aws

用户指南

AWS OpsWorks

API 版本 2013-02-18

AWS OpsWorks: 用户指南

Table of Contents

什么是 AWS OpsWorks？

> **Important**
>
> 这些 AWS OpsWorks 服务已接近使用寿命，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

AWS OpsWorks 是一项配置管理服务，可帮助您使用 Puppet 或 Chef 在云企业中配置和操作应用程序。 AWS OpsWorks 堆叠并 AWS OpsWorks for Chef Automate 允许你使用 Chef 食谱和解决方案进行配置管理，而对于 Puppet Enterprise，则 OpsWorks 允许你在中配置 Puppet Enterpr ise 主服务器。 AWS Puppet 提供了一组工具，用于为基础设施实施所需的状态和自动按需任务。

AWS OpsWorks 服务

适用于 Pupp OpsWorks et 企业的 AWS

OpsWorks for Puppet Enterprise 允许你创建 AWS托管的 Puppet 主服务器。Puppet Master 服务器管理基础设施中的节点，存储有关节点的事实，并用作 Puppet 模块的中央存储库。模块是可重用、可共享的 Puppet 代码单元，其中包含有关应如何配置基础设施的指令。您可以从 Puppet Forge 下载社区模块，或者使用 Puppet 开发工具包创建自己的自定义模块，然后使用 Puppet Code Manager 管理其部署。

OpsWorks for Puppet Enterprise 提供完全托管的 Puppet Master，一套自动化工具，使您能够检查、交付、操作应用程序并为未来做好准备，还可以访问允许您查看有关节点和 Puppet 活动的信息的用户界面。 OpsWorks for Puppet Enterprise 允许你使用 Puppet 自动配置、部署和管理节点，无论它们是 Amazon EC2 实例还是本地设备。fo OpsWorks r Puppet Enterprise master 通过处理软件和操作系统配置、软件包安装、数据库设置、变更管理、策略实施、监控和质量保证等任务来提供全栈自动化。

由 OpsWorks 于 Puppet Enterprise 管理的 Puppet Enterprise 软件，因此您的服务器可以在您选择的时间自动备份，始终运行最新的 AWS 兼容版本的 Puppet，并且始终应用最新的安全更新。您可以使用 Amazon EC2 Auto Scaling 组，自动将新的 Amazon EC2 节点与您的服务器关联。

AWS f OpsWorks or Chef 自动化

AWS OpsWorks for Chef Automate 允许你创建包含 Chef AWS A utomate 高级功能的托管 Chef 服务器，并使用 Chef DK 和其他 Chef 工具来管理它们。Chef 服务器管理您环境中的各个节点、存储有关这些节点的信息，并用作 Chef 说明书的中央存储库。说明书包含您使用 Chef 管理的每个节点上的 Chef Infra 客户端 (`chef-client`) 代理运行的配方。你可以使用诸如`knife`和 T est Kitch en 之类的 Chef 工具来管理 AWS OpsWorks for Chef Automate 服务中 Chef 服务器上的节点和食谱。

Chef Automate 是一个随附的服务器软件包，它为持续部署和合规性检查提供自动化工作流程。 AWS OpsWorks for Chef Automate 使用单个亚马逊弹性计算云实例安装和管理 Chef Automate、Chef InSpec Infra 和 Chef。使用 AWS OpsWorks for Chef Automate，您可以使用社区创作或自定义 Chef 食谱，而无需进行 AWS OpsWorks特定更改。

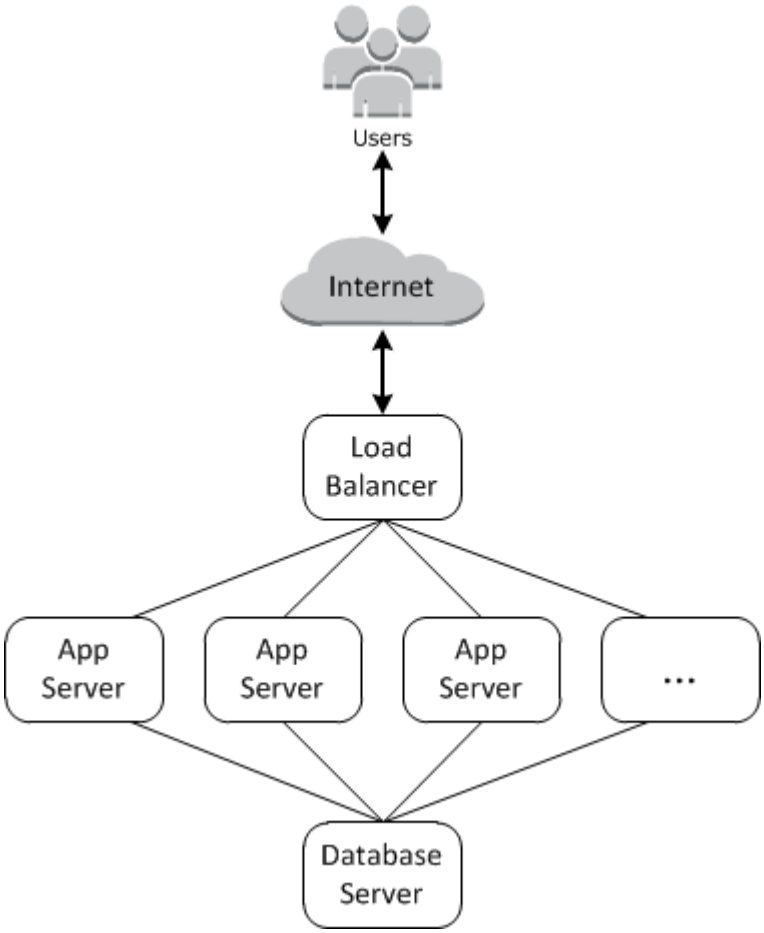
由于在单个实例上 AWS OpsWorks for Chef Automate 管理 Chef Automate 组件，因此您的服务器可以在您选择的时间自动备份，始终运行最新的 Chef 次要版本，并且始终应用最新的安全更新。您可以使用 Amazon EC2 Auto Scaling 组，自动将新的 Amazon EC2 节点与您的服务器关联。

AWS OpsWorks 堆栈

基于云的计算通常涉及各组 AWS 资源，如 EC2 实例和 Amazon关系数据库 (RDS, Relational Database Service) 实例。例如，一个 Web 应用程序通常需要应用程序服务器、数据库服务器、负载均衡器以及其他资源。此组实例通常称为堆栈。

AWS OpsWorks Stacks 是最初的服务，它提供了一种简单而灵活的方式来创建和管理堆栈和应用程序。 AWS OpsWorks Stacks 允许您在堆栈中部署和监控应用程序。您可以创建堆栈，以帮助您在专门的组 (名为层) 中管理云资源。层代表一组服务特定目的 (如提供应用程序服务或承载数据库服务器) 的 EC2 实例。层依靠 Chef 配方来处理诸如在实例上安装程序包、部署应用程序和运行脚本等任务。

与之不同的是 AWS OpsWorks for Chef Automate， AWS OpsWorks Stacks 不需要或创建 Chef 服务器； AWS OpsWorks Stacks 可以为你执行 Chef 服务器的部分工作。 AWS OpsWorks Stacks 监控实例的运行状况，并在需要时通过自动修复和自动扩缩，为您预配置新的实例。简单的应用程序服务器堆栈可能类似于下图。


Users
Internet
Load Balancer
App Server
App Server
App Server
...
Database Server

适用于 Pupp OpsWorks et 企业的 AWS

> **Important**
>
> 该 AWS OpsWorks for Puppet Enterprise 服务于 2024 年 3 月 31 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

OpsWorks for Puppet Enterprise 允许你在几分钟内启动 Puppet Enter prise 主服务器，然后让我们 AWS OpsWorks 处理其操作、备份、恢复和软件升级。 OpsWorks for Puppet Enterprise 可以让你腾出时间专注于核心配置管理任务，而不是管理 Puppet 大师。通过使用 OpsWorks Puppet Enterprise，您可以使用相同的配置来管理本地和云基础架构，从而帮助您在混合环境中高效地扩展运营。Puppet Enterprise 控制台、 AWS Management Console和 AWS CLI简化了 Puppet Master 服务器的管理。

Puppet Master 将特定节点的配置目录提供给 `puppet-agent` 软件，并用作 Puppet 模块的中央存储库，从而管理环境中节点的配置。Puppet Enterprise 中的 OpsWorks Puppet 大师会部署`puppet-agent`到你的托管节点，并提供 Puppet Enterprise 的高级功能。

OpsWorks 适用于 Puppet Enterprise 的主服务器在亚马逊弹性计算云实例上运行。 OpsWorks for Puppet Enterprise 服务器配置为运行最新版本的亚马逊 Linux（亚马逊 Linux 2）和最新版本的 Puppet Enterprise Master，版本 2019.8.5。有关 Puppet Enterprise 2019.8.5 中的更改的更多信息，请参阅 Puppet Enterprise 发布说明。

当 Puppet 软件的新版本可用时，系统维护的目的是在服务器上的 Puppet Enterprise 版本通过 AWS 测试后立即自动更新。AWS 会进行大量测试，确认 Puppet 升级已达到生产就绪状态，并且不会中断现有客户环境。

您可以将任何运行支持的操作系统并具有网络访问权限的 Puppet Enterprise 主服务器 OpsWorks 的本地计算机或 EC2 实例连接起来。`puppet` 代理软件由 Puppet Master 安装在您要管理的节点上。

主题

- Puppet Enterprise OpsWorks 的区域支持
- AWS OpsWorks for Puppet Enterprise 生命周期终结常见问题解答
- Puppet Enterp OpsWorks rise 入门
- 使用创建 AWS OpsWorks for Puppet Enterprise 大师 AWS CloudFormation

- 更新 Pupp OpsWorks et Enterprise 服务器以使用自定义域
- 使用 AWS OpsWorks for Puppet Enterprise 资源上的标签
- 备份和恢复 Puppe OpsWorks t Enterprise Server
- Puppet Enterp OpsWorks rise 系统维护中
- OpsWorks 为 Puppet Enterprise 自动添加节点
- 取消节点与 for Puppet OpsWorks 企业服务器的关联
- 删除 Pupp OpsWorks et 企业服务器的
- 如何将 Puppe OpsWorks t Enterprise 服务器迁移到亚马逊弹性计算云 (Amazon EC2)
- 使用登录 OpsWorks Puppet 企业 API 调用 AWS CloudTrail
- Pupp OpsWorks et 企业版故障排除

Puppet Enterprise OpsWorks 的区域支持

以下区域终端节点支持 OpsWorks Puppet Enterprise 主服务器。 OpsWorks for Puppet Enterprise 在与你的 Puppet 主服务器相同的区域终端节点中创建与你的 Puppet 主节点关联的资源，例如实例配置文件、用户和服务角色。您的 Puppet Master 必须在 VPC 中。您可以使用您创建的或已有的 VPC，或使用默认 VPC。

- 美国东部（俄亥俄州）区域
- 美国东部（弗吉尼亚州北部）区域
- 美国西部（北加利福尼亚）区域
- 美国西部（俄勒冈州）区域
- Asia Pacific（Tokyo）Region
- 亚太地区（新加坡）区域
- 亚太地区（悉尼）区域
- 欧洲地区（法兰克福）区域
- 欧洲地区（爱尔兰）区域

AWS OpsWorks for Puppet Enterprise 生命周期终结常见问题解答

> **Important**
>
> 该 AWS OpsWorks for Puppet Enterprise 服务于 2024 年 3 月 31 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

主题

生命周期终止将如何影响现有客户？

在2024年3月31日（Puppet Enterprise的生命周期终止日期）之前，现有客户将不受影响。 OpsWorks 在生命周期终止日期之后，客户将无法再使用 OpsWorks 控制台或 API 管理其服务器。

如果我不采取任何行动，我的服务器会怎么样？

从 2024 年 3 月 31 日起，您将无法再使用 OpsWorks 控制台或 API 管理服务器。届时，我们将停止对您的服务器执行任何正在进行中的管理功能，例如备份或维护。为了限制对客户的影响，我们将让备份 Puppet Enterprise 服务器的 EC2 实例保持运行，但它们的许可证将不再有效，因为 for Puppet Enterprise 服务协议不再涵盖使用量（或计费）。 OpsWorks 如果你想继续使用 Puppet Enterprise 管理基础设施，请参阅如何将 OpsWorks 适用于 Puppet Enterprise 的服务器迁移到亚马逊弹性计算云 (Amazon EC2)。

正在 AWS OpsWorks for Puppet Enterprise 接受新客户吗？

不是。 AWS OpsWorks for Puppet Enterprise 不再接受新客户。

生命的尽头会同时影响所有 AWS 区域 人吗？

是的。自 2024 年 3 月 31 日起，API 和控制台将处于使用生命周期终止状态，且无法在所有地区使用。有关可用 AWS 区域 地点 AWS OpsWorks for Puppet Enterprise 的列表，请参阅AWS 区域服务列表。

可获得什么级别的技术支持 AWS OpsWorks for Puppet Enterprise？

AWS 在生命周期终止日期之前 AWS OpsWorks for Puppet Enterprise ，将继续为客户提供与当今相同水平的支持。如果您有任何疑问或疑虑，可以通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

我是 Puppet Enterprise 的现有客户，我需要使用以前未使用该服务的账户启动服务器。 OpsWorks 我能做到吗？

通常不能，除非有特殊情况需要这样做。如果您有特殊情况，请通过 re AWS : Post 或通过 Pre mium Su AWS pport 联系 AWS Support 团队，提供详细信息和理由，我们将审核您的请求。

会有新功能发布吗 AWS OpsWorks for Puppet Enterprise？

没有。由于该服务已将至生命周期终止，因此我们不会发布任何新功能。但是，在生命周期终止之前，我们将继续改进安全性并按预期管理服务器。

Puppet Enterp OpsWorks rise 入门

> **Important**
>
> 该 AWS OpsWorks for Puppet Enterprise 服务于 2024 年 3 月 31 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

OpsWorks for Puppet Enterprise 允许你在中运行 Puppet Enter p AWS您可以在大约 15 分钟内预配置一个 Puppet Enterprise Master 服务器。

从 2021 年 5 月 3 日起，Puppet Enterprise 将在其中存储一些 Puppet Enterprise OpsWorks AWS Secrets Manager有关更多信息，请参阅 与 AWS Secrets Manager 集成。

以下演练可帮助你在 Puppet Enterprise 中 OpsWorks 创建第一个 Puppet 大师。

先决条件

在开始之前，您必须完成以下前提条件：

主题

- 安装 Puppet 开发工具包
- 安装 Puppet Enterprise 客户端工具
- 设置 Git 控制存储库
- 设置 VPC
- 设置 EC2 密钥对（可选）
- 使用自定义域的先决条件（可选）

安装 Puppet 开发工具包

1. 从 Puppet 网站下载 Puppet 开发工具包，该工具包适合您的本地计算机的操作系统。
2. 安装 Puppet 开发工具包。
3. 将 Puppet 开发工具包添加到本地计算机的 PATH 变量。

   - 在 Linux 或 macOS 操作系统上，您可以通过在 Bash shell 中运行以下命令来将 Puppet 开发工具包添加到 PATH 变量。

     ```
     echo 'export PATH=/opt/puppetlabs/pdk/bin/pdk:$PATH' >> ~/.bash_profile && source
      ~/.bash_profile
     ```

   - 在基于 Windows 的操作系统上，可以在 PowerShell 会话中或通过 “系统属性” 访问的 “环境**PATH**变量” 对话框中使用以下.NET Framework 命令将 Puppet 开发套件添加到变量中。您可能需要以管理员身份运行 PowerShell 会话才能运行以下命令。

     ```
     [Environment]::SetEnvironmentVariable("Path","new path value","Machine")
     ```

安装 Puppet Enterprise 客户端工具

Puppet Enterprise (PE) 客户端工具是一组命令行工具，可让您从工作站访问 Puppet Enterprise 服务。这些工具可安装在许多不同的操作系统上，也可以安装在使用 Puppet 管理的节点上。有关支持这

些工具的操作系统以及如何安装这些工具的信息，请参阅 Puppet Enterprise 文档中的安装 PE 客户端工具。

设置 Git 控制存储库

您必须先在 Git 中配置一个控制存储库来存储 Puppet 模块和类并对它们进行变更管理，然后才能启动 Puppet Master。在启动 Puppet Enterprise Master 服务器的步骤中，需要指向 Git 存储库的 URL 以及用于访问该存储库的 HTTPS 或 SSH 账户信息。有关如何设置 Puppet Enterprise Master 将使用的控制存储库的更多信息，请参阅设置控制存储库。您还可以在上的 Puppet `control-repo`示例存储库的自述文件中找到控制存储库设置说明。 GitHub控制存储库的结构类似于以下内容。

```
### LICENSE
### Puppetfile
### README.md
### environment.conf
### hieradata
#   ### common.yaml
#   ### nodes
#       ### example-node.yaml
### manifests
#   ### site.pp
### scripts
#   ### code_manager_config_version.rb
#   ### config_version.rb
#   ### config_version.sh
### site
    ### profile
    #   ### manifests
    #       ### base.pp
    #       ### example.pp
    ### role
        ### manifests
            ### database_server.pp
            ### example.pp
            ### webserver.pp
```

使用设置存储库 CodeCommit

您可以使用创建新存储库 CodeCommit。有关 CodeCommit 如何使用创建控制存储库的更多信息，请参阅本指南the section called “可选：使用 CodeCommit”中的。有关如何开始使用 Git 的更多信息 CodeCommit，请参阅 AWS 入门 CodeCommit。要 OpsWorks 为您的存储库授权 Puppet Enterprise 服务器，请将该`AWSCodeCommitReadOnly`策略附加到您的 IAM 实例配置文件角色。

设置 VPC

你的 OpsWorks Puppet Enterprise 主服务器必须在亚马逊 Virtual Private Cloud 中运行。您可以将其添加到现有 VPC、使用默认 VPC，或者创建新 VPC 以包含服务器。有关创建 Amazon VPC 以及如何创建新 VPC 的信息，请参阅 Amazon VPC 入门指南。

如果您创建自己的 VPC，或者使用现有的 VPC，则它应具有以下设置或属性。

- VPC 应具有至少一个子网。

  如果您的 OpsWorks Puppet Enterprise 主服务器可以公开访问，请将子网设为公有子网，然后启用自动分配公有 IP。
- 应该启用 DNS resolution。
- 在子网上，启用 Auto-assign public IP。

如果您不熟悉创建 VPC 或在其中运行实例，则可以使用为您 AWS OpsWorks 提供的 AWS CloudFormation 模板运行以下 AWS CLI 命令来创建具有单个公有子网的 VPC。如果您更喜欢使用 AWS Management Console，也可以将模板上传到 AWS CloudFormation 控制台。

```
aws cloudformation create-stack --stack-name OpsWorksVPC --template-url https://
s3.amazonaws.com/opsworks-cm-us-east-1-prod-default-assets/misc/opsworks-cm-vpc.yaml
```

设置 EC2 密钥对（可选）

对于 Puppet 服务器的典型管理，不需要也不建议使用 SSH 连接；您可以使用 AWS Management Console 和 AWS CLI 命令在 Puppet 服务器上执行许多管理任务。

如果您丢失或者希望更改 Puppet Enterprise 基于 Web 的控制台的登录密码，则需要使用 EC2 密钥对通过 SSH 连接到您的服务器。您可以使用现有的密钥对，或者创建新的密钥对。有关如何创建新 EC2 密钥对的更多信息，请参阅 Amazon EC2 密钥对。

如果您不需要 EC2 密钥对，那么您已准备好创建 Puppet Enterprise Master。

使用自定义域的先决条件（可选）

您可以在自己的域上设置 Puppet Enterprise 主服务器，同时在自定义域中指定一个公有终端节点用作服务器的终端节点。当您使用自定义域时，需要执行以下所有操作，如本节中详细介绍。

主题

- 设置自定义域
- 获取证书
- 获取私有密钥

设置自定义域

要在自己的自定义域上运行 Puppet Enterprise 主服务器，您需要服务器的公有终端节点，例如 `https://aws.my-company.com`。如果指定自定义域，还必须提供证书和私有密钥，如前面各节所述。

要在创建服务器后访问此服务器，请在首选 DNS 服务中添加 CNAME DNS 记录。此记录必须将自定义域指向由 Puppet 主服务器创建过程生成的终端节点（服务器的 `Endpoint` 属性的值）。如果服务器使用自定义域，则无法使用生成的 `Endpoint` 值访问服务器。

获取证书

要在您自己的自定义域上设置您的 Puppet 主服务器，您需要 PEM 格式的 HTTPS 证书。这可以是单个自签名证书或证书链。在完成 Create a Puppet Enterprise Master (创建 Puppet Enterprise 主服务器) 工作流时，如果您指定此证书，则还必须提供自定义域和私有密钥。

以下是证书值的要求：

- 您可以提供自签名的自定义证书或完整的证书链。
- 证书必须是有效的 X509 证书或 PEM 格式的证书链。
- 证书在上传时必须是有效的。您不能在证书有效期开始（证书的 `NotBefore` 日期）之前或证书有效期到期（证书的 `NotAfter` 日期）之后使用证书。
- 证书的公用名称或使用者备用名称 (SAN)（如果存在）必须与自定义域值匹配。
- 证书必须与 Custom private key (自定义私有密钥) 字段的值匹配。

获取私有密钥

要在自己的自定义域上设置 Puppet 主服务器，您需要一个 PEM 格式的私有密钥，以便使用 HTTPS 连接到服务器。私有密钥不得加密；无法使用密码或密码短语保护它。如果指定自定义私有密钥，则还必须提供自定义域和证书。

创建 Puppet Enterprise Master

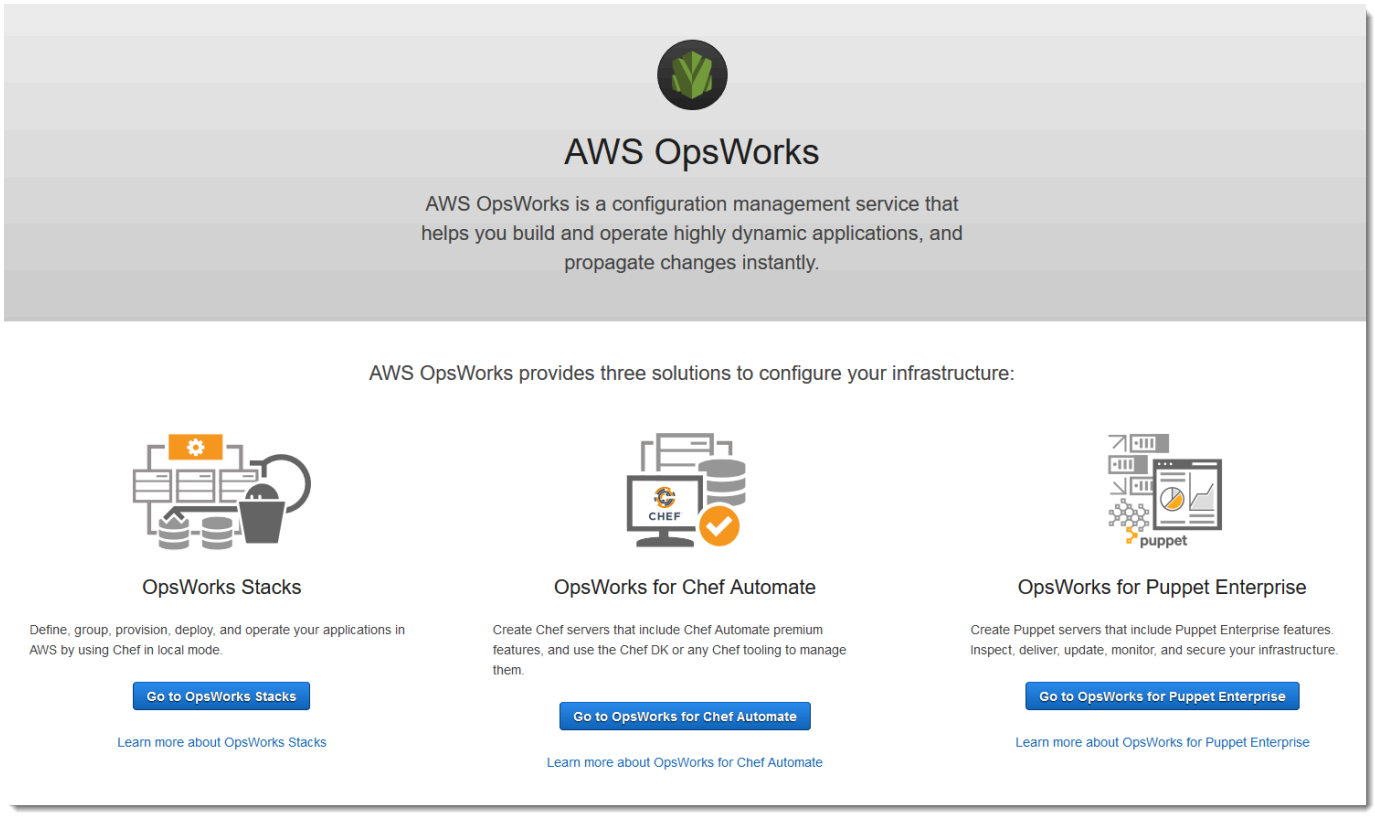
> **Important**
>
> 该 AWS OpsWorks for Puppet Enterprise 服务于 2024 年 3 月 31 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

你可以使用 for Puppet Enterprise 控制台创建 Puppet 大师，或者。 OpsWorks AWS CLI

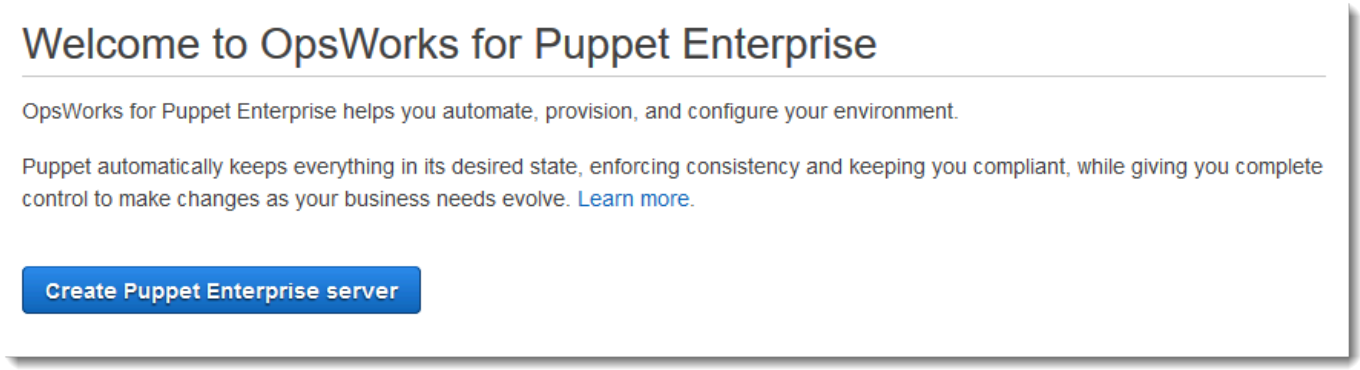
主题

使用创建 Puppet Enterprise Master AWS Management Console

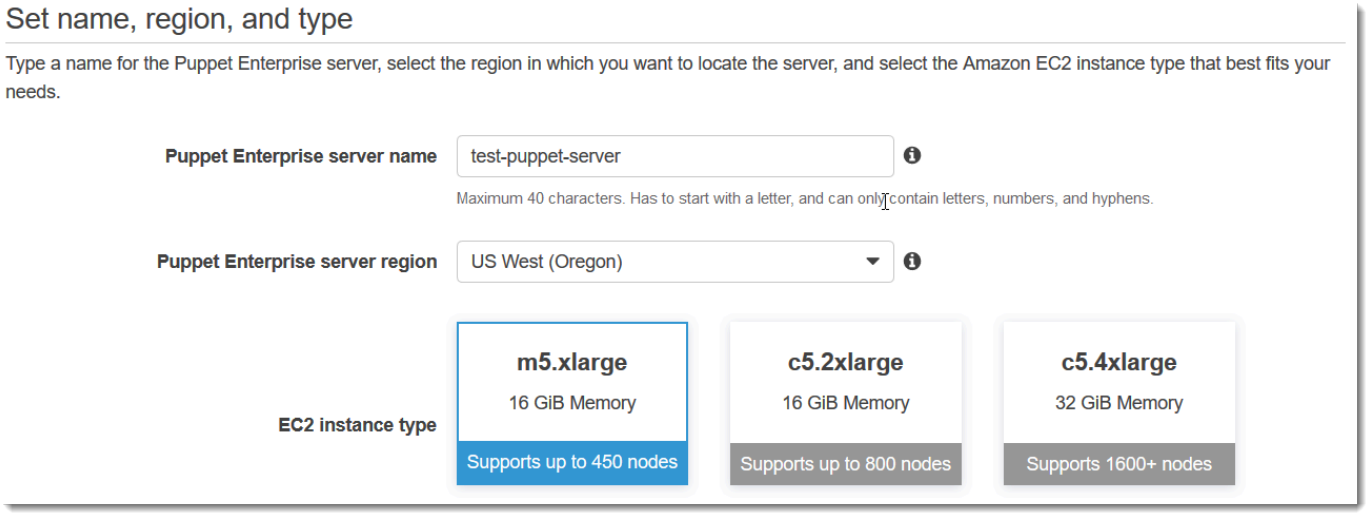
1. 登录 AWS Management Console 并打开 AWS OpsWorks 控制台，网址为 https://console.aws.amazon.com/opsworks/。
2. 在 AWS OpsWorks 主页上，选择 Puppet Enterprise OpsWorks 的 “前往”。



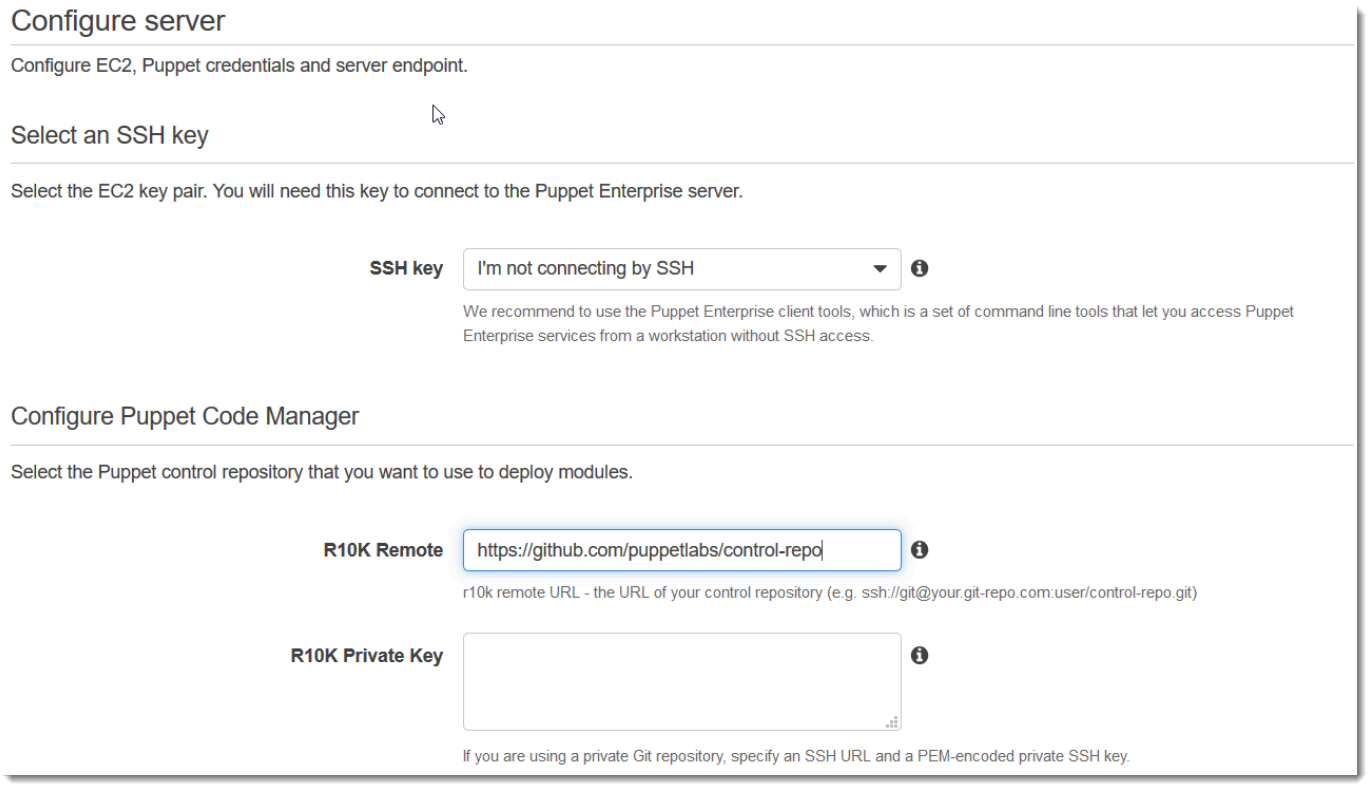
3. 在 Puppe OpsWorks t Enterprise 的主页上，选择创建 Puppet Enter prise 服务器



4. 在 Set name, region, and type (设置名称、区域和类型) 页面上，指定服务器的名称。Puppet Master 名称最多可包含 40 个字符，必须以字母开头，并且只能包含字母数字字符和短划线。选择支持的区域，然后选择支持所要管理节点数量的实例类型。如果需要，您可以在服务器创建之后更改实例类型。对于本演练，我们在美国西部（俄勒冈州）区域中创建一个 m5.xlarge 实例类型。选择下一步。



5. 在 Configure server 页面上，除非您要指定密钥对名称，否则保留 SSH key 下拉列表中的默认选择。在 Configure Puppet Code Manager 区域的 r10k remote 字段中，指定 Git remote 的有效 SSH 或 HTTPS URL。在 r10k 私钥字段中，粘贴 AWS OpsWorks 可用于访问 r10k 远程存储库的 SSH 私钥。此密钥是 Git 在您创建私有存储库时提供的，但如果您使用 HTTPS 身份验证来访问控制存储库，则不需要它。选择下一步。



6. 对于 Specify server endpoint (指定服务器终端节点)，保留默认值 Use an automatically-generated endpoint (使用自动生成的终端节点)，然后选择 Next (下一步)，除非您希望您的服务器位于您自己的自定义域中。要配置自定义域，请继续执行下一步。

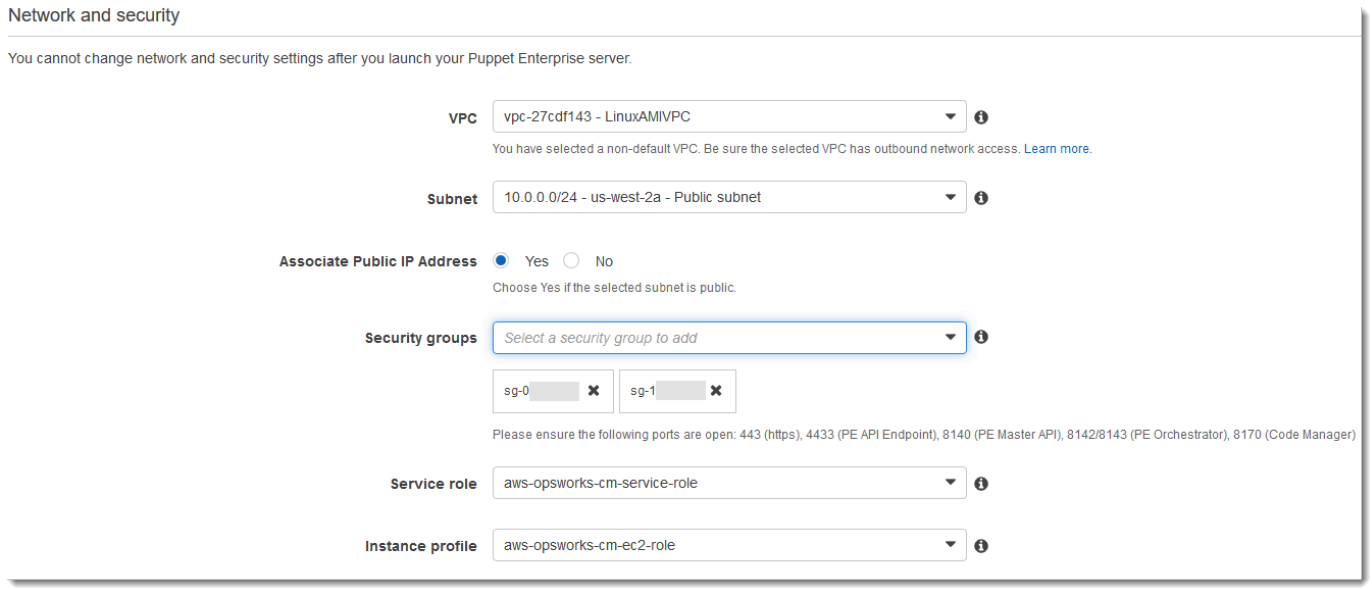
7. 要使用自定义域，对于 Specify server endpoint (指定服务器终端节点)，从下拉列表中选择 Use a custom domain (使用自定义域)。

   a. 对于完全限定域名 (FQDN)，指定 FQDN。您必须拥有要使用的域名。

   b. 对于 SSL 证书，请粘贴整个 PEM 格式的证书，以 `-----BEGIN CERTIFICATE-----` 开头并以 `-----END CERTIFICATE-----` 结尾。SSL 证书主题必须与您在上一步中输入的 FQDN 相匹配。删除证书之前和之后的任何多余行。

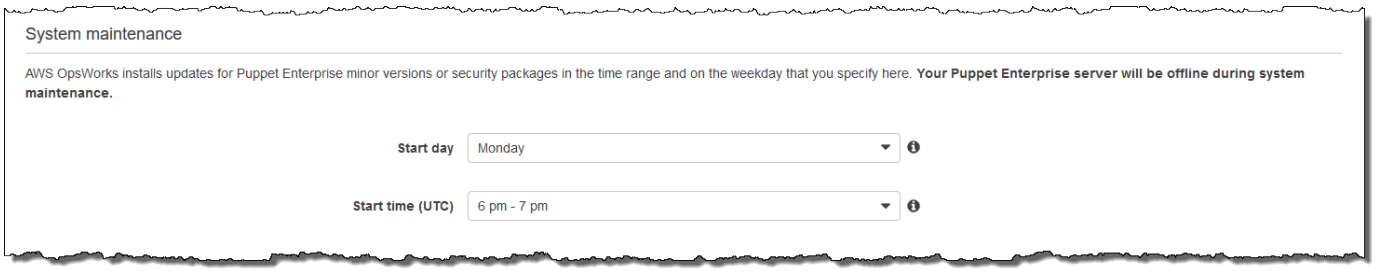
   c. 对于 SSL private key (SSL 私有密钥)，请粘贴整个 RSA 私有密钥，以 `-----BEGIN RSA PRIVATE KEY-----` 开头并以 `-----END RSA PRIVATE KEY-----` 结尾。SSL 私有密钥必须与您在上一步中输入的 SSL 证书中的公有密钥匹配。删除私钥之前和之后的任何多余行。选择下一步。

8. 在配置高级设置页面的网络和安全区域中，选择一个 VPC、子网以及一个或多个安全组。 AWS OpsWorks 如果您还没有要使用的安全组、服务角色和实例配置文件，则可以为您生成安全组、服务角色和实例配置文件。您的服务器可属于多个安全组。在离开此页后，您无法更改 Puppet Master 的网络和安全设置。

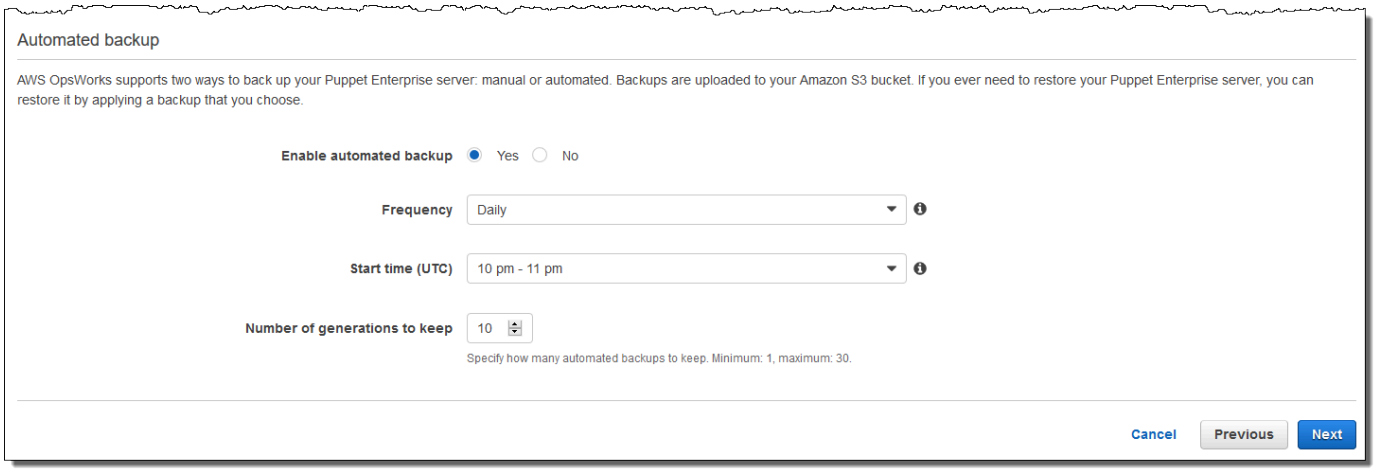


9. 在 System maintenance (系统维护) 部分中，设置您希望系统维护开始的日期和时间。由于您可以预见到服务器会在系统维护期间脱机，因此请选择正常工作时间中对服务器需求较低的时间。

   维护时段是必需的。您可以稍后使用 AWS Management Console AWS CLI、或 API 更改开始日期和时间。



10. 配置备份。默认情况下会启用自动备份。设置自动备份启动的首选频率和时间，并设置在 Amazon Simple Storage Service 中存储的备份生成数。最多可以保留 30 个备份；当达到最大值时，OpsWorks Puppet Enterprise 会删除最旧的备份，以便为新备份腾出空间。



11. （可选）在 Tags (标签) 中，向服务器和相关资源（如 EC2 实例、弹性 IP 地址、安全组、S3 存储桶和备份）添加标签。有关为 Puppet Enterprise 服务器添加标签 OpsWorks 的更多信息，请参阅。使用 AWS OpsWorks for Puppet Enterprise 资源上的标签
12. 在您配置完高级设置后，选择 Next (下一步)。
13. 在审核页面上，审核您的选择。当您准备好创建服务器时，选择 Launch (启动)。

    在等待 AWS OpsWorks 创建 Puppet 大师时，请继续下载入门套件使用初学者工具包配置 Puppet Master和 Puppet Enterprise 主机凭证。请不要一直等到您的服务器联机再下载这些项。

    服务器创建完成后，您的 Puppet 大师将在 Puppet Enterprise 的主页上可用，状态 OpsWorks 为在线。服务器联机之后，Puppet Enterprise 控制台在服务器的域上可用，位于以下格式的 URL 上：`https://your_server_name-randomID.region.opsworks-cm.io`。

使用创建 Puppet Enterprise Master AWS CLI

> **Important**
>
> 该 AWS OpsWorks for Puppet Enterprise 服务于 2024 年 3 月 31 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

通过运行 AWS CLI 命令 OpsWorks 为 Puppet Enterprise 创建主服务器不同于在控制台中创建服务器。如果您未指定要使用的现有服务角色和安全组，则可以在控制台中为您 AWS OpsWorks 创建服务角色和安全组。在中 AWS CLI，如果您未指定安全组，则 AWS OpsWorks 可以为您创建安全组，但它不会自动创建服务角色；您必须在命令中提供服务角色 ARN。`create-server`在主机中创建 Pupp AWS OpsWorks et 大师时，你可以下载 Puppet Enterprise 主机的入门套件和登录凭证。由于在使用创建 Puppet Enterprise 主服务器时无法执行此操作 AWS CLI，因此在新 OpsWorks 的 Puppet Enterprise 主服务器上线后，您可以使用 JSON 处理实用程序从`create-server`命令结果中获取登录凭证和入门套件。 OpsWorks

如果您的本地计算机尚未运行 AWS CLI，请 AWS CLI 按照 AWS 命令行界面用户指南中的安装说明下载并安装。本部分未介绍可以与 `create-server` 命令结合使用的所有参数。有关 `create-server` 参数的更多信息，请参阅 `create-server` 参考 中的 AWS CLI 。

1. 确保完成先决条件 。要创建 Puppet Master，您需要子网 ID，因此您必须有一个 VPC。
2. 创建服务角色和实例配置文件。 AWS OpsWorks 提供了一个可用于创建两者的 AWS CloudFormation 模板。运行以下 AWS CLI 命令创建一个 AWS CloudFormation 堆栈，用于为您创建服务角色和实例配置文件。

   ```
   aws cloudformation create-stack --stack-name OpsWorksCMRoles --template-url
    https://s3.amazonaws.com/opsworks-cm-us-east-1-prod-default-assets/misc/opsworks-
   cm-roles.yaml --capabilities CAPABILITY_NAMED_IAM
   ```

3. 创建 AWS CloudFormation 完堆栈后，在您的账户中查找并复制服务角色的 ARN。

   ```
   aws iam list-roles --path-prefix "/service-role/" --no-paginate
   ```

   在 `list-roles` 命令的结果中，查找类似于以下内容的服务角色 ARN 条目。记下服务角色 ARN。您需要使用这些值来创建 Puppet Enterprise 主服务器。

```
{
    "AssumeRolePolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Action": "sts:AssumeRole",
                "Effect": "Allow",
                "Principal": {
                    "Service": "ec2.amazonaws.com"
                }
            }
        ]
    },
    "RoleId": "AROZZZZZZZZZZQG6R22HC",
    "CreateDate": "2018-01-05T20:42:20Z",
    "RoleName": "aws-opsworks-cm-ec2-role",
    "Path": "/service-role/",
    "Arn": "arn:aws:iam::000000000000:role/service-role/aws-opsworks-cm-ec2-role"
},
{
    "AssumeRolePolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Action": "sts:AssumeRole",
                "Effect": "Allow",
                "Principal": {
                    "Service": "opsworks-cm.amazonaws.com"
                }
            }
        ]
    },
    "RoleId": "AROZZZZZZZZZZZZZZZ6QE",
    "CreateDate": "2018-01-05T20:42:20Z",
    "RoleName": "aws-opsworks-cm-service-role",
    "Path": "/service-role/",
    "Arn": "arn:aws:iam::000000000000:role/service-role/aws-opsworks-cm-service-
role"
}
```

4. 查找并复制您的账户中的实例配置文件的 ARN。

```
aws iam list-instance-profiles --no-paginate
```

在 `list-instance-profiles` 命令的结果中，查找类似于以下内容的实例配置文件 ARN 条目。记录实例配置文件 ARN。您需要使用这些值来创建 Puppet Enterprise 主服务器。

```
{
    "Path": "/",
    "InstanceProfileName": "aws-opsworks-cm-ec2-role",
    "InstanceProfileId": "EXAMPLEDC6UR3LTUW7VHK",
    "Arn": "arn:aws:iam::123456789012:instance-profile/aws-opsworks-cm-ec2-role",
    "CreateDate": "2017-01-05T20:42:20Z",
    "Roles": [
        {
            "Path": "/service-role/",
            "RoleName": "aws-opsworks-cm-ec2-role",
            "RoleId": "EXAMPLEE4STNUQG6R22HC",
            "Arn": "arn:aws:iam::123456789012:role/service-role/aws-opsworks-cm-
ec2-role",
            "CreateDate": "2017-01-05T20:42:20Z",
            "AssumeRolePolicyDocument": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {
                            "Service": "ec2.amazonaws.com"
                        },
                        "Action": "sts:AssumeRole"
                    }
                ]
            }
        }
    ]
},
```

5. 运行`create-server`命令 OpsWorks 为 Puppet Enterprise 主服务器创建。

   - 该 `--engine` 值是`Puppet`，`--engine-model`是`Monolithic`，`--engine-version`可以是`2019`或`2017`。
   - 在您的 AWS 账户中，每个区域内的服务器名称必须是唯一的。服务器名称必须以字母开头；然后允许字母、数字或连字符 (-)，最多 40 个字符。

- 使用您在步骤 3 和 4 中复制的实例配置文件 ARN 和服务角色 ARN。
- 有效实例类型为 `m5.xlarge`、`c5.2xlarge` 或 `c5.4xlarge`。有关这些实例类型的规范的更多信息，请参阅 Amazon EC2 用户指南中的实例类型。
- `--engine-attributes` 参数是可选的；如果您不指定 Puppet 管理员密码，则服务器创建过程会为您生成一个密码。如果您添加 `--engine-attributes`，请指定 `PUPPET_ADMIN_PASSWORD`，登录 Puppet Enterprise 控制台网页的管理员密码。该密码必须使用介于 8 和 32 个之间的 ASCII 字符。
- SSH 密钥对是可选的，但可以帮助您连接到 Puppet Master (如果您需要重置控制台管理员密码)。有关创建 SSH 密钥对的更多信息，请参阅《Amazon EC2 用户指南》中的 Amazon EC2 密钥对。
- 要使用自定义域，请将以下参数添加到命令中。否则，Puppet 主服务器创建过程会自动为您生成端点。配置自定义域需要所有三个参数。有关使用这些参数的其他要求的信息，请参阅 AWS OpsWorks CM API 参考CreateServer中的。
  - `--custom-domain` - 服务器的可选公有端点，例如 `https://aws.my-company.com`。
  - `--custom-certificate` - PEM 格式的 HTTPS 证书。该值可以是单个自签名证书或证书链。
  - `--custom-private-key` - PEM 格式的私有密钥，用于通过 HTTPS 连接到服务器。私有密钥不得加密；无法使用密码或密码短语保护它。
- 需要每周进行系统维护。有效值必须按以下格式指定：`DDD:HH:MM`。指定的时间为协调世界时 (UTC)。如果您不指定 `--preferred-maintenance-window` 的值，则默认值为星期二、星期三或星期五的一小时随机时间段。
- `--preferred-backup-window` 的有效值必须按以下格式之一指定：`HH:MM` (针对每日备份) 或 `DDD:HH:MM` (针对每周备份)。指定的时间采用 UTC 格式。默认值为随机每日开始时间。要退出自动备份，请改为添加参数 `--disable-automated-backup`。
- 对于 `--security-group-ids`，输入一个或多个安全组 ID，用空格分隔。
- 对于 `--subnet-ids`，输入子网 ID。

```
aws opsworks-cm create-server --engine "Puppet" --engine-model "Monolithic"
 --engine-version "2019" --server-name "server_name" --instance-profile-arn
 "instance_profile_ARN" --instance-type "instance_type" --engine-attributes
 '{"PUPPET_ADMIN_PASSWORD":"ASCII_password"}' --key-pair "key_pair_name" --
preferred-maintenance-window "ddd:hh:mm" --preferred-backup-window "ddd:hh:mm"
 --security-group-ids security_group_id1 security_group_id2 --service-role-arn
 "service_role_ARN" --subnet-ids subnet_ID
```

示例如下：

```
aws opsworks-cm create-server --engine "Puppet" --engine-model
 "Monolithic" --engine-version "2019" --server-name "puppet-02" --
instance-profile-arn "arn:aws:iam::111122223333:instance-profile/aws-
opsworks-cm-ec2-role" --instance-type "m5.xlarge" --engine-attributes
 '{"PUPPET_ADMIN_PASSWORD":"zZZzDj2DLYXSZFRv1d"}' --key-pair "amazon-test"
 --preferred-maintenance-window "Mon:08:00" --preferred-backup-window
 "Sun:02:00" --security-group-ids sg-b00000001 sg-b0000008 --service-role-arn
 "arn:aws:iam::111122223333:role/service-role/aws-opsworks-cm-service-role" --
subnet-ids subnet-383daa71
```

以下示例创建使用自定义域的 Puppet 主服务器。

```
aws opsworks-cm create-server \
    --engine "Puppet" \
    --engine-model "Monolithic" \
    --engine-version "2019" \
    --server-name "puppet-02" \
    --instance-profile-arn "arn:aws:iam::111122223333:instance-profile/aws-
opsworks-cm-ec2-role" \
    --instance-type "m5.xlarge" \
    --engine-attributes '{"PUPPET_ADMIN_PASSWORD":"zZZzDj2DLYXSZFRv1d"}' \
    --custom-domain "my-puppet-master.my-corp.com" \
    --custom-certificate "-----BEGIN CERTIFICATE----- EXAMPLEqEXAMPLE== -----END
 CERTIFICATE-----" \
    --custom-private-key "-----BEGIN RSA PRIVATE KEY----- EXAMPLEqEXAMPLE= -----END
 RSA PRIVATE KEY-----" \
    --key-pair "amazon-test"
    --preferred-maintenance-window "Mon:08:00" \
    --preferred-backup-window "Sun:02:00" \
    --security-group-ids sg-b00000001 sg-b0000008 \
    --service-role-arn "arn:aws:iam::111122223333:role/service-role/aws-opsworks-
cm-service-role" \
    --subnet-ids subnet-383daa71
```

以下示例创建了添加两个标签的 Puppet 主服务器：`Stage: Production` 和 `Department: Marketing`。有关在 Puppet Enterprise 服务器上 OpsWorks 添加和管理标签的更多信息，请参阅本指南使用 AWS OpsWorks for Puppet Enterprise 资源上的标签中的。

```
aws opsworks-cm create-server \
```

```
    --engine "Puppet" \
    --engine-model "Monolithic" \
    --engine-version "2019" \
    --server-name "puppet-02" \
    --instance-profile-arn "arn:aws:iam::111122223333:instance-profile/aws-
opsworks-cm-ec2-role" \
    --instance-type "m5.xlarge" \
    --engine-attributes '{"PUPPET_ADMIN_PASSWORD":"zZZzDj2DLYXSZFRv1d"}' \
    --key-pair "amazon-test"
    --preferred-maintenance-window "Mon:08:00" \
    --preferred-backup-window "Sun:02:00" \
    --security-group-ids sg-b00000001 sg-b0000008 \
    --service-role-arn "arn:aws:iam::111122223333:role/service-role/aws-opsworks-
cm-service-role" \
    --subnet-ids subnet-383daa71 \
    --tags [{\"Key\":\"Stage\",\"Value\":\"Production\"},{\"Key\":\"Department\",
\"Value\":\"Marketing\"}]
```

6. OpsWorks 对于 Puppet Enterprise，创建新服务器大约需要 15 分钟。请勿关闭 `create-server` 命令的输出或关闭您的 Shell 会话，因为该输出可能包含不再显示的重要信息。要从 `create-server` 结果中获取密码和初学者工具包，请继续执行下一步。

   如果要在服务器上使用自定义域，请在 `create-server` 命令的输出中复制 `Endpoint` 属性的值。示例如下：

   ```
   "Endpoint": "puppet-07-exampleexample.opsworks-cm.us-east-1.amazonaws.com"
   ```

7. 如果你选择让 Puppet Enterprise 为你生成密码，你可以使用 jq 等 JSON 处理器从`create-server`结果中提取出可用的格式。 OpsWorks 安装 jq 后，您可以运行以下命令来提取 Puppet 管理员密码和初学者工具包。如果您未在步骤 3 中提供自己的密码，请确保将提取的管理员密码保存在方便且安全的位置。

   ```
   #Get the Puppet password:
   cat resp.json | jq -r '.Server.EngineAttributes[] | select(.Name ==
    "PUPPET_ADMIN_PASSWORD") | .Value'

   #Get the Puppet Starter Kit:
   cat resp.json | jq -r '.Server.EngineAttributes[] | select(.Name ==
    "PUPPET_STARTER_KIT") | .Value' | base64 -D > starterkit.zip
   ```

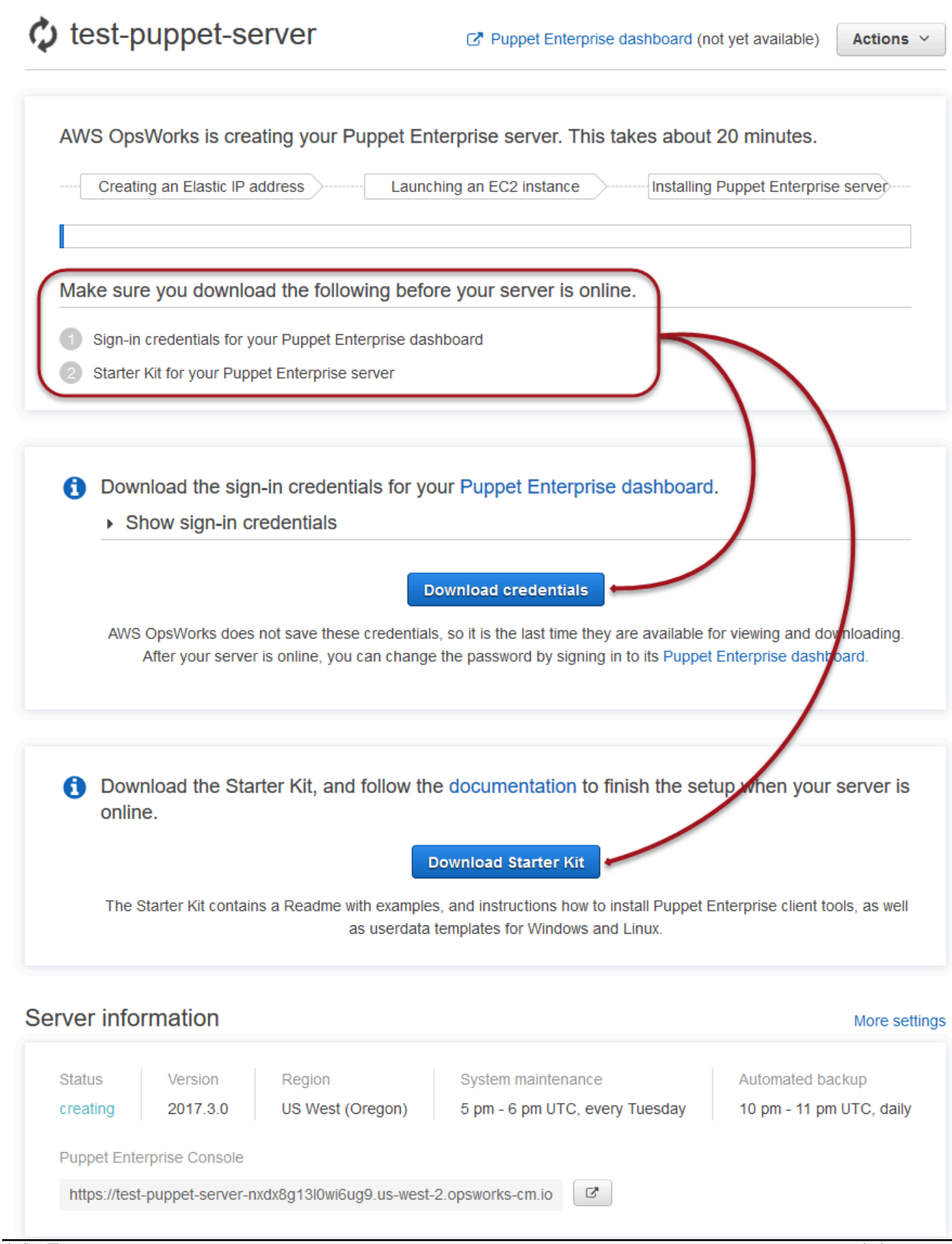
> **Note**
>
> 在 AWS Management Console中，您无法重新生成新的 Puppet Master 初学者工具包。使用创建 Puppet 大师时 AWS CLI，请运行前面的`jq`命令将`create-server`结果中的 base64 编码的入门套件保存为 ZIP 文件。

8. 如果您不使用自定义域，请继续下一步。如果您在服务器上使用自定义域，请在企业的 DNS 管理工具中创建 CNAME 条目，将您的自定义域指向您在步骤 6 中复制的 for Puppet Enterprise 终端节点。 OpsWorks 在完成此步骤之前，您无法访问或登录具有自定义域的服务器。
9. 继续执行下一部分，the section called “完成配置”。

使用初学者工具包配置 Puppet Master

> **Important**
>
> 该 AWS OpsWorks for Puppet Enterprise 服务于 2024 年 3 月 31 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

当 Puppet 大师的创建仍在进行中，服务器的 “属性” 页面将在 Puppet Enterprise 控制台中打开。 OpsWorks 在您首次使用新 Puppet Master 时，“Properties”页面提示您下载两个必需项。应在 Puppet 服务器联机之前下载这些项；新服务器联机之后，下载按钮将不可用。


test-puppet-server
Puppet Enterprise dashboard (not yet available)
Actions
AWS OpsWorks is creating your Puppet Enterprise server. This takes about 20 minutes.
Creating an Elastic IP address
Launching an EC2 instance
Installing Puppet Enterprise server
Make sure you download the following before your server is online.
1 Sign-in credentials for your Puppet Enterprise dashboard
2 Starter Kit for your Puppet Enterprise server
Download the sign-in credentials for your Puppet Enterprise dashboard.
Show sign-in credentials
Download credentials
AWS OpsWorks does not save these credentials, so it is the last time they are available for viewing and downloading.
After your server is online, you can change the password by signing in to its Puppet Enterprise dashboard.
Download the Starter Kit, and follow the documentation to finish the setup when your server is online.
Download Starter Kit
The Starter Kit contains a Readme with examples, and instructions how to install Puppet Enterprise client tools, as well as userdata templates for Windows and Linux.
Server information
More settings
Status
creating
Version
2017.3.0
Region
US West (Oregon)
System maintenance
5 pm - 6 pm UTC, every Tuesday
Automated backup
10 pm - 11 pm UTC, daily
Puppet Enterprise Console
https://test-puppet-server-nxdx8g13l0wi6ug9.us-west-2.opsworks-cm.io

- Puppet Master 的登录凭证。您将使用这些凭据登录 Puppet Enterprise 控制台，在那里您可以执行大部分节点管理。 AWS OpsWorks 不会保存这些凭据；这是它们最后一次可供查看和下载。如果需要，您可以在登录之后更改随这些凭证提供的密码。
- 初学者工具包。初学者工具包中包含一个自述文件，该文件包含相关信息、描述如何完成设置的示例以及 Puppet Enterprise 控制台的管理员凭证。每次下载初学者工具包时，将生成新凭证，并且旧凭证将失效。

先决条件

1. 当服务器创建仍在进行时，下载 Puppet Master 的登录凭证并将其保存到安全而方便的位置。
2. 下载初学者工具包，将初学者工具包 .zip 文件解压缩到您的工作区目录中。请不要共享您的登录凭证。如果其他用户将管理 Puppet Master，以后可以在 Puppet Enterprise 控制台中将其作为管理员添加。有关如何将用户添加到 Puppet Master 的更多信息，请参阅 Puppet Enterprise 文档中的创建并管理用户和用户角色。

安装 Puppet Master 证书

要使用 Puppet Master 并添加要管理的节点，您将需要安装其证书。通过运行以下 AWS CLI 命令进行安装。您无法在中执行此任务 AWS Management Console。

```
aws --region region opsworks-cm describe-servers --server-name server_name --query
 "Servers[0].EngineAttributes[?Name=='PUPPET_API_CA_CERT'].Value" --output text
 > .config/ssl/cert/ca.pem
```

生成短期令牌

要使用 Puppet API，您必须自行创建短期令牌。此步骤不是使用 Puppet Enterprise 控制台所必需的。通过运行以下命令生成令牌。

默认令牌生命周期为 5 分钟，不过您可以更改此默认值。

```
puppet-access login --config-file .config/puppetlabs/client-tools/puppet-access.conf --
lifetime 8h
```

> **Note**
>
> 由于默认令牌生命周期为 5 分钟，因此，上一个示例命令添加了 `--lifetime` 参数以将令牌生命周期延长一段时间。您可以将令牌生命周期设置为一个最多为 10 年的周期 (`10y`)。有关如

何更改默认令牌生命周期的更多信息，请参阅 Puppet Enterprise 文档中的更改令牌的默认生命周期。

设置初学者工具包 Apache 示例

在下载并解压缩初学者工具包后，您可以使用示例 `control-repo-example` 文件夹中的示例分支来在托管节点上配置 Apache Web 服务器。

初学者工具包包含两个 `control-repo` 文件夹：`control-repo` 和 `control-repo-example`。该`control-repo`文件夹包含一个`production`分支，该分支与您在 Puppet GitHub 存储库中看到的分支没有变化。`control-repo-example` 文件夹也包含一个 `production` 分支，该分支包含用于通过测试网站设置 Apache 服务器的示例代码。

1. 将 `control-repo-example` `production` 分支推送到 Git remote (Puppet Master 的 `r10k_remote` URL)。在您的入门套件根目录中，运行以下命令，将 *`r10 kRemoteUrl`* 替换为您的 `r10k_remote` URL。

   ```
   cd  control-repo-example
   git remote add origin r10kRemoteUrl
   git push origin production
   ```

   Puppet 的 Code Manager 使用 Git 分支作为环境。默认情况下，所有节点都在生产环境中。

   > **Important**
   >
   > 请不要推送到 `master` 分支。为 Puppet Master 保留 `master` 分支。

2. 将 `control-repo-example` 分支中的代码部署到您的 Puppet Master。这可让 Puppet Master 从 Git 存储库 (`r10k_remote`) 下载 Puppet 代码。在初学者工具包的根目录中，运行以下项。

   ```
   puppet-code deploy --all --wait --config-file .config/puppet-code.conf
   ```

有关如何将示例 Apache 配置应用于您在 Amazon EC2 中创建的托管代码的更多信息，请参阅本指南中的 第 2 步：使用自动化关联脚本创建实例。

为 Puppet Master 添加要管理的节点

> **Important**
>
> 该 AWS OpsWorks for Puppet Enterprise 服务于 2024 年 3 月 31 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

主题

添加节点的推荐方法是使用 AWS OpsWorks `associateNode()` API。Puppet Enterprise Master 服务器托管您用来在要管理的节点上安装 Puppet 代理软件的存储库，无论这些节点是在本地物理计算机上还是在虚拟机上。作为启动过程的一部分，适用于某些操作系统的 Puppet 代理软件安装在 OpsWorks Puppet Enterprise 服务器上。下表显示了 Puppet Enterprise OpsWorks 服务器启动时可用的操作系统代理。

预安装的操作系统代理

支持的操作系统	版本
Ubuntu	16.04、18.04、20.04
Red Hat Enterprise Linux (RHEL)	6、7、8
Windows	64 位版本的所有 Puppet 支持的 Windows 版本

对于其他操作系统，您可以将 `puppet-agent` 添加到您的服务器。请注意，系统维护将删除您在启动后添加到服务器的代理。虽然大多数运行已删除代理的现有附加节点将继续签入，但运行 Debian 操作系统的节点会停止报告。我们建议您在运行操作系统的节点`puppet-agent`上手动安装，而您 OpsWorks 的 puppet Enterprise 服务器上未预装代理软件。有关如何使 `puppet-agent` 适用于面向带其他操作系统的节点的服务器的详细信息，请参阅 Puppet Enterprise 文档中的安装代理。

有关如何通过填充 EC2 实例用户数据来自动将节点与 Puppet Master 关联的信息，请参阅 OpsWorks 为 Puppet Enterprise 自动添加节点。

运行 **associateNode()** API 调用

通过安装添加节点后puppet-agent，节点会向 Puppet Enterprise 服务器发送证书签名请求 (CSR)。OpsWorks 您可以在 Puppet console;中查看 CSR。有关节点 CSR 的更多信息，请参阅 Puppet Enterprise 文档中的管理证书签名请求。运行 puppet Enterprise associateNode() API 调用会处理节点 CSR，并将该节点与您的服务器关联起来。 OpsWorks 以下是如何在中使用此 API 调用关联单个节点的 AWS CLI 示例。您需要节点发送的 PEM 格式的 CSR；您可以从 Puppet 控制台获取它。

```
aws opsworks-cm associate-node --server-name "test-puppet-
server" --node-name "node or instance ID" --engine-attributes
 "Name=PUPPET_NODE_CSR,Value='PEM_formatted_CSR_from_the_node'
```

有关如何使用 associateNode() 自动添加节点的更多信息，请参阅 OpsWorks 为 Puppet Enterprise 自动添加节点。

添加本地节点的注意事项

在本地计算机或虚拟机puppet-agent上安装后，您可以使用两种方式中的任何一种将本地节点与 for Puppet Enterprise OpsWorks 主节点相关联。

- 如果某个节点支持安装 AWS 开发工具包、AWS CLI 或 AWS Tools for PowerShell ，您可以使用推荐的方法关联节点，即运行 associateNode() API 调用。首次创建 Puppet Enterpr OpsWorks ise 大师版时下载的入门套件展示了如何使用标签为节点分配角色。在将节点与 Puppet Master 关联的同时，您可以通过在 CSR 中指定可信事实来应用标签。例如，初学者工具包中随附的演示控制存储库被配置为使用标签 pp_role 将角色分配到 Amazon EC2 实例。有关如何将标签作为可信事实添加至 CSR 的更多信息，请参阅 Puppet 平台文档中的扩展请求 (永久证书数据)。
- 如果该节点无法运行 AWS 管理或开发工具，你仍然可以在你的 Puppet Enterprise 主节点中注册该节点，就像向任何非托管的 Puppet Enterprise 主节点注册一样。 OpsWorks 如本主题所述，安装puppet-agent会向 Puppet Enterprise 主服务器发送一个 CSR。 OpsWorks 授权 Puppet 用户可以手动签署 CSR，或通过编辑存储在 Puppet Master 上的 autosign.conf 文件配置 CSR 的自动签署。有关配置自动签署和编辑 autosign.conf 的更多信息，请参阅 Puppet 平台文档中的 SSL 配置：自动签署证书请求。

要将本地节点与 Puppet Master 关联，并允许 Puppet Master 接受所有 CSR，请在 Puppet Enterprise 控制台中执行以下操作。控制此行为的参数为 `puppet_enterprise::profile::master::allow_unauthenticated_ca`。

> **Important**
>
> 出于安全考虑，建议不要启用 Puppet Master 来接受自签名 CSR 或所有 CSR。默认情况下，允许未经身份验证的 CSR 将使 Puppet Master 可供任何人访问。默认情况下，将证书请求上传设置为启用会使您的 Puppet Master 易受拒绝服务 (DoS) 攻击。

1. 登录 Puppet Enterprise 控制台。
2. 依次选择 Configure (配置)、Classification (分类)、PE Master、Configuration (配置) 选项卡。
3. 在 Classification (分类) 选项卡上，找到类 puppet_enterprise::profile::master。
4. 将 allow_unauthenticated_ca 参数的值设置为 true。
5. 保存您的更改。您的更改将在下一次运行 Puppet 时应用。您可以允许更改在 30 分钟后生效（并添加本地节点），也可以在 PE 控制台的 Run (运行) 部分中手动启动 Puppet 运行。

更多信息

访问 Le arn Puppet 教程网站，详细了解如何使用 OpsWorks Puppet Enterprise 服务器和 Puppet Enterprise 控制台功能。

登录 Puppet Enterprise 控制台

> **Important**
>
> 该 AWS OpsWorks for Puppet Enterprise 服务于 2024 年 3 月 31 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

在您从 Puppet Master 的“Properties”页面下载了登录凭证并且服务器联机之后，登录 Puppet Enterprise 控制台。在本演练中，我们已指导您指定包含模块的控制存储库，并添加至少一个要管理的节点。这样您就可以在控制台中查看有关代理和节点的信息。

当您尝试连接到 Puppet Enterprise 控制台网页时，您的浏览器中会显示证书警告，直到您在用于管理 Puppet 服务器的客户端计算机上安装 AWS OpsWorks特定的、由 CA 签名的 SSL 证书。如果您不希望在继续到控制面板页面之前看到警告，请先安装 SSL 证书，然后登录。

安装 AWS OpsWorks SSL 证书

- 选择与您的系统匹配的证书。
  - 对于基于 Linux 或 macOS 的系统，请从以下亚马逊 S3 位置下载文件扩展名为 PEM 的文件：https://s3.amazonaws.com/ opsworks-cm-us-east--/misc/-2016-prod-default-assets root.pem。opsworks-cm-ca

    > **Note**
    >
    > 此外，请从以下位置下载更新的PEM文件：https://s3.amazonaws.com/opsworks-cm-us-east-1-prod-default-assets/misc/opsworks-cm-ca-2020-root.pem。由 OpsWorks 于 Puppet Enterprise 目前正在续订其根证书，因此您必须同时信任新旧证书。

    有关如何在 macOS 上管理 SSL 证书的更多信息，请参阅 Apple Support 网站的 在 Mac 上的 Keychain Access 中获取有关证书的信息。
  - 对于基于 Windows 的系统，请从以下亚马逊 S3 位置下载文件扩展名为 P7B 的文件：https://s3.amazonaws.com/ opsworks-cm-us-east--/misc/- 2016-root.p7b。prod-default-assets opsworks-cm-ca

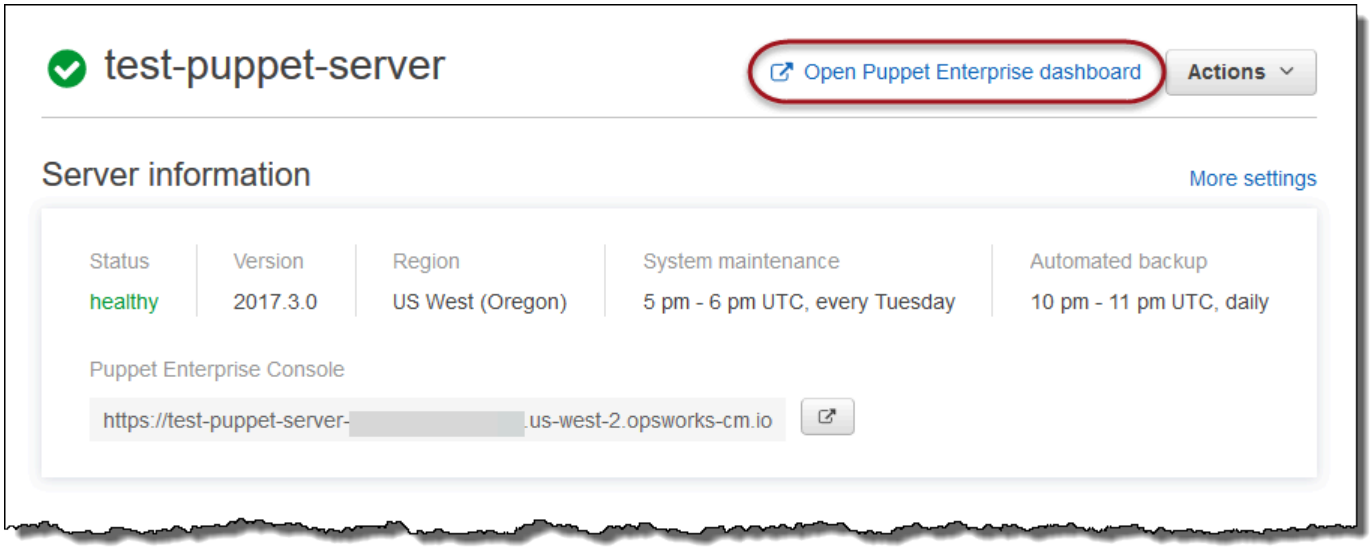
    有关如何在 Windows 上安装 SSL 证书的更多信息，请参阅在 Microsoft 上管理可信根证书 TechNet。

    > **Note**
    >
    > 此外，请从以下位置下载更新的 P7B 文件：https://s3.amazonaws.com/opsworks-cm-us-east-1-prod-default-assets/misc/opsworks-cm-ca-2020-root.p7b。由 OpsWorks 于 Puppet Enterprise 目前正在续订其根证书，因此您必须同时信任新旧证书。

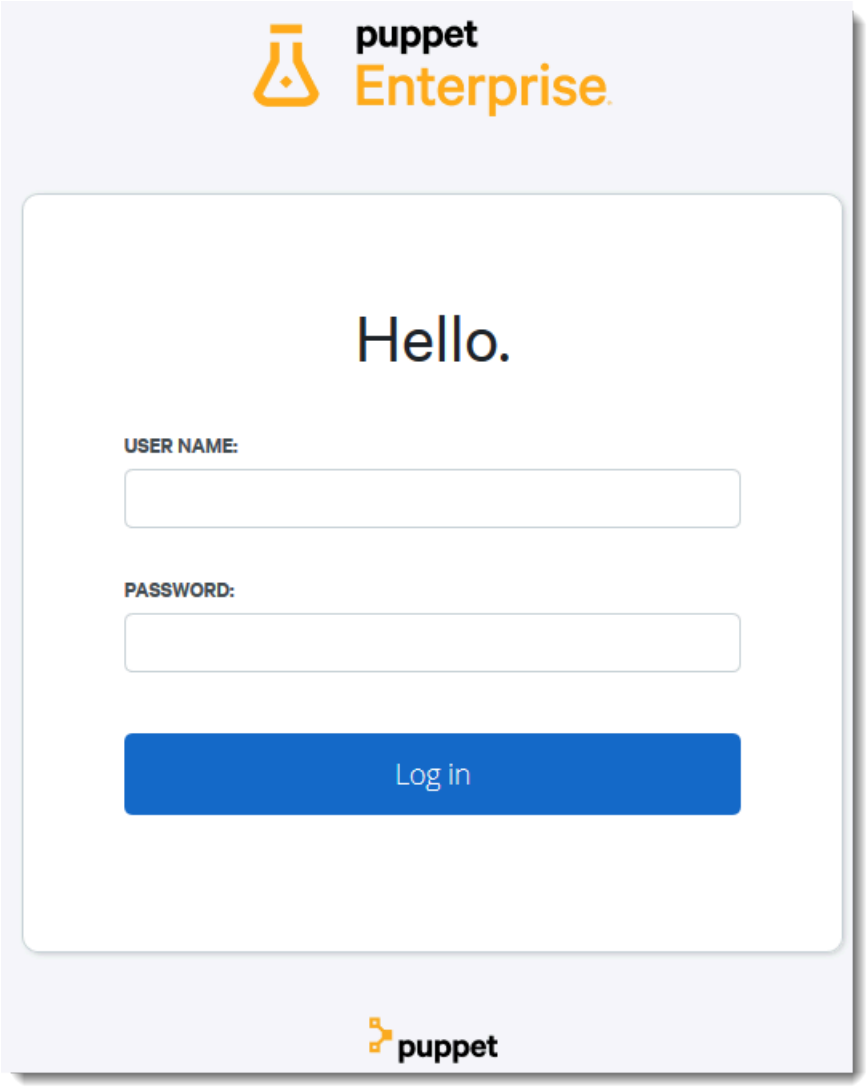
在安装了客户端 SSL 证书之后，您可以登录到 Puppet Enterprise 控制台而不会看到警告消息。

登录 Puppet Enterprise 控制台

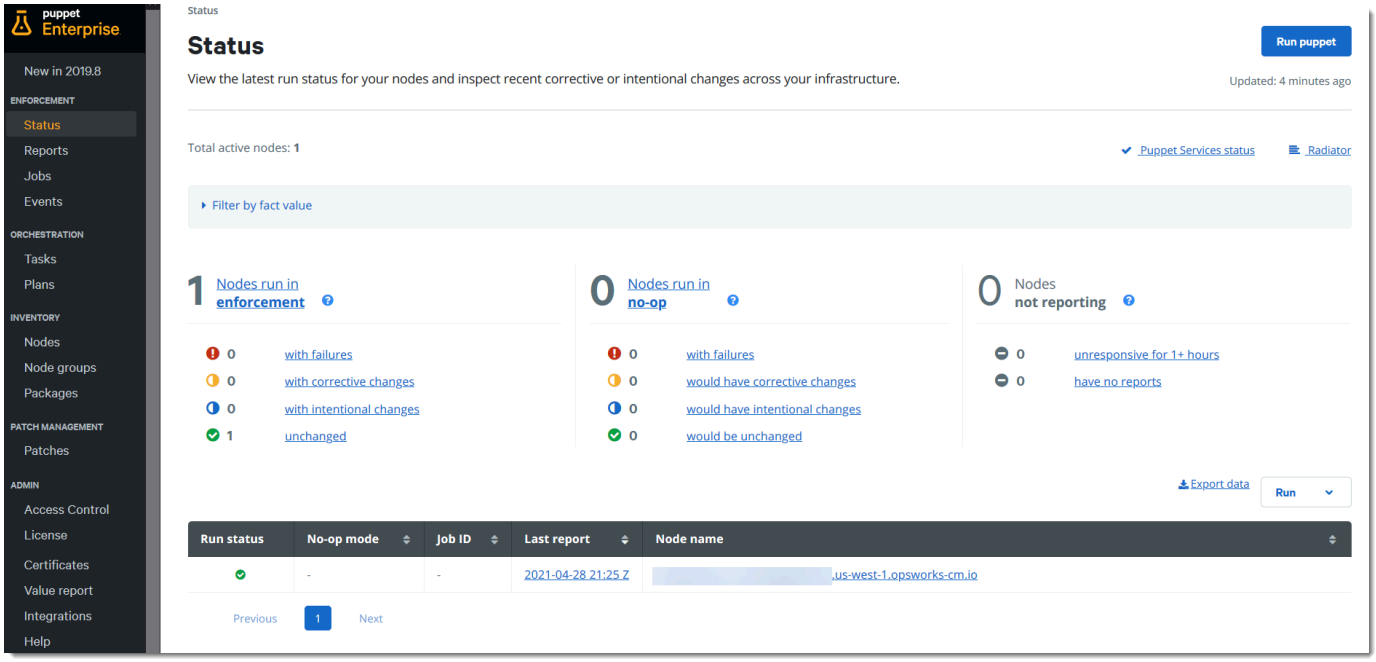
1. 解压缩并打开您在先决条件 中下载的 Puppet Enterprise 凭证。您需要这些凭证才能登录。
2. 在中 AWS Management Console，打开 Puppet 服务器的 “属性” 页面。
3. 在 Properties 页面的右上角，选择 Open Puppet Enterprise console。



4. 使用步骤 1 中的凭证登录。



5. 在 Puppet Enterprise 控制台中，您可以查看有关您管理的节点的详细信息、模块运行进度和事件、节点的合规性水平等等。有关 Puppet Enterprise 控制台的功能以及如何使用这些功能的更多信息，请参阅 Puppet Enterprise 文档中的管理节点。



对节点进行分组和分类

在通过将类应用于节点来指定节点所需的配置之前，根据节点在 Enterprise 中的作用或节点的常见特征来对其进行分组。对节点进行分组和分类涉及以下高级任务。您可使用 PE 控制台完成这些任务。有关如何对节点进行分组和分类的详细信息，请参阅 Puppet Enterprise 文档中的对节点进行分组和分类。

1. 创建节点组。
2. 手动将节点添加到组或通过应用您创建的规则来自动执行此操作。
3. 将类分配给节点组。

重置管理员密码和用户密码

有关如何更改您用于登录 Puppet Enterprise 控制台的密码的信息，请参阅 Puppet Enterprise 文档中的重置控制台管理员密码。

默认情况下，在尝试登录 10 次后，将禁止用户登录 Puppet 控制台。有关如何在锁定情况下重置用户密码的更多信息，请参阅 Puppet Enterprise 文档中的密码终端节点。

可选：用 AWS CodeCommit 作 Puppet r10k 远程控制存储库

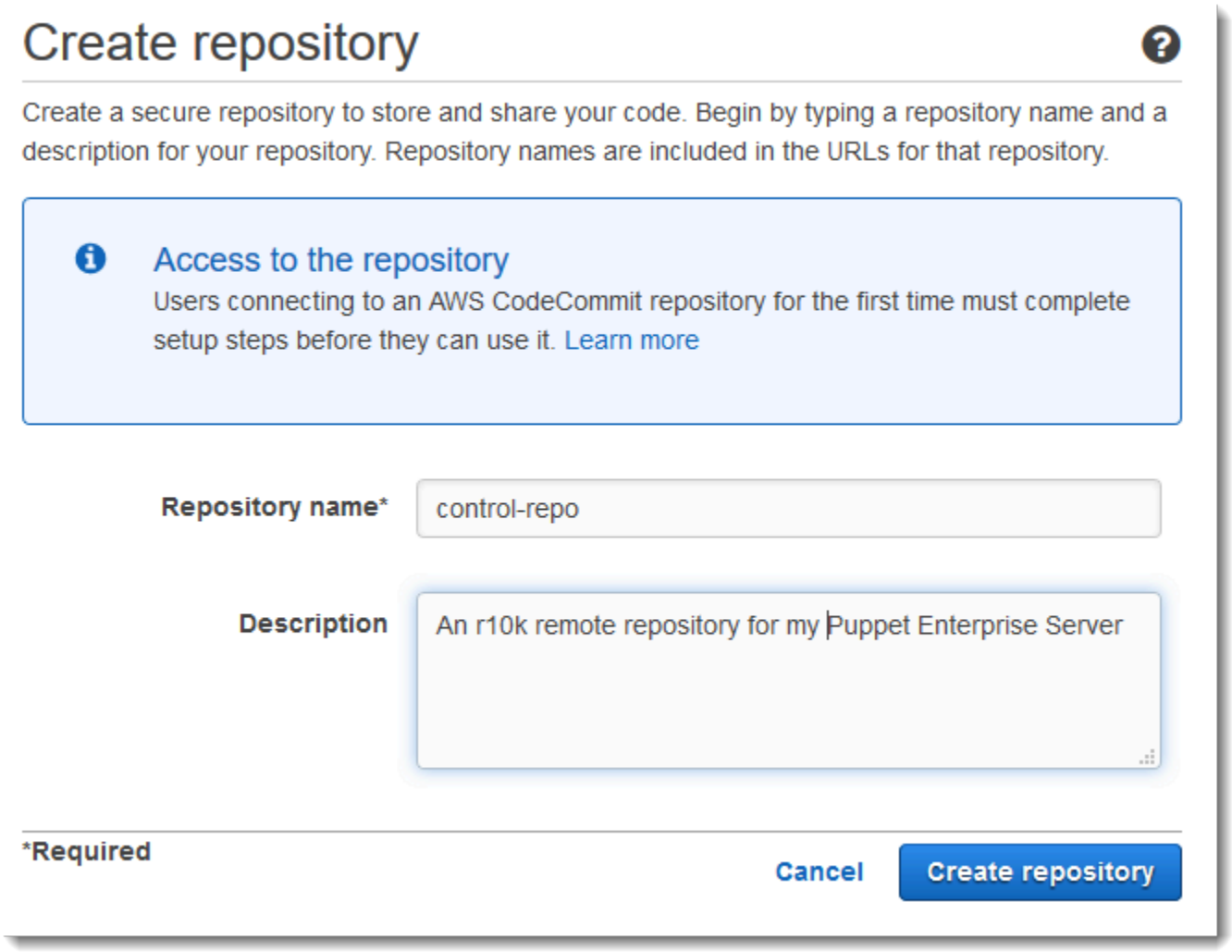
> **Important**
>
> 该 AWS OpsWorks for Puppet Enterprise 服务于 2024 年 3 月 31 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

您可以使用创建新存储库 AWS CodeCommit，并将其用作 r10k 远程控制存储库。要完成本节中的步骤并使用 CodeCommit 存储库，您需要具有AWSCodeCommitReadOnly托管策略所提供权限的用户。

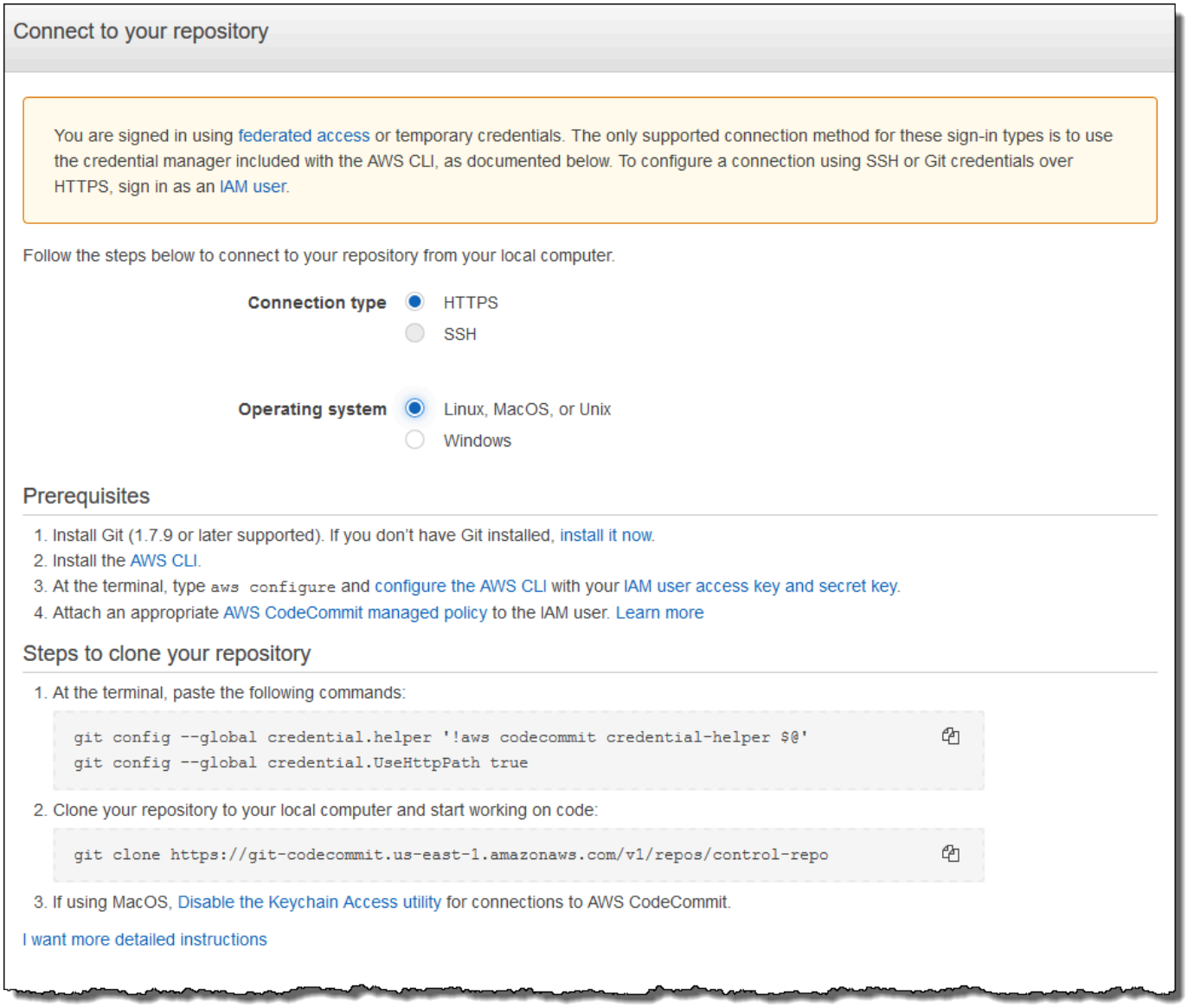
主题

步骤 1：用 CodeCommit 作 HTTPS 连接类型的存储库

1. 在 CodeCommit 控制台中，创建一个新存储库。



2. 选择 Skip 以跳过设置 Amazon SNS 主题这一操作。
3. 在 Code 页面上，选择 Connect to your repository。
4. 在 Connect to your repository 页面上，选择 HTTPS 作为 Connection type，然后选择您的操作系统。



在克隆存储库的步骤区域中，您的 `git clone` URL 应与以下内容类似：`https://git-codecommit.`*`region`*`.amazonaws.com/v1/repos/control-repo`。将此 URL 复制到方便位置以便在 Puppet 服务器设置中使用。

5. 关闭 “连接到您的存储库” 页面，然后返回 Puppet Enterprise 服务器设置。 OpsWorks
6. 将您在步骤 4 中复制的 URL 粘贴到 Puppet Master 设置向导的 Configure credentials 页面中的 r10k remote 字符串框中。将 r10k private key 框保留为空。完成您的 Puppet Master 的创建和启动。
7. 在 IAM 控制台中，将AWSCodeCommitReadOnly策略附加到您的 Puppet 主服务器的实例配置文件角色。有关如何将策略附加到 IAM 角色的更多信息，请参阅 IAM 用户指南 中的添加 IAM 身份权限（控制台）。

8. 按照用户指南中使用 Git 凭据的 HTTPS AWS CodeCommit 用户设置中的步骤将现有`control-repo`内容推送到新 CodeCommit 存储库。
9. 现在，您可以按照 the section called “完成配置”中的说明继续操作，并使用初学者工具包将代码部署到您的 Puppet Master。以下命令是一个示例。

```
puppet-code deploy --all --wait --config-file .config/puppet-code.conf
```

步骤 2：（可选）用 CodeCommit 作 SSH 连接类型的存储库

您可以将 AWS CodeCommit r10k 远程控制存储库配置为使用 SSH key pair 身份验证。在开始此过程之前，必须完成以下先决条件。

- 您必须已使用前一节所述的 OpsWorks HTTPS 控制存储库启动了 for Puppet Enterprise 服务器。the section called “步骤 1：用 CodeCommit 作 HTTPS 连接类型的存储库”必须先完成此操作，以便您能够将所需配置上传到 Puppet Master。
- 请确保您的用户已附加AWSCodeCommitReadOnly托管策略。有关如何创建用户的更多信息，请参阅 IAM 用户指南中的在您的 AWS 账户中创建 IAM 用户。
- 创建一个 SSH 密钥并将该密钥与 用户关联。按照AWS CodeCommit 用户指南中的步骤 3：在 Linux、macOS 或 Unix 上配置凭证中的说明，用`ssh-keygen`创建公有密钥和私有密钥对。

1. 在会 AWS CLI 话中，运行以下命令将私钥文件内容上传到 P AWS Systems Manager arameter Store。你 OpsWorks 的 puppet Enterprise 服务器查询此参数以获取所需的证书文件。将 *`private_key_file`* 替换为您的 SSH 私有密钥文件的路径。

```
aws ssm put-parameter --name puppet_user_pk --type String --value
 "`cat private_key_file`"
```

2. 将 Systems Manager Parameter Store 权限添加到 Puppet Master。
   a. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
   b. 在左侧导航窗格中，选择 Roles（角色）。
   c. 选择 aws-opsworks-cm-ec2 个角色。
   d. 在 Permissions（权限）选项卡上，选择 Attach policies（附加策略）。
   e. 在 Search (搜索) 栏中，输入 **`AmazonSSMManagedInstanceCore`**。
   f. 在搜索结果中，选择 AmazonSSM ManagedInstanceCore。

g. 选择附加策略。

3. 创建配置文件清单。如果您使用的是初学者工具包中提供的 control-repo-example 存储库，请在示例存储库中显示的位置创建以下文件。否则，请根据您自己的控制存储库结构创建这些文件。将 *IAM_USER_SSH_KEY* 值替换为您在此过程的先决条件中创建的 SSH 密钥 ID。

```
control-repo-example/site/profile/manifests/codecommit.pp
```

```
class profile::codecommit {
  $configfile = @(CONFIGFILE)
      Host git-codecommit.*.amazonaws.com
      User IAM_USER_SSH_KEY
      IdentityFile /etc/puppetlabs/puppetserver/ssh/codecommit.rsa
      StrictHostKeyChecking=no
      | CONFIGFILE

  # Replace REGION with the correct region for your server.
  $command = @(COMMAND)
      aws ssm get-parameters \
      --region REGION \
      --names puppet_user_pk \
      --query "Parameters[0].Value" \
      --output text >| /etc/puppetlabs/puppetserver/ssh/codecommit.rsa
      | COMMAND

  $dirs = [
            '/opt/puppetlabs/server/data/puppetserver/.ssh',
            '/etc/puppetlabs/puppetserver/ssh',
          ]

  file { $dirs:
    ensure => 'directory',
    group  => 'pe-puppet',
    owner  => 'pe-puppet',
    mode   => '0750',
  }

  file { 'ssh-config':
    path    => '/opt/puppetlabs/server/data/puppetserver/.ssh/config',
    require => File[$dirs],
    content => $configfile,
    group   => 'pe-puppet',
    owner   => 'pe-puppet',
```

```
    mode    => '0600',
  }

  exec { 'download-codecommit-certificate':
    command => $command,
    require => File[$dirs],
    creates => '/etc/puppetlabs/puppetserver/ssh/codecommit.rsa',
    path    => '/bin',
    cwd     => '/etc/puppetlabs',
  }

  file { 'private-key-permissions':
    subscribe => Exec['download-codecommit-certificate'],
    path      => '/etc/puppetlabs/puppetserver/ssh/codecommit.rsa',
    group     => 'pe-puppet',
    owner     => 'pe-puppet',
    mode      => '0600',
  }
}
```

4. 将您的控制存储库推送到 CodeCommit。运行以下命令以将新的清单文件推送到您的存储库。

```
git add ./site/profile/manifests/codecommit.pp
git commit -m 'Configuring for SSH connection to CodeCommit'
git push origin production
```

5. 部署清单文件。运行以下命令将更新的配置部署到你 OpsWorks 的 for Puppet Enterprise 服务器。将 *STARTER_KIT_DIRECTORY* 替换为 Puppet 配置文件的路径。

```
cd STARTER_KIT_DIRECTORY

puppet-access login --config-file .config/puppetlabs/client-tools/puppet-
access.conf

puppet-code deploy --all --wait \
--config-file .config/puppet-code.conf \
--token-file .config/puppetlabs/token
```

6. 更新 Pupp OpsWorks et Enterprise 服务器的分类。默认情况下，Puppet 代理每 30 分钟在节点（包括 Master）上运行一次。要避免等待，您可以手动在 Puppet Master 上运行代理。运行代理将选取新清单文件。

   a. 登录 Puppet Enterprise 控制台。

b. 选择分类。

c. 展开 PE 基础设施。

d. 选择 PE Master。

e. 在配置选项卡上，在添加新类中输入**`profile::codecommit`**。

新类 `profile::codecommit` 可能不会在运行 `puppet-code deploy` 后立即出现。如果该类未出现，请在此页面上选择刷新。

f. 选择添加类，然后选择提交 1 更改。

g. 在 Puppet Enterprise 服务器上手动运行 Puppet 代理。 OpsWorks 选择节点，在列表中选择您的服务器，再选择运行 Puppet，然后选择运行。

7. 在 Puppet Enterprise 控制台中，更改存储库 URL 以使用 SSH 而不是 HTTPS。您在这些步骤中执行的配置将 OpsWorks 在 Puppet Enterprise 备份和还原过程中保存，因此在维护活动结束后，您无需手动更改存储库配置。

a. 选择分类。

b. 展开 PE 基础设施。

c. 选择 PE Master。

d. 在配置选项卡上，找到 `puppet_enterprise::profile::master` 类。

e. 选择 `r10k_remote` 参数旁边的编辑。

f. 将 HTTPS URL 替换为存储库的 SSH URL，然后选择提交 1 更改。

g. 在 Puppet Enterprise 服务器上手动运行 Puppet 代理。 OpsWorks 选择节点，在列表中选择您的服务器，再选择运行 Puppet，然后选择运行。

使用创建 AWS OpsWorks for Puppet Enterprise 大师 AWS CloudFormation

> **Important**
>
> 该 AWS OpsWorks for Puppet Enterprise 服务于 2024 年 3 月 31 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

AWS OpsWorks for Puppet Enterprise 允许你在中运行 Pu ppet Enterpris e 服务器。 AWS您可以在大约 15 分钟内预配置一个 Puppet Enterprise Master 服务器。

从 2021 年 5 月 3 日起，Puppet Enterprise 将在其中存储一些 Puppet Enterprise OpsWorks AWS Secrets Manager有关更多信息，请参阅 与 AWS Secrets Manager 集成。

以下演练通过在中创建堆栈来帮助你在 Puppet Enterprise 中 OpsWorks 创建 Puppet 大师。 AWS CloudFormation

主题

先决条件

在创建新的 Puppet 大师之前，请在外部 OpsWorks 为 Puppet Enterprise 创建访问和管理人偶大师所需的资源。有关更多信息，请参阅本指南的“入门”部分的先决条件 。

如果要创建使用自定义域的服务器，则需要自定义域、证书和私有密钥。您必须在 AWS CloudFormation 模板中为所有这三个参数指定值。有关、和`CustomPrivateKey`参数要求的更多信息`CustomDomainCustomCertificate`，请参阅 AWS OpsWorks CM API 参考CreateServer中的。

查看《AWS CloudFormation 用户指南模板参考》的 “OpsWorks-CM” 部分，了解用于创建服务器的 AWS CloudFormation 模板中支持的值和必填值。

在 AWS CloudFormation中创建 Puppet Enterprise Master

本节介绍如何使用 AWS CloudFormation 模板构建堆栈，从而为 Puppet Enterprise 主服务器创建一个 OpsWorks 。您可以使用 AWS CloudFormation 控制台或 AWS CLI. 您可以使用示例 AWS CloudFormation 模板来构建 Puppet Enterpr OpsWorks ise 服务器堆栈。请务必使用您自己的服务器名称、IAM 角色、实例配置文件、服务器描述、备份保留计数、维护选项和可选标签来更新示例模板。如果您的服务器将使用自定义域，则必须在 AWS CloudFormation 模板中为 `CustomDomain`、`CustomCertificate` 和 `CustomPrivateKey` 参数指定值。有关这些选项的更多信息，请参阅本指南的“入门”部分的the section called “使用创建 Puppet Enterprise Master AWS Management Console”。

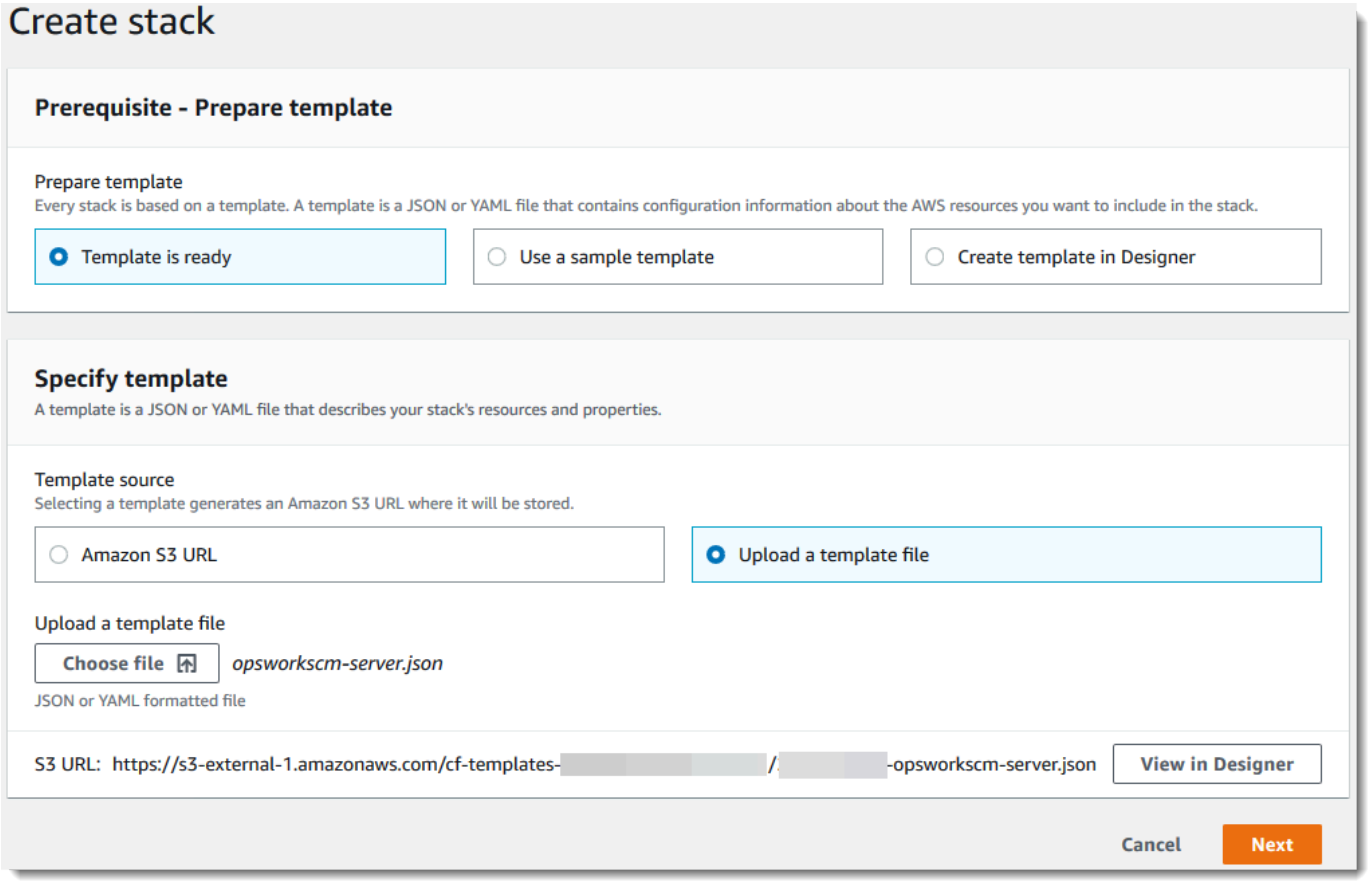
主题

- 使用 AWS CloudFormation (CLI) 创建 Puppet Enterprise Master

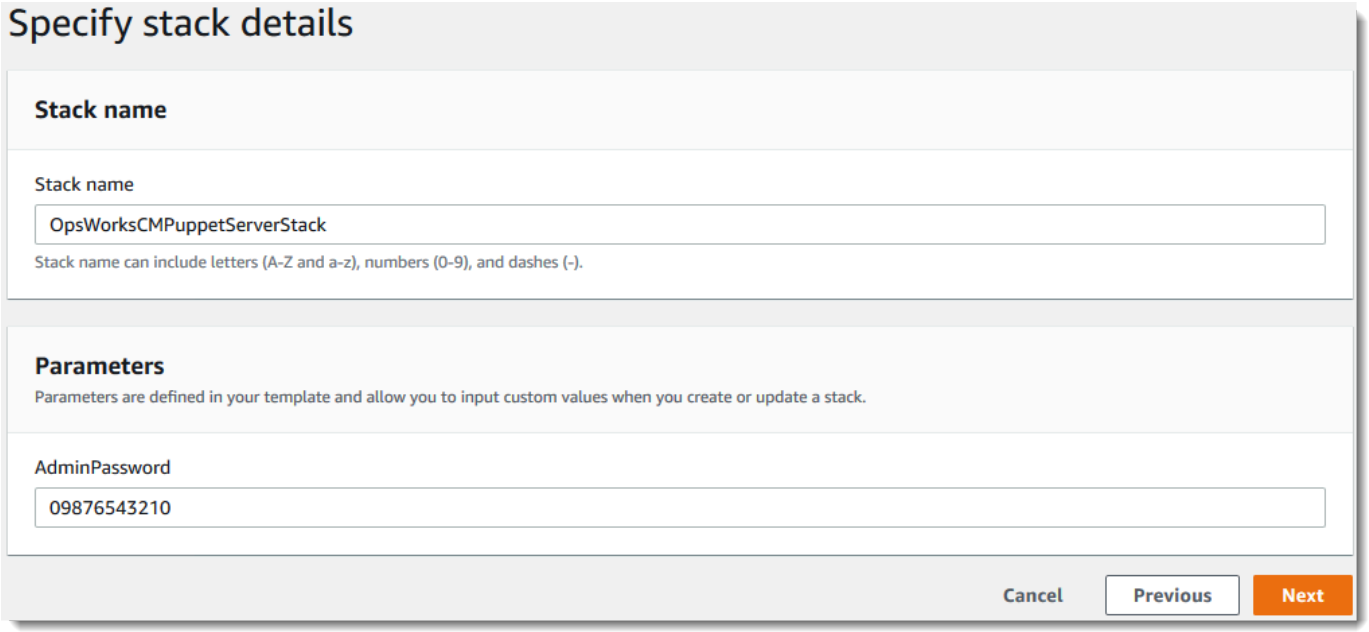
使用 AWS CloudFormation （控制台）创建 Puppet Enterprise Master

1. 登录 AWS Management Console 并打开 AWS CloudFormation 控制台，网址为 https://console.aws.amazon.com/cloudformation。
2. 在 AWS CloudFormation 主页上，选择创建堆栈。
3. 在 Prerequisite - Prepare template (先决条件 – 准备模板) 中，如果您使用的是示例 AWS CloudFormation 模板，请选择 Template is ready (模板已准备就绪)。
4. 在 Specify template (指定模板) 中，选择模板的源。在本演练中，选择上传模板文件，然后上传用于创建 Puppet Enterprise 服务器的 AWS CloudFormation 模板。浏览查找您的模板文件，然后选择 Next (下一步)。

   AWS CloudFormation 模板可以采用 YAML 或 JSON 格式。有一个示例 AWS CloudFormation 模板可供您使用；请务必用自己的示例值替换示例值。您可以使用 AWS CloudFormation 模板设计器来构建新模板或验证现有模板。有关如何执行此操作的更多信息，请参阅《AWS CloudFormation 用户指南》中的AWS CloudFormation Designer 界面概述。



5. 在 指定详细信息 页面上，输入您的堆栈的名称。这将不会与您的服务器的名称相同，它仅仅是一个堆栈名称。在参数区域中，输入用于登录 Puppet Enterprise 控制台网页的管理员密码。该密码必须使用介于 8 和 32 个之间的 ASCII 字符。选择下一步。



6. 在选项页面上，您可以向您使用堆栈创建的服务器添加标签，如果您尚未在您的模板中指定要使用的 IAM 角色，还可以选择用于创建资源的 IAM 角色。在指定选项之后，选择 Next (下一步)。有关高级选项（例如回滚触发器）的更多信息，请参阅《AWS CloudFormation 用户指南》中的设置 AWS CloudFormation 堆栈选项。

7. 在审核页面上，审核您的选择。在准备好创建服务器堆栈时，选择创建堆栈。

   在等待创建堆栈时， AWS CloudFormation 请查看堆栈创建状态。如果堆栈创建失败，请查看控制台中显示的错误消息，以帮助您解决问题。有关对 AWS CloudFormation 堆栈中的错误进行故障排除的更多信息，请参阅《AWS CloudFormation 用户指南》中的排查错误。

   服务器创建完成后，您的 Puppet 大师将在 Puppet Enterprise 的主页上可用，状态 OpsWorks 为在线。服务器联机之后，Puppet Enterprise 控制台在服务器的域上可用，位于以下格式的 URL 上：`https://`*`your_server_name`*`-`*`randomID`*`.`*`region`*`.opsworks-cm.io`。

   > **Note**
   >
   > 如果您为服务器指定了自定义域、证书和私钥，请在企业的 DNS 管理工具中创建一个 CNAME 条目，该条目将您的自定义域映射到 Puppet Enterprise OpsWorks 为服务器自动生成的终端节点。在将生成的端点映射到自定义域值之前，无法管理服务器或连接到服务器的 Puppet Enterprise 管理网站。

要获取生成的端点值，请在服务器联机后运行以下 AWS CLI 命令：

```
aws opsworks describe-servers --server-name server_name
```

使用 AWS CloudFormation (CLI) 创建 Puppet Enterprise Master

如果您的本地计算机尚未运行 AWS CLI，请 AWS CLI 按照 AWS 命令行界面用户指南中的安装说明下载并安装。本部分未介绍可以与 create-stack 命令结合使用的所有参数。有关 create-stack 参数的更多信息，请参阅 create-stack 参考 中的 AWS CLI 。

1. 请务必完成为 Puppe OpsWorks t Enterprise 大师创建。先决条件
2. 创建服务角色和实例配置文件。 AWS OpsWorks 提供了一个可用于创建两者的 AWS CloudFormation 模板。运行以下 AWS CLI 命令创建一个 AWS CloudFormation 堆栈，用于为您创建服务角色和实例配置文件。

   ```
   aws cloudformation create-stack --stack-name OpsWorksCMRoles --template-url
    https://s3.amazonaws.com/opsworks-cm-us-east-1-prod-default-assets/misc/opsworks-
   cm-roles.yaml --capabilities CAPABILITY_NAMED_IAM
   ```

   创建 AWS CloudFormation 完堆栈后，在您的账户中查找并复制服务角色的 ARN。

   ```
   aws iam list-roles --path-prefix "/service-role/" --no-paginate
   ```

   在 `list-roles` 命令的结果中，查找类似于以下内容的服务角色和实例配置文件条目。记下服务角色和实例配置文件的 ARN，然后将其添加到用于创建 Puppet 主服务器堆栈的 AWS CloudFormation 模板中。

   ```
   {
       "AssumeRolePolicyDocument": {
           "Version": "2012-10-17",
           "Statement": [
               {
                   "Action": "sts:AssumeRole",
                   "Effect": "Allow",
                   "Principal": {
                       "Service": "ec2.amazonaws.com"
                   }
               }
   ```

```
        ]
    },
    "RoleId": "AROZZZZZZZZZZQG6R22HC",
    "CreateDate": "2018-01-05T20:42:20Z",
    "RoleName": "aws-opsworks-cm-ec2-role",
    "Path": "/service-role/",
    "Arn": "arn:aws:iam::000000000000:role/service-role/aws-opsworks-cm-ec2-role"
},
{
    "AssumeRolePolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Action": "sts:AssumeRole",
                "Effect": "Allow",
                "Principal": {
                    "Service": "opsworks-cm.amazonaws.com"
                }
            }
        ]
    },
    "RoleId": "AROZZZZZZZZZZZZZZZ6QE",
    "CreateDate": "2018-01-05T20:42:20Z",
    "RoleName": "aws-opsworks-cm-service-role",
    "Path": "/service-role/",
    "Arn": "arn:aws:iam::000000000000:role/service-role/aws-opsworks-cm-service-
role"
}
```

3. 再次运行该create-stack命令 OpsWorks 为 Puppet Enterprise Master 创建。

   - 将 *stack_name* 替换为您的堆栈的名称。这是 AWS CloudFormation 堆栈的名称，不是你的 Puppet 大师。Puppet master 名称是 AWS CloudFormation 模板中 `ServerName` 的值。
   - 将 *template* 替换为您的模板文件的路径，并将扩展名 *yaml # json* 相应地替换为 `.yaml` 或 `.json`。
   - 的值对`--parameters`应于 CreateServerAPI EngineAttributes中的值。对于 Puppet，以下是用户提供的用于创建服务器的引擎属性。r10k 引擎属性将你的 Puppet 主服务器连接到代码存储库以管理服务器的环境配置。有关 r10k 引擎属性的更多信息，请参阅 Puppet Enterprise 文档中的使用 r10k 来管理代码。

- PUPPET_ADMIN_PASSWORD，登录 Puppet Enterprise 控制台网页的管理员密码。密码必须使用 8 到 32 个 ASCII 字符，至少需要一个大写字母、一个小写字母、一个数字和一个特殊字符。
- PUPPET_R10K_REMOTE，您控件存储库的 URL（例如 ssh://git@your.git-repo.com:user/control-repo.git）。指定 r10k remote 可打开 TCP 端口 8170。
- PUPPET_R10K_PRIVATE_KEY。如果您使用的是私有 Git 存储库，请添加 PUPPET_R10K_PRIVATE_KEY 以指定一个 SSH URL 和一个 PEM 编码的私有 SSH 密钥。

```
aws cloudformation create-stack --stack-name stack_name
 --template-body file://template.yaml or json --parameters
 ParameterKey=AdminPassword,ParameterValue="password"
```

示例如下：

```
aws cloudformation create-stack --stack-name "OpsWorksCMPuppetServerStack"
 --template-body file://opsworkscm-puppet-server.json --parameters
 ParameterKey=AdminPassword,ParameterValue="09876543210Ab#"
```

以下示例将 r10k 引擎属性指定为参数，而模板中未提供这些属性。 AWS CloudFormation 一个包含 r10k 引擎属性的示例模板-puppet-server-param-attributes.yaml，包含在示例 AWS CloudFormation 模板中。

```
aws cloudformation create-stack --stack-name MyPuppetStack --
template-body file://puppet-server-param-attributes.yaml --parameters
 ParameterKey=AdminPassword,ParameterValue="superSecret1%3"
 ParameterKey=R10KRemote,ParameterValue="https://www.yourRemote.com"
 ParameterKey=R10KKey,ParameterValue="$(cat puppet-r10k.pem)"
```

以下示例在 AWS CloudFormation 模板中指定 r10k 引擎属性及其值；该命令只需指向模板文件即可。指定为 --template-body 的值的模板-puppet-server-in-file-attributes.yaml，包含在示例 AWS CloudFormation 模板中。

```
aws cloudformation create-stack --stack-name MyPuppetStack --template-body file://
puppet-server-in-file-attributes.yaml
```

4. （可选）要获取堆栈创建状态，请运行以下命令。

```
aws cloudformation describe-stacks --stack-name stack_name
```

5. 堆栈创建完成后，请继续执行下一节-the section called “完成配置”。如果堆栈创建失败，请查看控制台中显示的错误消息，以帮助您解决问题。有关对 AWS CloudFormation 堆栈中的错误进行故障排除的更多信息，请参阅《AWS CloudFormation 用户指南》中的故障排除。

更新 Pupp OpsWorks et Enterprise 服务器以使用自定义域

> **Important**
>
> 该 AWS OpsWorks for Puppet Enterprise 服务于 2024 年 3 月 31 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

本节介绍如何使用服务器 OpsWorks 的备份创建新服务器，更新现有的 Puppet Enterprise 服务器以使用自定义域和证书。本质上，您是在复制现有 OpsWorks 的 Puppet Enterprise 2.0 服务器，方法是从备份中创建新服务器，然后将新服务器配置为使用自定义域、证书和私钥。

主题

先决条件

以下是更新现有 OpsWorks 的 Puppet Enterprise 服务器以使用自定义域和证书的要求。

- 要更新（或复制）的服务器必须运行 Puppet Enterprise 2019.8.5。
- 决定要用于创建新服务器的备份。您必须至少有一个要更新的服务器的备份可用。有关 Puppet Enterprise 中备份 OpsWorks 的更多信息，请参阅为 Puppet Enterprise 服务器 OpsWorks 备份。
- 准备好用于创建现有服务器（作为备份源）的服务角色和实例配置文件 ARN。
- 请确保您正在运行最新版本的 AWS CLI。有关更新 AWS CLI 工具的更多信息，请参阅 AWS 命令行界面用户指南 AWS CLI中的安装。

限制

通过使用备份创建新服务器来更新现有服务器时，新服务器不能与 Puppet Enterprise 服务器 OpsWorks 的现有服务器完全相同。

- 您只能使用 AWS CLI 或其中一个AWS 软件开发工具包来完成此过程。无法通过使用 AWS Management Console从备份创建新的服务器。
- 新服务器不能使用与账户内和 Amazon Web Services Region 内的现有服务器相同的名称。名称必须与用作备份源的现有服务器不同。
- 连接到现有服务器的节点不由新服务器管理。您必须执行以下操作之一。
  - 附加不同的节点，因为节点不能由多个 Puppet 主服务器管理。
  - 将节点从现有服务器（备份源）迁移到新服务器和新的自定义域端点。有关如何迁移节点的更多信息，请参阅 Puppet Enterprise 文档。

更新服务器以使用自定义域

要更新现有的 Puppet 主服务器，您可以通过运行 `create-server` 命令、添加参数以指定备份、自定义域、自定义证书和自定义私有密钥来创建该服务器的副本。

1. 如果您的 `create-server` 命令中没有可供指定的服务角色或实例配置文件 ARN，请按照使用 Chef Automate 服务器创建 AWS CLI中的步骤 1-5 创建您可以使用的服务角色和实例配置文件。
2. 如果您尚未执行此操作，请查找现有 Puppet 主服务器的备份，将以此备份为基础创建使用自定义域的新服务器。运行以下命令以显示有关您的账户和区域中 Puppet Enterprise 备份的所有 OpsWorks 信息。确保记下要使用的备份的 ID。

   ```
   aws opsworks-cm --region region name describe-backups
   ```

3. 运行`create-server`命令 OpsWorks 为 Puppet Enterprise 服务器创建。

   - `--engine` 值为 `Puppet`，`--engine-model` 为 `Monolithic`，`--engine-version` 为 `2019` 或 `2017`。
   - 在您的 AWS 账户中，服务器名称在每个区域内必须是唯一的。服务器名称必须以字母开头；然后允许字母、数字或连字符 (-)，最多 40 个字符。
   - 使用您在步骤 3 和 4 中复制的实例配置文件 ARN 和服务角色 ARN。
   - 有效实例类型为 `c4.large`、`c4.xlarge` 或 `c4.2xlarge`。有关这些实例类型的规范的更多信息，请参阅 Amazon EC2 用户指南中的实例类型。

- `--engine-attributes` 参数是可选的；如果您不指定 Puppet 管理员密码，则服务器创建过程会为您生成一个密码。如果您添加 `--engine-attributes`，请指定 `PUPPET_ADMIN_PASSWORD`，登录 Puppet Enterprise 控制台网页的管理员密码。该密码必须使用介于 8 和 32 个之间的 ASCII 字符。
- SSH 密钥对是可选的，但可以帮助您连接到 Puppet Master (如果您需要重置控制台管理员密码)。有关创建 SSH 密钥对的更多信息，请参阅《Amazon EC2 用户指南》中的 Amazon EC2 密钥对。
- 要使用自定义域，请将以下参数添加到命令中。否则，Puppet 主服务器创建过程会自动为您生成端点。配置自定义域需要所有三个参数。有关使用这些参数的其他要求的信息，请参阅 AWS OpsWorks CM API 参考CreateServer中的。
  - `--custom-domain` - 服务器的可选公有端点，例如 `https://aws.my-company.com`。
  - `--custom-certificate` - PEM 格式的 HTTPS 证书。该值可以是单个自签名证书或证书链。
  - `--custom-private-key` - PEM 格式的私有密钥，用于通过 HTTPS 连接到服务器。私有密钥不得加密；无法使用密码或密码短语保护它。
- 需要每周进行系统维护。有效值必须按以下格式指定：`DDD:HH:MM`。指定的时间为协调世界时 (UTC)。如果您不指定 `--preferred-maintenance-window` 的值，则默认值为星期二、星期三或星期五的一小时随机时间段。
- `--preferred-backup-window` 的有效值必须按以下格式之一指定：`HH:MM` (针对每日备份) 或 `DDD:HH:MM` (针对每周备份)。指定的时间采用 UTC 格式。默认值为随机每日开始时间。要退出自动备份，请改为添加参数 `--disable-automated-backup`。
- 对于 `--security-group-ids`，输入一个或多个安全组 ID，用空格分隔。
- 对于 `--subnet-ids`，输入子网 ID。

```
aws opsworks-cm create-server --engine "Puppet" --engine-model "Monolithic"
 --engine-version "2019" --server-name "server_name" --instance-profile-arn
 "instance_profile_ARN" --instance-type "instance_type" --engine-attributes
 '{"PUPPET_ADMIN_PASSWORD":"ASCII_password"}' --key-pair "key_pair_name" --
preferred-maintenance-window "ddd:hh:mm" --preferred-backup-window "ddd:hh:mm"
 --security-group-ids security_group_id1 security_group_id2 --service-role-arn
 "service_role_ARN" --subnet-ids subnet_ID
```

以下示例创建使用自定义域的 Puppet 主服务器。

```
aws opsworks-cm create-server \
```

```
    --engine "Puppet" \
    --engine-model "Monolithic" \
    --engine-version "2019" \
    --server-name "puppet-02" \
    --instance-profile-arn "arn:aws:iam::1019881987024:instance-profile/aws-
opsworks-cm-ec2-role" \
    --instance-type "c4.large" \
    --engine-attributes '{"PUPPET_ADMIN_PASSWORD":"zZZzDj2DLYXSZFRv1d"}' \
    --custom-domain "my-puppet-master.my-corp.com" \
    --custom-certificate "-----BEGIN CERTIFICATE----- EXAMPLEqEXAMPLE== -----END
 CERTIFICATE-----" \
    --custom-private-key "-----BEGIN RSA PRIVATE KEY----- EXAMPLEqEXAMPLE= -----END
 RSA PRIVATE KEY-----" \
    --key-pair "amazon-test"
    --preferred-maintenance-window "Mon:08:00" \
    --preferred-backup-window "Sun:02:00" \
    --security-group-ids sg-b00000001 sg-b0000008 \
    --service-role-arn "arn:aws:iam::044726508045:role/service-role/aws-opsworks-
cm-service-role" \
    --subnet-ids subnet-383daa71
```

4. OpsWorks 对于 Puppet Enterprise，创建新服务器大约需要 15 分钟。在 `create-server` 命令的输出中，复制 `Endpoint` 属性的值。示例如下：

```
"Endpoint": "puppet-2019-exampleexample.opsworks-cm.us-east-1.amazonaws.com"
```

请勿关闭 `create-server` 命令的输出或关闭您的 Shell 会话，因为该输出可能包含不再显示的重要信息。要从 `create-server` 结果中获取密码和初学者工具包，请继续执行下一步。

5. 如果你选择让 Puppet Enterprise 为你生成密码，你可以使用 jq 等 JSON 处理器从`create-server`结果中提取出一个可用的格式。 OpsWorks 安装 jq 后，您可以运行以下命令来提取 Puppet 管理员密码和初学者工具包。如果您未在步骤 3 中提供自己的密码，请确保将提取的管理员密码保存在方便且安全的位置。

```
#Get the Puppet password:
cat resp.json | jq -r '.Server.EngineAttributes[] | select(.Name ==
 "PUPPET_ADMIN_PASSWORD") | .Value'

#Get the Puppet Starter Kit:
cat resp.json | jq -r '.Server.EngineAttributes[] | select(.Name ==
 "PUPPET_STARTER_KIT") | .Value' | base64 -D > starterkit.zip
```

> **Note**
>
> 在 AWS Management Console中，您无法重新生成新的 Puppet Master 初学者工具包。使用创建 Puppet 大师时 AWS CLI，请运行前面的`jq`命令将`create-server`结果中的 base64 编码的入门套件保存为 ZIP 文件。

6. 或者，如果您没有从`create-server`命令结果中提取入门套件，则可以从 for Puppet Enterprise 控制台的服务器属性页面下载新的入门套件。 OpsWorks
7. 如果您不使用自定义域，请继续下一步。如果您在服务器上使用自定义域，请在企业的 DNS 管理工具中创建 CNAME 条目，将您的自定义域指向您在步骤 4 中复制的 for Puppet Enterprise 终端节点。 OpsWorks 在完成此步骤之前，您无法访问或登录具有自定义域的服务器。
8. 服务器创建过程完成后，请转到使用初学者工具包配置 Puppet Master。

另请参阅

- 使用创建 Puppet Enterprise Master AWS CLI
- 备份和恢复 Puppe OpsWorks t Enterprise Server
- CreateServer在 AWS OpsWorks CM API 参考中
- 《AWS CLI Command Reference》中的 create-server。

使用 AWS OpsWorks for Puppet Enterprise 资源上的标签

> **Important**
>
> 该 AWS OpsWorks for Puppet Enterprise 服务于 2024 年 3 月 31 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

标签是一些充当元数据的词和短语，用于标识和组织 AWS 资源。在 OpsWorks Puppet Enterprise 中，一个资源最多可以有 50 个用户应用的标签。每个标签由一个键和一个可选的值组成。您可以为 Puppet Enterprise 中的 OpsWorks 以下资源应用标签：

- OpsWorks 适用于 Puppet 企业服务器

- Pupp OpsWorks et 企业服务器的备份

AWS 资源标签可以帮助您跟踪成本、控制对资源的访问权限、对资源进行分组以实现任务自动化，或者按用途或生命周期阶段组织资源。有关标签优势的更多信息，请参阅 AWS Answers 中的 AWS 标签策略和《AWS Billing and Cost Management 用户指南》中的使用成本分配标签。

要使用标签控制对 Puppet Enterprise 服务器或备份的访问权限，请在 AWS Identity and Access Management (IAM) 中创建或编辑策略声明。 OpsWorks 有关更多信息，请参阅《AWS Identity and Access Management IAM 用户指南》中的使用资源标签控制对 AWS 资源的访问。

当您将标签应用于 Puppet Enterprise 主服务器时，标签还会应用于主服务器的备份、存储备份的 Amazon S3 存储桶、主服务器的 Amazon EC2 实例 AWS Secrets Manager、存储在中的主服务器的密钥以及主服务器使用的弹性 IP 地址。 OpsWorks 标签不会传播到用于创建 Pupp AWS OpsWorks et 大师的 AWS CloudFormation 堆栈。

主题

标签的工作原理 AWS OpsWorks for Puppet Enterprise

在此版本中，您可以使用 AWS OpsWorks CM API 或 AWS Management Console添加和管理标签。AWS OpsWorks CM 还尝试将您添加到服务器的标签添加到与服务器关联的 AWS 资源中，包括 EC2 实例、Secrets Manager 中的密钥、弹性 IP 地址、安全组、S3 存储桶和备份。

下表概述了如何在 Puppet Enterprise 中 OpsWorks 添加和管理标签。

操作	要使用的内容
OpsWorks 为新的 Puppet Enterprise 服务器或手动创建的备份添加标签。	• 选择 Create Puppet Enterprise server (创建 Puppet Enterprise 服务器)，然后在 Configure advanced settings (配置高级设置) 页面上添加标签。

操作	要使用的内容
	• 在备份页面上为现有服务器选择创建备份，然后在创建 Puppet Enterprise 服务器的备份页面上添加标签。 • 向 CreateServer 或 CreateBackup 命令添加 Tags 参数。
查看资源上的标签。	• 在服务器的详细信息页面上，选择导航窗格中的 Tags (标签)。 • 在服务器的 Backups (备份) 页面上，选择一个备份，然后选择 Edit backup (编辑备份)。 • 运行 ListTagsForResource 命令。
OpsWorks 为现有的 Puppet Enterprise 服务器或备份添加标签，无论备份是手动创建还是自动创建。	• 在服务器的详细信息页面上，选择导航窗格中的 Tags (标签)，然后选择 Edit (编辑)。 • 在服务器的 Backups (备份) 页面上，选择一个备份，然后选择 Edit backup (编辑备份)。 • 运行 TagResource 命令。
从资源中删除标签。	• 在服务器的详细信息页面上，选择导航窗格中的 Tags (标签)，然后选择 Edit (编辑)。选择要删除的标签旁边的 X。 • 在服务器的 Backups (备份) 页面上，选择一个备份，然后选择 Edit backup (编辑备份)。选择要删除的标签旁边的 X。 • 运行 UntagResource 命令。

DescribeServers 和 DescribeBackups 响应不包含标签信息。要显示标签，请使用 ListTagsForResource API。

在 Puppet Enterprise（控制台）中 OpsWorks 添加和管理标签

本节中的过程将在 AWS Management Console中执行。

如果添加标签，标签密钥不能为空。密钥最多可包含 127 个字符，并且只能包含 Unicode 字母、数字或分隔符，或以下特殊字符：+ - = . _ : / @。标签值是可选的。您可以添加具有密钥但没有值的

标签。值最多可包含 255 个字符，并且只能包含 Unicode 字母、数字或分隔符，或以下特殊字符：+ - = . _ : / @

主题

为 Puppet Enterprise Server（控制台）的新 OpsWorks 版本添加标签

1. 请务必完成创建 Puppet Enterpr OpsWorks ise 主版的所有先决条件。
2. 按照使用创建 Puppet Enterprise Master AWS Management Console中的步骤 1-8 操作。
3. 指定自动备份设置后，在配置高级设置页面的标签区域中添加标签。您最多可添加 50 个标签。添加完标签后，选择 Next。
4. 继续前往使用创建 Puppet Enterprise Master AWS Management Console的步骤 11，并查看您为新服务器选择的设置。

向新备份添加标签（控制台）

1. 在 f OpsWorks or Puppet Enterprise 主页上，选择现有的 Puppet 大师。
2. 从服务器的详细信息页面中，选择导航窗格中的 Backups (备份) 。
3. 在 Backups (备份) 页面上，选择 Create backup (创建备份)。
4. 添加标签。在添加完标签后，请选择 Create (创建)。

在现有服务器上添加或查看标签（控制台）

1. 在 f OpsWorks or Puppet Enterprise 主页上，选择现有的 Puppet 大师以打开其详细信息页面。
2. 在导航窗格中选择 Tags (标签)，或在详细信息页面底部选择 View all tags (查看所有标签)。
3. 在 Tags (标签) 页面上，选择 Edit (编辑)。
4. 在服务器上添加或编辑标签。完成后，选择 Save (保存)。

> **Note**
>
> 请注意，更改 Puppet 主服务器上的标签也会更改与服务器关联的资源上的标签，例如 EC2 实例、弹性 IP 地址、安全组、S3 存储桶和备份。

在现有备份上添加或查看标签（控制台）

1. 在 f OpsWorks or Puppet Enterprise 主页上，选择现有的 Puppet 大师以打开其详细信息页面。
2. 在导航窗格中选择 Backups (备份)，或在详细信息页面的 Recent backups (最近备份) 区域中选择 View all backups (查看所有备份)。
3. 在 Backups (备份) 页面上，选择要管理的备份，然后选择 Edit backup (编辑备份)。
4. 在备份上添加或编辑标签。完成后，选择 Update (更新)。

从服务器中删除标签（控制台）

1. 在 f OpsWorks or Puppet Enterprise 主页上，选择现有的 Puppet 大师以打开其详细信息页面。
2. 在导航窗格中选择 Tags (标签)，或在详细信息页面底部选择 View all tags (查看所有标签)。
3. 在 Tags (标签) 页面上，选择 Edit (编辑)。
4. 选择标签旁边的 X 以删除标签。完成后，选择 Save (保存)。

> **Note**
>
> 请注意，更改 Puppet 主服务器上的标签也会更改与服务器关联的资源上的标签，例如 EC2 实例、弹性 IP 地址、安全组、S3 存储桶和备份。

从备份中删除标签（控制台）

1. 在 f OpsWorks or Puppet Enterprise 主页上，选择现有的 Puppet 大师以打开其详细信息页面。
2. 在导航窗格中选择 Backups (备份)，或在详细信息页面的 Recent backups (最近备份) 区域中选择 View all backups (查看所有备份)。
3. 在 Backups (备份) 页面上，选择要管理的备份，然后选择 Edit backup (编辑备份)。
4. 选择标签旁边的 X 以删除标签。完成后，选择 Update (更新)。

在 Puppet Enterprise (CLI) 中 OpsWorks 添加和管理标签

本节中的过程将在 AWS CLI中执行。在开始使用标签 AWS CLI 之前，请务必运行的是的最新版本。有关安装或更新的更多信息 AWS CLI，请参阅AWS Command Line Interface 用户指南 AWS CLI中的安装。

如果添加标签，标签密钥不能为空。密钥最多可包含 127 个字符，并且只能包含 Unicode 字母、数字或分隔符，或以下特殊字符：+ - = . _ : / @。标签值是可选的。您可以添加具有密钥但没有值的标签。值最多可包含 255 个字符，并且只能包含 Unicode 字母、数字或分隔符，或以下特殊字符：+ - = . _ : / @

主题

- 为 Puppet Enterprise Server (CLI) 的新 OpsWorks 版本添加标签
- 向新备份添加标签 (CLI)
- 将标签添加到现有服务器或备份 (CLI)
- 列出资源标签 (CLI)
- 从资源中删除标签 (CLI)

为 Puppet Enterprise Server (CLI) 的新 OpsWorks 版本添加标签

在 AWS CLI 创建 Puppet Enterprise OpsWorks 服务器时，可以使用来添加标签。此过程并不完整描述如何创建服务器。有关如何使用创建适用 OpsWorks 于 Puppet Enterprise 服务器的详细信息 AWS CLI，请参阅本指南使用创建 Puppet Enterprise Master AWS CLI中的。您最多可以向服务器添加 50 个标签。

1. 请务必完成创建 Puppet Enterpr OpsWorks ise 服务器的所有先决条件。
2. 完成使用创建 Puppet Enterprise Master AWS CLI的步骤 1-4。
3. 对于步骤 5，当您运行 `create-server` 命令时，请将 `--tags` 参数添加到命令，如以下示例所示。

```
aws opsworks-cm create-server ... --tags Key=Key1,Value=Value1
 Key=Key2,Value=Value2
```

以下是仅显示 `create-server` 命令的标签部分的示例。

```
aws opsworks-cm create-server ... --tags Key=Stage,Value=Production
 Key=Department,Value=Marketing
```

4. 完成使用创建 Puppet Enterprise Master AWS CLI 中的其余步骤。要验证标签是否已添加到新服务器，请按照本主题的列出资源标签 (CLI)中的步骤操作。

向新备份添加标签 (CLI)

在 AWS CLI 为 Puppet Enterpr OpsWorks ise 服务器创建新的手动备份时，可以使用来添加标签。此过程并不完整描述如何创建手动备份。有关如何创建手动备份的详细信息，请参阅中的 “要执行手动备份 AWS CLI” 为 Puppet Enterprise 服务器 OpsWorks 备份。您最多可以向备份添加 50 个标签。如果服务器具有标签，则会自动使用服务器的标签标记新的备份。

默认情况下，当您为 Puppet Enterprise 服务器创建新的 OpsWorks 服务器时，会启用自动备份。您可以通过运行 `tag-resource` 命令向自动备份添加标签，如本主题的将标签添加到现有服务器或备份 (CLI)中所述。

- 要在创建备份时向手动备份添加标签，请运行以下命令。仅显示命令的标签部分。有关完整 `create-backup` 命令的示例，请参阅 为 Puppet Enterprise 服务器 OpsWorks 备份 中的 “在 AWS CLI执行手动备份”。

  ```
  aws opsworks-cm create-backup ... --tags Key=Key1,Value=Value1
   Key=Key2,Value=Value2
  ```

  以下示例仅显示 `create-backup` 命令的标签部分。

  ```
  aws opsworks-cm create-backup ... --tags Key=Stage,Value=Production
   Key=Department,Value=Marketing
  ```

将标签添加到现有服务器或备份 (CLI)

您可以运行该`tag-resource`命令 OpsWorks 为现有的 Puppet Enterprise 服务器或备份添加标签（无论备份是自动创建还是手动创建）。指定目标资源的 Amazon 资源编号 (ARN) 以向其添加标签。

1. 要获取要应用标签的资源的 ARN，请执行以下操作：

   - 对于服务器，请运行 `describe-servers --server-name` *`server_name`*。命令的结果显示服务器 ARN。

- 对于备份，请运行 `describe-backups --backup-id` *`backup_ID`*。命令的结果显示备份 ARN。您还可以运行`describe-backups --server-name` *`server_name`*以显示有关特定 OpsWorks 的 Puppet Enterprise 服务器的所有备份的信息。

以下示例仅显示 `describe-servers --server-name opsworks-cm-test` 命令的结果中的 `ServerArn`。`ServerArn` 值将添加到 `tag-resource` 命令中以向服务器添加标签。

```
{
    "Servers": [
        {
            ...
            "ServerArn": "arn:aws:opsworks-cm:us-west-2:123456789012:server/
opsworks-cm-test/EXAMPLEd-66b0-4196-8274-d1a2bEXAMPLE"
        }
    ]
}
```

2. 使用您在步骤 1 中返回的 ARN 运行 `tag-resource` 命令。

```
aws opsworks-cm tag-resource --resource-arn "server_or_backup_ARN" --tags
 Key=Key1,Value=Value1 Key=Key2,Value=Value2
```

示例如下：

```
aws opsworks-cm tag-resource --resource-arn "arn:aws:opsworks-cm:us-
west-2:123456789012:server/opsworks-cm-test/EXAMPLEd-66b0-4196-8274-d1a2bEXAMPLE"
 --tags Key=Stage,Value=Production Key=Department,Value=Marketing
```

3. 要验证标签是否已成功添加，请继续下一个过程，即列出资源标签 (CLI)。

列出资源标签 (CLI)

你可以运行该`list-tags-for-resource`命令来显示 Puppet Enterprise 服务器或备份所附的标签。OpsWorks 指定目标资源的 ARN 以查看其标签。

1. 要获取要列出其标签的资源的 ARN，请执行以下操作：

- 对于服务器，请运行 `describe-servers --server-name` *`server_name`*。命令的结果显示服务器 ARN。

- 对于备份，请运行 `describe-backups --backup-id backup_ID`。命令的结果显示备份 ARN。您还可以运行`describe-backups --server-name server_name`以显示有关特定 OpsWorks 的 Puppet Enterprise 服务器的所有备份的信息。

2. 使用您在步骤 1 中返回的 ARN 运行 `list-tags-for-resource` 命令。

```
aws opsworks-cm list-tags-for-resource --resource-arn "server_or_backup_ARN"
```

示例如下：

```
aws opsworks-cm tag-resource --resource-arn "arn:aws:opsworks-cm:us-
west-2:123456789012:server/opsworks-cm-test/EXAMPLEd-66b0-4196-8274-d1a2bEXAMPLE"
```

如果资源上有标签，则命令返回如下所示的结果。

```
{
    "Tags": [
        {
            "Key": "Stage",
            "Value": "Production"
        },
        {
            "Key": "Department",
            "Value": "Marketing"
        }
    ]
}
```

从资源中删除标签 (CLI)

你可以运行`untag-resource`命令删除 Puppet Enterprise 服务器或备份的标签。 OpsWorks 如果资源被删除，资源上的标签也会被删除。指定目标资源的 Amazon 资源编号 (ARN)，以便从其中删除标签。

1. 要获取要从中删除标签的资源的 ARN，请执行以下操作：

- 对于服务器，请运行 `describe-servers --server-name server_name`。命令的结果显示服务器 ARN。

- 对于备份，请运行 `describe-backups --backup-id backup_ID`。命令的结果显示备份 ARN。您还可以运行`describe-backups --server-name server_name`以显示有关特定 OpsWorks 的 Puppet Enterprise 服务器的所有备份的信息。

2. 使用您在步骤 1 中返回的 ARN 运行 `untag-resource` 命令。仅指定要删除的标签。

```
aws opsworks-cm untag-resource --resource-arn "server_or_backup_ARN" --tags
 Key=Key1,Value=Value1 Key=Key2,Value=Value2
```

在此示例中，`untag-resource` 命令仅删除密钥为 `Stage` 且值为 `Production` 的标签。

```
aws opsworks-cm untag-resource --resource-arn "arn:aws:opsworks-cm:us-
west-2:123456789012:server/opsworks-cm-test/EXAMPLEd-66b0-4196-8274-d1a2bEXAMPLE"
 --tags Key=Stage,Value=Production
```

3. 要验证标签是否已成功删除，请按照本主题的列出资源标签 (CLI)中的步骤操作。

另请参阅

- 使用创建 Puppet Enterprise Master AWS CLI
- 为 Puppet Enterprise 服务器 OpsWorks 备份
- AWS 标记策略
- 使用AWS Identity and Access Management 用户指南中的资源标签控制对资源的访问权限 AWS
- 《AWS Billing and Cost Management 用户指南》中的使用成本分配标签
- CreateBackup在 AWS OpsWorks CM API 参考中
- CreateServer在 AWS OpsWorks CM API 参考中
- TagResource在 AWS OpsWorks CM API 参考中
- ListTagsForResource在 AWS OpsWorks CM API 参考中
- UntagResource在 AWS OpsWorks CM API 参考中

备份和恢复 Puppe OpsWorks t Enterprise Server

> **Important**
>
> 该 AWS OpsWorks for Puppet Enterprise 服务于 2024 年 3 月 31 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

本节介绍如何备份和恢复 Puppe OpsWorks t Enterprise 服务器。

主题

为 Puppet Enterprise 服务器 OpsWorks 备份

> **Important**
>
> 该 AWS OpsWorks for Puppet Enterprise 服务于 2024 年 3 月 31 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

您可以为 Puppet Enterprise 服务器备份定义每日或每周的定期 OpsWorks 备份，并让该服务代表您将备份存储在亚马逊简单存储服务 (Amazon S3) 中。或者，您可以按需进行手动备份。

由于备份存储在 Amazon S3 中，因此它们会产生额外费用。您最多可以将备份的保留数量设定为 30 个。您可以使用 AWS 支持渠道提交服务请求以更改该限制。发送到 Amazon S3 存储桶的内容可能包含客户内容。有关删除敏感数据的更多信息，请参阅如何清空 S3 存储桶？或如何删除 S3 存储桶？。

您可以在 Puppet Enterprise 主 OpsWorks 服务器的备份中添加标签。如果您已向 Puppe OpsWorks t Enterprise 主节点添加了标签，Puppet 主节点的自动备份将继承这些标签。有关如何在备份中添加和管理标签的更多信息，请参阅本指南中的使用 AWS OpsWorks for Puppet Enterprise 资源上的标签。

主题

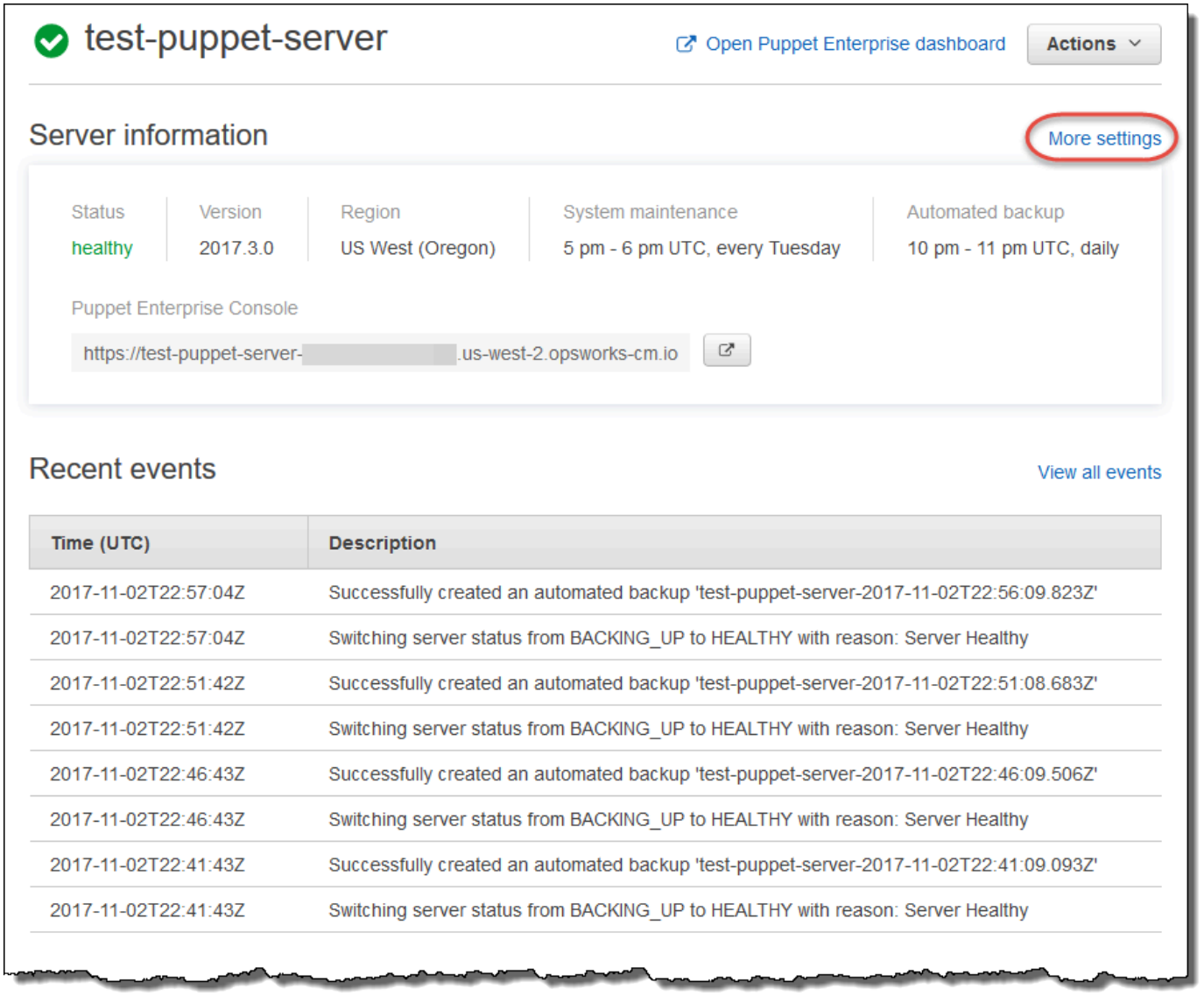
- 手动备份
- 删除备份

自动备份

在为 Puppet Enterprise 服务器配置时，可以选择自动备份或手动备份。 OpsWorks OpsWorks for Puppet Enterprise 将在设置向导的 “配置高级设置” 页面的 “自动备份” 部分中选择的一小时和当天开始自动备份。当服务器处于联机状态之后，您可以通过在服务器的属性页面上执行以下步骤来更改备份设置。

更改自动备份设置

1. 在服务器的属性页面上，选择 More settings。



2. 要关闭自动备份，Enable automated backups 选项请选择 No。保存您的更改；您不需要继续执行下一步骤。
3. 在 Automated Backup 部分中，更改频率、开始时间或要保留的生成。保存您的更改。

手动备份

您可以随时在中启动手动备份，也可以通过运行 AWS CLI [create- AWS Management Console backup 命令启动手动备份]。手动备份不包括在存储的最多 30 代自动备份中。最多存储 10 个手动备份，并且必须从 Amazon S3 中手动删除它们。

要在中执行手动备份 AWS Management Console

1. 在 Puppet Enterprise servers 页面上，选择您要备份的服务器。
2. 在服务器的属性页面上，在左侧导航窗格中选择 Backups。
3. 选择创建备份。
4. 当页面在备份的 Status 列中显示绿色复选标记时，手动备份完成。

要在中执行手动备份 AWS CLI

在为 Puppet Enterpr OpsWorks ise 服务器创建新的手动备份时，可以添加标签。有关如何在创建手动备份时添加标签的更多信息，请参阅向新备份添加标签 (CLI)。

- 要启动手动备份，请运行以下 AWS CLI 命令。

```
aws opsworks-cm --region region name create-backup --server-name "Puppet server
 name" --description "optional descriptive string"
```

删除备份

删除某个备份会从存储该备份的 S3 存储桶中永久删除它。

要在中删除备份 AWS Management Console

1. 在 Puppet Enterprise servers 页面上，选择您要备份的服务器。
2. 在服务器的属性页面上，在左侧导航窗格中选择 Backups。
3. 选择您要删除的备份，然后选择 Delete backup。您一次只能选择一个备份。
4. 出现确认删除提示时，选中 Delete the backup, which is stored in an S3 bucket 的复选框，然后选择 Yes, Delete。

要在中删除备份 AWS CLI

- 要删除备份，请运行以下 AWS CLI 命令，将的`--backup-id`值替换为要删除的备份的 ID。Backup ID 的格式为 *`ServerName-yyyymmddhhmmssss`* s。例如，**`puppet-server-20171218132604388`**。

```
aws opsworks-cm --region region name delete-backup --backup-id ServerName-
yyyyMMddHHmmssSSS
```

从 Backu OpsWorks p 中恢复 for Puppet 企业服务器

> **Important**
>
> 该 AWS OpsWorks for Puppet Enterprise 服务于 2024 年 3 月 31 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

浏览完可用备份后，您可以轻松地选择从哪个时间点恢复 for Puppet Enterprise 服务器。 OpsWorks 服务器备份包含配置管理软件持久性数据，例如模块、类、节点关联、数据库信息 (包括报告、事实等)。对服务器执行就地恢复（即将 Puppet Enterprise 服务器的现有服务器恢复到新的 EC2 实例）会重新注册在备份时注册的 OpsWorks 用于恢复服务器的节点，如果恢复成功且 Puppet Enterprise 服务器的恢复状态为，则将流量切换到新实例。 OpsWorks `Healthy`恢复到新创建的 OpsWorks Puppet Enterprise 服务器不会保持节点连接。还原一个服务器并不会更新 Puppet 软件的版本；它将应用与您所选的备份中相同的可用 Puppet 版本和配置管理数据。

还原服务器通常比创建新服务器花费更多的时间；时间取决于您选择的备份大小。还原完成后，旧 EC2 实例仍处于 `Running` 或 `Stopped` 状态，但只是暂时的。它最终被终止。

在此版本中，您可以使用恢复 Puppet E AWS CLI nterprise 中的 OpsWorks Puppet 大师。

> **Note**
>
> 您还可以运行 restore-server 命令来更改当前实例类型，或者还原或设置您的 SSH 密钥 (如果该密钥丢失或泄露)。

从备份中还原服务器

1. 在中 AWS CLI，运行以下命令以返回可用备份及其 ID 的列表。请记下要使用的备份的 ID。Backup ID 的格式为 *myServerName-yyyymmddhhmmssss* s。

   ```
   aws opsworks-cm --region region name describe-backups
   ```

2. 运行以下命令。

```
aws opsworks-cm --region region name restore-server --backup-id "myServerName-
yyyyMMddHHmmssSSS" --instance-type "Type of instance" --key-pair "name of your EC2
 key pair" --server-name "name of Puppet master"
```

示例如下：

```
aws opsworks-cm --region us-west-2 restore-server --backup-id
 "MyPuppetServer-20161120122143125" --server-name "MyPuppetServer"
```

3. 等待直到还原操作完成。

Puppet Enterp OpsWorks rise 系统维护中

> **Important**
>
> 该 AWS OpsWorks for Puppet Enterprise 服务于 2024 年 3 月 31 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

强制性系统维护可确保 Puppet AWS Server 的最新测试版本（包括安全更新）始终在 for Puppet Enterpr OpsWorks ise 服务器上运行。每周至少需要进行一次系统维护。如果需要 AWS CLI，您可以使用，配置每日自动维护。除了定期的 AWS CLI 系统维护外，您还可以使用按需执行系统维护。

如果有新的 Puppet 软件版本可用，一旦其通过了 AWS 测试，系统维护机制会自动在服务器上更新 Puppet 服务器的版本。AWS 进行了广泛的测试，以验证 Puppet 升级是否可以投入生产，并且不会中断现有客户环境，因此 Puppet 软件的发布与其应用于 Puppet Enterprise 服务器的现有 OpsWorks 版本之间可能会存在延迟。要按需更新 Puppet 软件的可用版本，请参阅本主题中的 按需启动系统维护。

系统维护从在维护过程中执行的备份启动新实例，这有助于减少经过定期维护的降级或受损 Amazon EC2 实例带来的风险。

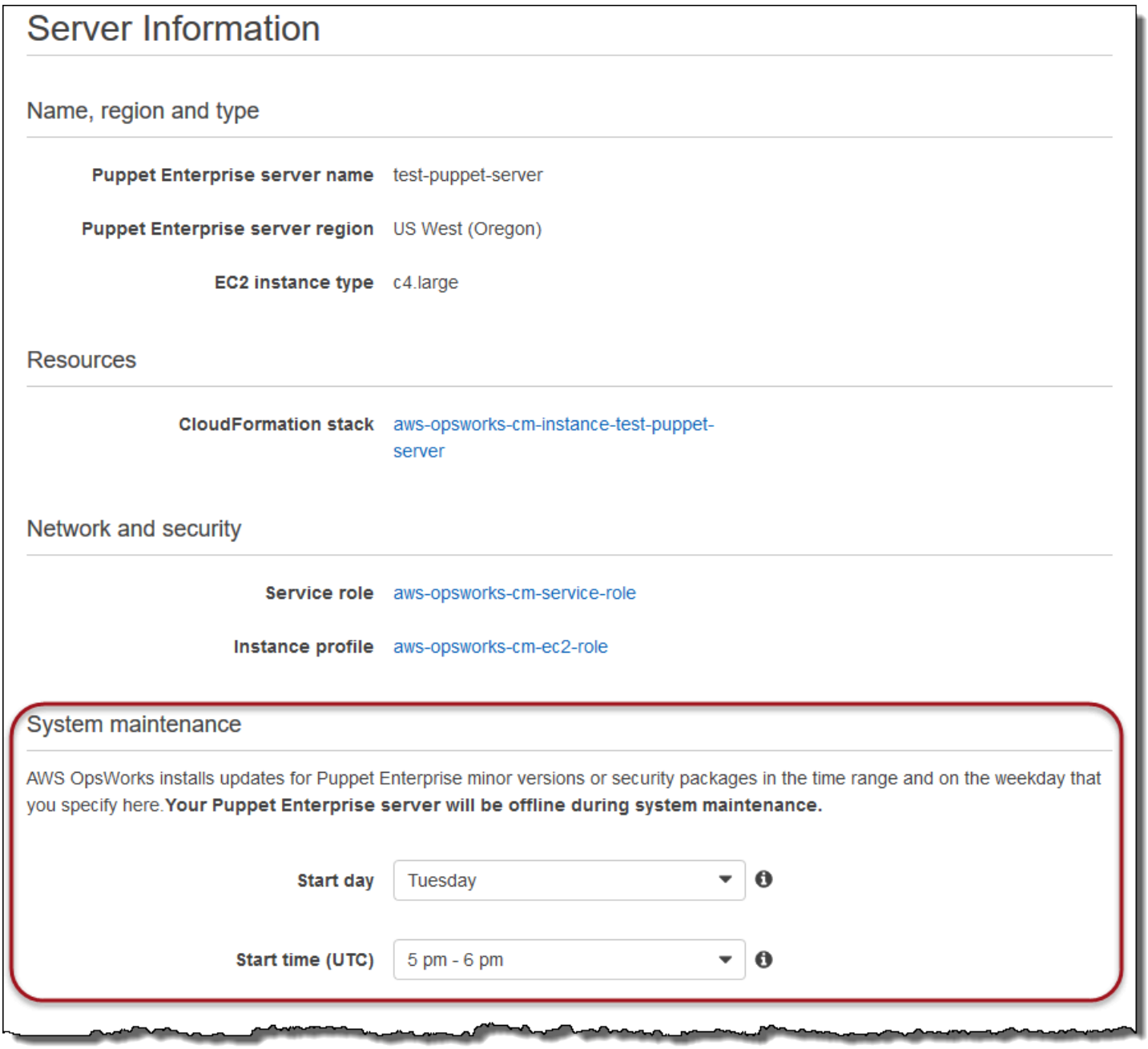
> **Important**
>
> 系统维护会删除您已添加到 Puppet Enterprise 服务器的所有文件或自定义配置。 OpsWorks 有关如何修复配置或文件丢失的更多信息，请参阅本主题中的在维护之后恢复自定义配置和文件。

主题

配置系统维护

在 OpsWorks 为 Puppet Enterprise 服务器创建新的服务器时，可以按协调世界时 (UTC) 配置工作日和时间，以便开始系统维护。维护在您指定的小时开始。由于您可以预见到服务器会在系统维护期间脱机，因此请选择正常工作时间中对服务器需求较低的时间。进行维护时，服务器的状态为 UNDER_MAINTENANCE。

您还可以更改现有 OpsWorks 的 Puppet Enterprise 服务器上的系统维护设置，方法是更改服务器 “设置” 页面的 “系统维护” 区域中的设置，如以下屏幕截图所示。



在 System maintenance (系统维护) 部分中，设置您希望系统维护开始的日期和时间。

使用配置系统维护 AWS CLI

您还可以使用 AWS CLI配置系统维护自动开始时间。如果需要， AWS CLI 可以省略三个字符的工作日前缀，从而配置每日自动维护。

在 `create-server` 命令中，指定创建服务器实例的要求 (例如实例类型、实例配置文件 ARN 和服务角色 ARN) 后，向您的命令添加 `--preferred-maintenance-window` 参数。在以下 `create-`

server 示例中，--preferred-maintenance-window 设置为 Mon:08:00，这意味着您已设置维护在每周一上午 8:00 开始。(UTC)。

```
aws opsworks-cm create-server --engine "Puppet" --engine-model "Monolithic"
 --engine-version "2017" --server-name "puppet-06" --instance-profile-arn
 "arn:aws:iam::1119001987000:instance-profile/aws-opsworks-cm-ec2-role"
 --instance-type "c4.large" --key-pair "amazon-test" --service-role-arn
 "arn:aws:iam::044726508045:role/aws-opsworks-cm-service-role" --preferred-maintenance-
window "Mon:08:00"
```

在 update-server 命令中，您可以根据需要单独更新 --preferred-maintenance-window 值。在以下示例中，维护时段设置为周五晚上 6:15 (UTC)。

```
aws opsworks-cm update-server --server-name "puppet-06" --preferred-maintenance-window
 "Fri:18:15"
```

将维护时段的开始时间更改为每天晚上 6:15 (UTC)，请忽略三个字符的工作日前缀，如以下示例中所示。

```
aws opsworks-cm update-server --server-name "puppet-06" --preferred-maintenance-window
 "18:15"
```

有关使用设置首选系统维护时段的更多信息 AWS CLI，请参阅创建服务器和更新服务器。

按需启动系统维护

要在配置的每周或每日自动维护之外按需启动系统维护，请运行以下 AWS CLI 命令。您不能在 AWS Management Console中启动按需维护。

```
aws opsworks-cm start-maintenance --server-name server_name
```

有关此命令的更多信息，请参阅 start-maintenance。

在维护之后恢复自定义配置和文件

系统维护人员可以删除或更改您已添加到 for Puppet Enterprise 服务器 OpsWorks 的自定义文件或配置。

如果在维护运行之后，您的 Puppet Master 缺少了使用 RunCommand 或 SSH 添加的文件或设置，您可以使用亚马逊机器映像（AMI）来启动新的 Amazon EC2 实例。提供的 AMI 是从服务器的维护前配置生成的。

新的实例与维护前的 Puppet Master 处于相同状态，且应包含您缺少的文件和设置。

> **Important**
>
> 您无法使用新实例来恢复服务器；该实例无法作为 Puppet Master 运行。您只能使用实例来恢复您的文件和配置设置。

要从 AMI 启动 EC2 实例，请在 Amazon EC2 控制台中打开 Launch 向导，选择 My AMIs，然后选择具有您的服务器名称的 AMI。按照 Amazon EC2 向导中的步骤执行，就像您启动任何其他实例一样操作。

OpsWorks 为 Puppet Enterprise 自动添加节点

> **Important**
>
> 该 AWS OpsWorks for Puppet Enterprise 服务于 2024 年 3 月 31 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

本主题介绍如何将亚马逊弹性计算云 (Amazon EC2) 节点自动添加到 for Puppet Enterp OpsWorks rise 服务器中。在为 Puppet Master 添加要管理的节点中，您学会了如何使用 `associate-node` 命令，一次向您的 Puppet Enterprise 服务器中添加一个节点。本主题中的代码显示如何使用无人参与的方法自动添加节点。对于如何在无人参与的情况下 (或自动) 将新的节点关联起来，我们推荐的方法是配置 Amazon EC2 用户数据。默认情况下， OpsWorks 适用于 Puppet Enterprise 的服务器已经`puppet-agent`可用于 Ubuntu、Amazon Linux 和 RHEL 节点操作系统。

有关如何取消关联节点的信息，请参阅取消节点与 for Puppet OpsWorks 企业服务器的关联本指南和`disassociate-node`for Puppet Enterprise API 文档。 OpsWorks

第 1 步：创建一个 IAM 角色，以用作您的实例配置文件

创建一个 AWS Identity and Access Management (IAM) 角色用作您的 EC2 实例配置文件，并将以下策略附加到该 IAM 角色。该策略准许 `opsworks-cm` API 在节点注册期间与 EC2 实例通信。有关实例

配置文件的更多信息，请参阅 Amazon EC2 文档中的使用实例配置文件。有关如何创建 IAM 角色的信息，请参阅 Amazon EC2 文档中的在控制台中创建 IAM 角色。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "opsworks-cm:AssociateNode",
                "opsworks-cm:DescribeNodeAssociationStatus",
                "opsworks-cm:DescribeServers",
                "ec2:DescribeTags"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

AWS OpsWorks 提供了一个 AWS CloudFormation 模板，您可以使用该模板通过上述策略声明创建 IAM 角色。以下 AWS CLI 命令使用此模板为您创建实例配置文件角色。如果您想在默认区域中创建新 AWS CloudFormation 堆栈，则可以省略该`--region`参数。

```
aws cloudformation --region region ID create-stack --stack-name myPuppetinstanceprofile
 --template-url https://s3.amazonaws.com/opsworks-cm-us-east-1-prod-default-assets/
misc/owpe/opsworks-cm-nodes-roles.yaml --capabilities CAPABILITY_IAM
```

第 2 步：使用自动化关联脚本创建实例

要创建 EC2 实例，您可以将入门套件中包含的用户数据脚本复制到 EC2 实例说明`userdata`部分、Amazon EC2 Auto Scaling 组启动配置或 AWS CloudFormation 模板中。仅对于运行 Ubuntu 和 Amazon Linux 操作系统的 EC2 实例支持该脚本。有关将脚本添加将到用户数据的更多信息，请参阅 Amazon EC2 文档中的启动时对您的 Linux 实例运行命令。创建新节点最简单的方法是使用 Amazon EC2 实例启动向导。此演练使用 Puppet Enterp OpsWorks rise 入门 中所述的 Apache web 服务器示例模块设置。

1. 初学者工具包中的用户数据脚本运行 `opsworks-cm` API `associate-node` 命令，将一个新的节点与您的 Puppet Master 相关联。在此版本中，它还会为您在节点 AWS CLI 上安装当前版本的，以防它尚未运行大多数 up-to-date 版本。将此脚本以名称 `userdata.sh` 保存到方便的位置。

默认情况下，新注册节点的名称是实例 ID。

2. 按照 EC2 文档中启动实例的说明操作，修改见此处。在 EC2 实例启动向导中，选择 Amazon Linux AMI。
3. 在 Configure Instance Details 页面上，选择 myPuppetinstanceprofile，将您在 第 1 步：创建一个 IAM 角色，以用作您的实例配置文件中创建的角色作为 IAM 角色。
4. 在 Advanced Details 区域中，上传您在步骤 1 中创建的 `userdata.sh` 脚本。
5. 无需在 Add Storage 页面上进行更改。转到 Add Tags。

   通过将标签应用到 EC2 实例，您可以自定义 `userdata.sh` 的行为。对于此示例，通过添加值为 **`apache_webserver`** 的标签 **`pp_role`**，将角色 `apache_webserver` 应用到您的节点。

   在节点上设置 `pp_role` 值将设置永久存储在节点的代理证书中的数据值，为节点启用可信分类。有关更多信息，请参阅 Puppet 平台文档中的扩展请求 (永久证书数据)。
6. 在配置安全组页上，选择添加规则，然后选择类型 HTTP，在此例中为 Apache Web 服务器打开端口 8080。
7. 选择 Review and Launch，然后选择 Launch。新节点启动时，它会应用您在 设置初学者工具包 Apache 示例 中设置的示例模块的 Apache 配置。
8. 当您打开链接到新节点的公有 DNS 的网页时，您应看到网站托管在由 Puppet 管理的 Apache Web 服务器上。

取消节点与 for Puppet OpsWorks 企业服务器的关联

> **Important**
>
> 该 AWS OpsWorks for Puppet Enterprise 服务于 2024 年 3 月 31 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

本节介绍如何从 for Puppet Enterprise 服务器中取消关联或移除托管节点 OpsWorks 的管理。此操作在命令行或 Puppet Enterprise 控制台中执行；您无法在 for Puppet Enterprise 管理控制台中取消关联节点。 OpsWorks 目前，f OpsWorks or Puppet Enterprise API 不允许批量删除多个节点。本节中的命令将一次对一个节点取消关联。

我们建议在删除 Puppet Master 之前解除节点与该服务器的关联，这样节点就能继续工作而不用尝试重新连接服务器。为此，请运行`disassociate-node` AWS CLI 命令。要从 PE 中完全删除节点，您必须取消节点的关联并撤销其证书，这样节点不会持续尝试签入到 Puppet Master。当您不再希望使用 Puppet Master 管理代理时，您还应从节点卸载 `puppet-agent`。

解除节点的关联

1. 在中 AWS CLI，运行以下命令取消节点的关联。*`Node_name`* 是要取消关联的节点的名称；对于 Amazon EC2 实例，这是实例 ID。*`Server_name`* 是您要将节点与之解除关联的 Puppet 主服务器的名称。两个参数都是必需的。`--region` 参数不是必需的，除非您要取消节点与不在您的默认区域内的 Puppet Master 的关联。

   ```
   aws opsworks-cm --region Region_name disassociate-node --node-name Node_name --
   server-name Server_name
   ```

   以下命令是一个示例。

   ```
   aws opsworks-cm --region us-west-2 disassociate-node --node-name
    i-0010zzz00d66zzz90 --server-name opsworkstest
   ```

2. 请耐心等待，直到响应消息指示已完成关联断开。

有关如何删除 Puppet Enterprise 服务器 OpsWorks 的更多信息，请参阅删除 Pupp OpsWorks et 企业服务器的。

另请参阅

- Puppet Enterprise 文档中的删除节点

删除 Pupp OpsWorks et 企业服务器的

> **Important**
>
> 该 AWS OpsWorks for Puppet Enterprise 服务于 2024 年 3 月 31 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

本节介绍如何删除 Puppe OpsWorks t Enterprise 服务器。删除服务器的同时会删除存储在服务器上的事件、日志和任何模块。支持资源（Amazon Elastic Compute Cloud 实例、Amazon Elastic Block Store 卷等）以及所有自动备份也将被删除。

尽管删除某个服务器并不会删除节点，但这些节点不再由删除的服务器进行管理，因此将不断尝试重新连接。因此，我们建议您在删除 Puppet Master 之前，解除托管节点的关联。在此版本中，您可以通过运行 AWS CLI 命令来取消与节点的关联。

步骤 1：解除托管节点的关联

请在删除服务器之前解除节点与 Puppet Master 的关联，以便节点继续工作，而不会尝试重新连接服务器。为此，请运行`disassociate-node` AWS CLI 命令。

解除节点的关联

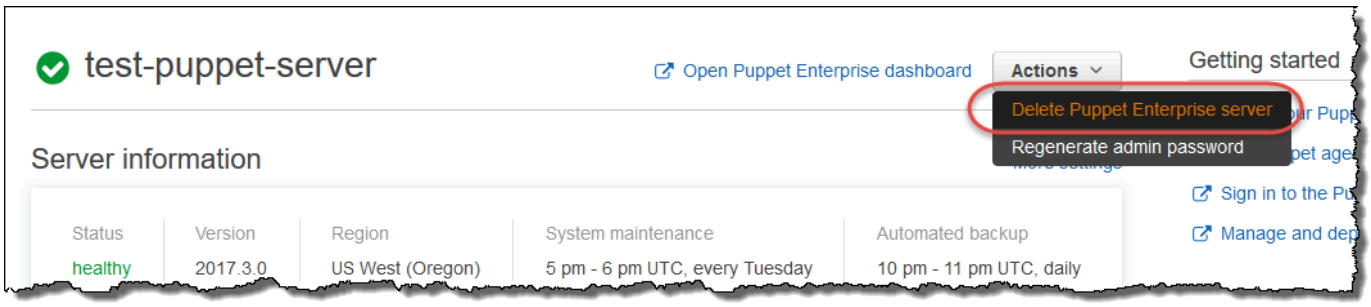
1. 在中 AWS CLI，运行以下命令取消节点的关联。*`Server_name`* 是您要将节点与之解除关联的 Puppet Master。`--node-name` 的值可以为实例 ID。

   ```
   aws opsworks-cm --region Region_name disassociate-node --node-name Node_name --
   server-name Server_name
   ```

2. 请耐心等待，直到响应消息指示已完成关联断开。

步骤 2：删除服务器

1. 在仪表板的服务器磁贴上，展开 Actions 菜单。



2. 选择 Delete Puppet Enterprise server。
3. 出现确认删除提示时，选中复选框，删除关联的角色和资源，然后选择 Yes, Delete。

另请参阅

- 取消节点与 for Puppet OpsWorks 企业服务器的关联

如何将 Puppe OpsWorks t Enterprise 服务器迁移到亚马逊弹性计算云 (Amazon EC2)

> **Important**
>
> 该 AWS OpsWorks for Puppet Enterprise 服务于 2024 年 3 月 31 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

以下说明描述了如何将现有 Puppet Enterprise 服务器迁移到 Amazon EC2，以备您想继续使用 Puppet Enterprise 来满足外部的配置管理需求。 OpsWorks

主题

第 1 步：联系 Puppet 购买许可证

当您将服务器迁移到 EC2 时，新实例不附带 Puppet 许可证。要购买许可证密钥，请按照 Puppet 网站上的说明进行操作。

第 2 步：获取有关 Puppet Enterprise 服务器 OpsWorks 的详细信息

查找并保存您的 for Puppet Enterprise 服务器 OpsWorks 的值。

1. 登录 AWS Management Console 并打开 Amazon S3 控制台，网址为 https://console.aws.amazon.com/s3/。

   复制你的 for Puppet Enterprise 服务器的现有 Amazon S3 存储桶 OpsWorks 的名称。存储桶名称的格式为 `aws-opsworks-cm-server-name-random-string`。

2. 运行aws opsworks-cm describe-servers命令以获取 Puppet Enterprise 服务器 OpsWorks 的配置。

   ```
   aws opsworks-cm describe-servers \
      --server-name server-name \
      --region region
   ```

   存储响应中的 `InstanceType`、`KeyPair`、`SubnetIds`、`SecurityGroupIds`、`InstanceProfileArn`、和 `Endpoint` 的值。

3. 使用 SSH 连接到现有 OpsWorks 的 Puppet Enterprise 服务器。您可以使用 EC2 控制台中的 Session Manager 而不是 SSH。

   运行以下命令。

   ```
   rpm -qa | grep opsworks-cm-puppet-enterprise | cut -d '-' -f 5
   ```

   响应提供了 Puppet Enterprise 版本（例如，2019.8.10）。存储此值。

在下一个步骤中，您将使用 SSH 或 Session manager。

第 3 步：备份你 OpsWorks 的 for Puppet Enterprise 服务器

1. 运行以下命令进行本地备份。

   ```
   mkdir /tmp/puppet-backup/
   sudo /opt/puppetlabs/bin/puppet-backup create --dir=/tmp/puppet-backup/
   ```

2. 运行以下命令以存储备份的名称。

```
ls /tmp/puppet-backup/
PUPPET_BACKUP=$(ls /tmp/puppet-backup/)
```

3. 运行以下命令以将备份上传到 S3 存储桶。将 *S3-Bucket* 替换为 第 2 步：获取有关 Puppet Enterprise 服务器 OpsWorks 的详细信息 中步骤 1 中的值。

```
aws s3 cp /tmp/puppet-backup/PUPPET_BACKUP s3://S3_Bucket/tmp/puppet-backup/
```

存储 PUPPET_BACKUP 和 S3_BUCKET 值。您将把这些值导入到新的 EC2 实例。

您可以退出 SSH 或 Session Manager 会话。

第 4 步：启动一个 EC2 实例

使用与 Pupp@@ et Enterprise 服务器相同的配置，从 EC2 控制台 https://console.aws.amazon.com/ec2/ 启动一个新 OpsWorks 的 EC2 实例。

参数名称	值
OS	Amazon Linux 2
实例类型	第 2 步：获取有关 Puppet Enterprise 服务器 OpsWorks 的详细信息 的步骤 2 中的 InstanceType 值。
密钥对名称	第 2 步：获取有关 Puppet Enterprise 服务器 OpsWorks 的详细信息 的步骤 2 中的 KeyPair 值。
VPC	第 2 步：获取有关 Puppet Enterprise 服务器 OpsWorks 的详细信息 的步骤 2 中SubnetIds 的 VPC。
子网	第 2 步：获取有关 Puppet Enterprise 服务器 OpsWorks 的详细信息 的步骤 2 中的 SubnetIds 。
选择现有安全组 -> 常见安全组	第 2 步：获取有关 Puppet Enterprise 服务器 OpsWorks 的详细信息 的步骤 2 中的 SecurityGroupIds 。
存储	至少 120 GB。

参数名称	值
IAM 实例配置文件	第 2 步：获取有关 Puppet Enterprise 服务器 OpsWorks 的详细信息 的步骤 2 中的 `InstanceProfileArn` 。

如果您想创建 Elastic IP 并将其附加到新实例，请复制新实例的实例 ID，然后完成 （可选）步骤 4.1：创建并附加 Elastic IP 中的步骤。

（可选）步骤 4.1：创建并附加 Elastic IP

使用弹性 IP 地址，您可以快速将地址重新映射到您的账户中的另一个实例，从而屏蔽实例故障。

创建并关联弹性 IP 地址

1. 登录 AWS Management Console 并打开亚马逊 EC2 控制台，网址为 https://console.aws.amazon.com/ec2/。
2. 选择Elastic IP。
3. 选择 Allocate Elastic IP address (分配弹性 IP 地址)。
4. 在分配弹性 IP 地址页面，选择分配。这将创建公有 IPv4 地址。
5. 复制分配的 IPv4 地址。
6. 从操作中选择关联弹性 IP 地址。
7. 对于实例，输入新实例的实例 ID。
8. 选择关联。

第 5 步：在新的 EC2 实例上安装 Puppet Enterprise

使用 SSH 连接到新的 EC2 实例。您可以使用 EC2 控制台中的 Session Manager 而不是 SSH。

```
# switch to sudo user
sudo -i

# Setup environment variables
PUPPET_ENTERPRISE_VERSION=Puppet Enterprise version from step 2.3
hostname Public IPv4 DNS or Custom Domain if available

# Install Puppet Enterprise
```

```
curl -JLO https://pm.puppetlabs.com/puppet-enterprise/$PUPPET_ENTERPRISE_VERSION/
puppet-enterprise-$PUPPET_ENTERPRISE_VERSION-el-7-x86_64.tar.gz
tar -xf puppet-enterprise-$PUPPET_ENTERPRISE_VERSION-el-7-x86_64.tar.gz

./puppet-enterprise-$PUPPET_ENTERPRISE_VERSION-el-7-x86_64/puppet-enterprise-installer
```

您可以保持 SSH 或 Session Manager 会话处于打开状态，以备下一步使用。

步骤 6：在新的 EC2 实例上恢复备份

```
# Setup environment variables
S3_BUCKET=S3 bucket name from step 2.1
PUPPET_BACKUP=Puppet backup file name from step 3.2

# download backup
aws s3 cp s3://$S3_BUCKET/tmp/puppet-backup/$PUPPET_BACKUP

# Prepare Puppet Enterprise backup to remove OpsWorks metadata
mkdir output
tar -xf $PUPPET_BACKUP -C output/
cd output/
rm -f opt/puppetlabs/facter/facts.d/opsworks.json
tar -cf ../$PUPPET_BACKUP *
cd ..
rm -rf output/

# Restore from backup
PATH=$PATH:/opt/puppetlabs/puppet/bin/
puppet-backup restore $PUPPET_BACKUP
puppet agent -t
```

您可以访问已恢复的 EC2 实例的 Puppet 控制台，网址为 https://###### *IPv4*。您可以在 EC2 控制台中的实例详细信息页面上找到公有 IPv4 DNS。登录凭据与您用于访问 Puppet Enterprise 服务器 OpsWorks 的凭据相同。

您可以保持 SSH 或 Session Manager 会话处于打开状态，以备下一步使用。

步骤 7：配置您的 Puppet 许可证

按照 Puppet 网站上的步骤配置您的许可证。

您可以保持 SSH 或 Session Manager 会话处于打开状态，以备下一步使用。

步骤 8：迁移节点

Puppet Enterprise 服务器支持两种类型的域： OpsWorks

- BYODC（带上您自己的域名和证书）
- OpsWorks 端点

步骤 8.1：对于 BYODC（带上您自己的域名和证书）

对于这些节点，您只需将 DNS 提供商中的自定义域指向新 EC2 实例的公有 IPv4 DNS 或公有 IPv4 地址即可。

步骤 8.2：对于 OpsWorks 终端节点

对于 OpsWorks 端点，Puppet 文档建议在节点上卸载 Puppet 代理，然后使用新恢复的 Puppet Enterprise 服务器安装 Puppet 代理。

> **Note**
>
> 虽然 Puppet 没有移动代理节点的自动化程序，但 Puppet 社区成员已在 Puppet Forge 网站上发布了一些模块来完成自动节点迁移。这些模块包括`pe_migrate`模块和另一位作者的第二个迁移模块。Puppet 不支持 Puppet Forge 网站上的模块，或者 OpsWorks除非在 Forge 模块中明确说明。我们建议谨慎使用这些模块，并在广泛使用之前对其进行测试。

以下各节提供了在 Linux 实例上卸载和重新安装 Puppet 代理的步骤。

主题

- 步骤 8.2.1：从 Puppet 服务器复制卸载程序
- 步骤 8.2.2：下载卸载程序并在节点上运行
- 步骤 8.2.3：在节点上重新安装 Puppet 代理

步骤 8.2.1：从 Puppet 服务器复制卸载程序

在卸载代理之前，请确保节点的 IAM 实例配置文件提供 S3 ReadOnly 权限。

运行以下命令以将卸载程序从 Puppet 服务器复制到 S3 存储桶。

```
aws s3 cp \
```

```
/opt/puppetlabs/bin/puppet-enterprise-uninstaller \
s3://$S3_BUCKET/tmp/puppet-enterprise-uninstaller
```

运行命令后，您可以注销 Puppet 服务器的 SSH 或 Session Manager 会话。

步骤 8.2.2：下载卸载程序并在节点上运行

使用 SSH 连接节点。如果节点是一个 EC2 实例，您可以使用 EC2 控制台中的 Session Manager 而不是 SSH。

```
sudo -i

S3_BUCKET=aws-opsworks-cm-abcdefg-uuhtyn6messn
aws s3 cp s3://$S3_BUCKET/tmp/puppet-enterprise-uninstaller /opt/puppetlabs/bin/
chmod 700 /opt/puppetlabs/bin/puppet-enterprise-uninstaller
/opt/puppetlabs/bin/puppet-enterprise-uninstaller
```

您可以保持 SSH 或 Session Manager 会话处于打开状态，以备下一步使用。

步骤 8.2.3：在节点上重新安装 Puppet 代理

完成以下步骤以在节点上重新安装 Puppet Agent。

主题

- 步骤 8.2.3.1：使用正确的配置安装 Puppet 代理
- 步骤 8.2.3.2：在 Puppet 控制台中接受证书
- 步骤 8.2.3.3：将节点签入 Puppet Enterprise 服务器

步骤 8.2.3.1：使用正确的配置安装 Puppet 代理

运行以下命令安装 Puppet 代理。

```
curl -k https://Public_IPv4_DNS:8140/packages/current/install.bash | bash
```

您可以保持 SSH 或 Session Manager 会话处于打开状态，以备步骤 8.2.2.3 使用。

步骤 8.2.3.2：在 Puppet 控制台中接受证书

1. 在 `https://Public_IPv4_DNS` 前往 Puppet 服务器的控制台。
2. 选择证书，然后选择未签名证书。

3. 选择接受以签署 Puppet 代理的证书。

步骤 8.2.3.3：将节点签入 Puppet Enterprise 服务器

在节点上运行以下命令以将其签入服务器。

```
puppet agent -t
```

现在，该节点应该在 Puppet 服务器的控制台中可见。

第 9 步：删除你的 OpsWorks Puppet Enterprise 服务器

您可以使用 OpsWorks 控制台或 AWS CLI 删除您 OpsWorks 的 for Puppet Enterprise 服务器。

使用 OpsWorks 控制台删除服务器

1. 登录 AWS Management Console 并打开 AWS OpsWorks 控制台，网址为 https://console.aws.amazon.com/opsworks/。
2. 从导航窗格中，选择 Puppet Enterprise 服务器。
3. 在 Puppet Enterprise 服务器页面上，选择您要删除的服务器。
4. 从操作中，选择删除 Puppet Enterprise 服务器。

要删除您的服务器，请使用 AWS CLI

运行以下命令。

```
aws opsworks-cm delete-server \
   --server-name server-name \
   --region region
```

使用登录 OpsWorks Puppet 企业 API 调用 AWS CloudTrail

> **Important**
>
> 该 AWS OpsWorks for Puppet Enterprise 服务于 2024 年 3 月 31 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

OpsWorks for Puppet Enterprise 与 AWS CloudTrail一项服务集成，该服务提供用户、角色或 AWS 服务在 Puppet Enterprise 中采取的操作 OpsWorks 的记录。 CloudTrail 将 Puppet Enterprise 的所有 API 调用捕获 OpsWorks 为事件，包括来自 Puppet Enterprise 控制台的调用和对 Puppet Enterprise API OpsWorks 的代码调用。 OpsWorks 如果您创建跟踪，则可以允许将 CloudTrail 事件持续传输到 Amazon S3 存储桶，包括 Puppet Enterprise 的事件。 OpsWorks 如果您未配置跟踪，您仍然可以在 CloudTrail 控制台的 “事件历史记录” 中查看最新的事件。使用收集的信息 CloudTrail，您可以确定向 Puppet Enterprise 发出的请求、发出请求的 IP 地址、谁发出了请求、何时发出请求以及其他详细信息。 OpsWorks

要了解更多信息 CloudTrail，请参阅《AWS CloudTrail 用户指南》。

OpsWorks 查看 Puppet 企业信息 CloudTrail

CloudTrail 在您创建 AWS 账户时已在您的账户上启用。当 Puppet Enterprise 中 OpsWorks 发生活动时，该活动将与其他 AWS 服务 CloudTrail 事件一起记录在事件历史记录中。您可以在自己的 AWS 账户中查看、搜索和下载最近发生的事件。有关更多信息，请参阅使用事件历史记录查看 CloudTrail 事件。

要持续记录 AWS 账户中的事件，包括 Puppet Enterprise 的事件，请创建跟踪。 OpsWorks 跟踪允许 CloudTrail 将日志文件传输到 Amazon S3 存储桶。默认情况下，在控制台中创建跟踪记录时，此跟踪记录应用于所有区域。跟踪记录 AWS 分区中所有区域的事件，并将日志文件传送到您指定的 Amazon S3 存储桶。此外，您可以配置其他 AWS 服务，以进一步分析和处理 CloudTrail 日志中收集的事件数据。有关更多信息，请参阅：

- 创建跟踪概述
- CloudTrail 支持的服务和集成
- 配置 Amazon SNS 通知 CloudTrail
- 接收来自多个区域的 CloudTrail 日志文件和接收来自多个账户的 CloudTrail 日志文件

Puppet Enterprise 的所有 OpsWorks 操作都由 Pu ppet Enterprise API 参考记录 CloudTrail 并记录在案。OpsWorks 例如，对CreateServerCreateBackup、和DescribeServers操作的调用会在 CloudTrail 日志文件中生成条目。

每个事件或日记账条目都包含有关生成请求的人员信息。身份信息可帮助您确定以下内容：

- 请求是使用根用户凭证还是 IAM 用户凭证发出的。
- 请求是使用角色还是联合用户的临时安全凭证发出的。
- 请求是否由其他 AWS 服务发出。

有关更多信息，请参阅CloudTrail 用户身份元素。

了解 P OpsWorks uppet 企业日志文件条目

跟踪是一种配置，允许将事件作为日志文件传输到您指定的 Amazon S3 存储桶。 CloudTrail 日志文件包含一个或多个日志条目。事件代表来自任何来源的单个请求，包括有关请求的操作、操作的日期和时间、请求参数等的信息。 CloudTrail 日志文件不是公共 API 调用的有序堆栈跟踪，因此它们不会按任何特定的顺序出现。

以下示例显示了 Puppet Enterprise `CreateServer` 操作 OpsWorks 的 CloudTrail 日志条目。

```
{"eventVersion":"1.05",
"userIdentity":{
    "type":"AssumedRole",
    "principalId":"ID number:OpsWorksCMUser",
    "arn":"arn:aws:sts::831000000000:assumed-role/Admin/OpsWorksCMUser",
    "accountId":"831000000000","accessKeyId":"ID number",
    "sessionContext":{
        "attributes":{
            "mfaAuthenticated":"false",
            "creationDate":"2017-01-05T22:03:47Z"
            },
        "sessionIssuer":{
            "type":"Role",
            "principalId":"ID number",
            "arn":"arn:aws:iam::831000000000:role/Admin",
            "accountId":"831000000000",
            "userName":"Admin"
            }
        }
    },
"eventTime":"2017-01-05T22:18:23Z",
"eventSource":"opsworks-cm.amazonaws.com",
"eventName":"CreateServer",
"awsRegion":"us-west-2",
"sourceIPAddress":"101.25.190.51",
"userAgent":"console.amazonaws.com",
"requestParameters":{
    "serverName":"test-puppet-server",
    "engineModel":"Single",
    "engine":"Puppet",
    "instanceProfileArn":"arn:aws:iam::831000000000:instance-profile/aws-opsworks-cm-
ec2-role",
```

```
    "backupRetentionCount":3,"serviceRoleArn":"arn:aws:iam::831000000000:role/service-
role/aws-opsworks-cm-service-role",
    "engineVersion":"12",
    "preferredMaintenanceWindow":"Fri:21:00",
    "instanceType":"t2.medium",
    "subnetIds":["subnet-1e111f11"],
    "preferredBackupWindow":"Wed:08:00"
    },
"responseElements":{
    "server":{
        "endpoint":"test-puppet-server-xxxx8u4390xo6pd9.us-west-2.opsworks-cm.io",
        "createdAt":"Jan 5, 2017 10:18:22 PM",
        "serviceRoleArn":"arn:aws:iam::831000000000:role/service-role/aws-opsworks-cm-
service-role",
        "preferredBackupWindow":"Wed:08:00",
        "status":"CREATING",
        "subnetIds":["subnet-1e111f11"],
        "engine":"Puppet",
        "instanceType":"t2.medium",
        "serverName":"test-puppet-server",
        "serverArn":"arn:aws:opsworks-cm:us-west-2:831000000000:server/test-puppet-
server/8ezz7f6z-e91f-4z10-89z5-8c6219zzz09f",
        "engineModel":"Single",
        "backupRetentionCount":3,
        "engineAttributes":[
            {"name":"PUPPET_ADMIN_PASSWORD","value":"*** Redacted ***"},
            {"name":"PUPPET_API_CA_CERT","value":"*** Redacted ***"},
            ],
        "engineVersion":"12.11.1",
        "instanceProfileArn":"arn:aws:iam::831000000000:instance-profile/aws-opsworks-
cm-ec2-role",
        "preferredMaintenanceWindow":"Fri:21:00"
        }
    },
"requestID":"de7z64z9-d394-12ug-8081-7zz0386fbcb6",
"eventID":"8z7z18dz-6z90-47bz-87cf-e8346428zzz3",
"eventType":"AwsApiCall",
"recipientAccountId":"831000000000"
}
```

Pupp OpsWorks et 企业版故障排除

> **Important**
>
> 该 AWS OpsWorks for Puppet Enterprise 服务于 2024 年 3 月 31 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

本主题包含 Puppet Enterprise 问题的一些常见 OpsWorks 问题以及这些问题的建议解决方案。

主题

一般故障排除技巧

如果您无法创建或使用 Puppet Master，可查看错误消息或日志来帮助您对问题进行故障排除。下列任务描述了在您排除 Puppet Master 问题故障时通常应从哪些方面入手。有关特定错误和解决方案的信息，请参阅本主题的针对特定错误进行故障排除部分。

- 使用 f OpsWorks or Puppet Enterprise 控制台在 Puppet Master 无法启动时查看错误消息。在 Puppet Master 属性页面上，与服务器启动和运行相关的错误消息将显示在页面顶部。 OpsWorks 用于创建 Puppet Master 的 Puppet Enterprise 或 Amazon EC2 服务可能会出现错误。 AWS CloudFormation在属性页面上，您还可查看正在运行的服务器上发生的事件，其中可能会包含故障事件消息。
- 要帮助解决 EC2 问题，请通过使用 SSH 连接到您的服务器实例并查看日志。EC2 实例日志存储在 `/var/log/aws/opsworks-cm` 目录中。这些日志会捕获 Puppet Enterprise 启动 Puppet Master 时的 OpsWorks 命令输出。

针对特定错误进行故障排除

主题

- 服务器创建失败，并返回“requested configuration is currently not supported”消息
- 无法创建服务器的 Amazon EC2 实例
- 服务角色错误阻止服务器创建
- 超出弹性 IP 地址限制
- 无人参与节点关联失败
- 系统维护失败

服务器处于连接丢失状态

问题：服务器的状态显示为连接丢失。

原因：这种情况最常发生在外部实体对适用于 Pupp AWS OpsWorks et Enterprise OpsWorks 的服务器或其支持资源进行更改时。 AWS OpsWorks 无法连接到处于连接中断状态的 Puppet Enterprise 服务器来处理维护任务，例如创建备份、应用操作系统补丁或更新 Puppet。因此，您的服务器可能缺少重要更新，容易受到安全问题的影响，或者无法按预期运行。

解决方案：尝试以下步骤来恢复服务器的连接。

1. 请确保您的服务角色具有所有必需的权限。
    a. 在服务器的设置页面上，在网络和安全中，选择服务器正在使用的服务角色的链接。这将将服务角色打开以供在 IAM 控制台中查看。
    b. 在权限选项卡上，确认 `AWSOpsWorksCMServiceRole` 是否在权限策略列表中。如果未列出该托管策略，请手动将 `AWSOpsWorksCMServiceRole` 托管策略添加到角色中。
    c. 在信任关系选项卡上，验证服务角色是否具有信任 `opsworks-cm.amazonaws.com` 服务代表您代入角色的信任策略。有关如何对角色使用信任策略的更多信息，请参阅修改角色（控制台）或 AWS 安全博客文章《如何在 IAM 角色中使用信任策略》。
2. 请确保您的实例配置文件具有所有必需的权限。
    a. 在服务器的设置页面上，在网络和安全中，选择服务器正在使用的实例配置文件的链接。这将打开实例配置文件以在 IAM 控制台中查看。
    b. 在权限选项卡上，确认 `AmazonEC2RoleforSSM` 和 `AWSOpsWorksCMInstanceProfileRole` 是否在权限策略列表中。如果未列出其中一个或两个托管策略，请手动将这些托管策略添加到角色中。

c. 在信任关系选项卡上，验证服务角色是否具有信任 `ec2.amazonaws.com` 服务代表您代入角色的信任策略。有关如何对角色使用信任策略的更多信息，请参阅修改角色（控制台）或 AWS 安全博客文章《如何在 IAM 角色中使用信任策略》。

3. 在 Amazon EC2 控制台中，确保您与 OpsWorks 适用于 Puppet Enterprise 服务器的区域位于同一区域，然后重启您的服务器正在使用的 EC2 实例。

   a. 选择名为 `aws-opsworks-cm-instance- #####`的 EC2 实例。

   b. 在实例状态菜单，选择启动实例。

   c. 等待最多 15 分钟让您的服务器重新启动并完全联机。

4. 在 f OpsWorks or Puppet Enterprise 控制台的服务器详细信息页面上，验证服务器状态现在是否正常。

如果执行上述步骤后服务器状态仍为连接丢失，请尝试以下方法之一。

- 通过创建新服务器并删除原始服务器来替换服务器。如果当前服务器上的数据对您很重要，请从最近的备份中恢复服务器，并验证数据是否为最新，然后再删除未响应的原始服务器。
- 请联系 AWS 支持。

服务器创建失败，并返回“requested configuration is currently not supported”消息

问题：您尝试创建一台 Puppet Enterprise 服务器，但服务器创建失败，并返回与“The requested configuration is currently not supported。Please check the documentation for supported configurations.”类似的错误消息。

原因：可能为 Puppet Master 指定了不支持的实例类型。如果您选择在具有非默认租赁的 VPC 中创建 Puppet 服务器，例如适用于专用实例的 VPC，则指定 VPC 内的所有实例也必须为专用或主机租赁。由于某些实例类型 (如 t2) 只适用于默认租赁，指定 VPC 可能不支持 Puppet Master 实例类型，因此服务器创建失败。

解决方案：如果您选择具有非默认租赁的 VPC，请使用 m4 实例类型，此类型可以支持专用租赁。

无法创建服务器的 Amazon EC2 实例

问题：服务器创建失败，并返回类似以下的错误消息：“The following resource(s) failed to create: [EC2Instance]。Failed to receive 1 resource signal(s) within the specified duration.”

原因：很可能的原因是 EC2 实例没有网络访问权限。

解决方案：确保实例具有出站 Internet 访问权限，并且 AWS 服务代理能够发出命令。请确保您的 VPC (具有单一公有子网的 VPC) 已启用 DNS resolution，并且您的子网已启用 Auto-assign Public IP 设置。

服务角色错误阻止服务器创建

问题：服务器创建失败，并显示一条错误消息，上面写着 “未授权执行 sts:” AssumeRole。

原因：当您使用的服务角色缺少足够的权限创建新服务器时，可能会出现此问题。

解决方案：打开 Puppet Enterprise 控制台；使用控制台生成新的服务角色和实例配置文件角色。OpsWorks 如果您希望使用自己的服务角色，请将AWSOpsWorksCMServiceRole策略附加到该角色。验证 opsworks-cm.amazonaws.com 在角色的 Trust Relationships 中随服务一起列出。验证与 Puppet 主服务器关联的服务角色是否已附加AWSOpsWorksCMServiceRole托管策略。

超出弹性 IP 地址限制

问题：服务器创建失败，并返回错误消息，指示“The following resource(s) failed to create: [EIP, EC2Instance]。Resource creation cancelled, the maximum number of addresses has been reached.”

原因：当您的账户已使用最大数量的弹性 IP (EIP) 地址时，将会出现此问题。默认的 EIP 地址数量限制为 5。

解决方案：您可以释放现有 EIP 地址或删除您的账户未使用的 EIP 地址，也可以联系 Cust AWS omer Support 以增加与您的账户关联的 EIP 地址的限制。

无人参与节点关联失败

问题：新的 Amazon EC2 节点的无人参与或自动关联失败。Puppet Enterprise 控制面板中未显示应已添加到 Puppet Master 的节点。

原因：当您未将 IAM 角色设置为允许 `opsworks-cm` API 调用与新的 EC2 实例通信的实例配置文件时，可能会出现此问题。

解决方案：将一个策略附加到您的 EC2 实例配置文件以允许 `AssociateNode` 和 `DescribeNodeAssociationStatus` API 调用与 EC2 一起工作，如 OpsWorks 为 Puppet Enterprise 自动添加节点中所述。

系统维护失败

AWS OpsWorks CM 每周执行系统维护，以确保 AWS经过测试的最新版本的 Puppet Server（包括安全更新）始终在 Puppet Enterpr OpsWorks ise 服务器上运行。如果由于任何原因导致系统维护

失败，则 AWS OpsWorks CM 会通知您该故障。有关系统维护的更多信息，请参阅 Puppet Enterp OpsWorks rise 系统维护中。

本节介绍可能的失败原因并提出解决方案。

主题

- 服务角色或实例配置文件错误会阻止系统维护

服务角色或实例配置文件错误会阻止系统维护

问题：系统维护失败，并显示一条错误消息，上面写着 “未授权执行 sts:AssumeRole” 或类似的权限错误消息。

原因：当您使用的服务角色或实例配置文件缺少在服务器上执行系统维护的足够权限时，可能会发生这种情况。

解决方案：确保您的服务角色和实例配置文件具有所有必需的权限。

1. 请确保您的服务角色具有所有必需的权限。
   a. 在服务器的设置页面上，在网络和安全中，选择服务器正在使用的服务角色的链接。这将将服务角色打开以供在 IAM 控制台中查看。
   b. 在权限选项卡上，验证 `AWSOpsWorksCMServiceRole` 是否已附加到该服务角色。如果 `AWSOpsWorksCMServiceRole` 未列出，则将此策略添加到角色。
   c. 验证 opsworks-cm.amazonaws.com 在角色的 Trust Relationships 中随服务一起列出。有关如何对角色使用信任策略的更多信息，请参阅修改角色（控制台）或 AWS 安全博客文章《如何在 IAM 角色中使用信任策略》。
2. 请确保您的实例配置文件具有所有必需的权限。
   a. 在服务器的设置页面上，在网络和安全中，选择服务器正在使用的实例配置文件的链接。这将打开实例配置文件以在 IAM 控制台中查看。
   b. 在权限选项卡上，确认 `AmazonEC2RoleforSSM` 和 `AWSOpsWorksCMInstanceProfileRole` 是否在权限策略列表中。如果未列出其中一个或两个托管策略，请手动将这些托管策略添加到角色中。
   c. 在信任关系选项卡上，验证服务角色是否具有信任 `ec2.amazonaws.com` 服务代表您代入角色的信任策略。有关如何对角色使用信任策略的更多信息，请参阅修改角色（控制台）或 AWS 安全博客文章《如何在 IAM 角色中使用信任策略》。

其他帮助和支持

如果本主题没有描述您的特定问题，或者您已尝试本主题中的建议，但问题仍然存在，请访问 AWS OpsWorks 论坛。

您也可访问 AWS Support Center 。 AWS 支持中心是创建和管理 AWS 支持案例的中心。Su AWS pport Center 还包括指向其他有用资源的链接，例如论坛、技术常见问题解答、服务运行状况和 AWS Trusted Advisor。

AWS f OpsWorks or Chef 自动化

> **Important**
>
> AWS OpsWorks for Chef Automate 已于 2024 年 5 月 5 日停用，新客户和现有客户均已禁用。我们建议现有客户迁移到 Chef SaaS 或其他替代解决方案。如果您有任何疑问，可以通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

AWS OpsWorks for Chef Automate 允许你在 AWS 中运行 Chef Automat e 服务器。你可以在几分钟内配置一个 Chef 服务器，然后让它 AWS OpsWorks for Chef Automate 处理它的操作、备份、恢复和软件升级。 AWS OpsWorks for Chef Automate 让您可以腾出时间专注于核心配置管理任务，而不必管理 Chef 服务器。

Chef Automate 服务器通过指示在节点上运行`chef-client`哪些 Chef 配方来管理环境中节点的配置，存储有关节点的信息，并充当 Chef 食谱的中央存储库。 AWS OpsWorks for Chef Automate 提供包含 Chef Automate：Chef Infra 和 Chef 高级功能的 Che InSpec f 服务器。

AWS OpsWorks for Chef Automate 服务器在 Amazon 弹性计算云实例上运行。 AWS OpsWorks for Chef Automate 服务器配置为运行最新版本的亚马逊 Linux（亚马逊 Linux 2）。有关此版本的 Chef Automate 中所做更改的信息，请参阅 Chef Automate 发布说明。下表描述了安装在 AWS OpsWorks for Chef Automate 服务器上的 Chef 组件。

组件名称	描述	AWS OpsWorks for Chef Automate 服务器上安装的版本
Chef Automate	Chef Automate 是一个企业服务器软件包，可提供自动化的持续部署工作流，且可用于深入了解基于 Web 的管理控制台中的托管节点。Chef Automate通过包括Chef Infra来实现基础设施自动化，通过包括Chef来提供安全和合规性信息以及通过包括Chef Habitat来实现强制执行	2.0

组件名称	描述	AWS OpsWorks for Chef Automate 服务器上安装的版本
	InSpec，以及通过包括Chef Habitat 有关 Chef Automate 的更多信息，请参阅 Chef 网站上的 Chef Automate。	
Chef Infra	Chef Infra Server（以前称为 Chef Server）使用 Chef Infra Client (`chef-client`) 代理将配置持续应用到托管节点以保持所需状态。 有关更多信息，请参阅 Chef 网站上的 Chef Infra。	12.x
厨师 InSpec	Chef InSpec 描述了可以在软件工程师、运营和安全工程师之间共享的安全和合规性规则。合规性、安全性和其他策略要求构成了自动化测试的框架，`chef-client` 代理可以针对托管节点运行这些测试，以确保一致的实施标准。 有关更多信息 InSpec，请参阅 Chef InSpec 网站上的 Chef。	3.9.0

在与 AWS OpsWorks for Chef Automate 服务器关联的节点上，`chef-client` 的最低支持版本为 13.x。我们建议至少运行 14.10.9 或最新的、稳定的`chef-client`版本。

如果有新的 Chef 软件次要版本可用，并通过了 AWS 测试，系统维护机制会自动在服务器上更新 Chef Automate 和 Chef Server 的次要版本。AWS 会进行广泛的测试，以验证 Chef 升级是否可以投入生产，并且不会中断现有客户环境，因此 Chef 软件的发布与其应用到 Chef Automate 现有 OpsWorks服务器的可用性之间可能会存在延迟。系统维护还会将您的服务器升级到 Amazon Linux 的最新版本。

您可以将任何运行支持的操作系统并具有网络访问权限的本地计算机或 EC2 实例连接到 AWS OpsWorks for Chef Automate 服务器。要获取对应您要管理的节点的受支持操作系统列表，请登录 Chef 网站。`chef-client` 代理软件安装在您要通过 Chef 服务器管理的节点上。

主题

Region Support AWS OpsWorks for Chef Automate

以下区域终端节点支持 AWS OpsWorks for Chef Automate 服务器。 AWS OpsWorks for Chef Automate 在与 Chef 服务器相同的区域端点中创建与 Chef 服务器关联的资源，例如实例配置文件、用户和服务角色。您的 Chef 服务器必须在 VPC 中。您可以使用您创建的或已有的 VPC，或使用默认 VPC。

- 美国东部（俄亥俄州）区域
- 美国东部（弗吉尼亚州北部）区域
- 美国西部（北加利福尼亚）区域
- 美国西部（俄勒冈州）区域

- Asia Pacific（Tokyo）Region
- 亚太地区（新加坡）区域
- 亚太地区（悉尼）区域
- 欧洲地区（法兰克福）区域
- 欧洲地区（爱尔兰）区域

AWS OpsWorks for Chef Automate 生命周期终结

> **Important**
>
> AWS OpsWorks for Chef Automate 已于 2024 年 5 月 5 日停用，新客户和现有客户均已禁用。我们建议现有客户迁移到 Chef SaaS 或其他替代解决方案。

主题

生命周期终止将如何影响现有用户？

在2024年5月5日（Chef Automate的生命周期终止日期）之前，现有客户将不受影响。 OpsWorks 在生命周期终止日期之后，客户将无法再使用 OpsWorks 控制台或 API 管理其服务器。

如果我不采取任何行动，我的服务器会怎么样？

从 2024 年 5 月 5 日起，您将无法再使用 OpsWorks 控制台或 API 管理服务器。届时，我们将停止对您的服务器执行任何正在进行中的管理功能，例如备份或维护。为了限制对客户的影响，我们将让所

有正在备份 Chef Automate 服务器的 EC2 实例保持运行状态，但它们的许可证将不再有效，因为与 Chef 签订的 for Chef Automate 服务协议不再涵盖使用量（或计费）。 OpsWorks 您需要联系 Chef 以获得新的许可证。当你联系 Chef 时，一定要告诉他们你是 Chef Automate OpsWorks 的现有客户，而且你正在从中 OpsWorks过渡。

我可以过渡到哪些替代方案？

AWS Progress Chef 建议你迁移到他们的新 Chef SaaS 产品，这样你就可以继续从完全托管的 Chef Automate 服务中受益。要开始使用 Chef SaaS，您可以联系 Chef 获取有关如何设置 Chef SaaS 账户以及转换数据和节点的文档。

如果 Chef SaaS 无法满足你的需求，因为你更喜欢在自己控制的 AWS 账户中的 EC2 实例上运行 Chef Automate，那么 Chef 提供了多种选择，包括自AWS Marketplace 带许可 (BYOL) 模式和在 EC2 上自托管。您可以联系 Progress Chef 以获取有关如何执行此类过渡的更多信息。

该服务还在接受新客户吗？

不是。 AWS OpsWorks 因为 Chef Automate 不再接受新客户。

生命的尽头会同时影响所有 AWS 区域 人吗？

是。自 2024 年 5 月 5 日起，API 和控制台将进入生命周期终止期，无法在所有 AWS 区域中使用。有关 Chef Automate AWS 区域 在哪里 AWS OpsWorks 可用的信息，请参阅AWS 区域服务列表。

提供什么级别的技术支持？

AWS 将继续为Chef Automate OpsWorks 提供与客户今天相同的支持水平，直到生命周期终止日期。如果您有任何疑问或疑虑，可以通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。如需过渡支持，我们建议客户联系 Progress Chef。

我是 Chef Automate 的 OpsWorks 现有客户，我需要使用以前未使用该服务的账户启动服务器。我能做到吗？

通常不能，除非有特殊情况需要这样做。如果您有特殊情况，请通过 re AWS : Post 或通过 Pre mium Su AWS pport 联系 AWS Support 团队，提供详细信息和理由，我们将审核您的请求。

明年会有新主要功能的发布吗？

没有。由于该服务已将至生命周期终止，因此我们不会发布任何新功能。但是，在生命周期终止之前，我们将继续改进安全性并按预期管理服务器。

将 AWS OpsWorks for Chef Automate 服务器升级到 Chef Automate 2

> **Important**
>
> AWS OpsWorks for Chef Automate 已于 2024 年 5 月 5 日停用，新客户和现有客户均已禁用。我们建议现有客户迁移到 Chef SaaS 或其他替代解决方案。如果您有任何疑问，可以通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

升级到 Chef Automate 2 的先决条件

在开始之前，请确保您了解 Chef Automate 2 添加的新功能以及 Chef Automate 2 不支持的功能。有关 Chef Automate 2 中的新功能和不支持的功能的信息，请参阅 Chef 网站上的 Chef Automate 2 文档。

运行 Chef Automate 1 的服务器必须在 2019 年 11 月 1 日之后至少有一次成功的维护运行，才有资格升级。

与 AWS OpsWorks for Chef Automate 服务器上的任何维护操作一样，服务器在升级期间处于脱机状态。您应该计划在升级过程中有长达三个小时的停机时间。

对于 Chef Automate 控制面板网站，您需要此服务器的登录凭证。升级完成后，您应登录 Chef Automate 控制面板，并验证节点和配置信息是否未更改。

> **Important**
>
> 当你准备好将 AWS OpsWorks for Chef Automate 服务器升级到 Chef Automate 2 时，请仅使用此处的说明进行升级。由于 AWS OpsWorks for Chef Automate 可以自动执行许多升级过程，例如创建备份，因此请不要按照Chef网站上的升级说明进行操作。

关于升级过程

在升级过程中，您的服务器会在开始升级之前和完成升级之后进行备份。将创建以下备份：

- 服务器仍在运行 Chef Automate 1（版本 12.17.33）时的备份。
- 完成升级且服务器正在运行 Chef Automate 2（版本 2019-08）后的服务器备份。

升级过程会终止服务器在运行 Chef Automate 1 时所使用的 Amazon EC2 实例。创建一个新实例来运行 Chef Automate 2 服务器。

升级到 Chef Automate 2（控制台）

1. 登录 AWS Management Console 并打开 AWS OpsWorks 控制台，网址为 https://console.aws.amazon.com/opsworks/。
2. 在左侧导航窗格中，选择 AWS OpsWorks for Chef Automate。
3. 选择服务器以查看其属性页面。页面顶部的蓝色横幅应表明服务器是否有资格升级到 Chef Automate 2。

   > **Note**
   >
   > 运行 Chef Automate 1 的服务器必须在 2019 年 11 月 1 日之后至少有一次成功的维护运行，才有资格升级。

4. 如果服务器符合升级条件，请选择 Start upgrade (开始升级)。
5. 留出多达三个小时进行升级。在升级过程中，属性页面将服务器状态显示为 Under maintenance (正在维护中)。
6. 升级完成后，属性页将显示以下两条消息： Successfully upgraded to Automate 2 (成功升级到 Automate 2) 和 Maintenance completed successfully (成功完成维护)。服务器状态应为 HEALTHY (运行状况良好)。
7. 使用您的现有凭证登录到 Chef Automate 控制面板，并验证您的节点是否正确报告。

升级到 Chef Automate 2 (CLI)

1. （可选）如果您不确定哪些 AWS OpsWorks for Chef Automate 服务器符合升级条件，请运行以下命令。如果您要列出不同于默认 AWS 区域的 AWS 区域中的 AWS OpsWorks for Chef Automate 服务器，请务必添加`--region`参数。

   ```
   aws opsworks-cm describe-servers
   ```

   在结果中，查找属性 `CHEF_MAJOR_UPGRADE_AVAILABLE` 的 `true` 值。这表明服务器有资格升级到 Chef Automate 2。记下符合升级条件的 AWS OpsWorks for Chef Automate 服务器的名称。
2. 运行以下命令，将 s *erver_name* 替换为服务器的 AWS OpsWorks for Chef Automate 名称。要升级到 Chef Automate 2，而不是执行日常系统维护，请添加 `CHEF_MAJOR_UPGRADE` 引擎属

性，如此命令中所示。如果目标服务器不在您的默认 Amazon Web Services Region 中，请添加 `--region` 参数。每个命令只能升级一个服务器。

```
aws opsworks-cm start-maintenance --server-name server_name --engine-attributes
 Name=CHEF_MAJOR_UPGRADE,Value=true --region region
```

如果由于任何原因 AWS OpsWorks for Chef Automate 无法升级服务器，则此命令将导致验证异常。

3. 留出多达三个小时来进行升级。您可以通过运行以下命令定期检查升级状态。

```
aws opsworks-cm describe-servers --server-name server_name
```

在结果中，查找 `Status` 值。`Status` 为 `UNDER_MAINTENANCE` 意味着升级仍在进行中。成功升级后将返回类似于以下内容的消息。

```
2019/10/24 00:27:56 UTC          Successfully upgraded to Automate 2.
2019/10/23 23:50:38 UTC          Upgrading Chef server from Automate 1 to Automate
 2
```

如果升级不成功，则 AWS OpsWorks for Chef Automate 自动将服务器回滚到 Chef Automate 1。

如果升级成功，但服务器的运行与升级之前不同（例如，如果托管节点不报告），则可以手动回滚服务器。有关手动回滚信息，请参阅 [将 AWS OpsWorks for Chef Automate 服务器回滚到 Chef Automate 1 (CLI)]。

将 AWS OpsWorks for Chef Automate 服务器回滚到 Chef Automate 1 (CLI)

如果升级过程失败，则 AWS OpsWorks for Chef Automate 自动将服务器回滚到 Chef Automate 1。如果升级成功但服务器的运行方式与升级前不同，则可以使用手动将 AWS OpsWorks for Chef Automate 服务器回滚到 Chef Automate 1 AWS CLI。

1. 运行以下命令以显示您在尝试升级之前在服务器上执行的上次备份的 `BackupId`。如果您的服务器位于与默认 Amazon Web Services Region 不同的 Amazon Web Services Region 中，请添加 `--region` 参数。

```
aws opsworks-cm describe-backups server_name
```

Backup ID 的格式为 *ServerName-yyyymmddhhmmssss* s。在结果中查找以下 Chef Automate 1 属性。

```
"Engine": "Chef"
"EngineVersion": "12.17.33"
```

2. 运行以下命令，使用您在步骤 1 中返回的备份 ID 作为 `--backup-id` 的值。

```
aws opsworks-cm restore-server --server-name server_name --backup-id ServerName-yyyyMMddHHmmssSSS
```

根据您存储在服务器上的数据量，请花费 20 分钟到 3 小时的时间来还原服务器。在还原操作期间，服务器的状态为 `RESTORING`。此状态显示在服务器的属性页面上 AWS Management Console，并在describe-servers命令的结果中返回。

3. 还原完成后，控制台显示消息 Restore completed successfully (还原成功完成)。您的 AWS OpsWorks for Chef Automate 服务器处于联机状态，与您开始升级过程之前相同。

另请参阅

- 中的系统维护 AWS OpsWorks for Chef Automate
- 从 Backup 中恢复 AWS OpsWorks for Chef Automate 服务器
- 《AWS OpsWorks API 参考》中的 DescribeServers
- 《AWS OpsWorks API 参考》中的 StartMaintenance

入门 AWS OpsWorks for Chef Automate

> **Important**
>
> AWS OpsWorks for Chef Automate 已于 2024 年 5 月 5 日停用，新客户和现有客户均已禁用。我们建议现有客户迁移到 Chef SaaS 或其他替代解决方案。如果您有任何疑问，可以通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

AWS OpsWorks for Chef Automate 允许你在中运行 Chef Automat e 服务器 AWS。您可以在大约 15 分钟内预配置一个 Chef Server。

从 2021 年 5 月 3 日起，将一些 Chef Automate 服务器属性 AWS OpsWorks for Chef Automate 存储在中 AWS Secrets Manager。有关更多信息，请参阅 与 AWS Secrets Manager 集成。

以下演练可帮助您在中创建第一台 Chef 服务器。 AWS OpsWorks for Chef Automate

先决条件

在开始之前，您必须完成以下前提条件：

主题

设置 VPC

您的 AWS OpsWorks for Chef Automate 服务器必须在亚马逊 Virtual Private Cloud 中运行。您可以将其添加到现有 VPC、使用默认 VPC，或者创建新 VPC 以包含服务器。有关创建 Amazon VPC 以及如何创建新 VPC 的信息，请参阅 Amazon VPC 入门指南。

如果您创建自己的 VPC，或者使用现有的 VPC，则它应具有以下设置或属性。

- VPC 应具有至少一个子网。

  如果您的 AWS OpsWorks for Chef Automate 服务器可以公开访问，请将子网设为公有子网，然后启用自动分配公有 IP。
- 应该启用 DNS resolution。
- 在子网上，启用 Auto-assign public IP。

如果您不熟悉创建 VPC 或在其中运行实例，则可以使用为您 AWS OpsWorks 提供的 AWS CloudFormation 模板运行以下 AWS CLI 命令来创建具有单个公有子网的 VPC。如果您更喜欢使用 AWS Management Console，也可以将模板上传到 AWS CloudFormation 控制台。

```
aws cloudformation create-stack --stack-name OpsWorksVPC --template-url https://
s3.amazonaws.com/opsworks-cm-us-east-1-prod-default-assets/misc/opsworks-cm-vpc.yaml
```

使用自定义域的先决条件（可选）

您可以在自己的域上设置 Chef Automate 服务器，同时在自定义域中指定一个公有终端节点用作服务器的终端节点。当您使用自定义域时，需要执行以下所有操作，如本节中详细介绍。

主题

设置自定义域

要在自己的自定义域上运行 Chef Automate 服务器，您需要服务器的公有终端节点，例如 `https://aws.my-company.com`。如果指定自定义域，还必须提供证书和私有密钥，如前面各节所述。

要在创建服务器后访问此服务器，请在首选 DNS 服务中添加 CNAME DNS 记录。此记录必须将自定义域指向由 Chef Automate 服务器创建过程生成的终端节点（服务器的 `Endpoint` 属性的值）。如果服务器使用自定义域，则无法使用生成的 `Endpoint` 值访问服务器。

获取证书

要在您自己的自定义域上设置您的 Chef Automate 服务器，您需要 PEM 格式的 HTTPS 证书。这可以是单个自签名证书或证书链。在完成 创建 Create Chef Automate 服务器 工作流时，如果您指定此证书，则还必须提供自定义域和私有密钥。

以下是证书值的要求：

- 您可以提供自签名的自定义证书或完整的证书链。
- 证书必须是有效的 X509 证书或 PEM 格式的证书链。
- 证书在上传时必须是有效的。您不能在证书有效期开始（证书的 `NotBefore` 日期）之前或证书有效期到期（证书的 `NotAfter` 日期）之后使用证书。
- 证书的公用名称或使用者备用名称 (SAN)（如果存在）必须与自定义域值匹配。
- 证书必须与 Custom private key (自定义私有密钥) 字段的值匹配。

获取私有密钥

要在自己的自定义域上设置 Chef Automate 服务器，您需要一个 PEM 格式的私有密钥，以便使用 HTTPS 连接到服务器。私有密钥不得加密；无法使用密码或密码短语保护它。如果指定自定义私有密钥，则还必须提供自定义域和证书。

设置 EC2 密钥对（可选）

对于典型的 Chef Server 管理，通常不需要或不建议使用 SSH 连接，您可以使用 `knife` 命令在 Chef Server 上执行大部分管理任务。

如果您丢失或者希望更改 Chef Automate 控制面板的登录密码，则需要使用 EC2 密钥对通过 SSH 连接到您的服务器。您可以使用现有的密钥对，或者创建新的密钥对。有关如何创建新 EC2 密钥对的更多信息，请参阅 Amazon EC2 密钥对。

如果您不需要 EC2 密钥对，那么您已准备好创建 Chef Server。

创建 Chef Automate 服务器

> **Important**
>
> AWS OpsWorks for Chef Automate 已于 2024 年 5 月 5 日停用，新客户和现有客户均已禁用。我们建议现有客户迁移到 Chef SaaS 或其他替代解决方案。如果您有任何疑问，可以通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

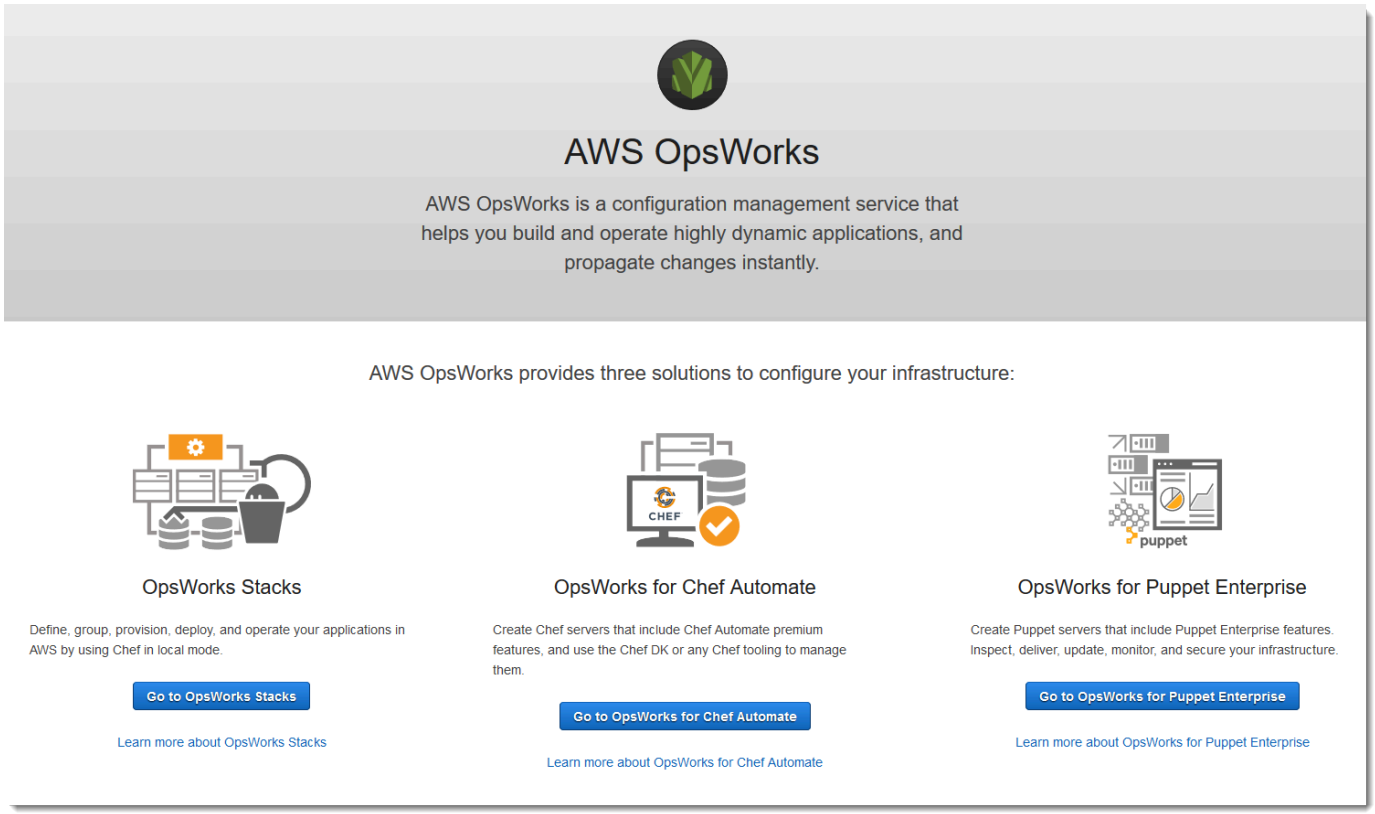
您可以使用 AWS OpsWorks for Chef Automate 控制台创建 Chef 服务器，或者 AWS CLI.

主题

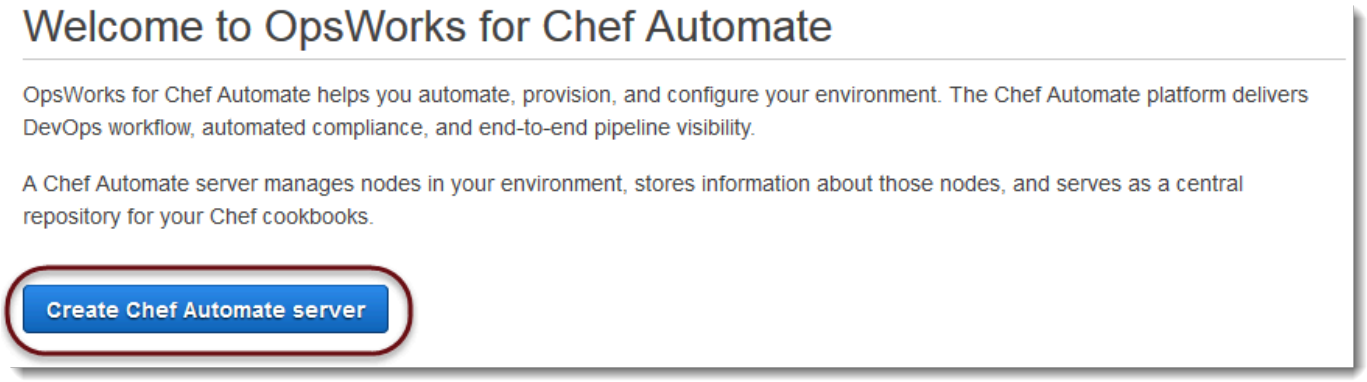
- 在中创建一个 Chef Automate 服务器 AWS Management Console
- 使用 Chef Automate 服务器创建 AWS CLI

在中创建一个 Chef Automate 服务器 AWS Management Console

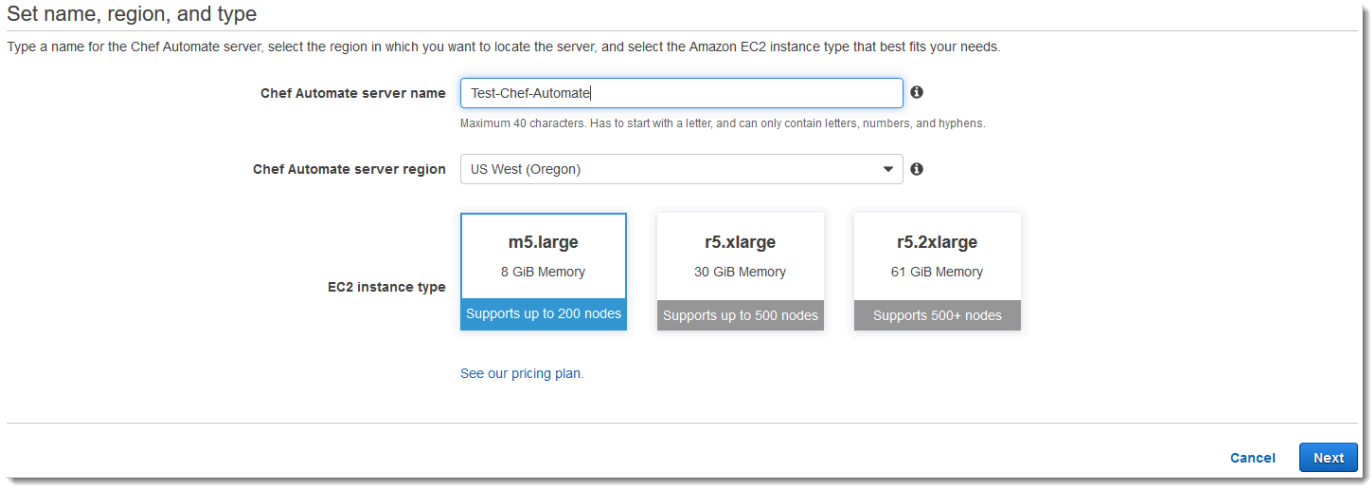
1. 登录 AWS Management Console 并打开 AWS OpsWorks 控制台，网址为 https://console.aws.amazon.com/opsworks/。
2. 在 AWS OpsWorks 主页上，选择 Chef Automate OpsWorks 的 “前往”。



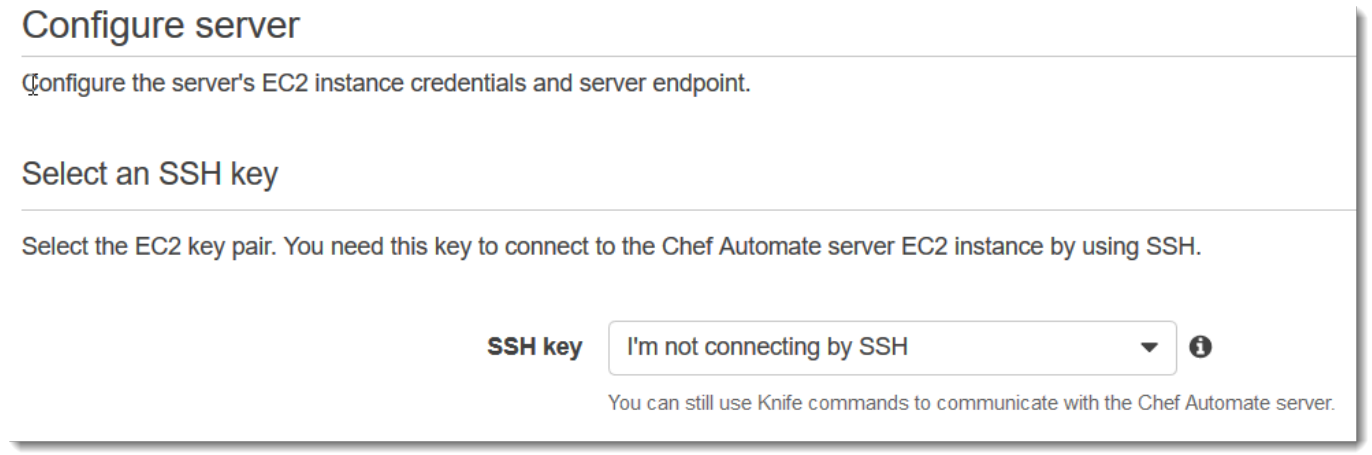
3. 在 AWS OpsWorks for Chef Automate 主页上，选择创建 Chef 自动服务器。



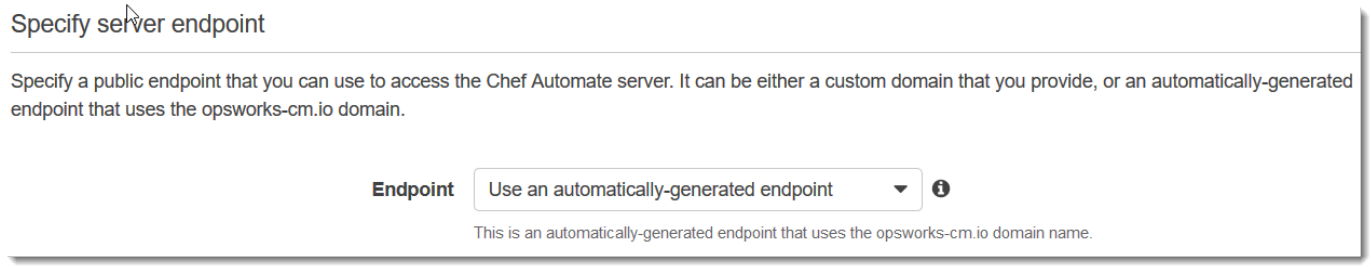
4. 在 Set name, region, and type (设置名称、区域和类型) 页面上，指定服务器的名称。Chef Server 名称最多可包含 40 个字符，并且只能包含字母数字字符和短划线。选择支持的区域，然后选择支持所要管理节点数量的实例类型。如果需要，您可以在服务器创建之后更改实例类型。对于本演练，我们在美国西部（俄勒冈州）区域中创建一个 m4.large 实例类型。选择下一步。



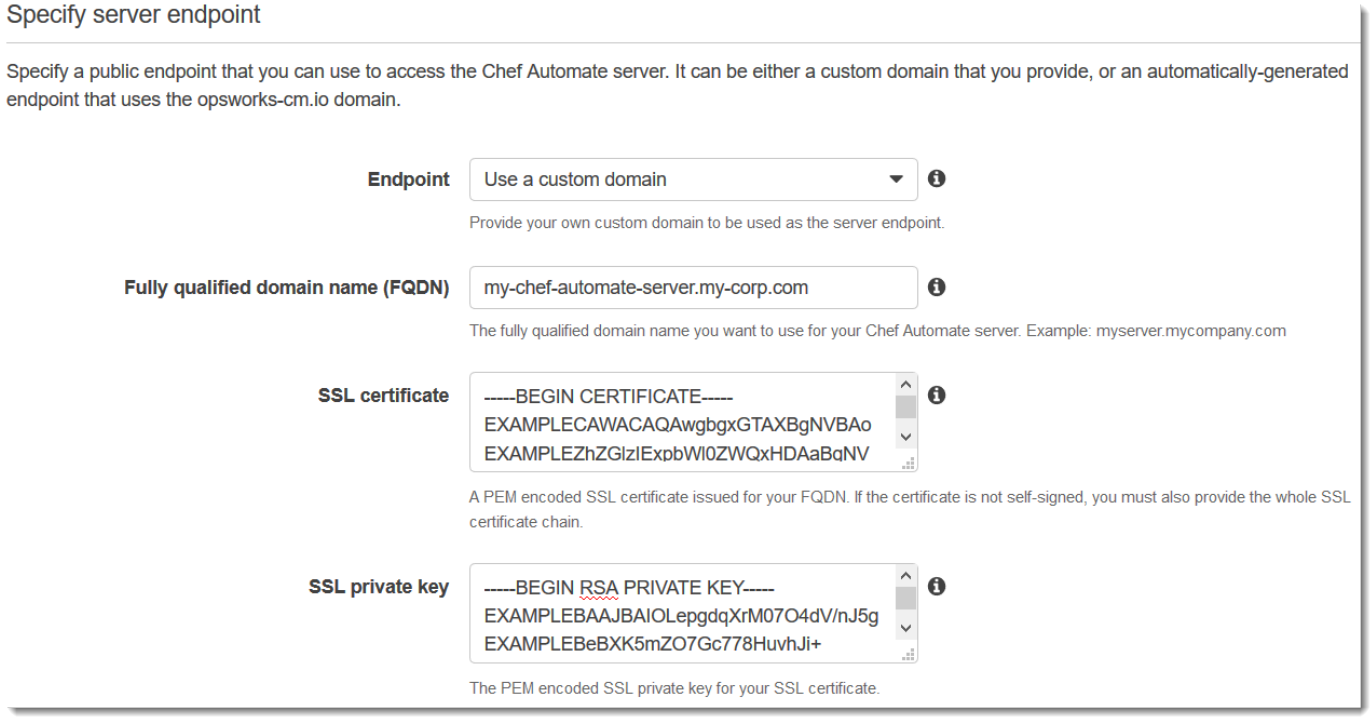
5. 在 Configure server 页面上，除非您要指定密钥对名称，否则保留 SSH key 下拉列表中的默认选择。



6. 对于 Specify server endpoint (指定服务器终端节点)，保留默认值 Use an automatically-generated endpoint (使用自动生成的终端节点)，然后选择 Next (下一步)，除非您希望您的服务器位于您自己的自定义域中。要配置自定义域，请继续执行下一步。



7. 要使用自定义域，对于 Specify server endpoint (指定服务器终端节点)，从下拉列表中选择 Use a custom domain (使用自定义域)。



a. 对于完全限定域名 (FQDN)，指定 FQDN。您必须拥有要使用的域名。

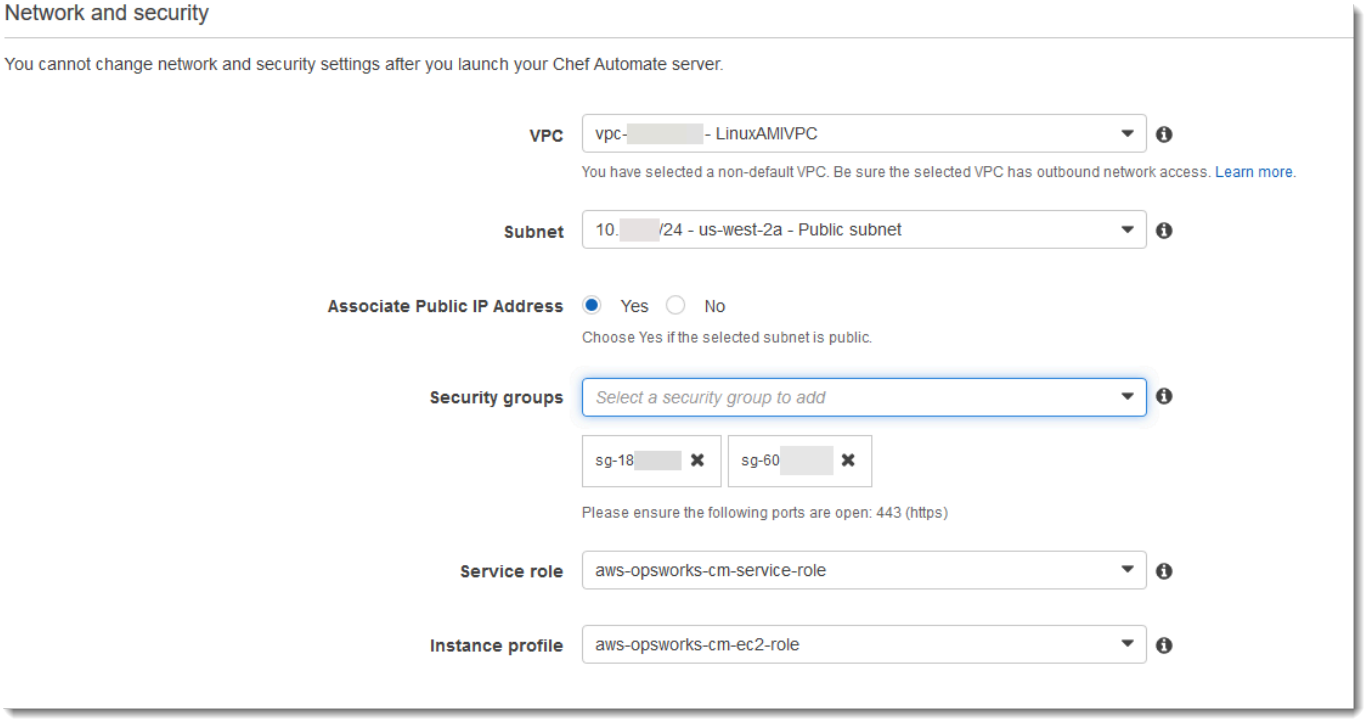
b. 对于 SSL 证书，请粘贴整个 PEM 格式的证书，以 `-----BEGIN CERTIFICATE-----` 开头并以 `-----END CERTIFICATE-----` 结尾。SSL 证书主题必须与您在上一步中输入的 FQDN 相匹配。

c. 对于 SSL private key (SSL 私有密钥)，请粘贴整个 RSA 私有密钥，以 `-----BEGIN RSA PRIVATE KEY-----` 开头并以 `-----END RSA PRIVATE KEY-----` 结尾。SSL 私有密钥必须与您在上一步中输入的 SSL 证书中的公有密钥匹配。选择下一步。

8. 在 Configure Advanced Settings 页面上的 Network and Security 区域中，选择 VPC、子网以及一个或多个安全组。以下是 对您 VPC 的要求：

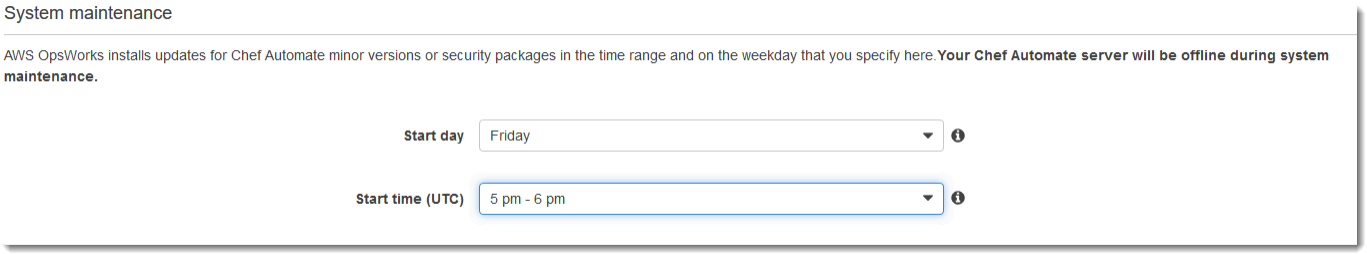
   - VPC 必须只有具有一个公有子网。
   - 必须启用 DNS 解析。
   - 必须在公有子网上启用自动分配公有 IP。

   AWS OpsWorks 如果您还没有要使用的安全组、服务角色和实例配置文件，则可以为您生成安全组、服务角色和实例配置文件。您的服务器可属于多个安全组。在离开此页后，您无法更改 Chef Server 的网络和安全设置。

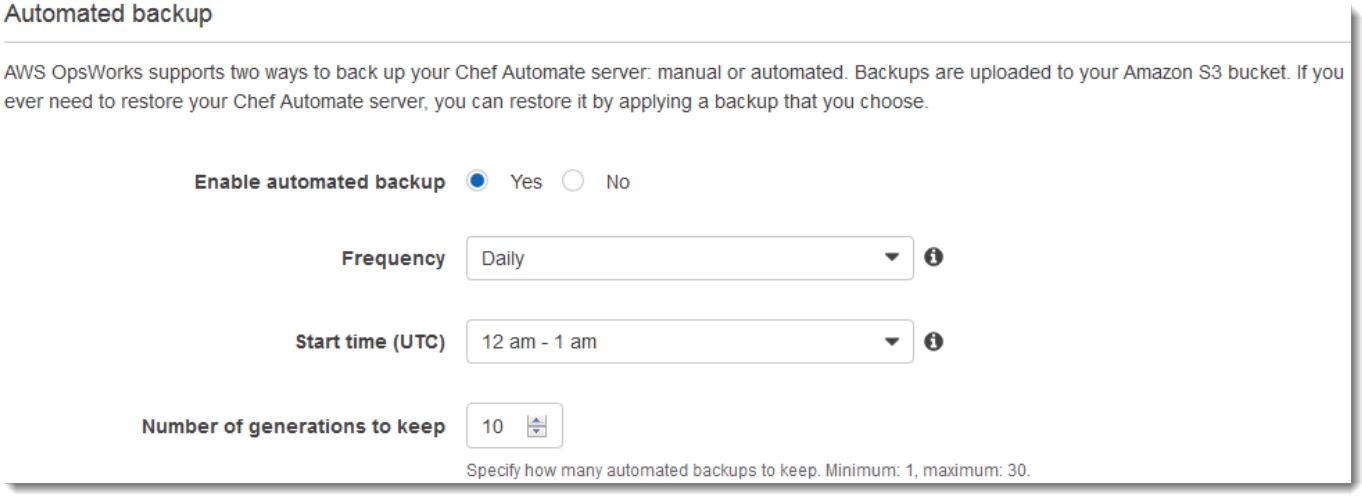


9. 在 System maintenance (系统维护) 部分中，设置您希望系统维护开始的日期和时间。由于您可以预见到服务器会在系统维护期间脱机，因此请选择正常工作时间中对服务器需求较低的时间。维护完成之前，已连接的节点进入 `pending-server` 状态。

   维护时段是必需的。您可以稍后使用 AWS Management Console AWS CLI、或 API 更改开始日期和时间。



10. 配置备份。默认情况下会启用自动备份。设置自动备份启动的首选频率和时间，并设置在 Amazon Simple Storage Service 中存储的备份生成数。最多保留 30 个备份；当达到最大值时， AWS OpsWorks for Chef Automate 会删除最旧的备份，以便为新备份腾出空间。



11. （可选）在 Tags (标签) 中，向服务器和相关资源（如 EC2 实例、弹性 IP 地址、安全组、S3 存储桶和备份）添加标签。有关为 AWS OpsWorks for Chef Automate 服务器添加标签的更多信息，请参见使用 AWS OpsWorks for Chef Automate 资源上的标签。
12. 在您配置完高级设置后，选择 Next (下一步)。
13. 在审核页面上，审核您的选择。当您准备好创建服务器时，选择 Launch (启动)。

    在等待 AWS OpsWorks 创建 Chef 服务器时，请继续下载入门套件使用初学者工具包配置 Chef Server和 Chef Automate 仪表板凭据。请不要一直等到您的服务器联机再下载这些项。

    服务器创建完成后，您的 Chef Server 可在 AWS OpsWorks for Chef Automate 主页上使用，其状态为 online (在线)。服务器联机之后，Chef Automate 控制面板在服务器的域上可用，位于以下格式的 URL 上：`https://your_server_name-random.region.opsworks-cm.io`。

使用 Chef Automate 服务器创建 AWS CLI

通过运行 AWS CLI 命令创建 AWS OpsWorks for Chef Automate 服务器不同于在控制台中创建服务器。如果您未指定要使用的现有服务角色和安全组，则可以在控制台中为您 AWS OpsWorks 创建服务角色和安全组。在中 AWS CLI，如果您未指定安全组，则 AWS OpsWorks 可以为您创建安全组，但它不会自动创建服务角色；您必须在命令中提供服务角色 ARN。`create-server`在控制台中，在创建 Ch AWS OpsWorks ef Automate 服务器时，你可以下载 Chef Automate 入门套件和 Chef Automate 仪表板的登录凭证。由于使用创建服务器时无法执行此操作，因此在新 AWS OpsWorks for Chef Automate AWS OpsWorks for Chef Automate 服务器联机后 AWS CLI，您可以使用 JSON 处理实用程序从`create-server`命令的结果中获取登录凭证和入门套件。或者， AWS OpsWorks for Chef Automate 服务器联机之后，您可以在控制台中生成一组新的登录凭证和一个新的初学者工具包。

如果您的本地计算机尚未运行 AWS CLI，请 AWS CLI 按照 AWS 命令行界面用户指南中的安装说明下载并安装。本部分未介绍可以与 `create-server` 命令结合使用的所有参数。有关 `create-server` 参数的更多信息，请参阅 `create-server` 参考 中的 AWS CLI 。

1. 请确保完成先决条件，尤其是设置 VPC，或确保您有要使用的现有 VPC。要创建 Chef Automate 服务器，您需要一个子网 ID。
2. (可选) 通过使用 OpenSSL 来生成 Chef 关键密钥，将该密钥保存到您的本地计算机上安全、方便的文件中。如果您在 `create-server` 命令中不提供，则该关键密钥将作为服务器创建过程的一部分自动生成。如果要跳过此步骤，您可以改为从 `create-server` 命令的结果中获取 Chef Automate 关键密钥。如果您选择使用以下命令生成关键密钥，请务必包含 `-pubout` 参数，因为 Chef Automate 关键密钥值是 RSA 密钥对的公开部分。有关更多信息，请参阅步骤 6。

   ```
   umask 077
   openssl genrsa -out "pivotal" 2048
   openssl rsa -in "pivotal" -pubout
   ```

3. 创建服务角色和实例配置文件。 AWS OpsWorks 提供了一个可用于创建两者的 AWS CloudFormation 模板。运行以下 AWS CLI 命令创建一个 AWS CloudFormation 堆栈，用于为您创建服务角色和实例配置文件。

   ```
   aws cloudformation create-stack --stack-name OpsWorksCMRoles --template-url
    https://s3.amazonaws.com/opsworks-cm-us-east-1-prod-default-assets/misc/opsworks-
   cm-roles.yaml --capabilities CAPABILITY_NAMED_IAM
   ```

4. 创建 AWS CloudFormation 完堆栈后，在您的账户中查找并复制服务角色的 ARN。

   ```
   aws iam list-roles --path-prefix "/service-role/" --no-paginate
   ```

   在 `list-roles` 命令的结果中，查找类似于以下内容的服务角色 ARN 条目。记下服务角色 ARN。您需要这些值来创建您的 Chef Automate 服务器。

   ```
   {
       "AssumeRolePolicyDocument": {
           "Version": "2012-10-17",
           "Statement": [
               {
                   "Action": "sts:AssumeRole",
                   "Effect": "Allow",
                   "Principal": {
                       "Service": "ec2.amazonaws.com"
   ```

```
                }
            }
        ]
    },
    "RoleId": "AROZZZZZZZZZZQG6R22HC",
    "CreateDate": "2018-01-05T20:42:20Z",
    "RoleName": "aws-opsworks-cm-ec2-role",
    "Path": "/service-role/",
    "Arn": "arn:aws:iam::000000000000:role/service-role/aws-opsworks-cm-ec2-role"
},
{
    "AssumeRolePolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Action": "sts:AssumeRole",
                "Effect": "Allow",
                "Principal": {
                    "Service": "opsworks-cm.amazonaws.com"
                }
            }
        ]
    },
    "RoleId": "AROZZZZZZZZZZZZZZZ6QE",
    "CreateDate": "2018-01-05T20:42:20Z",
    "RoleName": "aws-opsworks-cm-service-role",
    "Path": "/service-role/",
    "Arn": "arn:aws:iam::000000000000:role/service-role/aws-opsworks-cm-service-
role"
}
```

5. 查找并复制您的账户中的实例配置文件的 ARN。

```
aws iam list-instance-profiles --no-paginate
```

在 `list-instance-profiles` 命令的结果中，查找类似于以下内容的实例配置文件 ARN 条目。记录实例配置文件 ARN。您需要这些值来创建您的 Chef Automate 服务器。

```
{
    "Path": "/",
    "InstanceProfileName": "aws-opsworks-cm-ec2-role",
    "InstanceProfileId": "EXAMPLEDC6UR3LTUW7VHK",
    "Arn": "arn:aws:iam::123456789012:instance-profile/aws-opsworks-cm-ec2-role",
```

```
    "CreateDate": "2017-01-05T20:42:20Z",
    "Roles": [
        {
            "Path": "/service-role/",
            "RoleName": "aws-opsworks-cm-ec2-role",
            "RoleId": "EXAMPLEE4STNUQG6R22HC",
            "Arn": "arn:aws:iam::123456789012:role/service-role/aws-opsworks-cm-
ec2-role",
            "CreateDate": "2017-01-05T20:42:20Z",
            "AssumeRolePolicyDocument": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {
                            "Service": "ec2.amazonaws.com"
                        },
                        "Action": "sts:AssumeRole"
                    }
                ]
            }
        }
    ]
},
```

6. 通过运行`create-server`命令创建 AWS OpsWorks for Chef Automate 服务器。

   - `--engine` 值为 `ChefAutomate`，`--engine-model` 为 `Single`，`--engine-version` 为 `12`。
   - 在您的 AWS 账户中，服务器名称在每个区域内必须是唯一的。服务器名称必须以字母开头；然后允许字母、数字或连字符 (-)，最多 40 个字符。
   - 使用您在步骤 4 和 5 中复制的实例配置文件 ARN 和服务角色 ARN。
   - 有效实例类型为 `m5.large`、`r5.xlarge` 或 `r5.2xlarge`。有关这些实例类型的规范的更多信息，请参阅 Amazon EC2 用户指南中的实例类型。
   - `--engine-attributes` 参数是可选的；如果您不指定一个或两个值，则服务器创建过程会为您生成这些值。如果您添加 `--engine-attributes`，请指定您在步骤 2 中生成的 `CHEF_AUTOMATE_PIVOTAL_KEY` 值、`CHEF_AUTOMATE_ADMIN_PASSWORD`，或两者。

     如果您不设置 `CHEF_AUTOMATE_ADMIN_PASSWORD` 的值，一个密码将生成并作为 `create-server` 响应的一部分返回。您也可以在控制台中再次下载初学者工具包，该工具包会重新生成此密码。该密码的最小长度为 8 个字符，最大长度为 32 个字符。该密码可以包含字母、数字和

特殊字符 (!/@#$%^+=_)。该密码必须至少包含一个小写字母、一个大写字母、一个数字和一个特殊字符。

- SSH 密钥对是可选的，但可以帮助您连接到 Chef Automate 服务器 (如果您需要重置 Chef Automate 控制面板管理员密码)。有关创建 SSH 密钥对的更多信息，请参阅《Amazon EC2 用户指南》中的 Amazon EC2 密钥对。
- 要使用自定义域，请将以下参数添加到命令中。否则，Chef Automate 服务器创建过程会自动为您生成终端节点。配置自定义域需要所有三个参数。有关使用这些参数的其他要求的信息，请参阅 AWS OpsWorks CM API 参考CreateServer中的。
  - `--custom-domain` - 服务器的可选公有端点，例如 `https://aws.my-company.com`。
  - `--custom-certificate` - PEM 格式的 HTTPS 证书。该值可以是单个自签名证书或证书链。
  - `--custom-private-key` - PEM 格式的私有密钥，用于通过 HTTPS 连接到服务器。私有密钥不得加密；无法使用密码或密码短语保护它。
- 需要每周进行系统维护。有效值必须按以下格式指定：`DDD:HH:MM`。指定的时间为协调世界时 (UTC)。如果您不指定 `--preferred-maintenance-window` 的值，则默认值为星期二、星期三或星期五的一小时随机时间段。
- `--preferred-backup-window` 的有效值必须按以下格式之一指定：`HH:MM` (针对每日备份) 或 `DDD:HH:MM` (针对每周备份)。指定的时间采用 UTC 格式。默认值为随机每日开始时间。要退出自动备份，请改为添加参数 `--disable-automated-backup`。
- 对于 `--security-group-ids`，输入一个或多个安全组 ID，用空格分隔。
- 对于 `--subnet-ids`，输入子网 ID。

```
aws opsworks-cm create-server --engine "ChefAutomate" --engine-model "Single"
 --engine-version "12" --server-name "server_name" --instance-profile-arn
 "instance_profile_ARN" --instance-type "instance_type" --engine-attributes
 '{"CHEF_AUTOMATE_PIVOTAL_KEY":"pivotal_key","CHEF_AUTOMATE_ADMIN_PASSWORD":"password"}'
 --key-pair "key_pair_name" --preferred-maintenance-window
 "ddd:hh:mm" --preferred-backup-window "ddd:hh:mm" --security-group-
ids security_group_id1 security_group_id2 --service-role-arn "service_role_ARN" --
subnet-ids subnet_ID
```

示例如下：

```
aws opsworks-cm create-server --engine "ChefAutomate" --engine-
model "Single" --engine-version "12" --server-name "automate-06" --
```

```
instance-profile-arn "arn:aws:iam::12345678912:instance-profile/aws-
opsworks-cm-ec2-role" --instance-type "m5.large" --engine-attributes
 '{"CHEF_AUTOMATE_PIVOTAL_KEY":"MZZE...Wobg","CHEF_AUTOMATE_ADMIN_PASSWORD":"zZZzDj2DLYXSZF
 --key-pair "amazon-test" --preferred-maintenance-window "Mon:08:00" --preferred-
backup-window "Sun:02:00" --security-group-ids sg-b00000001 sg-b0000008 --service-
role-arn "arn:aws:iam::12345678912:role/service-role/aws-opsworks-cm-service-role"
 --subnet-ids subnet-300aaa00
```

以下示例创建使用自定义域的 Chef Automate 服务器。

```
aws opsworks-cm create-server --engine "ChefAutomate" --engine-model "Single" --
engine-version "12" \
    --server-name "my-custom-domain-server" \
    --instance-profile-arn "arn:aws:iam::12345678912:instance-profile/aws-opsworks-
cm-ec2-role" \
    --instance-type "m5.large" \
    --engine-attributes
 '{"CHEF_AUTOMATE_PIVOTAL_KEY":"MZZE...Wobg","CHEF_AUTOMATE_ADMIN_PASSWORD":"zZZzDj2DLYXSZF
 \
    --custom-domain "my-chef-automate-server.my-corp.com" \
    --custom-certificate "-----BEGIN CERTIFICATE----- EXAMPLEqEXAMPLE== -----END
 CERTIFICATE-----" \
    --custom-private-key "-----BEGIN RSA PRIVATE KEY----- EXAMPLEqEXAMPLE= -----END
 RSA PRIVATE KEY-----" \
    --key-pair "amazon-test" \
    --preferred-maintenance-window "Mon:08:00" \
    --preferred-backup-window "Sun:02:00" \
    --security-group-ids sg-b00000001 sg-b0000008 \
    --service-role-arn "arn:aws:iam::12345678912:role/service-role/aws-opsworks-cm-
service-role" \
    --subnet-ids subnet-300aaa00
```

以下示例创建了添加两个标签的 Chef Automate 服务器：`Stage: Production` 和 `Department: Marketing`。有关在 AWS OpsWorks for Chef Automate 服务器上添加和管理标签的更多信息，请参阅本指南使用 AWS OpsWorks for Chef Automate 资源上的标签中的。

```
aws opsworks-cm create-server --engine "ChefAutomate" --engine-model "Single" --
engine-version "12" \
    --server-name "my-test-chef-server" \
    --instance-profile-arn "arn:aws:iam::12345678912:instance-profile/aws-opsworks-
cm-ec2-role" \
    --instance-type "m5.large" \
```

```
    --engine-attributes
 '{"CHEF_AUTOMATE_PIVOTAL_KEY":"MZZE...Wobg","CHEF_AUTOMATE_ADMIN_PASSWORD":"zZZzDj2DLYXSZF
 \
    --key-pair "amazon-test" \
    --preferred-maintenance-window "Mon:08:00" \
    --preferred-backup-window "Sun:02:00" \
    --security-group-ids sg-b00000001 sg-b0000008 \
    --service-role-arn "arn:aws:iam::12345678912:role/service-role/aws-opsworks-cm-
service-role" \
    --subnet-ids subnet-300aaa00 \
    --tags [{\"Key\":\"Stage\",\"Value\":\"Production\"},{\"Key\":\"Department\",
\"Value\":\"Marketing\"}]
```

7. AWS OpsWorks for Chef Automate 创建新服务器大约需要 15 分钟。请勿关闭 `create-server` 命令的输出或关闭您的 Shell 会话，因为该输出可能包含不再显示的重要信息。要从 `create-server` 结果中获取密码和初学者工具包，请继续执行下一步。

   如果要在服务器上使用自定义域，请在 `create-server` 命令的输出中复制 `Endpoint` 属性的值。示例如下：

   ```
   "Endpoint": "automate-07-exampleexample.opsworks-cm.us-east-1.amazonaws.com"
   ```

8. 如果您选择为您 AWS OpsWorks for Chef Automate 生成密钥和密码，则可以使用 j q 等 JSON 处理器从`create-server`结果中提取可用格式的密钥和密码。安装 jq 后，您可以运行以下命令来提取关键密钥、Chef Automate 控制面板管理员密码和初学者工具包。如果您未在步骤 4 中提供自己的关键密钥和密码，请确保将提取的关键密钥和管理员密码保存在方便而安全的位置。

   ```
   #Get the Chef password:
   cat resp.json | jq -r '.Server.EngineAttributes[] | select(.Name ==
    "CHEF_AUTOMATE_ADMIN_PASSWORD") | .Value'

   #Get the Chef Pivotal Key:
   cat resp.json | jq -r '.Server.EngineAttributes[] | select(.Name ==
    "CHEF_AUTOMATE_PIVOTAL_KEY") | .Value'

   #Get the Chef Starter Kit:
   cat resp.json | jq -r '.Server.EngineAttributes[] | select(.Name ==
    "CHEF_STARTER_KIT") | .Value' | base64 -D > starterkit.zip
   ```

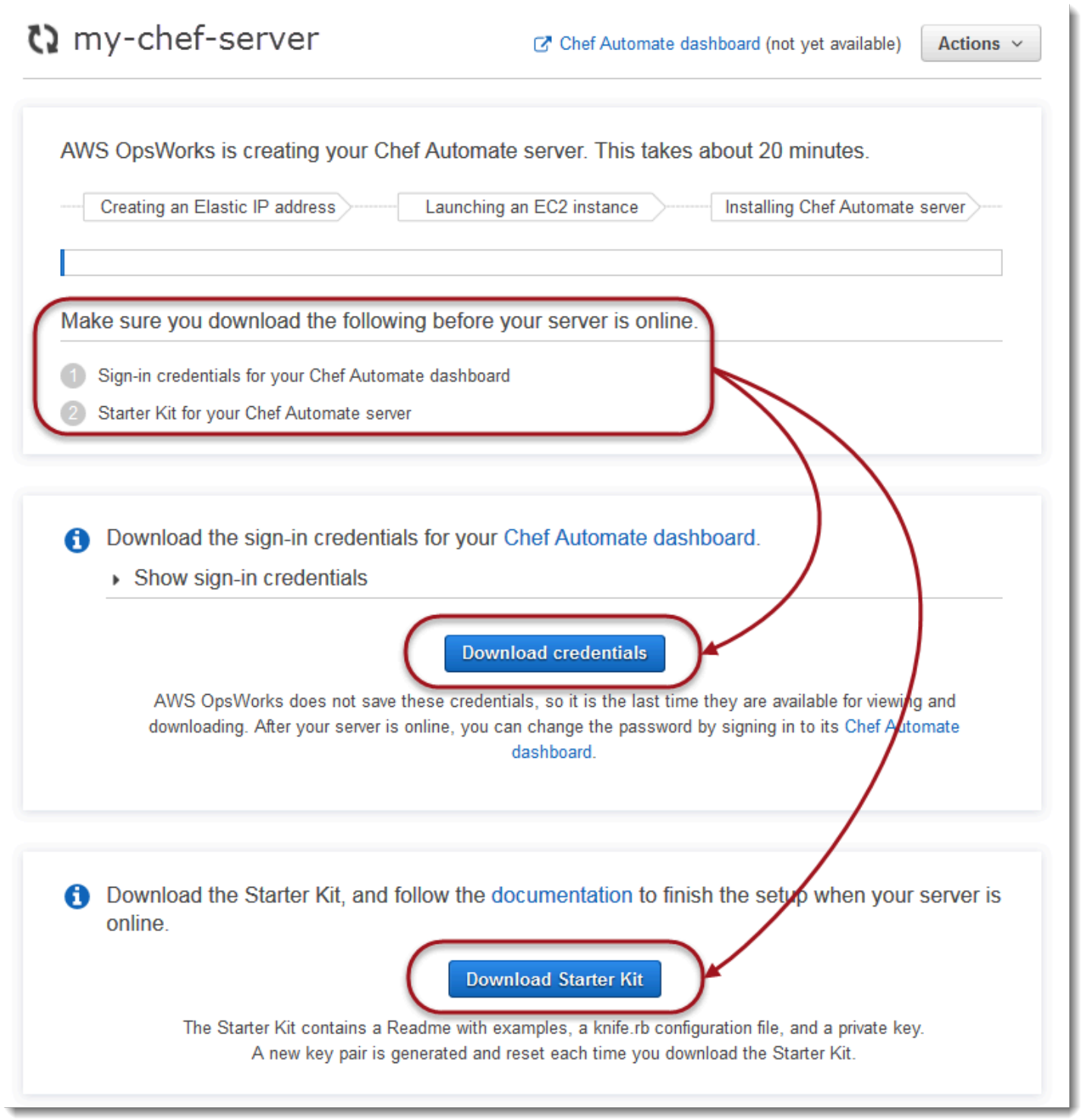
9. 或者，如果您没有从`create-server`命令结果中提取入门套件，则可以在 AWS OpsWorks for Chef Automate 控制台中从服务器的 “属性” 页面下载新的入门套件。下载新的初学者工具包将重置 Chef Automate 控制面板管理员密码。

10. 如果您不使用自定义域，请继续下一步。如果您在服务器上使用自定义域，请在企业的 DNS 管理工具中创建 CNAME 条目，将您的自定义域指向您在步骤 7 中复制的 AWS OpsWorks for Chef Automate 终端节点。在完成此步骤之前，您无法访问或登录具有自定义域的服务器。
11. 服务器创建过程完成后，请转到the section called “完成配置和上传说明书”。

使用初学者工具包配置 Chef Server

> **Important**
>
> AWS OpsWorks for Chef Automate 已于 2024 年 5 月 5 日停用，新客户和现有客户均已禁用。我们建议现有客户迁移到 Chef SaaS 或其他替代解决方案。如果您有任何疑问，可以通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

当 Chef Server 创建仍在进行中时，在 AWS OpsWorks for Chef Automate 控制台中打开其“Properties”页面。在您首次使用新 Chef Server 时，“Properties”页面提示您下载两个必需项。应在 Chef Server 联机之前下载这些项，新服务器联机之后，下载按钮将不可用。



- Chef 服务器的登录凭证。您将使用这些凭据登录 Chef Automate 仪表板，在那里您可以使用 Chef Automate 高级功能，例如工作流程和合规性扫描。 AWS OpsWorks 不会保存这些凭据；这是它们最后一次可供查看和下载。如果需要，您可以在登录之后更改随这些凭证提供的密码。

- 初学者工具包。初学者工具包包括 README 文件，带有示例、knife.rb 配置文件以及面向主用户或关键用户的私有密钥。每次下载初学者工具包时，将生成新密钥对，旧密钥对将会重置。

除了仅适用于新服务器的凭据外，Starter Kit .zip 文件还包括一个适用于任何 AWS OpsWorks for Chef Automate 服务器的 Chef 存储库的简单示例。在 Chef 存储库中，您可以存储用于通过 Chef 管理节点的说明书、角色、配置文件和其他项目。我们建议您将此存储库存储到版本控制系统中，例如 Git，并将其作为源代码处理。有关如何设置在 Git 中跟踪的 Chef 存储库的信息和演示示例，请参阅 Chef 文档中的关于 chef-repo。

先决条件

1. 当服务器创建仍在进行时，下载 Chef Server 的登录凭证并将其保存到安全而方便的位置。
2. 下载初学者工具包，将初学者工具包 .zip 文件解压缩到您的工作区目录中。请不要共享初学者工具包私有密钥。如果其他用户将管理 Chef Server，以后可以在 Chef Automate 控制面板中将其作为管理员添加。
3. 在要用于管理 Chef Server 和节点的计算机上下载并安装Chef Workstation （之前称为 Chef 开发工具包 或 Chef DK)。该knife实用程序是 Chef 工作站的一部分。有关说明，请参阅 Chef 网站上的安装 Chef Workstation。

浏览初学者工具包内容

此初学者工具包具有以下内容。

- cookbooks/-您创建的说明书目录。该 cookbooks/ 文件夹包含 opsworks-webserver 说明书，这是一本包装程序说明书，依赖于 Chef Supermarket 网站上的 nginx 说明书。 如果 cookbooks/ 目录中没有说明书依赖项，则Policyfile.rb默认为 Chef supermarket 作为辅助来源。
- Policyfile.rb-基于 Ruby 的策略文件，定义成为您节点策略的说明书、依赖项和属性。
- userdata.sh 和 userdata.ps1-在启动 Chef Automate 服务器后，您可以使用用户数据文件自动关联节点。 userdata.sh 用于引导基于 Linux 的节点，userdata.ps1 用于基于 Windows 的节点。
- Berksfile-如果您更喜欢使用 Berkshelf 和 berks 命令上传说明书及其依赖项，则可以使用此文件。在本演练中，我们使用 Policyfile.rb 和 Chef 命令上传说明书、依赖项和属性。
- README.md，一个基于 Markdown 的文件，它描述了如何使用初学者工具包首次设置 Chef Automate 服务器。
- .chef 是一个隐藏目录，其中包含 knife 配置文件 (knife.rb) 和秘密身份验证密钥文件 (.pem)。

- `.chef/knife.rb` - knife 配置文件 (`knife.rb`)。该`knife.rb`文件的配置使 Chef 的`knife`工具操作能够在 AWS OpsWorks for Chef Automate 服务器上运行。
- `.chef/ca_certs/opsworks-cm-ca-2020-root.pem` – 由 AWS OpsWorks提供的证书颁发机构 (CA) 签名的 SSL 私有密钥。此密钥允许服务器向您的服务器管理的节点上的 Chef Infra client 代理表明自己的身份。

设置 Chef 存储库

Chef 存储库包含多个目录。初学者工具包中的每个目录包含一个 README 文件，其中介绍了目录的用途，以及如何使用它通过 Chef 来管理系统。有两种方法可以在 Chef 服务器上安装说明书：运行 `knife` 命令，或者运行 Chef 命令将策略文件 (`Policyfile.rb`) 上传到服务器，下载并安装指定说明书。此演练使用 Chef 命令和 `Policyfile.rb` 在服务器上安装说明书。

1. 在本地计算机上创建目录来存储说明书，例如 `chef-repo`。将食谱、角色和其他文件添加到此存储库后，我们建议您将其上传或存储在安全的版本控制系统（例如 CodeCommit Git 或 Amazon S3）中。
2. 在 `chef-repo` 目录中，创建以下目录：
   - `cookbooks/`-存放说明书。
   - `roles/` - 以 `.rb` 或 `.json` 格式存储角色。
   - `environments/` - 以 `.rb` 或 `.json` 格式存储环境。

使用 Policyfile.rb 从远程源获取说明书

在本节中，编辑 `Policyfile.rb` 以指定说明书，然后运行 Chef 命令将文件上传到服务器并安装说明书。

1. 在您的初学者工具包中查看 `Policyfile.rb`。默认情况下，`Policyfile.rb` 包括 `opsworks-webserver` 包装程序说明书，该说明书依赖于 Chef Supermarket 网站上提供的`nginx`说明书。`nginx` 说明书在托管节点上安装和配置 Web 服务器。还指定了所需的 `chef-client` 说明书，该说明书在托管节点上安装 Chef Infra 客户端代理。

   `Policyfile.rb` 还指向可选 Chef Audit 说明书，您可以用它来在节点上设置合规性扫描。有关为托管节点设置合规性扫描和获取合规性结果的更多信息，请参阅：中的合规性扫描 AWS OpsWorks for Chef Automate。如果您不想立即配置合规性扫描和审计，请从 `run_list` 部分中删除 `'audit'`，并且不要在文件末尾指定 `audit` 说明书属性。

```
# Policyfile.rb - Describe how you want Chef to build your system.
#
# For more information about the Policyfile feature, visit

# https://docs.chef.io/policyfile.html


# A name that describes what the system you're building with Chef does.

name 'opsworks-demo-webserver'

# The cookbooks directory is the preferred source for external cookbooks

default_source :chef_repo, "cookbooks/" do |s|

  s.preferred_for "nginx", "windows", "chef-client", "yum-epel", "seven_zip",
                  "build-essential", "mingw", "ohai", "audit", "logrotate", "cron"

end
# Alternative source
default_source :supermarket


# run_list: chef-client runs these recipes in the order specified.

run_list  'chef-client',
          'opsworks-webserver',
          'audit'
# add 'ssh-hardening' to your runlist to fix compliance issues detected by the ssh-
baseline profile

# Specify a custom source for a single cookbook:

cookbook 'opsworks-webserver', path: 'cookbooks/opsworks-webserver'


# Policyfile defined attributes

# Define audit cookbook attributes
default["opsworks-demo"]["audit"]["reporter"] = "chef-server-automate"
default["opsworks-demo"]["audit"]["profiles"] = [
  {
    "name": "DevSec SSH Baseline",
```

```
    "compliance": "admin/ssh-baseline"
  }
]
```

如果您现在只想配置 nginx Web 服务器，以下为一个没有 audit 说明书和属性的 Policyfile.rb 示例。

```
# Policyfile.rb - Describe how you want Chef to build your system.
#
# For more information on the Policyfile feature, visit
# https://docs.chef.io/policyfile.html

# A name that describes what the system you're building with Chef does.
name 'opsworks-demo-webserver'

# Where to find external cookbooks:
default_source :supermarket

# run_list: chef-client will run these recipes in the order specified.
run_list  'chef-client',
          'opsworks-webserver'

# Specify a custom source for a single cookbook:
cookbook 'opsworks-webserver', path: 'cookbooks/opsworks-webserver'
```

如果您对 Policyfile.rb 进行了更改，请务必保存该文件。

2. 下载并安装 Policyfile.rb 中定义的说明书。

```
chef install
```

所有说明书的版本均记录在说明书的 metadata.rb 文件中。每次更改说明书后，必须在 metadata.rb 中提高该说明书的版本。

3. 如果您选择配置合规性扫描，并将 audit 说明书信息保留在策略文件中，请推送策略 opsworks-demo 到您的服务器。

```
chef push opsworks-demo
```

4. 如果您完成了步骤 3，请验证您的策略是否已安装。运行以下命令。

```
chef show-policy
```

结果应与以下内容类似：

```
opsworks-demo-webserver
=======================
* opsworks-demo:  ec0fe46314
```

5. 现在，您可以向您的 Chef Automate 服务器添加或引导节点了。您可以按照 在中自动添加节点 AWS OpsWorks for Chef Automate 中的步骤自动关联节点，或按照 单独添加节点 中的步骤，每次添加一个节点。

（替代）使用 Berkshelf 从远程源获取说明书

Berkshelf 是一个用于管理说明书及其依赖项的工具。相较于 `Policyfile.rb`，如果您更喜欢使用 Berkshelf 将说明书安装到本地存储中，请使用本节中的步骤而不是上一节的步骤。您可以指定要将哪些说明书和版本用于您的 Chef Server 并上传这些说明书和版本。初学者工具包中包含一个名为 `Berksfile` 的文件，其中您可以使用它列出您的说明书。

1. 要开始使用，请将 `chef-client` 说明书添加到随附的 Berksfile 中。`chef-client` 说明书配置连接到 Chef Automate 服务器的每个节点上的 Chef Infra 客户端代理软件。要了解有关此说明书的更多信息，请参阅 Chef 超市中的 Chef Client 说明书。
2. 使用文本编辑器，将其他说明书附加到在其中安装 Web 服务器应用程序的 Berksfile；例如，`apache2` 说明书，该说明书安装 Apache Web 服务器应用程序。您的 Berksfile 应类似于以下内容。

```
source 'https://supermarket.chef.io'
cookbook 'chef-client'
cookbook 'apache2'
```

3. 在本地计算机上下载并安装说明书。

```
berks install
```

4. 将说明书上传到 Chef Server。

   在 Linux 上，运行以下命令。

```
SSL_CERT_FILE='.chef/ca_certs/opsworks-cm-ca-2020-root.pem' berks upload
```

在 Windows 上，在 PowerShell 会话中运行以下 Chef Workstation 命令。在运行命令之前，请务必将执行策略设置 PowerShell 为`RemoteSigned`。添加`chef shell-init`以使 Chef Workstation 实用程序命令可供使用 PowerShell。

```
$env:SSL_CERT_FILE="ca_certs\opsworks-cm-ca-2020-root.pem"
chef shell-init berks upload
Remove-Item Env:\SSL_CERT_FILE
```

5. 通过显示当前在 Chef Automate 服务器上可用的说明书列表，验证说明书的安装。您可以使用以下 `knife` 命令进行这项操作：

   您已准备好添加要使用 AWS OpsWorks for Chef Automate 服务器管理的节点。

   ```
   knife cookbook list
   ```

（可选）配置 **knife** 使用自定义域

如果 Chef Automate 服务器使用自定义域，则可能需要添加用于签署服务器证书链的根 CA 的 PEM 证书，或者添加服务器 PEM 证书（如果证书是自签名的）。`ca_certs` 是 `chef/` 中包含由 Chef `knife` 实用程序信任的证书颁发机构 (CA) 的子目录。

如果您未使用自定义域，或者您的自定义证书由操作系统信任的根 CA 签名，则可以跳过此部分。否则，配置 `knife` 以信任您的 Chef Automate 服务器 SSL 证书，如以下步骤所述。

1. 运行以下命令。

   ```
   knife ssl check
   ```

   如果结果类似于以下内容，请跳过此过程的其余部分，然后继续 [为 Chef 服务器添加节点以进行管理](#)。

   ```
   Connecting to host my-chef-automate-server.my-corp.com:443
            Successfully verified certificates from 'my-chef-automate-server.my-
   corp.com'
   ```

如果您收到类似于以下内容的错误消息，请继续执行下一步。

```
Connecting to host my-chef-automate-server.my-corp.com:443
          ERROR: The SSL certificate of my-chef-automate-server.my-corp.com could
 not be verified.
          ...
```

2. 运行 `knife ssl fetch` 以信任 AWS OpsWorks for Chef Automate 服务器的证书。或者，您可以手动将服务器的根 CA PEM 格式的证书复制到作为 `knife ssl check` 输出中 `trusted_certs_dir` 的值的目录。默认情况下，此目录位于初学者工具包的 `.chef/ca_certs/` 中。输出应与以下内容类似：

```
WARNING: Certificates from my-chef-automate-server.my-corp.com will be fetched and
 placed in your trusted_cert
          directory (/Users/username/starterkit/.chef/../.chef/ca_certs).

          Knife has no means to verify these are the correct certificates. You
 should
          verify the authenticity of these certificates after downloading.

          Adding certificate for my-chef-automate-server in /Users/users/
starterkit/.chef/../.chef/ca_certs/servv-aqtswxu20swzkjgz.crt
          Adding certificate for MyCorp_Root_CA in /Users/users/
starterkit/.chef/../.chef/ca_certs/MyCorp_Root_CA.crt
```

3. 再次运行 `knife ssl check`。输出应与以下内容类似：

```
Connecting to host my-chef-automate-server.my-corp.com:443
          Successfully verified certificates from 'my-chef-automate-server.my-
corp.com'
```

您已准备好 `knife` 与您的 Chef Automate 服务器结合使用。

为 Chef 服务器添加节点以进行管理

> **Important**
>
> AWS OpsWorks for Chef Automate 已于 2024 年 5 月 5 日停用，新客户和现有客户均已禁用。我们建议现有客户迁移到 Chef SaaS 或其他替代解决方案。如果您有任何疑问，可以通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

`chef-client` 代理在称为节点 的物理或虚拟计算机上运行 Chef 配方，这些节点与服务器关联。只要节点运行的是支持的操作系统，您就可以将本地计算机或实例连接到 Chef 服务器进行管理。将节点注册到 Chef 服务器时，会在这些节点上安装 `chef-client` 代理软件。

您可以使用以下方法添加节点：

- 通过运行添加或引导 EC2 实例的 `knife` 命令来单独添加备注，以便 Chef 服务器可以对其进行管理。有关更多信息，请参阅 单独添加节点。
- 通过使用脚本来自动添加节点以便执行节点与 Chef Server 的自动关联。本初学者工具包中的代码显示如何使用无人参与的方法自动添加节点。有关更多信息，请参阅在中自动添加节点 AWS OpsWorks for Chef Automate。

主题

单独添加节点

> **Important**
>
> AWS OpsWorks for Chef Automate 已于 2024 年 5 月 5 日停用，新客户和现有客户均已禁用。我们建议现有客户迁移到 Chef SaaS 或其他替代解决方案。如果您有任何疑问，可以通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

此节介绍如何运行 `knife` 命令来添加或 引导 EC2 实例，以便 Chef 服务器可以管理它。

在与 AWS OpsWorks for Chef Automate 服务器关联的节点上，chef-client 的最低支持版本为 13.x。我们建议运行最新、最稳定的chef-client版本。

主题

（可选）指定 Chef Automate 服务器根 CA 的 URL

如果您的服务器使用自定义域和证书，则可能需要使用可用于获取服务器的根 CA PEM 格式证书的公共 URL 编辑 userdata 脚本中的 ROOT_CA_URL 变量。以下 AWS CLI 命令将您的根 CA 上传到 Amazon S3 存储桶，并生成可使用一小时的预签名 URL。

1. 将根 CA PEM 格式的证书上传到 S3。

   ```
   aws s3 cp ROOT_CA_PEM_FILE_PATH s3://bucket_name/
   ```

2. 生成一个预签名 URL，您可以使用该 URL 一小时（在本示例中为 3600 秒）来下载根 CA。

   ```
   aws s3 presign s3://bucket_name/ROOT_CA_PEM_FILE_NAME --expires-in 3600
   ```

3. 使用预签名 URL 的值编辑 userdata 脚本中的 ROOT_CA_URL 变量。

受支持的操作系统

有关当前支持的节点操作系统列表，请参阅 Chef 网站。

使用 Knife 添加节点

knife-ec2 插件包括在 Chef Workstation 中。如果您更熟悉 knife-ec2，也可以使用它取代 knife bootstrap 来预配置和引导新 EC2 实例。否则，启动新的 EC2 实例，然后按照本节中的步骤操作。

添加要管理的节点

1. 运行以下 knife bootstrap 命令：此命令在您的 Chef 服务器将管理的节点上引导 EC2 实例。请注意，您需要指示 Chef 服务器从您安装在 nginx 上的 the section called “使用 Policyfile.rb 从远程源获取说明书” 说明书运行配方。有关通过运行 knife bootstrap 命令来添加节点的更多信息，请参阅 Chef 文档中的引导节点。

下表显示了此步骤中 knife 命令的节点操作系统的有效用户名。如果 root 或 ec2-user 均无法使用，请与您的 AMI 供应商核实。有关连接到基于 Linux 的实例的更多信息，请参阅 AWS 文档中的使用 SSH 连接到 Linux 实例。

节点操作系统中用户名的有效值

操作系统	有效用户名
Amazon Linux	ec2-user
Red Hat Enterprise Linux 5	root 或 ec2-user
Ubuntu	ubuntu
Fedora	fedora 或 ec2-user
SUSE Linux	root 或 ec2-user

```
knife bootstrap INSTANCE_IP_ADDRESS -N INSTANCE_NAME -x USER_NAME --sudo --run-list
 "recipe[nginx]"
```

2. 运行以下命令，将 *INSTANCE_NAME* 替换为您刚刚添加的实例名称，验证新节点已添加。

```
knife client show INSTANCE_NAME
knife node show INSTANCE_NAME
```

在中自动添加节点 AWS OpsWorks for Chef Automate

> **Important**
>
> AWS OpsWorks for Chef Automate 已于 2024 年 5 月 5 日停用，新客户和现有客户均已禁用。我们建议现有客户迁移到 Chef SaaS 或其他替代解决方案。如果您有任何疑问，可以通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

本主题介绍如何将 Amazon Elastic Compute Cloud (Amazon EC2) 节点自动添加到 Chef 服务器。本初学者工具包中的代码显示如何使用无人参与的方法自动添加节点。对于如何在无人参与的情况下

(或自动) 将新的节点关联起来，我们推荐的方法是配置 Chef Client 说明书。您可以使用初学者工具包中的 userdata 脚本，并更改 userdata 脚本的 run_list 部分，或使用要应用于节点的说明书更改 Policyfile.rb。在您运行 chef-client 代理之前，请将 Chef Client 说明书安装到您的 Chef 服务器，然后使用某种角色（例如 HTTPD 角色）以服务模式安装 chef-client 代理，如以下示例命令所示。

```
chef-client -r "chef-client,role[httpd]"
```

要与 Chef 服务器通信，chef-client 代理软件必须拥有客户端节点的公有密钥的访问权。您可以在 Amazon EC2 中生成公私密钥对，然后将带有节点名称的公钥传递给 AWS OpsWorks associate-node API 调用。初学者工具包中包含的脚本为您收集您的组织名称、服务器名称和服务器端点。这可确保节点与 Chef 服务器相关联，运行于节点上的 chef-client 代理软件在与私有密钥匹配后，即可与服务器通信。

在与 AWS OpsWorks for Chef Automate 服务器关联的节点上，chef-client 的最低支持版本为 13.x。我们建议运行最新、最稳定的chef-client版本。

有关如何取消关联节点的信息，请参阅取消节点与服务器的关联 AWS OpsWorks for Chef Automate本指南和 AWS OpsWorks for Chef Automate API 文档disassociate-node中的。

主题

受支持的操作系统

有关当前支持的节点操作系统列表，请参阅 Chef 网站。

第 1 步：创建一个 IAM 角色，以用作您的实例配置文件

创建一个 AWS Identity and Access Management (IAM) 角色用作您的 EC2 实例配置文件，并将以下策略附加到该 IAM 角色。该策略准许 AWS OpsWorks for Chef Automate (opsworks-cm) API 在节点注册期间与 EC2 实例通信。有关实例配置文件的更多信息，请参阅 Amazon EC2 文档中的使用实例配置文件。有关如何创建 IAM 角色的信息，请参阅 Amazon EC2 文档中的在控制台中创建 IAM 角色。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "opsworks-cm:AssociateNode",
                "opsworks-cm:DescribeNodeAssociationStatus",
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

AWS OpsWorks 提供了一个 AWS CloudFormation 模板，您可以使用该模板通过上述策略声明创建 IAM 角色。以下 AWS CLI 命令使用此模板为您创建实例配置文件角色。如果您想在默认区域中创建新 AWS CloudFormation 堆栈，则可以省略该`--region`参数。

```
aws cloudformation --region region ID create-stack --stack-
name myChefAutomateinstanceprofile --template-url https://s3.amazonaws.com/opsworks-
cm-us-east-1-prod-default-assets/misc/opsworks-cm-nodes-roles.yaml --capabilities
 CAPABILITY_IAM
```

第 2 步：安装 Chef Client 说明书

如果您尚未执行上述操作，请按照（替代）使用 Berkshelf 从远程源获取说明书中的步骤操作，以确保您的 `Policyfile.rb` 文件引用 Chef Client 说明书并安装该说明书。

第 3 步：使用自动化关联脚本创建实例

1. 要创建 EC2 实例，您可以将该`userdata`脚本从入门工具包复制到 EC2 实例说明`userdata`部分、Amazon EC2 Auto Scaling 组启动配置或 AWS CloudFormation 模板中。有关将脚本添加将到用户数据的更多信息，请参阅 Amazon EC2 文档中的启动时对您的 Linux 实例运行命令。

   此脚本运行 `opsworks-cm` API `associate-node` 命令，以将一个新的节点与您的 Chef 服务器相关联。

   默认情况下，新注册的节点的名称即是实例 ID，但您可以通过修改 `userdata` 脚本中的 `NODE_NAME` 变量的值来更改这个名称。由于当前无法在 Chef 控制台 UI 上更改组织名称，请将 `CHEF_AUTOMATE_ORGANIZATION` 设置为 `default`。

2. 按照 EC2 文档中启动实例的说明操作，修改见此处。在 EC2 实例启动向导中，选择 Amazon Linux AMI。
3. 在 Configure Instance Details 页面上，将您在 第 1 步：创建一个 IAM 角色，以用作您的实例配置文件中创建的角色选为您的 IAM 角色。
4. 在 Advanced Details 区域中，上传您在此过程较早时创建的 `userdata.sh` 脚本。
5. 无需在 Add Storage 页面上进行更改。转到 Add Tags。
6. 在 Configure Security Group 页上，选择 Add Rule，然后选择类型 HTTP，在此例中为 Apache Web 服务器打开端口 443 和 80。
7. 选择 Review and Launch，然后选择 Launch。当您的新节点启动时，它会应用您在 `RUN_LIST` 参数中指定的配方所指定的配置。
8. 可选：如果您已将 `nginx` 说明书添加到运行列表中，当您打开与您的新节点的公有 DNS 相链接的网页时，您应看到由 nginx Web 服务器托管的网站。

自动重复运行 **`chef-client`** 的其他方法

尽管更难实现且不建议这样做，但您可以仅将本主题中的脚本作为独立实例用户数据的一部分运行，也可以使用 AWS CloudFormation 模板将其添加到新的实例用户数据中，将`cron`作业配置为定期运行脚本，或者在服务`chef-client`中运行。不过，我们推荐使用 Chef Client 说明书方法，因为其他自动化技术存在一些缺点：

要查看您可以为 `chef-client` 提供的完整参数列表，请参阅 Chef 文档。

相关主题

以下 AWS 博客文章提供了有关使用 Auto Scaling 群组或在多个账户中自动将节点与 Chef Automate 服务器关联的更多信息。

- 使用 AWS fo OpsWorks r Chef Automate 通过 Auto Scaling 管理 EC2 实例
- OpsWorks for Chef Automate — 自动引导不同账户中的节点

登录 Chef Automate 控制面板

> **Important**
>
> AWS OpsWorks for Chef Automate 已于 2024 年 5 月 5 日停用，新客户和现有客户均已禁用。我们建议现有客户迁移到 Chef SaaS 或其他替代解决方案。如果您有任何疑问，可以通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

在您从 Chef 服务器的“Properties”页面下载了登录凭证并且服务器联机之后，登录 Chef Automate 控制面板。在本演练中，我们将指导您先上传一个说明书，然后添加至少一个节点来管理。这样您就可以在控制面板中查看说明书和节点的信息。

当你尝试连接到仪表板网页时，证书警告会出现在浏览器中，直到你在用于管理 Chef 服务器的客户端计算机上安装了 AWS OpsWorks特定的、由 CA 签名的 SSL 证书。如果您不希望在继续到控制面板页面之前看到警告，请先安装 SSL 证书，然后登录。

安装 AWS OpsWorks SSL 证书

- 选择与您的系统匹配的证书。
  - 对于基于 Linux 或 macOS 的系统，请从以下亚马逊 S3 位置下载文件扩展名为 PEM 的文件：https://s3.amazonaws.com/ opsworks-cm-us-east--/misc/-2016-prod-default-assets root.pem。opsworks-cm-ca

    > **Note**
    >
    > 此外，请从以下位置下载更新的PEM文件：https://s3.amazonaws.com/opsworks-cm-us-east-1-prod-default-assets/misc/opsworks-cm-ca-2020-root.pem。由于当前 AWS OpsWorks for Chef Automate 正在续订其根证书，因此您必须同时信任新旧证书。

    有关如何在 macOS 上管理 SSL 证书的更多信息，请参阅 Apple Support 网站的 在 Mac 上的 Keychain Access 中获取有关证书的信息。
  - 对于基于 Windows 的系统，请从以下亚马逊 S3 位置下载文件扩展名为 P7B 的文件：https://s3.amazonaws.com/ opsworks-cm-us-east--/misc/- 2016-root.p7b。prod-default-assets opsworks-cm-ca

> **Note**
>
> 此外，请从以下位置下载更新的 P7B 文件：https://s3.amazonaws.com/opsworks-cm-us-east-1-prod-default-assets/misc/opsworks-cm-ca-2020-root.p7b。由于当前 AWS OpsWorks for Chef Automate 正在续订其根证书，因此您必须同时信任新旧证书。

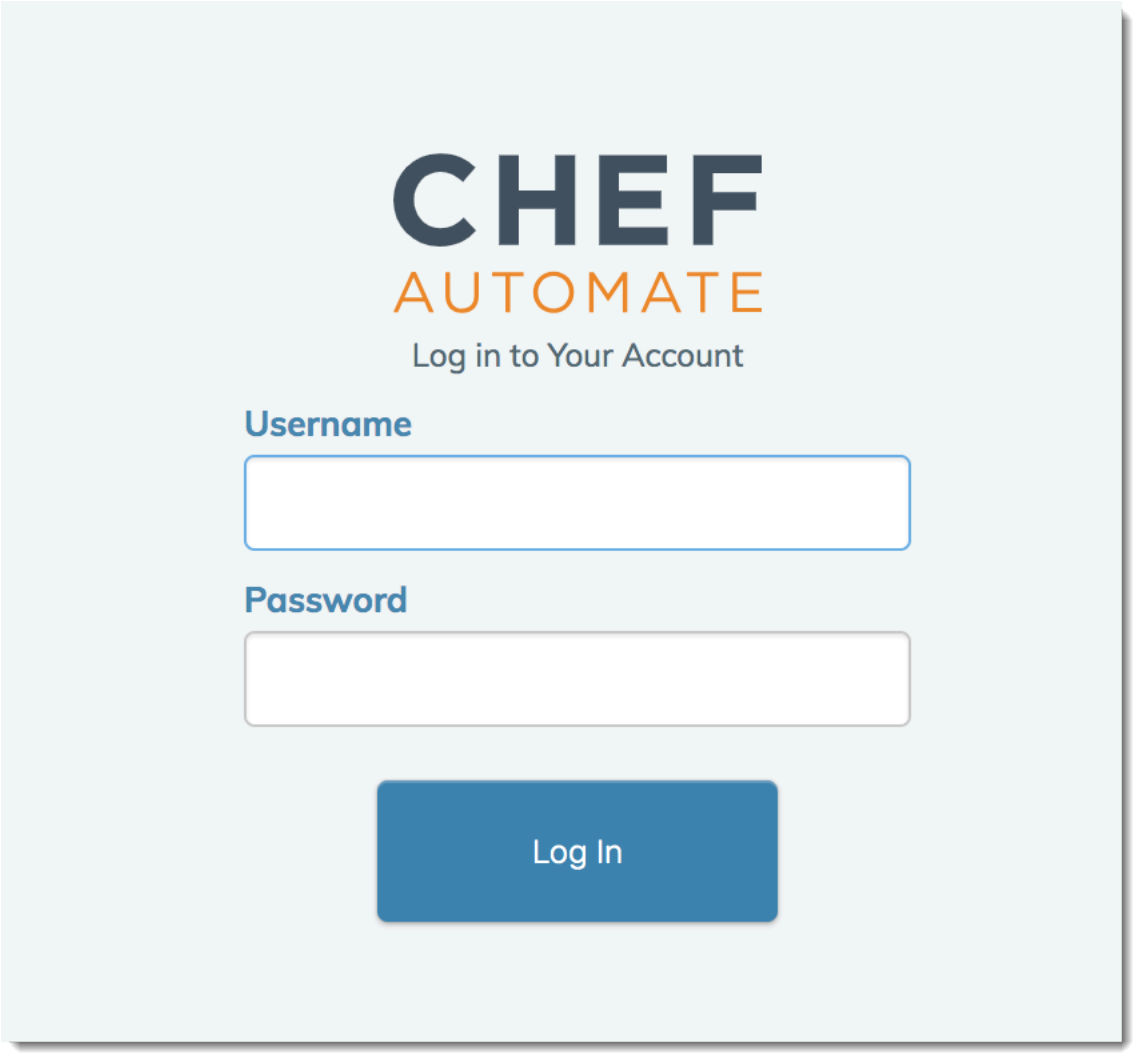
有关如何在 Windows 上安装 SSL 证书的更多信息，请参阅在 Microsoft 上管理可信根证书 TechNet。

在安装了客户端 SSL 证书之后，您可以登录到 Chef Automate 控制面板而不会看到警告消息。

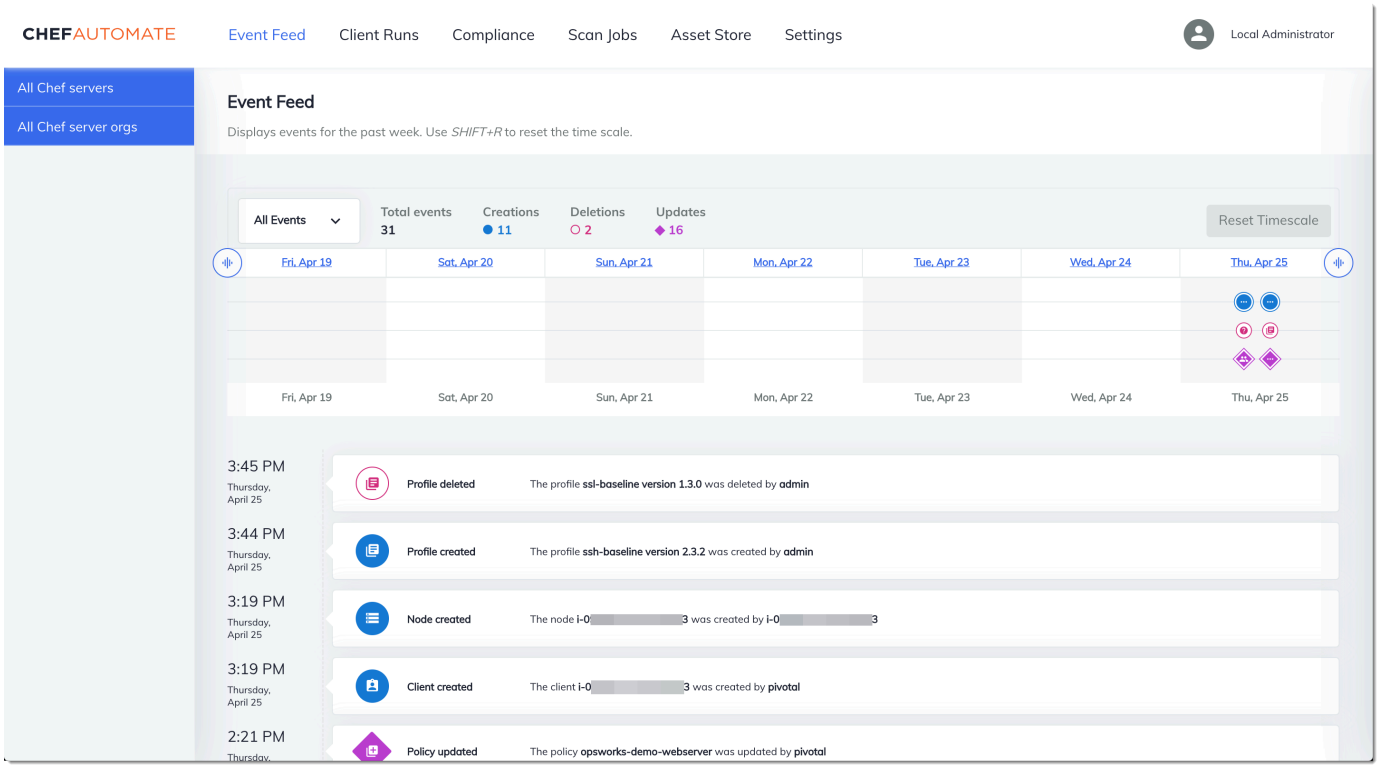
> **Note**
>
> Ubuntu 和 Linux Mint 操作系统上的 Google Chrome 用户在登录时可能会遇到问题。我们建议您在这些操作系统上使用 Mozilla Firefox 或其他浏览器来登录和使用 Chef Automate 控制面板。在 Windows 或 MacOS 上使用 Google Chrome 未发现问题。

登录 Chef Automate 控制面板

1. 解压缩并打开您在 先决条件 中下载的 Chef Automate 凭证。您需要这些凭证才能登录。
2. 打开您的 Chef 服务器的 Properties (属性) 页面。
3. 在 Properties (属性) 页面的右上角，选择 Open Chef Automate dashboard (打开 Chef Automate 控制面板)。
4. 使用步骤 1 中的凭证登录。



5. 在 Chef Automate 控制面板中，您可以查看有关已引导节点的详细信息、说明书运行进度和事件、节点的合规性水平等等。有关 Chef Automate 控制面板的功能以及用法的更多信息，请参阅 Chef Automate 文档。



Note

有关如何更改登录 Chef Automate 控制面板时使用的密码的信息，请参阅 重置 Chef Automate 控制面板的凭证。

使用创建 AWS OpsWorks for Chef Automate 服务器 AWS CloudFormation

Important

AWS OpsWorks for Chef Automate 已于 2024 年 5 月 5 日停用，新客户和现有客户均已禁用。我们建议现有客户迁移到 Chef SaaS 或其他替代解决方案。如果您有任何疑问，可以通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

AWS OpsWorks for Chef Automate 允许你在中运行 Chef Automat e 服务器 AWS。您可以在大约 15 分钟内预置一个 Chef Automate 服务器。

从 2021 年 5 月 3 日起，将一些 Chef Automate 服务器属性 AWS OpsWorks for Chef Automate 存储在中 AWS Secrets Manager。有关更多信息，请参阅 与 AWS Secrets Manager 集成。

以下演练可帮助您 AWS OpsWorks for Chef Automate 通过在中创建堆栈来创建服务器。 AWS CloudFormation

主题

先决条件

在创建新的 Chef Automate 服务器之前，请在 AWS OpsWorks for Chef Automate 外部创建一些用于访问和管理您的 Chef 服务器的资源。有关更多信息，请参阅本指南的“入门”部分的先决条件 。

查看《AWS CloudFormation 用户指南模板参考》的 “OpsWorks-CM” 部分，了解用于创建服务器的 AWS CloudFormation 模板中支持的值和必填值。

如果要创建使用自定义域的服务器，则需要自定义域、证书和私有密钥。您必须在 AWS CloudFormation 模板中为所有这三个参数指定值。有关、和CustomPrivateKey参数要求的更多信息CustomDomainCustomCertificate，请参阅 AWS OpsWorks CM API 参考CreateServer中的。

为 CHEF_AUTOMATE_ADMIN_PASSWORD 引擎属性创建一个密码值。该密码的最小长度为 8 个字符，最大长度为 32 个字符。该密码可以包含字母、数字和特殊字符 (!/@#$%^+=_)。该密码必须至少包含一个小写字母、一个大写字母、一个数字和一个特殊字符。您可以在 AWS CloudFormation 模板中指定此密码，或者在创建堆栈时指定为CHEF_AUTOMATE_ADMIN_PASSWORD参数的值。

在开始在中创建 Chef Automate 服务器之前，请生成一个 base64 编码的 RSA 密钥对。 AWS CloudFormation两者的公钥是 CreateServerAPI 中CHEF_AUTOMATE_PIVOTAL_KEY特定于 Chef EngineAttributes的值。此密钥作为 AWS CloudFormation 控制台中参数的值提供，或者在中的create-stack命令中提供 AWS CLI。要生成该密钥，建议采用以下方法。

- 在基于 Linux 的计算机上，您可以通过运行以下 OpenSSL 命令来生成该密钥。

  ```
  openssl genrsa -out pivotal_key_file_name.pem 2048
  ```

  然后，将该对的 RSA 公有密钥部分导出到文件。成为 CHEF_AUTOMATE_PIVOTAL_KEY 值的公有密钥。

```
openssl rsa -in pivotal_key_file_name.pem -pubout -out public.pem -outform PEM
```

- 在基于 Windows 的计算机上，您可以使用 PuTTYgen 实用工具生成 base64 编码的 RSA 密钥对。有关更多信息，请参阅 SSH.com 上的 PuTTYgen - Windows 上 PuTTY 的密钥生成器。

在 AWS CloudFormation中创建 Chef Automate 服务器

本节介绍如何使用 AWS CloudFormation 模板来构建用于创建 AWS OpsWorks for Chef Automate 服务器的堆栈。您可以使用 AWS CloudFormation 控制台或 AWS CLI. 您可以使用示例 AWS CloudFormation 模板来构建 AWS OpsWorks for Chef Automate 服务器堆栈。请务必使用您自己的服务器名称、IAM 角色、实例配置文件、服务器描述、备份保留计数、维护选项和可选标签来更新示例模板。如果您的服务器将使用自定义域，则必须在 AWS CloudFormation 模板中为`CustomDomainCustomCertificate`、和`CustomPrivateKey`参数指定值。可以在 AWS CloudFormation 模板中指定`CHEF_AUTOMATE_ADMIN_PASSWORD`和`CHEF_AUTOMATE_PIVOTAL_KEY`引擎属性及其值，或者仅提供属性，然后在 “ AWS CloudFormation 创建堆栈” 向导或create-stack命令中为属性指定值。有关这些属性的更多信息，请参阅本指南的“入门”部分的the section called “在中创建一个 Chef Automate 服务器 AWS Management Console”。

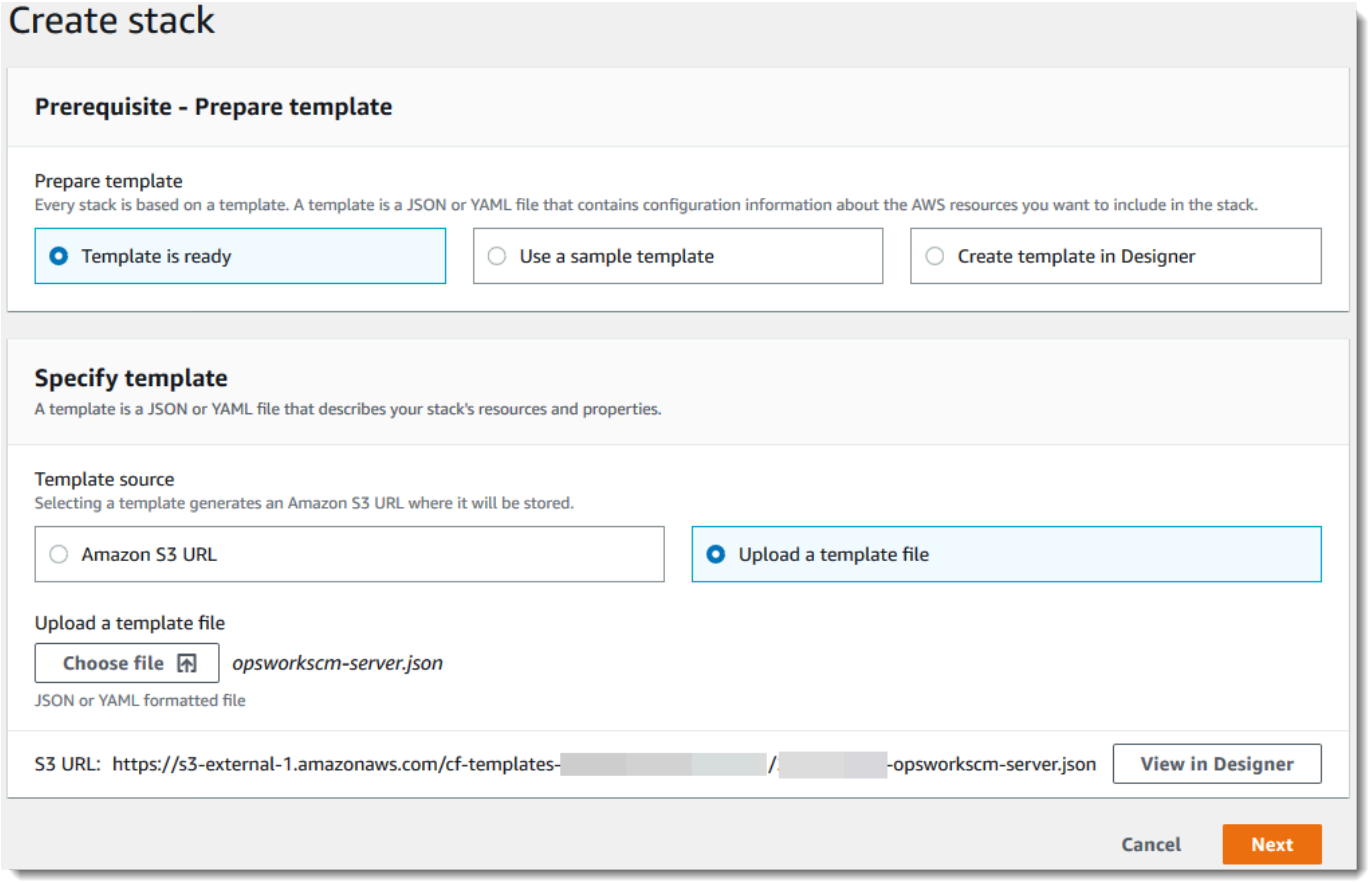
主题

- 使用 AWS CloudFormation 创建 Chef Automate 服务器（控制台）
- 使用 AWS CloudFormation 创建 Chef Automate 服务器 (CLI)

使用 AWS CloudFormation 创建 Chef Automate 服务器（控制台）

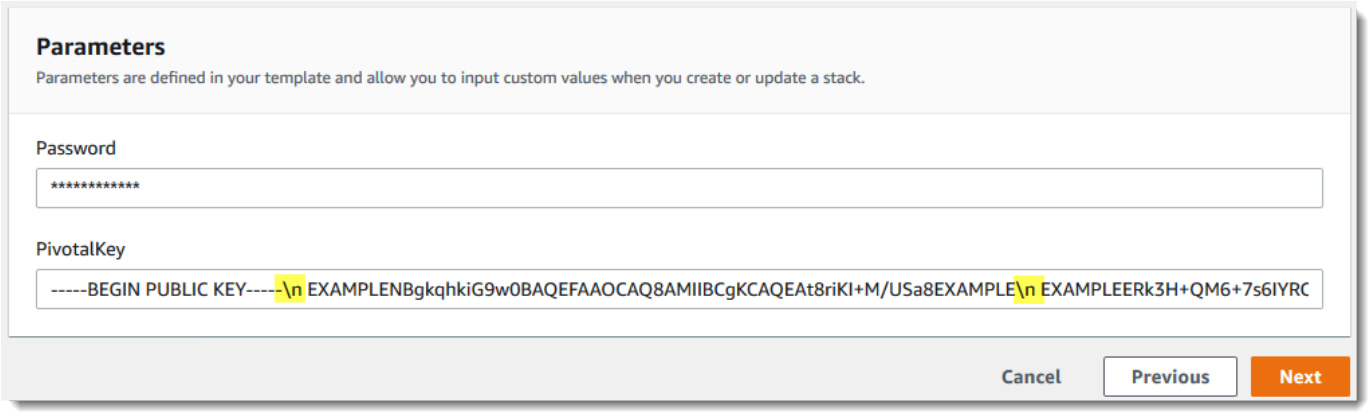
1. 登录 AWS Management Console 并打开 AWS CloudFormation 控制台，网址为 https://console.aws.amazon.com/cloudformation。
2. 在 AWS CloudFormation 主页上，选择创建堆栈。
3. 在 “先决条件-准备模板” 中，如果您使用的是示例 AWS CloudFormation 模板，请选择 “模板已准备就绪”。
4. 在 Specify template (指定模板) 中，选择模板的源。在本演练中，选择上传模板文件，然后上传用于创建 Chef Automate 服务器的 AWS CloudFormation 模板。浏览查找您的模板文件，然后选择 Next (下一步)。

AWS CloudFormation 模板可以采用 YAML 或 JSON 格式。有一个示例 AWS CloudFormation 模板可供您使用；请务必用自己的示例值替换示例值。您可以使用 AWS CloudFormation 模板设计器来构建新模板或验证现有模板。有关如何执行此操作的更多信息，请参阅《AWS CloudFormation 用户指南》中的AWS CloudFormation Designer 界面概述。



5. 在 指定详细信息 页面上，输入您的堆栈的名称。这将不会与您的服务器的名称相同，它仅仅是一个堆栈名称。在 Parameters (参数) 区域中，粘贴您在the section called “先决条件 ”中创建的值。在 Password (密码) 中输入密码。

   将 RSA 密钥文件的内容粘贴到中。PivotalKey在 AWS CloudFormation 控制台中，您必须在关键键值的每行末尾添加换行符 (**\n**)，如以下屏幕截图所示。选择下一步。



6. 在 配置堆栈选项 页面上，您可以向您使用堆栈创建的服务器添加标签，然后选择用于创建资源的 IAM 角色（如果您尚未在您的模板中指定要使用的 IAM 角色）。在指定选项之后，选择 Next (下一步)。有关高级选项（例如回滚触发器）的更多信息，请参阅《AWS CloudFormation 用户指南》中的设置 AWS CloudFormation 堆栈选项。

7. 在审核页面上，审核您的选择。在准备好创建服务器堆栈时，选择 Create stack (创建堆栈)。

   在等待创建堆栈时， AWS CloudFormation 请查看堆栈创建状态。如果堆栈创建失败，请查看控制台中显示的错误消息，以帮助您解决问题。有关对 AWS CloudFormation 堆栈中的错误进行故障排除的更多信息，请参阅《AWS CloudFormation 用户指南》中的排查错误。

   服务器创建完成之后，您的 Chef Automate 服务器将会出现在 AWS OpsWorks for Chef Automate 主页上，其状态为 online (在线)。从服务器的“属性”页面生成一个新的初学者工具包和 Chef Automate 控制面板凭证。服务器联机之后，Chef Automate 控制面板在服务器的域上可用，位于以下格式的 URL 上：`https://`*`your_server_name`*`-`*`randomID`*`.`*`region`*`.opsworks-cm.io`。

   > **Note**
   >
   > 如果您为服务器指定了自定义域、证书和私钥，请在企业的 DNS 管理工具中创建一个 CNAME 条目，该条目将您的自定义域映射到为服务器 AWS OpsWorks for Chef Automate 自动生成的终端节点。在将生成的终端节点映射到自定义域值之前，您无法管理服务器或连接到服务器的 Chef Automate 控制面板。
   > 要获取生成的端点值，请在服务器联机后运行以下 AWS CLI 命令：
   >
   > ```
   > aws opsworks describe-servers --server-name server_name
   > ```

使用 AWS CloudFormation 创建 Chef Automate 服务器 (CLI)

如果您的本地计算机尚未运行 AWS CLI，请 AWS CLI 按照 AWS 命令行界面用户指南中的安装说明下载并安装。本部分未介绍可以与 create-stack 命令结合使用的所有参数。有关 create-stack 参数的更多信息，请参阅 create-stack 参考 中的 AWS CLI 。

1. 请务必完成创建 AWS OpsWorks for Chef Automate 服务器的 先决条件 。
2. 创建服务角色和实例配置文件。 AWS OpsWorks 提供了一个可用于创建两者的 AWS CloudFormation 模板。运行以下 AWS CLI 命令创建一个 AWS CloudFormation 堆栈，用于为您创建服务角色和实例配置文件。

   ```
   aws cloudformation create-stack --stack-name OpsWorksCMRoles --template-url
    https://s3.amazonaws.com/opsworks-cm-us-east-1-prod-default-assets/misc/opsworks-
   cm-roles.yaml --capabilities CAPABILITY_NAMED_IAM
   ```

   创建 AWS CloudFormation 完堆栈后，在您的账户中查找并复制服务角色的 ARN。

   ```
   aws iam list-roles --path-prefix "/service-role/" --no-paginate
   ```

   在 `list-roles` 命令的结果中，查找类似于以下内容的服务角色和实例配置文件条目。记下服务角色和实例配置文件的 ARN，然后将其添加到用于创建服务器堆栈的 AWS CloudFormation 模板中。

   ```
   {
       "AssumeRolePolicyDocument": {
           "Version": "2012-10-17",
           "Statement": [
               {
                   "Action": "sts:AssumeRole",
                   "Effect": "Allow",
                   "Principal": {
                       "Service": "ec2.amazonaws.com"
                   }
               }
           ]
       },
       "RoleId": "AROZZZZZZZZZZQG6R22HC",
       "CreateDate": "2018-01-05T20:42:20Z",
       "RoleName": "aws-opsworks-cm-ec2-role",
       "Path": "/service-role/",
   ```

```
    "Arn": "arn:aws:iam::000000000000:role/service-role/aws-opsworks-cm-ec2-role"
},
{
    "AssumeRolePolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Action": "sts:AssumeRole",
                "Effect": "Allow",
                "Principal": {
                    "Service": "opsworks-cm.amazonaws.com"
                }
            }
        ]
    },
    "RoleId": "AROZZZZZZZZZZZZZZZ6QE",
    "CreateDate": "2018-01-05T20:42:20Z",
    "RoleName": "aws-opsworks-cm-service-role",
    "Path": "/service-role/",
    "Arn": "arn:aws:iam::000000000000:role/service-role/aws-opsworks-cm-service-
role"
}
```

3. 再次运行该create-stack命令来创建 AWS OpsWorks for Chef Automate 服务器。

   - 将 *stack_name* 替换为您的堆栈的名称。这是 AWS CloudFormation 堆栈的名称，而不是你的 Chef Automate 服务器的名称。Chef Automate 服务器名称是 AWS CloudFormation 模板ServerName中的值。
   - 将 *template* 替换为您的模板文件的路径，并将扩展名 *yaml # json* 相应地替换为 .yaml 或 .json。
   - 的值对--parameters应于 CreateServerAPI EngineAttributes中的值。对于 Chef，用户提供的用于创建服务器的引擎属性为 CHEF_AUTOMATE_PIVOTAL_KEY（您使用the section called “先决条件 ”中介绍的实用工具生成的 base64 编码的 RSA 公共密钥）和 CHEF_AUTOMATE_ADMIN_PASSWORD（您创建的一个长度介于 8 到 32 个字符的密码）。有关 CHEF_AUTOMATE_ADMIN_PASSWORD 的更多信息，请参阅 使用 Chef Automate 服务器创建 AWS CLI。您可以提供一个指向包含您的关键密钥的 PEM 文件的指针来作为 PivotalKey 参数的值，如示例中所示。如果模板中CHEF_AUTOMATE_PIVOTAL_KEY未指定CHEF_AUTOMATE_ADMIN_PASSWORD和的值，则必须在 AWS CLI 命令中提供这些值。

```
aws cloudformation create-stack --stack-name stack_name
 --template-body file://template.yaml or json --parameters
 ParameterKey=PivotalKey,ParameterValue="base64_encoded_RSA_public_key_value"
```

下面是一个示例，其中包括了 CHEF_AUTOMATE_ADMIN_PASSWORD 和 CHEF_AUTOMATE_PIVOTAL_KEY 属性的示例值。如果您未在 AWS CloudFormation 模板中为这些属性指定值，请运行类似的命令。

```
aws cloudformation create-stack --stack-name "OpsWorksCMChefServerStack"
 --template-body file://opsworkscm-server.yaml --parameters
 ParameterKey=PivotalKey,ParameterValue="$(openssl rsa -in "pivotalKey.pem" -
pubout)" ParameterKey=Password,ParameterValue="SuPer\$ecret890"
```

4. 堆栈创建完成后，在 AWS OpsWorks for Chef Automate 控制台中打开新服务器的 “属性” 页面，然后下载入门套件。下载新的初学者工具包将重置 Chef Automate 控制面板管理员密码。
5. 如果您的服务器将使用自定义域、证书和私钥，请按照（可选）配置 knife 使用自定义域 中的步骤配置 knife.rb，然后继续执行步骤 7。

   如果您不使用自定义域，请从以下 Amazon S3 存储桶位置下载根证书颁发机构 (CA) 证书：https://s3.amazonaws.com/opsworks-cm-us-east-1-prod-default-assets/misc/opsworks-cm-ca-2020-root.pem。将证书文件保存到一个安全便利的位置。在下一步中配置 knife.rb 时将需要使用该证书。
6. 要在新服务器上使用 knife 命令，请更新 Chef knife.rb 配置文件设置。初学者工具包中包括一个示例 knife.rb 文件。以下示例演示如何在不使用自定义域的服务器上设置 knife.rb。如果您使用的是自定义域，请参阅（可选）配置 knife 使用自定义域了解 knife 配置说明。

   - 将 *ENDPOINT* 替换为该服务器的终端节点值。这是堆栈创建操作的部分输出。您可以通过运行以下命令来获取终端节点。

     ```
     aws cloudformation describe-stacks --stack-name stack_name
     ```

   - 将 client_key 配置中的 *key_pair_file.pem* 替换为包含用于创建服务器的 CHEF_AUTOMATE_PIVOTAL_KEY 的 PEM 文件的名称。

     ```
     base_dir = File.join(File.dirname(File.expand_path(__FILE__)), '..')

     log_level                :info
     ```

```
log_location            STDOUT
node_name               'pivotal'
client_key              File.join(base_dir, '.chef', 'key_pair_file.pem')
syntax_check_cache_path File.join(base_dir, '.chef', 'syntax_check_cache')
cookbook_path           [File.join(base_dir, 'cookbooks')]

chef_server_url         'ENDPOINT/organizations/default'
ssl_ca_file             File.join(base_dir, '.chef', 'ca_certs', 'opsworks-cm-
ca-2020-root.pem')
trusted_certs_dir       File.join(base_dir, '.chef', 'ca_certs')
```

7. 服务器创建过程完成后，请转到the section called “完成配置和上传说明书”。如果堆栈创建失败，请查看控制台中显示的错误消息，以帮助您解决问题。有关对 AWS CloudFormation 堆栈中的错误进行故障排除的更多信息，请参阅《AWS CloudFormation 用户指南》中的故障排除。

更新 AWS OpsWorks for Chef Automate 服务器以使用自定义域

> **Important**
>
> AWS OpsWorks for Chef Automate 已于 2024 年 5 月 5 日停用，新客户和现有客户均已禁用。我们建议现有客户迁移到 Chef SaaS 或其他替代解决方案。如果您有任何疑问，可以通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

本节介绍如何使用 AWS OpsWorks for Chef Automate 服务器的备份创建新服务器，更新现有服务器以使用自定义域和证书。本质上，您是在复制现有的 AWS OpsWorks for Chef Automate 2.0 服务器，方法是从备份中创建新服务器，然后将新服务器配置为使用自定义域、证书和私钥。

主题

先决条件

以下是更新现有 AWS OpsWorks for Chef Automate 服务器以使用自定义域和证书的要求。

- 要更新（或复制）的服务器必须运行 Chef Automate 2.0。
- 决定要用于创建新服务器的备份。您必须至少有一个要更新的服务器的备份可用。有关中备份的更多信息 AWS OpsWorks for Chef Automate，请参阅备份 AWS OpsWorks for Chef Automate 服务器。
- 准备好用于创建现有服务器（作为备份源）的服务角色和实例配置文件 ARN。
- 请确保您正在运行最新版本的 AWS CLI。有关更新 AWS CLI 工具的更多信息，请参阅 AWS 命令行界面用户指南 AWS CLI中的安装。

限制

通过使用备份创建新服务器来更新现有服务器时，新服务器不能与现有 AWS OpsWorks for Chef Automate 服务器完全相同。

- 您只能使用 AWS CLI 或其中一个AWS 软件开发工具包来完成此过程。无法通过使用 AWS Management Console从备份创建新的服务器。
- 新服务器不能使用与账户内和 Amazon Web Services Region 内的现有服务器相同的名称。名称必须与用作备份源的现有服务器不同。
- 连接到现有服务器的节点不由新服务器管理。您必须执行以下操作之一。
  - 附加不同的节点，因为节点不能由多个 Chef Automate 服务器管理。
  - 将节点从现有服务器（备份源）迁移到新服务器和新的自定义域端点。有关如何迁移节点的更多信息，请参阅 Chef 文档。

更新服务器以使用自定义域

要更新现有的 Chef Automate 2.0 服务器，您可以通过运行 `create-server` 命令、添加参数以指定备份、自定义域、自定义证书和自定义私有密钥来创建该服务器的副本。

1. 如果您的 `create-server` 命令中没有可供指定的服务角色或实例配置文件 ARN，请按照使用 Chef Automate 服务器创建 AWS CLI中的步骤 1-5 创建您可以使用的服务角色和实例配置文件。
2. 如果您尚未执行此操作，请查找现有 Chef Automate 2.0 服务器的备份，将以此备份为基础创建使用自定义域的新服务器。运行以下命令以显示有关您的账户和某个区域中所有 AWS OpsWorks for Chef Automate 备份的信息。确保记下要使用的备份的 ID。

   ```
   aws opsworks-cm --region region name describe-backups
   ```

3. 通过运行`create-server`命令创建 AWS OpsWorks for Chef Automate 服务器。

- `--engine` 值为 `ChefAutomate`，`--engine-model` 为 `Single`，`--engine-version` 为 `12`。
- 在您的 AWS 账户中，服务器名称在每个区域内必须是唯一的。服务器名称必须以字母开头；然后允许字母、数字或连字符 (-)，最多 40 个字符。
- 使用步骤 1 中的实例配置文件 ARN 和服务角色 ARN。
- 有效实例类型为 `m5.large`、`r5.xlarge` 或 `r5.2xlarge`。有关这些实例类型的规范的更多信息，请参阅 Amazon EC2 用户指南中的实例类型。
- `--engine-attributes` 参数是可选的；如果您不指定一个或两个值，则服务器创建过程会为您生成这些值。如果您添加 `--engine-attributes`，请指定您在步骤 2 中生成的 `CHEF_AUTOMATE_PIVOTAL_KEY` 值、`CHEF_AUTOMATE_ADMIN_PASSWORD`，或两者。

  如果您不设置 `CHEF_AUTOMATE_ADMIN_PASSWORD` 的值，一个密码将生成并作为 `create-server` 响应的一部分返回。您也可以在控制台中再次下载初学者工具包，该工具包会重新生成此密码。该密码的最小长度为 8 个字符，最大长度为 32 个字符。该密码可以包含字母、数字和特殊字符 (`!/@#$%^+=_`)。该密码必须至少包含一个小写字母、一个大写字母、一个数字和一个特殊字符。
- SSH 密钥对是可选的，但可以帮助您连接到 Chef Automate 服务器 (如果您需要重置 Chef Automate 控制面板管理员密码)。有关创建 SSH 密钥对的更多信息，请参阅《Amazon EC2 用户指南》中的 Amazon EC2 密钥对。
- 要使用自定义域，请将以下参数添加到命令中。否则，Chef Automate 服务器创建过程会自动为您生成终端节点。配置自定义域需要所有三个参数。有关使用这些参数的其他要求的信息，请参阅 AWS OpsWorks CM API 参考CreateServer中的。
  - `--custom-domain` - 服务器的可选公有端点，例如 `https://aws.my-company.com`。
  - `--custom-certificate` - PEM 格式的 HTTPS 证书。该值可以是单个自签名证书或证书链。
  - `--custom-private-key` - PEM 格式的私有密钥，用于通过 HTTPS 连接到服务器。私有密钥不得加密；无法使用密码或密码短语保护它。
- 需要每周进行系统维护。有效值必须按以下格式指定：`DDD:HH:MM`。指定的时间为协调世界时 (UTC)。如果您不指定 `--preferred-maintenance-window` 的值，则默认值为星期二、星期三或星期五的一小时随机时间段。
- `--preferred-backup-window` 的有效值必须按以下格式之一指定：`HH:MM` (针对每日备份) 或 `DDD:HH:MM` (针对每周备份)。指定的时间采用 UTC 格式。默认值为随机每日开始时间。要退出自动备份，请改为添加参数 `--disable-automated-backup`。
- 对于 `--security-group-ids`，输入一个或多个安全组 ID，用空格分隔。

- 对于 `--subnet-ids`，输入子网 ID。
- 对于 `--backup-id`，输入您在步骤 2 中复制的备份的 ID。

```
aws opsworks-cm create-server --engine "ChefAutomate" --engine-model "Single"
 --engine-version "12" --server-name "server_name" --instance-profile-arn
 "instance_profile_ARN" --instance-type "instance_type" --engine-attributes
 '{"CHEF_AUTOMATE_PIVOTAL_KEY":"pivotal_key","CHEF_AUTOMATE_ADMIN_PASSWORD":"password"}'
 --key-pair "key_pair_name" --preferred-maintenance-window
 "ddd:hh:mm" --preferred-backup-window "ddd:hh:mm" --security-group-
ids security_group_id1 security_group_id2 --service-role-arn "service_role_ARN" --
subnet-ids subnet_ID --backup-id backup_ID
```

以下示例创建使用自定义域的 Chef Automate 服务器。

```
aws opsworks-cm create-server --engine "ChefAutomate" --engine-model "Single" --
engine-version "12" \
    --server-name "my-custom-domain-server" \
    --instance-profile-arn "arn:aws:iam::12345678912:instance-profile/aws-opsworks-
cm-ec2-role" \
    --instance-type "m5.large" \
    --engine-attributes
 '{"CHEF_AUTOMATE_PIVOTAL_KEY":"MZZE...Wobg","CHEF_AUTOMATE_ADMIN_PASSWORD":"zZZzDj2DLYXSZF
 \
    --custom-domain "my-chef-automate-server.my-corp.com" \
    --custom-certificate "-----BEGIN CERTIFICATE----- EXAMPLEqEXAMPLE== -----END
 CERTIFICATE-----" \
    --custom-private-key "-----BEGIN RSA PRIVATE KEY----- EXAMPLEqEXAMPLE= -----END
 RSA PRIVATE KEY-----" \
    --key-pair "amazon-test" \
    --preferred-maintenance-window "Mon:08:00" \
    --preferred-backup-window "Sun:02:00" \
    --security-group-ids sg-b00000001 sg-b0000008 \
    --service-role-arn "arn:aws:iam::12345678912:role/service-role/aws-opsworks-cm-
service-role" \
    --subnet-ids subnet-300aaa00 \
    --backup-id MyChefServer-20191004122143125
```

4. AWS OpsWorks for Chef Automate 创建新服务器大约需要 15 分钟。在 `create-server` 命令的输出中，复制 `Endpoint` 属性的值。示例如下：

```
"Endpoint": "automate-07-exampleexample.opsworks-cm.us-east-1.amazonaws.com"
```

请勿关闭 `create-server` 命令的输出或关闭您的 Shell 会话，因为该输出可能包含不再显示的重要信息。要从 `create-server` 结果中获取密码和初学者工具包，请继续执行下一步。

5. 如果您选择为您 AWS OpsWorks for Chef Automate 生成密钥和密码，则可以使用 j q 等 JSON 处理器从`create-server`结果中提取可用格式的密钥和密码。安装 jq 后，您可以运行以下命令来提取关键密钥、Chef Automate 控制面板管理员密码和初学者工具包。如果您未在步骤 3 中提供自己的关键密钥和密码，请确保将提取的关键密钥和管理员密码保存在方便而安全的位置。

```
#Get the Chef password:
cat resp.json | jq -r '.Server.EngineAttributes[] | select(.Name ==
 "CHEF_AUTOMATE_ADMIN_PASSWORD") | .Value'

#Get the Chef Pivotal Key:
cat resp.json | jq -r '.Server.EngineAttributes[] | select(.Name ==
 "CHEF_AUTOMATE_PIVOTAL_KEY") | .Value'

#Get the Chef Starter Kit:
cat resp.json | jq -r '.Server.EngineAttributes[] | select(.Name ==
 "CHEF_STARTER_KIT") | .Value' | base64 -D > starterkit.zip
```

6. 或者，如果您没有从`create-server`命令结果中提取入门套件，则可以在 AWS OpsWorks for Chef Automate 控制台中从服务器的 “属性” 页面下载新的入门套件。下载新的初学者工具包将重置 Chef Automate 控制面板管理员密码。
7. 在企业的 DNS 管理工具中创建 CNAME 条目，将您的自定义域指向您在步骤 4 中复制的 AWS OpsWorks for Chef Automate 终端节点。在完成此步骤之前，您无法访问或登录到服务器。
8. 服务器创建过程完成后，请转到the section called “完成配置和上传说明书”。

另请参阅

- 使用 Chef Automate 服务器创建 AWS CLI
- 从 Backup 中恢复 AWS OpsWorks for Chef Automate 服务器
- CreateServer在 AWS OpsWorks CM API 参考中
- 《AWS CLI Command Reference》中的 create-server。

为服务器重新生成入门套件 AWS OpsWorks for Chef Automate

> **Important**
>
> AWS OpsWorks for Chef Automate 已于 2024 年 5 月 5 日停用，新客户和现有客户均已禁用。我们建议现有客户迁移到 Chef SaaS 或其他替代解决方案。如果您有任何疑问，可以通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

的入门套件 AWS OpsWorks for Chef Automate 包含一个带有示例的自述文件、一个`knife.rb`配置文件和一个供主用户或关键用户使用的私钥。每次下载初学者工具包时，将生成新密钥对，旧密钥对将会重置。您可以通过以下两种方式之一为 AWS OpsWorks for Chef Automate 服务器重新生成入门套件：

- 在 AWS OpsWorks 控制台中，在 AWS OpsWorks for Chef Automate 服务器详细信息页面的操作菜单上。系统会提示您确认是否要重新生成和重置旧的关键密钥。
- 通过在中运行命令 AWS CLI。

有关如何使用初学者工具包的更多信息，请参阅使用初学者工具包配置 Chef Server。

使用重新生成入 AWS OpsWorks for Chef Automate 门套件 AWS CLI

> **Note**
>
> 当您重新生成初学者工具包时，您还会为您的 Chef Automate 服务器重新生成和重置身份验证密钥对，并删除当前的密钥对。

通过运行update-server-engine-attributes命令重新生成初学者工具包。在会 AWS CLI 话中，运行以下命令。将您的服务器名称指定为 `--server-name` 的值。要将您自己的公钥设置为 `CHEF_AUTOMATE_PIVOTAL_KEY` 的值，请在 `--attribute-value` 中指定公钥的值。否则，设置 `--attribute-value` 为 null。

```
aws opsworks-cm update-server-engine-attributes \
   --server-name server_name \
   --attribute-name "CHEF_AUTOMATE_PIVOTAL_KEY" \
   --attribute-value your_public_key
```

以下命令是指定服务器管理员想要使用的公钥值的示例。

```
aws opsworks-cm update-server-engine-attributes \
   --server-name your-test-server \
   --attribute-name "CHEF_AUTOMATE_PIVOTAL_KEY" \
   --attribute-value "-----BEGIN PUBLIC KEY-----ExamplePublicKey-----END PUBLIC
 KEY-----"
```

以下命令是一个允许 AWS OpsWorks for Chef Automate 重新生成公钥的示例。

```
aws opsworks-cm update-server-engine-attributes \
   --server-name your-test-server \
   --attribute-name "CHEF_AUTOMATE_PIVOTAL_KEY" \
   --attribute-value null
```

此命令的输出是有关服务器的信息以及一个 base64 编码的 ZIP 文件。ZIP 文件包含一个 Chef 初学者工具包，其中包括 README、配置文件和所需的 RSA 私钥。保存此文件，将其解压缩，然后切换到解压缩文件内容的目录。从这个目录中，您可以运行 `knife` 命令。

使用 AWS OpsWorks for Chef Automate 资源上的标签

> **Important**
>
> AWS OpsWorks for Chef Automate 已于 2024 年 5 月 5 日停用，新客户和现有客户均已禁用。我们建议现有客户迁移到 Chef SaaS 或其他替代解决方案。如果您有任何疑问，可以通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

标签是一些充当元数据的词和短语，用于标识和组织 AWS 资源。在中 AWS OpsWorks for Chef Automate，一个资源最多可以有 50 个用户应用的标签。每个标签由一个键和一个可选的值组成。您可以将标签应用于 AWS OpsWorks for Chef Automate中的以下资源：

- AWS OpsWorks for Chef Automate 服务器
- AWS OpsWorks for Chef Automate 服务器备份

AWS 资源标签可以帮助您跟踪成本、控制对资源的访问权限、对资源进行分组以实现任务自动化，或者按用途或生命周期阶段组织资源。有关标签优势的更多信息，请参阅 AWS Answers 中的 AWS 标签策略和《AWS Billing and Cost Management 用户指南》中的使用成本分配标签。

要使用标签控制对 AWS OpsWorks for Chef Automate 服务器或备份的访问权限，您可以在 AWS Identity and Access Management (IAM) 中创建或编辑策略声明。有关更多信息，请参阅《AWS Identity and Access Management IAM 用户指南》中的使用资源标签控制对 AWS 资源的访问。

当您将标签应用于 AWS OpsWorks for Chef Automate 服务器时，标签还会应用于服务器的备份、存储备份的 Amazon S3 存储桶、服务器的 Amazon EC2 实例 AWS Secrets Manager、存储在中的服务器机密以及服务器使用的弹性 IP 地址。标签不会传播到用于创建服务器的 AWS CloudFormation AWS OpsWorks 堆栈。

主题

标签的工作原理 AWS OpsWorks for Chef Automate

在此版本中，您可以使用 AWS OpsWorks CM API 或 AWS Management Console添加和管理标签。AWS OpsWorks CM 还尝试将您添加到服务器的标签添加到与服务器关联的 AWS 资源中，包括 EC2 实例、Secrets Manager 中的密钥、弹性 IP 地址、安全组、S3 存储桶和备份。下表概述如何在 AWS OpsWorks for Chef Automate中添加和管理标签。

操作	要使用的内容
向新 AWS OpsWorks for Chef Automate 服务器或手动创建的备份添加标签。	• 选择 Create Chef Automate server (创建 Chef Automate 服务器)，然后在 Configure advanced settings (配置高级设置) 页面上添加标签。 • 在备份页面上为现有服务器选择创建备份，然后在创建 Chef Automate 2 服务器的备份页面上添加标签。 • 向 `CreateServer` 或 `CreateBackup` 命令添加 `Tags` 参数。
查看资源上的标签。	• 在服务器的详细信息页面上，选择导航窗格中的 Tags (标签)。

操作	要使用的内容
	• 在服务器的 Backups (备份) 页面上，选择一个备份，然后选择 Edit backup (编辑备份)。 • 运行 ListTagsForResource 命令。
为现有 AWS OpsWorks for Chef Automate 服务器或备份添加标签，无论备份是手动创建还是自动创建。	• 在服务器的详细信息页面上，选择导航窗格中的 Tags (标签)，然后选择 Edit (编辑)。 • 在服务器的 Backups (备份) 页面上，选择一个备份，然后选择 Edit backup (编辑备份)。 • 运行 TagResource 命令。
从资源中删除标签。	• 在服务器的详细信息页面上，选择导航窗格中的 Tags (标签)，然后选择 Edit (编辑)。选择要删除的标签旁边的 X。 • 在服务器的 Backups (备份) 页面上，选择一个备份，然后选择 Edit backup (编辑备份)。选择要删除的标签旁边的 X。 • 运行 UntagResource 命令。

`DescribeServers` 和 `DescribeBackups` 响应不包含标签信息。要显示标签，请使用 `ListTagsForResource` API。

在 AWS OpsWorks for Chef Automate （控制台）中添加和管理标签

本节中的过程将在 AWS Management Console中执行。

如果添加标签，标签密钥不能为空。密钥最多可包含 127 个字符，并且只能包含 Unicode 字母、数字或分隔符，或以下特殊字符：`+ - = . _ : / @`。标签值是可选的。您可以添加具有密钥但没有值的标签。值最多可包含 255 个字符，并且只能包含 Unicode 字母、数字或分隔符，或以下特殊字符：`+ - = . _ : / @`

主题

- 向新 AWS OpsWorks for Chef Automate 服务器添加标签（控制台）
- 向新备份添加标签（控制台）
- 在现有服务器上添加或查看标签（控制台）

- 在现有备份上添加或查看标签（控制台）
- 从服务器中删除标签（控制台）
- 从备份中删除标签（控制台）

向新 AWS OpsWorks for Chef Automate 服务器添加标签（控制台）

1. 请务必完成创建 AWS OpsWorks for Chef Automate 服务器的所有先决条件。
2. 按照创建 Chef Automate 服务器中的步骤 1-10 操作。
3. 指定自动备份设置后，在配置高级设置页面的标签区域中添加标签。您最多可添加 50 个标签。添加完标签后，选择 Next。
4. 继续前往创建 Chef Automate 服务器的步骤 13，并查看您为新服务器选择的设置。

向新备份添加标签（控制台）

1. 在 AWS OpsWorks for Chef Automate 主页上，选择现有的 Chef Automate 服务器。
2. 从服务器的详细信息页面中，选择导航窗格中的 Backups (备份)。
3. 在 Backups (备份) 页面上，选择 Create backup (创建备份)。
4. 添加标签。在添加完标签后，请选择 Create (创建)。

在现有服务器上添加或查看标签（控制台）

1. 在 AWS OpsWorks for Chef Automate 主页上，选择现有的 Chef Automate 服务器以打开其详细信息页面。
2. 在导航窗格中选择 Tags (标签)，或在详细信息页面底部选择 View all tags (查看所有标签)。
3. 在 Tags (标签) 页面上，选择 Edit (编辑)。
4. 在服务器上添加或编辑标签。完成后，选择 Save (保存)。

> **Note**
>
> 请注意，更改 Chef Automate 服务器上的标签也会更改与服务器关联的资源上的标签，例如 EC2 实例、弹性 IP 地址、安全组、S3 存储桶和备份。

在现有备份上添加或查看标签（控制台）

1. 在 AWS OpsWorks for Chef Automate 主页上，选择现有的 Chef Automate 服务器以打开其详细信息页面。
2. 在导航窗格中选择 Backups (备份)，或在详细信息页面的 Recent backups (最近备份) 区域中选择 View all backups (查看所有备份)。
3. 在 Backups (备份) 页面上，选择要管理的备份，然后选择 Edit backup (编辑备份)。
4. 在备份上添加或编辑标签。完成后，选择 Update (更新)。

从服务器中删除标签（控制台）

1. 在 AWS OpsWorks for Chef Automate 主页上，选择现有的 Chef Automate 服务器以打开其详细信息页面。
2. 在导航窗格中选择 Tags (标签)，或在详细信息页面底部选择 View all tags (查看所有标签)。
3. 在 Tags (标签) 页面上，选择 Edit (编辑)。
4. 选择标签旁边的 X 以删除标签。完成后，选择 Save (保存)。

> **Note**
>
> 请注意，更改 Chef Automate 服务器上的标签也会更改与服务器关联的资源上的标签，例如 EC2 实例、弹性 IP 地址、安全组、S3 存储桶和备份。

从备份中删除标签（控制台）

1. 在 AWS OpsWorks for Chef Automate 主页上，选择现有的 Chef Automate 服务器以打开其详细信息页面。
2. 在导航窗格中选择 Backups (备份)，或在详细信息页面的 Recent backups (最近备份) 区域中选择 View all backups (查看所有备份)。
3. 在 Backups (备份) 页面上，选择要管理的备份，然后选择 Edit backup (编辑备份)。
4. 选择标签旁边的 X 以删除标签。完成后，选择 Update (更新)。

在 AWS OpsWorks for Chef Automate (CLI) 中添加和管理标签

本节中的过程将在 AWS CLI中执行。在开始使用标签 AWS CLI 之前，请务必运行的是的最新版本。有关安装或更新的更多信息 AWS CLI，请参阅《AWS Command Line Interface 用户指南》 AWS CLI 中的安装。

如果添加标签，标签密钥不能为空。密钥最多可包含 127 个字符，并且只能包含 Unicode 字母、数字或分隔符，或以下特殊字符：+ - = . _ : / @。标签值是可选的。您可以添加具有密钥但没有值的标签。值最多可包含 255 个字符，并且只能包含 Unicode 字母、数字或分隔符，或以下特殊字符：+ - = . _ : / @

主题

向新 AWS OpsWorks for Chef Automate 服务器添加标签 (CLI)

创建 AWS OpsWorks for Chef Automate 服务器时，您可以使用 AWS CLI 来添加标签。此过程并不完整描述如何创建服务器。有关如何使用创建 AWS OpsWorks for Chef Automate 服务器的详细信息，AWS CLI 请参阅本指南使用 Chef Automate 服务器创建 AWS CLI中的。您最多可以向服务器添加 50 个标签。

1. 请务必完成创建 AWS OpsWorks for Chef Automate 服务器的所有先决条件。
2. 完成使用 Chef Automate 服务器创建 AWS CLI的步骤 1-5。
3. 对于步骤 6，当您运行 `create-server` 命令时，请将 `--tags` 参数添加到命令，如以下示例所示。

   ```
   aws opsworks-cm create-server ... --tags Key=Key1,Value=Value1
    Key=Key2,Value=Value2
   ```

   以下是仅显示 `create-server` 命令的标签部分的示例。

```
aws opsworks-cm create-server ... --tags Key=Stage,Value=Production
 Key=Department,Value=Marketing
```

4. 完成使用 Chef Automate 服务器创建 AWS CLI 中的其余步骤。要验证标签是否已添加到新服务器，请按照本主题的列出资源标签中的步骤操作。

向新备份添加标签 (CLI)

在创建新的 AWS OpsWorks for Chef Automate 服务器手动备份时，可以使用 AWS CLI 来添加标记。此过程并不完整描述如何创建手动备份。有关如何创建手动备份的详细信息，请参阅中的 “要执行手动备份 AWS CLI” 备份 AWS OpsWorks for Chef Automate 服务器。您最多可以向备份添加 50 个标签。如果服务器具有标签，则会自动使用服务器的标签标记新的备份。

默认情况下，当您创建新 AWS OpsWorks for Chef Automate 服务器时，自动备份处于启用状态。您可以通过运行 `tag-resource` 命令向自动备份添加标签，如本主题的将标签添加到现有服务器或备份 (CLI)中所述。

- 要在创建备份时向手动备份添加标签，请运行以下命令。仅显示命令的标签部分。有关完整 `create-backup` 命令的示例，请参阅 备份 AWS OpsWorks for Chef Automate 服务器 中的 “在 AWS CLI执行手动备份”。

  ```
  aws opsworks-cm create-backup ... --tags Key=Key1,Value=Value1
   Key=Key2,Value=Value2
  ```

  以下示例仅显示 `create-backup` 命令的标签部分。

  ```
  aws opsworks-cm create-backup ... --tags Key=Stage,Value=Production
   Key=Department,Value=Marketing
  ```

将标签添加到现有服务器或备份 (CLI)

您可以运行 `tag-resource` 命令向现有 AWS OpsWorks for Chef Automate 服务器或备份添加标签（无论备份是自动创建还是手动创建的）。指定目标资源的 Amazon 资源编号 (ARN) 以向其添加标签。

1. 要获取要应用标签的资源的 ARN，请执行以下操作：

- 对于服务器，请运行 `describe-servers --server-name server_name`。命令的结果显示服务器 ARN。
- 对于备份，请运行 `describe-backups --backup-id backup_ID`。命令的结果显示备份 ARN。您也可以运行`describe-backups --server-name server_name`以显示有关特定 AWS OpsWorks for Chef Automate 服务器的所有备份的信息。

以下示例仅显示 `describe-servers --server-name opsworks-cm-test` 命令的结果中的 `ServerArn`。`ServerArn` 值将添加到 `tag-resource` 命令中以向服务器添加标签。

```
{
    "Servers": [
        {
            ...
            "ServerArn": "arn:aws:opsworks-cm:us-west-2:123456789012:server/
opsworks-cm-test/EXAMPLEd-66b0-4196-8274-d1a2bEXAMPLE"
        }
    ]
}
```

2. 使用您在步骤 1 中返回的 ARN 运行 `tag-resource` 命令。

```
aws opsworks-cm tag-resource --resource-arn "server_or_backup_ARN" --tags
 Key=Key1,Value=Value1 Key=Key2,Value=Value2
```

示例如下：

```
aws opsworks-cm tag-resource --resource-arn "arn:aws:opsworks-cm:us-
west-2:123456789012:server/opsworks-cm-test/EXAMPLEd-66b0-4196-8274-d1a2bEXAMPLE"
 --tags Key=Stage,Value=Production Key=Department,Value=Marketing
```

3. 要验证标签是否已成功添加，请继续下一个过程，即列出资源标签。

列出资源标签

您可以运行`list-tags-for-resource`命令来显示附加到 AWS OpsWorks for Chef Automate 服务器或备份的标签。指定目标资源的 ARN 以查看其标签。

1. 要获取要列出其标签的资源的 ARN，请执行以下操作：

- 对于服务器，请运行 `describe-servers --server-name server_name`。命令的结果显示服务器 ARN。
- 对于备份，请运行 `describe-backups --backup-id backup_ID`。命令的结果显示备份 ARN。您也可以运行`describe-backups --server-name server_name`以显示有关特定 AWS OpsWorks for Chef Automate 服务器的所有备份的信息。

2. 使用您在步骤 1 中返回的 ARN 运行 `list-tags-for-resource` 命令。

```
aws opsworks-cm list-tags-for-resource --resource-arn "server_or_backup_ARN"
```

示例如下：

```
aws opsworks-cm tag-resource --resource-arn "arn:aws:opsworks-cm:us-
west-2:123456789012:server/opsworks-cm-test/EXAMPLEd-66b0-4196-8274-d1a2bEXAMPLE"
```

如果资源上有标签，则命令返回如下所示的结果。

```
{
    "Tags": [
        {
            "Key": "Stage",
            "Value": "Production"
        },
        {
            "Key": "Department",
            "Value": "Marketing"
        }
    ]
}
```

从资源中删除标签

您可以运行 `untag-resource` 命令从 AWS OpsWorks for Chef Automate 服务器或备份中删除标签。如果资源被删除，资源上的标签也会被删除。指定目标资源的 Amazon 资源编号 (ARN)，以便从其中删除标签。

1. 要获取要从中删除标签的资源的 ARN，请执行以下操作：

- 对于服务器，请运行 `describe-servers --server-name server_name`。命令的结果显示服务器 ARN。
- 对于备份，请运行 `describe-backups --backup-id backup_ID`。命令的结果显示备份 ARN。您也可以运行`describe-backups --server-name server_name`以显示有关特定 AWS OpsWorks for Chef Automate 服务器的所有备份的信息。

2. 使用您在步骤 1 中返回的 ARN 运行 `untag-resource` 命令。仅指定要删除的标签。

```
aws opsworks-cm untag-resource --resource-arn "server_or_backup_ARN" --tags
 Key=Key1,Value=Value1 Key=Key2,Value=Value2
```

在此示例中，`untag-resource` 命令仅删除密钥为 `Stage` 且值为 `Production` 的标签。

```
aws opsworks-cm untag-resource --resource-arn "arn:aws:opsworks-cm:us-
west-2:123456789012:server/opsworks-cm-test/EXAMPLEd-66b0-4196-8274-d1a2bEXAMPLE"
 --tags Key=Stage,Value=Production
```

3. 要验证标签是否已成功删除，请按照本主题的列出资源标签中的步骤操作。

另请参阅

- 使用 Chef Automate 服务器创建 AWS CLI
- 备份 AWS OpsWorks for Chef Automate 服务器
- AWS 标记策略
- 使用AWS Identity and Access Management 用户指南中的资源标签控制对资源的访问权限 AWS
- 《AWS Billing and Cost Management 用户指南》中的使用成本分配标签
- CreateBackup在 AWS OpsWorks CM API 参考中
- CreateServer在 AWS OpsWorks CM API 参考中
- TagResource在 AWS OpsWorks CM API 参考中
- ListTagsForResource在 AWS OpsWorks CM API 参考中
- UntagResource在 AWS OpsWorks CM API 参考中

备份和恢复 AWS OpsWorks for Chef Automate 服务器

> **Important**
>
> AWS OpsWorks for Chef Automate 已于 2024 年 5 月 5 日停用，新客户和现有客户均已禁用。我们建议现有客户迁移到 Chef SaaS 或其他替代解决方案。如果您有任何疑问，可以通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

本节介绍如何备份和恢复 AWS OpsWorks for Chef Automate 服务器以及如何删除备份。

主题

备份 AWS OpsWorks for Chef Automate 服务器

> **Important**
>
> AWS OpsWorks for Chef Automate 已于 2024 年 5 月 5 日停用，新客户和现有客户均已禁用。我们建议现有客户迁移到 Chef SaaS 或其他替代解决方案。如果您有任何疑问，可以通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

您可以定义每日或每周定期 AWS OpsWorks for Chef Automate 服务器备份，并让该服务代表您将备份存储在亚马逊简单存储服务 (Amazon S3) 中。或者，您可以按需进行手动备份。

由于备份存储在 Amazon S3 中，因此它们会产生额外费用。您最多可以将备份的保留数量设定为 30 个。您可以使用 AWS 支持渠道提交服务请求以更改该限制。发送到 Amazon S3 存储桶的内容可能包含客户内容。有关删除敏感数据的更多信息，请参阅如何清空 S3 存储桶？或如何删除 S3 存储桶？。

您可以为 AWS OpsWorks for Chef Automate 服务器的备份添加标签。如果已向 AWS OpsWorks for Chef Automate 服务器添加标签，此服务器的自动备份将继承这些标签。有关如何在备份中添加和管理标签的更多信息，请参阅本指南中的使用 AWS OpsWorks for Chef Automate 资源上的标签。

主题

- 手动备份
- 删除备份

自动备份

配置 AWS OpsWorks for Chef Automate 服务器时，可以选择自动备份或手动备份。 AWS OpsWorks for Chef Automate 在安装程序的 “配置高级设置” 页面的 “自动备份” 部分中选择的一小时和当天启动自动备份。在服务器联机后，可以通过执行以下步骤更改备份设置 (从 Chef Automate 服务器主页上的服务器磁贴或服务器的“Properties”页)。

更改自动备份设置

1. 在 Chef servers 主页的服务器磁贴的 Actions 菜单中，选择 Change settings
2. 要关闭自动备份，Enable automated backups 选项请选择 No。保存您的更改；您不需要继续执行下一步骤。
3. 在 Automated Backup 部分中，更改频率、开始时间或要保留的生成。保存您的更改。

手动备份

您可以随时在中启动手动备份，也可以通过运行 AWS CLI create- AWS Management Console backup 命令启动手动备份。手动备份不包括在存储的最多 30 个生成的自动备份中；最多可存储 10 个手动备份，并且必须手动从 Amazon S3 中将其删除。

在创建新的 AWS OpsWorks for Chef Automate 服务器手动备份时，可以添加标签。有关如何在创建手动备份时添加标签的更多信息，请参阅向新备份添加标签 (CLI)。

要在中执行手动备份 AWS Management Console

1. 在 Chef Automate servers 页面上，选择您要备份的服务器。
2. 在服务器的属性页面上，在左侧导航窗格中选择 Backups。
3. 选择创建备份。
4. 当页面在备份的 Status 列中显示绿色复选标记时，手动备份完成。

要在中执行手动备份 AWS CLI

- 要启动手动备份，请运行以下 AWS CLI 命令。

```
aws opsworks-cm --region region name create-backup --server-name "Chef server name"
 --description "optional descriptive string"
```

删除备份

删除某个备份会从存储该备份的 S3 存储桶中永久删除它。

要在中删除备份 AWS Management Console

1. 在 Chef Automate servers 页面上，选择您要备份的服务器。
2. 在服务器的属性页面上，在左侧导航窗格中选择 Backups。
3. 选择您要删除的备份，然后选择 Delete backup。您一次只能选择一个备份。
4. 出现确认删除提示时，选中 Delete the backup, which is stored in an S3 bucket 的复选框，然后选择 Yes, Delete。

要在中删除备份 AWS CLI

- 要删除备份，请运行以下 AWS CLI 命令，`--backup-id`替换为要删除的备份的 ID。Backup ID 的格式为 *ServerName-yyyymmddhhmmssss* s。例如，**test-chef-server-20171218132604388**。

```
aws opsworks-cm --region region name delete-backup --backup-id ServerName-
yyyyMMddHHmmssSSS
```

从 Backup 中恢复 AWS OpsWorks for Chef Automate 服务器

> **Important**
>
> AWS OpsWorks for Chef Automate 已于 2024 年 5 月 5 日停用，新客户和现有客户均已禁用。我们建议现有客户迁移到 Chef SaaS 或其他替代解决方案。如果您有任何疑问，可以通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

浏览完可用备份后，您可以选择恢复 AWS OpsWorks for Chef Automate 服务器的时间点。服务器备份仅包含配置管理软件持久性数据 (说明书、注册节点等)。对服务器执行就地恢复 (即将现有 AWS

OpsWorks for Chef Automate 服务器恢复到新的 EC2 实例）会重新注册在备份时注册的用于恢复服务器的节点，如果恢复成功且恢复的 AWS OpsWorks for Chef Automate 服务器状态为，则将流量切换到新实例。`Healthy`还原到新创建的 AWS OpsWorks for Chef Automate 服务器不会维护节点连接。还原一个服务器并不会更新 Chef 软件的次要版本；它将应用与您所选的备份中相同的可用 Chef 版本和配置管理数据。

还原服务器通常比创建新服务器花费更多的时间；时间取决于您选择的备份大小。还原完成后，旧 EC2 实例仍处于 `Running` 或 `Stopped` 状态，但只是暂时的。它最终被终止。

在此版本中，您可以使用在中 AWS CLI 恢复 Chef 服务器 AWS OpsWorks for Chef Automate。

> **Note**
>
> 您还可以运行 restore-server 命令来更改当前实例类型，或者还原或设置您的 SSH 密钥 (如果该密钥丢失或泄露)。

从备份中还原服务器

1. 在中 AWS CLI，运行以下命令以返回可用备份及其 ID 的列表。请记下要使用的备份的 ID。Backup ID 的格式为 *`myServerName-yyyymmddhhmmssss`* s。

   ```
   aws opsworks-cm --region region name describe-backups
   ```

2. 运行以下命令。

   ```
   aws opsworks-cm --region region name restore-server --backup-id "myServerName-
   yyyyMMddHHmmssSSS" --instance-type "Type of instance" --key-pair "name of your EC2
    key pair" --server-name "name of Chef server"
   ```

   示例如下：

   ```
   aws opsworks-cm --region us-west-2 restore-server --backup-id
    "MyChefServer-20161120122143125" --server-name "MyChefServer"
   ```

3. 等待直到还原操作完成。

中的系统维护 AWS OpsWorks for Chef Automate

> **Important**
>
> AWS OpsWorks for Chef Automate 已于 2024 年 5 月 5 日停用，新客户和现有客户均已禁用。我们建议现有客户迁移到 Chef SaaS 或其他替代解决方案。如果您有任何疑问，可以通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

强制性的系统维护可确保 Chef Server 和 Chef Automate Server 的最新次要版本，包括安全更新，始终在 AWS OpsWorks for Chef Automate 服务器上运行。每周至少需要进行一次系统维护。如果需要 AWS CLI，您可以使用，配置每日自动维护。除了定期的 AWS CLI 系统维护外，您还可以使用按需执行系统维护。

如果有新的 Chef 软件次要版本可用，并通过了 AWS 测试，系统维护机制会自动在服务器上更新 Chef Automate 和 Chef Server 的次要版本。AWS 会进行广泛的测试，以验证 Chef 升级是否可以投入生产，并且不会中断现有客户环境，因此 Chef 软件的发布与其应用到 Chef Automate 现有 OpsWorks 服务器的可用性之间可能会存在延迟。要按需更新 Chef 软件的可用次要版本，请参阅本主题中的 按需启动系统维护。

系统维护从在维护过程中执行的备份启动新实例，这有助于减少经过定期维护的降级或受损 Amazon EC2 实例带来的风险。

> **Important**
>
> 系统维护会删除您已添加到 AWS OpsWorks for Chef Automate 服务器的任何文件或自定义配置。有关如何修复配置或文件丢失的更多信息，请参阅本主题中的在维护之后恢复自定义配置和文件。

主题

确保节点信任 AWS OpsWorks 证书颁发机构

> **Note**
>
> 如果您在 AWS OpsWorks for Chef Automate 服务器上使用自定义域和证书，则无需执行本节中的步骤。

您在 AWS OpsWorks for Chef Automate 服务器上管理的节点必须使用证书向服务器进行身份验证。在系统维护期间， AWS OpsWorks 替换服务器实例，并通过证书颁发机构 (CA) 重新生成新 AWS OpsWorks 证书。要在维护完成后自动恢复与托管节点的通信，节点应信任入门套件附带且托管在支持的区域的 AWS OpsWorks CA AWS OpsWorks for Chef Automate。当您使用 AWS OpsWorks CA 在节点和服务器之间建立信任时，节点将在维护后重新连接到新的服务器实例。如果您使用中所述的 EC2 `userdata` 脚本添加 EC2 节点[在中自动添加节点 AWS OpsWorks for Chef Automate](#)，则节点已配置为信任 AWS OpsWorks CA。

- 对于基于 Linux 的节点，CA 的 S3 存储桶位置为 `https://opsworks-cm-${REGION}-prod-default-assets.s3.amazonaws.com/misc/opsworks-cm-ca-2020-root.pem`。可 AWS OpsWorks 信 CA 必须存储在路径中`/etc/chef/opsworks-cm-ca-2020-root.pem`。
- 对于基于 Windows 的节点，CA 的 S3 存储桶位置为 `https://opsworks-cm-$env:AWS_REGION-prod-default-assets.s3.amazonaws.com/misc/opsworks-cm-ca-2020-root.pem`。C AWS OpsWorks A 必须存储在 Chef 的根文件夹中；例如，`C:\chef\opsworks-cm-ca-2020-root.pem`

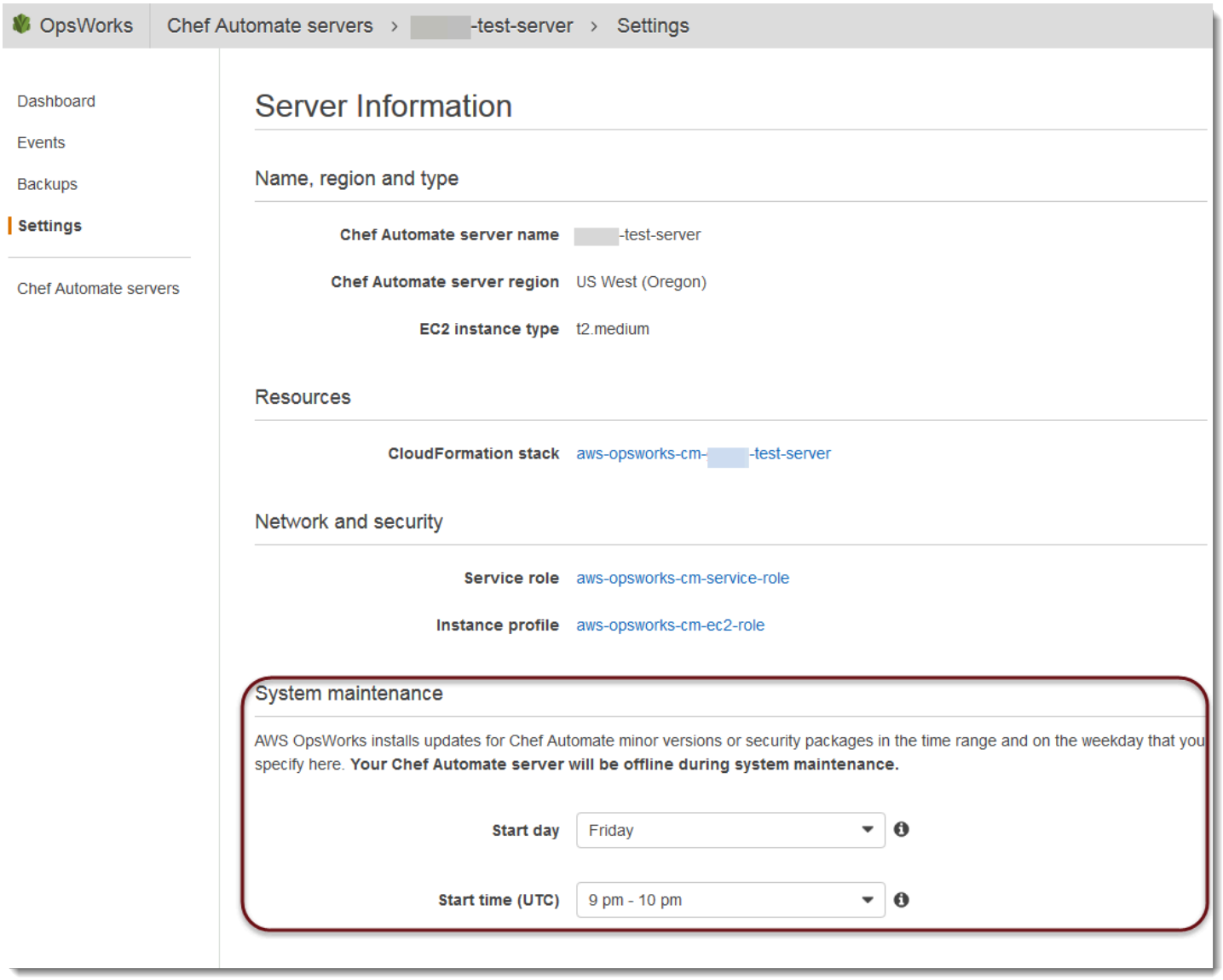
在这两个路径中，区域变量解析为以下之一。

- `us-east-2`
- `us-east-1`
- `us-west-1`
- `us-west-2`
- `ap-northeast-1`
- `ap-southeast-1`
- `ap-southeast-2`
- `eu-central-1`
- `eu-west-1`

配置系统维护

创建新 AWS OpsWorks for Chef Automate 服务器时，可以按协调世界时 (UTC) 配置工作日和时间，以便开始系统维护。维护在您指定的小时开始。由于您可以预见到服务器会在系统维护期间脱机，因此请选择正常工作时间中对服务器需求较低的时间。进行维护时，服务器的状态为 UNDER_MAINTENANCE。

您还可以通过更改 AWS OpsWorks for Chef Automate 服务器的 “设置” 页面的 “系统维护” 区域中的设置来更改现有服务器上的系统维护设置，如以下屏幕截图所示。



在 System maintenance (系统维护) 部分中，设置您希望系统维护开始的日期和时间。

使用配置系统维护 AWS CLI

您还可以使用 AWS CLI配置系统维护自动开始时间。如果需要， AWS CLI 可以省略三个字符的工作日前缀，从而配置每日自动维护。

在 `create-server` 命令中，指定创建服务器实例的要求 (例如实例类型、实例配置文件 ARN 和服务角色 ARN) 后，向您的命令添加 `--preferred-maintenance-window` 参数。在以下 `create-server` 示例中，`--preferred-maintenance-window` 设置为 `Mon:08:00`，这意味着您已设置维护在每周一上午 8:00 开始。(UTC)。

```
aws opsworks-cm create-server --engine "Chef" --engine-model "Single" --
engine-version "12" --server-name "automate-06" --instance-profile-arn
 "arn:aws:iam::1019881987024:instance-profile/aws-opsworks-cm-ec2-role"
 --instance-type "t2.medium" --key-pair "amazon-test" --service-role-arn
 "arn:aws:iam::044726508045:role/aws-opsworks-cm-service-role" --preferred-maintenance-
window "Mon:08:00"
```

在 `update-server` 命令中，您可以根据需要单独更新 `--preferred-maintenance-window` 值。在以下示例中，维护时段设置为周五晚上 6:15 (UTC)。

```
aws opsworks-cm update-server --server-name "shiny-kitchen" --preferred-maintenance-
window "Fri:18:15"
```

将维护时段的开始时间更改为每天晚上 6:15 (UTC)，请忽略三个字符的工作日前缀，如以下示例中所示。

```
aws opsworks-cm update-server --server-name "shiny-kitchen" --preferred-maintenance-
window "18:15"
```

有关使用设置首选系统维护时段的更多信息 AWS CLI，请参阅创建服务器和更新服务器。

按需启动系统维护

要在配置的每周或每日自动维护之外按需启动系统维护，请运行以下 AWS CLI 命令。您不能在 AWS Management Console中启动按需维护。

```
aws opsworks-cm start-maintenance --server-name server_name
```

有关此命令的更多信息，请参阅 start-maintenance。

在维护之后恢复自定义配置和文件

系统维护人员可以删除或更改您已添加到 AWS OpsWorks for Chef Automate 服务器的自定义文件或配置。

如果在维护运行之后，您的 Chef 服务器缺少了使用 RunCommand 或 SSH 添加的文件或设置，您可以使用 Amazon 系统映像 (AMI) 来启动新的 Amazon EC2 实例。提供的 AMI 是从服务器的维护前配置生成的。

新的实例与维护前的 Chef 服务器处于相同状态，且应包含您缺少的文件和设置。

> **Important**
>
> 您无法使用新实例来恢复服务器；该实例无法作为 Chef 服务器运行。您只能使用实例来恢复您的文件和配置设置。

要从 AMI 启动 EC2 实例，请在 Amazon EC2 控制台中打开 Launch 向导，选择 My AMIs，然后选择具有您的服务器名称的 AMI。按照 Amazon EC2 向导中的步骤执行，就像您启动任何其他实例一样操作。

中的合规性扫描 AWS OpsWorks for Chef Automate

> **Important**
>
> AWS OpsWorks for Chef Automate 已于 2024 年 5 月 5 日停用，新客户和现有客户均已禁用。我们建议现有客户迁移到 Chef SaaS 或其他替代解决方案。如果您有任何疑问，可以通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

合规性扫描可让您根据预定义的策略（也称为规则）跟踪您的基础设施中托管节点的合规性。合规性视图使您可以定期审核您的应用程序是否存在漏洞和不合规的配置。Chef 提供了超过 100 个预定义的合规性配置文件 - 适用于特定节点配置的规则集 - 您可以在合规性扫描中使用它们。您也可以使用 Chef InSpec 语言创建自己的自定义配置文件。

如果您的服务器尚未运行 Chef Automate 2.0，您可以通过安装 Audit 说明书手动设置 Chef Compliance。

> **Note**
>
> 与 AWS OpsWorks for Chef Automate 服务器关联的节点上支持的 Chef Infra 客户端代理软件 (`chef-client`) 的最低版本为 13。 x。我们建议运行最新、最稳定的`chef-client`版本，或至少 14.10.9。

主题

Chef Automate 2.0 中的合规性

如果你的 AWS OpsWorks for Chef Automate 服务器运行的是 Chef Automate 2.0，请使用本节中的过程设置 Chef 合规性。

使用 Chef Automate 2.0 运行合规性扫描作业

Chef Automate 2.0 包括 Chef InSpec 合规性扫描功能，以前需要手动设置和配置食谱。你可以在运行 Chef Automate 2.0 的 AWS OpsWorks for Chef Automate 服务器上运行扫描作业。作业可以立即运行（一次），也可以安排在以后运行，或者安排以特定的时间间隔运行（如每天或每隔两小时）。扫描作业的结果发送到合规性报告。您可以在 Chef Automate 控制面板中查看合规性扫描结果并对其执行操作。要打开 Compliance (合规性) 选项卡并查看报告，请在 Chef Automate 控制面板中的 Scan Jobs (扫描作业) 选项卡上，选择位于托管节点行右侧的 Report (报告)。

要在托管节点上运行扫描任务，您必须具有以下内容。

- 至少有一个合规性配置文件安装在您的命名空间中。
- 至少一个目标节点，手动添加或 EC2 实例自动添加。

在中 AWS OpsWorks for Chef Automate，以下目标支持扫描作业。

- 手动添加的节点

- aws-ec2 实例
- 亚马逊云科技 区域

有关如何运行扫描作业的详细说明，请参阅 Chef 文档中的 Chef Automate 扫描作业。

（可选，Chef Automate 2.0）使用 Audit 说明书设置合规性

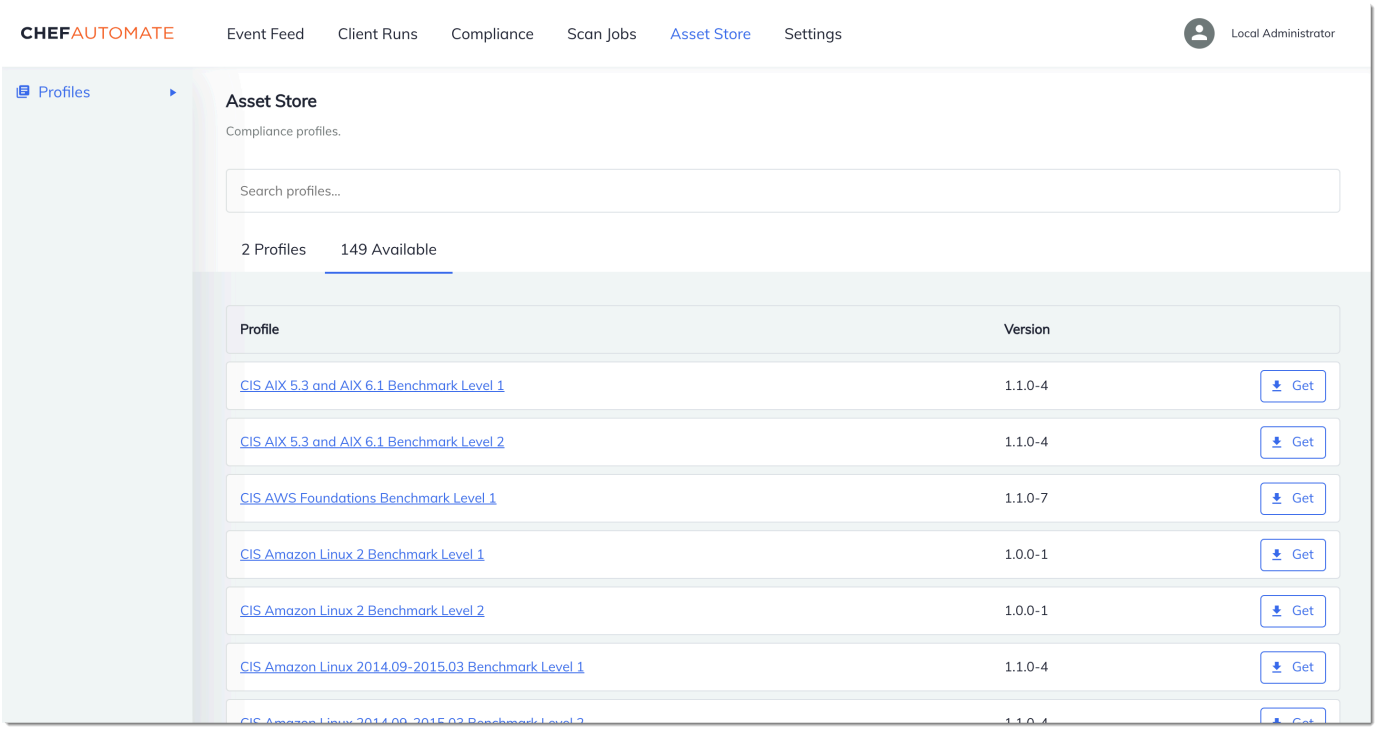
您可以在任何 AWS OpsWorks for Chef Automate 服务器上配置合规性。启动 AWS OpsWorks for Chef Automate 服务器后，您可以从 Chef Automate 控制面板安装配置文件，或者在 Policyfile.rb 策略文件中向 Audit 说明书属性添加所需的配置文件。初学者工具包中包含一个预填充的 Policyfile.rb 文件。

在将配置文件作为 Audit 说明书的属性编辑 Policyfile.rb 之后，运行 chef push 命令以将 Audit 说明书以及在 Policyfile.rb 中指定的其他说明书上传到您的 Chef Automate 服务器。安装 Audit cookbook 还会为 Chef 安装 gem InSpec，这是一款由 Chef 制作的开源测试和审计框架。对于 Chef Automate 2.0，选择 Audit 说明书的版本 7.1.0 或更高版本。该 InSpec 宝石必须是 2.2.102 或更高版本。

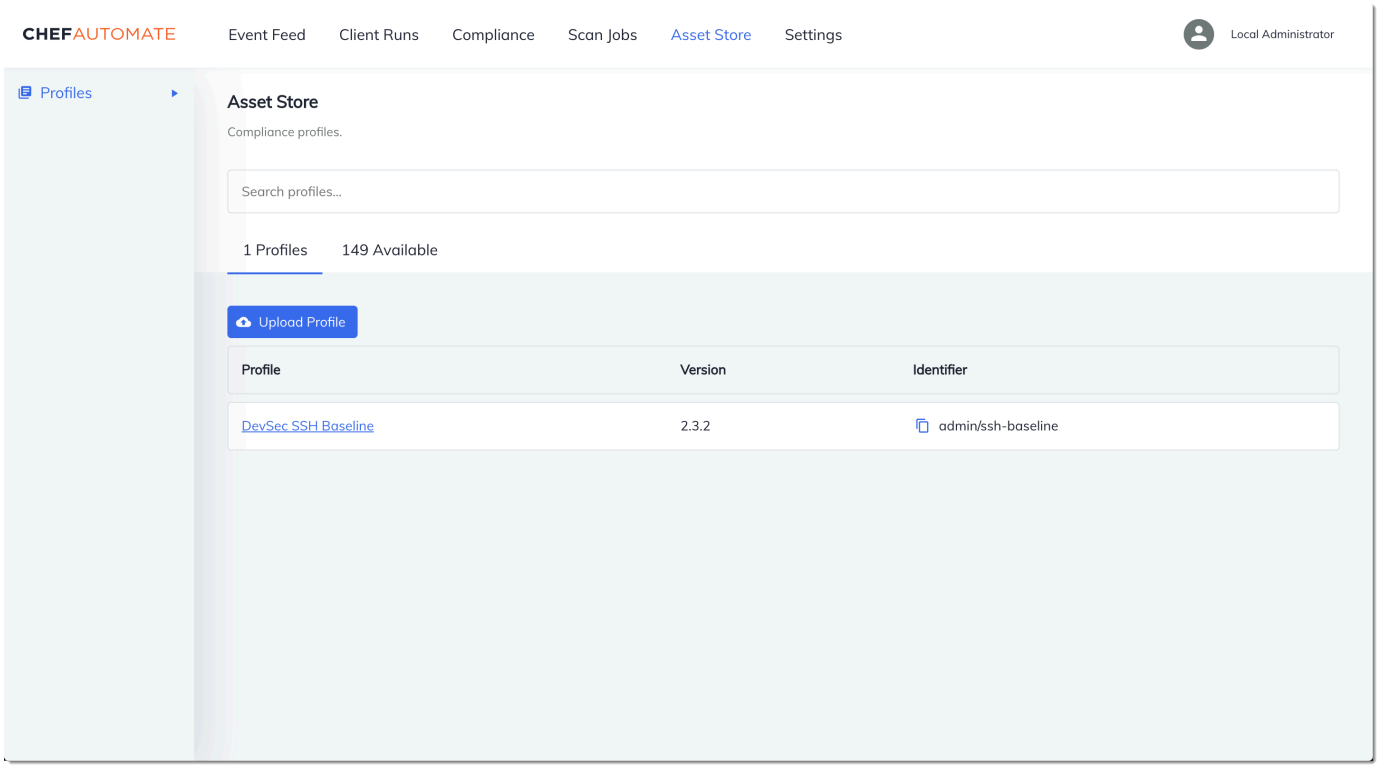
本部分的说明向您展示如何实施 opsworks-audit 说明书。审计食谱从 Chef Automate 服务器下载指定的配置文件，根据 DevSec SSH 基准配置文件评估节点，并在每次chef-client运行时报告合规性扫描的结果。

安装合规性配置文件

1. 如果您尚未登录到 Chef Automate 基于 Web 的控制面板，请先登录。使用在创建 AWS OpsWorks for Chef Automate 服务器的过程中下载初学者工具包时收到的凭证。
2. 在 Chef Automate 控制面板中，选择 Asset Store (资产存储) 选项卡。



3. 选择 Available (可用) 选项卡以查看预定义的配置文件。
4. 浏览配置文件列表。选择一个与至少一个受管节点的操作系统和配置相匹配的配置文件。要查看有关配置文件的详细信息，包括配置文件的目标违规和基本规则代码的说明，请选择配置文件条目右侧的 >。您可以选择多个配置文件。如果您要在入门套件中设置示例，请选择 DevSec SSH 基准。



5. 要在您的 Chef Automate 服务器上安装所选配置文件，请选择 Get。
6. 安装配置文件后，它们会显示在 Chef Automate 控制面板的 Profiles (配置文件) 选项卡中。

使用 **Policyfile.rb** 安装说明书

1. 查看初学者工具包中的 `Policyfile.rb`，可以看到 Audit 说明书的属性在 `['profiles']` 中指定 `ssh-baseline` 配置文件。

```
# Define audit cookbook attributes
default["opsworks-demo"]["audit"]["reporter"] = "chef-server-automate"
default["opsworks-demo"]["audit"]["profiles"] = [
  {
    "name": "DevSec SSH Baseline",
    "compliance": "admin/ssh-baseline"
  }
]
```

2. 下载并安装 `Policyfile.rb` 中定义的说明书。

```
chef install
```

所有说明书的版本均记录在说明书的 metadata.rb 文件中。每次更改说明书后，必须在 metadata.rb 中提高该说明书的版本。

3. 将策略 opsworks-demo（在 Policyfile.rb 中定义）推送到服务器。

```
chef push opsworks-demo
```

4. 验证策略的安装。运行以下命令。

```
chef show-policy
```

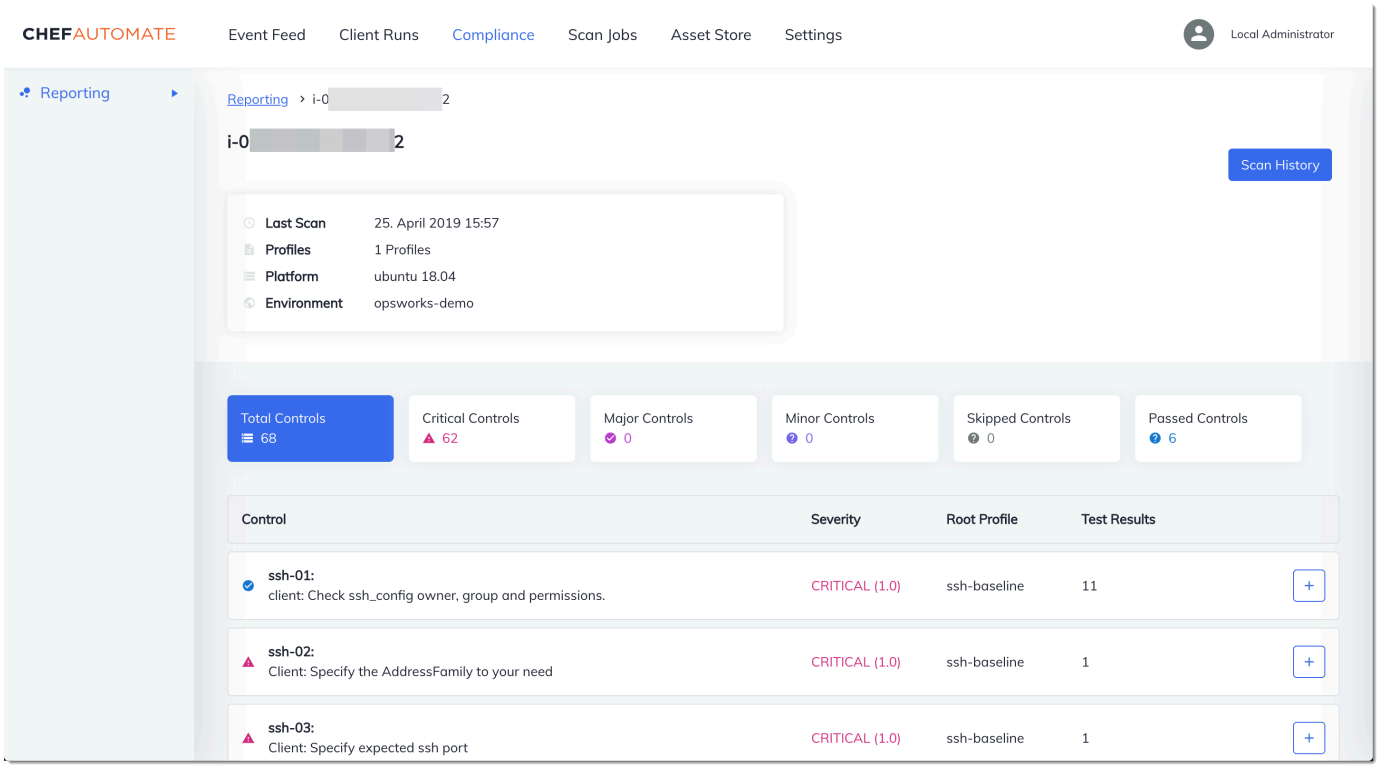
结果应与以下内容类似：

```
opsworks-demo-webserver
=======================
* opsworks-demo:  ec0fe46314
```

5. 如果您还没有将节点添加到服务器进行管理，请添加。要将您的第一个节点连接到 AWS OpsWorks for Chef Automate 服务器，请使用此入门套件中包含的userdata.sh脚本。它使用 AWS OpsWorks AssociateNode API 将节点连接到您的服务器。

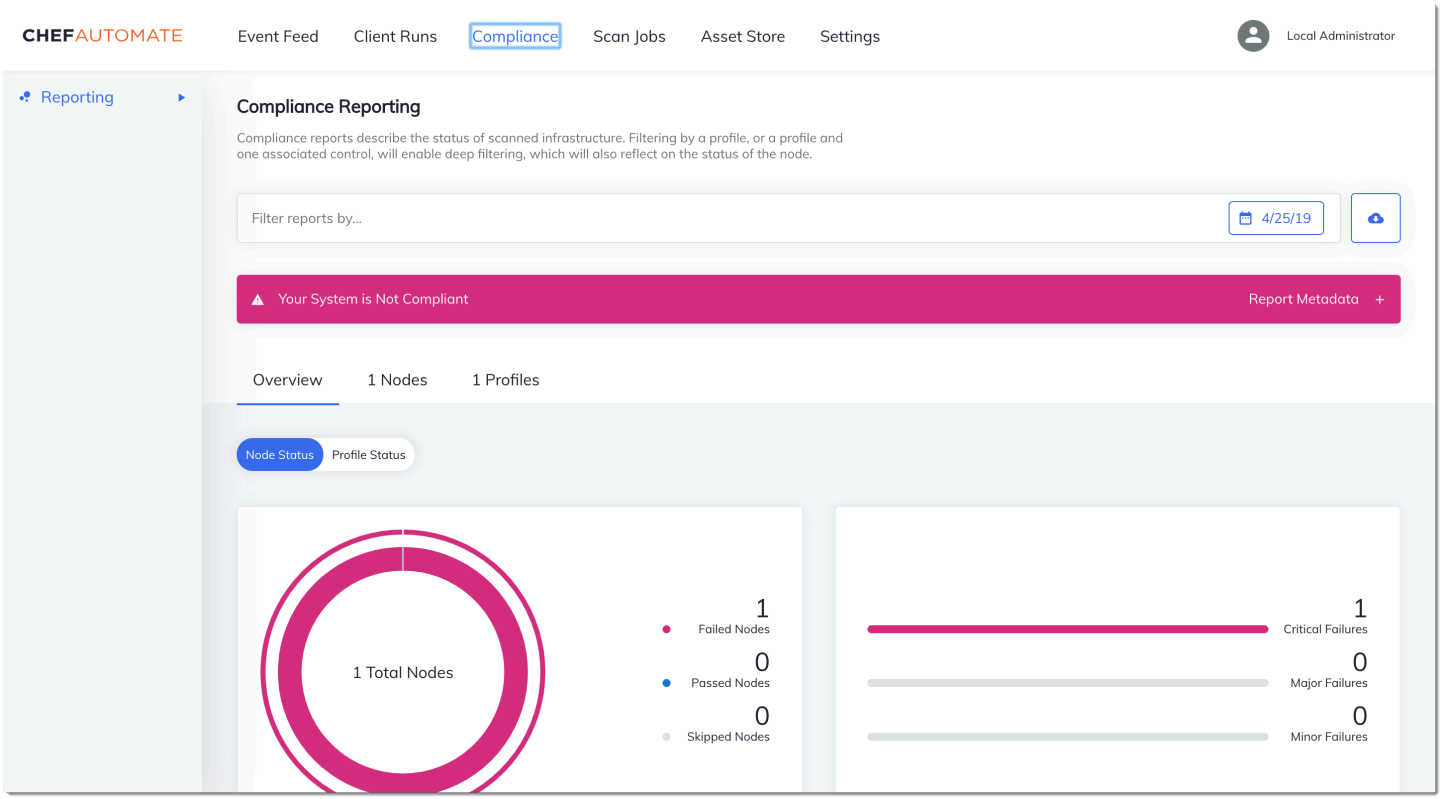
   您可以按照 在中自动添加节点 AWS OpsWorks for Chef Automate 中的步骤自动关联节点，或按照 单独添加节点 中的步骤，每次添加一个节点。

6. 您为节点更新运行列表后，chef-client 代理会在下次运行时运行您指定的配方。默认情况下每 1800 秒 (30 分钟) 发生一次。运行后，您可以从 Chef Automate 控制面板的 Compliance 选项卡中查看合规性结果并对其执行操作。

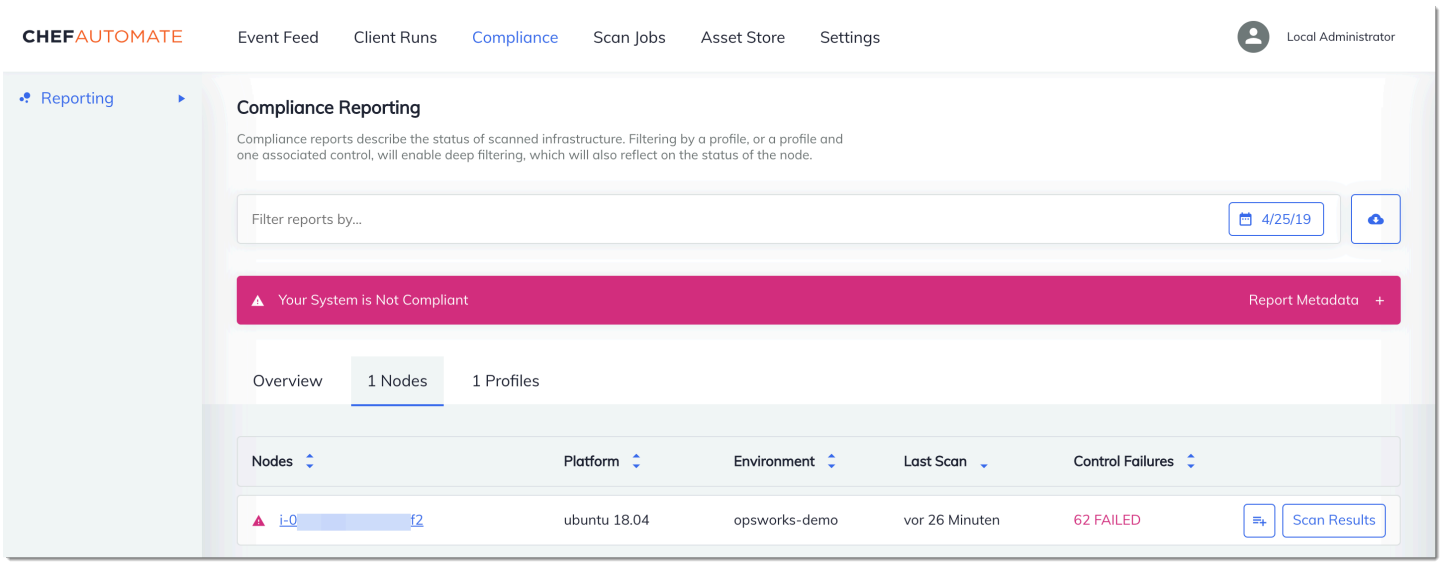


运行合规性扫描

在配置节点运行列表并首次运行代理之后，应该很快会在 Chef Automate 控制面板中看到合规性扫描结果。



在 Chef Automate 控制面板中，选择 Compliance 选项卡。在左侧导航窗格中，选择 Reporting。选择 Profiles 选项卡，选择 Scan Results，然后选择扫描失败的节点，以详细了解导致节点失败的规则。



通常，您会看到不合规的扫描结果，因为新节点尚未满足 DevSec SSH 基准配置文件中的所有规则。DevSec Hardening Framework 是一个基于社区的项目，它提供了解决违反 DevSec SSH 基准配置文件中规则的问题的食谱。

（可选）解决不合规结果

入门套件包括一本开源食谱ssh-hardening，你可以运行它来修复针对 DevSec SSH Bas eline 配置文件运行的不合规结果。

> **Note**
>
> 该ssh-hardening食谱会对您的节点进行更改以符合 DevSec SSH 基准规则。在任何生产节点上运行本食谱之前，请在Chef Automate控制台中查看有关 DevSec SSH基准配置文件的详细信息，以了解该食谱针对的规则违规行为。在任何生产节点上运行开源 ssh-hardening 说明书之前，请查看此说明书的信息。

运行 **ssh-hardening** 说明书

1. 在文本编辑器中，将 ssh-hardening 说明书追加到 Policyfile.rb 的运行列表之后。Policyfile.rb 运行列表应与以下内容匹配。

   ```
   run_list  'chef-client', 'opsworks-webserver', 'audit', 'ssh-hardening'
   ```

2. 更新 Policyfile.rb 并将其推送至您的 AWS OpsWorks for Chef Automate 服务器。

   ```
   chef update Policyfile.rb
      chef push opsworks-demo
   ```

3. 与 opsworks-demo 策略关联的节点将自动更新运行列表，并在下次运行 chef-client 时应用 ssh-hardening 说明书。

   由于您在使用 chef-client 说明书，将以固定间隔检查节点 (默认情况下为 30 分钟)。下次签入时，该ssh-hardening食谱将运行，它有助于提高节点安全性，以满足 DevSec SSH Baseline 配置文件的规则。

4. 在首次运行 ssh-hardening 说明书之后，请等待 30 分钟，以便合规性扫描再次运行。在 Chef Automate 控制面板中查看结果。应解决在首次运行 DevSec SSH 基准扫描时出现的不合规结果。

Chef Automate 1.x 中的合规性

如果你的 AWS OpsWorks for Chef Automate 服务器正在运行 Chef Automate 1. x，使用本节中的过程设置 Chef 合规性。

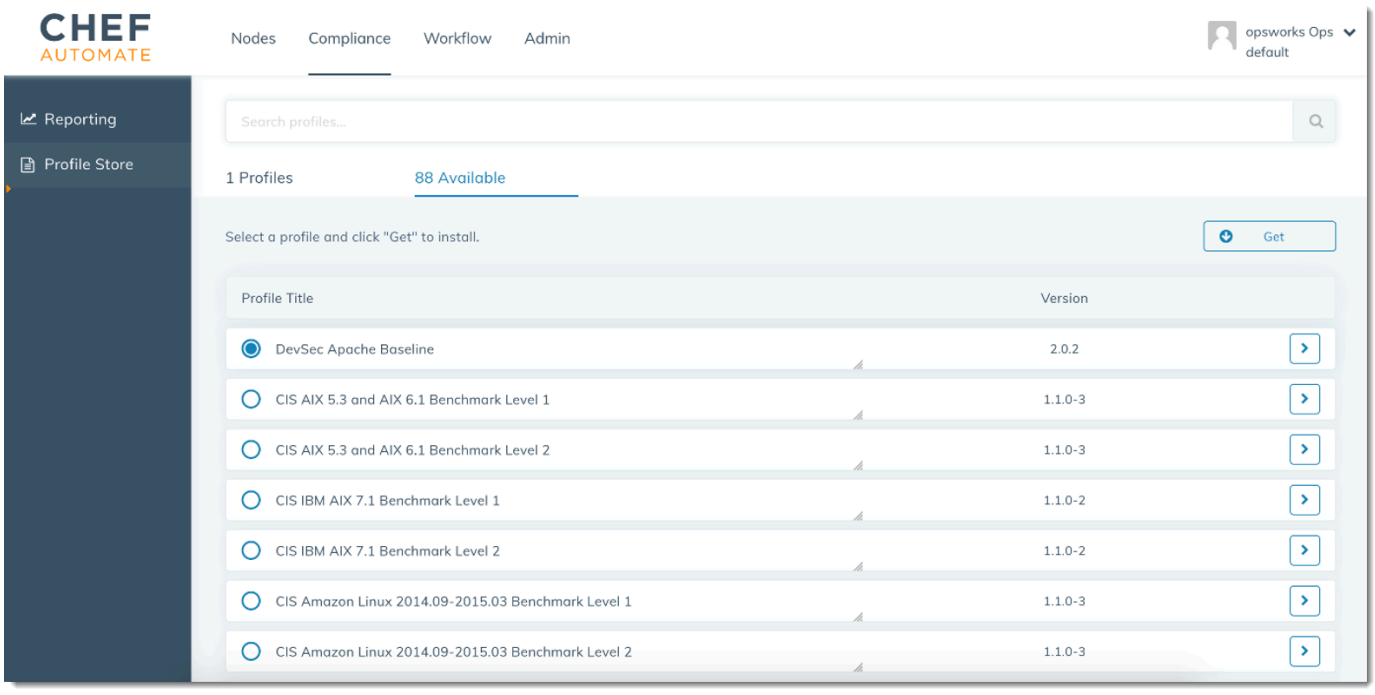
（可选，Chef Automate 1.x）设置 Chef Compliance

你可以在任何 AWS OpsWorks for Chef Automate 服务器上配置 Chef 合规性。启动 AWS OpsWorks for Chef Automate 服务器后，在 Chef Automate 控制面板的配置文件中选择要运行的配置文件。安装配置文件后，请运行 `berks` 命令以将 Audit 说明书上传到您的 Chef Automate 服务器。安装 Audit cookbook 还会安装 gem InSpec，这是一款由 Chef 制作的开源测试框架，可让您将自动化测试集成到部署管道的任何阶段。对于 Chef Automate 1.x，选择 Audit 说明书的版本 5.0.1 或更高版本。 InSpec Gem 必须是 1.24.0 或更高版本。

入 AWS OpsWorks for Chef Automate 门套件包括一本包装食谱`opsworks-audit`，它可以为你下载并安装正确版本的 Chef's Audit 食谱。`opsworks-audit`本食谱还指示`chef-client`代理根据你在本主题后面从 Chef 的合规性控制台安装的 DevSecSSH 基准配置文件来评估节点。您可以使用任一说明书设置合规性，以满足您的偏好。本部分的说明向您展示如何实施 `opsworks-audit` 说明书。

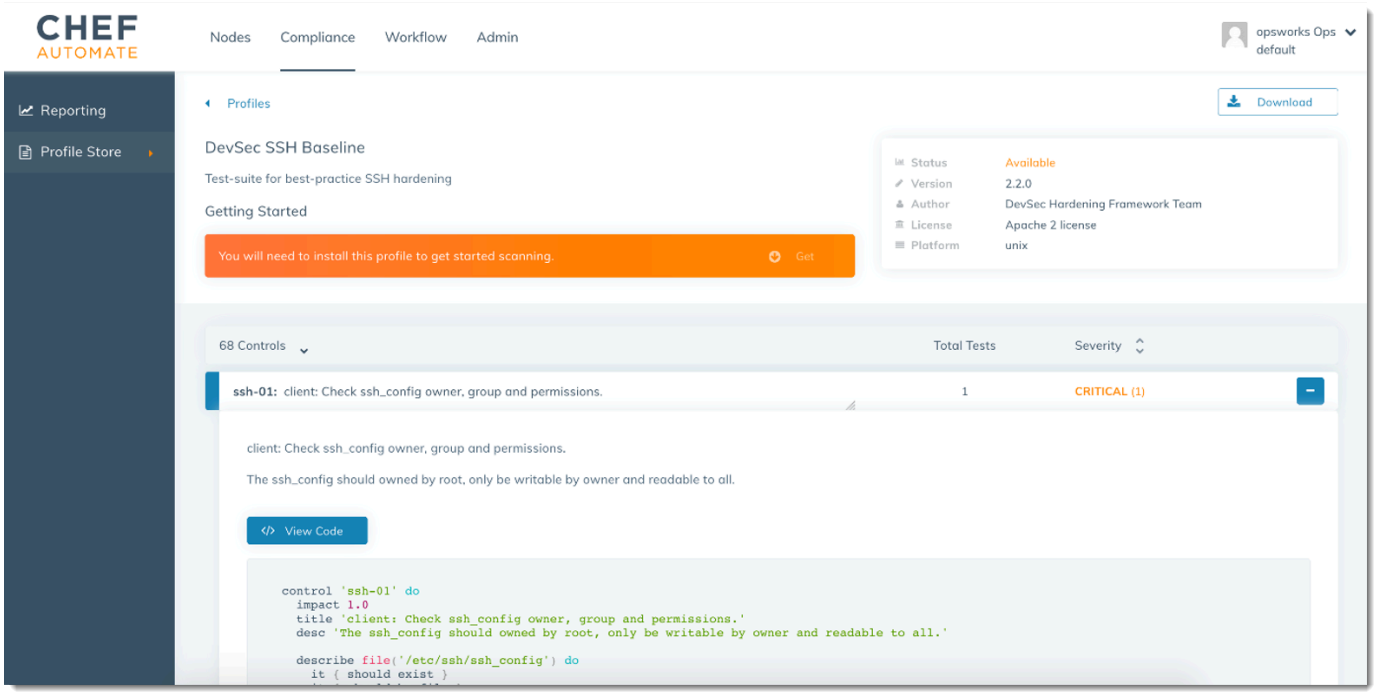
安装合规性配置文件

1. 如果您尚未登录到 Chef Automate 基于 Web 的控制面板，请先登录。使用您在创建 AWS OpsWorks for Chef Automate 服务器时下载入门套件时收到的凭据。
2. 在 Chef Automate 控制面板中，选择 Compliance 选项卡。



3. 在左侧导航栏中，选择 Profile Store，然后选择 Available 选项卡以查看预定义的配置文件。

4. 浏览配置文件列表。选择一个与至少一个受管节点的操作系统和配置相匹配的配置文件。要查看有关配置文件的详细信息，包括配置文件的目标违规和基本规则代码的说明，请选择配置文件条目右侧的 >。您可以选择多个配置文件。



5. 要在您的 Chef Automate 服务器上安装所选配置文件，请选择 Get。
6. 下载完成后，请转到下一过程。

安装并设置 **opsworks-audit** 说明书

1. 此步骤是可选的，但它可以节省步骤 6 中将配方添加到节点运行列表的时间。编辑 `roles/opsworks-example-role.rb` 文件，该文件包含在创建 AWS OpsWorks for Chef Automate 服务器时下载的初学者工具包中。添加以下行。最后一行有注释，因为添加 `ssh-hardening` 说明书和配方，在合规性扫描运行之后来解析不合规节点是可选的。

```
run_list(
     "recipe[chef-client]",
     "recipe[apache2]",
     "recipe[opsworks-audit]"
     # "recipe[ssh-hardening]"
       )
```

2. 使用文本编辑器在 Berksfile 中指定所需的说明书。初学者工具包中提供了示例 Berksfile。在此示例中，我们会安装 Chef Infra 客户端 (chef-client) 说明书、apache2 说明书和 opsworks-audit 说明书。您的 Berksfile 应类似于以下内容。

```
source 'https://supermarket.chef.io
     cookbook 'chef-client'
     cookbook 'apache2', '~> 5.0.1'
     cookbook 'opsworks-audit', path: 'cookbooks/opsworks-audit', '~> 1.0.0'
```

所有说明书的版本均记录在说明书的 metadata.rb 文件中。每次更改说明书后，必须在 metadata.rb 中提高该说明书的版本。

3. 运行以下命令，在本地或工作计算机上的 cookbooks 文件夹中下载并安装说明书。

```
berks vendor cookbooks
```

4. 运行以下命令，将分发的说明书上传到您的 AWS OpsWorks for Chef Automate 服务器。

```
knife upload .
```

5. 运行以下命令，显示服务器上当前可用的说明书列表，验证是否安装了 opsworks-audit 说明书。

```
knife cookbook list
```

6. 如果您还没有将节点添加到服务器进行管理，请添加。您可以按照 在中自动添加节点 AWS OpsWorks for Chef Automate 中的步骤自动关联节点，或按照 单独添加节点 中的步骤，每次添加一个节点。编辑节点的运行列表，添加您在步骤 1 中指定的角色 opsworks-example-role。在本示例中，我们编辑 RUN_LIST 脚本中的 userdata 属性，用于自动关联节点。

```
RUN_LIST="role[opsworks-example-role]"
```

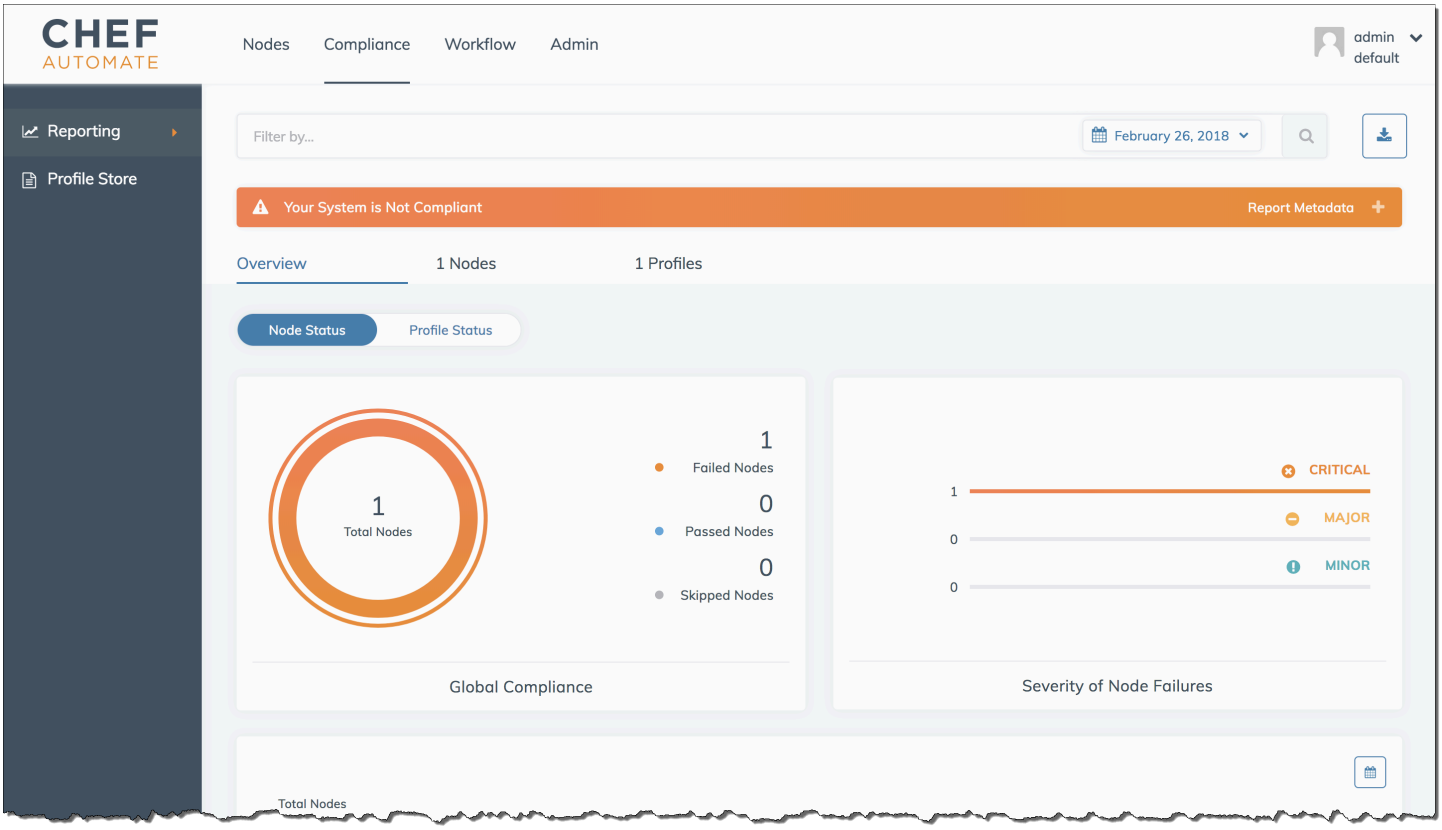
如果您跳过步骤 1，也没有设置角色，请将单个配方的名称添加到运行列表中。保存更改，然后按照 第 3 步：使用自动化关联脚本创建实例 中的步骤，将用户数据脚本应用于 Amazon EC2 实例。

```
RUN_LIST="recipe[chef-client],recipe[apache2],recipe[opworks-audit]"
```

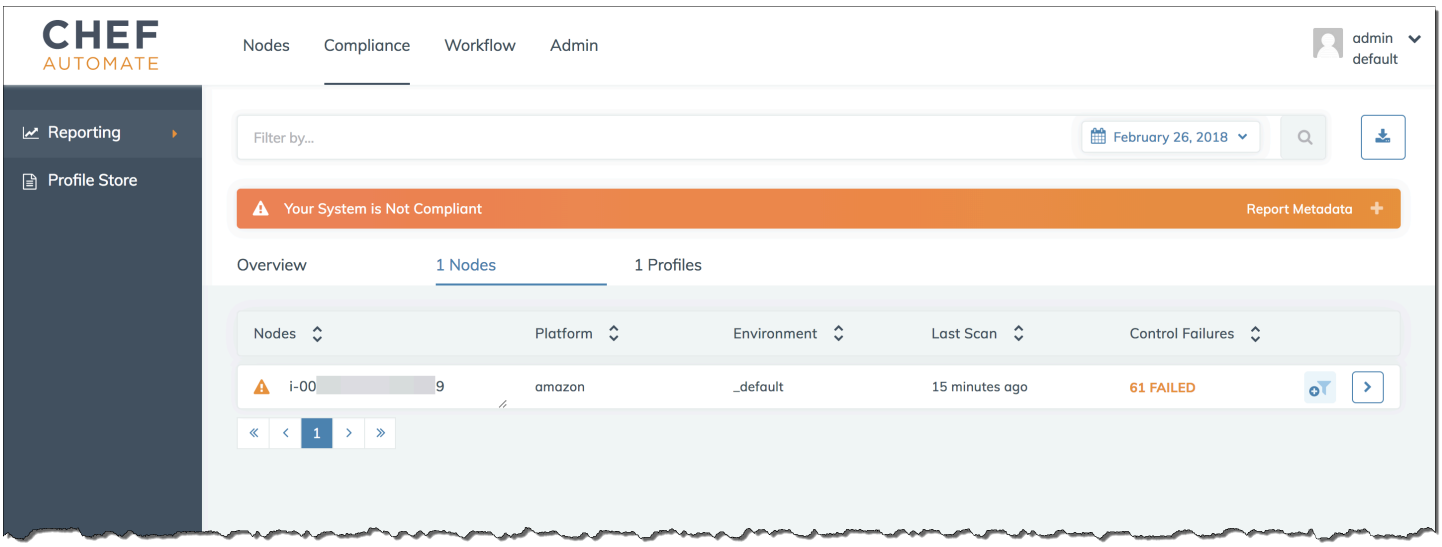
7. 您为节点更新运行列表后，chef-client 代理会在下次运行时运行您指定的配方。默认情况下每 1800 秒 (30 分钟) 发生一次。运行后，合规性结果将显示在 Chef Automate 控制面板中。

运行合规性扫描

在配置节点运行列表并首次运行代理守护程序之后，应该很快会在 Chef Automate 控制面板中看到合规性扫描结果。



在 Chef Automate 控制面板中，选择 Compliance 选项卡。在左侧导航窗格中，选择 Reporting。选择 Profiles 选项卡，选择 Scan Results，然后选择扫描失败的节点，以详细了解导致节点失败的规则。



通常，您会看到不合规的扫描结果，因为新节点尚未满足 DevSec SSH 基准配置文件中的所有规则。DevSec Hardening Framework 是一个基于社区的项目，它提供了解决违反 DevSec SSH 基准配置文件中规则的问题的食谱。

（可选）解决不合规结果

入门套件包括一本开源食谱`ssh-hardening`，你可以运行它来修复针对 DevSec SSH Bas eline 配置文件运行的不合规结果。

> **Note**
>
> 该`ssh-hardening`食谱会对您的节点进行更改以符合 DevSec SSH 基准规则。在任何生产节点上运行本食谱之前，请在Chef Automate控制台中查看有关 DevSec SSH基准配置文件的详细信息，以了解该食谱针对的规则违规行为。在任何生产节点上运行开源 `ssh-hardening` 说明书之前，请查看此说明书的信息。

运行 **ssh-hardening** 说明书

1. 在文本编辑器中，将 `ssh-hardening` 说明书附加到您的 Berksfile。您的 Berksfile 应类似于以下内容。

   ```
   source 'https://supermarket.chef.io'
        cookbook 'chef-client'
        cookbook 'apache2', '~> 5.0.1'
        cookbook 'opsworks-audit', path: 'cookbooks/opsworks-audit', '~> 1.0.0' #
    optional
        cookbook 'ssh-hardening'
   ```

2. 运行以下命令，将 `ssh-hardening` 说明书下载到本地说明书文件夹，然后再将它上传到 AWS OpsWorks for Chef Automate 服务器。

   ```
   berks vendor cookbooks
   knife upload .
   ```

3. 根据 安装并设置 opsworks-audit 说明书 步骤 1 和 6 的介绍，将 `ssh-hardening` 配方添加到您的节点运行列表中。

   如果您更新 `opsworks-example-role.rb` 文件，请运行以下命令，将改动上传到您的服务器。

```
knife upload .
```

如果您直接更新运行列表，请运行以下命令上传改动。节点名称通常为实例 ID。

```
knife node run_list add <node name> 'recipe[ssh-hardening]'
```

4. 由于您在使用 `chef-client` 说明书，将以固定间隔检查节点 (默认情况下为 30 分钟)。下次签入时，该`ssh-hardening`食谱将运行，它有助于提高节点安全性，以满足 DevSec SSH Baseline 配置文件的规则。
5. 在首次运行 `ssh-hardening` 说明书之后，请等待 30 分钟，以便合规性扫描再次运行。在 Chef Automate 控制面板中查看结果。应解决在首次运行 DevSec SSH 基准扫描时出现的不合规结果。

更新合规性

在 AWS OpsWorks for Chef Automate 服务器上，合规性功能由您的定期系统维护自动更新。随着 Chef Automate、Chef Infra Server 和 Chef InSpec 的更新版本可用于您的 AWS OpsWorks for Chef Automate 服务器，您可能需要检查和更新服务器上运行的 Audit cookbook 和 Chef InSpec gem 的支持版本。您已经安装在 AWS OpsWorks for Chef Automate 服务器上的配置文件不会作为维护的一部分进行更新。

社区和自定义合规性配置文件

Chef 目前包括超过 100 个合规性扫描配置文件。您可以将社区和自定义配置文件添加到列表中，然后根据这些配置文件下载和运行合规性扫描，就像对程序附带的配置文件一样。Chef Supermarket 提供了基于社区的合规性配置文件。自定义配置文件是基于 Ruby 的程序，其中包含用于指定扫描规则的控件文件夹。

另请参阅

- Chef Compliance 公告博客文章
- Chef Automate 合规性在线培训
- 厨师 InSpec网站
- 厨师 InSpec 教程

取消节点与服务器的关联 AWS OpsWorks for Chef Automate

> **Important**
>
> AWS OpsWorks for Chef Automate 已于 2024 年 5 月 5 日停用，新客户和现有客户均已禁用。我们建议现有客户迁移到 Chef SaaS 或其他替代解决方案。如果您有任何疑问，可以通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

本节介绍如何将托管节点从 AWS OpsWorks for Chef Automate 服务器的管理中解除关联或移除。此操作在命令行上执行；您无法在 AWS OpsWorks for Chef Automate 管理控制台中取消关联节点。目前，该 AWS OpsWorks for Chef Automate API 不允许批量移除多个节点。本节中的命令将一次对一个节点取消关联。

我们建议在删除 Chef 服务器之前解除节点与该服务器的关联，这样节点就能继续工作而不用尝试重新连接服务器。为此，请运行`disassociate-node` AWS CLI 命令。

解除节点的关联

1. 在中 AWS CLI，运行以下命令取消节点的关联。*Node_name* 是要取消关联的节点的名称；对于 Amazon EC2 实例，这是 ID。*Server_name* 是要取消节点与其关联的 Chef 服务器的名称。`--engine-attributes` 指定您的默认 `CHEF_AUTOMATE_ORGANIZATION` 名称。这三个参数全都是必需的。

   `--region` 参数不是必需的，除非您要取消节点与不在您的默认区域内的 Chef 服务器的关联。

   ```
   aws opsworks-cm --region Region_name disassociate-node --node-
   name Node_name --server-name Server_name --engine-attributes
    "Name=CHEF_AUTOMATE_ORGANIZATION,Value='default'"
   ```

   以下命令是一个示例。

   ```
   aws opsworks-cm --region us-west-2 disassociate-node --node-name
    i-0010zzz00d66zzz90 --server-name opsworkstest --engine-attributes
    "Name=CHEF_AUTOMATE_ORGANIZATION,Value='default'"
   ```

2. 请耐心等待，直到响应消息指示已完成关联断开。

成功取消节点与 AWS OpsWorks for Chef Automate 服务器的关联后，该节点可能仍会显示在 Chef Automate 仪表板中。默认情况下，Chef 会对节点状态信息实施保留，并在几天后自动清除该节点。

有关如何删除 AWS OpsWorks for Chef Automate 服务器的更多信息，请参见删除 AWS OpsWorks for Chef Automate 服务器。

相关主题

以下 AWS 博客文章提供了有关使用 Auto Scaling 群组或在多个账户中自动将节点与 Chef Automate 服务器关联的更多信息。

- 使用 AWS fo OpsWorks r Chef Automate 通过 Auto Scaling 管理 EC2 实例
- OpsWorks for Chef Automate — 自动引导不同账户中的节点

删除 AWS OpsWorks for Chef Automate 服务器

> **Important**
>
> AWS OpsWorks for Chef Automate 已于 2024 年 5 月 5 日停用，新客户和现有客户均已禁用。我们建议现有客户迁移到 Chef SaaS 或其他替代解决方案。如果您有任何疑问，可以通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

本节介绍如何删除 AWS OpsWorks for Chef Automate 服务器。删除服务器的同时会删除存储在服务器上的事件、日志和任何说明书。支持资源（Amazon Elastic Compute Cloud 实例、Amazon Elastic Block Store 卷等）以及所有自动备份也将被删除。

尽管删除某个服务器并不会删除节点，但这些节点不再由删除的服务器进行管理，因此将不断尝试重新连接。因此，我们建议您在删除 Chef 服务器之前，解除托管节点的关联。在此版本中，您可以通过运行 AWS CLI 命令来取消与节点的关联。

步骤 1：解除托管节点的关联

请在删除 Chef 服务器之前解除节点与该服务器的关联，以便节点继续工作，而不会尝试重新连接服务器。为此，请运行`disassociate-node` AWS CLI 命令。

解除节点的关联

1. 在中 AWS CLI，运行以下命令取消节点的关联。*Server_name* 是您要将节点与之解除关联的 Chef 服务器的名称。

   ```
   aws opsworks-cm --region Region_name disassociate-node --node-name Node_name --
   server-name Server_name
   ```

2. 请耐心等待，直到响应消息指示已完成关联断开。

步骤 2：删除服务器

1. 在仪表板的服务器磁贴上，展开 Actions 菜单。
2. 选择 Delete server。
3. 当系统提示您确认删除时，选择 Yes。

重置 Chef Automate 控制面板的凭证

> **Important**
>
> AWS OpsWorks for Chef Automate 已于 2024 年 5 月 5 日停用，新客户和现有客户均已禁用。我们建议现有客户迁移到 Chef SaaS 或其他替代解决方案。如果您有任何疑问，可以通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

您可能想要定期更改您用于登录 Chef Automate 控制面板的密码。如果您丢失了 Chef Automate 控制面板密码，也可以使用本节中显示的 Amazon EC2 Systems Manager AWS CLI 命令来更改该密码。您使用的命令取决于您的 Chef Automate 服务器运行的是 Chef Automate 的版本 1 还是版本 2。

1. 要返回 Chef 服务器的实例 ID，请打开以下页面。 AWS Management Console

   https://console.aws.amazon.com/ec2/v2/home?region = region_of_your_ser ver #Instances: search=-server_nam e aws-opsworks-cm

   例如，对于在美国西部（俄勒冈）MyChefServer地区命名的 Chef 服务器，控制台 URL 将如下所示。

https://console.aws.amazon.com/ec2/v2/home?region=us-west-2#Instances:search = aws-opsworks-cm-MyChefServer

记下控制台中显示的实例 ID；更改密码将需要用到该信息。

2. 要重置 Chef Automate 仪表板登录密码，请运行以下 AWS CLI 命令之一，具体取决于您的服务器运行的是 Chef Automate 1 还是 Chef Automate 2。将 *enterprise_name* 替换为您的企业或组织名称，将 *user_name* 替换为服务器上管理员的用户名称，将 *new_password* 替换为您要使用的密码，以及将 *region_name* 替换为服务器所在的区域。如果您没有指定企业名称，则企业名称将为 `default`。默认情况下，*enterprise_name* 为 `default` (这是始终预配置的组织名称)。对于 *user_name*， AWS OpsWorks for Chef Automate 仅创建名为的用户。`admin`记下新密码，然后将其存放在安全但方便的位置。

   对于 Chef Automate 1：

   ```
   aws ssm send-command --document-name "AWS-RunShellScript" --comment "reset admin
    password" --instance-ids "instance_id"
   --parameters commands="sudo delivery-ctl reset-
   password enterprise_name user_name new_password" --region region_name --output text
   ```

   对于 Chef Automate 2：

   ```
   aws ssm send-command --document-name "AWS-RunShellScript" --comment "reset admin
    password" --instance-ids "instance_id"
   --parameters commands="sudo chef-automate iam admin-access restore new_password" --
   region region_name --output text
   ```

3. 等待输出文本 (在本例中为命令 ID) 显示密码更改已完成。

使用记录 AWS OpsWorks for Chef Automate API 调用 AWS CloudTrail

> **Important**
>
> AWS OpsWorks for Chef Automate 已于 2024 年 5 月 5 日停用，新客户和现有客户均已禁用。我们建议现有客户迁移到 Chef SaaS 或其他替代解决方案。如果您有任何疑问，可以通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

AWS OpsWorks for Chef Automate 与 AWS CloudTrail一项服务集成，该服务提供 IAM 身份或中的 AWS 服务所采取的操作的记录 AWS OpsWorks for Chef Automate。 CloudTrail 将所有 API 调用捕获 AWS OpsWorks for Chef Automate 为事件，包括来自 AWS OpsWorks for Chef Automate 控制台的调用和对 AWS OpsWorks for Chef Automate API 的代码调用。如果您创建了跟踪，则可以允许将 CloudTrail 事件持续传输到 Amazon S3 存储桶，包括的事件 AWS OpsWorks for Chef Automate。如果您未配置跟踪，您仍然可以在 CloudTrail 控制台的 “事件历史记录” 中查看最新的事件。使用收集的信息 CloudTrail，您可以确定向哪个请求发出 AWS OpsWorks for Chef Automate、发出请求的 IP 地址、谁发出了请求、何时发出请求以及其他详细信息。

要了解更多信息 CloudTrail，请参阅《AWS CloudTrail 用户指南》。

AWS OpsWorks for Chef Automate 中的信息 CloudTrail

CloudTrail 在您创建 AWS 账户时已在您的账户上启用。当活动发生在中时 AWS OpsWorks for Chef Automate，该活动会与其他 AWS 服务 CloudTrail 事件一起记录在事件历史记录中。您可以在自己的 AWS 账户中查看、搜索和下载最近发生的事件。有关更多信息，请参阅使用事件历史记录查看 CloudTrail 事件。

要持续记录您 AWS 账户中的事件，包括其中的事件 AWS OpsWorks for Chef Automate，请创建跟踪。跟踪允许 CloudTrail 将日志文件传输到 Amazon S3 存储桶。默认情况下，在控制台中创建跟踪记录时，此跟踪记录应用于所有区域。跟踪记录 AWS 分区中所有区域的事件，并将日志文件传送到您指定的 Amazon S3 存储桶。此外，您可以配置其他 AWS 服务，以进一步分析和处理 CloudTrail 日志中收集的事件数据。有关更多信息，请参阅：

- 创建跟踪概述
- CloudTrail 支持的服务和集成
- 配置 Amazon SNS 通知 CloudTrail
- 接收来自多个区域的 CloudTrail 日志文件和接收来自多个账户的 CloudTrail 日志文件

所有 AWS OpsWorks for Chef Automate 操作均由《API 参考》记录 CloudTrail 并记录在《AWS OpsWorks for Chef Automate API 参考》中。例如，对CreateServerCreateBackup、和DescribeServers操作的调用会在 CloudTrail 日志文件中生成条目。

每个事件或日记账条目都包含有关生成请求的人员信息。身份信息可帮助您确定以下内容：

- 请求是使用根用户凭证还是 IAM 用户凭证发出的。
- 请求是使用角色还是联合用户的临时安全凭证发出的。
- 请求是否由其他 AWS 服务发出。

有关更多信息，请参阅CloudTrail 用户身份元素。

了解 AWS OpsWorks for Chef Automate 日志文件条目

跟踪是一种配置，允许将事件作为日志文件传输到您指定的 Amazon S3 存储桶。 CloudTrail 日志文件包含一个或多个日志条目。事件代表来自任何来源的单个请求，包括有关请求的操作、操作的日期和时间、请求参数等的信息。 CloudTrail 日志文件不是公共 API 调用的有序堆栈跟踪，因此它们不会按任何特定顺序出现。

以下示例显示了该 AWS OpsWorks for Chef Automate `CreateServer`操作的 CloudTrail 日志条目。

```
{"eventVersion":"1.05",
"userIdentity":{
    "type":"AssumedRole",
    "principalId":"ID number:OpsWorksCMUser",
    "arn":"arn:aws:sts::831000000000:assumed-role/Admin/OpsWorksCMUser",
    "accountId":"831000000000","accessKeyId":"ID number",
    "sessionContext":{
        "attributes":{
            "mfaAuthenticated":"false",
            "creationDate":"2017-01-05T22:03:47Z"
            },
        "sessionIssuer":{
            "type":"Role",
            "principalId":"ID number",
            "arn":"arn:aws:iam::831000000000:role/Admin",
            "accountId":"831000000000",
            "userName":"Admin"
            }
        }
    },
"eventTime":"2017-01-05T22:18:23Z",
"eventSource":"opsworks-cm.amazonaws.com",
"eventName":"CreateServer",
"awsRegion":"us-west-2",
"sourceIPAddress":"101.25.190.51",
"userAgent":"console.amazonaws.com",
"requestParameters":{
    "serverName":"OpsChef-test-server",
    "engineModel":"Single",
    "engine":"Chef",
    "instanceProfileArn":"arn:aws:iam::831000000000:instance-profile/aws-opsworks-cm-
ec2-role",
```

```
    "backupRetentionCount":3,"serviceRoleArn":"arn:aws:iam::831000000000:role/service-
role/aws-opsworks-cm-service-role",
    "engineVersion":"12",
    "preferredMaintenanceWindow":"Fri:21:00",
    "instanceType":"t2.medium",
    "subnetIds":["subnet-1e111f11"],
    "preferredBackupWindow":"Wed:08:00"
    },
"responseElements":{
    "server":{
        "endpoint":"OpsChef-test-server-thohsgreckcnwgz3.us-west-2.opsworks-cm.io",
        "createdAt":"Jan 5, 2017 10:18:22 PM",
        "serviceRoleArn":"arn:aws:iam::831000000000:role/service-role/aws-opsworks-cm-
service-role",
        "preferredBackupWindow":"Wed:08:00",
        "status":"CREATING",
        "subnetIds":["subnet-1e111f11"],
        "engine":"Chef",
        "instanceType":"t2.medium",
        "serverName":"OpsChef-test-server",
        "serverArn":"arn:aws:opsworks-cm:us-west-2:831000000000:server/OpsChef-test-
server/8epp7f6z-e91f-4z10-89z5-8c6219cdb09f",
        "engineModel":"Single",
        "backupRetentionCount":3,
        "engineAttributes":[
            {"name":"CHEF_STARTER_KIT","value":"*** Redacted ***"},
            {"name":"CHEF_PIVOTAL_KEY","value":"*** Redacted ***"},
            {"name":"CHEF_DELIVERY_ADMIN_PASSWORD","value":"*** Redacted ***"}],
        "engineVersion":"12.11.1",
        "instanceProfileArn":"arn:aws:iam::831000000000:instance-profile/aws-opsworks-
cm-ec2-role",
        "preferredMaintenanceWindow":"Fri:21:00"
        }
    },
"requestID":"de7f64f9-d394-12ug-8081-7bb0386fbcb6",
"eventID":"8r7b18df-6c90-47be-87cf-e8346428cfc3",
"eventType":"AwsApiCall",
"recipientAccountId":"831000000000"
}
```

故障排除 AWS OpsWorks for Chef Automate

> **Important**
>
> AWS OpsWorks for Chef Automate 已于 2024 年 5 月 5 日停用，新客户和现有客户均已禁用。我们建议现有客户迁移到 Chef SaaS 或其他替代解决方案。如果您有任何疑问，可以通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

本主题包含一些常见 AWS OpsWorks for Chef Automate 问题以及这些问题的建议解决方案。

主题

一般故障排除技巧

如果您无法创建或使用 Chef 服务器，可查看错误消息或日志来帮助您对问题进行故障排除。下列任务描述了在您排除 Chef 服务器问题故障时通常应从哪些方面入手。有关特定错误和解决方案的信息，请参阅本主题的针对特定错误进行故障排除部分。

- 在 Chef 服务器无法启动时，使用 AWS OpsWorks for Chef Automate 控制台查看错误消息。在 Chef 服务器详细信息页面上，与服务器启动和运行相关的错误消息将显示在页面顶部。错误可能来自用于创建 Chef 服务器的服务 AWS OpsWorks for Chef Automate AWS CloudFormation、或 Amazon EC2。在详细信息页面上，您还可查看正在运行的服务器上发生的事件，其中可能会包含故障事件消息。
- 要帮助解决 EC2 问题，请通过使用 SSH 连接到您的服务器实例并查看日志。EC2 实例日志存储在 `/var/log/aws/opsworks-cm` 目录中。这些日志在 AWS OpsWorks for Chef Automate 启动 Chef 服务器时捕获命令输出。

针对特定错误进行故障排除

主题

- 托管节点显示在 Chef Automate 控制面板的“Missing”列中
- 无法创建 Chef 保险库；knife vault 命令失败并返回错误
- 服务器创建失败，并返回“requested configuration is currently not supported”消息
- Chef 服务器未识别 Chef Automate 控制面板中添加的组织名称
- 无法创建服务器的 Amazon EC2 实例
- 服务角色错误阻止服务器创建
- 超出弹性 IP 地址限制
- 无法登录 Chef Automate 控制面板
- 无人参与节点关联失败
- 系统维护失败

服务器处于连接丢失状态

问题：服务器的状态显示为连接丢失。

原因：这种情况最常发生在外部实体对 AWS OpsWorks for Chef Automate 服务器或其支持资源 AWS OpsWorks 进行更改时。 AWS OpsWorks 无法连接到处于连接丢失状态的 Chef Automate 服务器来处理维护任务，例如创建备份、应用操作系统补丁或更新 Chef Automate。因此，您的服务器可能缺少重要更新，容易受到安全问题的影响，或者无法按预期运行。

解决方案：尝试以下步骤来恢复服务器的连接。

1. 请确保您的服务角色具有所有必需的权限。

    a. 在服务器的设置页面上，在网络和安全中，选择服务器正在使用的服务角色的链接。这将将服务角色打开以供在 IAM 控制台中查看。

    b. 在权限选项卡上，确认 `AWSOpsWorksCMServiceRole` 是否在权限策略列表中。如果未列出该托管策略，请手动将 `AWSOpsWorksCMServiceRole` 托管策略添加到角色中。

    c. 在信任关系选项卡上，验证服务角色是否具有信任 `opsworks-cm.amazonaws.com` 服务代表您代入角色的信任策略。有关如何对角色使用信任策略的更多信息，请参阅修改角色（控制台）或 AWS 安全博客文章《如何在 IAM 角色中使用信任策略》。

2. 请确保您的实例配置文件具有所有必需的权限。

    a. 在服务器的设置页面上，在网络和安全中，选择服务器正在使用的实例配置文件的链接。这将打开实例配置文件以在 IAM 控制台中查看。

b. 在权限选项卡上，确认 AmazonEC2RoleforSSM 和 AWSOpsWorksCMInstanceProfileRole 是否在权限策略列表中。如果未列出其中一个或两个托管策略，请手动将这些托管策略添加到角色中。

c. 在信任关系选项卡上，验证服务角色是否具有信任 ec2.amazonaws.com 服务代表您代入角色的信任策略。有关如何对角色使用信任策略的更多信息，请参阅修改角色（控制台）或 AWS 安全博客文章《如何在 IAM 角色中使用信任策略》。

3. 在 Amazon EC2 控制台中，确保您与 AWS OpsWorks for Chef Automate 服务器所在区域位于同一区域，然后重启服务器正在使用的 EC2 实例。

a. 选择名为 aws-opsworks-cm-instance- *#####*的 EC2 实例。

b. 在实例状态菜单，选择启动实例。

c. 等待最多 15 分钟让您的服务器重新启动并完全联机。

4. 在 AWS OpsWorks for Chef Automate 控制台的服务器详细信息页面上，验证服务器状态现在是否正常。

如果执行上述步骤后服务器状态仍为连接丢失，请尝试以下方法之一。

- 通过创建新服务器并删除原始服务器来替换服务器。如果当前服务器上的数据对您很重要，请从最近的备份中恢复服务器，并验证数据是否为最新，然后再删除未响应的原始服务器。
- 请联系 AWS 支持。

托管节点显示在 Chef Automate 控制面板的“Missing”列中

问题：一个托管节点显示在 Chef Automate 控制面板的 Missing 列中。

原因：如果某个节点未连接到 Chef Automate 服务器的时间超过 12 小时，并且 chef-client 无法在该节点上运行，则该节点的状态将从 12 小时前的状态发生变化，并移动到 Chef Automate 控制面板的 Missing 列中。

解决方案：确认该节点处于联机状态。尝试运行 knife node show *node_name* --run-list 以查看 chef-client 能否在该节点上运行，或运行 knife node show -l *node_name* 以显示有关该节点的所有信息。该节点可能处于脱机状态或已断开网络连接。

无法创建 Chef 保险库；**knife vault** 命令失败并返回错误

问题：您尝试运行 knife vault 命令来在您的 Chef Automate 服务器上创建一个保管库，例如一个用于存储基于 Windows 的节点加入域所用的凭证的保管库。该命令返回类似以下的错误消息。

```
WARN: Auto inflation of JSON data is deprecated. Please pass in the class to inflate or
 use #edit_hash (CHEF-1)
at /opt/chefdk/embedded/lib/ruby/2.3.0/forwardable.rb:189:in `edit_data'.Please see
 https://docs.chef.io/deprecations_json_auto_inflate.html
for further details and information on how to correct this problem.
WARNING: pivotal not found in users, trying clients.
ERROR: ChefVault::Exceptions::AdminNotFound: FATAL: Could not find pivotal in users or
 clients!
```

当您远程运行 `knife user list` 时未返回关键用户，但当您在 Chef Automate 服务器上本地运行 `chef-server-ctl user-show` 命令时，可在结果中看到关键用户。换句话说，您的 `knife vault` 命令无法找到关键用户，但您知道它存在。

原因：虽然关键用户在 Chef 中被视为超级用户，并且具有完整权限，但它不属于任何组织，包括在 AWS OpsWorks for Chef Automate中使用的 `default` 组织。命令 `knife user list` 将返回属于您的 Chef 配置中的当前组织中的所有用户。`chef-server-ctl user-show` 命令将返回包括关键用户在内的所有用户，而不论用户属于哪个组织。

解决方案：要修复此问题，请通过运行 `knife opc` 将关键用户添加到默认组织。

首先，您需要安装 knife-opc 插件。

```
chef gem install knife-opc
```

安装此插件后，运行以下命令将关键用户添加到默认组织。

```
knife opc org user add default pivotal
```

您可通过再次运行 `knife user list` 来确认关键用户是否为默认组织的一部分。`pivotal` 应会在结果中列出。然后，尝试再次运行 `knife vault`。

服务器创建失败，并返回“requested configuration is currently not supported”消息

问题：您尝试创建一台 Chef Automate 服务器，但服务器创建失败，并返回与“The requested configuration is currently not supported。Please check the documentation for supported configurations.”类似的错误消息。

原因：可能为 Chef Automate 服务器指定了不支持的实例类型。如果您选择在具有非默认租赁的 VPC 中创建 Chef Automate 服务器，例如适用于专用实例的 VPC，则指定 VPC 内的所有实例也必须为专

用或主机租赁。由于某些实例类型 (如 t2) 只适用于默认租赁，指定 VPC 可能不支持 Chef Automate 服务器实例类型，因此服务器创建失败。

解决方案：如果您选择具有非默认租赁的 VPC，请使用 m4 实例类型，此类型可以支持专用租赁。

Chef 服务器未识别 Chef Automate 控制面板中添加的组织名称

问题：您已在 Chef Automate 控制面板中添加新的工作流程组织名称，或在无人参与节点关联脚本中指定了 "default" 之外的 CHEF_AUTOMATE_ORGANIZATION 值，但节点关联失败。您的 AWS OpsWorks for Chef Automate 服务器未识别新的组织名称。

原因：工作流程组织名称和 Chef 服务器组织名称不同。您可在基于 Web 的 Chef Automate 控制面板中创建新的工作流程组织，而不是 Chef 服务器组织名称。您只能使用 Chef Automate 控制面板查看现有 Chef 服务器组织。您在 Chef Automate 控制面板中创建的新组织是一个工作流程组织，Chef 服务器未能识别。您无法通过在节点关联脚本中指定新的组织名称来创建它们。当在节点关联脚本中引用某个组织名称时，如果在此之前未将该组织添加到 Chef 服务器，则将导致节点关联失败。

解决方案：要创建在 Chef 服务器上识别的新组织，请使用 `knife opc org create` 命令，或运行 `chef-server-ctl org-create`。

无法创建服务器的 Amazon EC2 实例

问题：服务器创建失败，并返回类似以下的错误消息：“The following resource(s) failed to create: [EC2Instance]。Failed to receive 1 resource signal(s) within the specified duration.”

原因：很可能的原因是 EC2 实例没有网络访问权限。

解决方案：确保实例具有出站 Internet 访问权限，并且 AWS 服务代理能够发出命令。请确保您的 VPC (具有单一公有子网的 VPC) 已启用 DNS resolution，并且您的子网已启用 Auto-assign Public IP 设置。

服务角色错误阻止服务器创建

问题：服务器创建失败，并显示一条错误消息，上面写着 “未授权执行 sts:” AssumeRole。

原因：当您使用的服务角色缺少足够的权限创建新服务器时，可能会出现此问题。

解决方案：打开 AWS OpsWorks for Chef Automate 控制台；使用控制台生成新的服务角色和实例配置文件角色。如果您希望使用自己的服务角色，请将AWSOpsWorksCMServiceRole策略附加到该角色。验证 opsworks-cm.amazonaws.com 在角色的 Trust Relationships 中随服务一起列出。验证与 Chef 服务器关联的服务角色是否已附加AWSOpsWorksCMServiceRole托管策略。

超出弹性 IP 地址限制

问题：服务器创建失败，并返回错误消息，指示“The following resource(s) failed to create: [EIP, EC2Instance]。Resource creation cancelled, the maximum number of addresses has been reached.”

原因：当您的账户已使用最大数量的弹性 IP (EIP) 地址时，将会出现此问题。默认的 EIP 地址数量限制为 5。

解决方案：您可以释放现有 EIP 地址或删除您的账户未使用的 EIP 地址，也可以联系 Cust AWS omer Support 以增加与您的账户关联的 EIP 地址的限制。

无法登录 Chef Automate 控制面板

问题：Chef Automate 控制面板显示类似以下的错误：“Cross-Origin Request Blocked: The Same Origin Policy disallows reading the remote resource at https://myserver-name.region.opsworks-cm.io/api/v0/e/default/verify-token。(Reason: CORS header 'Access-Control-Allow-Origin' missing)”。错误也可能类似于“The User Id / Password combination entered is incorrect.”

原因：Chef Automate 控制面板显式设置了 FQDN，并且不接受相对 URL。此时，您无法使用 Chef 服务器的 IP 地址登录；您只能使用服务器的 DNS 名称登录。

解决方案：使用 Chef 服务器的 DNS 名称条目而不是其 IP 地址来登录 Chef Automate 控制面板。您也可通过运行 AWS CLI 命令尝试重置 Chef Automate 控制面板凭证，如 重置 Chef Automate 控制面板的凭证中所述。

无人参与节点关联失败

问题：新的 Amazon EC2 节点的无人参与或自动关联失败。应该已经添加到 Chef 服务器的节点未显示在 Chef Automate 控制面板中，并且未在 `knife client show` 或 `knife node show` 命令的结果中列出。

原因：当您未将 IAM 角色设置为允许 `opsworks-cm` API 调用与新的 EC2 实例通信的实例配置文件时，可能会出现此问题。

解决方案：将一个策略附加到您的 EC2 实例配置文件以允许 `AssociateNode` 和 `DescribeNodeAssociationStatus` API 调用与 EC2 一起工作，如在中自动添加节点 AWS OpsWorks for Chef Automate中所述。

系统维护失败

AWS OpsWorks CM 每周执行系统维护，以确保最新次要版本的 Chef Server 和 Chef Automate Server（包括安全更新）始终在 AWS OpsWorks for Chef Automate 服务器上运行。如果由于任何原

因导致系统维护失败，则 AWS OpsWorks CM 会通知您该故障。有关系统维护的更多信息，请参阅 中的系统维护 AWS OpsWorks for Chef Automate。

本节介绍可能的失败原因并提出解决方案。

主题

- 服务角色或实例配置文件错误会阻止系统维护

服务角色或实例配置文件错误会阻止系统维护

问题：系统维护失败，并显示一条错误消息，上面写着 “未授权执行 sts:AssumeRole” 或类似的权限错误消息。

原因：当您使用的服务角色或实例配置文件缺少在服务器上执行系统维护的足够权限时，可能会发生这种情况。

解决方案：确保您的服务角色和实例配置文件具有所有必需的权限。

1. 请确保您的服务角色具有所有必需的权限。
   a. 在服务器的设置页面上，在网络和安全中，选择服务器正在使用的服务角色的链接。这将将服务角色打开以供在 IAM 控制台中查看。
   b. 在权限选项卡上，验证 AWSOpsWorksCMServiceRole 是否已附加到该服务角色。如果 AWSOpsWorksCMServiceRole 未列出，则将此策略添加到角色。
   c. 验证 opsworks-cm.amazonaws.com 在角色的 Trust Relationships 中随服务一起列出。有关如何对角色使用信任策略的更多信息，请参阅修改角色（控制台）或 AWS 安全博客文章《如何在 IAM 角色中使用信任策略》。
2. 请确保您的实例配置文件具有所有必需的权限。
   a. 在服务器的设置页面上，在网络和安全中，选择服务器正在使用的实例配置文件的链接。这将打开实例配置文件以在 IAM 控制台中查看。
   b. 在权限选项卡上，确认 AmazonEC2RoleforSSM 和 AWSOpsWorksCMInstanceProfileRole 是否在权限策略列表中。如果未列出其中一个或两个托管策略，请手动将这些托管策略添加到角色中。
   c. 在信任关系选项卡上，验证服务角色是否具有信任 ec2.amazonaws.com 服务代表您代入角色的信任策略。有关如何对角色使用信任策略的更多信息，请参阅修改角色（控制台）或 AWS 安全博客文章《如何在 IAM 角色中使用信任策略》。

其他帮助和支持

如果本主题没有描述您的特定问题，或者您已尝试本主题中的建议，但问题仍然存在，请访问 AWS OpsWorks 论坛。

您也可访问 AWS Support Center 。 AWS 支持中心是创建和管理 AWS 支持案例的中心。Su AWS pport Center 还包括指向其他有用资源的链接，例如论坛、技术常见问题解答、服务运行状况和 AWS Trusted Advisor。

AWS OpsWorks 配置管理 (CM) 中的安全性

AWS十分重视云安全性。作为AWS客户，您将从专为满足大多数安全敏感型企业的要求而打造的数据中心和网络架构中受益。

安全性是AWS和您的共同责任。责任共担模式将其描述为云的安全性和云中的安全性：

- 云的安全性 – AWS 负责保护在 AWS 云中运行 AWS 服务的基础设施。AWS 还向您提供可安全使用的服务。作为AWS合规性计划的一部分，第三方审核人员将定期测试和验证安全性的有效性。要了解适用于 AWS OpsWorks CM 的合规性计划，请参阅合规性计划范围内的 AWS 服务。
- 云中的安全性——您的责任由您使用的 AWS 服务决定。您还需要对其他因素负责，包括数据的敏感性、公司的要求以及适用的法律法规。

此文档将帮助您了解如何在使用 AWS OpsWorks CM 时应用责任共担模型。以下主题说明如何配置 AWS OpsWorks CM 以实现您的安全性和合规性目标。您还将了解如何使用其他 Amazon Web Service 来帮助您监控和保护您的 AWS OpsWorks CM 资源。

主题

AWS OpsWorks CM 中的数据保护

AWS 责任共担模式适用于 AWS OpsWorks Configuration Management 中的数据保护。如该模式中所述，AWS 负责保护运行所有 AWS Cloud 的全球基础设施。您负责维护对托管在此基础设施上的内容的控制。您还负责您所使用的 AWS 服务 的安全配置和管理任务。有关数据隐私的更多信息，

请参阅数据隐私常见问题。有关欧洲数据保护的信息，请参阅 AWS 安全性博客 上的博客文章 AWS Shared Responsibility Model and GDPR。

出于数据保护目的，我们建议您保护 AWS 账户 凭证并使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 设置单个用户。这样，每个用户只获得履行其工作职责所需的权限。我们还建议您通过以下方式保护数据：

- 对每个账户使用多重身份验证（MFA）。
- 使用 SSL/TLS 与 AWS 资源进行通信。我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 使用 AWS CloudTrail 设置 API 和用户活动日志记录。
- 使用 AWS 加密解决方案以及 AWS 服务 中的所有默认安全控制。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Amazon S3 中的敏感数据。
- 如果您在通过命令行界面或 API 访问 AWS 时需要经过 FIPS 140-2 验证的加密模块，请使用 FIPS 端点。有关可用的 FIPS 端点的更多信息，请参阅《美国联邦信息处理标准 (FIPS) 第 140-2 版》。

我们强烈建议您切勿将机密信息或敏感信息（如您客户的电子邮件地址）放入标签或自由格式文本字段（如名称字段）。这包括使用控制台、API、AWS CLI 或 AWS SDK 处理 OpsWorks CM 或其他 AWS 服务 时。在用于名称的标签或自由格式文本字段中输入的任何数据都可能会用于计费或诊断日志。如果您向外部服务器提供网址，我们强烈建议您不要在网址中包含凭证信息来验证对该服务器的请求。

OpsWorks CM 服务器的名称是未加密的。

OpsWorks CM 在创建和维护您的 AWS OpsWorks for Chef Automate 和 AWS OpsWorks for Puppet Enterprise 服务器的过程中收集以下客户数据。

- 对于 OpsWorks for Puppet Enterprise，我们收集 Puppet Enterprise 用于启用 Puppet 主节点和托管节点之间的通信的私有密钥。
- 对于 AWS OpsWorks for Chef Automate，如果您使用的是自定义域，我们将收集您附加到服务的证书的私有密钥。您在创建带自定义域的 Chef Automate 服务器时提供的私有密钥将传递到您的服务器。

OpsWorks CM 服务器存储您的配置代码，例如 Chef 说明书或 Puppet Enterprise 模块。虽然此代码存储在服务器备份中，但 AWS 无权访问它。此内容已加密，仅您的 AWS 账户中的管理员能够访问它。我们建议您使用针对源存储库的推荐协议来保护您的 Chef 或 Puppet 配置代码。例如，您可以限制 AWS CodeCommit 中存储库的权限，或遵循 GitHub 网站上有关保护 GitHub 存储库的指南。

OpsWorks CM 不会使用客户提供的内容来维护服务或保留客户日志。有关 OpsWorks CM 服务器的日志存储在您的账户中的 Amazon S3 存储桶中。连接到您的 OpsWorks CM 服务器的用户的 IP 地址由 AWS 记录。

与 AWS Secrets Manager 集成

从 2021 年 5 月 3号开始，在 OpsWorks CM 中创建新的服务器时，OpsWorks CM 会将服务器的密钥存储在 AWS Secrets Manager 中。对于新服务器，以下属性作为密钥存储在 Secrets Manager 中。

- Chef Automate 服务器
  - HTTPS 私钥（仅限不使用自定义域的服务器）
  - Chef Automate 管理密码（CHEF_AUTOMATE_ADMIN_PASSWORD）
- Puppet Enterprise master
  - HTTPS 私钥（仅限不使用自定义域的服务器）
  - Puppet 管理密码（PUPPET_ADMIN_PASSWORD）
  - Puppet r10k 遥控器（PUPPET_R10K_REMOTE）

对于不使用自定义域的现有服务器，Chef Automate 和 Puppet Enterprise 服务器存储在 Secrets Manager 中的唯一密钥是 HTTPS 私钥，因为这是在每周自动系统维护期间生成的。

OpsWorks CM 会自动将密钥存储在 Secrets Manager 中，用户无法配置此行为。

数据加密

AWS OpsWorks CM 加密服务器备份以及授权 AWS 用户与其 AWS OpsWorks CM 服务器之间的通信。但是，AWS OpsWorks CM 服务器的根 Amazon EBS 卷是未加密的。

静态加密

AWS OpsWorks CM 服务器备份是加密的。但是，AWS OpsWorks CM 服务器的根 Amazon EBS 卷是未加密的。用户无法对其进行配置。

传输中加密

AWS OpsWorks CM 将 HTTP 与 TLS 加密结合使用。AWS OpsWorks如果用户未提供签名证书，则 CM 默认使用自签名证书来预置和管理服务器。我们建议您使用由证书颁发机构 (CA) 签名的证书。

密钥管理

AWS OpsWorks CM 目前不支持 AWS Key Management Service 客户托管密钥和 AWS 托管密钥。

适用于 CM 的 Identity and A AWS OpsWorks ccess 管理

AWS Identity and Access Management (IAM) 是一项 AWS 服务，可帮助管理员安全地控制对 AWS 资源的访问。IAM 管理员控制谁可以进行身份验证（登录）和授权（有权限）使用 OpsWorks CM 资源。IAM 是一项无需额外付费即可使用的 AWS 服务。

主题

受众

您的使用方式 AWS Identity and Access Management (IAM) 会有所不同，具体取决于您在 OpsWorks CM 中所做的工作。

服务用户-如果您使用 OpsWorks CM 服务完成工作，则管理员会为您提供所需的凭证和权限。当你使用更多的 OpsWorks CM 功能来完成工作时，你可能需要额外的权限。了解如何管理访问权限有助于您向管理员请求适合的权限。如果您无法访问 OpsWorks CM 中的某项功能，请参阅AWS OpsWorks CM 身份和访问疑难解答。

服务管理员-如果您负责公司的 OpsWorks CM 资源，则可能拥有对 OpsWorks CM 的完全访问权限。您的工作是确定您的服务用户应访问哪些 OpsWorks 内容管理功能和资源。然后，您必须向 IAM 管理员提交请求以更改服务用户的权限。请查看该页面上的信息以了解 IAM 的基本概念。要详细了解贵公司如何将 IAM 与 OpsWorks CM 结合使用，请参阅AWS OpsWorks CM 如何与 IAM 配合使用。

IAM 管理员 — 如果您是 IAM 管理员，则可能需要详细了解如何编写策略来管理 OpsWorks CM 的访问权限。要查看您可以在 IA OpsWorks M 中使用的基于身份的 CM 策略示例，请参阅。AWS OpsWorks CM 基于身份的策略示例

使用身份进行身份验证

身份验证是您 AWS 使用身份凭证登录的方式。您必须以 IAM 用户身份或通过担 AWS 账户根用户任 IAM 角色进行身份验证（登录 AWS）。

您可以使用通过身份源提供的凭据以 AWS 联合身份登录。 AWS IAM Identity Center （IAM Identity Center）用户、贵公司的单点登录身份验证以及您的 Google 或 Facebook 凭据就是联合身份的示例。当您以联合身份登录时，您的管理员以前使用 IAM 角色设置了身份联合验证。当你使用联合访问 AWS 时，你就是在间接扮演一个角色。

根据您的用户类型，您可以登录 AWS Management Console 或 AWS 访问门户。有关登录的更多信息 AWS，请参阅《AWS 登录 用户指南》中的如何登录到您 AWS 账户的。

如果您 AWS 以编程方式访问，则会 AWS 提供软件开发套件 (SDK) 和命令行接口 (CLI)，以便使用您的凭据对请求进行加密签名。如果您不使用 AWS 工具，则必须自己签署请求。有关使用推荐的方法自行签署请求的更多信息，请参阅 IAM 用户指南中的签署 AWS API 请求。

无论使用何种身份验证方法，您可能需要提供其他安全信息。例如， AWS 建议您使用多重身份验证 (MFA) 来提高账户的安全性。要了解更多信息，请参阅《AWS IAM Identity Center 用户指南》中的 多重身份验证和《IAM 用户指南》中的 在 AWS中使用多重身份验证（MFA）。

AWS 账户 root 用户

创建时 AWS 账户，首先要有一个登录身份，该身份可以完全访问账户中的所有资源 AWS 服务 和资源。此身份被称为 AWS 账户 root 用户，使用您创建账户时使用的电子邮件地址和密码登录即可访问该身份。强烈建议您不要使用根用户执行日常任务。保护好根用户凭证，并使用这些凭证来执行仅根用户可以执行的任务。有关需要您以根用户身份登录的任务的完整列表，请参阅《IAM 用户指南》中的需要根用户凭证的任务。

IAM 用户和群组

I AM 用户是您 AWS 账户 内部对个人或应用程序具有特定权限的身份。在可能的情况下，我们建议使用临时凭证，而不是创建具有长期凭证（如密码和访问密钥）的 IAM 用户。但是，如果您有一些特定的使用场景需要长期凭证以及 IAM 用户，建议您轮换访问密钥。有关更多信息，请参阅《IAM 用户指南》中的 对于需要长期凭证的使用场景定期轮换访问密钥。

IAM 组是一个指定一组 IAM 用户的身份。您不能使用组的身份登录。您可以使用组来一次性为多个用户指定权限。如果有大量用户，使用组可以更轻松地管理用户权限。例如，您可能具有一个名为 IAMAdmins 的组，并为该组授予权限以管理 IAM 资源。

用户与角色不同。用户唯一地与某个人员或应用程序关联，而角色旨在让需要它的任何人代入。用户具有永久的长期凭证，而角色提供临时凭证。要了解更多信息，请参阅《IAM 用户指南》中的何时创建 IAM 用户（而不是角色）。

> **Warning**
>
> IAM 用户拥有长期证书，这会带来安全风险。为帮助减轻这种风险，我们建议仅向这些用户提供执行任务所需的权限，并在不再需要这些用户时将其移除。

IAM 角色

I AM 角色是您内部具有特定权限 AWS 账户 的身份。它类似于 IAM 用户，但与特定人员不关联。您可以 AWS Management Console 通过切换角色在中临时担任 IAM 角色。您可以通过调用 AWS CLI 或 AWS API 操作或使用自定义 URL 来代入角色。有关使用角色的方法的更多信息，请参阅《IAM 用户指南》中的使用 IAM 角色。

具有临时凭证的 IAM 角色在以下情况下很有用：

- 联合用户访问 – 要向联合身份分配权限，请创建角色并为角色定义权限。当联合身份进行身份验证时，该身份将与角色相关联并被授予由此角色定义的权限。有关联合身份验证的角色的信息，请参阅《IAM 用户指南》中的为第三方身份提供商创建角色。如果您使用 IAM Identity Center，则需要配置权限集。为控制您的身份在进行身份验证后可以访问的内容，IAM Identity Center 将权限集与 IAM 中的角色相关联。有关权限集的信息，请参阅《AWS IAM Identity Center 用户指南》中的 权限集。
- 临时 IAM 用户权限 – IAM 用户可代入 IAM 用户或角色，以暂时获得针对特定任务的不同权限。
- 跨账户存取 – 您可以使用 IAM 角色以允许不同账户中的某个人（可信主体）访问您的账户中的资源。角色是授予跨账户访问权限的主要方式。但是，对于某些资源 AWS 服务，您可以将策略直接附加到资源（而不是使用角色作为代理）。要了解用于跨账户访问的角色和基于资源的策略之间的差别，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。
- 跨服务访问 — 有些 AWS 服务 使用其他 AWS 服务服务中的功能。例如，当您在某个服务中进行调用时，该服务通常会在 Amazon EC2 中运行应用程序或在 Simple Storage Service (Amazon S3) 中存储对象。服务可能会使用发出调用的主体的权限、使用服务角色或使用服务相关角色来执行此操作。

- 转发访问会话 (FAS) — 当您使用 IAM 用户或角色在中执行操作时 AWS，您被视为委托人。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS 使用调用委托人的权限以及 AWS 服务 向下游服务发出请求的请求。 AWS 服务只有当服务收到需要与其他 AWS 服务 或资源交互才能完成的请求时，才会发出 FAS 请求。在这种情况下，您必须具有执行这两个操作的权限。有关发出 FAS 请求时的策略详情，请参阅转发访问会话。
- 服务角色 - 服务角色是服务代表您在您的账户中执行操作而分派的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的创建向 AWS 服务委派权限的角色。
- 服务相关角色-服务相关角色是一种链接到的服务角色。 AWS 服务服务可以代入代表您执行操作的角色。服务相关角色出现在您的中 AWS 账户 ，并且归服务所有。IAM 管理员可以查看但不能编辑服务相关角色的权限。

- 在 A@@ mazon EC2 上运行的应用程序 — 您可以使用 IAM 角色管理在 EC2 实例上运行并发出 AWS CLI 或 AWS API 请求的应用程序的临时证书。这优先于在 EC2 实例中存储访问密钥。要向 EC2 实例分配 AWS 角色并使其可供其所有应用程序使用，您需要创建附加到该实例的实例配置文件。实例配置文件包含角色，并使 EC2 实例上运行的程序能够获得临时凭证。有关更多信息，请参阅《IAM 用户指南》中的使用 IAM 角色为 Amazon EC2 实例上运行的应用程序授予权限。

要了解是使用 IAM 角色还是 IAM 用户，请参阅IAM 用户指南中的何时创建 IAM 角色（而不是用户）。

使用策略管理访问

您可以 AWS 通过创建策略并将其附加到 AWS 身份或资源来控制中的访问权限。策略是其中的一个对象 AWS ，当与身份或资源关联时，它会定义其权限。 AWS 在委托人（用户、root 用户或角色会话）发出请求时评估这些策略。策略中的权限确定是允许还是拒绝请求。大多数策略都以 JSON 文档的 AWS 形式存储在中。有关 JSON 策略文档的结构和内容的更多信息，请参阅 IAM 用户指南中的 JSON 策略概览。

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

默认情况下，用户和角色没有权限。要授予用户对所需资源执行操作的权限，IAM 管理员可以创建 IAM 策略。管理员随后可以向角色添加 IAM policy，用户可以代入角色。

IAM 策略定义操作的权限，无关乎您使用哪种方法执行操作。例如，假设您有一个允许 `iam:GetRole` 操作的策略。拥有该策略的用户可以从 AWS Management Console AWS CLI、或 AWS API 获取角色信息。

基于身份的策略

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的创建 IAM policy。

基于身份的策略可以进一步归类为内联策略或托管策略。内联策略直接嵌入单个用户、组或角色中。托管策略是独立的策略，您可以将其附加到中的多个用户、群组和角色 AWS 账户。托管策略包括 AWS 托管策略和客户托管策略。要了解如何在托管式策略和内联策略之间进行选择，请参阅 IAM 用户指南中的在托管式策略与内联策略之间进行选择。

OpsWorks CM 支持您在 IAM 中创建并附加到用户、角色或群组的自定义策略。

基于资源的策略

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。委托人可以包括账户、用户、角色、联合用户或 AWS 服务。

基于资源的策略是位于该服务中的内联策略。您不能在基于资源的策略中使用 IAM 中的 AWS 托管策略。

OpsWorks CM 不支持基于资源的策略。

访问控制列表（ACL）

访问控制列表（ACL）控制哪些主体（账户成员、用户或角色）有权访问资源。ACL 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

Amazon S3 和 Amazon VPC 就是支持 ACL 的服务示例。 AWS WAF要了解有关 ACL 的更多信息，请参阅《Amazon Simple Storage Service 开发人员指南》中的访问控制列表（ACL）概览。

OpsWorks CM 不使用 ACL。

其他策略类型

OpsWorks CM 不支持以下其他策略类型。

AWS 支持其他不太常见的策略类型。这些策略类型可以设置更常用的策略类型向您授予的最大权限。

- 权限边界：权限边界是一个高级特征，用于设置基于身份的策略可以为 IAM 实体（用户或角色）授予的最大权限。您可为实体设置权限边界。这些结果权限是实体的基于身份的策略及其权限边界的交集。在 `Principal` 中指定用户或角色的基于资源的策略不受权限边界限制。任一项策略中的显式拒绝将覆盖允许。有关权限边界的更多信息，请参阅《IAM 用户指南》中的 IAM 实体的权限边界。
- 服务控制策略 (SCP)-SCP 是 JSON 策略，用于指定组织或组织单位 (OU) 的最大权限。 AWS Organizations AWS Organizations 是一项用于对您的企业拥有的多个 AWS 账户进行分组和集中管理的服务。如果在组织内启用了所有功能，则可对任意或全部账户应用服务控制策略 (SCP)。SCP 限制成员账户中的实体（包括每个 AWS 账户根用户实体）的权限。有关 Organization 和 SCP 的更多信息，请参阅 AWS Organizations 用户指南中的 SCP 的工作原理。
- 会话策略 – 会话策略是当您以编程方式为角色或联合用户创建临时会话时作为参数传递的高级策略。结果会话的权限是用户或角色的基于身份的策略和会话策略的交集。权限也可以来自基于资源的策略。任一项策略中的显式拒绝将覆盖允许。有关更多信息，请参阅《IAM 用户指南》中的会话策略。

多个策略类型

当多个类型的策略应用于一个请求时，生成的权限更加复杂和难以理解。要了解在涉及多种策略类型时如何 AWS 确定是否允许请求，请参阅 IAM 用户指南中的策略评估逻辑。

AWS OpsWorks CM 如何与 IAM 配合使用

在使用 IAM 管理对 AWS OpsWorks CM 的访问权限之前，您应该了解有哪些 IAM 功能可用于 AWS OpsWorks CM。要全面了解 AWS OpsWorks CM 和其他 AWS 服务如何与 IAM 配合使用，请参阅 IAM 用户指南中的与 IAM 配合使用的AWS 服务。

主题

- AWS OpsWorks CM 基于身份的策略
- AWS OpsWorks CM 和基于资源的政策
- 基于 AWS OpsWorks CM 标签的授权
- AWS OpsWorks CM IAM 角色

AWS OpsWorks CM 基于身份的策略

借助 IAM 基于身份的策略，您可以指定允许或拒绝的操作和资源，以及允许或拒绝操作的条件。 AWS OpsWorks CM 支持特定的操作、资源和条件键。要了解在 JSON 策略中使用的所有元素，请参阅《IAM 用户指南》中的 IAM JSON 策略元素参考。

在 AWS OpsWorks CM 中，您可以将自定义策略声明附加到用户、角色或组。

操作

基于 IAM 身份的策略的 Action 元素描述该策略将允许或拒绝的特定操作。策略操作通常与关联的 AWS API 操作同名。此策略用于策略中以授予执行关联操作的权限。

AWS OpsWorks CM 中的策略操作在操作前使用以下前缀：opsworks-cm:。例如，要授予某人使用 API 操作创建 AWS OpsWorks CM 服务器的权限，您应将 opsworks-cm:CreateServer 操作纳入其策略中。策略声明必须包含Action或NotAction元素。 AWS OpsWorks CM 定义了一组自己的操作，以描述您可以使用该服务执行的任务。

要在单个语句中指定多项操作，请使用逗号将它们隔开，如下所示：

```
"Action": [
      "opsworks-cm:action1",
      "opsworks-cm:action2"
```

您也可以使用通配符 （*) 指定多个操作。例如，要指定以单词 Describe 开头的所有操作，包括以下操作：

```
"Action": "opsworks-cm:Describe*"
```

当您在策略语句中使用通配符以允许多个操作时，请注意您只应允许授权服务或用户使用这些操作。

要查看 AWS OpsWorks CM 操作列表，请参阅 IA M 用户指南 OpsWorks中的 AWS 操作、资源和条件密钥。

资源

Resource 元素指定要向其应用操作的对象。语句必须包含 Resource 或 NotResource 元素。您可使用 ARN 来指定资源，或使用通配符 (*) 以指明该语句适用于所有资源。

您可以通过运行或 DescribeBackupsAPI 操作获取 AWS OpsWorks CM 服务器或备份的 Amazon 资源编号 (ARN)，并以这些DescribeServers资源为基础制定资源级策略。

C AWS OpsWorks M 服务器资源具有以下格式的 ARN：

```
arn:aws:opsworks-cm:{Region}:${Account}:server/${ServerName}/${UniqueId}
```

C AWS OpsWorks M 备份资源具有以下格式的 ARN：

```
arn:aws:opsworks-cm:{Region}:${Account}:backup/${ServerName}-{Date-and-Time-Stamp-of-Backup}
```

有关 ARN 格式的更多信息，请参阅 A mazon 资源名称 (ARN) 和 AWS 服务命名空间。

例如，要在语句中指定 `test-chef-automate` Chef Automate 服务器，请使用以下 ARN：

```
"Resource": "arn:aws:opsworks-cm:us-west-2:123456789012:server/test-chef-automate/EXAMPLE-d1a2bEXAMPLE"
```

要指定属于特定账户的所有 AWS OpsWorks CM 服务器，请使用通配符 (*)：

```
"Resource": "arn:aws:opsworks-cm:us-west-2:123456789012:server/*"
```

以下示例将 AWS OpsWorks CM 服务器备份指定为资源：

```
"Resource": "arn:aws:opsworks-cm:us-west-2:123456789012:backup/test-chef-automate-server-2018-05-20T19:06:12.399Z"
```

某些 AWS OpsWorks CM 操作（例如用于创建资源的操作）无法对特定资源执行。在这些情况下，您必须使用通配符 （*)。

```
"Resource": "*"
```

许多 API 操作涉及多种资源。要在单个语句中指定多个资源，请使用逗号分隔 ARN。

```
"Resource": [
      "resource1",
      "resource2"
```

要查看 AWS OpsWorks CM 资源类型及其 ARN 的列表，请参阅 IA M 用户指南中的 AWS OpsWorks CM 的操作、资源和条件密钥。要了解您可以使用哪些操作来指定每种资源的 ARN，请参阅 IA M 用户指南中的 AWS OpsWorks CM 的操作、资源和条件密钥。

条件键

AWS OpsWorks CM 没有可在策略声明`Condition`元素中使用的特定于服务的上下文密钥。有关所有服务都可使用的全局上下文密钥列表，请参阅《IAM policy 参考》中的AWS 全局条件上下文密钥。要查看所有 AWS 全局条件键，请参阅 IAM 用户指南中的AWS 全局条件上下文密钥。

在 Condition 元素（或 Condition 块）中，可以指定语句生效的条件。Condition 元素是可选的。您可以构建使用条件运算符（例如，等于或小于）的条件表达式，以使策略中的条件与请求中的值相匹配。

如果您在一个语句中指定多个 Condition 元素，或在单个 Condition 元素中指定多个键，则 AWS 使用逻辑 AND 运算评估它们。如果您为单个条件键指定多个值，则使用逻辑OR运算来 AWS 评估条件。在授予语句的权限之前必须满足所有的条件。

在指定条件时，您也可以使用占位符变量。例如，只有在资源被标记为用户名时，才能授予该用户访问该资源的权限。有关更多信息，请参阅 IAM 用户指南 中的 IAM policy 元素：变量和标签。

示例

要查看 AWS OpsWorks CM 基于身份的策略的示例，请参阅。AWS OpsWorks CM 基于身份的策略示例

AWS OpsWorks CM 和基于资源的政策

AWS OpsWorks CM 不支持基于资源的策略。

基于资源的策略是 JSON 策略文档，它们指定了指定委托人可以对资源执行的操作以及在什么条件下执行操作。

基于 AWS OpsWorks CM 标签的授权

您可以将标签附加到 AWS OpsWorks CM 资源或在请求中将标签传递给 AWS OpsWorks CM。要根据标签控制访问，您需要在策略的条件元素中使用 `aws:RequestTag/key-name` 或 `aws:TagKeys` 条件密钥提供标签信息。有关为 AWS OpsWorks CM 资源添加标签的更多信息，请参阅本指南使用 AWS OpsWorks for Puppet Enterprise 资源上的标签中的使用 AWS OpsWorks for Chef Automate 资源上的标签或。

AWS OpsWorks CM IAM 角色

I AM 角色是您的 AWS 账户中具有特定权限的实体。

AWS OpsWorks CM 使用两个角色：

- 一种服务角色，它向 AWS OpsWorks CM 服务授予在用户 AWS 账户中工作的权限。如果您使用 OpsWorks CM 提供的默认服务角色，则此角色的名称为`aws-opsworks-cm-service-role`。
- 允许 AWS OpsWorks CM 服务调用 C OpsWorks M API 的实例配置文件角色。此角色授予访问 Amazon S3 以及 AWS CloudFormation 创建用于备份的服务器和 S3 存储桶的权限。如果您使用

OpsWorks CM 提供的默认实例配置文件，则此实例配置文件角色的名称为`aws-opsworks-cm-ec2-role`。

AWS OpsWorks CM 不使用服务相关角色。

将临时凭证用于 AWS OpsWorks CM

AWS OpsWorks CM 支持使用临时证书，并从中 AWS Security Token Service继承该功能。

可以使用临时凭证进行联合身份验证登录，分派 IAM 角色或分派跨账户角色。您可以通过调用AssumeRole或之类的 AWS STS API 操作来获取临时安全证书GetFederationToken。

服务相关角色

AWS OpsWorks CM 不使用服务相关角色。

服务相关角色允许 AWS 服务访问其他服务中的资源以代表您完成操作。服务相关角色显示在 IAM 账户中，并归该服务所有。IAM 管理员可以查看但不能编辑服务相关角色的权限。

服务角色

此功能允许服务代表您担任服务角色。此角色允许服务访问其他服务中的资源以代表您完成操作。服务角色显示在 IAM 账户中，并归该账户所有。这意味着，IAM 管理员可以更改该角色的权限。但是，这样做可能会中断服务的功能。

AWS OpsWorks CM 使用两个角色：

- 一种服务角色，它向 AWS OpsWorks CM 服务授予在用户 AWS 账户中工作的权限。如果您使用 OpsWorks CM 提供的默认服务角色，则此角色的名称为`aws-opsworks-cm-service-role`。
- 允许 AWS OpsWorks CM 服务调用 C OpsWorks M API 的实例配置文件角色。此角色授予访问 Amazon S3 以及 AWS CloudFormation 创建用于备份的服务器和 S3 存储桶的权限。如果您使用 OpsWorks CM 提供的默认实例配置文件，则此实例配置文件角色的名称为`aws-opsworks-cm-ec2-role`。

在 AWS OpsWorks CM 中选择 IAM 角色

在 AWS OpsWorks CM 中创建服务器时，必须选择一个角色以允许 AWS OpsWorks CM 代表您访问 Amazon EC2。如果您已经创建了服务角色，那么 AWS OpsWorks CM 会为您提供可供选择的角色列表。 OpsWorks 如果您未指定角色，CM 可以为您创建角色。选择一个允许访问以启动和停止 Amazon

EC2 实例的角色很重要。有关更多信息，请参阅创建 Chef Automate 服务器或创建 Puppet Enterprise Master。

AWS OpsWorks CM 基于身份的策略示例

默认情况下，用户或角色无权创建或修改 AWS OpsWorks CM 资源。他们也无法使用 AWS Management Console AWS CLI、或 AWS API 执行任务。IAM 管理员必须创建 IAM policy，以便为角色授予 IAM 身份权限以对所需的指定资源执行特定的 API 操作。然后，管理员必须将这些策略附加到需要这些权限的用户或组。

要了解如何使用这些示例 JSON 策略文档创建基于 IAM 身份的策略，请参阅《IAM 用户指南》中的创建 IAM policy。

在 AWS OpsWorks CM 中，你可以将`AWSOpsWorksCMServiceRole`策略分配给用户，让用户使用或创建和管理 Chef Automate 或 Puppet Enterprise 服务器。 AWS Management Console AWS CLI

主题

策略最佳实践

基于身份的策略决定了某人是否可以在您的账户中创建、访问或删除 OpsWorks CM 资源。这些操作可能会使 AWS 账户产生成本。创建或编辑基于身份的策略时，请遵循以下指南和建议：

- 开始使用 AWS 托管策略并转向最低权限权限 — 要开始向用户和工作负载授予权限，请使用为许多常见用例授予权限的AWS 托管策略。它们在你的版本中可用 AWS 账户。我们建议您通过定义针对您的用例的 AWS 客户托管策略来进一步减少权限。有关更多信息，请参阅《IAM 用户指南》中的 AWS 托管策略 或 工作职能的AWS 托管策略。
- 应用最低权限 – 在使用 IAM policy 设置权限时，请仅授予执行任务所需的权限。为此，您可以定义在特定条件下可以对特定资源执行的操作，也称为最低权限许可。有关使用 IAM 应用权限的更多信息，请参阅《IAM 用户指南》中的 IAM 中的策略和权限。
- 使用 IAM policy 中的条件进一步限制访问权限 – 您可以向策略添加条件来限制对操作和资源的访问。例如，您可以编写策略条件来指定必须使用 SSL 发送所有请求。如果服务操作是通过特定的方式使用的，则也可以使用条件来授予对服务操作的访问权限 AWS 服务，例如 AWS CloudFormation。有关更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素：条件 。

- 使用 IAM Access Analyzer 验证您的 IAM policy，以确保权限的安全性和功能性 – IAM Access Analyzer 会验证新策略和现有策略，以确保策略符合 IAM policy语言 (JSON) 和 IAM 最佳实践。IAM Access Analyzer 提供 100 多项策略检查和可操作的建议，以帮助您制定安全且功能性强的策略。有关更多信息，请参阅《IAM 用户指南》中的 IAM Acess Analyzer 策略验证。
- 需要多重身份验证 (MFA)-如果 AWS 账户您的场景需要 IAM 用户或根用户，请启用 MFA 以提高安全性。若要在调用 API 操作时需要 MFA，请将 MFA 条件添加到您的策略中。有关更多信息，请参阅《IAM 用户指南》中的 配置受 MFA 保护的 API 访问。

有关 IAM 中的最佳实操的更多信息，请参阅《IAM 用户指南》中的 IAM 中的安全最佳实操。

允许用户查看他们自己的权限

此示例显示您可以如何创建策略，以便允许 用户查看附加到其用户身份的内联和托管策略。此策略包括在控制台上或使用 AWS CLI 或 AWS API 以编程方式完成此操作的权限。

```
{
       "Version": "2012-10-17",
       "Statement": [
           {
               "Sid": "ViewOwnUserInfo",
               "Effect": "Allow",
               "Action": [
                   "iam:GetUserPolicy",
                   "iam:ListGroupsForUser",
                   "iam:ListAttachedUserPolicies",
                   "iam:ListUserPolicies",
                   "iam:GetUser"
               ],
               "Resource": [
                   "arn:aws:iam::*:user/${aws:username}"
               ]
           },
           {
               "Sid": "NavigateInConsole",
               "Effect": "Allow",
               "Action": [
                   "iam:GetGroupPolicy",
                   "iam:GetPolicyVersion",
                   "iam:GetPolicy",
                   "iam:ListAttachedGroupPolicies",
                   "iam:ListGroupPolicies",
```

```
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

根据标签查看 AWS OpsWorks CM 服务器

您可以使用基于身份的策略中的条件根据标签控制对 AWS OpsWorks CM 服务器和备份的访问权限。此示例说明如何创建允许查看 AWS OpsWorks CM 服务器的策略。但是，只有当 AWS OpsWorks CM 服务器标签的值为Owner该用户的用户名时，才会授予权限。此策略还授予在控制台上完成此操作的必要权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ListServersInConsole",
            "Effect": "Allow",
            "Action": "opsworks-cm:DescribeServers",
            "Resource": "*"
        },
        {
            "Sid": "ViewServerIfOwner",
            "Effect": "Allow",
            "Action": "opsworks-cm:DescribeServers",
            "Resource": "arn:aws:opsworks-cm:region:master-account-ID:server/server-
name",
            "Condition": {
                "StringEquals": {"opsworks-cm:ResourceTag/Owner": "${aws:username}"}
            }
        }
    ]
}
```

您可以将此策略附加到您账户中的用户。如果名为的用户richard-roe尝试查看 AWS OpsWorks CM 服务器，则必须对该服务器进行标记Owner=richard-roe或owner=richard-roe。否则，将拒绝其访问。条件标签键 Owner 匹配 Owner 和 owner，因为条件键名称不区分大小写。有关更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素：条件 。

AWS OpsWorks CM 身份和访问疑难解答

使用以下信息可帮助您诊断和修复在使用 IAM 工作时可能遇到的常见问题。有关特定于 AWS OpsWorks CM 的疑难解答信息，请参阅故障排除 AWS OpsWorks for Chef Automate和Pupp OpsWorks et 企业版故障排除。

主题

- 我无权在 AWS OpsWorks CM 中执行操作
- 我无权执行 iam：PassRole
- 我想允许 AWS 账户之外的人访问我的 AWS OpsWorks CM 资源

我无权在 AWS OpsWorks CM 中执行操作

如果 AWS Management Console 告诉您您无权执行某项操作，则必须联系管理员寻求帮助。管理员是向您提供登录凭证的人。

当用户`mateojackson`尝试使用控制台查看有关 AWS OpsWorks CM 服务器的详细信息但没有`opsworks-cm:DescribeServers`权限时，会出现以下示例错误。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 opsworks-cm:DescribeServers on resource: test-chef-automate-server
```

在此情况下，Mateo 请求管理员更新策略，以允许其使用 `opsworks-cm:DescribeServers` 操作访问 `test-chef-automate-server` 资源。

我无权执行 iam：PassRole

如果您收到错误消息，提示您无权执行 `iam:PassRole` 操作，则必须联系您的管理员寻求帮助。管理员是向您提供登录凭证的人。要求该人更新您的政策，允许您将角色传递给 OpsWorks CM。

某些 AWS 服务允许您将现有角色传递给该服务，而不是创建新的服务角色或服务相关角色。为此，您必须具有将角色传递到服务的权限。

当名为的用户`marymajor`尝试使用控制台在 OpsWorks CM 中执行操作时，会出现以下示例错误。但是，服务必须具有服务角色所授予的权限才可执行操作。Mary 不具有将角色传递到服务的权限。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在这种情况下，Mary 请求她的管理员来更新其策略，以允许她执行 `iam:PassRole` 操作。

我想允许 AWS 账户之外的人访问我的 AWS OpsWorks CM 资源

您可以创建一个角色，以便其他账户中的用户或您组织外的人员可以使用该角色来访问您的资源。您可以指定谁值得信赖，可以担任角色。对于支持基于资源的策略或访问控制列表（ACL）的服务，您可以使用这些策略向人员授予对您的资源的访问权。

要了解更多信息，请参阅以下内容：

- AWS OpsWorks CM 支持向来自多个账户的用户授予管理 AWS OpsWorks CM 服务器的访问权限。
- 要了解如何通过您拥有的 AWS 账户提供对资源的访问权限，请参阅 IAM 用户指南中的向您拥有的另一个 AWS 账户中的 IAM 用户提供访问权限。
- 要了解如何向第三方 AWS 账户提供对您的资源的访问权限，请参阅 IAM 用户指南中的向第三方 AWS 账户提供访问权限。
- 要了解如何通过联合身份验证提供访问权限，请参阅《IAM 用户指南》中的为经过外部身份验证的用户（联合身份验证）提供访问权限。
- 要了解使用角色和基于资源的策略进行跨账户访问之间的差别，请参阅《IAM 用户指南》中的 IAM 角色与基于资源的策略有何不同。

AWS OpsWorks Configuration Management 的 AWS 托管策略

要向用户、组和角色添加权限，与自己编写策略相比，使用 AWS 托管策略更简单。创建仅为团队提供所需权限的 IAM 客户托管策略需要时间和专业知识。要快速入门，您可以使用我们的 AWS 托管策略。这些策略涵盖常见使用案例，可在您的 AWS 账户中使用。有关 AWS 托管策略的更多信息，请参阅 IAM 用户指南中的 AWS 托管策略。

AWS 服务负责维护和更新 AWS 托管策略。您无法更改 AWS 托管策略中的权限。服务偶尔会向 AWS 托管式策略添加额外权限以支持新特征。此类更新会影响附加策略的所有身份（用户、组和角色）。当启动新特征或新操作可用时，服务最有可能会更新 AWS 托管式策略。服务不会从 AWS 托管式策略中删除权限，因此策略更新不会破坏您的现有权限。

此外，AWS 还支持跨多种服务的工作职能的托管策略。例如，ReadOnlyAccess AWS 托管策略提供对所有 AWS 服务和资源的只读访问权限。当服务启动新特征时，AWS 会为新操作和资源添加只读权限。有关工作职能策略的列表和说明，请参阅 IAM 用户指南中的适用于工作职能的 AWS 托管策略。

AWS 托管策略：**AWSOpsWorksCMServiceRole**

您可以将 AWSOpsWorksCMServiceRole 附加到您的 IAM 实体。OpsWorks CM 还会将此策略附加到服务角色，允许 OpsWorks CM 代表您执行操作。

此策略授予##权限，允许 OpsWorks CM 管理员创建、管理和删除 OpsWorks CM 服务器和备份。

权限详细信息

此策略包含以下权限。

- opsworks-cm—允许委托人删除现有服务器并开始运行维护。
- acm— 允许委托人从 AWS Certificate Manager 删除或导入允许用户连接到 OpsWorks CM 服务器的证书。
- cloudformation— 允许 OpsWorks CM 在委托人创建、更新或删除 OpsWorks CM 服务器时创建和管理 AWS CloudFormation 堆栈。
- ec2— 允许 OpsWorks CM 在委托人创建、更新或删除 OpsWorks CM 服务器时启动、预配置、更新和终止 Amazon Elastic Compute Cloud 实例。
- iam— 允许 OpsWorks CM 创建和管理 OpsWorks CM 服务器所需的服务角色。
- tag— 允许委托人在 OpsWorks CM 资源（包括服务器和备份）中应用和删除标签。
- s3— 允许 OpsWorks CM 创建用于存储服务器备份的 Amazon S3 存储桶，根据委托人请求管理 S3 存储桶中的对象（例如，删除备份），以及删除存储桶。
- secretsmanager— 允许 OpsWorks CM 创建和管理 Secrets Manager 密钥，以及应用或删除密钥中的标签。
- ssm— 允许 OpsWorks CM 在作为 OpsWorks CM 服务器的实例上使用 Systems Manager 运行命令。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::aws-opsworks-cm-*"
            ],
            "Action": [
                "s3:CreateBucket",
                "s3:DeleteObject",
```

```
                "s3:DeleteBucket",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:PutBucketPolicy",
                "s3:PutObject",
                "s3:GetBucketTagging",
                "s3:PutBucketTagging"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": [
                "*"
            ],
            "Action": [
                "tag:UntagResources",
                "tag:TagResources"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": [
                "*"
            ],
            "Action": [
                "ssm:DescribeInstanceInformation",
                "ssm:GetCommandInvocation",
                "ssm:ListCommandInvocations",
                "ssm:ListCommands"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringLike": {
                    "ssm:resourceTag/aws:cloudformation:stack-name": "aws-opsworks-cm-
*"
                }
            },
            "Action": [
                "ssm:SendCommand"
```

```
            ]
        },
        {
            "Effect": "Allow",
            "Resource": [
                "arn:aws:ssm:*::document/*",
                "arn:aws:s3:::aws-opsworks-cm-*"
            ],
            "Action": [
                "ssm:SendCommand"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": [
                "*"
            ],
            "Action": [
                "ec2:AllocateAddress",
                "ec2:AssociateAddress",
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:CreateImage",
                "ec2:CreateSecurityGroup",
                "ec2:CreateSnapshot",
                "ec2:CreateTags",
                "ec2:DeleteSecurityGroup",
                "ec2:DeleteSnapshot",
                "ec2:DeregisterImage",
                "ec2:DescribeAccountAttributes",
                "ec2:DescribeAddresses",
                "ec2:DescribeImages",
                "ec2:DescribeInstanceStatus",
                "ec2:DescribeInstances",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSnapshots",
                "ec2:DescribeSubnets",
                "ec2:DisassociateAddress",
                "ec2:ReleaseAddress",
                "ec2:RunInstances",
                "ec2:StopInstances"
            ]
        },
        {
            "Effect": "Allow",
```

```
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringLike": {
                    "ec2:ResourceTag/aws:cloudformation:stack-name": "aws-opsworks-cm-
*"
                }
            },
            "Action": [
                "ec2:TerminateInstances",
                "ec2:RebootInstances"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": [
                "arn:aws:opsworks-cm:*:*:server/*"
            ],
            "Action": [
                "opsworks-cm:DeleteServer",
                "opsworks-cm:StartMaintenance"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": [
                "arn:aws:cloudformation:*:*:stack/aws-opsworks-cm-*"
            ],
            "Action": [
                "cloudformation:CreateStack",
                "cloudformation:DeleteStack",
                "cloudformation:DescribeStackEvents",
                "cloudformation:DescribeStackResources",
                "cloudformation:DescribeStacks",
                "cloudformation:UpdateStack"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": [
                "arn:aws:iam::*:role/aws-opsworks-cm-*",
                "arn:aws:iam::*:role/service-role/aws-opsworks-cm-*"
            ],
```

```
            "Action": [
                "iam:PassRole"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": "*",
            "Action": [
                "acm:DeleteCertificate",
                "acm:ImportCertificate"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": "arn:aws:secretsmanager:*:*:opsworks-cm!aws-opsworks-cm-
secrets-*",
            "Action": [
                "secretsmanager:CreateSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:UpdateSecret",
                "secretsmanager:DeleteSecret",
                "secretsmanager:TagResource",
                "secretsmanager:UntagResource"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "ec2:DeleteTags",
            "Resource": [
                "arn:aws:ec2:*:*:instance/*",
                "arn:aws:ec2:*:*:elastic-ip/*",
                "arn:aws:ec2:*:*:security-group/*"
            ]
        }
    ]
}
```

AWS 托管策略：**AWSOpsWorksCMInstanceProfileRole**

您可以将 AWSOpsWorksCMInstanceProfileRole 附加到您的 IAM 实体。OpsWorks CM 还会将此策略附加到服务角色，允许 OpsWorks CM 代表您执行操作。

该策略授予##权限，允许用作 OpsWorks CM 服务器的 Amazon EC2 实例从 AWS CloudFormation 和 AWS Secrets Manager 获取信息，并将服务器备份存储在 Amazon S3 存储桶中。

权限详细信息

此策略包含以下权限。

- `acm`— 允许 OpsWorks CM 服务器 EC2 实例从 AWS Certificate Manager 中获取证书，允许用户连接到 OpsWorks CM 服务器。
- `cloudformation`— 允许 OpsWorks CM 服务器 EC2 实例在实例创建或更新过程中获取关于 AWS CloudFormation 堆栈的信息，并向 AWS CloudFormation 发送有关其状态的信号。
- `s3`— 允许 OpsWorks CM 服务器 EC2 实例上传服务器备份并将其存储在 S3 存储桶中，必要时停止或回滚上传，以及从 S3 存储桶中删除备份。
- `secretsmanager`— 允许 OpsWorks CM 服务器 EC2 实例获取与 OpsWorks CM 相关的 Secrets Manager 密钥的值。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "cloudformation:DescribeStackResource",
                "cloudformation:SignalResource"
            ],
            "Effect": "Allow",
            "Resource": [
                "*"
            ]
        },
        {
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:DeleteObject",
                "s3:GetObject",
                "s3:ListAllMyBuckets",
                "s3:ListBucket",
                "s3:ListMultipartUploadParts",
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::aws-opsworks-cm-*",
```

```
            "Effect": "Allow"
        },
        {
            "Action": "acm:GetCertificate",
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "arn:aws:secretsmanager:*:*:opsworks-cm!aws-opsworks-cm-
secrets-*",
            "Effect": "Allow"
        }
    ]
}
```

OpsWorks CM 对 AWS 托管策略的更新

查看有关 OpsWorks CM 的 AWS 托管策略更新的详细信息（从该服务开始跟踪这些更改开始）。有关此页面更改的自动提示，请订阅 OpsWorks CM 文档历史记录页面上的 RSS 源。

更改	说明	日期
awsopsworkscminstanceProfileRole-更新的托管策略	OpsWorks CM 更新了托管策略，允许用作 OpsWorks CM 服务器的 EC2 实例与 CloudFormation 和 Secrets Manager 共享信息并管理备份。此更改添加 `opsworks-cm!` 到 Secrets Manager 密钥的资源名称，因此允许 OpsWorks CM 拥有这些密钥。	2021 年 4 月 23 日
AWSOpsWorksCMServiceRole-更新的托管策略	OpsWorks CM 更新了托管策略，该策略允许 OpsWorks CM 管理员创建、管理和删除 OpsWorks CM 服务器和备份。此更改添加 `opsworks-cm!` 到 Secrets Manager	2021 年 4 月 23 日

更改	说明	日期
	密钥的资源名称，因此允许 OpsWorks CM 拥有这些密钥。	
OpsWorks CM 已开启跟踪更改	OpsWorks CM为其 AWS 托管式策略开启了跟踪更改。	2021 年 4 月 23 日

防止在 AWS OpsWorks CM 中出现跨服务混淆代理

混淆代理问题是一个安全性问题，即不具有操作执行权限的实体可能会迫使具有更高权限的实体执行该操作。在 AWS 中，跨服务模拟可能会导致混淆代理问题。一个服务（呼叫服务）调用另一项服务（所谓的服务)时，可能会发生跨服务模拟。可以操纵调用服务，使用其权限以在其他情况下该服务不应有访问权限的方式对另一个客户的资源进行操作。为防止这种情况，AWS 提供可帮助您保护所有服务的数据的工具，而这些服务中的服务委托人有权限访问账户中的资源。

我们建议使用资源策略中的 aws:SourceArn 和 aws:SourceAccount 全局条件上下文键，限制 AWS OpsWorks CM 为另一项服务提供的资源访问权限。如果 aws:SourceArn 值不包含账户 ID，例如 Amazon S3 存储桶 ARN，您必须使用两个全局条件上下文密钥来限制权限。如果同时使用全局条件上下文密钥和包含账户 ID 的 aws:SourceArn 值，则 aws:SourceAccount 值和 aws:SourceArn 值中的账户在同一策略语句中使用时，必须使用相同的账户 ID。如果您只希望将一个资源与跨服务访问相关联，请使用 aws:SourceArn。如果您想允许该账户中的任何资源与跨服务使用操作相关联，请使用 aws:SourceAccount。

aws:SourceArn 的值必须是 OpsWorks CM Chef 或 Puppet 服务器的 ARN。

防范混淆代理问题最有效的方法是使用 aws:SourceArn 全局条件上下文键和 AWS OpsWorks CM 服务器的完整 ARN。如果不知道完整 ARN，或者正在指定多个服务器 ARN，请针对 ARN 未知部分使用带有通配符（*）的 aws:SourceArn 全局上下文条件键。例如，arn:aws:*servicename*:*:*123456789012*:*。

以下一节演示如何使用 AWS OpsWorks CM 中的 aws:SourceArn 和 aws:SourceAccount 全局条件上下文键来防范混淆代理问题。

防止在 AWS OpsWorks CM 中出现混淆代理漏洞

本节介绍如何帮助防止在 AWS OpsWorks CM 中出现混淆代理漏洞，并列出了一些权限策略示例，您可以将这些示例附加到用于访问 AWS OpsWorks CM 的 IAM 角色。作为安全最佳实践，建议您向

IAM 角色与其他服务的信任关系中添加 aws:SourceArn 和 aws:SourceAccount 条件键。信任关系允许 AWS OpsWorks CM 承担这样一个角色：在其他服务中执行创建或管理 AWS OpsWorks CM Stacks 堆栈所需的操作。

编辑信任关系来添加 **aws:SourceArn** 和 **aws:SourceAccount** 条件键

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在左侧导航窗格中，选择 Roles（角色）。
3. 在搜索框中，搜索您用于访问 AWS OpsWorks CM 的角色。AWS 托管角色是 aws-opsworks-cm-service-role。
4. 在角色的 摘要 页面上，选择 信任关系 选项卡。
5. 在信任关系选项卡上，选择编辑信任关系。
6. 在策略文档中，向策略中添加至少一个 aws:SourceArn 或 aws:SourceAccount 条件键。使用 aws:SourceArn 将跨服务（例如 AWS Certificate Manager 和 Amazon EC2）和 AWS OpsWorks CM 之间的信任关系限制为特定的 AWS OpsWorks CM 服务器，限制性更强。添加 aws:SourceAccount 将跨服务与 AWS OpsWorks CM 之间的信任关系限制为特定账户中的服务器，限制性较低。以下是示例。请注意，如果您同时使用两个条件键，则账户 ID 必须相同。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "opsworks-cm.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:opsworks-cm:us-east-2:123456789012:server/my-
opsworks-server/EXAMPLEabcd-1234-efghEXAMPLE-ID"
        }
      }
    }
  ]
}
```

7. 添加完条件键，选择更新信任策略。

以下是使用 aws:SourceArn 和 aws:SourceAccount 来限制 AWS OpsWorks CM 服务器访问的其他角色示例。

主题

示例：访问特定区域的 AWS OpsWorks CM 服务器

以下角色信任关系语句访问美国东部（俄亥俄州）区域 (us-east-2) 中的任何 AWS OpsWorks CM 服务器。请注意，该区域在 aws:SourceArn 的 ARN 值中指定，但服务器 ID 值是通配符 (*)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "opsworks-cm.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:opsworks-cm:us-east-2:123456789012:server/*"
        }
      }
    }
  ]
}
```

示例：向 **aws:SourceArn** 添加多个服务器 ARN

以下示例限制对账户 ID 为 123456789012 的由两个 AWS OpsWorks CM 服务器组成的数组的访问。

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "opsworks-cm.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnEquals": {
          "aws:SourceArn": [
             "arn:aws:opsworks-cm:us-east-2:123456789012:server/my-chef-
server/unique_ID",
             "arn:aws:opsworks-cm:us-east-2:123456789012:server/my-puppet-
server/unique_ID"
           ]
       }
      }
    }
  ]
}
```

互联网络流量保密性

AWS OpsWorks CM 使用 AWS 通常使用的相同传输安全协议：HTTPS 或带 TLS 加密的 HTTP。

AWS OpsWorks CM 中的日志记录和监控

AWS OpsWorks CM 将所有 API 操作记录到 CloudTrail。有关更多信息，请参阅以下主题：

- 使用登录 OpsWorks Puppet 企业 API 调用 AWS CloudTrail
- 使用记录 AWS OpsWorks for Chef Automate API 调用 AWS CloudTrail

AWS OpsWorks CM 的合规性验证

AWS OpsWorks CM 支持以下合规性计划和法规：

- 支付卡行业 (PCI)
- 1996 年版健康保险流通与责任法案 (HIPAA)
- AWS 系统和组织控制 (SOC) 1、2 和 3
- 通用数据保护条例 (GDPR)

作为多个 AWS 合规性计划的一部分，第三方审计员将评估 AWS OpsWorks CM 的安全性和合规性。其中包括 SOC、PCI、FedRAMP、HIPAA 及其它。

有关特定合规性计划范围内的 AWS 服务的列表，请参阅合规性计划范围内的亚马逊云科技服务。有关一般信息，请参阅 AWS合规性计划。

您可以使用 AWS Artifact 下载第三方审计报告。有关更多信息，请参阅在 AWS Artifact 中下载报告。

您在使用 AWS OpsWorks CM 时的合规性责任由您数据的敏感性、公司的合规性目标以及适用的法律法规决定。AWS 提供以下资源来帮助满足合规性：

- 安全性与合规性快速入门指南 – 这些部署指南讨论了架构注意事项，并提供了在 AWS 上部署基于安全性和合规性的基准环境的步骤。
- 设计符合 HIPAA 安全性和合规性要求的架构白皮书 — 此白皮书介绍公司如何使用 AWS 创建符合 HIPAA 标准的应用程序。
- AWS 合规性资源 – 此业务手册和指南集合可能适用于您的行业和位置。
- AWS Config – 此 AWS 服务评估您的资源配置对内部实践、行业指南和法规的遵循情况。
- AWS Security Hub – 此 AWS 服务提供了 AWS 中安全状态的全面视图，可帮助您检查是否符合安全行业标准和最佳实操。

AWS OpsWorks CM 中的弹性

默认情况下，当您创建服务器时，AWS OpsWorks CM 会启用每日服务器备份。备份将进行加密并存储在 Amazon S3 存储桶中。默认情况下，仅创建服务器的账户有权访问此存储桶。您可以自行决定在 Amazon S3 中为其他用户账户添加存储桶访问权限或配置跨区域备份。Chef 和 Puppet 都支持跨区域加密，因为这两种产品都会对 AWS OpsWorks CM 服务器和托管节点之间的流量进行加密。

AWS OpsWorks CM 不支持高可用性 (HA) 配置。

AWS全球基础设施围绕 AWS 区域和可用区构建。AWS区域提供多个在物理上独立且隔离的可用区，这些可用区通过延迟低、吞吐量高且冗余性高的网络连接在一起。利用可用区，您可以设计和操作在

可用区之间无中断地自动实现故障转移的应用程序和数据库。与传统的单个或多个数据中心基础设施相比，可用区具有更高的可用性、容错性和可扩展性。

有关如何在 AWS OpsWorks CM 中备份和还原服务器的更多信息，请参阅以下内容：

- 备份和恢复 Puppe OpsWorks t Enterprise Server
- 备份和恢复 AWS OpsWorks for Chef Automate 服务器

有关 AWS 区域和可用区的更多信息，请参阅 AWS全球基础设施。

AWS OpsWorks CM 中的基础设施安全性

作为一项托管式服务，AWS OpsWorks Configuration Management 受 AWS 全球网络安全保护。有关 AWS 安全服务以及 AWS 如何保护基础设施的信息，请参阅 AWS 云安全。要按照基础设施安全最佳实践设计您的 AWS 环境，请参阅《安全性支柱 AWS Well-Architected Framework》中的基础设施保护。

您可以使用 AWS 发布的 API 调用通过网络访问 OpsWorks CM。客户端必须支持以下内容：

- 传输层安全性协议 (TLS) 我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 具有完全向前保密 (PFS) 的密码套件，例如 DHE（临时 Diffie-Hellman）或 ECDHE（临时椭圆曲线 Diffie-Hellman）。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

此外，必须使用访问密钥 ID 和与 IAM 委托人关联的秘密访问密钥来对请求进行签名。或者，您可以使用 AWS Security Token Service (AWS STS) 生成临时安全凭证来对请求进行签名。

AWS OpsWorks CM 不支持私有链接或 VPC 私有终端节点。

AWS OpsWorks CM 不支持基于资源的策略。有关更多信息，请转到AWS Identity and Access Management用户指南中的与 IAM 结合使用的 AWS服务。

AWS OpsWorks CM 中的配置和漏洞分析

AWS OpsWorks CM 对 AWS OpsWorks CM 服务器上运行的操作系统执行定期内核和安全更新。用户可以设置一个时段，以使自动更新从当前日期起最多持续两周。AWS OpsWorksCM 推动了 Chef 和 Puppet Enterprise 次要版本的自动更新。有关配置 AWS OpsWorks for Chef Automate 的更新的更多信息，请参阅本指南中的系统维护 (Chef)。有关配置 OpsWorks for Puppet Enterprise 的更新的更多信息，请参阅本指南中的系统维护 (Puppet)。

AWS OpsWorks CM 的安全最佳实践

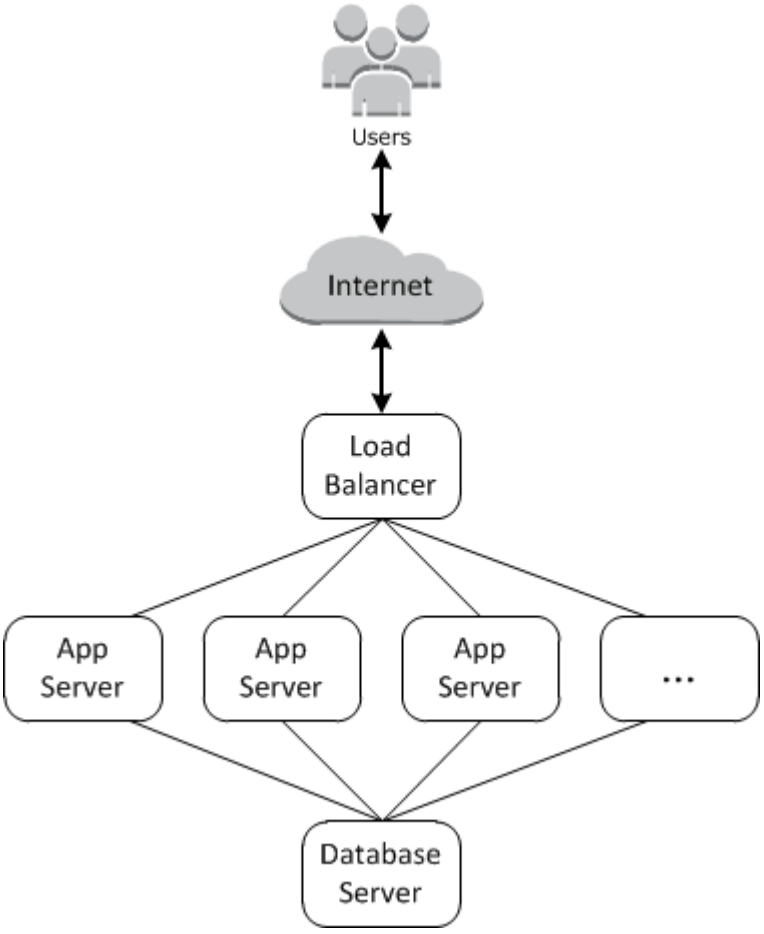
与所有 AWS OpsWorks 服务一样，AWS CM 提供了您在开发和实施自己的安全策略时需要考虑的安全功能。以下最佳实践是一般指导原则，并不代表完整安全解决方案。这些最佳实践可能不适合环境或不满足环境要求，请将其视为有用的考虑因素而不是惯例。

- 保护您的初学者工具包和下载的登录凭证。在创建新的 AWS OpsWorks CM 服务器或从 AWS OpsWorks CM 控制台下载新的初学者工具包和凭证时，请将这些项目存储在至少需要一个身份验证因素的安全位置。凭证提供对服务器的管理员级别访问权限。
- 保护您的配置代码。使用针对源存储库的推荐协议来保护您的 Chef 或 Puppet 配置代码（说明书和模块）。例如，您可以限制 AWS CodeCommit 中存储库的权限，或遵循 GitHub 网站上有关保护 GitHub 存储库的指南。
- 使用 CA 签名证书连接到节点。虽然您可以在 AWS OpsWorks CM 服务器上注册或引导节点时使用自签名证书，但最佳实践是使用 CA 签名证书。我们建议您使用由证书颁发机构 (CA) 签名的证书。
- 请勿与其他用户共享 Chef 或 Puppet 管理控制台登录凭证。管理员应为 Chef 或 Puppet 控制台网站的每个用户创建单独的用户。
  - 在 Chef Automate 中管理用户
  - 在 Puppet Enterprise 中管理用户
- 配置自动备份和系统维护更新。在 AWS OpsWorks CM 服务器上配置自动维护更新可帮助确保您的服务器正在运行与安全相关的最新操作系统更新。配置自动备份有助于在发生事故或故障时减小灾难恢复的难度并缩短恢复时间。限制对存储了 AWS OpsWorks CM 服务器备份的 Amazon S3 存储桶的访问权限；请勿向 每个人 授予访问权限。根据需要向其他用户单独授予读取或写入访问权限，或在 IAM 中为这些用户创建一个安全组，并将访问权限分配给该安全组。
  - 系统维护 (Chef)
  - 系统维护 (Puppet)
  - 备份和恢复 AWS OpsWorks for Chef Automate 服务器
  - 备份和恢复 Puppe OpsWorks t Enterprise Server
  - 在AWS Identity and Access Management用户指南中创建您的第一个 IAM 委派用户和组
  - 《Amazon Simple Storage Service 开发人员指南》中的 Amazon S3 安全最佳实践

AWS OpsWorks 堆栈

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

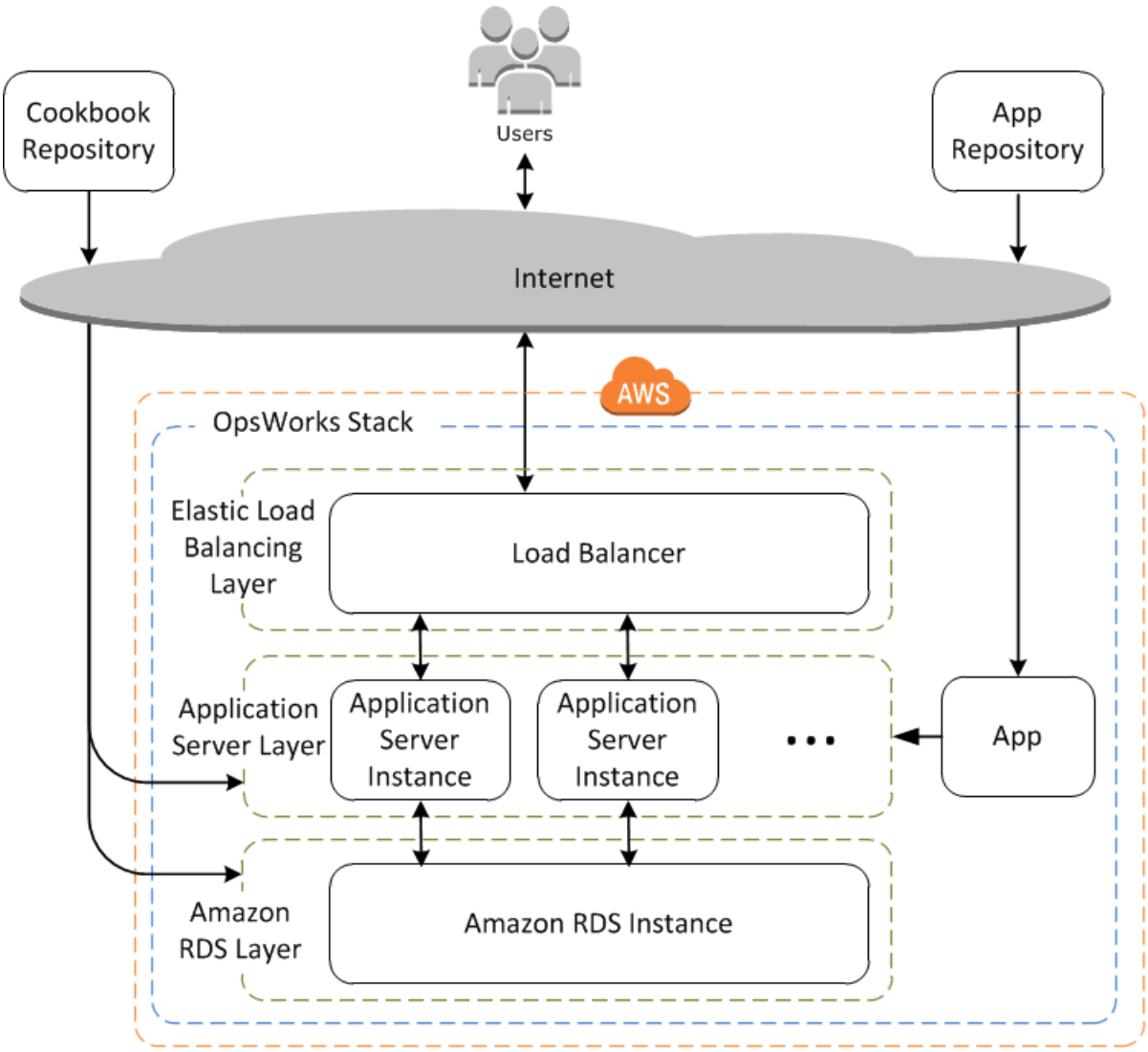
基于云的计算通常涉及 AWS 资源组，如 Amazon EC2 实例和 Amazon 关系数据库 (RDS) 实例，这些实例必须被共同创建和管理。例如，Web 应用程序通常需要应用程序服务器、数据库服务器、负载均衡器等。该组实例通常称为堆栈；一个简单的应用程序服务器堆栈可能像下面这样：



除了创建实例和安装必要的软件包外，您通常需要通过某种方式向应用程序服务器分发应用程序、监控堆栈的性能、管理安全性和权限，等等。

AWS OpsWorks Stacks 提供了一种简单而灵活的方式来创建和管理堆栈和应用程序。

以下是 Stacks 中基本应用服务器堆栈的 AWS OpsWorks 样子。它包含一组在 Elastic Load Balancing 负载均衡器背后运行的应用程序服务器，并且带一个后端 Amazon RDS 数据库服务器。



尽管相对简单，但此堆栈显示了 Stac AWS OpsWorks ks 的所有关键功能。下面介绍了这些功能是如何结合在一起的。

主题

- 堆栈
- 图层
- 食谱和 LifeCycle 活动
- 实例
- 应用程序
- 自定义您的堆栈

- 资源管理
- 安全性和权限
- 监控和日志记录
- CLI、软件开发工具包和 AWS CloudFormation 模板
- AWS OpsWorks Stacks 生命周期终结常见问题解答
- 将 AWS OpsWorks Stacks 应用程序迁移到 AWS Systems Manager 应用程序管理器
- 使用 “就地 AWS OpsWorks Stacks 分离” 工具
- AWS OpsWorks 堆栈入门
- AWS OpsWorks 堆栈最佳实践
- 堆栈
- 图层
- 实例
- 应用程序
- 说明书和诀窍
- 资源管理
- 标签
- 监控
- 安全性和权限
- AWS OpsWorks Stacks 对 Chef 12 Linux 的支持
- Support 以 AWS OpsWorks 堆栈形式支持以前的 Chef 版本
- 将 AWS OpsWorks 堆栈与其他 AWS 服务一起使用
- 使用 AWS OpsWorks Stacks CLI
- 调试和故障排除指南
- AWS OpsWorks Stacks Agent CLI
- AWS OpsWorks 堆栈数据包参考
- OpsWorks 代理变更

堆栈

堆栈是堆栈的核心 AWS OpsWorks 组件。它基本上是一个容器，用于存放 AWS 资源（Amazon EC2 实例、Amazon RDS 数据库实例等），这些资源具有共同的用途，应一起进行逻辑管理。堆栈帮助

您将这些资源作为一个组来管理，同时它还定义了一些默认配置设置，如实例的操作系统和 Amazon Web Services Region。如果您想要隔离一些堆栈组件以避免用户直接交互，可以在 VPC 中运行堆栈。

图层

可通过添加一个或多个层 来定义堆栈的组成。层代表一组服务特定目的 (如提供应用程序服务或承载数据库服务器) 的 Amazon EC2 实例。

您可以通过修改包的默认配置、添加 Chef 配方来执行诸如安装附加软件包等任务的方式自定义或扩展层。

对于所有堆栈， AWS OpsWorks 堆栈都包含服务层，它们代表以下 AWS 服务。

- Amazon Relational Database Service
- Elastic Load Balancing
- Amazon Elastic Container Service

层让您能够完全控制安装哪些软件包、如何配置它们以及如何部署应用程序等。

食谱和 LifeCycle 活动

层依靠 Chef 配方来处理诸如在实例上安装软件包、部署应用程序、运行脚本等任务。 AWS OpsWorks Stacks 的关键功能之一是一组生命周期事件（设置、配置、部署、取消部署和关闭），它们会在每个实例的适当时间自动运行一组指定的配方。

每个层都可以有一组配方分配给每个生命周期事件，它们会为该事件和该层处理各种任务。例如，在属于 Web 服务器层的实例完成启动后， AWS OpsWorks Stacks 会执行以下操作。

1. 运行该层的 Setup 配方，这会执行诸如安装和配置 Web 服务器之类的任务。
2. 运行该层的 Deploy 配方，这会将该层的应用程序从存储库部署到实例并执行相关任务，如重启服务等。
3. 在堆栈中的每个实例上运行 Configure 配方，以便每个实例可根据需要调整其配置来容纳新实例。

   例如，在运行负载均衡器的实例上，Configure 配方可以修改该负载均衡器的配置以包括新实例。

如果一个实例属于多个层， AWS OpsWorks Stacks 会运行每个层的配方，这样您就可以拥有一个支持 PHP 应用服务器和 MySQL 数据库服务器的实例。

如果您已经实现了配方，则可以将每个食谱分配给相应的图层和事件， AWS OpsWorks Stacks 会在适当的时间自动为您运行它们。您也可以随时手动运行配方。

实例

一个实例代表一个计算资源，如一个 Amazon EC2 实例。它定义资源的基本配置，如操作系统和大小。其他配置设置，例如弹性 IP 地址或 Amazon EBS 卷都由实例的层来定义。层的配方通过执行诸如安装和配置软件包和部署应用程序等任务来完成配置。

您可以使用 AWS OpsWorks Stacks 创建实例并将其添加到图层中。当您启动实例时， AWS OpsWorks Stacks 会使用实例及其层指定的配置设置启动一个 Amazon EC2 实例。Amazon EC2 实例完成启动之后， AWS OpsWorks Stacks 安装一个代理来处理实例和服务之间的通信，以及运行适当的配方来响应生命周期事件。

AWS OpsWorks 堆栈支持以下实例类型，其特点是它们的启动和停止方式。

- 全天候实例 手动启动并运行到您停止它们为止。
- 基于时间的实例由 AWS OpsWorks Stacks 按指定的每日和每周计划运行。

  它们允许您的堆栈自动调整实例的数量以顺应可预测的使用模式。
- AWS OpsWorks 堆栈根据指定的@@ 负载指标（例如 CPU 利用率）自动启动和停止基于负载的实例。

  它们允许您的堆栈自动调整实例的数量以顺应传入流量的变化。基于负载的实例仅可用于基于 Linux 的堆栈。

AWS OpsWorks 堆栈支持实例自动修复。如果代理停止与服务通信， AWS OpsWorks Stacks 会自动停止并重启实例。

您还可以将基于 Linux 的计算资源整合到堆栈之外创建的 AWS OpsWorks 堆栈中。

- 直接使用 Amazon EC2 控制台、CLI 或 API 创建的 Amazon EC2 实例。
- 您自己的硬件上运行的本地 实例，包括虚拟机上运行的实例。

注册其中一个实例后，它就会变成 AWS OpsWorks Stacks 实例，您可以用与使用 AWS OpsWorks Stacks 创建的实例大致相同的方式对其进行管理。

应用程序

您将应用程序和相关文件存储在存储库中，如 Amazon S3 存储桶。每个应用程序由一个 app 表示，它指定应用程序的类型，并包含将应用程序从存储库部署到您的实例所需的信息，如存储库 URL 和密码。部署应用程序时， AWS OpsWorks Stacks 会触发 Deploy 事件，该事件会在堆栈的实例上运行 Deploy 配方。

您可以通过下列方式来部署应用程序：

- 自动-当您启动实例时， AWS OpsWorks Stacks 会自动运行实例的 Deploy 配方。
- 手动 - 如果您有新的应用程序或打算更新现有应用程序，可以手动运行在线实例的 Deploy 配方。

您通常会让 AWS OpsWorks Stacks 在整个堆栈上运行 Deploy 配方，这样其他层的实例就可以相应地修改其配置。不过，您可以将部署限制为一个实例子集，例如，如果您要测试新的应用程序以便将其部署到每个应用程序服务器实例的话。

自定义您的堆栈

AWS OpsWorks Stacks 提供了各种自定义层的方法来满足您的特定要求：

- 您可以通过覆盖代表各种配置设置的属性，甚至覆盖用于创建配置文件的模板来修改 AWS OpsWorks Stacks 配置软件包的方式。
- 您可以扩展现有层，方法是提供您自己的配方来执行诸如运行脚本或安装和配置非标准软件包等任务。

所有堆栈都可以包含一个或多个层，最开始只有最低限度的一组配方。您通过实施配方向层添加功能以处理诸如安装软件包、部署应用程序等任务。您将您的自定义配方和相关文件打包为一个或多个说明书，并将说明书存储在诸如 Amazon S3 或 Git 等的存储库中。

你可以手动运行配方，但是 AWS OpsWorks Stacks 还允许你通过支持一组五个生命周期事件来实现流程自动化：

- Setup 发生在成功启动后的新实例上。
- Configure 发生在堆栈的所有实例上 (在实例上线或离线时)。
- Deploy 发生在部署应用程序时。
- Undeploy 发生在删除应用程序时。
- Shutdown 发生在停止实例时。

每个层可以有任意数量的配方分配给每个事件。当图层的实例上发生生命周期事件时， AWS OpsWorks Stacks 会运行相关的配方。例如，当应用程序服务器实例上发生 Deploy 事件时， AWS OpsWorks Stacks 会运行该层的 Deploy 配方来下载应用程序或执行相关任务。

资源管理

您可以将其他 AWS 资源（如弹性 IP 地址）纳入您的堆栈。您可以使用 AWS OpsWorks Stacks 控制台或 API 向堆栈注册资源，将注册的资源附加到实例或将其与实例分离，以及将资源从一个实例移动到另一个实例。

安全性和权限

AWS OpsWorks Stacks 与 AWS Identity and Access Management (IAM) 集成，提供了控制用户如何访问 AWS OpsWorks 堆栈的强大方法，包括：

- 单个用户如何与每个堆栈交互，例如，他们能否创建堆栈资源 (如层和实例)，或者，能否使用 SSH 或 RDP 连接到堆栈的 Amazon EC2 实例。
- AWS OpsWorks Stacks 如何代表您与 Amazon EC2 实例等 AWS 资源进行交互。
- 在 AWS OpsWorks Stacks 实例上运行的应用程序如何访问 AWS 资源，例如 Amazon S3 存储桶。
- 如何管理用户的公有 SSH 密钥和 RDP 密码并连接到实例。

监控和日志记录

AWS OpsWorks Stacks 提供了多种功能，可帮助您监控堆栈并解决堆栈和任何配方的问题。对于所有堆栈：

- AWS OpsWorks Stacks 为 Linux 堆栈提供了一组自定义 CloudWatch 指标，为了方便起见，在 “监控” 页面上对这些指标进行了汇总。

  AWS OpsWorks 堆栈支持 Windows 堆栈的标准 CloudWatch 指标。您可以使用 CloudWatch 控制台对其进行监控。
- CloudTrail 日志，用于记录您的 AWS 账户中由 Stacks 或代表 AWS OpsWorks 堆栈进行的 API 调用。
- 事件日志，其中列出您的堆栈中的所有事件。
- Chef 日志，其中详细记录每个实例上每个生命周期事件的细节，例如，运行了哪些配方以及发生了哪些错误。

基于 Linux 的堆栈也可以包含一个 Ganglia 主层，您可以使用它来收集和显示您的堆栈中实例的详细监控数据。

CLI、软件开发工具包和 AWS CloudFormation 模板

除控制台外， AWS OpsWorks Stacks 还支持命令行界面 (CLI) 和多种语言的 SDK，可用于执行任何操作。请考虑以下功能：

- AWS OpsWorks Stacks CLI 是 AWS CLI 的一部分，可用于从命令行执行任何操作。

  AWS CLI 支持多种 Amazon Web Service，可安装在 Windows、Linux 或 OS X 系统上。
- AWS OpsWorks 堆栈包含在适用于 Windows PowerShell 的 AWS 工具中，可用于从 Windows PowerShell 命令行执行任何操作。
- AWS OpsWorks Stacks 软件开发工具包包含在 AWS 开发工具包中，可用于：Java、JavaScript（基于浏览器和 Node.js）、.NET、PHP 、 Pyth on（bot o）或 Ruby 实现的应用程序。

您也可以使用 AWS CloudFormation 模板来配置堆栈。有关一些示例，请参阅 AWS OpsWorks 代码片段。

AWS OpsWorks Stacks 生命周期终结常见问题解答

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。

主题

- 生命周期终止将如何影响现有客户？
- 正在 AWS OpsWorks Stacks 接受新客户吗？
- 我应该将现有堆栈迁移到哪里？
- 在生命周期结束后，如何保留我现有的 Amazon EC2 实例？
- 生命的尽头会同时影响所有 AWS 区域 人吗？
- 可获得什么级别的技术支持 AWS OpsWorks Stacks？

- 会有新功能发布吗 AWS OpsWorks Stacks？

生命周期终止将如何影响现有客户？

在 2024 年 5 月 26 日之前，也就是 AWS OpsWorks Stacks的生命周期终止日期之前，现有客户将不受影响。2024 年 5 月 26 日之后，客户将无法使用 OpsWorks 控制台、API、CLI 和 CloudFormation 资源。

正在 AWS OpsWorks Stacks 接受新客户吗？

不是。 AWS OpsWorks Stacks 不再接受新客户，目前只有现有客户才能创建新堆栈。

我应该将现有堆栈迁移到哪里？

我们建议 AWS OpsWorks Stacks 客户将其工作负载迁移到可以利用以下功能 AWS Systems Manager 的地方：

- 现代 Chef 版本
- SSM Agent
- 应用程序负载均衡器
- 通过 Auto Scaling 组增强扩展功能
- 能够使用 EC2 启动模板定义所需的主机特征
- 较新的实例类型
- 较新的 EBS 卷类型

有关 Systems Manager 的更多信息，请参阅 AWS Systems Manager 用户指南。有关迁移到的信息 AWS Systems Manager，请参见 将 AWS OpsWorks Stacks 应用程序迁移到 AWS Systems Manager 应用程序管理器

在生命周期结束后，如何保留我现有的 Amazon EC2 实例？

到生命周期结束日期后，您的 Amazon EC2 实例将保留在您的账户中，但您将无法再使用 OpsWorks Stacks 服务来控制和管理这些实例。

您可以使用 “就地 AWS OpsWorks Stacks 分离” 工具将您的 OpsWorks 实例与 OpsWorks Stacks 服务分离。分离后，您可以使用 Amazon EC2 或任何与 EC2 兼容的方法来配置和管理实例。 AWS Systems Manager有关更多信息，请参阅 使用 “就地 AWS OpsWorks Stacks 分离” 工具。

生命的尽头会同时影响所有 AWS 区域 人吗？

是。 OpsWorks 控制台、API、CLI 和 CloudFormation 资源将于 2024 年 5 月 26 日 AWS 区域 同时停产。有关可用 AWS 区域 地点 AWS OpsWorks Stacks 的列表，请参阅AWS 区域服务列表。

可获得什么级别的技术支持 AWS OpsWorks Stacks？

AWS 在生命周期终止日期之前 AWS OpsWorks Stacks ，将继续为客户提供与当今相同水平的支持。如果您有任何疑问或疑虑，可以通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

会有新功能发布吗 AWS OpsWorks Stacks？

没有。由于该服务已将至生命周期终止，因此我们不会发布任何新功能。但是，在生命周期终止之前，我们将继续改进安全性并按预期管理 Amazon EC2 实例。

将 AWS OpsWorks Stacks 应用程序迁移到 AWS Systems Manager 应用程序管理器

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。

现在，您可以使用迁移脚本将 AWS OpsWorks Stacks 应用程序迁移到应用程序管理器 AWS Systems Manager，这是一项功能。将堆栈应用程序迁移到 Systems Manager Application Manager 允许您使用中未提供的 AWS 功能 AWS OpsWorks Stacks，例如新的 Amazon EC2 实例类型（例如 Graviton）、新的亚马逊弹性区块存储 (EBS) 卷（如 gp3）、新的操作系统、与 Auto Scaling 组的集成以及应用程序负载均衡器。

在此版本中，您现在可以使用 Systems Manager Application Manager 提供的新实例选项卡对迁移的实例进行监控和运行操作。您可以使用 “实例” 选项卡在一个位置查看多个 AWS 实例。使用此选项卡，您可以查看有关实例运行状况的信息并对问题进行故障排除。有关使用实例选项卡的更多信息，请参阅AWS Systems Manager 用户指南中的使用应用程序实例。

主题

脚本的工作原理

AWS OpsWorks 提供了一个脚本，您可以使用该脚本使用 CloudFormation 模板将 AWS OpsWorks Stacks 应用程序迁移到 Systems Manager 应用程序管理器。该脚本获取有关现有 OpsWorks 层的信息，根据脚本`--provision-application`参数的值，要么配置应用程序的克隆，要么提供可以修改的入门 CloudFormation模板 AWS CloudFormation。

先决条件

- 请确保 AWS CLI 已安装和配置。有关安装的更多信息 AWS CLI，请参阅《AWS Command Line Interface 用户指南》 AWS CLI中的安装或更新最新版本的。

  > **Note**
  >
  > 如果您不想配置 AWS CLI，也可以使用运行命令 AWS CloudShell。有关使用的更多信息 CloudShell，请参阅《AWS CloudShell 用户指南》 AWS CloudShell中的使用。

- 请确保 Python 3.6 或更高版本已安装或具有亚马逊机器映像 (AMI)。
- 请确保您的操作系统受支持。您可以在以下操作系统上下载并运行迁移脚本。
  - Amazon Linux 和 Amazon Linux 2
  - Ubuntu 18.04 LTS, 20.04 LTS, 22.04 LTS
  - Red Hat Enterprise Linux 8
  - Windows Server 2019、Windows 10 企业版

  > **Note**
  >
  > 不支持 Windows Server 2022。

限制

新 OpsWorks 架构不同于的架构 AWS OpsWorks Stacks。本节介绍此架构的已知局限性。

新 OpsWorks 架构不支持以下内容。

- 在 Windows 和 CentOS 实例上运行 Chef 配方
- 内置 Chef 11 层和 Berkshelf
- Chef 属性和数据包
- 本地实例
- 从 EC2 导入的实例
- 不支持安装用户指定的操作系统包列表
- 不支持或迁移应用程序

支持以下功能，但有限制。

- 迁移脚本克隆 EBS 卷信息，但不包括装载点和卷中包含的实际数据。
- 将迁移基于时间和负载的扩展实例，但不会迁移与这些实例关联的任何扩展规则。您可以修改自动扩缩组以获得类似结果。
- 不会创建或生成 OpsWorks 控制台中堆栈权限页面中定义的 IAM 实体。
- 迁移脚本只能在 Systems Manager 中预配置单层应用程序。例如，如果您为同一堆栈中的两个层运行脚本两次，则在 Systems Manager 中会出现两个不同的应用程序。

开始使用

迁移脚本 `stack_exporter.py` 是一个 Python 脚本，可以在本地运行，也可以在 EC2 实例上运行。在运行脚本之前，请确保满足所有先决条件。有关这些先决条件的更多信息，请参阅 先决条件 。

以下各节中的步骤向您展示了如何将 OpsWorks 堆栈迁移到 Systems Manager 应用程序管理器。

主题

- 步骤 5：配置堆 CloudFormation 栈
- 步骤 6：审查预配置的资源
- 步骤 7：启动实例
- 步骤 8：审查实例
- 第 9 步：使用 Systems Manager Application Manager 监控和运行实例上的操作

步骤 1：准备脚本运行环境

运行适用于您操作系统的命令，准备环境。

主题

- Amazon Linux 2
- Amazon Linux
- Ubuntu 18.04、20.04、22.04
- Red Hat Enterprise Linux 8
- Windows Server 2019、Windows 10 企业版

Amazon Linux 2

```
sudo su
python3 -m pip install pipenv
PATH="$PATH:/usr/local/bin"
yum update
yum install git
```

Amazon Linux

```
sudo su
PATH="$PATH:/usr/local/bin"
export LC_ALL=en_US.utf-8
export LANG=en_US.utf-8
yum update
yum list | grep python3
yum install python36 // Any python version
yum install git
```

对于 Python 3.6，还要运行：

```
python3 -m pip install pipenv==2022.4.8
```

对于 Python 3.7或更高版本，还要运行：

```
python3 -m pip install pipenv
```

Ubuntu 18.04、20.04、22.04

```
sudo su
export PATH="${HOME}/.local/bin:$PATH"
apt-get update
apt install python3-pip
apt-get install git // if git is not installed
python3 -m pip install --user pipenv==2022.4.8
```

Red Hat Enterprise Linux 8

```
sudo su
sudo dnf install python3
PATH="$PATH:/usr/local/bin"
yum update
yum install git
python3 -m pip install pipenv==2022.4.8
```

Windows Server 2019、Windows 10 企业版

> **Note**
>
> 对于 Windows Server 2019，安装 Python 3.6.1 或更高版本。

```
pip install pipenv
```

下载并安装 Git（如果尚未安装）。

如果您使用 Git 作为说明书源，请先将您的 Git 服务器添加到 known_hosts 文件中，然后再在 Windows 上运行脚本。您可以使用 PowerShell 创建以下函数。

```
function add_to_known_hosts($server){
    $new_host=$(ssh-keyscan $server 2> $null)
```

```
    $existing_hosts=''
    if (!(test-path "$env:userprofile\.ssh")) {
        md "$env:userprofile\.ssh"
    }
    if ((test-path "$env:userprofile\.ssh\known_hosts")) {
        $existing_hosts=Get-Content "$env:userprofile\.ssh\known_hosts"
    }
    $host_added=0
    foreach ($line in $new_host) {
        if (!($existing_hosts -contains $line)) {
            Add-Content -Path "$env:userprofile\.ssh\known_hosts" -Value $line
            $host_added=1
    }
   }
   if ($host_added) {
       echo "$server has been added to known_hosts."
   } else {
       echo "$server already exists in known_hosts."
   }
}
```

然后当您运行该函数时，您可以提供你的 Git 服务器（例如 github.com、git-codecommit.*repository_region*.amazonaws.com）。

```
add_to_known_hosts "myGitServer"
```

步骤 2：下载迁移脚本

运行以下命令下载包含迁移脚本和所有相关文件的 zip 文件。

```
aws s3api get-object \
    --bucket export-opsworks-stacks-bucket-prod-us-east-1 \
    --key export_opsworks_stacks_script.zip export_opsworks_stacks_script.zip
```

如果您使用的是 Linux，请使用以下命令安装解压缩实用程序。

```
sudo apt-get install unzip
sudo yum install unzip
```

运行适用于您的操作系统的命令解压文件。

对于 Linux，请使用以下命令。

```
unzip export_opsworks_stacks_script.zip
```

对于 Windows，请使用中的`Expand-Archive`命令 PowerShell。

```
Expand-Archive -LiteralPath PathToZipFile -DestinationPath PathToDestination
```

文件解压缩后，以下目录和文件可用。

- README.md
- LICENSE
- NOTICE
- requirements.txt
- templates/
  - OpsWorkscfnTemplate.yaml
  - MountEBSVolumes.yaml
- opsworks/
- cloudformation/
- instances_tab/
- cfn_stack_deployer.py
- s3.py
- stack_exporter_context.py
- stack_exporter.py

步骤 3：设置环境运行脚本

使用以下命令设置环境以运行脚本。

```
pipenv install -r requirements.txt
pipenv shell
```

> **Note**
>
> 目前，该脚本只能在 Application Manager 中预配置单层应用程序。例如，如果您为同一堆栈中的两个层运行脚本两次，则脚本将在 Application Manage 中创建两个不同的应用程序。

设置环境后，请查看脚本参数。您可以通过运行 `python3 stack_exporter.py --help` 命令来查看迁移脚本的可用选项。

参数	描述	必填	类型	默认值
`--layer-id`	导出此 OpsWorks 图层 ID 的 CloudFormation 模板。	是	字符串	
`--region`	OpsWorks 堆栈的 AWS 区域。如果您的 OpsWorks 堆栈区域和 API 终端节点区域不同，请使用堆栈区域。此区域与 OpsWorks堆栈中的其他资源部分（例如，EC2 实例和子网）属于同一区域。	否	字符串	us-east-1
`--provision-application`	默认情况下，该脚本会置备由 CloudFormation 模板导出的应用程序。将此参数传递到脚本中，值为 FALSE 可跳过 CloudFormation 模板的配置。	否	布尔值	TRUE
`--launch-template`	此参数定义是否使用现有启动模板还是创建新的启动模板。您可以创建使用推荐的实例属性的新启动模板，也可以创建使用与在线实例匹配的实例属性的新启动模板。 有效值包括： • RECOMMENDED -使用来自最新 AMI 的实例特征作为 OpsWorks 堆栈操作系统和 c5.large 实例大小。 • MATCH_LAST_INSTANCE -使用最新的可用在线实例特征。 • *LaunchTemplateID* / [*LaunchTemplateVersion*] ：选择现有启动模板。	否	字符串	RECOMMENDED

参数	描述	必填	类型	默认值
	（可选）您可以提供模板版本。如果您未提供模板版本，则脚本将使用默认版本。			
--system-updates	定义是否在实例启动时执行内核和软件包更新。 有效值包括： • ALL_UPDATES -在实例启动时为内核和软件包执行系统更新。 • NO_UPDATES -实例启动时不执行系统更新。 • MATCH_LAYER_SETTINGS -使用 OpsWorks图层或实例的InstallUpdatesOnBoot 属性来确定是否安装系统更新。	否	字符串	ALL_UPDATES
--http-username	Systems Manager SecureString 参数的名称，该参数存储用于对包含自定义说明书的 HTTP 存档进行身份验证的用户名。	否	字符串	
--http-password	Systems Manager SecureString 参数的名称，该参数存储用于对包含自定义说明书的 HTTP 存档进行身份验证的密码。	否	字符串	

参数	描述	必填	类型	默认值
`--repo-private-key`	Systems Manager `SecureString` 参数的名称，该参数存储用于对包含自定义说明书的存储库进行身份验证的 SSH 密钥。如果存储库已开启 GitHub，则必须生成新的 `Ed25519` SSH 密钥。如果不生成新的 `Ed25519` SSH 密钥，则与 GitHub 存储库的连接将失败。	否	字符串	
`--lb-type`	迁移现有负载均衡器时要创建的负载均衡器的类型（如果有）。 有效值包括： • `ALB` (应用程序负载均衡器) • `Classis` (经典负载均衡器) • `None`（如果您不想创建负载均衡器）	否	字符串	`ALB`
`--lb-access-logs-path`	现有 S3 存储桶的路径和存储负载均衡器访问日志的前缀。该 S3 存储桶与负载均衡器必须位于同一区域。如果您未提供值且 `--lb-type` 参数值设置为 `None`，则脚本会创建新的 S3 存储桶和前缀。请确保此前缀有适当的存储桶策略。	否	字符串	

参数	描述	必填	类型	默认值
--enable-instance-protection	如果设置为 TRUE，则脚本会为您的自动扩缩组创建自定义终止策略（Lambda 函数）。带有 protected_instance 标签的 EC2 实例受到保护，免受横向缩减事件的影响。为每个 EC2 实例添加一个 protected_instance 标签，使其免受横向缩减事件的影响。	否	布尔值	FALSE
--command-logs-bucket	用于存储 AWS ApplyChefRecipe 和 MountEBSVolumes 日志的现有 S3 存储桶名称。如果您未提供值，脚本将创建一个新的 S3 存储桶。	否	字符串	aws-opsworks-application-manager-logs-*account-id*
--custom-json-bucket	用于存储自定义 JSON 的现有 S3 存储桶名称。如果您未提供值，脚本将创建一个新的 S3 存储桶。	否	字符串	aws-apply-chef-application-manager-transition-data-*account-id*

备注：

- 如果您使用私有 GitHub 存储库，则必须为 SSH 创建新的Ed25519主机密钥。这是因为 GitHub 更改了 SSH 中支持的密钥并删除了未加密的 Git 协议。有关Ed25519主机密钥的更多信息，请参阅 GitHub博客文章 “改进 Git 协议安全” GitHub。生成新的 Ed25519 主机密钥后，为 SSH 密钥创建一个 Systems Manager SecureString 参数，并将 SecureString 参数名称用作 --repo-private-key 参数的值。有关如何创建 System SecureString s Manager 参数的更多信息，请参阅《AWS Systems Manager 用户指南》中的创建 SecureString 参数 (AWS CLI) 或创建 Systems Manager 参数（控制台）。

- `--http-username`、`--http-password` 和 `--repo-private-key` 参数指的是 Systems Manager `SecureString` 参数的名称。当您运行 `AWS-ApplyChefRecipes` 文档时，迁移脚本会使用这些参数。
- `--http-username` 参数要求您还为 `--http-password` 参数指定一个值。
- `--http-password` 参数要求您还为 `--http-username` 参数指定一个值。
- 请勿同时为 `--http-password` 和 `--repo-private-key` 设置值。提供 SSH 密钥的 Systems Manager `SecureString` 参数名称 (`--repo-private-key`)，或者存储库用户名 (`--http-username`) 和密码 (`--http-password`)。

步骤 4：运行脚本

运行 `python3 stack_exporter.py` 时，您可以预配置应用程序，也可以通过将 `--provision-application` 参数的值设置为 `FALSE` 来创建初学者模板。

示例 1：预配置 Systems Manager Application Manager 应用程序

以下命令获取有关现有 OpsWorks 层的信息，并使用较新的 OpsWorks 架构配置应用程序，其结果类似于为堆栈配置的 Chef 版本。该脚本使用预置所有必需的资源，例如 Auto Scaling 组 CloudFormation，然后在 Systems Manager 应用程序管理器中注册应用程序。

将*###### ID ########*的值。 OpsWorks

```
python3 stack_exporter.py \
     --layer-id layer-id \
     --region stack-region
```

示例 2：生成模板

以下命令获取有关现有 OpsWorks 图层的信息并生成 CloudFormation 模板。如果预配置了模板，其结果与使用 Chef 14 类似。在此示例中，由于 `--provision-application` 参数设置为 `FALSE`，因此未预配置任何资源。

将*###### ID ########*的值。 OpsWorks

```
python3 stack_exporter.py \
    --layer-id layer-id \
    --region stack-region \
    --provision-application FALSE
```

运行该命令后，您可以在 Systems Manager 的 Application Manager 模板库中查看模板，也可以预配置模板。有关查看模板库的更多信息，请参阅AWS Systems Manager 用户指南中的使用模板库。

步骤 5：配置堆 CloudFormation 栈

> **Note**
>
> 如果将脚本的 `--provision-application` 参数设置为 `FALSE`，则您只需要完成此步骤。

当您指定值为的`--provision-application`参数时`FALSE`，脚本输出将提供 CloudFormation 模板的名称和 URL。此模板代表了现有 OpsWorks 堆栈和层的拟议替代方案。

您可以使用 Application Manager 模板库（推荐）或使用来配置模板 CloudFormation。有关使用模板库的更多信息，请参阅AWS Systems Manager 用户指南中的使用模板库。

步骤 6：审查预配置的资源

您现在已准备就绪，可审查预配置的资源。

1. 使用 AWS CloudFormation 控制台查看已配置堆栈的资源。
   a. 通过 https://console.aws.amazon.com/cloudformation 打开 AWS CloudFormation 主机然后选择 Stacks。
   b. 在堆栈页面，选择堆栈，然后选择资源选项卡。
   c. 在资源选项卡上，审查您堆栈中列出的资源。资源列表包括一个 EC2 Auto Scaling 组，您可以在 Auto Scaling 控制台中查看该组，或者 AWS CLI。
2. 使用 Systems Manager Application Manager 审查应用程序的资源。
   a. 通过 https://console.aws.amazon.com/systems-manager/ 打开 Systems Manager 控制台。
   b. 在导航窗格中，选择 Application Manager。
   c. 在应用程序一节中，选择自定义应用程序。Application Manager 打开概述选项卡。
   d. 选择资源选项卡。资源选项卡显示了为 OpsWorks 堆栈和层迁移的所有资源。应用程序名称包括 OpsWorks 堆栈的名称，其格式为 *app-stack-nam e-##*，其中后#代表堆栈 ID 的前六个字符。有关在 Application Manager 中查看资源的更多信息，请参阅AWS Systems Manager 用户指南中的查看应用程序资源。

步骤 7：启动实例

预配置完实例后，就可以测试该实例了。此时，没有实例在运行。

要使您的实例联机，请将自动扩缩组的 Min、Max 和 Desired capacity 值调整为适合您应用程序的数字。最初，您可能需要将这些值设置为 1，以使单个实例联机，并验证该实例是否执行了所有预期的操作，包括运行您的自定义 Chef 配方。

步骤 8：审查实例

启动实例后，请验证其是否按预期运行。

1. 审查脚本 `--command-logs-bucket` 参数指定的 S3 存储桶中的 Chef `startup` 和 `terminate` 日志。默认情况下，日志存储在名称为 `aws-opsworks-application-manager-logs-`*`account-id`* 的存储桶中。

   a. 登录 AWS Management Console 并打开 Amazon S3 控制台，网址为 https://console.aws.amazon.com/s3/。

   b. 选择包含您日志的存储桶。

   c. 导航到 `ApplyChefRecipes` 前缀以查看您的日志。

2. 检查 Application Load Balancer 的连接和运行状况。

   请执行以下步骤，查看负载均衡器的访问日志。您可以使用脚本的 `--lb-access-logs-path` 参数指定要将负载均衡器访问日志存储到的 S3 存储桶。

   a. 登录 AWS Management Console 并打开 Amazon S3 控制台，网址为 https://console.aws.amazon.com/s3/。

   b. 选择您的 S3 存储桶，然后导航到包含您日志的前缀。

3. 验证实例通过了所有自动扩缩 和 Application Load Balancer 运行状况检查（如果您已配置）。

   您可以在新的实例选项卡上查看有关自动扩缩运行状况的信息。

   a. 通过 https://console.aws.amazon.com/systems-manager/ 打开 Systems Manager 控制台。

   b. 在导航窗格中，选择 Application Manager。

   c. 在应用程序一节中，选择自定义应用程序。

   d. 在列表中选择应用程序。Application Manager 打开概述选项卡。

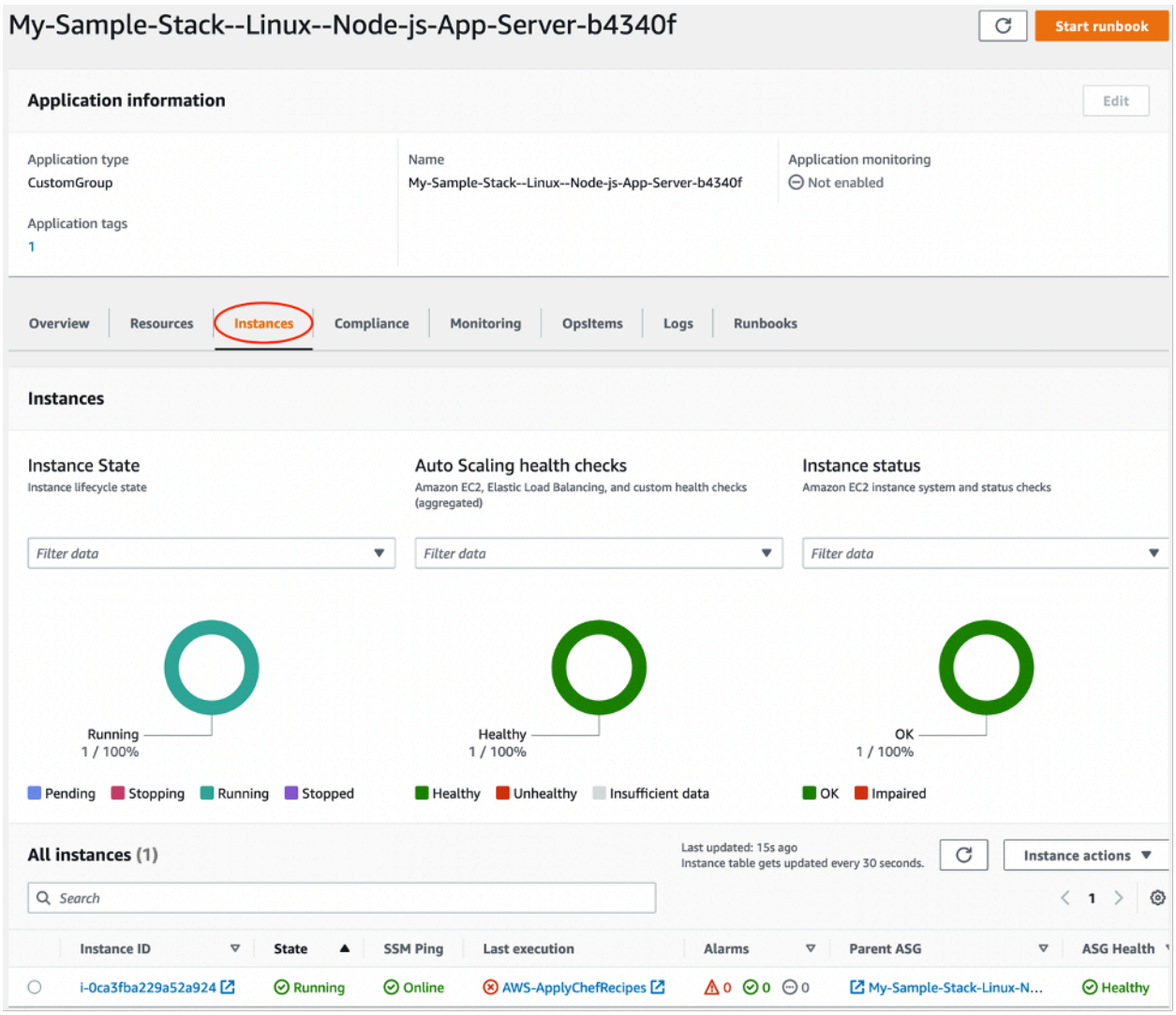
   e. 选择实例选项卡可查看有关自动扩缩运行状况的信息。

在验证 Chef 配方成功运行后，您可以减少自动扩缩组的容量来终止实例。如果您有任何自定义终止配方，请验证配方是否按预期运行。

第 9 步：使用 Systems Manager Application Manager 监控和运行实例上的操作

现在，您可以使用 Application Manager 页面上的新实例选项卡对您的实例进行监控和运行操作。有关使用实例选项卡的更多信息，请参阅AWS Systems Manager 用户指南中的使用应用程序实例。

您可以使用实例选项卡在一个位置查看多个 AWS 实例。使用此选项卡，您可以查看有关实例运行状况的信息并对问题进行故障排除。



按照以下步骤查看实例选项卡。

1. 通过 https://console.aws.amazon.com/systems-manager/ 打开 Systems Manager 控制台。
2. 在导航窗格中，选择 Application Manager。

3. 在应用程序一节中，选择自定义应用程序。
4. 在列表中选择应用程序。Application Manager 打开概述选项卡。
5. 选择实例选项卡，查看有关您的实例状态和 EC2 运行状况的信息。

常见问题解答

以下常见问题解答提供了一些常见问题的答案。

主题

我可以迁移哪些 AWS OpsWorks Stacks 版本？

您只能迁移 Chef 11.10 和 Chef 12, Amazon Linux, Amazon Linux 2, Ubuntu, 和 Red Hat Enterprise Linux 7 堆栈。

我的迁移实例可以使用哪些 Chef 版本？

迁移的实例可以使用 Chef 11 到 14。

> **Note**
>
> 不支持 Windows 堆栈迁移。

我可以迁移哪些存储库类型？

您可以迁移 S3、Git 和 HTTP 存储库类型。

我可以继续使用私有 Git 存储库吗？

是的，您可以继续使用私有 Git 存储库。

如果您使用私有 GitHub 存储库，则必须为 SSH 创建新的Ed25519主机密钥。这是因为 GitHub 更改了 SSH 中支持的密钥并删除了未加密的 Git 协议。有关Ed25519主机密钥的更多信息，请参阅 GitHub 博客文章 “改进 Git 协议安全” GitHub。生成新的 Ed25519 主机密钥后，为此 SSH 密钥创建一个 Systems Manager SecureString 参数，并使用参数名称作为 --repo-private-key 参数的值。有关如何创建 System SecureString s Manager 参数的更多信息，请参阅《AWS Systems Manager 用户指南》中的创建 SecureString 参数 (AWS CLI)。

对于任何其他 Git 存储库类型，请为此 SSH 密钥创建一个 Systems Manager SecureString 参数，并将该参数名称用作脚本 --repo-private-key 参数的值。

我可以使用哪些 SSH 密钥来访问我的实例？

运行脚本时，脚本会迁移堆栈中配置的 SSH 密钥和实例。您可以使用 SSH 密钥访问您的实例。如果为堆栈和实例提供了 SSH 密钥，则脚本将使用堆栈中的密钥。如果您不确定要使用哪些 SSH 密钥，请在 EC2 控制台中查看实例 (https://console.aws.amazon.com/ec2/)。EC2 控制台中的详细信息页面显示您实例的 SSH 密钥。

为什么我的实例会自动扩展和缩减？

自动扩缩根据自动扩缩组的伸缩规则来扩缩实例。您可以为组设置最小、最大和所需容量值。当您更新这些值时，自动扩缩组会自动相应地扩缩您的容量。

我可以关闭自动扩缩吗？

您可以通过将自动扩缩组的最小、最大和所需容量值设置为相同的数字来关闭自动扩缩。例如，如果您希望始终有十个实例，请将最小、最大和所需容量值设置为十。

我能否对启动的 EC2 实例执行内核和软件包更新？

默认情况下，内核和软件包更新在 EC2 实例启动时发生。使用以下步骤对已启动的 EC2 实例执行内核或软件包更新。例如，您可能希望在运行部署或配置配方后应用更新。

1. 连接到您的 EC2 实例。
2. 创建以下 `perform_upgrade` 函数并在您的实例上运行它。

```
perform_upgrade() {
    #!/bin/bash
    if [ -e '/etc/system-release' ] || [ -e '/etc/redhat-release' ]; then
     sudo yum -y update
    elif [ -e '/etc/debian_version' ]; then
     sudo apt-get update
     sudo apt-get dist-upgrade -y
    fi
}
perform_upgrade
```

3. 内核和软件包更新后，您可能需要重启您的 EC2 实例。要检查是否需要重启，请创建以下 `reboot_if_required` 函数并在您的 EC2 实例上运行该函数。

```
reboot_if_required () {
 #!/bin/bash
 if [ -e '/etc/debian_version' ]; then
   if [ -f /var/run/reboot-required ]; then
     echo "reboot is required"
   else
     echo "reboot is not required"
   fi
 elif [ -e '/etc/system-release' ] || [ -e '/etc/redhat-release' ]; then
  export LC_CTYPE=en_US.UTF-8
```

```
  export LC_ALL=en_US.UTF-8
  LATEST_INSTALLED_KERNEL=`rpm -q --last kernel | perl -X -pe 's/^kernel-(\S+).*/
$1/' | head -1`
  CURRENTLY_USED_KERNEL=`uname -r`
  if [ "${LATEST_INSTALLED_KERNEL}" != "${CURRENTLY_USED_KERNEL}" ];then
     echo "reboot is required"
  else
     echo "reboot is not required"
  fi
 fi
}
reboot_if_required
```

4. 如果在 `reboot is required` 消息中运行 `reboot_if_required` 结果，请重启 EC2 实例。如果您收到 `reboot is not required` 消息，则无需重启 EC2 实例。

为什么我的实例中的 EBS 卷不包含任何数据？

运行该脚本时，该脚本会迁移 EBS 卷的配置，从而为您的 OpsWorks 堆栈和层创建替代架构。该脚本不会迁移实际实例或实例中包含的数据。该脚本仅在层级迁移 EBS 卷的配置，并将空的 EBS 卷附加到已启动的 EC2 实例。

按照以下步骤从先前实例的 EBS 卷中提取数据。

1. 为您的之前的实例 EBS 卷拍摄快照。有关创建快照的更多信息，请参阅《Amazon EC2 用户指南》中的创建 Amazon EBS 快照。
2. 从快照创建卷。有关通过快照创建卷的更多信息，请参阅 Amazon EC2 用户指南中的从快照创建卷。
3. 将您创建的卷附加到实例。有关附加卷的更多信息，请参阅《Amazon EC2 用户指南》中的将 Amazon EBS 卷附加到实例。

为什么我的启动模板中描述的 EBS 卷没有挂载？

如果您为带有 EBS 卷的 `--launch-template` 参数提供启动模板 ID，则该脚本会附加 EBS 卷，但不会挂载卷。您可以通过运行脚本为已启动的 EC2 实例创建的`MountEBSVolumes` RunCommand 文档来挂载附加的 EBS 卷。

如果您未设置 `--launch-template` 参数，则脚本会创建一个模板，当自动扩缩组启动新的 EC2 实例时，自动扩缩组会自动附加 EBS 卷，然后运行 `SetupAutomation` 命令将附加的卷挂载到层设置中配置的挂载点。

我在哪里可以找到 Chef 配方和 Mount EBS 卷日志？

OpsWorks 将日志传送到 S3 存储桶，您可以通过为`--command-logs-bucket`参数提供值来指定该存储桶。默认S3 存储桶名称的格式为 `aws-opsworks-stacks-application-manager-logs-`*`account-id`*。Chef 配方日志存储在 `ApplyChefRecipes` 前缀中。挂载 EBS 卷日志存储在 `MountEBSVolumes` 前缀中。从堆栈迁移的所有层都会将日志传送到同一 S3 存储桶。

> **Note**
>
> - S3 存储桶的生命周期配置包括一条在 30 天后删除日志的规则。如果您想保存日志超过 30 天，则必须更新 S3 存储桶生命周期配置中的规则。
> - 目前， OpsWorks 仅记录厨师`setup`和`terminate`食谱。

在哪里可以找到迁移脚本的调试日志？

该脚本将调试日志放在名为 `aws-opsworks-stacks-transition-logs-`*`account-id`* 的存储桶中。您可以在 S3 存储桶的 `migration_script` 文件夹中找到与您迁移的层名称相匹配的文件夹下的调试日志。

迁移脚本是否支持 CloudFormation 模板版本控制？

该脚本生成类型的 Systems Manager 文档 CloudFormation ，用于替换要迁移的图层或堆栈。即使使用相同的参数，再次运行脚本也会导出先前导出的层模板的新版本。模板版本与脚本日志存储在同一 S3 存储桶中。

我可以迁移多个图层吗？

脚本的 `--layer-id` 参数在单个层中传递。要迁移多个图层，请重新运行脚本并在不同的 `--layer-id` 传递。

在应用程序管理器中，属于同一 OpsWorks 堆栈的图层列在同一个应用程序下。

1. 通过 https://console.aws.amazon.com/systems-manager/ 打开 Systems Manager 控制台。
2. 在导航窗格中，选择 Application Manager。
3. 在应用程序一节中，选择自定义应用程序。
4. 选择您的应用程序。应用程序名称以 app-*`stack-name`*-*`first-six-characters-stack-id`* 开头。

5. 以 app 开头的顶级元素显示了与您的 OpsWorks 堆栈对应的所有组件。这包括与您的 OpsWorks 图层对应的组件。
6. 选择与该层对应的组件以查看该层的资源。在“自定义应用程序”部分中，代表 OpsWorks 图层的组件也可作为单个应用程序查看。

我该如何创建 SecureString 参数？

您可以使用 Systems Manager 来创建 SecureString 参数。有关如何创建 System SecureString s Manager 参数的更多信息，请参阅《AWS Systems Manager 用户指南》中的创建 SecureString 参数 (AWS CLI) 或创建 Systems Manager 参数（控制台）。

必须提供一个 SecureString 参数作为 --http-username、--http-password、或 --repo-private-key 参数的值。

如何保护新自动扩缩组中的实例免受终止事件的影响？

您可以通过将 --enable-instance-protection 参数设置为 TRUE 并向要保护的每个 EC2 实例添加 protected_instance 标签密钥来保护实例，使其免受终止事件的影响。当您将 --enable-instance-protection 参数设置为 TRUE 并添加 protected_instance 标签密钥时，脚本会将自定义终止策略添加到您的新自动扩缩组并暂停 ReplaceUnhealthy 进程。使用 protected_instance 标签密钥的实例受到保护，免受以下终止事件的影响：

- 横向缩减事件
- 实例刷新
- 再平衡
- 实例最大生命周期
- 允许列出实例终止
- 终止和替换运行状况不佳的实例

> **Note**
>
> 您必须在要保护的实例上设置 protected_instance 标签密钥。此标签密钥区分大小写。无论标签值如何，任何具有该标签密钥的实例都将受到保护。
> 要缩短自定义终止策略的运行时间，您可以通过更新 default_sample_size 函数代码变量的值来增加 Lambda 函数用于筛选受保护实例的默认实例数。默认值为 15。如果您增加

`default_sample_size`，则可能需要增加分配给 Lambda 函数的内存，这将增加 Lambda 函数的成本。有关 AWS Lambda 定价的信息，请参阅 AWS Lambda 定价。

迁移脚本有哪些负载均衡器可用？

该脚本提供了三个负载均衡器选项。

- （推荐）创建新的 Application Load Balancer。默认情况下，该脚本会创建一个新的 Application Load Balancer。您也可以将 `--lb-type` 参数设置为 `ALB`。有关应用程序负载均衡器的更多信息，请参阅《Elastic Load Balancing 用户指南》中的什么是应用程序负载均衡器？。
- 如果无法选择 Application Load Balancer，请通过将 `--lb-type` 参数设置为来创建 Classic Load Balancer`Classic`。如果您选择此选项，则连接到您的 OpsWorks层的现有 Classic Load Balancer 将与您的应用程序分开。有关应用程序负载均衡器的更多信息，请参阅《Elastic Load Balancing：经典负载均衡器用户指南》中的什么是经典负载均衡器？。
- 您可以通过将 `--lb-type` 参数设置为 `None` 来连接现有的负载均衡器。

> **Important**
>
> 我们建议为您的 AWS OpsWorks Stacks 层创建新的 Elastic Load Balancing 负载均衡器。如果您选择使用现有的 Elastic Load Balancing 负载均衡器，应先确认它当前未用于其他用途并且没有挂载实例。将负载均衡器连接到层后， OpsWorks 移除所有现有实例，并将负载均衡器配置为仅处理该层的实例。从技术上来说，在将某个负载均衡器挂载到层之后使用 Elastic Load Balancing 控制台或 API 来修改它的配置尽管是可行的，但您不应如此操作；更改将不会是永久的。

将现有 OpsWorks 层负载均衡器连接到 Auto Scaling 群组

1. 将 `--lb-type` 参数设置为 `None`，运行迁移脚本。当该值设置为 `None` 时，脚本不会克隆或创建负载均衡器。
2. 脚本部署 CloudFormation 堆栈后，更新 Auto Scaling 组`Min``Max`和`Desired capacity`值，然后测试您的应用程序。
3. 选择 `Link to the template` 显示在脚本输出中。如果您关闭了终端，请按照以下步骤访问模板。

   a. 通过 https://console.aws.amazon.com/systems-manager/ 打开 Systems Manager 控制台。

b. 在导航窗格中，选择 Application Manager。

c. 选择CloudFormation 堆栈，然后选择模板库。

d. 选择我拥有并找到您的模板。

4. 在 CloudFormation 模板中，从 “操作” 菜单中选择 “编辑”。

5. 在 CloudFormation模板的`ApplicationAsg`资源部分中更新该`LabelBalancerNames`属性。

```
ApplicationAsg:
   DependsOn: CustomTerminationLambdaPermission
   Properties:
   #(other properties in ApplicationAsg to remain unchanged)
      LoadBalancerNames:
        - load-balancer-name
      HealthCheckType: ELB
```

6. 如果您希望自动扩缩组实例的运行状况检查也使用负载均衡器的运行状况检查，请删除 `HealthCheckType` 下的一节并输入 `ELB`。如果您只需要 EC2 运行状况检查，则无需更改模板。

7. 保存您的更改。保存会创建模板的新默认版本。如果这是您第一次运行层的脚本，也是您第一次在控制台中保存更改，则较新的版本是 2。

8. 从操作中，选择预配置堆栈。

9. 确认您要使用模板的默认版本。确保选中 “选择现有堆栈”，然后选择要更新的 CloudFormation 堆栈。

10. 在接下来的每个页面中选择下一步，直到看到审查和预配置页面。在 “查看和配置” 页面上，选择我确认 AWS CloudFormation 可能会使用自定义名称创建 IAM 资源，并且我知道所选模板中的更改可能会 AWS CloudFormation 导致更新或删除现有 AWS 资源。

11. 选择配置堆栈。

如果您需要回滚更新，请执行以下步骤。

1. 选择操作，然后选择预置堆栈。

2. 选择选择现有版本之一，然后选择以前的模板版本。

3. 选择 “选择现有堆栈”，然后选择要更新的 CloudFormation 堆栈。

自定义说明书配置配方是否已迁移？

不支持在设置事件期间运行配置自定义配方。该脚本迁移自定义说明书配置配方，并为您创建 Systems Manager Automation 运行手册。但是，您必须手动运行配方。

请执行以下步骤来运行配置配方。

1. 通过 https://console.aws.amazon.com/systems-manager/ 打开 Systems Manager 控制台。
2. 在导航窗格中，选择 Application Manager。
3. 在应用程序一节中，选择自定义应用程序。
4. 选择您的应用程序。应用程序名称以 `app-stack-name` 开头。
5. 选择资源，然后选择配置运行手册。
6. 选择执行自动化。
7. 选择要为其运行配置配方的实例 ID，然后选择执行。

我能否在新创建的实例上运行部署和取消部署配方？

根据层的配置，该脚本可以创建三个可能的自动化运行手册。

- 设置
- 配置
- 终止

该脚本还可以创建以下 Systems Manager 参数，这些参数包含 `AWS-ApplyChefRecipes Run Command` 文档的输入值。

- 设置
- 部署
- 配置
- Undeploy
- 终止

当发生横向扩展事件时，安装程序自动化运行手册会自动运行。这包括设置和部署原始 OpsWorks 图层中的自定义食谱食谱。当发生横向缩减事件时，安装程序自动化运行手册会自动运行。终止自动化运行手册包含原始 OpsWorks 图层中的关闭方法。

如果要手动运行取消部署或配置配方，请执行以下步骤。

1. 通过 https://console.aws.amazon.com/systems-manager/ 打开 Systems Manager 控制台。
2. 在导航窗格中，选择 Application Manager。
3. 在应用程序一节中，选择自定义应用程序。
4. 选择您的应用程序。应用程序名称以 app-*stack-name*-*first-six-characters-stack-id* 开头。Application Manager 打开概述选项卡。
5. 选择资源，然后选择配置自动化运行手册。
6. 选择执行自动化。
7. 有关 applyChefRecipesPropertiesParameter 自动运行手册输入参数，请引用正确的 Systems Manager 参数。Systems Manager 参数名称遵循格式 /ApplyChefRecipes-Preset/*OpsWorks-stack-name*-*OpsWorks-layer-name*-*first-six-characters-stack-id*/*event*，其中 *event* 的值为Configure、Deploy、或Undeploy，具体取决于您要运行的配方。
8. 选择要为其运行配方的实例 ID，然后选择执行。

我能否更改自动扩缩组跨越的子网？

默认情况下，Auto Scaling 组跨越您的堆栈 OpsWorks VPC 中的所有子网。要更新要跨越的子网，请执行以下步骤。

1. 选择 Link to the template 显示在脚本输出中。如果您关闭了终端，请按照以下步骤访问模板。
   a. 通过 https://console.aws.amazon.com/systems-manager/ 打开 Systems Manager 控制台。
   b. 在导航窗格中，选择 Application Manager。
   c. 选择CloudFormation 堆栈，然后选择模板库。
   d. 选择我拥有并找到您的模板。
2. 从操作中，选择预配置堆栈。
3. 确认您要使用默认模板。选择 “选择现有堆栈”，然后选择要更新的 CloudFormation 堆栈。

> **Note**
>
> 如果在--provision-application参数设置为的情况下运行脚本FALSE，则必须创建一个新 CloudFormation 堆栈。

4. 对于 SubnetIDs 参数，请提供您希望自动扩缩组跨越的子网 ID 列表，以逗号分隔。
5. 选择下一步，直到看到审查和配置页面。
6. 在“查看和配置”页面上，选择“我确认 AWS CloudFormation 可能会使用自定义名称创建 IAM 资源”，并且我知道所选模板中的更改可能会 AWS CloudFormation 导致更新或删除现有 AWS 资源。
7. 选择配置堆栈。

故障排除

本主题包含一些常见的问题以及这些问题的建议的解决方案。

主题

提供的委托人无效

问题：您收到一条错误消息，指出您提供的委托人无效。

原因：出现这种情况是因为自动扩缩组没有服务角色。

解决方案：在发生错误的区域创建一个自动扩缩组。创建自动扩缩组可为您的自定义终止策略创建必要的服务相关角色。

启用 Auto Scaling 组保护实例后无法删除 CloudFormation 堆栈

问题： `--enable-instance-protection` 参数设置为 TRUE，并且您的自动扩缩组的某些 EC2 实例受到 `protected_instance` 标签密钥的保护，这样可以防止您的 AWS CloudFormation 堆栈被完全删除。

原因：EC2 实例有一个 `protected_instance` 标签密钥，可以保护它们免受终止事件的影响。

解决方案：从 EC2 实例中移除 `protected_instance` 标签密钥。这允许自动扩缩组缩减。在 Auto Scaling 组缩小规模后，您可以删除该 AWS CloudFormation 堆栈。

提供现有 S3 存储桶和前缀时出现访问被拒绝错误

问题：当您提供现有 S3 存储桶和前缀时，您会收到一条 `AccessDenied` 错误。

原因：S3 存储桶策略未提供将负载均衡器日志传送到存储桶所需的权限。

解决方案：更新 S3 存储桶策略以允许脚本将负载均衡器访问日志传送到存储桶。有关如何更新存储桶策略的更多信息，请参阅Elastic Load Balancing：应用程序负载均衡器用户指南中的为应用程序负载均衡器启用访问日志。

使用 “就地 AWS OpsWorks Stacks 分离” 工具

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。

本节介绍如何使用原地 AWS OpsWorks Stacks 分离工具将您的 OpsWorks 实例与 OpsWorks Stacks 服务分离。

您分离的实例将保留在您的中 AWS 账户，但您将无法再使用 OpsWorks来管理它们。相反，您将使用 Amazon EC2 或任何与 EC2 兼容的方法来配置和管理实例。 AWS Systems Manager

从总体上讲，分离过程包括以下步骤：

1. 该工具会执行验证检查，以确保资源已准备就绪，可以分离。
2. 该工具从您的 OpsWorks 堆栈中导出自定义 JSON，并将其作为对象存储在 Amazon S3 中。
3. 该工具可创建代表每个 OpsWorks Stacks 生命周期事件的 Systems Manager 自动化文档。
4. 该工具为所有正在分离的实例创建 AWS Service Catalog AppRegistry 目录，并将所有 Elastic Load Balancing (ELB) 负载均衡器与层分离。 OpsWorks
5. 最后，该工具会分离和注销其他资源，包括亚马逊关系数据库服务 (Amazon RDS) 实例。

流程是如何运作的

Detach In Place 工具提供以下 3 个命令和类似向导的体验，可指导您完成一系列步骤，在继续分离图层之前检查和配置实例。

命令	描述
`handle-prerequisites`	此命令分析图层中的所有实例是否符合分离条件并解析先决条件。这些实例必须处于健康状

命令	描述
	态 OpsWorks，不能有基于时间或负载的自动缩放器，并且必须安装最新的 OpsWorks 代理版本。 此外，该命令还会检查是否所有实例都具有支持 SSM 代理所需的权限，以及是否安装了最新的 SSM 代理版本。如果 SSM 代理不存在，则该命令将安装该代理；如果 SSM 代理未使用最新版本，则会更新 SSM 代理。该命令还将添加所有必要的权限。
`detach`	此命令分离指定层的所有 OpsWorks 实例。 首先，该命令将运行先决条件检查，以确保该层符合分离条件。如果您不想解决先决条件，则可以选择强制分离。 接下来，该命令将指示通过标 OpsWorks 记 API 或通过从图层和堆栈传播标签添加到您的实例的所有标签都将被保留。分离完成后，您可以使用相关的 EC2 API 删除这些标签中的任何一个。 然后，该命令将检查您是否要将与 Chef 相关的配置导出到 SSM 参数。 如果您在层上连接了 Classic Load Balancer，则该命令将询问它是否可以分离负载均衡器以防止停机。

命令	描述
`cleanup`	此命令会 OpsWorks 从您的账户中删除所有实体。它将终止实例并删除所有堆栈。这应该用于清理账户的最后一步不再需要的资源。 **Note** 我们建议您在运行该`cleanup`命令之前运行新安装程序几天。这样可以确保堆栈中的任何必要配置在需要时随时可用。

限制

就地分离工具的主要目的是安全地分离 OpsWorks 堆栈实例。本节总结了该工具的局限性。

- Windows SSM 代理 — 如果实例上未安装 SSM 代理，则需要手动安装它。如果代理未更新到最新版本，则同样适用。
- 时间/加载 Auto Scaling 实例 — 分离工具不支持启用了 Auto Scaling 的实例。您必须在要分离的实例上禁用 Auto Scaling。
- 权限 — 分离工具不会创建或生成 OpsWorks 控制台权限页面上指定的 IAM 实体。
- 应用程序-分离工具不会在外部创建或生成应用程序。 OpsWorks

开始使用

步骤 1：验证是否满足先决条件

就地分离工具的所有 3 个命令都是 Python 脚本，您可以在本地、在 EC2 实例上运行或通过使用AWS CloudShell来运行这些脚本。

AWS CloudShell 是一个基于浏览器的外壳，可让您通过命令行访问选定 AWS 资源中的资源。 AWS 区域 AWS CloudShell 预装了常用工具（例如 AWS CLI 和 Python）。使用时 AWS CloudShell，您使用的凭证与登录控制台时使用的凭证相同。

本演练假设您正在使用 AWS CloudShell.

第 2 步：下载脚本

1. 通过运行以下命令下载包含迁移脚本和所有相关文件的 zip 文件：

```
aws s3api get-object \
--bucket detach-in-place-bucket-prod-us-east-1 \
--key detach_in_place_script.zip detach_in_place_script.zip
```

2. 运行以下命令解压缩文件。

```
unzip detach_in_place_script.zip
```

文件解压缩后，以下文件可用：

- README.md
- LICENSE
- NOTICE
- requirements.txt
- TODO.py

3. 如有必要，请运行以下命令进行安装pipenv。

```
pip install pipenv
```

步骤 3：运行脚本

首先，设置您的环境，以便您可以通过运行以下命令来运行脚本。

```
pipenv install -r requirements.txt
pipenv shell
```

然后，查看脚本参数。

命令	参数	描述	类型	必需	默认
handle-prerequisites	--layer-id	要分离的图层的 ID。	String	是	-

命令	参数	描述	类型	必需	默认
	--region	OpsWorks 堆栈的区域。如果您的 OpsWorks 堆栈区域和 API 终端节点区域不同，请使用堆栈区域。此区域与 OpsWorks 堆栈中的其他资源部分（例如，EC2 实例和子网）属于同一区域。	String	否	us-east-1
detach	--layer-id	要分离的图层的 ID。	String	是	-
	--batch-size	要从图层分离的实例数（例如，5）。	String	否	-
	--region	OpsWorks 堆栈的区域。如果您的 OpsWorks 堆栈区域和 API 终端节点区域不同，请使用堆栈区域。此区域与 OpsWorks 堆栈中的其他资源部分（例如，EC2 实例和子网）属于同一区域。	String	否	us-east-1
cleanup	--stack-id	要删除的堆栈的 ID。	String	否	互不相容，必须指定层 ID 或堆栈 ID
	--layer-id	要删除的图层的 ID	String	否	

命令	参数	描述	类型	必需	默认
	`--region`	OpsWorks 堆栈的区域。如果您的 OpsWorks 堆栈区域和 API 终端节点区域不同，请使用堆栈区域。此区域与 OpsWorks 堆栈中的其他资源部分（例如，EC2 实例和子网）属于同一区域。	String	否	us-east-1

您可以通过运行带有选项的`detachcleanup`命令来查看`handle-prerequisites`和命令的可用`--help`选项，如下所示：

```
python3 layer_detacher.py detach --help
python3 layer_detacher.py handle-prerequisites --help
python3 layer_detacher.py cleanup --help
```

现在，您已准备好开始了。以下示例显示了如何针对不同的用例运行命令。

示例：

- 示例 1：检查图层是否满足所有先决条件并且符合分离条件
- 示例 2：分离图层的所有实例
- 示例 3：批量分离图层的所有实例
- 示例 4：清理图层的所有资源并删除该图层
- 示例 5：清理堆栈的所有资源并删除堆栈

示例 1：检查图层是否满足所有先决条件并且符合分离条件

以下命令读取有关 OpsWorks 层（及其包含的实例）的信息，并检查是否满足以下先决条件：

- 所有实例都处于联机状态。
- 没有 Load/Time Auto Scaling 实例。
- 所有实例都有最新的 OpsWorks 代理。
- 所有实例都安装并配置了最新的 SSM 代理。

- 所有实例都有 SSH 密钥对。
- 每个实例恰好属于一个图层。

```
python3 layer_detacher.py handle-prerequisites \
--layer-id opsworks-layer-id \
--region opsworks-stack-region
```

示例 2：分离图层的所有实例

以下命令将遍历该层的所有实例，检查这些实例是否满足先决条件，并尝试并行分离所有满足先决条件的实例。如果不满足一个或多个先决条件，则该命令将为其余的不合规实例提供强制分离选项。

在分离任何实例之前，该命令将：

1. 保存自定义 JSON 并将其上传到 S3。
2. 为层的每个 OpsWorks 生命周期事件创建 SSM 自动化文档，并将自动化文档的执行日志上传到 S3。
3. 为所有要分离的实例创建 AppRegistry 应用程序。该应用程序有一个与之关联的资源组，其中包含所有分离的实例和资源。这些资源包括 SSM 自动化文档和 SSM 参数，其中包含有关生命周期事件和自定义 Chef 食谱的信息。
4. 将 Classic Load Balancer 与该层（如果存在）分离。

此命令将仅修改 OpsWorks 资源。EC2 实例的状态将保持不变。

```
python3 layer_detacher.py detach \
--layer-id opsworks-layer-id \
--region opsworks-stack-region
```

示例 3：批量分离图层的所有实例

以下命令的作用与前面的示例相同。唯一的区别是它会批量分离实例。

此命令将仅修改 OpsWorks 资源。EC2 实例的状态将保持不变。

```
python3 layer_detacher.py detach \
--layer-id opsworks-layer-id \
--region opsworks-stack-region \
--batch-size 5
```

示例 4：清理图层的所有资源并删除该图层

以下命令将遍历图层的所有资源并将其删除。更详细地说，它将停止并删除和 EC2 中的 OpsWorks 所有实例，分离负载均衡器并注销 Amazon RDS 实例、弹性 IP 和卷。清理资源后，它将删除该图层。

此命令将删除 OpsWorks 资源和 EC2 实例。如果您想让 EC2 实例保持不变，请在使用`detach`命令之前使用该`cleanup`命令。这样，该`cleanup`命令将删除所有剩余的资源。

```
python3 layer_detacher.py cleanup \
--layer-id opsworks-layer-id \
--region opsworks-stack-region
```

示例 5：清理堆栈的所有资源并删除堆栈

以下命令将遍历所有层，然后迭代每个层的资源。对于每一层，该命令将停止并删除和 EC2 中的所有实例，分离负载均衡器， OpsWorks 并注销 Amazon RDS 实例、弹性 IP 和卷。然后，该命令将删除该图层。将在属于该堆栈的每一层中执行相同的过程。最后，删除所有图层后，堆栈将被移除。

此命令将删除 OpsWorks 资源和 EC2 实例。如果您想让 EC2 实例保持不变，请在使用`detach`命令之前使用该`cleanup`命令。这样，该`cleanup`命令将删除所有剩余的资源。

```
python3 layer_detacher.py cleanup \
--stack-id opsworks-stack-id \
--region opsworks-stack-region
```

第 4 步：从资源分离后继续操作您的资源 OpsWorks

运行该`detach`命令后，该工具将创建一个与分离的图层对应的新 AWS Service Catalog AppRegistry 应用程序。应用程序名称遵循以下格式*`layer-name`*`---`*`layer-id`*。它还添加了`OpsWorksLayerId`标记，以唯一标识与分离层匹配的应用程序。

要向该应用程序添加新 AWS 资源（例如，新的 EC2 实例），您可以执行以下操作之一：

1. 使用应用程序的唯一应用程序标签来 AppRegistry标记资源：

   标签密钥：`awsApplication`

   值：`arn:aws:resource-groups:`*`region`*`:`*`account-id`*`:group/`*`application-name`*`/`*`application-id`*`>`
2. 运行 associate-resource 命令。

此外，还会为每个 AppRegistry 应用程序创建一个资源组。资源组包含以下标签。

标签密钥	值
EnableAWSServiceCatalogAppRegistry	TRUE
aws:servicecatalog:applicationName	*application-name*
aws:servicecatalog:applicationId	*application-id*
aws:servicecatalog:applicationArn	arn:aws:servicecatalog: *region*:*account-id* :/applications/ *application-id*

分队后执行任务

下表提供了有关在分离后如何执行任务的信息：

任务	描述
执行生命周期事件	运行detach命令后，如果您选择了该选项，则脚本将创建 5 个与 5 个 OpsWorks 生命周期事件匹配的 Automation 文档。 每个自动化文档的名称都遵循以下格式: *layer-id*_*lifecycle-event* _automation_document . 要模拟 Systems Manager 中的 OpsWorks 行为，您需要在配置、终止 EC2 实例或部署/删除配方时手动触发自动化执行。
更新自定义 JSON	堆栈和层的自定义 JSON 存储在分离期间指定的 S3 存储桶中，或者存储在创建的新 S3 存储桶中。 为 JSON 文件存储的文件名如下：

任务	描述
	• `layercustomjson.json` • `stackcustomjson.json`
更改生命周期事件的运行列表	每个生命周期事件的运行列表在相应的 Automation 文档中定义。要更改运行列表，请在 AppRegistry 应用程序中找到 Automation 文档并修改`RunList`参数。 更新食谱和食谱的过程没有变化`AWS-ApplyChefRecipes`，因为自动化文档触发的支持与之相同的 OpsWorks来源。
管理自动修复/自动缩放	当您分离实例时， OpsWorks 代理会卸载。如果没有代理， OpsWorks 就无法自动修复或替换运行状况不佳的实例，也无法自动扩展您的队列。要继续自动扩展和替换失败的实例，请创建一个 Amazon EC2 Auto Scaling 组。当 Amazon EC2 检测到需要更换的不健康实例时，该组将启动新实例以保持其所需容量。
管理 Load Balancer	如果您的层使用 Classic Load Balancer，则该`detach`命令将在注销实例之前将其分离。这样做是为了确保在整个分离过程中所有 ELB 实例关联都保留在 Amazon EC2 上，从而实现零停机时间。该过程完成后，您将能够在 EC2 上管理您的 ELB。

任务	描述
连接到实例	运行`handle-prerequisites` 或`detach`命令时，会进行两项检查： • SSM 代理的版本和权限 • SSH 密钥 这些命令还为您提供更新 SSM 代理和添加所需权限的选项，以便您可以使用会话管理器连接到实例。如果存在 SSH 密钥，您还可以选择 SSH 进入实例。

使用 Systems Manager 应用程序管理器实例选项卡

分离后，您将能够在 Application Manager 实例选项卡上查看和管理您的实例。

实例选项卡提供有关应用程序的 EC2 实例的汇总信息，例如其状态、运行状况和上次命令状态。使用此选项卡，您可以查看有关单个实例的详细信息，例如命令历史记录、警报状态、Systems Manager 代理运行状况等。“实例” 选项卡还提供各种操作，例如能够应用 Chef 配方、启动或停止实例，或者在 Auto Scaling 组中添加或删除实例。

AWS OpsWorks 堆栈入门

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

AWS OpsWorks Stacks 提供了一组丰富的可自定义组件，您可以混合搭配这些组件，以创建满足您特定目的的堆栈。对于新用户来说，难点在于了解如何将这些组件整合到一个工作堆栈中并有效地管理它。下面介绍了如何入手。

如果要...	完成本演练：
尽快创建示例堆栈	入门：示例
试验基于 Linux 的堆栈	入门：Linux
试验基于 Windows 的堆栈	入门：Windows
了解如何创建您自己的 Chef 说明书	入门：说明书

如果您有现有的计算资源（Amazon EC2 实例，甚至是在您自己的硬件上运行的本地实例），则可以将它们与使用堆栈创建的实例一起合并到堆栈中。 AWS OpsWorks 然后，您可以使用 AWS OpsWorks Stacks 将所有相关实例作为一个组进行管理，无论它们是如何创建的。

区域支持

您可以全局访问 AWS OpsWorks 堆栈；也可以在全球范围内创建和管理实例。用户可以将 AWS OpsWorks Stacks 实例配置为在除 AWS GovCloud （美国西部）和中国（北京） AWS 地区以外的任何地区启动。要使用 AWS OpsWorks 堆栈，实例必须能够连接到以下 AWS OpsWorks Stacks 实例服务 API 终端节点之一。

只能在创建资源的区域中对资源进行管理。在一个区域终端节点中创建的资源对其他区域终端节点不可用，也不能克隆到其他区域终端节点中。您可以在以下任一区域中启动实例。

- 美国东部（俄亥俄州）区域
- 美国东部（弗吉尼亚州北部）区域
- 美国西部（俄勒冈州）区域
- 美国西部（北加利福尼亚）区域
- 加拿大（中部）区域（仅限 API，不适用于在 AWS Management Console中创建的堆栈。）
- 亚太地区（孟买）区域
- 亚太地区（新加坡）区域
- 亚太地区（悉尼）区域
- Asia Pacific（Tokyo）Region
- 亚太地区（首尔）区域
- 欧洲地区（法兰克福）区域

- 欧洲地区（爱尔兰）区域
- 欧洲地区（伦敦）区域
- 欧洲地区（巴黎）区域
- 南美洲（圣保罗）区域

示例堆栈入门

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

本演练展示了如何使用 AWS OpsWorks Stacks 快速创建示例 Node.js 应用程序环境，只需点击几下鼠标，无需编写代码。完成后，您将有一个运行 Chef 12 的 Amazon Elastic Compute Cloud (Amazon EC2) 实例、一台 Node.js HTTP 服务器和一个可用来与 Twitter 交互并在网页上留下评论的 Web 应用程序。

> **Note**
>
> 由于完成本演练将自动创建一个类型为 c3.large 的实例，您无法在免费套餐中使用本演练或使用 AWS OpsWorks Stacks 中的 AWS Sample Stack 创建工具。尽管在 VPC 中使用 Sample Stack 创建工具可以创建一个 t2.medium 实例，但目前 VPC 在AWS 免费套餐中并不可用。

步骤 1：完成前提条件

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

您必须完成以下设置步骤，然后才能开始演练。这些设置步骤包括注册 AWS 帐户、创建管理用户以及为 AWS OpsWorks Stacks 分配访问权限。

主题

注册获取 AWS 账户

如果您没有 AWS 账户，请完成以下步骤来创建一个。

要注册 AWS 账户

1. 打开 https://portal.aws.amazon.com/billing/signup。
2. 按照屏幕上的说明进行操作。

   在注册时，将接到一通电话，要求使用电话键盘输入一个验证码。

   当您注册时 AWS 账户，就会创建AWS 账户根用户一个。根用户有权访问该账户中的所有 AWS 服务 和资源。作为安全最佳实践，请为用户分配管理访问权限，并且只使用根用户来执行需要根用户访问权限的任务。

AWS 注册过程完成后会向您发送一封确认电子邮件。在任何时候，您都可以通过转至 https://aws.amazon.com/ 并选择我的账户来查看当前的账户活动并管理您的账户。

创建具有管理访问权限的用户

注册后，请保护您的安全 AWS 账户 AWS 账户根用户 AWS IAM Identity Center，启用并创建管理用户，这样您就不会使用 root 用户执行日常任务。

保护你的 AWS 账户根用户

1. 选择 Root 用户并输入您的 AWS 账户 电子邮件地址，以账户所有者的身份登录。AWS Management Console在下一页上，输入您的密码。

   要获取使用根用户登录方面的帮助，请参阅 《AWS 登录 用户指南》中的以根用户身份登录。
2. 为您的根用户启用多重身份验证 (MFA)。

   有关说明，请参阅 I A M 用户指南中的为 AWS 账户 根用户启用虚拟 MFA 设备（控制台）。

创建具有管理访问权限的用户

1. 启用 IAM Identity Center。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的启用 AWS IAM Identity Center。

2. 在 IAM Identity Center 中，为用户授予管理访问权限。

   有关使用 IAM Identity Center 目录 作为身份源的教程，请参阅《用户指南》 IAM Identity Center 目录中的使用默认设置配置AWS IAM Identity Center 用户访问权限。

以具有管理访问权限的用户身份登录

- 要使用您的 IAM Identity Center 用户身份登录，请使用您在创建 IAM Identity Center 用户时发送到您的电子邮件地址的登录网址。

  有关使用 IAM Identity Center 用户登录的帮助，请参阅AWS 登录 用户指南中的登录 AWS 访问门户。

将访问权限分配给其他用户

1. 在 IAM Identity Center 中，创建一个权限集，该权限集遵循应用最低权限的最佳做法。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的创建权限集。

2. 将用户分配到一个组，然后为该组分配单点登录访问权限。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的添加组。

分配服务访问权限

通过向您的角色或用户添加和权限，启用对 AWS OpsWorks Stacks 服务（以及 Stacks 所依赖的`AWSOpsWorks_FullAccess`相关服务）的`AmazonS3FullAccess`访问权限。 AWS OpsWorks

有关添加权限的更多信息，请参阅：添加 IAM 身份权限（控制台）。

您现已完成所有设置步骤，可开始此演练。

步骤 2：创建堆栈

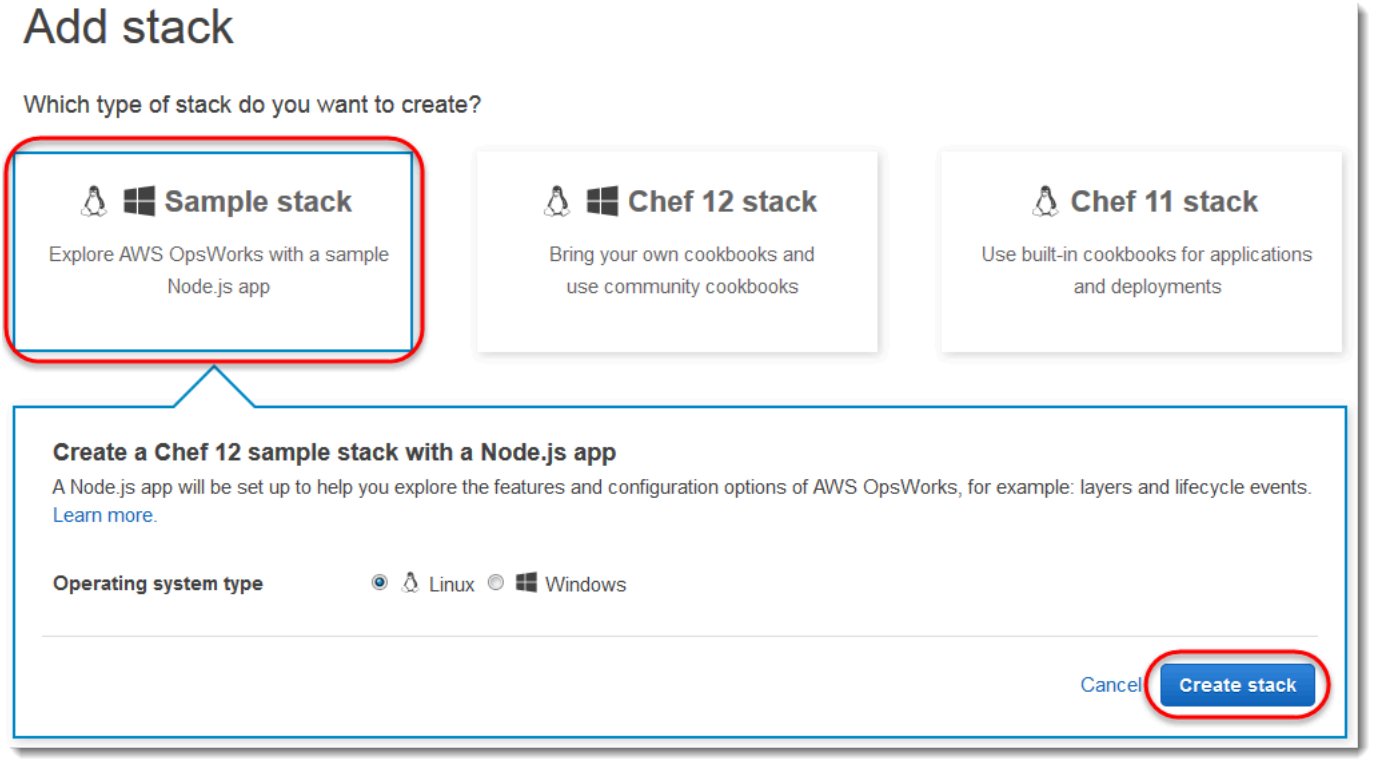
> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

在此步骤中，您将使用 AWS OpsWorks Stacks 控制台创建堆栈。堆栈是实例（例如 Amazon EC2 实例）和相关 AWS 资源的集合，它们具有共同的用途，您想一起管理。（有关更多信息，请参阅 堆栈。）本演练将只有一个实例。

在开始本步骤之前，请满足先决条件。

要创建 堆栈，请执行以下操作：

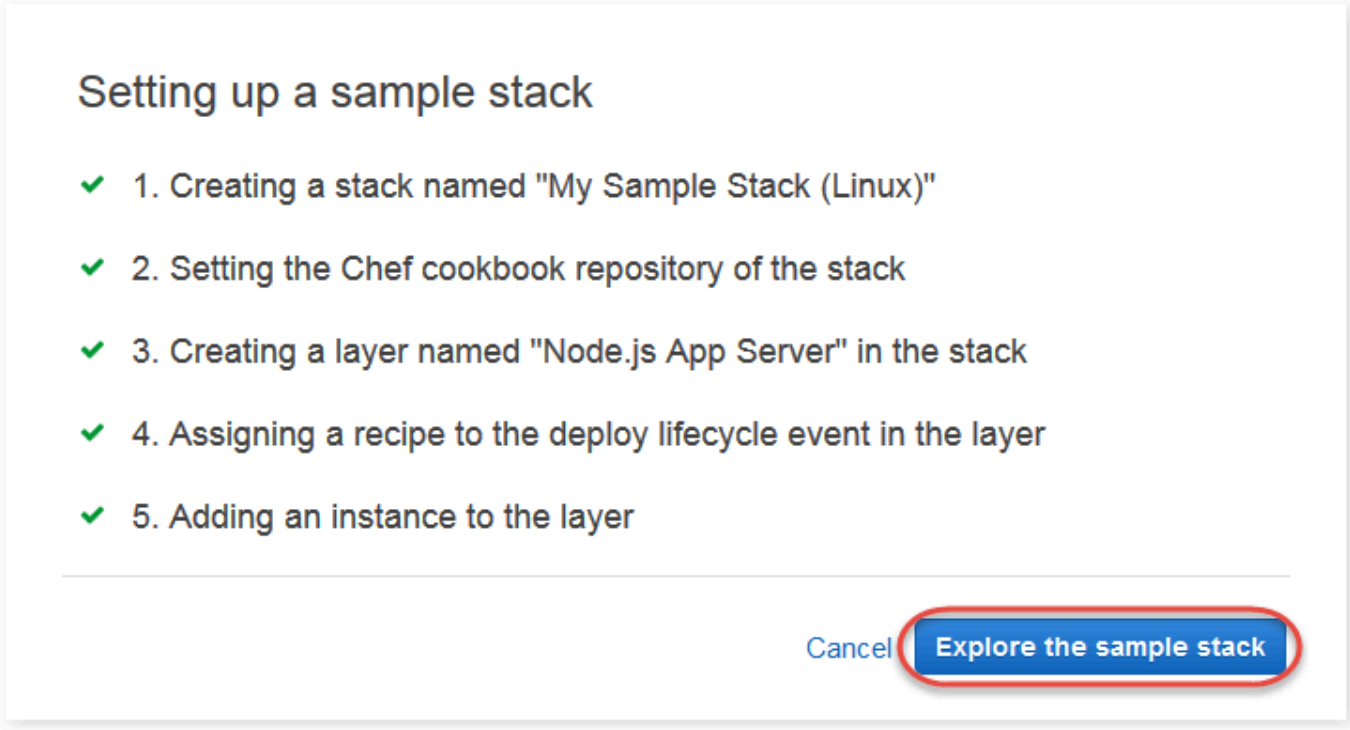
1. 登录 AWS Management Console 并打开 AWS OpsWorks 控制台，网址为 https://console.aws.amazon.com/opsworks/。
2. 执行以下任一操作 (如果适用)：
   - 如果显示 “欢迎使用 AWS OpsWorks 堆栈” 页面，请选择 “添加您的第一个堆栈” 或 “添加您的第一个 AWS OpsWorks 堆栈”（两个选项的作用相同）。这将显示 Add stack 页面。
   - 如果显示OpsWorks 控制面板页面，请选择添加堆栈。这将显示 Add stack 页面。
3. 在出现 Add stack 页面后，选择 Sample stack (如果尚未选择)。
4. 在已选择 Linux 作为 Operating system type 的情况下，选择 Create stack：



5. AWS OpsWorks 堆栈会创建一个名为 “我的示例堆栈” (Linux) 的堆栈。 AWS OpsWorks Stacks 还添加了将应用程序部署到堆栈所需的所有必要组件：

   - 层，它是一组实例的蓝图。它将指定实例的设置、资源、已安装的程序包和安全组等项目。（有关更多信息，请参阅 图层。）该层名为 Node.js App Server。
   - 实例，在本例中是一个 Amazon Linux 2 EC2 实例。(有关实例的更多信息，请参阅实例。) 实例的主机名为 nodejs-server1。
   - 应用程序，它是要在实例上运行的代码。(有关应用程序的更多信息，请参阅应用程序。) 应用程序名为 Node.js Sample App。

6. AWS OpsWorks 堆栈创建堆栈后，选择 “浏览示例堆栈” 以显示 “我的示例堆栈 (Linux)” 页面（如果您多次完成此演练，“我的示例堆栈 (Linux)” 后面可能有一个序号，例如 2 或 3）：



在下一个步骤中，您将启动实例并将应用程序部署到实例。

步骤 3：启动实例并部署应用程序

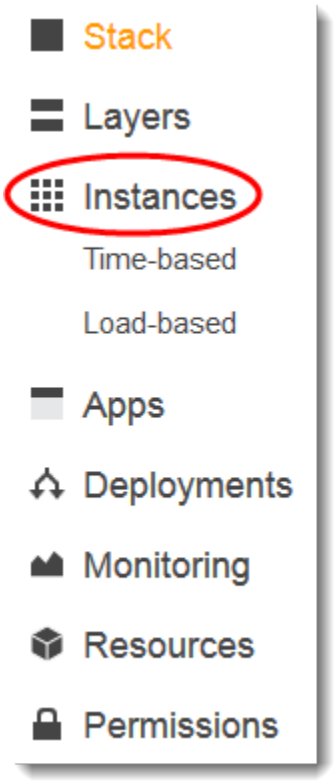
> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

既然您已经有了一个实例和一个应用程序，请启动该实例并将该应用程序部署到该实例。

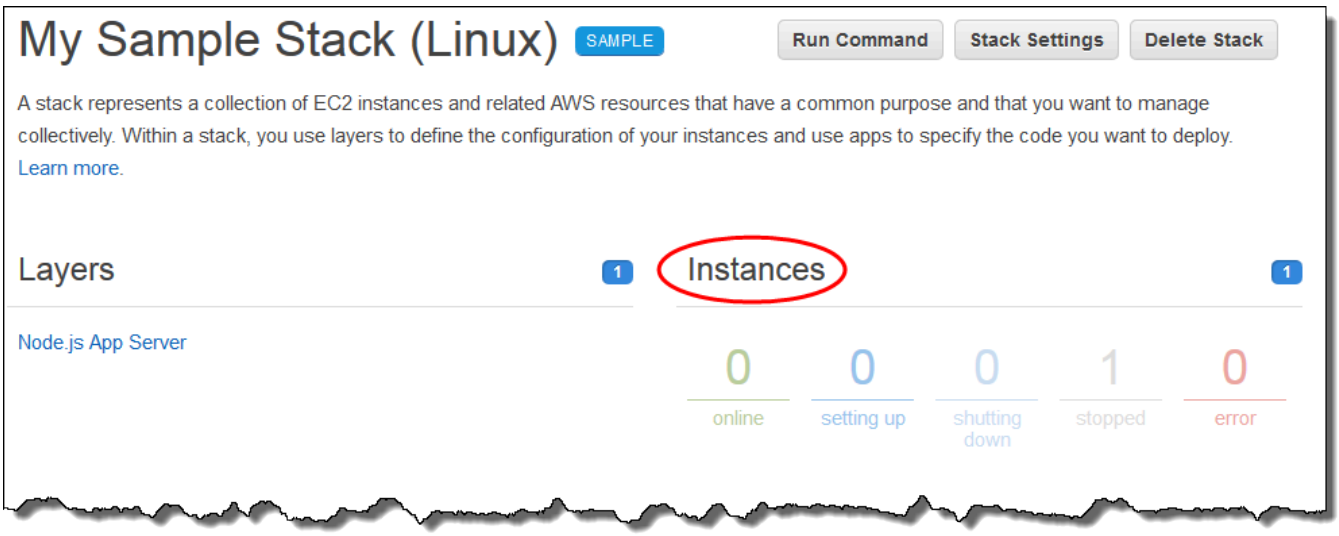
启动实例并部署应用程序

1. 请执行以下操作之一：

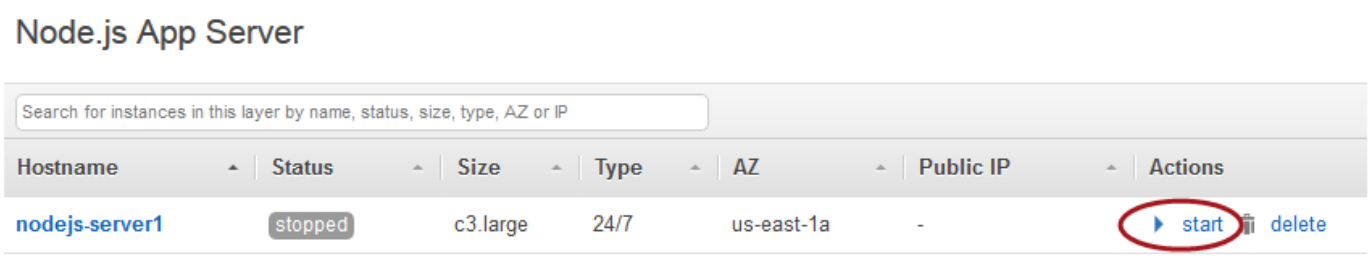
   - 在服务导航窗格中，选择 Instances：



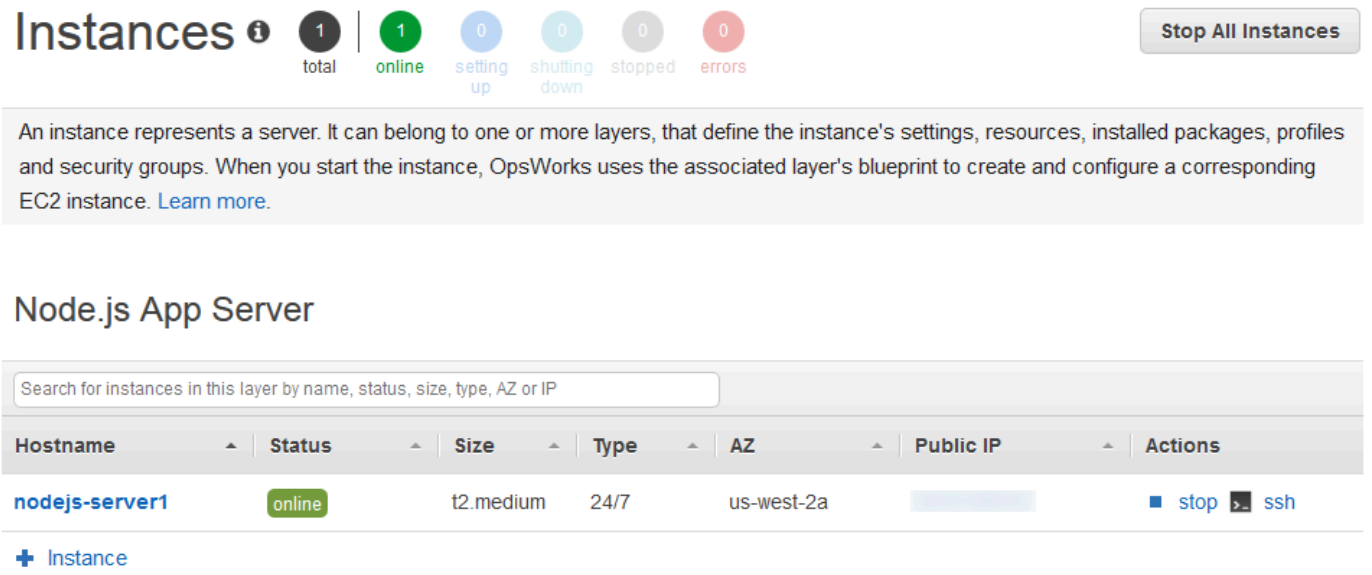
- 在 My Sample Stack (Linux) 页面上，选择 Instances：



2. 在 Instances 页面上，对于 Node.js App Server 中的 nodejs-server1，选择 start：



3. 在 online 圆圈呈亮绿色后再继续。(如果您看到了失败消息，请参阅调试和故障排除指南。)
4. 在实例设置过程中， AWS OpsWorks Stacks 会将应用程序部署到该实例。
5. 您的结果应该与以下屏幕截图类似，然后才能继续 (如果您收到失败消息，则可能需要参阅调试和故障排除指南)：



您现在有一个实例，该实例具有已部署到它的应用程序。

在下一个步骤中，您将测试实例上的应用程序。

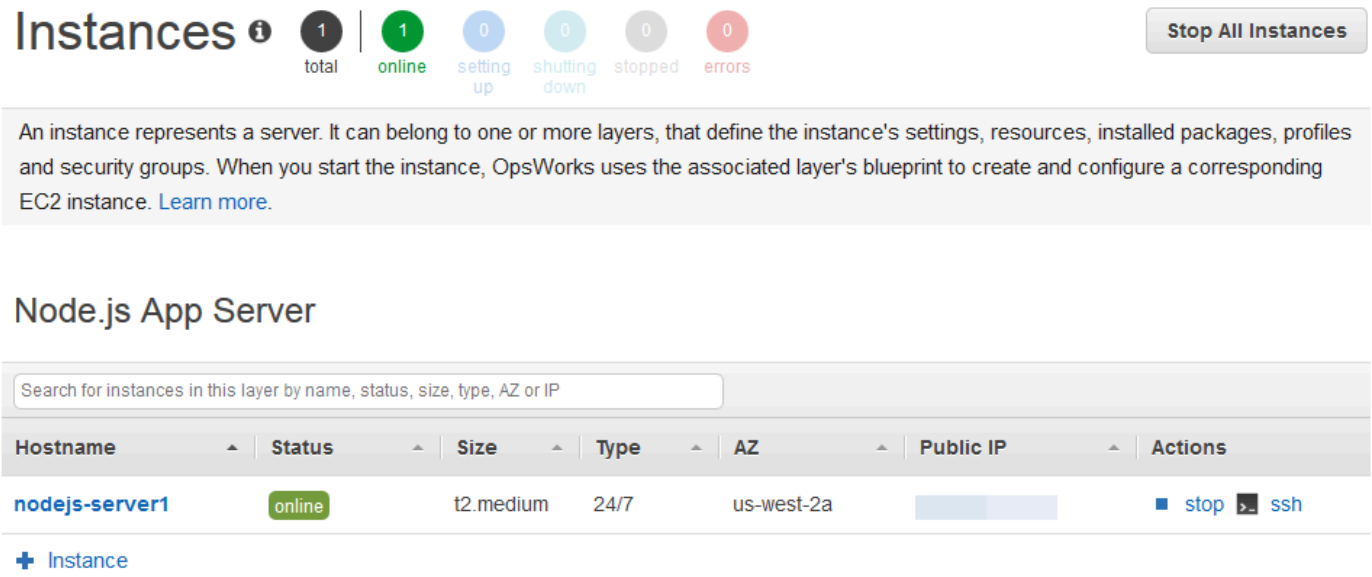
步骤 4：测试实例上的已部署应用程序

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

测试实例上的应用程序部署的结果。

测试实例上的部署

1. 在上一步中显示的 Instances 页面上，对于 Node.js App Server 中的 nodejs-server1 的 Public IP，选择 IP 地址。



2. 在祝贺网页上的 Leave a comment 文本框中，键入注释，然后选择 Send 来测试应用程序。该应用程序将您的注释添加到网页中。继续输入注释并根据需要选择 Send。

3. 如果你有 Twitter 账号，请选择 “推文” 或 “关注 @”AWSOpsWorks，然后按照屏幕上的说明在推特上发布有关该应用程序的推文或关注 @ AWSOpsWorks。

现在，您已成功地测试了实例上部署的应用程序。

在剩下的步骤中，您可以使用 AWS OpsWorks Stacks 控制台浏览堆栈及其组件的设置。在下一个步骤中，您可以通过检查堆栈的设置来开始您的探索。

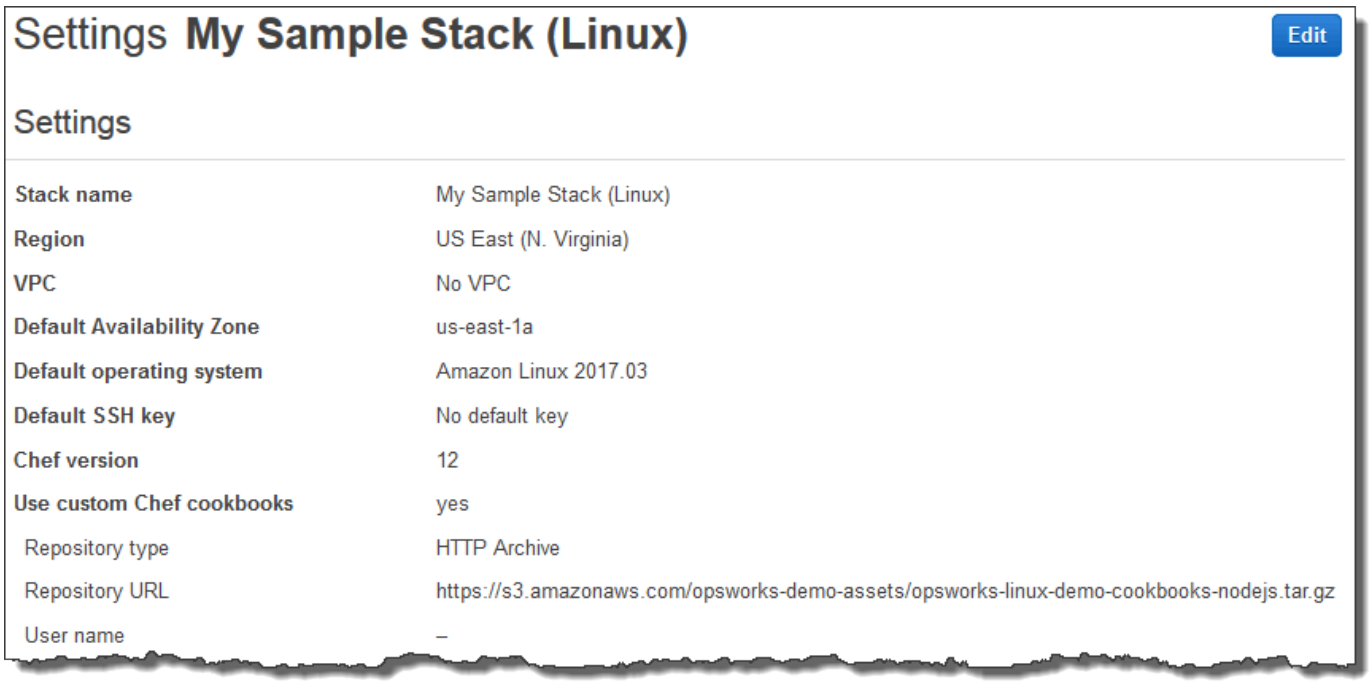
步骤 5：探索堆栈的设置

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

检查 AWS OpsWorks 堆栈是如何设置堆栈的。

显示堆栈的设置

1. 在服务导航栏中，选择 Stack。此时将显示 My Sample Stack (Linux) 页面。
2. 选择 Stack Settings。此时将显示 Settings My Sample Stack (Linux) 页面：



要详细了解多种设置，请选择 Edit，然后将光标悬停在各个设置上。(并非所有设置都具备屏幕上的说明。) 有关这些设置的更多信息，请参阅 创建新堆栈。

要浏览本演练中使用的 Chef 食谱，请打开上的 opsworks-linux-demo-cookbooks-nod ejs 存储库。GitHub

在下一个步骤中，您可以探索层的设置。

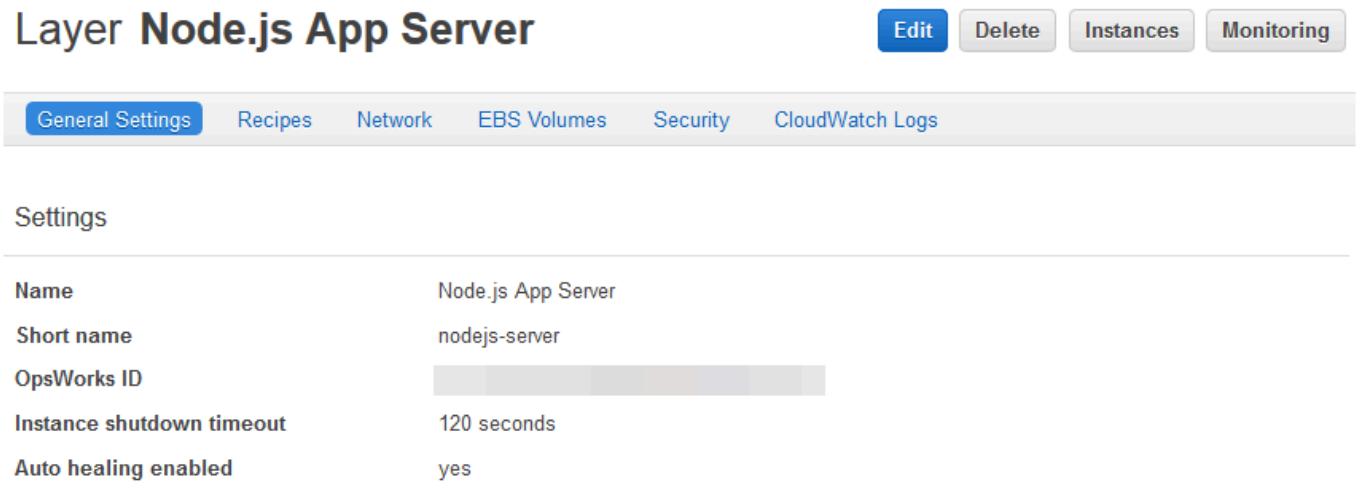
步骤 6：探索层的设置

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

检查 AWS OpsWorks Stacks 是如何设置图层的。

显示层的设置

1. 在服务导航窗格中，选择 Layers。此时将显示 Layers 页面。
2. 选择 Node.js App Server。此时将显示 Layer Node.js App Server 页面。要查看层的设置，请选择 General Settings、Recipes、Network、EBS Volumes 和 Security：



要详细了解多种设置，请选择 Edit，然后将光标悬停在各个设置上。(并非所有设置都具备屏幕上的说明。) 有关这些设置的更多信息，请参阅 编辑图 OpsWorks 层的配置。

在下一个步骤中，您可以探索实例的设置和日志。

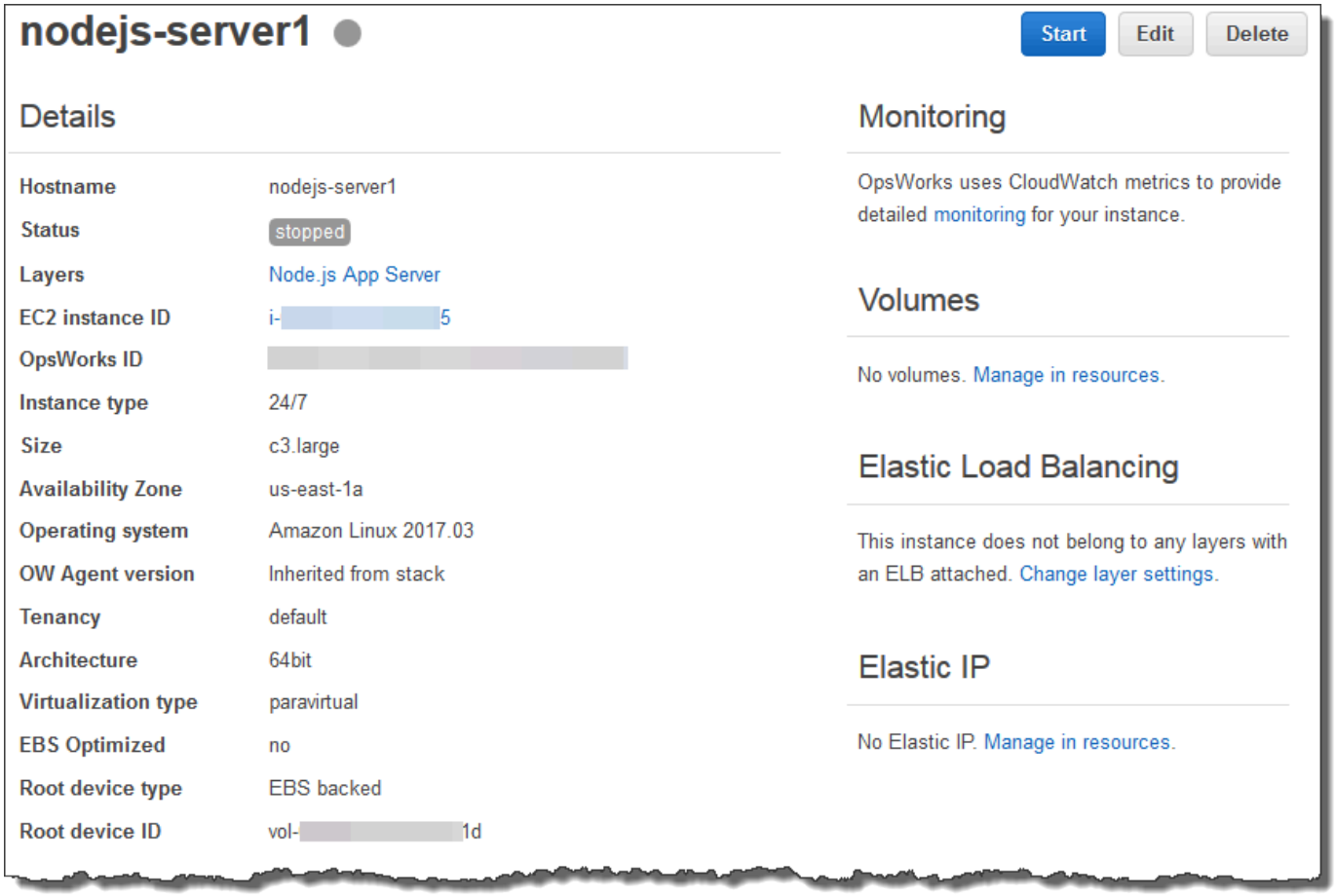
步骤 7：探索实例的设置和日志

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

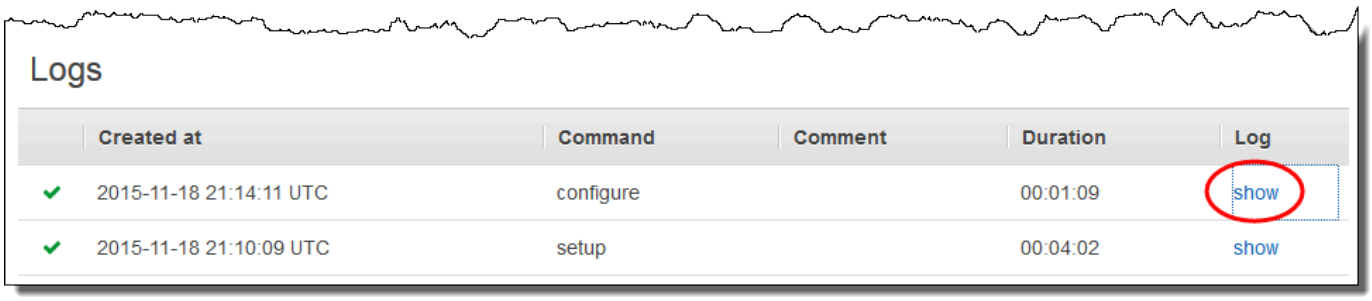
检查 AWS OpsWorks Stacks 用于启动实例的设置。您还可以检查 AWS OpsWorks Stacks 创建的实例日志。

显示实例的设置和日志

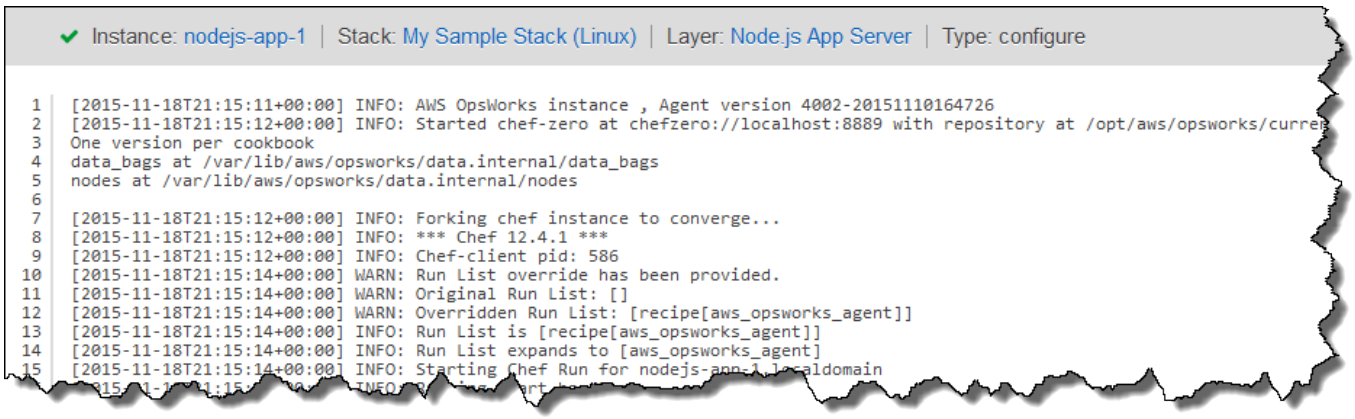
1. 在服务导航窗格中，选择 Instances。此时将显示 Instances 页面。
2. 对于 Node.js App Server，请选择 nodejs-server1。实例的属性页面随即显示。



3. 要浏览实例日志，请在 Logs 部分中，对于 Log，选择 show。



4. AWS OpsWorks Stacks 在单独的 Web 浏览器选项卡中显示日志。



要详细了解某些实例设置的含义，请返回到 nodejs-server1 页面，选择 Stop，当您看到确认消息时，选择 Stop。在 Status 从 stopping 变为 stopped 后，选择 Edit，然后将光标悬停在各个设置上。(并非所有设置都具备屏幕上的说明。) 有关这些设置的更多信息，请参阅 将实例添加到层。

查看完设置后，选择 Start 以重新启动实例，然后等到 Status 变为 online。否则，您之后将无法测试应用程序，因为实例将保持停止状态。

> **Note**
>
> 如果您想登录实例进行进一步探索，则必须先向 AWS OpsWorks 堆栈提供有关您的 SSH 公钥的信息（您可以使用 ssh-keygen 或 PuttyGen 等工具创建），然后必须在 “我的示例堆栈” (Linux) 堆栈上设置权限以允许您的用户登录实例。有关说明，请参阅注册用户的公有 SSH 密钥和使用 SSH 登录。

在下一个步骤中，探索应用程序的设置。

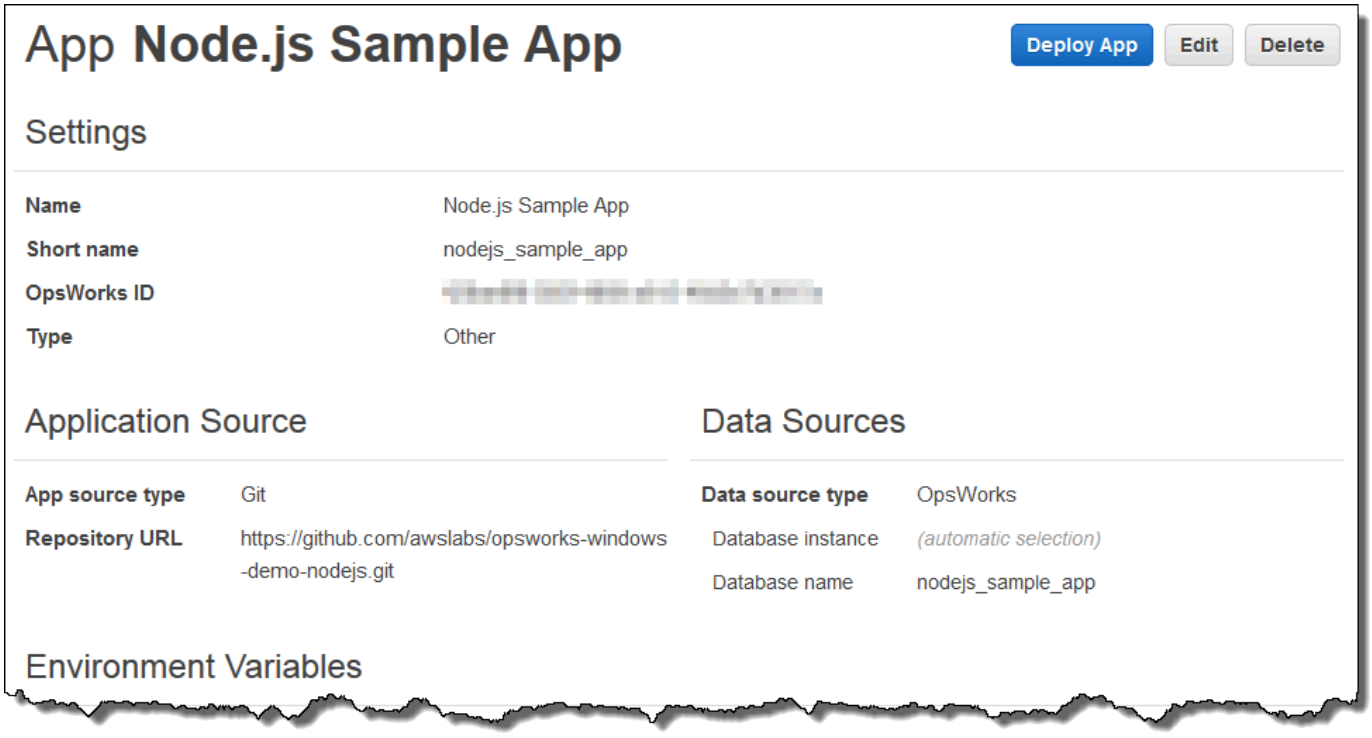
步骤 8：探索应用程序的设置

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

检查 AWS OpsWorks Stacks 用于该应用程序的设置。

显示应用程序的设置

1. 在服务导航窗格中，选择 Apps。将出现 Apps 页面。
2. 选择 Node.js Sample App。此时将显示 App Node.js Sample App 页面。



要了解某些设置的含义，请选择 Edit，然后将光标悬停在各个设置上。(并非所有设置都具备屏幕上的说明。) 有关这些设置的更多信息，请参阅添加应用程序。

在下一个步骤中，您可以探索层监控报告。

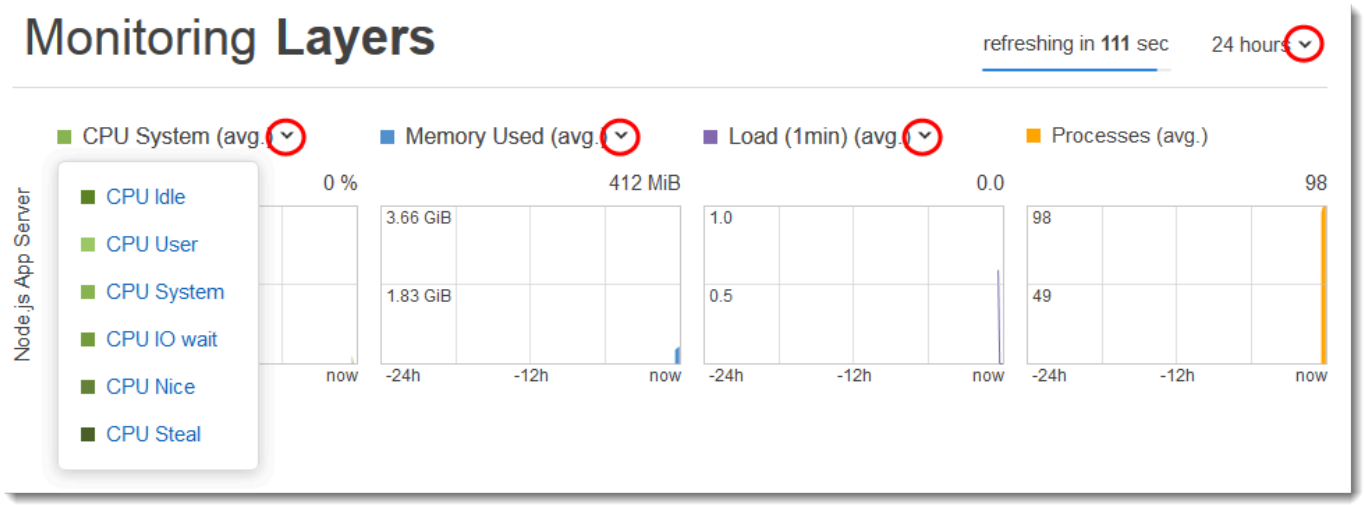
步骤 9：探索层监控报告

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

检查 AWS OpsWorks Stacks 生成的有关该层计算性能的报告。

显示层监控报告

1. 在服务导航窗格中，选择 Monitoring。此时将显示 Monitoring Layers 页面。
2. 要浏览更多视图，请选择 CPU、Memory、Load 和时间旁边的箭头：



有关这些报告和其他报告的更多信息，请参阅使用亚马逊 CloudWatch 和监控。

在下一个步骤中，您可以探索更多堆栈设置。

步骤 10：探索更多堆栈设置

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

在本步骤中，您可以检查更多堆栈设置。

AWS OpsWorks 堆栈没有运行单独的部署，也没有配置其他资源，也没有调整任何其他权限作为堆栈的一部分，因此 “部署和命令”、“资源” 和 “权限” 页面上没有太多兴趣。如果您仍然要查看这些设置，请分别选择服务导航窗格中的 Deployments、Resources 和 Permissions。如果您需要有关这些页面代表的意义的更多信息，请参阅部署应用程序、资源管理和管理用户权限。

在下一步中，您可以清理用于本演练的 AWS 资源。下一步是可选的。随着您继续了解 AWS OpsWorks Stacks的更多信息，您可能需要继续使用这些 AWS 资源。但是，保留这些 AWS 资源可能会导致您的 AWS 账户持续收取一些费用。如果你想保留这些 AWS 资源以备后用，现在你已经完成了本演练，你可以直接跳到后续步骤。

步骤 11：(可选) 清除

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

为防止您的 AWS 账户产生额外费用，您可以删除用于本演练的应用程序和 AWS 资源，包括实例和堆栈 AWS OpsWorks 堆栈。（有关更多信息，请参阅 AWS OpsWorks 定价。）但是，随着您继续了解 AWS OpsWorks Stacks的更多信息，您可能需要继续使用这些 AWS 资源。如果你想让这些 AWS 资源保持可用，现在你已经完成了本演练，你可以跳到后续步骤。

为本演练创建的资源中存储的内容可以包含个人可识别信息。如果您不再希望由 AWS存储此信息，请执行本主题中的步骤。

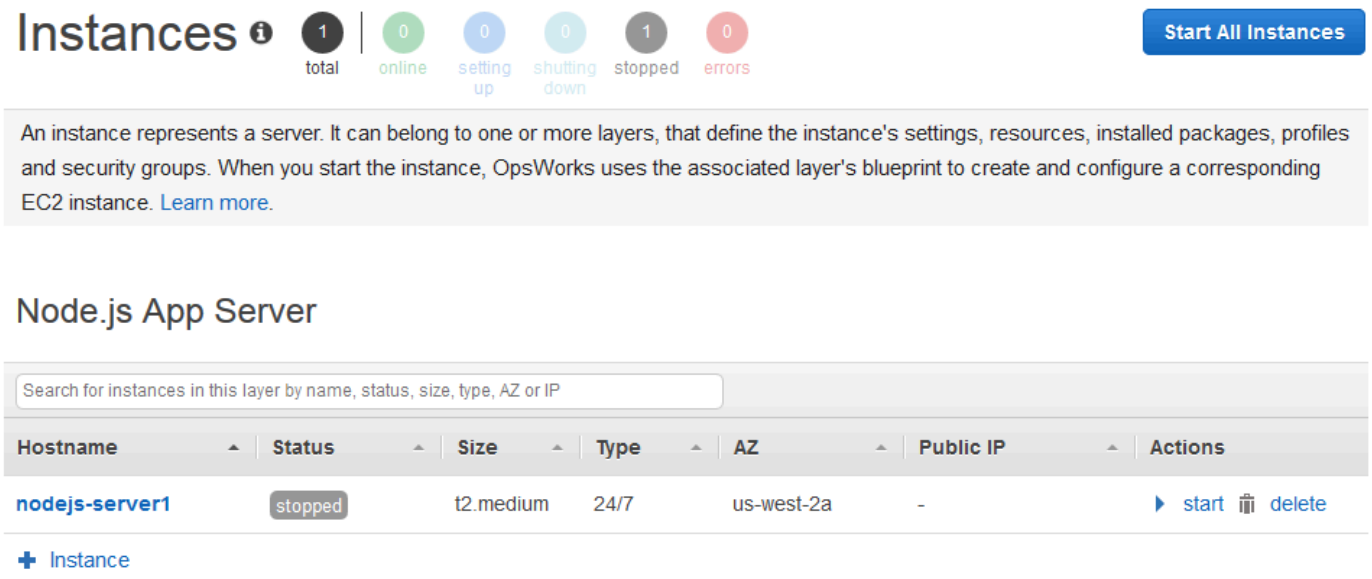
从堆栈中删除应用程序

1. 在服务导航窗格中，选择 Apps。此时将显示 Apps 页面。
2. 对于 Node.js Sample App 中的 Actions，选择 delete。看到确认消息后，选择 Delete。删除应用程序后，您将看到 No apps 消息。

删除堆栈的实例

1. 在服务导航窗格中，选择 Instances。此时将显示 Instances 页面。
2. 对于 Node.js App Server 中的 nodejs-server1 的 Actions，选择 stop。看到确认消息后，选择 Stop。

   此过程可能耗时数分钟。 AWS OpsWorks 堆栈完成后，将显示以下结果。



3. 对于 Actions，选择 delete。看到确认消息后，选择 Delete。实例将被删除，随即显示 No instances 消息。

删除堆栈

1. 在服务导航窗格中，选择 Stack。此时将显示 My Sample Stack (Linux) 页面。
2. 选择 Delete Stack。看到确认消息后，选择 Delete。堆栈已删除，并显示OpsWorks控制面板页面。

或者，如果您不想重复使用用户和 Amazon EC2 密钥对来访问其他 AWS 服务和 EC2 实例，则可以将其删除。有关说明，请参阅：删除 IAM 用户及 Amazon EC2 密钥对和 Linux 实例。

现在，您已完成本演练。有关更多信息，请参阅 后续步骤。

后续步骤

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS：Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

现在您已经完成了本演练，可以进一步了解如何使用 AWS OpsWorks Stacks：

- 练习使用 Stacks AWS OpsWorks 自己手动重新创建此堆栈。请参阅 入门：Linux。
- 探索 AWS OpsWorks Stacks 在本演练中使用的食谱和应用程序。请参阅随附的了解更多：探索本演练中用到的说明书 演练中的了解更多：探索本演练中用到的应用程序和入门：Linux。
- 在 Windows 实例中练习使用 AWS OpsWorks 堆栈。请参阅 入门：Windows。
- 通过了解如何创建新堆栈来了解有关堆栈的更多信息。
- 通过编辑图 OpsWorks 层的配置了解有关层的更多信息。
- 通过将实例添加到层了解有关实例的更多信息。
- 通过部署应用程序了解有关应用程序的更多信息。
- 了解有关 说明书和诀窍 的更多信息。
- 创建您自己的说明书。请参阅 入门：说明书。
- 利用安全性和权限了解如何控制对堆栈的访问权限。

Linux 堆栈入门

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS：Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

在本演练中，您将学习如何使用 AWS OpsWorks 堆栈创建 Node.js 应用程序环境。完成后，您将有一个运行 Chef 12 的 Amazon Elastic Compute Cloud (Amazon EC2) 实例、一台 Node.js HTTP 服务器和一个可用来与 Twitter 交互并在网页上留下评论的 Web 应用程序。

Chef 是一个第三方框架，用来配置和维护服务器 (如 EC2 实例) 以及在这些服务器上部署和维护应用程序的方式。如果你不熟悉 Chef，在完成本演练后，我们建议你更多地了解 Chef，这样你就可以充分利用 AWS OpsWorks Stacks 所提供的一切。(有关更多信息，请参阅了解 Chef 网站。)

AWS OpsWorks Stacks 支持四种 Linux 发行版：亚马逊 Linux、Ubuntu Server、CentOS 和红帽企业 Linux。在本演练中，我们使用 Ubuntu 服务器。 AWS OpsWorks 堆栈也适用于 Windows 服务器。尽管我们有针对 Windows Server 堆栈的同等演练，但我们建议你先完成本演练，以了解关于 AWS OpsWorks 堆栈和 Chef 的基本概念，这些概念不会在那里重复。完成本演练后，请参阅入门：Windows 演练。

主题

步骤 1：完成前提条件

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS：Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

完成以下设置步骤，然后您才能开始演练。这些设置步骤包括注册 AWS 帐户、创建管理用户以及为 AWS OpsWorks Stacks 分配访问权限。

如果您已经完成了入门：示例演练，则您已经满足本演练的先决条件，可以向前跳到步骤 2：创建堆栈。

主题

- 注册获取 AWS 账户
- 创建具有管理访问权限的用户
- 分配服务访问权限

注册获取 AWS 账户

如果您没有 AWS 账户，请完成以下步骤来创建一个。

报名参加 AWS 账户

1. 打开 https://portal.aws.amazon.com/billing/signup。
2. 按照屏幕上的说明进行操作。

   在注册时，将接到一通电话，要求使用电话键盘输入一个验证码。

   当您注册时 AWS 账户，就会创建AWS 账户根用户一个。根用户有权访问该账户中的所有 AWS 服务 和资源。作为安全最佳实践，请为用户分配管理访问权限，并且只使用根用户来执行需要根用户访问权限的任务。

AWS 注册过程完成后会向您发送一封确认电子邮件。在任何时候，您都可以通过转至 https://aws.amazon.com/ 并选择我的账户来查看当前的账户活动并管理您的账户。

创建具有管理访问权限的用户

注册后，请保护您的安全 AWS 账户 AWS 账户根用户 AWS IAM Identity Center，启用并创建管理用户，这样您就不会使用 root 用户执行日常任务。

保护你的 AWS 账户根用户

1. 选择 Root 用户并输入您的 AWS 账户 电子邮件地址，以账户所有者的身份登录。AWS Management Console在下一页上，输入您的密码。

   要获取使用根用户登录方面的帮助，请参阅 《AWS 登录 用户指南》中的以根用户身份登录。

2. 为您的根用户启用多重身份验证 (MFA)。

   有关说明，请参阅 I A M 用户指南中的为 AWS 账户 根用户启用虚拟 MFA 设备（控制台）。

创建具有管理访问权限的用户

1. 启用 IAM Identity Center。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的启用 AWS IAM Identity Center。
2. 在 IAM Identity Center 中，为用户授予管理访问权限。

   有关使用 IAM Identity Center 目录 作为身份源的教程，请参阅《用户指南》 IAM Identity Center 目录中的使用默认设置配置AWS IAM Identity Center 用户访问权限。

以具有管理访问权限的用户身份登录

- 要使用您的 IAM Identity Center 用户身份登录，请使用您在创建 IAM Identity Center 用户时发送到您的电子邮件地址的登录网址。

  有关使用 IAM Identity Center 用户登录的帮助，请参阅AWS 登录 用户指南中的登录 AWS 访问门户。

将访问权限分配给其他用户

1. 在 IAM Identity Center 中，创建一个权限集，该权限集遵循应用最低权限的最佳做法。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的创建权限集。
2. 将用户分配到一个组，然后为该组分配单点登录访问权限。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的添加组。

分配服务访问权限

通过向您的角色或用户添加和权限，启用对 AWS OpsWorks Stacks 服务（以及 Stacks 所依赖的`AWSOpsWorks_FullAccess`相关服务）的`AmazonS3FullAccess`访问权限。 AWS OpsWorks

有关添加权限的更多信息，请参阅：添加 IAM 身份权限（控制台）。

您现已完成所有设置步骤，可开始此演练。

步骤 2：创建堆栈

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

您将使用 AWS OpsWorks Stacks 控制台创建堆栈。堆栈是实例和相关 AWS 资源的集合，它们具有共同的用途，您希望共同管理。（有关更多信息，请参阅 堆栈。）在本演练中，仅有一个实例。

在开始之前，请完成前提条件 (如果您尚未完成)。

要创建 堆栈，请执行以下操作：

1. 登录 AWS Management Console 并打开 AWS OpsWorks 控制台，网址为 https://console.aws.amazon.com/opsworks/。
2. 执行以下任一操作 (如果适用)：
   - 如果显示 “欢迎使用 AWS OpsWorks 堆栈” 页面，请选择 “添加您的第一个堆栈” 或 “添加您的第一个 AWS OpsWorks 堆栈”（两个选项的作用相同）。这将显示 Add stack 页面。
   - 如果显示 “OpsWorks 控制面板” 页面，请选择 “添加堆栈”。这将显示 Add stack 页面。
3. 在显示 Add stack 页面后，选择 Chef 12 stack (如果尚未选择)。
4. 在 Stack name (堆栈名称) 框中，键入一个名称，例如 **`MyLinuxDemoStack`**。(您可以键入一个不同的名称，但务必在整个演练中用它来替换 `MyLinuxDemoStack`。)
5. 对于区域，选择美国西部（俄勒冈）。
6. 对于 VPC，请执行下列操作之一：
   - 如果 VPC 可用，请选择它。（有关更多信息，请参阅 在 VPC 中运行堆栈。）
   - 否则，请选择 No VPC (无 VPC)。
7. 对于 Default operating system，选择 Linux 和 Ubuntu 18.04 LTS。
8. 对于 Use custom Chef cookbooks，选择 Yes。
9. 对于 Repository type，选择 Http Archive。
10. 对于 Repository URL (存储库 URL)，键入 **`https://s3.amazonaws.com/opsworks-demo-assets/opsworks-linux-demo-cookbooks-nodejs.tar.gz`**

11. 对以下项目保留默认值：

    - Default Availability Zone (us-west-2a)
    - Default SSH key (Do not use a default SSH key)
    - User name (空白)
    - Password (空白)
    - Stack color (深蓝色)

12. 选择 Advanced (高级)。
13. 对于 IAM role，执行下列操作之一 (有关更多信息，请参阅允许 AWS OpsWorks Stacks 代表你行事)：

    - 如果 aws-opsworks-service-role 可用，选择该选项。
    - 如果不可aws-opsworks-service-role用，请选择 “新建 IAM 角色”。

14. 对于 Default IAM instance profile，执行下列操作之一 (有关更多信息，请参阅为在 EC2 实例上运行的应用程序指定权限)：

    - 如果有 aws-opsworks-ec2 个角色可用，请选择它。
    - 如果aws-opsworks-ec双角色不可用，请选择新建 IAM 实例配置文件。

15. 对于 API endpoint region，选择您希望堆栈与其关联的区域 API 终端节点。如果您希望堆栈位于美国东部（弗吉尼亚州北部）区域终端节点内的美国西部（俄勒冈州）区域，请选择 us-east-1。如果您希望堆栈既位于美国西部（俄勒冈州）区域内，也与美国西部（俄勒冈州）区域终端节点关联，则选择 us-west-2。

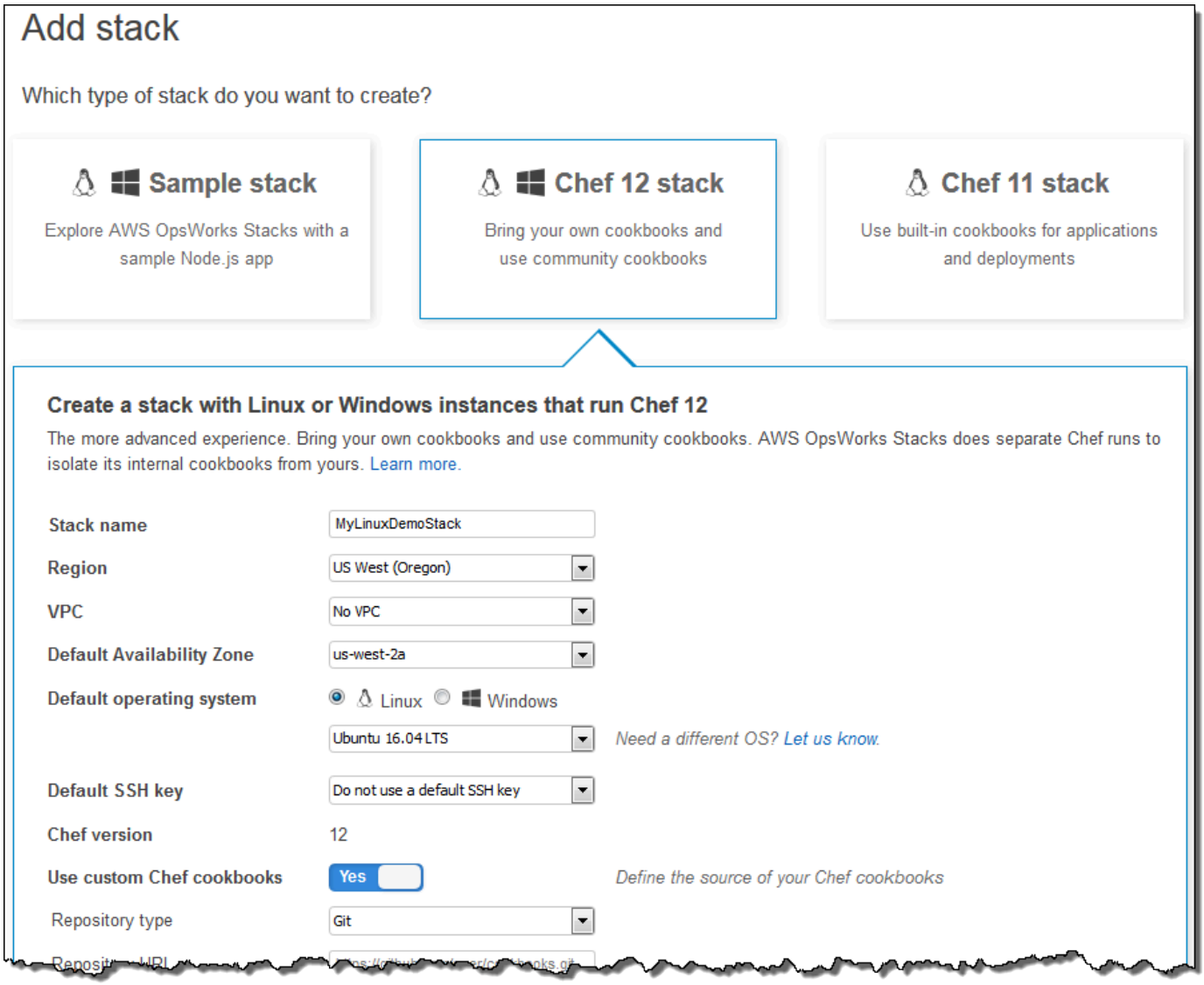
    > **Note**
    >
    > AWS 区域 为了向后兼容，美国东部（弗吉尼亚北部）区域终端节点包括较旧的终端节点，但最佳做法是选择最接近您管理位置的区域终端节点 AWS。有关更多信息，请参阅区域支持。

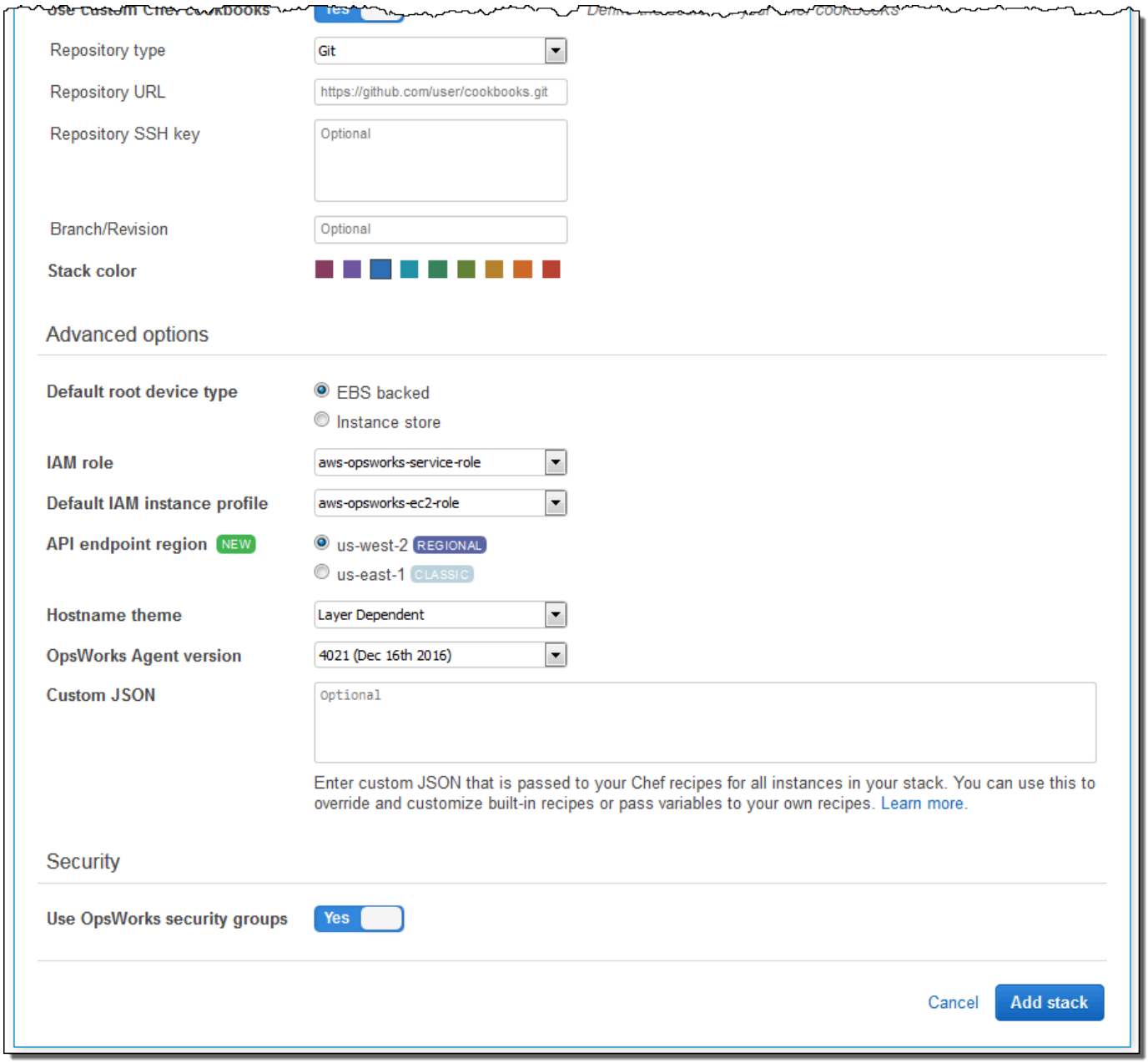
16. 对以下项目保留默认值：

    - Default root device type (EBS backed)
    - Hostname theme (Layer Dependent)
    - OpsWorks 代理版本（最新版本）
    - Custom JSON (空白)

- 使用 OpsWorks 安全组（是）

17. 可能除了 VPC、IAM role 和 Default IAM 实例配置文件 外，您的结果应与以下屏幕截图匹配：





18. 选择 “添加堆栈”。 AWS OpsWorks Stacks 创建堆栈并显示MyLinuxDemoStack页面。

现在，您有了一个包含正确设置的堆栈可用于本演练。

在下一步中，您将在堆栈中添加层。

步骤 3：将层添加到堆栈中

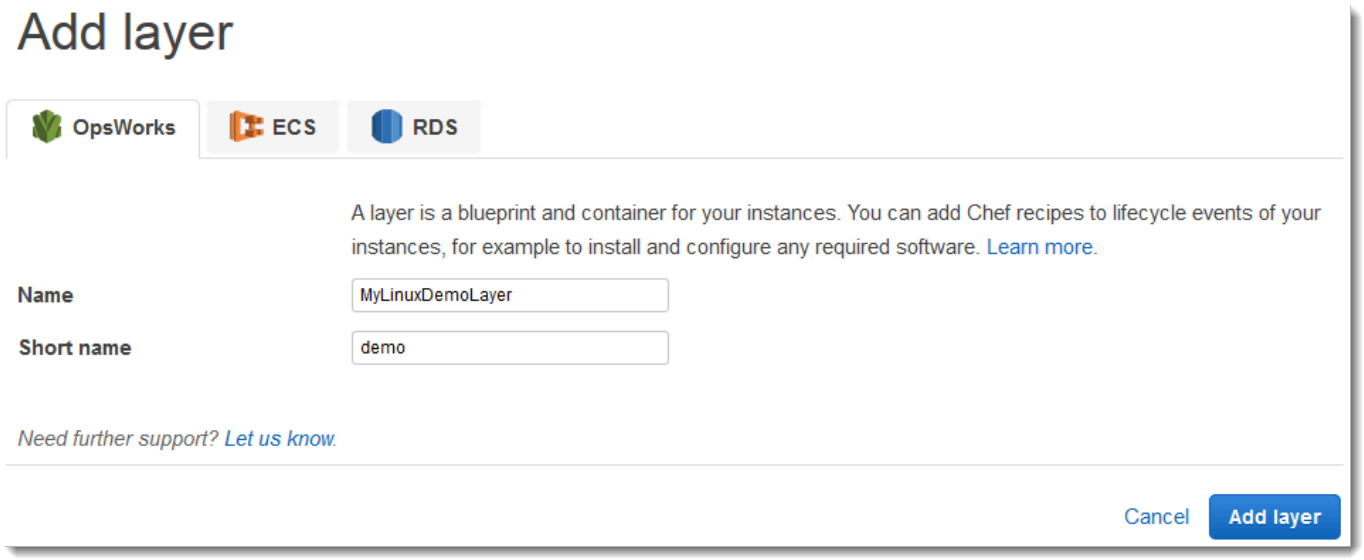
> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

层 是一组实例 (如 Amazon EC2 实例) 的蓝图。它指定如实例设置、资源、已安装的软件包和安全组等信息。下面，我们将一个层添加到堆栈中。(有关层的更多信息，请参阅图层。)

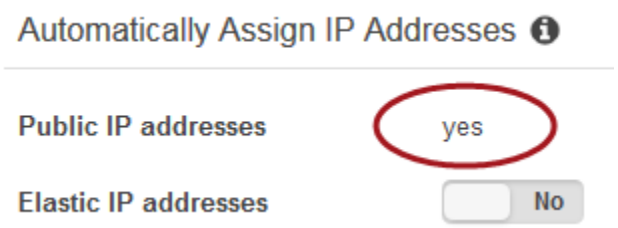
将层添加到堆栈中

1. 显示上一步中的MyLinuxDemoStack页面后，在 “图层” 中选择 “添加图层”：

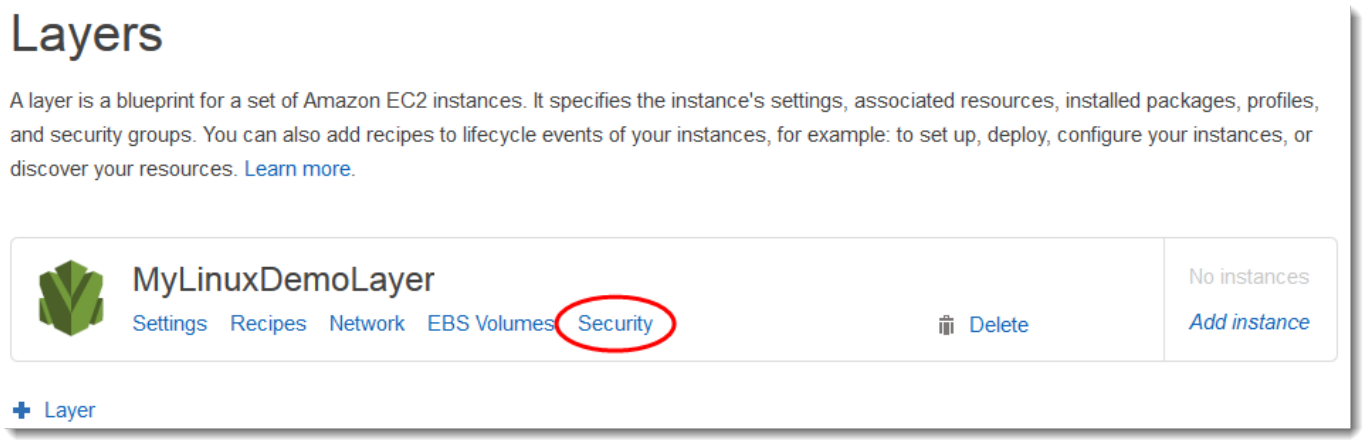
2. 这将显示 Add Layer 页面。在OpsWorks选项卡上，在 “名称” 中键入**MyLinuxDemoLayer**。(您可以键入一个不同的名称，但务必在整个演练中用它来替换 MyLinuxDemoLayer。)
3. 对于 Short name (短名称)，键入 **demo**（您可以键入一个不同的值，但务必在整个演练中用它来替换 demo）：



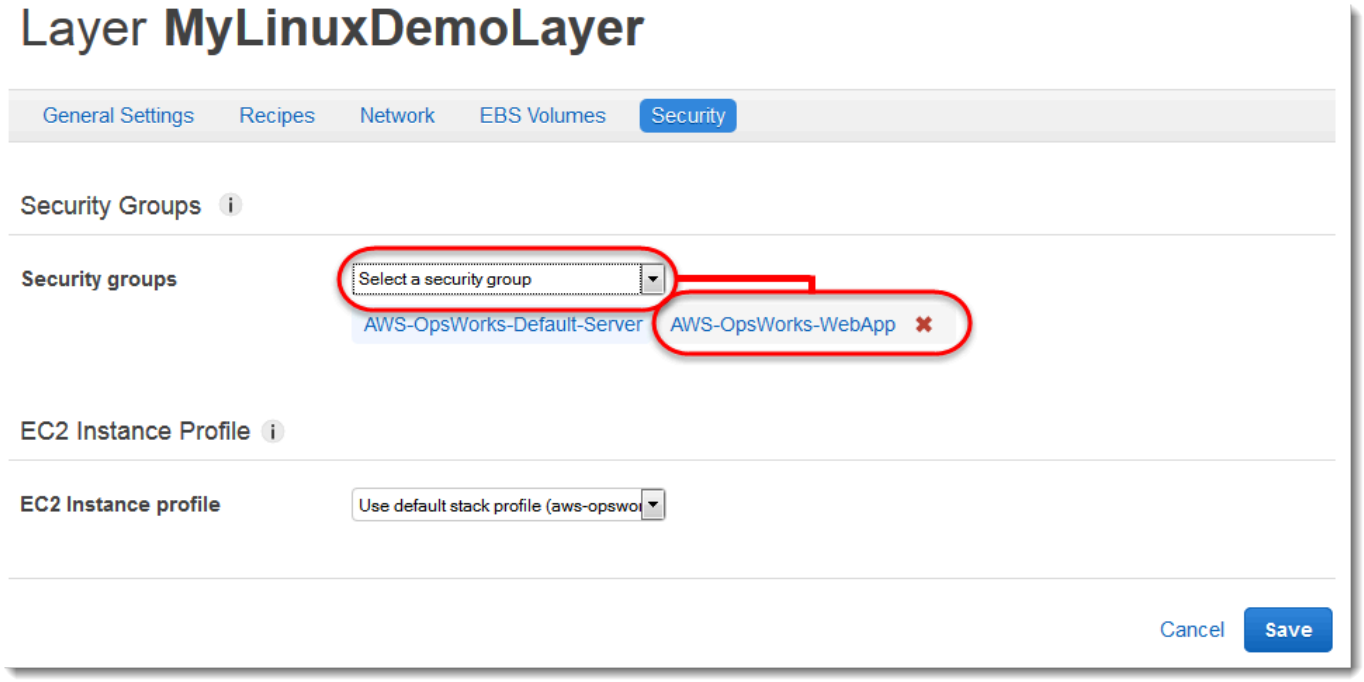
4. 选择 “添加图层”。 AWS OpsWorks Stacks 创建图层并显示 “图层” 页面。
5. 在 “图层” 页面上 MyLinuxDemoLayer，选择 “网络”。
6. 在 Network 选项卡上的 Automatically Assign IP Addresses 下，确认 Public IP addresses 设置为 yes。如果进行了更改，请选择 Save。



7. 在 Layers 页面上，选择 Security：



8. 将显示 MyLinuxDemoLayer “层” 页面，“安全” 选项卡处于打开状态。对于安全组，选择 AWS-OpsWorks-WebApp，然后选择保存：



9. AWS-OpsWorks-WebApp 安全组被添加到层中。（此安全组允许用户在本演练的稍后部分连接到实例上的应用程序。如果没有此安全组，用户将在其 Web 浏览器中收到一条消息，告知他们无法连接到该实例。）

现在，您有了一个包含正确设置的层可用于本演练。

在下一步中，您将指定要部署到实例上的应用程序。

步骤 4：指定要部署到实例上的应用程序

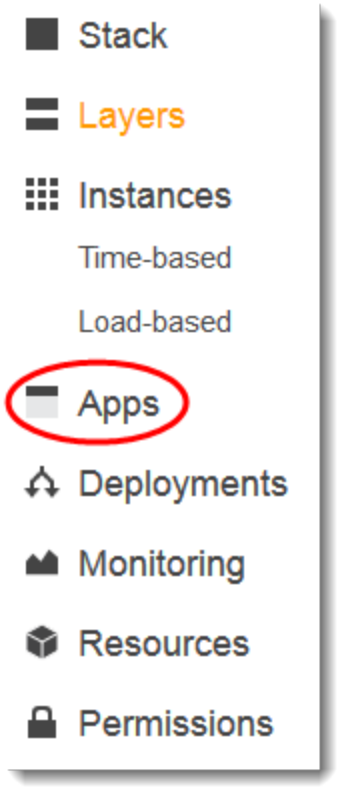
> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

在本演练稍后将要部署到实例的应用程序告知 AWS OpsWorks Stacks。在这种情况下，AWS OpsWorks Stacks 将应用程序定义为要在实例上运行的代码。（有关更多信息，请参阅 应用程序。）

本部分中的过程适用于 Chef 12 和更新的堆栈。有关如何向 Chef 11 堆栈的层中添加应用程序的信息，请参阅步骤 2.4：创建和部署应用程序 - Chef 11。

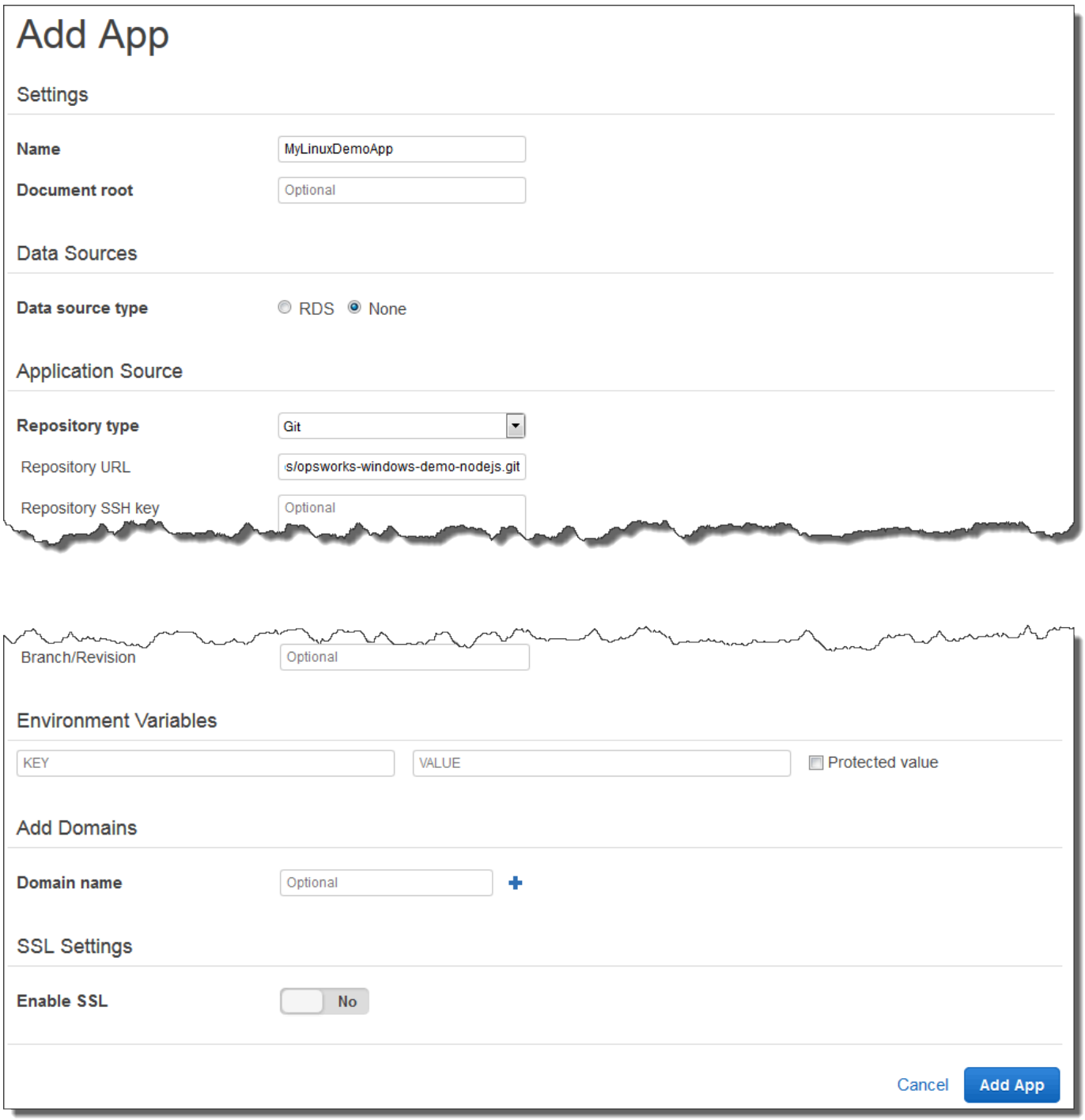
指定要部署的应用程序

1. 在服务导航窗格中，选择 Apps：



2. 将出现 Apps 页面。选择 Add an app。这将显示 Add App 页面。
3. 对于 Settings (设置) 中的 Name (名称)，键入 **MyLinuxDemoApp**。(您可以键入一个不同的名称，但务必在整个演练中用它来替换 MyLinuxDemoApp。)
4. 对于 Application Source (应用程序源) 的 Repository URL (存储库 URL)，键入 **https://github.com/awslabs/opsworks-windows-demo-nodejs.git**
5. 对以下项目保留默认值：
   - Settings，Document root (空白)
   - Data Sources，Data source type (None)
   - Repository type (Git)
   - Repository SSH key (空白)

- Branch/Revision (空白)
- Environment Variables (空白 KEY，空白 VALUE，未选中 Protected Value)
- Add Domains，Domain Name (空白)
- SSL Settings，Enable SSL (No)



6. 选择 “添加应用程序”。 AWS OpsWorks Stacks 添加应用程序并显示应用程序页面。

现在，您有了一个包含正确设置的应用程序可用于本演练。

在下一步中，您将启动实例。

步骤 5：启动实例

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

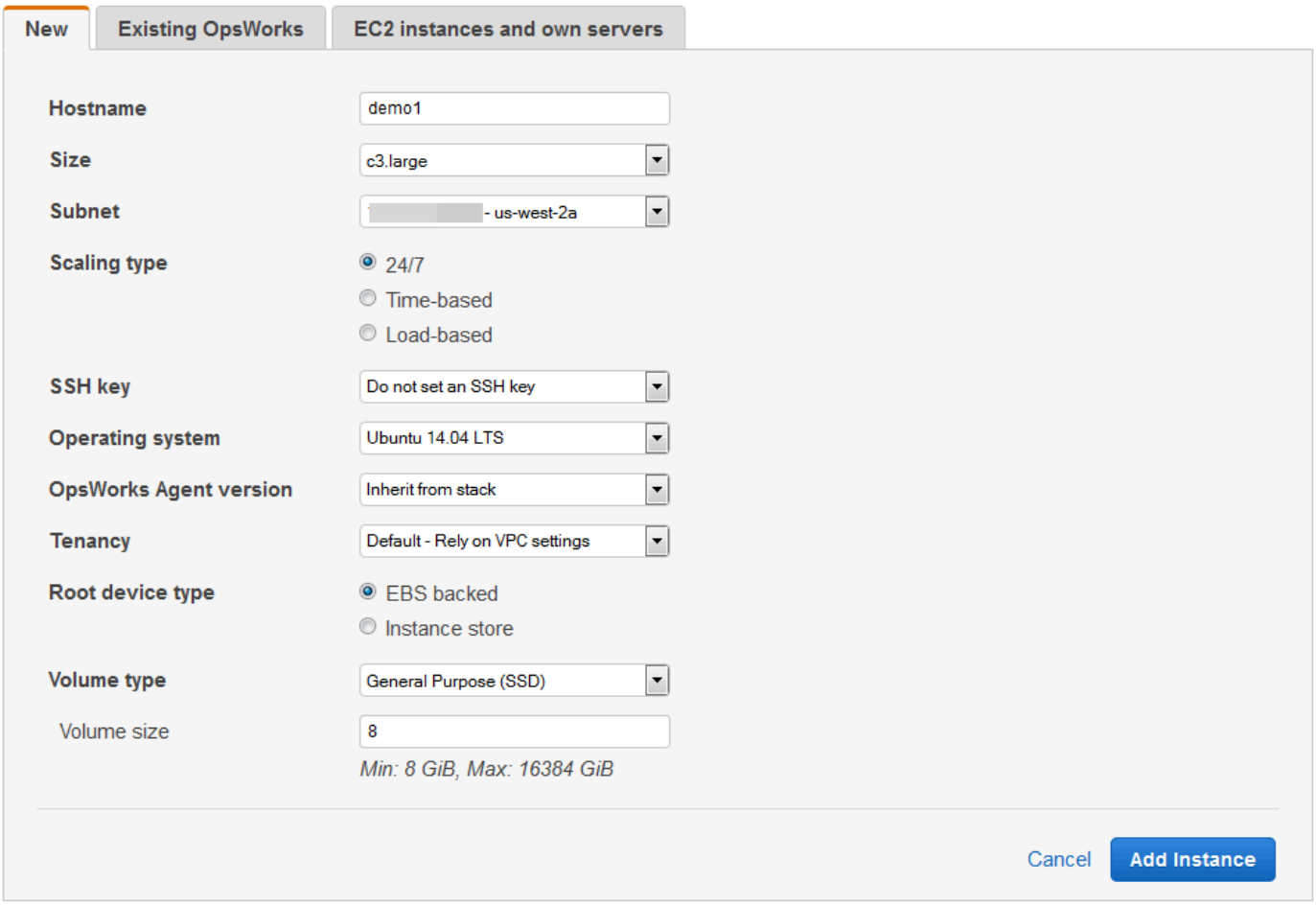
使用 AWS OpsWorks 堆栈启动 Ubuntu Server Amazon EC2 实例。该实例使用您在本演练前面创建的层中定义的设置。（有关更多信息，请参阅 实例。）

启动实例

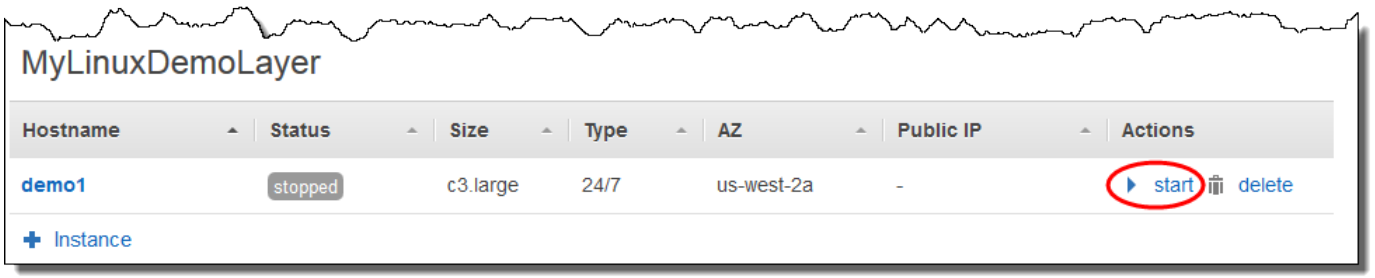
1. 在服务导航窗格中，选择 Instances。这将显示 Instances 页面。
2. 对于 MyLinuxDemoLayer，选择添加实例。
3. 在 New 选项卡上，对以下项目保留默认值：
   - Hostname (demo1)
   - Size (c3.large)
   - Subnet (*IP ##* us-west-2a)
4. 选择 Advanced (高级)。
5. 对以下项目保留默认值：
   - Scaling type (24/7)
   - SSH key (Do not use a default SSH key)
   - Operating system (Ubuntu 18.04 LTS)
   - OpsWorks 代理版本（继承自堆栈）
   - Tenancy (Default - Rely on VPC settings)
   - Root device type (EBS backed)

- Volume type (General Purpose (SSD))
- Volume size (8)

6. 您的结果将类似于以下屏幕截图：



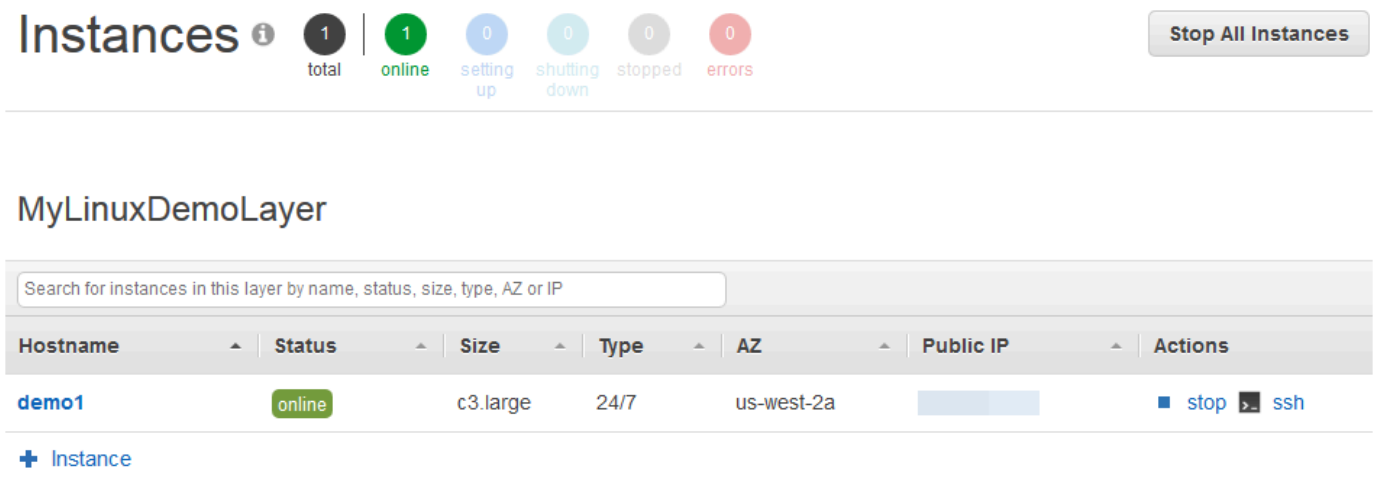
7. 选择添加实例。 AWS OpsWorks Stacks 将实例添加到层并显示 “实例” 页面。
8. 对于 MyLinuxDemoLayerdemo1，在 “操作” 中选择 “开始”：



9. 几分钟后，将出现以下情况：

- setting up 圆圈从 0 变为 1。
- Status 依次变为 stopped、requested、pending、booting、running_setup，最后变成 online。注意，此过程可能耗时数分钟。
- 当 Status 变成 online 后，setting up 圆圈指示器从 1 变成 0，online 圆圈从 0 变成 1 并变为亮绿色。请在 online 圆圈变成亮绿色并显示 1 个实例联机后再继续。

10. 您的结果必须与以下屏幕截图匹配，然后才能继续 (如果您收到失败消息，可能需要查阅调试和故障排除指南)：



现在，您已经准备好一个实例来用于部署应用程序了。

> **Note**
>
> 如果您想登录该实例以进一步探索它，必须先向 AWS OpsWorks Stacks 提供您的公有 SSH 密钥的信息（可以使用诸如 ssh-keygen 或 PuTTYgen 等工具创建），然后必须在 MyLinuxDemoStack 堆栈上设置许可以便您的用户能够登录该实例。有关说明，请参阅注册用户的公有 SSH 密钥和使用 SSH 登录。如果你计划使用 SSH 通过 PuTTY 连接实例，请参阅文档中的使用 PuTTY 从 Windows 连接到你的 Linux 实例。 AWS

在下一步中，您将把应用程序部署到实例上。

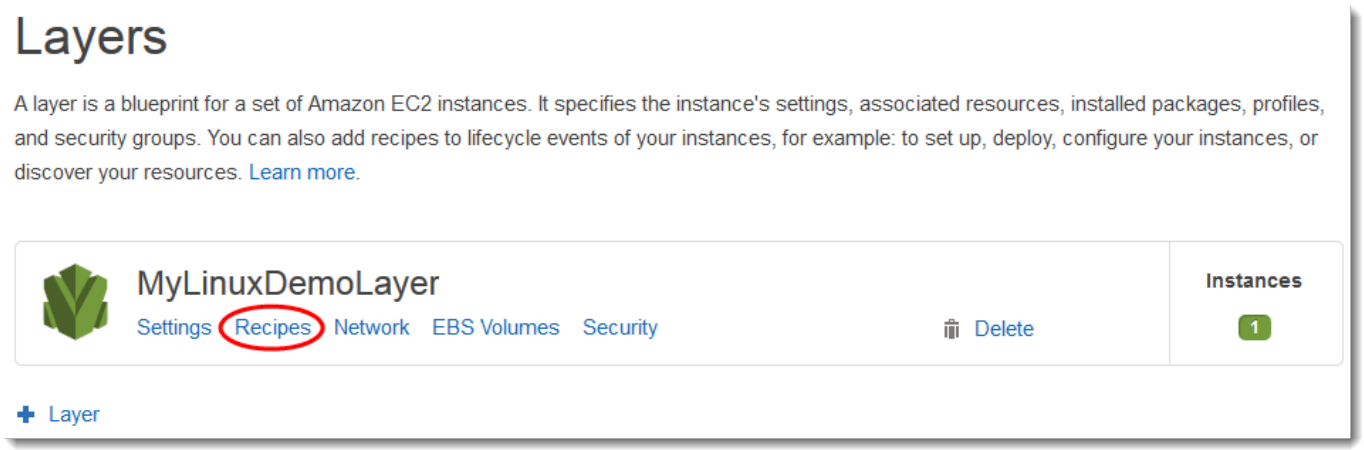
步骤 6：将应用程序部署到实例上

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

在此步骤中，您将从将应用程序部署 GitHub 到正在运行的实例。（有关更多信息，请参阅 部署应用程序。）在部署应用程序之前，必须指定配方以用于协调部署。配方是一个 Chef 概念。配方是一些说明，用 Ruby 语言语法编写，指定要使用的资源和应用这些资源的顺序。(有关更多信息，请转至了解 Chef 网站上的关于配方。)

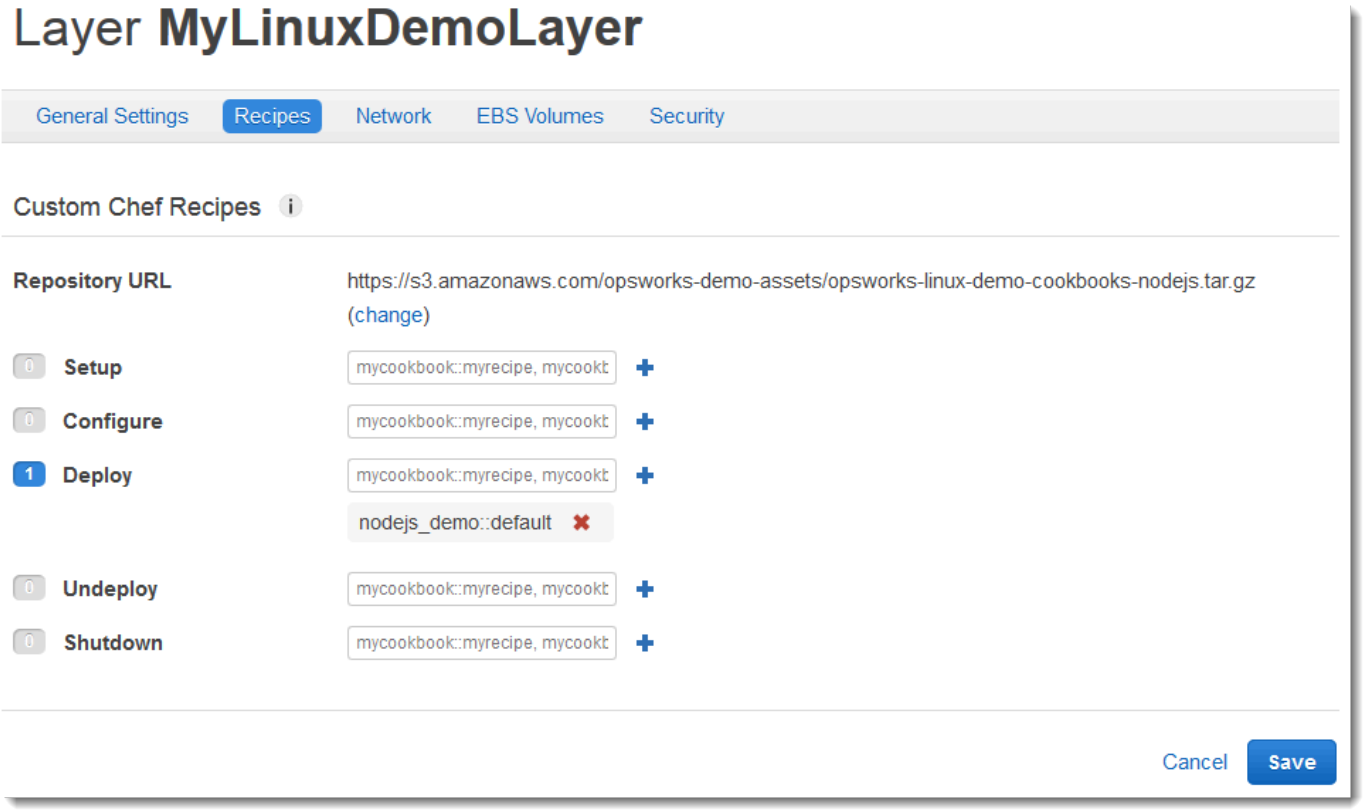
指定配方以用来将应用程序部署到实例上

1. 在服务导航窗格中，选择 Layers。此时将显示 Layers 页面。
2. 对于 MyLinuxDemoLayer，请选择食谱：



将显示 MyLinuxDemoLayer “图层” 页面，“食谱” 选项卡处于打开状态。

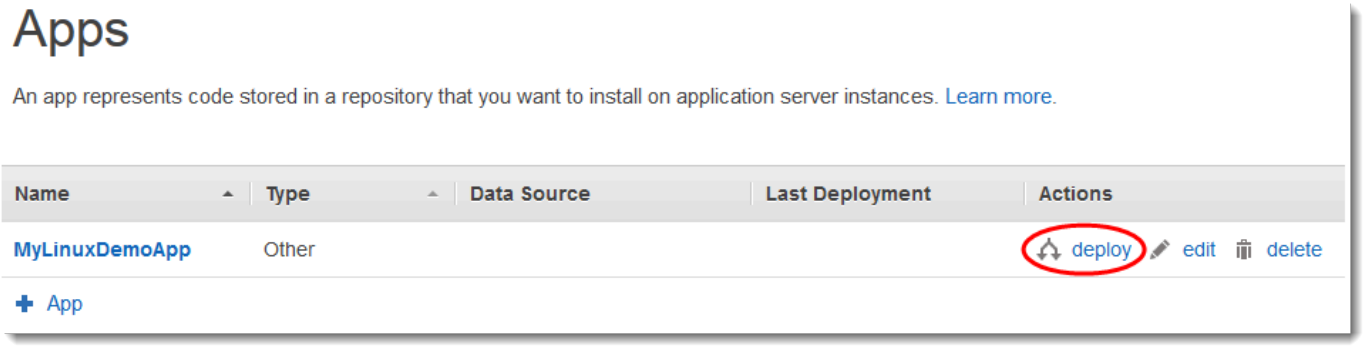
3. 对于 Custom Chef Recipes (自定义 Chef 配方) 的 Deploy (部署)，键入 **`nodejs_demo::default`**，然后按 Enter。`nodejs_demo` 是说明书的名称，`default` 是说明书内目标配方的名称。(要探索配方的代码，请参阅了解更多：探索本演练中用到的说明书。) 您的结果必须与以下屏幕截图匹配：



4. 选择 “保存”。 AWS OpsWorks Stacks 将配方添加到图层的 Deploy 生命周期事件中。

将应用程序部署到实例上

1. 在服务导航窗格中，选择 Apps。此时将显示 Apps 页面。
2. 对于 MyLinuxDemoApp“操作”，选择 “部署”，如以下屏幕截图所示：

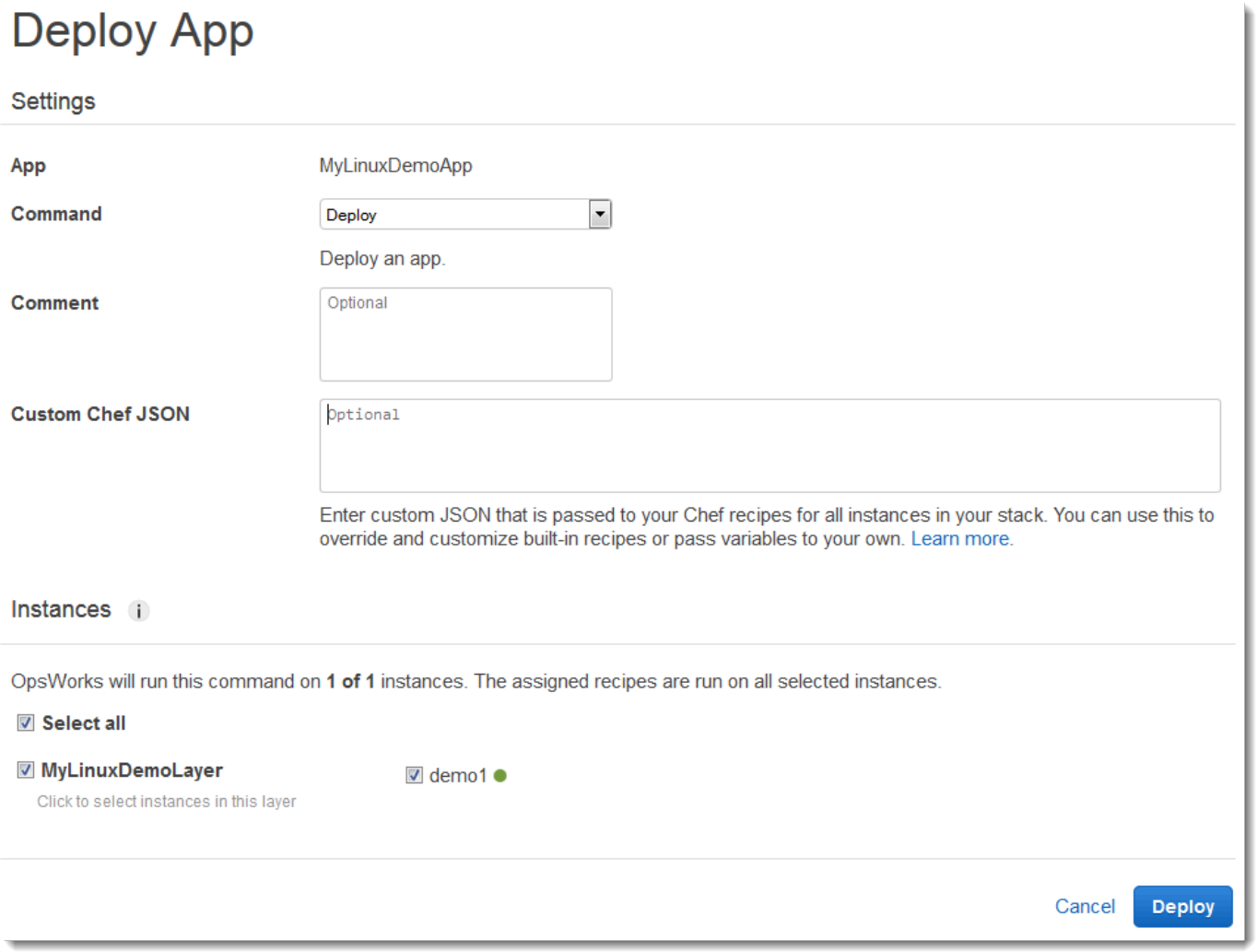


3. 在 Deploy App 页面上，对以下项目保留默认值：

   - Command (Deploy)

- Comment (空白)
- Settings、Advanced、Custom Chef JSON (空白)
- 实例，高级（选中全选，选中 MyLinuxDemoLayer，选中 demo1）

4. 您的结果必须与以下屏幕截图匹配：



5. 选择部署。将显示 “部署 MyLinuxDemoApp -部署” 页面。Status 从 running 变为 successful。在 demo1 旁边会显示一个旋转圆圈，然后又变成绿色的对勾。注意，此过程可能耗时数分钟。在没有看到 Status 变为 successful 以及绿色的对勾图标之前，不要继续。
6. 您的结果必须与以下屏幕截图匹配，当然 Created at、Completed at、Duration 和 User 除外。如果 status 为 failed，则排查问题，对于 Log，选择 show 以获取有关失败的详细信息：

现在，您已成功地将应用程序部署到实例上。

在下一步中，您将测试实例上部署的应用程序。

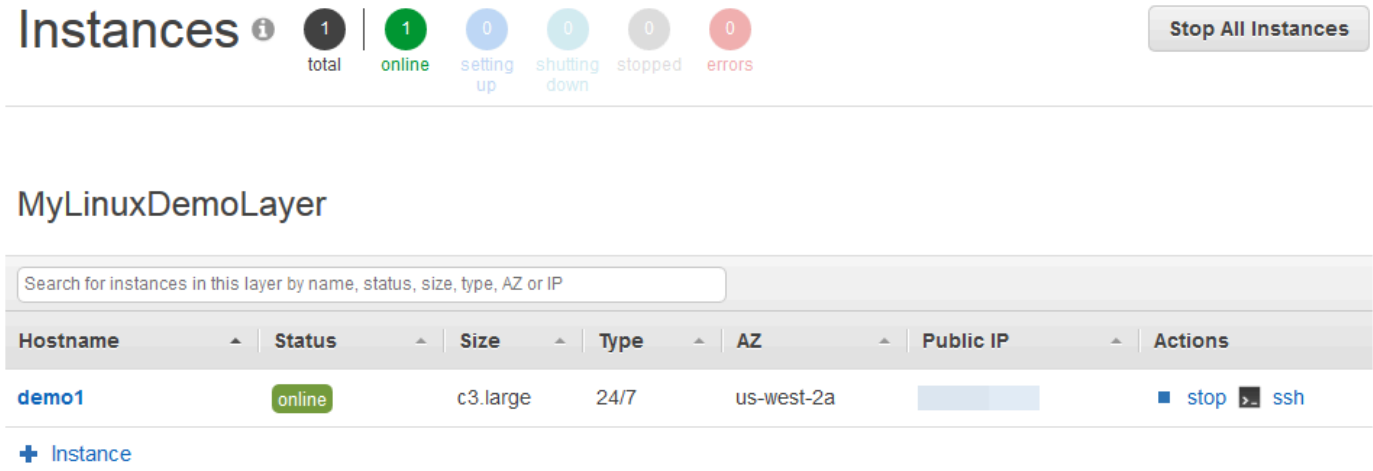
步骤 7：测试实例上部署的应用程序

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

现在，测试实例上部署的应用程序。

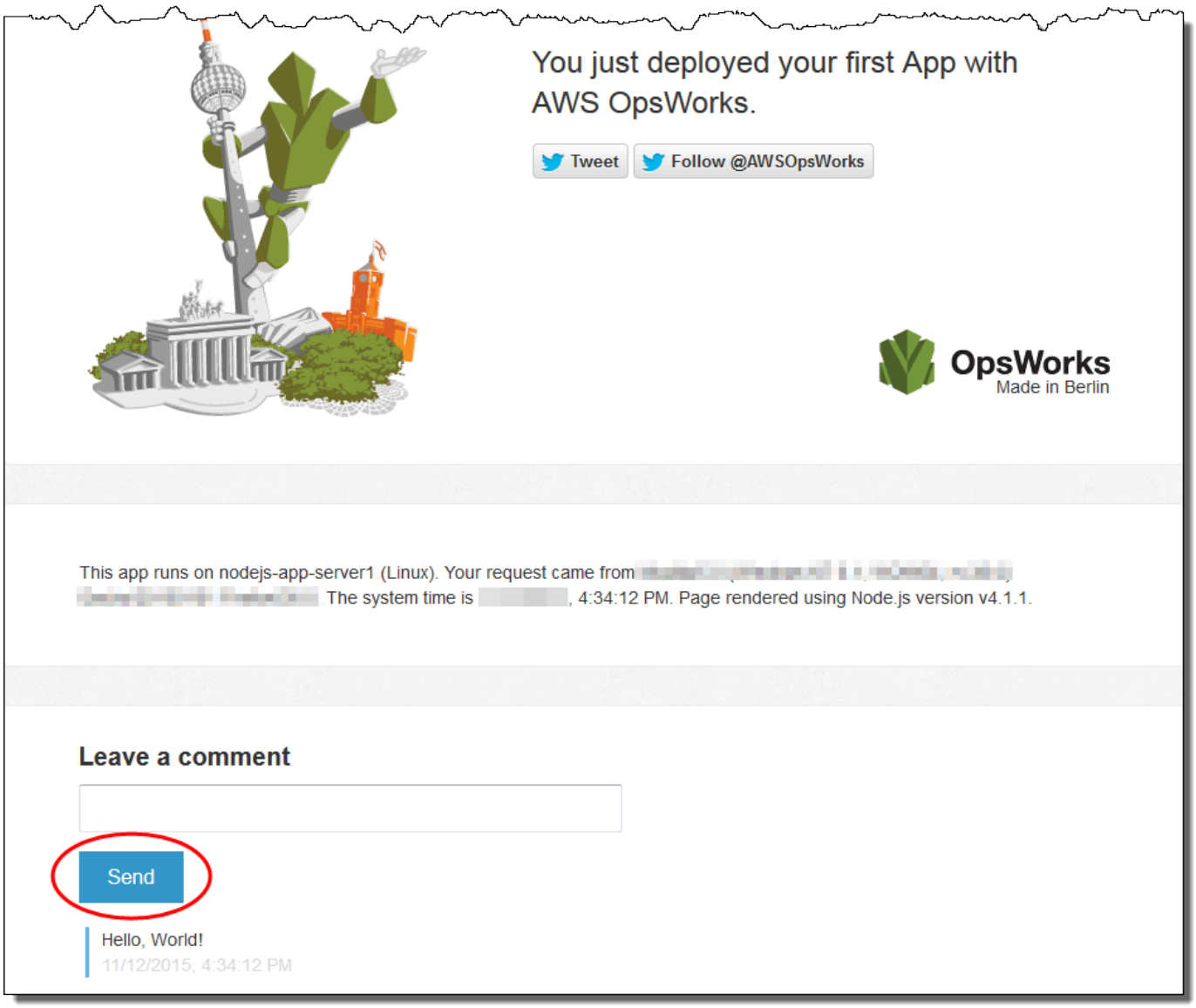
测试实例上的部署

1. 在服务导航窗格中，选择 Instances。这将显示 Instances 页面。
2. 对于 MyLinuxDemoLayerdemo1，对于公有 IP，请选择 IP 地址：



一个新的 Web 浏览器选项卡显示该应用程序。

3. 在祝贺网页上的 Leave a comment 文本框中，键入注释，然后选择 Send 来测试应用程序。该应用程序将您的注释添加到网页中。继续键入注释并根据需要选择 Send：



4. 如果你有 Twitter 账号，请选择 “推文” 或 “关注 @”AWSOpsWorks，然后按照屏幕上的说明在推特上发布有关该应用程序的推文或关注 @ AWSOpsWorks。

现在，您已成功地测试了实例上部署的应用程序。

在下一步中，您可以清理用于本演练的 AWS 资源。下一步是可选的。随着您继续了解 AWS OpsWorks Stacks的更多信息，您可能需要继续使用这些 AWS 资源。但是，保留这些 AWS 资源可能会导致您的 AWS 账户持续收取一些费用。如果你想保留这些 AWS 资源以备后用，现在你已经完成了本演练，你可以直接跳到后续步骤。

步骤 8：(可选) 清除

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

为防止您的 AWS 账户产生额外费用，您可以删除用于本演练的 AWS 资源。这些 AWS 资源包括堆栈 AWS OpsWorks 堆栈和堆栈的组件。（有关更多信息，请参阅 AWS OpsWorks 定价。）但是，随着您继续了解 AWS OpsWorks Stacks的更多信息，您可能需要继续使用这些 AWS 资源。如果你想让这些 AWS 资源保持可用，现在你已经完成了本演练，你可以跳到后续步骤。

为本演练创建的资源中存储的内容可以包含个人可识别信息。如果您不再希望由 AWS存储此信息，请执行本主题中的步骤。

从堆栈中删除应用程序

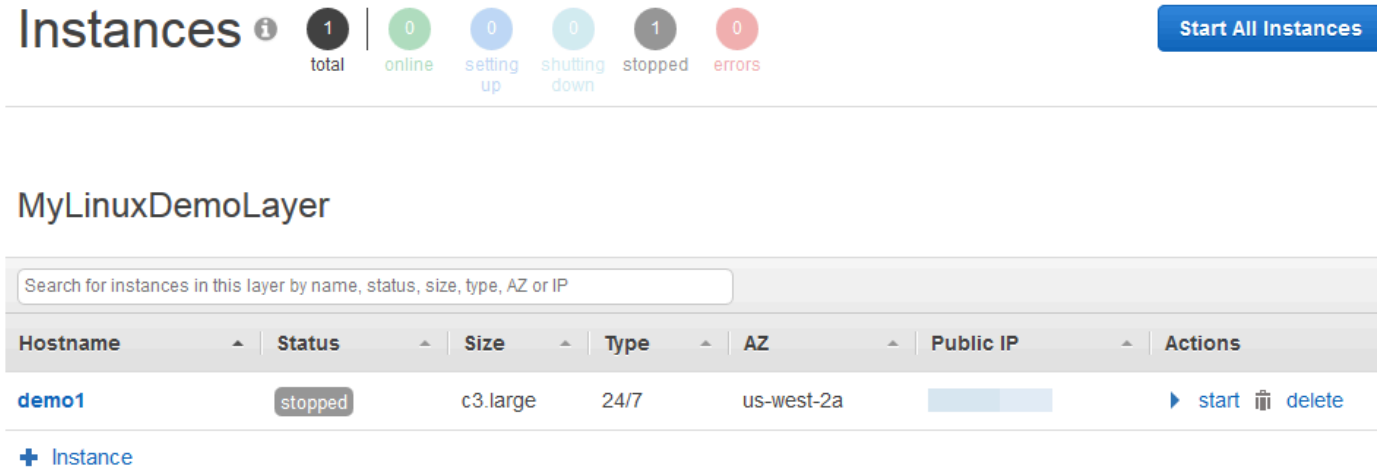
1. 在 AWS OpsWorks Stacks 控制台的服务导航窗格中，选择应用程序。将出现 Apps 页面。
2. 对于 MyLinuxDemoApp，在 “操作” 中选择 “删除”。显示确认消息时，选择 “删除”。 AWS OpsWorks 堆栈会删除应用程序。

删除堆栈的实例

1. 在服务导航窗格中，选择 Instances。这将显示 Instances 页面。
2. 对于 demo1 MyLinuxDemoLayer，对于 “操作”，选择 “停止”。看到确认消息后，选择 Stop。会发生以下情况。

   - Status 从 online 变为 stopping，最终变为 stopped。
   - online 从 1 变为 0。
   - shutting down 从 0 变为 1，最终又变回 0。
   - stopped 最终从 0 变为 1。

   此过程可能耗时数分钟。 AWS OpsWorks 堆栈完成后，将显示以下结果。



3. 对于 Actions，选择 delete。当您看到确认消息时，选择删除。 AWS OpsWorks Stacks 会删除实例并显示 “无实例” 消息。

删除堆栈

1. 在服务导航窗格中，选择 Stack。屏幕上随即显示 MyLinuxDemoStack 页面。
2. 选择 Delete Stack。当您看到确认消息时，选择删除。 AWS OpsWorks Stacks 会删除堆栈并显示OpsWorks 控制面板页面。

或者，如果您不想重复使用用户和 Amazon EC2 密钥对来访问其他 AWS 服务和 EC2 实例，则可以将其删除。有关说明，请参阅：删除 IAM 用户及 Amazon EC2 密钥对和 Linux 实例。

现在，您已完成本演练。有关更多信息，请参阅 后续步骤。

后续步骤

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

现在您已经完成了本演练，可以进一步了解如何使用 AWS OpsWorks Stacks：

- 探索您用于本演练的说明书和应用程序。请参阅了解更多：探索本演练中用到的说明书和了解更多：探索本演练中用到的应用程序。
- 在 Windows 实例中练习使用 AWS OpsWorks 堆栈。请参阅 入门：Windows。
- 通过了解如何创建新堆栈来了解有关堆栈的更多信息。
- 通过编辑图 OpsWorks 层的配置了解有关层的更多信息。
- 通过将实例添加到层了解有关实例的更多信息。
- 通过部署应用程序了解有关应用程序的更多信息。
- 了解有关 说明书和诀窍 的更多信息。
- 创建您自己的说明书。请参阅 入门：说明书。
- 利用安全性和权限了解如何控制对堆栈的访问权限。

了解更多：探索本演练中用到的说明书

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS：Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

本主题介绍了 AWS OpsWorks Stacks 在演练中使用的食谱。

说明书 是一个 Chef 概念。说明书是一些存档文件，其中包含配置信息，如配方、属性值、文件、模板、库、定义和自定义资源。配方 也是一个 Chef 概念。配方是一些说明，用 Ruby 语言语法编写，指定要使用的资源和应用这些资源的顺序。有关更多信息，请转至了解 Chef 网站上的关于说明书和关于配方。

要查看本演练中使用的食谱内容，请opsworks-linux-demo-cookbooks将-nodejs.tar.gz 文件的内容解压缩到本地工作站上的空目录中。(您也可以登录到部署了说明书的实例并浏览 `/var/chef/cookbooks` 目录的内容。)

`cookbooks/nodejs_demo/recipes` 目录中的 `default.rb` 文件是说明书运行其代码的位置：

```
app = search(:aws_opsworks_app).first
app_path = "/srv/#{app['shortname']}"

package "git" do
```

```
  options "--force-yes" if node["platform"] == "ubuntu" && node["platform_version"] ==
 "18.04"
end

application app_path do
  javascript "4"
  environment.update("PORT" => "80")

  git app_path do
    repository app["app_source"]["url"]
    revision app["app_source"]["revision"]
  end

  link "#{app_path}/server.js" do
    to "#{app_path}/index.js"
  end

  npm_install
  npm_start
end
```

下面是该文件执行的操作：

- `search(:aws_opsworks_app).first` 使用 Chef 搜索来查找最终将被部署到实例的应用程序的信息。这些信息包括如应用程序的短名及其源存储库详细信息等设置。由于本演练中只部署了一个应用程序，所以 Chef 搜索从实例上 `aws_opsworks_app` 搜索索引内的第一个信息项获得这些设置。每当启动实例时， AWS OpsWorks Stacks 都会将这些信息和其他相关信息作为一组数据袋存储在实例本身上，然后您可以通过 Chef 搜索获得数据包中的内容。尽管您可以将这些设置硬编码到此配方中，但使用数据包和 Chef 仍不失为一种更可靠的方式。有关数据包的更多信息，请参阅 AWS OpsWorks 堆栈数据包参考。另请参阅了解 Chef 网站上的关于数据包。有关 Chef 搜索的更多信息，请转至了解 Chef 网站上的关于搜索。
- `package` 资源将 Git 安装在实例上。
- `application` 资源描述和部署 Web 应用程序：
  - `javascript`是要安装的 JavaScript 运行时版本。
  - `environment` 设置环境变量。
  - `git` 从指定存储库和分支获取源代码。
  - `app_path` 是要将存储库克隆到的目标路径。如果实例上不存在该路径，则 AWS OpsWorks Stacks 会创建该路径。
  - `link` 创建一个符号链接。

- `npm_install` 安装 Node Package Manager，这是 Node.js 的默认软件包管理器。
- `npm_start` 运行 Node.js。

尽管 AWS OpsWorks Stacks 创建了用于本演练的食谱，但你可以创建自己的食谱。要了解如何操作，请参阅 入门：说明书。另外，您也可以转至了解 Chef 网站上的关于说明书、关于配方和了解 Ubuntu 上的 Chef 基础，以及 Chef 入门网站上使用 Chef 的第一步中“我们的第一个 Chef 说明书”一节。

了解更多：探索本演练中用到的应用程序

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS：Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

本主题介绍了 AWS OpsWorks Stacks 在本演练中部署到实例的应用程序。

要查看应用程序的源代码，请将opsworks-windows-demo-nodejs GitHub 存储库的内容解压缩到本地工作站上的空目录中。您也可以登录到部署了说明书的实例并浏览 `/srv/mylinuxdemoapp` 目录的内容。

`index.js` 文件包含该应用程序最重要的代码：

```
var express = require('express');
var app = express();
var path = require('path');
var os = require('os');
var bodyParser = require('body-parser');
var fs = require('fs');

var add_comment = function(comment) {
  var comments = get_comments();
  comments.push({"date": new Date(), "text": comment});
  fs.writeFileSync('./comments.json', JSON.stringify(comments));
};

var get_comments = function() {
  var comments;
```

```
  if (fs.existsSync('./comments.json')) {
    comments = fs.readFileSync('./comments.json');
    comments = JSON.parse(comments);
  } else {
    comments = [];
  }
  return comments;
};

app.use(function log (req, res, next) {
  console.log([req.method, req.url].join(' '));
  next();
});
app.use(express.static('public'));
app.use(bodyParser.urlencoded({ extended: false }))

app.set('view engine', 'jade');
app.get('/', function(req, res) {
  var comments = get_comments();
  res.render("index",
    { agent: req.headers['user-agent'],
      hostname: os.hostname(),
      nodeversion: process.version,
      time: new Date(),
      admin: (process.env.APP_ADMIN_EMAIL || "admin@unconfigured-value.com" ),
      comments: get_comments()
    });
});

app.post('/', function(req, res) {
  var comment = req.body.comment;
  if (comment) {
    add_comment(comment);
    console.log("Got comment: " + comment);
  }
  res.redirect("/#form-section");
});

var server = app.listen(process.env.PORT || 3000, function() {
  console.log('Listening on %s', process.env.PORT);
});
```

下面是该文件执行的操作：

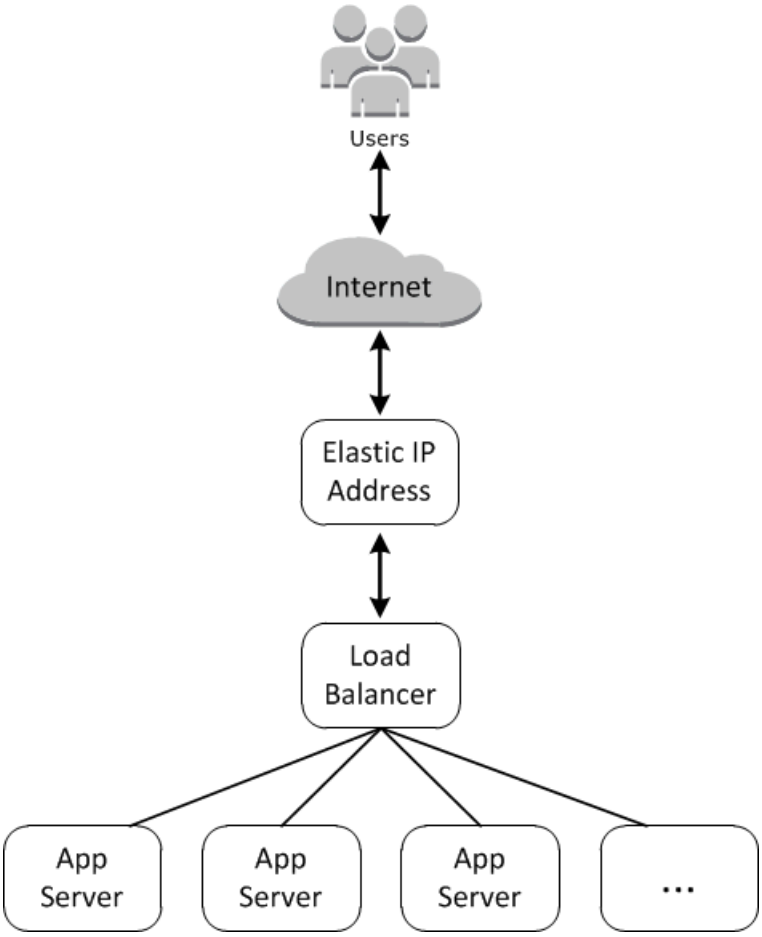
- `require` 加载模块，这些模块中包含此 Web 应用程序按预期运行所需要的一些相关代码。
- `add_comment` 和 `get_comments` 函数将信息写入 `comments.json` 文件并从中读取信息。
- 有关 `app.get`、`app.listen`、`app.post`、`app.set` 和 `app.use` 的信息，请参阅 Express API 参考。

要了解如何创建和打包应用程序以便部署，请参阅应用程序源。

Windows 堆栈入门

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

基于云的应用程序通常需要一组相关的资源 (应用程序服务器、数据库服务器等等)，且必须集体创建并管理这些资源。此实例集合称为堆栈。简单的应用程序堆栈可能如下所示。



基础架构包含以下内容：

- 用于接收用户请求的弹性 IP 地址。
- 用于在应用程序服务器间均匀分发传入请求的负载均衡器。
- 一组应用程序服务器实例，数量必须足以处理流量。

此外，您通常需要通过某种方法来将应用程序分发到应用程序服务器、管理用户权限等。

AWS OpsWorks Stacks 提供了一种简单明了的方式来创建和管理堆栈及其关联的应用程序和资源。本章通过在图表中向您展示创建应用程序服务器堆栈的过程，来介绍 AWS OpsWorks Stacks 的基础知识，以及它的一些更加复杂的功能。它使用 AWS OpsWorks Stacks 易于遵循的增量开发模型：设置一个基本堆栈，并在其正常运行后添加组件，直到你得到一个功能齐全的实现。

- 步骤 1：完成前提条件演示如何设置以开始演练。
- 步骤 2：创建基本应用程序服务器堆栈 演示如何创建基本堆栈以支持互联网信息服务器 (IIS, Internet Information Service) 并将应用程序部署到服务器。

- 步骤 3：扩展 IISExample演示如何通过添加多台应用程序服务器、一个用于分发传入流量的负载均衡器和一个用于接收传入请求的弹性 IP 地址来扩展堆栈以处理增加的负载。

主题

步骤 1：完成前提条件

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS：Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

完成以下设置步骤，然后您才能开始演练。这些设置步骤包括注册 AWS 帐户、创建管理用户以及为 AWS OpsWorks Stacks 分配访问权限。

如果您已完成入门：示例或入门：Linux 演练，则您已满足本演练的先决条件，可以向前跳至步骤 2：创建基本应用程序服务器堆栈。

主题

注册获取 AWS 账户

如果您没有 AWS 账户，请完成以下步骤来创建一个。

要注册 AWS 账户

1. 打开 https://portal.aws.amazon.com/billing/signup。
2. 按照屏幕上的说明进行操作。

   在注册时，将接到一通电话，要求使用电话键盘输入一个验证码。

   当您注册时 AWS 账户，就会创建AWS 账户根用户一个。根用户有权访问该账户中的所有 AWS 服务 和资源。作为安全最佳实践，请为用户分配管理访问权限，并且只使用根用户来执行需要根用户访问权限的任务。

AWS 注册过程完成后会向您发送一封确认电子邮件。在任何时候，您都可以通过转至 https://aws.amazon.com/ 并选择我的账户来查看当前的账户活动并管理您的账户。

创建具有管理访问权限的用户

注册后，请保护您的安全 AWS 账户 AWS 账户根用户 AWS IAM Identity Center，启用并创建管理用户，这样您就不会使用 root 用户执行日常任务。

保护你的 AWS 账户根用户

1. 选择 Root 用户并输入您的 AWS 账户 电子邮件地址，以账户所有者的身份登录。AWS Management Console在下一页上，输入您的密码。

   要获取使用根用户登录方面的帮助，请参阅 《AWS 登录 用户指南》中的以根用户身份登录。
2. 为您的根用户启用多重身份验证 (MFA)。

   有关说明，请参阅 I A M 用户指南中的为 AWS 账户 根用户启用虚拟 MFA 设备（控制台）。

创建具有管理访问权限的用户

1. 启用 IAM Identity Center。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的启用 AWS IAM Identity Center。
2. 在 IAM Identity Center 中，为用户授予管理访问权限。

   有关使用 IAM Identity Center 目录 作为身份源的教程，请参阅《用户指南》 IAM Identity Center 目录中的使用默认设置配置AWS IAM Identity Center 用户访问权限。

以具有管理访问权限的用户身份登录

- 要使用您的 IAM Identity Center 用户身份登录，请使用您在创建 IAM Identity Center 用户时发送到您的电子邮件地址的登录网址。

  有关使用 IAM Identity Center 用户登录的帮助，请参阅AWS 登录 用户指南中的登录 AWS 访问门户。

将访问权限分配给其他用户

1. 在 IAM Identity Center 中，创建一个权限集，该权限集遵循应用最低权限的最佳做法。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的创建权限集。
2. 将用户分配到一个组，然后为该组分配单点登录访问权限。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的添加组。

分配服务访问权限

通过向您的角色或用户添加和权限，启用对 AWS OpsWorks Stacks 服务（以及 Stacks 所依赖的`AWSOpsWorks_FullAccess`相关服务）的`AmazonS3FullAccess`访问权限。 AWS OpsWorks

有关添加权限的更多信息，请参阅：添加 IAM 身份权限（控制台）。

确保 AWS OpsWorks Stacks 用户已添加到您的域中

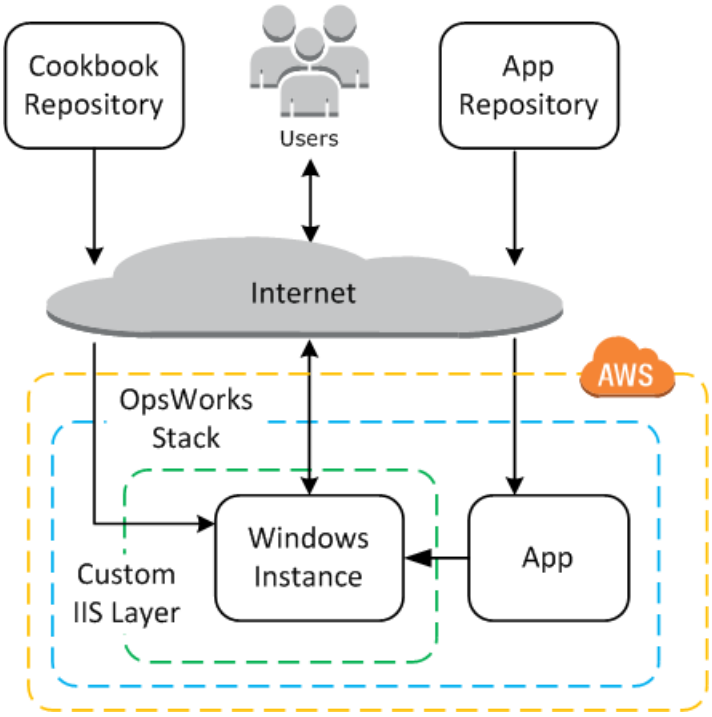
在 Chef 12.2 堆栈中，随附的 `aws_opsworks_users` 说明书创建了对基于 Windows 的实例拥有 SSH 和远程桌面协议 (RDP) 访问权限的用户。当你将堆栈中的 Windows 实例加入 Active Directory 域时，如果 Active Directory 中不存在 AWS OpsWorks 堆栈用户，则此食谱运行可能会失败。如果在 Active Directory 中未识别到用户，则在将实例加入域后重启时，实例会进入 `setup failed` 状态。对于加入域的 Windows 实例，在用户权限页面向 AWS OpsWorks Stacks 用户授予 SSH/RDP 访问权限是不足够的。

在将 Chef 12.2 堆栈中的 Windows 实例加入 Active Directory 域之前，请确保基于 Windows 的 AWS OpsWorks 堆栈的所有堆栈用户都是该域的成员。实现此目的的最佳方法是在创建基于 Windows 的堆栈之前使用 IAM 配置联合身份，然后将 AWS OpsWorks 联合身份用户导入堆栈，然后再将堆栈中的实例加入到域中。有关如何执行此操作的更多信息，请参阅 AWS 安全博客中的使用 Windows Active Directory、ADFS 和 SAML 2.0 启用 AWS 的联合身份验证和 IAM 用户指南中的联合现有用户。

步骤 2：创建基本应用程序服务器堆栈

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

一个基本的应用程序服务器堆栈包含一个具有公有 IP 地址的应用程序服务器实例以接收用户请求。应用程序代码和任何相关文件均存储在单独的存储库中，并从该位置部署到服务器。下图阐明了此类堆栈。



该堆栈拥有以下组件：

- 一个层，它表示一组实例并指定这些实例的配置方式。

  在此示例中，层表示一组 IIS 实例。
- 一个实例，它表示一个 Amazon EC2 实例。

  在此示例中，层配置单个实例以运行 IIS，但层可具有任意数量的实例。
- 一个应用程序，包含在实例上安装应用程序所需的信息。

- 一个说明书，包含支持自定义 IIS 层的自定义 Chef 配方。说明书和应用程序代码存储在远程存储库中，如 Amazon S3 存储桶或 Git 存储库中的存档文件。

以下各节介绍如何使用 AWS OpsWorks Stacks 控制台创建堆栈和部署应用程序。

主题

步骤 2.1：创建堆栈

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS：Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

您可以通过创建 AWS OpsWorks 堆栈来启动 Stacks 项目，该堆栈充当您的实例和其他资源的容器。堆栈配置指定一些由堆栈的所有实例共享的基本设置，如 Amazon Web Services Region 和默认操作系统。

创建新堆栈

1. 添加堆栈

   如果您尚未执行此操作，请登录 AWS OpsWorks Stacks 控制台。

   - 如果该账户没有现有堆栈，您将看到 “欢迎使用 AWS OpsWorks” 页面；选择添加您的第一个堆栈。
   - 否则，您将看到 AWS OpsWorks Stacks 控制面板，其中列出了您账户的堆栈；选择添加堆栈。

2. 配置堆栈

   在 Add Stack 页面上，选择 Chef 12 stack 并指定以下设置：

   堆栈名称

   为您的堆栈输入一个名称，可以包含字母数字字符 (a–z、A–Z、0-9) 和连字符 (-)。此演练的示例堆栈名为 **IISWalkthrough**。

   区域

   选择 美国西部（俄勒冈州） 作为堆栈的区域。

   虽然您可在任何区域中创建堆栈，但在教程中，我们建议您在美国西部（俄勒冈州）中创建堆栈。

   默认操作系统

   选择 Windows，然后指定 Microsoft Windows Server 2022 Base，这是默认设置。

   使用自定义 Chef 说明书

   在本演练中，为此选项指定 No。

3. 选择 Advanced 以确认您具有 IAM 角色并且已选择 IAM 实例配置文件。

   IAM 角色

   指定堆栈的 IAM (AWS Identity and Access Management) 角色。 AWS OpsWorks 堆栈需要访问其他 AWS 服务才能执行诸如创建和管理 Amazon EC2 实例之类的任务。IAM 角色指定 AWS OpsWorks Stacks 在代表您使用其他 AWS 服务时担任的角色。有关更多信息，请参阅 允许 AWS OpsWorks Stacks 代表你行事。

   - 如果您的账户已有 AWS OpsWorks Stacks IAM 角色，则可以从列表中选择该角色。

     如果角色由 AWS OpsWorks Stacks 创建，则该角色将被命名`aws-opsworks-service-role`。

   - 否则，请选择 “新建 IAM 角色” 以指示 AWS OpsWorks Stacks 为您创建具有正确权限的新角色。

     注意：如果您具有 AWS OpsWorks Stacks 完全访问权限，则创建新角色需要额外几个 IAM 权限。有关更多信息，请参阅 示例策略。

4. 接受其他设置的默认值并选择 Add Stack。有关各种堆栈设置的更多信息，请参阅创建新堆栈。

步骤 2.2：授予 RDP 访问权

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

现在您已创建一个堆栈，您将创建一个层并向该层添加 Windows 实例。但是，您必须先配置堆栈以允许您使用 RDP 连接到自定义层的实例，然后才能执行上述操作。为此，您必须执行以下操作：

- 将入站规则添加到控制 RDP 访问权的安全组。
- 设置 AWS OpsWorks 此堆栈的堆栈权限以允许 RDP 访问。

当您在某个区域中创建第一个堆栈时， AWS OpsWorks Stacks 会创建一组安全组。它们包括一个名为类似的名字`AWS-OpsWorks-RDP-Server`， AWS OpsWorks 堆栈将其连接到所有 Windows 实例以允许 RDP 访问。但是，默认情况下，此安全组没有任何规则，因此您必须添加入站规则以允许 RDP 访问您的实例。

允许 RDP 访问

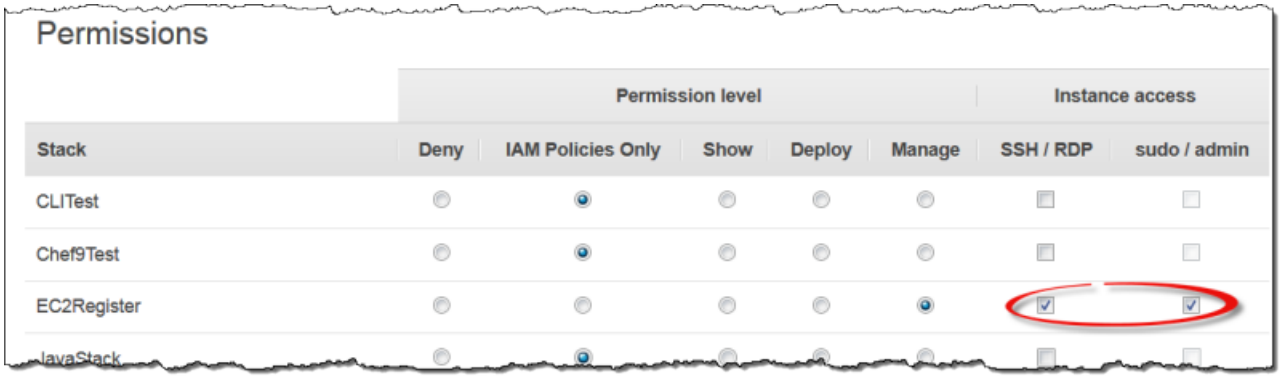
1. 打开 Amazon S3 控制台，将其设置为堆栈的区域，然后从导航窗格中选择安全组。
2. 选择 AWS-OpsWorks RDP-Server，选择 “入站” 选项卡，然后选择 “编辑”。
3. 选择 Add Rule，然后指定以下设置：
   - 类型：RDP。
   - 源 -允许的源 IP 地址。

     通常您会允许来自您的 IP 地址或指定 IP 地址范围（一般是公司的 IP 地址范围）的入站 RDP 请求。就学习而言，通常指定 0.0.0.0/0 已足够，这将允许来自任何 IP 地址的 RDP 访问。

此安全组允许实例接收 RDP 连接请求，但这只是此安全组所发挥的一半作用。普通用户将使用 AWS OpsWorks Stacks 提供的密码登录实例。要让 AWS OpsWorks Stacks 生成该密码，您必须明确授权用户访问 RDP。

为用户授予 RDP 访问权

1. 在 AWS OpsWorks 堆栈仪表板中，选择 iis WalkThrough 堆栈。
2. 在堆栈的导航窗格中，选择 Permissions。
3. 在“Permissions”页面上，选择 Edit。
4. 在用户列表中，对于要为其授予必需权限的用户，选中该用户的 SSH/RDP 所对应的复选框。如果您希望此用户还具有管理员权限，同时选择 sudo/admin。



5. 选择保存。

随后，此用户可获取密码并使用密码登录实例，如下文所述。

> **Note**
>
> 您也可以管理员身份登录。有关更多信息，请参阅 作为管理员登录。

步骤 2.3：实施自定义说明书

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

尽管堆栈基本上是实例的容器，但您不能直接将实例添加到堆栈。您可添加一个或多个层 (每个层代表一组相关实例)，然后将实例添加到层。

层基本上是一个蓝图， AWS OpsWorks Stacks 使用它来创建一组具有相同配置的 Amazon EC2 实例。实例从基本版的操作系统开始，实例的层在实例上执行各种任务来实施此蓝图，这些任务可能包括：

- 创建目录和文件
- 管理用户
- 安装和配置软件
- 启动或停止服务器
- 部署应用程序代码和相关文件。

层通过运行 简称“配方” 在实例上执行任务。配方是使用 Chef 的域专用语言 (DSL) 描述实例的最终状态的 Ruby 应用程序。在 AWS OpsWorks Stacks 中，每个配方通常会分配给该层的生命周期事件之一：设置、配置、部署、取消部署和关闭。当实例上发生生命周期事件时， AWS OpsWorks Stacks 会运行该事件的配方来执行相应的任务。例如，Setup 事件发生在实例完成启动之后。 AWS OpsWorks 然后，Stacks 会运行安装配方，这些配方通常会执行诸如安装和配置服务器软件以及启动相关服务之类的任务。

AWS OpsWorks Stacks 为每个图层提供了一组用于执行标准任务的内置配方。您可扩展层的功能，方式为实现自定义配方来执行其他任务并将配方分配给层的生命周期事件。Windows 堆栈支持自定义层，这些层具有一组仅执行几项基本任务的最少量的配方。要向您的 Windows 实例添加功能，您必须实现自定义配方来安装软件、部署应用程序等。本主题介绍如何创建一个简单的自定义层来支持 IIS 实例。

主题

- 说明书和配方的快速简介
- 实施配方以安装和启动 IIS
- 启用自定义说明书

说明书和配方的快速简介

配方定义实例预期状态的一个或多个方面：应包含哪些目录，应安装哪些软件程序包、应部署哪些应用程序等。配方包装在说明书 中，说明书通常包含一个或多个相关配方以及关联文件 (例如，用于创建配置文件的模板)。

本主题对配方做了最基本的介绍，足以为您演示如何实施说明书以支持简单的自定义 IIS 层。有关说明书的更具体的介绍，请参阅说明书和诀窍。有关实施说明书的详细指导性介绍 (包括一些 Windows 特定的主题)，请参阅说明书 101。

从技术上说，Chef 配方是 Ruby 应用程序，但大部分 (而非所有) 代码位于 Chef DSL 中。DSL 主要由一组资源 组成，可用于声明地指定实例状态的一个方面。例如，directory 资源定义要添加到系统的目录。以下示例定义一个拥有完整控制权的 C:\data 目录，此目录属于指定用户并且不会从父目录继承权限。

```
directory 'C:\data' do
  rights :full_control, 'WORKGROUP\username'
  inherits false
  action :create
end
```

当 Chef 运行配方时，它通过将数据传递给关联的提供程序 (一个处理修改实例状态的详细信息的 Ruby 对象) 来执行每项资源。在此示例中，提供程序使用指定配置创建一个新目录。

自定义 IIS 层的自定义说明书必须执行下列任务：

- 安装 IIS 功能并启动此服务。

  通常，您在设置期间、实例完成启动后立即执行此任务。
- 将应用程序部署到实例 (此示例中为一个简单的 HTML 页面)。

  您通常在设置期间执行此任务。但是，应用程序通常需要定期更新，因此您还需要在实例处于联机状态时部署更新。

您可让单个配方执行所有这些任务。但是，首选方法是将单独的配方用于设置和部署任务。这样一来，无需运行设置代码，也可随时部署应用程序更新。下面介绍了如何设置说明书以支持自定义 IIS 层。后续主题将演示如何实施配方。

开始使用

1. 在您的工作站上的方便位置创建一个名为 iis-cookbook 的目录。
2. 将包含以下内容的 metadata.rb 文件添加到 iis-cookbook。

   ```
   name "iis-cookbook"
   version "0.1.0"
   ```

此示例使用最小 metadata.rb。有关如何使用此文件的更多信息，请参阅 metadata.rb。

3. 将 recipes 目录添加 iis-cookbook 中。

   此目录必须命名为 recipes，它包含说明书的配方。

通常，说明书可以包含各种其他目录。例如，如果配方使用模板创建配置文件，则此模板通常位于 templates\default 目录中。此示例的说明书完全由配方构成，因此无需其他目录。此外，此示例使用单个说明书，但您可使用所需数量的说明书；对于复杂项目，通常使用多个说明书更可取。例如，对于设置和部署任务，您可具有单独的说明书。有关更多说明书示例，请参阅说明书和诀窍。

实施配方以安装和启动 IIS

IIS 是一项 Windows 功能，它是可安装在 Windows Server 上的可选系统组件之一。您可让配方通过以下任一方式安装 IIS：

- 使用 powershell_script 资源运行 Install-WindowsFeature cmdlet。
- 使用 Chef windows 说明书 windows_feature 资源。

  windows 说明书包含一组资源，该资源的提供程序使用部署映像服务和管理 (DISM) 在 Windows 实例上执行各种任务（包括功能安装）。

> **Note**
>
> powershell_script 位于 Windows 配方的最有用资源之间。通过运行 PowerShell 脚本或 cmdlet，您可以使用它在实例上执行各种任务。它尤其适用于不受 Chef 资源支持的任务。

此示例运行 PowerShell 脚本来安装和启动 Web 服务器 (IIS)。下文将介绍 windows 说明书。有关如何使用 windows_feature 安装 IIS 的示例，请参阅安装 Windows 功能：IIS。

将包含下列内容的名为 install.rb 的配方添加到说明书的 recipes 目录。

```
powershell_script 'Install IIS' do
  code 'Install-WindowsFeature Web-Server'
  not_if "(Get-WindowsFeature -Name Web-Server).Installed"
end

service 'w3svc' do
```

```
  action [:start, :enable]
end
```

此配方包含两种资源。

powershell_script

powershell_script运行指定的 PowerShell脚本或 cmdlet。此示例具有下列属性设置：

- code— 要运行的 PowerShell cmdlet。

  此示例运行一个用于安装 Web Server (IIS) 的 Install-WindowsFeature cmdlet。通常，code 属性可以包含任意数量的行，因此您可运行所需数量的 cmdlet。

- not-if：一个 guard 属性，可确保配方仅在 IIS 未安装时安装它。

  您通常希望配方是幂等 的，因此它们不会浪费时间执行同一任务多次。

每项资源均有一个操作，它指定提供程序要采取的操作。此示例没有明确的操作，因此提供程序采用默认:run操作，即运行指定的 PowerShell 脚本。有关更多信息，请参阅 运行 Windows PowerShell 脚本。

service

一个 service 管理一项服务 (在此示例中为 Web Server IIS 服务 (W3SVC))。此示例使用默认属性并指定 :start 和 :enable 这两项操作来启动和启用 IIS。

> **Note**
>
> 如果您要安装使用程序包安装程序的软件 (如 MSI)，可使用 windows_package 资源。有关更多信息，请参阅 安装程序包。

启用自定义说明书

AWS OpsWorks Stacks 在每个实例上运行本地缓存中的配方。要运行您的自定义配方，您必须执行以下操作：

- 将说明书存储在远程存储库中。

  AWS OpsWorks Stacks 将食谱从此存储库下载到每个实例的本地缓存中。

- 编辑堆栈以启用自定义说明书。

  默认情况下，禁用自定义说明书，因此您必须为堆栈启用自定义说明书并提供存储库 URL 和相关信息。

AWS OpsWorks Stacks 支持自定义食谱的 S3 存档和 Git 存储库；此示例使用 S3 存档。有关更多信息，请参阅 说明书存储库。

使用 S3 存档

1. 创建 `iis-cookbook` 目录的 `.zip` 存档。

   AWS OpsWorks 堆栈还支持 Windows 堆栈的`.tgz`（gzip 压缩的 tar）存档。

2. 将存档上传到美国西部（北加利福尼亚）区域中的 S3 存储桶并公开此文件。您还可使用私有 S3 存档，但公用存档对此示例来说已够用，并且使用起来更简单一些。

   a. 登录 AWS Management Console 并打开亚马逊 S3 控制台，网址为 https://console.aws.amazon.com/s3/。

   b. 如果您在 `us-west-1` 中还没有存储桶，请选择创建存储桶，在美国西部（北加利福尼亚）区域中创建存储桶。

   c. 在存储桶列表中，选择要将文件上传到的存储桶的名称，然后选择 Upload (上传)。

   d. 选择 Add Files。

   e. 选择要上传的存档文件，然后选择 Open (打开)。

   f. 在 Upload - Select Files and Folders (上传 - 选择文件和文件夹) 对话框的底部，选择 Set Details (设置详细信息)。

   g. 在 Set Details 对话框的底部，选择 Set Permissions。

   h. 在 Set Permissions 对话框中，选择 Make everything public。

   i. 在 Set Permissions (设置权限) 对话框的底部，选择 Start Upload (开始上传)。上传完成后，`iis-cookbook.zip` 文件显示在您的存储桶中。

   j. 选择存储桶，然后选择该存储桶的 Properties (属性) 选项卡。在 Link (链接) 旁边，记录存档文件的 URL 以供将来使用。

   更多有关将文件上传到 Amazon S3 存储桶的信息，请参阅Amazon S3 控制台用户指南中的如何将文件和文件夹上传到 S3 存储桶？。

> **Important**
>
> 此时，该演练只需您花一点点时间； AWS OpsWorks Stacks 服务本身是免费的。但是，您必须为使用的任何 AWS 资源 (如 Amazon S3 存储) 付费。一旦您上传存档，就会开始产生费用。有关更多信息，请参阅 AWS 定价。

为堆栈启用自定义说明书

1. 在 AWS OpsWorks 堆栈控制台中，选择导航窗格中的堆栈，然后选择右上角的堆栈设置。
2. 在 Settings 页的右上角，选择 Edit。
3. 在 Settings 页面上，将 Use custom Chef cookbooks 设置为 Yes，然后输入以下信息：
   - 存储库类型：S3 存档。
   - 存储库 URL：您之前记录的说明书存档文件的 S3 URL。
4. 选择 Save (保存) 以更新堆栈配置。

AWS OpsWorks Stacks 会在所有新实例上安装您的自定义食谱。请注意， AWS OpsWorks Stacks 不会自动在联机实例上安装或更新自定义说明书。您可手动执行此操作，如下文所述。

步骤 2.4：添加 IIS 层

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

您的说明书具有一个安装和启动 IIS 的配方。这足够用来创建层和确认您具有正在运行的 IIS 实例。之后，您将向层添加应用程序部署功能。

创建层

首先，将层添加到堆栈。然后，通过向相应的生命周期事件分配自定义配方来将此功能添加到层。

将 IIS 层添加到堆栈

1. 在导航窗格中选择 Layers，然后选择 Add a layer。

2. 按如下所示配置层：

   - 名称：**IISExample**
   - 短名称：**iisexample**

     AWS OpsWorks Stacks 使用短名称在内部标识图层。您还可在配方中使用短名称来标识层，但此示例未这样做。您可指定任何短名称，但它只能包含小写字母数字字符和少量标点符号。有关更多信息，请参阅 自定义层。

3. 选择 Add Layer。

如果您此时将实例添加到 IISWalkthrough 并启动它， AWS OpsWorks Stacks 将自动安装说明书，但它不会运行 `install.rb`。在实例处于联机状态后，您可使用“Execute Recipes”堆栈命令来手动运行配方。但是，更好的方法是将配方分配给该层的生命周期事件之一。 AWS OpsWorks 然后，Stacks 会在实例生命周期的适当时刻自动运行配方。

实例完成引导后立即安装并启动 IIS。要执行此操作，请将 `install.rb` 分配到层的 `Setup` 事件。

将配方分配给生命周期事件

1. 在导航窗格中选择 Layers
2. 在 IISExample 层的框中，选择 Recipes。
3. 在右上角，选择 Edit。
4. 在 Custom Chef Recipes (自定义 Chef 配方) 下方的 Setup (设置) 配方框中，键入 **iis-cookbook::install**。

   > **Note**
   >
   > 使用 *cookbook-name*::*recipe-name* 标识配方，其中您可省略配方名称的 `.rb` 后缀。

5. 选择 + 以将配方添加到层。红色 x 将出现在配方旁边以便随后轻松删除。
6. 选择 Save 以保存新配置。自定义设置配方现在应包含 `iis-cookbook::install`。

将实例添加到层并启动它

你可以通过向图层添加一个实例并启动实例来试试这个配方。 AWS OpsWorks 一旦实例完成启动，Stacks 就会自动安装食谱并在安装`install.rb`过程中运行。

将实例添加到层并启动它

1. 在 AWS OpsWorks 堆栈导航窗格中，选择实例。
2. 在 IISExample 层下，选择 Add an instance。
3. 选择合适的大小。t2.micro（或可供您使用的最小大小）对于此示例应已够用。
4. 选择 Add Instance。默认情况下， AWS OpsWorks Stacks 通过在图层的短名称后面附加一个整数来生成实例名称，因此该实例应命名为 iisexample1。
5. 在实例的操作列中选择启动以启动实例。 AWS OpsWorks 然后，堆栈将启动 EC2 实例并运行安装配方对其进行配置。如果该层此时有任何 Deploy 配方， AWS OpsWorks Stacks 将在安装配方完成后运行它们。

   此过程可能需要花费很长时间 (分钟)，期间 Status 列将显示一系列状态。当您进入 online 状态时，设置过程已完成，并且实例已可供使用。

确认 IIS 已安装并且正在运行

您可使用 RDP 连接到实例并验证您的设置配方是否运行正常。

确认 IIS 已安装并且正在运行

1. 在导航窗格中选择 “实例”，然后在 iisexample1 实例的 “操作” 列中选择 rdp。 AWS OpsWorks Stacks 会自动为您生成 RDP 密码，该密码将在指定时间段后过期。
2. 将 Session valid for 设置为 2 小时，然后选择 Generate Password。
3. AWS OpsWorks 为了方便起见，Stacks 还会显示密码以及实例的公有 DNS 名称和用户名。复制全部三项，然后单击 Acknowledge and close。
4. 打开您的 RDP 客户端并使用步骤 3 中的数据连接到实例。
5. 在实例上，打开 Windows 资源管理器并检查 `C:` 驱动器。它应具有由 IIS 安装创建的 `C:\inetpub` 目录。
6. 打开控制面板 Administrative Tools 应用程序，然后打开 Services。列表底部附近应显示 IIS 服务。它命名为 World Wide Web 发布服务，并且状态应为 running。
7. 返回 AWS OpsWorks Stacks 控制台并选择 iisexample1 实例的公有 IP 地址。请务必在 AWS OpsWorks 堆栈中执行此操作，而不是在 Amazon EC2 控制台中执行此操作。此操作将自动向地址发送 HTTP 请求，这应会打开默认 IIS 欢迎页面。

下一个主题讨论如何将应用程序部署到实例 (此示例中为一个简单的静态 HTML 页面)。不过，如果您想休息一下，请选择 iisexample1 实例的操作列中的停止以停止实例并避免产生不必要的费用。您可在准备好继续时重新启动实例。

步骤 2.5：部署应用程序

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS：Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

IIS 安装将为应用程序的代码和相关文件创建一个 `C:\inetpub\wwwroot` 目录。下一步是将应用程序安装在此目录中。在本示例中，您将在 `default.html`中安装静态 HTML 主页 `C:\inetpub\wwwroot`。您可轻松扩展常规方法以处理更复杂的方案，如 ASP.NET 应用程序。

您可在您的说明书中包含应用程序的文件并让 `install.rb` 将这些文件复制到 `C:\inetpub\wwwroot`。有关如何执行此操作的示例，请参阅示例 6：创建文件。但是，这种方法不是很灵活或高效，通常更好的方法是将说明书开发与应用程序开发分隔开。

首选解决方案是实施从存储库 (任何你喜欢的资源库，而不仅仅是说明书存储库) 检索应用程序的代码和相关文件的单独部署配方，并将此配方安装在每个 IIS 服务器实例上。此方法将说明书开发与应用程序开发分隔开，当您需要更新应用程序时，它允许您仅运行部署配方，而不必更新说明书。

本主题演示如何实施将 `default.htm` 部署到 IIS 服务器的简单部署配方。您可随时将此示例扩展到更复杂的应用程序。

主题

- 创建应用程序并将它存储在存储库中
- 实施配方以部署应用程序
- 更新实例的说明书
- 将配方添加到自定义 IIS 层
- 添加应用程序
- 部署应用程序和打开应用程序

创建应用程序并将它存储在存储库中

您可对应用程序使用您更喜欢的任何存储库。为简便起见，此示例将 default.htm 存储在公有 S3 存储桶中。

创建应用程序

1. 在您的工作站上的方便位置创建一个名为 iis-application 的目录。
2. 将包含以下内容的 default.htm 文件添加到 iis-application。

```
<!DOCTYPE html>
<html>
  <head>
    <title>IIS Example</title>
  </head>
  <body>
    <h1>Hello World!</h1>
  </body>
</html>
```

3. 创建 S3 存储桶，将 default.htm 上传到此存储桶，并记录此 URL 供以后使用。为简单起见，请将此文件设置为公用。

> **Note**
>
> 这是一个非常简单的应用程序，但您可扩展基本准则以处理生产级应用程序。
>
> - 对于具有多个文件的更复杂的应用程序，为 iis-application 创建 .zip 存档并将其上传至您的 S3 存储桶通常会更简单。
>
> 随后，您可下载此 .zip 文件并将其内容提取到相应目录。无需下载多个文件，创建一个目录结构等。
> - 对于生产应用程序，您可能希望保持您的文件私有。有关如何让配方从私有 S3 存储桶下载文件的示例，请参阅在 AWS OpsWorks Stacks Windows 实例上使用适用于 Ruby 的 SDK。
> - 您可将应用程序存储在任何适当的存储库中。
>
> 通常，您使用存储库的公用 API 下载应用程序。此示例使用 Amazon S3 API。例如，如果您在上存储应用程序 GitHub，则可以使用 GitHub API。

实施配方以部署应用程序

将名为 deploy.rb 的包含以下内容的配方添加到 iis-cookbook recipes 目录。

```
chef_gem "aws-sdk-s3" do
  compile_time false
  action :install
end

ruby_block "download-object" do
  block do
    require 'aws-sdk-s3'

    #1
    # Aws.config[:ssl_ca_bundle] = 'C:\ProgramData\Git\bin\curl-ca-bundle.crt'
    Aws.use_bundled_cert!

    #2
    query = Chef::Search::Query.new
    app = query.search(:aws_opsworks_app, "type:other").first
    s3region = app[0][:environment][:S3REGION]
    s3bucket = app[0][:environment][:BUCKET]
    s3filename = app[0][:environment][:FILENAME]

    #3
    s3_client = Aws::S3::Client.new(region: s3region)
    s3_client.get_object(bucket: s3bucket,
                         key: s3filename,
                         response_target: 'C:\inetpub\wwwroot\default.htm')
  end
  action :run
end
```

此示例使用 SDK for Ruby v2 下载文件。但是，AWS OpsWorks Stacks 不会在 Windows 实例上安装此 SDK，因此配方从处理该任务的chef_gem资源开始。

> **Note**
>
> chef_gem 资源将 gem 安装到 Chef 的专用 Ruby 版本 (配方使用的版本) 中。如果您要为系统范围的 Ruby 版本安装 gem，请使用 gem_package 资源。

大多数配方都是 `ruby_block` 资源，此资源运行使用 SDK for Ruby 下载 `default.htm` 的 Ruby 代码块。`ruby_block` 中的代码可分为下列各节，这些节对应于代码示例中的编号注释。

1：指定证书捆绑

Amazon S3 使用 SSL，因此您需要相应的证书来从 S3 存储桶下载对象。适用于 Ruby v2 的 SDK 不包含证书包，因此您必须提供一个证书包，然后配置适用于 Ruby 的 SDK 才能使用它。 AWS OpsWorks Stacks 不会直接安装证书包，但会安装 Git，其中包括证书包 (`curl-ca-bundle.crt`)。为方便起见，此示例将 SDK for Ruby 配置为使用 SSL 的 Git 证书捆绑。您也可以安装您自己的捆绑并相应地配置开发工具包。

2：检索存储库数据

要从 Amazon S3 下载对象，您需要 Amazon Web Services Region、存储桶名称和密钥名称。如下文所述，此示例通过将一组环境变量与应用程序关联来提供此信息。部署应用程序时， AWS OpsWorks Stacks 会向实例的节点对象添加一组属性。这些属性实际上是包含应用程序配置 (包括环境变量) 的哈希表。此应用程序的应用程序属性看上去将与以下内容类似 (JSON 格式)。

```
{
  "app_id": "8f71a9b5-de7f-451c-8505-3f35086e5bb3",
  "app_source": {
      "password": null,
      "revision": null,
      "ssh_key": null,
      "type": "other",
      "url": null,
      "user": null
  },
  "attributes": {
      "auto_bundle_on_deploy": true,
      "aws_flow_ruby_settings": {},
      "document_root": null,
      "rails_env": null
  },
  "data_sources": [{"type": "None"}],
  "domains": ["iis_example_app"],
  "enable_ssl": false,
  "environment": {
      "S3REGION": "us-west-2",
      "BUCKET": "windows-example-app",
      "FILENAME": "default.htm"
  },
  "name": "IIS-Example-App",
```

```
  "shortname": "iis_example_app",
  "ssl_configuration": {
      "certificate": null,
      "private_key": null,
      "chain": null
  },
  "type": "other",
  "deploy": true
}
```

应用程序的环境变量存储在 `[:environment]` 属性中。要检索它们，请使用 Chef 搜索查询检索应用程序的哈希表（在 `aws_opsworks_app` 节点下）。此应用程序将定义为 `other` 类型，因此查询将搜索此类型的应用程序。配方将利用此实例上只有一个应用程序的事实，因此相关哈希表为 `app[0]`。为方便起见，配方随后将向变量分配区域、存储桶和文件名。

有关如何使用 Chef 搜索的更多信息，请参阅使用 Chef 搜索获取属性值

3：下载文件

配方的第三部分创建一个 S3 客户端对象并使用其 `get_object` 方法将 `default.htm` 下载到实例的 `C:\inetpub\wwwroot` 目录。

> **Note**
>
> 配方是 Ruby 应用程序，因此 Ruby 代码不一定必须位于 `ruby_block` 中。但是，配方正文中的代码先运行，然后再运行资源。在此示例中，如果您将下载代码放入配方正文，则下载将失败，因为 `chef_gem` 资源尚未安装 SDK for Ruby。在 `chef_gem` 资源安装 SDK for Ruby 后，`ruby_block` 资源中的代码将在资源执行时执行。

更新实例的说明书

AWS OpsWorks Stacks 会自动在新实例上安装自定义食谱。但是，您正在使用现有实例，因此必须手动更新您的说明书。

更新实例的说明书

1. 创建 `iis-cookbook` 的 `.zip` 存档，然后将其传到 S3 存储桶。

   这会覆盖现有说明书，但 URL 保持不变，因此您无需更新堆栈配置。
2. 如果您的实例未处于联机状态，请重新启动它。

3. 在实例处于联机状态后，选择导航窗格中的 Stack，然后选择 Run Command。
4. 对于 Command，选择 Update Custom Cookbooks。此命令将在实例上安装更新后的说明书。
5. 选择 Update Custom Cookbooks。此命令可能需要几分钟才能完成。

将配方添加到自定义 IIS 层

与 `install.rb` 一样，处理部署的首选方式是将 `deploy.rb` 分配给相应的生命周期事件。通常，您将部署配方分配给部署事件，它们统称为部署配方。将配方分配给部署事件不会触发此事件。相反：

- 对于新实例， AWS OpsWorks Stacks 会在安装配方完成后自动运行 Deploy 配方，因此新实例会自动使用当前的应用程序版本。
- 对于联机实例，请使用部署命令手动安装新的或更新的应用程序。

  此命令将在堆栈实例上触发部署事件，该事件将运行部署配方。

将 deploy.rb 分配给层的部署事件

1. 在导航窗格中选择 Layers，然后在 Layer IISExample 下选择 Recipes。
2. 在 Custom Chef Recipes (自定义 Chef 配方) 下，将 **`iis-cookbook::deploy`** 添加到 Deploy (部署) 配方框并选择 + 以将配方添加到层。
3. 选择 Save 以保存新配置。自定义部署配方现在应包含 `iis-cookbook::deploy`。

添加应用程序

最后一项任务是向堆栈中添加一个应用程序，以在 AWS OpsWorks Stacks 环境中代表您的应用程序。应用程序包含元数据 (如应用程序的显示名称) 和从存储库下载应用程序所需的数据。

将应用程序添加到堆栈

1. 在导航窗格中选择 Apps，然后选择 Add an app。
2. 使用以下设置配置应用程序。

   - 名称：**`IIS-Example-App`**
   - 存储库类型：其他
   - 环境变量：添加以下三个环境变量：
     - **`S3REGION`**：存储桶的区域（在本例中为 `us-west-1`）。

- **BUCKET**：存储桶名称，例如 `windows-example-app`。
- **FILENAME**：文件名：**default.htm**。

3. 接受剩余设置的默认值，然后选择 Add App (添加应用程序) 以将应用程序添加到堆栈。

> **Note**
>
> 此示例使用环境变量来提供下载数据。另一种方法是使用 S3 存档存储库类型并提供文件的 URL。 AWS OpsWorks Stacks 会将信息以及可选数据（例如您的 AWS 证书）添加到应用程序的`app_source`属性中。您的部署配方必须从应用程序属性中获取 URL 并进行解析以提取区域、存储桶名称和文件名。

部署应用程序和打开应用程序

AWS OpsWorks Stacks 会自动将应用程序部署到新实例，但不会自动部署到在线实例。由于您的实例已在运行中，因此必须手动部署应用程序。

部署应用程序

1. 选择导航窗格中的 Apps，然后选择应用程序的 Actions 列中的 deploy。
2. Command (命令) 应设置为 Deploy (部署)。在部署应用程序页面右下角，选择部署。此命令可能需要几分钟才能完成。

   在部署完成后，您返回到 Apps (应用程序) 页面。Status (状态) 指示符显示绿色的 successful (成功)，并且应用程序名称旁会出现一个绿色对勾指示部署成功。

> **Note**
>
> Windows 应用程序始终为其他应用程序类型，因此部署应用程序将执行以下操作：
>
> - 将应用程序的数据添加到堆栈配置和部署属性，如上文所述。
> - 在堆栈的实例上触发部署事件，该事件将运行自定义部署配方。

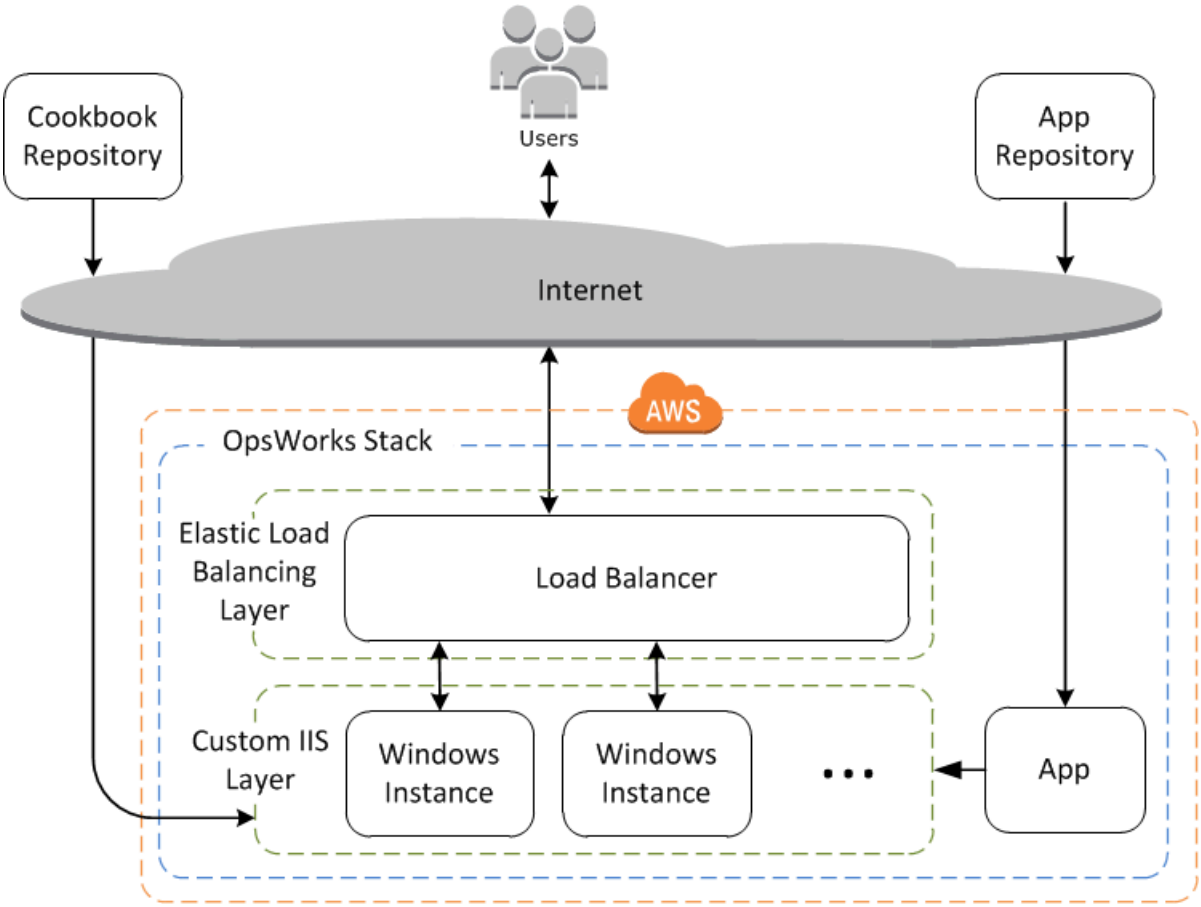
> **Note**
>
> 有关如何对失败的部署或应用程序执行故障排除的更多信息，请参阅调试配方。

应用程序现已安装。您可通过选择导航窗格中的实例，然后选择实例的公有 IP 地址来打开它。这将向实例发送 HTTP 请求，并且浏览器中应显示与以下内容类似的信息。

步骤 3：扩展 IISExample

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

如果您的传入用户请求数量开始接近您可通过单个 t2.micro 实例处理的请求的限制，则您需要增大服务器容量。您可移至更大的实例，但具有限制。一个更灵活的方法是，将实例添加到堆栈，然后将它们放在负载均衡器后面。基本架构看上去与以下内容类似。



除了其他优势，这种方法还比单个大型实例更可靠。

- 如果您的某个实例失败，负载均衡器会将传入请求分发至其余实例，并且您的应用程序将继续运行。
- 如果您将实例放在不同的可用区 (推荐的做法) 中，您的应用程序将继续运行，即使一个可用区遇到问题也是如此。

AWS OpsWorks 堆栈可以轻松扩展堆栈。本部分介绍如何通过将第二个全天候 PHP应用程序服务器实例添加到 IISExample 并将两个实例都置于 Elastic Load Balancing 负载均衡器后来横向扩展堆栈的基础知识。您可以轻松扩展该过程以添加任意数量的全天候实例，也可以使用基于时间的实例让 AWS OpsWorks Stacks 自动扩展您的堆栈。有关更多信息，请参阅 使用基于时间和基于负载的实例管理负载。

添加负载均衡器

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

Elastic Load Balancing 是一种 Amazon Web Service，将传入的应用程序流量自动分配到多个 Amazon EC2 实例。负载均衡器有两种用途。一个显而易见的用途是使应用程序服务器上的负载达到均衡。许多站点更愿意将其应用程序服务器和数据库与直接用户访问分隔开。除了分发流量之外，Elastic Load Balancing 还执行以下操作：

- 检测运行状况不佳的 Amazon EC2 实例。

  它将流量重新路由至其余运行正常的实例，直至运行不正常的实例恢复。
- 自动扩展请求处理容量来响应传入流量。

> **Note**
>
> AWS OpsWorks 堆栈不支持 Application Load Balancer。您只能将 Classic Load Balancer 与 AWS OpsWorks Stacks 配合使用。

尽管 Elastic Load Balancing 通常称为层，但其工作方式与其他内置层略有不同。您无需创建层并向其添加实例，而是使用 Amazon EC2 控制台创建 Elastic Load Balancing 负载均衡器，然后将其连接到现有层之一（通常是应用程序服务器层）。 AWS OpsWorks 然后，堆栈将图层的现有实例注册到该服务，并自动添加任何新实例。以下过程介绍如何添加负载均衡器。

将负载均衡器挂载到自定义 IIS 层

1. 使用 Amazon EC2 控制台为 IISExample 创建新的负载均衡器。有关更多信息，请参阅 Elastic Load Balancing 入门。在运行 Create Load Balancer 向导时，按如下所示配置负载均衡器：

   1：定义负载均衡器

   为负载均衡器分配一个易于识别的名称，例如 IIS-LB，以便更容易在 Stacks 控制台中 AWS OpsWorks 找到它。接受其余设置的默认值，然后选择 Next: Assign Security Groups。

2：分配安全组

如果您的账户支持默认 VPC，则该向导将显示此页以确定负载均衡器的安全组。它不会为 EC2 Classic 显示此页。

对于本演练，指定 default VPC security group，然后选择 Next: Configure Security Settings。

3：配置安全设置

本演练要求您的负载均衡器使用安全侦听器 (即，对其前端连接使用 HTTPS 或 SSL)，因此选择 Next: Configure Health Check 以继续。

4：配置运行状况检查

将 ping 路径设置为 **/**。接受其余设置的默认值，然后选择 Next: Add EC2 Instances。

5：添加 EC2 实例

AWS OpsWorks Stacks 会自动负责向负载均衡器注册实例。选择 Next Add Tags 以继续。

6：添加标签

此示例将不使用标签。选择 Review and Create。

7：复查

检查您的选择并选择 Create，然后选择 Close，这将启动负载均衡器。

2. 如果在启动负载均衡器后，您的账户支持默认 VPC，则必须确保其安全组具有合适的入站规则。默认规则不接受任何入站流量。

   1. 在 Amazon EC2 导航窗格中，选择 Security Groups。
   2. 选择 default VPC security group
   3. 在 Inbound (入站) 选项卡上，选择 Edit (编辑)。
   4. 在本演练中，将 Source 设置为 Anywhere，这将指示负载均衡器接受来自任何 IP 地址的传入流量。
   5. 单击保存。

3. 返回 AWS OpsWorks Stacks 控制台。在 Layers 页面上，选择 Network。
4. 在 Elastic Load Balancing 下，选择您在步骤 1 中创建的 IIS-LB 负载均衡器，然后单击 Save。

将负载均衡器连接到该层后， AWS OpsWorks Stacks 会自动注册该层的当前实例，并在新实例上线时添加这些实例。

5. 在 Layers 页面上，单击负载均衡器的名称以打开其详细信息页面。负载均衡器页上的实例旁的绿色复选标记指示实例已通过运行状况检查。

您现在可通过向负载均衡器发送请求来运行 IIS-Example-App。

通过负载均衡器运行 IIS-Example-App

1. 选择 Layers。IIS-ELB 负载均衡器应作为层列出，“Health”列应有一个绿色实例，这表示它是一个运行正常的实例。
2. 选择负载均衡器的 DNS 名称以运行 IIS-Example-App。它应在负载均衡器的名称下方列出，看上去类似于 `IIS-LB-1802910859.us-west-2.elb.amazonaws.com`。负载均衡器会将请求转发到实例并返回响应，响应应与单击实例的公有 IP 地址时获得的响应完全相同。

您此时只有一个实例，因此负载均衡器实际上不会添加更多实例。但是，您现在可向层添加其他实例。

将实例添加到层

1. 依次选择 Instances 和 + instance，将另一个实例添加到层。
2. 开启实例。

由于它们是新实例， AWS OpsWorks Stacks 会自动安装当前的自定义食谱并在安装过程中部署当前的应用程序版本。当实例上线时， AWS OpsWorks Stacks 会自动将其添加到负载均衡器中，因此您的实例将立即开始处理请求。要验证应用程序是否仍在运行，您可以再次选择负载均衡器的 DNS 名称。

后续步骤

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

本演练将指导您了解设置简单的 Windows 应用程序服务器堆栈的基础知识。以下是有关后续操作的一些建议。

- 如果您想了解更多信息，请入门：说明书提供实现食谱的教程介绍，并包括一些特定于 AWS OpsWorks Stacks 的示例。
- 您可以将 Amazon Relational Database Service (Amazon RDS) 层添加到堆栈中以用作后端数据库服务器。有关如何将应用程序连接到数据库的信息，请参阅使用自定义配方。

堆叠食谱入 AWS OpsWorks 门

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS：Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

生产级 AWS OpsWorks Stacks 堆栈通常需要一些自定义，这通常意味着要实现自定义 Chef 食谱。说明书 是一个程序包文件，其中包含配置信息（如称为配方的指令）。配方 是一组指令 (包含一个或多个指令)，使用 Ruby 语言语法编写，用于指定要使用的资源以及这些资源的使用顺序。资源（在 Chef 中使用）是配置策略的语句。本演练提供了实现 Stacks 的 Chef 食谱的基本 AWS OpsWorks 介绍。要了解有关 Chef、说明书、配方和资源的更多信息，请参阅后续步骤中的链接。

本演练主要介绍如何创建您自己的说明书。您还可以使用可在 Chef Supermarket 等网站上找到的社区提供配方。为了帮助您开始使用社区说明书，我们在本演练后面的部分中包括了关于使用 Chef Supermarket 中的社区说明书的说明。

在开始本演练之前，请完成几个设置步骤。如果您已经完成本章中的任何其他演练 (如入门：示例)，则您就满足了本演练的先决条件，可以跳至开始此演练。否则，请确保完成先决条件，然后再返回到此演练。

主题

- 第 1 步：创建说明书
- 第 2 步：创建堆栈及其组件
- 第 3 步：运行并测试配方
- 第 4 步：更新说明书以安装程序包
- 第 5 步：更新实例上的说明书并运行配方
- 第 6 步：更新说明书以添加用户
- 第 7 步：更新说明书以创建目录

第 1 步：创建说明书

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS：Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

首先，创建一个说明书。此说明书在一开始不会起太多作用，但它可以作为本演练其余部分的基础。

> **Note**
>
> 此步骤演示如何手动创建说明书。使用 Chef 开发工具包 (Chef DK)，您可以在本地工作站上运行 chef generate cookbook 命令，节省创建说明书所需的时间。但是，此命令创建的一些文件夹和文件在本演练中是用不到的。

创建说明书

1. 在您的本地工作站上，创建一个名为 `opsworks_cookbook_demo` 的目录。您可以使用一个不同的名称，但务必在本演练过程中将其替换为 `opsworks_cookbook_demo`。

2. 在 opsworks_cookbook_demo 目录中，使用文本编辑器创建一个名为 metadata.rb 的文件。添加以下代码以指定说明书的名称。有关 metadata.rb 的更多信息，请参阅 Chef 网站上的 metadata.rb。

```
name "opsworks_cookbook_demo"
```

3. 在 opsworks_cookbook_demo 目录中，创建名为 recipes 的子目录。此子目录包含您为本演练的说明书创建的所有配方。

4. 在 recipes 目录中创建名为 default.rb 的文件。此文件包含一个与之同名但没有文件扩展名的配方：default。将以下单独的代码行添加到 default.rb 文件中。此代码是一个一行配方，当配方运行时，将在日志中显示一条简单的消息：

```
Chef::Log.info("********** Hello, World! **********")
```

5. 在终端处或在命令提示符下，使用 tar 命令创建一个名为 opsworks_cookbook_demo.tar.gz 的文件，该文件包含 opsworks_cookbook_demo 目录及其内容。例如：

```
tar -czvf opsworks_cookbook_demo.tar.gz opsworks_cookbook_demo/
```

   您可以使用一个不同的文件名称，但务必在本演练过程中将其替换为 opsworks_cookbook_demo.tar.gz。

   > **Note**
   >
   > 当您在 Windows 上创建 tar 文件时，顶级目录必须是说明书的父目录。本演练已经在 Linux (使用 tar 程序包提供的 tar 命令) 和 Windows (使用 Git Bash 提供的 tar 命令)上经过了测试。使用其他命令或程序来创建压缩的 TAR (.tar.gz) 文件可能不会达到预期效果。

6. 创建一个 S3 存储桶或使用现有存储桶。有关更多信息，请参阅创建存储桶。

7. 将 opsworks_cookbook_demo.tar.gz 文件上传到 S3 存储桶。有关更多信息，请参阅将对象添加到存储桶。

您现在拥有一个您将在本演练的整个过程中都要用到的说明书。

在下一步中，您将创建一个 AWS OpsWorks Stacks 堆栈，稍后将使用该堆栈来上传食谱和运行食谱的食谱。

第 2 步：创建堆栈及其组件

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

创建 AWS OpsWorks Stacks 堆栈及其组件，包括一个层和一个实例。在稍后的步骤中，您需要将您的说明书上传到实例中，然后在该实例上运行说明书的配方。

要创建 堆栈，请执行以下操作：

1. 登录 AWS OpsWorks Stacks 控制台，网址为 https://console.aws.amazon.com/opsworks。
2. 执行以下任一操作 (如果适用)：
   - 如果显示 “欢迎使用 AWS OpsWorks 堆栈” 页面，请选择 “添加您的第一个堆栈” 或 “添加您的第一个 AWS OpsWorks 堆栈”（两个选项的作用相同）。这将显示 Add stack 页面。
   - 如果显示OpsWorks 控制面板页面，请选择添加堆栈。这将显示 Add Stack 页面。
3. 选择 Chef 12 stack。
4. 在 Stack name (堆栈名称) 框中，键入堆栈的名称，例如：**MyCookbooksDemoStack**。您可以键入一个不同的名称，但务必在本演练过程中将其替换为 `MyCookbooksDemoStack`。
5. 对于区域，选择美国西部（俄勒冈）。
6. 对于 VPC，请执行下列操作之一：
   - 如果 VPC 可用，请选择它。有关更多信息，请参阅 在 VPC 中运行堆栈。
   - 否则，请选择 No VPC (无 VPC)。
7. 对于 Use custom Chef cookbooks，选择 Yes。
8. 对于 Repository type，选择 S3 Archive。

   > **Note**
   >
   > 在 入门：Linux 演练中，您选择了 Http Archive。但在这里，请务必选择 S3 Archive。

9. 对于 Repository URL，键入 S3 中的 `opsworks_cookbook_demo.tar.gz` 文件的路径。要获取路径，请在 S3 控制台中选择 `opsworks_cookbook_demo.tar.gz` 文件。在 Properties 窗格

中，复制 Link 字段的值。(它应类似于以下内容：`https://s3.amazonaws.com/opsworks-demo-bucket/opsworks_cookbook_demo.tar.gz`。)

10. 默认情况下，您的 S3 存储桶是私有的，如果是这样，则对于 Access key ID 和 Secret access key，请键入您为本演练使用的 IAM; 用户的访问密钥 ID 和秘密访问密钥。有关更多信息，请参阅编辑对象许可和与其他用户共享对象。
11. 对以下项目保留默认值：
    - Default Availability Zone (us-west-2a)
    - 默认操作系统（Linux 和 Amazon Linux 2016.09）
    - Default SSH key (Do not use a default SSH key)
    - Stack color (深蓝色)
12. 选择 Advanced (高级)。
13. 对于 IAM 角色，执行以下操作之一：
    - 如果 aws-opsworks-service-role 可用，选择该选项。
    - 如果aws-opsworks-service-role不可用，请选择新建 IAM 角色。
14. 对于 Default IAM; 实例配置文件，执行下列操作之一：
    - 如果有 aws-opsworks-ec2 个角色可用，请选择它。
    - 如果aws-opsworks-ec双角色不可用，请选择新建 IAM 实例配置文件。
15. 对以下项目保留默认值：
    - Default root device type (EBS backed)
    - Hostname theme (Layer Dependent)
    - OpsWorks 代理版本（最新版本）
    - Custom Chef JSON (空白)
    - 安全，使用 OpsWorks 安全组（是）
16. 选择添加堆栈。 AWS OpsWorks Stacks 创建堆栈并显示MyCookbooksDemoStack页面。

创建层

1. 在服务导航窗格中，选择 Layers。此时将显示 Layers 页面。
2. 选择 Add a layer。

3. 在OpsWorks选项卡上，在 “名称” 中键入**MyCookbooksDemoLayer**。您可以键入一个不同的名称，但务必在本演练过程中将其替换为 MyCookbooksDemoLayer。
4. 对于 Short name (短名称)，键入 **cookbooks-demo**。您可以键入一个不同的名称，但务必在本演练过程中将其替换为 cookbooks-demo。
5. 选择 “添加图层”。 AWS OpsWorks Stacks 添加图层并显示 “图层” 页面。

创建并启动实例

1. 在服务导航窗格中，选择 Instances。这将显示 Instances 页面。
2. 选择添加一个实例。
3. 在 New 选项卡上，选择 Advanced。
4. 对以下项目保留默认值：
   - Hostname (cookbooks-demo1)
   - Size (c3.large)
   - Subnet (*IP ##* us-west-2a)
   - Scaling type (24/7)
   - SSH key (Do not use a default SSH key)
   - 操作系统 （ Amazon Linux 2016.09)
   - OpsWorks 代理版本（ 继承自堆栈 ）
   - Tenancy (Default - Rely on VPC settings)
   - Root device type (EBS backed)
   - Volume type (General Purpose (SSD))
   - Volume size (8)
5. 选择 Add instance。
6. 对于 MyCookbooksDemoLayercookbooks-demo1，在 “操作” 中，选择 “开始”。请在 Status 变为 online 之后继续。此过程可能需要几分钟的时间，请耐心等待。

您现在获得了一个堆栈、一个层以及一个实例 (系统会自动将说明书从您的 S3 存储桶复制到该实例)。在下一步中，您将在实例上的说明书中运行并测试默认配方。

第 3 步：运行并测试配方

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

从 AWS OpsWorks Stacks 复制到实例的`default`食谱中运行并测试食谱。您可能还记得，这是一个一行配方，当配方运行时，将在日志中显示一条简单的消息。

运行配方

1. 在服务导航窗格中，选择 Stack。屏幕上随即显示 MyCookbooksDemoStack 页面。
2. 选择 Run Command。此时将显示 Run Command 页面。
3. 对于 Command，选择 Execute Recipes。
4. 对于 Recipes to execute (要执行的配方)，请键入 **`opsworks_cookbook_demo::default`**。

   `opsworks_cookbook_demo` 是 `metadata.rb` 文件中定义的说明书的名称。**`default`** 是要运行的配方的名称，也就是，说明书的 `recipes` 子目录中 `default.rb` 文件的名称，不带文件扩展名。
5. 保留以下默认设置：

   - Comment (空白)
   - Advanced，Custom Chef JSON (空白)
   - 实例（选择所有已选中、已选中、MyCookbooksDemoLayer选中 cookbooks- demo1 ）
6. 选择 Execute Recipes。此时将显示 Running command execute_recipes 页面。请在 Status 变为 successful 之后继续。此过程可能需要几分钟的时间，请耐心等待。

检查配方的结果

1. 显示 Running command execute_recipes 页面时，对于 cookbooks-demo1 中的 Log，选择 show。此时将显示 execute_recipes 日志页面。
2. 向下滚动日志并查找类似于以下内容的条目：

```
[2015-11-13T19:14:39+00:00] INFO: ********** Hello, World! **********
```

您已成功运行您的第一个配方！在下一步中，您将通过添加一个可在实例上安装程序包的配方来更新您的说明书。

第 4 步：更新说明书以安装程序包

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

通过添加可在实例上安装包含受欢迎的文本编辑器 GNU Emacs 的程序包的配方，更新您的说明书。

尽管你可以同样轻松地登录实例并安装一次软件包，但编写配方可以让你从 AWS OpsWorks Stacks 运行一次配方，同时在堆栈中的多个实例上安装多个软件包。

更新说明书以安装程序包

1. 返回您的本地工作站，在 `opsworks_cookbook_demo` 目录的 `recipes` 子目录中，创建名为 `install_package.rb` 的包含以下代码的文件：

   ```
   package "Install Emacs" do
     package_name "emacs"
   end
   ```

   此配方将在实例上安装 `emacs` 程序包。(有关更多信息，请转到 package。)

   > **Note**
   >
   > 您可以将配方命名为您喜欢的任何文件名称。每当你想让 AWS OpsWorks Stacks 运行食谱时，一定要指定正确的食谱名称即可。

2. 在终端处或在命令提示符下，使用 tar 命令创建 `opsworks_cookbook_demo.tar.gz` 文件的新版本，该文件包含 `opsworks_cookbook_demo` 目录及其更新的内容。

3. 将更新后的 `opsworks_cookbook_demo.tar.gz` 文件上传到 S3 存储桶。

当您更新实例上的说明书，然后从更新后的说明书中运行新配方时，这个新配方就会运行。下一步骤将介绍如何执行此操作。

完成下一步后，您将能够登录实例，然后在命令提示符下键入 emacs 以启动 GNU Emacs。（有关更多信息，请参阅连接到您的 Linux 实例。）要退出 GNU Emacs，按 Ctrl+X，然后按 Ctrl+C。

> **Important**
>
> 要登录实例，您必须先向 AWS OpsWorks Stacks 提供有关您的 SSH 公钥的信息（您可以使用 ssh-keygen 或 PuttyGen 等工具创建该密钥），然后您必须在堆栈上`MyCookbooksDemoStack`设置权限以允许您的用户登录实例。有关说明，请参阅注册用户的公有 SSH 密钥和使用 SSH 登录。

第 5 步：更新实例上的说明书并运行配方

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

更新实例上的说明书，然后从实例上的更新后的说明书中运行配方。在本演练的其余步骤中，您每次通过添加新的配方来更新说明书时，就需要重复此步骤。

更新实例上的说明书

1. 在服务导航窗格中，选择 Stack。屏幕上随即显示 MyCookbooksDemoStack 页面。
2. 选择 Run Command。此时将显示 Run Command 页面。
3. 对于 Command，选择 Update Custom Cookbooks。
4. 保留以下默认设置：
   - Comment (空白)
   - Advanced，Custom Chef JSON (空白)

- 高级，实例（选择所有选中、选中、cookbooks- demo1 MyCookbooksDemoLayer选中）

5. 选择 Update Custom Cookbooks。此时将显示 Running command update_custom_cookbooks 页面。请在 Status 变为 successful 之后继续。此过程可能需要几分钟的时间，请耐心等待。

运行配方

1. 在服务导航窗格中，选择 Stack。屏幕上随即显示 MyCookbooksDemoStack 页面。
2. 选择 Run Command。此时将显示 Run Command 页面。
3. 对于 Command，选择 Execute Recipes。
4. 对于 Recipes to execute，键入要运行的配方的名称。当您第一次执行此操作时，配方名称为 **`opsworks_cookbook_demo::install_package`**。

> **Note**
>
> 稍后当您重复此步骤时，键入说明书的名称 (**`opsworks_cookbook_demo`**)，后面键入两个冒号 (**`::`**)，这两个冒号后面再键入配方的名称（配方的文件名，不带 `.rb` 文件扩展名）。

5. 保留以下默认设置：
   - Comment (空白)
   - Advanced，Custom Chef JSON (空白)
   - 实例选择所有选中、MyCookbooksDemoLayer选中、cookbooks- demo1 选中）
6. 选择 Execute Recipes。此时将显示 Running command execute_recipes 页面。请在 Status 变为 successful 之后继续。此过程可能需要几分钟的时间，请耐心等待。

> **Note**
>
> 您无需手动运行配方。您可以将配方分配给图层的生命周期事件，例如设置和配置事件，当事件发生时，AWS OpsWorks Stacks 将自动运行这些配方。有关更多信息，请参阅 AWS OpsWorks 堆栈生命周期事件。

在下一步中，您将更新说明书以将用户添加到实例。

第 6 步：更新说明书以添加用户

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

通过添加可将本地用户添加到实例并设置该用户的主目录和 Shell 的配方，来更新您的说明书。这与运行 Linux adduser 或 useradd 命令或者 Windows net user 命令相似。您可以将一个本地用户添加到实例，例如，当您想要控制对该实例的文件和目录的访问时。

您也可以在不使用说明书的情况下管理用户。有关更多信息，请参阅 管理用户。

更新实例上的说明书并运行新的配方

1. 在您的本地工作站上的 `opsworks_cookbook_demo` 目录的 `recipes` 子目录中，创建名为 `add_user.rb` 的包含以下代码的文件 (有关更多信息，请转到用户)：

   ```
   user "Add a user" do
     home "/home/jdoe"
     shell "/bin/bash"
     username "jdoe"
   end
   ```

2. 在终端处或在命令提示符下，使用 tar 命令创建 `opsworks_cookbook_demo.tar.gz` 文件的新版本，该文件包含 `opsworks_cookbook_demo` 目录及其更新的内容。
3. 将更新后的 `opsworks_cookbook_demo.tar.gz` 文件上传到 S3 存储桶。
4. 按照第 5 步：更新实例上的说明书并运行配方中的步骤，更新实例上的说明书并运行配方。在“运行配方”步骤中，对于 Recipes to execute (要执行的配方)，键入 **`opsworks_cookbook_demo::add_user`**。

测试配方

1. 如果您尚未登录实例，请登录。
2. 在命令提示符下，运行以下命令以确认已添加新用户：

```
grep jdoe /etc/passwd
```

此时将显示以下用户信息，包括用户的名称、ID 号、组 ID 号、主目录和 Shell 等详细信息：

```
jdoe:x:501:502::/home/jdoe:/bin/bash
```

在下一步中，您将更新说明书以在实例上创建一个目录。

第 7 步：更新说明书以创建目录

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

通过添加可向实例添加目录的配方，更新您的说明书。这与运行 Linux mkdir 命令或者 Windows md 或 mkdir 命令相似。

更新实例上的说明书并运行新的配方

1. 在您的本地工作站上的 opsworks_cookbook_demo 目录的 recipes 子目录中，创建名为 create_directory.rb 的包含以下代码的文件。有关更多信息，请转至 directory：

   ```
   directory "Create a directory" do
     group "root"
     mode "0755"
     owner "ec2-user"
     path "/tmp/create-directory-demo"
   end
   ```

2. 在终端处或在命令提示符下，使用 tar 命令创建 opsworks_cookbook_demo.tar.gz 文件的新版本，该文件包含 opsworks_cookbook_demo 目录及其更新的内容。
3. 将更新后的 opsworks_cookbook_demo.tar.gz 文件上传到 S3 存储桶。

4. 按照第 5 步：更新实例上的说明书并运行配方中的步骤，更新实例上的说明书并运行配方。在“运行配方”步骤中，对于 Recipes to execute (要执行的配方)，键入 **opsworks_cookbook_demo::create_directory**。

测试配方

1. 如果您尚未登录实例，请登录。
2. 在命令提示符下，运行以下命令以确认已添加新目录：

```
ls -la /tmp/create-directory-demo
```

此时将显示新添加目录的相关信息，包括许可、所有者名称和组名称等信息：

```
drwxr-xr-x 2 ec2-user root 4096 Nov 18 00:35 .
drwxrwxrwt 6 root     root 4096 Nov 24 18:17 ..
```

在下一步中，您将更新说明书以在实例上创建一个文件。

第 8 步：更新说明书以创建并复制文件

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

通过添加可向实例添加两个文件的配方，更新您的说明书。配方中的第一个资源创建一个全部内容都是配方代码的文件。这与运行 Linux cat、echo 或 touch 命令或者 Windows echo 或 fsutil 命令相似。此方法适用于文件数量少、文件小或简单的情况。配方中第二个资源将说明书中的一个文件复制到实例上的另一个目录中。这与运行 Linux cp 命令或 Windows copy 命令相似。此方法适用于文件数量多、文件大或复杂的情况。

在开始执行此步骤之前，请完成第 7 步：更新说明书以创建目录，以确保文件的父目录已经存在。

更新实例上的说明书并运行新的配方

1. 在您的本地工作站上的 `opsworks_cookbook_demo` 目录中，创建名为 `files` 的目录。

2. 在 files 子目录中，创建一个名为 hello.txt 的文件，该文件包含以下文本：**Hello, World!**
3. 在 opsworks_cookbook_demo 目录的 recipes 子目录中，创建名为 create_files.rb 的包含以下代码的文件。有关更多信息，请转至 file 和 cookbook_file。

```
file "Create a file" do
  content "<html>This is a placeholder for the home page.</html>"
  group "root"
  mode "0755"
  owner "ec2-user"
  path "/tmp/create-directory-demo/index.html"
end

cookbook_file "Copy a file" do
  group "root"
  mode "0755"
  owner "ec2-user"
  path "/tmp/create-directory-demo/hello.txt"
  source "hello.txt"
end
```

file 资源可在指定路径中创建文件。cookbook_file 资源将您刚刚在说明书中创建的 files 目录中的文件 (Chef 预计会找到一个可以复制其中文件的、名为 files 的标准命名的子目录) 复制到实例的另一个目录中。

4. 在终端处或在命令提示符下，使用 tar 命令创建 opsworks_cookbook_demo.tar.gz 文件的新版本，该文件包含 opsworks_cookbook_demo 目录及其更新的内容。
5. 将更新后的 opsworks_cookbook_demo.tar.gz 文件上传到 S3 存储桶。
6. 按照第 5 步：更新实例上的说明书并运行配方中的步骤，更新实例上的说明书并运行配方。在“运行配方”步骤中，对于 Recipes to execute (要执行的配方)，键入 **opsworks_cookbook_demo::create_files**。

测试配方

1. 如果您尚未登录实例，请登录。
2. 在命令提示符下，运行以下命令 (一次运行一个命令)，以确认已添加新文件：

```
sudo cat /tmp/create-directory-demo/index.html
```

```
sudo cat /tmp/create-directory-demo/hello.txt
```

随即将显示该文件的内容：

```
<html>This is a placeholder for the home page.</html>

Hello, World!
```

在下一步中，您将更新说明书以在实例上运行一个命令。

第 9 步：更新说明书以运行命令

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

通过添加可运行用于在实例上创建 SSH 秘钥的命令的配方，更新您的说明书。

更新实例上的说明书并运行新的配方

1. 在您的本地工作站上的 opsworks_cookbook_demo 目录的 recipes 子目录中，创建名为 run_command.rb 的包含以下代码的文件。有关更多信息，请转到 execute。

   ```
   execute "Create an SSH key" do
     command "ssh-keygen -f /tmp/my-key -N fLyC3jbY"
   end
   ```

2. 在终端处或在命令提示符下，使用 tar 命令创建 opsworks_cookbook_demo.tar.gz 文件的新版本，该文件包含 opsworks_cookbook_demo 目录及其更新的内容。
3. 将更新后的 opsworks_cookbook_demo.tar.gz 文件上传到 S3 存储桶。
4. 按照第 5 步：更新实例上的说明书并运行配方中的步骤，更新实例上的说明书并运行配方。在“运行配方”步骤中，对于 Recipes to execute (要执行的配方)，键入 **opsworks_cookbook_demo::run_command**。

测试配方

1. 如果您尚未登录实例，请登录。
2. 在命令提示符下，运行以下命令 (一次运行一个命令)，以确认已创建 SSH 密钥：

```
sudo cat /tmp/my-key

sudo cat /tmp/my-key.pub
```

随即显示 SSH 私有和公有密钥的内容：

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-128-CBC,DEF7A09C...541583FA
A5p9dCuo...wp0YYH1c
-----END RSA PRIVATE KEY-----

ssh-rsa AAAAB3N...KaNogZkT root@cookbooks-demo1
```

在下一步中，您将更新说明书以在实例上运行一个脚本。

第 10 步：更新说明书以运行脚本

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

通过添加可在实例上运行脚本的配方来更新您的说明书。此配方会创建一个目录，然后在该目录中创建一个文件。编写一个配方以运行包含多个命令的脚本，比一次运行其中一个命令简单。

更新实例上的说明书并运行新的配方

1. 在您的本地工作站上的 `opsworks_cookbook_demo` 目录的 `recipes` 子目录中，创建名为 `run_script.rb` 的包含以下代码的文件。有关更多信息，请转到 script。

```
script "Run a script" do
```

```
  interpreter "bash"
  code <<-EOH
    mkdir -m 777 /tmp/run-script-demo
    touch /tmp/run-script-demo/helloworld.txt
    echo "Hello, World!" > /tmp/run-script-demo/helloworld.txt
  EOH
end
```

2. 在终端处或在命令提示符下，使用 tar 命令创建 opsworks_cookbook_demo.tar.gz 文件的新版本，该文件包含 opsworks_cookbook_demo 目录及其更新的内容。
3. 将更新后的 opsworks_cookbook_demo.tar.gz 文件上传到 S3 存储桶。
4. 按照第 5 步：更新实例上的说明书并运行配方中的步骤，更新实例上的说明书并运行配方。在“运行配方”步骤中，对于 Recipes to execute (要执行的配方)，键入 **opsworks_cookbook_demo::run_script**。

测试配方

1. 如果您尚未登录实例，请登录。
2. 在命令提示符下，运行以下命令以确认已添加新文件：

```
sudo cat /tmp/run-script-demo/helloworld.txt
```

随即将显示该文件的内容：

```
Hello, World!
```

在下一步中，您将更新说明书以管理实例上的服务。

第 11 步：更新说明书以管理服务

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

通过在实例上添加可管理服务的配方来更新您的说明书。这与运行 Linux service 命令或者 Windows net stop、net start 和类似命令相似。此配方会停止实例上的 crond 服务。

更新实例上的说明书并运行新的配方

1. 在您的本地工作站上的 opsworks_cookbook_demo 目录的 recipes 子目录中，创建名为 manage_service.rb 的包含以下代码的文件。有关更多信息，请转到 service。

```
service "Manage a service" do
  action :stop
  service_name "crond"
end
```

2. 在终端处或在命令提示符下，使用 tar 命令创建 opsworks_cookbook_demo.tar.gz 文件的新版本，该文件包含 opsworks_cookbook_demo 目录及其更新的内容。
3. 将更新后的 opsworks_cookbook_demo.tar.gz 文件上传到 S3 存储桶。
4. 按照第 5 步：更新实例上的说明书并运行配方中的步骤，更新实例上的说明书并运行配方。在“运行配方”步骤中，对于 Recipes to execute (要执行的配方)，键入 **opsworks_cookbook_demo::manage_service**。

测试配方

1. 如果您尚未登录实例，请登录。
2. 在命令提示符下，运行以下命令以确认 crond 服务已停止：

```
service crond status
```

随即将显示以下内容：

```
crond is stopped
```

3. 要重新启动 crond 服务，运行以下命令：

```
sudo service crond start
```

随即将显示以下内容：

```
Starting crond:  [  OK  ]
```

4. 要确认 crond 服务已启动，再次运行以下命令：

```
service crond status
```

随即将显示如下信息：

```
crond (pid  3917) is running...
```

在下一步中，您将更新说明书，以引用以自定义 JSON 形式存储于实例上的信息。

第 12 步：更新说明书以使用自定义 JSON

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

通过添加可引用存储于实例上的自定义 JSON 的配方，更新您的说明书。

当您创建、更新或克隆堆栈时，或者当您运行部署或堆栈命令时，您可以以自定义的 JSON 格式指定信息。例如，这对为实例上您的配方提供小部分不变数据来代替从数据库中获得此数据来说非常有用。有关更多信息，请参阅 使用自定义 JSON。

在本演练中，您将使用自定义 JSON 来提供关于客户发票的一些虚构信息。本步骤后面部分将对自定义 JSON 进行讲解。

更新实例上的说明书并运行新的配方

1. 在您的本地工作站上的 `recipes` 目录的 `opsworks_cookbook_demo` 子目录中，创建名为 `custom_json.rb` 的包含以下配方代码的文件：

```
Chef::Log.info("********** For customer '#{node['customer-id']}' invoice
 '#{node['invoice-number']}' **********")
Chef::Log.info("********** Invoice line number 1 is a '#{node['line-items']
['line-1']}' **********")
Chef::Log.info("********** Invoice line number 2 is a '#{node['line-items']
['line-2']}' **********")
```

```
Chef::Log.info("********** Invoice line number 3 is a '#{node['line-items']
['line-3']}' **********")
```

此配方显示日志中关于自定义 JSON 中的值的消息。

2. 在终端处或在命令提示符下，使用 tar 命令创建 opsworks_cookbook_demo.tar.gz 文件的新版本，该文件包含 opsworks_cookbook_demo 目录及其更新的内容。
3. 将更新后的 opsworks_cookbook_demo.tar.gz 文件上传到 S3 存储桶。
4. 按照第 5 步：更新实例上的说明书并运行配方中的步骤，更新实例上的说明书并运行配方。在“运行配方”步骤中，对于 Recipes to execute (要执行的配方)，键入 **opsworks_cookbook_demo::custom_json**。对于 Advanced、Custom Chef JSON，键入以下自定义 JSON：

```
{
  "customer-id": "0123",
  "invoice-number": "9876",
  "line-items": {
    "line-1": "tractor",
    "line-2": "passenger car",
    "line-3": "trailer"
  }
}
```

测试配方

1. 通过上一步骤显示了 Running command execute_recipes 页面时，对于 cookbooks-demo1 中的 Log，选择 show。此时将显示 execute_recipes 日志页面。
2. 向下滚动日志以查找类似于以下内容的条目：

```
[2015-11-14T14:18:30+00:00] INFO: ********** For customer '0123' invoice '9876'
 **********
[2015-11-14T14:18:30+00:00] INFO: ********** Invoice line number 1 is a 'tractor'
 **********
[2015-11-14T14:18:30+00:00] INFO: ********** Invoice line number 2 is a 'passenger
 car' **********
[2015-11-14T14:18:30+00:00] INFO: ********** Invoice line number 3 is a 'trailer'
 **********
```

这些条目显示在 Advanced、Custom Chef JSON 框中键入的自定义 JSON 的信息。

在下一步中，您将更新食谱以从数据袋中获取信息，数据袋是 Stac AWS OpsWorks ks 在每个实例上存储的堆栈设置的集合。

第 13 步：更新说明书以使用数据包

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

通过添加一个引用 Stac AWS OpsWorks ks 存储在实例上的堆栈设置的食谱来更新你的食谱，这些设置存储在一组数据袋中。此配方会在日志中显示存储在实例上的具体堆栈设置的消息。有关更多信息，请参阅 AWS OpsWorks 堆栈数据包参考。

更新实例上的说明书并运行新的配方

1. 在您的本地工作站上的 `recipes` 目录的 `opsworks_cookbook_demo` 子目录中，创建名为 `data_bags.rb` 的包含以下代码的文件：

   ```
   instance = search("aws_opsworks_instance").first
   layer = search("aws_opsworks_layer").first
   stack = search("aws_opsworks_stack").first

   Chef::Log.info("********** This instance's instance ID is
    '#{instance['instance_id']}' **********")
   Chef::Log.info("********** This instance's public IP address is
    '#{instance['public_ip']}' **********")
   Chef::Log.info("********** This instance belongs to the layer '#{layer['name']}'
    **********")
   Chef::Log.info("********** This instance belongs to the stack '#{stack['name']}'
    **********")
   Chef::Log.info("********** This stack gets its cookbooks from
    '#{stack['custom_cookbooks_source']['url']}' **********")
   ```

   此配方会在日志中显示存储在实例上的具体堆栈设置的消息。
2. 在终端处或在命令提示符下，使用 tar 命令创建 `opsworks_cookbook_demo.tar.gz` 文件的新版本，该文件包含 `opsworks_cookbook_demo` 目录及其更新的内容。
3. 将更新后的 `opsworks_cookbook_demo.tar.gz` 文件上传到 S3 存储桶。

4. 按照第 5 步：更新实例上的说明书并运行配方中的步骤，更新实例上的说明书并运行配方。在“运行配方”步骤中，对于 Recipes to execute (要执行的配方)，键入 **opsworks_cookbook_demo::data_bags**。

测试配方

1. 通过上一步骤显示了 Running command execute_recipes 页面时，对于 cookbooks-demo1 中的 Log，选择 show。此时将显示 execute_recipes 日志页面。
2. 向下滚动日志并查找类似于以下内容的条目：

```
[2015-11-14T14:39:06+00:00] INFO: ********** This instance's instance ID is
 'f80fa119-81ab-4c3c-883d-6028e52c89EX' **********
[2015-11-14T14:39:06+00:00] INFO: ********** This instance's public IP address is
 '192.0.2.0' **********
[2015-11-14T14:39:06+00:00] INFO: ********** This instance belongs to the layer
 'MyCookbooksDemoLayer' **********
[2015-11-14T14:39:06+00:00] INFO: ********** This instance belongs to the stack
 'MyCookbooksDemoStack' **********
[2015-11-14T14:39:06+00:00] INFO: ********** This stack gets its cookbooks from
 'https://s3.amazonaws.com/opsworks-demo-bucket/opsworks_cookbook_demo.tar.gz'
 **********
```

   此配方会显示存储在实例上的具体堆栈设置的消息。

在下一步中，您将更新说明书以多次运行配方代码。

第 14 步：更新说明书以使用迭代

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

通过添加可使用迭代 (一种多次重复配方代码的方法) 的配方，更新您的说明书。此配方会在日志中显示包含多方面内容的数据包项目的消息。

更新实例上的说明书并运行新的配方

1. 在您的本地工作站上的 recipes 目录的 opsworks_cookbook_demo 子目录中，创建名为 iteration_demo.rb 的包含以下代码的文件：

```
stack = search("aws_opsworks_stack").first
Chef::Log.info("********** Content of 'custom_cookbooks_source' **********")

stack["custom_cookbooks_source"].each do |content|
  Chef::Log.info("********** '#{content}' **********")
end
```

> **Note**
>
> 编写上述配方代码比编写下面未使用迭代的配方代码用时更短、更灵活，且较不容易出错。
>
> ```
> stack = search("aws_opsworks_stack").first
> Chef::Log.info("********** Content of 'custom_cookbooks_source' **********")
>
> Chef::Log::info("********** '[\"type\", \"#{stack['custom_cookbooks_source']
> ['type']}\"]' **********")
> Chef::Log::info("********** '[\"url\", \"#{stack['custom_cookbooks_source']
> ['url']}\"]' **********")
> Chef::Log::info("********** '[\"username\",
>  \"#{stack['custom_cookbooks_source']['username']}\"]' **********")
> Chef::Log::info("********** '[\"password\",
>  \"#{stack['custom_cookbooks_source']['password']}\"]' **********")
> Chef::Log::info("********** '[\"ssh_key\",
>  \"#{stack['custom_cookbooks_source']['ssh_key']}\"]' **********")
> Chef::Log::info("********** '[\"revision\",
>  \"#{stack['custom_cookbooks_source']['revision']}\"]' **********")
> ```

2. 在终端处或在命令提示符下，使用 tar 命令创建 opsworks_cookbook_demo.tar.gz 文件的新版本，该文件包含 opsworks_cookbook_demo 目录及其更新的内容。
3. 将更新后的 opsworks_cookbook_demo.tar.gz 文件上传到 S3 存储桶。
4. 按照第 5 步：更新实例上的说明书并运行配方中的步骤，更新实例上的说明书并运行配方。在“运行配方”步骤中，对于 Recipes to execute (要执行的配方)，键入 **opsworks_cookbook_demo::iteration_demo**。

测试配方

1. 通过上一步骤显示了 Running command execute_recipes 页面时，对于 cookbooks-demo1 中的 Log，选择 show。此时将显示 execute_recipes 日志页面。
2. 向下滚动日志并查找类似于以下内容的条目：

```
[2015-11-16T19:56:56+00:00] INFO: ********** Content of 'custom_cookbooks_source'
 **********
[2015-11-16T19:56:56+00:00] INFO: ********** '["type", "s3"]' **********
[2015-11-16T19:56:56+00:00] INFO: ********** '["url", "https://s3.amazonaws.com/
opsworks-demo-bucket/opsworks_cookbook_demo.tar.gz"]' **********
[2015-11-16T19:56:56+00:00] INFO: ********** '["username", "secret-key-value"]'
 **********
[2015-11-16T19:56:56+00:00] INFO: ********** '["password", "secret-access-key-
value"]' **********
[2015-11-16T19:56:56+00:00] INFO: ********** '["ssh_key", nil]' **********
[2015-11-16T19:56:56+00:00] INFO: ********** '["revision", nil]' **********
```

此配方会在日志中显示包含多方面内容的数据包项目的消息。数据包项目位于 aws_opsworks_stack 数据包中。数据包项目包含名为 custom_cookbooks_source 的内容。此内容包含六个部分，名称分别为 type、url、username、password、ssh_key 和 revision；还会显示它们的值。

在下一步中，您将更新说明书以仅在满足了某些条件后才运行配方代码。

第 15 步：更新说明书以使用条件逻辑

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

现在，通过添加可使用条件逻辑 (只有在满足某些条件时才运行代码的一种方法) 的配方来更新您的说明书。有关更多信息，请转到 if Statements 和 case Statements。

此配方会根据数据包内容执行两项操作：在日志中显示一则消息以确定实例在哪个操作系统上运行，以及只有在操作系统是 Linux 的情况下，才会为所规定的 Linux 发行版使用相应的程序包管理器来安装程序包。此程序包名为“tree”，是一款用于显示目录列表的简单应用程序。

更新实例上的说明书并运行新的配方

1. 在您的本地工作站上的 opsworks_cookbook_demo directory 的 recipes 子目录中，创建名为 conditional_logic.rb 的包含以下代码的文件：

```
instance = search("aws_opsworks_instance").first
os = instance["os"]

if os == "Red Hat Enterprise Linux 7"
  Chef::Log.info("********** Operating system is Red Hat Enterprise Linux.
 **********")
elsif os == "Ubuntu 14.04 LTS" || os == "Ubuntu 16.04 LTS" || os == "Ubuntu 18.04
 LTS"
  Chef::Log.info("********** Operating system is Ubuntu. **********")
elsif os == "Microsoft Windows Server 2012 R2 Base"
  Chef::Log.info("********** Operating system is Windows. **********")
elsif os == "Amazon Linux 2015.03" || os == "Amazon Linux 2015.09" || os == "Amazon
 Linux 2016.03" || os == "Amazon Linux 2016.09" || os == "Amazon Linux 2017.03"
 || os == "Amazon Linux 2017.09" || os == "Amazon Linux 2018.03" || os == "Amazon
 Linux 2"
  Chef::Log.info("********** Operating system is Amazon Linux. **********")
elsif os == "CentOS Linux 7"
  Chef::Log.info("********** Operating system is CentOS 7. **********")
else
  Chef::Log.info("********** Cannot determine operating system. **********")
end

case os
when "Ubuntu 14.04 LTS", "Ubuntu 16.04 LTS", "Ubuntu 18.04 LTS"
  apt_package "Install a package with apt-get" do
    package_name "tree"
  end
when "Amazon Linux 2015.03", "Amazon Linux 2015.09", "Amazon Linux 2016.03",
 "Amazon Linux 2016.09", "Amazon Linux 2017.03", "Amazon Linux 2017.09", "Amazon
 Linux 2018.03", "Amazon Linux 2", "Red Hat Enterprise Linux 7", "CentOS Linux 7"
  yum_package "Install a package with yum" do
    package_name "tree"
  end
else
```

```
  Chef::Log.info("********** Cannot determine operating system type, or operating
 system is not Linux. Package not installed. **********")
end
```

2. 在终端处或在命令提示符下，使用 tar 命令创建 opsworks_cookbook_demo.tar.gz 文件的新版本，该文件包含 opsworks_cookbook_demo 目录及其更新的内容。
3. 将更新后的 opsworks_cookbook_demo.tar.gz 文件上传到 S3 存储桶。
4. 按照第 5 步：更新实例上的说明书并运行配方中的步骤，更新实例上的说明书并运行配方。在“运行配方”步骤中，对于 Recipes to execute (要执行的配方)，键入 **opsworks_cookbook_demo::conditional_logic**。

测试配方

1. 通过上一步骤显示了 Running command execute_recipes 页面时，对于 cookbooks-demo1 中的 Log，选择 show。此时将显示 execute_recipes 日志页面。
2. 向下滚动日志并查找类似于以下内容的条目：

```
[2015-11-16T19:59:05+00:00] INFO: ********** Operating system is Amazon Linux.
 **********
```

   由于实例的操作系统是 Amazon Linux 2016.09，日志中将仅显示 (配方代码中五个可能的条目的) 前一个条目。
3. 如果操作系统是 Linux，配方会安装树包。要查看形象化的目录内容，在预期目录的命令提示符下键入 **tree** 或键入预期目录的路径（例如 `tree /var/chef/runs`）。

在下一步中，您将更新说明书，以使用由 Chef 社区提供的外部说明书的功能。

第 16 步：更新说明书以使用社区说明书

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

最后，更新说明书以使用 Chef 社区提供的外部说明书中的功能。您用于此演练的外部说明书，可以通过 Chef Supermarket (访问外部 Chef 说明书的一个普通地方) 获得。此外部说明书提供自定义资源，利用这个资源，您可以下载和安装应用程序，这类似于您在第 4 步：更新说明书以安装程序包中执行的操作。不过，此资源除了可以安装程序包以外，还可以 Web 应用程序和其他类型的应用程序。

当一个说明书依赖另一个说明书时，您必须指定对另一个说明书的依赖项。要宣布和管理说明书依赖项，我们建议您使用一个名为 Berkshelf 的工具。有关如何在本地工作站上安装 Berkshelf 的更多信息，请参阅 Chef 网站上的关于 Berkshelf。

当您安装 Berkshelf 后，按照以下步骤来宣布说明书依赖项，然后创建可在外部说明书中调用资源的配方：

宣布说明书依赖项

1. 在您的本地工作站上的 opsworks_cookbook_demo 目录中，在 metadata.rb 文件的结尾添加以下行：

   ```
   depends "application", "5.0.0"
   ```

   这可宣布对一个名为 application 的说明书 (版本 5.0.0.) 的依赖项。
2. 从 opsworks_cookbook_demo 目录的根目录运行以下命令。命令末尾的句点是有意设计的。

   ```
   berks init .
   ```

   Berkshelf 创建的许多文件夹和文件都可以用于以后更高级的方案。对于此演练，我们需要的唯一文件是名为 Berksfile 的文件。
3. 将以下行添加到 Berksfile 文件的末尾处：

   ```
   cookbook "application", "5.0.0"
   ```

   这将告知 Berkshelf，您要使用 Berkshelf 从 Chef Supermarket 下载的应用程序说明书版本 5.0.0。
4. 在终端处或在命令提示符下，在 opsworks_cookbook_demo 目录的根目录下运行以下命令：

   ```
   berks install
   ```

   Berkshelf 将创建一份您的说明书以及应用程序说明书的依赖项列表。Berkshelf 在下一个步骤中使用此依赖项列表。

更新实例上的说明书并运行新的配方

1. 在 `recipes` 目录的 `opsworks_cookbook_demo` 子目录中，创建名为 `dependencies_demo.rb` 的包含以下代码的文件：

   ```
   application "Install NetHack" do
     package "nethack.x86_64"
   end
   ```

   此方法依赖于应用程序食谱中的应用程序资源，以便在实例 NetHack 上安装流行的基于文本的冒险游戏。(当然，您可以替换想要替换的其他任何程序包名称，但前提是实例上程序包管理器要能够随时获得程序包。)

2. 从 `opsworks_cookbook_demo` 目录的根目录运行以下命令：

   ```
   berks package
   ```

   Berkshelf 使用上一个步骤中创建的依赖项列表创建一个名为 `cookbooks-`*`timestamp`*`.tar.gz` 的文件，该文件包含 `opsworks_cookbook_demo` 目录和其更新后的内容 (包括说明书所依赖的说明书)。将此文件重命名为 `opsworks_cookbook_demo.tar.gz`。

3. 将更新后重命名为 `opsworks_cookbook_demo.tar.gz` 的文件上传到 S3 存储桶。

4. 按照第 5 步：更新实例上的说明书并运行配方中的步骤，更新实例上的说明书并运行配方。在“运行配方”步骤中，对于 Recipes to execute (要执行的配方)，键入 **`opsworks_cookbook_demo::dependencies_demo`**。

5. 运行配方后，您应当能够登录实例，然后在命令提示符下键入 **`nethack`** 以开始播放。（有关游戏的更多信息，请参阅NetHack和NetHack指南。）

在下一步中，您可以清理用于本演练的 AWS 资源。下一步是可选的。随着您继续了解 AWS OpsWorks Stacks的更多信息，您可能需要继续使用这些 AWS 资源。但是，保留这些 AWS 资源可能会导致您的 AWS 账户持续收取一些费用。如果你想保留这些 AWS 资源以备后用，现在你已经完成了本演练，你可以直接跳到后续步骤。

步骤 17：(可选) 清除

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

为防止您的 AWS 账户产生额外费用，您可以删除用于本演练的 AWS 资源。这些 AWS 资源包括 S3 存储桶、堆栈 AWS OpsWorks 堆栈和堆栈的组件。（有关更多信息，请参阅 AWS OpsWorks 定价。）但是，随着您继续了解 AWS OpsWorks Stacks的更多信息，您可能需要继续使用这些 AWS 资源。如果你想让这些 AWS 资源保持可用，现在你已经完成了本演练，你可以跳到后续步骤。

为本演练创建的资源中存储的内容可以包含个人可识别信息。如果您不再希望由 AWS存储此信息，请执行本主题中的步骤。

删除 S3 存储桶

- 请参阅删除 Amazon S3 存储桶。

删除堆栈的实例

1. 在 AWS OpsWorks 堆栈控制台的服务导航窗格中，选择实例。这将显示 Instances 页面。
2. 对于 MyCookbooksDemoLayercookbooks-demo1，在 “操作” 中，选择 “停止”。看到确认消息后，选择 Stop。
3. 以下更改需要几分钟的时间才能生效。请等待以下所有操作均完成后再继续。
   - Status 从 online 变为 stopping，最终变为 stopped。
   - online 从 1 变为 0。
   - shutting down 从 0 变为 1，最终变为 0。
   - stopped 最终从 0 变为 1。
4. 对于 Actions，选择 delete。当您看到确认消息时，选择删除。 AWS OpsWorks Stacks 会删除该实例并显示 “无实例”。

删除堆栈

1. 在服务导航窗格中，选择 Stack。屏幕上随即显示 MyCookbooksDemoStack 页面。
2. 选择 Delete Stack。当您看到确认消息时，选择删除。 AWS OpsWorks Stacks 会删除堆栈并显示控制面板页面。

或者，如果您不想重复使用用于访问其他 AWS 服务和 EC2 实例的 IAM 用户和 Amazon EC2 密钥对，则可以将其删除。有关说明，请参阅：删除 IAM 用户及 Amazon EC2 密钥对和 Linux 实例。

现在，您已完成本演练。有关更多信息，请参阅 后续步骤。

后续步骤

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

现在你已经完成了本演练，你可以通过查看以下资源来详细了解 AWS OpsWorks Stacks 对 Chef 食谱的支持：

- 说明书和诀窍— 描述 AWS OpsWorks Stacks 目前支持的 Chef 和 Ruby 版本。此外，还演示如何在实例上安装和更新自定义说明书，以及如何在实例上运行配方。
- Learn Chef – 提供 Chef 教程、Chef 技能库、完整 Chef 文档和 Chef 培训课程的链接。
- All about Chef – 提供完整的 Chef 文档。相关特定主题包括：
  - About Cookbooks – 介绍主要说明书组件，如属性、配方、文件、元数据和模板。
  - About Recipes – 介绍配方的基础知识，例如，如何使用数据包、如何包括其他配方以及如何在配方中使用 Ruby 代码。
  - Resources – 介绍如何使用所有内置 Chef 资源，如 `apt_package`、`cookbook_file`、`directory`、`execute`、`file` 和 `package`。
  - About the Recipe DSL – 介绍如何使用语句 (如 `if`、`case`、`data_bag`、`data_bag_item` 和 `search`) 为 Chef 配方编写代码。
- About Templates – 介绍如何使用 Embedded Ruby (ERB) 模板不断生成静态文本文件，如配置文件。

- Learning Tracks – 介绍如何使用 Chef 管理实例、管理基本的 Web 应用程序、开发和测试基础设施代码、使用 Chef 分析等等。
- Learning Chef – Chef 简介。由 O'Reilly Media 出版。
- Learning Chef code examples – 提供随附于由 O'Reilly Media 出版的书籍 Learning Chef 的代码示例。

AWS OpsWorks 堆栈最佳实践

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

本节中的策略、技巧和建议可以帮助您从 AWS OpsWorks Stacks 中获得最大的收益和最佳结果。

主题

最佳实践：实例的根设备存储

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 本主题不适用于必须由 Amazon Elastic Block Store 支持的 Windows 实例。

Amazon Elastic Compute Cloud (Amazon EC2) Linux 实例具有以下根设备存储选项。

- 实例存储支持的实例-根设备是临时的。

  如果您停止此实例，根设备上的数据将消失并且无法恢复。有关更多信息，请参阅 Amazon EC2 实例存储。
- Amazon EBS 支持的实例-根设备为 Amazon EBS 卷。

  如果您停止此实例，Amazon EBS 卷仍然存在。如果您重新启动此实例，此卷将自动重新安装，并还原实例状态和存储的所有数据。您也可以在其他实例上安装此卷。有关更多信息，请参阅 Amazon Elastic Block Store（Amazon EBS）。

决定要使用的根设备存储选项时，请考虑以下事项。

启动时间

在初始启动之后，Amazon EBS 实例重新启动的速度通常更快。

任一存储类型的初始启动时间相差无几。这两种类型必须执行完整设置，其中包括相对耗时的任务，如通过远程存储库安装程序包。但是，当您随后重新启动实例时，应注意以下区别：

- 实例存储支持的实例将执行初始启动时执行过的相同设置任务，包括程序包安装。

  重新启动与初始启动所需时间大致相同。
- Amazon EBS 支持的实例将重新安装根卷并运行设置配方。

  重新启动通常明显快于初始启动，因为设置配方不必执行一些任务，例如重新安装已在根卷上安装的程序包。

费用

Amazon EBS 支持的实例成本更高：

- 在使用实例存储支持的实例时，您仅在实例运行时付费。
- 在使用 Amazon EBS 支持的实例时，无论实例是否运行，您都要为 Amazon EBS 卷付费。

  有关更多信息，请参阅 Amazon EBS 定价。

日志记录

Amazon EBS 支持的实例将自动保留日志：

- 在使用实例存储支持的实例时，日志将在实例停止时消失。

  您必须在停止实例之前检索日志，或者使用诸如CloudWatch 日志之类的服务来远程存储所选日志。
- 在使用 Amazon EBS 支持的实例时，日志将存储在 Amazon EBS 卷上。

  您可通过重新启动实例或将卷安装在另一实例上来查看日志。

依赖项

这两种存储类型都具有不同的依赖项：

- 实例存储支持的实例依赖 Amazon EBS。

  当您启动实例时，实例必须从 Amazon EBS 下载 AMI。
- Amazon EBS 支持的实例依赖 Amazon EBS。

  当您启动实例时，实例必须安装 Amazon EBS 根卷。

建议：如果您不确定最适合您的要求的存储类型，我们建议从 Amazon EBS 实例开始。尽管 Amazon EBS 卷会产生适度的费用，但数据意外丢失的风险更低。

最佳实践：优化应用程序服务器的数目

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

生产堆栈通常包含跨多个可用区分布的多个应用程序服务器。但是，传入请求的数量可能会有很大不同，具体取决于当日时间或星期几。您可以只运行足量服务器来处理预期的最大负载，而大多数情况下，您最终将为比所需容量更多的服务器容量付费。要高效运行您的站点，建议的做法是使服务器数量与当前请求卷匹配。

AWS OpsWorks Stacks 提供了三种管理服务器实例数量的方法。

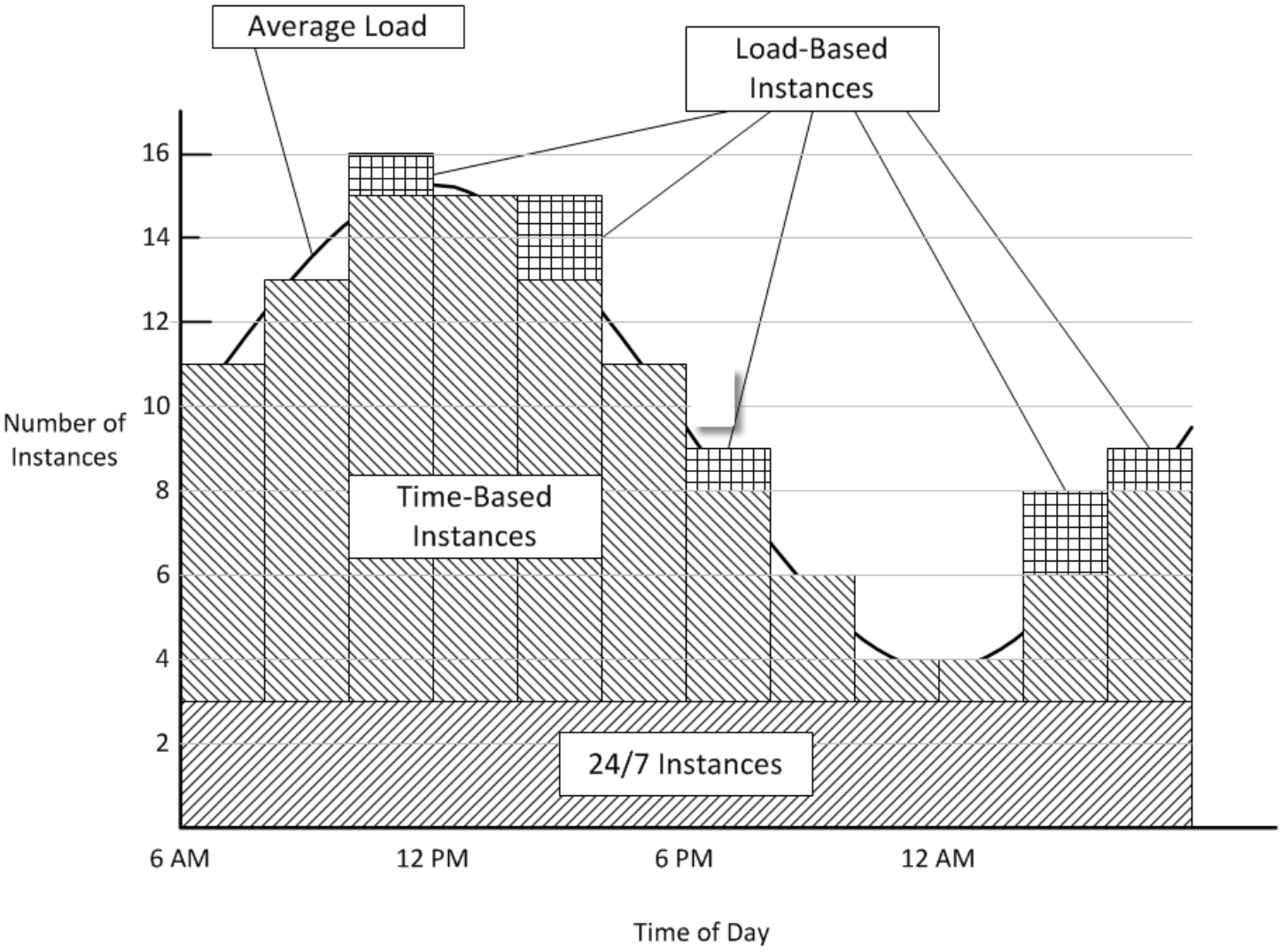
- 手动启动并运行全天候实例，直至其被手动终止。
- AWS OpsWorks Stacks 会按照用户指定的@@ 时间表自动启动和停止基于时间的实例。
- 当@@ 基于负载的实例超过用户指定的负载指标（例如 CPU 或内存利用率）的阈值时， AWS OpsWorks 堆栈会自动启动和停止。

> **Note**
>
> 在您创建并配置堆栈的基于时间和基于负载的实例后， AWS OpsWorks Stacks 将根据指定的配置自动启动和停止这些实例。除非您决定更改实例的配置或数目，否则您无需再次接触这些实例。

建议：如果您正在管理具有多个应用程序服务器实例的堆栈，建议您混合使用所有三个实例类型。以下示例说明如何管理堆栈的服务器容量以处理具备以下特性的可变每日请求量。

- 一天中的平均请求量呈正弦周期变化。
- 最小平均请求量需要 5 个应用程序服务器实例。
- 最大平均请求量需要 16 个应用程序服务器实例。
- 请求量峰值通常可由一个或两个应用程序服务器实例处理。

这是一个为了方便讨论而提供的模型，但您可以轻松调整来适应请求量的任何变化，并可对其进行扩展以处理每周变化。下图介绍如何使用三个实例类型来管理此请求量。



该示例具有以下特征：

- 堆栈具有 3 个全天候实例，这些实例始终运行并处理基本负载。
- 堆栈具有 12 个基于时间的实例，这些实例已配置为处理平均每日变化。

  1 个实例从 10 PM 运行至 2 AM，两个以上的实例从 8 PM 运行至 10 PM 以及从 2 AM 运行至 4 AM，以此类推。为简单起见，该图修改了每两小时的基于时间的实例数，但如果您希望实施更精细的控制，可修改每小时数量。
- 堆栈具有足够的基于负载的实例来处理流量峰值，这已超出全天候实例和基于时间的实例可处理的范围。

  AWS OpsWorks 只有当当前运行的所有服务器的负载超过指定指标时，堆栈才会启动基于负载的实例。未运行实例的成本最低 (由 Amazon EBS 提供支持的实例) 或没有成本 (由实例存储提供支持的

实例)，因此，建议的做法是创建足量的实例以顺利处理预期的最大请求量。在本示例中，堆栈应具有至少 3 个基于负载的实例。

> **Note**
>
> 确保您拥有跨多个可用区分布的所有三个实例类型，以减小任何服务中断所带来的影响。

最佳实践：管理权限

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

您必须拥有某种形式的 AWS 凭证才能访问您的账户资源。下面是一些向员工提供访问权限的一般准则。

- 首先，最重要的是，我们建议您不要使用您账户的根凭证来访问 AWS 资源。

  取而代之的是，为您的员工创建 IAM 身份并添加提供适当访问的权限。然后每个员工都可以使用凭证来访问资源。
- 员工应仅有权访问完成其工作所需的那些资源。

  例如，应用程序开发人员只需要访问运行其应用程序的堆栈。
- 员工应仅有权使用完成其工作所需的那些操作。

  应用程序开发人员可能需要完全权限，以获得将其应用程序部署到相应生产堆栈所需的开发堆栈和权限。他们可能不需要在生产堆栈上启动或停止实例、创建或删除层以及其他操作的权限。

有关管理权限的更多一般信息，请参阅 AWS 安全凭证。

您可以使用 AWS OpsWorks Stacks 或 IAM 来管理用户权限。请注意，两个选项并不是互斥的；有时需要同时使用它们。

AWS OpsWorks 堆栈权限管理

每个堆栈都有一个 Permissions 页面，您可以借助该页面向用户授予访问堆栈的权限并指定他们可执行哪些操作。通过设置以下其中一个权限等级，您可以指定用户的权限。每个等级都表示一个 IAM policy，该策略授予执行一组标准操作的权限。

- Deny 拒绝以任何方式与堆栈交互的权限。
- Show 授予查看堆栈配置的权限，但不能以任何方式修改堆栈状态。
- Deploy 包含 Show 权限，并且授予用户部署应用程序的权限。
- Manage 包含 Deploy 权限，并允许用户执行各种堆栈管理操作，如创建或删除实例和层。

> **Note**
>
> 管理权限级别不授予少量高级 AWS OpsWorks 堆栈操作的权限，包括创建或克隆堆栈。您必须使用 IAM policy 来授予这些权限。

除了设置权限等级，您还可以使用堆栈的 Permissions 页面来指定用户是否拥有对堆栈实例的 SSH/RDP 和 sudo/admin 权限。有关 AWS OpsWorks Stacks 权限管理的更多信息，请参阅授予每堆栈权限。有关管理 SSH 访问权限的更多信息，请参阅 管理 SSH 访问。

IAM 权限管理

借助 IAM 权限管理，您可以使用 IAM 控制台、API 或 CLI 将 JSON 格式的策略附加到显式指定其权限的用户。有关 IAM 权限管理的更多信息，请参阅什么是 IAM？。

建议：从 AWS OpsWorks 堆栈权限管理开始。如果您需要微调用户权限，或者授予用户 Manage 权限等级中不包含的权限，您可以结合使用这两种方法。 AWS OpsWorks 然后，Stacks 会评估这两个策略以确定用户的权限。

> **Important**
>
> 如果用户具有多个包含冲突权限的策略，则始终应该拒绝。例如，假设您将 IAM policy 附加到一个允许访问特定堆栈的用户，并且还使用该堆栈的 Permissions 页面向用户分配一个 Deny 权限等级。Deny 权限等级优先，并且用户将无法访问该堆栈。有关更多信息，请参阅 IAM policy 评估逻辑。

例如，假设您希望用户能够对堆栈执行除了添加或删除层以外的大多数操作。

- 指定 Manage 权限等级，它允许用户执行大多数堆栈管理操作，包括创建和删除层。
- 将以下客户管理的策略附加到用户，该策略将拒绝在该堆栈上使用CreateLayer和DeleteLayer操作的权限。您可以通过其 Amazon 资源名称 (ARN) 来标识该堆栈，可在堆栈的 Settings (设置) 页面上找到此名称。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "opsworks:CreateLayer",
        "opsworks:DeleteLayer"
      ],
      "Resource": "arn:aws:opsworks:*:*:stack/2f18b4cb-4de5-4429-a149-ff7da9f0d8ee/"
    }
  ]
}
```

有关包括示例策略在内的更多信息，请参阅 通过附加 IAM 策略管理 AWS OpsWorks 堆栈权限。

> **Note**
>
> 使用 IAM policy 的另一种方法是设置一个条件，用以限制员工对指定 IP 地址或地址范围的堆栈访问。例如，为了确保员工仅从企业防火墙内部访问堆栈，可以设置一个条件，限制对您的公司 IP 地址范围的访问。有关更多信息，请参阅条件。

最佳实践：管理和部署应用程序和说明书

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

AWS OpsWorks Stacks 会将应用程序和食谱部署到远程存储库中的每个新实例。在实例的生命周期内，您经常必须在堆栈的联机实例上更新应用程序和说明书，以添加功能、修复错误等。管理堆栈的应用程序和说明书的方法有很多种，但您使用的方法应满足以下一般要求：

- 所有生产堆栈实例应具有相同的应用程序和自定义说明书代码，除非出于 A/B 测试目的等有限的例外情况。
- 即使出现问题，部署更新也不应中断站点的操作。

本部分介绍管理和部署应用程序和自定义说明书的推荐做法。

主题

维持一致性

一般来说，您需要对您的生产堆栈上运行的应用程序或说明书代码保持严密的控制。通常，所有实例都应运行当前已获批准的代码版本。如下文所述，在更新您的应用程序或说明书时会出现例外，在适应特殊情况 (例如，执行 A/B 测试) 时也会出现例外。

通过两种方式将应用程序和说明书代码从指定的源存储库部署到您的堆栈的实例：

- 当您启动实例时， AWS OpsWorks Stacks 会自动将当前应用程序和食谱代码部署到该实例。
- 对于联机实例，您必须通过运行部署命令 (对于应用程序) 或更新自定义说明书命令 (对于说明书) 来手动部署当前应用程序或说明书代码。

由于有两种部署机制，因此请务必仔细地管理您的源代码，以避免在不同实例上意外地运行不同的代码。例如，如果您从 Git 主分支部署应用程序或食谱， AWS OpsWorks Stacks 会部署该分支中当时的内容。如果您更新主分支中的代码，然后启动一个新实例，则该实例的代码版本将比原来的实例更新。更新的版本可能甚至不会被批准用于生产。

建议：Amazon S3 存档

> 要确保您的所有实例均具有已批准的代码版本，我们建议您从 Amazon Simple Storage Service (Amazon S3) 存档部署您的应用程序和说明书。这可确保代码是必须是显式更新的静态构件 (.zip 或其他存档文件)。此外，Amazon S3 高度可靠，因此您无法访问存档的情况将极少 (如果有的

话)。为了进一步确保一致性，请使用命名约定或使用 Amazon S3 版本控制（提供审计跟踪记录并提供一种简单的方式来转换为之前的版本）来对每个存档文件进行显式版本管理。

例如，您可以使用一个工具 (如 Jenkins) 创建部署管道。提交并测试您要部署的代码之后，创建一个存档文件并将其上传到 Amazon S3。所有应用程序部署或说明书更新都将安装该存档文件中的代码，且每个实例将具有相同的代码。

建议：Git 或 Subversion 存储库

如果您倾向于使用 Git 或 Subversion 存储库，请勿从主分支部署。应改为标记已批准的版本并将该版本指定为应用程序或说明书源。

将代码部署到联机实例

AWS OpsWorks Stacks 不会自动将更新的代码部署到在线实例。您必须手动执行该操作，这会带来以下挑战：

- 高效地部署更新，而不影响站点在部署过程中处理客户请求的能力。
- 处理不成功的部署，不论是因为部署的应用程序或说明书的问题还是因为部署过程本身的问题。

最简单的方法是运行默认部署命令 (针对应用程序) 或者运行用于将更新同时部署到每个实例的更新自定义说明书命令 (对于说明书)。此方法简单快捷，但不允许犯错误。如果部署失败或更新的代码有任何问题，则您的生产堆栈中的每个实例都将受到影响，可能会中断或禁用您的站点，直到您解决问题或回滚到之前的版本。

建议：使用一个健壮的部署策略，允许运行旧版本代码的实例继续处理请求，直到您确认部署已成功并且可以放心地将所有传入流量转移到新版本。

以下部分提供了两个健壮的部署策略的示例，然后讨论了如何在部署期间管理后端数据库。为简洁起见，它们只介绍了应用程序更新，但您可以对说明书使用类似的策略。

主题

- 使用滚动部署
- 使用单独的堆栈
- 管理后端数据库

使用滚动部署

滚动部署将分多个阶段在堆栈的联机应用程序服务器实例上更新应用程序。在每个阶段中，您将更新联机实例的一个子集，并在开始下一个阶段之前确认更新已成功。如果您遇到问题，仍在运行旧的应用程序版本的实例可以继续处理传入流量，直到您解决该问题。

以下示例假设，您正在使用建议做法跨多个可用区分配堆栈的应用程序服务器实例。

执行滚动部署

1. 在“Deploy App”页面上，选择 Advanced，选择单个应用程序服务器实例，并将应用程序部署到该实例。

   如果您想要保持谨慎，可以从负载均衡器中删除该实例，然后再部署应用程序。这可以确保在您确认应用程序正常运行之前，用户不会遇到更新的应用程序。如果您使用 Elastic Load Balancing，请使用 Elastic Load Balancing 控制台、CLI 或 SDK 从负载均衡器中删除实例。
2. 确认已更新的应用程序运行正常，且实例具有可接受的性能指标。

   如果您从 Elastic Load Balancing 负载均衡器移除了该实例，请使用 Elastic Load Balancing 控制台、CLI 或 SDK 来还原它。更新后的应用程序版本现在正在处理用户请求。
3. 将更新部署至可用区中剩余的实例，并验证它们是否正常运行，以及是否具有可接受的指标。
4. 对堆栈的其他可用区重复第 3 步，一个可用区重复一次。如果您需要特别小心，请重复步骤 1~3。

> **Note**
>
> 如果您使用 Elastic Load Balancing 负载均衡器，则可以使用其运行状况检查来验证部署是否成功。但是，将 ping 路径部署为一个用于检查依赖项并确认一切正常的应用程序，而不是一个只是用于确认应用程序服务器正在运行的静态文件。

使用单独的堆栈

管理应用程序的另一种方法是对应用程序生命周期的每个阶段使用单独的堆栈。不同的堆栈有时称为环境。这种安排使您能够对不可公开访问的堆栈进行开发和测试。当您准备好部署更新时，将用户流量从托管当前应用程序版本的堆栈切换到托管更新后的版本的堆栈。

主题

- 使用开发、暂存和生产堆栈

- 使用蓝-绿部署策略

使用开发、暂存和生产堆栈

最常见的方法使用以下堆栈。

开发堆栈

将开发堆栈用于实现新功能或修复错误等任务。开发堆栈实质上是一个原型生产堆栈，具有与生产堆栈上相同的层、应用程序和资源等。由于开发堆栈通常不必处理与生产堆栈相同的负载，因此您通常可以使用更少或更小的实例。

开发堆栈不是面向公众的；您可按如下方式控制访问权限：

- 通过配置应用程序服务器或负载均衡器的安全组入站规则来限制网络访问权限，以便仅接受来自指定的 IP 地址或地址范围的传入请求。

  例如，限制 HTTP、HTTPS 和 SSH 访问您的公司地址范围内的地址。
- 使用堆 AWS OpsWorks 栈的 “权限” 页面，控制对堆栈堆栈管理功能的访问权限。

  例如，向开发团队授予管理权限级别，并向所有其他员工授予显示权限级别。

暂存堆栈

使用暂存堆栈测试并最终确定更新后的生产堆栈的候选项。完成开发后，可通过克隆开发堆栈来创建一个暂存堆栈。然后，在该暂存堆栈上运行您的测试套件并将更新部署到该堆栈，以修复出现的问题。

暂存堆栈也不是面向公众的；您可以使用与开发堆栈相同的方式控制堆栈和网络访问权限。请注意，在克隆开发堆栈以创建暂存堆栈时，可以克隆 AWS OpsWorks 堆栈权限管理授予的权限。但是，克隆不会影响用户的 IAM 策略所授予的权限。您必须使用 IAM 控制台、CLI 或开发工具包来修改这些权限。有关更多信息，请参阅 管理用户权限。

生产堆栈

生产堆栈是面向公众的堆栈，可支持您当前的应用程序。在暂存堆栈通过测试后，您可以将其提升到生产环境并停用旧的生产堆栈。有关如何执行此操作的示例，请参阅 使用蓝-绿部署策略。

> **Note**
>
> 与其使用 AWS OpsWorks Stacks 控制台手动创建堆栈，不如为每个堆栈创建一个 AWS CloudFormation 模板。这种方法有以下优势：

- 速度和便利性：当您启动模板时， AWS CloudFormation 会自动创建堆栈，包括所有必需的实例。
- 一致性：将每个堆栈的模板存储在您的源存储库内，以确保开发人员出于相同目的使用相同堆栈。

使用蓝-绿部署策略

蓝-绿 部署策略是高效利用不同的堆栈将应用程序更新部署到生产的一种常见方式。

- 蓝色环境是生产堆栈，用于托管当前的应用程序。
- 绿色环境是暂存堆栈，用于托管更新后的应用程序。

在您准备好将更新的应用程序部署到生产之后，便可以将用户流量从蓝色堆栈切换到绿色堆栈，后者将成为新的生产堆栈。然后，您可以停用原来的蓝色堆栈。

下面的示例介绍了如何结合 Route 53 和 Elastic Load Balancing 负载均衡器池，使用 AWS OpsWorks Stacks 堆栈执行蓝绿部署。进行切换之前，您应确保以下内容：

- 绿色堆栈上的应用程序更新通过了测试且可用于生产。
- 绿色堆栈与蓝色堆栈相同，只是它包含更新后的应用程序且不面向公众。

  这两个堆栈的权限相同、每层中的实例的数量和类型相同、基于时间和基于负载的配置相同，等等。
- 绿色堆栈的所有全天候实例和基于计划时间的实例均处于联机状态。
- 您有一个 Elastic Load Balancing 负载均衡器池，您可将它动态地挂载到任一堆栈中的层，也可以对它进行预热以处理预期的流量。
- 您已使用 Route 53 加权路由功能来在包括您的合并负载均衡器的托管区域中创建一个记录集。
- 您已将一个非零加权分配给挂载到您的蓝色堆栈的应用程序服务器层的负载均衡器，并将零加权分配到未使用的负载均衡器。这可以确保蓝色堆栈的负载均衡器处理所有传入的流量。

将用户切换到绿色堆栈

1. 将一个池的未使用的负载均衡器挂载到绿色堆栈的应用程序服务器层。在某些情况下，例如当您预计有闪现的流量时，或者如果您不能将负载测试配置为逐步增加流量，请对负载均衡器进行预热以处理预期的流量。

2. 在绿色堆栈的所有实例均通过 Elastic Load Balancing 运行状况检查后，更改 Route 53 记录集中的权重，以便绿色堆栈的负载均衡器具有非零权重，而蓝色堆栈的负载均衡器的权重相应减少。我们建议您开始时让绿色堆栈处理小部分请求，例如 5%，让蓝色堆栈处理剩余请求。现在您有两个生产堆栈，其中绿色堆栈处理部分传入请求，蓝色堆栈处理剩余请求。
3. 监控绿色堆栈的性能指标。如果它们是可接受的，则增加绿色堆栈的权重，让它处理约 10% 的传入流量。
4. 重复步骤 3，直至绿色堆栈处理约一半的传入流量。此时任何问题都应该已经浮现，因此，如果绿色堆栈的性能表现可接受，则可以通过将蓝色堆栈的权重减少到零来完成此过程。现在，绿色堆栈成为了新的蓝色堆栈，处理所有传入流量。
5. 从原来的蓝色堆栈的应用程序服务器层分离负载均衡器并将其退回到池。
6. 尽管原来的蓝色堆栈不再处理用户请求，我们仍然建议将其保留一段时间，以防新的蓝色堆栈出现问题。在这种情况下，您可以通过反向执行上述过程，将传入流量直接引导回原来的蓝色堆栈，从而回滚更新。当您确信新的蓝色堆栈的工作情况可接受之后，关闭原来的蓝色堆栈。

管理后端数据库

如果您的应用程序依赖于后端数据库，则需要从旧应用程序过渡到新应用程序。 AWS OpsWorks Stacks 支持以下数据库选项。

Amazon RDS 层

借助 Amazon Relational Database Service (Amazon RDS) 层，您可以单独创建 RDS 数据库实例，然后将它注册到您的堆栈。您一次仅可以将 RDS 数据库实例注册到一个堆栈，但您可以将 RDS 数据库实例从一个堆栈切换到另一个堆栈。

AWS OpsWorks Stacks 在您的应用程序服务器上安装一个包含连接数据的文件，其格式便于您的应用程序使用。 AWS OpsWorks 堆栈还将数据库连接信息添加到堆栈配置和部署属性中，配方可以访问这些信息。您也可以使用 JSON 向应用程序提供连接数据。有关更多信息，请参阅 连接到数据库。

更新依赖于数据库的应用程序带来了两个基本挑战：

- 在过渡期间要确保每个事务都得到正确记录，同时避免新旧应用程序版本之间出现竞用情况。
- 以限制对站点的性能影响且最大程度减少或消除停机时间的方式执行过渡。

当您使用本主题中所述的部署策略时，您不能只是将数据库从旧的应用程序分离并将它重新连接到新的应用程序。在过渡期间，两个版本的应用程序并行运行，且必须有权访问相同数据。下面介绍了管理过渡的两种方法，这两种方法各有优势和挑战。

方法 1：将这两个应用程序连接到同一数据库

优点

- 过渡期间没有停机时间。

  一个应用程序逐步停止访问数据库，而另一个应用程序逐步接管。

- 您不必在两个数据库之间同步数据。

挑战

- 两个应用程序均访问相同数据库，因此您必须管理访问权限，以防止数据丢失或损坏。
- 如果您需要迁移到新的数据库架构，旧的应用程序版本必须能够使用新的架构。

如果您使用单独的堆栈，这种方法可能最适合 Amazon RDS，因为实例没有与特定堆栈永久关联，并且可以通过在不同堆栈上运行的应用程序进行访问。但是，您不能将 RDS 数据库实例一次注册到多个堆栈，因此您必须为两个应用程序提供连接数据，例如，使用 JSON。有关更多信息，请参阅 使用自定义配方。

如果您使用滚动升级，则旧的和新的应用程序版本均在相同堆栈上托管，因此您既可以使用 Amazon RDS 层，也可以使用 MySQL 层。

方法 2：为每个应用程序版本提供各自的数据库

优点

- 每个版本均有各自的数据库，因此架构不必兼容。

挑战

- 过渡期间在两个数据库之间同步数据，而不会丢失或损坏数据。
- 确保您的同步过程不会导致严重停机或显著地降低站点的性能。

如果您使用不同的堆栈，则每个堆栈均有各自的数据库。如果您使用滚动部署，则可以将两个数据库挂载到堆栈，每个数据库针对一个应用程序。如果旧的和更新的应用程序的数据库架构不兼容，则此方法更好。

建议：一般来说，我们建议使用 Amazon RDS 层作为应用程序的后端数据库，因为它更灵活，并且可用于任何过渡场景。有关如何处理过渡的更多信息，请参阅《Amazon RDS 用户指南》。

本地打包说明书依赖项

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

您可以使用 Berkshelf 本地打包说明书依赖项，将该软件包上传到 Amazon S3，并修改您的堆栈以将 Amazon S3 上的软件包用作说明书源。发送到 Amazon S3 存储桶的内容可能包含客户内容。有关删除敏感数据的更多信息，请参阅如何清空 S3 存储桶？或如何删除 S3 存储桶？。

以下演练描述了如何将您的食谱及其依赖项预打包到.zip 文件中，然后使用.zip 文件作为 Stacks 中 Linux 实例的食谱来源。 AWS OpsWorks 第一个演练展示了如何打包一个说明书。第二个演练展示了如何打包多个说明书。

在开始之前，请先安装 Chef 开发工具包 (也称为 Chef DK)，该工具包是由 Chef 社区构建的各种工具。您需要此工具包才能使用 `chef` 命令行工具。

在 Chef 12 中打包依赖项

在 Chef 12 Linux 中，默认情况下，Berkshelf 不再安装在堆栈实例上。建议您在本地开发计算机上安装和使用 Berkshelf，以便在本地打包说明书依赖项。将您的程序包 (包含依赖项) 上传到 Amazon S3。最后，修改您的 Chef 12 Linux 堆栈，将已上传的程序包用作说明书源。当您在 Chef 12 中打包说明书时，请注意以下差异。

1. 在本地计算机上，通过运行 `chef` 命令行工具创建说明书。

   ```
   chef generate cookbook "server-app"
   ```

   此命令会创建一个说明书、一个 Berksfile、一个 `metadata.rb` 文件和一个配方目录，并将它们放置在与说明书同名的文件夹中。下面的示例显示所创建项目的结构。

   ```
   server-app <-- the cookbook you've just created
       ### Berksfile
       ### metadata.rb
       ### recipes
   ```

2. 在文本编辑器中，编辑 Berksfile 以指向 server-app 说明书将依赖的说明书。在我们的示例中，我们希望 server-app 依赖于 Chef Supermarket 中的 java 说明书。我们将指定版本 1.50.0 或更新的次要版本，但您可以在单引号中输入任何发布的版本。保存更改并关闭文件。

```
source 'https://supermarket.chef.io'
cookbook 'java', '~> 1.50.0'
```

3. 编辑 metadata.rb 文件以添加依赖项。保存更改并关闭文件。

```
depends 'java' , '~> 1.50.0'
```

4. 更改到 Chef 为您创建的 server-app 说明书目录，然后运行 package 创建说明书的 tar 文件。如果您要打包多个说明书，需要在存储所有说明书的根目录中运行此命令。要打包单个说明书，请在说明书目录级别运行此命令。在此示例中，我们在 server-app 目录中运行此命令。

```
berks package cookbooks.tar.gz
```

输出与以下内容类似。在本地目录中创建 tar.gz 文件。

```
Cookbook(s) packaged to /Users/username/tmp/berks/cookbooks.tar.gz
```

5. 在中 AWS CLI，将您刚刚创建的包上传到 Amazon S3。在将说明书程序包上传到 S3 后，记下此程序包的新 URL；您的堆栈设置需要此 URL。

```
aws s3 cp cookbooks.tar.gz s3://bucket-name/
```

输出与以下内容类似。

```
upload: ./cookbooks.tar.gz to s3://bucket-name/cookbooks.tar.gz
```

6. 在 AWS OpsWorks Stacks 中，修改您的堆栈以使用您上传的包作为食谱来源。

   a. 将 Use custom Chef cookbooks 设置设为 Yes。

   b. 将 Repository type 设置为 S3 Archive。

   c. 在 Repository URL 中，粘贴您在步骤 5 中上传的说明书程序包的 URL。

   保存您的堆栈更改。

本地打包一个说明书的依赖项

1. 在本地计算机上，使用 chef 命令行工具创建说明书：

   ```
   chef generate cookbook "server-app"
   ```

   此命令会创建一个说明书和 Berksfile，并将它们放置在与说明书同名的文件夹中。
2. 更改到 Chef 为您创建的说明书目录，然后通过运行以下命令打包所有内容：

   ```
   berks package cookbooks.tar.gz
   ```

   该输出类似于以下示例：

   ```
   Cookbook(s) packaged to /Users/username/tmp/berks/cookbooks.tar.gz
   ```
3. 在中 AWS CLI，将您刚刚创建的包上传到 Amazon S3：

   ```
   aws s3 cp cookbooks.tar.gz s3://bucket-name/
   ```

   该输出类似于以下示例：

   ```
   upload: ./cookbooks.tar.gz to s3://bucket-name/cookbooks.tar.gz
   ```
4. 在 AWS OpsWorks Stacks 中，修改您的堆栈以使用您上传的包作为食谱来源。

本地打包多个说明书的依赖项

此示例将创建两个说明书，并打包它们的依赖项。

1. 在本地计算机上，运行以下 `chef` 命令以生成两个说明书：

   ```
   chef generate cookbook "server-app"
   chef generate cookbook "server-utils"
   ```

   在本示例中，服务器应用程序说明书执行 Java 配置，因此我们需要添加对 Java 的依赖项。
2. 编辑 `server-app/metadata.rb` 以添加对社区 Java 说明书的依赖项：

```
maintainer "The Authors"
maintainer_email "you@example.com"
license "all_rights"
description "Installs/Configures server-app"
long_description "Installs/Configures server-app"
version "0.1.0"
depends "java"
```

3. 通过编辑说明书根目录中的 Berksfile 文件，来告诉 Berkshelf 要打包的内容，如下所示：

```
source "https://supermarket.chef.io"
cookbook "server-app", path: "./server-app"
cookbook "server-utils", path: "./server-utils"
```

现在，您的文件结构如下所示：

```
..
   ### Berksfile
   ### server-app
   ### server-utils
```

4. 最后，创建一个 zip 压缩包，将其上传到 Amazon S3，然后修改您的 AWS OpsWorks Stacks 堆栈以使用新的食谱来源。为此，请执行本地打包一个说明书的依赖项中的步骤 2 到步骤 4。

其他 资源

有关打包说明书依赖项的更多信息，请参阅以下内容。

- 如何在 AWS 博客上使用 Berkshelf 在本地打包 Cookbook 依赖关系 DevOps
- 论坛上@@ 有 Berkshelf 的 Linux Chef 12 AWS OpsWorks
- 论坛上的 Chef 12 中的 Berkshel f AWS OpsWorks
- 本指南中的安装自定义说明书
- 本指南中的说明书存储库

堆栈

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

堆栈是顶级 AWS OpsWorks 堆栈实体。它表示通常由于具有共同用途 (例如为 PHP 应用程序提供服务) 而需要您共同管理的一组实例。除了作为容器使用以外，堆栈还用于处理作为整体应用于实例组的任务，例如管理应用程序和说明书。

例如，一个用于向 Web 应用程序提供服务的堆栈可能与类似于下面这样：

- 一组应用程序服务器实例，每个实例均处理一部分的传入流量。
- 一个负载均衡器实例，用于接收传入流量，然后在多个应用程序服务器之间分配这些流量。
- 一个数据库实例，用作应用程序服务器的后端数据存储。

一种常见的做法是使用多个表示不同环境的堆栈。一组典型的堆栈包含：

- 一个开发堆栈，开发人员使用它来添加功能、修复错误以及执行其他开发和维护任务。
- 一个暂存堆栈，用于在发布前验证更新或修复。
- 一个生产堆栈，这是面向公众的版本，用于处理来自用户的传入请求。

本部分介绍了使用堆栈的基础知识。

主题

- 删除堆栈

将堆栈从 Amazon EC2-Classic 迁移到 VPC

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

本主题介绍如何将 AWS OpsWorks Stacks 堆栈从 Amazon EC2 Classic 网络平台迁移到亚马逊虚拟私有云 (Amazon VPC) 网络。

如果您在 2013-12-04 之前创建了 AWS 账户，则某些地区可能支持 EC2-Classic。 AWS 一些 Amazon EC2 资源和功能（如增强联网和较新的实例类型）需要 Virtual Private Cloud (VPC)。有些资源可在 EC2-Classic 和 VPC 之间共享，而有些则不能。为避免服务中断，我们建议您将 AWS OpsWorks Stacks 堆栈迁移到 VPC。

主题

先决条件

在开始之前，您必须拥有符合 AWS OpsWorks Stacks 配置要求的 VPC。要在您的 VPC 中为配置私有子网 AWS OpsWorks Stacks，请参阅本指南在 VPC 中运行堆栈中的。您可以使用 Amazon VPC 管理控制台创建自定义 VPC。有关更多信息，请参阅 Amazon Virtual Private Cloud 用户指南中的亚马逊 VPC 控制台向导配置以及 VPC 和子网。

要继续迁移，您需要提供您要使用的 VPC ID 和子网 ID。

将 AWS OpsWorks Stacks 堆栈迁移到 VPC

首先，使用 AWS OpsWorks Stacks 控制台或 API 克隆现有的 EC2-Classic 堆栈。然后，将现有堆栈的资源移至新堆栈。在克隆的堆栈中启动新实例，然后部署应用程序。验证新堆栈是否正常运行。最后，从 EC2-Classic 堆栈中删除 EC2-Classic 资源，然后删除旧堆栈。

1. 将现有的 EC2-Classic 堆栈克隆到您的 VPC 中。克隆堆栈会将堆栈设置、图层、应用程序、用户和用户权限复制到新堆栈。有关如何开始克隆堆栈的更多信息，请参阅本指南中的 克隆堆栈。

   您也可以使用 AWS OpsWorks Stacks API 克隆堆栈。使用 AWS CLI 或 AWS 软件开发工具包克隆堆栈时，请将`VpcId`参数的值设置为您在中先决条件 创建的 VPC 的 ID。有关更多信息，请参阅《AWS OpsWorks Stacks API 参考》中的 CloneStack。

2. 在克隆堆栈的层中创建新实例。请务必指定您在 先决条件 中创建的子网 ID。有关如何在堆栈中创建实例的更多信息，请参阅此指南中的 将实例添加到层。

3. 将您的经典资源（例如 EC2 安全组、Elastic Load Balancing 负载均衡器和弹性 IP 地址）迁移到您的 VPC，然后将其与克隆的堆栈关联。有关更多信息，请参阅 Amazon EC2 用户指南中的将资源迁移到 VPC。

4. 使用克隆的堆栈注册亚马逊 EBS 卷和 Amazon RDS 实例。有关使用堆栈注册资源的更多信息，请参阅本指南的 将资源注册到堆栈。

   Amazon EBS 卷不与 VPC 关联，您可以在 EC2-Classic 堆栈和 VPC 中的堆栈中跨实例使用它们。您可以在 EC2-Classic 中使用 EC2-Classic 堆栈和 VPC 中的堆栈注册 Amazon RDS 实例。

5. 在克隆的堆栈中启动实例，然后将一小部分工作负载移到克隆的堆栈。例如，将一小部分流量移至克隆堆栈中的 Elastic Load Balancing 负载均衡器。如果您在使用 Amazon Route 53，请参阅《Amazon Route 53 开发人员指南》中的将流量路由到 ELB 负载均衡器。

   在确定新堆栈正常运行并支持您的应用程序之前，仅路由一小部分流量。让新堆栈在试用期（例如一周）内使用一小部分流量。验证新堆栈是否正常运行后，将剩余流量路由到堆栈。

6. 确定克隆的堆栈正常运行后，将剩余的生产流量或工作负载移至克隆的堆栈。现在，您可以停用 EC2-Classic 堆栈中的实例。我们建议您将旧堆栈保持几周的可用状态，以便在迁移后的几周内，如果新堆栈出现任何问题，则可以将工作负载移回旧堆栈。

7. 当新堆栈已运行数周后，删除 EC2-Classic 堆栈中的实例。有关如何删除实例的更多信息，请参阅此指南中的 删除 AWS OpsWorks 堆栈实例。

   > **Important**
   >
   > 请勿使用 Amazon EC2 控制台或 API 停止或删除 AWS OpsWorks 实例。

8. 删除 EC2-Classic 堆栈中的应用程序。有关如何删除应用程序的更多信息，请参阅此指南中的从堆栈删除应用程序。

9. 删除 EC2-Classic 堆栈。有关如何删除堆栈的更多信息，请参阅本指南中的 删除堆栈。

另请参阅

- 从 EC2-Classic 迁移到 VPC
- 调试和故障排除指南
- 在 VPC 中运行堆栈

创建新堆栈

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

要创建新堆栈，请在 “ AWS OpsWorks 堆栈” 控制面板上，单击 “添加堆栈”。然后，您可以使用 Add Stack 页来配置堆栈。完成后，单击 Add Stack。

主题

- 选择要创建的堆栈类型
- 基本选项
- 高级选项

选择要创建的堆栈类型

创建堆栈前，您必须决定要创建的堆栈的类型。有关帮助，请参阅下表。

如果需要创建...	在以下情况下，创建此类型的堆栈	要了解如何操作，请遵循以下说明：
示例堆栈	OpsWorks 使用基于 Linux 的 Chef 12 堆栈和示例 Node.js 应用程序，探索 AWS 的基础知识。	入门：示例

如果需要创建...	在以下情况下，创建此类型的堆栈	要了解如何操作，请遵循以下说明：
基于 Linux 的 Chef 12 堆栈	创建一个基于 Linux 的堆栈，该堆栈使用 AWS OpsWorks 支持的最新版本的 Chef。如果您是高级 Chef 用户，并且您想从大量可选的社区说明书中获益，或编写您自己的自定义说明书，请选择此选项。有关更多信息，请参阅 Chef 12 Linux。	入门：Linux
基于 Windows 的 Chef 12.2 堆栈	创建基于 Windows 的堆栈。	入门：Windows
基于 Linux 的 Chef 11.10 堆栈	如果您的组织需要将 Chef 11.10 与 Linux 结合使用以实现向后兼容性，请创建此堆栈。	Chef 11 Linux 堆栈入门

基本选项

Add Stack 页包含以下基本选项。

堆栈名称

（必需）用于在 Stack AWS OpsWorks s 控制台中标识堆栈的名称。名称不必是唯一的。 AWS OpsWorks 堆栈还会生成堆栈 ID，这是一个唯一标识堆栈的 GUID。例如，对于 AWS CLI 命令（如 update-stack），您使用堆栈 ID 标识特定堆栈。创建堆栈后，您可以通过在导航窗格中选择 Stack，然后选择 Stack Settings 来查找堆栈 ID。该身份证标有OpsWorks 身份证。

区域

(必需) 将从中启动实例的 Amazon Web Services Region。

VPC

(可选) 堆栈将启动到的 VPC 的 ID。所有实例都将启动到此 VPC 中，并且您稍后无法更改 ID。

- 如果您的账户支持 EC2 Classic，则当您不需要使用 VPC 时，可指定 No VPC (默认值)。

有关 EC2 Classic 的更多信息，请参阅支持的平台。

- 如果您的账户不支持 EC2 Classic，则必须指定 VPC。

  默认设置为 Default VPC，它将 EC2 Classic 的易用性与 VPC 联网功能的好处集于一身。如果您需要在常规 VPC 中运行堆栈，则必须使用 VPC 控制台、API 或 CLI 来创建堆栈。有关如何为 AWS OpsWorks Stacks 堆栈创建 VPC 的更多信息，请参阅在 VPC 中运行堆栈。有关一般信息，请参阅 Amazon Virtual Private Cloud。

默认可用区/默认子网

(可选) 此设置依赖于您是否在 VPC 中创建堆栈：

- 如果您的账户支持 EC2 Classic，并且您将 VPC 设置为 无 VPC，则此设置将标有默认可用区，这会指定将从中启动实例的默认 AWS 可用区。
- 如果您的账户不支持 EC2 Classic 或者您选择指定 VPC，则此字段将标有 Default subnet，这会指定将从中启动实例的默认子网。您可以通过在创建实例时覆盖此值来在其他子网中启动实例。每个子网均与一个可用区关联。

通过在创建实例时覆盖此设置，您可以让 AWS OpsWorks Stacks 在不同的可用区或子网中启动该实例。

有关如何在 VPC 中运行堆栈的更多信息，请参阅在 VPC 中运行堆栈。

默认操作系统

(可选) 默认情况下安装在每个实例上的操作系统。您有以下选项：

- 内置 Linux 操作系统之一。
- Microsoft Windows Server 2012 R2。
- 基于某个支持的操作系统的自定义 AMI。

  如果您选择 Use custom AMI，则操作系统由创建实例时指定的自定义 AMI 决定。有关更多信息，请参阅 使用自定义 AMI。

有关可用操作系统的更多信息，请参阅AWS OpsWorks 堆栈操作系统。

> **Note**
>
> 您可以在创建实例时覆盖默认操作系统。不过，您无法覆盖 Linux 操作系统来指定 Windows，也无法覆盖 Windows 来指定 Linux 操作系统。

默认 SSH 密钥

(可选) 来自该堆栈区域的 Amazon EC2密钥对。默认值为 none。如果您指定密钥对， AWS OpsWorks Stacks 会在实例上安装公钥。

- 对于 Linux 实例，您可以将私有密钥用于 SSH 客户端来登录堆栈的实例。

  有关更多信息，请参阅 使用 SSH 登录。

- 对于 Windows 实例，您可以将私有密钥用于 Amazon EC2 控制台或 CLI 来检索实例的管理员密码。

  随后，您可以将密码用于 RDP 客户端来以管理员身份登录到实例。有关更多信息，请参阅 使用 RDP 登录。

有关如何管理 SSH 密钥的更多信息，请参阅管理 SSH 访问。

> **Note**
>
> 您可以通过在创建实例时指定其他密钥对来覆盖此设置或不指定任何密钥对。

Chef 版本

这将显示您选定的 Chef 版本。

有关 Chef 版本的更多信息，请参阅Chef 版本。

使用自定义 Chef 说明书

是否在堆栈的实例上安装自定义 Chef 说明书。

对于 Chef 12，默认设置为 Yes。对于 Chef 11，默认设置为 “否”。 “是” 选项显示了几个其他设置，这些设置为 AWS OpsWorks Stacks 提供了将自定义食谱从存储库部署到堆栈实例所需的信息，例如存储库 URL。详细信息取决于您对说明书使用的存储库。有关更多信息，请参阅 安装自定义说明书。

堆栈颜色

（可选）用于在 Stack AWS OpsWorks s 控制台上表示堆栈的色调。您可以对不同的堆栈使用不同的颜色来帮助在开发期间区分临时和生产堆栈。

堆栈标签

您可以在堆栈和层级别应用标签。当您创建一个标签后，您将对标记的结构范围内的每个资源应用该标签。例如，如果您向堆栈应用一个标签，则将向每个层应用该标签，而在每个层中，将向层中的每个实例、Amazon EBS 卷或 Elastic Load Balancing 负载均衡器应用该标签。有关如何激活标签并使用它们来跟踪和管理 AWS OpsWorks 堆栈资源成本的更多信息，请参阅《B illing and Cost Management 用户指南》中的使用成本分配标签和激活用户定义的成本分配标签。有关在 AWS OpsWorks 堆栈中添加标签的更多信息，请参阅。标签

高级选项

对于高级设置，单击 Advanced >> 可显示 Advanced options 和 Security 部分。

Advanced options 部分具有以下选项：

默认根设备类型

确定要用于实例的根卷的存储类型。有关更多信息，请参阅存储。

- 默认情况下，Linux 堆栈使用由 Amazon EBS 提供支持的根卷，但您也可以指定由实例存储提供支持的根卷。
- Windows 堆栈必须使用由 Amazon EBS 提供支持的根卷。

IAM 角色

（可选）堆栈的 AWS 身份和访问管理 (IAM) 角色， AWS OpsWorks Stacks 使用该角色代表您与 AWS 进行交互。

默认 IAM 实例配置文件

(可选) 要与堆栈的 Amazon EC2 实例关联的默认 IAM 角色。此角色向正在堆栈的实例上运行的应用程序授予对 AWS 资源 (如 S3 存储桶) 的访问权。

- 要向应用程序授予特定权限，请选择具有适当策略的现有实例配置文件 (角色)。
- 最初，配置文件的角色不授予任何权限，但您可以使用 IAM 控制台、API 或 CLI 来附加适当的策略。有关更多信息，请参阅 为在 EC2 实例上运行的应用程序指定权限。

API 端点区域

此设置采用您在堆栈的基本设置中选择的区域中的值。您可以从以下区域端点中选择。

- US East (N. Virginia) Region

- 美国东部（俄亥俄）区域
- US West (Oregon) Region
- 美国西部（北加利福尼亚）区域
- 加拿大（中部）区域（仅限 API；不适用于在中创建的堆栈 AWS Management Console
- 亚太地区（孟买）区域
- 亚太地区（新加坡）区域
- 亚太地区（悉尼）区域
- Asia Pacific（Tokyo）Region
- 亚太地区（首尔）区域
- 欧洲地区（法兰克福）区域
- 欧洲地区（爱尔兰）区域
- 欧洲地区（伦敦）区域
- 欧洲地区（巴黎）区域
- 南美洲（圣保罗）区域

在一个 API 端点中创建的堆栈不适用于另一个 API 端点。由于 AWS OpsWorks 堆栈用户也是特定于区域的，因此如果您希望其中一个终端节点区域的 AWS OpsWorks 堆栈用户管理另一个终端节点区域中的堆栈，则必须将用户导入到与堆栈关联的终端节点。有关导入用户的更多信息，请参阅将用户导入 AWS OpsWorks 堆栈。

主机名主题

(可选) 一个用于为每个实例生成默认主机名的字符串。默认值为 Layer Dependent，它使用实例层的短名称并向每个实例附加一个唯一的编号。例如，依赖角色的负载均衡器 主题根为“lb”。您添加到层的第一个实例名为“lb1”，您添加到层的第二个实例名为“lb2”，以此类推。

OpsWorks 代理版本

（可选）是在有新版本可用时自动更新 AWS OpsWorks Stacks 代理，还是使用指定的代理版本并手动更新。此功能适用于 Chef 11.10 和 Chef 12 堆栈。默认设置为 Manual update，这将设置为最新代理版本。

AWS OpsWorks Stacks 在每个实例上安装一个代理，该代理与服务通信并处理诸如启动 Chef 运行以响应生命周期事件之类的任务。此代理将定期更新。对于为堆栈指定代理版本，您有两个选项。

- 自动更新 — 一旦更新可用， AWS OpsWorks Stacks 就会自动在堆栈的实例上安装每个新的代理版本。

- 手动更新 — AWS OpsWorks Stacks 在堆栈的实例上安装指定的代理版本。

  AWS OpsWorks 当有新的代理版本可用时，Stacks 会在堆栈页面上发布一条消息，但不会更新堆栈的实例。要更新代理，必须手动更新堆栈设置以指定新的代理版本，然后 AWS OpsWorks Stacks 将更新堆栈的实例。

您可以通过更新特定实例的配置来覆盖其默认OpsWorks 代理版本设置。在此情况下，实例的设置优先。例如，假定默认设置为 Auto-update，而您为特定实例指定 Manual update。当 AWS OpsWorks Stacks 发布新的代理版本时，它将自动更新堆栈的所有实例，但设置为 “手动更新” 的实例除外。要在该实例上安装新的代理版本，您必须手动更新其配置并指定新版本。

> **Note**
>
> 控制台将显示缩写的代理版本号。要查看完整版本号，请调用 AWS CLI describe-agent-versions命令或等效的 API 或 SDK 方法。它们将返回可用代理版本的完整版本号。

自定义 JSON

(可选) 一个或多个格式化为 JSON 结构的属性。这些属性将合并为堆栈配置和部署属性，它们将安装到每个实例上并且可由配方使用。您可以使用自定义 JSON 来自定义配置设置，方式是覆盖指定默认设置的内置属性。有关更多信息，请参阅 使用自定义 JSON。

安全有一个选项，即使用 OpsWorks 安全组，它允许您指定是否将堆栈内置安全组与堆栈的层相关联。 AWS OpsWorks

AWS OpsWorks Stacks 提供了一组标准的内置安全组（每层一个），默认情况下，这些安全组与图层关联。使用 OpsWorks安全组允许您改为提供自己的自定义安全组。有关更多信息，请参阅 使用安全组。

使用 OpsWorks 安全组具有以下设置：

- 是- AWS OpsWorks Stacks 会自动将相应的内置安全组与每个层关联起来（默认设置）。

  您可以在创建一个层后将该层与额外的安全组关联，但无法删除内置安全组。
- 否- AWS OpsWorks Stacks 不会将内置安全组与层关联。

  您必须创建适当的 EC2 安全组并将一个安全组与创建的每个层关联。但是，您仍然可以在创建时手动关联内置的安全组和层；只有需要自定义设置的层才必须要自定义安全组。

请注意以下几点：

- 如果将 “使用 OpsWorks 安全组” 设置为 “是”，则无法通过向层添加限制性更强的安全组来限制默认安全组的端口访问设置。对于多个安全组，Amazon EC2 会使用最宽松的设置。此外，您无法通过修改内置安全组配置来创建更严格的设置。创建堆栈时，Stac AWS OpsWorks ks 会使用标准设置覆盖内置安全组的配置，因此您所做的任何更改都将在下次创建堆栈时丢失。如果某个层需要比内置安全组更严格的安全组设置，请将 “使用 OpsWorks 安全组” 设置为 “否”，使用您的首选设置创建自定义安全组，并在创建时将其分配给图层。
- 如果您不小心删除了 AWS OpsWorks Stacks 安全组并想要重新创建它，则该安全组必须与原始安全组完全相同，包括组名的大写。建议您让 AWS OpsWorks Stacks 为您重新创建组，而不是手动执行此任务。只需在同一 AWS 区域和 VPC（如果有）中创建一个新堆栈， AWS OpsWorks Stacks 就会自动重新创建所有内置安全组，包括您删除的安全组。如果您不再需要该堆栈，则随后您可以删除它；安全组将保留。

在 VPC 中运行堆栈

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

您可以通过在虚拟私有云 (VPC) 中创建堆栈的实例来控制用户对这些实例的访问。例如，您可能不希望用户直接访问您堆栈的应用程序服务器或数据库，而要求所有公有流量通过弹性负载均衡器传送。

在 VPC 中运行堆栈的基本步骤如下：

1. 利用 Amazon VPC 控制台、API 或 AWS CloudFormation 模板创建一个正确配置的 VPC。
2. 在创建堆栈时指定 VPC ID。
3. 在适当的子网中启动堆栈的实例。

下面简要介绍了 VPC 在 AWS OpsWorks Stacks 中的工作原理。

Important

如果使用 VPC 端点功能，则请注意堆栈中的每个实例必须能够从 Amazon Simple Storage Service (Amazon S3) 中完成以下操作：

- 安装实例代理。
- 安装资产，例如 Ruby。
- 上传 Chef 运行日志。
- 检索堆栈命令。

要启用这些操作，您必须确保堆栈的实例有权访问与堆栈区域匹配的以下存储桶。否则，上述操作将失败。
对于 Chef 12 Linux 和 Chef 12.2 Windows，存储桶如下所示。

代理存储桶	资产存储桶	日志存储桶	DNA 存储桶
• opsworks-instance-agent-sa-east-1 • opsworks-instance-agent-ap-south-1 • opsworks-instance-agent-ap-东北-1 • opsworks-instance-agent-ap-东北-2 • opsworks-instance-agent-ap-southeast-1 • opsworks-instance-agent-ap-东南-2 • opsworks-instance-agent-ca-central-1 • opsworks-instance-agent-eu-central-1 • opsworks-instance-agent-eu-west-1 • opsworks-instance-agent-eu-west-2	• opsworks-instance-assets-us-east-2 • opsworks-instance-assets-us-east-1 • opsworks-instance-assets-ap-south-1 • opsworks-instance-assets-ap-东北-1 • opsworks-instance-assets-ap-东北-2 • opsworks-instance-assets-ap-southeast-1 • opsworks-instance-assets-ap-东南-2 • opsworks-instance-assets-ca-central-1 • opsworks-instance-assets-eu-central-1 • opsworks-instance-assets-eu-west-1	• opsworks-us-east-2 日志 • opsworks-us-east-1-日志 • opsworks-ap-south-1-日志 • opsworks-ap-northeast-1-日志 • opsworks-ap-northeast-2 日志 • opsworks-ap-southeast-1-日志 • opsworks-ap-southeast-2 日志 • opsworks-ca-central-1-日志 • opsworks-eu-central-1-日志 • opsworks-eu-west-1-日志 • opsworks-eu-west-2 日志 • opsworks-eu-west-3-log • opsworks-sa-east-1-日志 • opsworks-us-west-1-日志 • opsworks-us-west-2 日志	• opsworks-us-east-2-dna • opsworks-us-east-1-dna • opsworks-ap-south-1-dna • opsworks-ap-northeast-1-dna • opsworks-ap-northeast-2-dna • opsworks-ap-southeast-1-dna • opsworks-ap-southeast-2-dna • opsworks-ca-central-1-dna • opsworks-eu-central-1-dna • opsworks-eu-west-1-dna • opsworks-eu-west-2-dna • opsworks-eu-west-3-dna • opsworks-sa-east-1-dna • opsworks-us-west-1-dna • opsworks-us-west-2-dna

代理存储桶	资产存储桶	日志存储桶	DNA 存储桶
• opsworks-instance-agent-eu-west-3 • opsworks-instance-agent-us-east-1 • opsworks-instance-agent-us-east-2 • opsworks-instance-agent-us-west-1 • opsworks-instance-agent-us-west-2	• opsworks-instance-assets-eu-west-2 • opsworks-instance-assets-eu-west-3 • opsworks-instance-assets-sa-east-1 • opsworks-instance-assets-us-west-1 • opsworks-instance-assets-us-west-2		

对于 Chef 11.10 和 Linux 的早期版本，存储桶如下所示。美国东部（弗吉尼亚州北部）外的区域端点不支持 Chef 11.4 堆栈。

代理存储桶	资产存储桶	日志存储桶	DNA 存储桶
• opsworks-instance-agent-us-east-2 • opsworks-instance-agent-us-east-1 • opsworks-instance-agent-ap-south-1 • opsworks-instance-agent-ap-东北-1 • opsworks-instance-agent-ap-东北-2 • opsworks-instance-agent-ap-southeast-1 • opsworks-instance-agent-ap-东南-2 • opsworks-instance-agent-ca-central-1 • opsworks-instance-agent-eu-central-1 • opsworks-instance-agent-eu-west-1	• opsworks-instance-assets-us-east-2 • opsworks-instance-assets-us-east-1 • opsworks-instance-assets-ap-south-1 • opsworks-instance-assets-ap-东北-1 • opsworks-instance-assets-ap-东北-2 • opsworks-instance-assets-ap-southeast-1 • opsworks-instance-assets-ap-东南-2 • opsworks-instance-assets-ca-central-1 • opsworks-instance-assets-eu-central-1 • opsworks-instance-assets-eu-west-1	• prod_stage-log	• prod_stage-dna

代理存储桶	资产存储桶	日志存储桶	DNA 存储桶
• opsworks-instance-agent-eu-west-2 • opsworks-instance-agent-eu-west-3 • opsworks-instance-agent-us-east-1 • opsworks-instance-agent-us-west-1 • opsworks-instance-agent-us-west-2	• opsworks-instance-assets-eu-west-2 • opsworks-instance-assets-eu-west-3 • opsworks-instance-assets-sa-east-1 • opsworks-instance-assets-us-west-1 • opsworks-instance-assets-us-west-2		

有关更多信息，请参阅 VPC 端点。

> **Note**
>
> 为了使 AWS OpsWorks 堆栈连接到您启用的 VPC 终端节点，您还必须为 NAT 或公有 IP 配置路由，因为 AWS OpsWorks 堆栈代理仍然需要访问公有终端节点。

主题

- VPC 基础知识
- 为 AWS OpsWorks 堆栈堆栈创建 VPC

VPC 基础知识

有关 VPC 的详细讨论，请参阅 Amazon Virtual Private Cloud。简而言之，VPC 包含一个或多个子网，每个子网都包含一个或多个实例。每个子网有一个关联的路由表，可根据其目标 IP 地址定向出站流量。

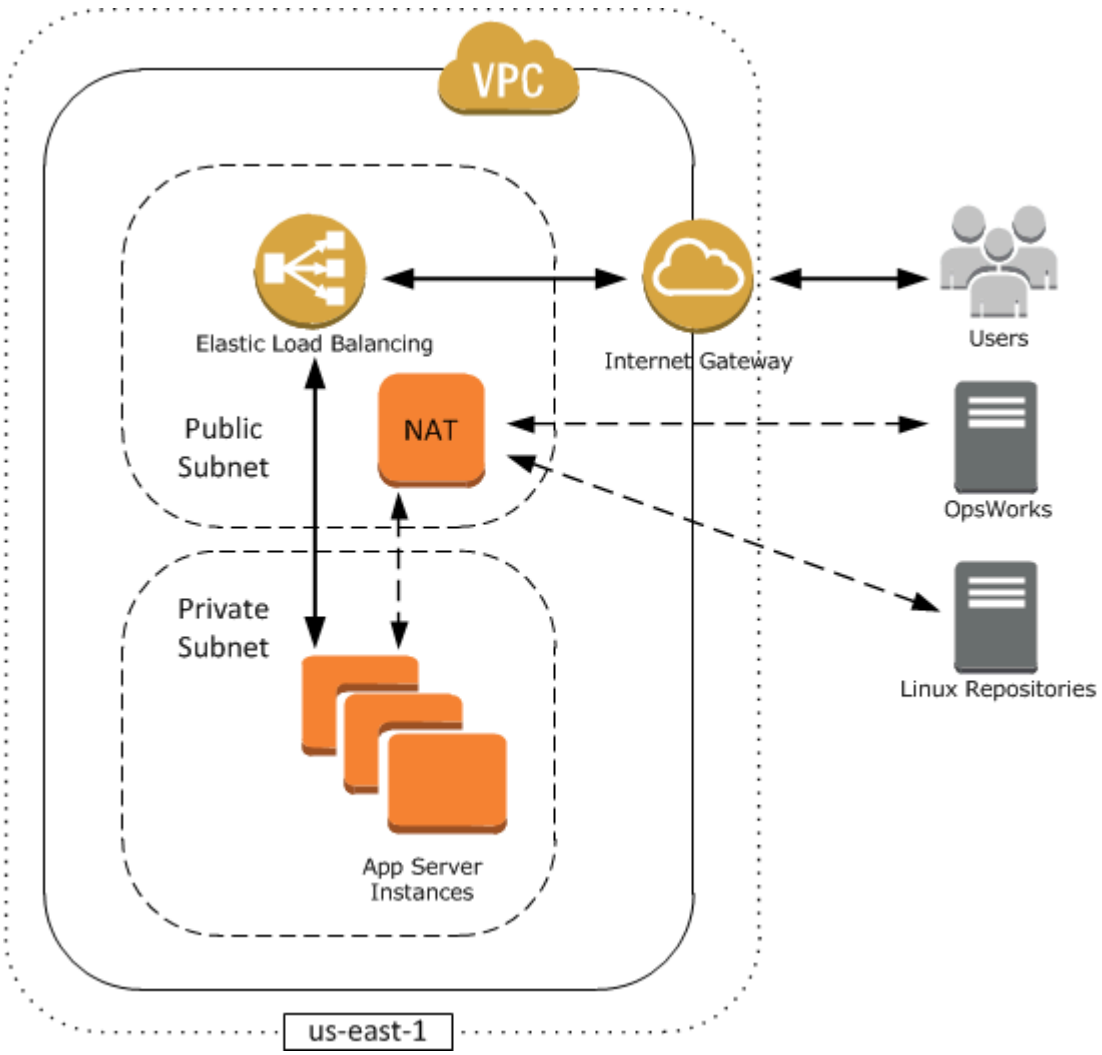
- VPC 内部的实例默认情况下可以相互通信，而不考虑其子网。但是，对网络访问控制列表 (ACL)、安全组策略的更改，或使用静态 IP 地址会破坏这种通信。
- 其实例可以与 Internet 通信的子网称为公有子网。
- 其实例仅可与 VPC 中的其他实例通信而不能直接与 Internet 通信的子网称为私有子网。

AWS OpsWorks 堆栈需要配置 VPC，以便堆栈中的每个实例（包括私有子网中的实例）都能访问以下终端节点：

- 的 “Region Support” 部分列出的 AWS OpsWorks Stacks 服务终端节点之一。AWS OpsWorks 堆栈入门
- 以下实例服务终端节点之一，由 AWS OpsWorks Stacks 代理使用。该代理在托管客户实例上运行以便与服务交换数据。
  - opsworks-instance-service.us-east-2.amazonaws.com
  - opsworks-instance-service.us-east-1.amazonaws.com
  - opsworks-instance-service.us-west-1.amazonaws.com
  - opsworks-instance-service.us-west-2.amazonaws.com
  - opsworks-instance-service.ap-south-1.amazows.com
  - opsworks-instance-service.ap-northeast-1.amazonaws.com
  - opsworks-instance-service.ap-northeast-2.amazonaws.com
  - opsworks-instance-service.ap-southeast-1.amazonaws.com
  - opsworks-instance-service.ap-southeast-2.amazonaws.com
  - opsworks-instance-service.ca-central-1.amazows.com
  - opsworks-instance-service.eu-central-1.amazonaws.com
  - opsworks-instance-service.eu-west-1.amazonaws.com
  - opsworks-instance-service.eu-west-2.amazonaws.com
  - opsworks-instance-service.eu-west-3.amazonaws.com
- Amazon S3

- 您的操作系统依靠的任何软件包存储库，例如 Amazon Linux 或 Ubuntu Linux 存储库。
- 您的应用程序和自定义说明书存储库。

可以使用多种方法来配置 VPC 以提供此连接。以下是如何为 AWS OpsWorks Stacks 应用程序服务器堆栈配置 VPC 的简单示例。



此 VPC 有几个组件：

子网

VPC 有两个子网：一个公有子网和一个私有子网。

- 公有子网中包含一个负载均衡器和一个网络地址转换 (NAT) 设备，它可以与外部地址以及私有子网中的实例进行通信。
- 私有子网包含应用程序服务器，它可与公有子网中的 NAT 和负载均衡器通信，但无法与外部地址直接通信。

Internet 网关

Internet 网关允许具有公有 IP 地址的实例 (例如负载均衡器) 与 VPC 外的地址进行通信。

负载均衡器

Elastic Load Balancing 负载均衡器接收用户的传入流量、将其分配到私有子网中的应用程序服务器，然后向用户返回响应。

NAT

(NAT) 设备为应用程序服务器提供有限的 Internet 访问，通常用于从外部存储库下载软件更新等用途。所有 AWS OpsWorks 堆栈实例都必须能够与 AWS OpsWorks 堆栈和相应的 Linux 存储库通信。处理此问题的一种方法是将具有关联的弹性 IP 地址的 NAT 设备放在公有子网中。然后，您就可以通过 NAT 对来自私有子网中实例的出站流量进行路由。

> **Note**
>
> 一个 NAT 实例可在您的私有子网的出站流量中创建一个单点故障。您可以配置具有一对 NAT 实例的 VPC 来提高可靠性，这对实例可在其中一个出现故障时由另一个接管。有关更多信息，请参阅 Amazon VPC NAT 实例的高可用性。您也可以使用 NAT 网关。有关更多信息，请参阅《Amazon VPC 用户指南》中的 NAT。

最佳 VPC 配置取决于您的堆栈 AWS OpsWorks 堆栈。以下是一些您可能使用特定 VPC 配置时的示例。有关其他 VPC 场景的示例，请参阅使用 Amazon VPC 的场景。

处理公有子网中的一个实例

如果您有一个没有关联私有资源的单一实例堆栈 (例如不应可公开访问的 Amazon RDS 实例)，您就可以创建具有一个公有子网的 VPC 并将实例放在该子网中。如果您不使用默认 VPC，则必须让实例的层为该实例分配弹性 IP 地址。有关更多信息，请参阅 OpsWorks 图层基础知识。

处理私有资源

如果您有不应公开访问的资源，则可以创建具有一个公有子网和一个私有子网的 VPC。例如，在负载均衡的自动扩展环境中，您可以将私有子网中的所有 Amazon EC2 实例和负载均衡器放在公有子网中。这样，就不能通过 Internet 直接访问 Amazon EC2 实例了；所有传入流量必须通过负载均衡器来路由。

私有子网将实例与 Amazon EC2 直接用户访问隔离开，但它们仍然必须向 AWS 和适当的 Linux 软件包存储库发送出站请求。若要允许此类请求，您可以使用诸如具有自己的弹性 IP 地址的网络地址

转换 (NAT) 设备，然后通过 NAT 对实例的出站流量进行路由。您可以将 NAT 放在与负载均衡器相同的公有子网中，如上述示例所示。

- 如果您使用的是诸如 Amazon RDS 实例等后端数据库，则可以将这些实例放在私有子网中。对于 Amazon RDS 实例，则必须在不同的可用区内指定至少两个不同子网。
- 如果需要直接访问私有子网中的实例 (例如，要使用 SSH 登录实例)，可以在公有子网中放置一台堡垒主机，代理来自互联网的请求。

将您自己的网络扩展到 AWS

如果您要将自己的网络扩展到云中并直接从您的 VPC 访问 Internet，则可以创建 VPN 网关。有关更多信息，请参阅场景 3：具有公有和私有子网的 VPC 以及硬件 VPN 访问。

为 AWS OpsWorks 堆栈堆栈创建 VPC

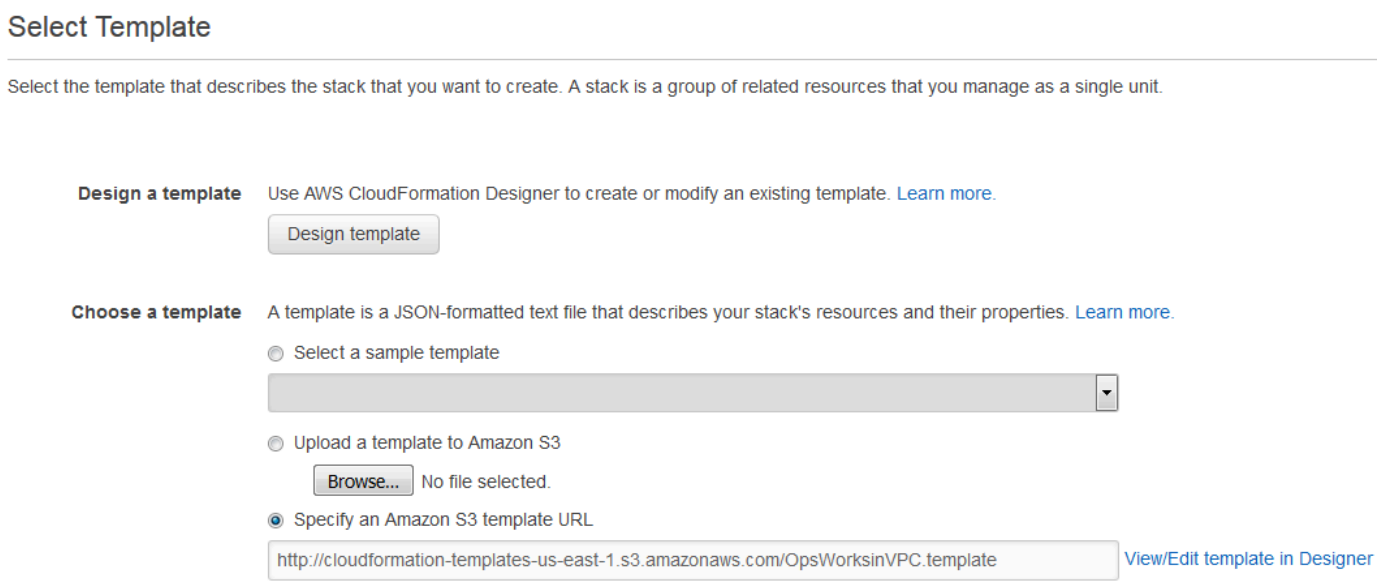
本节介绍如何使用示例 AWS CloudFormation 模板为 AWS OpsWorks 堆栈创建 VPC。您可以在 OpsWorksVPCtemplates.zip 文件中下载该模板。有关如何手动创建如本主题中所述 VPC 的更多信息，请参阅Scenario 2: VPC with Public and Private Subnets。有关如何配置路由表、安全组等信息，请参阅示例模板。

> **Note**
>
> 默认情况下， AWS OpsWorks Stacks 通过串联其 CIDR 范围和可用区来显示子网名称，例如。`10.0.0.1/24 - us-east-1b`为了使名称更具可读性，请为每个子网创建一个标记，密钥设置为**`Name`**，值设置为子网名称。 AWS OpsWorks 然后，堆栈会将子网名称附加到默认名称之后。例如，以下示例中的私有子网有一个标记，名称设置为**`Private`**， OpsWorks 显示为`10.0.0.1/24 us-east - 1b - Private`。

您只需几个步骤即可使用 AWS CloudFormation 控制台启动 VPC 模板。以下过程使用示例模板在美国东部（弗吉尼亚州北部）区域创建 VPC。有关如何使用模板在其他区域创建 VPC 的说明，请参阅过程后面的备注。

创建 VPC

1. 打开 AWS CloudFormation 控制台，选择 US East (N. Virginia) (美国东部 (弗吉尼亚北部)) 区域，然后选择 Create Stack (创建堆栈)。
2. 在 Select Template (选择模板) 页面上，选择 Upload a template (上传模板)。浏览您在 OpsWorksVPCtemplates.zip `OpsWorksinVPC.template` 文件中下载的文件。选择 “继续”。



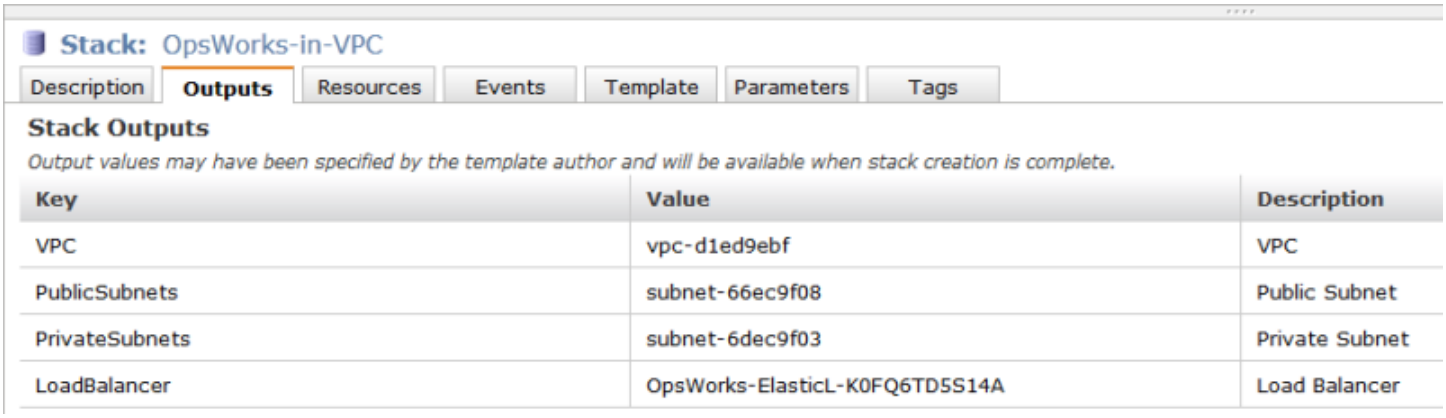
您也可以通过打开 AWS CloudFormation 示例模板，找到堆栈 VPC 模板并选择启动堆栈来启动此堆栈。 AWS OpsWorks

3. 在 Specify Parameters (指定参数) 页面上，接受默认值并选择 Continue (继续)。
4. 在 Add Tags (添加标签) 页面上，创建一个标签，将 Key (键) 设置为 **Name**，并将 Value (值) 设置为 VPC 名称。在创建堆栈 AWS OpsWorks 堆栈时，使用此标签可以更轻松地识别您的 VPC。
5. 选择 Continue (继续)，然后选择 Close (关闭) 以启动堆栈。

注意：您可以使用以下任一方法在其他区域中创建 VPC。

- 转至 “在不同区域使用模板”，选择相应的区域，找到 AWS OpsWorks Stacks VPC 模板，然后选择 La unch Stack。
- 将模板文件复制到您的系统中，在 AWS CloudFormation 控制台中选择适当的区域，并使用 Create Stack (创建堆栈) 向导的 Upload a template to Amazon S3 (将模板上传到 Amazon S3) 选项从您的系统中上传模板。

示例模板包括提供创建 AWS OpsWorks 堆栈所需的 VPC、子网和负载均衡器 ID 的输出。您可以通过选择 AWS CloudFormation 控制台窗口底部的 “输出” 选项卡来查看它们。


Stack: OpsWorks-in-VPC

Description | Outputs | Resources | Events | Template | Parameters | Tags

Stack Outputs

Output values may have been specified by the template author and will be available when stack creation is complete.

Key	Value	Description
VPC	vpc-d1ed9ebf	VPC
PublicSubnets	subnet-66ec9f08	Public Subnet
PrivateSubnets	subnet-6dec9f03	Private Subnet
LoadBalancer	OpsWorks-ElasticL-K0FQ6TD5S14A	Load Balancer

更新堆栈

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

创建堆栈后，您可以随时更新配置。在 Stack 页面上，单击 Stack Settings，然后单击 Edit，这将显示 Settings 页面。执行所需的更改，然后单击 Save。

这些设置与创建新堆栈中讨论的设置相同。有关详细信息，请参阅该主题。但是，请注意以下事项：

- 您无法修改区域或 VPC ID。
- 如果您的堆栈正在 VPC 中运行，这些设置将包含 Default subnet 设置，该设置列出了 VPC 的子网。如果您的堆栈未在 VPC 中运行，这些设置将被标上 Default Availability Zones 并列出区域的可用区。
- 您可以更改默认操作系统，但您无法为 Windows 堆栈指定 Linux 操作系统或为 Linux 堆栈指定 Windows 操作系统。
- 如果您更改了任一默认实例设置 (如 Hostname theme 或 Default SSH key)，新值将仅应用于您创建的所有新实例而非现有实例。
- 更改名称会更改控制台显示的名称；它不会更改 AWS OpsWorks Stacks 用来标识堆栈的基础短名称。
- 在将 “使用 OpsWorks 安全组” 从 “是” 更改为 “否” 之前，除了该层的内置安全组外，每层还必须至少有一个安全组。有关更多信息，请参阅 编辑图 OpsWorks 层的配置。

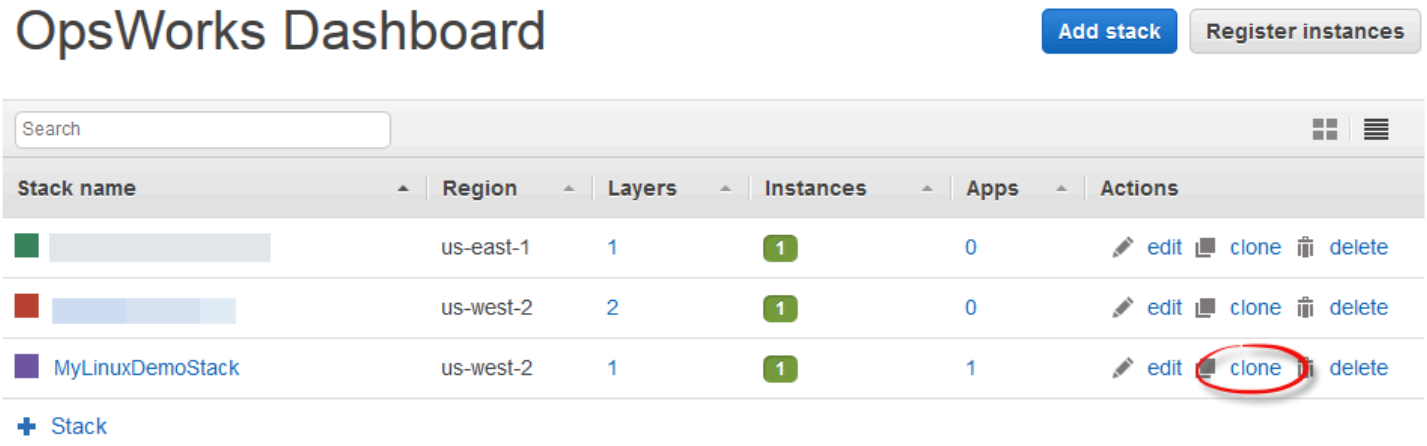
AWS OpsWorks 然后，堆栈会从每一层删除内置安全组。

- 如果您将 “使用 OpsWorks 安全组” 从 “否” 更改为 “是”，则 AWS OpsWorks Stacks 会向每层添加相应的内置安全组，但不会删除现有的安全组。

克隆堆栈

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

有时创建一个堆栈的多个副本非常有用。例如，您可能希望作为灾难恢复或预防性措施来添加冗余，或者您可以将现有堆栈用作新堆栈的起点。最简单的方法是克隆源堆栈。在 AWS OpsWorks 堆栈仪表板上，在要克隆的堆栈所在行的操作列中，选择克隆，这将打开克隆堆栈页面。



最初，克隆堆栈的设置与源堆栈的设置相同，但会将“copy”一词追加到堆栈名称后。有关这些设置的信息，请参阅创建新堆栈。还有两个额外的可选设置：

权限

如果已选中 all permissions (默认值)，则源堆栈权限会应用到克隆堆栈。

应用程序

列出部署在源堆栈上的应用程序。对于所列的每个应用程序，如果已选中相应的复选框 (默认值)，则应用程序将部署到克隆堆栈。

> **Note**
>
> 您不能将堆栈从一个区域端点克隆到另一个区域端点；例如，您不能将堆栈从美国西部（俄勒冈州）区域 (us-west-2) 克隆到亚太地区（孟买）区域 (ap-south-1)。

完成设置后，选择克隆堆栈。 AWS OpsWorks Stacks 会创建一个新的堆栈，该堆栈由源堆栈的层以及可选的应用程序和权限组成。层与原来的层配置相同，您可以做出任何修改。但是，克隆不会创建任何实例。您必须为每层克隆堆栈添加一组适当的实例，然后启动它们。对于任何堆栈，您可以对克隆堆栈执行常规管理任务，如添加、删除或修改层或者添加并部署应用程序。

要使克隆的堆栈正常运行，请启动实例。 AWS OpsWorks Stacks 根据每个实例的层成员资格来设置和配置每个实例。它还部署任何应用程序，就像处理新堆栈一样。

运行堆栈 AWS OpsWorks 堆栈命令

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

AWS OpsWorks Stacks 提供了一组堆栈命令，您可以使用这些命令对堆栈的实例执行各种操作。要运行堆栈命令，请单击堆栈页面上的运行命令。之后，选择相应的命令，指定任何选项，并按右下角标有命令名称的按钮。

> **Note**
>
> AWS OpsWorks Stacks 还支持一组部署命令，您可以使用这些命令来管理应用程序部署。有关更多信息，请参阅 部署应用程序。

您可对任何堆栈运行下列堆栈命令。

更新自定义说明书

使用存储库中的当前版本更新实例的自定义说明书。此命令不会运行任何配方。要运行更新后的配方，您可使用 Execute Recipes、Setup 或 Configure 堆栈命令，也可重新部署您的应用程序以运行部署配方。有关自定义说明书的更多信息，请参阅说明书和诀窍。

执行配方

对实例执行一组指定的配方。有关更多信息，请参阅 手动运行配方。

设置

运行实例的 Setup 配方。

配置

运行实例的 Configure 配方。

> **Note**
>
> 要使用 Setup 或 Configure 对某个实例运行配方，必须将配方分配给该实例层对应的生命周期事件。有关更多信息，请参阅 执行配方。

您只能对基于 Linux 的堆栈运行下列堆栈命令。

安装依赖项

安装实例的程序包。从 Chef 12 开始，此命令不可用。

Update Dependencies

（仅适用于 Linux。从 Chef 12 开始，此命令不可用。）安装常规操作系统更新和程序包更新。详细信息取决于实例的操作系统。有关更多信息，请参阅 管理安全更新。

使用 Upgrade Operating System (升级操作系统) 命令将实例升级到新的 Amazon Linux 版本。

升级操作系统

（仅限 Linux）将实例的 Amazon Linux 操作系统升级到最新版本。有关更多信息，请参阅 AWS OpsWorks 堆栈操作系统。

> **Important**
>
> 建议您运行 Upgrade Operating System (升级操作系统) 后也运行 Setup (设置)。这将确保服务正确地重新启动。

堆栈命令具有下列选项，部分选项仅针对特定命令显示。

注释

(可选) 输入您要添加的任何自定义注释。

要执行的配方

(必需) 此设置仅当您选择 Execute Recipes 命令时显示。使用标准 *cookbook_name*::*recipe_name* 格式输入要执行的配方 (用逗号分隔)。如果您指定多个配方，AWS OpsWorks Stacks 会按列出的顺序执行它们。

允许重新启动

(可选) 此设置仅当您选择 Upgrade Operating System 命令时显示。默认值为 “是”，这会指示 AWS OpsWorks Stacks 在安装升级后重启实例。

自定义 Chef JSON

(可选) 选择 Advanced 以显示此选项，此选项允许您指定要合并到堆栈配置和部署属性中的自定义 JSON 属性。

实例

(可选) 指定要对其执行命令的实例。默认情况下，将选择所有联机实例。要对一部分实例运行此命令，请选择适当的层或实例。

> **Note**
>
> 您可能会看到在 Deployment 和 Commands 页上列出了您未运行的 execute_recipes 执行。这通常是由于权限更改 (例如为用户授予 SSH 权限或删除用户的 SSH 权限) 造成的。当你进行这样的更改时， AWS OpsWorks Stacks 会使用 execute_recipes 来更新实例的权限。

使用自定义 JSON

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

有几个 AWS OpsWorks Stacks 操作允许你指定自定义 JSON， AWS OpsWorks Stacks 会将其安装在实例上并可供配方使用。

您可以在以下情况下指定自定义 JSON：

- 当您创建、更新或克隆堆栈时。

  AWS OpsWorks Stacks 在所有实例上安装自定义 JSON，用于所有后续生命周期事件。
- 当您运行部署或堆栈命令时。

  AWS OpsWorks Stacks 仅将该事件的自定义 JSON 传递给实例。

自定义 JSON 必须由有效的 JSON 对象表示，并且必须格式化为有效的 JSON 对象。例如：

```
{
  "att1": "value1",
  "att2": "value2"
  ...
}
```

AWS OpsWorks 堆栈将自定义 JSON 存储在以下位置：

在 Linux 实例上：

- `/var/chef/runs/run-ID/attribs.json`
- `/var/chef/runs/run-ID/nodes/hostname.json`

在 Windows 实例上：

- *drive*:\chef\runs*run-ID*\attribs.json
- *drive*:\chef\runs*run-ID*\nodes*hostname*.json

> **Note**
>
> 在 Chef 11.10 和较早版本的 Linux 中，自定义 JSON 位于 Linux 实例的以下路径中，Windows 实例不可用，且没有 attribs.json 文件。日志存储在与 JSON 相同的文件夹或目录中。有关 Chef 11.10 和较早版本的 Linux 中的自定义 JSON 的信息，请参阅用自定义 JSON 覆盖属性和 Chef 日志。
> /var/lib/aws/opsworks/chef/*hostname*.json

在上述路径中，*run-ID* 是 AWS OpsWorks Stacks 分配给实例上每个 Chef 运行的唯一 ID，*hostname* 是实例的主机名。

要访问 Chef 配方的自定义 JSON，请使用标准 Chef node 语法。

例如，假设您想为您要部署的某个应用程序定义简单的设置，如该应用程序最初是否显示，以及该应用程序的初始前台和后台颜色。假设您使用如下所示的 JSON 对象来定义这些应用程序设置：

```
{
  "state": "visible",
  "colors": {
    "foreground": "light-blue",
    "background": "dark-gray"
  }
}
```

声明堆栈的自定义 JSON：

1. 在堆栈页面上，选择 Stack Settings，然后选择 Edit。
2. 对于 Custom Chef JSON，键入 JSON 对象，然后选择 Save。

> **Note**
>
> 您可以在部署、层和堆栈级别声明自定义 JSON。如果您希望某些自定义 JSON 仅对单个部署或层显示，则建议您执行此操作。或者，例如，建议您使用在层级别声明的自定义 JSON 来临时覆盖在堆栈级别声明的自定义 JSON。如果您在多个级别声明自定义 JSON，则在部署级别

声明的自定义 JSON 将覆盖同时在层级别和堆栈级别声明的任何自定义 JSON。在层级别声明的自定义 JSON 会覆盖只在堆栈级别声明的任何自定义 JSON。
要使用 AWS OpsWorks Stacks 控制台为部署指定自定义 JSON，请在部署应用程序页面上选择高级。在 Custom Chef JSON 框中键入自定义 JSON，然后选择 Save。
要使用 AWS OpsWorks Stacks 控制台为图层指定自定义 JSON，请在图层页面上为所需图层选择设置。在 Custom JSON 框中键入自定义 JSON，然后选择 Save。
有关更多信息，请参阅 编辑图 OpsWorks 层的配置 和 部署应用程序。

当您运行部署或堆栈命令时，配方可通过使用标准 Chef node 语法来检索这些自定义值，该语法会直接映射到自定义 JSON 对象中的层次结构。例如，以下配方代码会将关于上述自定义 JSON 值的消息写入 Chef 日志：

```
Chef::Log.info("********** The app's initial state is '#{node['state']}' **********")
Chef::Log.info("********** The app's initial foreground color is '#{node['colors']
['foreground']}' **********")
Chef::Log.info("********** The app's initial background color is '#{node['colors']
['background']}' **********")
```

这种方法对于将数据传递给配方很有用。 AWS OpsWorks Stacks 将这些数据添加到实例中，配方可以使用标准的 Chef node 语法来检索数据。

Note

自定义 JSON 限制为 120 KB。如果您需要更多容量，则建议您将部分数据存储在 Amazon Simple Storage Service (Amazon S3) 上。然后，您的自定义配方可以使用 AWS CLI 或 AWS SDK for Ruby 将 Amazon S3 存储桶中的数据下载到您的实例。

删除堆栈

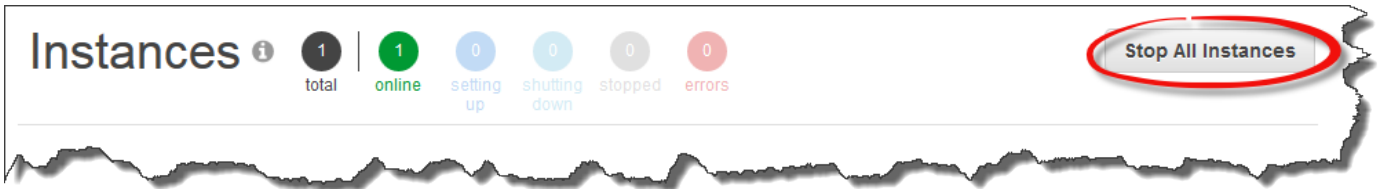
Important

该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

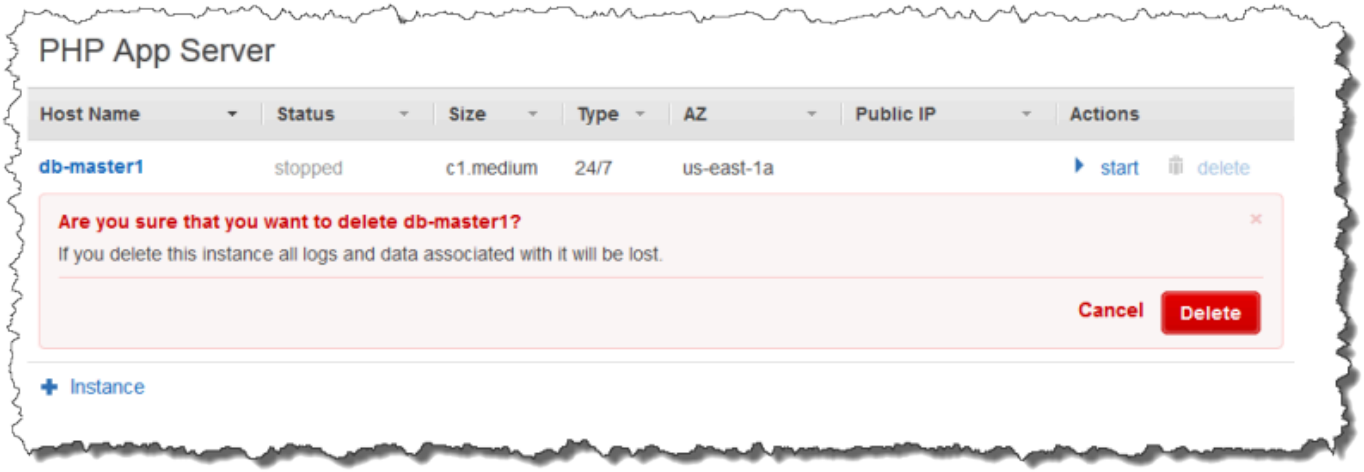
如果您不再需要某一堆栈，可将其删除。只能删除空堆栈；必须先删除该堆栈中的所有实例、应用程序和层。

删除堆栈

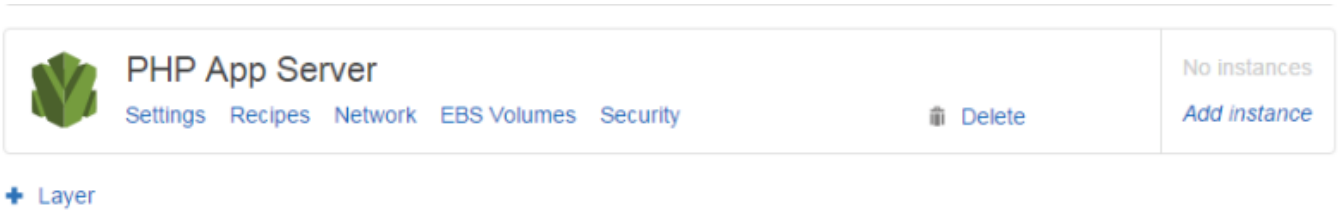
1. 在 AWS OpsWorks 堆栈控制面板上，选择要关闭并删除的堆栈。
2. 在导航窗格中，选择实例。
3. 在 Instances (实例) 页面上，选择 Stop all Instances (停止所有实例)。



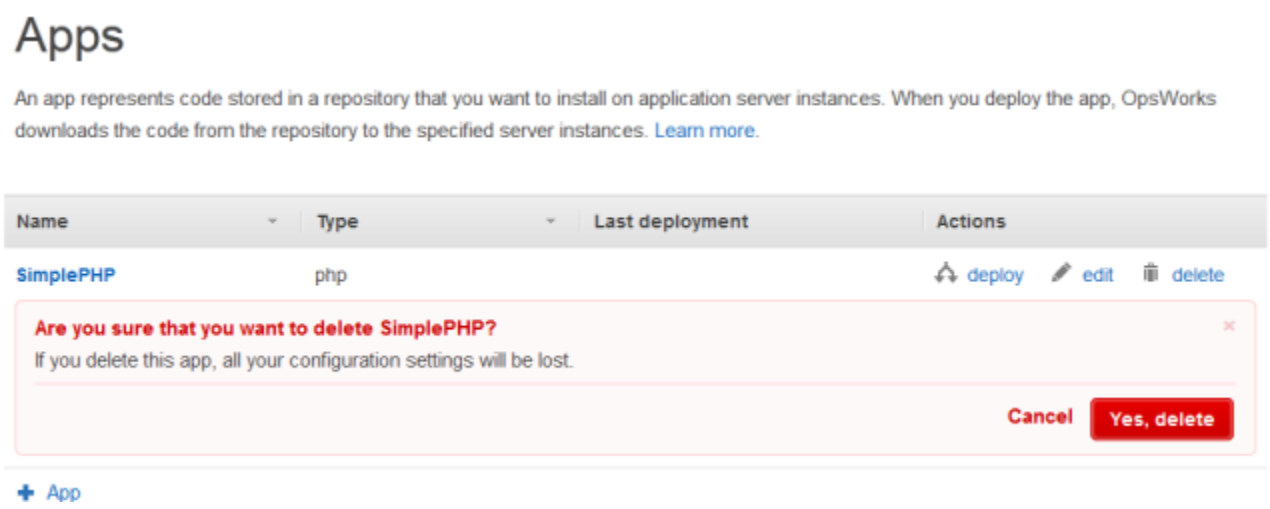
4. 在实例停止后，对该层中的每个实例，选择操作列中的删除。当系统提示进行确认时，选择 Yes, Delete (是，删除)。



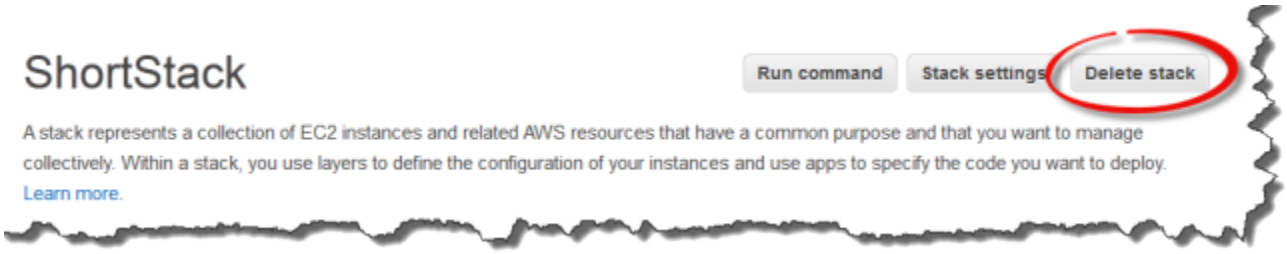
5. 在删除所有实例后，在导航窗格中选择 Layers (层)。
6. 在 Layers (层) 页面上，对堆栈中的每个层，选择 delete (删除)。在确认提示中，选择 Yes, Delete (是，删除)。



7. 在删除所有层后，在导航窗格中选择 Apps (应用程序)。
8. 在应用程序页面上，对堆栈中的每个应用程序，选择操作列中的删除。在确认提示中，选择 Yes, Delete (是，删除)。



9. 在删除所有应用程序后，在导航窗格中选择 Stack (堆栈)。
10. 在堆栈页面上，选择 Delete stack (删除堆栈)。在确认提示中，选择 Yes, Delete (是，删除)。



删除堆栈使用的其他 AWS 资源

您可以将其他 AWS 资源与 AWS OpsWorks Stacks 一起使用来创建和管理您的堆栈。删除堆栈时，如果其他堆栈未使用这些资源，而堆栈之外的资源未使用这些资源，则还应考虑删除与该堆 AWS OpsWorks 栈配合使用的资源。以下是清理您在堆栈中使用的外部 AWS 资源的建议理由。

- 外部 AWS 资源可能会继续在您的 AWS 账户中累积费用。
- Amazon S3 存储桶等资源可能包含个人可识别信息、敏感信息或机密信息。

> **Important**
>
> 如果这些资源正被其他堆栈使用，请勿将其删除。请注意，IAM 角色和安全组具有全局性，因此其他区域的堆栈可能正在使用这些相同的资源。

以下是堆栈使用的其他 AWS 资源，以及有关如何删除堆栈的信息的链接。

服务角色和实例配置文件

创建堆栈时，您需要指定 IAM 角色和实例配置文件， AWS OpsWorks Stacks 使用这些配置文件代表您创建允许的资源。 AWS OpsWorks 如果您不选择现有的角色和实例配置文件，则会为您创建角色和实例配置文件。为您 AWS OpsWorks 创建的角色和实例配置文件分别命名为 “aws-opsworks-service-role和aws-opsworks-ec2-role”。如果您的账户中没有其他堆栈在使用 IAM 角色和实例配置文件，则可安全地删除这些资源。有关如何删除 IAM 角色和实例配置文件的信息，请参阅《IAM 用户指南》中的删除角色或实例配置文件。

安全组

在 AWS OpsWorks Stacks 中，您可以在层级指定用户定义的安全组。您可以通过使用 Amazon EC2 控制台或 API 来创建安全组。其他区域的堆栈和层可以使用相同的安全组，因为安全组具有全局性。如果安全组未被其他 AWS 资源使用，则可以将其删除。有关如何删除安全组的更多信息，请参阅 Amazon EC2 用户指南中的删除安全组。

Amazon EBS 卷

在 AWS OpsWorks Stacks 中，您可以在层级别添加 EBS 卷，然后将它们连接到层中的实例。您可以使用 Amazon EC2 服务控制台或 API 创建 EBS 卷，然后在层级别将其附加到 AWS OpsWorks 堆栈实例。EBS 卷特定于可用区。如果在特定区域和可用区的任何堆栈中不再使用 EBS 卷，则可以删除该卷。有关如何删除 Amazon EBS 卷的更多信息，请参阅《Amazon EC2 用户指南》中的删除 Amazon EBS 卷。

Amazon Simple Storage Service (Amazon S3) 存储桶

在 AWS OpsWorks 堆栈中，您可以将 Amazon S3 存储桶用于以下用途。发送到 Amazon S3 存储桶的内容可能包含客户内容。有关删除敏感数据的更多信息，请参阅如何清空 S3 存储桶？或如何删除 S3 存储桶？。

- 存储应用程序代码
- 存储说明书和配方
- CloudTrail 日志，如果你启用了 AWS OpsWorks Stacks 中的 CloudTrail 登录功能
- Amaz CloudWatch on Logs 直播（ 如果您已在 AWS OpsWorks 堆栈中启用 ）

弹性 IP 地址

如果您在 AWS OpsWorks 堆栈中注册了弹性 IP 地址，并且不再需要弹性 IP 地址，则可以释放弹性 IP 地址。

Elastic Load Balancing 负载均衡器

如果不再需要此前一直对堆栈中的各层使用的 Elastic Load Balancing 经典负载均衡器，则可将其删除。有关更多信息，请参阅 Classic 负载均衡器用户指南中的删除负载均衡器。

Amazon Relational Database Service（Amazon RDS）实例

如果您在 AWS OpsWorks 堆栈中注册了 Amazon RDS 数据库 (DB) 实例，并且不再需要这些实例，则可以删除数据库实例。有关如何删除数据库实例的更多信息，请参阅《Amazon RDS 用户指南》中的删除数据库实例。

Amazon Elastic Container Service（Amazon ECS）集群

如果您的堆栈包含 ECS 集群层，但不再使用通过某一层注册的 ECS 集群，则可删除该 ECS 集群。有关如何删除 ECS 集群的更多信息，请参阅《Amazon ECS 开发人员指南》中的删除集群。

图层

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

每个堆栈包含一个或多个层，每个层代表一个堆栈组件，如负载均衡器或一组应用程序服务器。

在处理 AWS OpsWorks 堆栈图层时，请记住以下几点：

- 堆栈中的每个层必须至少具有一个实例并且可以选择具有多个实例。
- 堆栈中的每个实例必须是至少一个层的成员，但已注册的实例除外。

  除一些基本设置 (如 SSH 密钥和主机名) 之外，您无法直接配置实例。您必须创建和配置适当的层，然后将实例添加到该层。

Amazon EC2 实例可以是多个层中的成员（可选）。在这种情况下， AWS OpsWorks Stacks 会运行配方来为实例的每个层安装和配置软件包、部署应用程序等。

例如，通过将一个实例分配给多个层，您可执行以下操作：

- 通过在单个实例上托管数据库服务器和负载均衡器来减少开支。
- 使用您的一个应用程序服务器进行管理。

  创建一个自定义管理层并将应用程序服务器实例中的一个添加到该层。管理层的配方将配置该应用程序服务器实例以执行管理任务和安装任何其他必需软件。其他应用程序服务器实例仅仅是应用程序服务器。

本节描述如何使用层。

主题

OpsWorks 图层基础知识

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

本节介绍如何执行所有 AWS OpsWorks Stacks 图层通用的操作。

主题

创建图 OpsWorks 层

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

当您创建新的堆栈时，您将看到以下页面：

添加第一个 OpsWorks 图层

1. 单击 Add a Layer。
2. 在 Add Layer 页面上，选择适当的层，其中显示该层的配置选项。
3. 适当地配置该层，然后单击 Add Layer 以将其添加到堆栈中。以下部分介绍如何配置各种层。

   > **Note**
   >
   > Add Layer 页面仅显示每个层比较常用的配置设置。您可以通过编辑层来指定其他设置。

4. 将实例添加到层并启动实例。

   > **Note**
   >
   > 如果某个实例是多层的一个成员，则您必须将该实例添加到所有层，然后再启动该实例。您无法将一个联机实例添加到层。

要添加更多的层，请打开 Layers 页面，然后单击 + Layer 以打开 Add Layer 页面。

启动实例时， AWS OpsWorks Stacks 会自动为实例的每个层运行安装和部署配方，以安装和配置相应的软件包并部署相应的应用程序。您可以通过多种方式自定义图层的设置和配置过程，例如将自定义配方分配给相应的生命周期事件。 AWS OpsWorks Stacks 在每个事件的标准配方之后运行自定义食谱。有关更多信息，请参阅 说明书和诀窍。

以下特定于图层的部分描述了如何处理各个 AWS OpsWorks Stacks 图层的步骤 2 和 3。有关如何添加实例的更多信息，请参阅将实例添加到层。

编辑图 OpsWorks 层的配置

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

当您创建层后，一些属性 (如 Amazon Web Services Region) 是不可变的，但您可以随时更改大多数层配置。编辑层还可以提供对 Add Layer 页面上不可用的配置设置的访问权限。当您保存新配置后，这些设置便会立即生效。

编辑图 OpsWorks 层

1. 在导航窗格中，单击 Layers (层)。
2. 在 Layers 页面上，选择一个层名称以打开详细信息页面，该页面将显示当前配置。

   > **Note**
   >
   > 选择层名称下的一个名称，这会将您直接带到详细信息页面上的关联选项卡。

3. 单击 Edit，然后选择适当的选项卡：General Settings、Recipes、Network、EBS Volumes 或 Security。

以下部分介绍对所有层都可用的各种选项卡上的设置。某些层拥有其他特定于层的设置，这些设置显示在页面的顶部。此外，某些设置仅适用于基于 Linux 的堆栈，如下所述。

主题

- 常规设置
- 配方
- 网络
- EBS 卷
- 安全性
- CloudWatch 日志
- 标签

常规设置

所有层都具有以下设置：

自动修复已启用

是否为层的实例启用了自动修复。默认设置为 Yes。

自定义 JSON

针对此层中的所有实例，传递给您的 Chef 配方的 JSON 格式的数据。这个功能有多种用途，例如，您可以使用该功能将数据传递到您自己的配方中。有关更多信息，请参阅 使用自定义 JSON。

> **Note**
>
> 您可以在部署、层和堆栈级别声明自定义 JSON。如果您希望一些自定义 JSON 在整个堆栈中可见，或仅对某个部署可见，那么您可能希望执行上述操作。或者，例如，您可能希望暂时使用在部署级别声明的自定义 JSON 覆盖在层级别声明的自定义 JSON。如果您在多个级别声明自定义 JSON，则在部署级别声明的自定义 JSON 将覆盖同时在层级别和堆栈级别声明的任何自定义 JSON。在层级别声明的自定义 JSON 会覆盖只在堆栈级别声明的任何自定义 JSON。
>
> 要使用 AWS OpsWorks Stacks 控制台为部署指定自定义 JSON，请在部署应用程序页面上选择高级。在 Custom Chef JSON 框中键入自定义 JSON，然后选择 Save。
>
> 要使用堆栈控制台为 AWS OpsWorks 堆栈指定自定义 JSON，请在堆栈设置页面的“自定义 JSON”框中键入自定义 JSON，然后选择保存。
>
> 有关更多信息，请参阅 使用自定义 JSON 和 部署应用程序。

实例关闭超时

指定 AWS OpsWorks 堆栈在触发关闭生命周期事件后等待多长时间（以秒为单位），然后才会停止或终止 Amazon EC2 实例。默认设置为 120 秒。该设置的目的是为实例的 Shutdown 配方提供足够的时间来完成它们的任务，然后再终止该实例。如果您的自定义 Shutdown 配方可能需要更长的时间，相应地修改设置即可。有关实例关闭的更多信息，请参阅停止实例。

此选项卡上的其余设置因层的类型而异，但与层的 Add Layer 页面上的设置相同。

配方

Recipes 选项卡包括以下设置。

Custom Chef recipes

您可以将自定义 Chef 配方分配给层的生命周期事件。有关更多信息，请参阅 执行配方。

网络

Network 选项卡包括以下设置。

Elastic Load Balancing

您可以将 Elastic Load Balancing 负载均衡器连接到任何层。 AWS OpsWorks 然后，堆栈会自动向负载均衡器注册该层的在线实例，并在它们离线时将其注销。如果您启用了负载均衡器的连接耗尽功能，则可以指定 AWS OpsWorks Stacks 是否支持该功能。有关更多信息，请参阅 Elastic Load Balancing 层。

Automatically Assign IP Addresses

您可以控制 AWS OpsWorks Stacks 是自动为该层的实例分配公有 IP 地址还是弹性 IP 地址。当您启用此选项时，将发生以下情况：

- 例如，存储支持的实例， AWS OpsWorks Stacks 会在每次启动实例时自动分配地址。
- 对于由 Amazon EBS 支持的实例， AWS OpsWorks Stacks 会在实例首次启动时自动分配地址。
- 如果一个实例属于多个图层，如果您为至少一个图层启用了自动分配， AWS OpsWorks Stacks 会自动分配地址，

> **Note**
>
> 如果您启用公有 IP 地址的自动分配，则它仅适用于新实例。 AWS OpsWorks 堆栈无法更新现有实例的公有 IP 地址。

如果您的堆栈在 VPC 中运行，则公有 IP 地址与弹性 IP 地址分别使用不同的设置。下表说明了这些交互关系：

		Public IP addresses: Yes	Public IP addresses: No
Elastic IP addresses	Yes	Instances receive an Elastic IP address when they are started for the first time, or a public IP address if an Elastic IP cannot be assigned.	Instances receive an Elastic IP address when they are started for the first time.
	No	Instances receive a public IP address each time they are started.	Instances receive only a private IP address, which is not accessible from outside the VPC.

> **Note**
>
> 实例必须能够与 AWS OpsWorks Stacks 服务、Linux 软件包存储库和食谱存储库进行通信。如果您没有指定公有或弹性 IP 地址，则您的 VPC 必须包含一个允许层的实例与外部站点通信的组件，如 NAT。有关更多信息，请参阅 在 VPC 中运行堆栈。

如果您的堆栈未在 VPC 中运行，则 Elastic IP addresses 是您的唯一设置：

- Yes：实例在首次启动时获得弹性 IP 地址，或者，如果无法分配弹性 IP 地址，则获得公有 IP 地址。
- No：实例在每次启动都会获得公有 IP 地址。

EBS 卷

EBS Volumes 选项卡包括以下设置。

EBS 优化的实例

是否应针对Amazon Elastic Block Store (Amazon EBS) 优化层的实例。有关更多信息，请参阅 Amazon EBS 优化实例。

Additional EBS Volumes

(仅限 Linux) 您可以将 Amazon EBS 卷添加到层的实例或将其从层的实例中删除。当您启动实例时，AWS OpsWorks Stacks 会自动创建卷并将其连接到实例。您可以使用 Resources 页面管理堆栈的 EBS 卷。有关更多信息，请参阅 资源管理。

- 挂载点：（必填）指定 EBS 卷将挂载到的挂载点或目录。
- # 磁盘数：（可选）如果您指定了 RAID 阵列，此为阵列中的磁盘数。

  每个 RAID 级别都有一个默认的磁盘数，但您可以从列表中选择一个更大的数字。
- 总大小 (GiB)：（必填）卷的大小（以 GiB 为单位）。

  对于 RAID 阵列，此设置将指定总阵列大小，而不是每个磁盘的大小。

  下表显示了每个卷类型允许的最小和最大卷大小。

卷类型	最小大小 (GiB)	最大大小 (GiB)
磁介质	1	1024
预配置 IOPS (SSD)	4	16384
通用型 (SSD)	1	16384
吞吐量优化型 (HDD)	500	16384
Cold HDD	500	16384

- 卷类型：（可选）指定是创建磁性、通用型 SSD、吞吐量优化型 HDD、冷 HDD 还是 PIOPS 卷。

  默认值为 Magnetic。
- 加密：（可选）指定是否要加密 EBS 卷的内容。
- 每个磁盘的 IOPS：（对于预调配 IOPS SSD 和通用型 SSD 卷为必填）如果您指定预调配 IOPS SSD 或通用型 SSD 卷，则您还必须指定 每个磁盘的 IOPS。

  对于预配置 IOPS 卷，您可以在创建卷时指定 IOPS 速率。预配置的 IOPS 与请求的卷的大小比率最多为 30 (换言之，3000 IOPS 卷的大小至少为 100 GB)。通用型 (SSD) 卷有一个其大小为卷大小的三倍的基准 IOPS，最大为 10000 IOPS，它在 30 分钟内可突增至 3000 IOPS。

当您向层中添加卷或从层中删除卷时，请注意以下几点：

- 如果您添加卷，每个新实例都会获取新卷，但 AWS OpsWorks Stacks 不会更新现有实例。
- 如果您删除某个卷，它仅适用于新实例，而现有实例将保留其卷。

指定挂载点

您可以按照您的首选意愿指定任何挂载点。但是，请注意，有些挂载点已保留给 AWS OpsWorks Stacks 或 Amazon EC2 使用，不应用于亚马逊 EBS 卷。不要使用典型的 Linux 系统文件夹，例如 `/home` 或 `/etc`。

以下挂载点保留给 AWS OpsWorks Stacks 使用。

- `/srv/www`
- `/var/log/apache2` (Ubuntu)
- `/var/log/httpd` (Amazon Linux)
- `/var/log/mysql`
- `/var/www`

当实例启动或重启时，autofs (一种自动守护程序) 将使用临时设备挂载点 (如 `/media/ephemeral0`) 来进行绑定挂载。此操作发生在挂载 Amazon EBS 卷之前。为确保您的 Amazon EBS 卷的挂载点与 autofs 不冲突，请不要指定临时设备挂载点。可能的暂存设备挂载点取决于特定实例类型，以及它是由实例存储支持的还是由 Amazon EBS 支持的。为避免与 autofs 冲突，请执行以下操作：

- 验证适用于特定实例类型的临时设备挂载点和您要使用的后备存储。
- 请注意，如果您切换到 Amazon EBS 支持的实例，则适用于实例存储支持的实例的挂载点可能会与 autofs 冲突，反之亦然。

> **Note**
>
> 如果您希望更改实例存储块设备映射，您可以创建一个自定义 AMI。有关更多信息，请参阅 Amazon EC2 实例存储。有关如何为 AWS OpsWorks 堆栈创建自定义 AMI 的更多信息，请参阅使用自定义 AMI。

下面的示例说明了如何使用自定义配方来确保卷的挂载点不会与 autofs 冲突。您可以根据您的特定使用案例的需要来对其进行调整。

避免冲突的挂载点

1. 将 Amazon EBS 卷分配到所需的层，但使用永远不会与 autofs 冲突的挂载点，如 `/mnt/workspace`。
2. 实施以下自定义配方，该配方可在 Amazon EBS 卷上创建一个应用程序目录并从 `/srv/www/` 中与之链接。有关如何实施自定义配方的更多信息，请参阅说明书和诀窍和自定义堆栈 AWS OpsWorks。

```
mount_point = node['ebs']['raids']['/dev/md0']['mount_point'] rescue nil

if mount_point
  node[:deploy].each do |application, deploy|
    directory "#{mount_point}/#{application}" do
      owner deploy[:user]
      group deploy[:group]
      mode 0770
      recursive true
    end

    link "/srv/www/#{application}" do
      to "#{mount_point}/#{application}"
    end
  end
end
```

3. 在自定义说明书的 `depends 'deploy'` 文件中添加一个 `metadata.rb` 行。
4. 将此配方分配给层的 Setup 事件。

安全性

Security 选项卡包括以下设置。

安全组

层必须至少拥有一个关联的安全组。创建或更新堆栈时，您可以指定如何关联安全组。 AWS OpsWorks Stacks 提供了一组标准的内置安全组。

- 默认选项是让 AWS OpsWorks Stacks 自动将相应的内置安全组与每个层相关联。

- 您也可以选择不自动关联内置安全组，而是在创建层时将自定义安全组与各个层关联起来。

有关安全组的更多信息，请参阅使用安全组。

创建层后，您可以使用 Security Groups 将更多的安全组添加到层中，方法是从 Custom security groups 列表中选择这些安全组。将安全组添加到层后， AWS OpsWorks Stacks 会将其添加到所有新实例。（请注意，重新启动的实例存储实例将作为新实例启动，因此它们也将具有新的安全组。）AWS OpsWorks 堆栈不会向在线实例添加安全组。

您可以通过单击 x 删除现有安全组，如下所示：

- 如果您选择让 AWS OpsWorks Stacks 自动关联内置安全组，则可以通过单击 x 来删除之前添加的自定义安全组，但无法删除内置组。
- 如果您选择不自动关联内置安全组，您可以删除任何现有安全组 (包括原始安全组)，但层至少要保留一个组。

从层中移除安全组后， AWS OpsWorks Stacks 不会将其添加到任何新的或重启的实例中。 AWS OpsWorks 堆栈不会从在线实例中移除安全组。

> **Note**
>
> 如果您的堆栈在 VPC 中运行，则可以使用 Amazon EC2 控制台、API 或 CLI 为在线实例添加或删除安全组。但是，此安全组在 AWS OpsWorks Stacks 控制台中不可见。如果您希望移除此安全组，您还必须使用 Amazon EC2。有关更多信息，请参阅安全组。

请注意以下几点：

- 您无法通过添加限制性更强的安全组来限制内置安全组的端口访问设置。当有多个安全组时，Amazon EC2 会使用最宽松的设置。
- 您不应该修改内置安全组的配置。当您创建堆栈时，Stac AWS OpsWorks ks 会覆盖内置安全组的配置，因此您所做的任何更改都将在下次创建堆栈时丢失。

如果您发现自己需要为一个或多个层使用限制性更强的安全组设置，请执行以下步骤：

1. 创建拥有适当设置的自定义安全组，并将这些安全组添加到适当的层中。

   除了内置组，您堆栈中的每个层都必须至少拥有一个安全组，即使只有一个层需要自定义设置也是如此。

2. 编辑堆栈配置并将 “使用 OpsWorks安全组” 设置切换为 “否”。

AWS OpsWorks Stacks 会自动从每一层移除内置安全组。

有关安全组的更多信息，请参阅 Amazon EC2 安全组。

EC2 Instance Profile

您可以为层的实例更改 EC2 配置文件。有关更多信息，请参阅 为在 EC2 实例上运行的应用程序指定权限。

CloudWatch 日志

CloudWatch 日志选项卡允许您启用或禁用 Amazon CloudWatch 日志。 CloudWatch 日志集成适用于基于 Chef 11.10 和 Chef 12 Linux 的堆栈。有关启用 CloudWatch 日志集成和在日志控制台中指定要管理的 CloudWatch 日志的更多信息，请参阅使用带 AWS OpsWorks 堆栈的 Amazon CloudWatch 日志。

标签

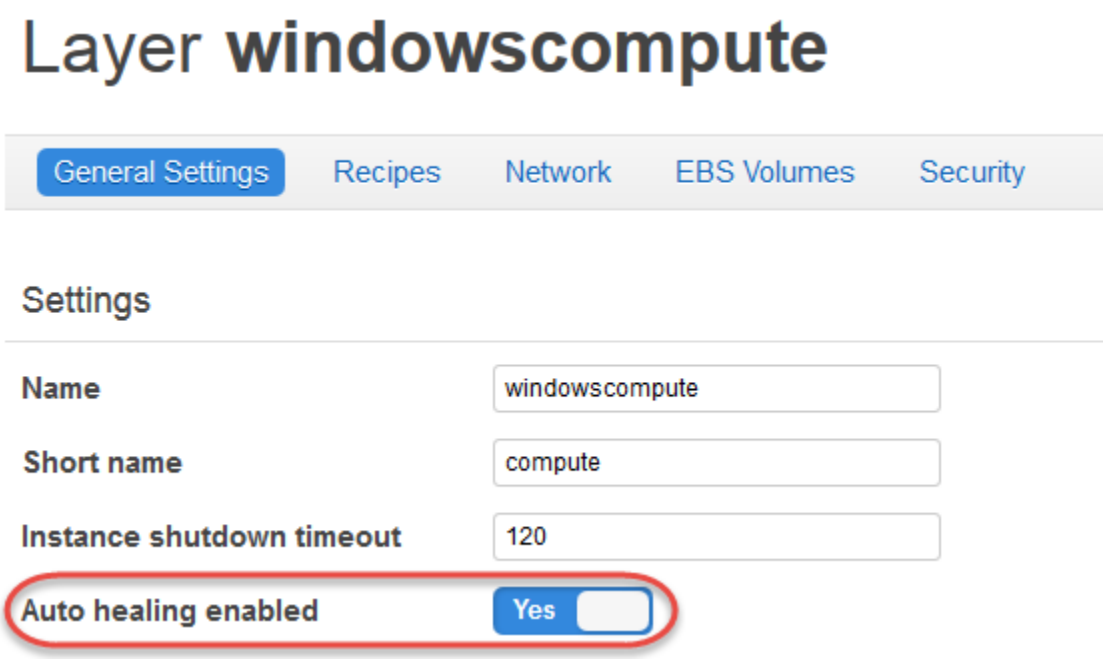
您可以使用 Tags 选项卡，对您的层应用成本分配标签。添加标签后，可以在 AWS Billing and Cost Management 控制台中激活它们。当您创建一个标签后，您将对标记的结构范围内的每个资源应用该标签。例如，如果您向某个层应用标签，该标签将应用于层中的每个实例、Amazon EBS 卷或 Elastic Load Balancing 负载均衡器。有关如何激活标签并使用它们来跟踪和管理 AWS OpsWorks 堆栈资源成本的更多信息，请参阅《B illing and Cost Management 用户指南》中的使用成本分配标签和激活用户定义的成本分配标签。有关在 AWS OpsWorks Stacks 中添加标签的更多信息，请参阅标签。

使用自动修复来更换失败的实例

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

每个实例都有一个 AWS OpsWorks Stacks 代理，可以定期与服务通信。 AWS OpsWorks Stacks 使用该通信来监控实例的运行状况。如果代理与服务通信的时间不超过大约五分钟， AWS OpsWorks Stacks 会认为该实例已失败。

自动修复是在层级别设置的；您可以通过编辑层设置来更改自动修复设置，如以下屏幕截图所示。



> **Note**
>
> 实例可能属于多个层中的某个层。如果其中任何一个层禁用了自动治疗，则在实例失败时，AWS OpsWorks Stacks 不会对其进行治疗。

如果图层启用了自动修复（默认设置），AWS OpsWorks Stacks 会自动替换该图层的失败实例，如下所示：

实例存储支持的实例

1. 停止 Amazon EC2 实例，并验证其是否已关闭。
2. 删除根卷上的数据。
3. 创建具有相同的主机名、配置和层成员关系的新 Amazon EC2 实例。
4. 重新连接任何 Amazon EBS 卷 (包括最初启动原有实例后连接的卷)。
5. 分配新的公有和私有 IP 地址。
6. 如果原有实例与弹性 IP 地址关联，则会将新的实例与相同的 IP 地址关联。

由 Amazon EBS 提供支持的实例

1. 停止 Amazon EC2 实例，并验证其是否已停止。
2. 启动 EC2 实例。

自动修复的实例恢复在线状态后，AWS OpsWorks Stacks 会在堆栈的所有实例上触发配置生命周期事件。关联的堆栈配置和部署属性包括实例的公有和私有 IP 地址。自定义配置配方可以从节点对象中获取新的 IP 地址。

如果您为层的实例指定 Amazon EBS 卷，AWS OpsWorks Stacks 会创建一个新卷，并在实例启动时将其连接到每个实例。如果您后来又想将卷与实例分离，可使用 Resources 页面。

当 AWS OpsWorks Stacks auto 治疗图层的其中一个实例时，它会通过以下方式处理体积：

- 如果在实例出现故障时卷已连接到实例，则会保存该卷及其数据，并且 AWS OpsWorks Stacks 会将其连接到新实例。
- 如果卷在实例失败时没有连接到实例，则 AWS OpsWorks Stacks 将创建一个新的空卷 (具有层指定的配置)，并将该卷连接到新的实例。

默认情况下为所有层启用了自动修复功能，但您可以编辑层的常规设置来禁用这项功能。

> **Important**
>
> 如果您启用了自动修复功能，请务必执行以下操作：
>
> - 仅使用 AWS OpsWorks Stacks 控制台、CLI 或 API 来停止实例。
>
> 如果您通过其他方式停止实例（例如，使用 Amazon EC2 控制台），则 AWS OpsWorks Stacks 会认为该实例已失败，并自动修复该实例。
> - 使用 Amazon EBS 卷存储您不想在自动修复实例后丢失的任何数据。
>
> 自动修复功能会停止原有的 Amazon EC2 实例，这会销毁没有存储到 Amazon EBS 卷上的任何数据。Amazon EBS 卷将重新连接到新的实例上，这将保留存储的所有数据。

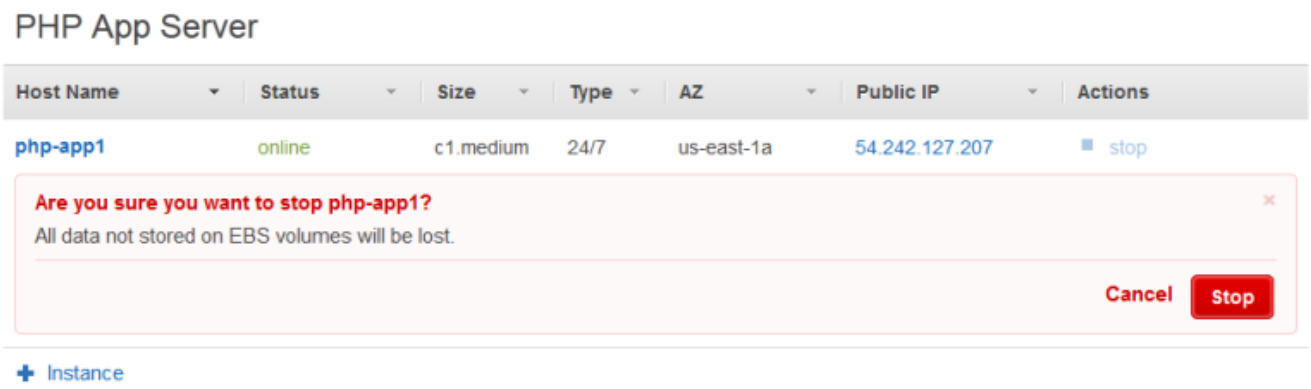
删除图 OpsWorks 层

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

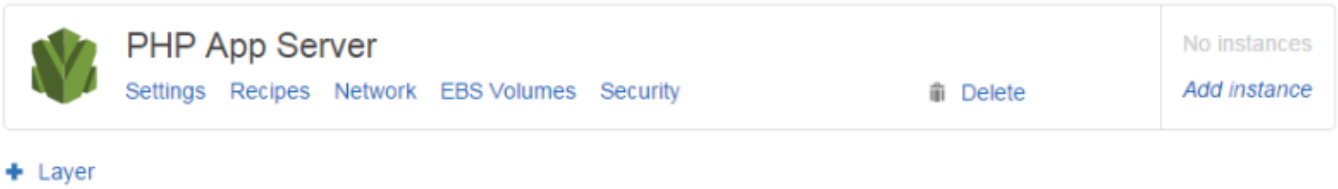
如果您不再需要 AWS OpsWorks Stacks 图层，可以将其从堆栈中删除。

删除图 OpsWorks 层

1. 在导航窗格中，单击 Instances。
2. 在 Instances 页面上，在您要删除的层名称下，单击每个实例的 Actions 列中的 stop。



3. 当每个实例停止后，单击 delete 以将其从层中删除。
4. 在导航窗格中，单击 Layers (层)。
5. 在 Layers 页面上，选择 Delete。



Elastic Load Balancing 层

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

Elastic Load Balancing 的工作原理与 AWS OpsWorks Stacks 层略有不同。原来的做法是创建一个层然后在其中添加实例，现在是使用 Elastic Load Balancing 控制台或 API 创建一个负载均衡器，然后将它挂载到现有层。除了将流量分发到层的实例之外，Elastic Load Balancing 将执行以下操作：

- 检测运行不正常的 Amazon EC2 实例并将流量重新路由至其他运行正常的实例，直到运行不正常的实例恢复。
- 自动扩展请求处理容量来响应传入流量。
- 如果您启用连接耗尽，负载均衡器会停止将新请求发送给运行不正常或即将注销的实例，但会保持连接处于活动状态直至指定的超时值，从而允许实例完成任何已在发送途中的请求。

将负载均衡器连接到层后， AWS OpsWorks Stacks 会执行以下操作：

- 注销当前已注册的任何实例。
- 在该层的实例（包括基于负载和基于时间的实例）联机时自动将其注册，并在它们脱机时自动将其注销。
- 自动开始将请求路由到可用区中的注册实例。

如果您启用了负载均衡器的连接耗尽功能，则可以指定 AWS OpsWorks Stacks 是否支持该功能。如果您启用连接耗尽支持（默认设置），则在实例关闭后， AWS OpsWorks Stacks 会执行以下操作：

- 从负载均衡器注销实例。

  负载均衡器将停止发送新请求并开始连接耗尽。
- 延迟触发关闭生命周期事件，直至负载均衡器完成连接耗尽。

如果您不启用连接耗尽支持， AWS OpsWorks Stacks 会在实例关闭后立即触发 Shutdown 事件，即使该实例仍连接到负载均衡器也是如此。

要对堆栈使用 Elastic Load Balancing，您必须先使用 Elastic Load Balancing 控制台、CLI 或 API 在同一区域中创建一个或多个负载均衡器。您应了解以下事项：

- 您只能将一个负载均衡器挂载到一个层。
- 每个负载均衡器只能处理一个层。
- AWS OpsWorks 堆栈不支持 Application Load Balancer。您只能将 Classic Load Balancer 与 AWS OpsWorks Stacks 配合使用。

这意味着，您必须为要平衡的每个堆栈中的每个层创建一个单独的 Elastic Load Balancing 负载均衡器，并且仅将其用于该用途。建议的做法是为计划与 AWS OpsWorks 堆栈一起使用的每个 Elastic Load Balancing 负载均衡器分配一个独特的名称，例如 MyStack 1-RailsLayer-ELB，以避免将负载均衡器用于多个目的。

> **Important**
>
> 我们建议为您的 AWS OpsWorks Stacks 层创建新的 Elastic Load Balancing 负载均衡器。如果您选择使用现有的 Elastic Load Balancing 负载均衡器，应先确认它当前未用于其他用途并且没有挂载实例。将负载均衡器连接到层后， OpsWorks 移除所有现有实例，并将负载均衡器配置为仅处理该层的实例。从技术上来说，在将某个负载均衡器挂载到层之后使用 Elastic Load Balancing 控制台或 API 来修改它的配置尽管是可行的，但您不应如此操作；更改将不会是永久的。

将 Elastic Load Balancing 负载均衡器附加到层

1. 如果您尚未这样操作，请使用 Elastic Load Balancing 控制台、API 或 CLI 在堆栈的区域中创建一个负载均衡器。当您创建负载均衡器时，请执行以下操作：

   - 请务必指定适用于您的应用程序的运行状况检查 Ping 路径。

     默认 Ping 路径为 `/index.html`，因此如果您的应用程序根目录不包括 `index.html`，则必须指定适当的 ping 路径，否则运行状况检查将失败。
   - 如果您要使用连接耗尽，请确保此功能已启用并且具有适当的超时值。

   有关更多信息，请参阅 Elastic Load Balancing。
2. 创建层 (您希望已平衡的层) 或编辑现有层的网络设置。

   > **Note**
   >
   > 您无法在创建自定义层时挂载负载均衡器。您必须编辑层的设置。
3. 在 Elastic L oad Balancing 下，选择要连接到该层的负载均衡器，并指定是否希望 AWS OpsWorks Stacks 支持连接耗尽。

将负载均衡器附加到层后， AWS OpsWorks Stacks 会在堆栈的实例上触发配置生命周期事件，以将更改通知它们。 AWS OpsWorks 当您分离负载均衡器时，堆栈还会触发配置事件。

> **Note**
>
> 实例启动后， AWS OpsWorks Stacks 会运行安装和部署配方，用于安装软件包和部署应用程序。这些配方完成后，实例将处于联机状态， AWS OpsWorks Stacks 将该实例注册到 Elastic

Load Balancing。 AWS OpsWorks 实例上线后，堆栈还会触发配置事件。这意味着，Elastic Load Balancing 注册和配置配方可并发运行，并且可在配置配方完成之前注册实例。要确保配方在向 Elastic Load Balancing 注册实例之前完成，您应将配方添加到层的设置或部署生命周期事件。有关更多信息，请参阅 执行配方。

从负载均衡器中删除实例有时也很有用。例如，当您更新一个应用程序时，建议您将该应用程序部署到单个实例并确认该应用程序在部署到每个实例之前正常运行。您通常会从负载均衡器删除该实例，因此该实例在您验证更新之前不会收到用户请求。

您必须使用 Elastic Load Balancing 控制台或 API 来从负载均衡器中暂时删除联机实例。下面介绍了如何使用此控制台。

从负载均衡器暂时删除实例

1. 打开 Amazon EC2 控制台并选择负载均衡器。
2. 选择适当的负载均衡器并打开 Instances 选项卡。
3. 在实例的 Actions 列中选择 Remove from Load Balancer。
4. 完成后，选择 Edit Instances，并将实例返回到负载均衡器。

Important

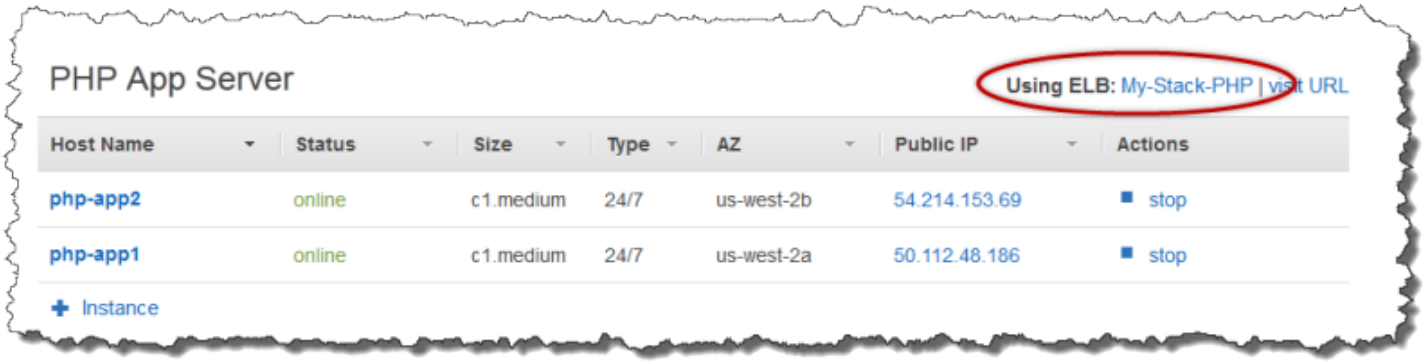
如果您使用 Elastic Load Balancing 控制台或 API 从负载均衡器中移除实例，则还必须使用 Elastic Load Balancing 将其放回。 AWS OpsWorks Stacks 不知道您使用其他服务控制台或 API 执行的操作，也不会为您将实例返回到负载均衡器。有时， AWS OpsWorks Stacks 可以将实例添加回 ELB，但这并不能保证行为，也不是在所有情况下都会发生。

您可将多个负载均衡器挂载到一组特定的实例，如下所示：

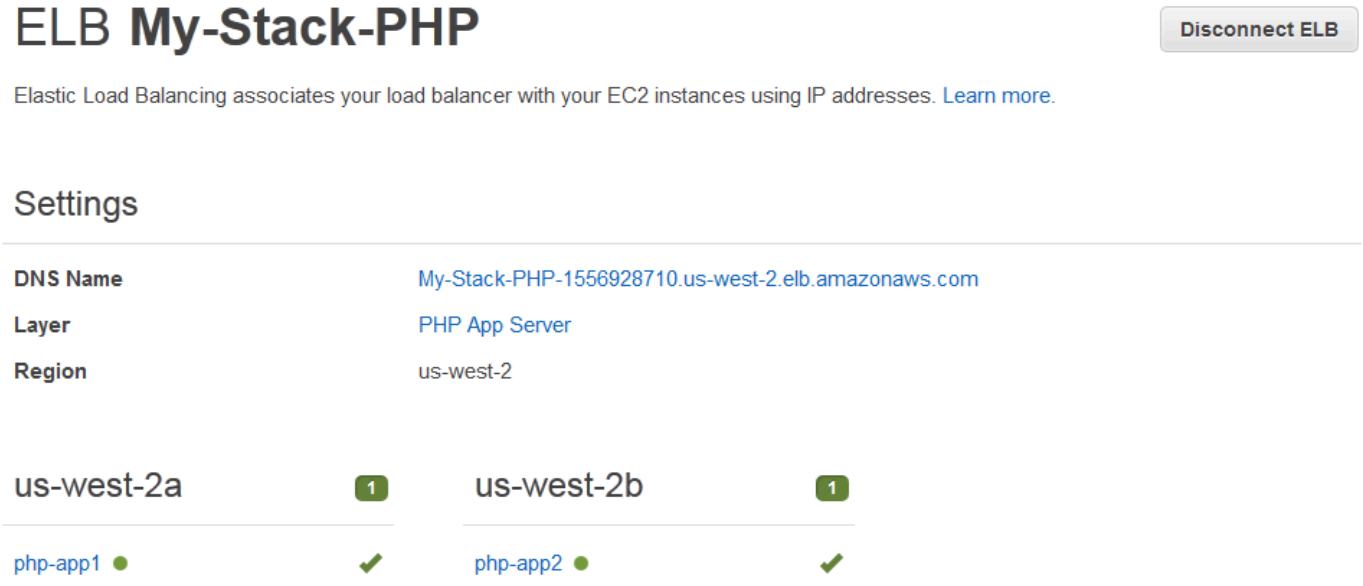
挂载多个负载均衡器

1. 使用 Elastic Load Balancing 控制台、API 或 CLI 创建一组负载均衡器。
2. 为每个负载均衡器创建一个自定义层并将其中一个负载均衡器挂载到该层。您无需为这些层实施任何自定义配方；默认自定义层已足够。
3. 为每个自定义层添加一组实例。

您可以转到“Instances”页并单击相应的负载均衡器名称，从而检查负载均衡器的属性。



ELB 页显示负载均衡器的基本属性，包括其 DNS 名称和关联实例的运行状况。如果堆栈在 VPC 中运行，则此页将显示子网而不是可用区。绿色对勾符号指示运行正常的实例。您可单击相应的名称来通过负载均衡器连接到某个服务器。



Amazon RDS 服务层

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

一个 Amazon RDS 服务层表示一个 Amazon RDS 实例。该层可以仅表示您必须使用 Amazon RDS 控制台或 API 单独创建的现有 Amazon RDS 实例。

将 Amazon RDS 服务层包含到堆栈中的基本过程如下所示：

1. 使用 Amazon RDS 控制台、API 或 CLI 创建实例。

   务必记录实例的 ID、主用户名称、主密码和数据库名称。
2. 要将 Amazon RDS 层添加到堆栈，请将 Amazon RDS 实例注册到堆栈。
3. 将该层添加到应用程序，这会将 Amazon RDS 实例的连接信息添加到应用程序的 `deploy` 属性。
4. 使用语言特定文件或 `deploy` 属性中的信息将应用程序连接到 Amazon RDS 实例。

   有关如何将应用程序连接到数据库服务器的更多信息，请参阅the section called “连接到数据库”

> **Warning**
>
> 请确保实例的主密码和用户名中的字符与应用程序服务器兼容。例如，对于 Java 应用程序服务器层，在任一字符串中包含 & 将导致阻止 Tomcat 服务器启动的 XML 分析错误。

主题

- 指定安全组
- 将 Amazon RDS 实例注册到 Stack
- 将 Amazon RDS 服务层与应用程序关联
- 从堆栈中删除 Amazon RDS 服务层

指定安全组

要使用带有 AWS OpsWorks 堆栈的 Amazon RDS 实例，数据库或 VPC 安全组必须允许从相应的 IP 地址进行访问。对于生产使用，安全组通常将访问对象限制为需要访问数据库的 IP 地址。它通常包括用于管理数据库的系统的地址以及需要访问数据库的 AWS OpsWorks Stacks 实例的地址。 AWS OpsWorks 当您在某个地区创建第一个堆栈时，堆栈会自动为每种类型的层创建一个 Amazon EC2 安全组。为 AWS OpsWorks 堆栈实例提供访问权限的一种简单方法是向 Amazon RDS 实例或 VPC 分配相应的 AWS OpsWorks 堆栈安全组。

为现有 Amazon RDS 实例指定安全组

1. 通过以下网址打开 Amazon RDS 控制台：https://console.aws.amazon.com/rds/。
2. 在导航窗格中，单击实例，然后选择适当的 Amazon RDS 实例。单击 Instance Actions 和 Modify。
3. 从 Security Group 列表中选择以下安全组，然后单击 Continue 和 Modify DB Instance 以更新实例。

   - AWS OpsWorks-db-Master-Server（security_group_id）安全组#
   - 其实例将连接到数据库的应用程序服务器层的安全组。组名称包含层名称。例如，要提供对 PHP 应用服务器实例的数据库访问权限，请指定 AWS--PHP-App OpsWorks-Server 组。

如果您要创建新的 Amazon RDS 实例，则可以在启动数据库实例向导的 “配置高级设置” 页面上指定相应的 AWS OpsWorks 堆栈安全组。有关如何使用此向导的说明，请参阅创建 MySQL 数据库实例并连接到 MySQL 数据库实例上的数据库。

有关如何指定 VPC 安全组的信息，请参阅您的 VPC 的安全组。

将 Amazon RDS 实例注册到 Stack

要在堆栈中添加 Amazon RDS 服务层，您必须将一个实例注册到堆栈。

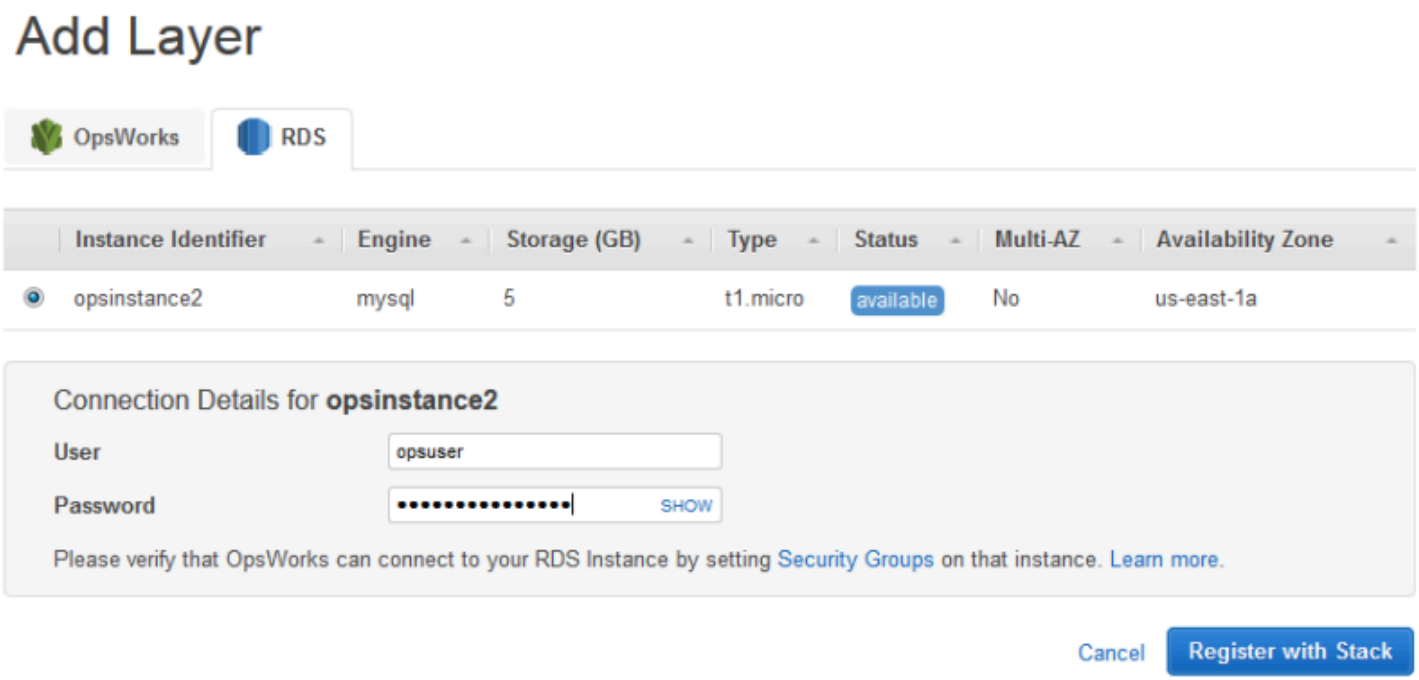
将 Amazon RDS 实例注册到堆栈

1. 在 AWS OpsWorks Stacks 控制台中，单击导航窗格中的图层，单击 + 图层或添加图层以打开添加层页面，然后单击 RDS 选项卡。
2. 必要时更新堆栈的服务角色，如更新堆栈的服务角色中所述。
3. 单击 RDS 选项卡以列出可用的 Amazon RDS 实例。

   > **Note**
   >
   > 如果您的账户没有任何 Amazon RDS 实例，您可通过单击“RDS”选项卡上的添加 RDS 实例创建一个，这会让您转到 Amazon RDS 控制台并启动启动数据库实例向导。您还可直接转到 Amazon RDS 控制台并单击启动数据库实例，或者使用 Amazon RDS API 或 CLI。有关如何创建 Amazon RDS 实例的更多信息，请参阅《Amazon RDS 入门指南》。

4. 选择合适的实例，将 User 和 Password 设置为合适的用户和密码值，然后单击 Register to Stack。

> **Important**
>
> 您必须确保您用于注册 Amazon RDS 实例的用户和密码与有效的用户和密码对应。如果它们不对应，您的应用程序将无法连接到该实例。但是，您可编辑层以提供有效的用户和密码值，然后重新部署应用程序。



当您将一个 Amazon RDS 服务层添加到堆栈时，Stac AWS OpsWorks ks 会为其分配一个 ID，并将关联的 Amazon RDS 配置添加到堆栈配置和部署属性的属性中`[:opsworks][:stack]`。

> **Note**
>
> 如果您更改已注册的 Amazon RDS 实例的密码，则必须手动更新 AWS OpsWorks 堆栈中的密码，然后重新部署应用程序以更新堆栈实例上的堆栈配置和部署属性。

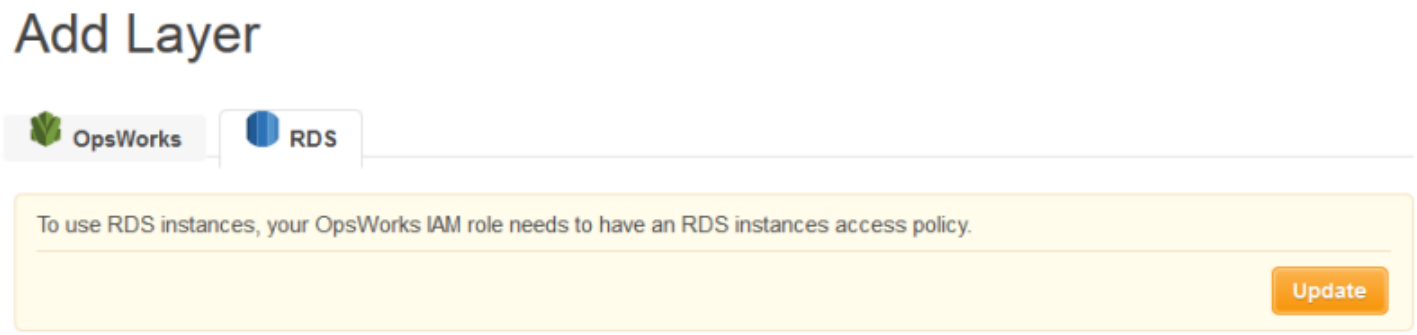
主题

- 更新堆栈的服务角色

更新堆栈的服务角色

每个堆栈都有一个 IAM 服务角色，用于指定 AWS OpsWorks 堆栈可以代表您使用其他 AWS 服务执行哪些操作。要向堆栈注册 Amazon RDS 实例，其服务角色必须向 AWS OpsWorks 堆栈授予访问 Amazon RDS 的权限。

首次将 Amazon RDS 服务层添加到您的堆栈之一时，服务角色可能缺少必需的权限。如果是这样的话，当您单击 Add Layer 页上的 RDS 选项卡时，您将看到以下内容。



单击 “更新”，让 AWS OpsWorks Stacks 将服务角色的策略更新为以下内容。

```
{"Statement": [{"Action": ["ec2:*", "iam:PassRole",
                        "cloudwatch:GetMetricStatistics",
                        "elasticloadbalancing:*",
                        "rds:*"],
               "Effect": "Allow",
               "Resource": ["*"] }]
}
```

> **Note**
>
> 您只需要执行一次更新。之后所有堆栈将自动使用更新后的角色。

将 Amazon RDS 服务层与应用程序关联

在添加 Amazon RDS 服务层之后，可将其与应用程序关联。

- 您可在创建应用程序时将 Amazon RDS 层与应用程序关联，也可在之后通过编辑应用程序的配置达到此目的。

- 要取消 Amazon RDS 层与应用程序的关联，请编辑应用程序的配置以指定其他数据库服务器或不指定服务器。

  Amazon RDS 层仍是堆栈的一部分，并且可与其他应用程序关联。

将 Amazon RDS 实例与应用程序关联后， AWS OpsWorks Stacks 会将数据库连接信息放在应用程序的服务器上。每个服务器实例上的应用程序随后可使用此信息连接到数据库。有关如何连接到 Amazon RDS 实例的更多信息，请参阅 the section called “连接到数据库”。

从堆栈中删除 Amazon RDS 服务层

要从堆栈中删除 Amazon RDS，请将其取消注册。

取消注册 Amazon RDS 服务层

1. 单击导航窗格中的层并单击 Amazon RDS服务层的名称。
2. 单击 Deregister 并确认您要取消注册该层。

此过程将从堆栈中删除该层，但它不会删除基础 Amazon RDS 实例。该实例和任何数据库将保留在您的账户中并且可注册到其他堆栈。您必须使用 Amazon RDS 控制台、API 或 CLI 删除该实例。有关更多信息，请参阅删除数据库实例。

ECS 集群层

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

Amazon Elastic Container Service (Amazon ECS) 管理名为容器实例的Amazon Elastic Compute Cloud (Amazon EC2) 实例集群上的 Docker 容器。ECS 集群层表示 Amazon ECS 集群，并通过提供相关功能来简化集群管理，这些功能包括：

- 简化的容器实例预配置和管理
- 容器实例操作系统和软件包更新
- 用户权限管理

- 容器实例性能监控
- Amazon Elastic Block Store（Amazon EBS）卷管理
- 公有和弹性 IP 地址管理
- 安全组管理

ECS 集群层具有以下限制和要求：

- 层仅适用于在 VPC（包括默认 VPC）中运行的 Chef 11.10 或 Chef 12 Linux 堆栈。
- 层的实例必须运行以下操作系统之一。
  - Amazon Linux 2
  - Amazon Linux 2018.03
  - Amazon Linux 2017.09
  - Amazon Linux 2017.03
  - Amazon Linux 2016.09
  - Amazon Linux 2016.03
  - Amazon Linux 2015.09
  - Amazon Linux 2015.03
  - Ubuntu 18.04 LTS
  - Ubuntu 16.04 LTS
  - Ubuntu 14.04 LTS
  - 自定义
- 层的实例上的 AWS OpsWorks Stacks 代理版本必须为 3425-20150727112318 或更高版本。

主题

向堆栈添加 ECS 集群层

AWS OpsWorks 堆栈简化了为现有 Amazon ECS 集群启动和维护容器实例的过程。要创建或启动其他 Amazon ECS 实体（如集群和任务），请使用 Amazon ECS 控制台、命令行界面 (CLI) 或 API。

（有关更多信息，请参阅《Amazon Elastic Container Service 开发人员指南》。）然后，您可以通过创建 ECS 集群层将群集与堆栈关联起来，您可以使用该层管理 AWS OpsWorks 堆栈中的群集。

您可以将集群与堆栈关联，如下所示：

- 每个堆栈均可具有一个 ECS 集群层，它表示一个集群。
- 一个集群只能与一个堆栈关联。

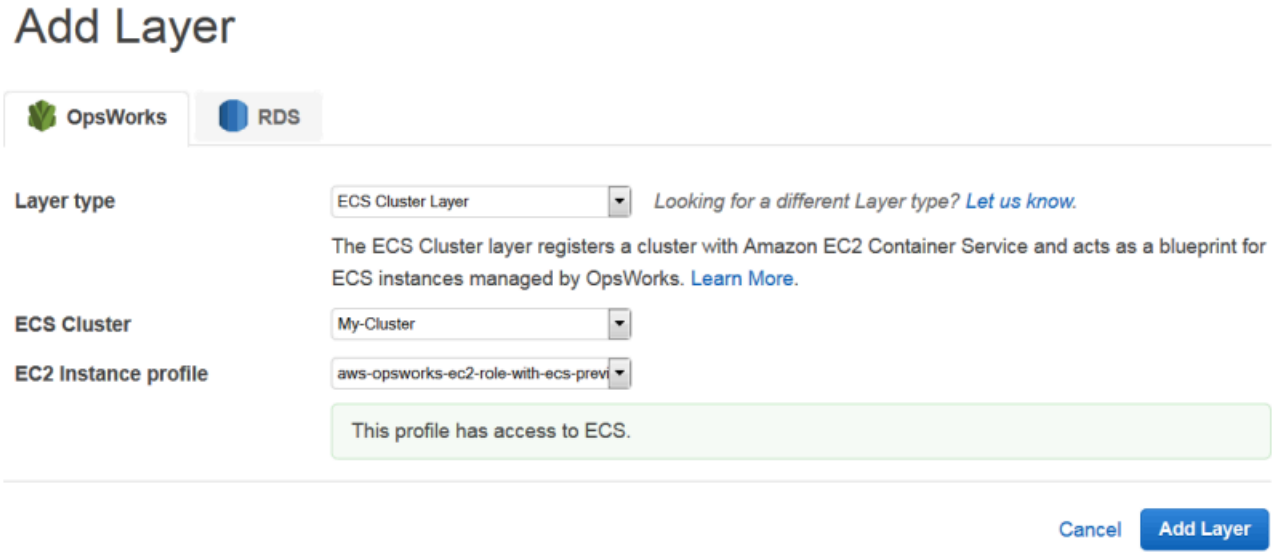
在将 ECS 集群层添加到堆栈之前，必须更新 AWS OpsWorks 堆栈 AWS Identity and Access Management (IAM) 服务角色（通常命名为）`aws-opsworks-service-role`，以允许 AWS OpsWorks 堆栈代表您与 Amazon ECS 进行交互。有关服务角色的更多信息，请参阅允许 AWS OpsWorks Stacks 代表你行事。

首次创建 ECS 集群层时，控制台会提供一个更新按钮，您可以选择该按钮指示 AWS OpsWorks Stacks 为您更新角色。 AWS OpsWorks 然后，Stacks 会显示 “添加图层” 页面，以便您可以将图层添加到堆栈中。您只需更新服务角色一次。随后，您可以使用更新后的角色将 ECS 集群层添加到任意堆栈。

> **Note**
>
> 如果您愿意，可以通过向现有策略添加 `ecs:*` 权限来手动更新服务角色的策略，如下所示：

```
{
  "Statement": [
    {
      "Action": [
        "ec2:*",
        "iam:PassRole",
        "cloudwatch:GetMetricStatistics",
        "elasticloadbalancing:*",
        "rds:*",
        "ecs:*"
      ],
      "Effect": "Allow",
      "Resource": ["*"]
    }
  ]
}
```

将集群与堆栈关联需要执行两项操作：将集群注册到堆栈，然后创建关联的层。 AWS OpsWorks Stacks 控制台结合了这些步骤；图层创建会自动注册指定的集群。如果您使用 AWS OpsWorks Stacks API、CLI 或 SDK，则必须使用单独的操作来注册集群并创建关联层。要使用控制台将 ECS 集群层添加到堆栈，请选择层，再选择+层或添加层，然后选择 ECS 集群层类型。



Add Layer 页面包括以下配置选项：

ECS 集群

要注册到堆栈的 Amazon ECS 集群。

EC2 实例配置文件

集群的 Amazon Elastic Compute Cloud（Amazon EC2）实例配置文件。此配置文件向正在集群的容器实例上运行的应用程序授予对其他 Amazon Web Service (包括 Amazon ECS) 的访问权。创建第一个 ECS 集群层时，选择新建配置文件并允许 ECS 访问直接 AWS OpsWorks 堆栈来创建所需的配置文件，该配置文件名`aws-opsworks-ec2-role-with-ecs`为。随后，您可以对所有后续 ECS 集群层使用该配置文件。有关实例配置文件的更多信息，请参阅为在 EC2 实例上运行的应用程序指定权限。

您可以通过编辑层的配置来指定其他设置，包括：

- 将 Elastic Load Balancing 负载均衡器连接到任何层。

  此方法可能适合某些用例，但 Amazon ECS 提供了更高级的选项。有关更多信息，请参阅服务负载均衡。
- 指定是否自动向容器实例分配公有 IP 地址或弹性 IP 地址。

如果您禁用针对两个地址类型的自动分配，则除非子网已正确配置 NAT，否则该实例将无法联机。有关更多信息，请参阅 在 VPC 中运行堆栈。

管理 ECS 集群

创建 ECS 集群层后，您可以使用 AWS OpsWorks 堆栈管理集群，如下所示：

预配置和管理容器实例

最初，ECS 集群层不包括任何容器实例，即使原始集群包含容器实例也是如此。一个选择是使用以下项的适当组合来管理层的实例：

- 手动向层添加全天候实例，并在不再需要这些实例时将其删除。
- 通过向层添加基于时间的实例来按计划添加或删除实例。
- 通过向层中添加基于负载的实例，根据 AWS OpsWorks Stacks 主机指标或 CloudWatch 警报添加或删除实例。

> **Note**
>
> 如果堆栈的默认操作系统不支持 Amazon ECS，则当创建容器实例时,必须明确指定支持的操作系统-Amazon Linux 2、Amazon Linux 2018.03、Amazon Linux 2017.09、Amazon Linux 2017.03、Amazon Linux 2016.09、Amazon Linux 2016.03、Amazon Linux 2015.09、Amazon Linux 2015.03、Ubuntu 18.04 LTS、Ubuntu 16.04 LTS、Ubuntu 14.04 LTS 或 Custom。请勿使用 ECS 优化的 AMI 在 ECS 层中创建实例，因为此 AMI 已经包含 ECS 代理。 AWS OpsWorks Stacks 还会在实例设置过程中尝试安装 ECS 代理，冲突可能导致安装失败。

有关更多信息，请参阅优化服务器数。 AWS OpsWorks 堆栈将 A WS OpsWorks-ECS-Cluster 安全组分配给每个实例。在每个新实例完成启动后， AWS OpsWorks Stacks 会通过安装 Docker 和 Amazon ECS 代理，然后在集群中注册该实例，将其转换为容器实例。

如果您更愿意使用现有的容器实例，您可以将其注册到堆栈并将其分配给 ECS 集群层。请注意，实例必须运行支持的操作系统、Amazon Linux 2015.03 或更高版本或 Ubuntu 14.04 LTS 或更高版本。

> **Note**
>
> 一个容器实例不能同时属于一个 ECS 集群层和另一个内置层。但是，容器实例可以属于一个 ECS 集群层和一个或多个自定义层。

运行操作系统和软件包更新

在新实例完成启动后， AWS OpsWorks Stacks 会安装最新的更新。然后，您可以使用 AWS OpsWorks Stacks 使容器实例保持最新状态。有关更多信息，请参阅 管理安全更新。

管理用户权限

AWS OpsWorks Stacks 提供了一种管理容器实例权限的简单方法，包括管理用户的 SSH 密钥。有关更多信息，请参阅 管理用户权限 和 管理 SSH 访问。

监控性能指标

AWS OpsWorks Stacks 提供了多种方法来监控堆栈、层或单个实例的性能指标。有关更多信息，请参阅 监控。

您通过 Amazon ECS 处理其他管理任务，例如创建任务或服务。有关更多信息，请参阅 Amazon Elastic Container Service 开发人员指南。

> **Note**
>
> 要直接转到 Amazon ECS 控制台上的集群页，请选择实例，然后选择 ECS 集群(位于 ECS 集群层的部分的右上角附近)。

从堆栈中删除 ECS 集群层

当您不再需要集群时，请删除 ECS 集群层并取消注册关联的集群。从堆栈中删除集群需要执行两项操作：取消注册集群，然后删除关联的层。 AWS OpsWorks Stacks 控制台结合了这些步骤；删除图层会自动注销指定集群的注册。如果您使用 AWS OpsWorks Stacks API、CLI 或 SDK，则必须使用单独的操作来注销集群并删除关联的层。

使用控制台删除 ECS 集群层

1. 如果您要控制任务的关闭方式，请使用 Amazon ECS 控制台、API 或 CLI 来缩减和删除集群的服务。有关更多信息，请参阅清除您的 Amazon ECS 资源。

2. 停止层的实例，然后将其删除。当您停止容器实例时， AWS OpsWorks Stacks 会自动停止所有正在运行的任务，从集群中注销该实例，并终止该实例。

   > **Note**
   >
   > 如果您具有与堆栈关联的现有容器实例，您可以从层取消分配实例，然后取消注册实例，这会将实例返回到 ECS 控制。

3. 删除图层。 AWS OpsWorks Stacks 会取消注册关联的集群，但不会将其删除。该集群将保留在 Amazon ECS 中。

自定义 AWS OpsWorks 堆栈图层

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

自定义层只有最少的一组配方。您随后可通过实施自定义配方并将它们分配给层的生命周期事件来将相应的功能添加到层。

自定义层具有以下配置设置。

> **Note**
>
> AWS OpsWorks 堆栈会自动在层的实例上安装 Ruby。如果您要在实例上运行 Ruby 代码但不想使用默认 Ruby 版本，则可使用自定义 JSON 或自定义属性文件来指定您的首选版本。有关更多信息，请参阅 Ruby 版本。

创建自定义层的基本过程包含以下步骤：

1. 实施说明书，其中包含安装和配置程序包、处理配置更改、部署应用程序等操作所需的配方和关联文件。

   根据您的要求，您还可能需要配方来处理取消部署和关闭任务。有关更多信息，请参阅 说明书和诀窍。

2. 创建自定义层。
3. 将您的配方分配给相应的生命周期事件。

您随后要将实例添加到层，启动它们，然后将应用程序部署到这些实例。

> **Important**
> 要将应用程序部署到自定义层的实例，您必须实施配方来处理部署操作并将配方分配给层的 Deploy 事件。

按层操作系统程序包安装

> **Important**
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

从 Chef 12 开始，您必须使用自定义配方将程序包安装到正在运行不同操作系统的层上。这种方法会在程序包安装方面为您提供极大的灵活性和控制力。

例如，假设你要将 Apache 安装在正在运行的各层 RedHat、Ubuntu 和 Amazon 版本的 Linux 操作系统上。Apache 软件包 RedHat 和 Amazon Linux 被调用`httpd`，但在 Ubuntu 上，它被称为。`apache2`

要解决程序包命名方面的这种差异，您可以使用类似于下面的示例配方中的语法。该配方可安装适用于各个操作系统的 Apache 程序包。此示例基于 Chef 文档。

```
package "Install Apache" do
   case node[:platform]
      when "redhat", "amazon"
         package_name "httpd"
      when "ubuntu"
         package_name "apache2"
   end
end
```

有关如何使用 package 资源来管理包的详细信息，请转到 Chef 文档中的包页面。

或者，您也可以使用 Chef 配方 DSL (领域特定的语言) 的 value_for_platform 帮助程序方法，这种方法可以更加简便地完成同样的任务：

```
package "Install Apache" do
   package_name value_for_platform(
      ["redhat", "amazon"] => { "default" => "httpd" },
      ["ubuntu"] => { "default" => "apache2" }
   )
end
```

有关使用 value_for_platform 帮助程序方法的信息，请转到 About the Recipe DSL。

实例

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

实例表示诸如 Amazon EC2 实例等计算资源，可处理应用程序服务和流量均衡等工作。实例的操作系统可以具有多个 Linux 发行版中的任意一个或 Windows Server 2012 R2。

可以通过以下任一方式向堆栈添加实例：

- 使用 AWS OpsWorks 堆栈向堆栈添加实例。您添加的实例表示 Amazon EC2 实例。
- 对于基于 Linux 的堆栈，您可以注册在其他位置创建的实例，包括用 Amazon EC2 创建的实例以及在您自己的硬件上运行的本地实例。

  然后，您可以使用 AWS OpsWorks 堆栈来管理这些实例，其方式与 AWS OpsWorks 使用堆栈创建的实例大致相同

本节介绍如何使用 AWS OpsWorks 堆栈创建和管理实例。

主题

- 使用 AWS OpsWorks 堆栈实例

- 使用在 AWS OpsWorks Stacks 外部创建的计算资源
- 编辑实例配置
- 删除 AWS OpsWorks 堆栈实例
- 使用 SSH 登录 Linux 实例
- 使用 RDP 登录 Windows 实例

使用 AWS OpsWorks 堆栈实例

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

您可以使用 AWS OpsWorks 堆栈创建实例并将其添加到堆栈中。

主题

- AWS OpsWorks 堆栈操作系统
- 将实例添加到层
- 使用自定义 AMI
- 手动启动、停止和重启全天候实例
- 使用基于时间和基于负载的实例管理负载

AWS OpsWorks 堆栈操作系统

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

AWS OpsWorks Stacks 支持多种内置操作系统的 64 位版本，包括亚马逊和 Ubuntu Linux 发行版以及微软 Windows 服务器。一些常规说明：

- 堆栈的实例可以运行 Linux 或 Windows。

  堆栈可以在不同的实例上拥有不同的 Linux 版本或发行版，但您不能将 Linux 和 Windows 实例混合使用。
- 您可以使用自定义 AMI（A mazon 系统映像），但它们必须基于本节主题中描述的 AWS OpsWorks Stacks 支持的 AMI 之一。虽然可以使用已经从自定义或社区生成的 AMI 创建的其他操作系统（例如 CentOS 6.x）创建或注册实例，但这些实例不受官方支持。
  - Linux 操作系统：
  - Microsoft Windows Server
- 您可以手动启动和停止实例，或让 AWS OpsWorks Stacks 自动扩展实例的数量。

  您可以对任何堆栈使用基于时间的自动扩展功能；Linux 堆栈还可以使用基于负载的扩展。
- 除了使用 AWS OpsWorks 堆栈创建 Amazon EC2 实例外，您还可以使用在堆栈之外创建的 Linux 堆栈注册实例。 AWS OpsWorks

  这包括 Amazon EC2 实例和您自己的硬件上运行的实例。但是，它们必须正在运行受支持的 Linux 发行版之一。您不能注册 Amazon EC2 或本地 Windows 实例。

你可以运行 AWS OpsWorks Stacks `DescribeOperatingSystems`API 来返回支持的操作系统及其支持的 Chef 版本的列表。以下是使用 AWS CLI的命令的示例。

```
aws opsworks describe-operating-systems
```

以下为响应示例。

```
{
    "OperatingSystems": [
        {
            "Name": "Amazon Linux",
            "Id": "Amazon Linux",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "11.10"
                },
                {
                    "Name": "Chef",
                    "Version": "11.4"
```

```
                },
                {
                    "Name": "Chef",
                    "Version": "0.9"
                }
            ],
            "ReportedName": "amazon",
            "ReportedVersion": "2014.03",
            "Supported": false
        },
        {
            "Name": "Amazon Linux 2",
            "Id": "Amazon Linux 2",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12"
                }
            ],
            "ReportedName": "amazon",
            "ReportedVersion": "2"
        },
        {
            "Name": "Amazon Linux 2014.09",
            "Id": "Amazon Linux 2014.09",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "11.10"
                },
                {
                    "Name": "Chef",
                    "Version": "11.4"
                },
                {
                    "Name": "Chef",
                    "Version": "0.9"
                }
            ],
            "ReportedName": "amazon",
            "ReportedVersion": "2014.09",
            "Supported": false
```

```
        },
        {
            "Name": "Amazon Linux 2015.03",
            "Id": "Amazon Linux 2015.03",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12"
                },
                {
                    "Name": "Chef",
                    "Version": "11.10"
                },
                {
                    "Name": "Chef",
                    "Version": "11.4"
                },
                {
                    "Name": "Chef",
                    "Version": "0.9"
                }
            ],
            "ReportedName": "amazon",
            "ReportedVersion": "2015.03",
            "Supported": false
        },
        {
            "Name": "Amazon Linux 2015.09",
            "Id": "Amazon Linux 2015.09",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12"
                },
                {
                    "Name": "Chef",
                    "Version": "11.10"
                },
                {
                    "Name": "Chef",
                    "Version": "11.4"
                },
```

```
                {
                    "Name": "Chef",
                    "Version": "0.9"
                }
            ],
            "ReportedName": "amazon",
            "ReportedVersion": "2015.09",
            "Supported": false
        },
        {
            "Name": "Amazon Linux 2016.03",
            "Id": "Amazon Linux 2016.03",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12"
                },
                {
                    "Name": "Chef",
                    "Version": "11.10"
                },
                {
                    "Name": "Chef",
                    "Version": "11.4"
                },
                {
                    "Name": "Chef",
                    "Version": "0.9"
                }
            ],
            "ReportedName": "amazon",
            "ReportedVersion": "2016.03"
        },
        {
            "Name": "Amazon Linux 2016.09",
            "Id": "Amazon Linux 2016.09",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12"
                },
                {
```

```
                    "Name": "Chef",
                    "Version": "11.10"
                },
                {
                    "Name": "Chef",
                    "Version": "11.4"
                },
                {
                    "Name": "Chef",
                    "Version": "0.9"
                }
            ],
            "ReportedName": "amazon",
            "ReportedVersion": "2016.09"
        },
        {
            "Name": "Amazon Linux 2017.03",
            "Id": "Amazon Linux 2017.03",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12"
                },
                {
                    "Name": "Chef",
                    "Version": "11.10"
                },
                {
                    "Name": "Chef",
                    "Version": "11.4"
                },
                {
                    "Name": "Chef",
                    "Version": "0.9"
                }
            ],
            "ReportedName": "amazon",
            "ReportedVersion": "2017.03"
        },
        {
            "Name": "Amazon Linux 2017.09",
            "Id": "Amazon Linux 2017.09",
            "Type": "Linux",
```

```
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12"
                },
                {
                    "Name": "Chef",
                    "Version": "11.10"
                },
                {
                    "Name": "Chef",
                    "Version": "11.4"
                },
                {
                    "Name": "Chef",
                    "Version": "0.9"
                }
            ],
            "ReportedName": "amazon",
            "ReportedVersion": "2017.09"
        },
        {
            "Name": "Amazon Linux 2018.03",
            "Id": "Amazon Linux 2018.03",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12"
                },
                {
                    "Name": "Chef",
                    "Version": "11.10"
                }
            ],
            "ReportedName": "amazon",
            "ReportedVersion": "2018.03"
        },
        {
            "Name": "CentOS Linux 7",
            "Id": "CentOS Linux 7",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
```

```
                "Name": "Chef",
                "Version": "12"
            }
        ],
        "ReportedName": "CentOS Linux",
        "ReportedVersion": "7"
    },
    {
        "Name": "Microsoft Windows Server 2012 R2 Base",
        "Id": "Microsoft Windows Server 2012 R2 Base",
        "Type": "Windows",
        "ConfigurationManagers": [
            {
                "Name": "Chef",
                "Version": "12.2"
            }
        ],
        "ReportedName": "microsoft windows server",
        "ReportedVersion": "2012 r2 standard",
        "Supported": false
    },
    {
        "Name": "Microsoft Windows Server 2012 R2 with SQL Server Express",
        "Id": "Microsoft Windows Server 2012 R2 with SQL Server Express",
        "Type": "Windows",
        "ConfigurationManagers": [
            {
                "Name": "Chef",
                "Version": "12.2"
            }
        ],
        "ReportedName": "microsoft windows server",
        "ReportedVersion": "2012 r2 standard",
        "Supported": false
    },
    {
        "Name": "Microsoft Windows Server 2012 R2 with SQL Server Standard",
        "Id": "Microsoft Windows Server 2012 R2 with SQL Server Standard",
        "Type": "Windows",
        "ConfigurationManagers": [
            {
                "Name": "Chef",
                "Version": "12.2"
            }
```

```
            ],
            "ReportedName": "microsoft windows server",
            "ReportedVersion": "2012 r2 standard",
            "Supported": false
        },
        {
            "Name": "Microsoft Windows Server 2012 R2 with SQL Server Web",
            "Id": "Microsoft Windows Server 2012 R2 with SQL Server Web",
            "Type": "Windows",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12.2"
                }
            ],
            "ReportedName": "microsoft windows server",
            "ReportedVersion": "2012 r2 standard",
            "Supported": false
        },
        {
            "Name": "Microsoft Windows Server 2019 Base",
            "Id": "Microsoft Windows Server 2019 Base",
            "Type": "Windows",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12.2"
                }
            ],
            "ReportedName": "microsoft windows server",
            "ReportedVersion": "2019 datacenter"
        },
        {
            "Name": "Microsoft Windows Server 2019 with SQL Server Express",
            "Id": "Microsoft Windows Server 2019 with SQL Server Express",
            "Type": "Windows",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12.2"
                }
            ],
            "ReportedName": "microsoft windows server",
            "ReportedVersion": "2019 datacenter"
```

```
        },
        {
            "Name": "Microsoft Windows Server 2019 with SQL Server Standard",
            "Id": "Microsoft Windows Server 2019 with SQL Server Standard",
            "Type": "Windows",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12.2"
                }
            ],
            "ReportedName": "microsoft windows server",
            "ReportedVersion": "2019 datacenter"
        },
        {
            "Name": "Microsoft Windows Server 2019 with SQL Server Web",
            "Id": "Microsoft Windows Server 2019 with SQL Server Web",
            "Type": "Windows",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12.2"
                }
            ],
            "ReportedName": "microsoft windows server",
            "ReportedVersion": "2019 datacenter"
        },
        {
            "Name": "Microsoft Windows Server 2022 Base",
            "Id": "Microsoft Windows Server 2022 Base",
            "Type": "Windows",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12.2"
                }
            ],
            "ReportedName": "microsoft windows server",
            "ReportedVersion": "2022 datacenter"
        },
        {
            "Name": "Microsoft Windows Server 2022 with SQL Server Express",
            "Id": "Microsoft Windows Server 2022 with SQL Server Express",
            "Type": "Windows",
```

```
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12.2"
                }
            ],
            "ReportedName": "microsoft windows server",
            "ReportedVersion": "2022 datacenter"
        },
        {
            "Name": "Microsoft Windows Server 2022 with SQL Server Standard",
            "Id": "Microsoft Windows Server 2022 with SQL Server Standard",
            "Type": "Windows",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12.2"
                }
            ],
            "ReportedName": "microsoft windows server",
            "ReportedVersion": "2022 datacenter"
        },
        {
            "Name": "Microsoft Windows Server 2022 with SQL Server Web",
            "Id": "Microsoft Windows Server 2022 with SQL Server Web",
            "Type": "Windows",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12.2"
                }
            ],
            "ReportedName": "microsoft windows server",
            "ReportedVersion": "2022 datacenter"
        },
        {
            "Name": "Red Hat Enterprise Linux 7",
            "Id": "Red Hat Enterprise Linux 7",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12"
                },
```

```
                {
                    "Name": "Chef",
                    "Version": "11.10"
                }
            ],
            "ReportedName": "Red Hat Enterprise Linux",
            "ReportedVersion": "7"
        },
        {
            "Name": "Ubuntu 12.04 LTS",
            "Id": "Ubuntu 12.04 LTS",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12"
                },
                {
                    "Name": "Chef",
                    "Version": "11.10"
                },
                {
                    "Name": "Chef",
                    "Version": "11.4"
                },
                {
                    "Name": "Chef",
                    "Version": "0.9"
                }
            ],
            "ReportedName": "ubuntu",
            "ReportedVersion": "12.04",
            "Supported": false
        },
        {
            "Name": "Ubuntu 14.04 LTS",
            "Id": "Ubuntu 14.04 LTS",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12"
                },
                {
```

```
                "Name": "Chef",
                "Version": "11.10"
            }
        ],
        "ReportedName": "ubuntu",
        "ReportedVersion": "14.04"
    },
    {
        "Name": "Ubuntu 16.04 LTS",
        "Id": "Ubuntu 16.04 LTS",
        "Type": "Linux",
        "ConfigurationManagers": [
            {
                "Name": "Chef",
                "Version": "12"
            }
        ],
        "ReportedName": "ubuntu",
        "ReportedVersion": "16.04"
    },
    {
        "Name": "Ubuntu 18.04 LTS",
        "Id": "Ubuntu 18.04 LTS",
        "Type": "Linux",
        "ConfigurationManagers": [
            {
                "Name": "Chef",
                "Version": "12"
            }
        ],
        "ReportedName": "ubuntu",
        "ReportedVersion": "18.04"
    },
    {
        "Name": "Ubuntu 20.04 LTS",
        "Id": "Ubuntu 20.04 LTS",
        "Type": "Linux",
        "ConfigurationManagers": [
            {
                "Name": "Chef",
                "Version": "12"
            }
        ],
        "ReportedName": "ubuntu",
```

```
            "ReportedVersion": "20.04"
        },
        {
            "Name": "Custom",
            "Id": "Custom",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12"
                },
                {
                    "Name": "Chef",
                    "Version": "11.10"
                },
                {
                    "Name": "Chef",
                    "Version": "11.4"
                },
                {
                    "Name": "Chef",
                    "Version": "0.9"
                }
            ]
        },
        {
            "Name": "CustomWindows",
            "Id": "CustomWindows",
            "Type": "Windows",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12.2"
                }
            ]
        }
    ]
}
```

主题

- Linux 操作系统：
- Microsoft Windows Server

Linux 操作系统：

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

AWS OpsWorks Stacks 支持以下 Linux 操作系统的 64 位版本。

- Amazon Linux和 Amazon Linux 2（请参阅 AWS OpsWorks Stacks 控制台以了解当前支持的版本）
- Ubuntu 20.04 LTS
- CentOS 7
- Red Hat Enterprise Linux 7

您还可以使用基于这些操作系统的自定义 AMI。

关于 Linux 实例的一些常规说明：

支持的包版本

支持的版本和包的补丁级别 (如 Ruby) 取决于操作系统和版本，如后面的部分中所述。

更新

默认情况下，AWS OpsWorks Stacks 通过自动调用`yum update`或在实例启动`apt-get update`后自动调用来确保 Linux 实例安装最新的安全补丁。要禁用自动更新，请使用CreateInstance、UpdateInstanceCreateLayer、或UpdateLayer操作（或等效的 AW S 开发工具包方法或 AWS CLI 命令）将`InstallUpdatesOnBoot`参数设置为。`false`

为避免服务中断，AWS OpsWorks Stacks 不会在实例联机后自动安装更新。您可以通过运行 Upgrade Operating System 堆栈命令，随时手动更新联机实例的操作系统。有关如何管理安全更新的更多信息，请参阅管理安全更新。

要更好地控制 AWS OpsWorks Stacks 如何更新您的实例，请基于支持的操作系统之一创建自定义 AMI。例如，通过自定义 AMI，您可以指定在实例上安装哪些程序包版本。每个 Linux 发行版都具有不同的支持时间表和程序包合并策略，因此，您应当考虑哪种方法最适合您的要求。有关更多信息，请参阅 使用自定义 AMI。

主机文件

每个在线实例都有一个将 IP 地址映射到主机名的/etc/hosts文件。 AWS OpsWorks 堆栈包含每个实例hosts文件中所有堆栈在线实例的公用地址和私有地址。例如，假设您有一个堆栈，该堆栈拥有两个 Node.js 应用程序服务器实例（nodejs-app1 和 nodejs-app2）以及一个 MySQL 实例 (db-master1)。nodejs-app1 实例的 hosts 文件将如以下示例所示，其他实例将拥有相似的 hosts 文件。

```
...
# OpsWorks Layer State
192.0.2.0 nodejs-app1.localdomain nodejs-app1
10.145.160.232 db-master1
198.51.100.0 db-master1-ext
10.243.77.78 nodejs-app2
203.0.113.0 nodejs-app2-ext
10.84.66.6 nodejs-app1
192.0.2.0 nodejs-app1-ext
```

AWS OpsWorks 堆栈代理代理支持

Chef 11.10 及更高版本 AWS OpsWorks 堆栈的堆栈代理包括对代理服务器的基本支持，代理服务器通常用于隔离的 VPC。要启用代理服务器支持，实例必须拥有为 HTTP 和 HTTPS 流量提供适当设置的 /etc/environment 文件。该文件应当如下所示，其中，您可以将突出显示的文本替换为您的代理服务器的 URL 和端口：

```
http_proxy="http://myproxy.example.com:8080/"
https_proxy="http://myproxy.example.com:8080/"
no_proxy="169.254.169.254"
```

要启用代理支持，我们建议创建自定义 AMI (其中包括一个合适的 /etc/environment 文件)，然后使用该 AMI 创建您的实例。

> **Note**
>
> 我们不建议使用自定义配方在您的实例上创建/etc/environment文件。 AWS OpsWorks Stacks 需要在设置过程的早期，也就是任何自定义配方执行之前的代理服务器数据。

主题

- Amazon Linux
- Ubuntu LTS
- CentOS
- Red Hat Enterprise Linux

Amazon Linux

AWS OpsWorks Stacks 支持 64 位版本的亚马逊 Linux 和亚马逊 Linux 2。除了定期更新和补丁外，Amazon Linux 大约每六个月还会发布新版本，新版本中可能涉及重大更改。当您创建堆栈或新的实例时，您必须指定要使用哪一个 Amazon Linux 版本。当 AWS 发布新版本后，您的实例将继续运行指定的版本，直到您明确更改该版本。新 Amazon Linux 版本发布后，有为期四周的迁移周期，在此周期内，AWS 将继续提供针对旧版本的定期更新。迁移周期结束后，您的实例仍可以继续运行旧版本，但 AWS 不再提供更新。有关更多信息，请参阅 Amazon Linux AMI 常见问题。

当新的 Amazon Linux 版本发布后，我们建议您在迁移周期内更新到新版本，以便您的实例继续获得安全更新。更新您的生产堆栈的实例之前，我们建议您启动一个新的实例，并验证您的应用程序可以在新版本上正常运行。然后，您可以更新生产堆栈实例。

> **Note**
>
> 默认情况下，基于 Amazon Linux 的自定义 AMI 会在发布后自动更新到新版本。建议的做法是：将您的自定义 AMI 保持在特定的 Amazon Linux 版本，这样，您就可以推迟更新，直到您测试了新版本。有关更多信息，请参阅如何将我的 AMI 保持在特定版本？
> 如果您使用 AWS CloudFormation 模板创建包含运行 Amazon Linux 的实例的堆栈，则模板应明确指定亚马逊 Linux 版本。具体而言，如果您的模板指定 `Amazon Linux`，则实例将继续运行版本 2016.09。有关更多信息，请参阅AWS::OpsWorks::Stack和AWS::OpsWorks::Instance。

要更新实例的 Amazon Linux 版本，请执行以下操作之一：

- 对于联机实例，运行 Upgrade Operating System 堆栈命令。

  当有新的 Amazon Linux 版本可用时，Instances 和 Stack 页面将显示一个通知，其中包含一个可让您转到 Run Command 页面的链接。然后，您可以运行 Upgrade Operating System 升级您的实例。

- 对于由 Amazon Elastic Block Store 支持（由 EBS 支持）的脱机实例，启动实例并运行升级操作系统，如前面的说明所述。
- 对于由实例存储支持的脱机实例 (包括基于时间和基于负载的实例)，编辑实例的 Operating system 设置以指定新版本。

  AWS OpsWorks 堆栈在实例重启时会自动将其更新到新版本。

Amazon Linux：受支持的 Node.js 版本

Amazon Linux 版本	Node.js 版本
2	(Not applicable to operating systems that are available for Chef 12 and higher stacks only)
2018.03	0.12.18
2017.09	0.12.18
2017.03	0.12.18
2016.09	0.12.18 0.12.17 0.12.16 0.12.15
2016.03	0.12.18 0.12.17 0.12.16 0.12.15 0.12.14 0.12.13 0.12.12 0.12.10

Amazon Linux：受支持的 Chef 版本

Chef 版本	受支持的 Amazon Linux 版本
12	Amazon Linux 2 Amazon Linux 2018.03 Amazon Linux 2017.09 Amazon Linux 2017.03 Amazon Linux 2016.09 Amazon Linux 2016.03
11.10	Amazon Linux 2018.03 Amazon Linux 2017.09 Amazon Linux 2017.03 Amazon Linux 2016.09 Amazon Linux 2016.03
11.4 (deprecated)	Amazon Linux 2016.09 Amazon Linux 2016.03

Important

更新 t1.micro 实例之前，请确保它们拥有临时交换文件 `/var/swapfile`。Chef 0.9 堆栈上的 t1.micro 实例没有交换文件。对于 Chef 11.4 和 Chef 11.10 堆栈，新版本的实例代理会自动为 t1.micro 实例创建交换文件。但是，此更改在几周之后引入，因此，您应当检查大约在 2014 年 3 月 24 日之前创建的实例上是否存在 `/var/swapfile`。
对于缺少交换文件的 t1.micro 实例，您可以根据以下方法创建一个交换文件：

- 对于 Chef 11.10 及更高版本的堆栈，创建新的 t1.micro 实例，这些实例自动拥有一个交换文件。
- 对于 Chef 0.9 堆栈，以根用户的身份对每个实例运行以下命令。

```
dd if=/dev/zero of=/var/swapfile bs=1M count=256
 mkswap /var/swapfile
 chown root:root /var/swapfile
 chmod 0600 /var/swapfile
 swapon /var/swapfile
```

如果您不希望创建新实例，还可以在 Chef 11.10 和更高版本上使用这些命令。

Ubuntu LTS

Ubuntu 大约每两年发布一次新的 Ubuntu LTS 版本，并为每个版本提供大约五年的支持。Ubuntu 在整个操作系统支持期间提供安全补丁和更新。有关更多信息，请参阅 LTS - Ubuntu Wiki。

- 您不能将现有 Ubuntu 实例更新到更新版本的 Ubuntu。

  您必须创建新的 Ubuntu 实例，并删除原来的实例。
- 对于 Chef 12 及更高版本的堆栈，仅支持 Ubuntu 20.04 LTS。

CentOS

AWS OpsWorks Stacks 支持 64 位版本的 CentOS 7。最开始受支持的版本是 CentOS 7，CentOS 大约每两年发布一次新版本。

当你在 CentOS 堆栈中启动一个新实例时， AWS OpsWorks Stacks 会自动安装最新的 CentOS 版本。由于在新的 CentOS 次要版本发布时， AWS OpsWorks Stacks 不会自动更新现有实例上的操作系统，因此新创建的实例可能会比堆栈的现有实例获得更新的版本。要使版本在您的堆栈中保持一致，您可以按照以下方法将您的现有实例更新到当前的 CentOS 版本：

- 对于联机实例，运行 Upgrade Operating System 堆栈命令，该命令将对指定实例运行 `yum update` 以将其更新到当前版本。

  当有新的 CentOS 7 次要版本可用时，Instances 和 Stack 页面将显示一个通知，其中包含一个可让您转到 Run Command 页面的链接。然后，您可以运行 Upgrade Operating System 升级您的实例。
- 对于由 Amazon EBS 支持的脱机实例，启动实例并运行升级操作系统，如上一列表项中所述。
- 对于离线实例存储支持的实例， AWS OpsWorks Stacks 会在实例重启时自动安装新版本。

CentOS：受支持的 Chef 版本

Chef 版本	受支持的 CentOS 版本
`12`	`CentOS 7`

Chef 版本	受支持的 CentOS 版本
11.10	(None supported)
11.4 (deprecated)	(None supported)

> **Note**
>
> AWS OpsWorks Stacks 支持 CentOS 实例的 Apache 2.4。

Red Hat Enterprise Linux

AWS OpsWorks Stacks 支持 64 位版本的红帽企业 Linux 7 (RHEL 7)。最初支持的版本是 RHEL 7.1，Red Hat 大约每 9 个月发布一次新的次要版本。次要版本应当与 RHEL 7.0 兼容。有关更多信息，请参阅生命周期和更新策略。

当您启动新实例时， AWS OpsWorks Stacks 会自动安装当前的 RHEL 7 版本。由于在发布新的 RHEL 7 次要版本时， AWS OpsWorks Stacks 不会自动更新现有实例上的操作系统，因此新创建的实例可能会收到比堆栈现有实例更新的版本。要使版本在您的堆栈中保持一致，您可以按照以下方法将您的现有实例更新到当前的 RHEL 7 版本：

- 对于联机实例，运行 Upgrade Operating System 堆栈命令，该命令将对指定实例运行 `yum update` 以将其更新到当前版本。

  当有新的 RHEL 7 版本可用时，Instances 和 Stack 页面将显示一个通知，其中包含一个可让您转到 Run Command 页面的链接。然后，您可以运行 Upgrade Operating System 升级您的实例。
- 对于由 Amazon EBS 支持的脱机实例，启动实例并运行升级操作系统，如上一列表项中所述。
- 对于离线实例存储支持的实例， AWS OpsWorks Stacks 会在实例重启时自动安装新版本。

Red Hat Enterprise Linux：受支持的 Node.js 版本

RHEL 版本	Node.js 版本
7	(Node.js versions only apply to Chef 11.10 stacks)

RHEL 版本	Node.js 版本
	0.8.19 0.8.26 0.10.11 0.10.21 0.10.24 0.10.25 0.10.27 0.10.29 0.10.40 0.12.10 0.12.12 0.12.13 0.12.15

Red Hat Enterprise Linux：受支持的 Chef 版本

Chef 版本	受支持的 RHEL 版本
12	Red Hat Enterprise Linux 7
11.10	Red Hat Enterprise Linux 7
11.4 (deprecated)	(None supported)

所有早于 0.10.40 的 Node.js 版本均已弃用。0.12.7 和 0.12.9 也已弃用。

> **Note**
>
> AWS OpsWorks Stacks 支持 RHEL 7 实例的 Apache 2.4。

Microsoft Windows Server

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

以下注释描述了 AWS OpsWorks 堆栈对 Windows 实例的支持。Windows 实例仅适用于 Chef 12.2 堆栈。Windows 堆栈中的准确 Chef 版本为 12.22。

目前，无法在 AWS OpsWorks 使用英语-美国 (en-US) 以外的系统用户界面语言的基于 Windows 的实例上安装 AWS OpsWorks Stacks 代理，Stacks 也无法管理。

版本

AWS OpsWorks Stacks 支持以下 Windows 64 位版本：

- Microsoft Windows Server 2022 Base
- Microsoft Windows Server 2022 with SQL Server Express
- Microsoft Windows Server 2022 with SQL Server Standard
- Microsoft Windows Server 2022 with SQL Server Web
- Microsoft Windows Server 2019 Base
- Microsoft Windows Server 2019 with SQL Server Express
- Microsoft Windows Server 2019 with SQL Server Standard
- Microsoft Windows Server 2019 with SQL Server Web

创建实例

你可以使用 AWS OpsWorks Stacks 控制台、API 或 CLI 创建 Windows 实例。Windows 实例由 Amazon EBS 提供支持，但您不能挂载额外的 Amazon EBS 卷。

Windows 堆栈可以使用全天候可用的实例，但您需要手动启动和停止这些实例。它们也可以使用基于时间的自动扩展，这将根据用户指定的计划来自动启动和停止实例。基于 Windows 的堆栈无法使用基于负载的自动扩展。

您无法使用堆栈注册在堆栈之 AWS OpsWorks 外创建的 Windows 实例。

更新

AWS 会为 Windows AMI 更新每组补丁，因此，当您创建实例时，该实例将包括最新的更新。但是， AWS OpsWorks Stacks 不提供对在线 Windows 实例应用更新的方法。确保 Windows 为最新版本的最简单的方法是定期更换您的实例，以便它们始终运行最新的 AMI。

图层

要处理诸如安装和配置软件或部署应用程序等任务，您将需要实施一个或多个包含自定义配方的自定义层。

Chef

Windows 实例使用 Chef 12.22 并以本地模式运行 chef-client，这会启动一个称为 chef-zero 的本地内存 Chef 服务器。此服务器启动后，使自定义配方可以使用 Chef 搜索和数据包。

远程登录

AWS OpsWorks Stacks 为获得授权的 IAM 用户提供了一个密码，他们可以使用该密码登录 Windows 实例。此密码在指定时间后过期。管理员可以使用 SSH 密钥对来检索实例的管理员密码，使用该密码，可以实现不受限制的 RDP 访问。有关更多信息，请参阅 使用 RDP 登录。

AWS SDK

AWS OpsWorks Stacks 会自动AWS SDK for .NET在每个实例上安装。该软件包包括 AWS .NET 库和适用于 Windows 的 AWS 工具，包括适用于 Windows 的 AWS 工具 PowerShell。要使用 Ruby 开发工具包，您可以让自定义配方安装适当的 Gem。

监控和指标

Windows 实例支持标准 Amazon CloudWatch (CloudWatch) 指标，您可以在 CloudWatch 控制台中查看这些指标。

Ruby

AWS OpsWorks Stacks 在 Windows 实例上安装的 Chef 12.22 客户端附带了 Ruby 2.3.6。但是， AWS OpsWorks Stacks 不会将可执行文件的目录添加到 PATH 环境变量中。要让您的应用程序使用此 Ruby 版本，通常您可以在 `C:\opscode\chef\embedded\bin\` 中找到它。

AWS OpsWorks Stacks Agent CLI

Windows 实例上的 AWS OpsWorks Stacks 代理不会公开命令行界面。

代理支持

执行以下操作为 Windows 实例设置代理支持：

1. 修改`machine.config`以添加以下内容，从而为 Windows PowerShell （初始引导）和.NET（AWS OpsWorks 堆栈代理）应用程序添加代理支持：

```
<system.net>
  <defaultProxy>
    <proxy autoDetect="false" bypassonlocal="true"
 proxyaddress="http://10.100.1.91:3128"  usesystemdefault="false" />
    <bypasslist>
      <add address="localhost" />
      <add address="169.254.169.254" />
    </bypasslist>
  </defaultProxy>
</system.net>
```

2. 运行以下命令来设置环境变量，供 Chef 和 Git 以后使用：

```
setx /m no_proxy "localhost,169.254.169.254"
setx /m http_proxy "http://10.100.1.91:3128"
setx /m https_proxy "http://10.100.1.91:3128"
```

> **Note**
>
> 要更好地控制 AWS OpsWorks Stacks 如何更新实例，请基于微软 Windows Server 2022 Base 创建自定义 AMI。例如，通过自定义 AMI，您可以指定在实例上安装哪一软件，如 Web Server (IIS)。有关更多信息，请参阅 使用自定义 AMI。

将实例添加到层

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

在创建层后，您通常会添加至少一个实例。如果当前设置无法处理负载，则可在稍后添加更多实例。您也可以使用基于负载或基于时间的实例来自动扩展实例的数量。

可以将新实例或现有实例添加到层：

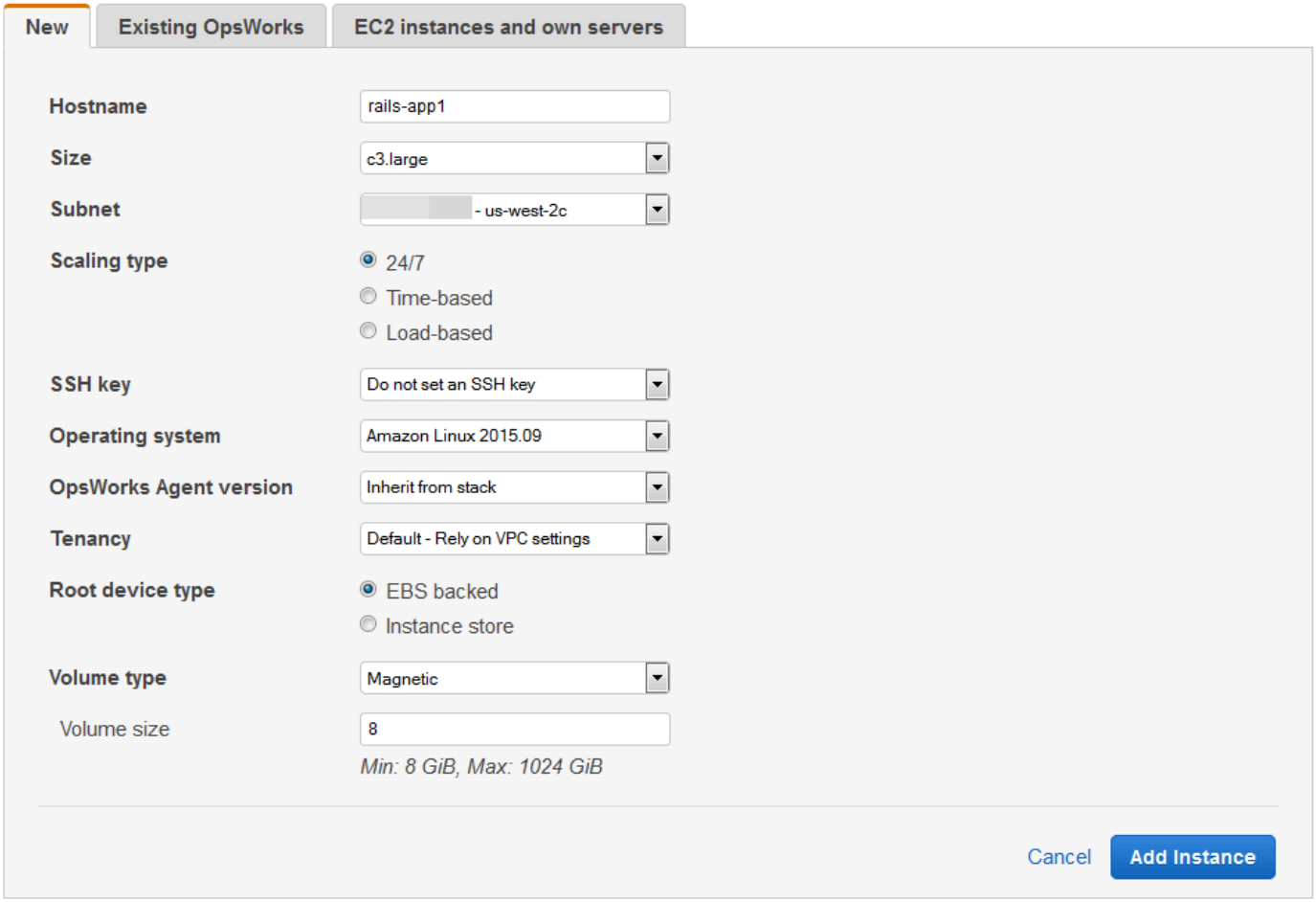
- 新OpsWorks 建-创建根据您的规格配置的新实例，并使其成为该层的成员。
- 现有实例：您可以从任何兼容的层添加一个现有实例，但它必须处于脱机（已停止）状态。

如果一个实例属于多个层， AWS OpsWorks Stacks 将在生命周期事件发生时或当您运行 stack 或 deployment 命令时针对该实例的每个层运行配方。

您还可以通过编辑实例的配置来使它成为多个层的成员。有关更多信息，请参阅 编辑实例配置。

将新实例添加到层

1. 在 Instances 页面上，为相应的层选择 +Instance，然后 (如果需要) 选择 New 选项卡。如果您要配置 Host name、Size 和 Subnet 或 Availability Zone 之外的内容，请选择 Advanced >> 以查看更多选项。下面显示了完整的选项集：



2. 如果需要，您可以覆盖默认配置，其中大部分内容是您在创建堆栈时指定的。有关更多信息，请参阅 创建新堆栈。

Hostname

在网络上标识实例。默认情况下，AWS OpsWorks Stacks 使用您在创建堆栈时指定的主机名主题生成每个实例的主机名。您可以覆盖此值并指定您的首选主机名。

大小

一种 Amazon EC2 实例类型，用于指定实例的资源，例如内存量或虚拟内核数量。AWS OpsWorks Stacks 为每个实例指定默认大小，您可以使用首选实例类型来覆盖该大小。

堆栈支持的 AWS OpsWorks 实例类型取决于堆栈是否在 VPC 中。如果您的账户使用 AWS Free Tier，则实例类型也有限。Size (大小) 下拉列表显示您的堆栈支持的 Chef 版本支持的实例类型。请注意，微型实例 (如 t1.micro) 可能没有足够资源来支持一些层。有关更多信息，请参阅 实例类型。

> **Note**
>
> 如果您使用的是负载均衡的实例，请注意，配置生命周期事件会产生可能持续一分钟或更长时间的显著 CPU 负载高峰。如果使用较小的实例，此负载高峰可能足以触发升级，特别是对于频繁出现配置事件的大型负载均衡的堆栈。以下是可能减少造成不必要升级的配置事件的一些方法。
>
> - 使用较大的实例，以使来自配置事件的额外负载不足以触发升级。
> - 请勿使用共享 CPU 资源的实例类型，如 T2。
>
>   这样可确保，当配置事件发生时，实例的所有 CPU 资源立即可用。
> - 使 `exceeded threshold` 时间显著长于处理配置事件所需的时间 (大约 5 分钟)。
>
>   有关更多信息，请参阅 使用基于负载的自动扩展。

可用区/子网

如果堆栈不在 VPC 中，则将此设置标记为 Availability Zone 并列出该区域的可用区。您可以使用此设置覆盖您在创建堆栈时指定的默认可用区。

如果堆栈正在 VPC 中运行，则将此设置标记为 Subnet 并列出该 VPC 的子网。您可以使用此设置覆盖您在创建堆栈时指定的默认子网。

> **Note**
>
> 默认情况下，AWS OpsWorks 堆栈会列出子网的 CIDR 范围。为了使列表更具可读性，请使用 VPC 控制台或 API 向每个子网添加标签，密钥设置为 **Name**，值设置为子网名称。AWS OpsWorks 堆栈会将该名称附加到 CIDR 范围。在上述示例中，子网的名称标签已设置为 **Private**。

扩展类型

确定如何启动和停止实例。

- 默认值是 24/7 实例，您可手动启动和停止该实例。
- AWS OpsWorks 堆栈根据指定的计划启动和停止基于时间的实例。
- （仅限 Linux）AWS OpsWorks 堆栈根据指定的负载指标启动和停止基于负载的实例。

> **Note**
>
> 您不自行启动或停止基于负载或基于时间的实例。相反，您可以配置实例，AWS OpsWorks Stacks 会根据配置启动和停止实例。有关更多信息，请参阅 使用基于时间和基于负载的实例管理负载。

SSH 密钥

亚马逊 EC2 密钥对。AWS OpsWorks Stacks 在实例上安装公钥。

- 对于 Linux 实例，您可以使用与 SSH 客户端对应的私有密钥来登录该实例。
- 对于 Windows 实例，您可以使用相应的私有密钥检索该实例的管理员密码。然后，您可以结合使用该密码和 RDP，以管理员身份登录该实例。

最初，此设置是您在创建堆栈时指定的 Default SSH key 值。

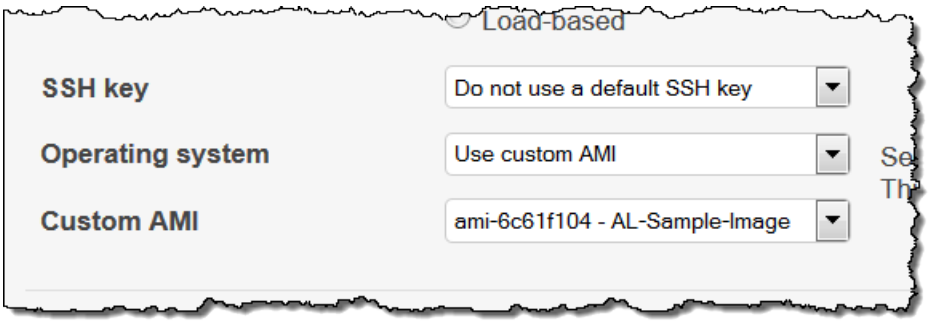
- 如果默认值设置为 不要使用默认的 SSH 密钥，您可以指定您账户中的 Amazon EC2 密钥之一。
- 如果默认值设置为 Amazon EC2 密钥，则可以指定其他密钥或不指定任何密钥。

操作系统

操作系统指定实例正在运行哪个操作系统。AWS OpsWorks 堆栈仅支持 64 位操作系统。

最初，此设置是您在创建堆栈时指定的 Default operating system 值。您可以覆盖默认值来指定其他 Linux 操作系统或自定义 Amazon 系统映像 (AMI)。但是，您不能从 Linux 切换到 Windows 或从 Windows 切换到 Linux。

如果您选择 Use custom AMI，则页面显示自定义 AMI 列表而不是 Architecture 和 Root device type。



有关更多信息，请参阅 使用自定义 AMI。

OpsWorks 代理版本

OpsWorks 代理版本指定您要在实例上运行的 AWS OpsWorks Stacks 代理的版本。如果您希望 AWS OpsWorks Stacks 自动更新代理，请选择 Inherit from stack (从堆栈继承)。要安装代理的特定版本，并且在实例上手动更新代理，请从下拉列表中选择一个版本。

> **Note**
>
> 并非所有的代理版本均适用于所有的操作系统版本。如果您的实例正在运行实例操作系统不完全支持的代理（或者您在实例上安装代理），AWS OpsWorks Stacks 控制台会显示错误消息，指示您安装兼容的代理。

租赁

为您的实例选择租赁选项。您可以选择在专门仅供您使用的物理服务器上运行您的实例。

- Default - Rely on VPC settings。没有租赁，或从您的 VPC 继承租赁设置。
- Dedicated - Run a dedicated instance。为在单一租户硬件上运行的实例按小时付费。有关更多信息，请参阅 Amazon VPC 用户指南中的专用实例和 Amazon EC2 专用实例。

- Dedicated host - Run this instance on a dedicated host。为完全专用于运行您的实例的物理主机付费，让您现有的按套接字、按内核或按 VM 计费的软件许可证降低成本。有关更多信息，请参阅 Amazon EC2文档中的专属主机概览和 Amazon EC2 专属主机。

根设备类型

指定实例的根设备存储。

- Linux 实例可以是由 Amazon EBS 支持或实例存储支持。
- Windows 实例必须由 Amazon EBS 支持。

有关更多信息，请参阅存储。

> **Note**
>
> 初始启动后，Amazon EBS 支持的实例的启动速度比实例存储支持的实例快，因为 AWS OpsWorks Stacks 不必从头开始重新安装实例的软件。有关更多信息，请参阅根设备存储。

卷类型

指定根设备卷类型：Magnetic、Provisioned IOPS (SSD) 或 General Purpose (SSD)。有关更多信息，请参阅 Amazon EBS 卷类型。

卷大小

为指定卷类型指定根设备卷大小。有关更多信息，请参阅 Amazon EBS 卷类型。

- 通用型 (SSD)。允许的最小大小：8 GiB；最大大小：16384 GiB。
- 预配置 IOPS (SSD)。允许的最小大小：8 GiB；最大大小：16384 GiB。您可以设置的每秒输入/输出操作数 (IOPS) 的最小值为 100，最大值为 240。
- 磁介质。允许的最小大小：8 GiB；最大大小：1024 GiB。

3. 选择 Add Instance 以创建新实例。

> **Note**
>
> 您不能在创建实例时覆盖堆栈的默认代理版本设置。要指定自定义代理版本设置，您必须创建实例，然后编辑实例的配置。

将现有实例添加到层

1. 在 Instances 页面上，为相应的层选择 +Instance，然后打开 Existing 选项卡。

> **Note**
>
> 如果您改变了主意而不想使用现有实例，请选择 New 来创建新实例，如上述步骤中所述。

2. 在 Existing 选项卡上，从列表中选择一个实例。
3. 选择 Add Instance 以创建新实例。

实例代表一个 Amazon EC2 实例，但基本上只是一个 AWS OpsWorks 堆栈数据结构。必须启动一个实例才能创建正在运行的 Amazon EC2 实例，如以下部分中所述。

> **Important**
>
> 如果在默认 VPC 中启动实例，则您必须谨慎地修改 VPC 配置。这些实例必须始终能够与 AWS OpsWorks Stacks 服务、Amazon S3 以及软件包存储库进行通信。例如，如果您移除默认网关，则这些实例将失去与 Stacks 服务的连接，然后 AWS OpsWorks 堆栈服务会将这些实例视为失败并自动修复它们。但是， AWS OpsWorks Stacks 将无法在修复的实例上安装实例代理。如果没有代理，这些实例将无法与该服务进行通信，而且启动过程将停留在 `booting` 状态。有关默认 VPC 的更多信息，请参阅支持的平台。

您还可以将 Linux 计算资源整合到堆栈之外创建的 AWS OpsWorks 堆栈中：

- 直接使用 Amazon EC2 控制台、CLI 或 API 创建的 Amazon EC2 实例。
- 您自己的硬件上运行的本地 实例，包括虚拟机上运行的实例。

有关更多信息，请参阅 使用在 AWS OpsWorks Stacks 外部创建的计算资源。

使用自定义 AMI

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

AWS OpsWorks Stacks 支持两种自定义实例的方式：自定义 Amazon 机器映像 (AMI) 和 Chef 食谱。这两种方法都可以让您对安装哪个软件包和软件包版本、如何配置等进行控制。但是，每种方法都有不同优势，因此哪种方法最佳取决于您的要求。

以下是考虑使用自定义 AMI 的主要理由：

- 您希望预捆绑特定软件包，而不是在实例启动之后安装它们。
- 您想要控制软件包更新的时间，以便为您的层提供一致的基本映像。
- 您希望实例 (特别是基于负载的实例) 尽快启动。

以下是考虑使用 Chef 配方的主要理由：

- 它们比自定义 AMI 更灵活。
- 更容易更新。
- 它们可以对正在运行的实例执行更新。

实际上，最佳的解决方案可能是这两种方法的结合。有关配方的更多信息，请参阅说明书和诀窍。

主题

- 自定义 AMI 如何与堆栈配合使用 AWS OpsWorks
- 为 AWS OpsWorks 堆栈创建自定义 AMI

自定义 AMI 如何与堆栈配合使用 AWS OpsWorks

要为您的实例指定自定义 AMI，请在创建新实例时选择使用自定义 AMI 作为实例的操作系统。 AWS OpsWorks 然后，堆栈会显示堆栈区域中的自定义 AMI 列表，您可以从列表中选择相应的自定义 AMI。有关更多信息，请参阅 将实例添加到层。

> **Note**
>
> 您不能将特定自定义 AMI 指定为堆栈的默认操作系统。您可以将 `Use custom AMI` 设置为堆栈的默认操作系统，但只有在向层中添加新实例时才可以指定特定 AMI。有关更多信息，请参阅 将实例添加到层 和 创建新堆栈。虽然可以使用已经从自定义或社区生成的 AMI 创建的其他操作系统（例如 CentOS 6.x）创建实例，但这些实例不受官方支持。

此主题将讨论在创建或使用自定义 AMI 之前应考虑的一些一般问题。

主题

- 启动行为
- 选择层
- 处理应用程序

启动行为

当您启动实例时， AWS OpsWorks Stacks 会使用指定的自定义 AMI 启动一个新的 Amazon EC2 实例。 AWS OpsWorks 然后，堆栈使用 c loud-init 在实例上安装 AWS OpsWorks 堆栈代理，然后代理运行实例的安装配方，然后运行部署配方。实例处于在线状态后，代理会为堆栈中的每个实例 (包括新添加的实例) 运行 Configure 配方。

选择层

AWS OpsWorks Stacks 代理通常不会与已安装的软件包发生冲突。但是，该实例必须是至少一个层的成员。 AWS OpsWorks Stacks 总是运行该层的配方，这可能会导致问题。您应准确了解在使用自定义 AMI 向层添加实例之前，该层的配方对实例要执行的操作。

要查看特定层类型在您的实例上运行哪些配方，请打开包含该层的堆栈。然后在导航窗格中单击 Layers，再单击所关注层的 Recipes。要查看实际代码，请单击配方名称。

> **Note**
>
> 对于 Linux AMI，减少冲突可能性的一种方法是使用 AWS OpsWorks 堆栈来配置和配置作为自定义 AMI 基础的实例。有关更多信息，请参阅 从 AWS OpsWorks 堆栈实例创建自定义 Linux AMI。

处理应用程序

除了软件包，您也可能想在 AMI 中包含一个应用程序。如果您有大型复杂应用程序，在 AMI 中包含该应用程序可以缩短实例的启动时间。您可以在 AMI 中加入小型应用程序，但是与让 AWS OpsWorks Stacks 部署应用程序相比，时间优势通常很小或根本没有。

一个选择是在您的 AMI 中包含应用程序并创建一个应用程序，该应用程序从存储库向实例部署应用程序。这种方法不但可以缩短启动时间，而且能够在运行实例后提供一种更新应用程序的便捷方法。请注意，Chef 配方是幂等的，因此只要存储库中的版本与实例上的版本相同，部署配方就不会修改应用程序。

为 AWS OpsWorks 堆栈创建自定义 AMI

要将自定义 AMI 与 AWS OpsWorks 堆栈配合使用，必须先从自定义实例创建 AMI。可从两个选项中选择：

- 根据一个 64 位版本的AWS OpsWorks Stacks 支持的 AMI，使用 Amazon EC2 控制台或 API 来创建和自定义实例。
- 对于 Linux AMI，使用 OpsWorks 根据其关联层的配置创建 Amazon EC2 实例。

在创建自定义 Linux AMI 之前，请在`/tmp`分区`noexec`上禁用以允许 AWS OpsWorks Stacks 在自定义 Linux 实例上安装其代理。

> **Note**
>
> 请注意，AMI 可能不适用于部分实例类型，因此请确保您启动的 AMI 与计划使用的实例类型兼容。具体而言，R3 实例类型需要硬件辅助虚拟化 (HVM) AMI。

然后，您可以使用 Amazon EC2 控制台或 API 从自定义实例创建自定义 AMI。您可以通过将实例添加到层并指定自定义 AMI 的方法在同一区域中的任一堆栈中使用自定义 AMI。有关如何使用自定义 AMI 创建实例的更多信息，请参阅 将实例添加到层。

> **Note**
>
> 默认情况下， AWS OpsWorks Stacks 会在启动时安装所有 Amazon Linux 更新，从而为您提供最新版本。此外，Amazon Linux 大约每六个月发布一个新版本，其中可能涉及重大更改。默认情况下，基于 Amazon Linux 的自定义 AMI 会在发布后自动更新到新版本。建议的做法

是，将您的自定义 AMI 锁定到特定的 Amazon Linux 版本，这样，您就可以推迟更新，直到您对新版本进行了测试。有关更多信息，请参阅如何将我的 AMI 保持在特定版本？

主题

- 使用 Amazon EC2 创建自定义 AMI
- 从 AWS OpsWorks 堆栈实例创建自定义 Linux AMI
- 创建自定义 Windows AMI

使用 Amazon EC2 创建自定义 AMI

使用 Amazon EC2 控制台或 API 执行整个任务是创建自定义 AMI 的最简单方法，也是 Windows AMI 的唯一选择。有关以下步骤的更多详细信息，请参阅创建您自己的 AMI。

使用 Amazon EC2 控制台或 API 创建自定义 AMI

1. 通过使用一个 64 位版本的 AWS OpsWorks Stacks 支持的 AMI 来创建实例。
2. 通过配置实例、安装程序包等自定义步骤 1 的实例。请记住，将根据 AMI 在每个实例上复制您安装的所有内容，因此请勿包括应特定于特定实例的项目。
3. 停止实例并创建自定义 AMI。

从 AWS OpsWorks 堆栈实例创建自定义 Linux AMI

要使用自定义 AWS OpsWorks Stacks Linux 实例创建 AMI，请注意，创建的每个 Amazon EC2 实例都 OpsWorks 包含一个唯一身份。如果您通过此类实例创建自定义 AMI，它将包含该身份并且基于 AMI 的所有实例都将具有相同的身份。为了确保基于您的自定义 AMI 的实例具有唯一的身份，必须在创建 AMI 之前先删除自定义实例中的身份。

从 AWS OpsWorks Stacks 实例创建自定义 AMI

1. 创建 Linux 堆栈并添加一个或多个层以定义自定义实例的配置。您可以使用内置层、根据需要自定义，以及完全自定义层。有关更多信息，请参阅 自定义堆栈 AWS OpsWorks。
2. 编辑图层并禁用 AutoHealing。
3. 利用首选 Linux 发行版向层中添加实例并启动它。我们建议使用 Amazon EBS 支持的实例。打开实例的详细信息页面并记录其 Amazon EC2 ID 以供将来使用。
4. 当实例处于联机状态时，使用 SSH 登录，然后执行接下来的四个步骤之一，具体取决于您的实例操作系统。

5. 对于 Chef 11 或 Chef 12 堆栈中的 Amazon Linux 实例或者 Chef 11 堆栈中的 Red Hat Enterprise Linux 7 实例，请执行以下操作。

   a. `sudo /etc/init.d/monit stop`

   b. `sudo /etc/init.d/opsworks-agent stop`

   c. `sudo rm -rf /etc/aws/opsworks/ /opt/aws/opsworks/ /var/log/aws/opsworks/ /var/lib/aws/opsworks/ /etc/monit.d/opsworks-agent.monitrc /etc/monit/conf.d/opsworks-agent.monitrc /var/lib/cloud/ /etc/chef`

   > **Note**
   >
   > 对于 Chef 12 堆栈中的实例，请将以下两个文件夹添加到此命令中：
   >
   > - `/var/chef`
   > - `/opt/chef`

   d. `sudo rpm -e opsworks-agent-ruby`

   e. `sudo rpm -e chef`

6. 对于 Chef 12 堆栈中的 Ubuntu 16.04 LTS 或 18.04 LTS 实例，请执行以下操作。

   a. `sudo systemctl stop opsworks-agent`

   b. `sudo rm -rf /etc/aws/opsworks/ /opt/aws/opsworks/ /var/log/aws/opsworks/ /var/lib/aws/opsworks/ /etc/monit.d/opsworks-agent.monitrc /etc/monit/conf.d/opsworks-agent.monitrc /var/lib/cloud/ /var/chef /opt/chef /etc/chef`

   c. `sudo apt-get -y remove chef`

   d. `sudo dpkg -r opsworks-agent-ruby`

   e. `systemctl stop apt-daily.timer`

   f. `systemctl stop apt-daily-upgrade.timer`

   g. `rm /var/lib/systemd/timers/stamp-apt-daily.timer`

   h. `rm /var/lib/systemd/timers/stamp-apt-daily-upgrade.timer`

7. 对于 Chef 12 堆栈中受支持的其他 Ubuntu 版本，请执行以下操作。

   a. `sudo /etc/init.d/monit stop`

b. `sudo /etc/init.d/opsworks-agent stop`

c. `sudo rm -rf /etc/aws/opsworks/ /opt/aws/opsworks/ /var/log/aws/opsworks/ /var/lib/aws/opsworks/ /etc/monit.d/opsworks-agent.monitrc /etc/monit/conf.d/opsworks-agent.monitrc /var/lib/cloud/ /var/chef /opt/chef /etc/chef`

d. `sudo apt-get -y remove chef`

e. `sudo dpkg -r opsworks-agent-ruby`

8. 对于 Chef 12 堆栈中的 Red Hat Enterprise Linux 7 实例，请执行以下操作。

a. `sudo systemctl stop opsworks-agent`

b. `sudo rm -rf /etc/aws/opsworks/ /opt/aws/opsworks/ /var/log/aws/opsworks/ /var/lib/aws/opsworks/ /etc/monit.d/opsworks-agent.monitrc /etc/monit/conf.d/opsworks-agent.monitrc /var/lib/cloud/ /etc/chef /var/chef`

c. `sudo rpm -e opsworks-agent-ruby`

d. `sudo rpm -e chef`

9. 此步骤取决于实例类型：

- 对于由亚马逊 EBS 支持的实例，请使用 AWS OpsWorks Stacks 控制台停止该实例并创建 AMI，如创建由亚马逊 EBS 提供支持的 L inux AMI 中所述。
- 对于由实例存储支持的实例，请按照创建实例存储支持的 Linux AMI 中所述创建 AMI，然后使用堆栈控制台停止 AWS OpsWorks 该实例。

  当您创建 AMI 时，请确保包含证书文件。例如，您可以调用 `ec2-bundle-vol` 命令（将 `-i` 参数设置为 `-i $(find /etc /usr /opt -name '*.pem' -o -name '*.crt' -o -name '*.gpg' | tr '\n' ',')`）。请勿在绑定时删除 apt 公有密钥。默认 `ec2-bundle-vol` 命令处理此任务。

10. 返回堆栈控制台并从 AWS OpsWorks 堆栈中删除实例，清理堆栈。

创建自定义 Windows AMI

以下过程将创建适用于 Windows Server 2022 Base 的自定义 AMI。您可以在 Amazon EC2 管理控制台中选择其他 Windows Server 操作系统。

> **Important**
>
> 目前，无法在 AWS OpsWorks 使用英语-美国 (en-US) 以外的系统用户界面语言的基于 Windows 的实例上安装 AWS OpsWorks Stacks 代理，Stacks 也无法管理。

主题

使用 **Sysprep** 创建自定义 Windows AMI

通过使用 Sysprep 创建自定义 Windows AMI 通常会导致实例启动过程变慢，但步骤更清晰。由于Sysprep活动、重启、堆栈配置和第一次 AWS OpsWorks AWS OpsWorks 堆栈运行（包括设置和配置），首次启动使用创建的映像创建的实例会Sysprep花费更多时间。在 Amazon EC2 控制台中完成创建自定义 Windows AMI 的步骤。

使用 Sysprep 创建自定义 Windows AMI

1. 在 Amazon EC2 控制台中，选择启动实例。
2. 查找 Microsoft Windows Server 2022 Base，然后选择选择。
3. 选择您需要的实例类型，然后选择 Configure Instance Details。对 AMI 进行配置更改，包括机器名称、存储和安全组设置。选择启动。
4. 实例启动过程完成后，获取密码，然后连接到 Windows Remote Desktop Connection 窗口中的实例。
5. 在 Windows “开始” 屏幕上，选择 “开始”，然后开始键入，**ec2configservice**直到结果显示 EC2 ConfigServiceSettings 控制台。打开 控制台。
6. 在 “常规” 选项卡上，确保选中 “启用 UserData 执行” 复选框（尽管此选项不是必需的Sysprep，但 AWS OpsWorks Stacks 需要安装其代理）。清除 Set the computer name of the instance... (设置实例的计算机名称...) 选项对应的复选框，因为此选项可导致 AWS OpsWorks Stacks 重新启动循环。
7. 在图像选项卡上，将管理员密码设置为随机以允许Amazon EC2 自动生成可以用 SSH 密钥检索的密码，或者设置为指定以指定您自己的密码。Sysprep 将保存此设置。如果您指定自己的密码，请将密码保存在方便的位置。建议您不要选择 Keep Existing。

8. 选择 Apply，然后选择 Shutdown with Sysprep。如果提示确认，请选择 Yes。
9. 实例停止后，在 Amazon EC2 控制台中，右键单击实例列表中的实例，选择图像，然后选择创建图像。
10. 在 Create Image 页面中，提供映像的名称和描述，并指定卷配置。完成后，选择 Create Image。
11. 打开 Images 页面，等待映像从 pending 阶段变为 available。新的 AMI 已准备就绪，可供使用。

不使用 **Sysprep** 创建自定义 Windows AMI

在 Amazon EC2 控制台中完成创建自定义 Windows AMI 的步骤。

不使用 Sysprep 创建自定义 Windows AMI

1. 在 Amazon EC2 控制台中，选择启动实例。
2. 查找 Microsoft Windows Server 2022 Base，然后选择选择。
3. 选择您需要的实例类型，然后选择 Configure Instance Details。对 AMI 进行配置更改，包括机器名称、存储和安全组设置。选择启动。
4. 实例启动过程完成后，获取密码，然后连接到 Windows Remote Desktop Connection 窗口中的实例。
5. 在实例上，打开 `C:\Program Files\Amazon\Ec2ConfigService\Settings\config.xml`，更改以下两项设置，然后保存并关闭文件：
   - `Ec2SetPassword` 到 `Enabled`
   - `Ec2HandleUserData` 到 `Enabled`
6. 断开远程桌面会话，并返回 Amazon EC2 控制台。
7. 在 Instances 列表中，停止实例。
8. 实例停止后，在 Amazon EC2 控制台中，右键单击实例列表中的实例，选择图像，然后选择创建图像。
9. 在 Create Image 页面中，提供映像的名称和描述，并指定卷配置。完成后，选择 Create Image。
10. 打开 Images 页面，等待映像从 pending 阶段变为 available。新的 AMI 已准备就绪，可供使用。

使用自定义 Windows AMI 添加新实例

您的映像变为 available 状态后，就可以创建基于自定义 Windows AMI 的新实例。当您从 Operating system (操作系统) 列表中选择 Use custom Windows AMI (使用自定义 Windows AMI) 时， AWS OpsWorks Stacks 显示自定义 AMI 的列表。

添加基于自定义 Windows AMI 的新实例

1. 当您的新 AMI 可用时，前往 AWS OpsWorks 堆栈控制台，打开 Windows 堆栈的 “实例” 页面，然后选择页面底部附近的 + Instance 来添加新实例。
2. 在 New 选项卡上，选择 Advanced。
3. 在 Operating system 下拉列表中，选择 Use custom Windows AMI。
4. 在 Custom AMI 下拉列表中，选择您创建的 AMI，然后选择 Add Instance。

现在您就可以启动和运行该实例了。

手动启动、停止和重启全天候实例

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 您可以将全天候实例用于 Linux 和 Windows 堆栈。

将全天候实例添加到层后，必须手动启动该实例以启动相应的 Amazon Elastic Compute Cloud (Amazon EC2) 实例，并手动停止该实例以终止 Amazon EC2 实例。您也可以手动重启无法正常运行的实例。 AWS OpsWorks 堆栈会自动启动和停止基于时间和基于负载的实例。有关更多信息，请参阅使用基于时间和基于负载的实例管理负载。

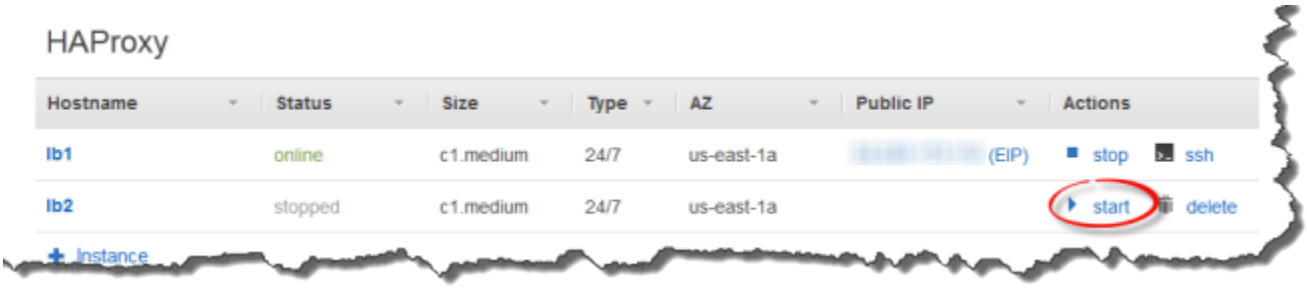
> **Important**
>
> AWS OpsWorks 只能在控制台中启动、停止和重启堆栈实例。 AWS OpsWorks AWS OpsWorks 无法识别在 Amazon EC2 控制台中执行的启动、停止或重启操作。

主题

- 启动或重新启动实例
- 停止实例
- 重启实例

启动或重新启动实例

要启动新实例，请在 Instances 页面上单击实例的 Actions 列中的 start。



您还可以创建多个实例，然后单击 Start all Instances 将它们同时全部启动。

启动实例后， AWS OpsWorks Stacks 会启动 Amazon EC2 实例并启动操作系统。启动过程通常需要几分钟的时间，而且通常对于 Windows 实例来说要比 Linux 实例慢。在启动过程中，实例的 Status 字段会显示下面一系列值：

1. 已@@ 请求- AWS OpsWorks Stacks 已调用亚马逊 EC2 服务来创建亚马逊 EC2 实例。
2. 待处理- AWS OpsWorks 堆栈正在等待 Amazon EC2 实例启动。
3. booting - Amazon EC2 实例正在启动。
4. running_setup- AWS OpsWorks Stacks 已触发安装事件并正在运行该层的Setup配方，然后运行其配方。Deploy有关更多信息，请参阅 执行配方。如果您已将自定义食谱添加到堆栈中， AWS OpsWorks Stacks 会在运行Setup和Deploy食谱之前安装存储库中的当前版本。
5. online - 实例已准备就绪，可供使用。

当 Status 变为 online 时，说明实例已在全面运行。

- 如果该层连接了负载均衡器，则 AWS OpsWorks Stacks 会向其添加实例。
- AWS OpsWorks Stacks 会触发一个Configure事件，该事件会运行每个实例的Configure配方。

  根据需要，这些配方更新实例以容纳新的实例。
- AWS OpsWorks Stacks 将实例的启动操作替换为停止，您可以使用停止操作来停止实例。

如果实例未成功启动或 Setup 配方失败，则状态将分别设置为 start_failed 或 setup_failed。您可以检查日志来查明原因。有关更多信息，请参阅 调试和故障排除指南。

停止的实例仍然是堆栈的一部分且保留所有资源。例如，Amazon EBS 卷和弹性 IP 地址仍然与停止的实例关联。可以通过在实例的操作列中选择启动来重新启动停止的实例。重启停止的实例时将执行以下操作：

- 实例存储支持的实例 — AWS OpsWorks Stacks 启动具有相同配置的新 Amazon EC2 实例。
- Amazon EBS 支持的实例 — AWS OpsWorks Stacks 会重新启动 Amazon EC2 实例，该实例会重新连接根卷。

实例完成启动后， AWS OpsWorks Stacks 会安装操作系统更新并运行Setup和Deploy配方，就像初始启动一样。 AWS OpsWorks Stacks 还会根据需要对重启的实例执行以下操作。

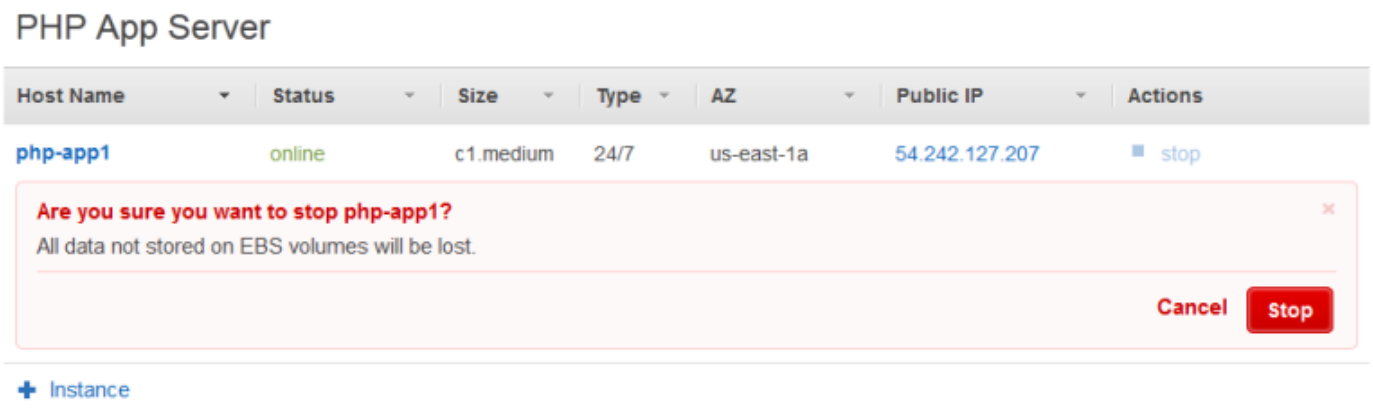
- 重新关联弹性 IP 地址。
- 重新挂载 Amazon Elastic Block Store（Amazon EBS）卷
- 对于实例存储支持的实例，安装最新的说明书版本。

  Amazon EBS 支持的实例继续使用存储在根卷上的自定义说明书。如果自您停止实例以来您的自定义说明书已改变，则在实例联机后您必须手动更新说明书。有关更多信息，请参阅 更新自定义说明书。

> **Note**
>
> 可能需要花费几分钟时间，弹性 IP 地址才能重新关联重启的实例。请注意，实例的 Elastic IP 设置代表元数据，并且只表示地址应与实例关联。Public IP 设置反映实例的状态并且最初可能为空。当弹性 IP 地址与实例相关联时，地址被分配给 Public IP 设置，后跟 (EIP)。

停止实例

在实例页面上，单击实例操作列中的停止，这将通知 AWS OpsWorks Stacks 运行关闭配方并终止 EC2 实例。



您也可以通过单击 Stop All Instances 来关闭堆栈中的所有实例。

停止实例后， AWS OpsWorks Stacks 会执行多项任务：

1. 如果实例的层附加了 Elastic Load Balancing 负载均衡器，则 AWS OpsWorks Stacks 会取消注册该实例。

   如果层支持负载均衡器的连接耗尽功能，则 AWS OpsWorks Stacks 会推迟触发 Shutdown 事件，直到连接耗尽完成。有关更多信息，请参阅 Elastic Load Balancing 层。
2. AWS OpsWorks Stacks 会触发一个Shutdown事件，该事件会运行实例的Shutdown配方。
3. 触发Shutdown事件后， AWS OpsWorks Stacks 会等待指定的时间让Shutdown配方有时间完成，然后执行以下操作：
   - 终止实例存储支持的实例，这将删除所有数据。
   - 停止 Amazon EBS 支持的实例，这将保留根卷上的数据。

   有关实例存储的更多信息，请参阅存储。

> **Note**
>
> 默认关闭超时设置为 120 秒。如果您的 Shutdown 配方需要更多时间，则可以编辑层配置来更改设置。

您可以通过查看实例的 Status 列来监控关闭过程。在关闭过程中，会显示以下一系列值：

1. 正在终止- AWS OpsWorks Stacks 正在终止 Amazon EC2 实例。
2. shutting_down- AWS OpsWorks Stacks 正在运行该层的配方。Shutdown

3. 已终止：Amazon EC2 实例已终止。
4. stopped - 实例已停止。

重启实例

在 Instances 页面上，单击不工作的实例的名称以打开详细信息页面，然后单击 Reboot。

此命令执行相关 Amazon EC2 实例的软重启。此操作不会删除实例的数据，即使对于实例存储支持的实例也是如此，并且不触发任何生命周期事件。

> **Note**
>
> 要让 AWS OpsWorks Stacks 自动替换失败的实例，请启用 auto 修复。有关更多信息，请参阅 使用自动修复。

使用基于时间和基于负载的实例管理负载

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

由于传入流量不尽相同，因此您的堆栈可能要么实例太少而无法从容处理负载，要么实例太多超出了需要的数量。通过使用基于时间或基于负载的实例来自动增加或减少层的实例，让您总是有足够的实例来妥善处理传入流量而无需为不需要的容量付费，从而同时节约时间和金钱。不需要监控服务器负载或手动启动或停止实例。此外，基于时间和基于负载的实例自动在一个区域内的多个可用区中分发、扩展和平衡应用程序，从而为您提供地理冗余和可扩展性。

自动扩展基于两种实例类型，它们基于不同的标准调整层的联机实例数：

- 基于时间的实例

  允许堆栈通过包括只在特定时间或特定日期运行的实例来处理遵循一定可预测模式的负载。例如，您可以在下午 6 点后启动某些实例来执行夜间备份任务，或在周末流量较低时停止某些实例。

- 基于负载的实例

  允许堆栈基于任意负载指标，通过在流量高时启动额外实例，在流量低时停止实例，以此来处理变化的负载。例如，您可以让 AWS OpsWorks 堆栈在平均 CPU 利用率超过 80% 时启动实例，并在平均 CPU 负载低于 60% 时停止实例。

Linux 堆栈支持基于时间和基于负载的实例，而 Windows 堆栈仅支持基于时间的实例。

与必须手动启动和停止的全天候实例不同，您无需自己启动或停止基于时间或基于负载的实例，相反，您可以配置实例， AWS OpsWorks 堆栈会根据其配置启动或停止它们。例如，您可以将基于时间的实例配置为按指定的计划启动和停止。 AWS OpsWorks 然后，堆栈会根据该配置启动和停止实例。

常见的做法是将全部三种实例类型综合起来使用，如下所示。

- 一组全天候实例，用来处理基本负载。您通常只要启动这些实例并让它们持续运行即可。
- 一组基于时间的实例， AWS OpsWorks Stacks 启动和停止这些实例以处理可预测的流量变化。例如，如果流量在工作时间达到最高，则应配置基于时间的实例在早上启动晚上关闭。
- 一组基于负载的实例， AWS OpsWorks Stacks 启动和停止这些实例以应对不可预测的流量变化。 AWS OpsWorks 当负载接近堆栈全天候和基于时间的实例的容量时，堆栈会启动它们，并在流量恢复正常时停止堆栈。

有关如何使用这些扩展时间的更多信息，请参阅优化服务器数。

> **Note**
>
> 如果您已经为实例层创建了应用程序或创建了自定义食谱， AWS OpsWorks Stacks 会在首次启动时自动将最新版本部署到基于时间和基于负载的实例。但是， AWS OpsWorks Stacks 不一定会将最新的食谱部署到重启的离线实例。有关更多信息，请参阅 编辑应用程序 和 更新自定义说明书。

主题

- 使用基于时间的自动扩展

- 使用基于负载的自动扩展
- 基于负载的扩展与自动修复有何不同

使用基于时间的自动扩展

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

基于时间的扩展允许您通过按指定计划启动或停止实例，来控制图层在一天中的特定时间或一周中的几天应在线的实例数量。 AWS OpsWorks Stacks 每隔几分钟检查一次，并根据需要启动或停止实例。您为每个实例指定单独的计划，如下所示：

- 一天中的时间。例如，您可以让多个实例在白天而不是在晚上运行。
- 一星期中的日子。例如，您可以让多个实例在工作日而不是周末运行。

> **Note**
>
> 不能指定特定的日期。

主题

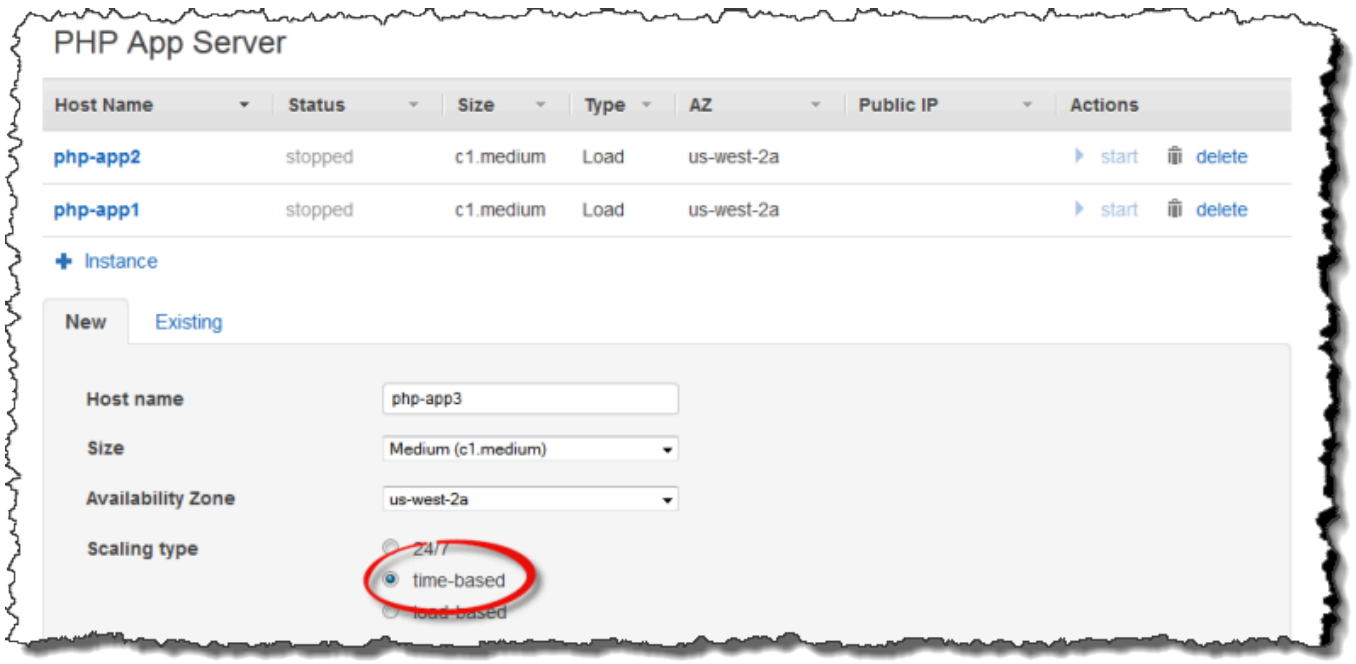
- 向层中添加基于时间的实例
- 配置基于时间的实例

向层中添加基于时间的实例

您既可以向层中添加基于时间的新实例，也可以使用现有实例。

添加基于时间的新实例

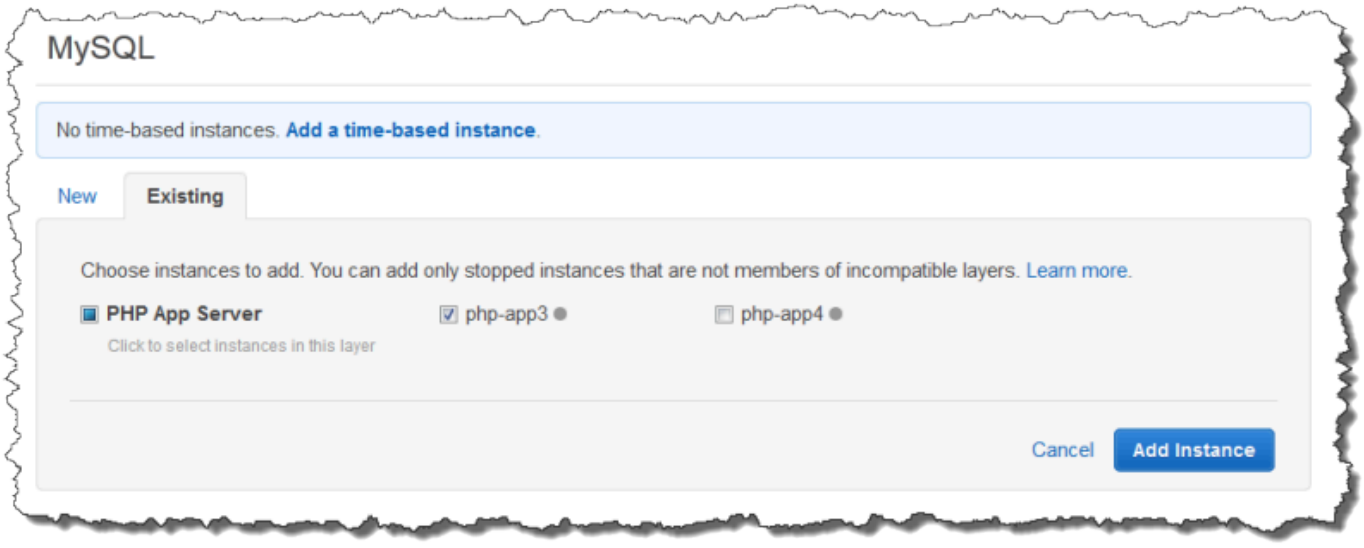
1. 在实例页面上，选择+实例添加一个实例。在新建选项卡上，选择高级，然后选择基于时间。



2. 配置实例。然后选择添加实例将实例添加到层。

向层中添加基于时间的现有实例

1. 在基于时间的实例页面上，如果某层已有基于时间的实例，则选择+实例。否则，选择添加基于时间的实例。然后选择现有选项卡。



2. 在现有选项卡上，从列表中选择一个实例。该列表仅显示基于时间的实例。

> **Note**
>
> 如果您改变了主意，不想使用现有实例，可在新建选项卡按照上一步骤所述创建一个新实例。

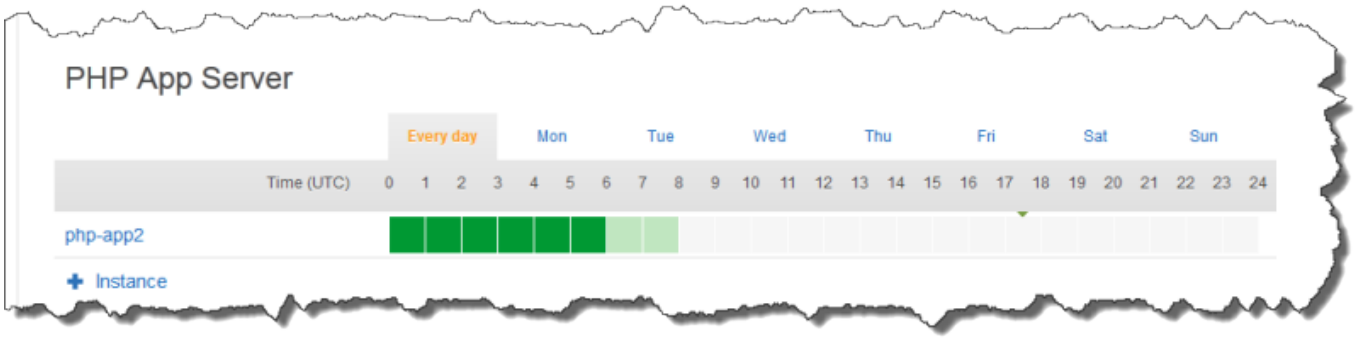
3. 选择添加实例将实例添加到层。

配置基于时间的实例

将基于时间的实例添加到层后，按如下配置其计划。

配置基于时间的实例

1. 在导航窗格中，选择实例下的基于时间。
2. 通过填写所需时间下方的相应框，为每个基于时间的实例指定在线时间段。
   - 要每天使用相同的计划，请单击每天选项卡，然后指定在线时间段。
   - 要在不同的日子使用不同的计划，请选择相应的日期并选择适当的时间段。



> **Note**
>
> 请务必留出启动实例所需的时间，并且 AWS OpsWorks Stacks 仅每隔几分钟检查一次，以确定应启动还是停止实例。例如，如果某个实例应在 UTC 时间 1:00 前运行，则应在 UTC 时间 0:00 启动它。否则， AWS OpsWorks Stacks 可能要等到世界标准时间 1:00 之后几分钟才能启动实例，并且实例需要几分钟才能上线。

您可以通过执行上述步骤随时更改实例的在线时间段。下次 AWS OpsWorks Stacks 检查时，它会使用新的计划来确定是启动还是停止实例。

> **Note**
>
> 您也可以向层中添加基于时间的新实例，方法是打开基于时间页面，单击添加基于时间的实例 (如果您尚未向层中添加任何基于时间的实例) 或+实例 (如果该层已有一个或多个基于时间的实例)。然后，按照上一步骤所述配置实例。

使用基于负载的自动扩展

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

基于负载的实例允许您快速启动或停止实例，以响应传入流量的变化。 AWS OpsWorks Stacks 使用 Amazon CloudWatch 数据计算每个层的以下指标，这些指标表示该层所有实例的平均值：

- CPU：平均 CPU 消耗，如 80%
- 内存：平均内存消耗，如 60%
- 负载：一个系统在一分钟内的平均计算工作。

您可以为这些指标中的任何一个或全部定义规模升级 和规模降级 阈值。您也可以使用自定义 CloudWatch 警报作为阈值。

超出阈值会触发扩展事件。通过指定以下值来确定 AWS OpsWorks Stacks 如何响应扩展事件：

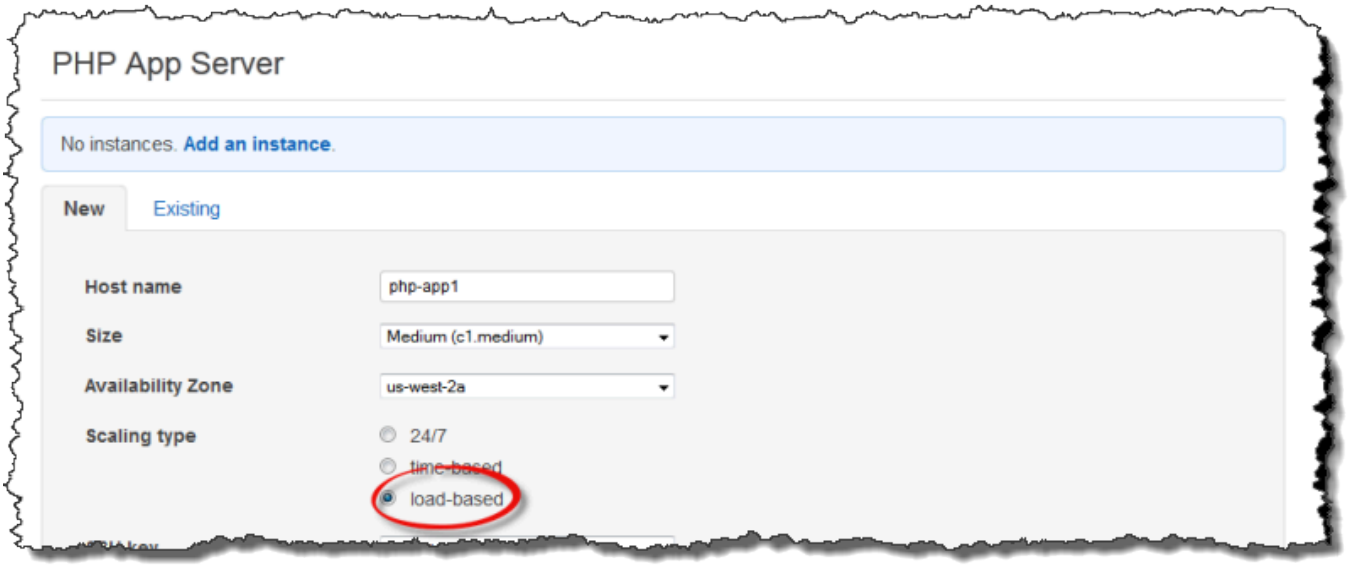
- 启动或停止多少个实例。
- AWS OpsWorks 堆栈在超过阈值后应等待多长时间才能启动或删除实例。例如，CPU 利用率必须超过阈值至少 15 分钟。该值允许您忽略短暂的流量波动。
- AWS OpsWorks Stacks 在启动或停止实例后应等待多长时间才能再次监控指标。通常，需要留有足够的时间来让启动的实例上线，或让停止的实例关闭，然后再评估层是否仍超出阈值。

发生扩展事件时，AWS OpsWorks Stacks 仅启动或停止基于负载的实例。而不启动或停止全天候实例或基于时间的实例。

> **Note**
>
> 基于负载的自动扩展不创建新实例；它仅启动和停止那些已创建的实例。因此，您必须提前配置足够的基于负载的实例来处理预期的最大负载。

创建基于负载的实例

1. 在实例页面上，选择+实例添加一个实例。选择高级，然后选择基于负载。



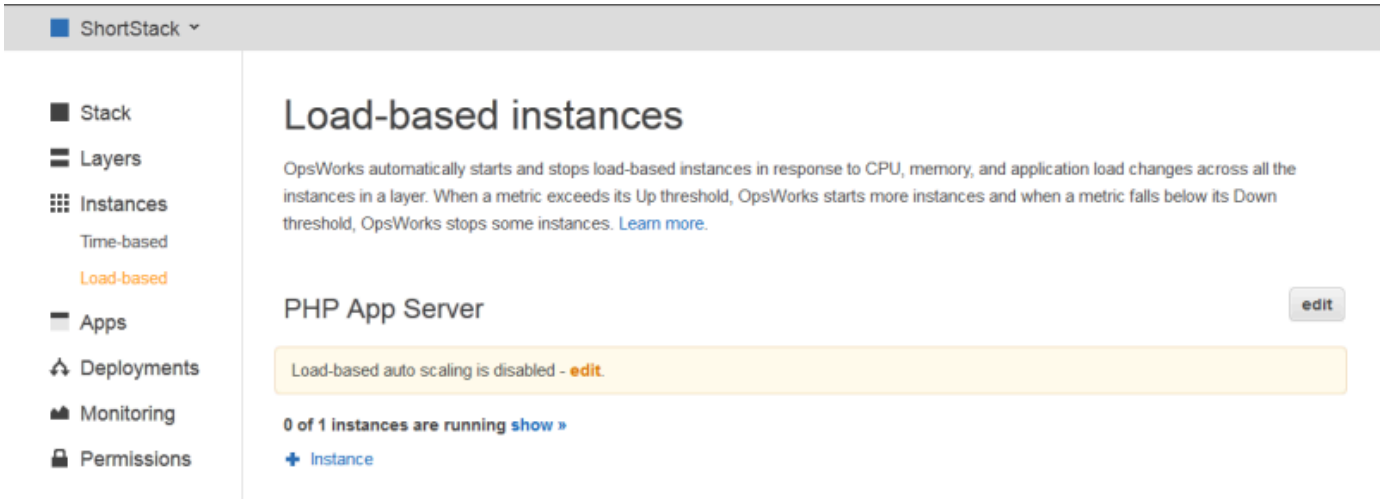
2. 配置实例，然后选择添加实例将实例添加到层。

重复此步骤，直到创建了足够数量的实例。稍后，您可以根据需要添加或移除实例。

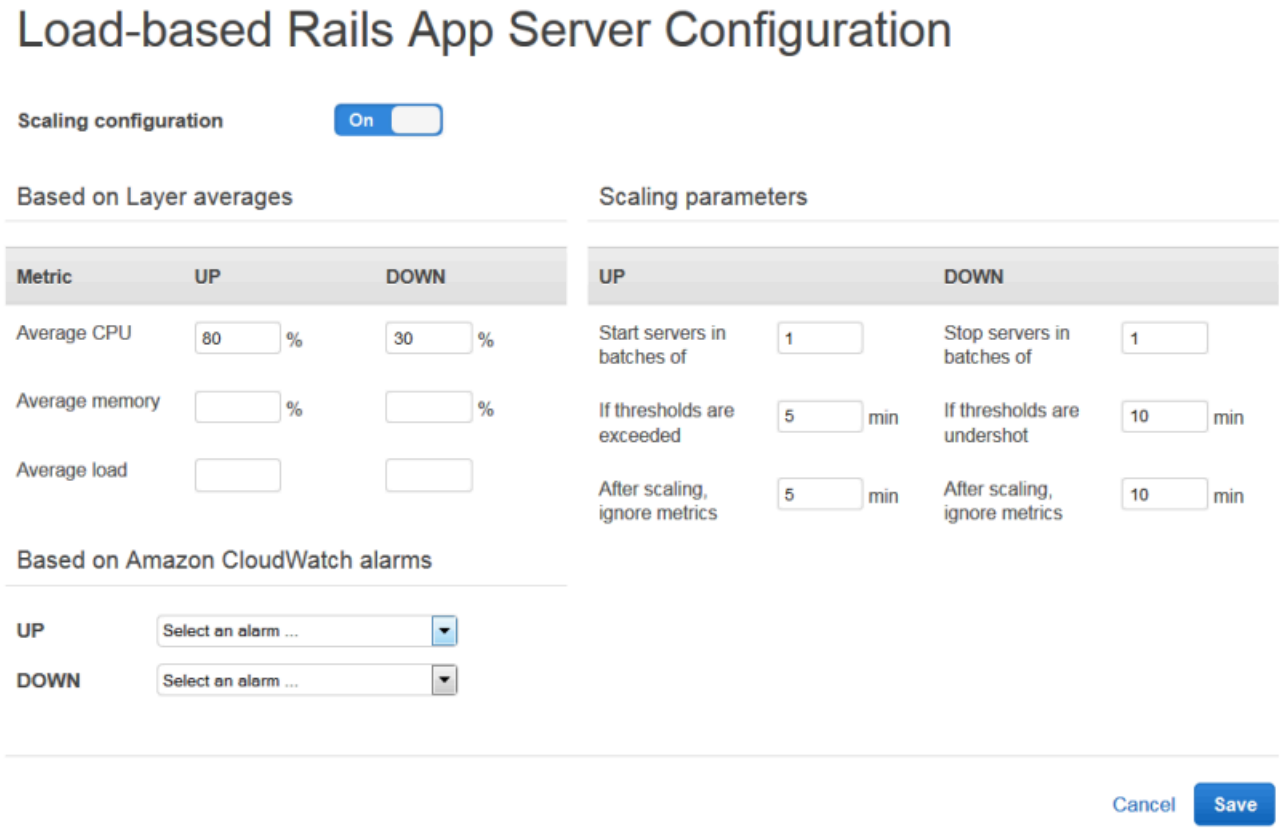
将基于负载的实例添加到层后，必须启用基于负载的扩展并指定配置。基于负载的扩展配置是一种层属性，而非实例属性，它指定一个层什么时候应启动或停止其基于负载的实例。必须为每个使用基于负载的实例的层单独指定该属性。

启用和配置基于负载的自动扩展

1. 在导航窗格中，单击实例下的基于负载，并针对相应的层单击编辑。



2. 将启用基于负载的自动扩缩设置为开启。然后，设置阈值和扩展参数来定义如何以及何时添加或删除实例。



层-平均阈值

您可以基于以下值设置扩展阈值，这些值为层所有实例的平均值。

- 平均 CPU：层的平均 CPU 使用率，用占总量的百分比表示。
- 平均内存：层的平均内存使用率，用占总量的百分比表示。

- 平均负载：层的平均负载。

  有关负载计算方法的更多信息，请参阅维基百科的负载（计算）。

超过阈值会导致扩展事件，如果需要更多实例，则会扩大规模，如果需要更少的实例，则会缩小规模。 AWS OpsWorks 然后，堆栈会根据扩展参数添加或删除实例。

自定义 CloudWatch 警报

您最多可以使用五个自定义 CloudWatch 警报作为扩大或缩小阈值。这些警报必须和堆栈在同一个区域。有关如何创建自定义警报的更多信息，请参阅创建 Amazon CloudWatch 警报。

> **Note**
>
> 要使用自定义警报，必须更新您的服务角色以允许 `cloudwatch:DescribeAlarms`。你可以让 AWS OpsWorks Stacks 在你第一次使用此功能时为你更新角色，也可以手动编辑角色。有关更多信息，请参阅 允许 AWS OpsWorks Stacks 代表你行事。
> 当为基于负载的配置配置了多个警报时，如果警报处于 `INSUFFICIENT_DATA` 指标警报状态，则即使另一个警报处于 `ALARM` 状态，也无法进行基于负载的实例扩展。只有当所有警报都处于 `OK` 或 `ALARM` 状态时，才能继续进行自动扩缩。有关使用亚马逊 CloudWatch 警报的更多信息，请参阅亚马逊 CloudWatch 用户指南中的使用亚马逊 CloudWatch 警报。

扩展参数

以下参数控制 AWS OpsWorks 堆栈管理扩展事件的方式。

- 批量启动的服务器数：当发生扩展事件时要添加或移除的实例数量。
- 如果超过阈值 — 在 AWS OpsWorks 堆栈触发扩展事件之前，负载必须保持在升级阈值或缩减阈值之下的时间（以分钟为单位）。
- 缩放后，忽略指标 — 扩展事件发生后， AWS OpsWorks 堆栈应忽略指标并抑制其他扩展事件的时间（以分钟为单位）。

  例如， AWS OpsWorks Stacks 在发生升级事件后会添加新实例，但这些实例在启动和配置后才会开始减少负载。引发更多的扩展事件直到新的实例在线并处理请求，这样的做法没有意义，实例在线和处理请求通常需要花费几分钟的时间。该设置允许您指示 AWS OpsWorks Stacks 抑制扩展事件足够长的时间以便让新的实例在线。

您可以提高此设置，防止当平均 CPU、平均内存或平均负载等层平均值临时偏离时扩展突然波动。

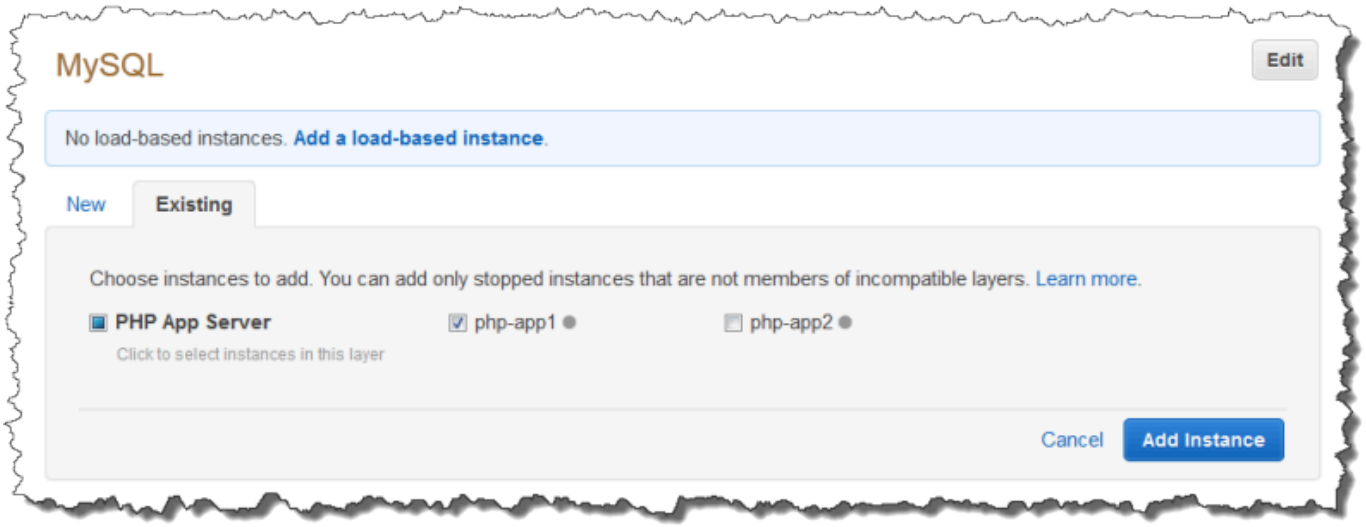
例如，如果 CPU 利用率超出限制并且内存使用率接近规模降级，则实例规模升级事件发生之后会紧接着发生内存规模降级事件。为防止这种情况发生，您可以增加扩展之后，忽略指标设置中的分钟数。在这个例子中，虽然会发生 CPU 扩展，但不会发生内存规模降级事件。

3. 要添加额外的基于负载的实例，请单击+实例，配置设置，然后单击添加实例。重复此步骤，直到您有足够的基于负载的实例来处理您的预期最高负载。然后选择保存。

> **Note**
>
> 您也可以向层中添加新的基于负载的实例，方法是打开基于负载页面，单击添加基于负载的实例 (如果您尚未向层中添加任何基于负载的实例) 或 +实例 (如果该层已有一个或多个基于负载的实例)。然后，如本节前面所述配置实例。

向层中添加基于负载的现有实例

1. 在导航窗格中，选择实例下的基于负载。
2. 如果您已为层启用了基于负载的自动扩展，则单击 +实例。否则，选择添加基于负载的实例。选择“现有” 选项卡。



3. 在现有选项卡上，选择一个实例。该列表仅显示基于负载的实例。

> **Note**
>
> 如果您改变了主意，不想使用现有实例，可在新建选项卡按照上一步骤所述创建一个新实例。

4. 选择添加实例将实例添加到层。

您可以随时修改基于负载的自动扩展的配置或将其禁用。

禁用基于负载的自动扩展

1. 在导航窗格中，单击实例下的基于负载，并针对相应的层单击编辑。
2. 将启用基于负载的自动扩缩切换为否。

基于负载的扩展与自动修复有何不同

基于负载的自动扩展使用所有正在运行的实例的平均负载指标。如果指标保持在指定的阈值之间，则 AWS OpsWorks Stacks 不会启动或停止任何实例。另一方面，借助 auto healin AWS OpsWorks g，当实例停止响应时，Stacks 会自动启动具有相同配置的新实例。旧实例可能因网络问题或一些与其关联的问题而无法响应。

例如，假设您的 CPU 规模升级阈值为 80%，并且有一个实例停止了响应。

- 如果禁用了自动修复，并且其余正在运行的实例可以将平均 CPU 利用率保持在 80% 以下，则 AWS OpsWorks Stacks 不会启动新实例。只有当其余实例的平均 CPU 利用率超过 80% 时，它才启动替换实例。
- 如果启用了自动修复，则无论负载阈值如何， AWS OpsWorks Stacks 都会启动替换实例。

使用在 AWS OpsWorks Stacks 外部创建的计算资源

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 仅 Linux 堆栈支持此功能。

实例 介绍如何使用 AWS OpsWorks Stacks 创建和管理多组 Amazon Elastic Compute Cloud (Amazon EC2) 实例。您还可以将 Linux 计算资源整合到堆栈之外创建的 AWS OpsWorks 堆栈中：

- 直接使用 Amazon EC2 控制台、CLI 或 API 创建的 Amazon EC2 实例。
- 您自己的硬件上运行的本地 实例，包括虚拟机上运行的实例。

这些计算资源变成 AWS OpsWorks 堆栈管理的实例，你可以像管理普通 AWS OpsWorks 堆栈实例一样管理它们：

- 管理用户权限：您可以使用 AWS OpsWorks Stacks 用户管理来指定允许哪些用户访问您的堆栈，允许他们对堆栈实例执行哪些操作，以及他们是否拥有 SSH 访问权限和 sudo 权限。
- 自动执行任务 — 你可以让 AWS OpsWorks Stacks 运行自定义 Chef 配方来执行任务，例如使用单个命令在堆栈的任何或所有实例上执行脚本。

  如果您将实例分配给某个层， AWS OpsWorks Stacks 会在实例生命周期的关键时刻（包括您的自定义食谱）自动在实例上运行一组指定的 Chef 食谱。请注意，您只能将注册的 Amazon EC2 实例分配给自定义层。
- 管理资源 — 堆栈允许您在中对资源进行分组和管理 AWS 区域， OpsWorks 控制面板显示所有区域的堆栈状态。
- 安装软件包：您可以使用 Chef 配方在堆栈的任何实例上安装软件包。
- 更新操作系统 — AWS OpsWorks Stacks 提供了一种在堆栈实例上安装操作系统安全补丁和更新的简单方法。
- 部署应用程序 — AWS OpsWorks Stacks 将应用程序一致地部署到堆栈的所有应用程序服务器实例。
- 监控 — AWS OpsWorks Stacks 创建自定义CloudWatch指标来监控您的所有堆栈实例。

有关定价信息，请参阅 AWS OpsWorks 定价。

以下是使用注册实例的基本步骤。

1. 向堆栈注册实例。

该实例现在是堆栈的一部分，由 AWS OpsWorks Stacks 管理。

2. 也可将实例分配给某个层。

   通过此步骤，您可以充分利用 AWS OpsWorks 堆栈管理功能。您可以将注册的本地实例分配给任何层；注册的 Amazon EC2 实例只能分配给自定义层。

3. 使用 AWS OpsWorks 堆栈管理实例。
4. 当您不再需要堆栈中的实例时，请取消注册该实例，这将从 AWS OpsWorks 堆栈中移除该实例。

以下部分详细介绍了此过程。

主题

使用 AWS OpsWorks 堆栈堆栈注册实例

Important

该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

Note

仅 Linux 堆栈支持此功能。

要注册 AWS OpsWorks 堆栈之外的实例，请运行 AWS CLI aws opsworks register命令。您可以从要注册的实例或其他计算机上运行此命令。您可以将`AWSOpsWorksRegisterCLI_EC2`或`AWSOpsWorksRegisterCLI_OnPremises`策略应用于用户

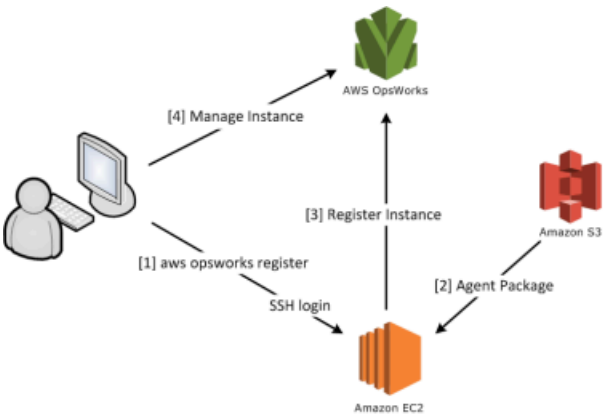
或组，以分别授予注册 EC2 或本地实例所需的权限。 AWS CLI 这些策略需要版本为 1.16.180 AWS CLI 或更高版本。

> **Note**
>
> 要防止用户或角色注册实例，请更新实例配置文件以拒绝访问 register 命令。

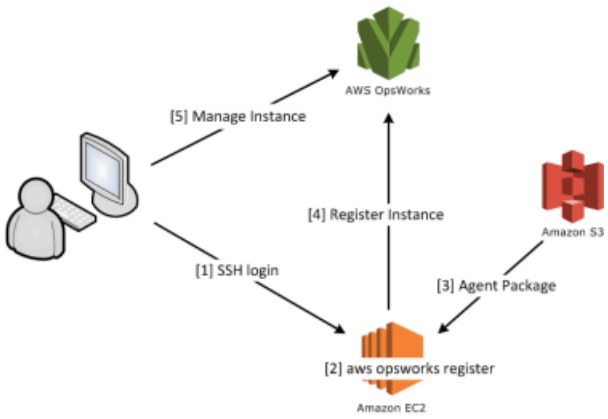
注册过程会在您要使用堆栈管理的实例上安装代理，然后使用 AWS OpsWorks 您指定的 AWS OpsWorks 堆栈注册该实例。注册实例后，实例是堆栈的一部分，由 AWS OpsWorks Stacks 管理。有关更多信息，请参阅 管理注册的实例。

> **Note**
>
> 尽管 AWS 工具 PowerShell包含调用 `register` API 操作的 `Register-OpsInstance`cmdlet，但我们建议您改用 AWS CLI 来运行该`register`命令。

下图显示了注册 Amazon EC2 实例的两种方法。您可以使用同样的方法注册本地实例。



Registering an instance from a workstation



Registering an instance from the instance itself

> **Note**
>
> 您可以使用 AWS OpsWorks Stacks 控制台来管理注册的实例，但必须运行 AWS CLI `register` 命令来注册该实例。此要求的原因是：注册过程必须从实例中运行，而无法通过控制台来完成。

以下部分详细介绍了该过程。

主题

演练：从工作站注册实例

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 仅 Linux 堆栈支持此功能。

注册过程支持多几种方案。本节将向您介绍一个场景的 end-to-end 示例：如何使用您的工作站注册 Amazon EC2 实例。其他注册场景使用类似的过程。有关更多信息，请参阅 注册 Amazon EC2 和本地实例。

> **Note**
>
> 您通常希望注册现有的 Amazon EC2 实例。不过，您可以只为本演练创建新实例和新堆栈，然后在完成后将其删除。

主题

步骤 1：创建堆栈和实例

首先，您需要一个堆栈，以及一个将在该堆栈中注册的 Amazon EC2 实例。

创建堆栈和实例

1. 使用 AWS OpsWorks Stacks 控制台创建一个新堆栈（名为 **EC2Register**）。您可以接受其他堆栈设置的默认值。
2. 从 Amazon EC2 控制台启动一个新实例。请注意以下几点。

   - 该实例必须与堆栈位于同一区域和 VPC。

     如果您使用的是 VPC，则为本演练选择一个公有子网。
   - 如果您需要创建 SSH 密钥，则将私有密钥文件保存到工作站，并记录名称和文件位置。

     如果您使用现有密钥，则记录名称和私有密钥文件位置。稍后将需要这些值。
   - 该实例必须基于某个受支持的 Linux 操作系统。例如，如果您的堆栈位于美国西部（俄勒冈州），则可使用 `ami-35501205` 启动该区域内的 Ubuntu 14.04 LTS 实例。

   否则，请接受默认值。

当该实例启动时，您可以继续下一部分。

步骤 2：安装和配置 AWS CLI

使用 AWS CLI aws opsworks register命令执行注册。在注册第一个实例之前，您必须运行版本为 1.16.180 AWS CLI 或更高版本。安装详细信息取决于您工作站的操作系统。有关安装的更多信息 AWS CLI，请参阅安装 AWS 命令行界面。要检查您正在运行的 AWS CLI 版本，请在 shell 会话中输入 `aws --version`。

> **Note**
>
> 要防止用户或角色注册实例，请更新实例配置文件以拒绝访问 register 命令。

我们强烈建议您不要跳过此步骤，即使您已经在工作站 AWS CLI 上运行了。使用最新版本的 AWS CLI 是安全最佳实践。

您必须为 `register` 提供一组具有相应权限的 AWS 凭证。为了避免直接在实例上安装凭证，推荐的方法是注册使用实例配置文件启动的实例，然后将 `--use-instance-profile` 开关添加到您的 `register` 命令中。如果要从实例配置文件获取凭证，请跳至本主题中的步骤 3：向 EC2Register 堆

栈注册实例。但是，如果未使用实例配置文件启动实例，可以创建 IAM 用户。以下过程创建具有相应权限的新用户，在工作站上安装该用户的凭证，然后将这些凭证传递给 `register`。

> **Warning**
>
> IAM 用户拥有长期证书，这会带来安全风险。为帮助减轻这种风险，我们建议仅向这些用户提供执行任务所需的权限，并在不再需要这些用户时将其移除。

创建用户

1. 在 IAM 控制台的导航窗格中选择用户，然后选择添加用户。
2. 添加名为 **`EC2Register`** 的用户。
3. 选择下一步。
4. 在设置权限页面上，选择 直接附加策略。
5. 在“权限策略筛选器”框**`OpsWorks`**中输入以显示 AWS OpsWorks 策略，选择以下策略之一，然后选择“下一步：查看”。此策略授予用户运行 `register` 时所需的权限。
   - 选择 `AWSOpsWorksRegisterCLI_EC2` 以允许注册使用实例配置文件的 EC2 实例的用户权限。
   - 选择 `AWSOpsWorksRegisterCLI_OnPremises` 以允许注册本地实例的用户权限。
6. 选择下一步。
7. 在审核页面上，选择创建用户。
8. 现在为您的用户创建访问密钥。从导航窗格中，选择用户，然后选择要为其创建访问密钥的用户。
9. 选择安全凭证选项卡，然后选择创建访问密钥。
10. 选择最符合您任务的访问密钥最佳实践和替代方法。
11. 选择下一步。
12. （可选）输入标识访问密钥的标签。
13. 选择下一步。
14. 选择下载 .csv 文件，将凭证文件保存到系统中的方便位置，然后选择完成。

您需要向 `register` 提供 IAM 用户的凭证。本演练通过在工作站的 `credentials` 文件中安装 EC2Register 凭证来处理该任务。有关管理凭证的其他方法的信息 AWS CLI，请参阅配置和凭据文件。

安装用户的凭证

1. 创建或打开工作站的 `credentials` 文件。文件位于 `~/.aws/credentials` (Linux、Unix 和 OS X) 或 `C:\Users\`*`User_Name`*`\.aws\credentials` (Windows 系统)。
2. 使用以下格式将 EC2Register 用户的配置文件添加到 `credentials` 文件。

   ```
   [ec2register]
   aws_access_key_id = access_key_id
   aws_secret_access_key = secret_access_key
   ```

   将 *`access_key_id`* 和 *`secret_access_key`* 替换为您之前下载的 EC2Register 密钥。

步骤 3：向 EC2Register 堆栈注册实例

您现在可以注册实例。

注册实例

1. 在 AWS OpsWorks 堆栈中，返回 ec2Register 堆栈，在导航窗格中选择实例，然后选择注册实例。
2. 选择 EC2 Instances (EC2 实例)，选择 Next: Select Instances (下一步: 选择实例)，然后从列表中选择您的实例。
3. 选择 “下一步：安装 AWS CLI”，然后选择 “下一步：注册实例”。 AWS OpsWorks 堆栈会自动使用堆栈 ID 和实例 ID 等可用信息来创建`register`命令模板，该模板显示在 “注册实例” 页面上。对于本示例，您将通过 `register` 使用 SSH 密钥登录该实例并明确指定密钥文件，因此应将 I use SSH keys to connect to my instances (我使用 SSH 密钥连接到我的实例) 设置为 Yes (是)。命令模板类似于以下内容。

   ```
   aws opsworks register --infrastructure-class ec2 --region region endpoint ID
     --stack-id 247be7ea-3551-4177-9524-1ff804f453e3 --ssh-username [username]
     --ssh-private-key [key-file] i-f1245d10
   ```

> **Note**
>
> 如果堆 AWS OpsWorks 栈位于与区域终端节点关联的经典区域内，则必须将区域设置为堆栈服务的终端节点区域，而不是堆栈的us-east-1区域。 AWS OpsWorks 堆栈根据堆栈 ID 确定堆栈的区域。

4. 命令模板包含多个用户特定的参数值，这些值通过方括号指示且必须用合适的值替换。将命令模板复制到文本编辑器，并按如下方式进行编辑。

> **Important**
>
> 在已注册实例的整个生命周期中，都需要注册过程中创建的 IAM 用户。删除用户会导致 AWS OpsWorks Stacks 代理无法与服务通信。为了帮助防止在用户被意外删除时无法正常管理已注册实例，请将 `--use-instance-profile` 参数添加到您的 `register` 命令，以便使用实例的内置实例配置文件。添加该`--use-instance-profile`参数还可以防止在每 90 天轮换 AWS 账户访问密钥时发生错误（这是推荐的最佳做法），因为它可以防止 AWS OpsWorks 代理可用的访问密钥与所需的 IAM 用户之间的访问密钥不匹配。

- 将*####*替换为您创建该实例时保存的 Amazon EC2 密钥对的私有密钥文件的完全限定路径。

  如果您愿意，可以使用相对路径。
- 将 *username* 替换为该实例的用户名称。

  在本示例中，对于 Ubuntu 实例，用户名称是 ubuntu；对于 Red Hat Enterprise Linux (RHEL) 或 Amazon Linux 实例，用户名称是 ec2-user。
- 添加 `--use-instance-profile`，它使用实例配置文件运行 `register`，以防止在密钥轮换期间出错或者主体 IAM 用户被意外删除。

您的命令应与以下内容类似。

```
aws opsworks register --use-instance-profile --infrastructure-class ec2 \
  --region us-west-2  --stack-id 247be7ea-3551-4177-9524-1ff804f453e3 --ssh-
username ubuntu \
  --ssh-private-key "./keys/mykeys.pem" i-f1245d10
```

5. 在您的工作站上打开一个终端窗口，粘贴来自编辑器的 `register` 命令，然后运行该命令。

注册通常需要大约五分钟时间。完成后，返回 AWS OpsWorks Stacks 控制台并选择完成。然后在导航窗格中选择 Instances (实例)。您的实例应在 Unassigned Instances 下列出。然后，您可以将该实例分配给某个层或将其留在原位，具体取决于您打算如何管理该实例。

6. 完成后，停止实例，然后使用 AWS OpsWorks Stacks 控制台或命令将其删除。这将终止 Amazon EC2 实例，这样就不会再产生任何进一步费用。

注册 Amazon EC2 和本地实例

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 仅 Linux 堆栈支持此功能。

本部分介绍如何向 AWS OpsWorks Stacks 堆栈注册 Amazon EC2 或本地实例。

主题

- 准备实例
- 安装和配置 AWS CLI
- 注册实例
- 使用 register 命令
- 示例 register 命令
- 实例注册策略

准备实例

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 仅 Linux 堆栈支持此功能。

注册实例之前，必须确保它与 AWS OpsWorks Stacks 兼容。详细信息取决于您是注册本地实例还是 Amazon EC2 实例。

本地实例

本地实例必须满足以下条件：

- 该实例必须运行某个受支持的 Linux 操作系统。虽然可以使用已经从自定义或社区生成的 AMI 创建的其他操作系统（例如 CentOS 6.x）创建或注册实例，但这些实例不受官方支持。

  您必须在该实例上安装 `libyaml` 软件包。对于 Ubuntu 实例，软件包名称为 `libyaml-0-2`。对于 CentOS 和 Red Hat Enterprise Linux 实例，软件包名称为 `libyaml`。
- 该实例必须拥有一个受支持的实例类型 (有时称为实例大小)。受支持的实例类型可能因操作系统而异，并且取决于您的堆栈是否位于 VPC 中。有关支持的实例类型的列表，请查看当您尝试在目标堆栈中创建新实例时， AWS OpsWorks 堆栈控制台中显示的 Si z e 下拉列表值。如果某个实例类型灰显，而无法在您的目标堆栈中创建，那么您无法注册该类型的实例。
- 实例必须具有互联网访问权限才能与 AWS OpsWorks Stacks 服务终端节点通信。`opsworks.us-east-1.amazonaws.com` (HTTPS)该实例还必须支持与 AWS 资源 (如 Amazon S3) 的出站连接。
- 如果您计划从独立的工作站注册实例，则注册的实例必须支持从该工作站进行 SSH 登录。

  如果您从该实例运行注册命令，则不需要进行 SSH 登录。
- AWS 访问密钥用于从 AWS OpsWorks 代理向 AWS OpsWorks Stacks 服务进行身份验证。如果您按照建议每 90 天轮换一次访问密钥，请手动更新 AWS OpsWorks 代理以使用新密钥。在本地计算

机或实例上，使用新的访问密钥和私有密钥编辑 `/etc/aws/opsworks/instance-agent.yml` 文件。以下命令显示了此文件中的访问密钥和私有密钥。使用旧密钥的代理可能会导致错误。

```
cat /etc/aws/opsworks/instance-agent.yml | egrep "access_key|secret_key"
:access_key_id: AKIAIOSFODNN7EXAMPLE
:secret_access_key: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

Amazon EC2 实例

Amazon EC2 实例必须满足以下条件：

- AMI 必须基于某个受支持的 Linux 操作系统。有关当前列表，请参阅 AWS OpsWorks 堆栈操作系统。

  有关更多信息，请参阅 使用自定义 AMI。

  如果该实例基于从受支持的标准 AMI 派生的自定义 AMI，或者该实例包含非常少的设置，则您必须在该实例上安装 `libyaml` 软件包。对于 Ubuntu 实例，软件包名称为 `libyaml-0-2`。对于 Amazon Linux 和 Red Hat Enterprise Linux 实例，软件包名称为 `libyaml`。
- 该实例必须拥有一个受支持的实例类型 (有时称为实例大小)。受支持的实例类型可能因操作系统而异，并且取决于您的堆栈是否位于 VPC 中。有关支持的实例类型的列表，请查看当您尝试在目标堆栈中创建新实例时， AWS OpsWorks 堆栈控制台中显示的 Si z e 下拉列表值。如果某实例类型灰显，而无法在您的目标堆栈中创建，那么您也无法注册该类型的实例。
- 该实例必须处于 `running` 状态。
- 该实例不得是 Auto Scaling 组的一部分。

  有关更多信息，请参阅从 Auto Scaling 组分离 EC2 实例。
- 该实例可以是 VPC 的一部分，但它必须与堆栈位于同一 VPC 中，并且必须将 VPC 配置为可以正常使用 AWS OpsWorks 堆栈。
- 不支持 Spot 实例，因为它们不支持自动修复。

当您注册 Amazon EC2 实例时， AWS OpsWorks Stacks 不会修改该实例的安全组或规则。请确保实例的安全组规则符合以下 AWS OpsWorks Stacks 要求。

入口规则

入口规则应允许以下操作。

- SSH 登录。
- 来自相应层的流量。

  例如，数据库服务器通常允许来自堆栈的应用程序服务器层的入站流量。
- 流向相应端口的流量。

  例如，应用程序服务器实例通常允许所有流向端口 80 (HTTP) 和 443 (HTTPS) 的入站流量。

出口规则

出口规则应允许以下操作。

- 从实例上运行的应用程序流向 AWS OpsWorks Stacks 服务的流量。
- 来自使用 AWS API 的应用程序的流量，用于访问 AWS 资源 (如 Amazon S3)。

一种常见方法是不指定任何出口规则，这样就不会对出站流量施加限制。

安装和配置 AWS CLI

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

在注册第一个实例之前，您必须在运行的计算机上运行版本 1.16.180 AWS CLI 或更高版本。`register`安装详细信息取决于您工作站的操作系统。有关安装的更多信息 AWS CLI，请参阅安装 AWS 命令行界面和配置 AWS 命令行接口。要检查您正在运行的 AWS CLI 版本，请在 shell 会话中输入 `aws --version`。

> **Note**
>
> 尽管 AWS 工具 PowerShell包含调用 `register` API 操作的 `Register-OpsInstance`cmdlet，但我们建议您改用 AWS CLI 来运行该`register`命令。

您必须具有相应权限，才可运行 `register`。您可以通过使用 IAM 角色，或者通过在要注册的工作站或实例上安装具有适当权限的用户凭证来获取权限，但这种方法并不理想。

然后，您可以使用这些凭证来运行 `register`，如下文所述。通过将 IAM policy 附加到用户或角色来指定权限。对于 `register`，请附加 `AWSOpsWorksRegisterCLI_EC2` 或 `AWSOpsWorksRegisterCLI_OnPremises` 策略，它们分别授予注册 Amazon EC2 或 本地实例的权限。

> **Note**
>
> 如果您在 Amazon EC2 实例上运行 `register`，最好使用 IAM 角色来提供凭证。有关如何将 IAM 角色附加到现有实例的更多信息，请参阅《Amazon EC2 用户指南》中的将 IAM 角色附加到实例或替换 IAM 角色。

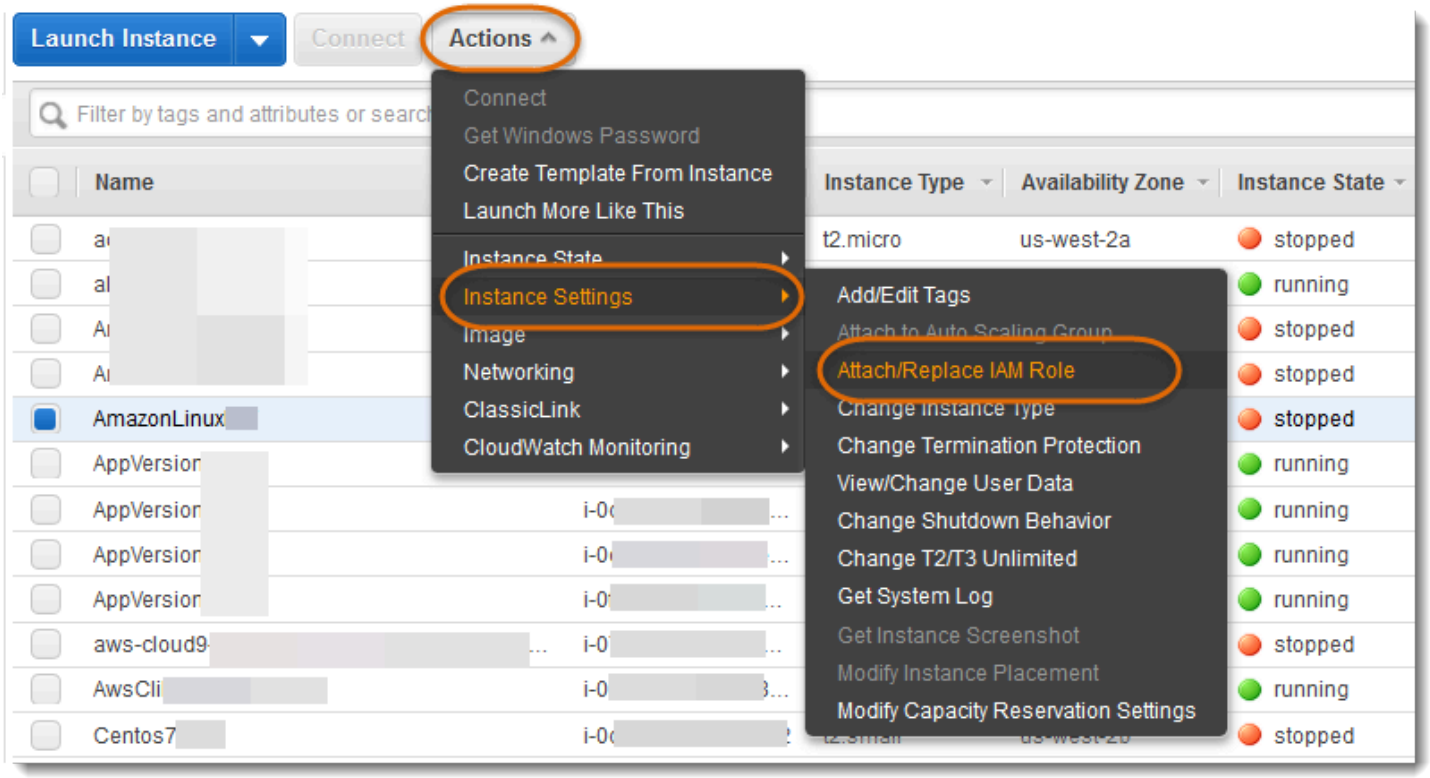
有关 `AWSOpsWorksRegisterCLI_EC2` 和 `AWSOpsWorksRegisterCLI_OnPremises` 策略的示例代码段，请参阅 实例注册策略。有关创建和管理 AWS 凭证的更多信息，请参阅 AWS 安全凭证。

主题

- 使用 IAM 角色
- 使用安装的凭证

使用 IAM 角色

如果您从打算注册的 Amazon EC2 实例运行命令，则向 `register` 提供凭证的首选策略是使用附加了 `AWSOpsWorksRegisterCLI_EC2` 策略或等效权限的 IAM 角色。此方法可以避免在实例上安装您的凭证。执行此操作的一种方法是在 EC2 控制台中使用 Attach/Replace IAM Role (附加/替换 IAM 角色) 命令，如下图所示。



有关如何将 IAM 角色附加到现有实例的更多信息，请参阅《Amazon EC2 用户指南》中的将 IAM 角色附加到实例或替换 IAM 角色。对于使用实例配置文件启动的实例（推荐），请向您的 `register` 命令添加 `--use-instance-profile` 开关以提供凭证；不要使用 `--profile` 参数。

如果实例正在运行并且具有角色，您可以通过将 `AWSOpsWorksRegisterCLI_EC2` 策略附加到该角色来授予所需权限。该角色将提供实例的一组默认凭证。只要您尚未在实例上安装任何凭证，`register` 便会自动承担该角色并使用其权限运行。

> **Important**
>
> 我们建议您不要在实例上安装凭证。除了造成安全风险外，实例的角色还位于默认提供者链的末端，AWS CLI 用于查找默认证书。安装的凭证可能优先于该角色，因此，`register` 可能不具有所需权限。有关更多信息，请参阅 AWS CLI入门。

如果运行中的实例没有角色，则您必须在实例上安装具有所需权限的凭证，如使用安装的凭证中所述。建议使用通过实例配置文件启动的实例，这样更容易、更不容易出错。

使用安装的凭证

有几种方法可以在系统上安装用户凭据并将其提供给 AWS CLI 命令。下面介绍了一种不再推荐的方法，但是如果要注册在没有实例配置文件的情况下启动的 EC2 实例，则可以使用该方法。您还可以使用现有 用户的凭证，只要附加的策略授予所需权限即可。有关更多信息，包括关于安装凭证的其他方法的说明，请参阅配置和凭证文件。

使用安装的凭证

1. 创建 IAM 用户，然后将访问密钥 ID 和秘密访问密钥保存在安全位置。

   > **Warning**
   >
   > IAM 用户拥有长期证书，这会带来安全风险。为帮助减轻这种风险，我们建议仅向这些用户提供执行任务所需的权限，并在不再需要这些用户时将其移除。

2. 将 AWSOpsWorksRegisterCLI_OnPremises 策略附加到用户。如果愿意，您可以附加一个授予更广泛权限的策略，只要它包含 `AWSOpsWorksRegisterCLI_OnPremises` 权限即可。
3. 在系统的 `credentials` 文件中创建该用户的配置文件。文件位于 `~/.aws/credentials` (Linux、Unix 和 OS X) 或 `C:\Users\`*`User_Name`*`\.aws\credentials` (Windows 系统)。该文件包含一个或多个以下格式的配置文件，每个配置文件均包含用户的访问密钥 ID 和秘密访问密钥。

   ```
   [profile_name]
   aws_access_key_id = access_key_id
   aws_secret_access_key = secret_access_key
   ```

   用您之前保存的 IAM 凭证替换 *`access_key_id`* 和 *`secret_access_key`* 值。您可以指定任何喜欢的名称作为配置文件名称，但有两个限制：该名称必须唯一，并且默认配置文件必须命名为 `default`。您也可以使用现有配置文件，只要它具有所需权限即可。
4. 使用 `register` 命令的 `--profile` 参数指定配置文件名称。`register` 命令将使用授予关联凭证的权限运行。

   您也可以省略 `--profile`。在这种情况下，`register` 将使用默认凭证运行。请注意，这些不一定是默认配置文件的凭证，因此，您必须确保默认凭证具有所需权限。有关如何 AWS CLI 确定默认证书的更多信息，请参阅配置 AWS 命令行界面。

注册实例

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 仅 Linux 堆栈支持此功能。

您可以通过从您的工作站或实例运行 AWS CLI `register` 命令来注册实例。处理该操作的最简单方法是使用 AWS OpsWorks Stacks 控制台的注册向导，这可简化构建命令字符串的流程。熟悉注册过程后，如果愿意，您可以跳过该向导并运行 `register` 命令。

下文介绍了如何使用注册向导向现有堆栈注册实例。

> **Note**
>
> 要使用新堆栈注册实例，您可以通过在 AWS OpsWorks 堆栈控制面板上选择注册实例来完成此操作。这将启动一个向导 (与用于现有堆栈的向导相同，只不过多了一个用于配置新堆栈的页面)。

使用注册向导来注册实例

1. 在 AWS OpsWorks Stacks 控制台中，创建一个堆栈或者打开现有堆栈。
2. 在导航窗格中选择 Instances，然后选择 register an instance。
3. 在选择一个实例类型页面，指定是要注册 Amazon EC2 实例还是本地实例：
   - 如果要注册 Amazon EC2 实例，请选择下一步：选择实例。
   - 如果要注册本地实例，则选择下一步：安装 AWS CLI，然后转到步骤 5。
4. 如果您正在注册 Amazon EC2 实例，请打开选择实例页面选择要注册的实例。 AWS OpsWorks Stacks 收集生成命令所需的信息。在您完成后，选择下一步：安装 AWS CLI。

5. 您计划运行的实例register必须运行版本为 1.16.180 AWS CLI 或更高版本。注册向导页面提供了指向安装和配置说明的链接，用于安装或更新 AWS CLI。验证 AWS CLI 安装后，请指定是在从要注册的实例运行命令，还是从独立的工作站运行命令，然后选择 Next: Register Instances (下一步: 注册实例)。

6. Register Instances 页面将显示 `register` 命令字符串的模板，其中包含您选择的选项。例如，如果您要从独立的工作站注册 Amazon EC2 实例，则默认模板类似于以下示例。

```
aws opsworks register --infrastructure-class ec2 --region us-west-2
  --stack-id 247be7ea-3551-4177-9524-1ff804f453e3 --ssh-username [username] i-
f1245d10
```

> **Important**
>
> 在已注册实例的整个生命周期中，都需要注册过程中创建的 IAM 用户。删除用户会导致 AWS OpsWorks Stacks 代理无法与服务通信。为了帮助防止在用户被意外删除时无法正常管理已注册实例，请将 `--use-instance-profile` 参数添加到您的 `register` 命令，以便使用实例的内置实例配置文件。添加该`--use-instance-profile`参数还可以防止在每 90 天轮换 AWS 账户访问密钥时发生错误（这是推荐的最佳做法），因为它可以防止 AWS OpsWorks 代理可用的访问密钥与所需的 IAM 用户之间的访问密钥不匹配。

如果您将 “我使用 SSH 密钥” 设置为 “是”， AWS OpsWorks Stacks 会将`--ssh-private-key`参数添加到字符串中，您可以使用该参数来指定 SSH 私钥文件。

> **Note**
>
> 如果您想让 `register` 使用密码登录，则将 I use SSH keys 设置为 No。当您运行 `register` 时，会提示您输入密码。

将此字符串复制到文本编辑器，并根据需要进行编辑。请注意以下几点。

- 方括号中的文本表示您必须提供的信息，例如 SSH 密钥文件的位置。
- 该模板假定您正在使用默认 AWS 凭证运行 `register`。如果不是，请向命令字符串添加一个 `--profile` 参数，并指定您要使用的凭证配置文件名称。

对于其他场景，您可能需要进一步更改命令。有关可用 `register` 参数的说明以及构建命令字符串的其他方法，请参阅使用 `register` 命令。您也可以通过从命令行运行 `aws opsworks help register` 来显示命令的说明文档。如需查看一些示例命令字符串，请参阅示例 register 命令。

7. 编辑完命令字符串后，请在您的工作站上打开一个终端窗口，或使用 SSH 登录到实例并运行该命令。整个操作通常需要大约五分钟时间，在此期间，该实例处于 Registering 状态。
8. 操作完成后，请选择 Done。该实例现在处于 Registered 状态，并在堆栈的 Instances 页面上作为未分配实例列出。

`register` 命令执行以下操作。

1. 如果 `register` 在工作站上运行，则该命令首先使用 SSH 登录到要注册的实例。

   此过程的剩余部分将在该实例上发生，并且无论您在哪里运行命令，步骤都相同。
2. 从 Amazon S3 下载 AWS OpsWorks Stacks 代理软件包。
3. 解包并安装代理及其依赖项，例如适用于 Ruby 的 AWS SDK。
4. 创建以下内容：
   - 一个 IAM 用户，它使用 AWS OpsWorks Stacks 服务引导代理以提供安全通信。

     该用户的权限仅允许 `opsworks:RegisterInstance` 操作，并且这些权限在 15 分钟后过期。
   - 堆栈的 IAM 组，其中包含已注册实例的用户。
5. 创建 RSA 密钥对并将公钥发送到 AWS OpsWorks Stacks。

   此密钥对用于加密代理与 AWS OpsWorks Stacks 之间的通信。
6. 使用 AWS OpsWorks 堆栈注册实例。该堆栈随后运行一组初始设置配方来配置实例，其中包括以下内容。
   - 覆盖实例的主机文件。

     通过注册实例，您已将用户管理移交给 AWS OpsWorks Stacks，Stacks 必须拥有自己的主机文件才能控制 SSH 登录权限。
   - 对于 Amazon EC2 实例，初始设置还包括向堆栈注册任何挂载的 Amazon EBS 卷或弹性 IP 地址。

     您必须确保 Amazon EBS 卷未挂载到预留的挂载点，包括 `/var/www` 以及实例各层预留的任何挂载点。有关管理堆栈资源的更多信息，请参阅资源管理。有关层挂载点的更多信息，请参阅AWS OpsWorks 堆栈图层参考。

有关初始设置配置更改的完整介绍，请参阅初始设置配置更改。

> **Note**
>
> 初始设置不会更新已注册实例的操作系统；您必须自行处理该任务。有关更多信息，请参阅管理安全更新。

使用 **register** 命令

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 仅 Linux 堆栈支持此功能。

要注册实例，请确保至少运行 AWS CLI的 1.16.180 版。以下示例显示了 `register` 命令的一般语法。

```
aws opsworks register \
  [--profile profile_name] \
  [--region region_name] \
  --infrastructure-class instance_type \
  --stack-id stack ID \
  [--local] | [--ssh-private-key key_file --ssh-username username] | [--override-
ssh command_string] \
  [--override-hostname hostname] \
  [--debug] \
  [--override-public-ip public IP] \
  [--override-private-ip private IP] \
..[--use-instance-profile] \
  [ [IP address] | [hostname] | [instance ID]
```

以下参数是所有 AWS CLI 命令的通用参数。

`--profile`

(可选) 凭证的配置文件名称。如果省略此参数，命令将使用默认凭证运行。有关如何 AWS CLI 确定默认证书的更多信息，请参阅配置 AWS 命令行界面。

`--region`

（可选）AWS OpsWorks Stacks 服务终端节点的区域。请勿设置`--region`为堆栈的区域。 AWS OpsWorks 堆栈会根据堆栈 ID 自动确定堆栈的区域。

> **Note**
>
> 如果已设置默认区域，则可省略此参数。有关如何指定默认区域的更多信息，请参阅配置 AWS 命令行界面。

对 Amazon EC2 实例和本地实例使用以下参数。

`--infrastructure-class`

(必需) 必须将此参数设置为 `ec2` 或 `on-premises`，以分别指明注册的是 Amazon EC2 实例，还是本地实例。

`--stack-id`

(必需) 注册实例时将使用的堆栈的 ID。

> **Note**
>
> 要查找堆栈 ID，请在 Stack (堆栈) 页面上选择 Settings (设置)。堆栈 ID 被标记为 OpsWorks ID，是一个看起来像`ad21bce6-7623-47f1-bf9d-af2affad8907`的 GUID。

SSH 登录参数

使用以下参数指定 `register` 应如何登录到实例。

`--local`

(可选) 使用此参数注册您在其中运行该命令的实例。

在这种情况下，register 不需要登录到实例。

--ssh-private-key 和 --ssh-username

(可选) 如果您要从独立的工作站注册实例，并希望明确指定用户名称或私有密钥文件，则使用这些参数。

- --ssh-username：使用此参数指定 SSH 用户名称。

  如果省略 --ssh-username，ssh 将使用默认用户名称。

- --ssh-private-key：使用此参数明确指定私有密钥文件。

  如果省略 --ssh-private-key，ssh 将尝试使用无需密码的身份验证技术进行登录，包括使用默认私有密钥。如果这些技术均不受支持，ssh 将查询您的密码。有关 ssh 如何处理身份验证的更多信息，请参阅 Secure Shell (SSH) 身份验证协议。

--override-ssh

(可选) 如果您要从独立的工作站注册实例，并希望指定自定义 ssh 命令字符串，则使用此参数。register 命令使用此命令字符串登录到注册的实例。

有关 ssh 的更多信息，请参阅 SSH。

--override-hostname

（可选）为实例指定主机名，该名称仅由 AWS OpsWorks Stacks 使用。默认值为实例的主机名。

--debug

(可选) 如果注册过程失败，则提供调试信息。有关故障排除信息，请参阅对实例注册进行故障排除。

--use-instance-profile

（可选，但强烈建议对 Amazon EC2 实例使用）让 register 命令使用附加的实例配置文件，而不是创建 IAM 用户。如果您在 IAM 用户已被意外删除时尝试管理已注册实例，则会出现错误，而添加此参数可帮助防止此类错误。

> **Important**
>
> 在已注册实例的整个生命周期中，都需要注册过程中创建的 IAM 用户。删除用户会导致 AWS OpsWorks Stacks 代理无法与服务通信。为了帮助防止在用户被意外删除时无法正常管理已注册实例，请将 --use-instance-profile 参数添加到您的 register 命令，以便使用实例的内置实例配置文件。添加该--use-instance-profile参数还可以防止

在每 90 天轮换 AWS 账户访问密钥时发生错误（这是推荐的最佳做法），因为它可以防止 AWS OpsWorks 代理可用的访问密钥与所需用户之间的访问密钥不匹配。

目标

(条件性) 如果您从工作站运行此命令，则命令字符串中的最终值按以下任一方式指定注册目标。

- 实例的公有 IP 地址。
- 实例的主机名。
- 对 Amazon EC2 实例 为 实例 ID。

AWS OpsWorks Stacks 使用实例 ID 来获取实例配置，包括实例的公有 IP 地址。默认情况下，AWS OpsWorks Stacks 使用此地址来构造用于登录实例的`ssh`命令字符串。如果您需要连接到私有 IP 地址，则必须使用 `--override-ssh` 提供自定义命令字符串。有关示例，请参阅从工作站注册本地实例。

> **Note**
>
> 如果您指定主机名，则 `ssh` 依赖 DNS 服务器将该名称解析为特定实例。如果您不确定主机名是否唯一，则使用 `ssh` 验证该主机名是否解析为正确的实例。

如果您从要注册的实例运行此命令，则省略实例标识符，改为使用 `--local` 参数。

以下参数仅适用于本地实例。

`--override-public-ip`

（可选）AWS OpsWorks Stacks 将指定地址显示为实例的公有 IP 地址。它不会更改实例的公有 IP 地址。但是，如果用户使用控制台连接到实例，例如通过在 “实例” 页面上选择地址， AWS OpsWorks Stacks 将使用指定的地址。 AWS OpsWorks 堆栈会自动确定参数的默认值。

`--override-private-ip`

（可选）AWS OpsWorks Stacks 将指定地址显示为实例的私有 IP 地址。它不会更改实例的私有 IP 地址。 AWS OpsWorks 堆栈会自动确定参数的默认值。

示例 register 命令

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 仅 Linux 堆栈支持此功能。

本部分包含一些 `register` 命令字符串的示例。

从工作站注册 Amazon EC2 实例

以下示例从工作站注册 Amazon EC2 实例。该命令字符串使用默认凭证，并通过 Amazon EC2 实例 ID 来标识实例。您可以通过将 `ec2` 更改为 `on-premises` 来使用本地实例的示例。

```
aws opsworks register \
  --region us-west-2 \
  --use-instance-profile \
  --infrastructure-class ec2 \
  --stack-id ad21bce6-7623-47f1-bf9d-af2affad8907 \
  --ssh-user-name my-sshusername \
  --ssh-private-key "./keys/mykeys.pem" \
  i-2422b9c5
```

从工作站注册本地实例

以下示例从独立的工作站注册本地实例。该命令字符串使用默认凭证，并使用指定的 `ssh` 命令字符串登录到实例。如果您的实例需要密码，`register` 会提示您。您可以通过将 `on-premises` 更改为 `ec2` 来使用 Amazon EC2 实例的示例。

```
aws opsworks register \
  --region us-west-2 \
  --infrastructure-class on-premises \
  --stack-id ad21bce6-7623-47f1-bf9d-af2affad8907 \
```

```
  --override-ssh "ssh your-user@192.0.2.0"
```

> **Note**
>
> 您可以使用--override-ssh来指定任何自定义 SSH 命令字符串。 AWS OpsWorks 然后，Stacks 使用指定的字符串登录实例，而不是构造命令字符串。有关另一个示例，请参阅使用自定义 SSH 命令字符串注册实例。

使用自定义 SSH 命令字符串注册实例

以下示例从工作站注册本地实例，并使用 `--override-ssh` 参数指定 `register` 用于登录实例的自定义 SSH 命令。此示例使用 `sshpass` 通过用户名和密码登录，但您可以指定任何有效的 `ssh` 命令字符串。

```
aws opsworks register \
  --region us-west-2 \
  --infrastructure-class on-premises \
  --stack-id 2f92ff9d-04f2-4728-879b-f4283b40783c \
  --override-ssh "sshpass -p 'mypassword' ssh your-user@192.0.2.0"
```

通过从实例运行 **`register`** 来注册实例

以下示例显示了如何通过从实例本身运行 `register` 来注册 Amazon EC2 实例。该命令字符串的权限取决于默认凭证。要使用本地实例的示例，请将 `--infrastructure-class` 更改为 `on-premises`。

```
aws opsworks register \
  --region us-west-2 \
  --infrastructure-class ec2 \
  --stack-id ad21bce6-7623-47f1-bf9d-af2affad8907 \
  --local
```

使用私有 IP 地址注册实例

默认情况下，`register` 使用实例的公有 IP 地址登录到实例。要使用私有 IP 地址注册实例 (例如 VPC 的私有子网中的实例)，则必须使用 `--override-ssh` 指定自定义 `ssh` 命令字符串。

```
aws opsworks register \
  --region us-west-2 \
  --infrastructure-class ec2 \
```

```
--stack-id 2f92ff9d-04f2-4728-879b-f4283b40783c \
--override-ssh "ssh -i mykey.pem ec2-user@10.183.201.93" \
i-2422b9c5
```

实例注册策略

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

AWSOpsWorksRegisterCLI_EC2 和 AWSOpsWorksRegisterCLI_OnPremises 策略分别为注册 EC2 和本地实例提供了正确的权限。您将 AWSOpsWorksRegisterCLI_EC2 添加到您的 IAM 用户以注册 EC2 实例，但将 AWSOpsWorksRegisterCLI_OnPremises 添加到您的用户以注册本地实例。要使用这些策略，您必须运行至少版本 1.16.180 AWS CLI 或更高版本。

AWSOpsWorksRegisterCLI_EC2 策略

将 AWSOpsWorksRegisterCLI_EC2 添加到您的用户以注册 EC2 实例。如果您计划仅注册 EC2 实例，则应使用此配置文件。当您使用此策略时，EC2 实例的实例配置文件提供权限。

```
{
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Action": [
            "opsworks:AssignInstance",
            "opsworks:CreateLayer",
            "opsworks:DeregisterInstance",
            "opsworks:DescribeInstances",
            "opsworks:DescribeStackProvisioningParameters",
            "opsworks:DescribeStacks",
            "opsworks:UnassignInstance"
          ],
          "Resource": [
            "*"
          ]
        },
```

```
      {
        "Effect": "Allow",
        "Action": [
          "ec2:DescribeInstances"
        ],
        "Resource": [
          "*"
        ]
      }
    ]
  }
```

AWSOpsWorksRegisterCLI_OnPremises 策略

将 AWSOpsWorksRegisterCLI_OnPremises 添加到您的用户以注册本地实例。此策略包括 IAM 权限，例如 AttachUserPolicy，但这些权限起作用的资源是受限的。

```
  {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": [
          "opsworks:AssignInstance",
          "opsworks:CreateLayer",
          "opsworks:DeregisterInstance",
          "opsworks:DescribeInstances",
          "opsworks:DescribeStackProvisioningParameters",
          "opsworks:DescribeStacks",
          "opsworks:UnassignInstance"
        ],
        "Resource": [
          "*"
        ]
      },
      {
        "Effect": "Allow",
        "Action": [
          "ec2:DescribeInstances"
        ],
        "Resource": [
          "*"
        ]
```

```
        },
        {
          "Effect": "Allow",
          "Action": [
            "iam:CreateGroup",
            "iam:AddUserToGroup"
          ],
          "Resource": [
            "arn:aws:iam::*:group/AWS/OpsWorks/OpsWorks-*"
          ]
        },
        {
          "Effect": "Allow",
          "Action": [
            "iam:CreateUser",
            "iam:CreateAccessKey"
          ],
          "Resource": [
            "arn:aws:iam::*:user/AWS/OpsWorks/OpsWorks-*"
          ]
        },
        {
          "Effect": "Allow",
          "Action": [
            "iam:AttachUserPolicy"
          ],
          "Resource": [
            "arn:aws:iam::*:user/AWS/OpsWorks/OpsWorks-*"
          ],
          "Condition": {
            "ArnEquals":
              {
                "iam:PolicyARN": "arn:aws:iam::aws:policy/
AWSOpsWorksInstanceRegistration"
              }
            }
        }
      ]
    }
```

（已淘汰）AWSOpsWorksRegisterCLI 策略

> **Important**
>
> AWSOpsWorksRegisterCLI 策略已被淘汰，不能用于注册新实例。它仅适用于已注册的实例的向后兼容性。AWSOpsWorksRegisterCLI 策略包含许多 IAM 权限，包括 CreateUser、PutUserPolicy 和 AddUserToGroup。由于这些是管理员级权限，因此您应该仅将 AWSOpsWorksRegisterCLI 策略分配给受信任的管理用户。

管理注册的实例

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 仅 Linux 堆栈支持此功能。

当你注册一个实例时，它就会变成一个 AWS OpsWorks Stacks 实例，你可以用与使用 AWS OpsWorks Stacks 创建的实例大致相同的方式来管理它。主要有两个区别：

- 注册的实例无需分配给某个层。
- 您可以取消注册已注册的实例，将其恢复为直接由您控制。

注册实例后，该实例将处于 “已注册” 状态。 AWS OpsWorks Stacks 为所有注册的实例提供以下管理功能：

- He@@ alth chec ks — AWS OpsWorks Stacks 监控代理以评估实例是否继续运行。

  如果实例未通过运行状况检查， AWS OpsWorks Stack s 会自动修复已注册的 Amazon EC2 实例，并将注册的本地实例的状态更改为。connection lost
- CloudWatch 监 CloudWatch 控-已为注册实例启用监控。

您可以监控 CPU 使用率和可用内存等指标，并可在某项指标超过指定阈值时收到通知 (可选)。

- 用户管理 — AWS OpsWorks Stacks 提供了一种简单的方法来指定哪些用户可以访问实例以及允许他们执行哪些操作。有关更多信息，请参阅 管理用户权限。
- 配方执行：您可以使用执行配方堆栈命令对实例执行 Chef 配方。
- 操作系统更新：您可以使用更新依赖项堆栈命令更新实例的操作系统。

要充分利用 AWS OpsWorks 堆栈管理功能，您可以将实例分配给一个层。有关更多信息，请参阅 将注册的实例分配给某个层。

AWS OpsWorks Stacks 管理 Amazon EC2 的方式与本地实例之间存在差异。

Amazon EC2 实例

- 如果您停止注册的 Amazon EC2 实例， AWS OpsWorks Stacks 会终止由实例存储支持的实例并停止 Amazon EBS 支持的实例。

  该实例仍然在堆栈中注册并分配给堆栈的层，因此，您可以在需要时重新启动该实例。您必须取消注册已注册的实例以将其从堆栈中删除，要么明确取消注册，要么通过删除实例 (这可自动取消注册)。
- 如果您重新启动注册的 Amazon EC2 实例或者该实例失败并自动修复，则效果与使用 Amazon EC2 停止和重启实例相同。请注意以下区别：
  - 实例存储支持的实例 — AWS OpsWorks Stacks 使用相同的 AMI 启动一个新实例。

    请注意， AWS OpsWorks Stacks 不知道您在实例注册之前对其执行的任何操作，例如安装软件包。如果您希望 AWS OpsWorks Stacks 在启动时安装软件包或执行其他配置任务，则必须提供执行所需任务的自定义 Chef 配方，并将它们分配给相应层的安装事件。
  - Amazon EBS 支持的实例 — AWS OpsWorks Stacks 使用相同的 AMI 启动一个新实例，然后重新连接根卷，从而将实例恢复到以前的配置。
- 如果您取消注册已注册的 Amazon EC2 实例，它会恢复为常规 Amazon EC2 实例。

本地实例

- AWS OpsWorks 堆栈无法停止或启动已注册的本地实例。

  取消分配已注册的本地实例会触发关机事件。但是，该事件只运行已分配层的 Shutdown 配方。它们执行某些任务 (如关闭服务)，但不会停止该实例。
- AWS OpsWorks 如果已注册的本地实例失败，堆栈将无法对其进行自动修复，但该实例会被标记为连接丢失。

- 本地实例不能使用 Elastic Load Balancing Amazon EBS 或弹性 IP 地址服务。

将注册的实例分配给某个层

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 仅 Linux 堆栈支持此功能。

注册实例后，您可以将其分配给一个或多个层。将实例分配给图层而不是将其置于未分配状态的好处是，您可以为该层的生命周期事件分配自定义配方。 AWS OpsWorks 然后，在图层为该事件准备好配方之后，堆栈会在适当的时间自动运行它们。

- 您可以将任何注册的实例分配给自定义层。自定义层有一个不安装任何软件包的最小配方集，因此它们应该不会与实例的现有配置产生任何冲突。
- 您可以将本地实例分配给 AWS OpsWorks Stacks 内置层。

  每个内置层都包含可自动安装一个或多个软件包的配方。例如，Java App Server 设置配方安装 Apache 和 Tomcat。该层的配方也可能执行其他操作，例如重启服务和部署应用程序。将本地实例分配给内置层之前，应确保该层的配方不会产生任何冲突（例如尝试安装与实例上的当前版本不同的应用程序服务器）。有关更多信息，请参阅 图层 和 AWS OpsWorks 堆栈图层参考。

将注册的实例分配给某个层

1. 将您要使用的层添加到堆栈 (如果尚未执行此操作)。
2. 在导航窗格中单击实例，然后单击实例的操作列中的分配。
3. 选择相应的层，然后选择 Save (保存)。

当你为图层分配实例时， AWS OpsWorks Stacks 会执行以下操作。

- 运行该层的 Setup 配方。
- 将任何附加的弹性 IP 地址或 Amazon EBS 卷添加到堆栈的资源中。

然后，您可以使用 AWS OpsWorks Stacks 来管理这些资源。有关更多信息，请参阅 资源管理。

完成后，实例将处于联机状态并完全合并到堆栈中。 AWS OpsWorks 然后，每次发生生命周期事件时，堆栈都会运行图层分配的配方。

取消分配注册的实例

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

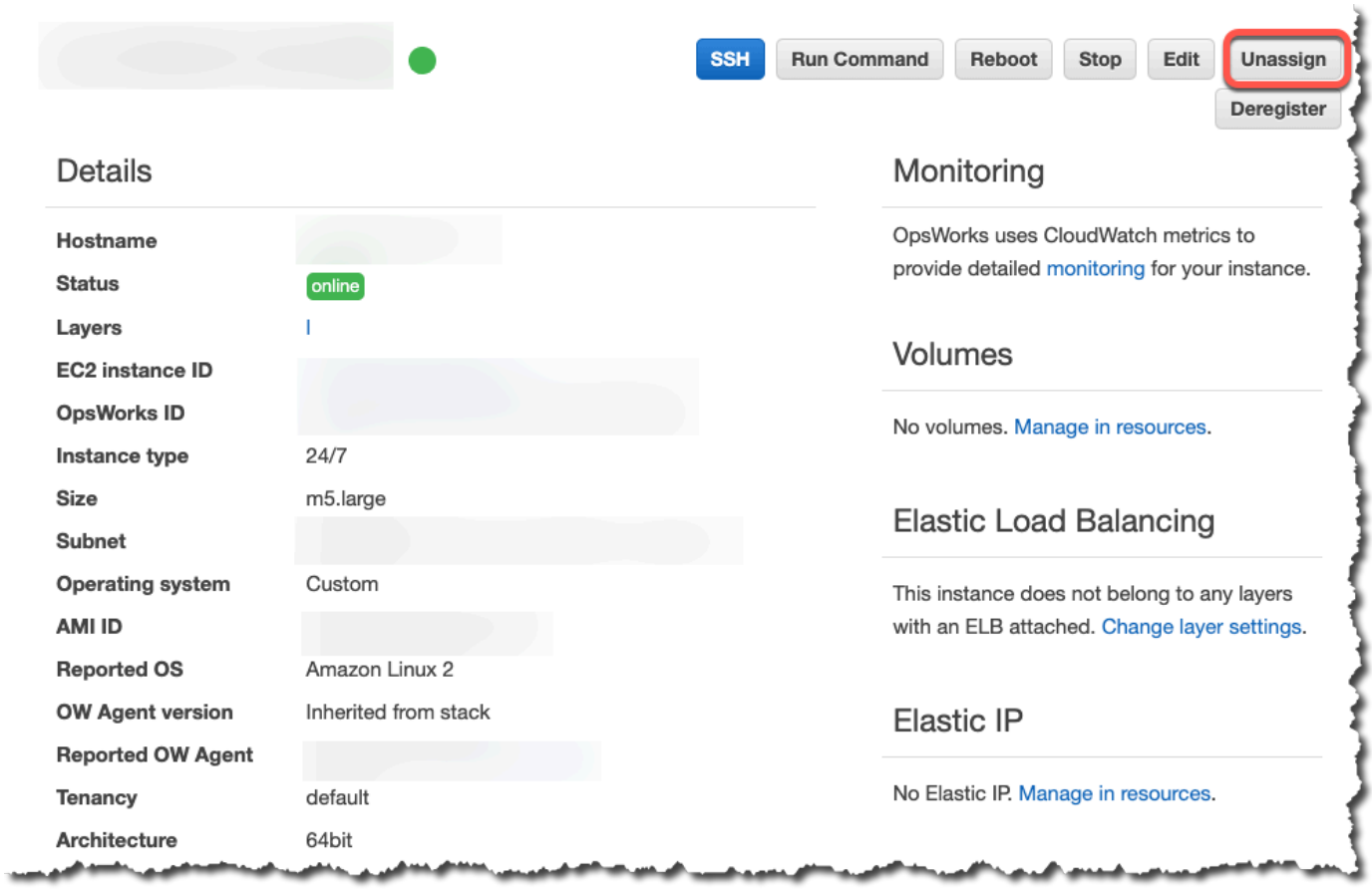
> **Note**
>
> 仅 Linux 堆栈支持此功能。

您可以使用 AWS OpsWorks 控制台、 AWS CLI或 SDK 操作将注册实例从其层中取消分配。

取消分配实例时， AWS OpsWorks Stacks 会在该实例上运行该层的关闭配方。这些配方执行某些任务 (如关闭服务)，但不会停止该实例。如果该实例被分配给多个层，则取消分配适用于每个层；您无法从层的子集取消分配实例。但是，该实例仍然在堆栈中注册，如果您愿意，可以将其分配给另一层。

使用控制台取消分配注册的实例

1. 在导航窗格中，选择实例。
2. 选择要取消分配的实例。
3. 在实例的详细信息页面上，选择取消分配。



要取消分配已注册的实例，请使用 AWS CLI

运行 aws opsworks unassign-instance 命令从正在使用该实例的所有层中取消分配注册的实例。

```
aws opsworks unassign-instance --region region --instance-id instance-id
```

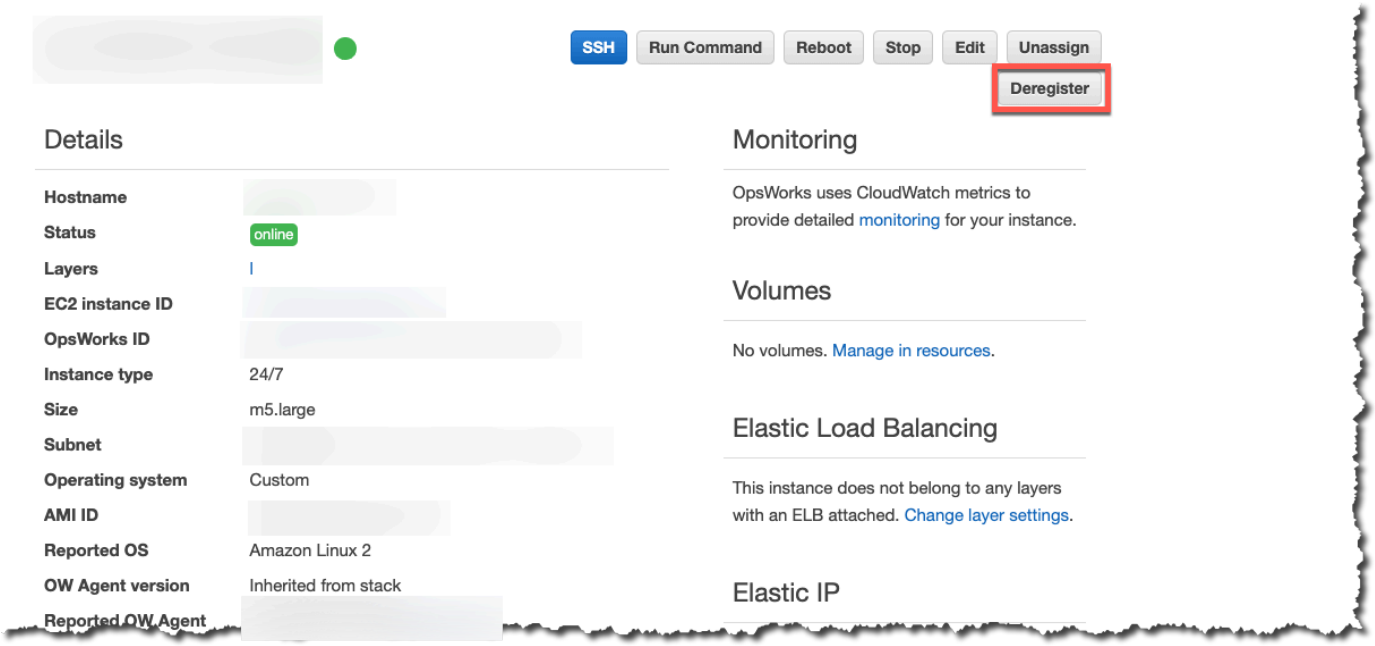
取消注册已注册的实例

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

您可以使用 AWS OpsWorks 控制台、 AWS CLI或 SDK 操作注销实例。

使用控制台取消注册实例

1. 在导航窗格中，选择实例。
2. 选择要取消注册的实例。
3. 在实例的详细信息页面上，选择取消注册。



要取消注册实例，请使用 AWS CLI

运行 aws opsworks deregister-instance 命令将实例从其堆栈中取消注册。

```
aws opsworks deregister-instance --region region --instance-id instance-id
```

当您注销实例时， AWS OpsWorks Stacks 会执行以下操作：

- 从堆栈中删除该实例。
- 从任何已分配层中取消分配实例。
- 关闭并卸载代理。
- 取消注册任何连接的资源 (弹性 IP 地址和 Amazon EBS 卷)。

  此过程包括注册前附加到实例的资源，以及您在实例作为 AWS OpsWorks 堆栈一部分时使用堆栈附加到实例的资源。取消注册后，这些资源不再是堆栈资源的一部分，但它们仍然连接至该实例。
- 对于本地实例，停止计费。
- 移除已 OpsWorks 添加到实例的所有标签。

该实例仍处于运行状态，但它由您直接控制，不再由 AWS OpsWorks Stacks 管理。

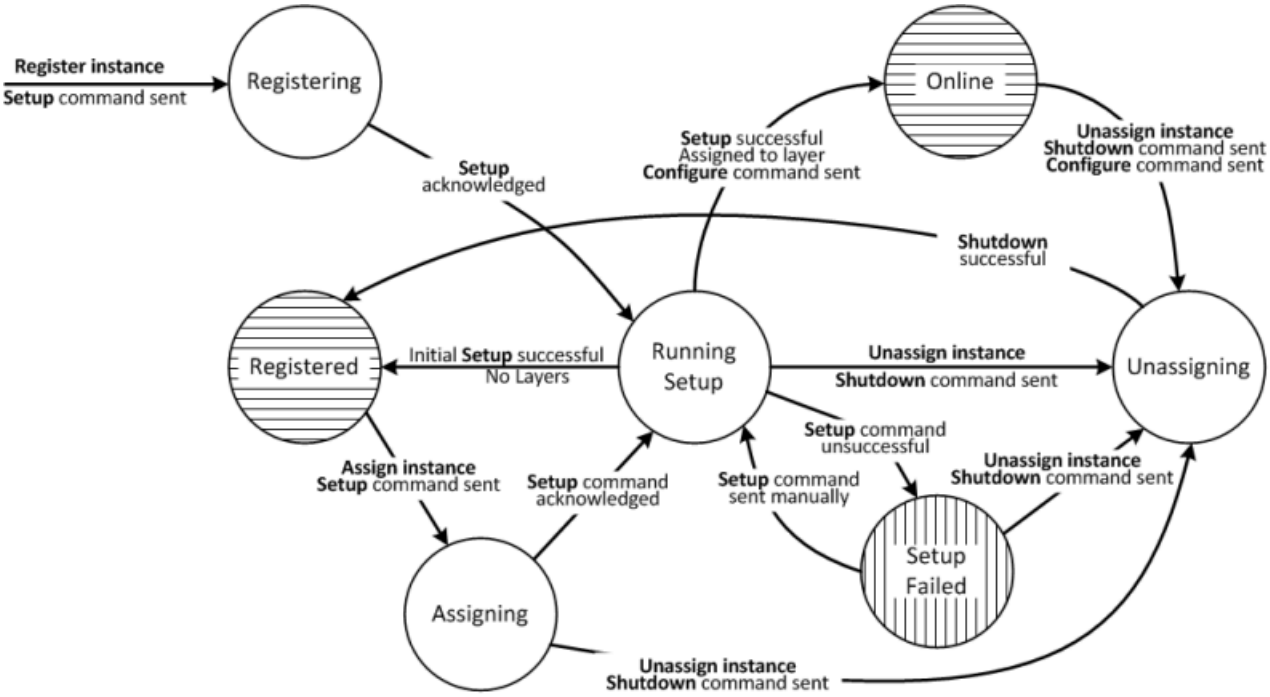
> **Note**
>
> 只有 Linux 堆栈完全支持注册和取消注册计算机或实例。对于 Windows 堆栈，允许取消注册实例，但不会从实例中卸载 OpsWorks 代理。取消注册不会删除所有已更改的文件，并且不会完全恢复到某些文件的备份副本。此列表适用于 Chef 11.10 和 Chef 12 两个堆栈；这里提到了两个版本之间的差异。
>
> - `/etc/hosts` 备份到 `/var/lib/aws/opsworks/local-mode-cache/backup/etc/`，但不还原。
> - `aws` 和 `opsworks` 的条目仍以密码、组和影子文件等形式存在。
> - `/etc/sudoers`包含对 AWS OpsWorks 堆栈目录的引用。
> - 以下文件可以安全保留；但长期来看，应考虑删除 `/var/lib/aws/opsworks`。
>   - `/var/log/aws/opsworks` 保留在 Chef 11.10 堆栈中的实例上。
>   - `/var/lib/aws/opsworks` 保留在 Chef 11.10 和 Chef 12 堆栈上。
>   - `/var/chef` 保留在 Chef 12 堆栈中的实例上。
> - 其他保留的文件：
>   - `/etc/logrotate.d/opsworks-agent`
>   - `/etc/cron.d/opsworks-agent-updater`
>   - `/etc/ld.so.conf.d/opsworks-user-space.conf`
>   - `/etc/motd.opsworks-static`
>   - `/etc/aws/opsworks`
>   - `/etc/sudoers.d/opsworks`
>   - `/etc/sudoers.d/opsworks-agent`

已注册实例的生命周期

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 仅 Linux 堆栈支持此功能。

已注册实例的生命周期从代理安装并开始运行之后开始。此时，它会指示 AWS OpsWorks Stacks 在堆栈中注册实例。以下状态示意图概述了密钥生命周期元素。



每个状态对应于一个实例状态。边缘代表以下 AWS OpsWorks Stacks 命令之一。以下各节讨论了相关详细信息。

- 设置：此命令对应于设置生命周期事件并运行实例的配置配方。
- 配置：此命令对应于配置生命周期事件。

  AWS OpsWorks 当实例进入或离开在线状态时，堆栈会在堆栈中的每个实例上触发此事件。这些实例运行其 Configure 配方，这会进行任何所需的更改以适应新实例。
- 关机：此命令对应于关机生命周期事件，可运行该实例的关机配方。

  这些配方执行某些任务 (如关闭服务)，但不会停止该实例。
- 取消注册：此命令会取消注册实例，且不对应于生命周期事件。

> **Note**
>
> 为简便起见，该示意图不显示 Deregistering 和 Deleted 状态。您可以从示意图中的任何状态取消注册某实例，这会向该实例发送 Deregister 命令并将其移至 Deregistering 状态。
>
> - 如果您注销在线实例， AWS OpsWorks Stacks 会向堆栈中的其余实例发送 Configure 命令，通知它们该实例即将离线。
> - 确认 Deregister 命令后，该实例仍在运行，但它处于 Deleted 状态且不再是该堆栈的一部分。如果您想再次将该实例合并到堆栈中，则必须重新注册。

主题

注册

代理发送注册请求后， AWS OpsWorks Stacks 会向实例发送安装命令，使其处于 Registing 状态，从而启动实例生命周期。在实例确认 Setup 命令后，它将移至 Running Setup 状态。

Running Setup

Running Setup 状态运行该实例的 Setup 配方。Setup 的工作原理取决于前面的状态。

> **Note**
>
> 如果您在实例处于 “运行设置” 状态时取消其分配， AWS OpsWorks Stacks 会发送关机命令，该命令会运行实例的关闭配方，但不会停止实例。该实例移至 取消分配 状态。

主题

- 注册
- 分配
- 设置失败

注册

在注册过程中，安装程序会创建一个 AWS OpsWorks Stacks 实例来表示堆栈中的注册实例，并在该实例上运行一组核心安装配方。

初始设置执行的一个关键更改是覆盖该实例的主机文件。通过注册实例，您将用户管理移交给 AWS OpsWorks Stacks，后者必须拥有自己的主机文件以控制 SSH 登录权限。初始设置还会创建或修改多个文件，并且在 Ubuntu 系统中修改包源并安装一组软件包。有关更多信息，请参阅 初始设置配置更改。

在注册期间，该进程将调用 IAM `AttachUserPolicy`，它是作为先决条件创建的附加到 IAM 用户的权限的一部分。如果 `AttachUserPolicy` 不存在（很可能是因为您运行的是旧版本的 AWS CLI），则该过程将回退到调用 `PutUserPolicy`。

> **Note**
>
> 为了保持一致性， AWS OpsWorks Stacks 会运行所有核心安装配方。然而，某些配方仅在实例已分配给至少一个层后才执行部分或全部任务，因此，它们不一定会影响初始设置。

- 如果设置成功，该实例将移至 已注册 状态。
- 如果设置失败，该实例将移至 设置失败 状态。

分配

该实例至少有一个已分配的层。 AWS OpsWorks Stacks 运行每个图层的设置配方，包括您分配给图层设置事件的任何自定义配方。

- 如果设置成功，该实例将移至 Online 状态，并且 AWS OpsWorks Stacks 将在堆栈中的每个实例上触发 Configure 生命周期事件以向它们通知新实例。
- 如果设置失败，该实例将移至 Setup Failed 状态。

> **Note**
>
> 此设置过程将再次运行核心配方。但是，Chef 配方是幂等的，因此它们不会重复已执行的任何任务。

设置失败

如果处于 分配 状态的实例的设置过程失败，您可以通过使用 Setup 堆栈命令重试，手动重新运行该实例的 Setup 配方。

- 如果设置成功，分配的实例将移至 在线 状态，并且 AWS OpsWorks Stacks 将在堆栈中的每个实例上触发 Configure 生命周期事件以向它们通知新实例。
- 如果设置尝试失败，该实例将返回到 Setup Failed 状态。

已注册

处于 “已注册” 状态的实例是堆栈的一部分，由 AWS OpsWorks 堆栈管理，但未分配给层。它们可以无限期地保持该状态。

如果您将实例分配给一个或多个图层， AWS OpsWorks Stacks 会向该实例发送安装命令，然后它会进入分配状态。

分配

在实例确认 Setup 命令后，它将移至 Running Setup 状态。

如果您在实例处于 “分配” 状态时取消分配该实例， AWS OpsWorks Stacks 会终止安装过程并发送 Shutdown 命令。该实例移至 取消分配 状态。

在线

该实例现在是至少一个层的成员，并且被视为常规 AWS OpsWorks Stacks 实例。它可以无限期地保持该状态。

如果您在实例处于联机状态时取消分配该实例， AWS OpsWorks Stacks 会向该实例发送关机命令，并向堆栈的其余实例发送配置命令。该实例移至 取消分配 状态。

设置失败

Setup 命令失败。

- 您可以通过运行 Setup 堆栈命令进行重试。

  该实例恢复为 Running Setup 状态。
- 如果您取消分配实例， AWS OpsWorks Stacks 会向该实例发送关闭命令。

  该实例移至 取消分配 状态。

取消分配

完成 Shutdown 命令后，该实例不再分配给任何层，并恢复为 已注册 状态。

> **Note**
>
> 如果某实例被分配给多个层，则取消分配将适用于每个层；您无法取消分配已分配层的子集。如果您想要一组不同的已分配层，则取消分配该实例，然后重新分配所需层。

初始设置配置更改

初始设置会在所有已注册实例上创建或修改以下文件和目录。

创建的文件

```
/etc/apt/apt.conf.d/99-no-pipelining
/etc/aws/
/etc/init.d/opsworks-agent
/etc/motd
/etc/motd.opsworks-static
/etc/sudoers.d/opsworks
/etc/sudoers.d/opsworks-agent
/etc/sysctl.d/70-opsworks-defaults.conf
/opt/aws/opsworks/
/usr/sbin/opsworks-agent-cli
/var/lib/aws/
/var/log/aws/
/vol/
```

修改的文件

```
/etc/apt/apt.conf.d/99-no-pipelining
/etc/crontab
```

```
/etc/default/monit
/etc/group
/etc/gshadow
/etc/monit/monitrc
/etc/passwd
/etc/security/limits.conf (removing limits only for EC2 micro instances)
/etc/shadow
/etc/sudoers
```

初始设置还在 Amazon EC2 微型实例上创建一个交换文件。

初始设置对 Ubuntu 系统进行以下更改。

软件包源

初始设置将对软件包源进行如下更改。

- deb http://archive.ubuntu.com/ubuntu/ ${code_name} main universe

  到：deb-src http://archive.ubuntu.com/ubuntu/ ${code_name} main universe
- deb http://archive.ubuntu.com/ubuntu/ ${code_name}-updates main universe

  到：deb-src http://archive.ubuntu.com/ubuntu/ ${code_name}-updates main universe
- deb http://archive.ubuntu.com/ubuntu ${code_name}-security main universe

  到：deb-src http://archive.ubuntu.com/ubuntu ${code_name}-security main universe
- deb http://archive.ubuntu.com/ubuntu/ ${code_name}-updates multiverse

  到：deb-src http://archive.ubuntu.com/ubuntu/ ${code_name}-updates multiverse
- deb http://archive.ubuntu.com/ubuntu ${code_name}-security multiverse

  到：deb-src http://archive.ubuntu.com/ubuntu ${code_name}-security multiverse
- deb http://archive.ubuntu.com/ubuntu/ ${code_name} multiverse

到：`deb-src http://archive.ubuntu.com/ubuntu/ ${code_name} multiverse`

- `deb http://security.ubuntu.com/ubuntu ${code_name}-security multiverse`

  到：`deb-src http://security.ubuntu.com/ubuntu ${code_name}-security multiverse`

软件包

初始设置卸载 `landscape` 并安装以下软件包。

autofs	libicu-dev	libopenssl-ruby
libssl-dev	libxml2-dev	libxslt-dev
libyaml-dev	monit	ntpd
procps	ruby	ruby-dev
rubygems	screen	sqlite
vim	xfs	

编辑实例配置

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

您可编辑实例配置（包括已注册的 Amazon Elastic Compute Cloud (Amazon EC2) 实例），但存在以下限制：

- 该实例必须处于已停止状态。

尽管您无法修改联机实例的属性，但您可通过编辑实例的层来更改实例的配置的某些方面。有关更多信息，请参阅 编辑图 OpsWorks 层的配置。

- 某些设置 (如 Availability Zone 和 Scaling Type) 是在您创建实例时确定的，之后无法修改。
- 某些设置只能为实例存储支持的实例修改，而不能为 Amazon Elastic Block Store 支持的实例修改。

  例如，您可更改实例存储支持的实例的操作系统。Amazon EBS-backed 支持的实例必须使用您创建实例时指定的操作系统。有关实例存储的更多信息，请参阅存储。
- 默认情况下，实例将继承堆栈的代理版本设置。

  您可以使用OpsWorks 代理版本来覆盖堆栈的代理版本设置，并为实例指定特定的代理版本。如果您指定实例的代理版本，则即使 AWS OpsWorks 堆栈的代理版本设置为自动更新，当有新版本可用时，Stacks 也不会自动更新代理。您必须通过编辑实例配置来手动更新实例的代理版本。 AWS OpsWorks 然后，Stacks 会在实例上安装指定的代理版本。

> **Note**
>
> 您无法编辑在本地实例上注册的配置。

编辑实例的配置

1. 停止实例 (如果尚未停止)。
2. 在 Instances 页面上，单击实例名称以显示 Details 页面。
3. 单击 Edit 以显示编辑页面。
4. 视情况编辑实例的配置。

有关 Host name、Size、SSH key 和 Operating system 设置的说明，请参阅将实例添加到层。Layers 设置可让您添加或删除层。实例的当前层显示在层列表后面。

- 要添加另一个层，请从列表中选择它。
- 要将实例从它的其中一个层中删除，请单击相应层旁边的 x。

  实例必须是至少一个层的成员，因此您无法删除最后一个层。

当您重启实例时， AWS OpsWorks Stacks 会使用更新的配置启动一个新的 Amazon EC2 实例。

删除 AWS OpsWorks 堆栈实例

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

您可以使用 AWS OpsWorks 堆栈来停止实例，包括注册的 Amazon EC2 实例。这样会停止 EC2 实例，但该实例仍留在堆栈中。您可以通过单击实例的操作列中的启动来重新启动它。如果您不再需要某个实例并希望将其从堆栈中移除，则可以将其删除，这会从堆栈中移除该实例，并终止关联的 Amazon EC2 实例。删除实例还会删除任何关联的日志或数据，以及该实例上的任何 Amazon Elastic Block Store (EBS) 卷。

> **Important**
>
> 本主题仅适用于由 AWS OpsWorks 堆栈管理的 Amazon EC2 实例。有关如何删除由 Amazon EC2 控制台或 API 管理的实例的更多信息，请参阅终止您的实例。

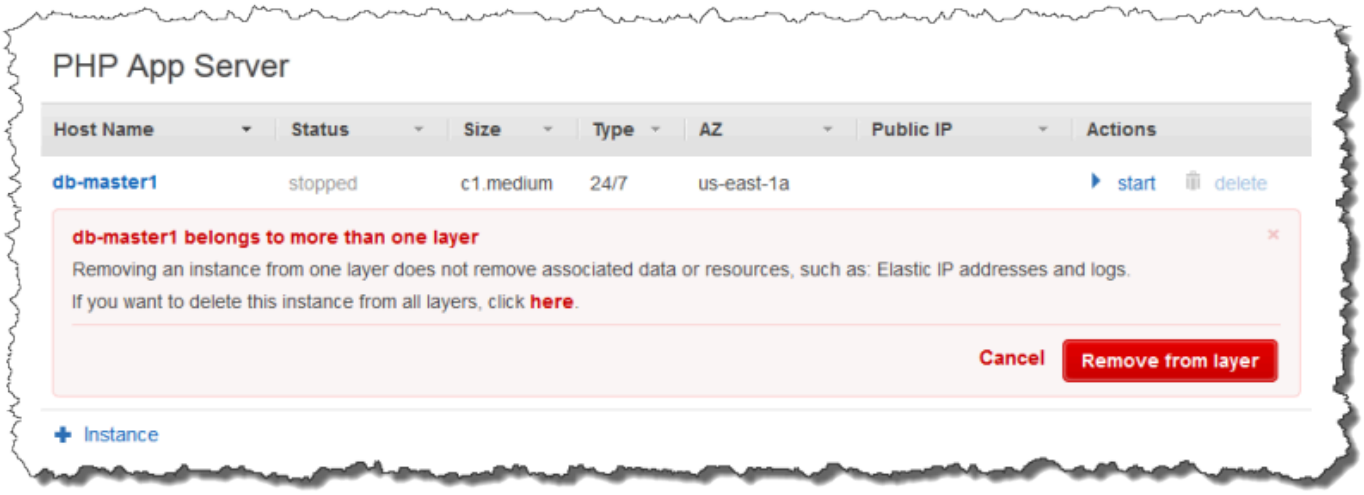
> **Note**
>
> 您不能使用 AWS OpsWorks Stacks 删除已注册的本地实例。

如果某个实例属于多个层，您可以从堆栈中删除实例，或仅移除特定的层。也可以通过编辑实例配置从实例中移除层，如编辑实例配置中所述。

> **Important**
>
> 您只能使用 AWS OpsWorks 堆栈控制台或 API 来删除 AWS OpsWorks Stacks 实例。特别是，您不应使用 Amazon EC2 控制台或 API 删除 AWS OpsWorks 堆栈实例，因为 Amazon EC2 操作不会自动与 AWS OpsWorks 堆栈同步。例如，如果启用了自动修复，并且通过使用 Amazon EC2 控制台终止实例，则 AWS OpsWorks Stacks 会将已终止的实例视为已失败实例，并启动另一个 Amazon EC2 实例来替换它。有关更多信息，请参阅 使用自动修复。

删除实例

1. 在 Instances 页面上，查找相应层下的实例。如果该实例正在运行，请单击 Actions 列中的 stop。
2. 在状态变为 stopped 后，单击 delete。如果实例是多个图层的成员，则层 AWS OpsWorks 堆栈将显示以下部分。



   - 要仅从选定层中删除实例，请单击 Remove from layer。

     该实例仍然是其他层的成员，可以重新启动。
   - 要从其所有层中删除实例 (这会将其从堆栈中移除)，请单击 here。
3. 如果您选择从堆栈中完全移除某个实例，或者该实例仅属于一个层， AWS OpsWorks Stacks 会提示您确认删除。

   选择 Delete (删除) 以确认。除了从堆栈中删除实例外，此操作还会删除所有关联的日志或数据；而且还会删除附加到该实例的根卷。要删除所有实例卷，请选择 Delete instance's EBS volumes (snapshots will not be deleted) (删除实例的 EBS 卷（不会删除快照）)，然后选择 Delete (删除)。

使用 SSH 登录 Linux 实例

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

您可以使用内置 MindTerm客户端或第三方客户端（例如 PuTTY）通过 SSH 登录您的在线 Linux 实例。SSH 通常根据 RSA 密钥对来进行身份验证。您在实例上安装公钥并向 SSH 客户端提供相应的私钥。 AWS OpsWorks Stacks 会为您在堆栈的实例上安装公钥，如下所示。

- Amazon Elastic Compute Cloud (Amazon EC2) 密钥对：如果堆栈的区域拥有一个或多个 Amazon EC2 密钥对，则您可以指定堆栈的默认 SSH 密钥对。

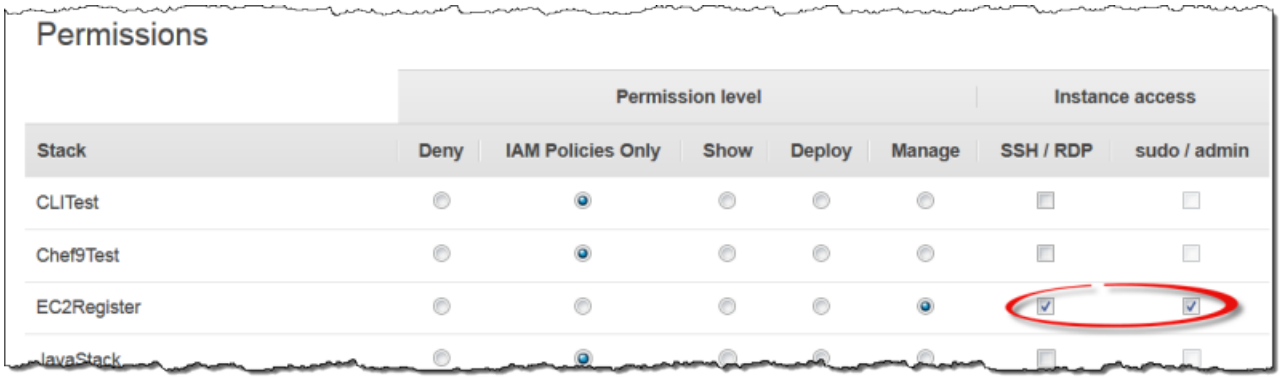
  当您创建实例时，您可以选择性地覆盖默认密钥对并指定其他密钥对。无论哪种情况， AWS OpsWorks Stacks 都会在实例上安装指定密钥对的公钥。有关如何创建 Amazon EC2 密钥对的更多信息，请参阅 Amazon EC2 密钥对。
- 个人密钥对：每个用户都可以在 AWS OpsWorks Stack 注册个人密钥对。

  用户或管理员向 AWS OpsWorks Stacks 注册公钥，用户将私钥存储在本地。为堆栈设置权限时，管理员可指定哪些用户应当拥有堆栈实例的 SSH 访问权限。 AWS OpsWorks Stacks 会自动在堆栈的实例上为每个授权用户创建一个系统用户，并安装用户的个人公钥。

用户必须获得 SSH 授权才能使用 MindTerm SSH 客户端或使用其个人密钥 pair 登录堆栈的实例。

为用户授予 SSH 访问权

1. 在 AWS OpsWorks 堆栈导航窗格中，单击 “权限”。
2. 为所需的 IAM 用户用户选择 SSH/RDP，以授予必要的权限。如果您希望允许用户使用 sudo 来提升权限 (例如，为运行代理 CLI 命令)，请一并选择 sudo/admin。



有关如何使用 AWS OpsWorks 堆栈管理 SSH 访问的更多信息，请参阅管理 SSH 访问。

主题

- 使用内置 MindTerm SSH 客户端
- 使用第三方 SSH 客户端

使用内置 MindTerm SSH 客户端

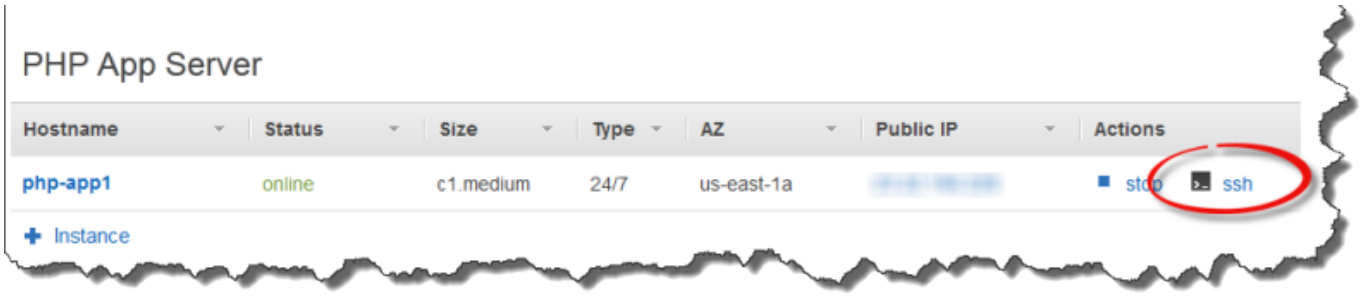
> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

登录 Linux 实例的最简单方法是使用内置的 MindTerm SSH 客户端。每个在线实例都包含一个 ssh 操作，您可以使用该操作启动 MindTerm 客户端。

> **Note**
>
> 您必须在浏览器中启用 Java 才能使用 MindTerm 客户端。

使用 MindTerm 客户端登录

1. 如果您尚未执行此操作，请向将连接到实例的 IAM 用户授予 SSH 访问权限，如上一部分中所述。
2. 以用户身份登录。
3. 在 Instances 页面上，选择相应实例的 Actions 列中的 ssh。



4. 对于私有密钥，提供指向用户的个人私有密钥或 Amazon EC2 私有密钥的路径，具体取决于您在实例上安装了哪些公有密钥。
5. 选择 Launch Mindterm 并使用终端窗口对实例运行命令。

使用第三方 SSH 客户端

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

您还可以使用第三方 SSH 客户端 (如 PuTTY) 来连接到 Linux 实例。

使用第三方 SSH 客户端

1. 如前所述，确保 AWS OpsWorks Stacks 已在实例上安装了 Amazon EC2 公钥或 IAM 用户的个人公钥。
2. 从实例的详细信息页面上获得实例的公有 DNS 名称或公有 IP 地址。
3. 为客户端提供实例的主机名，这取决于操作系统，如下所述：

   - Amazon Linux 和 Red Hat Enterprise Linux (RHEL)– `ec2-user@`*`DNSName/Address`*`.`
   - Ubuntu – `ubuntu@`*`DNSName/Address`*。

   将上一步中的 *`DNSName/Address`* 更换为 DNS 名称或 IP 地址。
4. 为客户端提供与安装的公有密钥对应的私有密钥。您可以使用 Amazon EC2 私有密钥或 IAM 用户的个人私有密钥，具体取决于安装在实例上的公有密钥。

使用 RDP 登录 Windows 实例

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

您可以使用 Windows 远程桌面协议 (RDP) 登录到联机 Windows 实例，如下所示：

- 该实例必须具有安全组以及允许 RDP 访问的入站规则。

有关使用安全组的更多信息，请参阅使用安全组。

- 普通用户 — AWS OpsWorks Stacks 为授权的普通用户提供 RDP 密码，该密码在有限的时间段内有效，范围从 30 分钟到 12 小时不等。

  除了获得授权外，用户还必须至少具有 Sho w 权限级别，或者其附加 AWS Identity and Access Management (IAM) 策略必须允许该`opsworks:GrantAccess`操作。

- 管理员：您可以使用管理员密码登录，登录时长不受限制。

  如后文所述，如果您指定了实例的 Amazon Elastic Compute Cloud (Amazon EC2) 密钥对，就可以用它来检索管理员密码。

> **Note**
>
> 本主题介绍如何从 Windows 工作站使用 Windows 远程桌面连接客户端登录。您还可以为 Linux 或 OS X 使用可用的 RDP 客户端之一，但过程可能会有些不同。有关与 Microsoft Windows Server 2012 R2 兼容的 RDP 客户端的更多信息，请参阅 Microsoft 远程桌面客户端。

主题

提供允许 RDP 访问的安全组

实例的安全组入站规则必须允许 RDP 连接，然后您才能使用 RDP 登录 Windows 实例。在区域中创建第一个堆栈时， AWS OpsWorks Stacks 创建一组安全组。它们包括一个名为类似的名字`AWS-OpsWorks-RDP-Server`， AWS OpsWorks 堆栈将其连接到所有 Windows 实例以允许 RDP 访问。但是，默认情况下，此安全组没有任何规则，因此您必须添加入站规则以允许 RDP 访问您的实例。

允许 RDP 访问

1. 打开 Amazon S3 控制台，将其设置为堆栈的区域，然后从导航窗格中选择安全组。
2. 选择 AWS-OpsWorks RDP-Server，选择 “入站” 选项卡，然后选择 “编辑”。

3. 选择 Add Rule，然后指定以下设置：

   - 类型：RDP
   - 源 -允许的源 IP 地址。

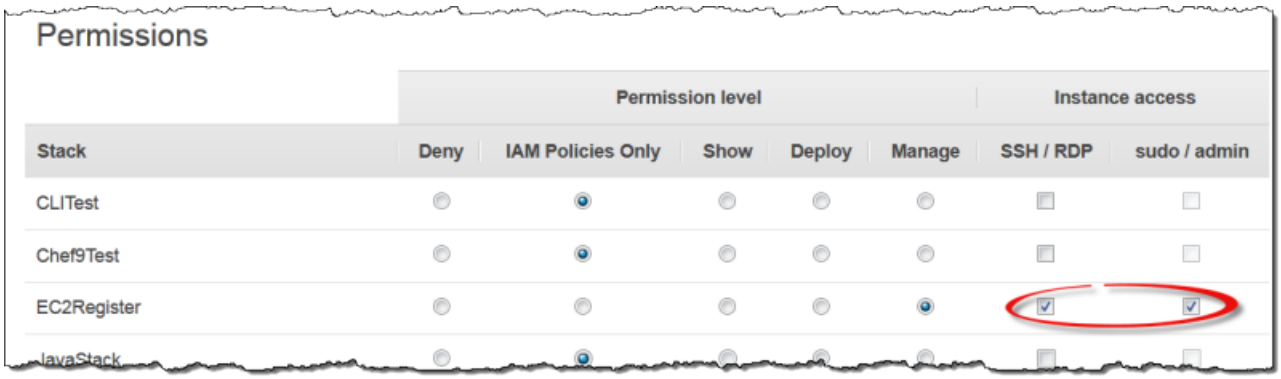
     通常您会允许来自您的 IP 地址或指定 IP 地址范围（一般是公司的 IP 地址范围）的入站 RDP 请求。

作为普通用户登录

授权用户可以使用 AWS OpsWorks Stacks 提供的临时密码登录到实例。

为用户授予 RDP 访问权；

1. 在 AWS OpsWorks 堆栈导航窗格中，单击 “权限”。
2. 选中所需用户的 SSH/RDP 复选框以授予必要的权限。如果您希望用户具有管理员权限，则还应选择 sudo/admin。



授权用户可以登录到堆栈的任意联机实例，如下所示。

作为普通 IAM 用户登录

1. 作为 IAM 用户登录。
2. 在 Instances 页面上，选择相应实例的 Actions 列中的 rdp。
3. 指定会话长度，范围可从 30 分钟到 12 小时，然后选择 Generate Password。密码仅在指定的会话持续时间内有效。
4. 记录 public DNS name、username 和 password 值，然后选择 Acknowledge and close。
5. 打开 Windows 远程桌面连接客户端，选择 Show Options，然后使用在步骤 4 中记录的信息提供以下信息：

- 计算机：实例的公有 DNS 名称。

  如果愿意，您也可以使用公有 IP 地址。选择 Instances，并复制实例的 Public IP 列中的地址。
- 用户名：用户名。

6. 在客户端提示输入凭证时，输入您在步骤 4 中保存的密码。

> **Note**
>
> AWS OpsWorks Stacks 仅为在线实例生成用户密码。例如，如果您启动一个实例，您的自定义 Setup 配方之一失败，则实例最终将处于 `setup_failed` 状态。尽管就 AWS OpsWorks Stacks 而言，该实例未处于联机状态，但 EC2 实例仍在运行，登录以解决问题通常很有用。AWS OpsWorks 在这种情况下，堆栈不会为您生成密码，但是如果您已为实例分配了 SSH 密钥对，则可以使用 EC2 控制台或 CLI 来检索实例的管理员密码并以管理员身份登录。有关更多信息，请参阅以下 章节。

作为管理员登录

您可以使用相应的密码，作为管理员登录实例。如果您将 EC2 密钥对分配到某个实例，则 Amazon EC2 在实例启动时使用它来自动创建并加密管理员密码。然后，您可以通过 EC2 控制台、API 或 CLI 使用密钥对的私有密钥检索和解密密码。

> **Note**
>
> 您不能使用个人 SSH 密钥对来检索管理员密码；必须使用 EC2 密钥对。

以下介绍了如何使用 EC2 控制台来检索管理员密码并登录实例。如果您偏好命令行工具，则还可以使用 AWS CLI `get-password-data` 命令来检索密码。

作为管理员登录

1. 确保您为实例指定了 EC2 密钥对。创建堆栈时，您可以为堆栈的所有实例指定默认密钥对，或者可以在创建实例时为特定实例指定密钥对。
2. 打开 EC2 控制台，将其设置为堆栈的区域，然后从导航窗格中选择 Instances (实例)。
3. 依次选择实例、Connect 和 Get Password。

4. 提供您的工作站上 EC2 密钥对中私有密钥的路径，然后选择 Decrypt Password。复制解密的密码供以后使用。
5. 打开 Windows 远程桌面连接客户端，选择 Show Options，然后提供以下信息：
   - 计算机：实例的公有 DNS 名称或公有 IP 地址，您可以从实例的详细信息页面获取这些信息。
   - 用户名：`Administrator`。
6. 客户端提示输入凭证时，提供从步骤 4 中获得的已解密密码。

应用程序

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

AWS OpsWorks Stacks 应用程序表示您要在应用程序服务器上运行的代码。该代码本身位于存储库 (如 Amazon S3 存档) 中；该应用程序包含将该代码部署到相应的应用程序服务器实例所需的信息。

部署应用程序时， AWS OpsWorks Stacks 会触发 Deploy 事件，该事件会运行每个层的 Deploy 配方。 AWS OpsWorks Stacks 还会安装堆栈配置和部署属性，其中包含部署应用程序所需的所有信息，例如应用程序的存储库和数据库连接数据。

您必须实施自定义配方，这些配方从堆栈配置和部署属性中检索该应用程序的部署数据并处理部署任务。

主题

- 使用自定义域
- 使用 SSL

添加应用程序

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

向您的应用程序服务器部署应用程序的第一步是将应用程序添加到堆栈。应用表示应用程序，包含各种元数据，例如应用程序的名称和类型以及向服务器实例部署应用程序所需的信息，如存储库 URL。您必须拥有管理权限才能将应用程序添加到堆栈。有关更多信息，请参阅 管理用户权限。

> **Note**
>
> 本部分中的过程适用于 Chef 12 和更新的堆栈。有关如何向 Chef 11 堆栈的层中添加应用程序的信息，请参阅步骤 2.4：创建和部署应用程序 - Chef 11。

将应用程序添加到堆栈

1. 将代码置于您的首选存储库中：Amazon S3 存档、Git 存储库、Subversion 存储库或 HTTP 存档。有关更多信息，请参阅 应用程序源。
2. 在导航窗格中，单击 Apps。在 Apps 页面上，单击 Add an app 以添加您的第一个应用程序。对于后续应用程序，单击 +App。
3. 使用 App New 页面来配置应用程序，如以下部分中所述。

配置应用程序

Add App 页面包含以下部分：Settings、Application source、Data Sources、Environment Variables、Add Domains 和 SSL Settings。

主题

- 设置

- 应用程序源
- 数据源
- 环境变量
- 域和 SSL 设置

设置

名称

应用程序名称，用于在界面中表示应用程序。 AWS OpsWorks Stacks 还使用此名称为内部使用的应用程序生成短名称，并在堆栈配置和部署属性中标识该应用程序。在将应用程序添加到堆栈后，您就可以查看短名称了，方法是在导航窗格中单击 Apps，然后单击该应用程序的名称以打开详细信息页面。

Document root

AWS OpsWorks Stacks 将文档根目录设置分配给应用程序`[:document_root]`属性中的属性。`deploy`默认值为 `null`。您的部署配方可以使用标准 Chef 节点语法从 `deploy` 属性中获取该值，并将指定代码部署到服务器上的合适位置。有关如何部署应用程序的更多信息，请参阅 Deploy 配方

应用程序源

您可以从以下类型的存储库部署应用程序：Git、Amazon S3 包、HTTP 包及其他。所有存储库类型都要求您指定存储库类型和存储库 URL。各个存储库类型有自己的要求，说明如下。

> **Note**
>
> AWS OpsWorks Stacks 会自动将应用程序从标准存储库部署到内置服务器层。如果您使用其他存储库类型（这是 Windows 堆栈的唯一选项），AWS OpsWorks Stacks 会将存储库信息放在应用程序的`deploy`属性中，但您必须实现自定义配方才能处理部署任务。

主题

- HTTP 存档
- Amazon S3 归档
- Git 存储库

- 其他存储库

HTTP 存档

要将可公开访问的 HTTP 服务器用作存储库，请执行以下步骤：

1. 创建包含应用程序代码和任何关联文件的文件夹的压缩存档：zip、gzip、bzip2、Java WAR 或 tarball。

   > **Note**
   >
   > AWS OpsWorks 堆栈不支持未压缩的压缩包。

2. 将该存档文件上传到服务器。
3. 要在控制台中指定存储库，请选择 HTTP 存档作为存储库类型，然后输入 URL。

   如果存档受密码保护，请在应用程序源下指定登录凭证。

Amazon S3 归档

要使用 Amazon Simple Storage Service 存储桶作为存储库：

1. 创建公有或私有 Amazon S3 存储桶。有关更多信息，请参阅 Amazon S3 文档。
2. 要 AWS OpsWorks 让 Stacks 访问私有存储桶，您必须是至少拥有 Amazon S3 存储桶只读权限的用户，并且需要访问密钥 ID 和私有访问密钥。有关更多信息，请参阅 AWS Identity and Access Management 文档。
3. 将代码和任何关联的文件置于一个文件夹中，并将该文件夹保存在压缩存档中 - zip、gzip、bzip2、Java WAR 或 tarball。

   > **Note**
   >
   > AWS OpsWorks 堆栈不支持未压缩的压缩包。

4. 将该存档文件上传到 Amazon S3 存储桶并记录 URL。
5. 要在 AWS OpsWorks 堆栈控制台中指定存储库，请将存储库类型设置为 S3 存档，然后输入档案的 URL。对于私有存档，您还必须提供其策略可授予访问该存储桶权限的 AWS 访问密钥 ID 和秘密访问密钥。对于公有存档，这些设置保留为空。

Git 存储库

Git 存储库提供源代码控制和版本控制。 AWS OpsWorks Stacks 支持公共托管的存储库站点，例如GitHub或 Bitbucke t 以及私有托管的 Git 服务器。对于应用程序和 Git 子模块而言，您在 Application Source 中用于指定存储库 URL 的格式取决于存储库是公有的还是私有的：

公有存储库：使用 HTTPS 或 Git 只读协议。例如，Chef 11 Linux 堆栈入门使用可通过以下任一 URL 格式访问的公共 GitHub 存储库：

- Git 只读：**`git://github.com/amazonwebservices/opsworks-demo-php-simple-app.git`**
- HTTPS：**`https://github.com/amazonwebservices/opsworks-demo-php-simple-app.git`**

私有存储库：使用以下示例中所示的 SSH 读/写格式：

- Github 存储库：**`git@github.com:`*`project`*`/`*`repository`*。
- Git 服务器上的存储库：*`user`*`@`*`server`*`:`*`project`*`/`*`repository`*

选择 Source Control 下的 Git 会显示两个额外的可选设置：

存储库 SSH 密钥

您必须指定一个部署 SSH 密钥来访问私有 Git 存储库。此字段需要私有密钥；公有密钥已分配到您的 Git 存储库。对于 Git 子模块，指定的密钥必须有权访问这些子模块。有关更多信息，请参阅 使用 Git 存储库 SSH 密钥。

> **Important**
>
> 部署 SSH 密钥不需要密码； AWS OpsWorks Stacks 无法通过密码。

分支/修订

如果存储库有多个分支， AWS OpsWorks Stacks 会默认下载主分支。要指定特定分支，请输入分支名称、SHA1 哈希或标签名称。要指定特定提交，请输入完整的 40 位十六进制提交标识符。

其他存储库

如果标准存储库不符合您的要求，您也可以使用其他存储库，例如 Bazaar。但是， AWS OpsWorks Stacks 不会自动部署来自此类存储库的应用程序。您必须实施自定义配方来处理部署流程并将其分配到合适层的部署事件。有关如何实现“部署”配方的示例，请参阅 Deploy 配方。

数据源

本部分将数据库附加到应用程序。您有以下选项：

- RDS：附加堆栈的 Amazon RDS 服务层之一。
- 无：不附加数据库服务器。

如果您选择 RDS，则必须指定以下内容。

数据库实例

该列表包括每个 Amazon RDS 服务层。您还可以选择以下任一项：

(必填项) 指定要将哪个数据库服务器附加到应用程序。列表的内容取决于数据源。

- RDS：堆栈的 Amazon RDS 服务层列表。

数据库名称

(可选) 指定数据库名称。

- Amazon RDS 层：输入您为 Amazon RDS 实例指定的数据库名称。

  您可以从 Amazon RDS 控制台获取数据库名称。

当您部署带有附加数据库的应用程序时， AWS OpsWorks Stacks 会将数据库实例的连接添加到应用程序的`deploy`属性中。

您可以编写自定义配方，以检索 `deploy` 属性中的信息并将其保存在该应用程序可访问的文件中。这是向“Other”应用程序类型提供数据库连接信息的唯一选项。

有关如何处理数据库连接的更多信息，请参阅 连接到数据库。

要将数据库服务器与应用程序分离，编辑应用程序的配置以指定一个不同数据库服务器或无服务器。

环境变量

您可以为每个应用程序指定一组环境变量，这些变量是特定于应用程序的。例如，如果您有两个应用程序，那么您为第一个应用程序定义的环境变量不适用于第二个应用程序，反之亦然。您也可以为多个应用程序定义相同的环境变量，并为每个应用程序分配一个不同值。

> **Note**
>
> 环境变量没有特定的数量限制。但是，关联的数据结构（其中包括这些变量名称、值和受保护标记值）的大小不能超过 20 KB。此限制应适用于大部分使用案例（如果不是所有使用案例的话）。超过此大小将导致服务错误 (控制台) 或异常 (API)，并显示消息“环境：过大 (最大值为 20 KB)。”

AWS OpsWorks Stacks 将变量作为属性存储在应用程序的deploy属性中。您可以通过使用标准 Chef 节点语法让自定义配方检索这些值。有关如何访问应用程序的环境变量的示例，请参阅 使用环境变量。

键

变量名称。可以包含最多 64 个大小写字母、数字和下划线 (_)，但必须以字母或下划线开始。

值

变量的值。最多可包含 256 个字符，并且必须都是可打印字符。

受保护的值

该值是否受保护。此设置可让您隐藏密码等敏感信息。如果您为变量设置 Protected value，在您创建应用程序后：

- 该应用程序的详细信息页面将仅显示变量名称，而不显示值。
- 如果您有权编辑应用程序，则可以单击 Update value 来指定一个新值，但不能查看或编辑旧值。

> **Note**
>
> Chef 部署日志有时包含环境变量。这意味着受保护变量可能显示在控制台中。为了防止受保护的变量显示在控制台中，我们建议您使用 Amazon S3 存储桶来存储不希望显示在控制台中的受保护变量。本指南中的使用 Amazon S3 存储桶提供了有关如何使用 S3 存储桶以达到此目的的示例。

域和 SSL 设置

对于其他应用程序类型， AWS OpsWorks Stacks 会将设置添加到应用程序的deploy属性中。您的配方可以从这些属性中检索数据并根据需要配置服务器。

域设置

本部分有一个用于指定域的可选 Add Domains 字段。有关更多信息，请参阅 使用自定义域。

SSL 设置

本部分有一个 SSL Support 开关，您可以用它来启用或禁用 SSL。如果您单击 Yes，将需要提供 SSL 证书信息。有关更多信息，请参阅 使用 SSL。

部署应用程序

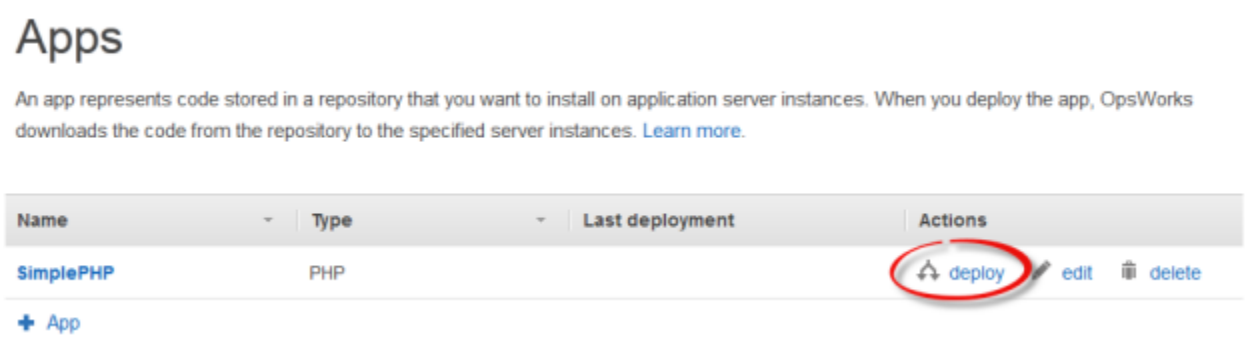
> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

部署的主要目的是将应用程序代码和相关文件部署到应用程序服务器实例。部署操作由每个实例的“部署”配方处理，而该配方由相应实例的层确定。

启动实例时，在安装配方完成后， AWS OpsWorks Stacks 会自动运行该实例的 Deploy 配方。但是，当您添加或修改应用程序时，您必须将其手动部署到任何联机实例。您必须拥有管理或部署权限才能部署应用程序。有关更多信息，请参阅 管理用户权限。

部署应用程序

1. 在 Apps 页面上，单击应用程序的 deploy 操作。



> **Note**
>
> 您也可单击导航窗格中的 Deployments 来部署应用程序。在 Deployments & Commands 页面上单击 Deploy an app，执行此操作时，您还可选择要部署的应用程序。

2. 指定以下内容：

   - (必需) 将 Command: 设置为 deploy (如果尚未选择此项)。
   - (可选) 包含注释。

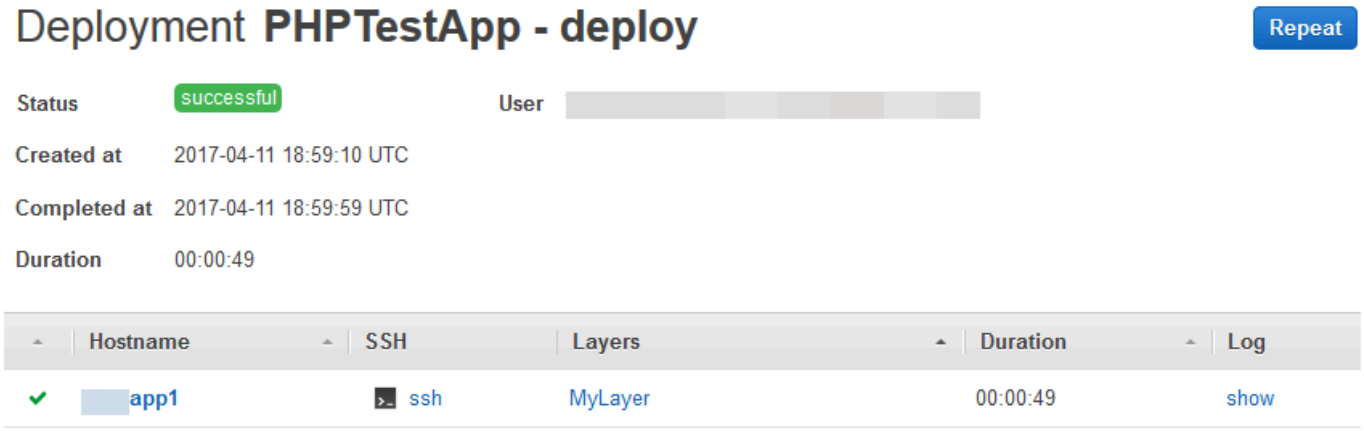
3. 单击 “高级 >>” 以指定自定义 JSON。 AWS OpsWorks Stacks 向节点对象添加了一组堆栈配置和部署属性。deploy 属性包含部署详细信息，可供“部署”配方用来处理安装和配置。在 Linux 堆栈上，您可以使用自定义 JSON 字段来覆盖默认 AWS OpsWorks 堆栈设置或将自定义设置传递给您的自定义配方。有关如何使用自定义 JSON 的更多信息，请参阅使用自定义 JSON。

   > **Note**
   >
   > 如果您在此处指定了自定义 JSON，则它只会添加到此部署的堆栈配置和部署属性。如果您要永久添加自定义 JSON，则必须将它添加到堆栈。自定义 JSON 限制为 120 KB。如果您需要更多容量，则建议您将部分数据存储在 Amazon S3 上。您的自定义配方随后可使用 AWS CLI 或适用于 Ruby 的 AWS SDK将数据从存储桶下载到您的实例。有关示例，请参阅使用适用于 Ruby 的 SDK。

4. 在 Instances 下，单击 Advanced >> 并指定要运行部署命令的实例。

   部署命令将触发一个部署事件，该事件将在选定实例上运行部署配方。关联应用程序服务器的部署配方会将代码和相关文件从存储库下载并安装到实例上，因此您通常会选择所有关联应用程序服务器实例。但是，其他实例类型可能需要进行一些配置更改才能适应新应用程序，因此在此类实例上运行部署配方通常很有用。此类配方会按需更新配置，但不会安装应用程序的文件。有关配方的更多信息，请参阅说明书和诀窍。

5. 单击 Deploy 在指定实例上运行部署配方，这将显示“Deployment”页面。该过程完成后， AWS OpsWorks Stacks 会用绿色勾号标记应用程序，表示部署成功。如果部署失败， AWS OpsWorks Stacks 会用红色 X 标记应用程序。在这种情况下，您可以转到 “部署” 页面并查看部署日志以获取更多信息。



> **Note**
>
> 当您为 JSP 应用程序部署更新时，Tomcat 可能无法识别该更新，而是继续运行现有的应用程序版本。在某些情况下，比如当您将您的应用程序作为仅包含一个 JSP 页面的 .zip 文件时，可能会发生这种情况。要确保 Tomcat 将运行最近部署的版本，项目的根目录应包含放置了 `web.xml` 文件的 WEB-INF 目录。`web.xml` 文件可包含各种内容，但以下内容足以确保 Tomcat 识别更新并运行当前部署的应用程序版本。您无需为每个更新更改版本。Tomcat 将识别更新，即使版本未发生更改。
>
> ```
> <context-param>
>   <param-name>appVersion</param-name>
>   <param-value>0.1</param-value>
> </context-param>
> ```

其他部署命令

Deploy app 页包含用于管理您的应用程序和关联服务器的另外几个命令。在以下命令中，只有 Undeploy 适用于 Chef 12 堆栈上的应用程序。

Undeploy

触发取消部署生命周期事件，这将运行“取消部署”配方以从指定实例中删除应用程序的所有版本。

回滚

恢复以前部署的应用程序版本。例如，如果您部署了应用程序三次，然后运行 Rollback，则服务器将从第二次部署为应用程序提供服务。如果您再次运行 Rollback，服务器将从第一次部署为应用程序提供服务。默认情况下， AWS OpsWorks Stacks 存储五个最新的部署，这允许您最多回滚四个版本。如果您超出了已存储版本的数量，此命令将失败并保持最旧的版本不变。此命令在 Chef 12 堆栈中不可用。

启动 Web 服务器

运行在指定实例上启动应用程序服务器的配方。此命令在 Chef 12 堆栈中不可用。

停止 Web 服务器

运行在指定实例上停止应用程序服务器的配方。此命令在 Chef 12 堆栈中不可用。

重新启动 Web 服务器

运行在指定实例上重新启动应用程序服务器的配方。此命令在 Chef 12 堆栈中不可用。

Important

Start Web Server、Stop Web Server、Restart Web Server 和 Rollback 本质上是 Execute Recipes 堆栈命令的自定义版本。它们运行一组在指定实例上执行任务的配方。

- 这些命令不会触发生命周期事件，因此您无法挂钩它们以运行自定义代码。
- 这些命令仅适用于内置 应用程序服务器层。

  具体而言，这些命令对自定义层无效，即使它们支持应用程序服务器也是如此。要在自定义层上启动、停止或重新启动服务器，您必须实施自定义配方来执行这些任务并使用“执行配方”堆栈命令来运行这些配方。有关如何实施和安装自定义配方的更多信息，请参阅说明书和诀窍。

编辑应用程序

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

您可通过编辑应用程序来修改应用程序的配置。例如，如果您准备部署新版本，则可以编辑应用程序的 AWS OpsWorks Stacks 设置以使用新的存储库分支。您必须拥有管理或部署权限才能编辑应用程序的配置。有关更多信息，请参阅 管理用户权限。

编辑应用程序

1. 在 Apps 页面上，单击应用程序名称以打开其详细信息页面。
2. 单击 Edit 以更改应用程序的配置。
   - 如果您修改应用程序的名称， AWS OpsWorks Stacks 将使用新名称在控制台中标识该应用程序。

     更改该名称不会更改关联的短名称。短名称是在将应用程序添加到堆栈时设置的，之后无法修改。
   - 如果您已指定一个受保护的环境变量，则无法查看或编辑该值。不过，您可通过单击 Update value 来指定新值。
3. 单击 Save 保存新配置，然后单击 Deploy App 部署应用程序。

编辑应用程序会更改 AWS OpsWorks 堆栈的设置，但不会影响堆栈的实例。当您首次部署应用程序时，“部署”配方会将代码和相关文件下载到应用程序服务器实例，该实例随后将运行本地副本。如果您修改存储库中的应用程序或更改任何其他设置，则必须部署该应用程序才能在您的应用程序服务器实例上安装更新，如下所示。 AWS OpsWorks Stacks 在新实例启动时会自动将当前应用程序版本部署到新实例。然而，对于现有实例，情况则有所不同：

- AWS OpsWorks Stacks 在新实例启动时会自动将当前应用程序版本部署到新实例。
- AWS OpsWorks 当离线实例重启时，Stacks 会自动将最新的应用程序版本部署到离线实例，包括基于负载和基于时间的实例。
- 您必须手动将更新后的应用程序部署到联机实例。

有关如何部署应用程序的更多信息，请参阅部署应用程序

将应用程序连接到数据库服务器

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

您可在创建应用程序时将 Amazon RDS 数据库服务器与应用程序关联，也可在之后通过编辑应用程序实现此目的。然后，您的应用程序可以使用数据库连接信息 (用户名、密码...) 连接到数据库服务器。部署应用程序时， AWS OpsWorks Stacks 通过两种方式向应用程序提供此信息：

- 对于 Linux 堆栈， AWS OpsWorks Stacks 会在每个内置应用程序服务器实例上创建一个文件，在其中包含应用程序可用于连接到数据库服务器的连接数据。
- AWS OpsWorks 堆栈在每个实例上安装的堆栈配置和部署属性中包含连接信息。

  您可以实施一个自定义配方来从这些属性中提取连接信息，并将这些信息按照您首选的格式放置在一个文件中。有关更多信息，请参阅 传递数据到应用程序。

> **Important**
>
> 对于 Linux 堆栈，如果您要将 Amazon RDS 服务层与您的应用程序相关联，则必须将适当的驱动程序包添加到关联的应用程序服务器层，如下所示：
>
> 1. 在导航窗格中单击 Layers，然后打开应用程序服务器的 Recipes 选项卡。
> 2. 单击 Edit 并将适当的驱动程序包添加到 OS Packages。例如，如果层包含 Amazon Linux 实例，则您应指定 `mysql`；如果层包含 Ubuntu 实例，则应指定 `mysql-client`。
> 3. 保存更改并重新部署应用程序。

使用自定义配方

您可以实施一个自定义配方，从应用程序的 `deploy` 属性中提取连接数据并采用应用程序可读取的格式 (如 YAML 文件) 保存这些数据。

您可在创建应用程序时将数据库服务器挂载到一个应用程序，或者稍后通过编辑应用程序来执行此操作。部署应用程序时，AWS OpsWorks Stacks 会在每个实例上安装包含数据库连接信息的堆栈配置和部署属性。然后，您的应用程序可以检索相应的属性。详细信息取决于您使用的是 Linux 堆栈还是 Windows 堆栈。

连接到 Linux 堆栈的数据库服务器

对于 Linux 堆栈，堆栈配置和部署属性`deploy`的命名空间为每个已部署的应用程序包含一个对应的属性并使用应用程序的短名称命名。当您将数据库服务器连接到应用程序时，AWS OpsWorks Stacks 会使用连接信息填充应用程序的`[:database]`属性，然后将其安装在堆栈的实例上，供后续每次部署使用。这些属性值由用户提供或由 AWS OpsWorks Stacks 生成。

> **Note**
>
> AWS OpsWorks Stacks 允许您将数据库服务器连接到多个应用程序，但每个应用程序只能连接一个数据库服务器。如果您要将一个应用程序连接到多个数据库服务器，请将其中一个服务器挂载到该应用程序，然后使用应用程序的 `deploy` 属性中的信息连接到该服务器。使用自定义 JSON 将其他数据库服务器的连接信息传递到应用程序。有关更多信息，请参阅 传递数据到应用程序。

应用程序可使用实例的 `deploy` 属性中的连接信息来连接到数据库。但是，应用程序无法直接访问该信息，只有配方才可以访问 `deploy` 属性。您可以通过以下方法来解决这一问题：实施一个自定义配方，让它提取 `deploy` 属性中的连接信息，并将这些将信息放置在应用程序可读取的文件中。

使用环境变量

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 本主题中的建议适用于 Chef 11.10 和早期版本的 Chef。要获取 Chef 12 和更新版本中的环境变量，您必须使用应用程序数据包。有关更多信息，请参阅 AWS OpsWorks 数据包参考和应用程序数据包 (aws_opsworks_ app)。

当您为应用程序指定环境变量时， AWS OpsWorks Stacks 会将变量定义添加到应用程序的deploy属性中。

自定义层可以使用配方来检索变量的值 (通过使用标准节点语句)，并以该层的应用程序可以访问的格式存储该值。

您必须实现一个自定义配方，从实例的 deploy 属性获取环境变量值。然后，该配方可以将数据以应用程序可以访问的格式 (例如 YAML 文件格式) 存储在实例上。一个应用程序的环境变量定义存储在该应用程序的 deploy 中的 environment_variables 属性中。以下示例显示了名为 simplephpapp 的应用程序的这些属性的位置 (使用 JSON 代表属性结构)。

```
{
  ...
  "ssh_users": {
  },
  "deploy": {
    "simplephpapp": {
      "application": "simplephpapp",
      "application_type": "php",
      "environment_variables": {
        "USER_ID": "168424",
        "USER_KEY": "somepassword"
      },
    ...
  }
}
```

通过使用标准节点语法，配方可以获取变量值。以下示例显示了如何从上述 JSON 中获取 USER_ID 值并将其放在 Chef 日志中。

```
Chef::Log.info("USER_ID: #{node[:deploy]['simplephpapp'][:environment_variables]
[:USER_ID]}")
```

有关如何从堆栈配置和部署 JSON 中检索信息并将其存储在实例上的更多详细说明，请参阅传递数据到应用程序。

传递数据到应用程序

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

将键/值对之类的数据传递到服务器上的应用程序通常会很有用。为此，请使用自定义 JSON 添加数据到堆栈。 AWS OpsWorks Stacks 将每个生命周期事件的数据添加到每个实例的节点对象。

不过请注意，虽然配方可以使用 Chef 属性从节点对象获取自定义 JSON 数据，但应用程序不能。将自定义 JSON 数据传输到一个或多个应用程序的方法之一是，实施从 node 对象提取数据的自定义配方并将其写入到应用程序可读取的文件中。本主题中的示例显示了如何将数据写入 YAML 文件，但您可以为其他格式 (例如 JSON 或 XML) 使用相同的基本方法。

要将键/值数据传递到堆栈的实例，请将类似于下文的自定义 JSON 添加到堆栈。有关如何将自定义 JSON 添加到堆栈的更多信息，请参阅使用自定义 JSON。

```
{
  "my_app_data": {
    "app1": {
      "key1": "value1",
      "key2": "value2",
      "key3": "value3"
    },
    "app2": {
      "key1": "value1",
      "key2": "value2",
      "key3": "value3"
    }
  }
}
```

该示例假定您有两个应用程序，其短名称为 app1 和 app2，每个应用程序均有三个数据值。随附的配方假设您使用应用程序的短名称来确定关联的数据，其他名称任意。有关应用程序短名称的更多信息，请参阅设置。

以下示例中的配方显示了如何为每个应用程序从 deploy 属性提取数据并将其放到 .yml 文件中。该配方假定您的自定义 JSON 包含每个应用程序的数据。

```
node[:deploy].each do |app, deploy|
  file File.join(deploy[:deploy_to], 'shared', 'config', 'app_data.yml') do
    content YAML.dump(node[:my_app_data][app].to_hash)
  end
end
```

deploy 属性为每个应用程序包含一个属性，使用该应用程序的短名称命名。每个应用程序属性包含一组属性，表示有关应用程序的各种信息。此示例使用应用程序的部署目录，由 [:deploy][:*app_short_name*][:deploy_to] 属性表示。有关 [:deploy] 的更多信息，请参阅 deploy 属性。

对于 deploy 中的各个应用程序，配方执行以下操作：

1. 在应用程序 [:deploy_to] 目录的 shared/config 子目录中创建名为 app_data.yml 的文件。

   有关 AWS OpsWorks Stacks 如何安装应用程序的更多信息，请参阅Deploy 配方。
2. 将应用程序的自定义 JSON 值转换为 YAML，并将格式化数据写入 app_data.yml。

将数据传递到应用程序

1. 将应用程序添加到堆栈并记录其短名称。有关更多信息，请参阅 添加应用程序。
2. 将自定义 JSON 以及应用程序的数据添加到 deploy 属性，如上文中所述。有关如何将自定义 JSON 添加到堆栈的更多信息，请参阅使用自定义 JSON。
3. 使用基于前例中的代码创建说明书并将配方添加到其中，根据需要修改在自定义 JSON 中使用的属性名称。有关如何创建说明书和配方的更多信息，请参阅说明书和诀窍。如果您已有此堆栈的自定义说明书，则还可以将配方添加到现有的说明书，甚至可以将代码添加到现有的部署说明书。
4. 在堆栈上安装说明书。有关更多信息，请参阅 安装自定义说明书。
5. 将配方分配给应用服务器层的 Deploy 生命周期事件。 AWS OpsWorks 然后，Stacks 将在每个新实例启动后对其运行配方。有关更多信息，请参阅 执行配方。

6. 部署应用程序，这还将安装当前包含您数据的堆栈配置和部署属性。

> **Note**
>
> 如果数据文件必须在部署应用程序之前到位，您还可以将配方分配到层的设置生命周期事件，该事件在实例完成启动之后立即发生一次。但是， AWS OpsWorks Stacks 尚未创建部署目录，因此您的配方应在创建数据文件之前明确创建所需的目录。以下示例明确创建应用程序的 `/shared/config` 目录，然后在该目录中创建数据文件。
>
> ```
> node[:deploy].each do |app, deploy|
>
>  directory "#{deploy[:deploy_to]}/shared/config" do
>       owner "deploy"
>       group "www-data"
>       mode 0774
>       recursive true
>       action :create
>     end
>
>   file File.join(deploy[:deploy_to], 'shared', 'config', 'app_data.yml') do
>     content YAML.dump(node[:my_app_data][app].to_hash)
>   end
> end
> ```

要加载数据，您可以使用类似于以下 Sinatra 代码的命令：

```
#!/usr/bin/env ruby
# encoding: UTF-8
require 'sinatra'
require 'yaml'

get '/' do
  YAML.load(File.read(File.join('..', '..', 'shared', 'config', 'app_data.yml')))
End
```

您可以随时更新自定义 JSON 来更新应用程序的数据值，如下所示。

更新应用程序数据

1. 编辑自定义 JSON 来更新数据值。
2. 再次部署应用程序，这会指示 AWS OpsWorks Stacks 在堆栈的实例上运行 Deploy 配方。配方将使用来自更新后堆栈配置和部署属性的属性，因此您的自定义配方将使用当前值更新数据文件。

使用 Git 存储库 SSH 密钥

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

Git 存储库 SSH 密钥，有时称为“部署 SSH 密钥”，是一种没有密码的 SSH 密钥，可通过它访问私有 Git 存储库。理想情况下，它不属于任何特定开发人员。其目的是允许 AWS OpsWorks Stacks 异步部署来自 Git 存储库的应用程序或食谱，而无需您提供任何进一步的输入。

下面介绍了创建存储库 SSH 密钥的基本过程。有关详细信息，请参阅您的存储库的相关文档。例如，管理部署密钥描述了如何为存储库创建存储库 SSH 密钥，Bitbucket 上的部署密钥描述了如何为 Bitbuck et 存储库创建存储库 SSH 密钥。 GitHub 请注意，某些文档介绍了如何在服务器上创建密钥。对于 AWS OpsWorks 堆栈，只需在说明中将 “服务器” 替换为 “工作站” 即可。

创建存储库 SSH 密钥

1. 使用 `ssh-keygen` 等程序在您的工作站上为 Git 存储库创建部署 SSH 密钥对。

   > **Important**
   >
   > AWS OpsWorks 堆栈不支持 SSH 密钥密码。

2. 将公有密钥分配给存储库，并将私有密钥存储在您的工作站上。
3. 在您添加应用程序或指定说明书存储库时，在 Repository SSH Key 框中输入私有密钥。有关更多信息，请参阅 添加应用程序。

AWS OpsWorks Stacks 将存储库 SSH 密钥传递给每个实例，然后内置配方使用该密钥连接到存储库并下载代码。在deploy 属性中将密钥存储为 node[:deploy]['*appshortname*'][:scm][:ssh_key]，并且仅可由根用户访问。

使用自定义域

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

如果您通过第三方托管一个域名，则可将该域名映射到一个应用程序。基本步骤如下所示：

1. 通过 DNS 注册商创建一个子域，然后将它映射到您的负载均衡器的弹性 IP 地址或您的应用程序服务器的公有 IP 地址。
2. 将您的应用程序的配置更新为指向该子域并重新部署应用程序。

> **Note**
>
> 确保您将未限定的域名 (如 myapp1.example.com) 转发至已限定的域名 (如 www.myapp1.example.com)，以便它们都能映射到您的应用程序。

为应用程序配置域时，域在服务器的配置文件中将作为服务器别名列出。如果您正在使用负载均衡器，则负载均衡器会在请求传入时检查 URL 中的域名并基于域重定向流量。

将子域映射到 IP 地址

1. 如果您使用的是负载均衡器，在 Instances 页面上，单击负载均衡器实例以打开其详细信息页并获取实例的 Elastic IP 地址。或者，从应用程序服务器实例的详细信息页中获取公有 IP 地址。
2. 按照您的 DNS 注册商提供的说明来创建您的子域并将其映射到步骤 1 中的 IP 地址。

> **Note**
>
> 如果负载均衡器实例在某个时间点终止，则会向您分配一个新的弹性 IP 地址。您需要更新 DNS 注册商设置以映射到新的弹性 IP 地址。

AWS OpsWorks Stacks 只是将域名设置添加到应用程序的`deploy`属性中。您必须实施自定义配方从节点对象中检索信息并正确配置服务器。有关更多信息，请参阅 说明书和诀窍。

在同一应用程序服务器上运行多个应用程序

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 本主题中的信息不适用于 Node.js 应用程序。

如果您有同一类型的多个应用程序，则在同一应用程序服务器实例上运行它们有时会更为经济高效。

在同一服务器上运行多个应用程序

1. 针对每个应用程序向堆栈添加应用程序。
2. 为每个应用程序获取单独的子域并将子域映射到应用程序服务器或负载均衡器的 IP 地址。
3. 编辑每个应用程序的配置以指定适当的子域。

有关如何执行这些任务的更多信息，请参阅使用自定义域。

> **Note**
>
> 如果您的应用程序服务器运行多个 HTTP 应用程序，则可使用 Elastic Load Balancing 进行负载均衡。对于多个 HTTPS 应用程序，您必须在负载均衡器处终止 SSL 连接或者为每个应用程

序创建单独的堆栈。HTTPS 请求已加密，这意味着，如果您在服务器处终止 SSL 连接，则负载均衡器无法检查域名来确定哪个应用程序应处理请求。

使用 SSL

Important

该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

要对您的应用程序使用 SSL，必须首先获取证书颁发机构 (CA) 签发的数字服务器证书。为简单起见，本演练将创建一个证书，然后自行签署该证书。自签名证书可用于学习和测试目的，但您应始终为生产堆栈使用由 CA 签署的证书。

在本演示中，您将执行以下操作：

1. 安装和配置 OpenSSL。
2. 创建私有密钥。
3. 创建证书签名请求。
4. 生成自签名证书。
5. 用您的证书信息编辑应用程序。

Important

如果您的应用程序使用 SSL，我们建议您尽可能在应用程序服务器层中禁用 SSLv3，以解决 CVE-2014-3566 中所述的漏洞。如果您的堆栈包含一个 Ganglia 层，您也应该为该层禁用 SSL v3。详细信息取决于特定层；有关更多信息，请参阅以下内容。

- Java 应用服务器 AWS OpsWorks 堆栈层
- Node.js App Server AWS OpsWorks 堆栈层
- PHP 应用服务器 AWS OpsWorks 堆栈层
- Rails 应用程序服务器 AWS OpsWorks 堆栈层
- 静态 Web 服务器 AWS OpsWorks 堆栈层

- Ganglia 层

主题

- 步骤 1：安装和配置 OpenSSL
- 步骤 2：创建私有密钥
- 步骤 3：创建证书签名请求
- 步骤 4：将 CSR 提交给证书颁发机构
- 步骤 5：编辑应用程序

步骤 1：安装和配置 OpenSSL

创建和上传服务器证书时，需要使用支持 SSL 和 TLS 协议的工具。OpenSSL 是一种开源工具，提供创建 RSA 令牌以及使用私有密钥进行签名所需的基本加密功能。

以下步骤假定您的计算机上还没有安装 OpenSSL。

在 Linux 和 Unix 系统上安装 OpenSSL

1. 请访问 OpenSSL：源、Tarball。
2. 下载最新源。
3. 构建软件包。

如要在 Windows 系统上安装 OpenSSL

1. 如果您的系统上尚未安装 Microsoft Visual C++ 2008 Redistributable Package，下载该软件包。
2. 运行安装程序并按照 Microsoft Visual C++ 2008 Redistributable Setup Wizard 提供的说明来安装可重新分发软件。
3. 转到 OpenSSL: Binary Distributions，单击您的环境所适用的 OpenSSL 二进制文件版本，然后在本地保存安装程序。
4. 运行安装程序，然后按照 OpenSSL Setup Wizard 中的说明来安装二进制文件。

创建指向 OpenSSL 安装点的环境变量，方法是打开一个终端窗口或命令窗口，然后使用以下命令行。

- 在 Linux 和 Unix 上

```
export OpenSSL_HOME=path_to_your_OpenSSL_installation
```

- 在 Windows 上

```
set OpenSSL_HOME=path_to_your_OpenSSL_installation
```

将 OpenSSL 二进制文件的路径添加到计算机的路径变量，方法是打开一个终端窗口或命令窗口，然后使用以下命令行。

- 在 Linux 和 Unix 上

```
export PATH=$PATH:$OpenSSL_HOME/bin
```

- 在 Windows 上

```
set Path=OpenSSL_HOME\bin;%Path%
```

> **Note**
>
> 通过使用这些命令行对环境变量所做的任何更改只对当前的命令行会话有效。

步骤 2：创建私有密钥

您需要使用一个唯一的私有密钥创建证书签名请求 (CSR)。通过使用以下命令行创建密钥：

```
openssl genrsa 2048 > privatekey.pem
```

步骤 3：创建证书签名请求

证书签名请求 (CSR) 是发送到证书颁发机构 (CA) 的文件，用于申请数字服务器证书。通过使用以下命令行创建 CSR。

```
openssl req -new -key privatekey.pem -out csr.pem
```

命令的输出与以下内容类似：

```
You are about to be asked to enter information that will be incorporated
 into your certificate request.
 What you are about to enter is what is called a Distinguished Name or a DN.
 There are quite a few fields but you can leave some blank
 For some fields there will be a default value,
 If you enter '.', the field will be left blank.
```

下表可帮助您创建证书请求。

证书请求数据

名称	描述	示例
国家/地区名称	代表国家/地区的两个字母 ISO 缩写。	US = 美国
州或省	组织所在州或省的名称。此名称不可使用缩写。	Washington
所在地名称	组织所在城市的名称。	Seattle
组织名称	组织的法定全称。请勿缩写组织名称。	CorporationX
组织部门	(可选) 用于获取额外组织信息。	市场营销
公用名	别名记录的完全限定域名。如果两者不能精确匹配，那么您会收到一条证书名称检测警告。	www.example.com
电子邮件地址	服务器管理员的电子邮件地址	someone@example.com

> **Note**
>
> 通常情况下，“Common Name”字段很容易出现误解，也不容易填写正确。公用名通常指的是您的主机加上域名。它会采取类似“www.example.com”或“example.com”的形式。您需要使用正确的公用名创建 CSR。

步骤 4：将 CSR 提交给证书颁发机构

在实际生产使用中，您可以通过将 CSR 提交给证书颁发机构 (CA) 来获取服务器证书，这可能要求提供其他证书或身份证明。如果您的应用程序成功，则 CA 将返回带有数字签名的身份证书，并可能返回一个证书链文件。AWS 不建议特定的 CA。如需获取可用的部分 CA 名单，请参阅 Wikipedia 上的证书颁发机构 – 提供商。

您也可以生成自签名证书，该证书仅可用于测试目的。对于本示例，请使用以下命令行来生成自签名证书。

```
openssl x509 -req -days 365 -in csr.pem -signkey privatekey.pem -out server.crt
```

输出将类似如下：

```
Loading 'screen' into random state - done
Signature ok
subject=/C=us/ST=washington/L=seattle/O=corporationx/OU=marketing/CN=example.com/
emailAddress=someone@example.com
Getting Private key
```

步骤 5：编辑应用程序

在生成您的证书并签名后，请更新您的应用程序以启用 SSL 并提供您的证书信息。在 Apps (应用程序) 页面上，选择一个应用程序以打开详细信息页面，然后单击 Edit App (编辑应用程序)。要启用 SSL 支持，请将 Enable SSL (启用 SSL) 设置为 Yes (是)，这将显示以下配置选项。

SSL 证书

将公有密钥证书文件 (.crt) 的内容粘贴到相应的输入框。证书应与以下内容类似：

```
-----BEGIN CERTIFICATE-----
```

```
MIICuTCCAiICCQCtqFKItVQJpzANBgkqhkiG9w0BAQUFADCBoDELMAkGA1UEBhMC
dXMxEzARBgNVBAgMCndhc2hpbmd0b24xEDAOBgNVBAcMB3NlYXR0bGUxDzANBgNV
BAoMBmFtYXpvbjEWMBQGA1UECwwNRGV2IGFuZCBUb29sczEdMBsGA1UEAwwUc3Rl
cGhhbmllYXBpZXJjZS5jb20xIjAgBgkqhkiG9w0BCQEWE3NhcGllcmNlQGFtYXpv
...
-----END CERTIFICATE-----
```

> **Note**
>
> 如果您使用的是 Nginx 并且您拥有证书链文件，则应将内容追加到公有密钥证书文件中。

如果您要更新现有证书，请执行以下操作：

- 选择 Update SSL certificate (更新 SSL 证书) 以更新证书。
- 如果新证书与现有私有密钥不匹配，请选择 Update SSL certificate key (更新 SSL 证书密钥)。
- 如果新证书与现有证书链不匹配，请选择 Update SSL certificates (更新 SSL 证书)。

SSL Certificate Key

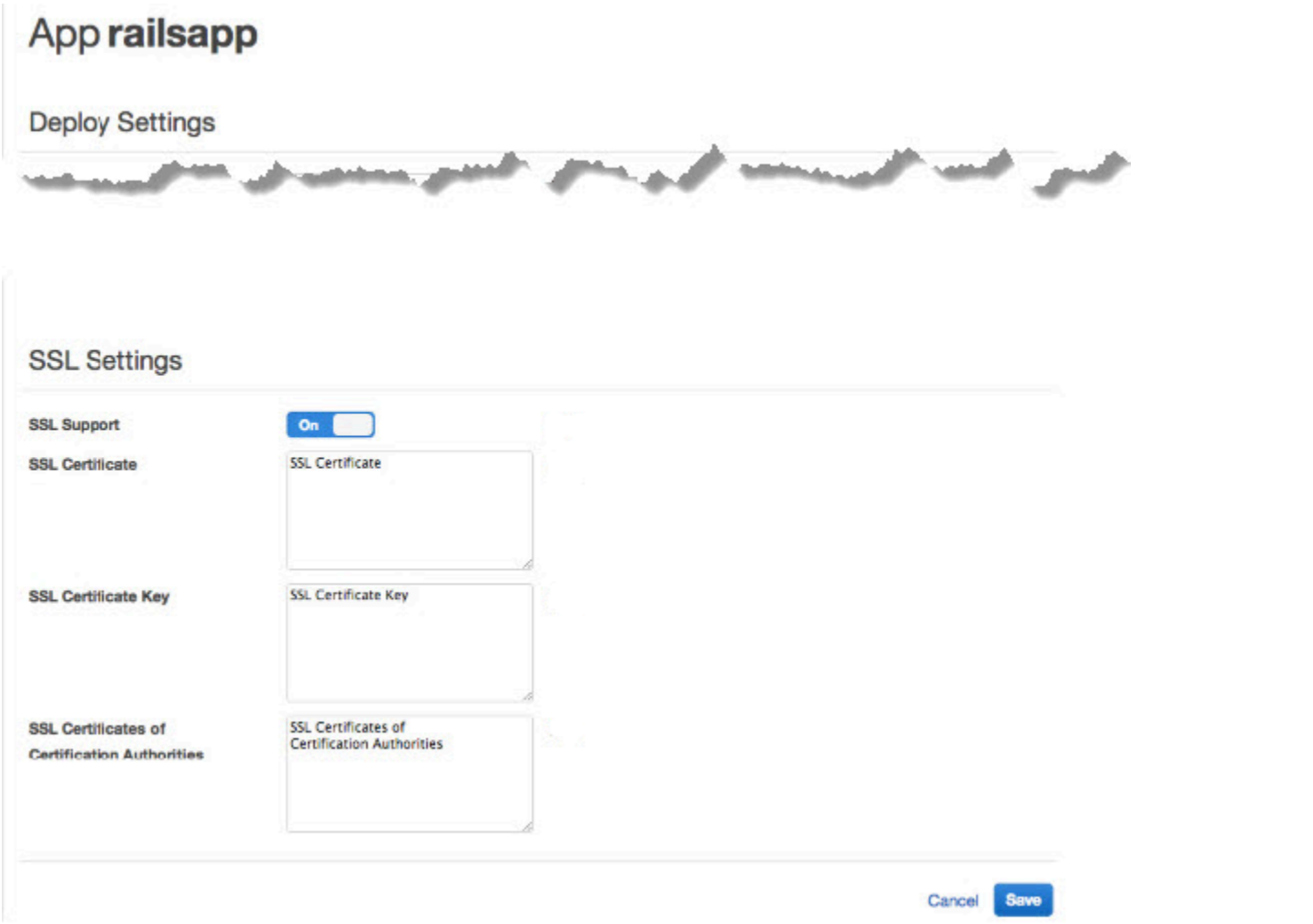
将私有密钥文件 (.pem 文件) 的内容粘贴到相应的输入框。它应与以下内容类似：

```
----BEGIN RSA PRIVATE KEY-----
MIICXQIBAAKBgQC0CYklJY5r4vV2NHQYEpwtsLuMMBhylMrgBShKq+HHVLYQQCL6
+wGIiRq5qXqZlRXje3GM5Jvcm6q0R71MfRIl1FuzKyqDtneZaAIEYniZibHiUnmO
/UNqpFDosw/6hY3ONk0fSBlU4ivD0Gjpf6J80jL3DJ4R23Ed0sdL4pRT3QIDAQAB
AoGBAKmMfWrNRqYVtGKgnWB6Tji9QrKQLMXjmHeGg95mppdJELiXHhpMvrHtpIyK
...
-----END RSA PRIVATE KEY-----
```

SSL certificates of Certification Authorities

如果您拥有证书链文件，请将内容粘贴到相应的输入框中。

> **Note**
>
> 如果您使用的是 Nginx，则应将此框留为空。如果您拥有证书链文件，请将其附加到 SSL Certificate (SSL 证书) 中的公有密钥证书文件。



单击 Save 后，重新部署应用程序以更新您的联机实例。

对于内置的应用程序服务器层， AWS OpsWorks Stacks 会自动更新服务器配置。部署完成后，您可以验证 OpenSSL 安装是否正确，如下所示。

验证 OpenSSL 安装

1. 转至 Instances 页面。
2. 通过单击应用程序服务器实例的 IP 地址来运行应用程序，或者如果您使用的是负载均衡器，也可以单击负载均衡器的 IP 地址来运行应用程序。
3. 将 IP 地址前缀从 **`http://`** 更改为 **`https://`**，然后刷新浏览器以验证页面是否使用 SSL 正确加载。

已配置应用程序要在 Mozilla Firefox 中运行的用户有时会收到以下证书错误：SEC_ERROR_UNKNOWN_ISSUER。此错误可能是由您组织的防病毒和反恶意软件程序中的证书替

换功能、某些类型的网络流量监控和过滤软件或恶意软件导致的。有关如何解决此错误的更多信息，请参阅 Mozilla Firefox 支持网站上的如何对安全网站上的安全错误代码进行故障排除。

对于包括自定义层在内的所有其他层，AWS OpsWorks Stacks 直接将 SSL 设置添加到应用程序的 deploy 属性。您必须实施自定义配方从节点对象中检索信息并正确配置服务器。

说明书和诀窍

Important

该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

AWS OpsWorks Stacks 使用 Chef 食谱来处理诸如安装和配置软件包以及部署应用程序之类的任务。本节介绍如何在 AWS OpsWorks Stacks 中使用食谱。有关更多信息，请参阅 Chef。

Note

AWS OpsWorks Stacks 目前支持 Chef 版本 12、11.10.4、11.4.4 和 0.9.15.5。但是，Chef 0.9.15.5 已弃用，建议不要将它用于新堆栈。为方便起见，通常仅使用这些版本的主要和次要版本号来指代它们。运行 Chef 0.9 或 11.4 的堆栈使用 Chef Solo，运行 Chef 12 或 11.10 的堆栈使用本地模下式的 Chef 客户端。对于 Linux 堆栈，您可以在创建堆栈时使用 Configuration Manager 指定要使用 Chef 的哪个版本。Windows 堆栈必须使用 Chef 12.2。有关更多信息 (包括关于将堆栈迁移到较新的 Chef 版本的准则)，请参阅Chef 版本。

主题

说明书存储库

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

您的自定义说明书必须存储在联机存储库中，它或者是一个存档 (如 .zip 文件)，或者是一个源代码控制管理器 (如 Git)。一个堆栈只能有一个自定义说明书存储库，但一个存储库可以包含任意多个说明书。当你安装或更新食谱时，AWS OpsWorks Stacks 会将整个存储库安装到每个堆栈实例的本地缓存中。例如，当实例需要运行一个或多个配方时，它会使用来自该本地缓存的代码。

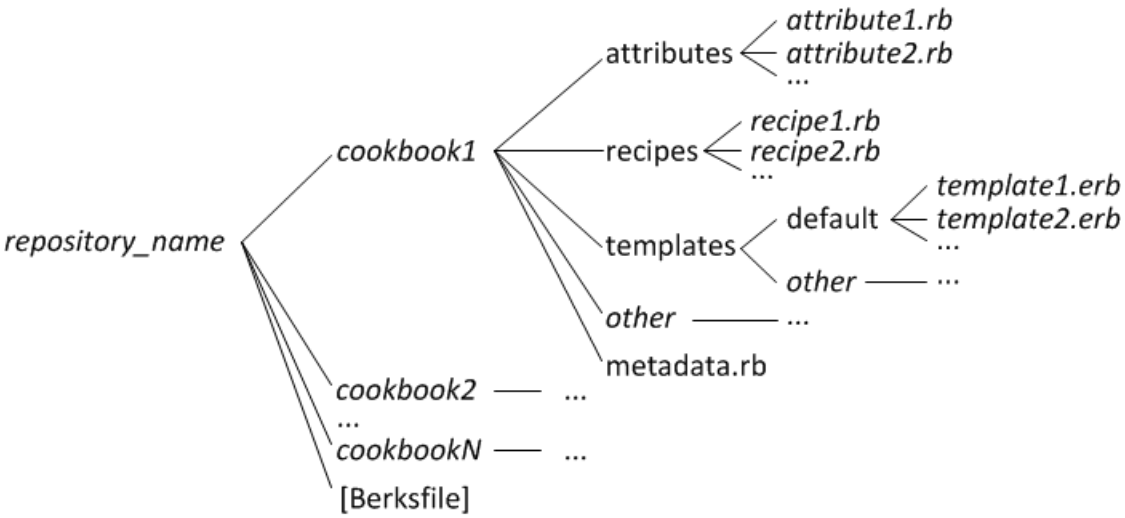
下面介绍如何构造您的说明书存储库，这取决于类型。图示中的斜体字代表用户定义的目录和文件名，包括存储库或存档名称。

源代码控制管理器

AWS OpsWorks Stacks 支持以下源代码控制管理器：

- Linux 堆栈：Git 和 Subversion
- Windows 堆栈：Git

下图显示了所需的目录和文件结构：



- 说明书目录必须全都在最高一级。

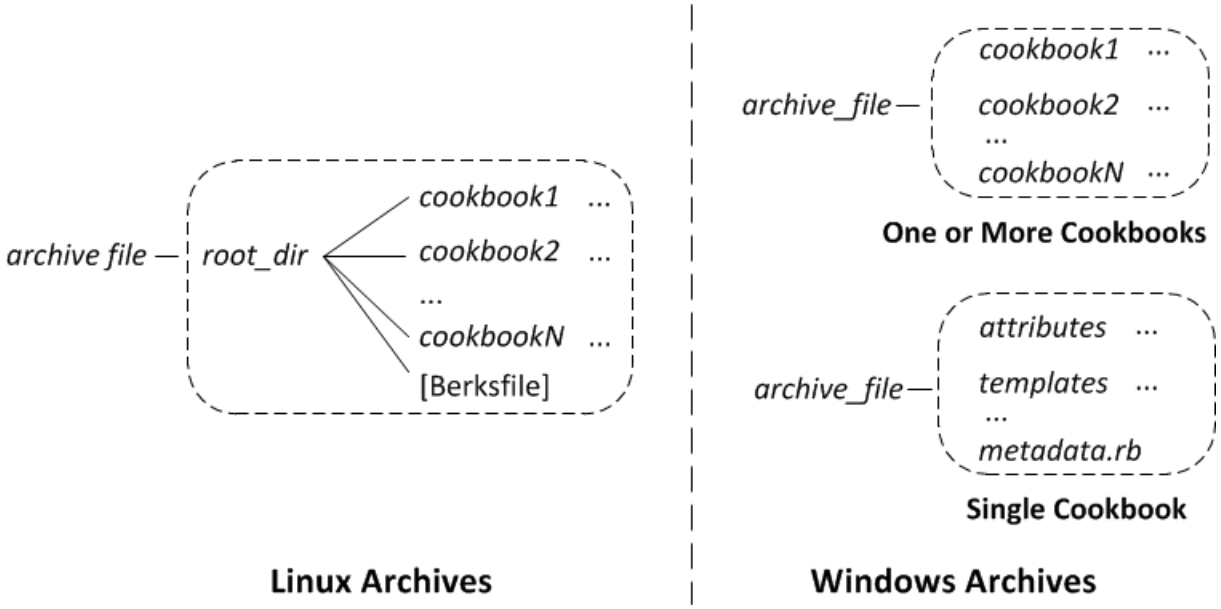
档案

AWS OpsWorks Stacks 支持以下存档：

- Linux 堆栈：zip、gzip、bzip2 或 tarball 文件，存储在 Amazon S3 或网站（HTTP 存档）上。

  AWS OpsWorks 堆栈不支持未压缩的压缩包。
- Windows 堆栈：zip 和 tgz（gzip 压缩的 tar）文件，存储在 Amazon S3 上。

下图显示了所需的目录和文件结构，这取决于您是运行 Linux 还是 Windows 堆栈。说明书结构与 SCM 存储库的一样，所以就用省略号 (...) 来表示。



- Linux 堆栈：说明书目录必须包含在根目录中。
- Windows 堆栈：说明书必须位于存档的最高一级。

  如果您只有一个说明书，可以选择省略说明书目录而将说明书文件放到最顶层。在那种情况下，AWS OpsWorks Stacks 从 metadata.rb 获取说明书名称。

每个说明书目录都至少拥有下面一个标准目录和文件，但通常拥有所有这些标准目录和文件，这些目录和文件必须使用标准名称：

- `attributes`：说明书的属性文件。
- `recipes`：说明书的配方文件。
- `templates`：说明书的模板文件。
- `##`：可选的用户定义目录，其中包含其他文件类型，如定义或规范。

- `metadata.rb`：说明书的元数据。

  对于 Chef 11.10 及更高版本，如果您的配方依靠其他说明书，则您必须在说明书的 `depends` 文件中包括相应的 `metadata.rb` 语句。例如，如果您的说明书包括含有类似于 `include_recipe anothercookbook::somerecipe` 的语句的配方，则您说明书的 `metadata.rb` 文件必须包括以下行：`depends "anothercookbook"`。有关更多信息，请参阅关于说明书元数据。

模板必须位于 `templates` 的子目录下，其中包含至少一个或多个子目录。这些子目录下还可以有子目录。

- 模板通常具有一个 `default` 子目录，其中包含 Chef 默认使用的模板文件。
- other 代表可用于操作系统特定模板的可选子目录。
- Chef 会基于文件特异性中描述的命名约定自动使用相应子目录中的模板。例如，对于 Linux 和 操作系统，您可以将操作系统特定的模板放在名为 amazonamazon 或 ubuntuubuntu 的子目录中。

具体如何处理自定义说明书取决于您的首选存储库类型。

使用存档

1. 使用上一节所示的文件夹结构实现您的说明书。
2. 创建压缩存档文件并将其上传到 Amazon S3 存储桶或网站。

   如果您更新说明书，则必须创建并上传新的存档文件。发送到 Amazon S3 存储桶的内容可能包含客户内容。有关删除敏感数据的更多信息，请参阅如何清空 S3 存储桶？或如何删除 S3 存储桶？。

使用 SCM

1. 使用前面所示的结构设置 Git 或 Subversion 存储库。
2. (可选) 使用存储库的版本控制功能来实施多个分支或版本。

   如果您更新食谱，则可以在新分支中进行更新，然后直接 OpsWorks 使用新版本。您也可以指定特定标记的版本。有关更多信息，请参阅 指定自定义说明书存储库。

安装自定义说明书描述了如何让 AWS OpsWorks Stacks 在堆栈的实例上安装你的食谱存储库。

> **Important**
>
> 更新存储库中的现有食谱后，必须运行 AWS OpsWorks st update_cookbooks ack 命令指示 Stacks 更新每个在线实例的本地缓存。有关更多信息，请参阅 运行堆栈命令。

Chef 版本

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

AWS OpsWorks Stacks 支持多个版本的 Chef。您在创建堆栈时选择版本。 AWS OpsWorks 然后，Stacks 在堆栈的所有实例上安装该版本的 Chef 以及一组与该版本兼容的内置配方。如果您安装任何自定义配方，它们必须与堆栈的 Chef 版本兼容。

AWS OpsWorks Stacks 目前支持 Linux 堆栈的 Chef 版本 12、11.10、11.4 和 0.9，支持 Windows 堆栈的 Chef 12.2（目前为 Chef 12.22）。为方便起见，通常仅使用这些版本的主要和次要版本号来指代它们。对于 Linux 堆栈，您可以在创建堆栈时使用 Configuration Manager 指定要使用 Chef 的哪个版本。Windows 堆栈必须使用 Chef 12.2。有关更多信息 (包括关于将堆栈迁移到较新的 Chef 版本的准则)，请参阅Chef 版本。有关完整的版本信息，请参阅AWS OpsWorks 堆栈操作系统。

Chef 12.2

Chef 12.2 支持于 2015 年 5 月引入，仅供 Windows 堆栈使用。Windows 堆栈中的当前 Chef 版本为 Chef 12.22。它与 Ruby 2.3.6 一起运行，并以本地模式使用 chef-client，这会启动一个称为 chef-zero 的本地内存 Chef Server。此服务器启动后，使配方可以使用 Chef 搜索和数据包。这种支持存在一些限制 (实施配方：Chef 12.2 中有相关介绍)，但您可以在未经修改的情况下运行许多社区说明书。

Chef 12

Chef 12 支持于 2015 年 12 月引入，仅供 Linux 堆栈使用。它与 Ruby 2.1.6 或 2.2.3 一起运行并以本地模式使用 chef-client，这使配方可以使用 Chef 搜索和数据包。有关更多信息，请参阅 AWS OpsWorks 堆栈操作系统。

Chef 11.10

Chef 11.10 支持于 2014 年 3 月引入，仅供 Linux 堆栈使用。它与 Ruby 2.0.0 一起运行并以本地模式使用 chef-client，这使配方可以使用 Chef 搜索和数据包。这种支持存在一些限制 (实施配方：Chef 11.10 中有相关介绍)，但您可以在未经修改的情况下运行许多社区说明书。您也可以使用 Berkshelf 管理您的说明书依赖项。哪个版本的 Berkshelf 受支持，取决于操作系统。有关更多信息，请参阅 AWS OpsWorks 堆栈操作系统。您不能创建使用 11.10 的 CentOS 堆栈。

Chef 11.4

Chef 11.4 于 2013 年 7 月引入，仅供 Linux 堆栈使用。它与 Ruby 1.8.7 一起运行并使用 chef-solo (不支持 Chef 搜索或数据包)。您通常可以在 AWS OpsWorks Stacks 中使用依赖于这些功能的社区食谱，但必须按照中所述对其进行修改。迁移到新的 Chef 版本您不能创建使用 11.4 的 CentOS 堆栈。美国东部（弗吉尼亚州北部）外的区域端点不支持 Chef 11.4 堆栈。

Chef 0.9

Chef 0.9 只能由 Linux 堆栈使用，且不再受支持。请注意以下详细信息：

- 您不能使用控制台创建新的 Chef 0.9 堆栈。

  您必须使用 CLI 或 API，或者您必须创建使用不同的 Chef 版本的堆栈，然后编辑堆栈配置。
- 新的 AWS OpsWorks 堆栈功能不适用于 Chef 0.9 堆栈。
- 新的操作系统版本将只提供针对 Chef 0.9 堆栈的有限支持。

  具体而言，Amazon Linux 2014.09 及更高版本不支持具有依赖 Ruby 1.8.7 的 Rails 应用程序服务器层的 Chef 0.9 堆栈。
- 新的 Amazon Web Services Region (包括欧洲地区 [法兰克福]) 不支持 Chef 0.9 堆栈。

> **Note**
>
> 我们不建议对新堆栈使用 Chef 0.9。您应该尽快将任何现有堆栈迁移到最新的 Chef 版本。

如果你想在堆栈中使用社区食谱，我们建议你为新的 Linux AWS OpsWorks 堆栈指定 Chef 12，并将现有的 Linux 堆栈迁移到 Chef 12。你可以使用 AWS OpsWorks Stacks 控制台、API 或 CLI 将现有堆栈迁移到较新的 Chef 版本。有关更多信息，请参阅 迁移到新的 Chef 版本。

主题

- 为 Chef 12.2 堆栈实施配方
- 为 Chef 12 堆栈实施配方

- 为 Chef 11.10 堆栈实施配方
- 为 Chef 11.4 堆栈实施配方
- 将现有 Linux 堆栈迁移到新的 Chef 版本

为 Chef 12.2 堆栈实施配方

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

Chef 12.2 (当前为 Chef 12.22) 仅对 Windows 堆栈可用，而这些堆栈必须运行该 Chef 版本。

- 配方必须使用特定于 Windows 的属性和资源来实现一些目的。

  有关更多信息，请参阅 Chef for Microsoft Windows。
- Chef 运行使用 Ruby 2.3.6，所以您的配方可以使用新的 Ruby 语法。
- 配方可以使用 Chef 搜索和数据包。

  Chef 12.2 堆栈可以使用许多社区说明书，而无需修改。有关更多信息，请参阅 使用 Chef 搜索 和 使用数据包。
- AWS OpsWorks 堆栈数据包参考和内置说明书属性中介绍的大多数堆栈配置和部署属性都可用于 Windows 配方。

  您可以使用 Chef 搜索来获取这些属性值。有关示例，请参阅使用 Chef 搜索获取属性值。有关属性的列表，请参阅AWS OpsWorks 堆栈数据包参考。

为 Chef 12 堆栈实施配方

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

相对于 Chef 11.10 堆栈，Chef 12 堆栈提供以下优势：

- Chef 运行使用 Ruby 2.1.6，所以您的配方可以使用新的 Ruby 语法。
- Chef 12 堆栈可以使用更多社区说明书，而无需修改。由于没有任何内置说明书，因此，不可能出现内置说明书与自定义说明书之间发生名称冲突的情况。
- 您不再局限于 AWS OpsWorks Stacks 为其提供预建软件包的 Berkshelf 版本。在 Chef 12 中，Berkshelf 不再安装在 AWS OpsWorks Stacks 实例上。相反，您可以在您的本地工作站上使用任何 Berkshelf 版本。
- 现在， AWS OpsWorks Stacks 在 Chef 12 中提供的内置食谱（Elastic Load Balancing、Amazon RDS 和 Amazon ECS）和自定义食谱之间有明显的区别。这就使得对失败的 Chef 运行进行故障排除变得更加容易。

为 Chef 11.10 堆栈实施配方

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

相对于 Chef 11.4 堆栈，Chef 11.10 堆栈提供以下优势：

- Chef 运行使用 Ruby 2.0.0，所以您的配方可以使用新的 Ruby 语法。
- 配方可以使用 Chef 搜索和数据包。

  Chef 11.10 堆栈可以使用许多社区说明书，而无需修改。
- 您可以使用 Berkshelf 管理说明书。

  Berkshelf 提供了一个更灵活的方法来管理您的自定义说明书，以及在堆栈中使用社区说明书。
- 说明书必须在 `metadata.rb` 中宣布依赖项。

  如果您的说明书依赖另一个说明书，则您必须在您说明书的 `metadata.rb` 文件中包括该依赖项。例如，如果您的说明书包括含有类似于 `include_recipe anothercookbook::somerecipe` 的语句的配方，则您说明书的 `metadata.rb` 文件必须包括以下行：`depends "anothercookbook"`。
- AWS OpsWorks 只有当堆栈包含 MySQL 层时，堆栈才会在堆栈的实例上安装 MySQL 客户端。

- AWS OpsWorks 只有当堆栈包含 Ganglia 层时，Stacks 才会在堆栈的实例上安装 Ganglia 客户端。
- 如果部署运行 `bundle install`，并且安装失败，则部署也会失败。

> **Important**
>
> 请勿为自定义或社区说明书重复使用内置说明书名称。与内置说明书具有相同名称的自定义说明书可能会失败。有关 Chef 11.10、11.4 和 0.9 堆栈中可用的内置食谱的完整列表，请参阅 opsworks-cookbooks 存储库。 GitHub
>
> 包含非 ASCII 字符的说明书在 Chef 0.9 和 11.4 堆栈上可以成功运行，而在 Chef 11.10 堆栈上可能会失败。原因在于，Chef 11.10 堆栈为 Chef 运行使用 Ruby 2.0.0，这在编码上要比 Ruby 1.8.7 严格得多。为确保这些说明书在 Chef 11.10 堆栈上成功运行，使用非 ASCII 字符的每个文件都应当在顶部包含一个注释，以提供关于编码的提示。例如，对于 UTF-8 编码，注释将为 `# encoding: UTF-8`。有关 Ruby 2.0.0 编码的更多信息，请参阅 Encoding。

主题

说明书安装和优先顺序

Chef 11.10 AWS OpsWorks 堆栈的安装 Stacks 食谱的过程与早期的 Chef 版本略有不同。对于 Chef 11.10 堆栈，在 AWS OpsWorks Stacks 安装内置、自定义和 Berkshelf 食谱后，它会按以下顺序将它们合并到公共目录中：

1. 内置说明书。
2. Berkshelf 说明书 (如果有)。
3. 自定义说明书 (如果有)。

当 AWS OpsWorks Stacks 执行此合并时，它会复制目录的全部内容，包括配方。如果有重复项，则将依照以下规则处理：

- Berkshelf 说明书的内容优先于内置说明书。

- 自定义说明书的内容优先于 Berkshelf 说明书。

为了说明此过程的原理，请想一想以下场景，其中所有这三个说明书目标都包括名为 mycookbook 的说明书：

- 内置说明书：mycookbook 包括一个名为 someattributes.rb 的属性文件、一个名为 sometemplate.erb 的模板文件以及一个名为 somerecipe.rb 的配方。
- Berkshelf 说明书：mycookbook 包括 sometemplate.erb 和 somerecipe.rb。
- 自定义说明书：mycookbook 包括 somerecipe.rb。

合并的说明书包含以下内容：

- 内置说明书中的 someattributes.rb。
- Berkshelf 说明书中的 sometemplate.erb。
- 自定义说明书的 somerecipe.rb。

> **Important**
>
> 您不应当通过将整个内置说明书复制到您的存储库，然后修改该说明书的部分内容来自定义您的 Chef 11.10 堆栈。这样做会覆盖整个内置说明书，包括配方。如果 AWS OpsWorks Stacks 更新了该食谱，则除非您手动更新私有副本，否则您的堆栈将无法从这些更新中受益。有关如何自定义堆栈的更多信息，请参阅自定义堆栈 AWS OpsWorks。

使用 Chef 搜索

您可以在您的配方中使用 Chef search 方法来查询堆栈数据。你使用与 Chef 服务器相同的语法，但是 AWS OpsWorks Stacks 从本地节点对象获取数据，而不是查询 Chef 服务器。这些数据包括：

- 实例的堆栈配置和部署属性。
- 实例的内置和自定义说明书的属性文件中的属性。
- Ohai 收集的系统数据。

堆栈配置和部署属性包含配方通常通过搜索获得的大部分信息，包括堆栈中每个在线实例的主机名和 IP 地址等数据。 AWS OpsWorks 堆栈会为每个生命周期事件更新这些属性，从而确保它们准确反映当

前堆栈状态。这意味着，您可以经常在您的堆栈中使用与搜索相关的社区配方，而无需修改。搜索方法仍返回适当的数据；但这些数据来自堆栈配置和部署属性，而不是服务器。

AWS OpsWorks Stacks 搜索的主要局限性在于，它只能处理本地节点对象中的数据，尤其是堆栈配置和部署属性。因此，可能无法通过搜索获得以下类型的数据：

- 其他实例上本地定义的属性。

  如果配方在本地定义了属性，则该信息不会报告给 AWS OpsWorks Stacks 服务，因此您无法使用搜索从其他实例访问该数据。
- 自定义 `deploy` 属性。

  您可以在部署应用程序时指定自定义 JSON，然后便会在堆栈针对该部署的实例上安装相应的属性。但是，如果您仅部署到所选的实例上，则将仅在这些实例上安装属性。针对这些自定义 JSON 属性的查询在所有其他实例上都将失败。此外，自定义属性仅包含在针对该特定部署的堆栈配置和部署 JSON 中。只有当下一个生命周期事件安装了一组新的堆栈配置和部署属性后，才能访问这些属性。请注意，如果您为堆栈指定自定义 JSON，则会在每个生命周期事件的每个实例上安装这些属性，并且这些属性在整个搜索期间始终可以访问。
- 其他实例中的 Ohai 数据。

  Chef 的 Ohai 工具获取实例上的各种系统数据，然后将这些数据添加到节点对象中。这些数据存储在本地，并且不会反馈给 AWS OpsWorks Stacks 服务，因此搜索无法从其他实例中访问 Ohai 数据。但是，堆栈配置和部署属性中可能包含其中一些数据。
- 离线实例。

  堆栈配置和部署属性仅包含联机实例的数据。

以下配方摘录显示了如何使用搜索获取 PHP 层的实例的私有 IP 地址。

```
appserver = search(:node, "role:php-app").first
Chef::Log.info("The private IP is '#{appserver[:private_ip]}'")
```

> **Note**
>
> 当 AWS OpsWorks Stacks 向节点对象添加堆栈配置和部署属性时，它实际上会创建两组图层属性，每组属性都具有相同的数据。一个集合位于`layers`命名空间中，这就是 AWS OpsWorks Stacks 存储数据的方式。另一组属性位于 `role` 命名空间中，用于定义 Chef 服务

器存储等效数据的方式。`role`命名空间的目的是允许为 Chef 服务器实现的搜索代码在 AWS OpsWorks Stacks 实例上运行。如果您正在专门为 AWS OpsWorks Stacks 编写代码，则可以在前面的示例`role:php-app`中使用`layers:php-app`或并`search`返回相同的结果。

使用数据包

您可以在您的配方中使用 Chef `data_bag_item` 方法来查询数据包中的信息。您使用与将为 Chef 服务器使用的语法相同的语法，但 AWS OpsWorks Stacks 从实例的堆栈配置和部署属性中获取数据。但是， AWS OpsWorks Stacks 目前不支持 Chef 环境，因此请`node.chef_environment`务必返回`_default`。

您可以通过使用自定义 JSON 向 `[:opsworks][:data_bags]` 属性添加一个或多个属性，来创建数据包。以下示例显示了在自定义 JSON 中创建数据包的一般格式。

> **Note**
>
> 您不能通过将数据包添加到您的说明书存储库中来创建数据包。您必须使用自定义 JSON。

```
{
  "opsworks": {
    "data_bags": {
      "bag_name1": {
        "item_name1: {
          "key1" : “value1”,
          "key2" : “value2”,
          ...
        }
      },
      "bag_name2": {
        "item_name1": {
          "key1" : “value1”,
          "key2" : “value2”,
          ...
        }
      },
      ...
    }
  }
```

```
}
```

您通常为堆栈指定自定义 JSON，这会在后继的每个生命周期事件的每个实例上安装自定义属性。您还可以在部署应用程序时指定自定义 JSON，但仅为该部署安装这些属性，并且这些属性可能将仅安装到一组所选的实例中。有关更多信息，请参阅 部署应用程序。

以下自定义 JSON 示例创建名为 myapp 的数据包。它包含一个项目 mysql 以及两个密钥值对。

```
{ "opsworks": {
    "data_bags": {
      "myapp": {
        "mysql": {
          "username": "default-user",
          "password": "default-pass"
        }
      }
    }
  }
}
```

要使用您的配方中的数据，您可以调用 data_bag_item 并向其传递数据包和值名称，如以下摘录所示。

```
mything = data_bag_item("myapp", "mysql")
Chef::Log.info("The username is '#{mything['username']}' ")
```

要修改数据包中的数据，仅修改自定义 JSON 即可，然后自定义 JSON 将安装在下一个生命周期事件的堆栈的实例上。

使用 Berkshelf

对于 Chef 0.9 和 Chef 11.4 堆栈，您仅可以安装一个自定义说明书存储库。对于 Chef 11.10 堆栈，您可以使用 Berkshelf 管理您的说明书及其依赖项，这使您可以从多个存储库中安装说明书。（有关更多信息，请参阅 本地打包说明书依赖项。）特别是，使用 Berkshelf，您可以直接从其存储库中安装与 AWS OpsWorks Stacks 兼容的社区食谱，而不必将它们复制到您的自定义食谱存储库中。哪个版本的 Berkshelf 受支持，取决于操作系统。有关更多信息，请参阅 AWS OpsWorks 堆栈操作系统。

要使用 Berkshelf，您必须明确启用它，如安装自定义说明书中所述。然后，将一个 Berksfile 文件放入您的说明书存储库的根目录 (指明要安装哪些说明书)。

要指定 Berksfile 中的外部说明书源，可在该文件的顶部添加一个源属性以指定默认存储库 URL。Berkshelf 将在源 URL 中查找说明书，除非您明确指定一个存储库。然后为您希望安装的每个说明书添加一行，格式如下：

```
cookbook 'cookbook_name', ['>= cookbook_version'], [cookbook_options]
```

cookbook 后面的字段指定特定的说明书。

- *cookbook_name*：（必需）指定说明书的名称。

  如果您没有添加任何其他字段，Berkshelf 将从指定的源 URL 中安装说明书。
- *cookbook_version*：（可选）指定说明书版本。

  您可以使用诸如 = 或 >= 等前缀指定特定版本或一系列可接受的版本。如果您不指定版本，则 Berkshelf 将安装最新的版本。
- *cookbook_options*：（可选）最后一个字段是一个哈希值，它包含一个或多个用于指定存储库位置等选项的密钥值对。

  例如，您可以添加一个 git 密钥来指定特定的 Git 存储库，添加一个 tag 密钥来指定特定的存储库分支。指定存储库分支通常是确保安装您的首选说明书的最佳方式。

> **Important**
>
> 切勿通过以下方式宣布说明书：在您的 Berksfile 中添加 metadata 行，并在 metadata.rb 中宣布说明书依赖项。为了能够正确执行这一操作，这两个文件必须位于同一目录中。对于 AWS OpsWorks Stacks，Berksfile 必须位于存储库的根目录中，但metadata.rb文件必须位于各自的食谱目录中。您应当在 Berksfile 中明确宣布外部说明书。

下面是 Berksfile 的一个示例，介绍了指定说明书的不同方法。有关如何创建 Berksfile 的更多信息，请参阅 Berkshelf。

```
source "https://supermarket.chef.io"
```

```
cookbook 'apt'
cookbook 'bluepill', '>= 2.3.1'
cookbook 'ark', git: 'git://github.com/opscode-cookbooks/ark.git'
cookbook 'build-essential', '>= 1.4.2', git: 'git://github.com/opscode-cookbooks/build-
essential.git', tag: 'v1.4.2'
```

此文件安装以下说明书：

- 社区说明书存储库中最新版本的 `apt`。
- 社区说明书中最新版本的 `bluepill`，但前提是它是 2.3.1 版本或更高版本。
- 指定存储库中最新版本的 `ark`。

  此示例的 URL 适用于上的公共社区食谱存储库 GitHub，但您可以安装来自其他存储库（包括私有存储库）的食谱。有关更多信息，请参阅 Berkshelf。
- 指定存储库的 v1.4.2 分支中的 `build-essential` 说明书。

除了 Berksfile，自定义说明书存储库还可以包含自定义说明书。在这种情况下， AWS OpsWorks Stacks 会安装两套食谱，这意味着一个实例可以有多达三个食谱存储库。

- 内置说明书安装到 `/opt/aws/opsworks/current/cookbooks` 中。
- 如果您的自定义说明书存储库包含说明书，则它们将安装到 `/opt/aws/opsworks/current/site-cookbooks` 中。
- 如果您已启用了 Berkshelf，并且您的自定义说明书存储库包含 Berksfile，则指定说明书将安装到 `/opt/aws/opsworks/current/berkshelf-cookbooks` 中。

在安装过程中，内置食谱和您的自定义食谱会安装在每个实例上，除非您手动运行更新自定义食谱堆栈命令，否则不会随后进行更新。 AWS OpsWorks Stacks 在`berks install`每次 Chef 运行时都会运行，因此您的 Berkshelf 食谱会根据以下规则针对每个生命周期事件进行更新：

- 如果存储库中有新的说明书版本，则此操作将从存储库更新说明书。
- 否则，此操作将从本地缓存更新 Berkshelf 说明书。

> **Note**
>
> 此操作将覆盖 Berkshelf 说明书，因此，如果您修改了任何说明书的本地副本，则所做的更改将被覆盖。有关更多信息，请参阅 Berkshelf。

您还可以通过以下方法更新您的 Berkshelf 说明书：运行 Update Custom Cookbooks 堆栈命令，这会同时更新 Berkshelf 说明书和您的自定义说明书。

为 Chef 11.4 堆栈实施配方

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Important**
>
> 请勿为自定义或社区说明书重复使用内置说明书名称。与内置说明书具有相同名称的自定义说明书可能会失败。有关 Chef 11.10、11.4 和 0.9 堆栈中可用的内置食谱的完整列表，请参阅 opsworks-cookbooks 存储库。 GitHub

Chef 11.4 堆栈的主要限制是：配方不能使用 Chef 搜索或数据包。但是， AWS OpsWorks Stacks 会在每个实例上安装堆栈配置和部署属性，这些属性包含您通过搜索获得的大部分信息，包括以下内容：

- 控制台中用户定义的数据，如主机名或应用程序名称。
- Stac AWS OpsWorks ks 服务生成的堆栈配置数据，例如堆栈的层、应用程序和实例，以及有关每个实例的详细信息，例如 IP 地址。
- 包含用户提供的、几乎能够充当数据包来进行使用的数据的自定义 JSON 属性。

AWS OpsWorks 在事件的 Chef 运行开始之前，Stacks 会在每个生命周期事件的每个实例上安装最新版本的堆栈配置和部署属性。配方可通过标准 `node[:attribute][:child_attribute][...]` 语法获得这些数据。例如，堆栈配置和部署属性包括堆栈名称 `node[:opsworks][:stack][:name]`。

以下内容摘自其中一个内置配方，这部分摘录内容可获得堆栈名称并使用这个名称创建一个配置文件。

```
template '/etc/ganglia/gmetad.conf' do
  source 'gmetad.conf.erb'
  mode '0644'
```

```
  variables :stack_name => node[:opsworks][:stack][:name]
  notifies :restart, "service[gmetad]"
end
```

许多堆栈配置和部署属性值包含多个属性。您必须循环访问这些属性来获取所需的信息。下面的示例显示了堆栈配置和部署属性 (为方便起见，我们将其表示为 JSON 对象) 的摘录内容。它包含一个顶级属性 deploy，这个属性包含每个堆栈的应用程序的属性，并以应用程序的简称命名。

```
{
  ...
  "deploy": {
    "app1_shortname": {
      "document_root": "app1_root",
      "deploy_to": "deploy_directory",
      "application_type": "php",
      ...
    },
    "app2_shortname": {
      "document_root": "app2_root",
      ...
    }
  },
  ...
}
```

每个应用程序属性都包含一组描述应用程序特征的属性。例如，deploy_to 属性表示应用程序的部署目录。以下摘录设定每个应用程序的部署目录的用户、组和路径。

```
node[:deploy].each do |application, deploy|
  opsworks_deploy_dir do
    user deploy[:user]
    group deploy[:group]
    path deploy[:deploy_to]
  end
  ...
end
```

有关堆栈配置和部署属性的更多信息，请参阅自定义堆栈 AWS OpsWorks。有关部署目录的更多信息，请参阅Deploy 配方。

Chef 11.4 堆栈不支持数据包，但您可以通过指定自定义 JSON 向堆栈配置和部署属性中添加任意数据。然后，您的配方可以使用标准 Chef 节点语法访问这些数据。有关更多信息，请参阅 使用自定义 JSON。

如果您需要使用加密数据包的功能，一种方法是将敏感属性存储到安全的位置，如私有 Amazon S3 存储桶。然后，您的配方可以使用安装在所有 AWS OpsWorks Stacks 实例上的 AWS Ruby 软件开发工具包从存储桶下载数据。

> **Note**
>
> 每个 AWS OpsWorks Stacks 实例都有一个实例配置文件。关联的IAM 角色指定在实例上运行的应用程序可以访问哪些 AWS 资源。要使您的配方能够访问 Amazon S3 存储桶，该角色的策略必须包括类似以下内容的语句，这可授予从指定的存储桶检索文件的权限。
>
> ```
> "Action": ["s3:GetObject"],
> "Effect": "Allow",
> "Resource": "arn:aws:s3:::yourbucketname/*",
> ```
>
> 有关实例配置文件的更多信息，请参阅为在 EC2 实例上运行的应用程序指定权限。

将现有 Linux 堆栈迁移到新的 Chef 版本

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

你可以使用 AWS OpsWorks Stacks 控制台、API 或 CLI 将你的 Linux 堆栈迁移到较新的 Chef 版本。但是，您可能需要修改配方以使其与更新的版本兼容。在准备迁移堆栈时，请考虑以下内容。

- 您无法通过编辑或克隆 AWS OpsWorks堆栈将堆栈版本从 Chef 11 更改为 Chef 12。使用这部分介绍的过程无法执行 Chef 主要版本升级。有关将 Chef 11.10 转换为 Chef 12 的更多信息，请参阅实施配方：Chef 12。
- 从一个 Chef 版本转换为另一版本需要进行许多更改，其中一些是重大更改。

有关将 Chef 0.9 转换为 Chef 11.4 的更多信息，请参阅迁移到新的 Chef 版本。有关将 Chef 11.4 转换为 Chef 11.10 的更多信息，请参阅实施配方：Chef 11.10。有关将 Chef 11.10 转换为 Chef 12 的更多信息，请参阅实施配方：Chef 12。

- Chef 运行对不同的 Chef 版本使用不同的 Ruby 版本：Chef 0.9 和 Chef 11.4 堆栈 (Ruby 1.8.7)、Chef 11.10 堆栈 (Ruby 2.0.0) 以及 Chef 12 堆栈 (Ruby 2.1.6)。

  有关更多信息，请参阅 Ruby 版本。

- Chef 11.10 堆栈安装说明书的方式与 Chef 0.9 或 Chef 11.4 堆栈不同。

  在将使用自定义说明书的堆栈迁移到 Chef 11.10 时，这种差别可能会引发问题。有关更多信息，请参阅 说明书安装和优先顺序。

以下是针对将 Chef 堆栈迁移到更新的 Chef 版本的建议指南：

将堆栈迁移到更新的 Chef 版本

1. 克隆您的生产堆栈。在 Clone Stack 页面上，单击 Advanced>> 可显示 Configuration Management 部分，然后将 Chef version 更改为高一级别的版本。

   > **Note**
   >
   > 如果您从 Chef 0.9 堆栈开始，无法直接升级到 Chef 11.10；您必须先升级到 Chef 11.4。如果您希望在测试您的配方之前将您的堆栈迁移到 Chef 11.10，请等待 20 分钟，以便执行更新，然后将堆栈从 11.4 升级到 11.10。

2. 将实例添加到层中，并在测试或临时系统中测试克隆的堆栈的应用程序和说明书。有关更多信息，请参阅 All about Chef ...。
3. 当测试结果符合您的要求时，请执行以下操作之一：

   - 如果这是您所需的 Chef 版本，您可以使用克隆的堆栈作为您的生产堆栈，否则，您可以重置您的生产堆栈上的 Chef 版本。
   - 如果要分两个阶段将 Chef 0.9 堆栈迁移到 Chef 11.10，可以重复执行将堆栈从 Chef 11.4 迁移到 Chef 11.10 的过程。

> **Note**
>
> 测试配方时，您可以使用 SSH 连接到实例，然后使用实例代理 CLI run_command 命令运行与各个生命周期事件关联的配方。代理 CLI 对于测试 Setup 配方尤其有用，因为您甚至可以在 Setup 失败且实例没有处于在线状态的情况下使用它。您还可以使用 Setup 堆栈命令来返回 Setup 配方，但该命令仅在 Setup 成功且实例处于在线状态的情况下可用。

可以将正在运行的堆栈更新到新的 Chef 版本。

将正在运行的堆栈更新到新的 Chef 版本

1. 编辑堆栈以更改 Chef version 堆栈设置。
2. 保存新设置并等待 AWS OpsWorks Stacks 更新实例，这通常需要 15-20 分钟。

> **Important**
>
> AWS OpsWorks Stacks 不会将 Chef 版本更新与生命周期事件同步。如果要更新生产堆栈上的 Chef 版本，您必须谨慎，以确保在更新完成之后再进入下一个生命周期事件。如果发生了事件（通常为 Deploy 或 Configure 事件），实例代理会更新您的自定义说明书，并运行该事件的已分配配方，无论版本更新是否已完成。关于确定版本更新何时已完成，没有直接的方法，但部署日志中包含 Chef 版本。

Ruby 版本

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

Linux 堆栈中的所有实例都安装了 Ruby。 AWS OpsWorks Stacks 在每个实例上安装一个 Ruby 软件包，用于运行 Chef 配方和实例代理。 AWS OpsWorks 堆栈根据堆栈运行的 Chef 版本来确定 Ruby 版本。切勿试图修改此版本；这样做可能会禁用实例代理。

AWS OpsWorks 堆栈不会在 Windows 堆栈上安装应用程序 Ruby 可执行文件。Chef 12.2 客户端附带 Ruby 2.0.0 p451，但不会将 Ruby 可执行文件添加到实例的 PATH 环境变量。如果您想要使用此可执行文件来运行 Ruby 代码，可以在您的 Windows 驱动器上的 `\opscode\chef\embedded\bin\ruby.exe` 中找到该代码。

下表汇总了 AWS OpsWorks Stacks Ruby 版本。可用的应用程序 Ruby 版本还取决于实例的操作系统。有关更多信息，包括可用的补丁版本的信息，请参阅 AWS OpsWorks 堆栈操作系统。

Chef 版本	Chef Ruby 版本	可用的应用程序 Ruby 版本
0.9 (c)	1.8.7	1.8.7(a)、1.9.3(e)、2.0.0
11.4 (c)	1.8.7	1.8.7(a)、1.9.3(e)、2.0.0、2.1、2.2.0、2.3
11.10	2.0.0-p481	1.9.3(c, e)、2.0.0、2.1、2.2.0、2.3、2.6.1
12 (b)	2.1.6、2.2.3	无
12.22 (d)	2.3.6	无

(a) 对 Amazon Linux 2014.09 及更高版本、Red Hat Enterprise Linux (RHEL) 或 Ubuntu 14.04 LTS 不可用。

(b) 仅在 Linux 堆栈上可用。

(c) 对 RHEL 不可用。

(d) 仅在 Windows 堆栈上可用。主要版本为 12.2。当前次要版本为 12.22。

(e) 弃用已完成；支持已结束。

安装位置取决于 Chef 版本：

- 应用程序对所有的 Chef 版本使用 `/usr/local/bin/ruby` 可执行文件。
- 对于 Chef 0.9 和 11.4，实例代理和 Chef 配方使用 `/usr/bin/ruby` 可执行文件。
- 对于 Chef 11.10，实例代理和 Chef 配方使用 `/opt/aws/opsworks/local/bin/ruby` 可执行文件。

安装自定义说明书

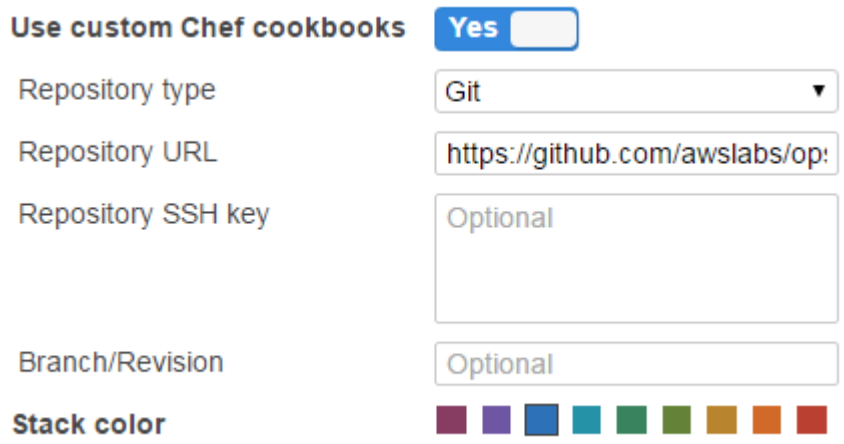
> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

要让堆栈安装和使用自定义说明书，您必须将堆栈配置为启用自定义说明书 (如果尚未配置)。然后，您必须提供存储库 URL 和任何相关信息，如密码。

> **Important**
>
> 将堆栈配置为支持自定义食谱后， AWS OpsWorks Stacks 会在启动时自动将您的食谱安装到所有新实例上。但是，您必须通过运行 “更新自定义 Cookbook AWS OpsWorks s 堆栈” 命令明确指示 Stacks 在任何现有实例上安装新的或更新的食谱。有关更多信息，请参阅 更新自定义说明书。在堆栈上启用 Use custom Chef cookbooks (使用自定义 Chef 说明书) 之前，请确保您运行的自定义和社区说明书支持堆栈所使用的 Chef 版本。

为自定义说明书配置堆栈

1. 在您的堆栈页面上，单击 Stack Settings 以显示其 Settings 页面。单击 Edit 以编辑设置。
2. 将 Use custom Chef cookbooks 切换为 Yes。



3. 配置您的自定义说明书。

完成后，单击 Save 以保存更新后的堆栈。

指定自定义说明书存储库

Linux 堆栈可从以下任意一种存储库类型中安装自定义说明书：

- HTTP 或 Amazon S3 存档。

  它们可以是公有或私有，但 Amazon S3 通常是私有存档的首选选项。
- Git 和 Subversion 存储库提供源代码控制，并且让您可以有多个版本。

Windows 堆栈可从 Amazon S3 存档和 Git 存储库中安装自定义说明书。

所有存储库类型均具有以下必填字段。

- 存储库类型：存储库的类型
- 存储库 URL：存储库的 URL

AWS OpsWorks Stacks 支持公共托管的 Git 存储库站点，例如GitHub或 Bitbucke t 以及私有托管的 Git 服务器。对于 Git 存储库，您必须使用以下一种 URL 格式，具体取决于存储库是公有还是私有。遵循与 Git 子模块相同的 URL 准则。

对于公有 Git 存储库，请使用 HTTPS 或 Git 只读协议：

- Git 只读：`git://github.com/amazonwebservices/opsworks-example-cookbooks.git`。
- HTTPS：`https://github.com/amazonwebservices/opsworks-example-cookbooks.git`。

对于私有 Git 存储库，您必须使用 SSH 读/写格式，如下例所示：

- Github 存储库：`git@github.com:`*`project`*`/`*`repository`*。
- Git 服务器上的存储库：*`user`*`@`*`server`*`:`*`project`*`/`*`repository`*

剩余的设置会因存储库类型而异，以下各节将介绍这些设置。

HTTP 存档

为 Repository type 选择 Http Archive 时将显示两个额外的设置，如果该存档受密码保护，则您必须完成这两个设置。

- 用户名：您的用户名
- Password - 您的密码

Amazon S3 归档

为存储库类型选择 S3 存档会显示以下其他可选设置。 AWS OpsWorks 无论您使用堆栈 API 还是控制台，堆栈都可以使用 Amazon EC2 角色（主机操作系统管理员身份验证）访问您的存储库。 AWS OpsWorks

- 访问密钥 ID - AWS 访问密钥 ID，例如 AKIAIOSFODNN7EXAMPLE。
- 秘密访问密钥 — 相应的 AWS 秘密访问密钥，例如 wjalr bPxRfi xutnfemi/k7mdeng/CYEXAMPLEKEY。

Git 存储库

选择 Source Control 下的 Git 会显示以下额外可选设置：

存储库 SSH 密钥

您必须指定一个部署 SSH 密钥来访问私有 Git 存储库。对于 Git 子模块，指定的密钥必须有权访问这些子模块。有关更多信息，请参阅 使用 Git 存储库 SSH 密钥。

> **Important**
>
> 部署 SSH 密钥不需要密码； AWS OpsWorks Stacks 无法通过密码。

分支/修订

如果存储库有多个分支， AWS OpsWorks Stacks 会默认下载主分支。要指定特定分支，请输入分支名称、SHA1 哈希或标签名称。要指定特定提交，请输入完整的 40 位十六进制提交 ID。

Subversion 存储库

选择 Source Control 下的 Subversion 会显示以下额外可选设置：

- 用户名：您的针对私有存储库的用户名称。
- 密码：您的针对私有存储库的密码。
- 修订：[可选] 修订名称（如果您有多个修订）。

  要指定分支或标记，您必须修改存储库 URL，例如：**`http://repository_domain/repos/myapp/branches/my-apps-branch`** 或 **`http://repository_domain_name/repos/calc/myapp/my-apps-tag`**。

更新自定义说明书

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

当你为 AWS OpsWorks Stacks 提供自定义食谱时，内置的安装配方会在每个新启动的实例上创建一个本地缓存，并将食谱下载到缓存中。 AWS OpsWorks 然后，Stacks 从缓存中运行配方，而不是存储库。如果您修改存储库中的自定义食谱，则必须确保已将更新的食谱安装在实例的本地缓存中。AWS OpsWorks Stacks 在启动时会自动将最新的食谱部署到新实例。然而，对于现有实例，情况则有所不同：

- 您必须手动将更新后的自定义说明书部署到在线实例。
- 您不必将更新后的自定义说明书部署到由实例存储支持的离线实例，包括基于负载和基于时间的实例。

  AWS OpsWorks 当实例重新启动时，堆栈会自动部署当前的食谱。
- 您必须启动不是基于负载也不是基于时间的 EBS 支持的离线全天候实例。
- 您不能启动 EBS 支持的基于负载和基于时间的离线实例，因此，最简单的方法是删除离线实例并添加新实例来替换它们。

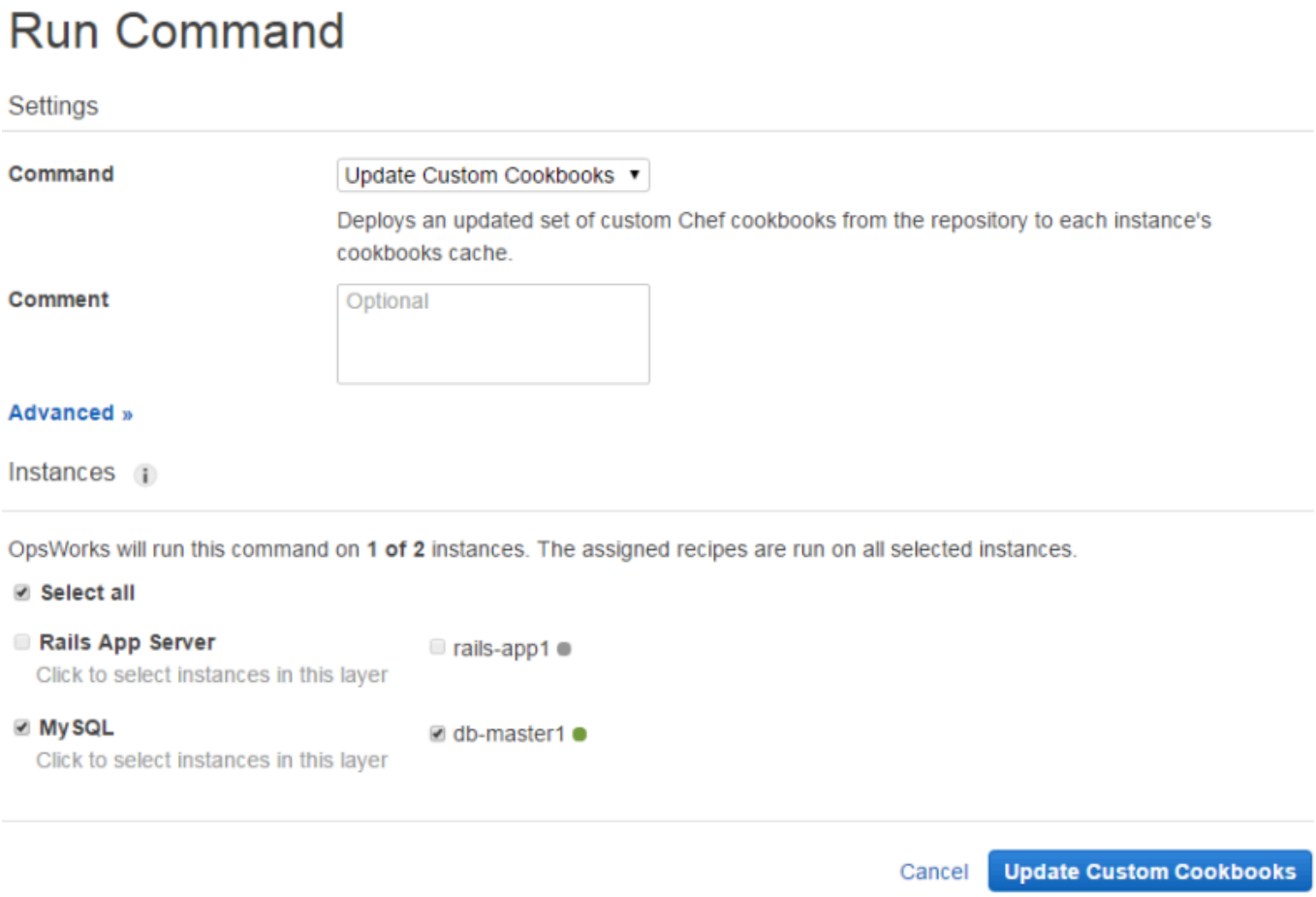
  由于它们现在是新实例，因此 AWS OpsWorks Stacks 会在实例启动时自动部署当前的自定义食谱。

手动更新自定义说明书

1. 使用修改后的食谱更新您的存储库。 AWS OpsWorks Stacks 使用你最初安装食谱时提供的缓存 URL，因此食谱根文件名、存储库位置和访问权限不应更改。

   - 对于 Amazon S3 或 HTTP 存储库，用新的同名 .zip 文件替换原始的 .zip 文件。
   - 对于 Git 或 Subversion 存储库，编辑您的堆栈设置以将 Branch/Revision 字段改为新版本。

2. 在堆栈的页面上，单击 Run Command，然后选择 Update Custom Cookbooks 命令。



3. 根据需要添加注释。
4. （可选）为命令指定自定义 JSON 对象，以向 Stack AWS OpsWorks s 在实例上安装的堆栈配置和部署属性添加自定义属性。有关更多信息，请参阅 使用自定义 JSON 和 覆盖属性。
5. 默认情况下， AWS OpsWorks Stacks 会在每个实例上更新食谱。要指定更新特定的实例，可从位于页面底端的列表中选择相应的实例。要选择某个层中的全部实例，请选中左列中该层对应的复选框。

6. 单击 “更新自定义食谱” 以安装更新的食谱。 AWS OpsWorks Stacks 会删除指定实例上缓存的自定义食谱，并从存储库中安装新的食谱。

> **Note**
>
> 此过程只用于其缓存中包含旧版本说明书的现有实例。如果您随后向图层添加实例， AWS OpsWorks Stacks 会部署存储库中当前的食谱，以便它们自动获取最新版本。

执行配方

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

您可通过两种方式运行配方：

- 自动，通过将配方分配给相应层的生命周期事件来实现。
- 手动，通过运行“执行配方”堆栈命令或使用代理 CLI 来实现。

主题

AWS OpsWorks 堆栈生命周期事件

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

每个层有一组五个生命周期事件，每个事件有一组特定于层的关联配方。当某个层的实例上发生事件时， AWS OpsWorks Stacks 将自动运行一组相应的配方。要对这些事件提供自定义响应，请实现自定义配方并将其分配给每个图层的相应事件。 AWS OpsWorks Stacks 在活动内置食谱之后运行这些食谱。

Setup

此事件在已启动的实例完成引导后发生。您也可以使用安装堆栈命令手动触发Setup事件。 AWS OpsWorks Stacks 运行的配方可以根据实例的层设置实例。例如，如果实例是 Rails App Server 层的成员，则 Setup 配方将安装 Apache、Ruby Enterprise Edition、Passenger 和 Ruby on Rails。

> **Note**
>
> Setup 事件将导致实例中断服务。由于实例在 Setup 生命周期事件运行时未处于 Online 状态，因此将从负载均衡器中删除运行 Setup 事件的实例。

Configure

如果出现以下情况之一，此事件将在堆栈的所有实例上发生：

- 实例进入或退出联机状态。
- 您将弹性 IP 地址与实例关联或取消弹性 IP 地址与实例的关联。
- 您将 Elastic Load Balancing 负载均衡器附加到层，或将该负载均衡器与层分离。

例如，假设您的堆栈有实例 A、B 和 C，您启动了一个新实例 D。D 运行完设置配方后， AWS OpsWorks 堆栈会在 A、B、C 和 D 上触发Configure事件。如果您随后停止 A， AWS OpsWorks 堆栈会在 B、C 和 D 上触发Configure事件。 AWS OpsWorks 堆栈通过运行每个层的配方来响应Configure事件，这些Configure配方会更新实例的配置以反映当前的在线实例集。因此，Configure事件是重新生成配置文件的好时机。例如，HAProxy Configure 配方将重新配置负载均衡器以适应联机应用程序服务器实例组中的任何更改。

您还可使用“配置”堆栈命令手动触发配置事件。

Deploy

此事件在您运行 Deploy 命令时发生，通常用于将应用程序部署到一组应用程序服务器实例。这些实例将运行这样的配方：将应用程序和任何相关文件从其存储库部署到层的实例。例如，对于 Rails 应用程序服务器实例，Deploy 配方将签出指定 Ruby 应用程序并要求 Phusion Passenger 重新加载它。您还可在其他实例上运行 Deploy，使这些实例能够更新其配置以适应新部署的应用程序 (举例)。

> **Note**
>
> “设置”包含“部署”；它在设置完成后运行“部署”配方。

Undeploy

该事件在您删除应用程序或运行 Undeploy 命令以从一组应用程序服务器实例中删除应用程序时发生。指定的实例将运行配方来删除所有应用程序版本并执行所有必需的清理。

Shutdown

此事件发生在您指示 AWS OpsWorks Stacks 关闭实例之后，但在关联的 Amazon EC2 实例实际终止之前。 AWS OpsWorks Stacks 将运行配方来执行清理任务，如关闭服务。

如果您已将 Elastic Load Balancing 负载均衡器连接到该层并启用了对连接耗尽的支持，则 AWS OpsWorks Stacks 会等到连接耗尽完成后再触发事件。Shutdown

触发Shutdown事件后， AWS OpsWorks Stacks 允许Shutdown配方在指定的时间内执行任务，然后停止或终止 Amazon EC2 实例。默认 Shutdown 超时值为 120 秒。如果您的 Shutdown 配方需要更长时间，您可以编辑层配置以更改超时值。有关实例 Shutdown 的更多信息，请参阅停止实例。

> **Note**
>
> 重新启动实例不会触发任何生命周期事件。

有关 Deploy 和 Undeploy 应用程序命令的更多讨论，请参阅部署应用程序。

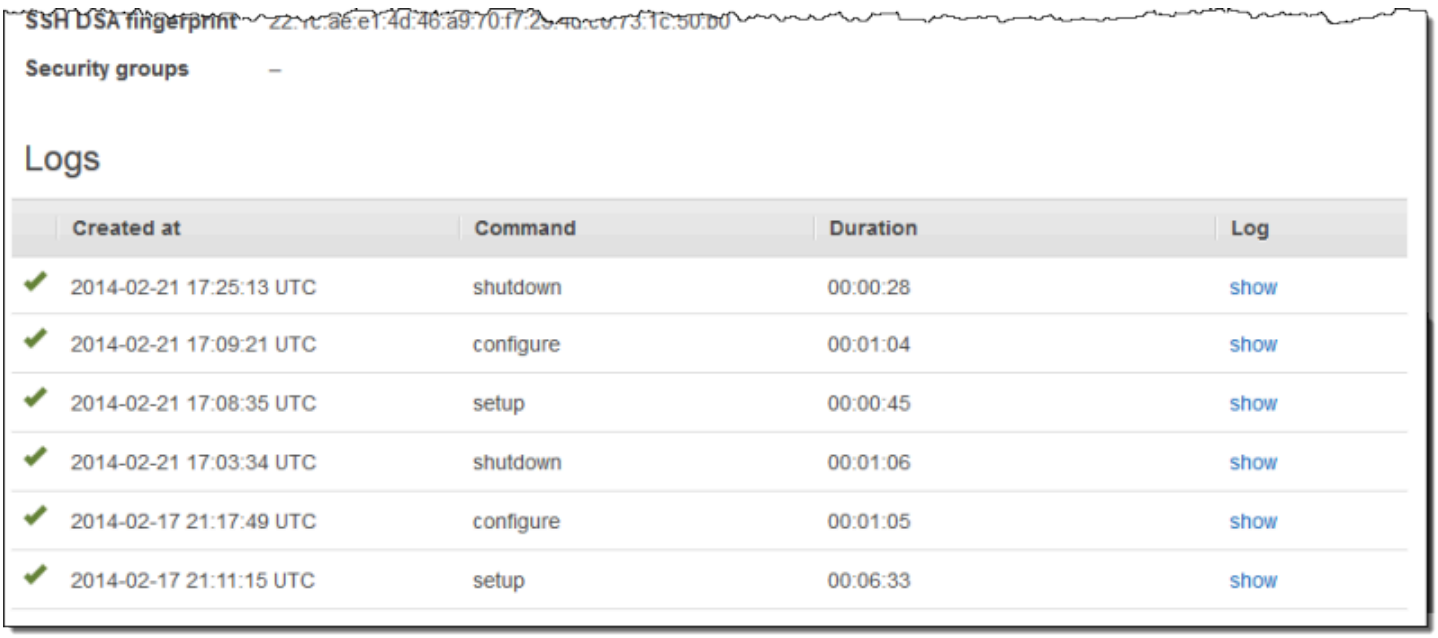
在已启动的实例完成引导后，剩余的启动序列如下所示：

1. AWS OpsWorks Stacks 运行实例的内置Setup配方，然后运行任何自定义Setup配方。
2. AWS OpsWorks Stacks 运行实例的内置Deploy配方，然后运行任何自定义Deploy配方。

   实例现在处于联机状态。
3. AWS OpsWorks Stacks 会在堆栈中的所有实例（包括新启动的实例）上触发一个Configure事件。

   AWS OpsWorks Stacks 运行实例的内置Configure配方，然后运行任何自定义配方。Configure

> **Note**
>
> 要查看特定实例上发生的生命周期事件，请转到 Instances 页并单击实例名称，打开其详细信息页面。事件列表位于该页面底部的 Logs 部分。您可单击日志列中的显示以检查 Chef 日志中是否有事件。它提供了有关如何处理事件的详细信息，包括运行了哪些配方。有关如何解释 Chef 日志的更多信息，请参阅 Chef 日志。



对于每个生命周期事件， AWS OpsWorks Stacks 会在每个实例上安装一组堆栈配置和部署属性，其中包含当前堆栈状态以及有关部署的信息（对于Deploy事件）。这些属性包含有关可用的实例、此类实例的 IP 地址等信息。有关更多信息，请参阅 堆栈配置和部署属性。

> **Note**
>
> 同时启动或停止大量实例可能迅速生成大量的 Configure 事件。为避免不必要的处理， AWS OpsWorks Stacks 仅响应最后一个事件。该事件的堆栈配置和部署属性包含针对一整套更改更新堆栈实例所需的一切信息。这样就不必同时处理早期Configure的事件。 AWS OpsWorks Stacks 将未处理Configure的事件标记为已取代。

自动运行配方

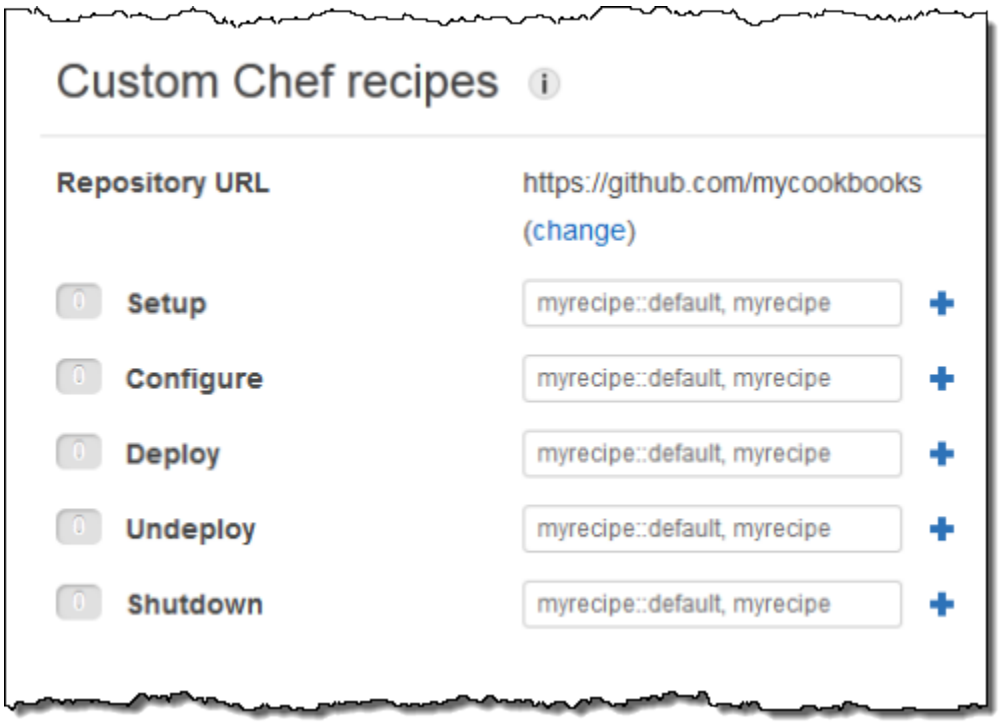
> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

每个层都有一组分配给每个生命周期事件的内置配方，但某些层缺少“取消部署”配方。当实例上发生生命周期事件时， AWS OpsWorks Stacks 会为关联层运行相应的配方集。

如果您安装了自定义食谱，则可以通过将每个食谱分配给图层的生命周期事件来让 AWS OpsWorks Stacks 自动运行部分或全部食谱。事件发生后， AWS OpsWorks Stacks 会在图层的内置配方之后运行指定的自定义配方。

将自定义配方分配给层事件

1. 在 Layers 页上，对于相应的层，单击 Recipes，然后单击 Edit。如果您尚未启用自定义说明书，请单击 configure cookbooks 以打开堆栈的 Settings 页。将 Use custom Chef Cookbooks 切换到 Yes，并提供说明书的存储库信息。然后，单击 Save 并导航回 Recipes 选项卡的编辑页面。有关更多信息，请参阅 安装自定义说明书。
2. 在 Recipes 选项卡上，在相应事件字段中输入各个自定义配方，然后单击 + 以将其添加到列表。按照下面所示指定配方：*cookbook*::*somerecipe* (忽略 `.rb` 扩展名)。



当您启动新实例时， AWS OpsWorks Stacks 会在运行标准配方后自动为每个事件运行自定义配方。

> **Note**
>
> 自定义配方将按您在控制台中输入它们的顺序执行。控制执行顺序的一个替代方法是实施按正确顺序执行配方的元配方。

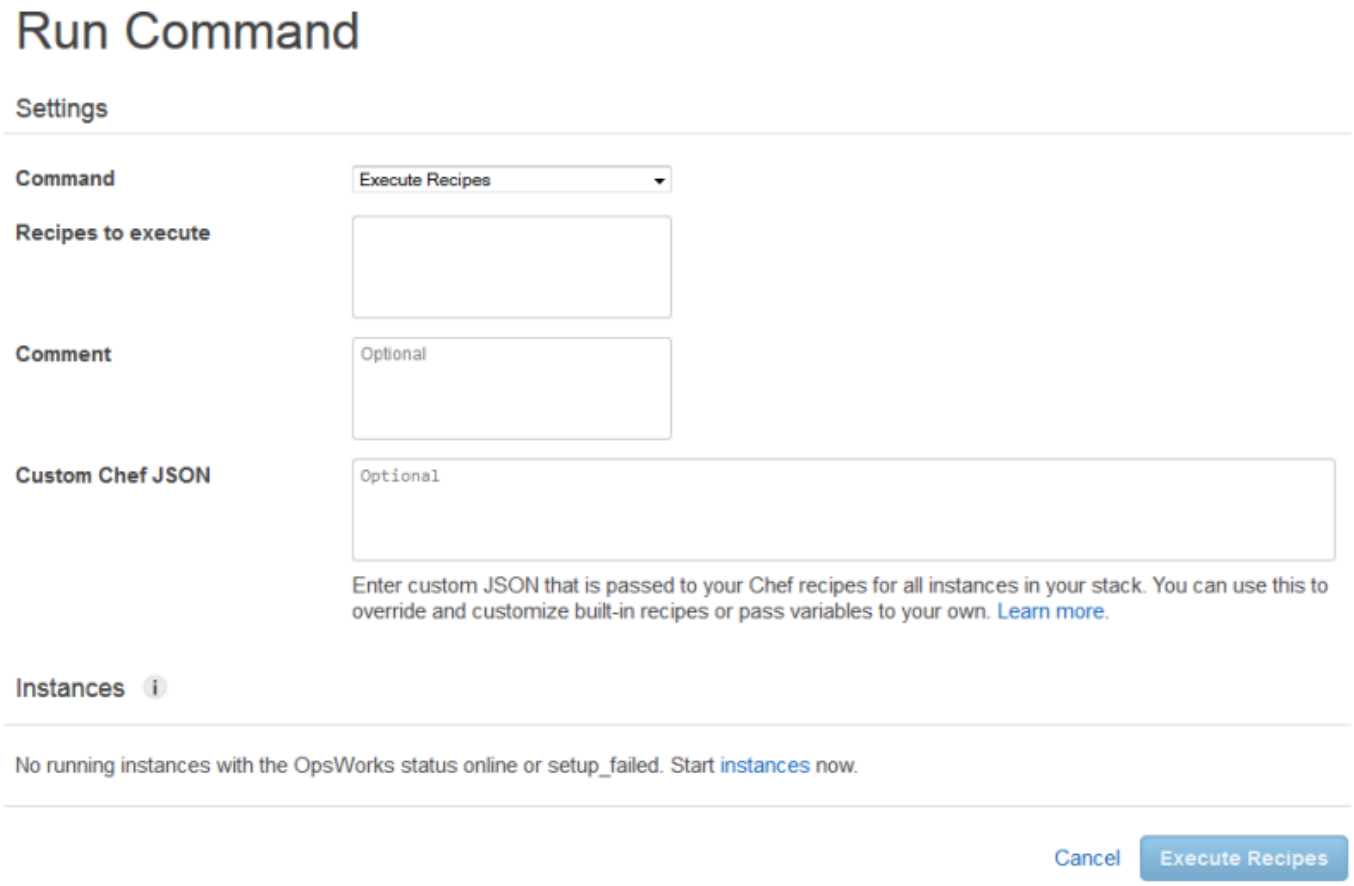
手动运行配方

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

尽管配方通常会自动运行以响应生命周期事件，但您可以随时在任一或所有堆栈实例上手动运行配方。此功能通常用于无法很好地映射到生命周期事件的任务，如备份实例。要手动运行自定义配方，该配方必须位于您的其中一个自定义说明书中，但它不必分配给生命周期事件。当你手动运行配方时， AWS OpsWorks Stacks 会安装与 Deploy 事件相同的`deploy`属性。

在堆栈实例上手动运行配方

1. 在 Stack 页面上，单击 Run command。对于 Command，选择 Execute Recipes。



2. 使用标准 cookbookname::*recipename* 格式在######框中输入要运行的配方。使用逗号来分隔多个配方；这些配方将按您列出它们的顺序运行。
3. (可选) 使用 Custom Chef JSON 框添加自定义 JSON 对象，用于自定义属性，这些属性将合并到安装在实例上的堆栈配置和部署属性。有关使用自定义 JSON 对象的更多信息，请参阅使用自定义 JSON 和覆盖属性。
4. 在 “实例” 下，选择 AWS OpsWorks 堆栈应在其上运行配方的实例。

当生命周期事件发生时， AWS OpsWorks Stacks 代理会收到一条命令来运行相关配方。您可在特定实例上手动运行这些命令，方法为使用合适的堆栈命令或使用代理 CLI 的 run_command 命令。有关如何使用代理 CLI 的更多信息，请参阅 AWS OpsWorks Stacks Agent CLI。

资源管理

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

资源页面允许您在 AWS OpsWorks 堆栈堆栈中使用账户的弹性 IP 地址、Amazon EBS 卷或 Amazon RDS 实例资源。您可使用 Resources 执行以下操作：

- 注册资源 - 将资源注册到一个堆栈，从而让您可以将资源挂载到该堆栈的一个实例。
- 挂载资源 - 将资源挂载到堆栈的一个实例。
- 移动资源 - 将资源从一个实例移动到另一个实例。
- 分离资源 - 将资源与一个实例分离。资源将保持注册状态，并且可以挂载到另一个实例。
- 注销资源。未注册的资源不能被 AWS OpsWorks Stacks 使用，但除非您将其删除，否则它会保留在您的账户中，并且可以在其他堆栈中注册。

请注意以下约束：

- 您无法将已注册的 Amazon EBS 卷挂载到 Windows 实例。
- Resources 页面管理标准、PIOPS、吞吐量优化型 HDD、冷 HDD 或通用型 (SSD) Amazon EBS 卷，但不管理 RAID 阵列。
- Amazon EBS 卷必须为 xfs 格式。

  AWS OpsWorks 堆栈不支持其他文件格式，例如 ext4。有关准备 Amazon EBS 卷的更多信息，请参阅使 Amazon EBS 卷可用。
- 您不能将 Amazon EBS 卷挂载到正在运行的实例或将它从正在运行的实例中分离。

  您只能对脱机实例进行操作。例如，您可以将使用中的卷注册到堆栈并将它挂载到一个脱机实例，但在启动新实例前，您必须停止原来的实例并分离卷。否则，启动过程将失败。
- 所有注册的资源都只能在中进行管理 AWS OpsWorks。这可以覆盖资源生命周期属性，例如，对 EC2 卷使用 `DeleteOnTermination`。
- 您可以将弹性 IP 地址挂载到正在运行的实例或将它从正在运行的实例中分离。

您可以对联机或脱机实例进行操作。例如，您可以注册一个正在使用的地址并将其分配给正在运行的实例， AWS OpsWorks Stacks 将自动重新分配该地址。

- 要注册和注销资源，您的 IAM 策略必须为以下操作授予权限：

Amazon EBS 卷	弹性 IP 地址	Amazon RDS 实例
RegisterVolume	RegisterElasticIp	RegisterRdsDbInstance
UpdateVolume	UpdateElasticIp	UpdateRdsDbInstance
DeregisterVolume	DeregisterElasticIp	DeregisterRdsDbInstance

管理权限级别为所有这些操作授予权限。为了防止管理用户注册或注销特定资源，请对其 IAM 策略进行编辑以拒绝相应操作的权限。有关更多信息，请参阅 安全性和权限。

主题

将资源注册到堆栈

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

Amazon EBS 卷或 Elastic IP 地址必须先注册到某个堆栈，然后才能挂载到实例上。当 AWS OpsWorks 堆栈为堆栈创建资源时，它们会自动注册到该堆栈。如果要使用外部创建的资源，则必须显式注册它们。请注意以下几点：

- 您一次仅可将资源注册到一个堆栈。

- 当您删除堆栈时，AWS OpsWorks Stacks 会取消注册所有资源。

主题

将 Amazon EBS 卷注册到 Stack

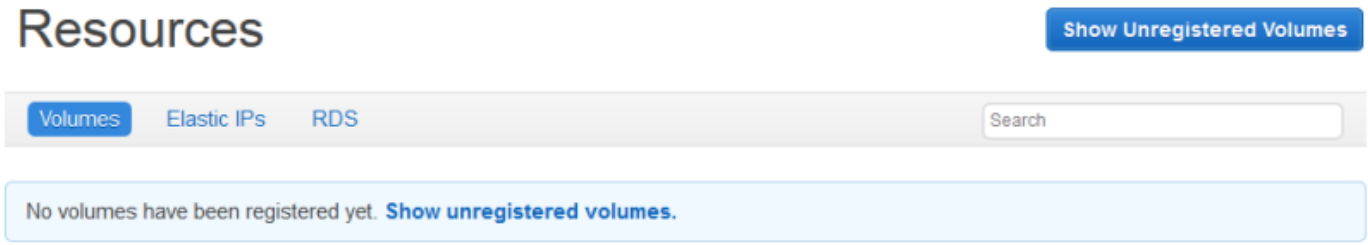
> **Note**
>
> 此资源仅可用于 Linux 堆栈。虽然您可以将 Amazon EBS 卷注册到 Windows 堆栈，但无法将它挂载到实例。

您可以使用 Resources 页面来将 Amazon EBS 卷注册到堆栈，但要受到以下约束：

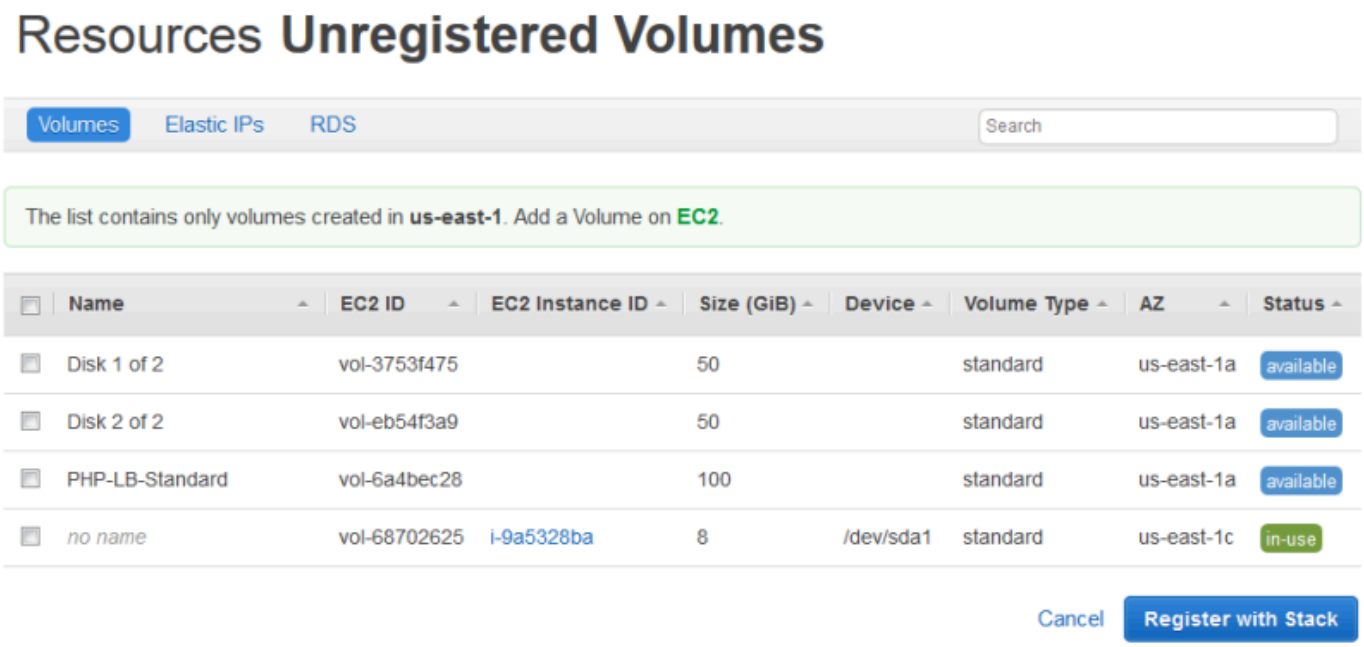
- 挂载的非根 Amazon EBS 卷必须是标准、吞吐优化型 HDD、冷 HDD、PIOPS 或通用型 (SSD)，而不是 RAID 阵列。有关最大和最小卷大小的更多信息，请参阅本指南中的EBS 卷。
- 卷必须为 XFS 格式。
- 对于非根 Amazon EBS 卷，AWS OpsWorks Stacks 不支持其他文件格式（例如第四个扩展文件系统 (ext4)）。有关准备 Amazon EBS 卷的更多信息，请参阅使 Amazon EBS 卷可用。请注意，该主题中的示例介绍的是如何创建基于 ext4 的卷，但您可以对基于 XFS 的卷使用相同的步骤。

注册 Amazon EBS 卷

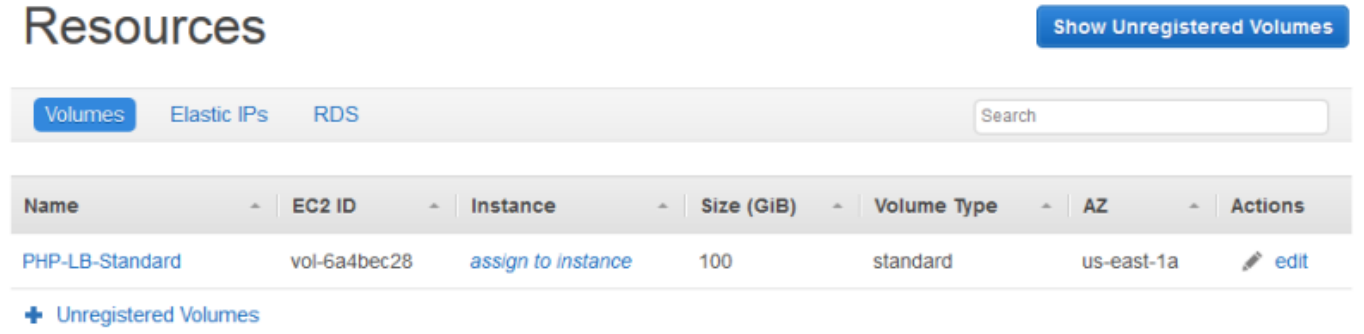
1. 打开所需的堆栈并在导航窗格中单击 Resources。
2. 单击 Volumes 以显示可用的 Amazon EBS 卷。最初，堆栈没有注册的卷，如下图中所示。



3. 单击 Show Unregistered Volumes (显示未注册的卷)以显示您账户中位于堆栈的区域以及堆栈的 VPC（如果可用）中的 Amazon EBS 卷。Status 列指示卷是否可用。Volume Type 指示该卷是标准卷 (`standard`)、通用型 SSD (`gp2`)、PIOPS (`io1`，后根用括号括起来的每卷 IOPS 值)、吞吐优化型 HDD (`st1`) 还是冷 HDD (`sc1`)。



4. 选择相应的卷并单击 Register to Stack。现在，Resources 页面将列出新注册的卷。



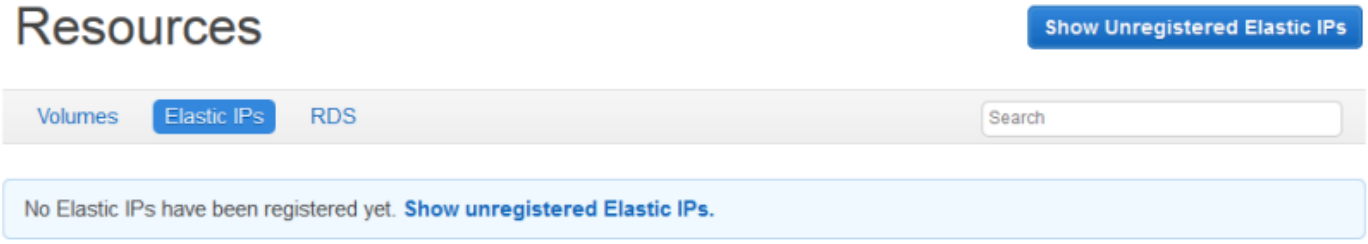
要注册其他卷，请单击 Show Unregistered Volumes 或 + Unregistered Volumes 并重复此过程。

将弹性 IP 地址注册到堆栈

使用以下过程注册弹性 IP 地址。

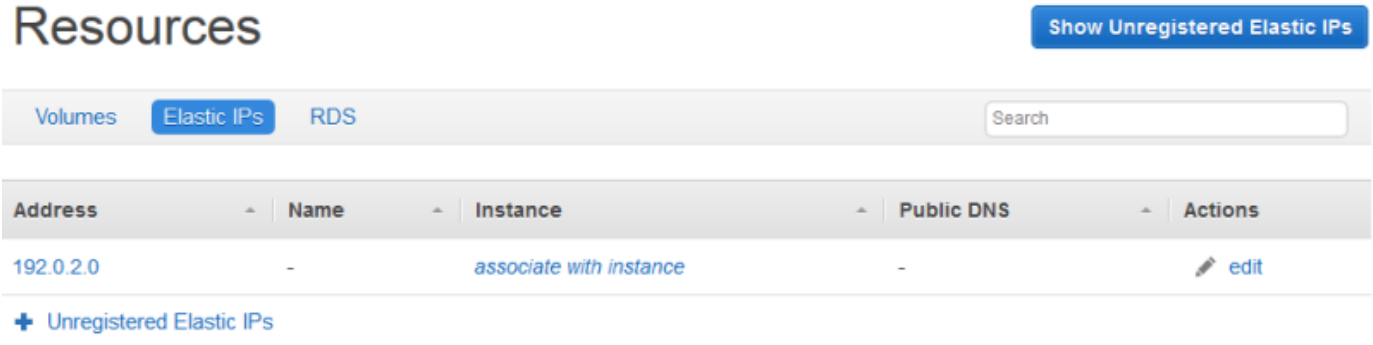
注册弹性 IP 地址

1. 打开堆栈的 Resources 页面，然后单击 Elastic IPs 以显示可用的弹性 IP 地址。最初，堆栈没有注册的地址，如下图中所示。



2. 单击 Show Unregistered Elastic IPs 以在您的账户中显示位于堆栈的区域中的可用弹性 IP 地址。

3. 选择相应的地址并单击 Register to Stack。这将返回到 Resources 页面，该页面现在列出了新注册的地址。



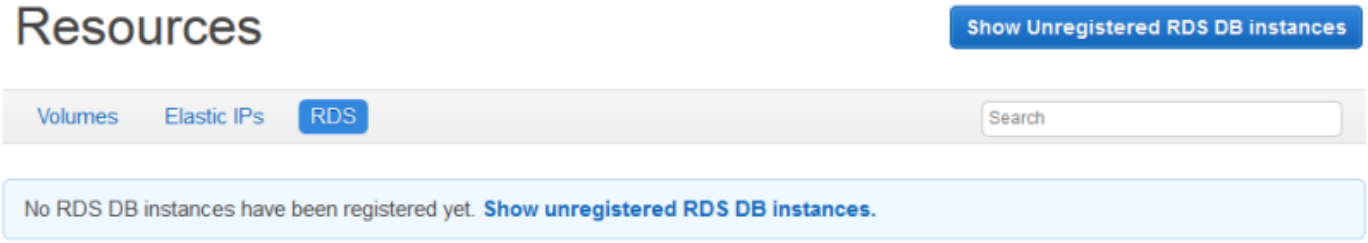
要注册其他地址，请单击 Show Unregistered Elastic IPs 或 + Unregistered Elastic IPs 并重复此过程。

将 Amazon RDS 实例注册到 Stack

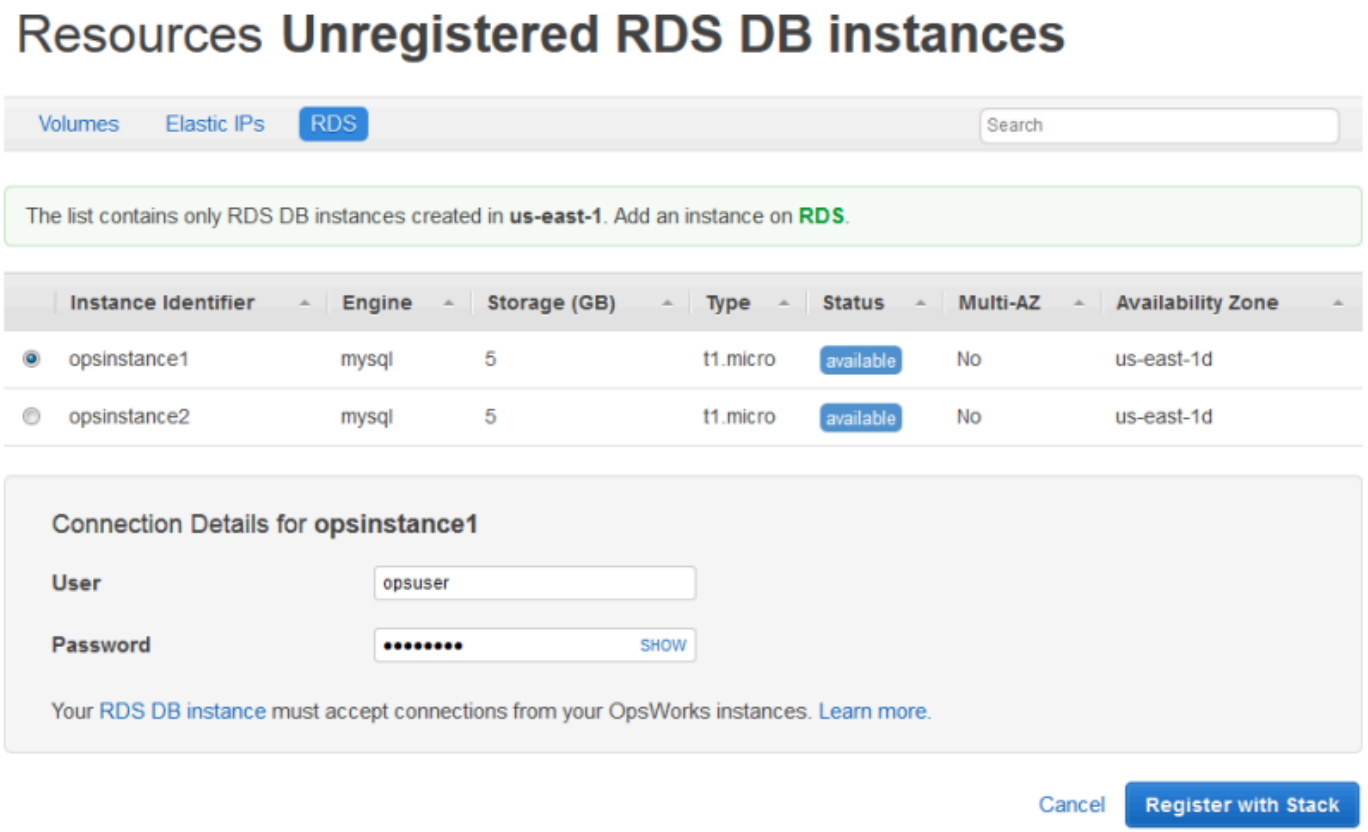
使用以下过程注册 Amazon RDS 实例。

注册 Amazon RDS 实例

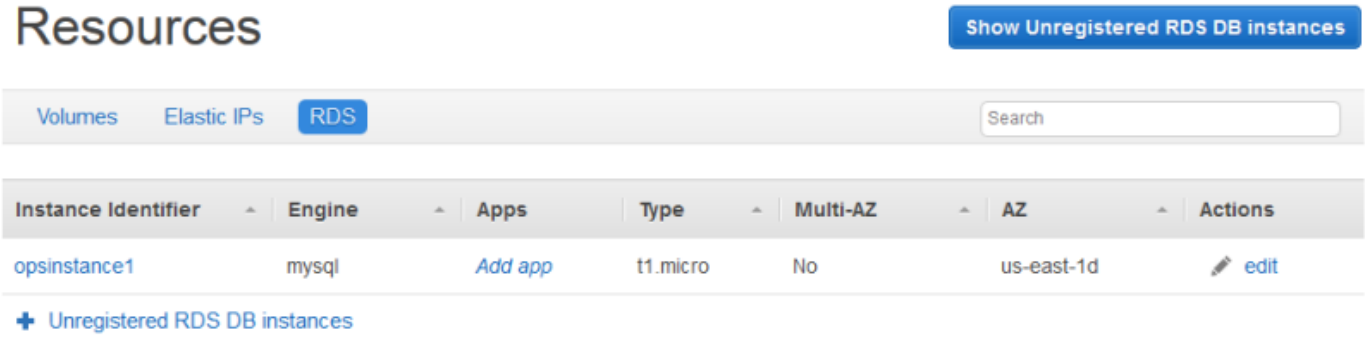
1. 打开堆栈的 Resources 页面，然后单击 RDS 以显示可用的 Amazon RDS 实例。最初，堆栈没有注册的实例，如下图中所示。



2. 单击 显示未注册的 RDS 数据库实例 以在您的账户中显示位于堆栈的区域中的可用 Amazon RDS 实例。



3. 选择合适的实例，为 User 和 Password 输入该实例的主用户和主密码值，然后单击 Register to Stack。这将返回到 Resources 页面，该页面现在列出了新注册的实例。



> **Important**
>
> 您必须确保您用于注册 Amazon RDS 实例的用户和密码与有效的用户和密码对应。如果它们不对应，您的应用程序将无法连接到该实例。

要注册其他地址，请单击 Show Unregistered RDS DB instances 或 + Unregistered RDS DB instances 并重复此过程。有关如何使用带有 AWS OpsWorks 堆栈的 Amazon RDS 实例的更多信息，请参阅Amazon RDS 服务层。

> **Note**
>
> 您也可以通过 Layers 页面注册 Amazon RDS 实例。有关更多信息，请参阅 Amazon RDS 服务层。

挂载和移动资源

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

将资源注册到某个堆栈后，您可以将它挂载到该堆栈的一个实例。您也可以将已挂载的资源从一个实例移动到另一个实例。请注意以下几点：

- 在挂载或移动 Amazon EBS 卷时，操作中涉及的实例必须是脱机的。如果您感兴趣的实例不在 Resources 页面上，请转到 Instances 页面并停止实例。在实例停止后，您可以返回到 Resources 页面并挂载或移动资源。
- 当您挂载或移动弹性 IP 地址时，实例既可以是联机的，也可以是脱机的。
- 如果您删除某个实例，任何已挂载的资源将保留在堆栈上的注册状态。然后，您可以将该资源挂载到另一个实例，或者如果您不再需要该资源，可注销该资源。

主题

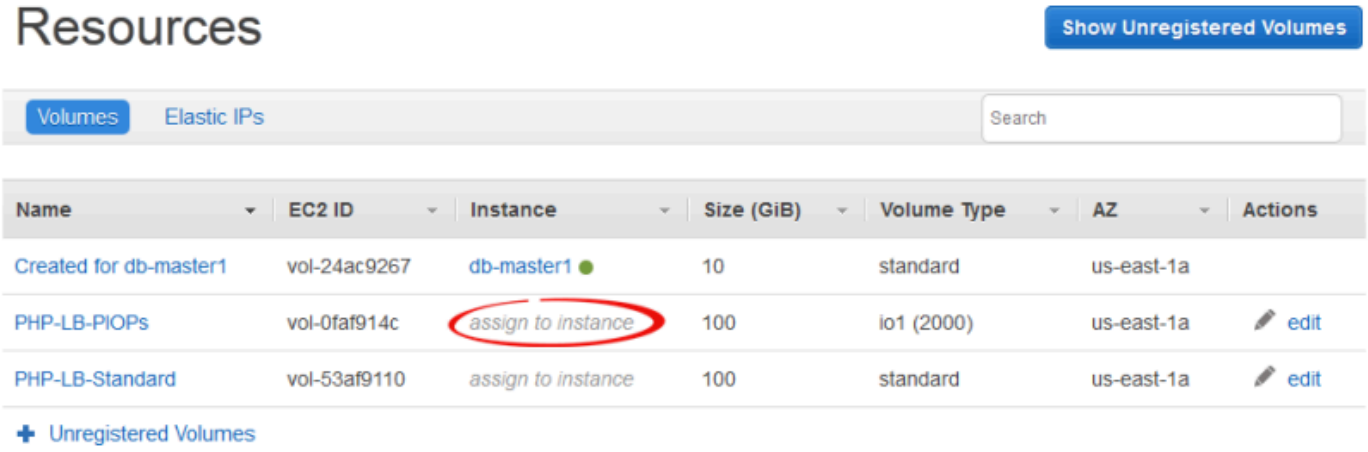
将 Amazon EBS 卷分配给实例

> **Note**
>
> 您不能将 Amazon EBS 卷分配给 Windows 实例

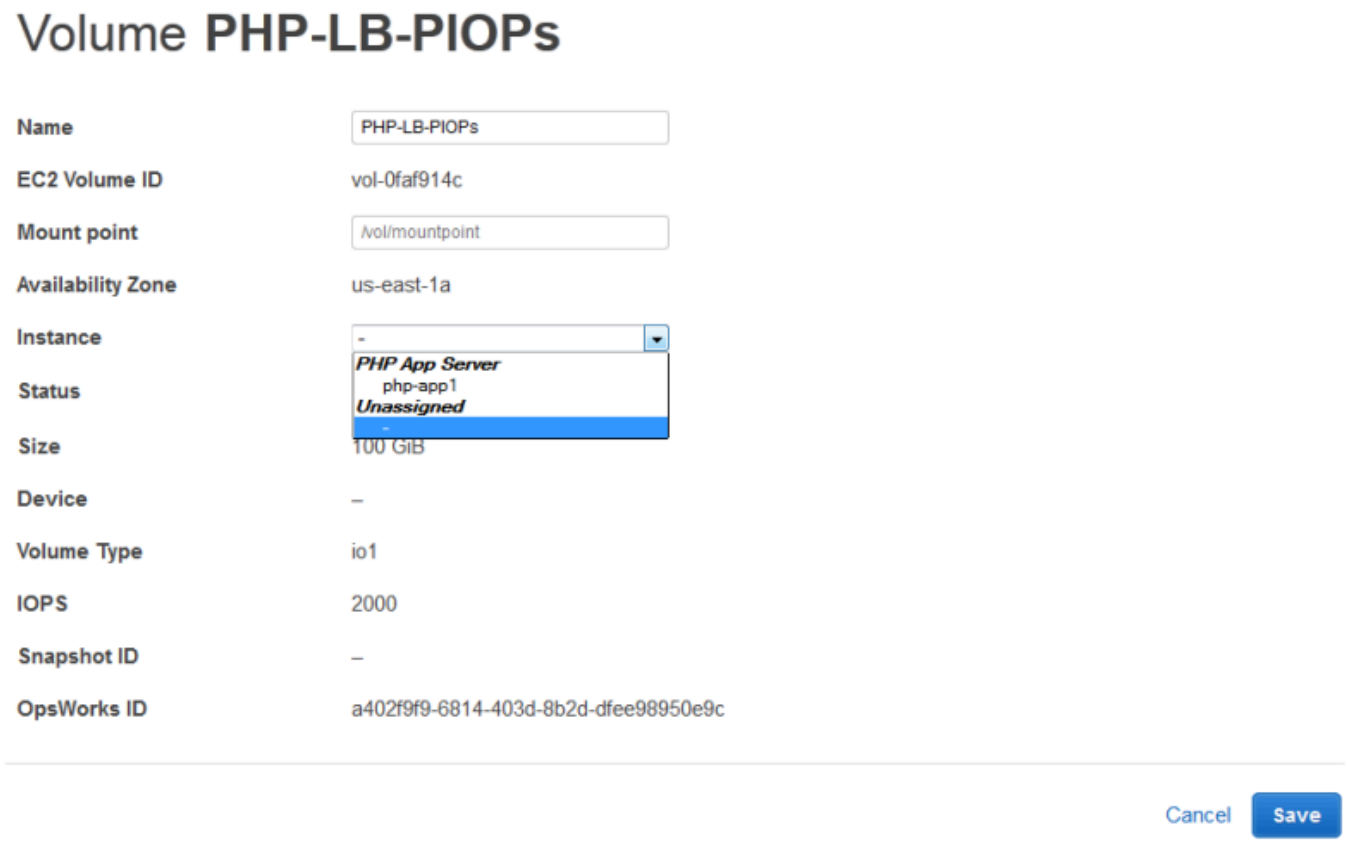
您可以将已注册的 Amazon EBS 卷挂载到实例并将它从一个实例移动到另一个实例，但这两个实例都必须是脱机的。

将 Amazon EBS 卷分配给实例

1. 在“Resources”页面上，单击相应卷的 Instance 列中的 assign to instance。



2. 在该卷的详细信息页面，选择相应实例，指定卷的名称和挂载点，然后单击 Save 以将该卷挂载到该实例。



> **Important**
>
> 如果您将外部的使用中的卷分配给您的实例，则必须使用 Amazon EC2 控制台、API 或 CLI 将它从原来的实例中取消分配，否则启动过程将失败。

您还可以使用详细信息页面将分配的 Amazon EBS 卷移动到堆栈中的另一个实例。

将 Amazon EBS 卷移动到另一个实例

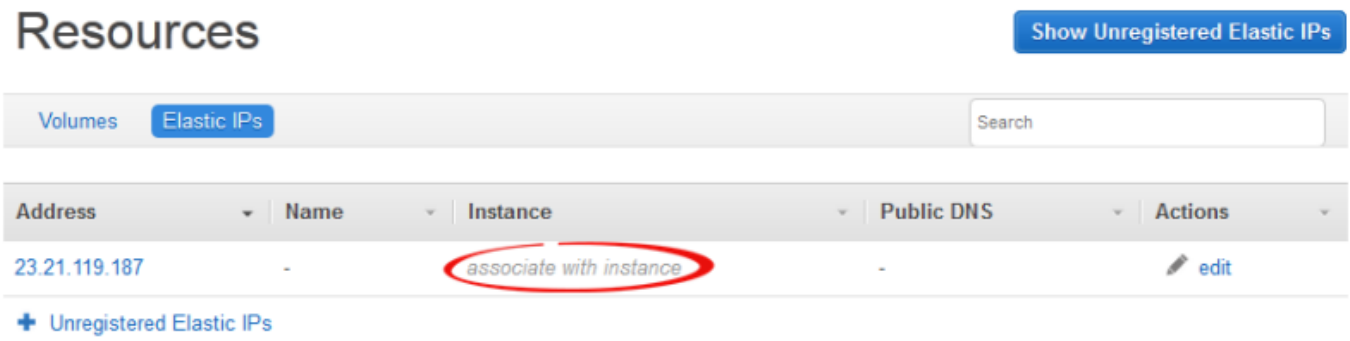
1. 确保两个实例均处于脱机状态。
2. 在 Resources 页面上，单击 Volumes，然后单击卷的 Actions 列中的 edit。
3. 请执行以下操作之一：
   - 要将卷移动到堆栈中的另一个实例，请从 Instance 列表中选择相应的实例并单击 Save。
   - 要将卷移动到另一个堆栈中的实例，请注销卷、向新堆栈注册卷，然后向新实例挂载卷。

关联弹性 IP 地址和实例

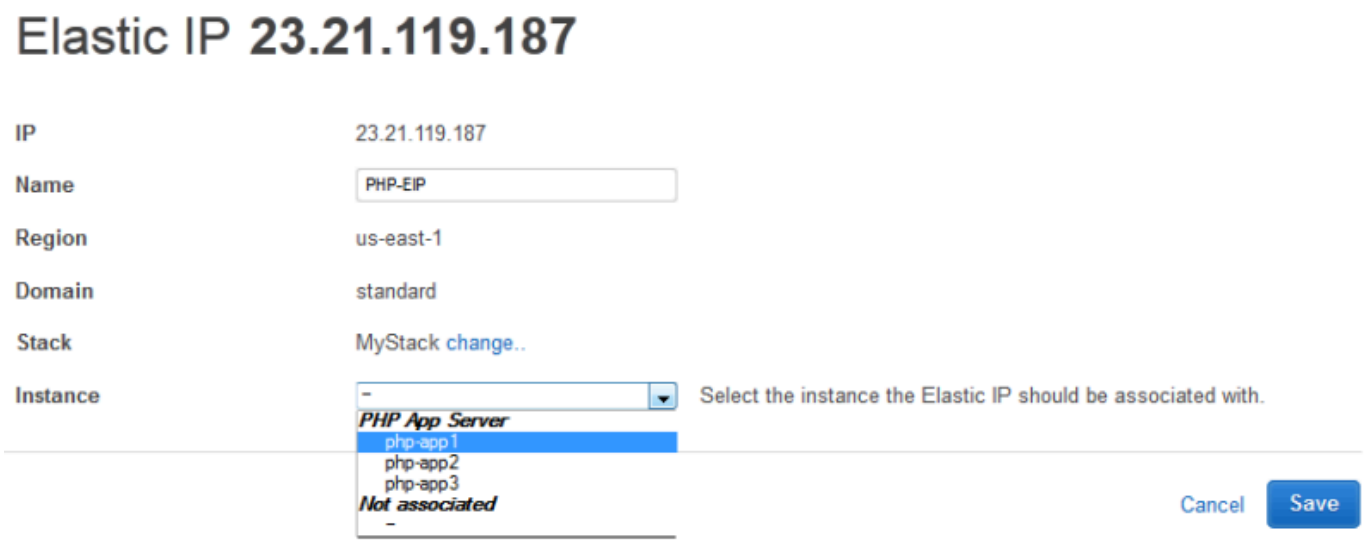
您可以将已注册的弹性 IP 地址与实例关联，并将它从一个实例移动到另一个实例，包括其他堆栈中的实例。这些实例既可以是联机的，也可以是脱机的。

要关联弹性 IP 地址和实例

1. 在 Resources 页面上，单击相应地址的 Instance 列中的 associate with instance。



2. 在地址的详细信息页面上，选择适当的实例，指定地址的名称，然后单击 Save 以将该地址与实例关联。



> **Note**
>
> 如果弹性 IP 地址当前与另一个在线实例关联， AWS OpsWorks Stacks 会自动将该地址重新分配给新实例。

您也可以使用详细信息页面将关联的弹性 IP 地址移动到另一个实例。

将弹性 IP 地址移动到另一个实例

1. 在 Resources 页面上，单击 Elastic IPs，然后单击地址的 Actions 列中的 edit。
2. 请执行以下操作之一：
   - 要将地址移动到堆栈中的另一个实例，请从 Instance 列表中选择相应的实例并单击 Save。
   - 要将地址移动到另一个堆栈中的实例，请单击 Stack 设置中的 change，以查看可用堆栈的列表。从 Stack 列表中选择一个堆栈，并从 Instance 列表中选择一个实例。然后单击 Save (保存)。

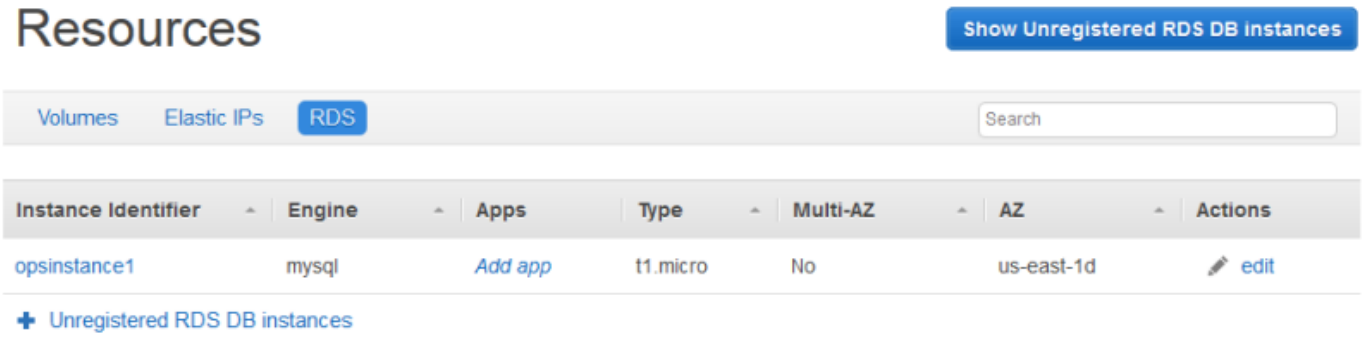
附加或移动地址后，AWS OpsWorks Stacks 会触发配置生命周期事件，将更改通知堆栈的实例。

将 Amazon RDS 实例挂载到应用程序

您可以将 Amazon RDS 实例挂载到一个或多个应用程序。

将 Amazon RDS 实例挂载到应用程序

1. 在 Resources 页面上，单击相应实例的 Apps 列中的 Add app。



2. 使用 添加应用程序 页面挂载 Amazon RDS 实例。有关更多信息，请参阅 添加应用程序。

由于 Amazon RDS 可挂载到多个应用程序中，因此并没有一个专门的过程来将实例从一个应用程序移动到另一个应用程序。只需编辑第一个应用程序以删除 RDS 实例或编辑第二个应用程序以添加 RDS 实例即可。有关更多信息，请参阅 编辑应用程序。

分离资源

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

如果您不再需要某个挂载的资源，则可以将其分离。此资源将保持在堆栈中的注册状态，并且可以挂载到任何其他位置。

主题

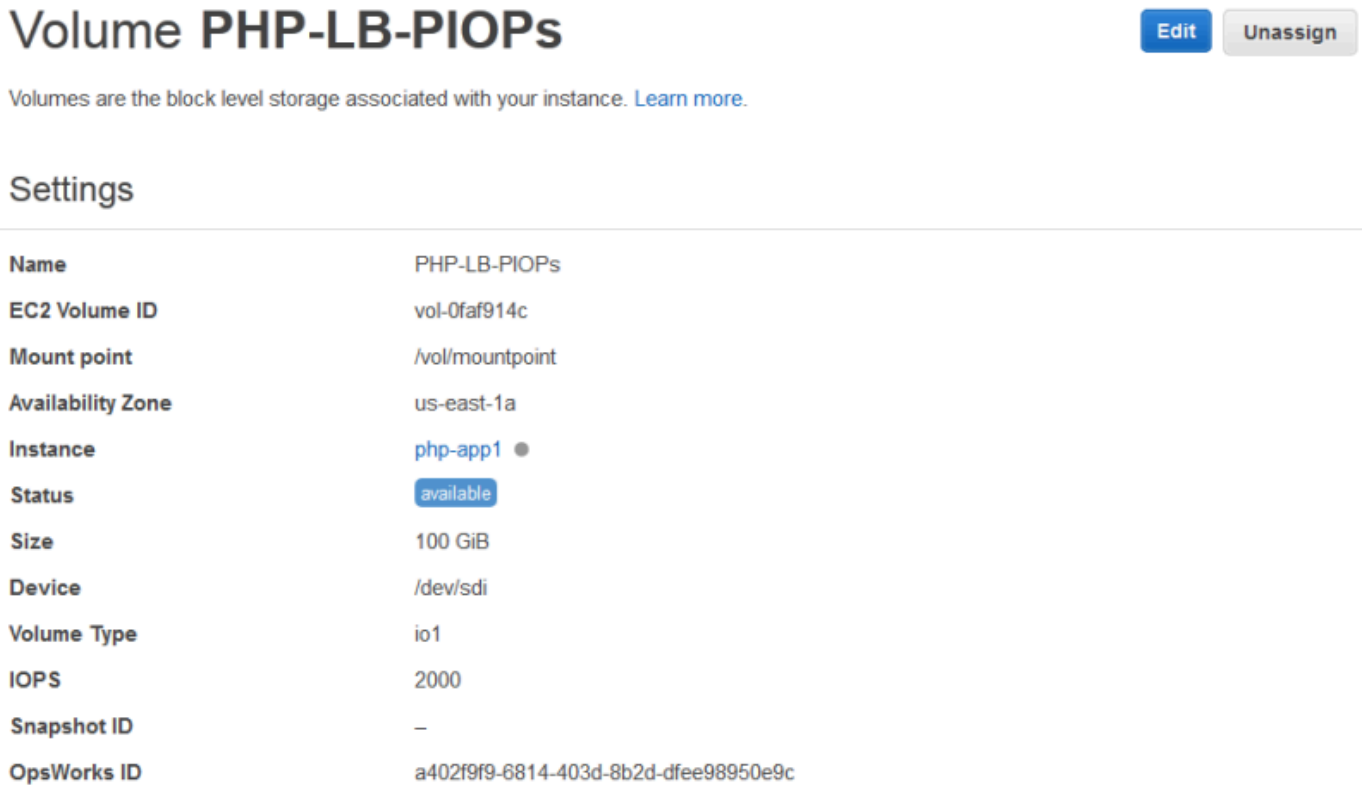
- 取消分配 Amazon EBS 卷
- 取消关联弹性 IP 地址
- 分离 Amazon RDS 实例

取消分配 Amazon EBS 卷

使用以下过程可从实例中取消分配 Amazon EBS 卷。

取消分配 Amazon EBS 卷

1. 确认实例处于脱机状态。
2. 在 Resources 页面上，单击 Volumes，然后单击卷名称。
3. 在卷的详细信息页面上，单击 Unassign。

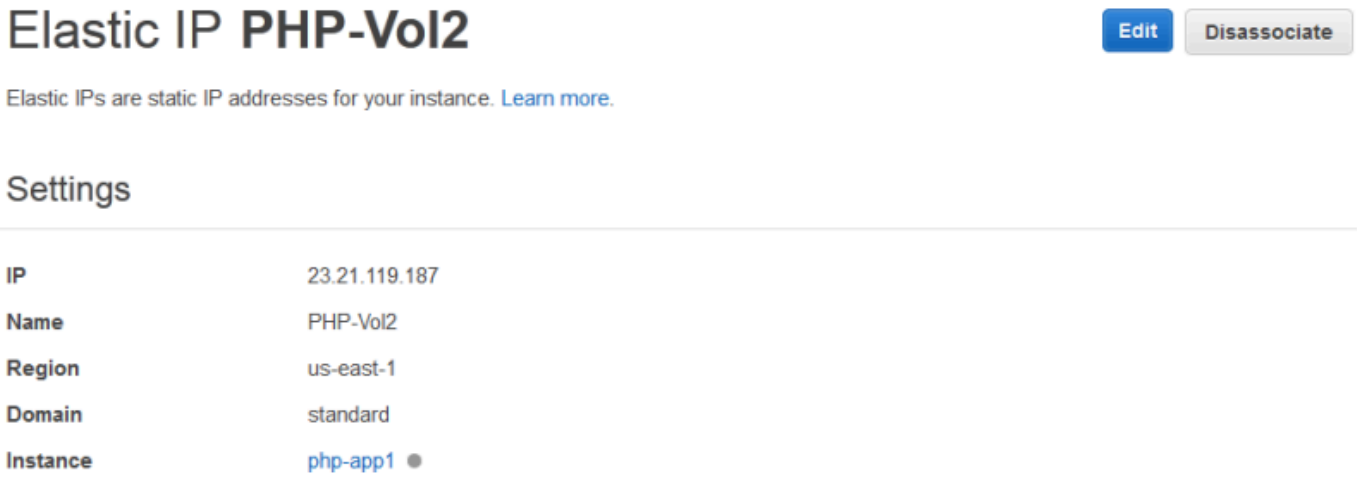


取消关联弹性 IP 地址

使用以下过程可从实例中取消关联弹性 IP 地址。

撤销弹性 IP 地址的关联

1. 在 Resources 页面上，单击 Elastic IPs，然后单击地址的 Actions 列中的 edit。
2. 在地址的详细信息页面上，单击 Disassociate。



取消关联地址后，AWS OpsWorks Stacks 会触发配置生命周期事件，将更改通知堆栈的实例。

分离 Amazon RDS 实例

使用以下过程可从应用程序中分离 Amazon RDS。

分离 Amazon RDS 实例

1. 在 Resources 页面上，单击 RDS，然后单击 Apps 列中的适当的应用程序。
2. 单击 Edit 并编辑应用程序配置以分离实例。有关更多信息，请参阅 编辑应用程序。

> **Note**
>
> 此过程会将 Amazon RDS 从单个应用程序中分离。如果将实例挂载到多个应用程序，则必须对每个应用程序重复此过程。

注销资源

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

如果您不再需要将某个资源注册到堆栈，则可以注销该资源。取消注册不会将资源从您的账户中删除；它会保留在那里，可以在其他堆栈中注册或在堆栈之外 AWS OpsWorks 使用。如果要完全删除资源，您有两个选择：

- 如果某个 Elastic IP 或 Amazon EBS 资源已挂载到某个实例，则可以在删除该实例时删除该资源。

  转至 Instances 页面，单击实例的 Actions 列中的 delete，然后选择 Delete instance's EBS volumes 或 Delete the instance's Elastic IP。
- 注销该资源，然后使用 Amazon EC2; 或 Amazon RDS 控制台、API 或 CLI 来删除它。

主题

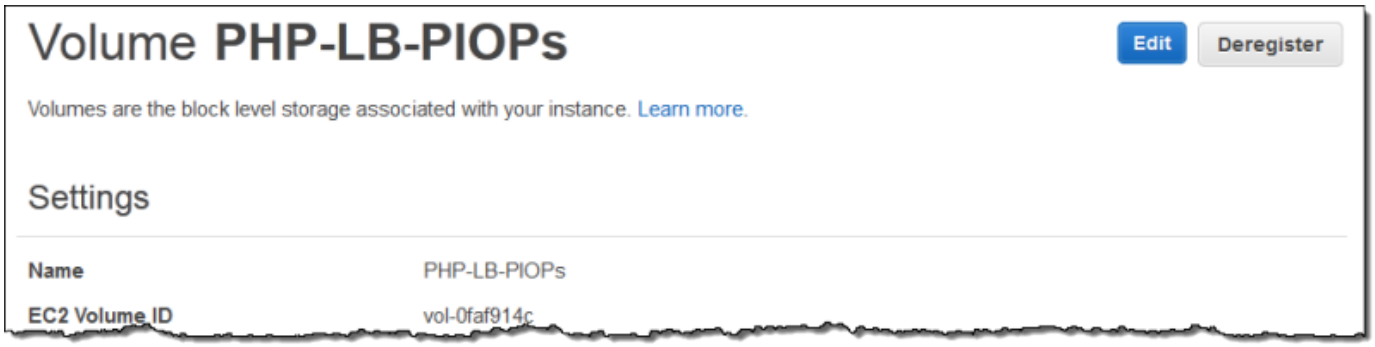
- 注销Amazon EBS 卷
- 注销弹性 IP 地址
- 注销Amazon RDS 实例

注销Amazon EBS 卷

使用以下过程可注销 Amazon EBS 卷。

取消注册 Amazon EBS 卷

1. 如果卷已挂载到某个实例，则取消分配该卷，如取消分配 Amazon EBS 卷中所述。
2. 在 Resources 页面上，单击 Name 列中的卷名称。
3. 在卷的详细信息页面上，单击 Deregister。

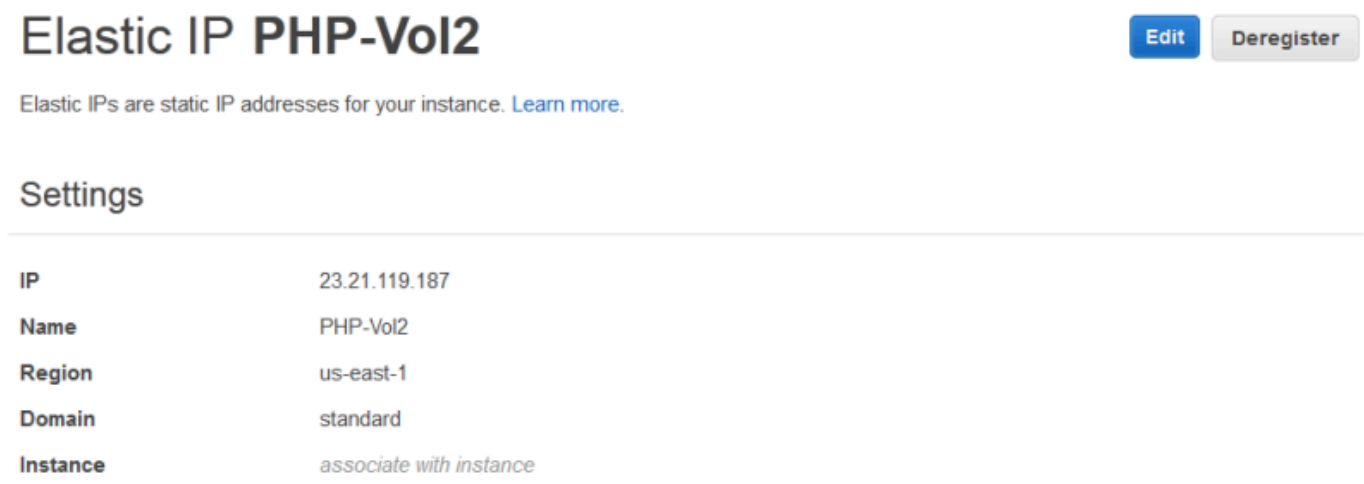


注销弹性 IP 地址

使用以下过程可注销弹性 IP 地址。

注销弹性 IP 地址

1. 如果地址与某个实例关联，则可注销地址，如取消关联弹性 IP 地址中所述。
2. 在 Resources 页面上，单击 Elastic IPs，然后单击 Address 列中的 IP 地址。
3. 在地址的详细信息页面上，单击 Deregister。



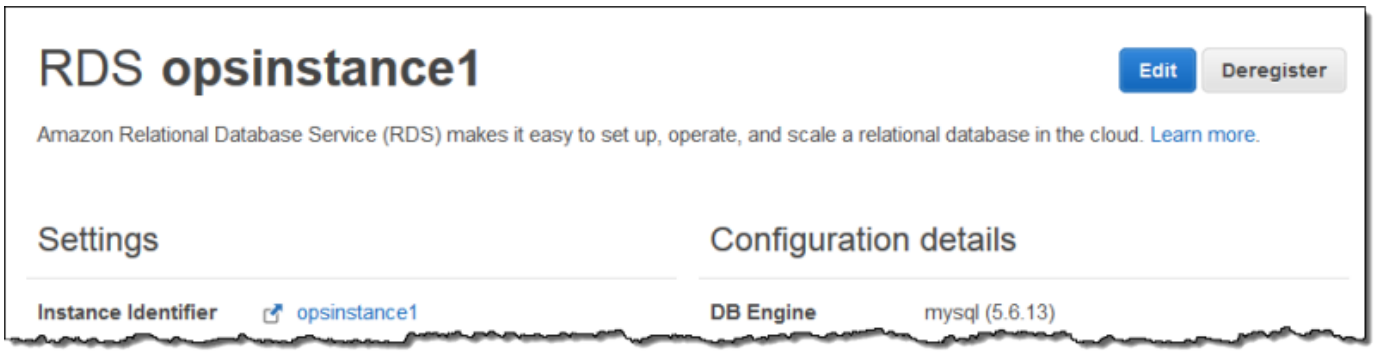
> **Note**
>
> 如果您只想将某个弹性 IP 地址注册到另一个堆栈，则必须将其从当前堆栈中注销，然后将其注册到新的堆栈。不过，您可以将挂载的弹性 IP 地址直接移动到另一个堆栈中的实例。有关更多信息，请参阅 挂载和移动资源。

注销Amazon RDS 实例

使用以下过程可注销 Amazon RDS 实例。

取消注册 Amazon RDS 实例

1. 如果实例与某个应用程序关联，则分离实例，如分离资源中所述。
2. 在 Resources 页面上，单击 RDS，然后单击实例的名称。
3. 在实例的详细信息页面上，单击 Deregister。



标签

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

标签可以帮助您对 Chef 11.10、Chef 12 和 Chef 12.2 堆栈中的资源进行分组，并跟踪在 AWS Billing and Cost Management 中使用这些资源的成本。

您可以在堆栈和层级别应用标签。当您创建一个标签后，您将对标记的结构范围内的每个资源应用该标签。例如，如果您向某个层应用标签，该标签将应用于层中的每个实例、Amazon EBS 卷（不包括根）或 Elastic Load Balancing 负载均衡器。标签当前不能应用于实例的根，或默认的 EBS 卷。

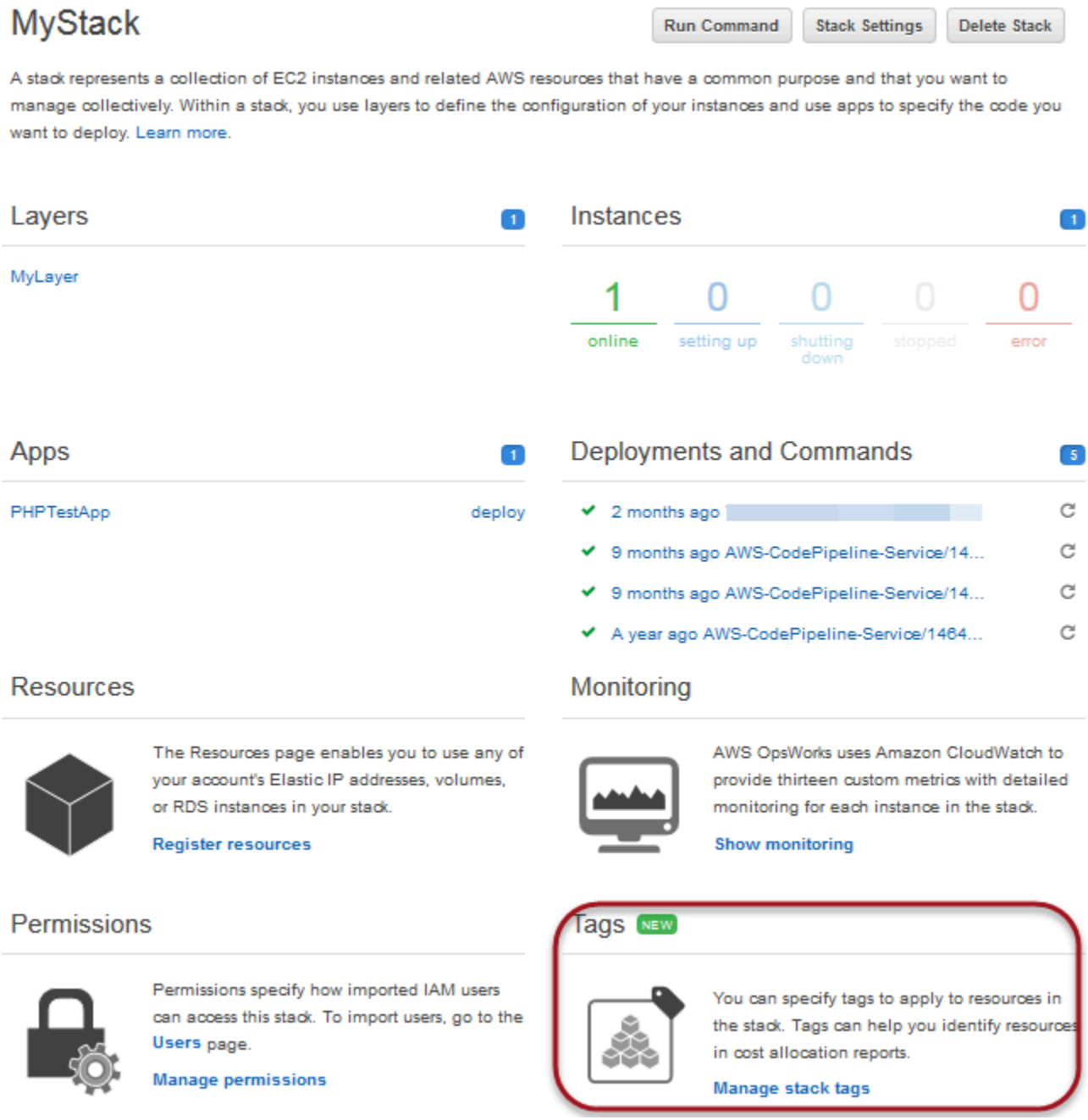
标签是您分配给堆栈或堆栈中图层的键值对。 AWS OpsWorks 创建标签后，打开 Billing and Cost Management 控制台以激活用户定义的标签。有关如何激活标签并使用它们来跟踪和管理 AWS OpsWorks 堆栈资源成本的更多信息，请参阅《B illing and Cost Management 用户指南》中的使用成本分配标签和激活用户定义的成本分配标签。

标签的工作方式类似于 AWS OpsWorks Stacks 中的自定义属性。您应用于堆栈的标签将由该堆栈中的每个层继承。在图层级别，您可以覆盖继承标签的值（但不能覆盖密钥名称），并添加新的特定于图层的标签。 AWS OpsWorks 将生成的标签集应用于图层中的所有资源。当您创建新资源或将现有资源分配到某个层时，该层中的新资源将用相同的标签集进行标记。

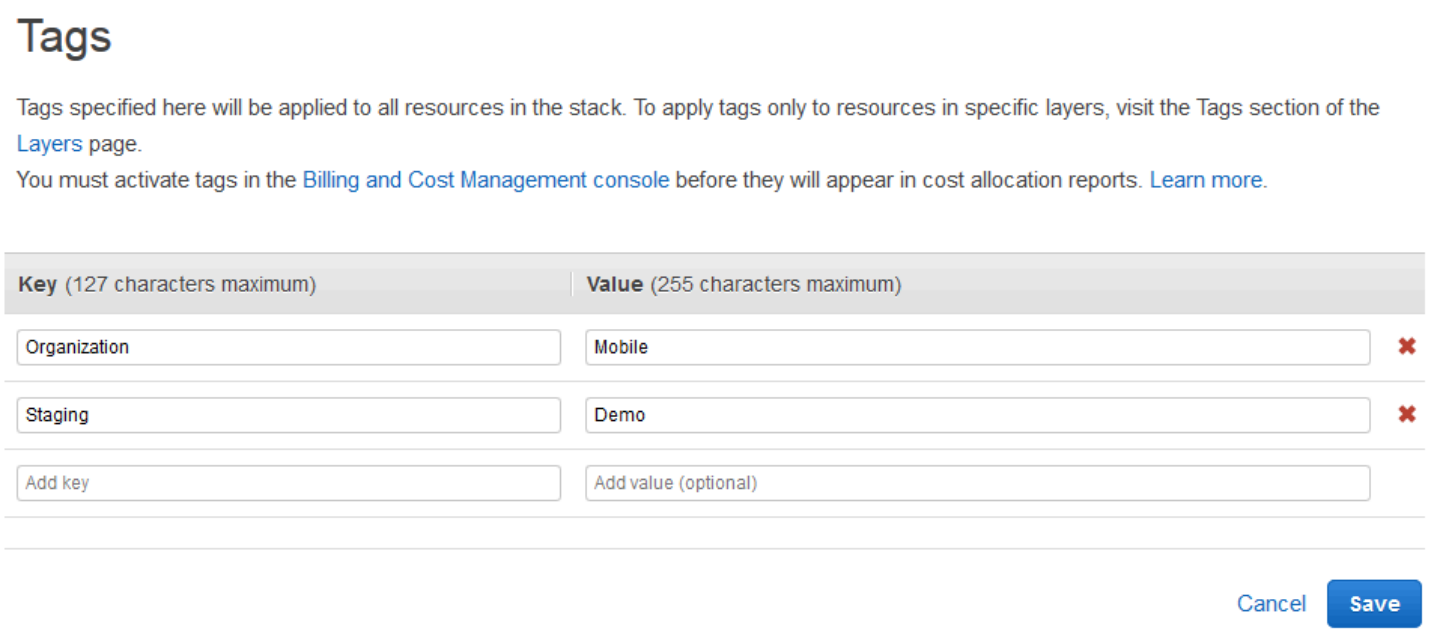
主题

在堆栈级别设置标签

在堆栈级别，您可以通过在堆栈主页上选择 Tags 来添加和管理标签。

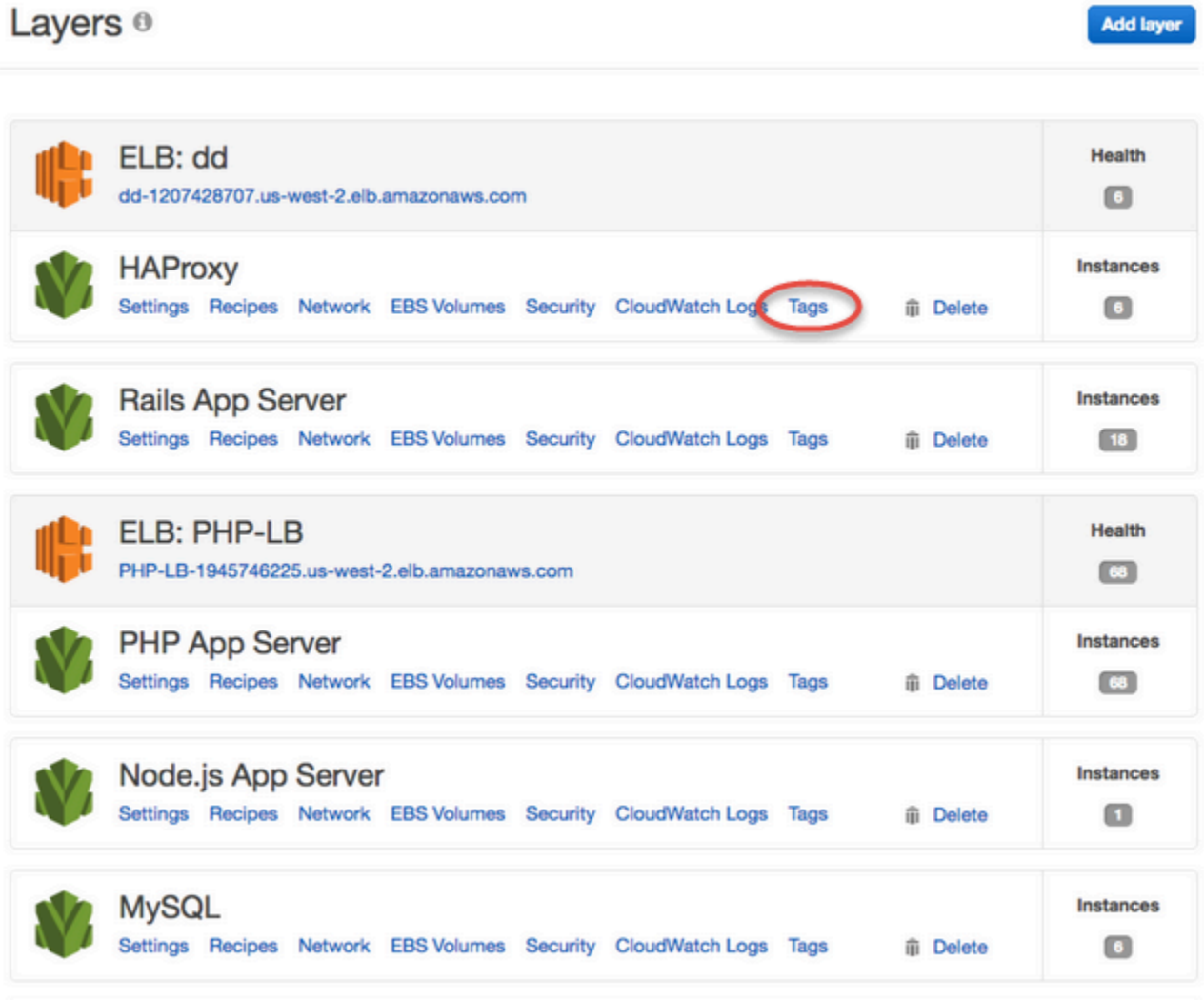


在 Tags 页面上，将标签添加为键值对。以下屏幕截图显示了一些示例标签。您可以通过选择键值对右侧的红色 X 来删除标签。

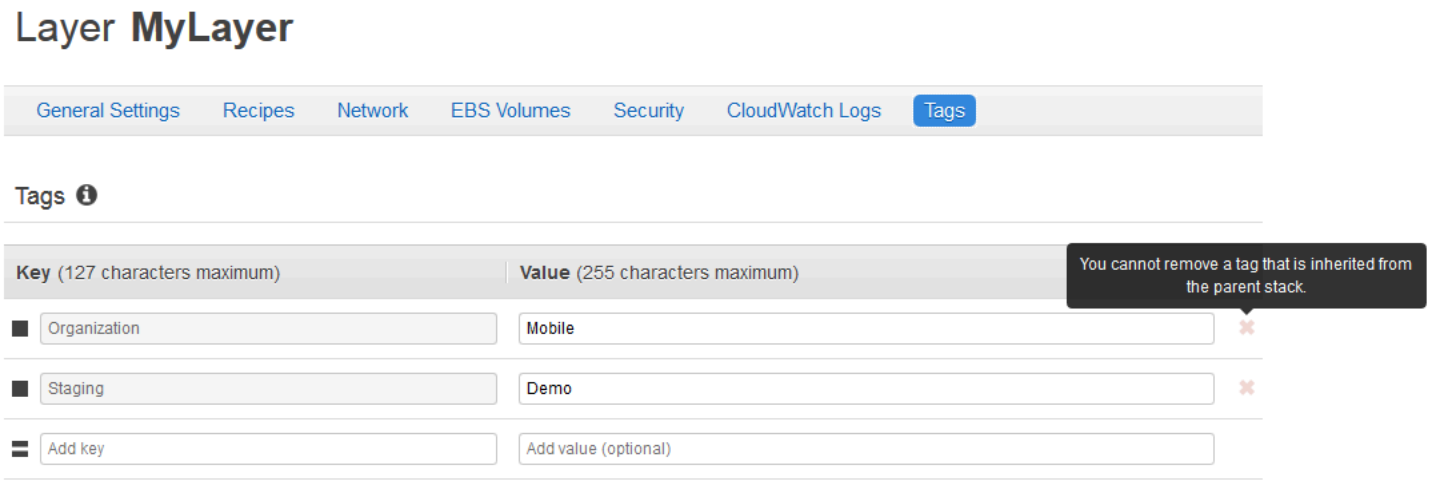


在层级别设置标签

在层级别，通过选择 Tags 选项卡来设置标签。您可以在 Layers (层) 主页以及每个层的主页上找到此选项卡。



在层级别更改或添加标签时，请注意，已在父堆栈级别添加的标签将由该层及其资源继承。虽然您可以更改继承标签的值，但无法更改键名称或删除继承标签。可在堆栈设置中更改键名称或删除从父堆栈继承的标签。以下屏幕截图显示了从堆栈级别继承的标签示例。继承标签将灰显。



有关向堆栈添加标签的更多信息，请参阅创建新堆栈。有关向层添加标签的更多信息，请参阅编辑图 OpsWorks 层的配置。

使用管理标签 AWS CLI

您还可以使用 AWS CLI 命令在堆栈和图层级别添加和移除标签。有关下载和安装的更多信息 AWS CLI，请参阅安装 AWS 命令行界面。请记住，如果要标记的堆栈不在默认区域内，则将 `--region` 参数添加到您的命令中。图层 ARN 当前未显示在。 AWS Management Console要获取层的 ARN，请运行 describe-layers 命令。

要添加标签，请使用 AWS CLI

- 在 AWS CLI 命令提示符处，键入以下命令，替换 ***stack_or_layer_arn ##############*** Enter。使用反斜杠转义双引号。

```
aws opsworks tag-resource --resource-arn stack_or_layer_ARN --tags "{\"key\":
\"value\",\"key\":\"value\"}"
```

示例如下：

```
aws opsworks tag-resource --resource-arn arn:aws:opsworks:us-
east-2:800000000003:stack/500b99c0-ec00-4cgg-8a0d-1000000jjd1b --tags "{\"Stage\":
\"Production\",\"Organization\":\"Mobile\"}"
```

要移除标签，请使用 AWS CLI

- 在 AWS CLI 命令提示符处，键入以下内容，然后按 Enter。

```
aws opsworks untag-resource --resource-arn stack_or_layer_ARN --tag-keys "[\"key\",
\"key\"]"
```

要删除标签，您只需指定要删除的标签的键即可。示例如下：

```
aws opsworks untag-resource --resource-arn arn:aws:opsworks:us-
east-2:800000000003:stack/500b99c0-ec00-4cgg-8a0d-1000000jjd1b --tag-keys "[\"Stage
\",\"Organization\"]"
```

> **Note**
>
> 您无法从层中删除继承的标签 (在父堆栈级别添加的标签)；应该从堆栈中删除继承的标签。

标签限制

创建标签时，请注意以下限制。

- AWS OpsWorks 堆栈将堆栈和层级的用户定义标签数量限制为 40 个，包括继承自父级的用户定义标签。这就留下了 10 个可用插槽，用于存放前置的`opsworks:`默认标签以及由其他 AWS 进程设置的标签。一个资源上最多允许 50 个标签，包括由创建的用户定义和默认标签 AWS。
- 标签键不得以 **`aws:`**、**`opsworks:`** 或 **`rds:`** 开头。请勿使用**`name`**或**`Name`**作为标签密钥，因为**`Name`**该密钥由 AWS OpsWorks Stacks 保留。
- 一个键最多可包含 127 个字符，并且只能包含 Unicode 字母、数字或分隔符，或以下特殊字符：`+ - = . _ : /`。
- 一个值最多可包含 255 个字符，并且只能包含 Unicode 字母、数字或分隔符，或以下特殊字符：`+ - = . _ : /`。

监控

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

您可以通过以下方法来监控堆栈。

- AWS OpsWorks Stacks 使用 Amazon CloudWatch 提供 13 个自定义指标，并对堆栈中的每个实例进行详细监控。
- AWS OpsWorks Stacks 与集成 AWS CloudTrail ，可记录每个 AWS OpsWorks Stacks API 调用，并将数据存储在 Amazon S3 存储桶中。

- 您可以使用 Amazon CloudWatch Logs 来监控堆栈的系统、应用程序和自定义日志。

主题

使用 Amazon 监控堆栈 CloudWatch

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

AWS OpsWorks 堆栈使用 Amazon CloudWatch (CloudWatch) 来监控堆栈。

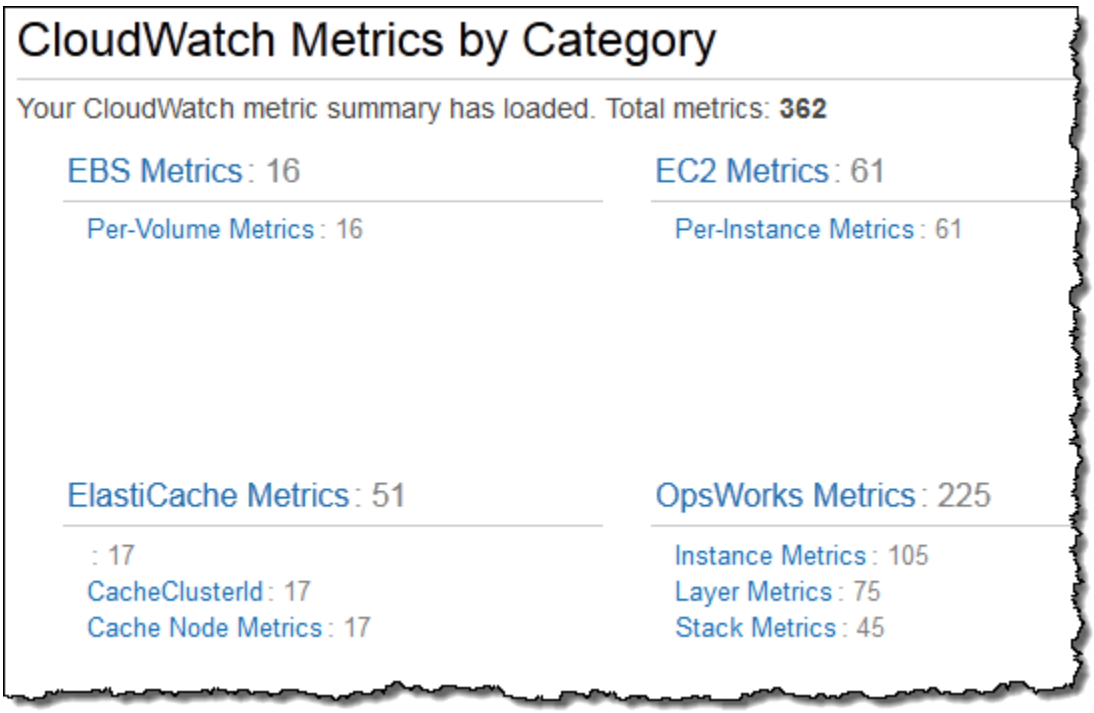
- 对于 Linux 堆栈， AWS OpsWorks Stacks 支持 13 个自定义指标，以提供堆栈中每个实例的详细监控，并在 “监控” 页面上汇总数据，方便您使用。
- 对于 Windows 堆栈，您可以使用CloudWatch 控制台监控实例的标准 Amazon EC2 指标。

  Monitoring 页面不会显示 Windows 指标。

“监控” 页面显示整个堆栈、层或实例的指标。 AWS OpsWorks 堆栈指标不同于 Amazon EC2 指标。您也可以通过 CloudWatch 控制台启用其他指标，但这些指标通常需要额外收费。您还可以在 CloudWatch 控制台上查看底层数据，如下所示：

要在中查看 OpsWorks 自定义指标 CloudWatch

1. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
2. 在导航栏中，选择堆栈的区域。
3. 在导航窗格中，选择指标。
4. 在 OpsWorks指标中，选择实例指标、层指标或堆栈指标。



> **Note**
>
> AWS OpsWorks Stacks 通过在每个实例（实例代理）上运行一个进程来收集指标。由于使用虚拟机管理程序 CloudWatch 收集指标的方式不同，因此控制台中的值可能与 AWS OpsWorks Stacks CloudWatch 控制台的 “监控” 页面上的相应值略有不同。

您也可以使用 CloudWatch 控制台设置警报。有关如何创建警报的更多信息，请参阅创建 Amazon CloudWatch 警报。有关 CloudWatch 自定义指标的列表，请参阅 AWS OpsWorks 指标和维度。有关更多信息，请参阅 Amazon CloudWatch。

主题

- AWS OpsWorks 堆栈指标
- AWS OpsWorks 堆栈指标的维度
- 堆栈指标
- 层指标
- 实例指标

AWS OpsWorks 堆栈指标

AWS OpsWorks Stacks CloudWatch 每五分钟向其发送一次以下指标。

CPU 指标

指标	描述
cpu_idle	CPU 空闲时间的百分比。 有效维度：您正在查看其指标的各个资源的 ID：StackId LayerId、或 InstanceId。 有效统计数据：Average, Minimum, Maximum, Sum, or Data Samples。 单位：无
cpu_nice	CPU 处理具有正 nice 值且调度优先级较低的进程的时间百分比。关于更多衡量内容的信息，请参阅 nice (Unix)。 有效维度：您正在查看其指标的各个资源的 ID：StackId LayerId、或 InstanceId。 有效统计数据：Average, Minimum, Maximum, Sum, or Data Samples。 单位：无
cpu_steal	随着 AWS 在越来越多的实例中分配虚拟机管理程序 CPU 资源，虚拟化负载会增加，并可能影响虚拟机管理程序在实例上执行请求工作的频率。cpu_steal 衡量实例等待虚拟机管理程序分配物理 CPU 资源的时间百分比。 有效维度：您正在查看其指标的各个资源的 ID：StackId LayerId、或 InstanceId。 有效统计数据：Average, Minimum, Maximum, Sum, or Data Samples。

指标	描述
	单位：无
cpu_system	CPU 正处理系统操作的时间百分比。 有效维度：您正在查看其指标的各个资源的 ID：StackId LayerId、或 InstanceId。 有效统计数据：Average, Minimum, Maximum, Sum, or Data Samples。 单位：无
cpu_user	CPU 正处理用户操作的时间百分比。 有效维度：您正在查看其指标的各个资源的 ID：StackId LayerId、或 InstanceId。 有效统计数据：Average, Minimum, Maximum, Sum, or Data Samples。 单位：无
cpu_waitio	CPU 等待输入/输出操作完成的时间百分比。 有效维度：您正在查看其指标的各个资源的 ID：StackId LayerId、或 InstanceId。 有效统计数据：Average, Minimum, Maximum, Sum, or Data Samples。 单位：无

内存指标

指标	描述
memory_buffers	缓冲内存量。

指标	描述
	有效维度：您正在查看其指标的各个资源的 ID：StackId LayerId、或 InstanceId。 有效统计数据：Average, Minimum, Maximum, Sum, or Data Samples。 单位：无
memory_cached	缓存的内存量。 有效维度：您正在查看其指标的各个资源的 ID：StackId LayerId、或 InstanceId。 有效统计数据：Average, Minimum, Maximum, Sum, or Data Samples。 单位：无
memory_free	可用内存量。 有效维度：您正在查看其指标的各个资源的 ID：StackId LayerId、或 InstanceId。 有效统计数据：Average, Minimum, Maximum, Sum, or Data Samples。 单位：无
memory_swap	交换空间量。 有效维度：您正在查看其指标的各个资源的 ID：StackId LayerId、或 InstanceId。 有效统计数据：Average, Minimum, Maximum, Sum, or Data Samples。 单位：无

指标	描述
memory_total	内存的总量。 有效维度：您正在查看其指标的各个资源的 ID：StackId LayerId、或 InstanceId。 有效统计数据：Average, Minimum, Maximum, Sum, or Data Samples。 单位：无
memory_used	使用中的内存量。 有效维度：您正在查看其指标的各个资源的 ID：StackId LayerId、或 InstanceId。 有效统计数据：Average, Minimum, Maximum, Sum, or Data Samples。 单位：无

负载指标

指标	描述
load_1	一分钟窗口内的平均负载。 有效维度：您正在查看其指标的各个资源的 ID：StackId LayerId、或 InstanceId。 有效统计数据：Average, Minimum, Maximum, Sum, or Data Samples。 单位：无
load_5	五分钟窗口内的平均负载。 有效维度：您正在查看其指标的各个资源的 ID：StackId LayerId、或 InstanceId。

指标	描述
	有效统计数据：Average, Minimum, Maximum, Sum, or Data Samples。 单位：无
load_15	15 分钟窗口内的平均负载。 有效维度：您正在查看其指标的各个资源的 ID：StackId LayerId、或 InstanceId。 有效统计数据：Average, Minimum, Maximum, Sum, or Data Samples。 单位：无

进程指标

指标	描述
procs	活动进程的数量。 有效维度：您正在查看其指标的各个资源的 ID：StackId LayerId、或 InstanceId。 有效统计数据：Average, Minimum, Maximum, Sum, or Data Samples。 单位：无

AWS OpsWorks 堆栈指标的维度

AWS OpsWorks 堆栈指标使用 AWS OpsWorks Stacks 命名空间，并为以下维度提供指标：

维度	描述
StackId	堆栈的平均值。

维度	描述
LayerId	分层的平均值。
InstanceId	实例的平均值。

堆栈指标

要查看整个堆栈的指标摘要，请在堆栈控制面板中选择一个堆栈，然后在导航窗格中单击监控。 AWS OpsWorks 以下示例针对的是具有 PHP 和 DB 层的堆栈。

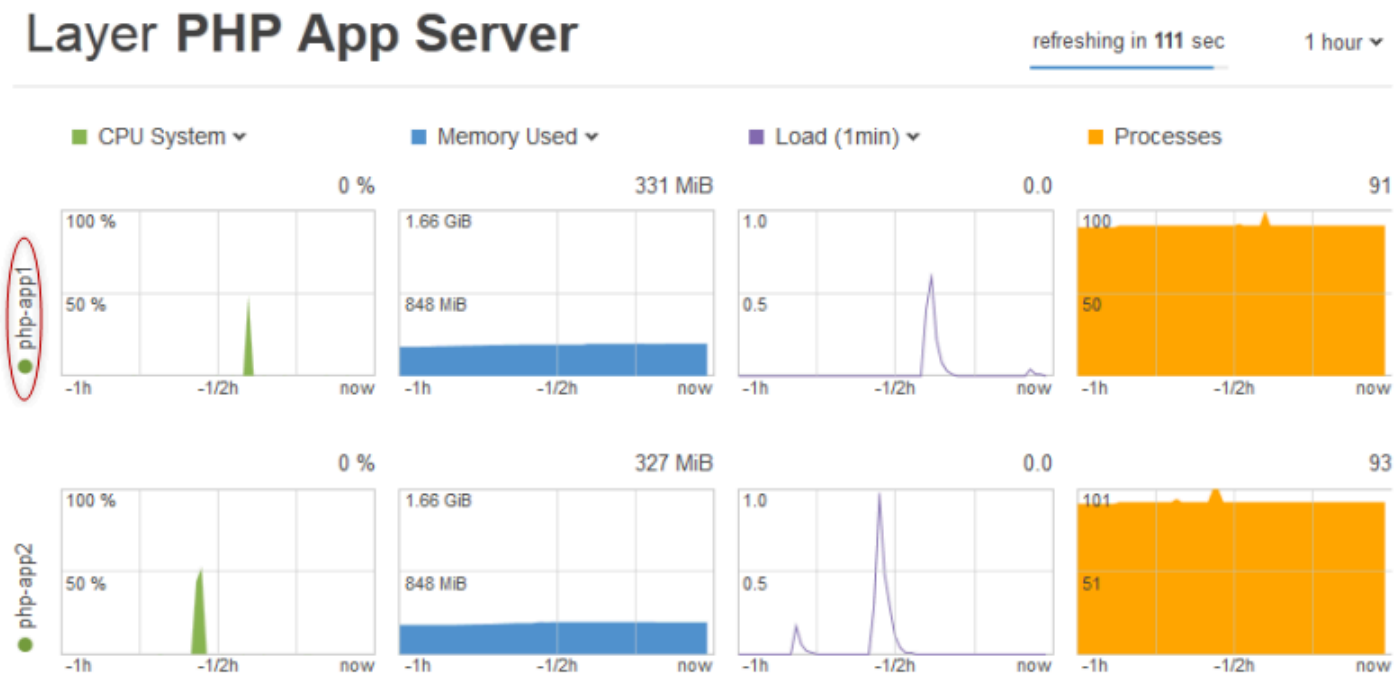
堆栈视图针对每个层显示了指定时间段 (1 小时、8 小时、24 小时、1 周或 2 周) 内四个类型的指标的图表。请注意以下几点：

- AWS OpsWorks Stacks 会定期更新图表；右上角的倒计时器表示距离下次更新的剩余时间，
- 如果某个层具有多个实例，则图表将显示该层的平均值。
- 您可以通过单击右上角的列表并选择您首选的值来指定时间段。

对于每个指标类型，您可以使用图表顶部的列表来选择要查看的特定指标。

层指标

要查看特定层的指标，请单击 Monitoring Layers 视图中的层名称。以下示例显示了 PHP 层的指标，该层有两个实例。

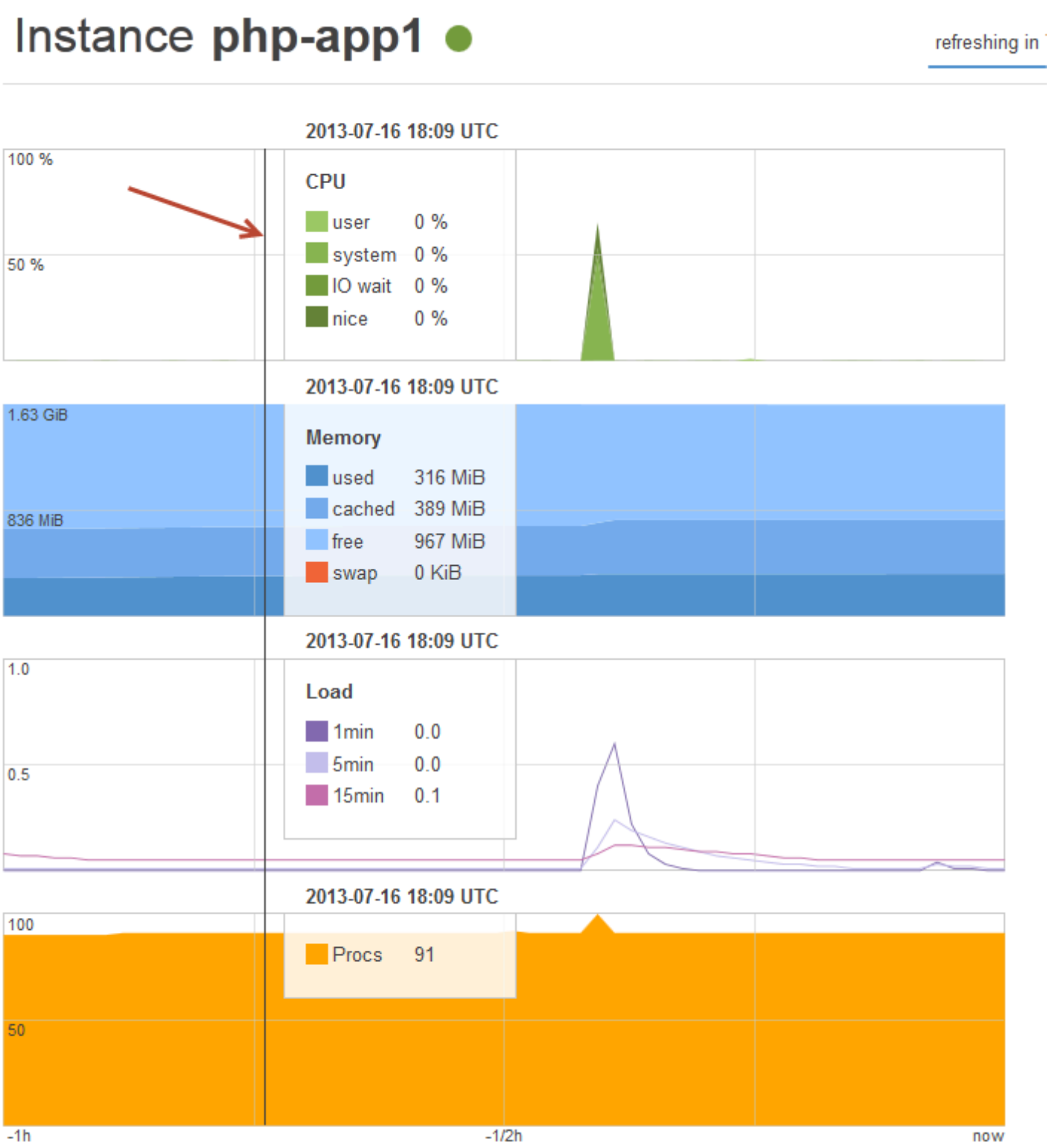


这些指标类型与堆栈指标的类型相同，对于每种类型，您都可以使用图表顶部的列表来选择要查看的特定指标。

> **Note**
>
> 您也可以显示层指标，方法是转到层的详细信息页面并单击右上角的 Monitoring。

实例指标

要查看特定实例的指标，请单击层监控视图中的实例名称。以下示例显示了 PHP 层的 php-app1 实例的指标。



这些图表汇总了每个指标类型的所有可用指标。要获取特定时间点的准确值，请使用鼠标将滑块 (在上图中以红色箭头表示) 移动到合适的位置。

> **Note**
>
> 您也可以显示实例指标，方法是转到实例的详细信息页面并选择右上角的 Monitoring。

使用记录 AWS OpsWorks 堆栈 API 调用 AWS CloudTrail

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

AWS OpsWorks Stacks 与 AWS CloudTrail提供 IAM 身份所采取操作记录的服务或 AWS OpsWorks Stacks 中的 AWS 服务集成。 CloudTrail 将 AWS OpsWorks 堆栈的所有 API 调用捕获为事件，包括来自 AWS OpsWorks 堆栈控制台的调用和对 AWS OpsWorks Stacks API 的代码调用。如果您创建跟踪，则可以将 CloudTrail 事件持续传输到 Amazon S3 存储桶，包括 AWS OpsWorks 堆栈的事件。如果您未配置跟踪，您仍然可以在 CloudTrail 控制台的 “事件历史记录” 中查看最新的事件。使用收集的信息 CloudTrail，您可以确定向 AWS OpsWorks Stacks 发出的请求、发出请求的 IP 地址、谁发出了请求、何时发出请求以及其他详细信息。

要了解更多信息 CloudTrail，请参阅AWS CloudTrail 用户指南。

AWS OpsWorks 将信息堆叠在 CloudTrail

CloudTrail 在您创建 AWS 账户时已在您的账户上启用。当活动发生在 AWS OpsWorks Stacks 中时，该活动会与其他 AWS 服务 CloudTrail 事件一起记录在事件历史记录中。您可以在自己的 AWS 账户中查看、搜索和下载最近发生的事件。有关更多信息，请参阅使用事件历史记录查看 CloudTrail 事件。

要持续记录您的 AWS 账户中的事件，包括 AWS OpsWorks 堆栈的事件，请创建跟踪。跟踪允许 CloudTrail 将日志文件传输到 Amazon S3 存储桶。默认情况下，在控制台中创建跟踪记录时，此跟踪记录应用于所有区域。跟踪记录 AWS 分区中所有区域的事件，并将日志文件传送到您指定的 Amazon S3 存储桶。此外，您可以配置其他 AWS 服务，以进一步分析和处理 CloudTrail 日志中收集的事件数据。有关更多信息，请参阅：

- 创建跟踪概述
- CloudTrail 支持的服务和集成
- 配置 Amazon SNS 通知 CloudTrail

- 接收来自多个区域的 CloudTrail 日志文件和接收来自多个账户的 CloudTrail 日志文件

所有 AWS OpsWorks 堆栈操作均由 Stacks API 参考记录 CloudTrail 并记录在 AWS OpsWorks Stacks API 参考中。例如，对CreateLayerDescribeInstances、和StartInstance操作的调用会在 CloudTrail 日志文件中生成条目。

每个事件或日记账条目都包含有关生成请求的人员信息。身份信息可帮助您确定以下内容：

- 请求是使用根用户凭证还是 IAM 用户凭证发出的。
- 请求是使用角色还是联合用户的临时安全凭证发出的。
- 请求是否由其他 AWS 服务发出。

有关更多信息，请参阅CloudTrail 用户身份元素。

了解 AWS OpsWorks 堆栈日志文件条目

跟踪是一种配置，允许将事件作为日志文件传输到您指定的 Amazon S3 存储桶。 CloudTrail 日志文件包含一个或多个日志条目。事件代表来自任何来源的单个请求，包括有关请求的操作、操作的日期和时间、请求参数等的信息。 CloudTrail 日志文件不是公共 API 调用的有序堆栈跟踪，因此它们不会按任何特定顺序出现。

以下示例显示了演示该CreateLayer操作的 CloudTrail 日志条目。

```
  {
"Records": [
    {
        "awsRegion": "us-west-2",
        "eventID": "342cd1ec-8214-4a0f-a68f-8e6352feb5af",
        "eventName": "CreateLayer",
        "eventSource": "opsworks.amazonaws.com",
        "eventTime": "2014-05-28T16:05:29Z",
        "eventVersion": "1.01"ed,
        "requestID": "e3952a2b-e681-11e3-aa71-81092480ee2e",
        "requestParameters": {
            "attributes": {},
            "customRecipes": {},
            "name": "2014-05-28 16:05:29 +0000 a073",
            "shortname": "customcf4571d5c0d6",
            "stackId": "a263312e-f937-4949-a91f-f32b6b641b2c",
            "type": "custom"
```

```
            },
            "responseElements": null,
            "sourceIPAddress": "198.51.100.0",
            "userAgent": "aws-sdk-ruby/2.0.0 ruby/2.1 x86_64-linux",
            "userIdentity": {
                "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
                "accountId": "111122223333",
                "arn": "arn:aws:iam::111122223333:user/A-User-Name",
                "principalId": "AKIAI44QH8DHBEXAMPLE",
                "type": "IAMUser",
                "userName": "A-User-Name"
            }
        },
        {
            "awsRegion": "us-west-2",
            "eventID": "a860d8f8-c1eb-449b-8f55-eafc373b49a4",
            "eventName": "DescribeInstances",
            "eventSource": "opsworks.amazonaws.com",
            "eventTime": "2014-05-28T16:05:31Z",
            "eventVersion": "1.01",
            "requestID": "e4691bfd-e681-11e3-aa71-81092480ee2e",
            "requestParameters": {
                "instanceIds": [
                    "218289c4-0492-473d-a990-3fbe1efa25f6"
                ]
            },
            "responseElements": null,
            "sourceIPAddress": "198.51.100.0",
            "userAgent": "aws-sdk-ruby/2.0.0 ruby/2.1x86_64-linux",
            "userIdentity": {
                "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
                "accountId": "111122223333",
                "arn": "arn:aws:iam::111122223333:user/A-User-Name",
                "principalId": "AKIAI44QH8DHBEXAMPLE",
                "type": "IAMUser",
                "userName": "A-User-Name"
            }
        }
    ]
}
```

使用带 AWS OpsWorks 堆栈的 Amazon CloudWatch 日志

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

为了简化监控多个实例上的日志的过程， AWS OpsWorks Stacks 支持 Amazon CloudWatch 日志。您可以在 AWS OpsWorks Stacks 中启用图层级别的 CloudWatch 日志。 CloudWatch 日志集成适用于基于 Chef 11.10 和 Chef 12 Linux 的堆栈。启用 “ CloudWatch 日志” 会产生额外费用，因此请在开始之前查看 Amaz CloudWatch on P ricing。

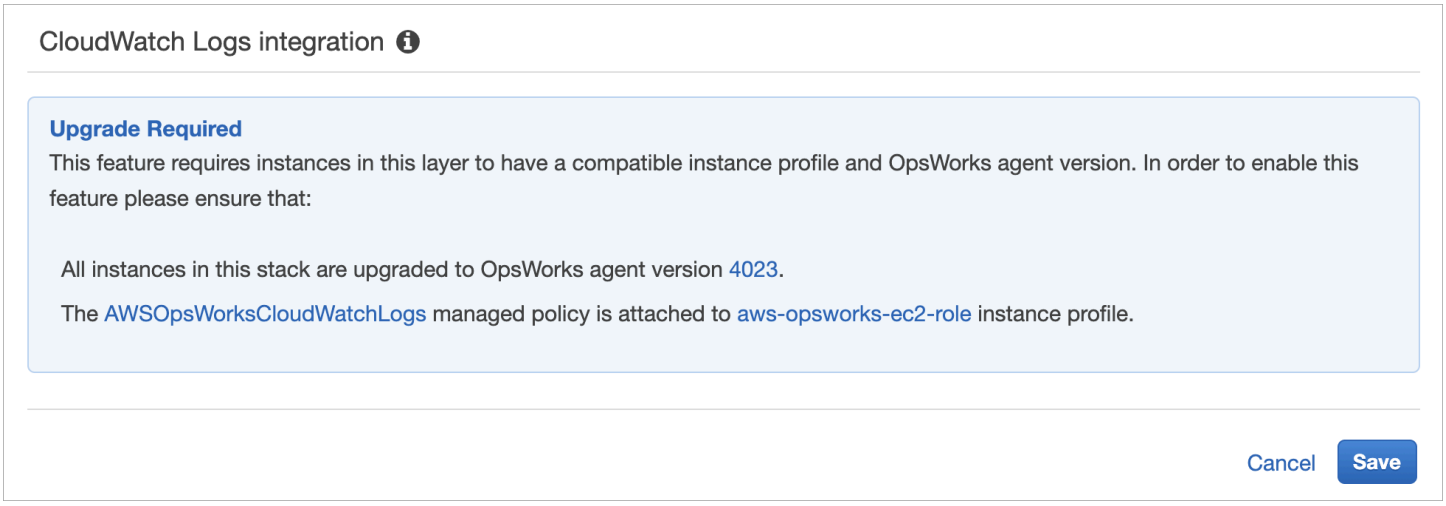
CloudWatch Logs 监视选定的日志中是否出现了用户指定的模式。例如，您可以监控关于某个字词 (例如 `NullReferenceException`) 的出现的日志，或计算出现这个字词的数量。启用 CloudWatch Logs in AWS OpsWorks Stacks 后，Stac AWS OpsWorks ks 代理会将日志发送到 CloudWatch Logs。有关 CloudWatch 日志的更多信息，请参阅 CloudWatch 日志入门。

先决条件

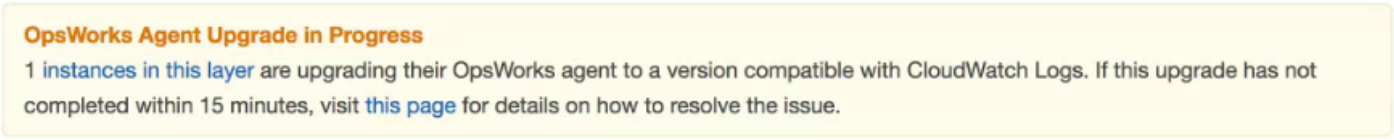
在启用 CloudWatch 日志之前，您的实例必须在 Chef 11.10 堆栈中运行 3444 或更高版本的 AWS OpsWorks 堆栈代理，在 Chef 12 堆栈中运行 4023 或更高版本。对于使用 CloudWatch 日志监控的任何实例，您还必须使用兼容的实例配置文件。

如果您使用的是自定义实例配置文件（ AWS OpsWorks Stacks 在创建堆栈时未提供该配置文件），则 AWS OpsWorks Stacks 无法自动升级实例配置文件。您必须使用 IAM 手动将AWSOpsWorksCloudWatchLogs策略附加到您的个人资料。相关信息，请参阅IAM 用户指南中的管理 IAM 策略。

如果您需要升级代理版本或实例配置文件，当您打开 “层” 页面上的 “ CloudWatch 日志” 选项卡时， AWS OpsWorks Stacks 会显示类似于以下屏幕截图的提醒。



更新层中所有实例上的代理可能要花费一些时间。如果您尝试在代理升级完成之前在层上启用 CloudWatch 日志，则会看到类似于以下内容的消息。



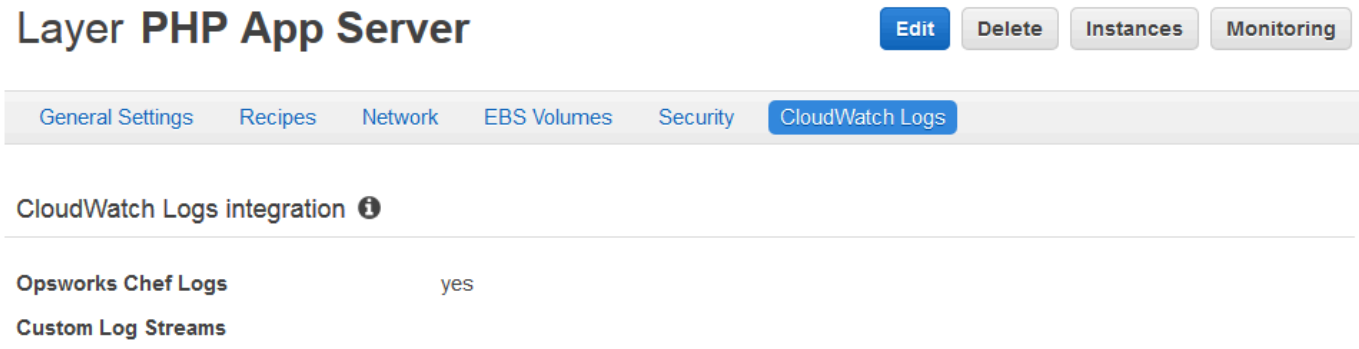
启用 CloudWatch 日志

1. 完成所有必需的代理和实例配置文件升级后，您可以通过将 CloudWatch 日志选项卡上的滑块控件设置为开启来启用CloudWatch 日志。

2. 要流式传输命令日志，请将 Stream command logs 滑块设置为 On。这会将您层实例上的 Chef 活动和用户启动的命令的 CloudWatch 日志发送到 Logs。

   当您打开日志 URL 的目标时，这些日志中包含的数据与您在DescribeCommands操作结果中看到的数据非常吻合。它包括关于 `setup`、`configure`、`deploy`、`undeploy`、`start`、`stop` 和配方运行命令的数据。

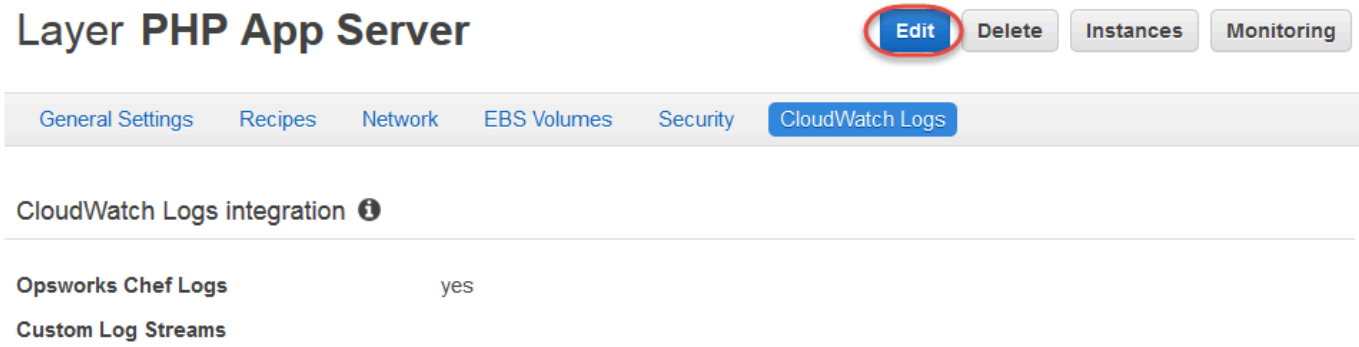
3. 要流式传输在您的层实例上的自定义位置中存储的活动日志 (如 `/var/log/apache/myapp/mylog*`)，请在 Stream custom logs 字符串框中键入自定义位置，然后选择 Add (+)。
4. 选择保存。几分钟后， AWS OpsWorks Stacks 日志流应会在 CloudWatch 日志控制台中可见。



关闭 CloudWatch 日志

要关闭 CloudWatch 日志，请编辑您的图层设置。

1. 在您的层属性页面上，选择 Edit。



2. 在编辑页面上，选择CloudWatch 日志选项卡。
3. 在 “CloudWatch 日志” 区域中，关闭 “流式传输” 命令日志。选择自定义日志上的 X 以从日志流中删除它们 (如果适用)。
4. 选择保存。

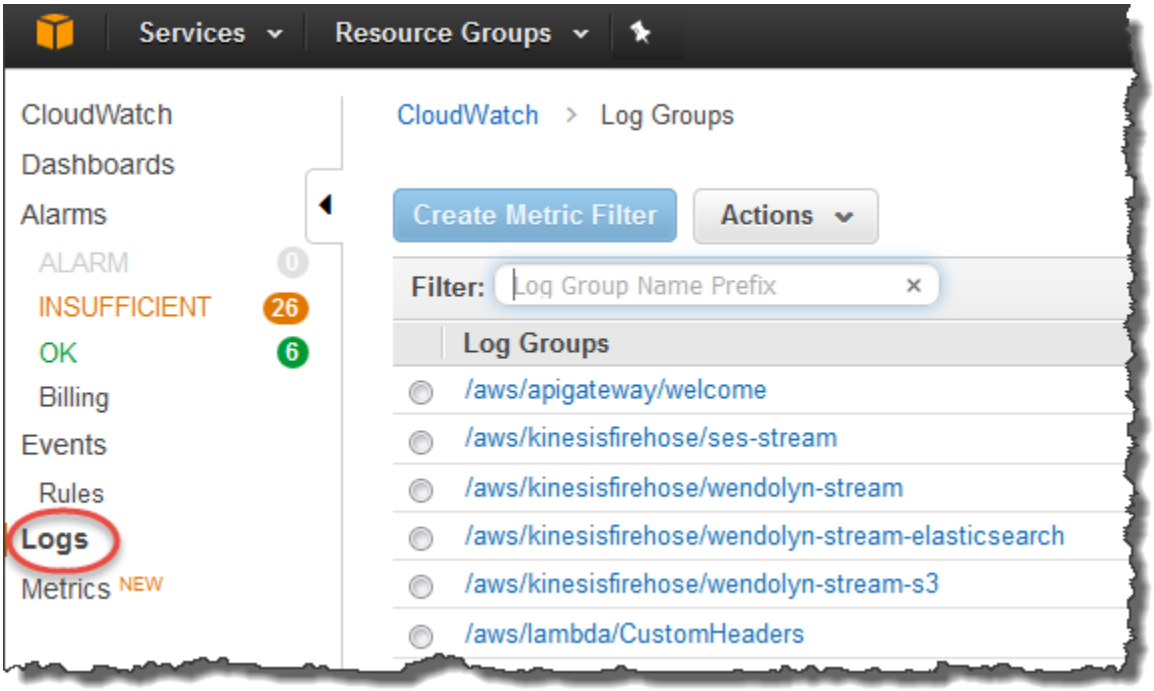
从日志中删除流式处理日 CloudWatch 志

关闭从 AWS OpsWorks 堆栈流式传输 CloudWatch 日志后，日志管理控制台中仍可使用现有 CloudWatch 日志。您存储的日志仍然产生费用，除非您将日志导出到 Amazon S3 中或删除它们。有关将日志导出到 S3 的更多信息，请参阅将日志数据导出到 Amazon S3。

您可以在日志管理控制台中删除日志流和日志组，也可以通过运行delete-log-stream和delete-log-group AWS CLI 命令来删除 CloudWatch 日志流和日志组。有关更改日志保留期限的更多信息，请参阅更改日志中的 CloudWatch 日志数据保留期。

在日志中管理您的 CloudWatch 日志

您正在流式传输的日志在 CloudWatch 日志控制台中进行管理。



AWS OpsWorks 自动创建默认日志组和日志流。 AWS OpsWorks Stacks 数据的日志组拥有与以下样式相匹配的名称：

stack_name/*layer_name*/*chef_log_name*

自定义日志拥有与以下样式相匹配的名称：

/stack_name/layer_short_name/file_path_name。删除特殊字符 (如星号 (*)) 可使路径名称更具可读性。

在日志中找到日志后，您可以将 CloudWatch 日志组织成组，通过创建指标筛选器来搜索和筛选日志，以及创建自定义警报。

将 Chef 12.2 Windows 层配置为使用 CloudWatch 日志

CloudWatch 基于 Windows 的实例不支持日志自动集成。“日CloudWatch 志” 选项卡在 Chef 12.2 堆栈中的图层上不可用。要为基于 Windows 的实例手动启用流式传输到 CloudWatch 日志，请执行以下操作。

- 更新基于 Windows 的实例的实例配置文件，以便 Logs CloudWatch 代理具有相应的权限。AWSOpsWorksCloudWatchLogs策略声明显示需要哪些权限。

  通常，您仅执行一次该任务。之后，对于层中的所有 Windows 实例，您可以使用更新后的实例配置文件。
- 编辑每个实例上的以下 JSON 配置文件。此文件包括日志流首选项，如要监控哪些日志。

  ```
  %PROGRAMFILES%\Amazon\Ec2ConfigService\Settings
  \AWS.EC2.Windows.CloudWatch.json
  ```

您可以创建用于处理所需任务的自定义配方，并将它们分配给 Chef 12.2 层的 Setup 事件，从而自动执行上述两项任务。每次您在这些图层上启动新实例时， AWS OpsWorks Stacks 会在实例完成启动后自动运行您的配方，从而 CloudWatch 启用 Logs。

要关闭基于 Windows 的实例上的 CloudWatch 日志，请撤消该过程。清除 “EC2 服务属性” 对话框中的 “启用 CloudWatch 日志集成” 复选框，从`AWS.EC2.Windows.CloudWatch.json`文件中删除日志流首选项；并停止运行任何自动为 Chef 12.2 层中的新实例分配 CloudWatch 日志权限的 Chef 配方。

使用 Amazon CloudWatch 事件监控堆栈

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

您可以在 Amazon CloudWatch Events 中配置规则，提醒您注意 AWS OpsWorks 堆栈资源的变化，并指示 CloudWatch 事件根据事件内容采取行动。有关如何开始使用 CloudWatch 事件和设置规则的更多信息，请参阅《活动用户指南》中的 CloudWatch CloudWatch 事件入门。

事件中支持以下 AWS OpsWorks Stacks CloudWatch 事件类型。

实例状态更改

表示 AWS OpsWorks Stacks 实例的状态发生了变化。

命令状态更改

表示 AWS OpsWorks Stacks 命令的状态发生了变化。

部署状态更改

表示 AWS OpsWorks Stacks 部署的状态发生了变化。

提醒

表示引发了 AWS OpsWorks Stacks 服务错误。

有关事件支持的 AWS OpsWorks 堆栈事件类型的更多信息，请参阅《 CloudWatch 事件用户指南》中的 AWS OpsWorks Stacks CloudWatch 事件。

安全性和权限

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

您的每个用户都必须拥有相应的 AWS 凭证才能访问您账户的 AWS 资源。向用户提供证书的推荐方法是使用 AWS Identity and Access Management(IAM)。 AWS OpsWorks 堆栈与 IAM 集成，可让您控制以下内容：

- 个人用户如何与 AWS OpsWorks Stacks 互动。

  例如，您可以允许某些用户将应用程序部署到任意堆栈，但不能修改堆栈本身，同时允许其他用户完全访问特定堆栈等。
- AWS OpsWorks 堆栈如何代表您访问堆栈资源，例如 Amazon EC2 实例和 Amazon S3 存储桶。

  AWS OpsWorks Stacks 提供了一个服务角色，用于授予这些任务的权限。
- 在 AWS OpsWorks 堆栈控制的 Amazon EC2 实例上运行的应用程序如何访问其他 AWS 资源，例如存储在 Amazon S3 存储桶中的数据。

您可以为层的实例分配实例配置文件，从而向在这些实例上运行的应用程序授予访问其他 AWS 资源的权限。

- 管理基于用户的 SSH 密钥并使用 SSH 或 RDP 连接到实例的方式。

  对于每个堆栈，管理用户可为每个 用户分配一个个人 SSH 密钥，或允许用户指定其自己的密钥。您也可以为每个用户授予堆栈实例的 SSH 或 RDP 访问权、sudo 或管理员权限。

其他安全方面包括：

- 如何对使用最新安全补丁更新实例的操作系统的操作进行管理。

  有关更多信息，请参阅 管理安全更新。

- 如何配置 Amazon EC2 安全组以控制流入和流出实例的网络流量。

  如何指定自定义安全组而不是 AWS OpsWorks Stacks 默认安全组。有关更多信息，请参阅 使用安全组。

主题

管理 AWS OpsWorks 堆栈用户权限

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

最佳做法是，将 AWS OpsWorks Stacks 用户限制为使用一组指定的操作或堆栈资源。您可以通过两种方式控制 AWS OpsWorks Stacks 用户权限：使用 AWS OpsWorks 堆栈权限页面和应用适当的 IAM 策略。

“ OpsWorks 权限” 页面（或等效的 CLI 或 API 操作）允许您通过为每个用户分配多个权限级别中的一个，在每个堆栈的基础上控制多用户环境中的用户权限。每个级别均为特定堆栈资源的一组标准操作授予权限。您可以使用 Permissions 页面来控制：

- 谁可以访问每个堆栈。
- 每个用户可以在每个堆栈上执行哪些操作。

  例如，您可以允许一些用户只能查看堆栈，而其他用户可以部署应用程序、添加实例等等。
- 谁可以管理每个堆栈。

  您可以将每个堆栈的管理工作委托给一个或多个指定用户。
- 谁对每个堆栈的 Amazon EC2 实例拥有用户级 SSH 访问权和 sudo 特权 (Linux)，或 RDP 访问权和管理员特权 (Windows)。

  您可以随时针对每个用户单独授予或删除这些权限。

> **Important**
>
> 拒绝 SSH/RDP 访问权不一定会阻止用户登录到实例。如果为实例指定 Amazon EC2 密钥对，则具有相应私有密钥的任何用户都可以登录或使用密钥来找回 Windows 管理员密码。有关更多信息，请参阅 管理 SSH 访问。

您可以使用 IAM 控制台、CLI 或 API，将策略附加至针对各种 AWS OpsWorks Stacks 资源和操作授予显式权限的用户。

- 使用 IAM policy 指定权限比使用权限级别更灵活。
- 您可以设置 IAM 身份（用户、用户组和角色），向用户和用户组等 IAM 身份授予权限，或者定义可以与联合用户关联的角色。
- IAM 策略是授予某些关键 AWS OpsWorks 堆栈操作权限的唯一方法。

  例如，您必须使用 IAM; 授予针对 `opsworks:CreateStack` 和 `opsworks:CloneStack` 的权限，这两个代码分别用于创建和克隆堆栈。

虽然无法在控制台中明确导入联合用户，但联合用户可以通过选择 AWS OpsWorks Stacks 控制台右上角的 “我的设置”，然后选择右上角的 “用户” 来隐式创建用户个人资料。在 用户 页面上，联合身份用户 (其账户是通过使用 API 或 CLI 创建的，或通过控制台隐式创建的) 可像非联合身份用户一样管理其账户。

这两种方法并不相互排斥，有时将这两种方法结合使用非常有用；随后 AWS OpsWorks Stacks 会评估两组权限。例如，假设您想允许用户添加或删除实例，但不允许其添加或删除层。 AWS OpsWorks Stacks 的所有权限级别均未授予该特定权限集。不过，您可以使用 权限 页面授予用户 管理 权限级别，这允许他们执行大部分堆栈操作，然后附加可拒绝添加或删除层的权限的 IAM policy。有关更多信息，请参阅使用策略控制访问 AWS 资源。

以下是管理用户权限的典型模型。在每种情况下，均假定读者 (您) 为管理用户。

1. 使用 IAM 控制台将 AWSOpsWorks_FullAccess 策略应用于一个或多个管理用户。
2. 利用不授予任何 AWS OpsWorks Stacks 权限的策略为每个非管理用户创建一个用户。

   如果用户只需要访问 AWS OpsWorks 堆栈，则可能根本不需要应用策略。相反，您可以通过 “ AWS OpsWorks 堆栈权限” 页面管理他们的权限。
3. 使用 “ AWS OpsWorks 堆栈用户” 页面将非管理用户导入堆栈。 AWS OpsWorks
4. 对于每个堆栈，请使用堆栈的 Permissions 页面向每个用户分配权限级别。
5. 根据需要，通过附加适当配置的 IAM policy 自定义用户的权限级别。

有关管理用户的更多建议，请参阅 最佳实践：管理权限。

有关 IAM 最佳实践的更多信息，请参阅IAM 用户指南中的 IAM 中的安全最佳实践。

主题

- 管理 AWS OpsWorks 堆栈用户
- 向 AWS OpsWorks 堆栈用户授予每个堆栈的权限
- 通过附加 IAM 策略管理 AWS OpsWorks 堆栈权限
- 示例策略
- AWS OpsWorks 堆栈权限级别

管理 AWS OpsWorks 堆栈用户

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

必须先为每个人创建一个用户，然后才能将用户导入 AWS OpsWorks Stacks 并向他们授予权限。要创建 IAM 用户，首先要以 AWS 已被授予 IAM FullAccess 策略中定义的权限的用户身份登录。然后，您可以使用 IAM 控制台为需要访问 AWS OpsWorks 堆栈的所有人创建 IAM 用户。然后，您可以将这些用户导入 AWS OpsWorks Stacks 并按如下方式授予用户权限：

普通 AWS OpsWorks 堆栈用户

常规用户不需要附加策略。如果他们有 Stacks 权限，则通常不包含任何 AWS OpsWorks Stacks 权限。取而代之的是，使用 AWS OpsWorks Stacks 权限页面为普通用户分配以下权限级别之一。stack-by-stack

- Show 权限允许用户查看堆栈，但不允许执行任何操作。
- Deploy 权限包括 Show 权限，还允许用户部署和更新应用程序。
- Manage 权限包括 Deploy 权限，还允许用户执行堆栈管理操作 (例如添加层或实例)、使用 Permissions 页面设置用户权限，并启用他们自己的 SSH/RDP 和 sudo/管理员权限。
- Deny 权限拒绝访问堆栈。

如果这些权限级别不太适合特定用户，则可以通过附加 IAM policy 来自定义用户的权限。例如，您可能希望使用 “ AWS OpsWorks 堆栈权限” 页面为用户分配 “管理” 权限级别，这会授予他们执行所有堆栈管理操作的权限，但不允许他们创建或克隆堆栈。然后，您可以附加一个策略以限制这些权限 (通过拒绝它们添加或删除层)，或扩充这些权限 (通过允许它们创建或克隆堆栈)。有关更多信息，请参阅 通过附加 IAM 策略管理 AWS OpsWorks 堆栈权限。

AWS OpsWorks 堆栈管理用户

管理用户是账户所有者或拥有策略定义权限的 IAM 用户。AWSOpsWorks_FullAccess 此策略除了包括授予 Manage 用户的权限之外，还包括无法通过 Permissions 页面授予的操作权限，例如以下权限：

- 将用户导入 AWS OpsWorks 堆栈

- 创建和克隆堆栈

有关完整策略的信息，请参阅示例策略。有关只能通过附加 IAM policy 来授予用户的权限的详细列表，请参阅 AWS OpsWorks 堆栈权限级别。

主题

- 用户和区域
- 创建 AWS OpsWorks Stacks 管理用户
- 为 AWS OpsWorks 堆栈创建 IAM 用户
- 将用户导入 AWS OpsWorks 堆栈
- 编辑 AWS OpsWorks 堆栈用户设置

用户和区域

AWS OpsWorks 堆栈用户可在创建堆栈的区域终端节点中使用。您可以在以下任一区域中创建用户。

- 美国东部（俄亥俄州）区域
- 美国东部（弗吉尼亚州北部）区域
- 美国西部（俄勒冈州）区域
- 美国西部（北加利福尼亚）区域
- 加拿大（中部）区域（仅限 API；未在 AWS Management Console
- 亚太地区（孟买）区域
- 亚太地区（新加坡）区域
- 亚太地区（悉尼）区域
- Asia Pacific（Tokyo）Region
- 亚太地区（首尔）区域
- 欧洲地区（法兰克福）区域
- 欧洲地区（爱尔兰）区域
- 欧洲地区（伦敦）区域
- 欧洲地区（巴黎）区域
- 南美洲（圣保罗）区域

将用户导入 AWS OpsWorks Stacks 时，将其导入其中一个区域终端节点；如果您希望用户在多个区域可用，则必须将用户导入该区域。您也可以将 AWS OpsWorks Stacks 用户从一个区域导入到另一个区域；如果您将用户导入到已有同名用户的区域，则导入的用户将替换现有用户。有关导入用户的更多信息，请参阅导入用户。

创建 AWS OpsWorks Stacks 管理用户

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

您可以通过向用户添加策略来创建 AWS OpsWorks Stacks 管理用户，该`AWSOpsWorks_FullAccess`策略授予该用户 AWS OpsWorks Stacks 完全访问权限。有关创建管理用户的更多信息，请参阅创建管理用户。

> **Note**
>
> 该 AWSOpsWorks_FullAccess 策略允许用户创建和管理 AWS OpsWorks Stacks 堆栈，但用户无法为堆栈创建 IAM 服务角色；他们必须使用现有角色。如 管理权限中所述，创建堆栈的第一个用户必须拥有额外的 IAM; 权限。当该用户创建第一个堆栈时， AWS OpsWorks Stacks 会创建一个具有所需权限的 IAM 服务角色。此后，拥有 `opsworks:CreateStack` 权限的任何用户均可使用该角色创建其他堆栈。有关更多信息，请参阅 允许 AWS OpsWorks Stacks 代表你行事。

创建用户时，您可以根据需要添加其他客户管理型策略来微调用户的权限。例如，您可能希望管理用户能够创建或删除堆栈，但不能导入新用户。有关更多信息，请参阅 通过附加 IAM 策略管理 AWS OpsWorks 堆栈权限。

如果您有多个管理用户，则无需为每个用户单独设置权限，而是可以将 AWSOpsWorks_FullAccess 策略添加到 IAM 群组并将用户添加到该群组。

有关创建组的信息，请参阅创建 IAM 用户组。创建群组时，添加AWSOpsWorks_FullAccess策略。您也可以添加AdministratorAccess策略，其中包括AWSOpsWorks_FullAccess权限。

有关向现有群组添加权限的信息，请参阅将策略附加到 IAM 用户组。

为 AWS OpsWorks 堆栈创建 IAM 用户

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

在将 IAM 用户导入 AWS OpsWorks 堆栈之前，您需要先创建它们。可以使用 IAM 控制台、命令行或 API 来执行此操作。有关完整说明，请参阅在您的 AWS 账户中创建 IAM 用户。

请注意，与管理用户不同的是，您不需要附加策略即可定义权限。如中所述，您可以在 AWS OpsWorks 将用户导入到 Stacks管理用户权限 后设置权限。

有关创建 IAM; 用户和组的更多信息，请参阅 IAM 入门。

将用户导入 AWS OpsWorks 堆栈

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

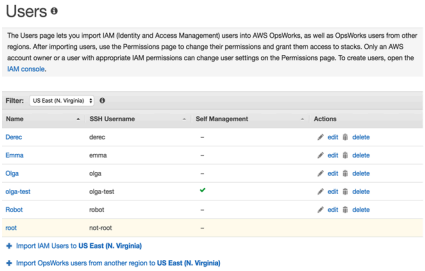
管理用户可以将用户导入 AWS OpsWorks Stacks；他们也可以将 AWS OpsWorks Stacks 用户从一个区域端点导入到另一个区域端点。将用户导入 AWS OpsWorks Stacks 时，会将他们导入到其中一个 AWS OpsWorks Stacks 区域终端节点。如果您希望用户在多个区域中可用，则必须将该用户导入到该区域中。

虽然无法在控制台中明确导入联合用户，但联合用户可以通过选择 AWS OpsWorks Stacks 控制台右上角的 “我的设置”，然后选择右上角的 “用户” 来隐式创建用户个人资料。在 用户 页面上，联合身份用户 (其账户是通过使用 API 或 CLI 创建的，或通过控制台隐式创建的) 可像非联合身份用户一样管理其账户。

将用户导入 AWS OpsWorks 堆栈

1. 以管理员用户或账户所有者的身份登录 AWS OpsWorks Stacks。

2. 在右上角选择 Users，以打开 Users 页面。



3. 选择 将 IAM 用户导入 <**##**> 以显示可用但尚未导入的用户。

4. 选中 Select all 复选框，或选择一个或多个单独用户。完成后，选择 “导入到” OpsWorks。

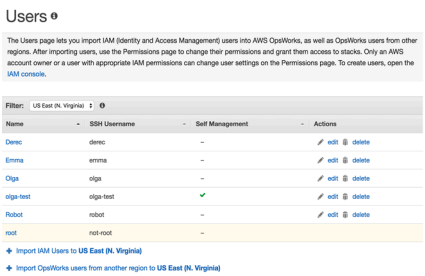
> **Note**
>
> 将用户导入 AWS OpsWorks Stacks 后，如果您使用 IAM 控制台或 API 从您的账户中删除该用户，则该用户不会自动失去您通过 AWS OpsWorks Stacks 授予的 SSH 访问权限。您还必须通过打开 “用户” 页面，然后在用户的 “操作” 列中选择删除，将用户从 AWS OpsWorks Stacks 中删除。

将 AWS OpsWorks Stacks 用户从一个区域导入到另一个区域

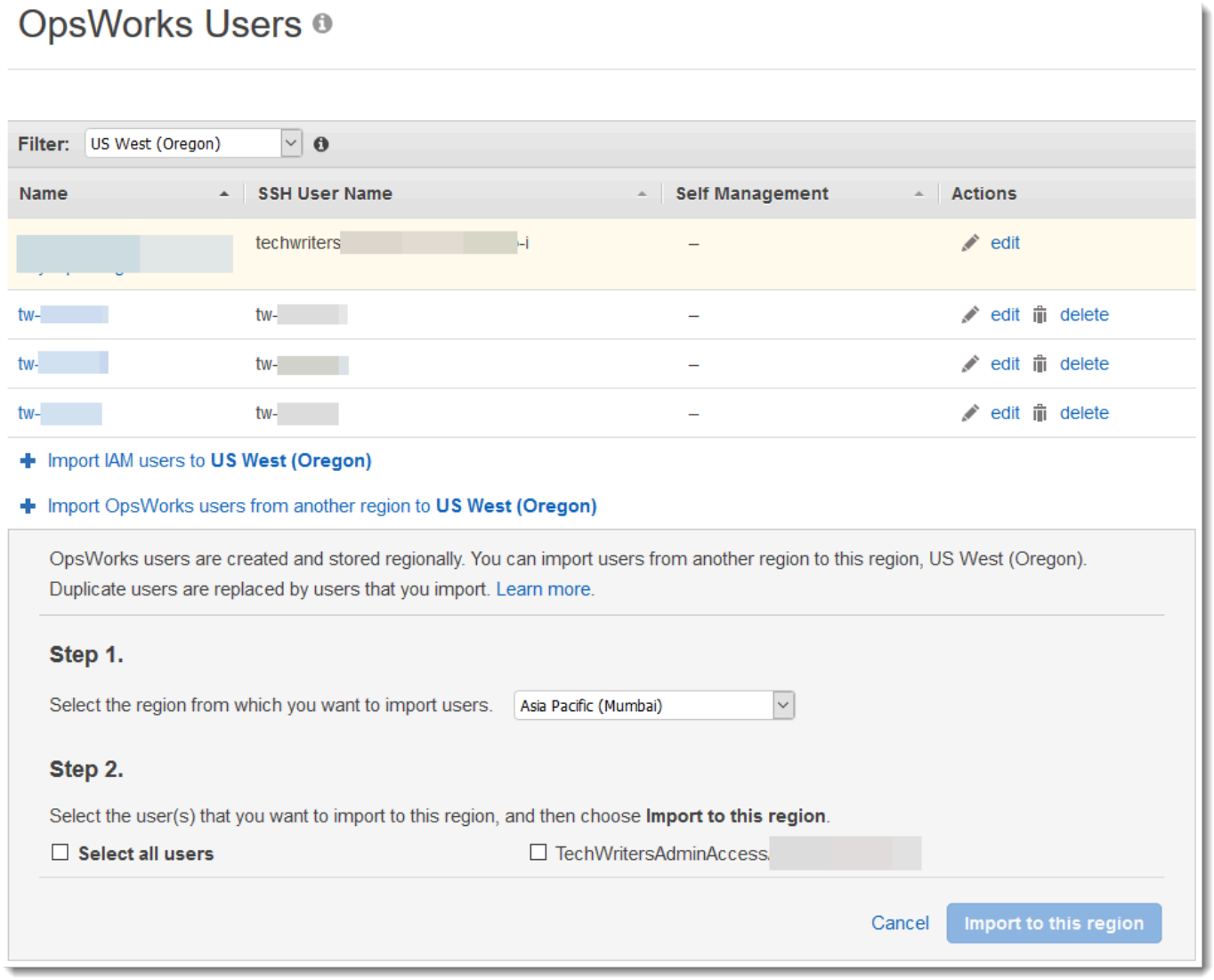
AWS OpsWorks 堆栈用户可在创建堆栈的区域终端节点中使用。您可以在用户和区域中所显示的区域中创建用户。

您可以将 AWS OpsWorks Stacks 用户从一个区域导入到当前筛选用户列表的区域。如果您将用户导入到已有具有相同名称的用户的区域中，则导入的用户将替换现有用户。

1. 以管理员用户或账户所有者的身份登录 AWS OpsWorks Stacks。
2. 在右上角选择 Users，以打开 Users 页面。如果您在多个区域中有 AWS OpsWorks Stacks 用户，请使用筛选控件筛选要将用户导入到的区域。



3. 选择从其他区域导入 AWS OpsWorks Stacks 用户到 < **####** >。



4. 选择要从中导入 AWS OpsWorks Stacks 用户的区域。
5. 选择一个或多个要导入的用户，或选择所有用户，然后选择 Import to this region。等待 AWS OpsWorks Stacks 在用户列表中显示导入的用户。

在 AWS OpsWorks 堆栈外创建的 Unix ID 和用户

AWS OpsWorks 为 AWS OpsWorks 堆栈实例上的用户分配介于 2000 到 4000 之间 Unix ID (UID) 的值。由于 AWS OpsWorks 保留了 2000-4000 个 UID 范围，因此您在之外创建的用户 AWS OpsWorks（例如，通过使用食谱食谱或 AWS OpsWorks 从 IAM 导入用户）可以拥有被 Stacks 覆盖的其他用户的 UID。 AWS OpsWorks 这可能会导致您在 AWS OpsWorks 堆栈之外创建的用户不会出现在数据包搜索结果中，或者被排除在 AWS OpsWorks Stacks 内置`sync_remote_users`操作之外。

外部进程也可以使用 AWS OpsWorks 堆栈可以覆盖的 UID 创建用户。例如，某些操作系统软件包可以创建用户，这是安装后流程的一部分。*########## Linux ################ UID##### ##Stacks ### UID # <##### UID> + 1# AWS OpsWorks AWS OpsWorks*

最佳做法是，创建 AWS OpsWorks Stacks 用户并在 AWS OpsWorks Stacks 控制台中或使用 SDK 管理他们的访问权限。 AWS CLI AWS 如果您确实在以外的 AWS OpsWorks 堆栈实例上创建用户 AWS OpsWorks，请使用大于 4000 的 *UnixID* 值。

编辑 AWS OpsWorks 堆栈用户设置

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

导入用户后，您可以按如下所示编辑其设置：

编辑用户设置

1. 在 Users 页面上，选择用户的 Actions 列中的 edit。
2. 您可以指定以下设置。

   Self Management

   选择 “是” 以允许用户使用该 MySettings 页面来指定他或她的个人 SSH 密钥。

   > **Note**
   >
   > 您还可以通过向 IAM 身份添加 IAM 策略来启用自我管理，该策略授予DescribeMyUserProfile和UpdateMyUserProfile操作的权限。

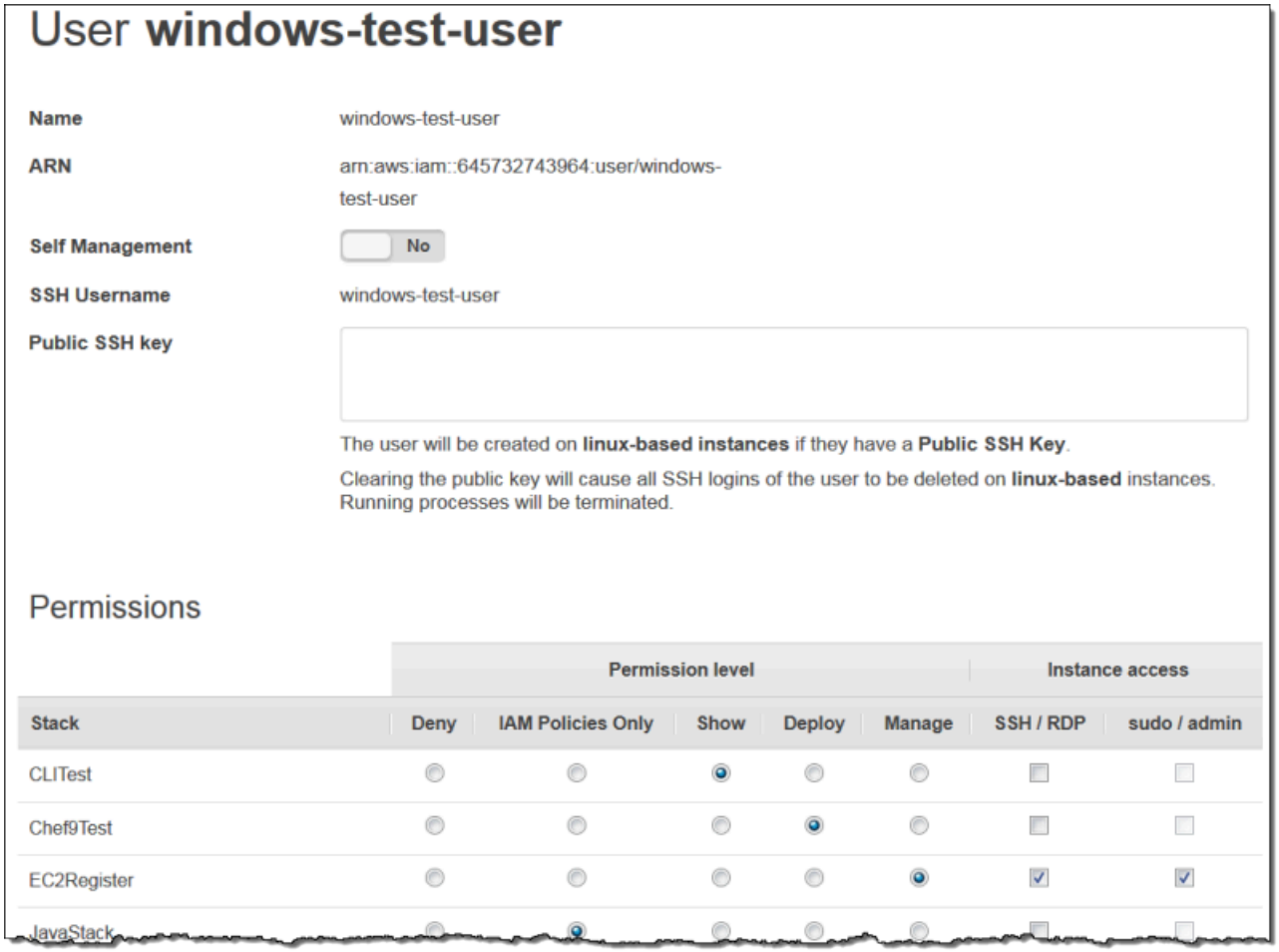
   Public SSH key

   (可选) 输入用户的公有 SSH 密钥。此密钥将显示在用户的 My Settings 页面上。如果您启用自我管理，则用户可以编辑 My Settings 并指定自己的密钥。有关更多信息，请参阅 注册用户的公有 SSH 密钥。

AWS OpsWorks Stacks 在所有 Linux 实例上安装此密钥；用户可以使用关联的私钥登录。有关更多信息，请参阅 使用 SSH 登录。不能将此密钥用于 Windows 堆栈。

权限

(可选) 在一个位置设置用户对每个堆栈的权限级别，而不是通过使用每个堆栈的 Permissions 页面来单独设置它们。有关权限级别的更多信息，请参阅授予每堆栈权限。



向 AWS OpsWorks 堆栈用户授予每个堆栈的权限

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

管理 AWS OpsWorks Stacks 用户权限的最简单方法是使用堆栈的 “权限” 页面。每个堆栈均有其自己的页面，该页面可授予针对该堆栈的权限。

您必须以管理员用户或者 Manage 用户的身份登录，才能修改任何权限设置。该列表仅显示那些已导入 AWS OpsWorks Stacks 的用户。有关如何创建和导入用户的信息，请参阅管理用户。

默认权限级别为“IAM Policies Only”，该级别仅向用户授予其附加 IAM policy 中的那些权限。

- 当您从 IAM; 或另一个区域导入用户时，将使用 IAM Policies Only 权限级别将用户添加到所有现有堆栈的列表中。
- 默认情况下，您刚从其他区域导入的用户无法访问目标区域中的堆栈。如果从其他区域导入用户，要让他们管理目标区域中的堆栈，则在导入用户之后，必须为这些堆栈分配权限。
- 当您创建新堆栈时，会使用 IAM Policies Only 权限级别将所有当前用户添加到列表中。

主题

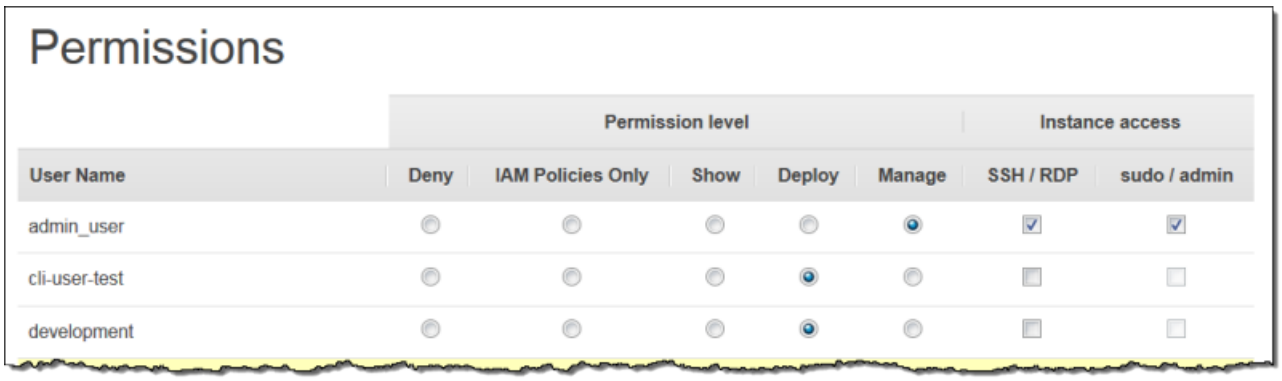
设置用户的权限

设置用户的权限

1. 在导航窗格中，选择 Permissions (权限)。
2. 在 Permissions (权限) 页面上，选择 Edit (编辑)。
3. 更改 Permission level (权限级别) 和 Instance access (实例访问) 设置：

   - 使用 Permissions level 设置为每个用户分配其中一个标准权限级别，从而确定用户是否可访问此堆栈以及用户可以执行哪些操作。如果用户拥有 IAM 策略， AWS OpsWorks Stacks 会评估这两组权限。有关示例，请参阅示例策略。
   - Instance access SSH/RDP 设置可指定用户是否对堆栈的实例具有 SSH (Linux) 或 RDP (Windows) 访问权。

     如果您授予 SSH/RDP 访问权，则可以选择 sudo/admin，这样可授予用户对堆栈实例的 sudo (Linux) 或管理 (Windows) 特权。



您可以将每个用户分配到以下某个权限级别。有关每个级别所允许的操作列表，请参阅AWS OpsWorks 堆栈权限级别。

拒绝

用户无法在堆栈上执行任何 AWS OpsWorks Stacks 操作，即使他们拥有授予 AWS OpsWorks Stacks 完全访问权限的 IAM 策略。例如，您可以利用这一点来拒绝某些用户访问堆栈以获取未发布的产品。

IAM Policies Only

默认级别，该级别分配给所有新导入的用户，以及新创建堆栈的所有用户。用户的权限由其所附加的 IAM policy 确定。如果用户没有 IAM 策略，或者他们的策略没有明确的 AWS OpsWorks 堆栈权限，则他们无法访问堆栈。通常会向管理用户分配该级别，因为它们的附加 IAM 策略已经授予完全访问权限。

Show (显式)

用户可查看堆栈，但不能执行任何操作。例如，管理员可能希望监控账户的堆栈，但不需要以任何方式部署应用程序或修改堆栈。

部署

包括 Show 权限，并允许用户部署应用程序。例如，应用程序开发人员可能需要将更新部署到堆栈的实例，但不能向堆栈添加层或实例。

Manage

包括 Deploy 权限，并允许用户执行各种堆栈管理工作，其中包括：

- 添加或删除层和实例。

- 使用堆栈的 Permissions 页面将权限级别分配给用户。
- 注册或取消注册资源。

例如，每个堆栈均可能具有指定的管理员，负责确保堆栈具有适当数量和类型的实例、处理软件包和操作系统更新等等。

> **Note**
>
> Manage 级别不允许用户创建或克隆堆栈。必须由附加的 IAM policy 授予这些权限。有关示例，请参阅Manage 权限。

如果用户还有 IAM 策略， AWS OpsWorks Stacks 会评估这两组权限。这将允许您为用户分配权限级别，然后将策略附加到用户以限制或扩充相应级别允许的操作。例如，您可以附加一种策略，该策略允许 Manage 用户创建或克隆堆栈，或拒绝该用户注册或取消注册资源。要获取此类策略的一些示例，请参阅示例策略。

> **Note**
>
> 如果用户的策略允许其他操作，则可能显示结果以覆盖 Permissions 页面设置。例如，如果用户拥有允许该CreateLayer操作的策略，但您使用 “权限” 页面来指定部署权限，则仍允许该用户创建图层。此规则的例外是 “拒绝” 选项，该选项甚至拒绝使用 AWSOpsWorks_FullAccess 策略的用户访问堆栈。有关更多信息，请参阅使用策略控制对 AWS 资源的访问权限。

查看您的权限

如果启用了自我管理，则用户通过选择右上角的 My Settings，可查看其针对每个堆栈的权限级别汇总。如果用户的策略授予DescribeMyUserProfile和UpdateMyUserProfile操作的权限，则用户也可以访问 “我的设置”。

使用 IAM; 条件键来验证临时凭证

AWS OpsWorks Stacks 具有内置的授权层，支持其他授权案例（例如简化个人用户对堆栈的只读或读写访问权限的管理）。此授权层需要使用临时凭证。因此，如 IAM 文档的 JSON 策略元素参考所述，您不能使用 `aws:TokenIssueTime` 条件来验证用户是否正在使用长期凭证，或阻止使用临时凭证的用户执行相应操作。

通过附加 IAM 策略管理 AWS OpsWorks 堆栈权限

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

您可以通过附加 IAM 策略来指定用户的 AWS OpsWorks 堆栈权限。如下这些权限需要附加策略：

- 管理用户权限，例如导入用户。
- 针对某些操作的权限，例如创建或克隆堆栈。

有关需要附加策略的完整操作列表，请参阅AWS OpsWorks 堆栈权限级别。

您还可以使用策略来自定义通过 Permissions 页面授予的权限级别。本节简要概述了如何将 IAM 策略应用于用户以指定 AWS OpsWorks Stacks 权限。有关更多信息，请参阅 AWS 资源访问管理。

IAM policy 是一个 JSON 对象，其中包含一个或多个语句。每个语句元素都有一个权限列表，这些权限拥有它们自己的三个基本元素：

操作

权限所影响的操作。您可以将 AWS OpsWorks Stacks 操作指定为。`opsworks:`*`action`*可将 `Action` 设置为诸如 `opsworks:CreateStack` 等特定操作，该操作可指定是否允许用户调用 `CreateStack`。您还可以使用通配符来指定多组操作。例如，`opsworks:Create*` 可指定所有创建操作。有关 AWS OpsWorks 堆栈操作的完整列表，请参阅 Stack AWS OpsWorks s API 参考。

效果

是允许还是拒绝指定的操作。

资源

权限影响的 AWS 资源。 AWS OpsWorks 堆栈有一种资源类型，即堆栈。要指定特定堆栈资源的权限，请将 `Resource` 设置为堆栈的 ARN，其格式如下：`arn:aws:opsworks:`*`region`*`:`*`account_id`*`:stack/`*`stack_id`*`/`。

还可使用通配符。例如，将 `Resource` 设置为 `*` 可授予针对每种资源的权限。

例如，以下策略会拒绝用户停止 ID 为 2860-2f18b4cb-4de5-4429-a149-ff7da9f0d8ee 的堆栈上的实例。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "opsworks:StopInstance",
      "Effect": "Deny",
      "Resource": "arn:aws:opsworks:*:*:stack/2f18b4cb-4de5-4429-a149-ff7da9f0d8ee/"
    }
  ]
}
```

有关为 IAM 用户添加权限的信息，请参阅 https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users_change-permissions.html#users_change_permissions-add-console。

有关如何创建或修改 IAM 策略的更多信息，请参阅策略和 IAM 中的权限。有关 AWS OpsWorks 堆栈策略的一些示例，请参阅示例策略。

示例策略

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

本节介绍可应用于 AWS OpsWorks Stacks 用户的 IAM 策略示例。

- 管理权限 介绍了可用于向管理用户授予权限的两个策略。
- Manage 权限和 Deploy 权限介绍了可附加到用户以扩充或限制 Manage 和 Deploy 权限级别的策略示例。

  AWS OpsWorks Stacks 通过评估 IAM 策略授予的权限以及 “权限” 页面授予的权限来确定用户的权限。有关更多信息，请参阅使用策略控制对 AWS 资源的访问权限。有关 Permissions 页面权限的更多信息，请参阅AWS OpsWorks 堆栈权限级别。

管理权限

使用 IAM 控制台 https://console.aws.amazon.com/iam/ 访问 AWSOpsWorks_FullAccess 策略，将此策略附加到用户以授予他们执行所有 AWS OpsWorks Stacks 操作的权限。除其他权限外，还需要 IAM 权限，以允许管理用户导入用户。

您必须创建一个 IAM 角色，允许 AWS OpsWorks 堆栈代表您访问其他 AWS 资源，例如 Amazon EC2 实例。通常，您可以通过让管理用户创建第一个堆栈，然后让 AWS OpsWorks Stacks 为您创建角色来处理此任务。然后，您可以针对所有后续堆栈使用该角色。有关更多信息，请参阅 允许 AWS OpsWorks Stacks 代表你行事。

创建第一个堆栈的管理用户必须拥有 AWSOpsWorks_FullAccess 策略中未包含的某些 IAM 操作的权限。为策略中的 `Actions` 节添加以下权限。要获得正确的 JSON 语法，请务必在操作之间添加逗号，并删除操作列表末尾的尾部逗号。

```
"iam:PutRolePolicy",
"iam:AddRoleToInstanceProfile",
"iam:CreateInstanceProfile",
"iam:CreateRole"
```

Manage 权限

Manage 权限级别允许用户执行各种堆栈管理操作，包括添加或删除层。本主题介绍了几种策略，您可以使用这些策略 Manage 用户，以扩充或限制标准权限。

拒绝 Manage 用户添加或删除层

您可以通过使用以下 IAM policy，将 Manage 权限级别限制为允许用户执行除添加或删除层之外的所有 Manage 操作。将*##*、*## ID* 和 *stack_id* 替换为适合您配置的值。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "opsworks:CreateLayer",
        "opsworks:DeleteLayer"
      ],
      "Resource": "arn:aws:opsworks:region:account_id:stack/stack_id/"
    }
  ]
```

```
}
```

允许 Manage 用户创建或克隆堆栈

Manage 权限级别不允许用户创建或克隆堆栈。您可以通过附加以下 IAM policy 来更改 Manage 权限，从而允许用户创建或克隆堆栈。将*##*和*## ID* 替换为适合您配置的值。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:GetRolePolicy",
        "iam:ListRoles",
        "iam:ListInstanceProfiles",
        "iam:ListUsers",
        "opsworks:DescribeUserProfiles",
        "opsworks:CreateUserProfile",
        "opsworks:DeleteUserProfile"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": "arn:aws:opsworks::account_id:stack/*/",
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "opsworks.amazonaws.com"
        }
      }
    }
  ]
}
```

拒绝 Manage 用户注册或取消注册资源

管理权限级别允许用户在堆栈中注册和取消注册 Amazon EBS 和 Elastic IP 地址资源。您可以通过附加以下策略来限制 Manage 权限，从而允许用户执行除注册资源之外的所有 Manage 操作。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "opsworks:RegisterVolume",
        "opsworks:RegisterElasticIp"
      ],
      "Resource": "*"
    }
  ]
}
```

允许 Manage 用户导入用户

管理权限级别不允许用户将用户导入 AWS OpsWorks 堆栈。您可以通过附加以下 IAM policy 来扩充 Manage 权限，从而允许用户导入和删除用户。将*##*和*## ID* 替换为适合您配置的值。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:GetRolePolicy",
        "iam:ListRoles",
        "iam:ListInstanceProfiles",
        "iam:ListUsers",
        "iam:PassRole",
        "opsworks:DescribeUserProfiles",
        "opsworks:CreateUserProfile",
        "opsworks:DeleteUserProfile"
      ],
      "Resource": "arn:aws:iam:region:account_id:user/*",
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "opsworks.amazonaws.com"
        }
      }
    }
  ]
```

```
}
```

Deploy 权限

Deploy 权限级别不允许用户创建或删除应用程序。您可以通过附加以下 IAM policy 来扩充 Deploy 权限，从而允许用户创建和删除应用程序。将*##*、*## ID* 和 *stack_id* 替换为适合您配置的值。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "opsworks:CreateApp",
        "opsworks:DeleteApp"
      ],
      "Resource": "arn:aws:opsworks:region:account_id:stack/stack_id/"
    }
  ]
}
```

AWS OpsWorks 堆栈权限级别

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

本部分列出了 “ AWS OpsWorks 堆栈权限” 页面上的 “显示”、“部署” 和 “管理” 权限级别允许的操作。它还包括只能通过将 IAM policy 附加到用户来授予权限的操作列表。

Show (显式)

Show 级别允许执行 `Describe`*XYZ* 命令，以下情况除外：

```
DescribePermissions
DescribeUserProfiles
```

```
DescribeMyUserProfile
DescribeStackProvisioningParameters
```

如果管理用户已对用户启用自我管理，则 Show 用户还可以使用 DescribeMyUserProfile 和 UpdateMyUserProfile。有关自我管理的更多信息，请参阅编辑用户设置。

部署

除了 Show 级别所允许的操作之外，Deploy 级别还允许以下操作。

```
CreateDeployment
UpdateApp
```

Manage

除了 Deploy 和 Show 级别所允许的操作之外，Manage 级别还允许以下操作。

```
AssignInstance
AssignVolume
AssociateElasticIp
AttachElasticLoadBalancer
CreateApp
CreateInstance
CreateLayer
DeleteApp
DeleteInstance
DeleteLayer
DeleteStack
DeregisterElasticIp
DeregisterInstance
DeregisterRdsDbInstance
DeregisterVolume
DescribePermissions
DetachElasticLoadBalancer
DisassociateElasticIp
GrantAccess
GetHostnameSuggestion
RebootInstance
RegisterElasticIp
RegisterInstance
RegisterRdsDbInstance
RegisterVolume
SetLoadBasedAutoScaling
```

```
SetPermission
SetTimeBasedAutoScaling
StartInstance
StartStack
StopInstance
StopStack
UnassignVolume
UpdateElasticIp
UpdateInstance
UpdateLayer
UpdateRdsDbInstance
UpdateStack
UpdateVolume
```

需要 IAM policy 的权限

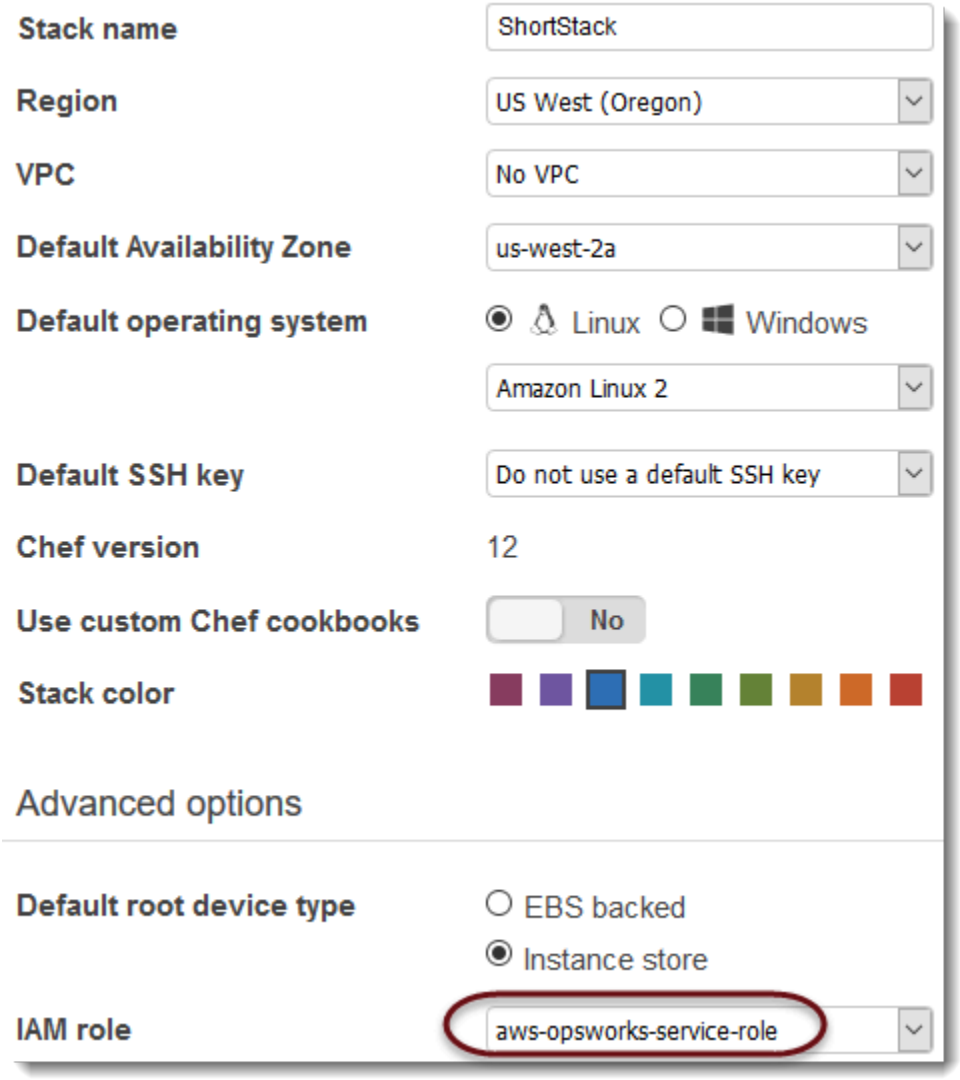
您必须通过将适当的 IAM policy 附加到用户，来授予针对以下操作的权限。要获取一些示例，请参阅示例策略。

```
CloneStack
CreateStack
CreateUserProfile
DeleteUserProfile
DescribeUserProfiles
UpdateUserProfile
```

允许 AWS OpsWorks Stacks 代表你行事

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

AWS OpsWorks Stacks 需要代表您与各种 AWS 服务进行交互。例如， AWS OpsWorks Stacks 与 Amazon EC2 交互以创建实例，与亚马逊交互 CloudWatch 以获取监控统计数据。创建堆栈时，您可以指定一个 IAM 角色（通常称为服务角色），该角色向 AWS OpsWorks Stacks 授予相应的权限。



当您指定一个新堆栈的服务角色时，可以执行下列操作之一：

- 指定您之前创建的标准服务角色。

  您通常可在创建第一个堆栈时创建一个标准服务角色，然后将该角色用于所有后续堆栈。
- 指定您使用 IAM; 控制台或 API 创建的自定义服务角色。

  如果您想向 AWS OpsWorks Stacks授予比标准服务角色更多的有限权限，则此方法非常有用。

> **Note**
>
> 要创建您的第一个堆栈，您必须拥有 IAM AdministratorAccess策略模板中定义的权限。这些权限允许 AWS OpsWorks Stacks 创建新的 IAM 服务角色并允许您导入用户，如前文所述。对于

所有后续堆栈，用户可选择为第一个堆栈创建的服务角色；这些用户无需完整的管理权限便可创建堆栈。

标准服务角色授予下列权限：

- 执行所有 Amazon EC2 操作 (`ec2:*`)。
- 获取 CloudWatch 统计数据 (`cloudwatch:GetMetricStatistics`)。
- 使用 Elastic Load Balancing 将流量分配到服务器（`elasticloadbalancing:*`）。
- 使用 Amazon RDS 实例作为数据库服务器 (`rds:*`)。
- 使用 IAM 角色 (`iam:PassRole`) 在 AWS OpsWorks 堆栈和您的 Amazon EC2 实例之间提供安全的通信。

如果您创建自定义服务角色，则必须确保该角色授予 Stacks 管理 AWS OpsWorks 堆栈所需的所有权限。以下 JSON 示例是标准服务角色的策略声明；自定义服务角色应在其策略声明中至少包含以下权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "ec2:*",
                "iam:PassRole",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:DescribeAlarms",
                "ecs:*",
                "elasticloadbalancing:*",
                "rds:*"
            ],
            "Effect": "Allow",
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "ec2.amazonaws.com"
                }
            }
        }
```

```
    ]
}
```

服务角色还具有信任关系。 AWS OpsWorks Stacks 创建的服务角色具有以下信任关系。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "StsAssumeRole",
      "Effect": "Allow",
      "Principal": {
        "Service": "opsworks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

服务角色必须具有这种信任关系， AWS OpsWorks Stacks 才能代表您行事。如果您使用默认服务角色，请勿修改信任关系。如果您正在创建自定义服务角色，请通过执行以下之一指定信任关系：

- 如果您在 IAM 控制台中使用创建角色向导，请在选择用例中选择 Opsworks。此角色具有相应的信任关系，但未隐含附加任何策略。要授予 AWS OpsWorks Stacks 代表您执行操作的权限，请创建包含以下内容的客户管理型策略，并将其附加到新角色。

  ```
  {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": [
          "cloudwatch:DescribeAlarms",
          "cloudwatch:GetMetricStatistics",
          "ec2:*",
          "ecs:*",
          "elasticloadbalancing:*",
          "iam:GetRolePolicy",
          "iam:ListInstanceProfiles",
          "iam:ListRoles",
          "iam:ListUsers",
  ```

```
        "rds:*"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "ec2.amazonaws.com"
        }
      }
    }
  ]
}
```

- 如果您使用的是 AWS CloudFormation 模板，则可以在模板的 “资源” 部分中添加类似以下内容的内容。

```
"Resources": {
  "OpsWorksServiceRole": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "AssumeRolePolicyDocument": {
            "Statement": [ {
              "Effect": "Allow",
              "Principal": {
                  "Service": [ "opsworks.amazonaws.com" ]
              },
              "Action": [ "sts:AssumeRole" ]
            } ]
        },
        "Path": "/",
        "Policies": [ {
            "PolicyName": "opsworks-service",
            "PolicyDocument": {
              ...
            } ]
```

```
        }
      },
    }
}
```

Stacks 中的 AWS OpsWorks 跨服务混淆了副手预防

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

混淆代理问题是一个安全性问题，即不具有操作执行权限的实体可能会迫使具有更高权限的实体执行该操作。在中 AWS，跨服务模仿可能会导致混乱的副手问题。一个服务（呼叫服务）调用另一项服务（所谓的服务）时，可能会发生跨服务模拟。可以操纵调用服务，使用其权限以在其他情况下该服务不应有访问权限的方式对另一个客户的资源进行操作。为防止这种情况， AWS 提供可帮助您保护所有服务的数据的工具，而这些服务中的服务主体有权限访问账户中的资源。

我们建议在堆栈访问策略中使用`aws:SourceArn`和`aws:SourceAccount`全局条件上下文密钥来限制 Stac AWS OpsWorks ks 向堆栈授予其他服务的权限。如果 `aws:SourceArn` 值不包含账户 ID，例如 Amazon S3 存储桶 ARN，您必须使用两个全局条件上下文密钥来限制权限。如果同时使用全局条件上下文密钥和包含账户 ID 的 `aws:SourceArn` 值，则 `aws:SourceAccount` 值和 `aws:SourceArn` 值中的账户在同一策略语句中使用时，必须使用相同的账户 ID。如果您只希望将一个堆栈与跨服务访问相关联，请使用 `aws:SourceArn`。如果您想允许该账户中的任何堆栈与跨服务使用操作相关联，请使用 `aws:SourceAccount`。

的值`aws:SourceArn`必须是堆栈的 ARN。 AWS OpsWorks

防范混淆代理问题最有效的方法是使用 `aws:SourceArn` 全局条件上下文键和 AWS OpsWorks Stacks 堆栈的完整 ARN。如果不知道完整 ARN，或者正在指定多个堆栈 ARN，请针对 ARN 未知部分使用带有通配符（`*`）的 `aws:SourceArn` 全局上下文条件键。例如，`arn:aws:`*`servicename`*`:*:`*`123456789012`*`:*`。

以下部分展示了如何使用 AWS OpsWorks Stacks 中的`aws:SourceArn`和`aws:SourceAccount`全局条件上下文键来防止出现混淆的副手问题。

防止在 Stacks 中 AWS OpsWorks 混淆副手漏洞

本节介绍如何帮助防止 AWS OpsWorks 堆栈中出现混淆的代理漏洞，并包括可以附加到用于访问 AWS OpsWorks 堆栈的 IAM 角色的权限策略示例。作为安全最佳实践，建议您向 IAM 角色与其他服务的信任关系中添加 `aws:SourceArn` 和 `aws:SourceAccount` 条件键。信任关系允许 AWS OpsWorks Stacks 扮演角色，在其他服务中执行创建或管理 AWS OpsWorks Stacks 堆栈所需的操作。

编辑信任关系来添加 `aws:SourceArn` 和 `aws:SourceAccount` 条件键

1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在左侧导航窗格中，选择 Roles（角色）。
3. 在搜索框中，搜索您用于访问 AWS OpsWorks 堆栈的角色。 AWS 托管角色是`aws-opsworks-service-role`。
4. 在角色的 摘要 页面上，选择 信任关系 选项卡。
5. 在 信任关系 选项卡上选择 编辑信任策略。
6. 在编辑信任策略页面上，向策略中添加至少一个 `aws:SourceArn` 或 `aws:SourceAccount` 条件键。用于`aws:SourceArn`将跨服务（例如 Amazon EC2）和 AWS OpsWorks 堆栈之间的信任关系限制为特定的堆 AWS OpsWorks 栈堆栈，限制性更强。`aws:SourceAccount`添加后，将跨服务与 AWS OpsWorks Stacks 之间的信任关系限制为特定账户中的堆栈，限制较少。示例如下：请注意，如果您同时使用两个条件键，则账户 ID 必须相同。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "opsworks.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnEquals": {
          "arn:aws:opsworks:us-east-2:123456789012:stack/
EXAMPLEd-5699-40a3-80c3-22c32EXAMPLE/"
        }
```

```
      }
    }
  ]
}
```

7. 添加条件键后，选择更新策略。

以下是使用 aws:SourceArn 和 aws:SourceAccount 来限制堆栈访问的其他角色示例。

主题

- 示例：在特定区域访问堆栈
- 示例：向 aws:SourceArn 添加多个堆栈 ARN

示例：在特定区域访问堆栈

以下角色信任关系声明访问美国东部（俄亥俄州）区域中的任何 AWS OpsWorks Stacks 堆栈 ()。us-east-2请注意，该区域在 aws:SourceArn 的 ARN 值中指定，但堆栈 ID 值是通配符 (*)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "opsworks.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:opsworks:us-east-2:123456789012:stack/*"
        }
      }
    }
  ]
}
```

示例：向 **aws:SourceArn** 添加多个堆栈 ARN

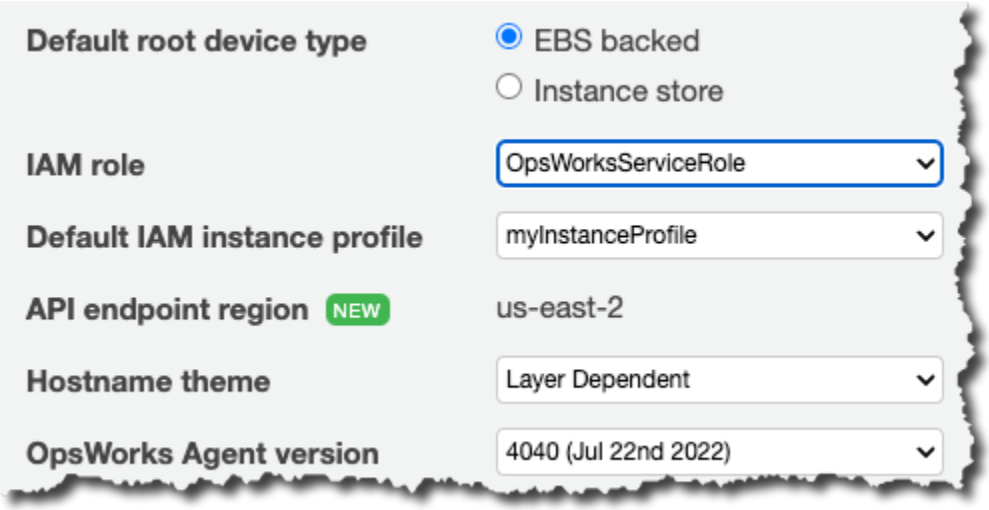
以下示例限制对账户 ID 为 123456789012 的两个 AWS OpsWorks 堆栈组成的数组的访问权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "opsworks.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnEquals": {
          "aws:SourceArn": [
             "arn:aws:opsworks:us-east-2:123456789012:stack/unique_ID1",
             "arn:aws:opsworks:us-east-2:123456789012:stack/unique_ID2"
           ]
       }
      }
    }
  ]
}
```

为在 EC2 实例上运行的应用程序指定权限

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

如果在您的堆栈的 Amazon EC2 实例上运行的应用程序需要访问其他 AWS 资源，如 Amazon EC2 存储桶，则它们必须具有适当的权限。要授予这些权限，您可使用实例配置文件。在创建 Stac AWS OpsWorks ks 堆栈时，您可以为每个实例指定实例配置文件。



您还可以通过编辑层配置来为层的实例指定配置文件。

实例配置文件指定一个 IAM 角色。在实例上运行的应用程序可担任该角色以访问 AWS 资源，具体取决于该角色的策略授予的权限。有关应用程序如何担任角色的更多信息，请参阅使用 API 调用担任角色。

您可以采用以下任一方式创建实例配置文件：

- 使用 IAM; 控制台或 API 创建配置文件。

  有关更多信息，请参阅角色（委托和联合）。
- 使用 AWS CloudFormation 模板创建个人资料。

  有关如何在模板中包含 IAM; 资源的一些示例，请参阅 Identity and Access Management (IAM) 模板代码段。

一个实例配置文件必须具有一种信任关系和一个授予访问 AWS 资源的权限的附加策略。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
```

```
    }
  ]
}
```

实例配置文件必须具有这种信任关系， AWS OpsWorks Stacks 才能代表您采取行动。如果您使用默认服务角色，请勿修改信任关系。如果您正在创建自定义服务角色，请指定信任关系，如下所示：

- 如果您是在 IAM 控制台中使用 Create Role 向导，请在此向导的第二页上的 AWS Service Roles 下指定 Amazon EC2 角色类型。
- 如果您使用的是 AWS CloudFormation 模板，则可以在模板的 “资源” 部分中添加类似以下内容的内容。

```
"Resources": {
      "OpsWorksEC2Role": {
         "Type": "AWS::IAM::Role",
         "Properties": {
            "AssumeRolePolicyDocument": {
               "Statement": [ {
                  "Effect": "Allow",
                  "Principal": {
                     "Service": [ "ec2.amazonaws.com" ]
                  },
                  "Action": [ "sts:AssumeRole" ]
               } ]
            },
            "Path": "/"
         }
      },
      "RootInstanceProfile": {
         "Type": "AWS::IAM::InstanceProfile",
         "Properties": {
            "Path": "/",
            "Roles": [ {
               "Ref": "OpsWorksEC2Role"
            }
            ]
         }
   }
}
```

当您创建您的实例配置文件时，您可将适当的策略附加到该配置文件的角色。在创建堆栈后，您必须使用 IAM; 控制台或 API 来将适当的策略附加到配置文件的角色。例如，以下策略授予对名为 DOC-EXAMPLE-BUCKET 的 Amazon S3 存储桶中的所有对象的完全访问权限。将##和 DOC-EXAMPLE-BUCKET 替换为适合您的配置的值。

```
{
  "Version": "2012-10-17",
  "Statement": [ {
    "Effect": "Allow",
    "Action": "s3:*",
    "Resource": "arn:aws:s3:region::DOC-EXAMPLE-BUCKET/*"
    }
  ]
}
```

有关如何创建和使用实例配置文件的示例，请参阅使用 Amazon S3 存储桶。

如果您的应用程序使用实例配置文件从 EC2 实例调用 AWS OpsWorks Stacks API，则除了对 AWS OpsWorks 堆栈和其他 AWS 服务执行相应操作外，策略还必须允许该操作。iam:PassRoleiam:PassRole 权限允许 AWS OpsWorks Stacks 代表您担任该服务角色。有关 AWS OpsWorks 堆栈 API 的更多信息，请参阅 AWS OpsWorks API 参考。

以下是 IAM 策略的示例，该策略允许您从 EC2 实例调用任何 AWS OpsWorks 堆栈操作以及任何 Amazon EC2 或 Amazon S3 操作。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:*",
        "s3:*",
        "opsworks:*",
        "iam:PassRole"
      ],
      "Resource": "arn:aws:ec2:region:account_id:instance/*",
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "opsworks.amazonaws.com"
        }
      }
```

```
    }
  ]
}
```

> **Note**
>
> 如果您不允许`iam:PassRole`，则任何调用 AWS OpsWorks Stacks 操作的尝试都会失败，并出现如下错误：
>
> ```
> User: arn:aws:sts::123456789012:federated-user/Bob is not authorized
> to perform: iam:PassRole on resource:
> arn:aws:sts::123456789012:role/OpsWorksStackIamRole
> ```

有关在 EC2 实例上使用角色以获取权限的更多信息，请参阅 https://docs.aws.amazon.com/IAM/latest/UserGuide/role-usecase-ec2app.html 用户指南 中的AWS Identity and Access Management 向在 Amazon EC2 实例上运行的应用程序授予访问 AWS 资源的权限。

管理 SSH 访问

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

AWS OpsWorks 堆栈支持 Linux 和 Windows 堆栈的 SSH 密钥。

- 对于 Linux 实例，您可以使用 SSH 登录到实例以运行代理 CLI 命令。

  有关更多信息，请参阅 使用 SSH 登录。
- 对于 Windows 实例，您可以使用 SSH 密钥来获取实例的管理员密码，您随后可使用该密码通过 RDP 进行登录。

  有关更多信息，请参阅 使用 RDP 登录。

身份验证基于 SSH 密钥对，该密钥对包含公有密钥和私有密钥：

- 您可以在实例上安装公有密钥。

  位置取决于特定的操作系统，但 AWS OpsWorks Stacks会为您处理细节。
- 您在本地存储私有密钥并将其提供给 SSH 客户端 (如 `ssh.exe`) 以访问该实例。

  SSH 客户端使用私有密钥连接到该实例。

要向堆栈用户提供 SSH 访问权，您需要通过某种方式创建 SSH 密钥对、在堆栈的实例上安装公有密钥并安全地管理私有密钥。

Amazon EC2 提供了一种简单的方式，可在实例上安装公有 SSH 密钥。您可以使用 Amazon EC2 控制台或 API 为计划使用的每个 Amazon Web Services Region 创建一个或多个密钥对。Amazon EC2 将在 AWS 上存储公有密钥，并且您将本地存储私有密钥。在启动实例时，您可以指定区域的一个密钥对，Amazon EC2 将自动在实例上安装该密钥对。然后，您可以使用相应的私有密钥来登录该实例。有关更多信息，请参阅 Amazon EC2 密钥对。

使用 AWS OpsWorks Stacks，您可以在创建堆栈时指定该地区的 Amazon EC2 密钥对之一，也可以选择在创建每个实例时使用不同的密钥对覆盖该密钥对。当 AWS OpsWorks Stacks 启动相应的 Amazon EC2 实例时，它会指定密钥对，并且 Amazon EC2 会在实例上安装公有密钥。然后，您可以使用私有密钥登录或检索管理员密码，就像您对标准 Amazon EC2 实例所做的一样。有关更多信息，请参阅 安装 Amazon EC2 密钥。

使用 Amazon EC2 密钥对很方便，但有两个重要限制：

- Amazon EC2 密钥对与特定的 Amazon Web Services Region 关联。

  如果您在多个区域中工作，则必须管理多个密钥对。
- 您只能在一个实例上安装一个 Amazon EC2 密钥对。

  如果您想允许多个用户登录，这些用户必须具有公有密钥的副本，这不是推荐的安全实践。

对于 Linux 堆栈， AWS OpsWorks Stacks 提供了一种更简单、更灵活的方式来管理 SSH 密钥对。

- 每个用户均注册一个个人密钥对。

  它们将私钥存储在本地并向 AWS OpsWorks Stacks 注册公钥，如中所述。注册用户的公有 SSH 密钥
- 当您设置堆栈的用户权限时，可以指定哪些用户应具有对堆栈实例的 SSH 访问权。

AWS OpsWorks Stacks 会自动在堆栈的实例上为每个授权用户创建一个系统用户，并安装他们的公钥。然后，该用户可以使用相应的私有密钥进行登录，如使用 SSH 登录中所述。

使用个人 SSH 密钥具有以下优势。

- 无需在实例上手动配置密钥； AWS OpsWorks Stacks 会自动在每个实例上安装相应的公钥。
- AWS OpsWorks Stacks 仅安装授权用户的个人公钥。

  未授权用户无法使用其个人私有密钥来获取对实例的访问权。利用 Amazon EC2 密钥对，任何具有相应私有密钥的用户均可登录，无论其是否具有授权的 SSH 访问权。
- 如果用户不再需要 SSH 访问权，则可使用 Permissions 页撤销用户的 SSH/RDP 权限。

  AWS OpsWorks 堆栈会立即从堆栈的实例中卸载公钥。
- 您可以对任意 Amazon Web Services Region 使用相同的密钥。

  用户必须仅管理一个私有密钥。
- 无需共享私有密钥。

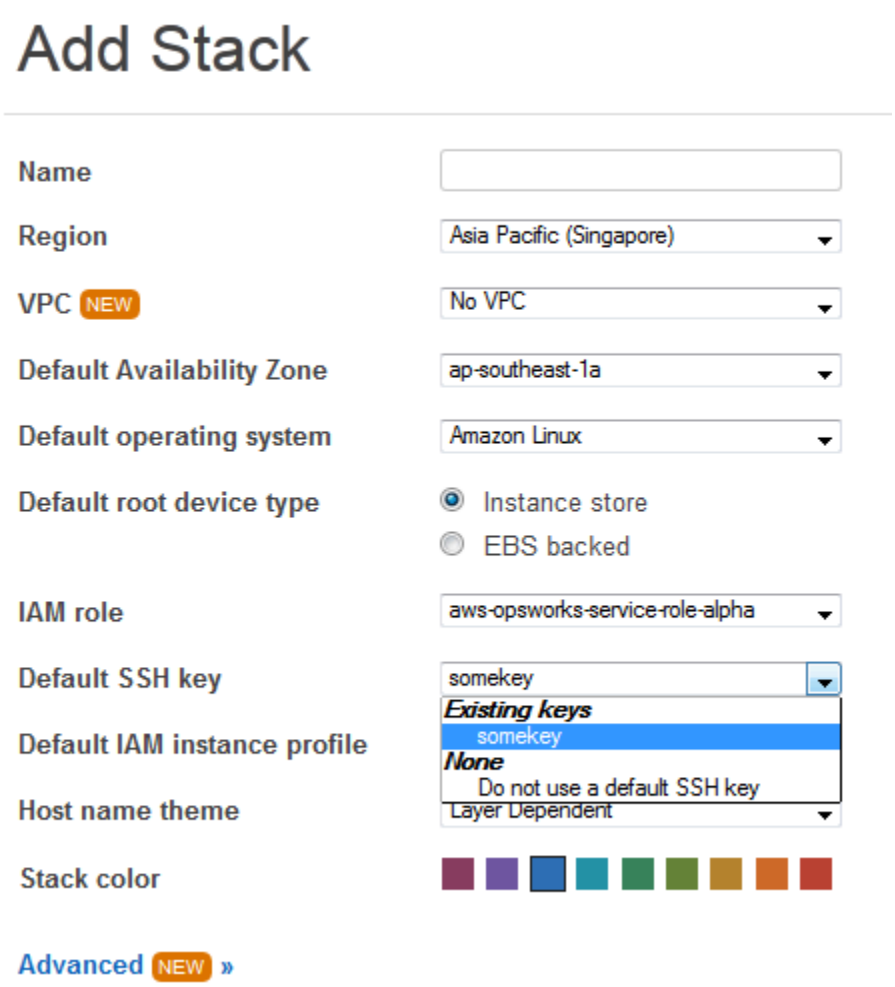
  每个用户均有其自己的私有密钥。
- 可轻松轮换密钥。

  您或用户更新 My Settings (我的设置) 中的公有密钥，而 AWS OpsWorks Stacks 将自动更新实例。

安装 Amazon EC2 密钥

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

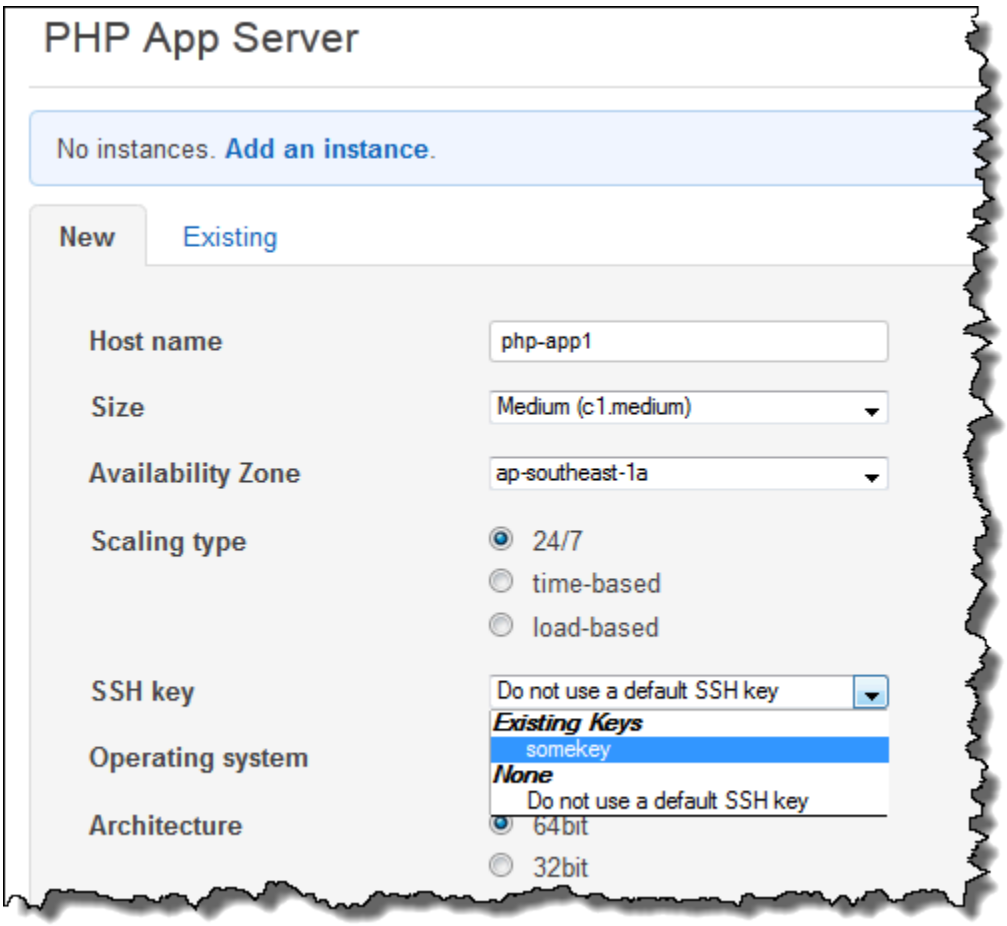
当您创建堆栈时，可以指定默认情况下安装在堆栈中的每个实例上的 Amazon EC2 SSH 密钥。



默认 SSH 密钥列表显示 Amazon Web Services account 的 Amazon EC2 密钥。您可以执行以下操作之一：

- 从该列表中选择适当的密钥。
- 选择 Do not use a default SSH key 可指定不使用密钥。

如果您选择了 Do not use a default SSH key 或需要覆盖堆栈的默认密钥，则可在创建实例时指定密钥。



启动实例时，AWS OpsWorks Stacks 会在`authorized_keys`文件中安装公钥。

注册用户的公有 SSH 密钥

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

可通过两种方法注册用户的公有 SSH 密钥：

- 管理用户可以将一个公有 SSH 密钥分配给一个或多个用户，并为这些用户提供相应的私有密钥。
- 管理用户可以为一个或多个用户启用自管理功能。

  随后，这些用户可以指定其自己的公有 SSH 密钥。

有关管理用户如何启用自管理功能或向用户分配公有密钥的更多信息，请参阅编辑用户设置。

通过在 PuTTY 终端中使用 SSH 连接到基于 Linux 的实例时，需要执行额外的步骤。有关更多信息，请参阅 AWS 文档中的使用 PuTTY 从 Windows 连接到您的 Linux 实例和排查实例的连接问题。

下面介绍了已启用自管理的用户如何指定其公有密钥。

指定您的 SSH 公有密钥

1. 创建 SSH 密钥对。

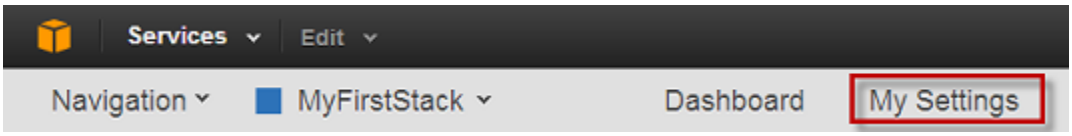
   最简单的方法是在本地生成密钥对。有关更多信息，请参阅如何生成您自己的密钥并将其导入 Amazon EC2 中。

   > **Note**
   >
   > 如果你使用 Puttygen 生成密钥对，请将公钥中的公钥复制粘贴到 OpenSSH authorized_keys 文件框中。单击 “保存公钥” 会以不支持的格式保存公钥 MindTerm。

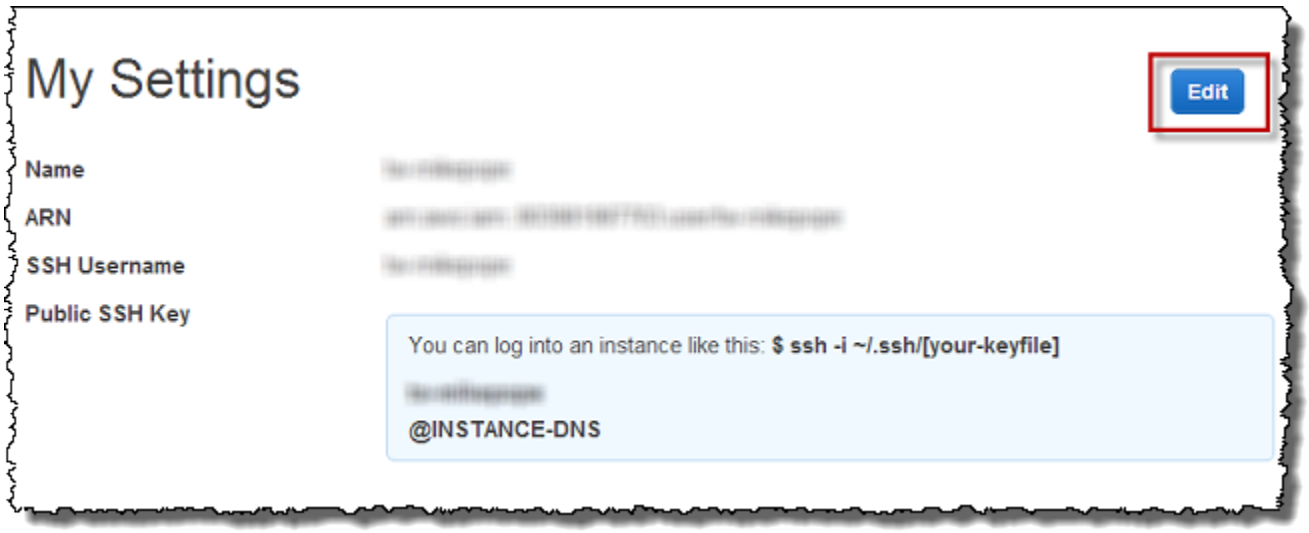
2. 以启用自我管理的 IAM 用户身份登录 AWS OpsWorks Stacks 控制台。

   > **Important**
   >
   > 如果您以账户所有者或未启用自我管理功能的 IAM 用户身份登录， AWS OpsWorks Stacks 不会显示 “我的设置”。如果您是管理用户或账户所有者，则可通过转至 Users 页并编辑用户设置来改为指定 SSH 密钥。

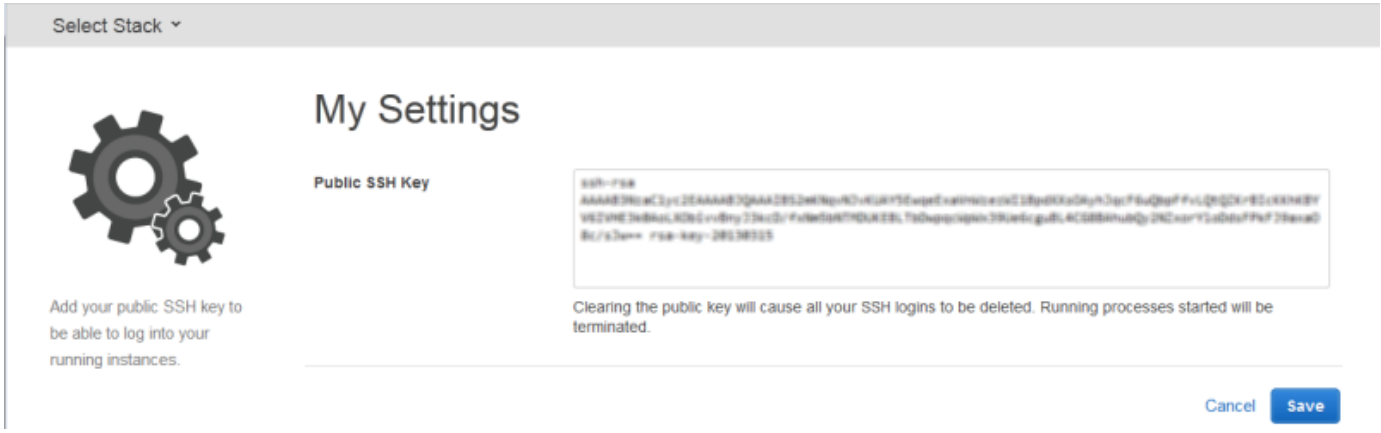
3. 选择 我的设置，这将显示已登录用户的设置。

4. 在 My Settings 页上，单击 Edit。



5. 在 Public SSH Key box 中，输入您的 SSH 公有密钥，然后单击 Save。



> **Important**
>
> 要使用内置的 MindTerm SSH 客户端连接到 Amazon EC2 实例，用户必须以 IAM 用户身份登录并在 AWS OpsWorks 堆栈中注册了 SSH 公钥。有关更多信息，请参阅 使用内置 MindTerm SSH 客户端。

管理 Linux 安全更新

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

安全更新

Linux 操作系统提供程序提供定期更新，大部分更新是操作系统安全修补，但也可能包括对已安装程序包的更新。您应确保您的实例操作系统是最新的操作系统并具有最新的安全修补。

默认情况下，在实例完成启动后， AWS OpsWorks Stacks 会在安装过程中自动安装最新的更新。AWS OpsWorks 为了避免诸如重启应用程序服务器之类的中断，Stacks 不会在实例联机后自动安装更新。由您自行管理联机实例的更新，因此可最大程度地减少中断。

我们建议您使用下列方法之一来更新您的联机实例。

- 创建并启动新实例来替换您当前的联机实例。然后，删除当前实例。

  新实例将具有在设置过程中安装的最新安全修补集。
- 在 Chef 11.10 或更早版本的堆栈中的基于 Linux 的实例上，运行 Update Dependencies 堆栈命令，以在指定实例上安装最新的安全修补集和其他更新。

对于这两种方法， AWS OpsWorks Stacks都通过在亚马逊Linux和红帽企业Linux（RHEL）或 `apt-get update` Ubuntu上运行`yum update`来执行更新。每次分发处理更新的方式略有不同，因此您应检查相关链接中的信息以准确了解更新将如何影响您的实例：

- Amazon Linux- Amazon Linux 更新将安装安全补丁并且还可能安装功能更新（包括程序包更新）。

  有关更多信息，请参阅 Amazon Linux AMI 常见问题。
- Ubuntu -更新主要限于安装安全修补，但可能还会针对数量有限的关键修复安装程序包更新。

  有关更多信息，请参阅 LTS - Ubuntu Wiki。
- CentOS - CentOS 更新通常会保留与早期版本的二进制兼容性。
- RHEL - RHEL 更新通常会保留与早期版本的二进制兼容性。

有关更多信息，请参阅 Red Hat Enterprise Linux 生命周期。

如果您想更好地控制更新，例如指定特定的软件包版本，则可以使用、、或UpdateLayer操作（或等效的 AWS 开发工具包方法或 AWS CLI 命令）来禁用自动更新，将参数设置为。CreateInstanceUpdateInstanceCreateLayerInstallUpdatesOnBootfalse以下示例演示了如何使用 AWS CLI 来禁用作为现有层默认设置的 InstallUpdatesOnBoot。

```
aws opsworks update-layer --layer-id layer ID --no-install-updates-on-boot
```

之后，您必须自行管理更新。例如，您可使用下列策略之一：

- 实现一个自定义配方以运行适当 shell 命令来安装您的首选更新。

  由于系统更新不会自然地映射到生命周期事件，因此应在您的自定义说明书中包含该配方，但手动执行。有关程序包更新，您还可使用 yum_package (Amazon Linux) 或 apt_package (Ubuntu) 资源来代替 shell 命令。
- 使用 SSH 登录每个实例并手动运行适当的命令。

使用安全组

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

安全组

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

每个 Amazon EC2 实例都有一个或多个关联的安全组，用于管理实例的网络流量，这与防火墙的功能类似。安全组有一个或多个规则，每个规则指定特定类别允许的流量。规则指定以下内容：

- 允许的流量类型，如 SSH 或 HTTP
- 流量的协议，如 TCP 或 UDP
- 流量可能源自的 IP 地址范围
- 流量允许的端口范围

安全组有两种类型的规则：

- 入站规则管理入站网络流量。

  例如，应用程序服务器实例通常有一个允许来自任何 IP 地址的入站 HTTP 流量至端口 80 的入站规则，以及另一个允许来自指定 IP 地址组的入站 SSH 流量至端口 22 的入站规则。
- 出站规则管理出站网络流量。

  常见做法是使用默认设置，这将允许所有出站流量。

有关安全组的更多信息，请参阅中Amazon EC2 安全组。

首次在某个区域中创建堆栈时， AWS OpsWorks Stacks 会使用一组适当的规则为每个层创建一个内置安全组。所有组均拥有默认出站规则，这些规则允许所有出站流量。一般情况下，入站规则允许以下流量：

- 来自相应 AWS OpsWorks 堆栈层的入站 TCP、UDP 和 ICMP 流量
- 端口 22 (SSH 登录) 上的入站 TCP 流量

> **Warning**
>
> 默认安全组配置打开 SSH (端口 22) 以连接到任何网络位置 (0.0.0.0/0)。这样，所有 IP 地址均可使用 SSH 访问您的实例。对于生产环境，您必须使用仅允许来自特定 IP 地址或地址范围的 SSH 访问的配置。您可以在创建后立即更新默认安全组，或使用自定义安全组。

- 对于 Web 服务器层，到端口 80 (HTTP) 和 443 (HTTPS) 的所有入站 TCP 和 UDP 流量

> **Note**
>
> 内置的 `AWS-OpsWorks-RDP-Server` 安全组被分配到所有 Windows 实例，以允许 RDP 访问。但默认情况下，它不具有任何规则。如果您运行的是 Windows 堆栈并希望使用 RDP 访问实例，则必须添加一条允许 RDP 访问的入站规则。有关更多信息，请参阅 使用 RDP 登录。

要查看每个组的详细信息，请转到 Amazon EC2 控制台，在导航窗格中选择 Security Groups，然后选择相应层的安全组。例如，AWS OpsWorks--Default-Server 是所有堆栈的默认内置安全组，而 A WS OpsWorks-WebApp 是 Chef 12 示例堆栈的默认内置安全组。

> **Note**
>
> 如果您不小心删除了 AWS OpsWorks Stacks 安全组，则重新创建该安全组的首选方法是让 AWS OpsWorks Stacks 为您执行任务。只需在同一 AWS 区域和 VPC（如果有）中创建一个新堆栈， AWS OpsWorks Stacks 就会自动重新创建所有内置安全组，包括您删除的安全组。如果您不再需要该堆栈，则随后您可以删除它；安全组将保留。如果您要手动重新创建安全组，则它必须是原始安全组的精确副本，包括组名称的大写形式。
> 此外，如果出现以下任一情况， AWS OpsWorks Stacks 将尝试重新创建所有内置安全组：
>
> - 您可以在 Stacks 控制台中对堆栈的设置页面进行 AWS OpsWorks 任何更改。
> - 您启动了其中一个堆栈实例。
> - 您创建了新堆栈。

您可以使用以下任一方法来指定安全组。创建堆栈时，您可以使用 “使用 OpsWorks 安全组” 设置来指定您的首选项。

- 是（默认设置）— AWS OpsWorks Stacks 会自动将相应的内置安全组与每个层关联起来。

  通过使用您的首选设置添加自定义安全组，您可以调整层的内置安全组。但是，当 Amazon EC2 评估多个安全组时，系统会使用严格程度最低的规则，因此您不能使用这种方法来指定比内置组更严格的规则。
- 否 — AWS OpsWorks Stacks 不会将内置安全组与层关联。

  您必须创建相应的安全组，并至少将一个安全组与您创建的每个层关联。使用这种方法可指定比内置组更严格的规则。请注意，如果您愿意，也可以手动关联内置安全组和层；只有需要自定义设置的层才必须要自定义安全组。

> **Important**
>
> 如果您使用内置安全组，则无法通过手动修改该组的设置来创建更严格的规则。每次创建堆栈时，Stac AWS OpsWorks ks 都会覆盖内置安全组的配置，因此您所做的任何更改都将在下次创建堆栈时丢失。如果某个层需要比内置安全组更严格的安全组设置，请将 “使用 OpsWorks 安全组” 设置为 “否”，使用您的首选设置创建自定义安全组，并在创建时将其分配给图层。

AWS OpsWorks Stacks 对 Chef 12 Linux 的支持

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

本节简要概述了 Chef 12 Linux 的 AWS OpsWorks 堆栈。有关 Windows 上的 Chef 12 的信息，请参阅入门：Windows。有关 Linux 上的早期 Chef 版本的信息，请参阅适用于 Linux 的 Chef 11.10 及早期版本。

概述

AWS OpsWorks Stacks 支持 Chef 12（最新版本的 Chef）适用于 Linux 堆栈。有关更多信息，请参阅了解 Chef。

AWS OpsWorks Stacks 继续支持 Chef 11.10 的 Linux 堆栈。不过，如果您是高级 Chef 用户，并且您想从大量可选的社区说明书中获益，或编写您自己的自定义说明书，建议您使用 Chef 12。相对于 Chef 11.10 以及早期 Linux 堆栈，Chef 12 堆栈提供了以下优势：

- 两个单独的 Chef 运行 ——在实例上执行命令时， AWS OpsWorks Stacks 代理现在会执行两次隔离的 Chef 运行：一次运行用于将实例与其他 AWS 服务 AWS Identity and Access Management（例如 (IAM)）集成的任务，另一次运行您的自定义食谱。第一次 Chef 运行会在实例上安装 AWS OpsWorks Stacks 代理并执行系统任务，例如用户设置和管理、卷设置和配置、 CloudWatch 指标配置等。第二个运行专用于运行AWS OpsWorks 堆栈生命周期事件的自定义配方。利用第二个运行，您可以使用自己的 Chef 说明书或社区说明书。
- 解决命名空间冲突 – 在 Chef 12 之前， AWS OpsWorks Stacks 在共享环境中执行系统任务并运行内置配方和自定义配方。这导致命名空间冲突以及对 AWS OpsWorks Stacks 运行了哪些配方不清

楚。必须手动覆盖不需要的默认配置，这是一项耗时且容易出错的任务。在 Linux 版 Chef 12 中，AWS OpsWorks Stacks 不再支持适用于 PHP、Node.js 或 Rails 等应用程序服务器环境的内置 Chef 食谱。通过消除内置配方， AWS OpsWorks Stacks 消除了内置配方和自定义配方之间命名冲突的问题。

- 对厨师社区食谱的大力支持 — AWS OpsWorks Stacks Chef 12 Linux 为厨师超市的社区食谱提供了更好的兼容性和支持。现在，您可以使用优于 AWS OpsWorks Stacks 之前提供的内置食谱的社区食谱，这些食谱专为在最新的应用程序服务器环境和框架中使用而设计。您可以运行大多数说明书而无需修改 Chef 12 for Linux。欲了解更多信息，请访问 Learn Chef 网站的 Chef Supermarket、Chef Super market 网站和上GitHub的 Chef Cookbooks 存储库。
- 及时更新 Chef 12 —— AWS OpsWorks Stacks 将在每次 Chef 发布后不久将其厨师环境更新到最新的 Chef 12 版本。在 Chef 12 中，Chef 的次要更新和新的 AWS OpsWorks Stacks 代理版本将同时发布。这可让您直接测试新的 Chef 版本，并使您的 Chef 配方和应用程序能够利用最新的 Chef 功能。

有关 Chef 12 之前的受支持的 Chef 版本的更多信息，请参阅适用于 Linux 的 Chef 11.10 及早期版本。

移至 Chef 12

与之前的 Chef 版本 11.10、11.4 和 0.9 的支持相比，Chef 12 Linux 的关键 AWS OpsWorks 堆栈变化如下：

- 对于适用于 Linux 堆栈的 Chef 12，不再提供或支持内置层。由于仅执行您的自定义配方，因此，删除此支持将使实例的设置方式变得完全透明，并使自定义说明书更易于编写和维护。例如，不再需要覆盖内置 AWS OpsWorks Stacks 配方的属性。移除内置层还可以让 AWS OpsWorks Stacks 更好地支持由 Chef 社区开发和维护的食谱，这样你就可以充分利用它们。Chef 12 for Linux 中不再提供的内置层类型包括：AWS Flow (Ruby)、Ganglia、HAProxy、Java App Server、Memcached、MySQL、Node.js App Server、PHP App Server、Rails App Server 和 Static Web Server。
  - 由于 AWS OpsWorks Stacks 正在运行您提供的食谱，因此不再需要通过运行自定义食谱来覆盖内置的 AWS OpsWorks Stacks 属性。要覆盖您自己的配方或社区配方中的属性，请按照 Chef 12 文档中的关于属性中的说明和示例执行操作。
- AWS OpsWorks Stacks 继续为 Chef 12 Linux 堆栈的以下层提供支持：
  - 自定义层
  - Amazon RDS 服务层
  - ECS 集群层

- Chef 12 Linux 的堆栈配置和数据包已更改，与 Chef 12.2 Windows 的对应项非常相似。这可让您更轻松地查询和分析这些数据包并排查其问题，尤其是在您使用具有不同的操作系统类型的堆栈时。请注意， AWS OpsWorks Stacks 不支持加密的数据包。要以加密形式存储敏感数据，例如密码或证书，我们建议将其存储在私有 S3 存储桶中。然后，您可以创建一个使用适用于 Ruby 的 Amazon SDK 来检索数据的自定义配方。有关示例，请参阅使用适用于 Ruby 的 SDK。有关更多信息，请参阅AWS OpsWorks 堆栈数据包参考。
- 在 Chef 12 Linux 中，Berkshelf 不再安装在堆栈实例上。相反，建议您在本地开发计算机上使用 Berkshelf，在本地打包说明书依赖项。然后，将您的程序包 (包含依赖项) 上传到 Amazon Simple Storage Service。最后，修改您的 Chef 12 Linux 堆栈，将已上传的程序包用作说明书源。有关更多信息，请参阅 本地打包说明书依赖项。
- 不再支持 EBS 卷的 RAID 配置。要提高性能，您可以使用 Amazon Elastic Block Store (Amazon EBS) 的预配置 IOPS。
- 不再支持 autofs。
- 不再支持 Subversion 存储库。
- 现在，必须使用自定义配方执行每层 OS 程序包安装。有关更多信息，请参阅 按层程序包安装。

受支持的操作系统

早期版本的 Chef 所支持的 Linux 操作系统也受 Chef 12 的支持。有关 Chef 12 Linux 堆栈可使用的 Linux 操作系统类型和版本的列表，请参阅Linux 操作系统：。

支持的实例类型

AWS OpsWorks 堆栈支持 Chef 12 Linux 堆栈的所有实例类型，但高性能计算 (HPC) 集群计算、集群 GPU 和高内存集群实例类型等特殊实例类型除外。

更多信息

要了解有关如何使用适用于 Linux 堆栈的 Chef 12 的更多信息，请参阅以下内容：

- 入门：示例

  通过指导你使用 AWS OpsWorks Stacks 控制台进行简短的动手练习，创建 Node.js 应用程序环境，向你介绍 AWS OpsWorks Stacks。
- 入门：Linux

通过指导你使用 AWS OpsWorks Stacks 控制台进行动手练习，向你介绍 AWS OpsWorks Stacks 和 Chef 12 Linux，创建一个基本的 Chef 12 Linux 堆栈，该堆栈包含一个带有提供流量的 Node.js 应用程序的简单层。

- 自定义层

  提供有关将包含说明书和配方的层添加到 Chef 12 Linux 堆栈的准则。您可以使用社区所提供的可用说明书和配方，也可以创建您自己的说明书和配方。

- 移至数据包

  将由运行 Chef 11 及早期版本的 Chef 的 Linux 堆栈使用的实例 JSON 与由运行 Chef 12 的 Linux 堆栈使用的实例 JSON 进行比较。还提供了 Chef 12 实例 JSON 格式的参考文档的暗示。

将堆栈设置从属性移至数据包

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

AWS OpsWorks Stacks 为你的 Chef 食谱提供了各种各样的堆栈设置。这些堆栈设置包括如下值：

- 堆栈说明书源 URL
- 层卷配置
- 实例主机名称
- Elastic Load Balancing DNS 名称
- 应用程序源 URL
- 用户名

引用配方中的堆栈设置使配方代码更可靠且比直接在配方中对堆栈设置进行硬编码更不容易出错。本主题描述如何访问这些堆栈设置以及如何将它们从适用于 Linux 的 Chef 11.10 及更早版本的属性移至 Chef 12 Linux 中的数据包。

在适用于 Linux 的 Chef 11.10 及更早版本中，堆栈设置作为 Chef 属性提供并且通过 Chef node 对象或 Chef 搜索访问。这些属性存储在 AWS OpsWorks Stacks 实例上的`/var/lib/aws/opsworks/chef`目录中一组 JSON 文件中。有关更多信息，请参阅 堆栈配置和部署属性：Linux。

在 Chef 12 Linux 中，堆栈设置作为 Chef 数据包提供并且仅可通过 Chef 搜索访问。数据袋存储在 AWS OpsWorks Stacks 实例上的`/var/chef/runs/`*`run-ID`*`/data_bags`目录中一组 JSON 文件中，其中 *`run-ID`* 是 AWS OpsWorks Stacks 分配给在实例上运行的每个 Chef 的唯一 ID。堆栈设置不再作为 Chef 属性提供，因此堆栈设置不再可通过 Chef node 对象访问。有关更多信息，请参阅 AWS OpsWorks 堆栈数据包参考。

例如，在适用于 Linux 的 Chef 11.10 及更早版本中，以下配方代码使用 Chef node 对象获取表示应用程序的短名称和源 URL 的属性。然后它使用 Chef 日志写入这两个属性值：

```
Chef::Log.info ("********** The app's short name is '#{node['opsworks']
['applications'].first['slug_name']}' **********")
Chef::Log.info("********** The app's URL is '#{node['deploy']['simplephpapp']['scm']
['repository']}' **********")
```

在 Chef 12 Linux 中，以下配方代码使用 `aws_opsworks_app` 搜索索引来获取 `aws_opsworks_app` 数据包中的第一个数据包项目的内容。然后，此代码将两条消息写入到 Chef 日志，一条消息包含应用程序的短名称数据包内容，另一条消息包含应用程序的源 URL 数据包内容：

```
app = search("aws_opsworks_app").first

Chef::Log.info("********** The app's short name is '#{app['shortname']}' **********")
Chef::Log.info("********** The app's URL is '#{app['app_source']['url']}' **********")
```

要将访问堆栈设置的配方代码从适用于 Linux 的 Chef 11.10 及早期版本迁移到 Chef 12 Linux，您必须修订您的代码，以便：

- 访问 Chef 数据包而不是 Chef 属性。
- 使用 Chef 搜索而不是 Chef node 对象。
- 使用 AWS OpsWorks 堆栈数据包名称（例如）`aws_opsworks_app`，而不是使用 AWS OpsWorks 堆栈属性名称，例如`opsworks`和。`deploy`

有关更多信息，请参阅AWS OpsWorks 堆栈数据包参考。

Support 以 AWS OpsWorks 堆栈形式支持以前的 Chef 版本

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

本节简要概述了以前的 Chef 版本的 AWS OpsWorks Stacks 文档。

适用于 Linux 的 Chef 11.10 及早期版本

提供有关 Lin AWS OpsWorks ux 堆栈对 Chef 11.10、11.4 和 0.9 的堆栈支持的文档。

适用于 Linux 的 Chef 11.10 及早期版本

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

本节简要概述了 Linux 版 Chef 11.10、11.4 和 0.9 的 AWS OpsWorks Stacks 文档。

Chef 11 Linux 堆栈入门

示范如何创建一个简单但功能完善的 PHP 应用程序服务器堆栈。

创建您的第一个 Node.js 堆栈

介绍如何创建支持 Node.js 应用程序服务器的 Linux 堆栈以及如何部署简单的应用程序。

自定义堆栈 AWS OpsWorks

描述如何自定义 AWS OpsWorks Stacks 以满足您的特定要求。

说明书 101

描述如何实现 AWS OpsWorks 堆栈实例的配方。

负载均衡一个层

描述如何使用可用的 AWS OpsWorks Stacks 负载平衡选项。

在 VPC 中运行堆栈

介绍如何在 Virtual Private Cloud 中创建和运行堆栈。

从 Chef Server 迁移

提供从 Chef 服务器迁移到 AWS OpsWorks Stacks 的指南。

AWS OpsWorks 堆栈图层参考

描述可用的 AWS OpsWorks 堆栈内置图层。

说明书组件

介绍三个标准说明书组件：属性、模板和配方。

堆栈配置和部署属性：Linux

介绍适用于 Linux 的堆栈配置和部署属性。

内置说明书属性

介绍如何使用内置配方属性来控制所安装软件的配置。

为 Linux 进行 Chef 11.10 和早期版本的故障排除

描述解决 AWS OpsWorks 堆栈中各种问题的方法。

Chef 11 Linux 堆栈入门

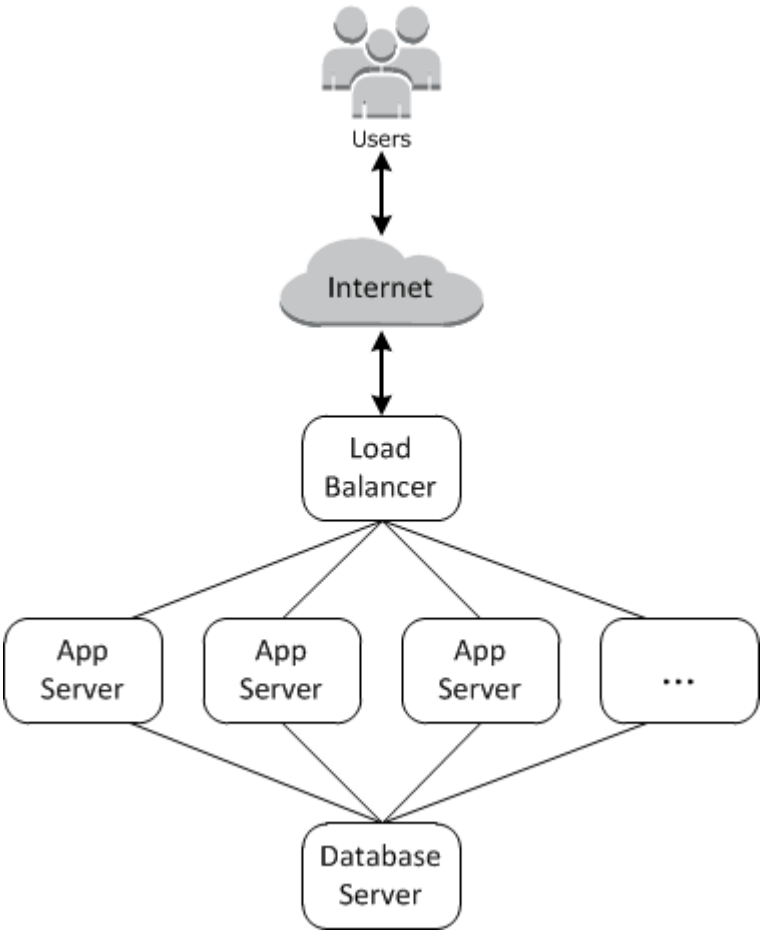
Important

该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS：Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

Note

这部分介绍关于使用 Chef 11 的 Linux 堆栈的入门知识。有关 Chef 12 Linux 堆栈入门的信息，请参阅入门：Linux。有关 Chef 12 Windows 堆栈入门的信息，请参阅入门：Windows。

基于云的应用程序通常需要一组相关的资源 (应用程序服务器、数据库服务器等等)，且必须集体创建并管理这些资源。此实例集合称为堆栈。简单的应用程序堆栈可能如下所示。



基础架构包含以下内容：

- 一个负载均衡器，用于在各个应用程序服务器之间平均分配来自用户的传入流量。
- 一组应用程序服务器实例，数量必须足以处理流量。
- 一个数据库服务器，用于为应用程序服务器提供后端数据存储。

此外，您通常需要一种可将应用程序分配给应用程序服务器、监控堆栈等的方式。

AWS OpsWorks Stacks 提供了一种简单明了的方式来创建和管理堆栈及其关联的应用程序和资源。本章通过在图表中向您展示创建应用程序服务器堆栈的过程，来介绍 AWS OpsWorks Stacks 的基础知识，以及它的一些更加复杂的功能。它使用 AWS OpsWorks Stacks易于遵循的增量开发模型：设置基本堆栈，并在其正常运行后添加组件，直到获得功能齐全的实现。

- 步骤 1：完成前提条件演示如何设置以开始演练。

- 步骤 2：创建简单的应用程序服务器堆栈 - Chef 11介绍如何创建仅包含一个应用程序服务器的最小堆栈。
- 步骤 3：添加后端数据存储介绍如何添加数据库服务器并将它连接到应用程序服务器。
- 第 4 步：横向扩展 MyStack显示如何通过添加更多应用程序服务器以及添加用于分配入站流量的负载均衡器来扩展堆栈以处理增加的负载。

主题

步骤 1：完成前提条件

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS：Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

完成以下设置步骤，然后您才能开始演练。这些设置步骤包括注册 AWS 帐户、创建管理用户以及为 AWS OpsWorks Stacks 分配访问权限。

如果您已经完成任何AWS OpsWorks 堆栈入门演练，则您已满足此演练的先决条件，可以向前跳至步骤 2：创建简单的应用程序服务器堆栈 - Chef 11。

主题

注册获取 AWS 账户

如果您没有 AWS 账户，请完成以下步骤来创建一个。

要注册 AWS 账户

1. 打开 https://portal.aws.amazon.com/billing/signup。
2. 按照屏幕上的说明进行操作。

   在注册时，将接到一通电话，要求使用电话键盘输入一个验证码。

   当您注册时 AWS 账户，就会创建AWS 账户根用户一个。根用户有权访问该账户中的所有 AWS 服务 和资源。作为安全最佳实践，请为用户分配管理访问权限，并且只使用根用户来执行需要根用户访问权限的任务。

AWS 注册过程完成后会向您发送一封确认电子邮件。在任何时候，您都可以通过转至 https://aws.amazon.com/ 并选择我的账户来查看当前的账户活动并管理您的账户。

创建具有管理访问权限的用户

注册后，请保护您的安全 AWS 账户 AWS 账户根用户 AWS IAM Identity Center，启用并创建管理用户，这样您就不会使用 root 用户执行日常任务。

保护你的 AWS 账户根用户

1. 选择 Root 用户并输入您的 AWS 账户 电子邮件地址，以账户所有者的身份登录。AWS Management Console在下一页上，输入您的密码。

   要获取使用根用户登录方面的帮助，请参阅 《AWS 登录 用户指南》中的以根用户身份登录。
2. 为您的根用户启用多重身份验证 (MFA)。

   有关说明，请参阅 I A M 用户指南中的为 AWS 账户 根用户启用虚拟 MFA 设备（控制台）。

创建具有管理访问权限的用户

1. 启用 IAM Identity Center。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的启用 AWS IAM Identity Center。
2. 在 IAM Identity Center 中，为用户授予管理访问权限。

有关使用 IAM Identity Center 目录 作为身份源的教程，请参阅《用户指南》 IAM Identity Center 目录中的使用默认设置配置AWS IAM Identity Center 用户访问权限。

以具有管理访问权限的用户身份登录

- 要使用您的 IAM Identity Center 用户身份登录，请使用您在创建 IAM Identity Center 用户时发送到您的电子邮件地址的登录网址。

  有关使用 IAM Identity Center 用户登录的帮助，请参阅AWS 登录 用户指南中的登录 AWS 访问门户。

将访问权限分配给其他用户

1. 在 IAM Identity Center 中，创建一个权限集，该权限集遵循应用最低权限的最佳做法。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的创建权限集。
2. 将用户分配到一个组，然后为该组分配单点登录访问权限。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的添加组。

为您的用户分配服务访问权限

通过向您的角色或用户添加和权限，启用对 AWS OpsWorks Stacks 服务（以及 Stacks 所依赖的`AWSOpsWorks_FullAccess`相关服务）的`AmazonS3FullAccess`访问权限。 AWS OpsWorks

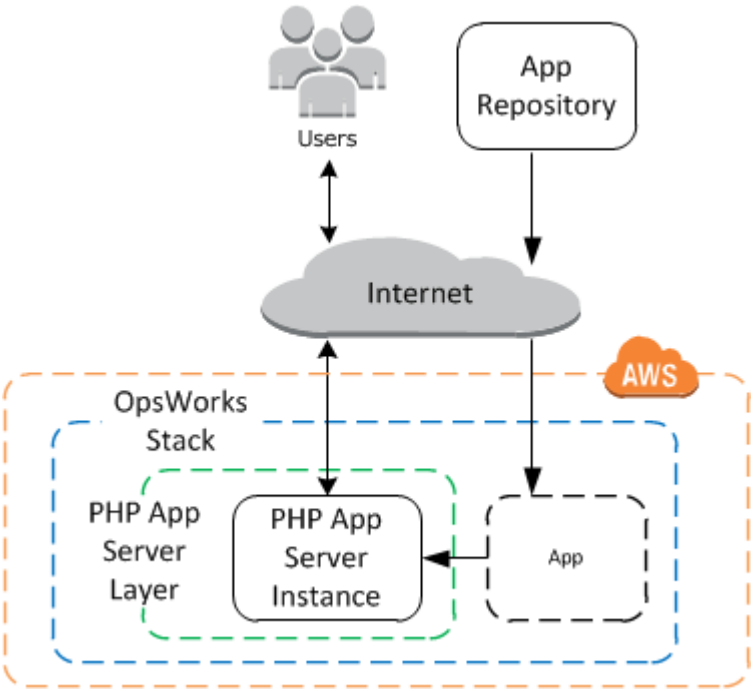
有关添加权限的更多信息，请参阅：添加 IAM 身份权限（控制台）。

您现已完成所有设置步骤，可开始此演练。

步骤 2：创建简单的应用程序服务器堆栈 - Chef 11

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

一个基本的应用程序服务器堆栈包含一个具有公有 IP 地址的应用程序服务器实例以接收用户请求。应用程序代码和任何相关文件均存储在单独的存储库中，并从该位置部署到服务器。下图阐明了此类堆栈。



该堆栈拥有以下组件：

- 一个层，它表示一组实例并指定这些实例的配置方式。

  在本例中，层表示一组 PHP App Server 实例。
- 一个实例，它表示一个 Amazon EC2 实例。

  在本例中，实例被配置为运行 PHP 应用程序服务器。图层可以有任意数量的实例。 AWS OpsWorks Stacks 还支持其他几个应用程序服务器。有关更多信息，请参阅 应用程序服务器层。
- 应用程序，其中包含在应用程序服务器上安装应用程序所需的信息。

  代码存储在一个远程存储库中，例如 Git 存储库或 Amazon S3 存储桶。

以下各节介绍如何使用 AWS OpsWorks Stacks 控制台创建堆栈和部署应用程序。您也可以使用 AWS CloudFormation 模板来配置堆栈。有关预置本主题中描述的堆栈的示例模板，请参阅 AWS S OpsWorks nippets。

主题

- 步骤 2.1：创建堆栈 - Chef 11

- 步骤 2.2：添加 PHP App Server 层 - Chef 11
- 步骤 2.3：向 PHP App Server 层添加实例 - Chef 11
- 步骤 2.4：创建和部署应用程序 - Chef 11

步骤 2.1：创建堆栈 - Chef 11

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS：Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

您可以通过创建 AWS OpsWorks 堆栈来启动 Stacks 项目，该堆栈充当您的实例和其他资源的容器。堆栈配置指定一些由堆栈的所有实例共享的基本设置，如 Amazon Web Services Region 和默认操作系统。

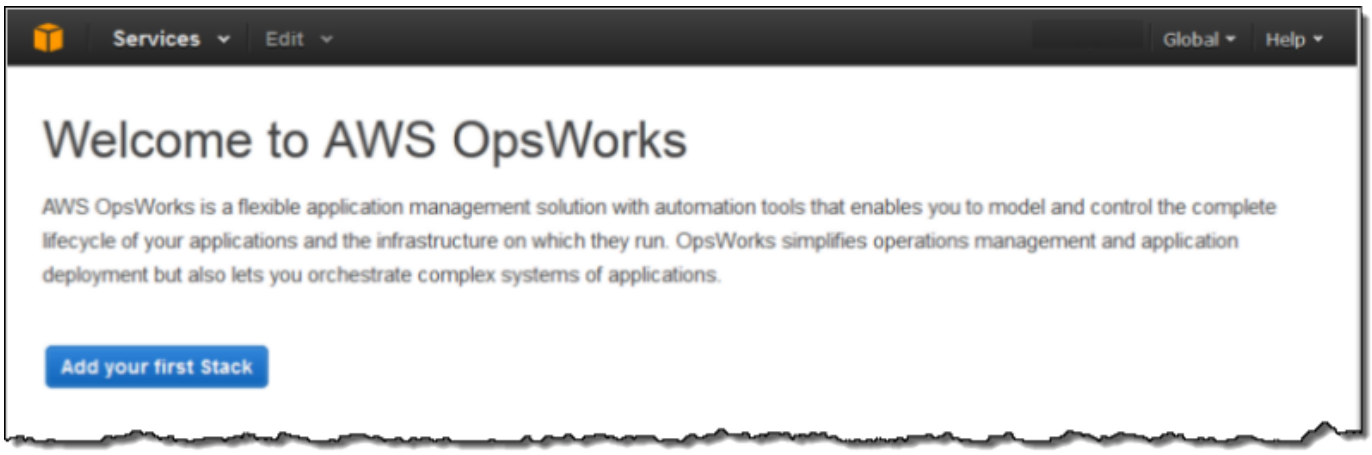
> **Note**
>
> 此页面帮助您创建 Chef 11 堆栈。有关如何创建 Chef 12 堆栈的信息，请参阅创建堆栈。

此页面帮助您在 Chef 11 中创建堆栈。

创建新堆栈

1. 添加堆栈

   登录到 AWS OpsWorks Stacks 控制台。如果该账户没有现有堆栈，您将看到 “欢迎使用 AWS OpsWorks” 页面；单击 “添加您的第一个堆栈”。否则，您将看到 AWS OpsWorks Stacks 控制面板，其中列出了您账户的堆栈；点击添加堆栈。



2. 配置堆栈

   在 Add Stack 页面上，选择 Chef 11 stack 并指定以下设置：

   堆栈名称

   为您的堆栈输入一个名称，可以包含字母数字字符 (a–z、A–Z、0-9) 和连字符 (-)。此演练的示例堆栈名为 **`MyStack`**。

   区域

   选择 美国西部（俄勒冈州）作为堆栈的区域。

   接受其他设置的默认值并单击 Add Stack。有关各种堆栈设置的更多信息，请参阅创建新堆栈。

步骤 2.2：添加 PHP App Server 层 - Chef 11

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

尽管堆栈基本上是实例的容器，但您不能直接将实例添加到堆栈。您需要添加一个层 (表示一组相关实例)，然后将实例添加到该层。

层基本上是一个蓝图， AWS OpsWorks Stacks 使用它来创建一组具有相同配置的 Amazon EC2 实例。您在堆栈中为每个相关实例组添加一个层。 AWS OpsWorks 堆栈包括一组内置层，用于表示运行

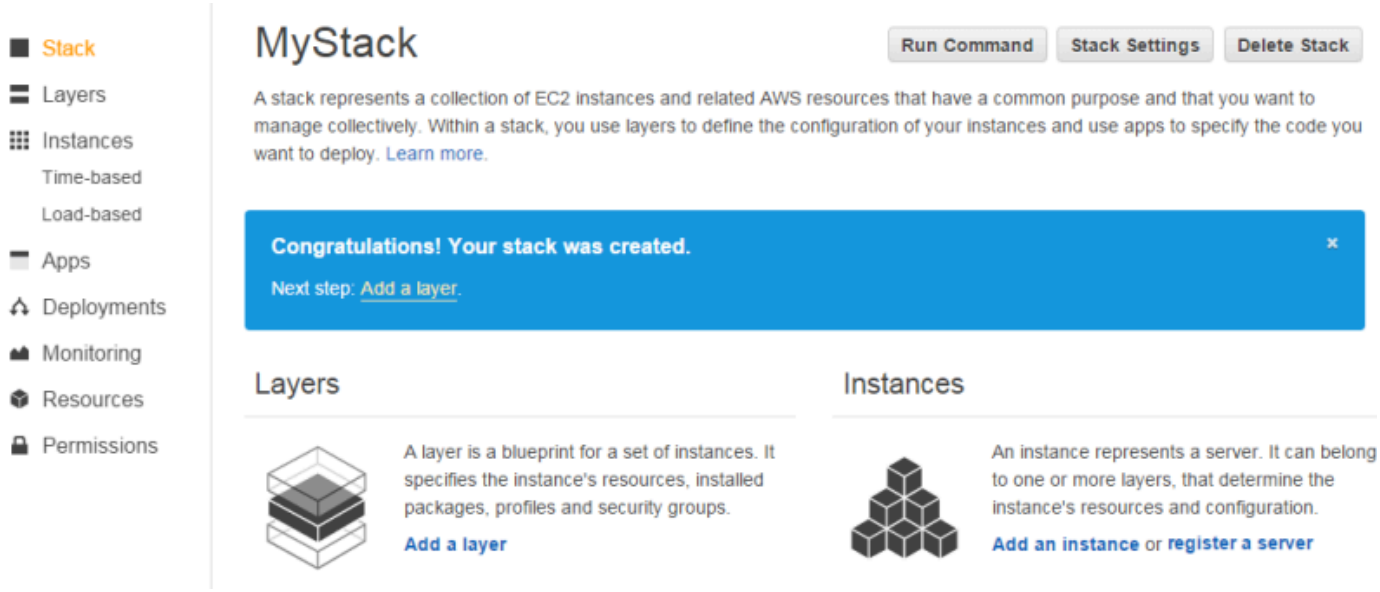
标准软件包（例如 MySQL 数据库服务器或 PHP 应用服务器）的实例组。此外，您还可以创建部分或完全自定义的层以满足特定要求。有关更多信息，请参阅 自定义堆栈 AWS OpsWorks。

MyStack 有一层，即内置的 PHP 应用服务器层，它表示一组用作 PHP 应用程序服务器的实例。有关更多信息，包括内置层的说明，请参阅图层。

将 PHP 应用服务器层添加到 MyStack

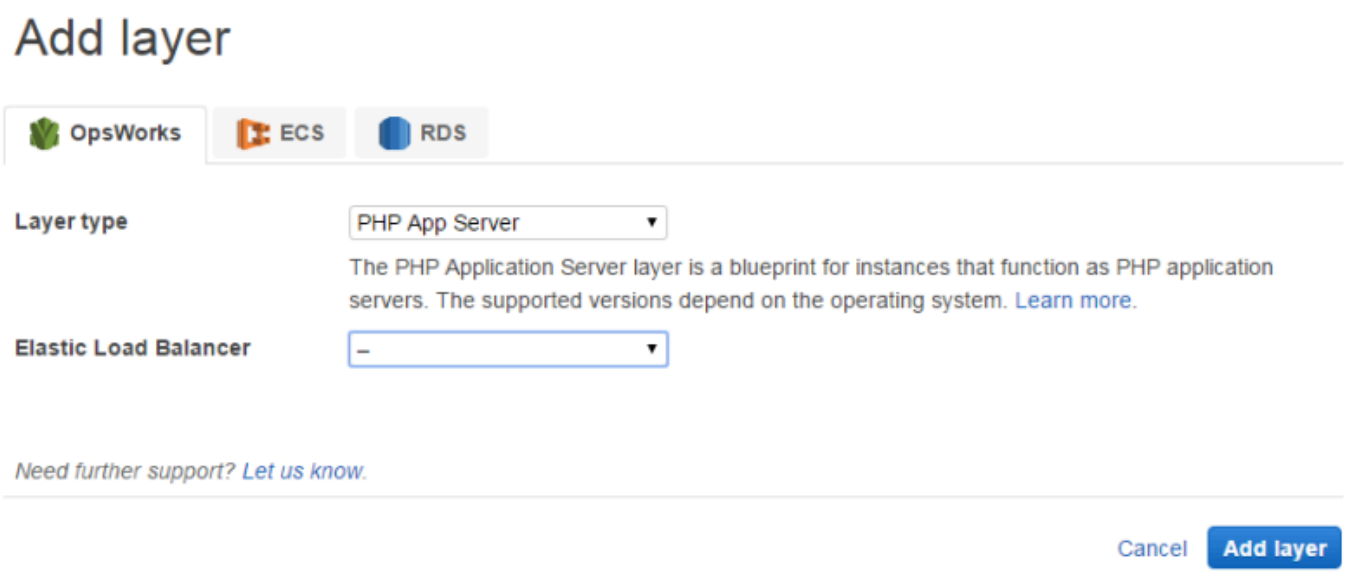
1. 打开“Add Layer”页面

   创建完堆栈后，Stac AWS OpsWorks ks 会显示 “堆栈” 页面。单击 Add a layer 以添加您的第一个层。



2. 指定层类型并配置层

   在 层类型框中，选择 PHP 应用程序服务器，接受默认 Elastic Load Balancer 设置并单击 添加层。创建层后，您可以通过编辑层来指定其他属性 (如 EBS 卷配置)。



步骤 2.3：向 PHP App Server 层添加实例 - Chef 11

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

AWS OpsWorks 堆栈实例代表一个特定的 Amazon EC2 实例：

- 该实例的配置指定了一些基本信息，如 Amazon EC2 操作系统和大小；它会运行，但不会经常运行。
- 该实例的层通过确定要安装哪些软件包以及该实例是否拥有弹性 IP 地址等内容而向该实例添加功能。

AWS OpsWorks Stacks 会在每个与服务交互的实例上安装一个代理。为了向实例添加层的功能，AWS OpsWorks Stacks 会指示代理运行名为 Chef recipes 的小应用程序，这些应用程序可以安装应用程序和软件包、创建配置文件等。 AWS OpsWorks Stacks 在实例生命周期的关键时刻运行配方。例如，在实例完成启动后 OpsWorks 运行安装配方以处理诸如安装软件之类的任务；在部署应用程序以安装代码和相关文件时运行 Deploy 配方。

> **Note**
>
> 如果你对食谱的工作原理感到好奇，那么所有 AWS OpsWorks Stacks 内置食谱都位于一个公共 GitHub 存储库中：OpsWorks Cookbook s。您也可以创建自己的自定义配方，并让 AWS OpsWorks Stacks 运行它们，如下文所述。

要将 PHP 应用程序服务器添加到中 MyStack，请将实例添加到您在上一步中创建的 PHP App Server 层。

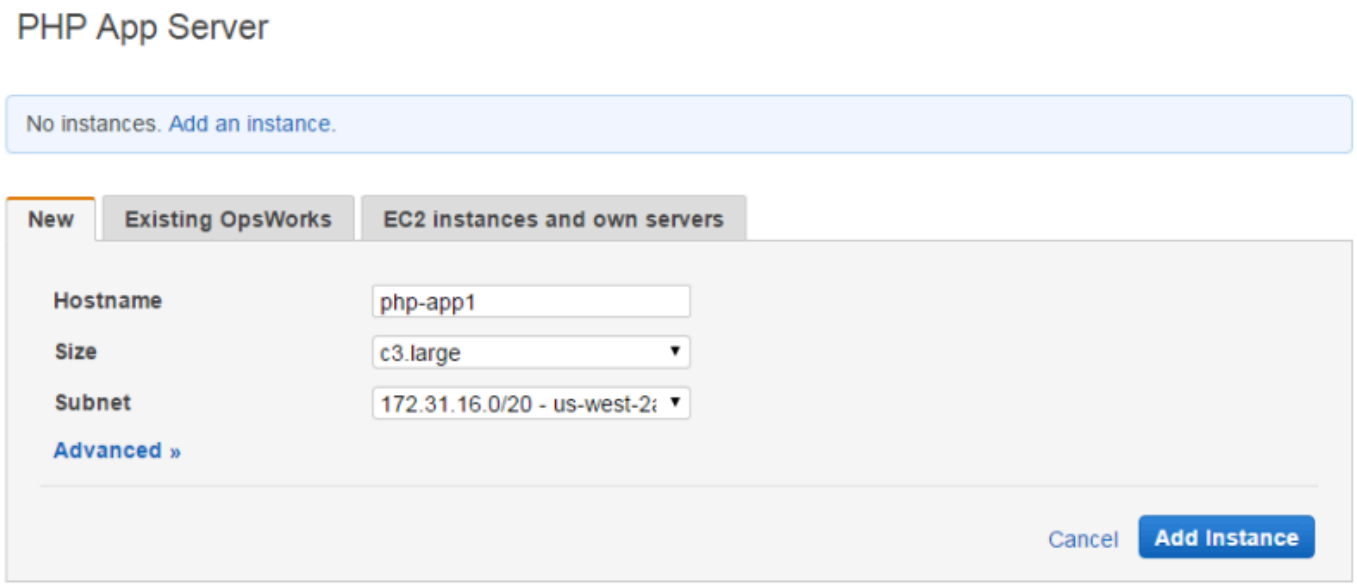
向 PHP App Server 层添加另一个实例

1. 打开“Add an Instance”

   添加完图层后， AWS OpsWorks Stacks 会显示 “图层” 页面。在导航窗格中，单击 Instances，然后在 PHP App Server 下单击 Add an instance。

2. 配置实例

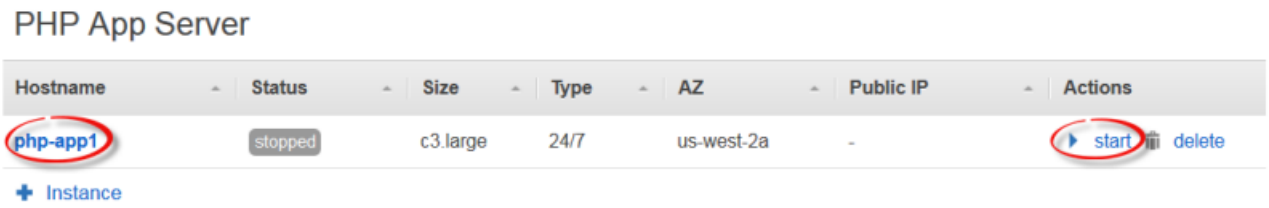
   每个实例都有一个由 AWS OpsWorks Stacks 为您生成的默认主机名。在此示例中， AWS OpsWorks Stacks 只是在图层的短名称中添加一个数字。您可以单独配置每个实例，包括覆盖您在创建堆栈时指定的某些默认设置，如可用区或操作系统。对于本演练，只需接受默认设置并单击 Add Instance 以将实例添加到该层即可。有关更多信息，请参阅 实例。



3. 启动实例

   到目前为止，您只是指定了该实例的配置。您必须启动一个实例才能创建正在运行的 Amazon EC2 实例。 AWS OpsWorks 然后，堆栈使用配置设置在指定的可用区启动 Amazon EC2 实例。有关如何启动实例的详细信息取决于该实例的扩展类型。在上一步中，您创建了具有默认扩展类型 (全天候) 的实例，该实例必须手动启动，然后一直运行，直至手动停止。您还可以创建基于时间和基于负载的扩展类型， AWS OpsWorks Stacks 会根据计划或当前负载自动启动和停止这些类型。有关更多信息，请参阅 使用基于时间和基于负载的实例管理负载。

   转到 PHP App Server 下的 php-app1，并单击该行的 Actions 列中的 start 以启动实例。

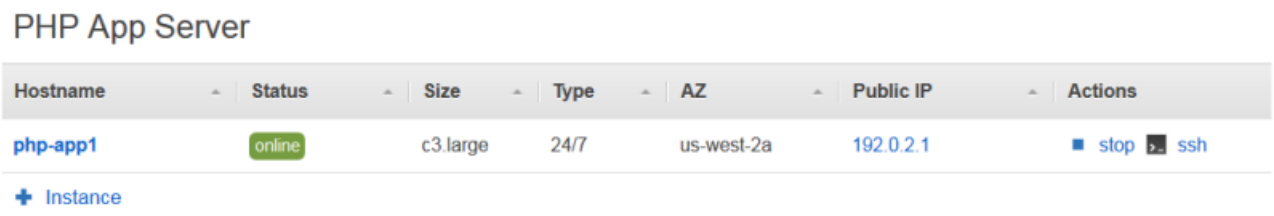


4. 启动期间监控实例的状态

   启动 Amazon EC2 实例并安装软件包通常需要几分钟时间。在启动过程中，实例的 Status 字段会显示下面一系列值：

   1. 已@@ 请求- AWS OpsWorks Stacks 已调用亚马逊 EC2 服务来创建亚马逊 EC2 实例。
   2. 待处理- AWS OpsWorks 堆栈正在等待 Amazon EC2 实例启动。
   3. booting - Amazon EC2 实例正在启动。
   4. running_setup- AWS OpsWorks Stacks 代理正在运行层的安装配方，用于处理诸如配置和安装软件包之类的任务，以及 Deploy 配方（用于将任何应用程序部署到实例）。
   5. online - 实例已准备就绪，可供使用。

   php-app1 联机后，Instances 页面应与以下内容类似：



   该页面首先简要汇总堆栈的所有实例。现在，它显示一个联机实例。在 php-app1 Actions 列中，请注意，stop (停止实例) 已取代 start 和 delete。

步骤 2.4：创建和部署应用程序 - Chef 11

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

为了使其 MyStack 更有用，您需要将应用程序部署到 PHP App Server 实例。您将应用程序代码和任何相关文件存储在 Git 之类的存储库中。要将这些文件放到应用程序服务器上，您需要采取以下几个步骤：

> **Note**
>
> 本部分中的过程适用于 Chef 11 堆栈。有关如何向 Chef 12 堆栈的层中添加应用程序的信息，请参阅添加应用程序。

1. 创建应用程序。

   应用程序包含 AWS OpsWorks Stacks 从存储库下载代码和相关文件所需的信息。您也可以指定其他信息，如应用程序的域。
2. 将应用程序部署到应用程序服务器。

   部署应用程序时， AWS OpsWorks Stacks 会触发 Deploy 生命周期事件。代理随后会运行该实例的 Deploy 配方，这会将文件下载到合适目录并执行相关任务 (如配置服务器、重启服务等)。

> **Note**
>
> 当您创建新实例时， AWS OpsWorks Stacks 会自动将所有现有应用程序部署到该实例。但是，当您创建新应用程序或更新现有应用程序时，必须手动部署该应用程序或对所有现有实例进行更新。

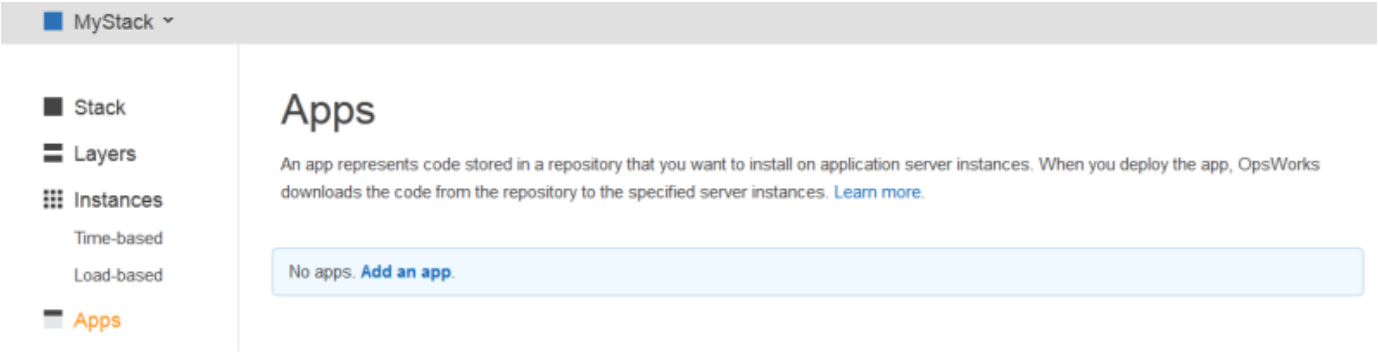
此步骤显示了如何手动将示例应用程序从公有 Git 存储库部署到应用程序服务器。如果你想查看应用程序，请前往 https://github.com/amazonwebservices/ opsworks-demo-php-simple-app。本示例中使用

的应用程序位于版本 1 分支中。 AWS OpsWorks 堆栈还支持其他几种存储库类型。有关更多信息，请参阅 应用程序源。

创建和部署应用程序

1. 打开应用程序页面

   在导航窗格中，单击 Apps，然后在 Apps 页面上单击 Add an app。



2. 配置应用程序

   在 App 页上，指定以下值：

   名称

   应用程序的名称， AWS OpsWorks Stacks 将其用于显示目的。示例应用程序名为**SimplePHPApp**。 AWS OpsWorks Stacks 还会生成一个简短的名称（在本例中为 simplephpapp），该名称供内部使用和 Deploy 配方使用，如后所述。

   类型

   应用程序的类型 (将决定在哪里部署该应用程序)。示例使用的是 PHP，这会将该应用程序部署到 PHP App Server 实例。

   数据来源类型

   关联的数据库服务器。现在选择 None；我们将在步骤 3：添加后端数据存储中介绍数据库服务器。

   存储库类型

   应用程序的存储库类型。示例应用程序存储在 Git 存储库中。

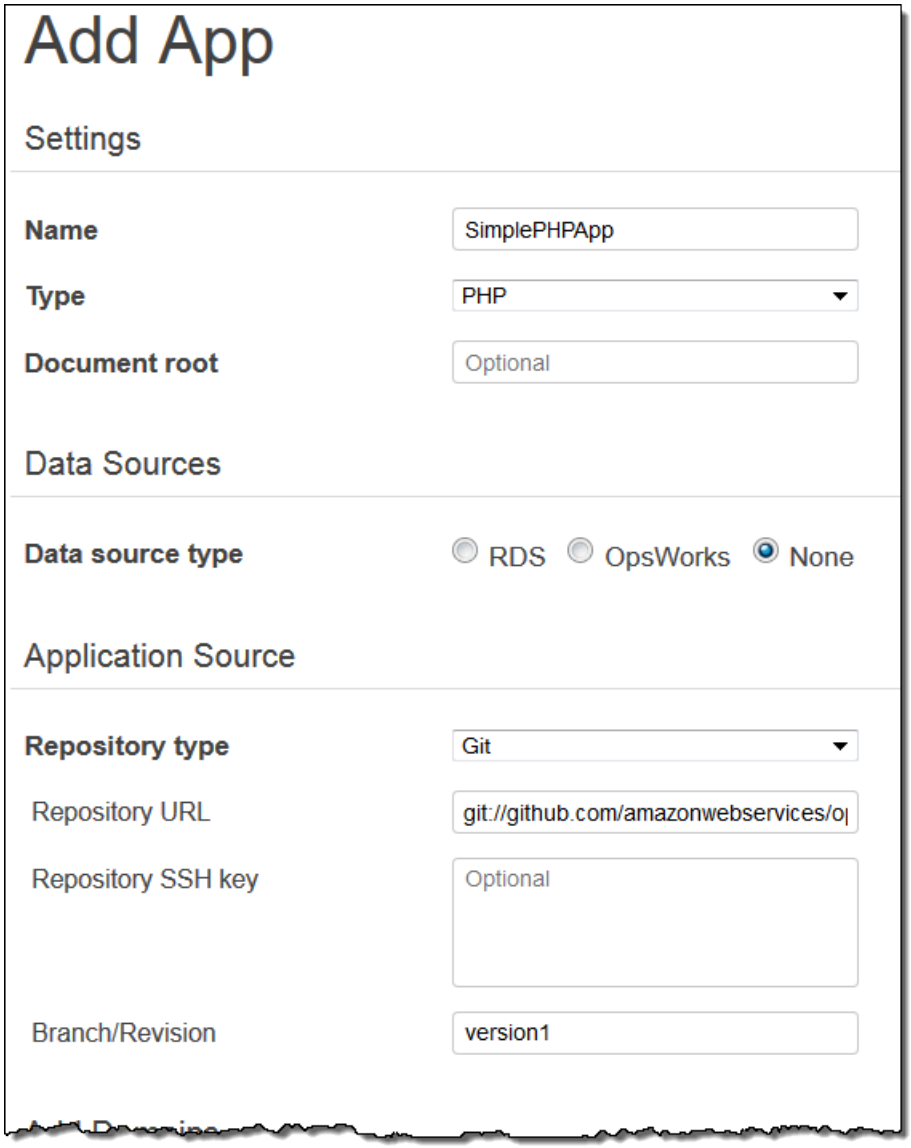
存储库 URL

应用程序的存储库 URL。示例 URL 为：**`git://github.com/awslabs/opsworks-demo-php-simple-app.git`**

分支/修订

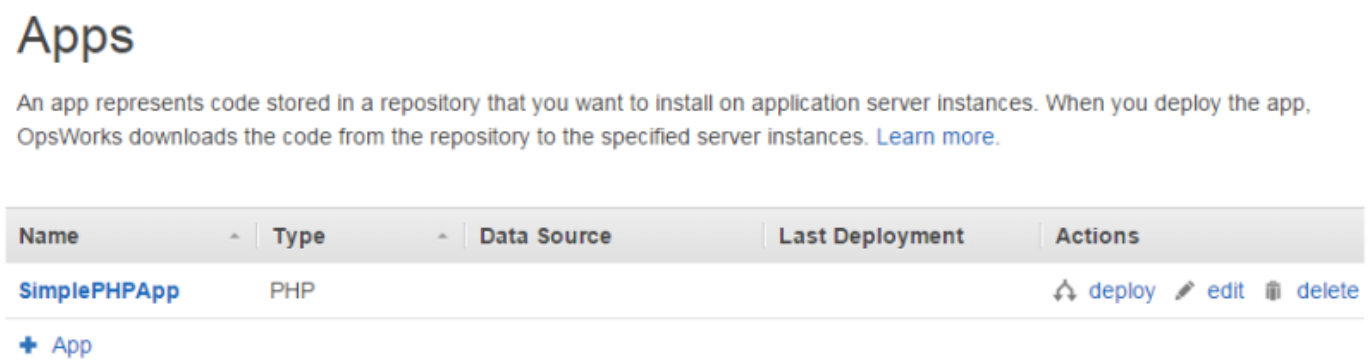
应用程序的分支或版本。本部分演练使用 **`version1`** 分支。

将其余设置保留默认值，然后单击 Add App。有关更多信息，请参阅 添加应用程序。



3. 打开部署页面

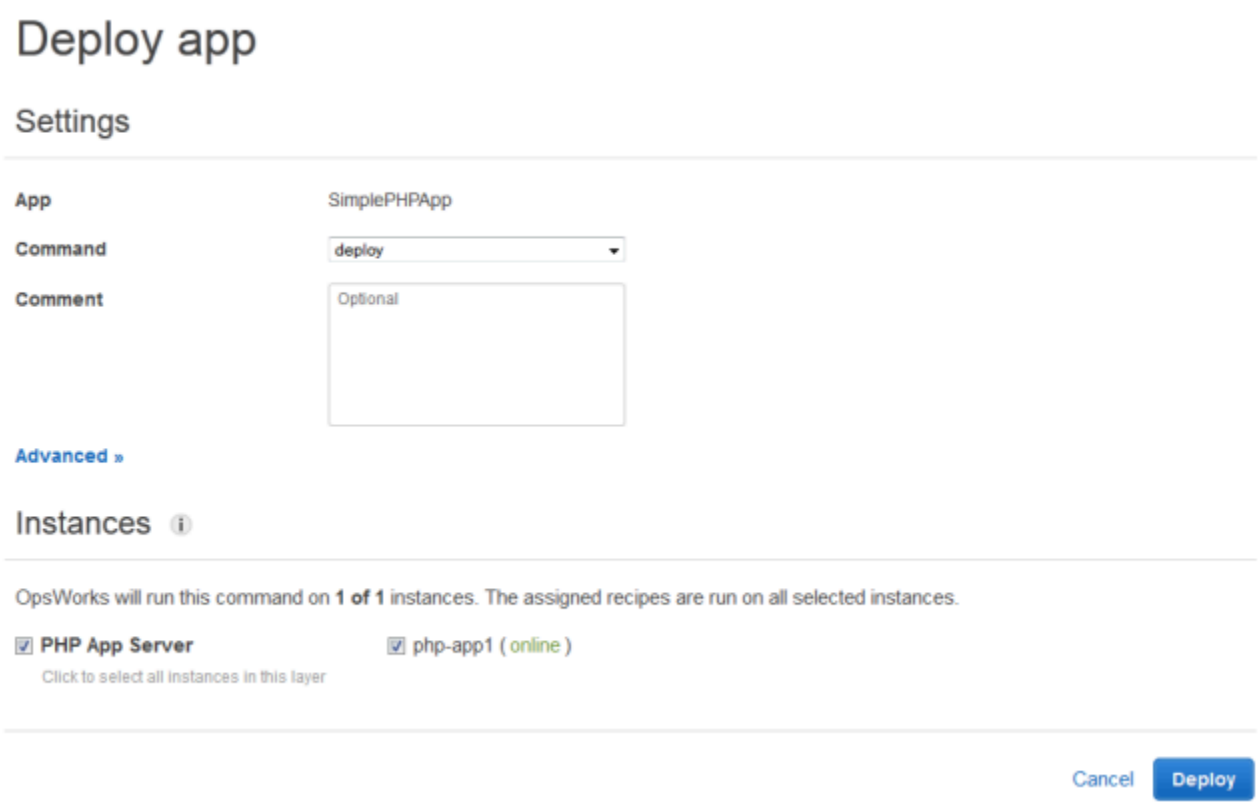
   要在服务器上安装代码，您必须部署应用程序。要执行此操作，请在 SimplePHPApp Actions 列中单击 deploy。



4. 部署应用程序

   部署应用程序时，代理将在 PHP App Server 实例上运行 Deploy 配方 (这将下载和配置应用程序)。

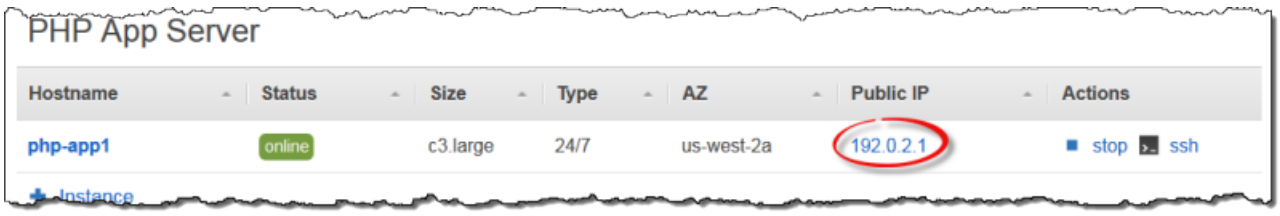
   Command 应该已经设置为 deploy。对于其他设置，保持默认设置不变，然后单击 Deploy 以部署应用程序。



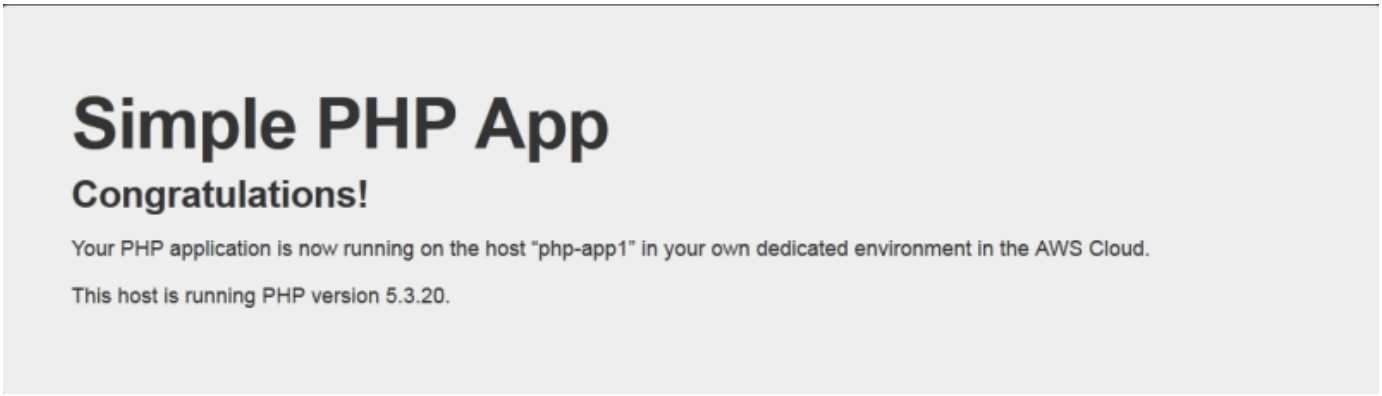
完成部署后，Deployment 页面将显示 Status 为 Successful，并且 php-app1 旁边会出现一个绿色对勾。

5. 运行 SimplePHPApp

SimplePHPApp 现已安装并可以运行。要运行它，请在导航窗格中单击 Instances 以转到 Instances 页面。然后单击 php-app1 实例的公有 IP 地址。



您应在浏览器中看到如下所示的页面。

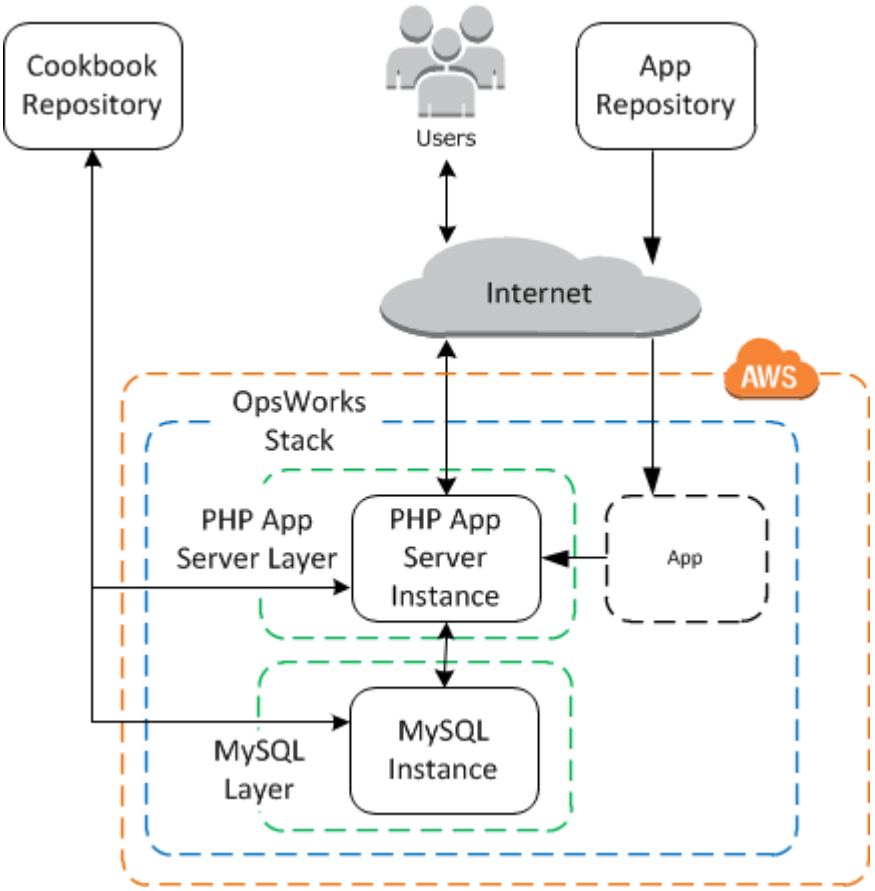


> **Note**
>
> 本演练假定您将继续执行下一部分并最终在一个会话中完成整个演练。如果你愿意，你可以随时停下来，稍后通过登录 AWS OpsWorks Stacks 并打开堆栈继续操作。但是，您需要为所使用的任何 AWS 资源 (如联机实例) 付费。为避免产生不必要的费用，您可以停止实例，从而终止相应的 EC2 实例。当您准备好继续时，可以再次启动该实例。

步骤 3：添加后端数据存储

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

步骤 2.1：创建堆栈 - Chef 11 向您展示如何创建为 PHP 应用程序提供服务的堆栈。但是，这是一个非常简单的应用程序，只能显示一些静态文本。生产应用程序通常使用后端数据存储，从而生成类似于下图所示的堆栈配置。



本节介绍如何扩展 MyStack 以包括后端 MySQL 数据库服务器。但是您需要做的不仅仅是向堆栈中添加一个 MySQL 服务器。您还必须将应用程序配置为与数据库服务器正常通信。 AWS OpsWorks Stacks 不会为你做这件事；你需要实现一些自定义配方来处理该任务。

主题

- 步骤 3.1：添加后端数据库
- 步骤 3.2：更新 SimplePHPApp
- 简短的题外话：食谱、食谱和堆栈属性 AWS OpsWorks
- 步骤 3.3：将自定义食谱添加到 MyStack
- 步骤 3.4：运行配方
- 步骤 3.5：部署 SimplePHPApp 版本 2
- 步骤 3.6：运行 SimplePHPApp

步骤 3.1：添加后端数据库

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

新版本的 SimplephApp 将其数据存储在后端数据库中。 AWS OpsWorks Stacks 支持两种类型的数据库服务器：

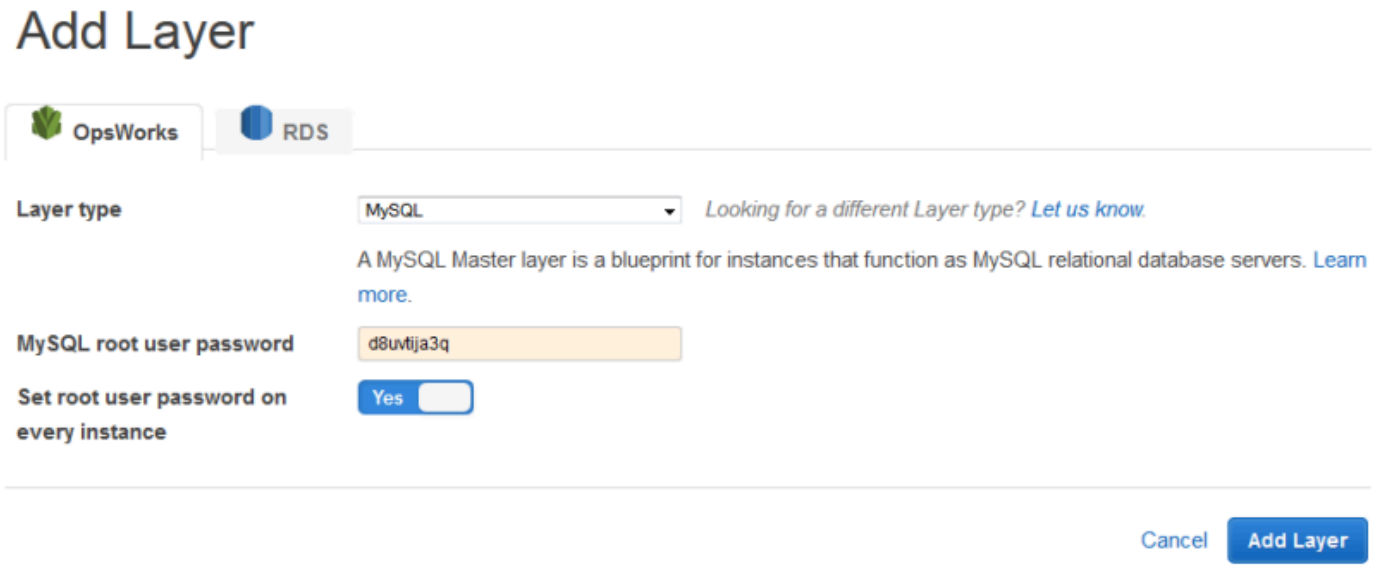
- MySQL AWS OpsWorks 堆栈层是创建托管 MySQL 数据库主服务器的 Amazon EC2 实例的蓝图。
- Amazon RDS 服务层提供了一种将 Amazon RDS 实例整合到堆栈中的方法。

您还可以使用诸如 Amazon DynamoDB 等其他数据库，或创建自定义层以支持 MongoDB 等数据库。有关更多信息，请参阅 the section called “使用后端数据存储”。

本示例使用 MySQL 层。

将 MySQL 层添加到 MyStack

1. 在 Layers 页面上，单击 + Layer。
2. 在 Add Layer 页面上，对于 Layer type，选择 MySQL，接受默认设置，然后单击 Add Layer。



将实例添加到 MySQL 层

1. 在 Layers 页面的 MySQL 行，单击 Add an instance。
2. 在 Instances 页面的 MySQL 下，单击 Add an instance。
3. 接受默认值，然后单击 Add instance，但尚不启动它。

> **Note**
>
> AWS OpsWorks Stacks 会自动创建一个使用应用程序的简称 simplephpapp 命名的数据库，在本例中为 simplephpapp。如果您想要使用 Chef 配方与数据库交互，那么您需要使用此名称。

步骤 3.2：更新 SimplePHPApp

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

要开始使用，您需要一个使用后端数据存储的新版本 SimplePHPApp。借助 AWS OpsWorks Stacks，可以轻松地更新应用程序。如果您使用 Git 或 Subversion 存储库，则可以针对每个应用程序版本拥有一个独立的存储库分支。该示例应用程序将存储一个应用程序版本，该应用程序使用 Git 存储库的 version2 分支中的后端数据库。您只需更新应用程序的配置来指定新分支并重新部署应用程序。

更新 SimplePHPApp

1. 打开应用程序的“Edit”页面

   在导航窗格中，单击 Apps，然后单击 SimplePHPApp 行的 Actions 列中的 edit。
2. 更新应用程序的配置

   更改以下设置。

分支/修订

此设置指示应用程序的存储库分支。第一个 SimplePHPApp 版本未连接到数据库。要使用支持数据库的应用程序版本，请将此值设置为 **`version2`**。

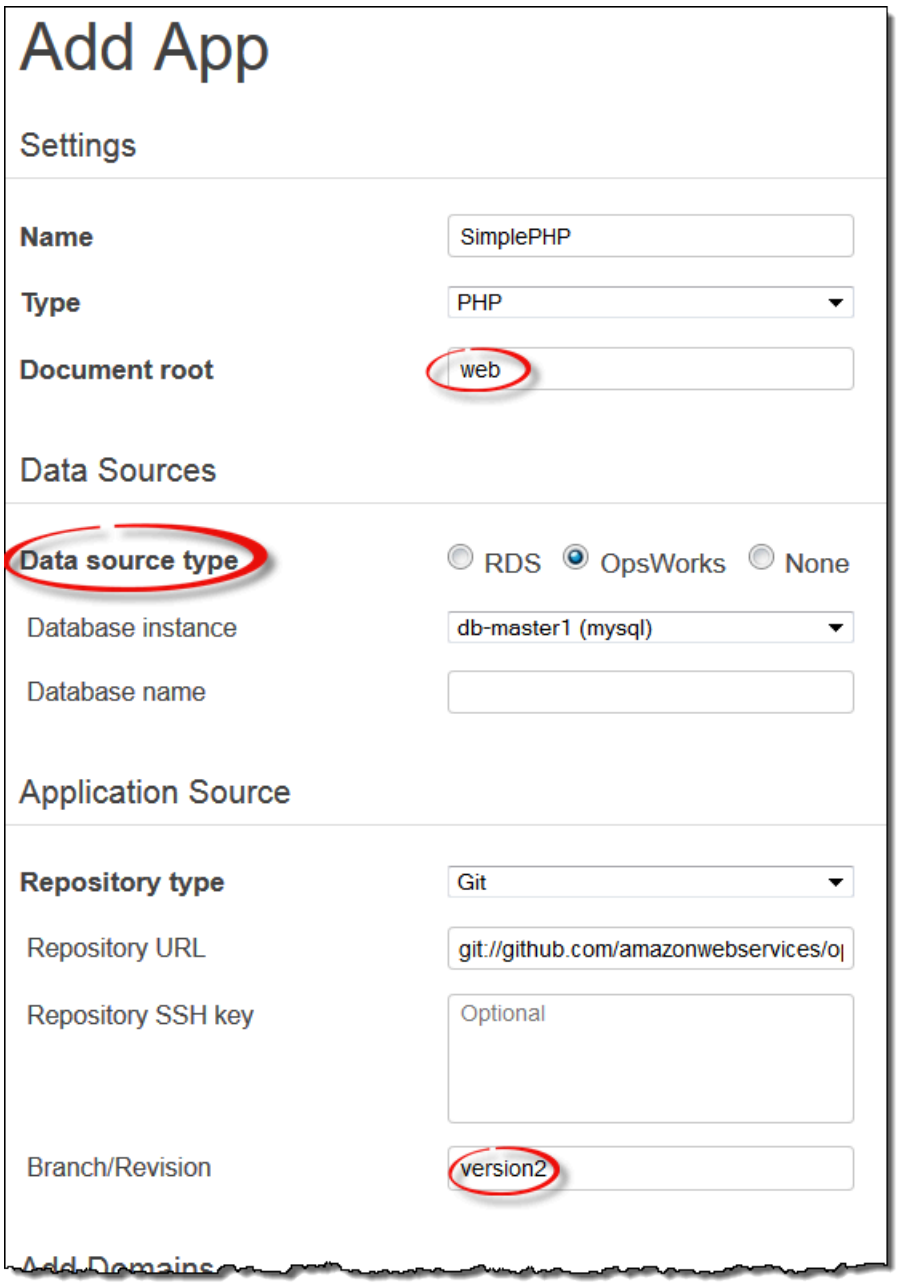
Document root

此设置将指定您的应用程序的根文件夹。使用默认设置的第一个 SimplePHPApp 版本，它在服务器的标准根文件夹（对于 PHP 应用程序为 `/srv/www`）中安装 `index.php`。如果您在此处指定子文件夹（仅指定名称，没有前导 “/”），AWS OpsWorks Stacks 会将其附加到标准文件夹路径。SimplePHPApp 的版本 2 应该在 `/srv/www/web` 中，因此请将 Document root (文档根) 设置为 **`web`**。

数据来源类型

此设置将数据库服务器与应用程序相关联。该示例使用您在上一步中创建的 MySQL 实例，因此请将数据源类型设置为，将数据库实例设置为 OpsWorks 您在上一步中创建的实例 db-master1 (mysql)。将数据库名称留空； AWS OpsWorks Stacks 将在服务器上创建一个以应用程序的短名称 simplephpapp 命名的数据库。

然后单击 Save 以保存新配置。



3. 开启 MySQL 实例。

更新应用程序后，AWS OpsWorks Stacks 会在您启动任何新的应用程序服务器实例时自动将新应用程序版本部署到这些实例。但是，AWS OpsWorks Stacks 不会自动将新的应用程序版本部署到现有服务器实例；您必须手动执行此操作，如中所步骤 2.4：创建和部署应用程序 - Chef 11述。您现在就可以部署更新的 SimplePHPApp 了，但在本示例中，最好再等一会儿。

简短的题外话：食谱、食谱和堆栈属性 AWS OpsWorks

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

现在，您拥有了应用程序和数据库服务器，但它们还未准备就绪，还不能使用。您仍然需要设置数据库并配置应用程序的连接设置。 AWS OpsWorks Stacks 不会自动处理这些任务，但它确实支持 Chef 食谱、食谱和动态属性。你可以实现两个配方，一个用于设置数据库，一个用于配置应用程序的连接设置，然后让 AWS OpsWorks Stacks 为你运行它们。

phpapp 说明书 (其中包含所需的配方) 已经实施并且准备就绪，可供使用；如果您愿意，您可以直接跳至 步骤 3.3：将自定义食谱添加到 MyStack。如果您想要了解更多信息，本部分提供了一些关于说明书和配方的背景知识，并介绍了配方的工作原理。要查看说明书，请转至 phpapp 说明书。

主题

- 配方和属性
- 设置数据库
- 将应用程序连接到数据库

配方和属性

Chef 配方基本上是一个专门的 Ruby 应用程序，可在实例上执行任务，例如安装程序包、创建配置文件、执行 shell 命令等。相关配方的组被组织成说明书，其中还包含配置文件创建模板等支持文件。

AWS OpsWorks Stacks 有一套支持内置图层的食谱。您也可以用自己的配方创建自定义说明书，以便在您的实例上执行自定义任务。本主题简要介绍配方，并展示如何使用配方来设置数据库并配置应用程序的连接设置。有关说明书和配方的更多信息，请参阅 说明书和诀窍 或 自定义堆栈 AWS OpsWorks。

配方通常依靠 Chef 属性来获得输入数据：

- 其中一些属性由 Chef 定义并提供关于实例的基本信息，例如操作系统。
- AWS OpsWorks 堆栈定义了一组属性，这些属性包含有关堆栈的信息（例如图层配置）和有关已部署应用程序的信息（例如应用程序存储库）。

您可以通过向堆栈或部署分配自定义 JSON 的方法向该组添加自定义属性。

- 您的说明书也可以定义特定于说明书的属性。

  在 `attributes/default.rb` 中定义了 phpapp 说明书属性。

有关 AWS OpsWorks 堆栈属性的完整列表，请参阅堆栈配置和部署属性：Linux和。内置说明书属性有关更多信息，请参阅 覆盖属性。

属性按层次结构进行组织，可以表示为 JSON 对象。

您通过使用如下所示的 Chef 节点语法将此数据整合到应用程序中：

```
[:deploy][:simplephpapp][:database][:username]
```

`deploy` 节点具有单个应用程序节点 `simplephpapp`，该节点包含有关应用程序数据库、Git 存储库等内容的信息。本示例表示数据库用户名称的值，该值被解析为 `root`。

设置数据库

MySQL 层的内置“设置”配方会自动为以应用程序的短名称命名的应用程序创建数据库，因此本示例中已经有一个名为 simplephpapp 的数据库。但是，您需要通过为应用程序创建表来存储其数据，以完成设置。你可以手动创建表，但更好的方法是实现自定义配方来处理任务，然后让 AWS OpsWorks Stacks 为你运行它。本部分介绍了如何实施 `dbsetup.rb` 配方。稍后将介绍让 AWS OpsWorks Stacks 运行配方的过程。

要查看存储库中的配方，请转至 dbsetup.rb。以下示例显示了 `dbsetup.rb` 代码。

`execute` 是执行指定命令的 Chef 资源。在本例中，它是一个用于创建表的 MySQL 命令。`not_if` 指令可确保如果指定表已经存在，则不会运行此命令。有关 Chef 资源的更多信息，请参阅关于资源和提供程序。

配方使用之前讨论的节点语法将属性值插入命令字符串中。例如，以下会插入数据库的用户名称。

```
#{deploy[:database][:username]}
```

让我们对这个有些神秘的代码进行解压缩：

- 对于每个迭代，将 `deploy` 设置为当前应用程序节点，因此它会解析为 `[:deploy][:app_name]`。在本示例中，它会解析为 `[:deploy][:simplephpapp]`。

- 使用之前所示的部署属性值，整个节点将解析为 root。
- 在 #{} 中封装该节点，以将其插入到字符串中。

其他大多数节点都以类似方式进行解析。但 #{node[:phpapp][:dbtable]} 例外，它由自定义说明书的属性文件定义，并解析为表名称 urler。因此在 MySQL 实例上运行的实际命令是：

```
"/usr/bin/mysql
    -uroot
    -pvjud1hw5v8
    simplephpapp
    -e'CREATE TABLE urler(
       id INT UNSIGNED NOT NULL AUTO_INCREMENT,
       author VARCHAR(63) NOT NULL,
       message TEXT,
       PRIMARY KEY (id))'
"
```

此命令使用部署属性中的凭证和数据库名称来创建一个名为 urler 的表，其中包含 ID、作者和消息字段。

将应用程序连接到数据库

第二个难题是应用程序，它需要数据库密码等连接信息来访问表。SimplePHPApp 实际上只有一个工作文件 app.php；index.php 所做的工作只是加载 app.php。

app.php 包含 db-connect.php，该文件可处理数据库连接，但它不在存储库中。您无法提前创建 db-connect.php，因为它根据特定实例来定义数据库。而是由 appsetup.rb 配方使用部署属性中的连接数据来生成 db-connect.php。

要查看存储库中的配方，请转至 appsetup.rb。以下示例显示了 appsetup.rb 代码。

像 dbsetup.rb 一样，appsetup.rb 会迭代 deploy 节点中的应用程序，simplephpapp 也同样。它运行包含 script 资源和 template 资源的代码块。

script 资源会安装 Composer，它是 PHP 应用程序的依存关系管理器。然后将运行 Composer 的 install 命令将示例应用程序的依赖项安装到应用程序的根目录中。

template 资源生成 db-connect.php 并将其放到 /srv/www/simplephpapp/current 中。请注意以下几点：

- 配方使用条件语句来指定文件所有者，这取决于实例的操作系统。
- 只有当指定目录存在时，`only_if` 指令才会指示 Chef 生成模板。

`template` 资源可对其内容和结构与关联文件基本相同、但包含各种数据值占位符的模板执行操作。`source` 参数指定模板 `db-connect.php.erb`，该模板在 phpapp 说明书的 `templates/default` 目录中，其中包含以下内容：

当 Chef 处理该模板时，它会用模板资源中相应变量的值替换 `<%= =>` 占位符，这些变量值又从部署属性中提取。因此，生成的文件将是：

步骤 3.3：将自定义食谱添加到 MyStack

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

您将自定义说明书存储在存储库中，与存储应用程序的方法非常相似。每个堆栈都可以有一个存储库，其中包含一组自定义说明书。然后，您可以指 AWS OpsWorks 示 Stacks 在堆栈的实例上安装您的自定义食谱。

1. 单击导航窗格中的 Stack 以查看当前堆栈的页面。
2. 单击 Stack Settings，然后单击 Edit。
3. 按如下方式修改堆栈配置。
   - 使用自定义 Chef 说明书 是
   - Repository type - Git
   - Repository URL (存储库 URL) - **`git://github.com/amazonwebservices/opsworks-example-cookbooks.git`**
4. 单击 Save 以更新堆栈配置。

AWS OpsWorks 然后，Stacks 会将食谱存储库的内容安装到堆栈的所有实例上。如果您创建新实例，AWS OpsWorks Stacks 会自动安装食谱存储库。

> **Note**
>
> 如果您需要更新任何食谱或向存储库中添加新的食谱，则无需触摸堆栈设置即可完成。 AWS OpsWorks Stacks 将在所有新实例上自动安装更新的食谱。但是， AWS OpsWorks Stacks 不会自动在堆栈的在线实例上安装更新的食谱。你必须通过运行 AWS OpsWorks stac `Update Cookbooks` k 命令明确指示 Stacks 更新食谱。有关更多信息，请参阅 运行堆栈命令。

步骤 3.4：运行配方

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS：Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

获得自定义说明书后，您需要在相应实例上运行配方。您可以手动运行这些配方。但是，通常需要在实例生命周期中的可预测点运行配方，例如在实例启动后或者在您部署应用程序时。本节介绍一种简单得多的方法：让 AWS OpsWorks Stacks 在适当的时间自动为你运行它们。

AWS OpsWorks Stacks 支持一组生命周期事件，可简化运行配方。例如，“设置”事件在实例完成启动后发生，“部署”事件在部署应用程序时发生。每个层都有一组与每个生命周期事件关联的内置配方。当实例上发生生命周期事件时，代理会为实例的每个层运行关联的配方。要让 AWS OpsWorks Stacks 自动运行自定义配方，请将其添加到相应层的相应生命周期事件中，代理将在内置配方完成后运行配方。

在本示例中，您需要运行两个配方：在 MySQL 实例上运行 `dbsetup.rb`，在 PHP App Server 实例上运行 `appsetup.rb`。

> **Note**
>
> 通过使用 *cookbook_name*::*recipe_name* 格式 (其中 *recipe_name* 不包含 .rb 扩展名)，您可以在控制台中指定配方。例如，您可以把 `dbsetup.rb` 作为 **phpapp::dbsetup**。

将自定义配方分配给生命周期事件

1. 在 层 页面上，对于 MySQL，单击 配方，然后单击 编辑。
2. 在 Custom Chef recipes 部分中，对于 Deploy，输入 **phpapp::dbsetup**。

3. 单击 + 图标将配方分配给事件，然后单击 Save 以保存新层配置。
4. 返回 Layers (层) 页面，然后重复该步骤将 **phpapp::appsetup** 分配给 PHP App Server (PHP 应用程序服务器) 层的 Deploy (部署) 事件。

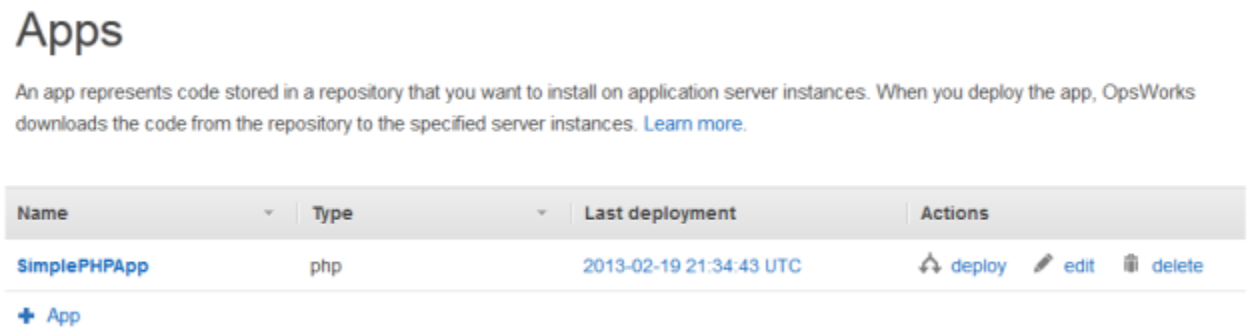
步骤 3.5：部署 SimplePHPApp 版本 2

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

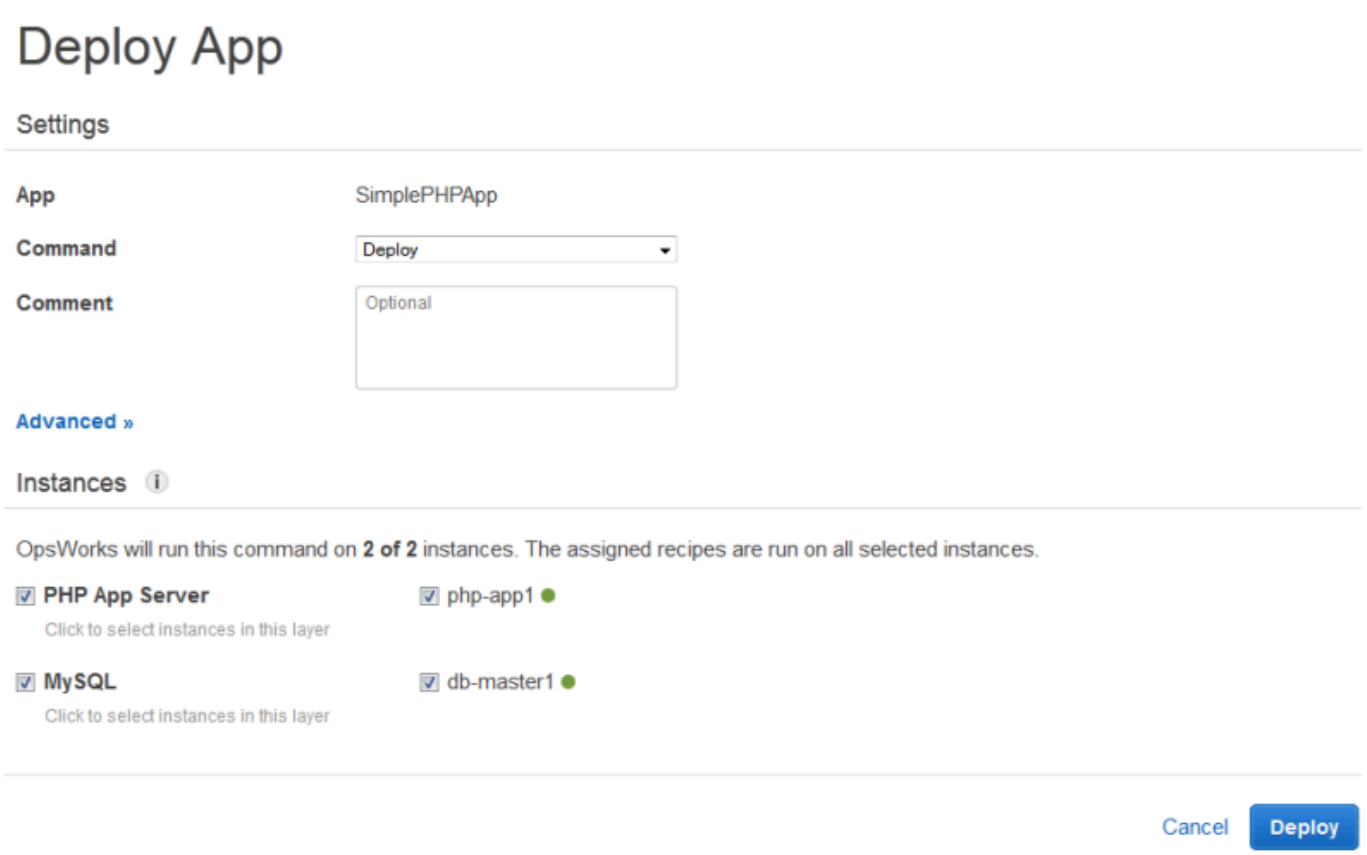
最后一步是部署 SimplePHPApp 的新版本。

部署 SimplePHPApp

1. 在 Apps 页面上，在 SimplePHPApp 应用程序的 Actions 中单击 deploy。



2. 接受默认设置，然后单击 Deploy。



单击 Deploy App 页面上的 Deploy 时，会触发“部署”生命周期事件，从而通知代理来运行“部署”配方。默认情况下，您会在所有堆栈实例上触发该事件。内置“部署”配方仅向该应用程序类型的相应实例 (在本案例中为 PHP App Server 实例) 部署应用程序。但是，在其他实例上触发“部署”事件的方法通常非常有用，这样可以让实例对应用程序部署进行响应。在本案例中，您还需要在 MySQL 实例上触发部署以设置数据库。

请注意以下几点：

- PHP App Server 实例上的代理会运行该层的内置配方，然后运行 `appsetup.rb`，该文件可以配置应用程序的数据库连接。
- MySQL 实例上的代理不会安装任何文件，但它会运行 `dbsetup.rb` 来创建 urler 表。

当部署完成后，Deployment 页面上的 Status 将变为 successful。

步骤 3.6：运行 SimplePHPApp

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

部署状态变为 successful 后，您可以运行新的 SimplePHPApp 版本，如下所示。

运行 SimplePHPApp

1. 在 Instances 页面上，单击 php-app1 行中的公有 IP 地址。

   您会在浏览器中看到以下页面。

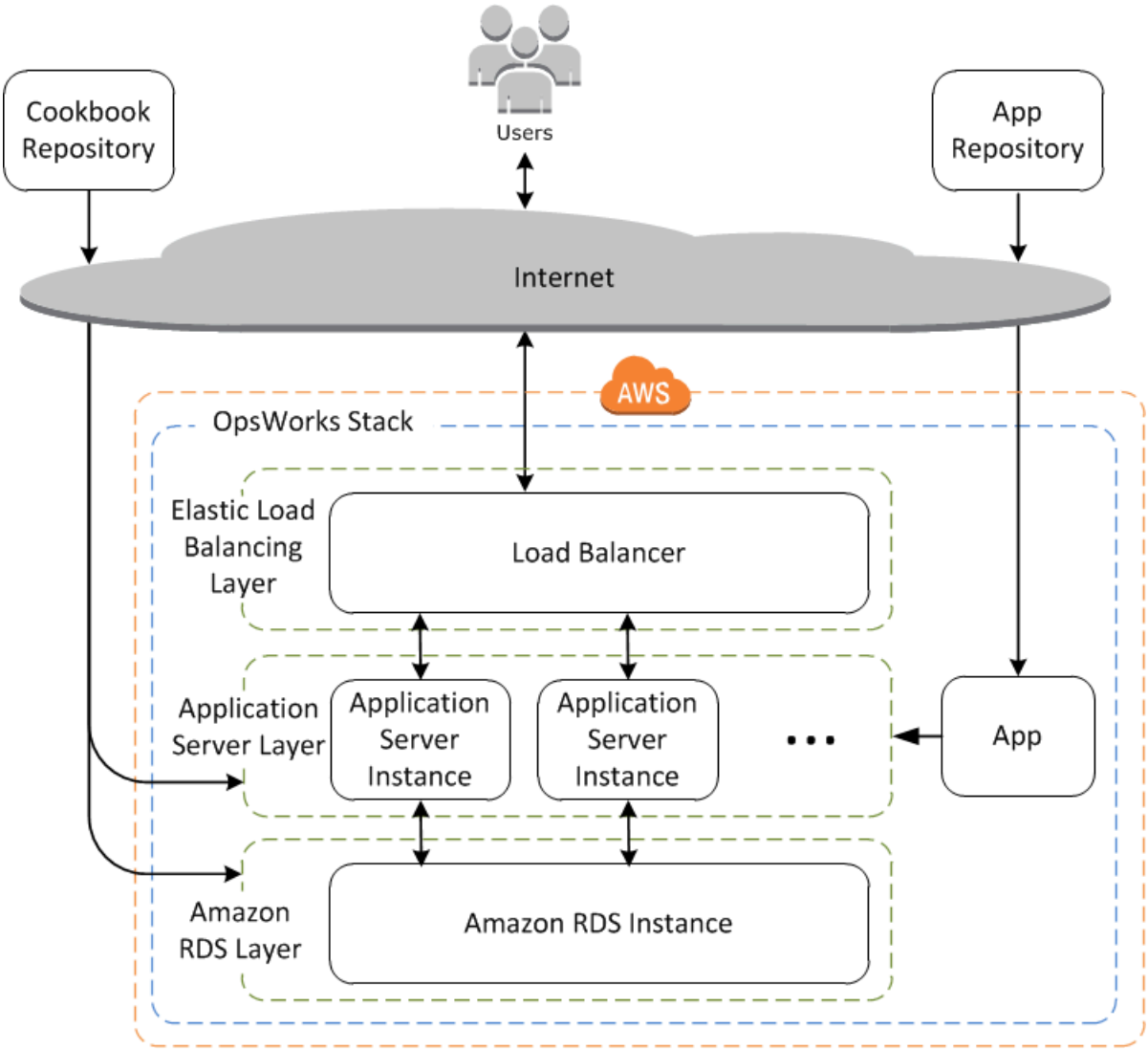
2. 单击分享您的想法，然后在您的想法中键入类似 **`Hello world!`** 的内容，在您的名字中键入您的名字。然后单击 Submit Your Thought 将消息添加到数据库。

3. 单击 Go Back 以查看数据库中的所有消息。

第 4 步：横向扩展 MyStack

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

MyStack 当前只有一台应用程序服务器。生产堆栈可能会需要多个应用程序服务器来处理传入流量和负载均衡器，从而将传入流量均匀地分布到应用程序服务器上。架构与以下内容类似：



AWS OpsWorks 堆栈可以轻松扩展堆栈。本节介绍如何通过向 Elastic Load Balancing 负载均衡器添加第二个全天候 PHP App Server 实例 MyStack 并将两个实例置于 Elastic Load Balancing 负载均衡器后面来扩展堆栈 您可以轻松扩展程序以添加任意数量的全天候实例，也可以使用基于时间或负载的实例让 Stacks 自动扩展您的 AWS OpsWorks 堆栈。有关更多信息，请参阅 使用基于时间和基于负载的实例管理负载。

步骤 4.1：添加负载均衡器

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

Elastic Load Balancing 是一种 Amazon Web Service，将传入的应用程序流量自动分配到多个 Amazon EC2 实例。除了分发流量之外，Elastic Load Balancing 还执行以下操作：

- 检测运行状况不佳的 Amazon EC2 实例。

  它将流量重新路由至其余运行正常的实例，直至运行不正常的实例恢复。
- 自动扩展请求处理容量以响应传入流量

> **Note**
>
> 负载均衡器有两种用途。一个显而易见的用途是使应用程序服务器上的负载达到均衡。此外，许多站点倾向于将其应用程序服务器和数据库与直接用户访问隔离开。使用 AWS OpsWorks Stacks，您可以通过在具有公有子网和私有子网的虚拟私有云 (VPC) 中运行堆栈来实现此目的，如下所示。
>
> - 将应用程序服务器和数据库置于私有子网中，VPC 中的其他实例可以访问该私有子网，但用户不能访问。
> - 将用户流量定向到公有子网中的负载均衡器，然后由负载均衡器将流量转发到私有子网中的应用程序服务器，并向用户返回响应。
>
> 有关更多信息，请参阅 在 VPC 中运行堆栈。要获取将本演练中的示例扩展到在 VPC 中运行的 AWS CloudFormation 模板，请下载该OpsWorksVPCtemplates.zip文件。

尽管 Elastic Load Balancing 通常称为层，但其工作方式与其他内置层略有不同。您无需创建层并向其添加实例，而是使用 Amazon EC2 控制台创建 Elastic Load Balancing 负载均衡器，然后将其连接到现有层之一（通常是应用程序服务器层）。 AWS OpsWorks 然后，堆栈将图层的现有实例注册到该服务，并自动添加任何新实例。以下过程介绍如何将负载均衡器添加到 MyStack的 PHP App Server 层。

> **Note**
>
> AWS OpsWorks 堆栈不支持 Application Load Balancer。您只能将 Classic Load Balancer 与 AWS OpsWorks 堆栈一起使用。

将负载均衡器附加到 PHP App Server 层

1. 使用 Amazon EC2 控制台为创建新的负载均衡器 MyStack。详细信息取决于您的账户是否支持 EC2 Classic。有关更多信息，请参阅 Elastic Load Balancing 入门。在运行 Create Load Balancer 向导时，按如下所示配置负载均衡器：

   定义负载均衡器

   为负载均衡器分配一个易于识别的名称（例如 PHP-LB），使其更容易在堆栈控制台中 AWS OpsWorks 找到。然后选择 Continue 以接受其余设置的默认值。

   如果您从 Create LB Inside 菜单中选择包含一个或多个子网的 VPC，则必须为要使用负载均衡器路由流量的每个可用区选择一个子网。

   分配安全组

   如果您的账户支持默认 VPC，则该向导将显示此页以确定负载均衡器的安全组。它不会为 EC2 Classic 显示此页。

   对于本演练，请选择 default VPC security group。

   配置安全设置

   如果您在 Define Load Balancer 页面上选择 HTTPS 作为 Load Balancer Protocol，请在此页面上配置证书、密码和 SSL 协议设置。对于本演练，请接受默认值并选择 Configure Health Check。

   配置运行状况检查

   将 Ping 路径设置为 **/** 并接受其余设置的默认值。

   添加 EC2 实例

   选择继续； AWS OpsWorks Stacks 会自动向负载均衡器注册实例。

   添加标签

   添加标签，以帮助您查找。每个标签都是一个键值对；例如，为了进行本演练，您可以将 **`Description`** 指定为键，将 **`Test LB`** 指定为值。

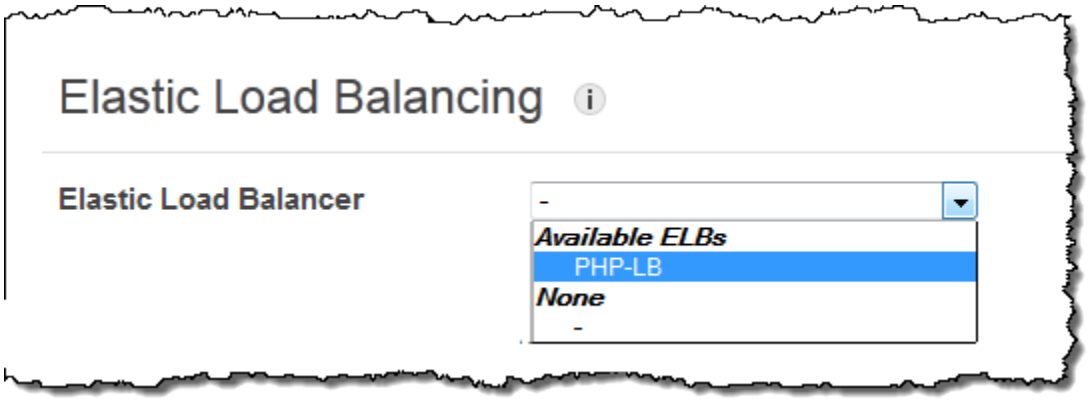
   审核

   检查您的选择，选择 Create，然后选择 Close，这将启动负载均衡器。

2. 如果在启动负载均衡器后您的账户支持默认 VPC，则必须确保其安全组具有合适的传入规则。默认规则不接受任何入站流量。

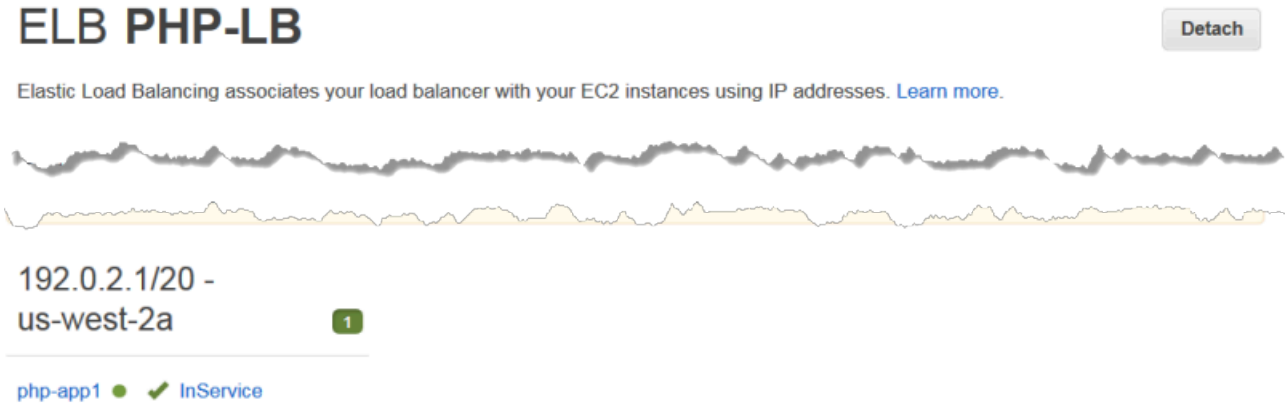
1. 在 Amazon EC2 导航窗格中，选择 Security Groups。
2. 选择 default VPC security group
3. 在 Inbound 选项卡上，选择 Edit。
4. 在本演练中，将 Source 设置为 Anywhere，这将指示负载均衡器接受来自任何 IP 地址的传入流量。

3. 返回 AWS OpsWorks Stacks 控制台。在 Layers 页面上，选择层的 Network 链接，然后选择 Edit。
4. 在 Elastic Load Balancing 下，选择您在步骤 1 中创建的负载均衡器，然后选择 Save。



将负载均衡器连接到层后， AWS OpsWorks Stacks 会自动注册该层的当前实例，并在新实例上线时添加这些实例。

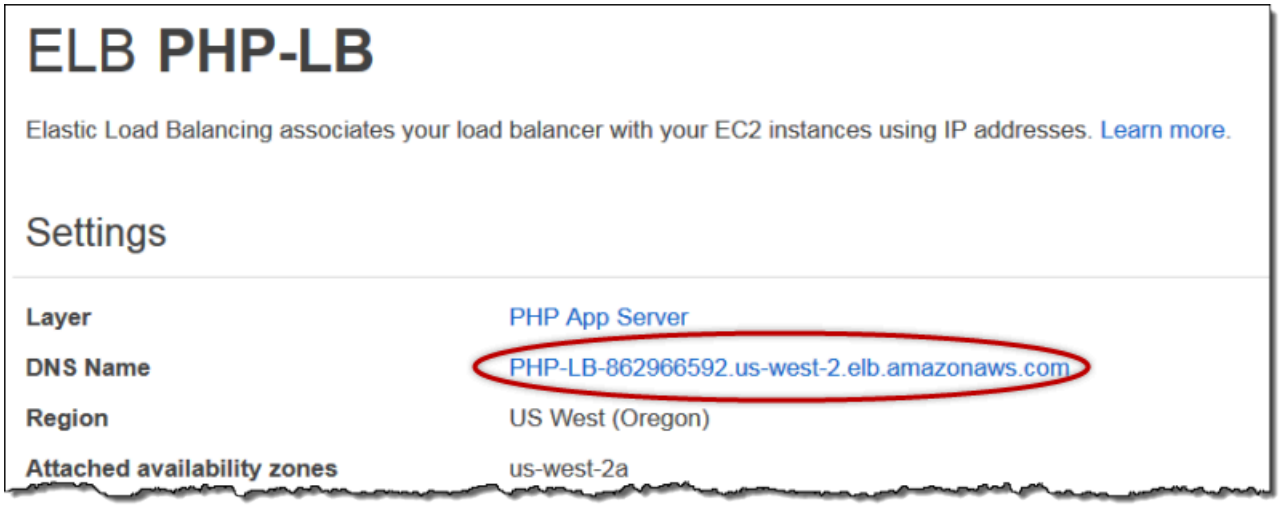
5. 在 Layers 页面上，单击负载均衡器的名称以打开其详细信息页面。注册完成且实例通过运行状况检查后， AWS OpsWorks Stacks 会在负载均衡器页面上的实例旁边显示一个绿色复选标记。



现在，您可以通过向负载均衡器发送请求的方法来运行 SimplePHPApp。

通过负载均衡器运行 SimplePHPApp

1. 再次打开负载均衡器的详细信息页面 (如果尚未打开的话)。
2. 在属性页面上，验证实例的运行状况检查状态，然后单击负载均衡器的 DNS 名称以运行 SimplePHPApp。负载均衡器会将请求转发到 PHP App Server 实例并返回响应，响应应与单击 PHP App Server 实例的公有 IP 地址时获得的响应完全相同。



> **Note**
>
> AWS OpsWorks 堆栈还支持 HAProxy 负载均衡器，这可能对某些应用程序有优势。有关更多信息，请参阅 HAProxy AWS OpsWorks 堆栈层。

步骤 4.2：添加 PHP 应用服务器实例

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

现在，负载均衡器已布置到位，您可以通过向 PHP App Server 层添加更多实例的方法来扩展堆栈。从您的角度来看，整个操作是无缝的。每次新的 PHP App Server 实例上线时， AWS OpsWorks Stacks 都会自动将其注册到负载均衡器并部署 SimplephPapp，这样服务器就可以立即开始处理传入流量。为

简洁起见，本主题介绍了如何添加一个额外 PHP App Server 实例，不过您可以使用相同的方法根据需要添加多个实例。

向 PHP App Server 层添加另一个实例

1. 在“Instances”页面上，单击 PHP App Server 下的 + Instance。
2. 接受默认设置，然后单击 Add Instance。
3. 单击 start 以启动实例。

步骤 4.3：监控 MyStack

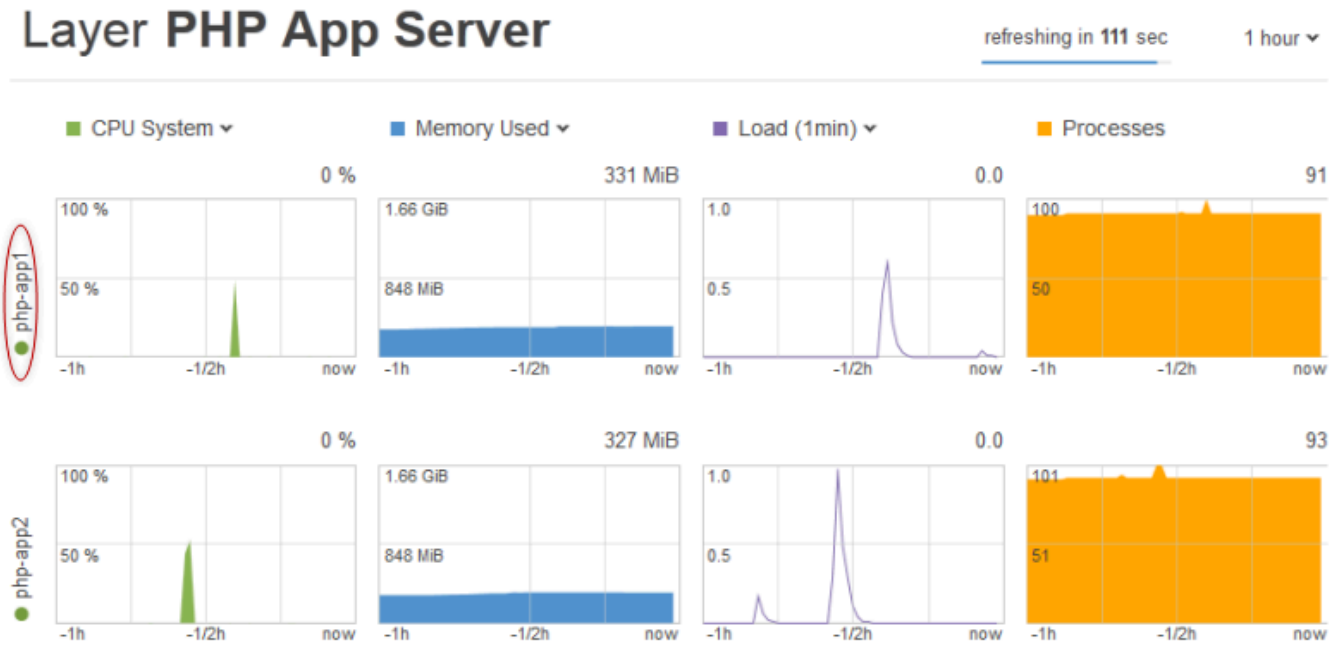
> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

AWS OpsWorks Stacks 使用 Amazon CloudWatch 为堆栈提供指标，并在 “监控” 页面上汇总这些指标，以方便您使用。您可以查看整个堆栈、指定层或指定实例的指标。

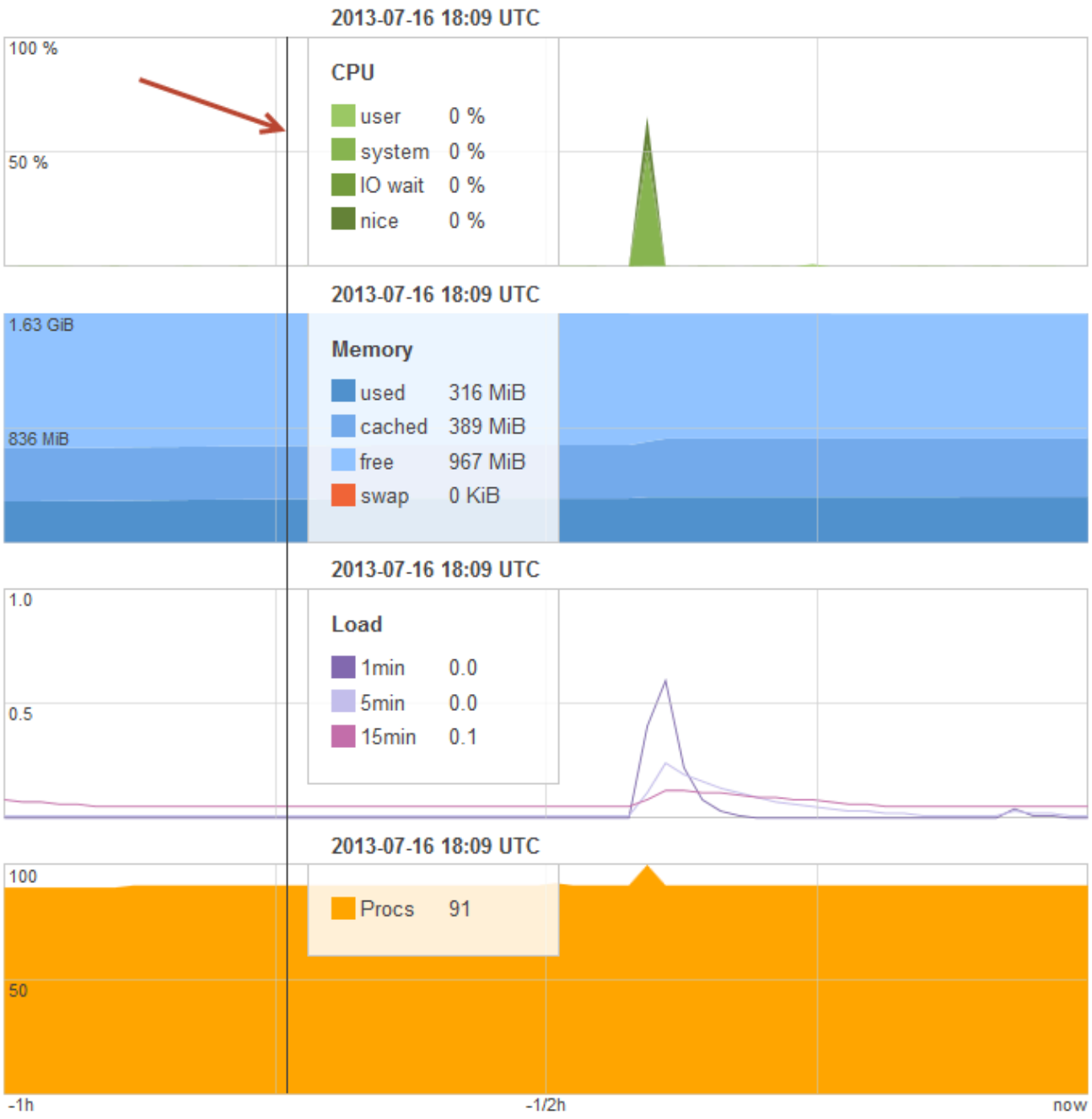
要监控 MyStack

1. 在导航窗格中，单击 Monitoring，这会显示一组图表，其中包含每个层的平均指标。您可以使用 CPU System、Memory Used 和 Load 的菜单，以显示不同的相关指标。

2. 单击 PHP App Server 以查看每个层实例的指标。



3. 单击 php-app1 以查看该实例的指标。您可以通过移动滑块的方法来查看任何特定时间点的指标。


Instance php-app1
refreshing in
2013-07-16 18:09 UTC
100 %
50 %
CPU
user 0 %
system 0 %
IO wait 0 %
nice 0 %
2013-07-16 18:09 UTC
1.63 GiB
836 MiB
Memory
used 316 MiB
cached 389 MiB
free 967 MiB
swap 0 KiB
2013-07-16 18:09 UTC
1.0
0.5
Load
1min 0.0
5min 0.0
15min 0.1
2013-07-16 18:09 UTC
100
50
Procs 91
-1h
-1/2h
now

> **Note**
>
> AWS OpsWorks Stacks 还支持 Ganglia 监控服务器，这可能对某些应用程序有优势。有关更多信息，请参阅 Ganglia 层。

第 5 步：删除 MyStack

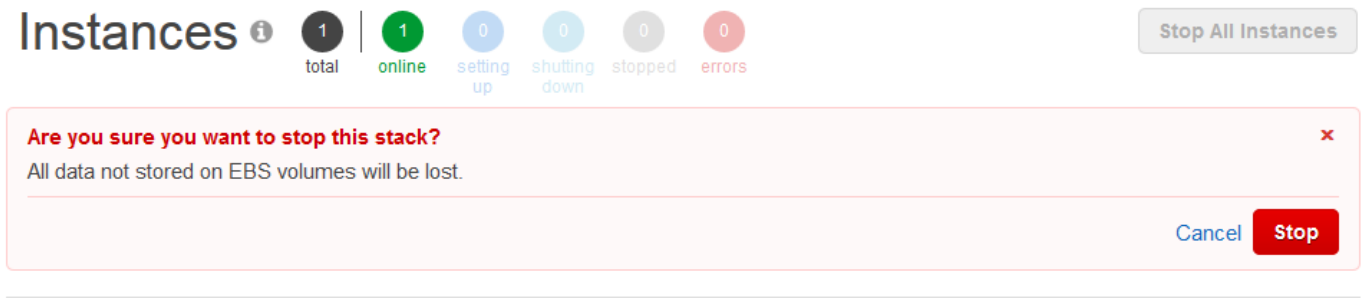
> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

一旦您开始使用 AWS 资源 (如 Amazon EC2 实例)，就需根据您的使用量付费。如果您现在已完成，则应停止实例，这样就不会产生任何不必要的费用。如果您不再需要堆栈，则可将其删除。

要删除 MyStack

1. 停止所有实例

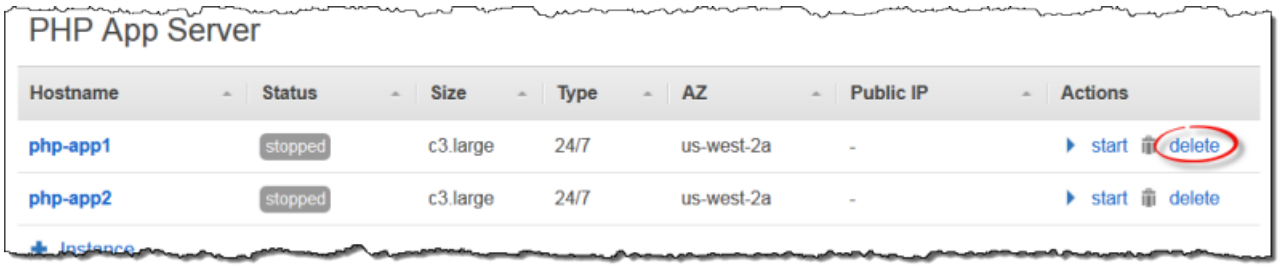
   在 Instances 页面上，单击 Stop All Instances，然后在要求您确认操作时单击 Stop。



   在您单击 “停止” 后， AWS OpsWorks Stacks 会终止关联的 Amazon EC2 实例，但不会终止任何关联资源，例如弹性 IP 地址或 Amazon EBS 卷。

2. 删除所有实例

   停止实例只会终止关联的 Amazon EC2 实例。当实例处于已停止状态时，您必须删除每个实例。在 PHP App Server 层中，单击 php-app1 实例的 Actions 列中的 delete。

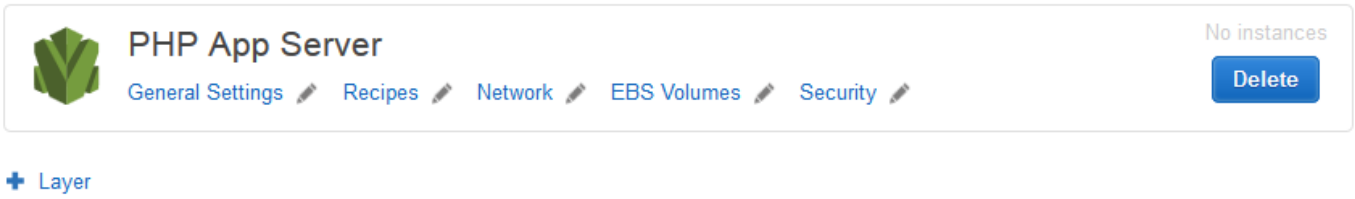


AWS OpsWorks 然后，Stacks 会要求您确认删除，并向您显示所有相关资源。您可以选择保留这些资源中的任意资源，或保留所有这些资源。此示例没有依赖资源，因此只需单击 Delete。

对 php-app2 和 MySQL 实例 db-master1 重复此过程。请注意，db-master1 具有关联的 Amazon Elastic Block Store 卷 (默认情况下已选定)。将其保持选定状态可删除卷以及实例。

3. 删除层。

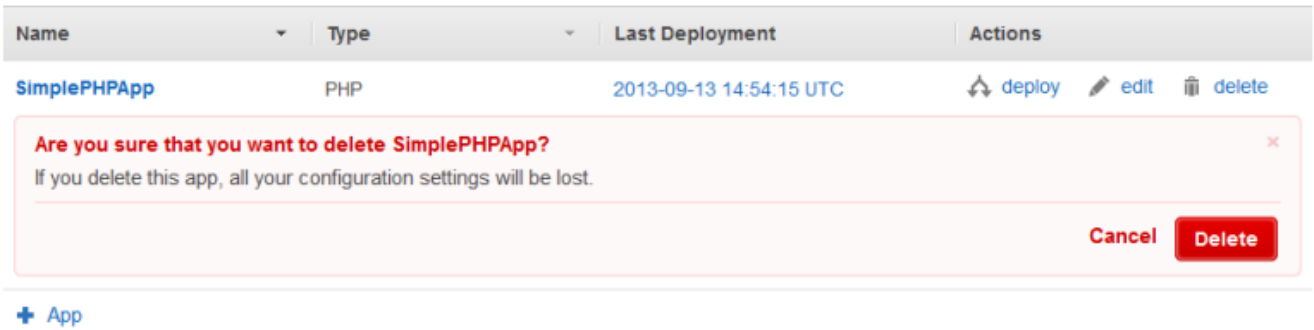
   在 Layers 页面上，单击 Delete，然后单击 Delete 以进行确认。



   对 MySQL 层重复此过程。

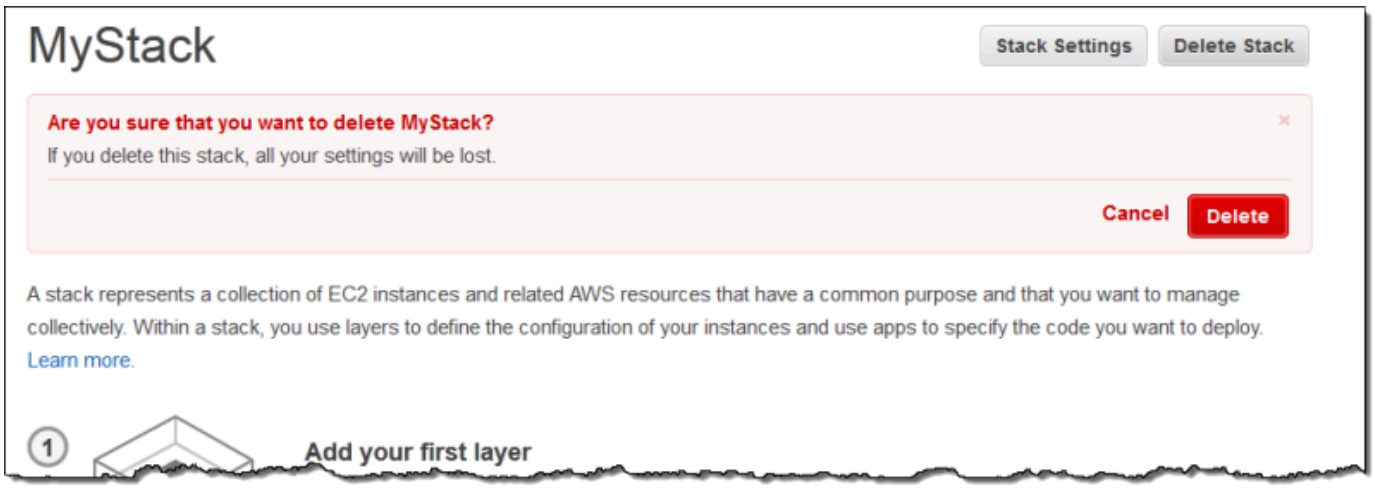
4. 删除应用程序

   在 Apps 页面上，单击 SimplePHPApp 应用程序的 Actions 列中的 delete，然后单击 Delete 以进行确认。



5. 删除 MyStack

   在 Stack 页面上，单击 Delete Stack，然后单击 Delete 以进行确认。



现在，您已经完成了本演练。

创建您的第一个 Node.js 堆栈

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

此示例介绍如何创建支持 Node.js 应用程序服务器的 Linux 堆栈以及如何部署简单的应用程序。堆栈包括以下组件：

- 包含两个实例的 Node.js App Server 层
- 一个 Elastic Load Balancing 负载均衡器，用于将流量分配给应用程序服务器实例
- Amazon Relational Database Service (Amazon RDS) 服务层，提供后端数据库

主题

- 先决条件
- 实施应用程序
- 创建数据库服务器和负载均衡器
- 创建堆栈
- 部署应用程序

- 接下来做什么？

先决条件

本演练假定：

- 您拥有 AWS 账户，并且对如何使用 AWS OpsWorks 堆栈有基本的了解。

  如果您不熟悉 AWS OpsWorks Stacks 或 AWS，请通过完成中的Chef 11 Linux 堆栈入门入门教程来学习基础知识。
- 您大致了解如何实施 Node.js 应用程序。

  如果您是首次使用 Node.js，请完成介绍性教程 (例如 节点：启动并运行) 以了解基础知识。
- 您已在 Amazon Web Services Region 中创建至少一个计划用于此示例的堆栈。

  当您在某个区域中创建第一个堆栈时，Stac AWS OpsWorks ks 会为每种层类型创建一个亚马逊弹性计算云 (Amazon EC2) 安全组。您需要这些安全组来创建 Amazon RDS 数据库 (DB) 实例。如果您不熟悉 AWS OpsWorks Stacks，我们建议您在本示例中Chef 11 Linux 堆栈入门使用与学习教程时相同的区域。如果您想使用新区域，请在该区域中创建一个新堆栈；该堆栈不需要具有层或实例。创建堆栈后， AWS OpsWorks Stacks 会自动向该区域添加一组安全组。
- 您将在默认 VPC 中创建您的堆栈。

  您可以在本演练中使用 EC2-Classic，但某些详细信息略为不同。例如，对于 EC2-Classic，您可以指定实例的可用区 (AZ) 而不是其子网。
- 您的 IAM 用户拥有 AWS OpsWorks 堆栈的完全访问权限。

  出于安全原因，强烈建议您不要在本演练中使用账户的根凭证。相反，创建一个拥有 AWS OpsWorks Stacks 完全访问权限的用户，然后在 Stacks 中使用这些证书。 AWS OpsWorks 有关更多信息，请参阅 创建 管理用户。

实施应用程序

本演练使用一个简单的 Express 应用程序，该应用程序连接到 Amazon RDS 数据库实例并列出该实例的数据库。

要实施应用程序，请在工作站上的方便位置创建一个名为 nodedb 的目录，并向该目录添加以下三个文件。

主题

- 程序包描述符
- 布局文件
- 代码文件

程序包描述符

要创建应用程序的程序包描述符，请将一个名为 package.json 的包含以下内容的文件添加到 nodedb 目录。package.json 是 Express 应用程序必需的，并且必须位于应用程序的根目录中。

```
{
  "name": "Nodejs-DB",
  "description": "Node.js example application",
  "version": "0.0.1",
  "dependencies": {
    "express": "*",
    "ejs": "*",
    "mysql": "*"
  }
}
```

此 package.json 示例是相当少的。它定义所需的 name 和 version 属性并列出依赖程序包：

- express 引用 Express 程序包。
- ejs 引用 EJS 程序包，应用程序使用此程序包将文本插入 HTML 布局文件中。
- mysql 引用 node-mysql 程序包，应用程序使用此程序包连接到 RDS 实例。

有关程序包描述符文件的更多信息，请参阅 package.json。

布局文件

要创建应用程序的布局文件，请将 views 目录添加到 nodedb 目录，然后将文件添加到一个名为 views 的包含以下内容的文件添加到 index.html：

```
<!DOCTYPE html>
<html>
<head>
  <title>AWS Opsworks Node.js Example</title>
</head>
```

```
<body>
  <h1>AWS OpsWorks Node.js Example</h1>
    <p>Amazon RDS Endpoint: <i><%= hostname %></i></p>
    <p>User: <i><%= username %></i></p>
    <p>Password: <i><%= password %></i></p>
    <p>Port: <i><%= port %></i></p>
    <p>Database: <i><%= database %></i></p>

    <p>Connection: <%= connectionerror %></p>
    <p>Databases: <%= databases %></p>
</body>
</html>
```

在本示例中，布局文件是一个简单的 HTML 文档，该文档将显示 Amazon RDS 中的某些数据。每个 <%= ... => 元素均表示一个在应用程序的代码文件 (我们接下来将创建该文件) 中定义的变量的值。

代码文件

要创建应用程序的代码文件，请将 server.js 文件与以下内容一起添加到 nodedb 目录。

> **Important**
>
> 使用 AWS OpsWorks Stacks，Node.js 应用程序的主代码文件必须命名server.js并位于应用程序的根文件夹中。

```
var express = require('express');
var mysql = require('mysql');
var dbconfig = require('opsworks'); //[1] Include database connection data
var app = express();
var outputString = "";

app.engine('html', require('ejs').renderFile);

//[2] Get database connection data
app.locals.hostname = dbconfig.db['host'];
app.locals.username = dbconfig.db['username'];
app.locals.password = dbconfig.db['password'];
app.locals.port = dbconfig.db['port'];
app.locals.database = dbconfig.db['database'];
app.locals.connectionerror = 'successful';
```

```
app.locals.databases = '';

//[3] Connect to the Amazon RDS instance
var connection = mysql.createConnection({
    host: dbconfig.db['host'],
    user: dbconfig.db['username'],
    password: dbconfig.db['password'],
    port: dbconfig.db['port'],
    database: dbconfig.db['database']
});

connection.connect(function(err)
{
    if (err) {
        app.locals.connectionerror = err.stack;
        return;
    }
});

// [4] Query the database
connection.query('SHOW DATABASES', function (err, results) {
    if (err) {
        app.locals.databases = err.stack;
    }

    if (results) {
        for (var i in results) {
            outputString = outputString + results[i].Database + ', ';
        }
        app.locals.databases = outputString.slice(0, outputString.length-2);
    }
});

connection.end();

app.get('/', function(req, res) {
    res.render('./index.html');
});

app.use(express.static('public'));

//[5] Listen for incoming requests
app.listen(process.env.PORT);
```

该示例显示数据库连接信息，还将查询数据库服务器并显示服务器的数据库。您可以根据需要轻松泛化它以便与数据库进行交互。以下说明是指上一代码中的已编号注释。

[1] 包括数据库连接数据

此 `require` 语句包含数据库连接数据。如后所述，当您将数据库实例附加到应用程序时， AWS OpsWorks Stacks 会将连接数据放在名为的文件中`opsworks.js`，该文件类似于以下内容：

```
exports.db = {
  "host":"nodeexample.cdlqlk5uwd0k.us-west-2.rds.amazonaws.com",
  "database":"nodeexampledb",
  "port":3306,
  "username":"opsworksuser",
  "password":"your_pwd",
  "reconnect":true,
  "data_source_provider":"rds",
  "type":"mysql"}
```

`opsworks.js` 位于应用程序的 `shared/config` 目录中，即 `/srv/www/app_shortname/shared/config`。但是， AWS OpsWorks Stacks 在应用程序的根目录`opsworks.js`中放置了指向的符号链接，因此您只需使用即可包含该对象。`require 'opsworks'`

[2] 获取数据库连接数据

这组语句通过以下方式显示 `opsworks.js` 的连接数据：将 db 对象中的值分配给一组 `app.locals` 属性（每个属性映射到 `index.html` 文件中的一个 <%= ... %> 元素）。呈现的文档将 <%= ... %> 元素替换为对应的属性值。

[3] 连接到 Amazon RDS 实例

该示例使用 `node-mysql` 访问数据库。为了连接到数据库，该示例会通过将连接数据传递到 `connection` 来创建 `createConnection` 对象，然后调用 `connection.connect` 来建立连接。

[4] 查询数据库

建立连接后，该示例会调用 `connection.query` 以查询数据库。此示例只查询服务器的数据库名称。`query` 返回一个 `results` 对象数组（每个数据库对应一个该对象）以及分配给 `Database` 属性的数据库名称。该示例将串联这些名称并将其分配给 `app.locals.databases,`，后者将在渲染的 HTML 页面中显示此列表。

本示例包含五个数据库，一个是您在创建 RDS 实例时指定的 nodeexampledb 数据库，其他四个是由 Amazon RDS 自动创建的数据库。

[5] 侦听传入请求

最后一个语句将侦听指定端口上的传入请求。您无需指定明确的端口值。将应用程序添加到堆栈时，需要指定该应用程序是支持 HTTP 还是 HTTPS 请求。 AWS OpsWorks 然后，Stacks 将PORT环境变量设置为 80 (HTTP) 或 443 (HTTPS)，您就可以在应用程序中使用该变量。

可以在其他端口上进行侦听，但是 Node.js App Server 层的内置安全组 AWS OpsWorks-nodejs-App-Server 仅允许进入端口 80、443 和 22 (SSH) 的入站用户流量。要允许流入其他端口的入站用户流量，请使用适当的入站规则创建安全组并将其分配给 Node.js App Server 层。请勿通过编辑内置安全组来修改入站规则。每次创建堆栈时，Stac AWS OpsWorks ks 都会使用标准设置覆盖内置安全组，因此您所做的任何更改都将丢失。

> **Note**
>
> 在创建或更新关联的应用程序时，可以将自定义环境变量与您的应用程序关联。您也可以通过使用自定义 JSON 和自定义配方将数据传递到您的应用程序。有关更多信息，请参阅 传递数据到应用程序。

创建数据库服务器和负载均衡器

此示例使用 Amazon RDS 数据库服务器和 Elastic Load Balancing 负载均衡器实例。您必须先单独创建每个实例，然后将其合并到堆栈中。本部分介绍如何创建新的数据库和负载均衡器实例。相反，您可以使用现有实例，但建议您通读此过程以确保正确配置这些实例。

下面介绍了如何创建足够此示例使用的最低配置的 RDS 数据库实例。有关更多信息，请参阅 Amazon RDS 用户指南。

创建 RDS 数据库实例

1. 打开 控制台。

   打开 Amazon RDS 控制台，并将区域设置为美国西部（俄勒冈州）。在导航窗格中，选择 RDS Dashboard，然后选择 Launch DB Instance。
2. 指定数据库引擎。

   选择 MySQL Community Edition 作为数据库引擎。

3. 拒绝多可用区部署。

   选择 No, this instance...，然后选择 Next。在此示例中，您不需要多可用区部署。

4. 配置基本设置。

   在 DB Instance Details 页面上，指定以下设置：

   - DB Instance Class：db.t2.micro
   - Multi-AZ Deployment：No
   - 分配的存储空间：**5** GB
   - 数据库实例标识符：**nodeexample**
   - 主用户名：**opsworksuser**
   - Master Password：所选密码

   记录实例标识符、用户名和密码以供将来使用，接受其他选项的默认设置，然后选择 Next。

5. 配置高级设置。

   在 Configure Advanced Settings 页面上，指定以下设置：

   - 数据库名称：**nodeexampledb**
   - 数据库安全组：AWS--db- Master OpsWorks- Server

   > **Note**
   >
   > AWS-OpsWorks db-Master-Server 安全组只允许您的堆栈的实例访问数据库。如果您需要直接访问数据库，请使用适当的入站规则将其他安全组连接到 RDS 数据库实例。有关更多信息，请参阅 Amazon RDS 安全组。您也可以通过将实例放置到 VPC 中来控制访问。有关更多信息，请参阅 在 VPC 中运行堆栈。

   记录数据库名称以供将来使用，接受其他设置的默认值，然后选择 Launch DB Instance。

以下过程介绍如何在此示例中创建 Elastic Load Balancing 负载均衡器。有关更多信息，请参阅 Elastic Load Balancing 用户指南。

创建负载均衡器

1. 打开 Amazon EC2 控制台。

   打开 Amazon EC2 控制台，并将区域设置为美国西部（俄勒冈州）。在导航窗格中，选择 Load Balancers，然后选择 Create Load Balancer。

2. 定义负载均衡器。

   在 Define Load Balancer 页面上，指定以下设置。

   - 名称 – **Node-LB**
   - 创建内部 LB - My Default VPC

   接受其他选项的默认设置，然后选择 Next。

3. 分配安全组。

   在 Assign Security Groups 页面上，指定以下组：

   - default VPC security group
   - AWS-nodej OpsWorks s-App-Server

   选择下一步。在 Configure Security Settings 页面上，选择 Next。在此示例中，您不需要安全侦听器。

4. 配置运行状况检查。

   在配置运行状况检查页面上，将 Ping 路径设置为 **/** 并接受其他设置的默认值。选择下一步。在 Add EC2 Instances 页面上，选择 Next。在 “添加标签” 页面上，选择 “查看并创建”。 AWS OpsWorks Stacks 负责向负载均衡器添加 EC2 实例，此示例不需要标签。

5. 创建负载均衡器。

   在 Review 页面上，选择 Create 以创建您的负载均衡器。

创建堆栈

现在，您已拥有创建堆栈所需的所有组件。

要创建 堆栈，请执行以下操作：

1. 登录 AWS OpsWorks Stacks 控制台。

   登录 AWS OpsWorks Stacks 控制台并选择 Add Stack (添加堆栈)。

2. 创建堆栈。

   要创建新的堆栈，请选择 Chef 11 stack，然后指定以下设置。

   - – **`NodeStack`**
   - 区域 - 美国西部（俄勒冈）

     虽然您可在任何 Amazon Web Services Region 中创建堆栈，但在教程中，建议您在美国西部（俄勒冈州）中创建堆栈。

   选择 Add Stack。有关堆栈配置设置的更多信息，请参阅创建新堆栈。

3. 使用连接的负载均衡器添加 Node.js App Server 层。

   在NodeStack页面上，选择添加图层，然后指定以下设置：

   - 图层类型 — Node.js App Server
   - 弹性负载均衡器 — Node-LB

   接受其他设置的默认值，然后选择 Add Layer。

4. 将实例添加到层并启动实例。

   在导航窗格中，选择 Instances，然后将两个实例添加到 Rails App Server 层，如下所示。

   1. 在 Node.js App Server 下，选择添加实例。

      将 Size 设置为 t2.micro，接受其他设置的默认值，然后选择 Add Instance。

   2. 选择 +Instance，然后将另一个 t2.micro 实例添加到其他子网中的层。

      这会将实例放置在其他可用区 (AZ) 中。

   3. 选择 Add instance。
   4. 要启动两个实例，请选择 Start All Instances。

您已向此层分配 Elastic Load Balancing 负载均衡器。当实例进入或离开在线状态时， AWS OpsWorks Stacks 会自动向负载均衡器注册或注销该实例。

> **Note**
>
> 对于生产堆栈，建议您跨多个可用区分配您的应用程序服务器实例。如果用户无法连接到可用区，负载均衡器会将传入流量路由到剩余区域中的实例，并且您的站点将继续工作。

5. 将 RDS 数据库实例注册到堆栈。

   在导航窗格中，选择 Resources 并将 RDS 数据库实例注册到堆栈，如下所示。

   1. 选择 RDS 选项卡，然后选择 Show Unregistered RDS DB 实例。
   2. 选择 nodeexampledb 实例，然后指定以下设置：
      - 用户 在创建实例时指定的主用户名；在此示例中为 **opsworksuser**。
      - 密码 - 创建实例时指定的主密码。
   3. 选择 注册到堆栈 以将 RDS 数据库实例作为 Amazon RDS 服务层添加到堆栈。

   > **Warning**
   >
   > AWS OpsWorks Stacks 不验证用户或密码值，它只是将它们传递给应用程序。如果您错误地输入这些值，您的应用程序将无法连接到数据库。

   选择 注册到堆栈 以将 RDS 数据库实例作为 Amazon RDS 服务层添加到堆栈。

部署应用程序

您必须在远程存储库中存储应用程序。部署时， AWS OpsWorks Stacks 会将代码和相关文件从存储库部署到应用服务器实例。为方便起见，此示例将公有 Amazon Simple Storage Service (Amazon S3) 存档用作存储库，但您也可以使用多种其他存储库类型，包括 Git 和 Subversion。有关更多信息，请参阅 应用程序源。

部署 应用程序

1. 将应用程序打包到存档文件中。

   创建 `.zip` 目录和子目录的 nodedb 存档，名为 nodedb.zip。您还可以使用其他类型的存档文件，包括 gzip、bzip2 和 tarball。请注意， AWS OpsWorks Stacks 不支持未压缩的压缩包。有关更多信息，请参阅 应用程序源。

2. 将存档文件上传到 Amazon S3。

   将 `nodedb.zip` 上传到 Amazon S3 存储桶，使该文件成为公有文件，然后复制文件的 URL 以供将来使用。有关如何创建存储桶和上传文件的更多信息，请转至 Amazon Simple Storage Service 入门指南。

   > **Note**
   >
   > AWS OpsWorks 堆栈也可以从 Amazon S3 存储桶部署私有文件，但为简单起见，此示例使用公共文件。有关更多信息，请参阅 应用程序源。

3. 创建 AWS OpsWorks Stacks 应用程序。

   返回 AWS OpsWorks Stacks 控制台，在导航窗格中选择 “应用程序”，然后选择 “添加应用程序”。指定以下设置：

   - 名称 – `NodeDB`。

     该字符串是应用程序的显示名称。在大多数情况下，您需要应用程序的短名称， AWS OpsWorks Stacks通过将所有字符转换为小写并删除标点符号从显示名称生成该名称。在本示例中，短名称是 nodedb。要验证应用程序的短名称，请在创建应用程序后，在 Apps 页上选择应用程序以显示其详细信息页面。

   - Type – `Node.js`。
   - Data source type - `RDS`。
   - Database instance - 选择之前注册的 Amazon RDS 数据库实例。
   - Database name - 指定您之前创建的数据库名称 (本示例中为 `nodeexampledb`)。
   - Repository type - `Http Archive`。

     您必须对公有 Amazon S3 文件使用此存储库类型。`S3 Archive` 类型仅用于私有存档。

   - Repository URL - 存档文件的 Amazon S3 URL。

对其余设置使用默认值，然后单击 Add App 创建应用程序。

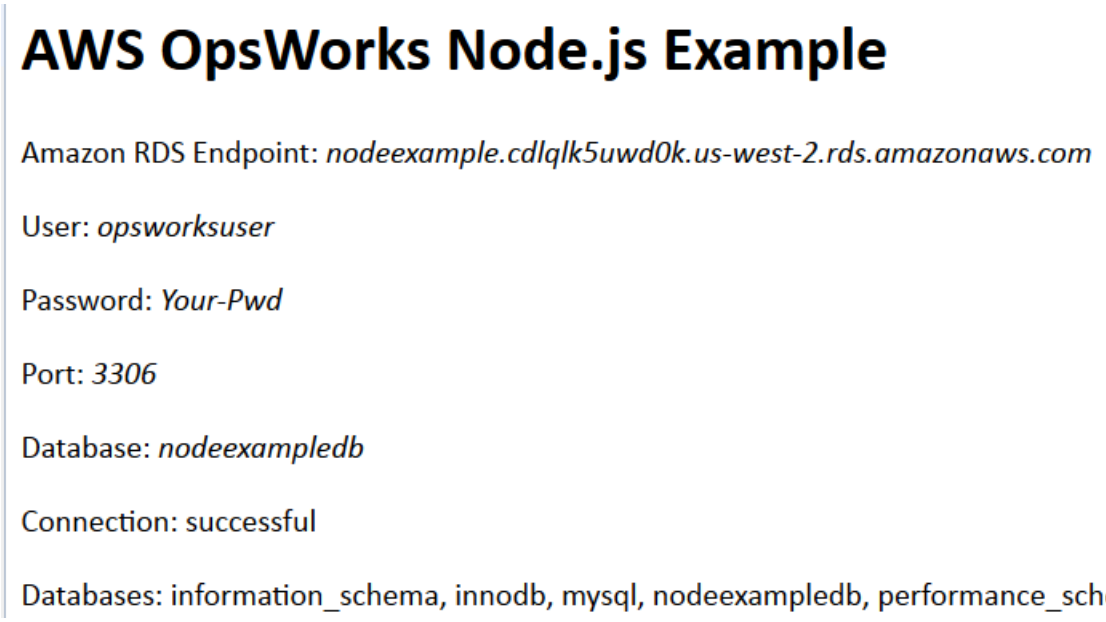
4. 部署应用程序。

   转至 Apps 页面，然后在 NodeDB 应用程序的 Actions 列中，选择 deploy。然后选择 De ploy 将应用程序部署到服务器实例。 AWS OpsWorks Stacks 在每个实例上运行 Deploy 配方，从存储库下载应用程序并重新启动服务器。当每个实例均有一个绿色复选标记且 Status 为 successful 时，表示部署已完成，并且应用程序已准备好开始处理请求。

   > **Note**
   >
   > 如果部署失败，请选择 Log 列中的 show 以显示部署的 Chef 日志。错误信息显示在底部附近。

5. 打开 应用程序。

   要打开应用程序，请选择 Layers，再选择负载均衡器，然后选择负载均衡器的 DNS 名称，这会将 HTTP 请求发送到负载均衡器。您应看到类似于以下内容的信息。

> **Note**
>
> AWS OpsWorks 在设置过程中，Stacks 会自动将应用程序部署到新实例。仅联机实例需要手动部署。有关更多信息，请参阅 部署应用程序。有关部署的一般讨论 (包括一些更高级的部署策略)，请参阅管理和部署应用程序和说明书。

接下来做什么？

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

本演练将指导您了解设置简单的 Node.js 应用程序服务器堆栈的基础知识。以下是有关后续操作的一些建议。

检查 Node.js 内置说明书

如果您想详细了解实例是如何配置的，请参阅该层的内置食谱 opsworks_nodejs，其中包含 AWS OpsWorks Stacks用于安装和配置软件的配方和相关文件，以及内置的部署食谱，其中包含Stacks用于部署应用程序的配方。 AWS OpsWorks

自定义服务器配置

该示例堆栈是非常基本的堆栈。要在生产中使用，您可能需要自定义堆栈。有关更多信息，请参阅自定义堆栈 AWS OpsWorks。

添加 SSL 支持

您可以为应用程序启用 SSL 支持，并在创建应用程序时向 AWS OpsWorks Stacks 提供相应的证书。 AWS OpsWorks 然后，Stacks 将证书安装到相应的目录中。有关更多信息，请参阅 使用 SSL。

添加内存中数据缓存

生产级站点通常会通过在内存中的关键字/值存储 (如 Redis 或 Memcache) 中缓存数据来提高性能。你可以将两者与堆 AWS OpsWorks 栈堆叠一起使用。有关更多信息，请参阅 ElastiCache Redis 和 Memcached。

使用更高级的部署策略

该示例使用一个简单的应用程序部署战略，该战略将更新同时部署到每个实例。此方法简单快捷，但不允许犯错误。如果部署失败或更新存在任何问题，则可能影响生产堆栈中的每个实例，从而可能中断或禁用您的站点，直到您解决该问题。有关部署策略的更多信息，请参阅管理和部署应用程序和说明书。

扩展 Node.js App Server 层

您可以通过多种方式扩展层。例如，您可以实施配方以在实例上运行脚本，或实施 Chef 部署说明书以自定义应用程序部署。有关更多信息，请参阅 扩展层。

定义环境变量

您可以通过定义关联应用程序的环境变量来将数据传递到您的应用程序。部署应用程序时， AWS OpsWorks Stacks 会导出这些变量，以便您可以从应用程序访问它们。有关更多信息，请参阅 使用环境变量。

自定义堆栈 AWS OpsWorks

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

AWS OpsWorks Stacks 内置层提供的标准功能足以满足多种用途。但是，您可能会遇到以下一种或多种情况：

- 某个内置层的标准配置只是够用，但不够理想；您想要针对特定要求优化它。

  例如，您可能希望为工作进程的最大数量或 `keepalivetimeout` 值等设置指定您自己的值，从而调整 Static Web Server 层的 Nginx 服务器配置。
- 某个内置层的功能很好，但您希望通过安装更多程序包或运行一些自定义安装脚本来扩展它。

  例如，您可能希望通过还安装 Redis 服务器来扩展 PHP App Server 层。
- 您有一些不由任何内置层处理的要求。

例如，AWS OpsWorks Stacks 不包括某些常用数据库服务器的内置层。您可以创建一个自定义层来将这些服务器安装在该层的实例上。

- 您正在运行一个 Windows 堆栈，该堆栈仅支持自定义层。

AWS OpsWorks Stacks 提供了多种自定义图层的方法，以满足您的特定要求。以下按照提高复杂性和增强功能的顺序列出了一些示例：

> **Note**
>
> 其中一些方法仅适用于 Linux 堆栈。有关详细信息，请参阅以下主题。

- 使用自定义 JSON 覆盖默认 AWS OpsWorks 堆栈设置。
- 使用可覆盖默认 AWS OpsWorks 堆栈设置的属性文件实现自定义 Chef 食谱。
- 使用覆盖或扩展默认 AWS OpsWorks Stacks 模板的模板实现自定义 Chef 食谱。
- 使用运行 shell 脚本的简单配方实施自定义 Chef 说明书。
- 使用执行创建和配置目录、安装程序包、创建配置文件、部署应用程序等任务的配方实施自定义 Chef 说明书。

您还可以覆盖配方，具体取决于堆栈的 Chef 版本和操作系统。

- 对于 Chef 0.9 和 11.4 堆栈，您无法通过实施具有与内置配方相同的说明书和配方名称的自定义配方来覆盖内置配方。

  对于每个生命周期事件，AWS OpsWorks Stacks 始终首先运行内置配方，然后运行任何自定义配方。由于这些 Chef 版本不会运行一个具有相同说明书和配方名称的配方两次，因此内置配方优先运行，而自定义配方不会执行。
- 您可以在 Chef 11.10 堆栈上覆盖内置配方。

  有关更多信息，请参阅 说明书安装和优先顺序。
- 您无法在 Windows 堆栈上覆盖内置配方。

  AWS OpsWorks Stacks 处理 Chef 在 Windows 堆栈中运行的方式不允许覆盖内置配方。

> **Note**
>
> 由于许多技术都使用自定义食谱，因此说明书和诀窍如果您还不熟悉食谱的实现，则应先阅读。 说明书基础知识提供了实现自定义食谱的详细教程介绍，并为 AWS OpsWorks 堆栈实现食谱介绍了有关如何为 AWS OpsWorks Stacks 实例实现食谱的一些细节。

主题

通过覆盖属性 AWS OpsWorks 来自定义堆栈配置

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 对于 Windows 堆栈和 Chef 12 Linux 堆栈， AWS OpsWorks Stacks 使用单独的 Chef 运行来制作内置食谱和自定义食谱。这意味着，您无法使用此节中讨论的方法来覆盖 Windows 堆栈和 Chef 12 Linux 堆栈的内置属性。

配方和模板取决于实例的各种 Chef 属性或堆栈特定信息 (如层配置或应用程序服务器设置)。这些属性具有多个源：

- 自定义 JSON （可选）-您可在创建、更新或克隆堆栈时或部署应用程序时指定自定义 JSON 属性。
- 堆栈配置属性 —AWS OpsWorks 堆栈定义这些属性以保存堆栈配置信息，包括您通过控制台设置指定的信息。

- 部署属性 — AWS OpsWorks 为 Deploy 事件定义与部署相关的属性。
- 说明书属性 内置说明书和自定义说明书通常包含一个或多个属性文件，这些文件中包含表示说明书特定值（如应用程序服务器配置设置）的属性。
- Chef - Chef 的 Ohai 工具定义了表示各种系统配置设置 (如 CPU 类型和安装的内存) 的属性。

有关堆栈配置和部署属性以及内置说明书属性的完整列表，请参阅堆栈配置和部署属性：Linux 和内置说明书属性。有关 Ohai 属性的更多信息，请参阅 Ohai。

当生命周期事件（如部署或配置）出现时，或您运行堆栈命令（如 `execute_recipes` 或 `update_packages`）时， AWS OpsWorks Stacks 将执行以下操作：

- 向每个受影响实例上的代理发送相应命令。

  代理将运行适当的配方。例如，对于部署事件，代理将运行内置部署配方，接着运行任何自定义部署配方。
- 将任何自定义 JSON 和部署属性与堆栈配置属性进行合并，然后在实例上安装它们。

自定义 JSON 中的属性、堆栈配置和部署属性、说明书属性以及 Ohai 属性将会合并到一个节点对象中，从而为配方提供属性值。涉及到堆栈配置属性时，包括任何自定义 JSON，实例实际上是无状态的。当您运行某个部署或堆栈命令时，关联的配方将使用通过该命令下载的堆栈配置属性。

主题

- 属性优先顺序
- 使用自定义 JSON 覆盖属性
- 使用自定义 AWS OpsWorks Cookbook 属性覆盖堆栈属性

属性优先顺序

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

如果一个属性是唯一定义的，Chef 则会简单地将它并入到节点对象中。但是，任何属性源可定义任何属性，因此有可能同一个属性会具有多个使用不同的值的定义。例如，内置 apache2 说明书定义 node[:apache][:keepalive]，但是您也可在自定义 JSON 或自定义说明书中定义该属性。如果一个属性具有多个定义，则将按下文中描述的顺序对这些定义进行评估，节点对象将接收具有最高最先顺序的定义。

属性的定义如下所示：

```
node.type[:attribute][:sub_attribute][:...]=value
```

如果一个属性有多个定义，则该类型决定哪个定义具有优先级，并将该定义合并到节点对象中。 AWS OpsWorks 堆栈使用以下属性类型：

- default-这是最常见的类型，并且它实际上意味着“如果属性尚未定义，请使用此值。” 如果属性的所有定义都是 default 类型，则评估顺序中的第一个定义具有优先顺序并且后续值将被忽略。请注意， AWS OpsWorks Stacks 将所有堆栈配置和部署属性定义设置为default类型。
- normal-此类型的属性将覆盖评估顺序中之前定义的任何 default 或 normal 属性。例如，如果第一个属性来自内置说明书并且具有 default 类型，第二个是具有 normal 类型的用户定义属性，则第二个定义具有优先顺序。
- set -这是您可能会在早期说明书中看到的已弃用类型。它已由具有相同的优先顺序的 normal 取代。

Chef 支持多种其他属性类型，包括优先于所有其他属性定义的 automatic 类型。Chef 的 Ohai 工具生成的属性定义均为 automatic 类型，因此它们实际上是只读的。这通常不是问题，因为没有理由重写它们，而且它们与 AWS OpsWorks Stacks 的属性不同。但是，您应谨慎为您的自定义说明书属性命名，以便将它们与 Ohai 属性区分开。有关更多信息，请参阅关于属性。

> **Note**
>
> Ohai 工具是您可通过命令行运行的可执行文件。要列出某个实例的 Ohai 属性，请登录该实例并在终端窗口中运行 ohai。请注意，此操作将生成一个非常长的输出。

下面是将各种属性定义并入到节点对象中的步骤：

1. 将任何自定义堆栈配置属性合并到堆栈配置和部署属性中。

可为堆栈或某个特殊部署设置自定义 JSON 属性。它们在评估顺序中优先并且实际上为 normal 类型。如果一个或多个堆栈配置属性也是在自定义 JSON 中定义的，则自定义 JSON 值将优先。否则， AWS OpsWorks Stacks 只是将自定义 JSON 属性并入到堆栈配置中。

2. 将任何部署自定义 JSON 属性合并到堆栈配置和部署属性中。

   部署自定义 JSON 属性实际上也是 normal 类型，因此它们优先于内置和自定义堆栈配置 JSON 以及内置部署 JSON。

3. 将堆栈配置和部署属性合并到实例的节点对象中。
4. 将实例的内置说明书属性合并到节点对象中。

   内置说明书属性均为 default 类型。如果一个或多个内置说明书属性也是在堆栈配置和部署属性中定义的(通常因为您使用自定义 JSON 定义它们)，堆栈配置定义将优先于内置说明书定义。所有其他内置说明书属性将简单地合并到节点对象中。

5. 将实例的自定义说明书属性合并到节点对象中。

   自定义说明书属性通常是 normal 或 default 类型。唯一属性将合并到节点对象中。如果步骤 1 至步骤 3 中还定义了任何自定义说明书属性 (通常是因为您使用自定义 JSON 定义了它们)，则优先顺序取决于自定义说明书属性的类型：

   - 步骤 1-步骤 3 中定义的属性优先于自定义说明书 default 属性。
   - 自定义说明书 normal 属性优先于步骤 1 - 步骤 3 中的定义。

> **Important**
>
> 请勿使用自定义说明书 default 属性覆盖堆栈配置或内置说明书属性。由于自定义说明书属性将最后进行评估，因此 default 属性具有最低的优先顺序，并且无法覆盖任何属性。

使用自定义 JSON 覆盖属性

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 由于 AWS OpsWorks Stacks 处理 Windows 堆栈的 Chef 运行方式与 Linux 堆栈不同，因此你不能将本节中讨论的技术用于 Windows 堆栈。

重写 AWS OpsWorks Stacks 属性的最简单方法是在自定义 JSON 中对其进行定义，该属性优先于堆栈配置和部署属性以及内置和自定义食谱`default`属性。有关更多信息，请参阅 属性优先顺序。

> **Important**
>
> 您应谨慎地覆盖堆栈配置和部署属性。例如，覆盖 `opsworks` 命名空间中的属性可能会影响内置配方。有关更多信息，请参阅 堆栈配置和部署属性。

您还可使用自定义 JSON 来定义唯一属性 (通常用于将数据传递到您的自定义配方)。这些属性将简单地并入到节点对象中，并且配方可通过使用标准 Chef 节点语法来引用它们。

如何指定自定义 JSON

要使用自定义 JSON 来覆盖某个属性值，您必须先确定该属性的完全限定属性名称。然后，您创建一个包含要覆盖、设置为您的首选值的属性的 JSON 对象。为方便起见，堆栈配置和部署属性：Linux 和内置说明书属性文档一般使用堆栈配置、部署和内置说明书属性，包括其完全限定名称。

对象的父子关系必须对应于适当的完全限定的 Chef 节点。例如，假设您需要更改以下 Apache 属性：

- `keepalivetimeout` 属性，其节点为 `node[:apache][:keepalivetimeout]` 并且具有默认值 `3`。
- `logrotate` `schedule` 属性，其节点为 `node[:apache][:logrotate][:schedule]`，并且具有默认值 `"daily"`。

要覆盖这些属性并将值分别设置为 `5` 和 `"weekly"`，可使用以下自定义 JSON：

```
{
  "apache" : {
    "keepalivetimeout" : 5,
    "logrotate" : {
```

```
        "schedule" : "weekly"
    }
  }
}
```

何时指定自定义 JSON

您可为下列任务指定自定义 JSON 结构：

- 创建新堆栈
- 更新堆栈
- 运行堆栈命令
- 克隆堆栈
- 部署应用程序

对于每项任务， AWS OpsWorks Stacks 都会将自定义 JSON 属性与堆栈配置和部署属性合并，然后将其发送到实例，然后合并到节点对象中。但是，请注意以下事项：

- 如果您在创建、克隆或更新堆栈时指定自定义 JSON，则这些属性将会合并到用于所有后续事件和堆栈命令的堆栈配置和部署属性中。
- 如果您为部署指定自定义 JSON，则这些属性将会合并到仅用于相应的事件的堆栈配置和部署属性中。

  如果您想要为后续部署使用这些自定义属性，必须再次显式指定自定义 JSON。

请务必记住，这些属性仅在配方使用它们时才会影响实例。如果您覆盖某个属性值，但没有后续配方引用该属性，则更改不会产生影响。您必须确保在关联的配方运行之前发送自定义 JSON，或者确保重新运行适当的配方。

自定义 JSON 的最佳实践

你可以使用自定义 JSON 来覆盖任何 AWS OpsWorks Stacks 属性，但是手动输入信息有点麻烦，而且不受任何形式的源代码控制。自定义 JSON 最适合以下用途：

- 您只想覆盖少量属性，并且您无需使用自定义说明书。

  通过自定义 JSON，您可避免仅为了覆盖一些属性而设置和维护说明书存储库所带来的开销。
- 敏感值，如密码或身份验证密钥。

说明书属性存储在一个存储库中，因此所有敏感信息都有泄露风险。可以改为定义使用虚拟值的属性，然后使用自定义 JSON 设置真实值。

- 预计会变化的值。

  例如，建议的做法是通过不同的开发和暂存堆栈来支持您的生产堆栈。假设这些堆栈支持某个用于接受付款的应用程序。如果您使用自定义 JSON 来指定付款终端节点，则可以为您的暂存堆栈指定一个测试 URL。当您准备好将更新后的堆栈迁移到生产堆栈时，可使用相同的说明书并使用自定义 JSON 来将付款终端节点设置为生产 URL。

- 特定于某个特殊堆栈或部署命令的值。

使用自定义 AWS OpsWorks Cookbook 属性覆盖堆栈属性

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 对于 Windows 堆栈， AWS OpsWorks Stacks 使用单独的 Chef 运行来生成内置食谱和自定义食谱。这意味着，您无法使用此节中讨论的方法来覆盖 Windows 堆栈的内置属性。

自定义 JSON 是覆盖 AWS OpsWorks Stacks 堆栈配置和内置食谱属性的便捷方法，但它有一些限制。具体而言，您必须在每次使用时手动输入自定义 JSON，因此您没有管理定义的稳健方式。更好的方式一般是使用自定义说明书属性文件来覆盖内置属性。这样做可让您将定义置于源代码管理之下。

使用自定义属性文件覆盖 AWS OpsWorks 堆栈定义的过程非常简单。

重写 AWS OpsWorks Stacks 属性定义

1. 如说明书和诀窍中所述设置说明书存储库。
2. 使用与包含您要覆盖的属性的内置说明书相同的名称创建一个说明书。例如，要覆盖 Apache 属性，说明书应命名为 apache2。
3. 将 `attributes` 文件夹添加到说明书并将一个名为 `customize.rb` 的文件添加到该文件夹中。

4. 将一个属性定义添加到您要覆盖、设置为您的首选值的每个内置说明书属性的文件中。该属性必须是normal类型或更高的类型，并且与相应的 AWS OpsWorks Stacks 属性具有完全相同的节点名称。有关 AWS OpsWorks 堆栈属性的详细列表，包括节点名称，请参阅堆栈配置和部署属性：Linux和。内置说明书属性有关属性和属性文件的更多信息，请参阅关于属性文件。

   > **Important**
   >
   > 您的属性必须是normal类型才能覆盖 AWS OpsWorks Stacks 属性；default类型没有优先级。例如，如果您的 customize.rb 文件包含 default[:apache][:keepalivetimeout] = 5 属性定义，则内置 apache.rb 属性文件中的对应属性将先进行评估，并且优先。有关更多信息，请参阅 覆盖属性。

5. 对于包含您要覆盖的属性的每个内置说明书，重复步骤 2 至 4。
6. 为您的堆栈启用自定义食谱，并提供 Stac AWS OpsWorks ks 将您的食谱下载到堆栈实例所需的信息。有关更多信息，请参阅 安装自定义说明书。

> **Note**
>
> 有关此过程的完整演练，请参阅覆盖内置属性。

后续生命周期事件、部署命令和堆栈命令使用的节点对象现在将包含您的属性定义而不是 AWS OpsWorks 堆栈值。

例如，要覆盖keepalivetimeout 中讨论的内置 logrotate schedule 和 如何指定自定义 JSON 设置，请将 apache2apache 说明书添加到您的存储库并将 customize.rb 文件添加到说明书中包含下列内容的 attributes 文件夹。

```
normal[:apache][:keepalivetimeout] = 5
normal[:apache][:logrotate][:schedule] = 'weekly'
```

> **Important**
>
> 不应通过修改关联的内置属性文件的副本来覆盖 AWS OpsWorks Stacks 属性。例如，如果您将 apache.rb 复制到您的 apache2/attributes 文件夹并修改其部分设置，实际上将会覆盖内置文件中的所有属性。配方将使用副本中的属性定义并忽略内置文件。如果 AWS OpsWorks Stacks 之后修改内置属性文件，除非您手动更新副本，否则配方将无权访问更改。

为避免这种情况，所有内置说明书都包含空的 `customize.rb` 属性文件，该文件是所有模块中通过 `include_attribute` 指令要求的。通过覆盖 `customize.rb` 的副本中的属性，您仅影响这些特定属性。配方将获取内置属性文件中的任何其他属性值，并且将自动获取您尚未覆盖的任何属性的当前值。
此方法将帮助您在说明书存储库中维持少量的属性，从而降低您的维护开销并使未来的升级管理变得更轻松。

使用自定义模板扩展 AWS OpsWorks 堆栈配置文件

Important

该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

Note

由于 AWS OpsWorks Stacks 处理 Windows 堆栈的 Chef 运行方式与 Linux 堆栈不同，因此你不能将本节中讨论的技术用于 Windows 堆栈。

AWS OpsWorks Stacks 使用模板来创建配置文件等文件，这些文件通常依赖于许多设置的属性。如果您使用自定义 JSON 或自定义食谱属性来覆盖 AWS OpsWorks 堆栈定义，则您的首选设置将合并到配置文件中，而不是 AWS OpsWorks 堆栈设置。但是， AWS OpsWorks Stacks 不一定要为每个可能的配置设置指定一个属性；它接受某些设置的默认值，而直接在模板中对其他设置进行硬编码。如果没有相应的 AWS OpsWorks Stacks 属性，则无法使用自定义 JSON 或自定义食谱属性来指定首选设置。

您可以通过创建自定义模板来扩展配置文件以包含额外的配置设置。随后，您可以向文件添加任何配置设置或其他必需内容，并覆盖任何硬编码的设置。有关模板的更多信息，请参阅模板。

Note

您可以覆盖任何内置模板，但 opsworks-agent.monitrc.erb 除外。

创建自定义模板

1. 创建一个具有与内置说明书相同的结构和目录名的说明书。然后，在适当的目录中创建一个与您要自定义的内置模板同名的模板文件。例如，要使用自定义模板扩展 Apache `httpd.conf` 配置文件，您必须在存储库中实施 `apache2` 说明书，并且您的模板文件必须为 `apache2/templates/default/apache.conf.erb`。使用完全相同的名称允许 AWS OpsWorks Stacks 识别自定义模板并使用它来代替内置模板。

   最简单的方法是将内置模板文件从内置食谱的 GitHub存储库复制到您的食谱中，然后根据需要进行修改。

   > **Important**
   >
   > 请不要复制内置说明书中的任何文件，但要自定义的模板文件除外。复制其他类型的说明书文件 (如配方) 将创建重复的 Chef 资源，并且可能导致错误。

   说明书还可以包含自定义属性、配方和相关文件，但其文件名不应与内置文件名重复。
2. 自定义模板文件以生成符合您要求的配置文件。您可以添加更多设置、删除现有设置、替换硬编码的属性等。
3. 如果您尚未执行此操作，请编辑堆栈设置以启用自定义说明书并指定说明书存储库。有关更多信息，请参阅 安装自定义说明书。

> **Note**
>
> 有关此过程的完整演练，请参阅覆盖内置模板。

您无需实现任何配方或在图层配置中添加配方即可覆盖模板。 AWS OpsWorks Stacks 始终运行内置配方。当此堆栈运行创建配置文件的配方时，它将自动使用您的自定义模板，而不是使用内置模板。

> **Note**
>
> 如果 AWS OpsWorks Stacks 对内置模板进行任何更改，则您的自定义模板可能会不同步，无法再正常工作。例如，假设您的模板引用了一个依赖文件，并且文件名发生了变化。 AWS OpsWorks Stacks 不经常进行此类更改，当模板发生更改时，它会列出更改，并允许您选择升

级到新版本。您应该监控 AWS OpsWorks Stacks 存储库中是否有更改，并根据需要手动更新您的模板。

扩展层

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

有时，您需要通过修改 AWS OpsWorks Stacks 属性或自定义模板来将内置层自定义到无法处理的程度。例如，假设您需要创建符号链接、设置文件或文件夹模式、安装其他程序包等。您必须扩展自定义层以提供最低限度功能之外的功能。在这种情况下，您将需要实施包含配方的一个或多个自定义说明书来处理自定义任务。本主题提供了一些介绍如何使用配方来扩展层的示例。

如果您在 Chef 方面是新手，您应当先阅读说明书 101，这是一个教程，旨在介绍关于如何实施说明书以执行各种常见任务的基本知识。有关如何实施自定义层的详细示例，请参阅创建自定义 Tomcat 服务器层。

主题

- 使用配方运行脚本
- 使用 Chef 部署挂钩
- 在 Linux 实例上运行 Cron 作业
- 在 Linux 实例上安装和配置程序包

使用配方运行脚本

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

如果您已经有一个用于执行所需的自定义任务的脚本，则扩展层的最简单方法通常是实施一个运行该脚本的简单配方。您随后可以将该配方分配到相应的生命周期事件 (通常是设置或部署)，或者使用 `execute_recipes` 堆栈命令手动运行该配方。

以下示例在 Linux 实例上运行 shell 脚本，但您可以对其他类型的脚本（包括 Windows PowerShell 脚本）使用相同的方法。

```
cookbook_file "/tmp/lib-installer.sh" do
  source "lib-installer.sh"
  mode 0755
end

execute "install my lib" do
  command "sh /tmp/lib-installer.sh"
end
```

`cookbook_file` 资源表示一个存储在说明书的 `files` 目录的子目录中的文件，然后将该文件传输到实例上的指定位置。本示例会将 shell 脚本 `lib-installer.sh` 传输到实例的 `/tmp` 目录并将该文件的模式设置为 `0755`。有关更多信息，请参阅 cookbook_file。

`execute` 资源表示一个命令，如 shell 命令。此示例运行 `lib-installer.sh`。有关更多信息，请参阅 execute。

您还可以通过将脚本集成到配方中来运行脚本。以下示例将运行 bash 脚本，但 Chef 还支持 Csh、Perl、Python 和 Ruby。

```
script "install_something" do
  interpreter "bash"
  user "root"
  cwd "/tmp"
  code <<-EOH
    #insert bash script
  EOH
end
```

`script` 资源表示一个脚本。该示例指定了 bash 解释器，将用户设置为 `"root"`，并将工作目录设置为 `/tmp`。然后，它在 `code` 块中运行了 bash 脚本，该块可根据需要包含任意数量的行。有关更多信息，请参阅脚本。

有关如何使用配方运行脚本的更多信息，请参阅示例 7：运行命令和脚本。有关如何在 Windows 实例上运行 PowerShell 脚本的示例，请参阅运行 Windows PowerShell 脚本。

使用 Chef 部署挂钩

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS：Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

您可以实施自定义配方来执行所需的任务并将该配方分配到相应层的部署事件，从而自定义部署。一个替代方法是使用 Chef 部署挂钩运行您的自定义代码，这个方法有时候更简单，尤其是在您出于其他原因无需实施说明书时。此外，在部署后运行的自定义部署配方已由内置配方执行。利用部署挂钩，您可以在部署期间进行交互，例如，在已将应用程序的代码从存储库签出但尚未重新启动 Apache 时。

Chef 分四个阶段部署应用程序：

- 签出-从存储库下载文件
- 迁移-根据需要运行迁移
- 符号链接-创建符号链接
- 重新启动-重新启动应用程序

Chef 部署挂钩提供了一种自定义部署的简单方法，即在每个阶段完成后选择性地运行用户提供的 Ruby 应用程序。要使用部署挂钩，请实施一个或多个 Ruby 应用程序，然后将其放在您的应用程序的 `/deploy` 目录中。(如果您的应用程序没有 `/deploy` 目录，请在 `APP_ROOT` 级别创建一个。) 该应用程序必须具有以下名称之一，该名称决定了应用程序运行的时间。

- `before_migrate.rb` 在签出阶段已完成但迁移尚未开始时运行。
- `before_symlink.rb` 在迁移阶段已完成但符号链接尚未开始时运行。
- `before_restart.rb` 在符号链接阶段已完成但重新启动尚未开始时运行。
- `after_restart.rb` 在重新启动阶段完成后运行。

Chef 部署挂钩可使用标准节点语法访问节点对象，就像配方一样。部署挂钩还可访问您已指定的任何应用程序环境变量的值。但是，您必须使用 `new_resource.environment["VARIABLE_NAME"]` 访问该变量的值，而不是使用 `ENV["VARIABLE_NAME"]`。

在 Linux 实例上运行 Cron 作业

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

Linux cron 作业指示 cron 守护程序按指定的时间表运行一个或多个命令。例如，假设您的堆栈支持某个 PHP 电子商务应用程序。您可以设置一个 cron 作业来让服务器在每周的指定时间向您发送销售报告。有关 cron 的更多信息，请参阅 Wikipedia 上的 cron。有关如何直接在基于 Linux 的计算机或实例上运行 cron 任务的更多信息，请参阅印第安纳大学知识库网站上的什么是 cron 和 crontab 以及如何使用它们？

尽管您可以通过使用 SSH 连接到 `cron` 任务并编辑其 `crontab` 条目来在各个基于 Linux 的实例上手动设置这些任务，但 AWS OpsWorks Stacks 的关键优势在于您可以指示它跨整层的实例运行该任务。以下过程将介绍如何在 PHP App Server 层的实例上设置 `cron` 任务，但您可以对任何层使用同一方法。

在层的实例上设置 **`cron`** 作业

1. 使用设置作业的 `cron` 资源实施一个包含配方的说明书。该示例假定配方已命名为 `cronjob.rb`；实施详细信息如下文所述。有关说明书和配方的更多信息，请参阅说明书和诀窍。
2. 在堆栈上安装说明书。有关更多信息，请参阅 安装自定义说明书。
3. 通过将配方分配给以下生命周期事件，让 AWS OpsWorks Stacks 在层的实例上自动运行配方。有关更多信息，请参阅 自动运行配方。

   - 设置 — `cronjob.rb` 为该事件分配指示 AWS OpsWorks Stacks 在所有新实例上运行配方。
   - 部署-分配`cronjob.rb`给此事件会指示 AWS OpsWorks Stacks 在将应用程序部署或重新部署到该层时在所有在线实例上运行配方。

   您也可以使用 `Execute Recipes` 堆栈命令来在联机实例上手动运行该配方。有关更多信息，请参阅 运行堆栈命令。

下面是 cronjob.rb 示例，该示例设置了一个 cron 作业来每周运行一次用户实施的 PHP 应用程序，该应用程序将从服务器收集销售数据并通过电子邮件发送报告。有关如何使用 cron 资源的更多示例，请参阅 cron。

```
cron "job_name" do
  hour "1"
  minute "10"
  weekday "6"
  command "cd /srv/www/myapp/current && php .lib/mailing.php"
end
```

cron 是一种表示 cron 作业的 Chef 资源。当 AWS OpsWorks Stacks 在实例上运行配方时，相关的提供商会处理设置任务的细节。

- *job_name* 是 cron 作业的用户定义的名称（如 weekly report）。
- hour/minute/weekday 指定这些命令应在何时运行。本示例在每周六凌晨 1:10 运行这些命令。
- command 指定要运行的命令。

  本示例运行两个命令。首先导航到 /srv/www/myapp/current 目录。第二个命令运行用户实施的 mailing.php 应用程序，该应用程序将收集销售数据并发送报告。

> **Note**
>
> 默认情况下，bundle 命令不适用于 cron 作业。原因是 AWS OpsWorks Stacks 在目录中安装了/usr/local/bin捆绑器。要将 bundle 与 cron 作业结合使用，您必须将路径 /usr/local/bin 显式添加到 cron 作业。此外，由于 $PATH 环境变量不能在 cron 作业中扩展，因此最佳实践是将所有必需的路径信息显式添加到该作业而不依赖于 $PATH 变量的扩展。以下示例显示了在 cron 中使用 bundle 作业的两种方法。
>
> ```
> cron "my first task" do
>   path "/usr/local/bin"
>   minute "*/10"
>   command "cd /srv/www/myapp/current && bundle exec my_command"
> end
> ```

```
cron_env = {"PATH" => "/usr/local/bin"}
cron "my second task" do
  environment cron_env
  minute "*/10"
  command "cd /srv/www/myapp/current && /usr/local/bin/bundle exec my_command"
end
```

如果您的堆栈具有多个应用程序服务器，则将 `cronjob.rb` 分配到 PHP App Server 层的生命周期事件可能不是理想方法。例如，配方在层的所有实例上运行，因此您将收到多个报告。更好的方法是使用自定义层来确保只有一台服务器发送报告。

仅在层的其中一个实例上运行配方

1. 例如，创建一个名为 PHPAdmin 的自定义层，然后将 `cronjob.rb` 分配到其设置和部署事件。自定义层不一定要执行很多操作。在本例中，PHPAdmin 只需在其实例上运行一个自定义配方。
2. 将其中一个 PHP 应用服务器实例分配给 AdminLayer。如果一个实例属于多个图层， AWS OpsWorks Stacks 会运行每个层的内置和自定义配方。

由于只有一个实例属于 PHP App Server 和 PHPAdmin 层，因此 `cronjob.rb` 仅在该实例上运行并且您只会收到一个报告。

在 Linux 实例上安装和配置程序包

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

内置层仅支持特定程序包。有关更多信息，请参阅 图层。您可以通过实施处理关联的设置、配置和部署任务的自定义配方来安装其他程序包 (如 Redis 服务器)。在某些情况下，最佳方法是扩展内置层以便让该层在其实例上安装该程序包以及该层的标准程序包。例如，如果您有一个支持 PHP 应用程序的堆栈，并且您想要包含 Redis 服务器，则可以扩展 PHP App Server 层以在该层的实例上安装和配置 Redis 服务器以及 PHP 应用程序服务器。

程序包安装配方通常需要执行如下任务：

- 创建一个或多个目录并设置其模式。
- 基于模板创建配置文件。
- 运行安装程序以在实例上安装程序包。
- 启动一个或多个服务。

有关如何安装 Tomcat 服务器的示例，请参阅创建自定义 Tomcat 服务器层。该主题介绍了如何设置自定义 Redis 层，但您可使用大致相同的代码在内置层上安装和配置 Redis。有关如何安装其他程序包的示例，请参阅 https://github.com/aws/opsworks-cookbooks 上的内置说明书。

创建自定义 Tomcat 服务器层

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 本主题介绍如何为 Linux 堆栈实施自定义层。但是，也可以修改基本原则和某些代码来为 Windows 堆栈实施自定义层，尤其是应用程序部署上的部分中的 Windows 堆栈。

在 AWS OpsWorks Stacks 实例上使用非标准包的最简单方法是扩展现有层。但是，这种方法会在该层的实例上同时安装并运行标准和非标准程序包，通常不需要这么做。一种要求更高但更有效的方法是实施自定义层，这使您几乎可以完全控制该层的实例，其中包括以下内容：

- 安装哪些程序包
- 如何配置每个程序包
- 如何将应用程序从存储库部署到实例

无论使用控制台还是 API，您都可以像任何其他层一样创建并管理自定义层，如自定义层中所述。但是，自定义层的内置配方仅执行一些非常基本的任务，如安装 Ganglia 客户端以向 Ganglia 主机报告指标。要使自定义层的实例不仅仅局限于最低的正常运行，您必须实施一个或多个包含 Chef 配方和相关文件的自定义说明书，以处理安装和配置程序包、部署应用程序等任务。不过，您没有必要从头开始实

施所有内容。例如，如果您将应用程序存储在一个标准存储库中，您可以使用内置部署配方来处理在该层的实例上安装应用程序的大部分工作。

> **Note**
>
> 如果您在 Chef 方面是新手，您应当先阅读说明书 101，这是一个教程，旨在介绍关于如何实施说明书以执行各种常见任务的基本知识。

以下演练介绍如何实施可支持 Tomcat 应用程序服务器的自定义层。该层基于一个名为 Tomcat 的自定义说明书，该说明书中包含用于处理程序包安装、部署等任务的配方。本演练包括 Tomcat 说明书的摘录。你可以从其GitHub 存储库中下载完整的食谱。如果您不熟悉 Opscode Chef，您应当先阅读说明书和诀窍。

> **Note**
>
> AWS OpsWorks Stacks 包括一个功能齐全的 Java 应用服务器层，供生产使用。Tomcat 说明书的目的在于说明如何实施自定义层，因此它仅支持不包括 SSL 等功能的有限版本的 Tomcat。有关功能完备的实施的示例，请参阅内置 opsworks_java 说明书。

Tomcat 说明书支持其实例拥有以下特征的自定义层：

- 它们支持拥有 Apache 前端的 Tomcat Java 应用程序服务器。
- Tomcat 经配置，允许应用程序使用 JDBC `DataSource` 对象来连接到单独的 MySQL 实例，该实例相当于一个后端数据存储。

针对这个项目的说明书涉及多个关键组件：

- 属性文件包含各个配方使用的配置设置。
- Setup 配方被分配给该层的 Setup 生命周期事件。它们在实例启动后运行，并执行安装程序包和创建配置文件等任务。
- Configure 配方分配给了该层的 Configure 生命周期事件。它们在堆栈的配置发生更改后 (主要是当实例变为在线或离线状态时) 运行，并处理任何所需的配置更改。
- Deploy 配方分配给了层的 Deploy 生命周期事件。它们在 Setup 配方之后以及当您手动部署应用程序以在层的实例上安装代码和相关文件并处理相关任务 (如重新启动服务) 之后运行。

最后一部分 创建堆栈并运行应用程序 介绍如何创建一个包含基于 Tomcat 说明书的自定义层的堆栈，以及如何部署和运行一个简单的 JSP 应用程序，该应用程序显示在属于单独的 MySQL 层的实例上运行的 MySQL 数据库中的数据。

> **Note**
>
> Tomcat 食谱依赖于一些 AWS OpsWorks Stacks 内置食谱。为了明确每个配方的来源，本主题使用 Chef cookbookname::recipename 约定确定配方。

主题

- 属性文件
- Setup 配方
- Configure 配方
- Deploy 配方
- 创建堆栈并运行应用程序

属性文件

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

在查看配方之前，先检查一下 Tomcat 说明书的属性文件 (该文件包含配方使用的各种配置设置) 是很有帮助的。属性不是必需的；您可以直接在您的配方或模板中对这些值进行硬编码。但是，如果您使用属性定义配置设置，则可以使用 AWS OpsWorks Stacks 控制台或 API 通过定义自定义 JSON 属性来修改值，这比每次要更改设置时都重写配方或模板代码更简单、更灵活。这种方法具有诸多好处，例如，允许您为多个堆栈使用相同的说明书，但对于每个堆栈，要为 Tomcat 服务器配置不同的值。有关属性以及如何覆盖属性的更多信息，请参阅覆盖属性。

以下示例显示了完整的属性文件 `default.rb`，该文件位于 Tomcat 说明书的 `attributes` 目录中。

```
default['tomcat']['base_version'] = 6
default['tomcat']['port'] = 8080
default['tomcat']['secure_port'] = 8443
default['tomcat']['ajp_port'] = 8009
default['tomcat']['shutdown_port'] = 8005
default['tomcat']['uri_encoding'] = 'UTF-8'
default['tomcat']['unpack_wars'] = true
default['tomcat']['auto_deploy'] = true
case node[:platform]
when 'centos', 'redhat', 'fedora', 'amazon'
  default['tomcat']['java_opts'] = ''
when 'debian', 'ubuntu'
  default['tomcat']['java_opts'] = '-Djava.awt.headless=true -Xmx128m -XX:
+UseConcMarkSweepGC'
end
default['tomcat']['catalina_base_dir'] = "/etc/tomcat#{node['tomcat']['base_version']}"
default['tomcat']['webapps_base_dir'] = "/var/lib/tomcat#{node['tomcat']
['base_version']}/webapps"
default['tomcat']['lib_dir'] = "/usr/share/tomcat#{node['tomcat']['base_version']}/lib"
default['tomcat']['java_dir'] = '/usr/share/java'
default['tomcat']['mysql_connector_jar'] = 'mysql-connector-java.jar'
default['tomcat']['apache_tomcat_bind_mod'] = 'proxy_http' # or: 'proxy_ajp'
default['tomcat']['apache_tomcat_bind_config'] = 'tomcat_bind.conf'
default['tomcat']['apache_tomcat_bind_path'] = '/tc/'
default['tomcat']['webapps_dir_entries_to_delete'] = %w(config log public tmp)
case node[:platform]
when 'centos', 'redhat', 'fedora', 'amazon'
  default['tomcat']['user'] = 'tomcat'
  default['tomcat']['group'] = 'tomcat'
  default['tomcat']['system_env_dir'] = '/etc/sysconfig'
when 'debian', 'ubuntu'
  default['tomcat']['user'] = "tomcat#{node['tomcat']['base_version']}"
  default['tomcat']['group'] = "tomcat#{node['tomcat']['base_version']}"
  default['tomcat']['system_env_dir'] = '/etc/default'
end
```

稍后将在相关部分中讨论这些设置本身。以下说明适用于一般情况：

- 所有节点定义都是 `default` 类型，因此，您可以使用自定义 JSON 属性覆盖这些定义。
- 该文件使用 `case` 语句按照条件并基于实例的操作系统设置一些属性值。

  `platform` 节点由 Chef 的 Ohai 工具生成，表示实例的操作系统。

Setup 配方

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

Setup 配方分配给了该层的 Setup 生命周期事件，这些配方在实例启动后运行。它们执行安装程序包、创建配置文件和启动服务等任务。安装配方完成运行后， AWS OpsWorks Stacks 会运行 Deploy 配方，将任何应用程序部署到新实例。

主题

- tomcat::setup
- tomcat::install
- tomcat::service
- tomcat::container_config
- tomcat::apache_tomcat_bind

tomcat::setup

`tomcat::setup` 配方适合分配给层的 Setup 生命周期事件。

```
include_recipe 'tomcat::install'
include_recipe 'tomcat::service'

service 'tomcat' do
  action :enable
end

# for EBS-backed instances we rely on autofs
bash '(re-)start autofs earlier' do
  user 'root'
  code <<-EOC
    service autofs restart
  EOC
  notifies :restart, resources(:service => 'tomcat')
```

```
end

include_recipe 'tomcat::container_config'
include_recipe 'apache2'
include_recipe 'tomcat::apache_tomcat_bind'
```

`tomcat::setup` 配方基本上是一个元配方。它包括一组从属配方，这些配方用于处理安装和配置 Tomcat 及相关程序包的大部分详细信息。`tomcat::setup` 的第一部分运行以下配方，我们将在后面的部分进行介绍：

- tomcat::install 配方可安装 Tomcat 服务器包。
- tomcat::service 配方可设置 Tomcat 服务。

`tomcat::setup` 的中间部分可启用并启动 Tomcat 服务：

- Chef service 资源可在启动时启用 Tomcat 服务。
- Chef bash 资源运行一个 Bash 脚本，以启动 autofs 守护进程，这对于 由Amazon EBS 提供支持的实例来说是必要的。然后，该资源通知 `service` 资源重新启动 Tomcat 服务。

  有关更多信息，请参阅：autofs (对于 Amazon Linux) 或 Autofs (对于 Ubuntu)。

`tomcat::setup` 的最后一部分可创建配置文件并安装和配置前端 Apache 服务器：

- tomcat::container_config 配方可创建配置文件。
- `apache2`配方（简写为`apache2::default`）是 AWS OpsWorks Stacks 的内置配方，用于安装和配置 Apache 服务器。
- tomcat::apache_tomcat_bind 配方可将 Apache 服务器配置为作为 Tomcat 服务器的前端运行。

> **Note**
>
> 您通常可以通过使用内置配方执行某些所需任务来节约时间和精力。此配方使用内置 `apache2::default` 配方来安装 Apache，而不是从头实施 Apache。有关另一个关于如何使用内置配方的示例，请参阅Deploy 配方。

后面的部分更加详细地介绍了 Tomcat 说明书的 Setup 配方。有关 `apache2` 配方的更多信息，请参阅 opsworks-cookbooks/apache2。

tomcat::install

`tomcat::install` 配方可安装 Tomcat 服务器、OpenJDK 以及 Java 连接器库，该连接器库可处理与 MySQL 服务器的连接。

```
tomcat_pkgs = value_for_platform(
  ['debian', 'ubuntu'] => {
    'default' => ["tomcat#{node['tomcat']['base_version']}", 'libtcnative-1',
 'libmysql-java']
  },
  ['centos', 'redhat', 'fedora', 'amazon'] => {
    'default' => ["tomcat#{node['tomcat']['base_version']}", 'tomcat-native', 'mysql-
connector-java']
  },
  'default' => ["tomcat#{node['tomcat']['base_version']}"]
)

tomcat_pkgs.each do |pkg|
  package pkg do
    action :install
  end
end

link ::File.join(node['tomcat']['lib_dir'], node['tomcat']['mysql_connector_jar']) do
  to ::File.join(node['tomcat']['java_dir'], node['tomcat']['mysql_connector_jar'])
  action :create
end

# remove the ROOT webapp, if it got installed by default
include_recipe 'tomcat::remove_root_webapp'
```

该配方执行以下任务：

1. 创建要安装的程序包的列表，具体取决于实例的操作系统。
2. 安装列表中的每个程序包。

   Chef package 资源使用适当的提供程序 (对于 Amazon Linux，使用 `yum`；对于 Ubuntu，使用 `apt-get`) 来处理安装任务。程序包提供程序将 OpenJDK 作为 Tomcat 依赖项进行安装，但必须显式安装 MySQL 连接器库。
3. 使用 Chef link 资源在 Tomcat 服务器的 lib 目录中创建一个指向 JDK 中的 MySQL 连接器库的符号链接。

如果使用默认属性值，则 Tomcat lib 目录为 /usr/share/tomcat6/lib，并且 MySQL 连接器库 (mysql-connector-java.jar) 位于 /usr/share/java/ 中。

tomcat::remove_root_webapp 配方可删除 ROOT Web 应用程序 (默认情况下为 /var/lib/tomcat6/webapps/ROOT)，以避免出现一些安全问题。

```
ruby_block 'remove the ROOT webapp' do
  block do
    ::FileUtils.rm_rf(::File.join(node['tomcat']['webapps_base_dir'], 'ROOT'), :secure
 => true)
  end
  only_if { ::File.exists?(::File.join(node['tomcat']['webapps_base_dir'], 'ROOT'))
 && !::File.symlink?(::File.join(node['tomcat']['webapps_base_dir'], 'ROOT')) }
end
```

only_if 语句可确保配方仅在文件存在时删除文件。

> **Note**
>
> Tomcat 版本由 ['tomcat']['base_version'] 属性指定，该属性在属性文件中被设置为 6。要安装 Tomcat 7，您可以使用自定义 JSON 属性来覆盖该属性。编辑您的堆栈设置，然后在 Custom Chef JSON 框中输入以下 JSON，或者将其添加到任何现有自定义 JSON 即可：
>
> ```
> {
>   'tomcat' : {
>     'base_version' : 7
>   }
> }
> ```
>
> 自定义 JSON 属性覆盖默认属性，并将 Tomcat 版本设置为 7。有关覆盖属性的更多信息，请参阅覆盖属性。

tomcat::service

tomcat::service 配方创建 Tomcat 服务定义。

```
service 'tomcat' do
  service_name "tomcat#{node['tomcat']['base_version']}"

  case node[:platform]
  when 'centos', 'redhat', 'fedora', 'amazon'
    supports :restart => true, :reload => true, :status => true
  when 'debian', 'ubuntu'
    supports :restart => true, :reload => false, :status => true
  end

  action :nothing
end
```

该配方使用 Chef service 资源指定 Tomcat 服务名称 (默认情况下为 tomcat6)，并设置 `supports` 属性以定义 Chef 如何管理服务在不同操作系统上的重新启动、重新加载和状态命令。

- `true` 表示 Chef 可以使用 init 脚本或其他服务提供程序来运行该命令。
- `false` 表示 Chef 必须尝试以其他方式运行该命令。

请注意，`action` 的设置为 `:nothing`。对于每个生命周期事件， AWS OpsWorks Stacks 都会启动 Chef 运行以执行相应的配方集。Tomcat 说明书采用常见模式：使配方创建服务定义，但不重新启动服务。Chef 运行中的其他配方处理重新启动，这通常是通过在用于创建配置文件的 `notifies` 资源中添加 `template` 命令来实现的。通知是重新启动服务的一种简便方法，因为它们仅在配置发生改变时重新启动服务。此外，如果 Chef 运行拥有针对一项服务的多个重新启动通知，Chef 最多重新启动服务一次。这种做法可以避免在尝试重新启动运行不正常的服务 (Tomcat 错误的常见起因) 时可能发生的问题。

必须为使用重新启动通知的任何 Chef 运行定义 Tomcat 服务。因此，多个配方都包含 `tomcat::service`，以确保为每个 Chef 运行都定义了该服务。如果一次 Chef 运行包含 `tomcat::service` 的多个实例，不会造成损失，因为 Chef 确保每一次运行，配方仅执行一次，而无论包括了几次实例。

tomcat::container_config

`tomcat::container_config` 配方基于说明书模板文件创建配置文件。

```
include_recipe 'tomcat::service'

template 'tomcat environment configuration' do
```

```
  path ::File.join(node['tomcat']['system_env_dir'], "tomcat#{node['tomcat']
['base_version']}")
  source 'tomcat_env_config.erb'
  owner 'root'
  group 'root'
  mode 0644
  backup false
  notifies :restart, resources(:service => 'tomcat')
end

template 'tomcat server configuration' do
  path ::File.join(node['tomcat']['catalina_base_dir'], 'server.xml')
  source 'server.xml.erb'
  owner 'root'
  group 'root'
  mode 0644
  backup false
  notifies :restart, resources(:service => 'tomcat')
end
```

该配方首先调用 `tomcat::service`，以在必要时定义该服务。批量配方包含两个 template 资源，每个资源都基于一个说明书模板文件创建配置文件，设置文件属性，以及通知 Chef 重新启动服务。

Tomcat 环境配置文件

第一个 `template` 资源使用 `tomcat_env_config.erb` 模板文件创建 Tomcat 环境配置文件，该配置文件用于设置环境变量，如 `JAVA_HOME`。默认文件名是 `template` 资源的参数。`tomcat::container_config` 使用 `path` 属性覆盖默认值，并将配置文件命名为 `/etc/sysconfig/tomcat6` (Amazon Linux) 或 `/etc/default/tomcat6` (Ubuntu)。`template` 资源还指定文件的所有者、组和模式设置，并指示 Chef 不要创建备份文件。

如果您查看源代码，实际上有三个版本的 `tomcat_env_config.erb`，这些版本分别位于 `templates` 目录的不同子目录中。`ubuntu` 和 `amazon` 目录包含适用于它们各自操作系统的模板。`default` 文件夹包含一个仅具有一个注释行的伪模板，仅当您尝试使用不受支持的操作系统在一个实例上运行此配方时，才会使用这个模板。`tomcat::container_config` 配方不需要指定要使用哪个 `tomcat_env_config.erb`。Chef 根据 File Specificity 中介绍的规则自动为实例的操作系统选择适当的目录。

此示例中的 `tomcat_env_config.erb` 文件主要包含注释。要设置其他环境变量，取消相应行的注释并提供您的首选值即可。

> **Note**
>
> 可能发生更改的任何配置设置都应当定义为一个属性，而不应当在模板中对其进行硬编码。这样，您就无需重写模板来更改设置，只需覆盖该属性即可。

Amazon Linux 模板仅设置一个环境变量，如以下摘录中所示。

```
...
# Use JAVA_OPTS to set java.library.path for libtcnative.so
#JAVA_OPTS="-Djava.library.path=/usr/lib"

JAVA_OPTS="${JAVA_OPTS} <%= node['tomcat']['java_opts'] %>"

# What user should run tomcat
#TOMCAT_USER="tomcat"
...
```

JAVA_OPTS 可用于指定 Java 选项，如库路径。如果使用默认的属性值，则模板不会为 Amazon Linux 设置 Java 选项。您可以通过覆盖 `['tomcat']['java_opts']` 属性 (例如，使用自定义 JSON 属性) 来设置您自己的 Java 选项。有关示例，请参阅创建堆栈。

Ubuntu 模板设置多个环境变量，如以下模板片段所示。

```
# Run Tomcat as this user ID. Not setting this or leaving it blank will use the
# default of tomcat<%= node['tomcat']['base_version'] %>.
TOMCAT<%= node['tomcat']['base_version'] %>_USER=tomcat<%= node['tomcat']
['base_version'] %>
...
# Run Tomcat as this group ID. Not setting this or leaving it blank will use
# the default of tomcat<%= node['tomcat']['base_version'] %>.
TOMCAT<%= node['tomcat']['base_version'] %>_GROUP=tomcat<%= node['tomcat']
['base_version'] %>
...
JAVA_OPTS="<%= node['tomcat']['java_opts'] %>"

<% if node['tomcat']['base_version'].to_i < 7 -%>
# Unset LC_ALL to prevent user environment executing the init script from
# influencing servlet behavior.  See Debian bug #645221
```

```
unset LC_ALL
<% end -%>
```

在使用默认属性值时，模板会将 Ubuntu 环境变量设置如下：

- TOMCAT6_USER 和 TOMCAT6_GROUP (表示 Tomcat 用户和组) 均被设置为 tomcat6。

  如果您将 ['tomcat']['base_version'] 设置为 tomcat7，则变量名称将解析为 TOMCAT7_USER 和 TOMCAT7_GROUP，并且均被设置为 tomcat7。
- JAVA_OPTS 被设置为 -Djava.awt.headless=true -Xmx128m -XX:+UseConcMarkSweepGC：
  - 将 -Djava.awt.headless 设置为 true 可告知图形引擎，实例处于无管模式，没有控制台，这会解决特定图形应用程序的错误行为。
  - -Xmx128m 确保 JVM 拥有足够的内存资源，在此示例中为 128 MB。
  - -XX:+UseConcMarkSweepGC 指定并发标记清除垃圾收集，这有助于限制垃圾收集引发的暂停。

    有关更多信息，请参阅：Concurrent Mark Sweep Collector Enhancements。
- 如果 Tomcat 版本低于 7，则模板取消 LC_ALL 的设置，这可以解决 Ubuntu 错误。

> **Note**
>
> 使用默认属性，其中一些环境变量会直接被设置为其默认值。但是，将环境变量显式设置为属性意味着：您可以定义自定义 JSON 属性来覆盖默认属性并提供自定义值。有关覆盖属性的更多信息，请参阅覆盖属性。

要查看完整的模板文件，请参阅 source code。

Server.xml 配置文件

第二个 template 资源使用 server.xml.erb 创建 system.xml 配置文件，它用于配置 servlet/JSP 容器。server.xml.erb 不包含操作系统特定的设置，因此位于 template 目录的 default 子目录中。

该模板使用标准设置，但它可以为 Tomcat 6 或 Tomcat 7 创建一个 system.xml 文件。例如，模板的服务器部分中的以下代码可为指定版本正确配置侦听器。

```
<% if node['tomcat']['base_version'].to_i > 6 -%>
  <!-- Security listener. Documentation at /docs/config/listeners.html
  <Listener className="org.apache.catalina.security.SecurityListener" />
  -->
<% end -%>
  <!--APR library loader. Documentation at /docs/apr.html -->
  <Listener className="org.apache.catalina.core.AprLifecycleListener" SSLEngine="on" />
  <!--Initialize Jasper prior to webapps are loaded. Documentation at /docs/jasper-
howto.html -->
  <Listener className="org.apache.catalina.core.JasperListener" />
  <!-- Prevent memory leaks due to use of particular java/javax APIs-->
  <Listener className="org.apache.catalina.core.JreMemoryLeakPreventionListener" />
<% if node['tomcat']['base_version'].to_i < 7 -%>
  <!-- JMX Support for the Tomcat server. Documentation at /docs/non-existent.html -->
  <Listener className="org.apache.catalina.mbeans.ServerLifecycleListener" />
<% end -%>
  <Listener className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListener" />
<% if node['tomcat']['base_version'].to_i > 6 -%>
  <Listener className="org.apache.catalina.core.ThreadLocalLeakPreventionListener" />
<% end -%>
```

该模板使用属性来代替硬编码设置，因此，您可以轻松地通过定义自定义 JSON 属性来更改设置。例如：

```
<Connector port="<%= node['tomcat']['port'] %>" protocol="HTTP/1.1"
           connectionTimeout="20000"
           URIEncoding="<%= node['tomcat']['uri_encoding'] %>"
           redirectPort="<%= node['tomcat']['secure_port'] %>" />
```

有关更多信息，请参阅 source code。

tomcat::apache_tomcat_bind

`tomcat::apache_tomcat_bind` 配方使 Apache 服务器能够充当 Tomcat 的前端，接收传入请求、将其转发到 Tomcat 并将响应返回给客户端。此示例使用 mod_proxy 作为 Apache 代理/网关。

```
execute 'enable mod_proxy for apache-tomcat binding' do
  command '/usr/sbin/a2enmod proxy'
```

```
  not_if do
    ::File.symlink?(::File.join(node['apache']['dir'], 'mods-enabled', 'proxy.load'))
 || node['tomcat']['apache_tomcat_bind_mod'] !~ /\Aproxy/
  end
end

execute 'enable module for apache-tomcat binding' do
  command "/usr/sbin/a2enmod #{node['tomcat']['apache_tomcat_bind_mod']}"
  not_if {::File.symlink?(::File.join(node['apache']['dir'], 'mods-enabled',
 "#{node['tomcat']['apache_tomcat_bind_mod']}.load"))}
end

include_recipe 'apache2::service'

template 'tomcat thru apache binding' do
  path ::File.join(node['apache']['dir'], 'conf.d', node['tomcat']
['apache_tomcat_bind_config'])
  source 'apache_tomcat_bind.conf.erb'
  owner 'root'
  group 'root'
  mode 0644
  backup false
  notifies :restart, resources(:service => 'apache2')
end
```

要启用 mod_proxy，您必须启用 proxy 模块和一个基于协议的模块。对于协议模块，您有两个选项：

- HTTP: proxy_http
- Apache JServ Protocol (AJP)：proxy_ajp

  AJP 是一个内部 Tomcat 协议。

配方的这两个 execute 资源 都运行 a2enmod 命令，该命令通过创建所需的符号链接来启用特定模块：

- 第一个 execute 资源启用 proxy 模块。
- 第二个 execute 资源启用协议模块，它在默认情况下被设置为 proxy_http。

  如果您希望使用 AJP，您可以定义自定义 JSON 以覆盖 apache_tomcat_bind_mod 属性并将其设置为 proxy_ajp。

该apache2::service配方是 AWS OpsWorks Stacks 的内置配方，用于定义 Apache 服务。有关更多信息，请参阅 AWS OpsWorks Stacks GitHub 存储库中的配方。

template 资源使用 apache_tomcat_bind.conf.erb 创建一个配置文件，默认情况下，该文件被命名为 tomcat_bind.conf。它将该配置文件放在 ['apache']['dir']/.conf.d 目录中。['apache']['dir'] 属性在内置 apache2 属性文件中定义，默认情况下它被设置为 /etc/httpd (Amazon Linux) 或 /etc/apache2 (Ubuntu)。如果 template 资源创建或更改配置文件，则 notifies 命令将计划重新启动 Apache 服务。

```
<% if node['tomcat']['apache_tomcat_bind_mod'] == 'proxy_ajp' -%>
ProxyPass <%= node['tomcat']['apache_tomcat_bind_path'] %> ajp://localhost:<%=
 node['tomcat']['ajp_port'] %>/
ProxyPassReverse <%= node['tomcat']['apache_tomcat_bind_path'] %> ajp://localhost:<%=
 node['tomcat']['ajp_port'] %>/
<% else %>
ProxyPass <%= node['tomcat']['apache_tomcat_bind_path'] %> http://localhost:<%=
 node['tomcat']['port'] %>/
ProxyPassReverse <%= node['tomcat']['apache_tomcat_bind_path'] %> http://localhost:<%=
 node['tomcat']['port'] %>/
<% end -%>
```

该模板使用ProxyPass和ProxyPassReverse指令来配置用于在 Apache 和 Tomcat 之间传递流量的端口。由于这两个服务器在同一个实例上，所以它们可以使用本地主机 URL，默认情况下，它们均被设置为 http://localhost:8080。

Configure 配方

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

Configure 配方被分配给层的 Configure 生命周期事件，每当实例进入在线状态或退出在线状态时，堆栈的所有实例上都会发生该事件。您可以使用 Configure 配方来调整实例的配置，以根据需要响应更改。当您实施 Configure 配方时，请记住，堆栈配置更改可能涉及与此层无关的实例。该配方必须能够作出相应的响应，但在某些情况下，可能不需要执行任何操作。

tomcat::configure

`tomcat::configure` 配方用于执行层的 Configure 生命周期事件。

```
include_recipe 'tomcat::context'
# Optional: Trigger a Tomcat restart in case of a configure event, if relevant
# settings in custom JSON have changed (e.g. java_opts/JAVA_OPTS):
#include_recipe 'tomcat::container_config'
```

`tomcat::configure` 配方基本上是一个元配方，可运行两个从属配方。

1. `tomcat::context` 配方可创建一个 Web 应用程序上下文配置文件。

   此文件可配置应用程序用来与 MySQL 实例通信的 JDBC 资源，如下一部分中所述。运行此配方以响应 configure 事件使层可以在数据库层发生变化后更新 Web 应用程序上下文配置文件。
2. `tomcat::container_config` Setup 配方再次运行，以捕获容器配置中的任何更改。

在本示例中，`include` 的 `tomcat::container_config` 被改为注释了。如果您希望使用自定义 JSON 来修改 Tomcat 设置，您可以删除注释。之后，Configure 生命周期事件运行 `tomcat::container_config`，这会更新 Tomcat 相关配置文件 (如tomcat::container_config中所述) 并重新启动 Tomcat 服务。

tomcat::context

Tomcat cookbook 允许应用程序使用 J DataSource 2EE 对象访问 MySQL 数据库服务器，该服务器可以在单独的实例上运行。借助 Tomcat，您可以通过为每个应用程序创建和安装 Web 应用程序上下文配置文件来启用连接。此文件定义应用程序与 JDBC 资源 (应用程序将使用该资源与数据库通信) 之间的关系。有关更多信息，请参阅 The Context Container。

`tomcat::context` 配方的主要目的是创建此配置文件。

```
include_recipe 'tomcat::service'

node[:deploy].each do |application, deploy|
  context_name = deploy[:document_root].blank? ? application : deploy[:document_root]

  template "context file for #{application} (context name: #{context_name})" do
    path ::File.join(node['tomcat']['catalina_base_dir'], 'Catalina', 'localhost',
 "#{context_name}.xml")
```

```
    source 'webapp_context.xml.erb'
    owner node['tomcat']['user']
    group node['tomcat']['group']
    mode 0640
    backup false
    only_if { node['datasources'][context_name] }
    variables(:resource_name => node['datasources'][context_name], :webapp_name =>
 application)
    notifies :restart, resources(:service => 'tomcat')
  end
end
```

除了 Tomcat 食谱属性外，此配方还使用 Stack AWS OpsWorks s 在配置事件中安装的堆栈配置和部署属性。 AWS OpsWorks Stacks 服务向每个实例的节点对象添加属性，这些属性包含配方通常通过使用数据袋或搜索获得的信息，并将属性安装到每个实例上。这些属性包含关于堆栈配置、部署的应用程序和用户希望包含的任何自定义数据的详细信息。配方可以通过使用标准 Chef 节点语法获取堆栈配置和部署属性的数据。有关更多信息，请参阅 堆栈配置和部署属性。对于 Chef 11.10 堆栈，您还可以使用 Chef 搜索来获取堆栈配置和部署数据。有关更多信息，请参阅 使用 Chef 搜索。

deploy属性是指[:deploy]命名空间，它包含与部署相关的属性，这些属性通过控制台或 API 定义，或者由 AWS OpsWorks Stacks 服务生成。deploy 属性包含各个已部署应用程序的以应用程序短名称命名的一个属性。每个应用程序属性都包含一组用于描述应用程序的特征的属性，如文档根目录 ([:deploy][:*appname*][:document_root])。

context 配方首先通过调用 tomcat::service 来确保为此 Chef 运行定义了服务。然后，它定义 context_name 变量，该变量表示配置文件的名称，不包括 .xml 扩展名。如果您使用默认文档根目录，则 context_name 将被设置为应用程序的短名称。否则，它将被设置为指定的文档根目录。创建堆栈并运行应用程序中讨论的示例将文档根目录设置为 "ROOT"，因此，上下文是 ROOT，配置文件被命名为 ROOT.xml。

批量配方查看部署的应用程序的列表，并为每个应用程序使用 webapp_context.xml.erb 模板来创建上下文配置文件。此示例仅部署一个应用程序，但 deploy 属性的定义需要您将其看作一组应用程序。

webapp_context.xml.erb 模板不是特定于操作系统的，因此它位于 templates 目录中的 default 子目录中。

该配方创建配置文件，如下所示：

- 使用默认属性值时，配置文件名称设置为 *context_name*.xml，并安装在 /etc/tomcat6/Catalina/localhost/ 目录中。

堆栈配置属性的 ['datasources'] 节点包含一个或多个属性，其中每个属性都会将一个上下文名称映射到关联的应用程序将用来与数据库通信的 JDBC 数据资源。当您创建堆栈时，该节点及其内容通过自定义 JSON 进行定义，如创建堆栈并运行应用程序后面的部分所述。此示例中仅有一个将 ROOT 上下文名称与名为 jdbc/mydb 的 JDBC 资源相关联的属性。

- 通过使用默认属性值，文件的用户和组均被设置为由 Tomcat 程序包定义的值：tomcat (Amazon Linux) 或 tomcat6 (Ubuntu)。
- template 资源仅在存在 ['datasources'] 节点时创建配置文件，并且包含 context_name 属性。
- template 资源会定义两个变量：resource_name 和 webapp_name。

  resource_name 被设置为与 context_name 相关联的资源名称，而 webapp_name 被设置为应用程序的短名称。
- template 资源可重新启动 Tomcat 服务以加载并激活更改。

webapp_context.xml.erb 模板包含一个 Context 元素，该元素包含一个具有其自己的一组属性的 Resource 元素。

这些 Resource 属性描述上下文配置的特征：

- name - JDBC 资源名称，该名称被设置为 tomcat::context 中定义的 resource_name 值。

  例如，资源名称设置为 jdbc/mydb。
- auth 和 type 这些是 JDBC DataSource 连接的标准设置。
- maxActive、maxIdle 和 maxWait -活动和空闲连接的最大数量，以及恢复连接的最长等待时间。
- username 和 password 数据库的用户名和根密码，这些从 deploy 属性中获得。
- driverClassName— JDBC 驱动程序的类名，设置为 MySQL 驱动程序。
- url -连接 URL。

  前缀取决于数据库。它的设置应当如下：jdbc:mysql (MySQL)、jdbc:postgresql (Postgres)，以及 jdbc:sqlserver (SQL Server)。此示例将 URL 设置为 jdbc:mysql://*host_IP_Address*:3306:simplejsp，其中 *simplejsp* 是应用程序的短名称。
- factory-DataSource工厂，这是 MySQL 数据库所必需的。

有关此配置文件的更多信息，请参阅 Tomcat wiki 的 “使用” DataSources 主题。

Deploy 配方

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

Deploy 配方分配给层的 Deploy 生命周期事件。每当你部署应用程序时，它通常会发生在堆栈的所有实例上，但你可以选择将事件限制为仅限指定的实例。 AWS OpsWorks 安装配方完成后，Stacks 还会在新实例上运行 Deploy 配方。Deploy 配方的主要目的是将存储库中的代码和相关文件部署到应用程序服务器层的实例。不过，您通常还可以在其他层上运行 Deploy 配方。这使这些层的实例可以执行更新，例如，更新它们的配置以适应新部署的应用程序。当您实施 Deploy 配方时，请记住，Deploy 事件并不一定意味着会将应用程序部署到实例。它只是一个通知，说明应用程序将部署到堆栈中的其他实例，以允许对实例进行任何必要的更新。该配方必须能够作出相应的响应，这种响应可能不需要执行任何操作。

AWS OpsWorks Stacks 会自动将标准应用程序类型的应用程序部署到相应的内置应用程序服务器层。要将应用程序部署到自定义层，您必须实施自定义 Deploy 配方，该配方可将存储库中的应用程序的文件下载到实例上的适当位置。但是，您通常可以通过使用内置部署说明书处理某些方面的部署来限制您必须编写的代码量。例如，如果您将文件存储到某个受支持的存储库中，则内置说明书可以处理关于从存储库将文件下载到层的实例的详细信息。

`tomcat::deploy` 配方旨在被分配给 Deploy 生命周期事件。

```
include_recipe 'deploy'

node[:deploy].each do |application, deploy|
  opsworks_deploy_dir do
    user deploy[:user]
    group deploy[:group]
    path deploy[:deploy_to]
  end

  opsworks_deploy do
    deploy_data deploy
    app application
  end
```

```
...
```

tomcat::deploy 配方使用内置部署说明书来完成部署的非特定于应用程序的方面。deploy 配方 (内置 deploy::default 配方的简写) 是一个可根据 deploy 属性的数据处理关于设置用户、组等的详细信息的内置配方。

该配方使用两个内置 Chef 定义 (opsworks_deploy_dir 和 opworks_deploy) 来安装应用程序。

opsworks_deploy_dir 定义根据应用程序的部署 JSON 中的数据来设置目录结构。从根本上说，定义是对资源定义进行打包的简便的方式，位于说明书的 definitions 目录中。配方使用定义的方式与使用资源很像，但定义本身并没有关联的提供程序，只是资源包括在定义中。您可以定义配方中的变量，这些变量将传递到基础资源定义。tomcat::deploy 配方根据部署 JSON 中的数据设置 user、group 和 path 变量。它们将传递到定义的 directory 资源，该资源用于管理目录。

> **Note**
>
> 您的已部署的应用程序的用户和组由 [:opsworks][:deploy_user][:user] 和 [:opsworks][:deploy_user][:group] 属性决定，这些属性在内置部署说明书的 deploy.rb 属性文件中进行定义。[:opsworks][:deploy_user][:user] 的默认值为 deploy。[:opsworks][:deploy_user][:group] 的默认值取决于实例的操作系统：
>
> - 对于 Ubuntu 实例，默认组为 www-data。
> - 对于属于使用 Nginx 和 Unicorn 的 Rails App Server 层的成员的 Amazon Linux 实例，默认组为 nginx。
> - 对于所有其他 Amazon Linux 实例，默认组为 apache。
>
> 您可以通过使用自定义 JSON 或自定义属性文件覆盖相应的属性来更改设置。有关更多信息，请参阅 覆盖属性。

另一个定义 opsworks_deploy 可根据 deploy 属性中的数据来处理关于查看存储库中的应用程序的代码和相关文件以及将其部署到实例的详细信息。您可以为任何应用程序类型使用此定义；部署详细信息 (例如目录名称) 在控制台中指定或通过 API 指定，并被置入到 deploy 属性中。但是，opsworks_deploy 仅适用于四种受支持的存储库类型：Git、Subversion、S3 和 HTTP。如果您希望使用不同的存储库类型，您必须自己实施此代码。

您将应用程序的文件安装在 Tomcat webapps 目录中。通常的做法是直接将文件复制到 webapps。但是， AWS OpsWorks Stacks 部署旨在在一个实例上保留多达五个版本的应用程序，因此如有必要，您可以回滚到较早的版本。 AWS OpsWorks 因此，堆栈会执行以下操作：

1. 将应用程序部署到一个不同的目录，该目录的名称中包含一个时间戳，如 /srv/www/my_1st_jsp/releases/20130731141527。
2. 创建一个指向此唯一的目录、名为 current 的符号链接，如 /srv/www/my_1st_jsp/current。
3. 如果尚不存在此目录，请在 webapps 目录中创建一个指向在第 2 步中创建的 current 符号链接的符号链接。

如果您需要回滚到早期版本，可修改 current 符号链接以指向包含相应时间戳的不同目录，例如，通过更改 /srv/www/my_1st_jsp/current 的链接目标。

tomcat::deploy 的中间部分可设置符号链接。

```
 ...
 current_dir = ::File.join(deploy[:deploy_to], 'current')
 webapp_dir = ::File.join(node['tomcat']['webapps_base_dir'],
deploy[:document_root].blank? ? application : deploy[:document_root])

 # opsworks_deploy creates some stub dirs, which are not needed for typical webapps
 ruby_block "remove unnecessary directory entries in #{current_dir}" do
   block do
     node['tomcat']['webapps_dir_entries_to_delete'].each do |dir_entry|
       ::FileUtils.rm_rf(::File.join(current_dir, dir_entry), :secure => true)
     end
   end
 end

 link webapp_dir do
   to current_dir
   action :create
 end
 ...
```

该配方首先创建两个变量 (current_dir 和 webapp_dir) 以分别表示 current 和 webapp 目录。然后，它使用 link 资源将 webapp_dir 链接到 current_dir。 AWS OpsWorks Stacks deploy::default 配方创建了一些本示例不需要的存根目录，因此摘录的中间部分将其删除。

tomcat::deploy 的最后部分可在必要时重新启动 Tomcat 服务。

```
  ...
  include_recipe 'tomcat::service'

  execute 'trigger tomcat service restart' do
    command '/bin/true'
    not_if { node['tomcat']['auto_deploy'].to_s == 'true' }
    notifies :restart, resources(:service => 'tomcat')
  end
end

include_recipe 'tomcat::context'
```

该配方首先运行 tomcat::service 以确保为此 Chef 运行定义了服务。然后，它使用 execute 资源通知服务重新启动，但前提是 ['tomcat']['auto_deploy'] 被设置为 'true'。否则，Tomcat 将侦听其 webapps 目录中的变化，这会使 Tomcat 服务执行一次不必要的显式重新启动。

> **Note**
>
> execute 资源实际上并不执行任何实质性操作；/bin/true 是一个虚拟 Shell 脚本，它只返回成功代码。它在这里只是用于生成重新启动通知的一种简便方式。如之前所提到过的，使用通知可确保不会过于频繁地重新启动服务。

最后，tomcat::deploy 运行 tomcat::context，如果您已经更改了后端数据库，这会更新 Web 应用程序上下文配置文件。

创建堆栈并运行应用程序

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

这部分介绍如何使用 Tomcat 说明书实施可运行名为 SimpleJSP 的简单的 Java 服务器页面 (JSP) 应用程序的基本堆栈设置。该堆栈由一个名为的基于 Tomcat 的自定义层和 TomCustom 一个 MySQL 层

组成。SimpleJSP 已部署到 MySQL 数据库 TomCustom 并显示一些来自 MySQL 数据库的信息。如果您还不熟悉如何使用 AWS OpsWorks Stacks 的基础知识，则应先阅读Chef 11 Linux 堆栈入门。

SimpleJSP 应用程序

SimpleJSP 应用程序演示了关于如何设置数据库连接以及从堆栈的 MySQL 数据库中检索数据的基础知识。

```
<html>
  <head>
    <title>DB Access</title>
  </head>
  <body>
    <%@ page language="java" import="java.sql.*,javax.naming.*,javax.sql.*" %>
    <%
      StringBuffer output = new StringBuffer();
      DataSource ds = null;
      Connection con = null;
      Statement stmt = null;
      ResultSet rs = null;
      try {
        Context initCtx = new InitialContext();
        ds = (DataSource) initCtx.lookup("java:comp/env/jdbc/mydb");
        con = ds.getConnection();
        output.append("Databases found:<br>");
        stmt = con.createStatement();
        rs = stmt.executeQuery("show databases");
        while (rs.next()) {
          output.append(rs.getString(1));
          output.append("<br>");
        }
      }
      catch (Exception e) {
        output.append("Exception: ");
        output.append(e.getMessage());
        output.append("<br>");
      }
      finally {
        try {
          if (rs != null) {
            rs.close();
          }
```

```
          if (stmt != null) {
            stmt.close();
          }
          if (con != null) {
            con.close();
          }
        }
        catch (Exception e) {
          output.append("Exception (during close of connection): ");
          output.append(e.getMessage());
          output.append("<br>");
        }
      }
    %>
    <%= output.toString() %>
  </body>
</html>
```

SimpleJSP 使用 DataSource 对象与 MySQL 数据库进行通信。Tomcat 使用 Web 应用程序上下文配置文件中的数据来创建并初始化 DataSource 对象，然后将其绑定到一个逻辑名称。然后，它向 Java 命名和目录接口 (JNDI) 命名服务注册该逻辑名称。要获取相应的 DataSource 对象的实例，您可以创建一个 InitialContext 对象，将资源的逻辑名称传递给该对象的 lookup 方法，该方法将检索相应的对象。SimpleJSP 示例的逻辑名称 java:comp/env/jdbc/mydb 拥有以下组件：

- 根命名空间 java 通过一个冒号 (:) 与名称的其余部分分隔开来。
- 任何其他命名空间都通过正斜杠 (/) 分隔开来。

  Tomcat 会自动将资源添加到 comp/env 命名空间。
- 资源名称，它在 Web 应用程序上下文配置文件中定义，并通过正斜杠与命名空间分隔开来。

  此示例中的资源名称为 jdbc/mydb。

为了建立与数据库之间的连接，SimpleJSP 将执行以下操作：

1. 调用 DataSource 对象的 getConnection 方法，该方法将返回一个 Connection 对象。
2. 调用 Connection 对象的 createStatement 方法以创建一个 Statement 对象，您可以使用该对象来与数据库通信。
3. 通过调用相应的 Statement 方法与数据库进行通信。

SimpleJSP 调用 `executeQuery` 以执行 SHOW DATABASES 查询，这会列出服务器的数据库。

`executeQuery` 方法将返回一个包含查询结果的 `ResultSet` 对象。SimpleJSP 从返回的 `ResultSet` 对象中获取数据库名称，并将这些名称联接起来创建一个输出字符串。最后，该示例关闭 `ResultSet`、`Statement` 和 `Connection` 对象。有关 JSP 和 JDBC 的更多信息，请分别参见 P JavaServer ages Tech nology 和 JD BC 基础知识。

要将 SimpleJSP 与堆栈一起使用，您必须将其放到存储库中。您可以使用任何受支持的存储库，但要将 SimpleJSP 与以下部分中讨论的示例堆栈一起使用，您必须将其放到公有 S3 存档中。有关如何使用其他标准存储库的信息，请参阅说明书存储库。

将 SimpleJSP 放到 S3 存档存储库中

1. 将示例代码复制到名为 `simplejsp.jsp` 的文件中，并将该文件放到名为 `simplejsp` 的目录中。
2. 创建 `simplejsp` 目录的 `.zip` 存档。
3. 创建一个 Amazon S3 存储桶，将 `simplejsp.zip` 上传到该存储桶，并将该文件设置为公有。

   有关如何执行此任务的说明，请参阅 Amazon Simple Storage Service 入门。

创建堆栈

要运行 SimpleJSP，您需要一个包含以下层的堆栈。

- 一个支持后端 MySQL 服务器的 MySQL 层。
- 一个使用 Tomcat 说明书以支持 Tomcat 服务器实例的自定义层。

要创建 堆栈，请执行以下操作：

1. 在 “ AWS OpsWorks 堆栈” 仪表板上，单击 “添加堆栈” 以创建新堆栈，然后单击 “高级 >>” 以显示所有选项。按如下所示配置堆栈。

   - 名称-用户定义的堆栈名称；本示例使用 TomStack。
   - 使用自定义 Chef 说明书 -将切换设置为 是 ，这会显示一些其他选项。
   - Repository type - Git

- 存储库 URL-`git://github.com/amazonwebservices/opsworks-example-cookbooks.git`。
- 自定义 Chef JSON -添加以下 JSON：

```
{
  "tomcat": {
    "base_version": 7,
    "java_opts": "-Djava.awt.headless=true -Xmx256m"
  },
  "datasources": {
    "ROOT": "jdbc/mydb"
  }
}
```

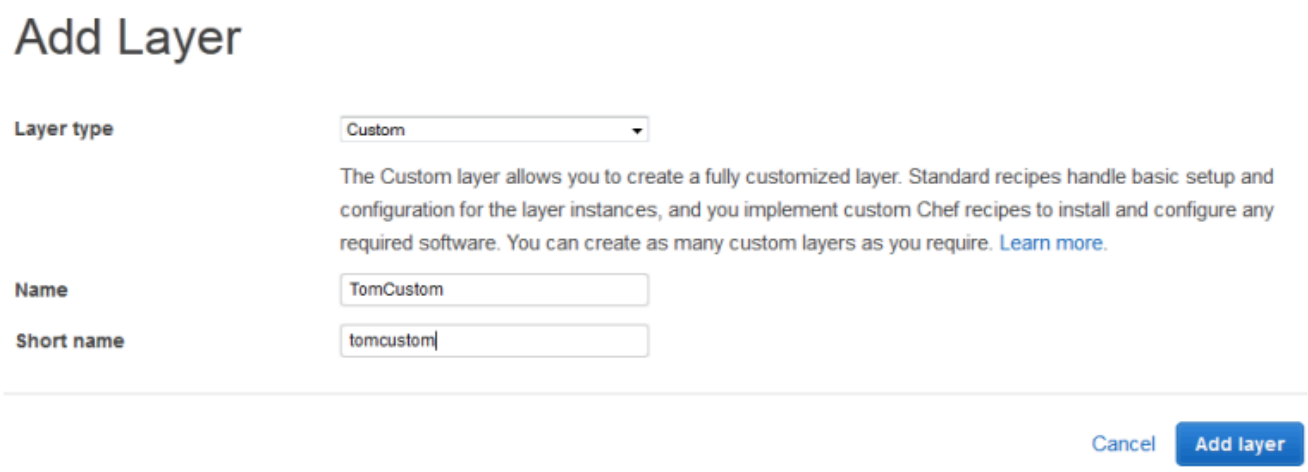
对于其余选项，您可以接受默认值。

自定义 JSON 执行以下操作：

- 覆盖 Tomcat 说明书的 `['base_version']` 属性，以将 Tomcat 版本设置为 7；默认值为 6。
- 覆盖 Tomcat 说明书的 `['java_opts']` 属性，以指定该实例处于无管控状态，并将 JVM 最大堆大小指定为 256MB；默认值并不会为运行 Amazon Linux 的实例设置选项。
- 指定 `['datasources]` 属性值，这会将 JDBC 资源名称 (jdbc/mydb) 分配给 Web 应用程序上下文名称 (ROOT)，如tomcat::context中所述。

  最后一个属性没有默认值；您必须使用自定义 JSON 设置该属性。

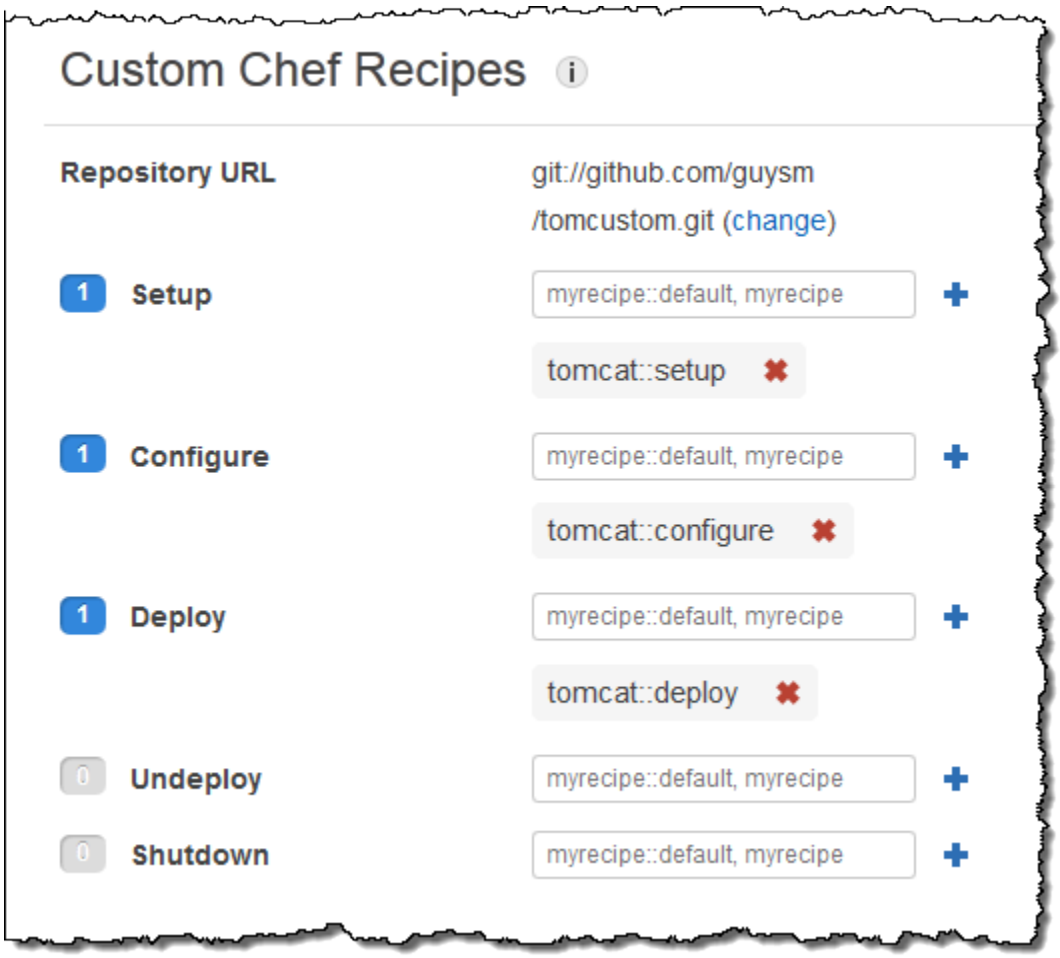
2. 单击 Add a layer。对于 Layer type，选择 MySQL。然后单击 Add Layer。
3. 在导航窗格中，单击 Instances，然后单击 Add an instance。单击 Add Instance 以接受默认值。在该实例对应的行中，单击 start。
4. 返回 Layers 页面并单击 + Layer 以添加层。对于 Layer type (层类型)，单击 Custom (自定义)。该示例分别使用 **`TomCustom`** 和 **`tomcustom`** 作为层的名称和短名称。



5. 在 Layers 页面上，对于自定义层，单击 Recipes，然后单击 Edit。在 Custom Chef Recipes 下，将 Tomcat 说明书配方分配给层的生命周期事件，如下所示：

   - 对于 Setup (设置)，键入 **`tomcat::setup`** 并单击 +。
   - 对于 Configure (配置)，键入 **`tomcat::configure`** 并单击 +。
   - 对于 Deploy (部署)，键入 **`tomcat::deploy`** 并单击 +。然后单击 Save (保存)。

.



6. 在导航窗格中，单击 Apps，然后单击 Add an app。指定以下选项，然后单击 Add App：

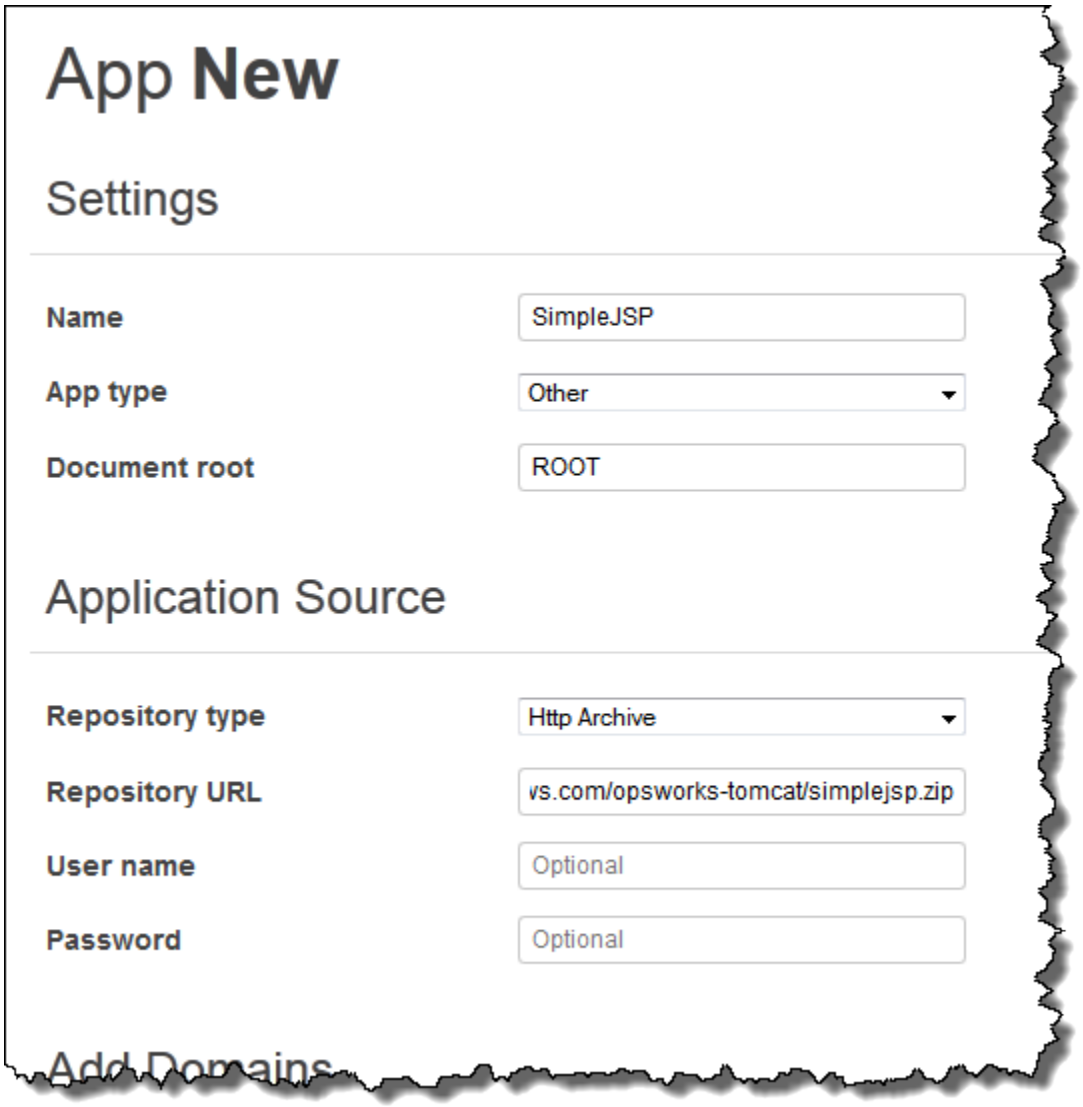
   - 名称 —应用程序的名称；该示例使用 simpleJSP， AWS OpsWorks Stacks 生成的短名称将是 simplejsp。
   - 应用程序类型 -将此选项设置为 其他。

     AWS OpsWorks Stacks 会自动将标准应用程序类型部署到关联的服务器实例。如果您将 App type (应用程序类型) 设置为“other (其他)”， AWS OpsWorks Stacks 将仅运行 Deploy 配方，并使这些配方处理部署作业。
   - Document root-将此选项设置为 **ROOT**。

     Document root 值指定上下文名称。
   - 存储库类型 -将此选项设置为 S3 存档。

- 存储库 URL -将此选项设置为您之前创建的应用程序的 URL。

为其他选项使用默认设置。



7. 使用 “实例” 页面向 TomCustom 图层添加实例并启动该实例。 AWS OpsWorks 安装配方完成后，堆栈会自动在新实例上运行 Deploy 配方，因此启动该实例也会部署 SimpleJSP。
8. 当 TomCustom 实例处于联机状态时，在 “实例” 页面上单击实例名称以查看其详细信息。复制公有 IP 地址。然后构造一个 URL，如下所示：http://*publicIP*/tc/*appname.jsp*。例如，此 URL 应类似于 **`http://50.218.191.172/tc/simplejsp.jsp`**。

> **Note**
>
> 将请求转发到 Tomcat 的 Apache URL 被设置为默认 `['tomcat']['apache_tomcat_bind_path']` 属性 `/tc/`。SimpleJSP 文档根目录被设置为 `ROOT`，这是一个解析为 `/` 的特殊值。因此，该 URL 为“.../tc/simplejsp.jsp”。

9. 将上一步骤中的 URL 粘贴到您的浏览器中。您将看到以下内容：

```
Databases found:
information_schema
simplejsp
test
```

> **Note**
>
> 如果您的堆栈有 MySQL 实例， AWS OpsWorks Stacks 会自动为每个应用程序创建一个数据库，并以应用程序的短名称命名。

堆栈配置和部署属性

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS：Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

当 AWS OpsWorks Stacks 在实例上运行命令（例如，响应 Deploy 生命周期事件的 deploy 命令）时，它会向实例的节点对象添加一组描述堆栈当前配置的属性。对于部署事件和执行配方堆栈命令，AWS OpsWorks Stacks 会安装部署属性，这些属性提供了一些额外的部署信息。有关节点对象的更多信息，请参阅覆盖属性。要查看常用堆栈配置和部署属性的列表 (包括完全限定的节点名称)，请参阅堆栈配置和部署属性：Linux和内置说明书属性。

> **Note**
>
> 在 Linux 堆栈上，您可以使用代理 CLI 的 get_json 命令，获得这些属性 (格式化为 JSON 对象) 的完整列表。

以下各部分介绍了与简单堆栈的配置事件和部署事件相关的属性，简单堆栈包括：

- 包含两个实例的 PHP App Server 层
- 带有一个实例的 HAProxy 层

这些示例都来自一个 实例 php-app1。为方便起见，这些属性被格式化为 JSON 对象。对象的结构映射到属性的完全限定名称。例如，`node[:opsworks][:ruby_version]` 属性的 JSON 表示形式如下所示。

```
{
  "opsworks": {
    ...
    "ruby_version": "1.8.7",
    ...
  }
}
```

主题

- 配置属性
- 部署属性

配置属性

以下 JSON 对象显示当实例联机或脱机时堆栈中各个实例上发生的配置事件的属性。这些属性包括嵌入式堆栈配置属性和事件发生之前为堆栈定义的任何自定义 JSON 属性 (此示例中未提及)。其长度已经过编辑。要了解各个属性的详细描述，请参阅堆栈配置和部署属性：Linux和内置说明书属性。

```
{
  "opsworks": {
    "layers": {
```

```
      "php-app": {
        "id": "4a2a56c8-f909-4b39-81f8-556536d20648",
        "instances": {
          "php-app2": {
            "elastic_ip": null,
            "region": "us-west-2",
            "booted_at": "2013-02-26T20:41:10+00:00",
            "ip": "192.0.2.0",
            "aws_instance_id": "i-34037f06",
            "availability_zone": "us-west-2a",
            "instance_type": "c1.medium",
            "private_dns_name": "ip-10-252-0-203.us-west-2.compute.internal",
            "private_ip": "10.252.0.203",
            "created_at": "2013-02-26T20:39:39+00:00",
            "status": "online",
            "backends": 8,
            "public_dns_name": "ec2-192-0-2-0.us-west-2.compute.amazonaws.com"
          },
          "php-app1": {
            ...
          }
        },
        "name": "PHP Application Server"
      },
      "lb": {
        "id": "15c86142-d836-4191-860f-f4d310440f14",
        "instances": {
          "lb1": {
           ...
          }
        },
        "name": "Load Balancer"
      }
    },
    "agent_version": "104",
    "applications": [

    ],
    "stack": {
      "name": "MyStack"
    },
    "ruby_version": "1.8.7",
    "sent_at": 1361911623,
    "ruby_stack": "ruby_enterprise",
```

```
    "instance": {
      "layers": [
        "php-app"
      ],
      "region": "us-west-2",
      "ip": "192.0.2.0",
      "id": "45ef378d-b87c-42be-a1b9-b67c48edafd4",
      "aws_instance_id": "i-32037f00",
      "availability_zone": "us-west-2a",
      "private_dns_name": "ip-10-252-84-253.us-west-2.compute.internal",
      "instance_type": "c1.medium",
      "hostname": "php-app1",
      "private_ip": "10.252.84.253",
      "backends": 8,
      "architecture": "i386",
      "public_dns_name": "ec2-192-0-2-0.us-west-2.compute.amazonaws.com"
    },
    "activity": "configure",
    "rails_stack": {
      "name": null
    },
    "deployment": null,
    "valid_client_activities": [
      "reboot",
      "stop",
      "setup",
      "configure",
      "update_dependencies",
      "install_dependencies",
      "update_custom_cookbooks",
      "execute_recipes"
    ]
  },
  "opsworks_custom_cookbooks": {
    "recipes": [

    ],
    "enabled": false
  },
  "recipes": [
    "opsworks_custom_cookbooks::load",
    "opsworks_ganglia::configure-client",
    "ssh_users",
    "agent_version",
```

```
    "mod_php5_apache2::php",
    "php::configure",
    "opsworks_stack_state_sync",
    "opsworks_custom_cookbooks::execute",
    "test_suite",
    "opsworks_cleanup"
  ],
  "opsworks_rubygems": {
    "version": "1.8.24"
  },
  "ssh_users": {
  },
  "opsworks_bundler": {
    "manage_package": null,
    "version": "1.0.10"
  },
  "deploy": {
  }
}
```

大部分信息都位于常常被称为命名空间的 opsworks 属性下。以下列表描述了主要属性：

- layers 属性-一组属性，其中各个属性分别描述一个堆栈层的配置。

  在此示例中，这些层是由其短名 php-app 和 lb 确定的。有关其他层的短名的更多信息，请参阅AWS OpsWorks 堆栈图层参考。

- instances 属性-各层都有一个 instances 元素，该元素包含一个针对各层的在线实例、以实例短名称命名的属性。

  PHP App Server 层有两个实例 php-app1 和 php-app2。HAProxy 层有一个实例 lb1。

  > **Note**
  >
  > instances 元素仅包含创建特定堆栈和部署属性时那些处于联机状态的实例。

- 实例属性-每个实例属性都包含一组可表明实例特征的属性，如实例的私有 IP 地址和私有 DNS 名称。为了简洁起见，该示例仅详细显示 php-app2 属性；其他示例包含类似信息。
- applications-一份已部署应用程序的列表，此示例中未用到。
- stack-堆栈名称，在此示例中为 MyStack。

- instance-安装有这些属性的实例，在此示例中为 php-app1。配方可以使用此属性来获得关于配方正在其上运行的实例的信息，如实例的公有 IP 地址。
- activity-生成属性的活动，在此示例中为一个配置事件。
- rails_stack-包含 Rails App Server 层的堆栈的 Rails 堆栈。
- deployment-指示这些属性是否与部署有关联。它在此示例中设置为 null，因为它们与配置事件相关联。
- valid_client_activities-一份有效客户端活动的列表。

opsworks 属性后跟多个其他顶级属性，包括以下各属性：

- opsworks_custom_cookbooks-指示是否启用自定义说明书。如果启用了自定义说明书，则该属性包含一份自定义配方的列表。
- recipes-通过此活动运行的配方。
- opsworks_rubygems— 实例的 RubyGems版本。
- ssh_users-一份 SSH 用户的列表，在此示例中未提及。
- opsworks_bundler-捆绑程序版本，并指示其是否已被启用。
- deploy-关于部署活动的信息，在此示例中未提及。

部署属性

部署事件或 Execute Recipes 堆栈命令的属性包括嵌入式堆栈配置和部署属性以及任何自定义堆栈或部署属性 (在此示例中没有出现)。以下 JSON 对象显示与可将 SimplePHP 应用程序部署到堆栈 PHP 实例的部署事件相关联的 php-app1 的属性。大部分对象都包含与上一部分中描述的配置事件的属性相类似的堆栈配置属性，因此，该示例主要关注特定于部署的属性。要了解各个属性的详细描述，请参阅堆栈配置和部署属性：Linux和内置说明书属性。

```
{
   ...
  "opsworks": {
    ...
    "activity": "deploy",
    "applications": [
      {
        "slug_name": "simplephp",
        "name": "SimplePHP",
        "application_type": "php"
```

```
      }
    ],
    "deployment": "5e6242d7-8111-40ee-bddb-00de064ab18f",
    ...
  },
  ...
{
  "ssh_users": {
  },
  "deploy": {
    "simplephpapp": {
      "application": "simplephpapp",
      "application_type": "php",
      "environment_variables": {
        "USER_ID": "168424",
        "USER_KEY": "somepassword"
      },
      "auto_bundle_on_deploy": true,
      "deploy_to": "/srv/www/simplephpapp",
      "deploying_user": "arn:aws:iam::123456789012:user/guysm",
      "document_root": null,
      "domains": [
        "simplephpapp"
      ],
      "migrate": false,
      "mounted_at": null,
      "rails_env": null,
      "restart_command": "echo 'restarting app'",
      "sleep_before_restart": 0,
      "ssl_support": false,
      "ssl_certificate": null,
      "ssl_certificate_key": null,
      "ssl_certificate_ca": null,
      "scm": {
        "scm_type": "git",
        "repository": "git://github.com/amazonwebservices/opsworks-demo-php-simple-
app.git",
        "revision": "version1",
        "ssh_key": null,
        "user": null,
        "password": null
      },
      "symlink_before_migrate": {
        "config/opsworks.php": "opsworks.php"
```

```
      },
      "symlinks": {
      },
      "database": {
      },
      "memcached": {
        "host": null,
        "port": 11211
      },
      "stack": {
        "needs_reload": false
      }
    }
  },
}
```

opsworks 属性与上一部分中的示例大致相似。以下各部分大多与部署有关：

- activity-与这些属性相关联的事件，在此示例中为一个部署事件。
- applications-包含各个应用程序的一组属性，这些属性提供应用程序的名称、临时名称和类型。

  slug 名称是应用程序的简称， AWS OpsWorks Stacks 根据应用程序名称生成该名称。SimplePHP 的临时名称是 simplephp。
- deployment-用于唯一标识部署的部署 ID。

deploy 属性包含正在部署的应用程序的信息。例如，嵌入式部署配方使用 deploy 属性中的数据，以在相应的目录中安装文件，并创建数据库连接文件。deploy 属性包含各个已部署应用程序的、以应用程序短名命名的一个属性。各个应用程序属性都包含以下属性：

- environment_variables-包含您为应用程序定义的任何环境变量。有关更多信息，请参阅 环境变量。
- domains-默认情况下，域是应用程序的短名称，在此示例中为 simplephpapp。如果您分配了自定义域名，则它们也会出现在此处。有关更多信息，请参阅 使用自定义域。
- application-应用程序的短名称。
- scm-此元素包含要从其存储库下载应用程序的文件所需的信息，在此示例中为 Git 存储库。
- database-数据库信息，前提是堆栈包含一个数据库层。
- document_root-文档根，在此示例中被设置为 null，表明根是公有的。

- `ssl_certificate_ca`、`ssl_support`、`ssl_certificate_key`-指示应用程序是否获得 SSL 支持。如果获得 SSL 支持，则 `ssl_certificate_key` 和 `ssl_certificate_ca` 属性会被设置为相应的证书。
- `deploy_to`-应用程序的根目录。

说明书 101

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

生产级 AWS OpsWorks Stacks 堆栈通常需要一些自定义，这通常意味着使用一个或多个食谱、属性文件或模板文件来实现自定义 Chef 食谱。本主题是实现 AWS OpsWorks 堆栈食谱的教程介绍。

有关 AWS OpsWorks Stacks 如何使用食谱的更多信息（包括对食谱的简要介绍），请参阅。说明书和诀窍有关如何实施和测试 Chef 配方的更多信息，请参阅使用 Chef 的测试驱动型基础设施，第 2 版。

该教程示例分为两个部分：

- 说明书基础知识 是一组演练示例，面向不熟悉 Chef 的用户；有经验的 Chef 用户可以跳过此部分。

  该示例向您演示如何实施说明书来执行常见任务的基础知识，例如安装软件包或者创建目录。为了简化该过程，您将使用两个非常有用的工具 Vagrant 和 Test Kitchen，以在虚拟机中本地运行大部分示例。在开始说明书基础知识之前，您应先阅读Vagrant 和 Test Kitchen 以了解如何安装和使用这些工具。由于 Test Kitchen 尚不支持 Windows，这些示例均面向 Linux，并带有备注来说明如何调整以用于 Windows。
- 为 AWS OpsWorks 堆栈实现食谱介绍如何实现 AWS OpsWorks 堆栈配方，包括 Windows 堆栈。

  它还包含一些更高级的主题，例如如何使用 Berkshelf 来管理外部说明书。这些示例面向新 Chef 用户编写，与 说明书基础知识 中的示例非常类似。但是 AWS OpsWorks Stacks 的工作方式与 Chef 服务器略有不同，因此我们建议有经验的 Chef 用户至少通读本节。

Vagrant 和 Test Kitchen

如果您在使用针对 Linux 实例的说明书，在学习和启动开发与测试时，Vagrant 和 Test Kitchen 是非常有用的工具。本主题提供了 Vagrant 和 Test Kitchen 的简要说明，并指向安装说明和演练，帮助您设置和熟悉如何使用这些工具的基本知识。由于 Vagrant 支持 Windows 而 Test Kitchen 不支持，因此只提供了这些工具的 Linux 示例。

Vagrant

Vagrant 为在虚拟机上执行和测试代码提供了一致的环境。它支持广泛的环境 (称为 Vagrant 盒子)，每个盒子表示一个配置的操作系统。对于 AWS OpsWorks Stacks，感兴趣的环境基于 Ubuntu、Amazon 或红帽企业 Linux (RHEL) 发行版，因此示例主要使用名为 Vagrant 的盒子。`opscode-ubuntu-12.04`

Vagrant 可用于 Linux、Windows 和 Macintosh 系统，因此，您可以使用偏好的工作站在任何受支持的操作系统上实施和测试配方。本章的示例在 Ubuntu Linux 系统上创建，不过可以直接转换为适用于 Windows 或 Macintosh 系统的过程。

Vagrant 本质上是面向虚拟提供程序的包装器。大多数示例都使用VirtualBox提供程序。 VirtualBox 是免费的，适用于 Linux、Windows 和 Macintosh 系统。如果您的系统上还没有安装说明，Vagrant 演练会提供安装说明。 VirtualBox 请注意，您可以在上运行基于 Ubuntu 的环境， VirtualBox但是 Amazon Linux 仅适用于亚马逊 EC2 实例。但是，你可以在上面运行类似的操作系统，比如 CentOS VirtualBox，这对于初始开发和测试很有用。

有关其他提供程序的信息，请参阅 Vagrant 文档。特别注意，`vagrant-aws` 插件提供程序允许您将 Vagrant 与 Amazon EC2 实例结合使用。此提供程序对于在 Amazon Linux 上测试配方尤为有用，不过这仅在 Amazon EC2 实例上可用。`vagrant-aws` 提供程序免费，不过您必须拥有 Amazon Web Services account 并且必须为使用的任何 AWS 资源付费。

此时，您应该通读 Vagrant 的入门演练，其中介绍了如何在工作站上安装 Vagrant 以及如何使用 Vagrant 的基础知识。请注意，本章中的示例不使用 Git 存储库，如果您希望，可以忽略掉演练中的这一部分。

Test Kitchen

Test Kitchen 简化了在 Vagrant 上执行和测试您的说明书的过程。在实际应用中，您几乎不需要直接使用 Vagrant。Test Kitchen 执行了大部分最常见任务，包括：

- 在 Vagrant 中启动实例。

- 将说明书传输到实例。
- 在实例上运行说明书的配方。
- 在实例上测试说明书的配方。
- 使用 SSH 登录到实例。

我们建议不要直接安装 Test Kitchen Gem，而是安装 Chef DK。除了 Chef 本身，这个软件包还包括 Test Kitchen、Berkshelf 和其他几个有用的工具。ChefSpec

此时，您应该通读 Test Kitchen 的入门演练，其中介绍了如何使用 Test Kitchen 来执行和测试配方的基础知识。

> **Note**
>
> 本章中的示例将 Test Kitchen 用作运行配方的便利方式。如果您愿意，您可在完成“手动验证”部分之后停止入门演练，这一部分介绍了您所需了解的全部示例信息。不过，Test Kitchen 基本上是一个支持 Bash 自动测试系统 (BATS) 等测试框架的测试平台。您应抽空完成剩余的演练以了解如何使用 Test Kitchen 来测试配方。

说明书基础知识

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

您可以使用说明书来完成各种任务。以下主题假定您是初次使用 Chef，并介绍了如何使用说明书来完成一些常见任务。由于 Test Kitchen 尚不支持 Windows，这些示例均面向 Linux，并带有备注来说明如何调整以用于 Windows。如果您是初次使用 Chef，即使您要使用的是 Windows，仍建议您浏览这些示例。本主题中的大部分示例均可在进行一些适度更改后用于 Windows 实例，示例中已指出这些更改。所有示例均在虚拟机上运行，因此您甚至不需要 Linux 计算机。只需将 Vagrant 和 Test Kitchen 安装在您的常规工作站上即可。

> **Note**
>
> 如果要在 Windows 实例上运行这些配方，最简单的方法是创建一个 Windows 堆栈，并在堆栈的其中一个实例上运行这些配方。有关如何在 AWS OpsWorks Stacks Windows 实例上运行配方的更多信息，请参阅在 Windows 实例上运行配方。

在继续操作之前，请确保您已安装 Vagrant 和 Test Kitchen，并已完成其入门演练。有关更多信息，请参阅 Vagrant 和 Test Kitchen。

主题

配方结构

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS：Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

说明书本质上是一组配方，可用来在实例上执行各种任务。要阐明如何实施配方，查看一个简单的示例会非常有用。以下是内置 HAProxy 层的设置配方。现在只需关注总体结构，不必过于关注细节；细节将在后续示例中加以介绍。

```
package 'haproxy' do
  action :install
end

if platform?('debian','ubuntu')
  template '/etc/default/haproxy' do
    source 'haproxy-default.erb'
    owner 'root'
    group 'root'
    mode 0644
  end
end

include_recipe 'haproxy::service'

service 'haproxy' do
  action [:enable, :start]
end

template '/etc/haproxy/haproxy.cfg' do
  source 'haproxy.cfg.erb'
  owner 'root'
  group 'root'
  mode 0644
  notifies :restart, "service[haproxy]"
end
```

> **Note**
>
> 有关此示例以及工作配方和相关文件的其他示例，请参阅 AWS OpsWorks Stacks 内置配方。

该示例重点介绍了关键配方元素，以下各节中分别介绍了这些元素。

主题

- 资源
- 流控制
- 包含的配方

资源

配方主要包含一组 Chef 资源。其中每种资源指定实例最终状态的一个特定方面，例如要安装的软件包或要启动的服务。该示例有四种资源：

- package 资源，该资源代表已安装的软件包，对于本示例为 HAProxy 服务器。
- service 资源，该资源代表服务，对于本示例为 HAProxy 服务。
- 两种 template 资源，这两种资源代表利用指定模板创建的文件，对于本示例为两个 HAProxy 配置文件。

资源提供了一种用于指定实例状态的声明方式。在幕后，每种资源都有一个关联的提供程序执行所需的任务，如安装软件包、创建和配置目录、启动服务等等。如果任务的详细信息取决于特定的操作系统，则资源有多个提供程序，并对系统使用适当的提供程序。例如，在 Red Hat Linux 系统上，package 提供程序使用 yum 来安装软件包。在 Ubuntu Linux 系统上，package 提供程序使用 apt-get。

您将资源实施为具有以下通用格式的 Ruby 代码块。

```
resource_type "resource_name" do
  attribute1 'value1'
  attribute2 'value2'
  ...
  action :action_name
  notifies : action 'resource'
end
```

元素有：

资源类型

(必选) 该示例包含三种资源类型，即 package、service 和 template。

资源名称

(必选) 该名称标识特定资源，有时用作其中一个属性的默认值。在示例中，package 代表名为 haproxy 的程序包资源，而第一个 template 资源代表名为 /etc/default/haproxy 的配置文件。

Attributes

(可选) 属性指定资源配置，并因资源类型以及您希望的资源配置方式而异。

- 该示例的 `template` 资源显式定义一组属性，以指定所创建文件的源、所有者、组和模式。
- 该示例的 `package` 和 `service` 资源没有显式定义任何属性。

  资源名称通常为必需属性的默认值，有时只需资源名称即可。例如，资源名称是 `package` 资源的 `package_name` 属性的默认值，该属性为唯一必需的属性。

还有一些称为 guard 属性的专有属性，这些属性指定资源提供程序何时执行操作。例如，`only_if` 属性指示资源提供程序只有在满足指定条件时才执行操作。HAProxy 配方不使用 guard 属性，但以下几个示例使用该属性。

操作和通知

(可选) 操作和通知指定提供程序要执行哪些任务。

- `action` 指示提供程序采取指定的操作，如安装或创建。

  每种资源都有一组取决于特定资源的操作，其中一个是默认操作。在示例中，`package` 资源的操作为 `install`，该操作会指示提供程序安装软件包。第一种 `template` 资源没有 `action` 元素，因此提供程序会执行默认的 `create` 操作。
- `notifies` 指示另一种资源的提供程序执行操作，但前提是资源的状态已更改。

  `notifies` 通常与 `template` 和 `file` 等资源一起使用，以执行诸如在修改配置文件后重新启动服务等操作。资源没有默认通知。如果需要通知，资源必须具有显式 `notifies` 元素。在 HAProxy 配方中，第二个 `template` 资源会在相关配置文件已更改的情况下通知 haproxy `service` 资源重新启动 HAProxy 服务。

有时资源依赖于操作系统。

- 有些资源只能在 Linux 或 Windows 系统上使用。

  例如，package 会将软件包安装在 Linux 系统上，而 windows_package 会将软件包安装在 Windows 系统上。
- 有些资源可以用在任何操作系统上，但却拥有特定于操作系统的属性。

  例如，file 资源可用于 Linux 或 Windows 系统上，但却拥有用于配置权限的独立属性集。

有关标准资源的说明，包括每种资源的可用属性、操作和通知，请参阅关于资源和提供程序。

流控制

由于配方为 Ruby 应用程序，因此您可以使用 Ruby 控制结构将流控制纳入到配方中。例如，您可以使用 Ruby 条件逻辑使配方在不同的系统上有不同的行为。HAProxy 配方包括一个 `if` 块，该块使用 `template` 资源创建配置文件，但前提是配方在 Debian 或 Ubuntu 系统上运行。

另一个常见情况是，使用循环利用不同的属性设置多次执行资源。例如，您可以使用循环创建一组目录，从而利用不同目录名称多次执行 `directory` 资源。

> **Note**
>
> 如果您不熟悉 Ruby，请参阅适用于 Chef 的 Ruby 基础知识，该部分介绍了您需要了解的有关大部分配方的内容。

包含的配方

`include_recipe` 会在您的代码中包括其他配方，这样有助于将您的配方模块化，并在多个配方中重复使用相同的代码。当您运行主机配方时，Chef 会在执行主机配方之前，使用指定配方的代码替换每个 `include_recipe` 元素。您通过使用标准 Chef `cookbook_name::recipe_name` 语法来标识包含的配方，其中 `recipe_name` 省略了 `.rb` 扩展名。该示例包括一个配方，即 `haproxy::service`，它表示 HAProxy 服务。

> **Note**
>
> 如果您在运行于 11.10 及更高版本的 Chef 上的配方中使用 `include_recipe`，以包含来自其他说明书的配方，则必须使用 `depends` 语句声明说明书的 `metadata.rb` 文件中的依赖关系。有关更多信息，请参阅 实施配方：Chef 11.10。

示例 1：安装软件包

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS：Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

软件包安装是配方的较常见用法之一，可能非常简单，具体取决于软件包。例如，以下配方会在 Linux 系统上安装 Git。

```
package 'git' do
  action :install
end
```

package 资源会处理软件包安装。对于本示例，您不需要指定任何属性。资源名称为 package_name 属性的默认值，可标识软件包。install 操作会指导提供程序安装软件包。您可以通过跳过 install 来简化代码；它是 package 资源的默认操作。当您运行配方时，Chef 会利用相应的提供程序来安装软件包。在将用于示例的 Ubuntu 系统上，提供程序会通过调用 apt-get 来安装 Git。

> **Note**
>
> 在 Windows 系统上安装软件的过程稍有不同。有关更多信息，请参阅 安装 Windows 软件。

要使用 Test Kitchen 在 Vagrant 中运行此配方，首先需要设置说明书，并初始化和配置 Test Kitchen。以下内容针对的是 Linux 系统，但 Windows 和 Macintosh 系统的过程与此过程基本类似。首先打开终端窗口；本章中的所有示例均使用命令行工具。

准备说明书

1. 在您的主目录中，创建一个名为 opsworks_cookbooks 的子目录，其中将包含本章的所有说明书。然后，为该说明书创建一个名为 installpkg 的子目录，并导航到该子目录。
2. 在 installpkg 中，创建一个名为 metadata.rb 的文件，其中包含以下代码。

   ```
   name "installpkg"
   version "0.1.0"
   ```

   为简单起见，本章中的示例仅指定说明书名称和版本，但是 metadata.rb 可以包含各种说明书元数据。有关更多信息，请参阅关于说明书元数据。

   > **Note**
   >
   > 确保在初始化 Test Kitchen 之前创建 metadata.rb；它会使用相应数据创建默认配置文件。

3. 在 installpkg 中，运行 kitchen init，它会初始化 Test Kitchen 并安装默认的 Vagrant 驱动程序。

4. kitchen init 命令在 installpkg 中创建名为 .kitchen.yml 的 YAML 配置文件。在您常用的文本编辑器中打开 文件。.kitchen.yml 文件中包含 platforms 部分，该部分指定在哪些系统上运行配方。Test Kitchen 创建一个实例，并在每个平台上运行指定的配方。

   > **Note**
   >
   > 默认情况下，Test Kitchen 每次在一个平台上运行配方。如果您将 -p 参数添加到创建实例的任何命令中，则 Test Kitchen 将在每个平台上并行运行配方。

   对于此示例，一个平台已足够，因此请编辑 .kitchen.yml 以删除 centos-6.4 平台。您的 .kitchen.yml 文件现在应如下所示：

   ```
   ---
   driver:
     name: vagrant

   provisioner:
     name: chef_solo

   platforms:
     - name: ubuntu-12.04

   suites:
     - name: default
       run_list:
         - recipe[installpkg::default]
       attributes:
   ```

   Test Kitchen 仅运行处于 .kitchen.yml 运行列表中的那些配方。您使用 [*cookbook_name*::*recipe_name*] 格式标识配方，其中，*recipe_name* 省略了 .rb 扩展名。最初，.kitchen.yml 运行列表中包含说明书的默认配方 installpkg::default。那是您要实施的配方，因此您不必修改运行列表。

5. 创建 installpkg 的子目录，该子目录名为 recipes。

   如果说明书中包含配方 (大部分均包含)，则这些配方一定位于 recipes 子目录中。

现在，您可以将配方添加到说明书中，并使用 Test Kitchen 在实例上运行它。

运行配方

1. 创建一个名为 `default.rb` 的文件 (其中包含本节开头的 Git 安装示例代码)，并将该文件保存到 `recipes` 子目录中。
2. 在 `installpkg` 目录中，运行 `kitchen converge`。该命令会在 Vagrant 中启动一个新的 Ubuntu 实例，将您的说明书复制到实例中，并开始运行 Chef，以执行 `.kitchen.yml` 运行列表中的配方。
3. 要验证配方是否已成功，请运行 `kitchen login`，这样可打开指向实例的 SSH 连接。然后运行 `git --version`，以验证是否已成功安装 Git。要返回到您的工作站，请运行 `exit`。
4. 完成后，运行 `kitchen destroy` 以关闭实例。下一个示例使用不同的说明书。

此示例是一个很好的入门方式，但它特别简单。其他软件包安装起来可能更复杂；您可能需要执行以下任意或所有操作：

- 创建和配置一个用户。
- 创建针对数据、日志等的一个或多个目录。
- 安装一个或多个配置文件。
- 为不同的操作系统指定不同的软件包名称或属性值。
- 启动一项服务，然后根据需要重新启动它。

以下示例描述了如何解决这些问题，并介绍了其他一些有用的操作。

示例 2：管理用户

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

另一个简单任务是在实例上管理用户。以下配方会向 Linux 实例中添加新用户。

```
user "myuser" do
```

```
  home "/home/newuser"
  shell "/bin/bash"
end
```

您使用 user 资源同时在 Linux 和 Windows 系统上管理用户，但某些属性仅适用于一个系统。该示例会创建名为 myuser 的用户，并指定其主目录和 Shell。未指定任何操作，因此资源会使用默认的 create 操作。您可以向 user 添加属性以指定各种其他设置，例如其密码或组 ID。也可以将 user 用于相关用户管理任务，例如修改用户设置或删除用户。有关更多信息，请参阅 user。

运行配方

1. 在 opsworks_cookbooks 中创建一个名为 newuser 的目录并导航到该目录。
2. 创建包含以下代码的 metadata.rb 文件，并将其保存至 newuser。

   ```
   name "newuser"
   version "0.1.0"
   ```

3. 按示例 1：安装软件包中所述初始化和配置 Test Kitchen，然后将 recipes 目录添加到 newuser 目录内。
4. 将 default.rb 文件连同示例配方一起添加到说明书的 recipes 目录中。
5. 运行 kitchen converge 以执行该配方。
6. 使用 kitchen login 登录到实例，并通过运行 cat /etc/passwd 验证新用户是否存在。myuser 用户应位于文件的底部。

示例 3：创建目录

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

当您在实例上安装软件包时，通常需要创建一些配置文件，并将这些配置文件放在适当的目录中。但是，这些目录可能尚不存在。您可能还需要针对数据、日志文件等创建目录。例如，首先启动您用于大多数示例的 Ubuntu 系统，/srv 目录没有子目录。如果您要安装应用程序服务器，则可能需要一个 /srv/www/ 目录，并且可能还需要用于数据文件、日志等的一些子目录。以下配方会在实例上创建 /srv/www/。

```
directory "/srv/www/" do
  mode 0755
  owner 'root'
  group 'root'
  action :create
end
```

您使用 directory 资源同时在 Linux 和 Windows 系统上创建和配置目录，但某些属性的用法不同。资源名称是资源的 path 属性的默认值，因此该示例会创建 /srv/www/，并指定其 mode、owner 和 group 属性。

运行配方

1. 在 opsworks_cookbooks 中创建名为 createdir 的目录并导航到该目录。
2. 如示例 1：安装软件包中所述，初始化和配置 Test Kitchen，然后将 recipes 目录添加到 createdir 内。
3. 将 default.rb 文件连同配方代码一起添加到说明书的 recipes 子目录中。
4. 运行 kitchen converge 以执行该配方。
5. 运行 kitchen login，导航到 /srv 并验证它是否拥有 www 子目录。
6. 运行 exit 以返回到您的工作站，但让实例保持运行状态。

> **Note**
>
> 要在实例上创建相对于主目录的目录，请使用 #{ENV['HOME']} 表示主目录。例如，以下内容会创建 ~/shared 目录。
>
> ```
> directory "#{ENV['HOME']}/shared" do
>   ...
> end
> ```

假设您要创建一个更深层嵌套的目录，例如 /srv/www/shared。您可以按如下所示修改前面的配方。

```
directory "/srv/www/shared" do
  mode 0755
  owner 'root'
  group 'root'
  action :create
end
```

运行配方

1. 用前面的配方替换 default.rb 中的代码。
2. 从 kitchen converge 目录运行 createdir。
3. 要验证是否确实已创建目录，请运行 kitchen login，导航到 /srv/www，并验证它是否包含 shared 子目录。
4. 运行 kitchen destroy 以关闭实例。

您将注意到 kitchen converge 命令运行速度更快。这是因为该实例已在运行，因此不需要启动实例、安装 Chef 等。Test Kitchen 直接将更新的说明书复制到实例中并开始运行 Chef。

现在，再次运行 kitchen converge，这将会在全新实例上执行配方。现在，您将看到以下结果。

```
Chef Client failed. 0 resources updated in 1.908125788 seconds
[2014-06-20T20:54:26+00:00] ERROR: directory[/srv/www/shared] (createdir::default line
 1) had an error: Chef::Exceptions::EnclosingDirectoryDoesNotExist: Parent directory /
srv/www does not exist, cannot create /srv/www/shared
[2014-06-20T20:54:26+00:00] FATAL: Chef::Exceptions::ChildConvergeError: Chef run
 process exited unsuccessfully (exit code 1)
>>>>>> Converge failed on instance <default-ubuntu-1204>.
>>>>>> Please see .kitchen/logs/default-ubuntu-1204.log for more details
>>>>>> ------Exception-------
>>>>>> Class: Kitchen::ActionFailed
>>>>>> Message: SSH exited (1) for command: [sudo -E chef-solo --config /tmp/kitchen/
solo.rb --json-attributes /tmp/kitchen/dna.json  --log_level info]
>>>>>> ----------------------
```

发生了什么？ 问题在于，默认情况下，directory 资源一次只能创建一个目录；它无法创建一系列目录。配方之前能够正常工作的原因是，您在实例上运行的第一个配方已经创建了 /srv/www，因此创建 /srv/www/shared 时仅创建了一个子目录。

> **Note**
>
> 在运行 `kitchen converge` 时，请确保了解您是在新实例上还是在现有实例上运行配方。您可能会得到不同的结果。

要创建一系列子目录，请将 `recursive` 属性添加到 `directory` 中，并将其设置为 `true`。以下配方会直接在干净的实例上创建 `/srv/www/shared`。

```
directory "/srv/www/shared" do
  mode 0755
  owner 'root'
  group 'root'
  recursive true
  action :create
end
```

示例 4：添加流控制

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

有些配方仅仅是一系列 Chef 资源。在这种情况下，当您运行配方时，它将依次执行每个资源提供程序。但是，拥有一个更复杂的执行路径通常很有用。以下是两种常见情境：

- 您希望配方使用不同的属性设置多次执行同一资源。
- 您希望在不同的操作系统上使用不同的属性设置。

您可以通过将 Ruby 控制结构纳入到配方中来应对诸如此类的情境。本部分将介绍如何从示例 3：创建目录中修改配方，从而应对上述两种情境。

主题

- 迭代

- 条件逻辑

迭代

示例 3：创建目录演示了如何使用 directory 资源创建一个目录或一系列目录。但是，假设您希望创建两个单独的目录，即 /srv/www/config 和 /srv/www/shared。您可以针对每个目录实施单独的目录资源，但是如果您希望创建非常多的目录，则该方法可能会非常麻烦。以下配方显示了处理该任务的一个更简单的方法。

```
[ "/srv/www/config", "/srv/www/shared" ].each do |path|
  directory path do
    mode 0755
    owner 'root'
    group 'root'
    recursive true
    action :create
  end
end
```

配方会使用包含子目录路径的字符串集合，而不是针对每个子目录使用单独的目录资源。Ruby each 方法会针对每个集合元素执行一次资源，从第一个元素开始。元素的值通过 path 变量在资源中表示，在此情况下，该变量表示目录路径。您可以轻松调整此示例，以创建任意数量的子目录。

运行配方

1. 不退出 createdir 目录；您将对接下来的几个示例使用该说明书。
2. 如果尚未运行 kitchen destroy，则运行它，以便从干净的实例开始。
3. 用示例替换 default.rb 中的代码，并运行 kitchen converge。
4. 登录到实例；您会在 /srv 下看到新创建的目录。

可以用哈希表来指定每个迭代的两个值。以下配方可创建 /srv/www/config 和 /srv/www/shared，每个都有不同的模式。

```
{ "/srv/www/config" => 0644, "/srv/www/shared" => 0755 }.each do |path, mode_value|
  directory path do
    mode mode_value
    owner 'root'
```

```
    group 'root'
    recursive true
    action :create
  end
end
```

运行配方

1. 如果尚未运行 `kitchen destroy`，则运行它，以便从干净的实例开始。
2. 用示例替换 `default.rb` 中的代码，并运行 `kitchen converge`。
3. 登录到实例；您会在 `/srv` 下看到新创建的具有指定模式的目录。

> **Note**
>
> AWS OpsWorks 堆栈配方通常使用这种方法从堆栈配置和部署 JSON（基本上是一个大型哈希表）中提取值，然后将其插入资源中。有关示例，请参阅Deploy 配方。

条件逻辑

您也可以使用 Ruby 条件逻辑来创建多个执行分支。以下配方使用 `if-elsif-else` 逻辑来扩展前面的示例，以便它可创建名为 `/srv/www/shared` 的子目录，但是仅适用于 Debian 和 Ubuntu 系统。对于所有其他系统，它会记录显示在 Test Kitchen 输出中的错误消息。

```
if platform?("debian", "ubuntu")
  directory "/srv/www/shared" do
    mode 0755
    owner 'root'
    group 'root'
    recursive true
    action :create
  end
else
  log "Unsupported system"
end
```

运行示例配方

1. 如果您的实例仍处于运行状态，请运行 `kitchen destroy` 将其关闭。

2. 用示例代码替换 default.rb 中的代码。
3. 编辑 .kitchen.yml，将 CentOS 6.4 系统添加到平台列表中。文件的 platforms 部分此时应类似于如下内容。

```
...
platforms:
  - name: ubuntu-12.04
  - name: centos-6.4
...
```

4. 运行 kitchen converge，这将创建一个实例，并按顺序在 .kitchen.yml 中针对每个平台运行配方。

> **Note**
>
> 如果希望仅收敛一个实例，请将实例名称作为参数添加。例如，要仅在 Ubuntu 平台上收敛配方，请运行 kitchen converge default-ubuntu-1204。如果您忘记了平台名称，则可运行 kitchen list。

您应该会在 Test Kitchen 输出的 CentOS 部分中看到您的日志消息，该消息将类似于以下内容：

```
...
Converging 1 resources
Recipe: createdir::default
* log[Unsupported system] action write[2014-06-23T19:10:30+00:00] INFO: Processing
 log[Unsupported system] action write (createdir::default line 12)
[2014-06-23T19:10:30+00:00] INFO: Unsupported system

[2014-06-23T19:10:30+00:00] INFO: Chef Run complete in 0.004972162 seconds
```

现在，您可以登录到实例，并验证是否已创建目录。但是，现在您不能简单地运行 kitchen login，而是必须通过附加平台名称来指定具体的实例，例如 kitchen login default-ubuntu-1204。

> **Note**
>
> 如果 Test Kitchen 命令采用了某个实例名称，则您无需键入完整的名称。Test Kitchen 会将实例名称视为 Ruby 正则表达式，因此只需足够的字符，即可提供一个唯一匹配项。例如，您可以通过运行 `kitchen converge ub` 仅收敛 Ubuntu 实例，或通过运行 `kitchen login 64` 登录到 CentOS 实例。

此时，您可能会遇到的问题是，配方如何知道它正在哪个平台上运行。Chef 针对收集系统数据 (包括平台) 的每次运行而运行名为 Ohai 的工具，并在称为节点对象的结构中将其表示为一组属性。Chef `platform?` 方法会将括号中的系统与 Ohai 平台值进行比较，如果其中一个匹配，则返回 true。

您可以使用 `node['attribute_name']` 直接在代码中引用节点属性的值。例如，平台值由 `node['platform']` 表示。例如，您可能已将前面的示例编写为如下内容。

```
if node[:platform] == 'debian' or node[:platform] == 'ubuntu'
  directory "/srv/www/shared" do
    mode 0755
    owner 'root'
    group 'root'
    recursive true
    action :create
  end
else
  log "Unsupported system"
end
```

将条件逻辑包含在配方中的一个常见原因是为了顺应如下事实：不同的 Linux 系列有时会对软件包、目录等使用不同的名称。例如，Apache 软件包名称在 CentOS 系统上为 `httpd`，而在 Ubuntu 系统上为 `apache2`。

如果您需要对不同的系统使用不同的字符串，则 Chef `value_for_platform` 方法是比 `if-elsif-else` 更简单的解决方案。以下配方会在 CentOS 系统上创建 `/srv/www/shared` 目录、在 Ubuntu 系统上创建 `/srv/www/data` 目录，而在所有其他系统上创建 `/srv/www/config`。

```
data_dir = value_for_platform(
  "centos" => { "default" => "/srv/www/shared" },
  "ubuntu" => { "default" => "/srv/www/data" },
```

```
  "default" => "/srv/www/config"
)
directory data_dir do
  mode 0755
  owner 'root'
  group 'root'
  recursive true
  action :create
end
```

value_for_platform 将相应的路径分配至 data_dir，而 directory 资源则使用该值创建目录。

运行示例配方

1. 如果您的实例仍处于运行状态，请运行 kitchen destroy 将其关闭。
2. 用示例代码替换 default.rb 中的代码。
3. 运行 kitchen converge，然后登录到每个实例，以验证相应的目录是否存在。

示例 5：使用属性

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

前面几节中的配方对除平台之外的一切内容均使用了硬编码值。如果您要在多个配方中使用相同的值，则此方法可能不太方便。您可以通过在说明书中包含属性文件，独立于配方单独定义值。

属性文件是一个 Ruby 应用程序，该应用程序可将值分配给一个或多个属性。它必须位于说明书的 attributes 文件夹中。Chef 将属性纳入到节点对象中，任何配方均可通过引用相应属性来使用属性值。本主题介绍了如何从迭代中修改配方以使用属性。以下是原始配方，供您参考。

```
[ "/srv/www/config", "/srv/www/shared" ].each do |path|
  directory path do
    mode 0755
```

```
    owner 'root'
    group 'root'
    recursive true
    action :create
  end
end
```

以下内容定义了子目录名称、模式、所有者和组值的属性。

```
default['createdir']['shared_dir'] = 'shared'
default['createdir']['config_dir'] = 'config'
default['createdir']['mode'] = 0755
default['createdir']['owner'] = 'root'
default['createdir']['group'] = 'root'
```

请注意以下几点：

- 每个定义均以属性类型开头。

  如果某个属性被定义多次 (可能在不同的属性文件中)，则属性类型会指定该属性的优先级，这样可确定将哪个定义纳入到节点对象中。有关更多信息，请参阅 属性优先顺序。此示例中的所有定义都具有 `default` 属性类型，该属性类型是用于此目的的通用类型。
- 属性具有嵌套名称。

  节点对象基本上是可具有任意嵌套深度的哈希表，因此属性名称可以且通常是嵌套的。该属性文件遵循使用包含说明书名称 `createdir` 的嵌套名称作为第一个元素的标准做法。

使用 createdir 作为属性的第一个元素的原因是，当您运行 Chef 时，Chef 会将来自每个说明书的属性纳入到节点对象中。在 AWS OpsWorks Stacks 中，节点对象除了您定义的任何属性外，还包括内置食谱中的大量属性。在属性名称中包含说明书名称可以降低与来自其他说明书的属性名称相冲突的风险，特别是在您的属性具有诸如 `port` 或 `user` 等名称的情况下更是如此。除非您希望覆盖属性的值，否则请勿将属性命名为诸如 `[:apache2][:user]` 等名称。有关更多信息，请参阅 使用自定义说明书属性。

以下示例显示了使用属性而不是硬编码值的原始配方。

```
[ "/srv/www/#{node['createdir']['shared_dir']}", "/srv/www/#{node['createdir']
['config_dir']}" ].each do |path|
```

```
  directory path do
    mode node['createdir']['mode']
    owner node['createdir']['owner']
    group node['createdir']['group']
    recursive true
    action :create
  end
end
```

> **Note**
>
> 如果要将属性值纳入到字符串中，请将其包含在 #{} 中。在前面的示例中，#{node['createdir']['shared_dir']} 将“shared”附加到“/srv/www/”。

运行配方

1. 运行 kitchen destroy，以便从全新的实例开始。
2. 将 recipes/default.rb 中的代码替换为前面的配方示例。
3. 创建一个名为 createdir 的 attributes 子目录，并添加一个名为 default.rb 的文件，其中包含属性定义。
4. 编辑 .kitchen.yml，从平台列表中删除 CentOS。
5. 运行 kitchen converge，然后登录到实例并验证存在 /srv/www/shared 和 /srv/www/config。

> **Note**
>
> 使用 AWS OpsWorks Stacks，将值定义为属性还有其他好处；您可以使用自定义 JSON 在每个堆栈甚至每个部署的基础上覆盖这些值。它可用于各种目的，其中包括：
>
> - 您可以自定义配方的行为，例如配置设置或用户名，而无需修改说明书。
>
>   例如，您可以对不同的堆栈使用同一说明书，并使用自定义 JSON 来指定特定堆栈的关键配置设置。这样可以节省您修改说明书或对每个堆栈使用不同说明书所需的时间和精力。
> - 您不必将潜在敏感信息 (如数据库密码) 置于您的说明书存储库中。
>
>   您可以使用属性来定义默认值，然后采用自定义 JSON 用实际值来覆盖该值。

有关如何使用自定义 JSON 覆盖属性的更多信息，请参阅覆盖属性。

属性文件名为 default.rb，因为它是一个 Ruby 应用程序，但是该应用程序相当简单。这意味着您可以使用条件逻辑来根据操作系统指定属性值。在条件逻辑中，您在配方中为不同 Linux 系列指定了不同的子目录名称。使用属性文件，您可以将条件逻辑放在属性文件中。

以下属性文件使用 value_for_platform 指定不同的 ['shared_dir'] 属性值，具体取决于操作系统。对于其他条件，您可以使用 Ruby if-elsif-else 逻辑或 case 语句。

```
data_dir = value_for_platform(
  "centos" => { "default" => "shared" },
  "ubuntu" => { "default" => "data" },
  "default" => "user_data"
)
default['createdir']['shared_dir'] = data_dir
default['createdir']['config_dir'] = "config"
default['createdir']['mode'] = 0755
default['createdir']['owner'] = 'root'
default['createdir']['group'] = 'root'
```

运行配方

1. 运行 kitchen destroy，以便从全新的实例开始。
2. 将 attributes/default.rb 中的代码替换为前面的示例。
3. 编辑 .kitchen.yml，将 CentOS 平台添加至平台部分，如 条件逻辑 中所述。
4. 运行 kitchen converge，然后登录到实例，以验证目录是否存在。

完成后，运行 kitchen destroy 以终止实例。下一个示例将使用新说明书。

示例 6：创建文件

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

创建目录后，通常需要用配置文件、数据文件等填充它们。本主题介绍了将文件安装在实例上的两种方法。

主题

- 从说明书安装文件
- 从模板创建文件

从说明书安装文件

将文件安装在实例上的最简单方法是使用 `cookbook_file` 资源，它将文件从说明书复制到 Linux 和 Windows 系统实例上的指定位置。此实例扩展 示例 3：创建目录 中的配方，以便在创建目录后将数据文件添加至 `/srv/www/shared`。以下是原始配方，供您参考。

```
directory "/srv/www/shared" do
  mode 0755
  owner 'root'
  group 'root'
  recursive true
  action :create
end
```

设置说明书

1. 在 `opsworks_cookbooks` 目录中，创建名为 `createfile` 的实例并导航到其中。
2. 将包含以下内容的 `metadata.rb` 文件添加到 `createfile`。

   ```
   name "createfile"
   version "0.1.0"
   ```

3. 如示例 1：安装软件包中所述，初始化和配置 Test Kitchen，并从 `platforms` 列表中删除 CentOS。
4. 将 `recipes` 子目录添加 `createfile` 中。

要安装的文件包含以下 JSON 数据。

```
{
  "my_name" : "myname",
  "your_name" : "yourname",
  "a_number" : 42,
  "a_boolean" : true
}
```

设置数据文件

1. 将 `files` 子目录添加到 `createfile`，并将 `default` 子目录添加到 `files`。您随 `cookbook_file` 安装的任何文件必须位于 `files` 的子目录中，例如此示例中的 `files/default`。

   > **Note**
   >
   > 如果要为不同的系统指定不同的文件，可以将每个特定于系统的文件放在根据系统命名的子文件夹中，例如 `files/ubuntu`。`cookbook_file` 资源会复制特定于系统的相应文件，如果该文件不存在，则会使用 `default` 文件。有关更多信息，请参阅 cookbook_file。

2. 使用前面示例中的 JSON 创建名为 `example_data.json` 的文件，并将其添加至 `files/default`。

以下配方会将 `example_data.json` 复制到指定位置。

```
directory "/srv/www/shared" do
  mode 0755
  owner 'root'
  group 'root'
  recursive true
  action :create
```

```
end

cookbook_file "/srv/www/shared/example_data.json" do
  source "example_data.json"
  mode 0644
  action :create_if_missing
end
```

在目录资源创建 /srv/www/shared 后，cookbook_file 资源会将 example_data.json 复制到该目录中，并设置该文件的用户、组和模式。

> **Note**
>
> cookbook_file 资源引入了一个新操作：create_if_missing。您也可以使用 create 操作，但是这样会覆盖现有文件。如果您不想覆盖任何内容，请使用 create_if_missing，这样会安装 example_data.json (如果该文件尚不存在)。

运行配方

1. 运行 kitchen destroy，以便从全新的实例开始。
2. 创建包含前面配方的 default.rb 文件，并将该文件保存至 recipes。
3. 运行 kitchen converge，然后登录到实例以验证 /srv/www/shared 包含 example_data.json。

从模板创建文件

cookbook_file 资源对某些用途来说非常有用，但它只会安装您在说明书中所拥有的任何文件。template 资源通过从模板动态创建文件，从而提供了一种将文件安装在 Windows 或 Linux 实例上的更灵活方法。然后，您可以在运行时确定文件内容的详细信息，并根据需要进行更改。例如，在启动实例时，您可能希望配置文件具有特定设置，并在以后向堆栈中添加更多实例时修改该设置。

此示例会修改 createfile 说明书，以使用 template 资源安装略经修改的 example_data.json 版本。

下面是已安装文件的外观。

```
{
```

```
  "my_name" : "myname",
  "your_name" : "yourname",
  "a_number" : 42,
  "a_boolean" : true,
  "a_string" : "some string",
  "platform" : "ubuntu"
}
```

模板资源通常与属性文件结合使用，因此该示例使用其中一个来定义以下值。

```
default['createfile']['my_name'] = 'myname'
default['createfile']['your_name'] = 'yourname'
default['createfile']['install_file'] = true
```

设置说明书

1. 删除 createfile 说明书的 files 目录及其内容。
2. 将 attributes 子目录添加到 createfile，将 default.rb 文件添加到 attributes (其中包含前面的属性定义)。

模板是一个 .erb 文件，该文件基本上为最终文件的副本，其中某些内容由占位符表示。当 template 资源创建该文件时，它会将模板的内容复制到指定文件，并用其分配的值覆盖占位符。此处为 example_data.json 的模板。

```
{
  "my_name" : "<%= node['createfile']['my_name'] %>",
  "your_name" : "<%= node['createfile']['your_name'] %>",
  "a_number" : 42,
  "a_boolean" : <%= @a_boolean_var %>,
  "a_string" : "<%= @a_string_var %>",
  "platform" : "<%= node['platform'] %>"
}
```

<%=...%> 值为占位符。

- <%=node[...]%> 表示节点属性值。

  在该示例中，“your_name”值为占位符，表示来自说明书的属性文件的其中一个属性值。

- <%=@...%> 表示模板资源中定义的变量值 (稍后将进行讨论)。

创建 模板文件

1. 将 templates 子目录添加到 createfile 说明书，并将 default 子目录添加到 templates。

   > **Note**
   >
   > templates 目录的工作方式与 files 目录很相似。您可以将特定于系统的模板放在子目录 (例如根据系统命名的 ubuntu) 中。template 资源会使用特定于系统的适当模板 (如果该模板存在)，否则会使用 default 模板。

2. 创建名为 example_data.json.erb 的目录并将其放在 templates/default 目录中。模板名称是任意性的，但您通常可以通过将 .erb 附加到文件名 (包括任何扩展名) 来创建模板名称。

以下配方使用 template 资源来创建 /srv/www/shared/example_data.json。

```
directory "/srv/www/shared" do
  mode 0755
  owner 'root'
  group 'root'
  recursive true
  action :create
end

template "/srv/www/shared/example_data.json" do
  source "example_data.json.erb"
  mode 0644
  variables(
    :a_boolean_var => true,
    :a_string_var => "some string"
  )
  only_if {node['createfile']['install_file']}
end
```

template 资源会从模板创建 example_data.json，并将其安装在 /srv/www/shared 中。

- 模板名称 /srv/www/shared/example_data.json 会指定所安装文件的路径和名称。
- source 属性会指定用于创建文件的模板。

- mode 属性会指定所安装文件的模式。
- 资源会定义两个变量，即 a_boolean_var 和 a_string_var。

  当资源创建 example_data.json 时，它会用来自资源的相应值覆盖模板中的变量占位符。
- 只有在 only_if 设置为 时， ['createfile']['install_file']guardtrue 属性才会指示资源创建文件。

运行配方

1. 运行 kitchen destroy，以便从全新的实例开始。
2. 将 recipes/default.rb 中的代码替换为前面的示例。
3. 运行 kitchen converge，然后登录到实例，以验证文件是否位于 /srv/www/shared 中以及是否拥有正确的内容。

完成后，运行 kitchen destroy 以关闭实例。下一节将使用新的说明书。

示例 7：运行命令和脚本

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

Chef 资源可以在实例上处理各种任务，但有时最好使用 shell 命令或脚本。例如，您可能已经拥有用于完成特定任务的脚本，继续使用它们而不是实施新的代码会更容易。本节介绍了如何在实例上运行命令或脚本。

主题

- 运行命令
- 运行脚本

运行命令

script 资源会运行一个或多个命令。它支持 csh、bash、Perl、Python 和 Ruby 命令解释器，因此可用于 Linux 或 Windows 系统，前提是它们已安装适当的解释器。本主题介绍了如何在 Linux 实例上运

行简单的 bash 命令。Chef 还支持 powershell_script 和 batch 资源在 Windows 上运行脚本。有关更多信息，请参阅 运行 Windows PowerShell 脚本。

开始使用

1. 在 opsworks_cookbooks 目录中，创建名为 script 的实例并导航到其中。
2. 将包含以下内容的 metadata.rb 文件添加到 script。

```
name "script"
version "0.1.0"
```

3. 如示例 1：安装软件包中所述，初始化和配置 Test Kitchen，并从 platforms 列表中删除 CentOS。
4. 在 script 中创建名为 recipes 的子目录。

您可以通过使用 script 资源本身来运行命令，但是 Chef 还支持一组特定于命令解释器的资源版本，这些资源版本根据解释器命名。以下配方使用 bash 资源运行简单的 bash 脚本。

```
bash "install_something" do
  user "root"
  cwd "/tmp"
  code <<-EOH
    touch somefile
  EOH
  not_if do
    File.exists?("/tmp/somefile")
  end
end
```

bash 资源的配置如下所示。

- 它使用默认操作 run，该操作会运行 code 块中的命令。

  此示例有一个命令 touch somefile，但是 code 块可能包含多个命令。
- user 属性会指定执行相应命令的用户。
- cwd 属性指定工作目录。

  在本示例中，touch 会在 /tmp 目录中创建一个文件。

- 如果该文件已经存在，则 not_if guard 属性会指示资源不采取任何行动。

运行配方

1. 创建包含前面示例代码的 default.rb 文件，并将其保存至 recipes。
2. 运行 kitchen converge，然后登录到实例，以验证相应文件是否在 /tmp 中。

运行脚本

script 资源很方便，尤其是当您只需要运行一两个命令时更是如此，但是通常最好将脚本存储在某个文件中，然后执行该文件。execute 资源在 Linux 或 Windows 上运行指定可执行文件 (包括脚本文件)。本主题修改了来自前面示例的 script 说明书，以使用 execute 运行简单的 shell 脚本。您可将此示例轻松扩展为更复杂的脚本，或其他类型的可执行文件。

设置脚本文件

1. 将 files 子目录添加到 script，并将 default 子目录添加到 files。
2. 创建名为 touchfile 的包含以下内容的文件，然后将其添加到 files/default 中。此示例中使用了常见的 Bash 解释器行，但如果需要，可以替换为适用于您的 shell 环境的解释器。

   ```
   #!/usr/bin/env bash
   touch somefile
   ```

   脚本文件可以包含任意数量的命令。为方便起见，此示例脚本只有单个 touch 命令。

以下配方执行该脚本。

```
cookbook_file "/tmp/touchfile" do
  source "touchfile"
  mode 0755
end

execute "touchfile" do
  user "root"
  cwd "/tmp"
  command "./touchfile"
end
```

cookbook_file 资源会将脚本文件复制到 /tmp，然后设置模式，以使该文件可执行。execute 资源随后会按如下所示执行文件：

- user 属性指定命令的用户 (在此示例中为 root)。
- cwd 属性指定工作目录 (在此示例中为 /tmp)。
- command 属性指定要执行的脚本 (在此示例中为 touchfile)，该脚本位于工作目录中。

运行配方

1. 将 recipes/default.rb 中的代码替换为前面的示例。
2. 运行 kitchen converge，然后登录到实例，以验证 /tmp 现在是否包含脚本文件 (模式设置为 0755) 以及 somefile。

完成后，运行 kitchen destroy 以关闭实例。下一节将使用新的说明书。

示例 8：管理服务

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

应用程序服务器等软件包通常具有必须启动、停止、重新启动的关联服务。例如，您需要在安装软件包后或在实例完成启动后启动 Tomcat 服务，并在每次修改配置文件时重新启动该服务。本主题以 Tomcat 应用程序服务器为例，讨论了如何在 Linux 实例上管理服务的基础知识。服务资源在 Windows 实例上的工作方式大致相同，但在细节上存在一些差异。有关更多信息，请参阅 service。

> **Note**
>
> 该示例执行极低限度的 Tomcat 安装，刚刚足以演示如何使用 service 资源的基础知识。有关如何对功能更强的 Tomcat 服务器实施配方的示例，请参阅创建自定义 Tomcat 服务器层。

主题

- 定义和启动服务

- 使用通知来启动或重新启动服务

定义和启动服务

本节介绍了有关如何定义和启动服务的基础知识。

开始使用

1. 在 opsworks_cookbooks 目录中，创建名为 tomcat 的实例并导航到其中。
2. 将包含以下内容的 metadata.rb 文件添加到 tomcat。

```
name "tomcat"
version "0.1.0"
```

3. 如示例 1：安装软件包中所述，初始化和配置 Test Kitchen，并从 platforms 列表中删除 CentOS。
4. 将 recipes 子目录添加 tomcat 中。

您使用 service 资源管理服务。以下默认配方会安装 Tomcat 并启动该服务。

```
execute "install_updates" do
  command "apt-get update"
end

package "tomcat7" do
    action :install
end

include_recipe 'tomcat::service'

service 'tomcat' do
  action :start
end
```

此配方会执行以下操作：

- execute 资源会运行 apt-get update 以安装当前系统更新。

对于此示例中使用的 Ubuntu 实例，必须在安装 Tomcat 之前先安装这些更新。其他系统的要求可能会有所不同。

- `package` 资源会安装 Tomcat 7。
- 包含的 `tomcat::service` 配方会定义该服务，稍后将讨论此内容。
- `service` 资源会启动 Tomcat 服务。

  您还可以使用此资源发出其他命令，例如停止和重新启动服务。

以下示例展示了 `tomcat::service` 配方。

```
service 'tomcat' do
  service_name "tomcat7"
  supports :restart => true, :reload => false, :status => true
  action :nothing
end
```

此配方将创建 Tomcat 服务定义，具体如下：

- 其他配方使用资源名称 `tomcat` 来引用该服务。

  例如，`default.rb` 会引用 `tomcat` 以启动服务。
- `service_name` 资源指定服务名称。

  当您在实例上列出服务时，Tomcat 服务将被命名为 tomcat7。
- `supports` 会指定 Chef 管理服务的 `restart`、`reload` 和 `status` 命令的方式。
  - `true` 表示 Chef 可以使用 init 脚本或其他服务提供程序来运行该命令。
  - `false` 表示 Chef 必须尝试以其他方式运行该命令。

请注意，`action` 设置为 `:nothing`，它会指示资源不执行任何操作。服务资源确实支持诸如 `start` 和 `restart` 等操作。但是，本说明书遵循以下标准做法：使用不执行任何操作的服务定义，并在其他地方启动或重新启动服务。启动或重新启动服务的每个配方首先必须定义该服务，因此最简单的方法是，将服务定义放在单独的配方中，并根据需要将其纳入到其他配方中。

> **Note**
>
> 为简便起见，此示例的默认配方会在运行服务定义后使用 service 资源启动服务。生产实施通常会通过使用 notifies 来启动或重新启动服务 (稍后将进行讨论)。

运行配方

1. 创建包含默认配方示例的 default.rb 文件，并将其保存至 recipes。
2. 创建包含服务定义示例的 service.rb 文件，并将其保存至 recipes。
3. 运行 kitchen converge，然后登录到实例，并运行以下命令，以验证相应服务是否正在运行。

```
sudo service tomcat7 status
```

> **Note**
>
> 如果您单独从 default.rb 运行 service.rb，则必须编辑 .kitchen.yml，以便将 tomcat::service 添加至运行列表中。但是，当您包含某个配方时，在执行该配方之前，会将其代码纳入到父配方中。因此 service.rb 基本上是 default.rb 的一部分，不需要提供单独的运行列表项。

使用通知来启动或重新启动服务

生产实施通常不使用 service 启动或重新启动服务。相反，它们会将 notifies 添加到任意几个资源中。例如，如果要在修改配置文件后重新启动服务，请将 notifies 包括在相关 template 资源中。在显式重新启动服务方面，与使用 notifies 资源相比，使用 service 具有如下优势。

- notifies 元素仅在关联配置文件发生更改时才会重新启动服务，因此不会导致不必要的服务重新启动的风险。
- 在每次运行结束时，Chef 最多重新启动服务一次，无论运行包含多少个 notifies 均是如此。

  例如，Chef 运行可能包含多个模板资源，其中每个模板资源均会修改不同的配置文件，并要求在文件发生更改时重新启动服务。但是，您通常只需要在 Chef 运行结束时重新启动服务一次。否则，您可能会尝试重新启动尚未从之前的重新启动中完全恢复正常运行的服务，从而可能导致错误。

本示例会修改 tomcat::default，以包括使用 template 来重新启动服务的 notifies 资源。实际的示例会使用一个模板资源来创建其中一个 Tomcat 配置文件的自定义版本，但是这些文件相当长且复杂。为简便起见，该示例仅使用来自从模板创建文件的模板资源。它与 Tomcat 没有任何关系，但它提供了一种简单的方式来显示如何使用 notifies。有关如何使用模板创建 Tomcat 配置文件的示例，请参阅Setup 配方。

设置说明书

1. 将 templates 子目录添加到 tomcat，并将 default 子目录添加到 templates。
2. 将 example_data.json.erb 模板从 createfile 说明书复制到 templates/default 目录。
3. 将 attributes 子目录添加到 tomcat 中。
4. 将 default.rb 属性文件从 createfile 说明书复制到 attributes 目录。

以下配方使用 notifies 重新启动 Tomcat 服务。

```
execute "install_updates" do
  command "apt-get update"
end

package "tomcat7" do
    action :install
end

include_recipe 'tomcat::service'

service 'tomcat' do
  action :enable
end

directory "/srv/www/shared" do
  mode 0755
  owner 'root'
  group 'root'
  recursive true
  action :create
end

template "/srv/www/shared/example_data.json" do
```

```
  source "example_data.json.erb"
  mode 0644
  variables(
    :a_boolean_var => true,
    :a_string_var => "some string"
  )
  only_if {node['createfile']['install_file']}
  notifies :restart, resources(:service => 'tomcat')
end
```

该示例将来自从模板创建文件的配方合并到前一节的配方中，其中包含如下两个重大更改：

- `service` 资源仍然存在，但该资源的用途目前略有不同。

  `:enable` 操作会在启动时启用 Tomcat 服务。
- 模板资源现在包含 `notifies`，它会在 `example_data.json` 发生更改时重新启动 Tomcat 服务。

  这样可确保在每次配置更改后首次安装和重新启动 Tomcat 时启动该服务。

运行配方

1. 运行 `kitchen destroy`，以便从全新的实例开始。
2. 将 `default.rb` 中的代码替换为前面的示例。
3. 运行 `kitchen converge`，然后登录到实例，并验证相应服务是否正在运行。

> **Note**
>
> 如果要重新启动服务，但配方不包含诸如 `template` 这样支持 `notifies` 的资源，则可以改用虚拟 `execute` 资源。例如
>
> ```
> execute 'trigger tomcat service restart' do
>   command 'bin/true'
>   notifies :restart, resources(:service => 'tomcat')
> end
> ```

execute 资源必须具有 command 属性，即使您仅将该资源用作运行 notifies 的一种方法也是如此。此示例通过运行 /bin/true 来解决上述要求，该命令是一个只返回成功代码的 shell 命令。

示例 9：使用 Amazon EC2 实例

Important

该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

到目前为止，您一直在本地运行实例 VirtualBox。虽然这种方法简单快捷，但最终您要在 Amazon EC2 实例上测试您的配方。具体而言，如果要在 Amazon Linux 上运行配方，则它仅在 Amazon EC2 上可用。您可以使用类似的系统 (例如 CentOS) 进行初步实施和测试，但是在 Amazon Linux 上全面测试您的配方的唯一方法是使用 Amazon EC2 实例。

本主题介绍了如何在 Amazon EC2 实例上运行配方。您使用 Test Kitchen 和 Vagrant 的方式与前面几节的方式大同小异，但有如下两个区别：

- 驱动程序为 kitchen-ec2 而不是 Vagrant。
- 必须使用启动 Amazon EC2 实例所需的信息配置说明书的 .kitchen.yml 文件。

Note

另一种替代方法是使用 vagrant-aws Vagrant 插件。有关更多信息，请参阅 Vagrant AWS 提供程序。

您需要 AWS 凭证才能创建 Amazon EC2 实例。如果您还没有 Amazon Web Services account，可以按如下方式获取一个。

注册获取 AWS 账户

如果您没有 AWS 账户，请完成以下步骤来创建一个。

要注册 AWS 账户

1. 打开 https://portal.aws.amazon.com/billing/signup。
2. 按照屏幕上的说明进行操作。

   在注册时，将接到一通电话，要求使用电话键盘输入一个验证码。

   当您注册时 AWS 账户，就会创建AWS 账户根用户一个。根用户有权访问该账户中的所有 AWS 服务 和资源。作为安全最佳实践，请为用户分配管理访问权限，并且只使用根用户来执行需要根用户访问权限的任务。

AWS 注册过程完成后会向您发送一封确认电子邮件。在任何时候，您都可以通过转至 https://aws.amazon.com/ 并选择我的账户来查看当前的账户活动并管理您的账户。

创建具有管理访问权限的用户

注册后，请保护您的安全 AWS 账户 AWS 账户根用户 AWS IAM Identity Center，启用并创建管理用户，这样您就不会使用 root 用户执行日常任务。

保护你的 AWS 账户根用户

1. 选择 Root 用户并输入您的 AWS 账户 电子邮件地址，以账户所有者的身份登录。AWS Management Console在下一页上，输入您的密码。

   要获取使用根用户登录方面的帮助，请参阅 《AWS 登录 用户指南》中的以根用户身份登录。
2. 为您的根用户启用多重身份验证 (MFA)。

   有关说明，请参阅 I A M 用户指南中的为 AWS 账户 根用户启用虚拟 MFA 设备（控制台）。

创建具有管理访问权限的用户

1. 启用 IAM Identity Center。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的启用 AWS IAM Identity Center。
2. 在 IAM Identity Center 中，为用户授予管理访问权限。

   有关使用 IAM Identity Center 目录 作为身份源的教程，请参阅《用户指南》 IAM Identity Center 目录中的使用默认设置配置AWS IAM Identity Center 用户访问权限。

以具有管理访问权限的用户身份登录

- 要使用您的 IAM Identity Center 用户身份登录，请使用您在创建 IAM Identity Center 用户时发送到您的电子邮件地址的登录网址。

  有关使用 IAM Identity Center 用户登录的帮助，请参阅AWS 登录 用户指南中的登录 AWS 访问门户。

将访问权限分配给其他用户

1. 在 IAM Identity Center 中，创建一个权限集，该权限集遵循应用最低权限的最佳做法。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的创建权限集。
2. 将用户分配到一个组，然后为该组分配单点登录访问权限。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的添加组。

您应该创建一个有权访问 Amazon EC2 的 IAM 用户，并将该用户的访问权限和密钥保存到工作站上的安全位置。Test Kitchen 将使用这些凭证来创建实例。向 Test Kitchen 提供凭证的首选方法是将密钥分配给工作站上的以下环境变量。

> **Warning**
>
> IAM 用户拥有长期证书，这会带来安全风险。为帮助减轻这种风险，我们建议仅向这些用户提供执行任务所需的权限，并在不再需要这些用户时将其移除。

- AWS_ACCESS_KEY - 您用户的访问密钥，它类似于 AKIAIOSFODNN7EXAMPLE。
- AWS_SECRET_KEY — 你的用户的密钥，看起来像 wjalrxutnfemi/k7mdeng/CYEXAMPLEKEY。bPxRfi

这种方法通过将包含您的凭证的项目上传到公共存储库，从而可减少意外泄露您的账户的几率。

设置说明书

1. 要使用 `kitchen-ec2` 驱动程序，必须将 `ruby-dev` 软件包安装在您的系统上。以下示例命令演示了如何使用 `aptitude` 将软件包安装在 Ubuntu 系统上。

```
sudo aptitude install ruby1.9.1-dev
```

2. kitchen-ec2 驱动程序是一个 Gem，可通过如下方法进行安装：

   ```
   gem install kitchen-ec2
   ```

   根据您的工作站，此命令可能需要 sudo，或者您也可以使用 Ruby 环境管理器 (如 RVM)。此过程已使用 0.8.0 版的 kitchen-ec2 驱动程序进行测试，但已推出更新版本。要安装某个特定版本，请运行 gem install kitchen-ec2 -v <*version number*>。

3. 您必须指定 Test Kitchen 可用于连接至相应实例的 Amazon EC2 SSH 密钥对。如果没有 密钥对，请参阅 Amazon EC2 密钥对，以获取有关如何创建密钥对的信息。请注意，该密钥对必须属于与实例相同的 Amazon Web Services Region。示例使用美国西部（北加利福尼亚）。

   选定密钥对后，创建 opsworks_cookbooks 的子目录 (名为 ec2_keys)，并将该密钥对的私有密钥 (.pem) 文件复制到该子目录中。请注意，将私有密钥置于 ec2_keys 中只是略微简化代码的一个便捷方法而已；它可以是您系统上的任何位置。

4. 创建 opsworks_cookbooks 的名为 createdir-ec2 的子目录并导航到该子目录。

5. 将包含以下内容的 metadata.rb 文件添加到 createdir-ec2。

   ```
   name "createdir-ec2"
   version "0.1.0"
   ```

6. 按照示例 1：安装软件包中所述初始化 Test Kitchen。以下部分介绍如何配置 .kitchen.yml，这对于 Amazon EC2 实例来说要复杂得多。

7. 将 recipes 子目录添加 createdir-ec2 中。

针对 Amazon EC2 配置 .kitchen.yml

您使用 kitchen-ec2 驱动程序启动适当配置的 Amazon EC2 实例所需的信息来配置 .kitchen.yml。以下是面向美国西部（北加利福尼亚）区域中 Amazon Linux 实例的 .kitchen.yml 文件示例。

```
driver:
  name: ec2
  aws_ssh_key_id: US-East1
  region: us-west-1
```

```
  availability_zone: us-west-1c
  require_chef_omnibus: true
  security_group_ids: sg........
  subnet_id: subnet-.........
  associate_public_ip: true
  interface: dns

provisioner:
  name: chef_solo

platforms:
  -name: amazon
  driver:
    image_id: ami-xxxxxxxx
  transport:
    username: ec2-user
    ssh_key: ../ec2_keys/US-East1.pem

suites:
  - name: default
    run_list:
      - recipe[createdir-ec2::default]
    attributes:
```

您可以使用 `provisioner` 和 `suites` 部分的默认设置，但必须修改默认的 `driver` 和 `platforms` 设置。此示例使用了最少的设置列表，并接受了其余部分的默认值。有关 `kitchen-ec2` 设置的完整列表，请参阅 Kitchen::Ec2：面向 Amazon EC2 的 Test Kitchen 驱动程序。

此示例设置了以下 `driver` 属性。它假设您已将用户的访问密钥和秘密密钥分配给标准环境变量 (如前所述)。默认情况下，该驱动程序将使用这些密钥。否则，您必须通过将 `aws_access_key_id` 和 `aws_secret_access_key` 添加至 `driver` 属性中来显式指定密钥，并设置为适当的密钥值。

name

(必选) 必须将此属性设置为 `ec2`。

aws_ssh_key_id

(必选) Amazon EC2 SSH 密钥对名称，在该示例中名为 `US-East1`。

transport.ssh_key

(必选) 您为 `aws_ssh_key_id` 指定的密钥的私有密钥 (`.pem`) 文件。对于本示例，该文件名为 `US-East1.pem`，且位于 `../opsworks/ec2_keys` 目录中。

region

(必选) 实例的 Amazon Web Services Region。示例使用美国西部（北加利福尼亚），用 `us-west-1`）表示。

availability_zone

(可选) 实例的可用区。如果您忽略了此设置，则 Test Kitchen 将使用指定区域的默认可用区，为美国西部（北加利福尼亚）的 `us-west-1b`。但是，默认区域可能不适用于您的账户。在这种情况下，您必须显式指定一个可用区。如果发生这种情况，用于准备相应示例的账户不支持 `us-west-1b`，因此示例会显式指定 `us-west-1c`。

require_chef_omnibus

如果设置为 `true`，此设置可确保使用 omnibus 安装程序将 `chef-client` 安装至所有平台实例。

security_group_ids

(可选) 应用于实例的安全组 ID 的列表。此设置会将 `default` 安全组应用于实例。确保安全组入口规则允许入站 SSH 连接，否则 Test Kitchen 无法与实例进行通信。如果您使用 `default` 安全组，则可能需要对其进行相应编辑。有关更多信息，请参阅 Amazon EC2 Security Groups。

subnet_id

实例的目标子网的 ID (如果适用)。

associate_public_ip

如果您希望能够通过 Internet 访问实例，则可以利用 Amazon EC2 将公有 IP 地址与该实例相关联。

接口

您用于访问实例的主机名配置类型。有效值为 `dns`、`public`、`private` 或 `private_dns`。如果您不指定此属性的值，则 `kitchen-ec2` 会按如下顺序设置主机名配置。如果您忽略此属性，则不会设置配置类型。

1. DNS 名称
2. 公有 IP 地址
3. 私有 IP 地址
4. 私有 DNS 名称

> **Important**
>
> 您不用将您的账户凭证用于访问密钥和秘密密钥，而应该创建一个用户，并向 Test Kitchen 提供这些凭证。有关更多信息，请参阅管理 AWS 访问密钥的最佳实践。
> 注意不要.kitchen.yml放在可公开访问的位置，例如将其上传到公共存储库 GitHub 或 Bitbucket 存储库。这样做会暴露您的凭证，并可能危及您的账户安全。

kitchen-ec2 驱动程序为以下平台提供默认支持：

- ubuntu-10.04
- ubuntu-12.04
- ubuntu-12.10
- ubuntu-13.04
- ubuntu-13.10
- ubuntu-14.04
- centos-6.4
- debian-7.1.0
- windows-2012r2
- windows-2008r2

如果您要使用这些平台中的一个或多个，请将相应的平台名称添加至 platforms。kitchen-ec2 驱动程序将自动选择适当的 AMI，并生成一个 SSH 用户名。您可以使用其他平台 (此示例使用 Amazon Linux)，但是必须显式指定如下 platforms 属性。

name

平台名称。此示例使用 Amazon Linux，因此 name 设置为 amazon。

driver

driver 属性，它包括以下内容：

- image_id-平台的 AMI，必须属于指定区域。该示例使用 ami-ed8e9284，这是来自美国西部（北加利福尼亚）区域的 Amazon Linux AMI。
- transport.username- Test Kitchen 将用于与实例进行通信的 SSH 用户名。

  将 ec2-user 用于 Amazon Linux。其他 AMI 可能具有不同的用户名。

将 `.kitchen.yml` 中的代码替换为示例，并将相应值分配给特定于账户的属性，例如 `aws_access_key_id`。

运行配方

此示例使用来自迭代的配方。

运行配方

1. 使用以下代码创建名为 `default.rb` 的文件，并将其保存至说明书的 `recipes` 文件夹。

```
directory "/srv/www/shared" do
  mode 0755
  owner 'root'
  group 'root'
  recursive true
  action :create
end
```

2. 运行 `kitchen converge` 以执行该配方。请注意，由于启动和初始化 Amazon EC2 实例需要时间，因此与以前的示例相比，该命令需要更长的时间才能完成。
3. 转到 Amazon EC2 控制台，选择美国西部（北加利福尼亚）区域，并在导航窗格中单击 实例。您将会在列表中看到新创建的实例。
4. 运行`kitchen login`以登录实例，就像您对正在运行的实例所做的那样 VirtualBox。您将在 `/srv` 下看到新创建的目录。也可以使用您常用的 SSH 客户端连接到实例。

后续步骤

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

本章演练了如何实施 Chef 方法的基础知识，但还提供了更多信息：

- 这些示例向您介绍了如何使用一些更加常用的资源，但还有更多资源。

对于已介绍的资源，这些示例仅使用了一些可用属性和操作。有关完整的参考，请参阅关于资源和提供程序。

- 这些示例仅使用核心说明书元素：`recipes`、`attributes`、`files` 和 `templates`。

  说明书还可以包含其他各种元素，例如 `libraries`、`definitions` 和 `specs`。有关更多信息，请参阅 Chef 文档。
- 这些示例仅将 Test Kitchen 用作启动实例、运行配方和登录实例的一种便捷方法。

  Test Kitchen 主要是一个测试平台，您可以用它对您的配方运行各种测试。如果您尚未执行此操作，请完成 Test Kitchen 演练的其余部分，它会向您介绍其测试功能。
- 为 AWS OpsWorks 堆栈实现食谱提供了一些更高级的示例，并演示了如何为 AWS OpsWorks Stacks 实现食谱。

为 AWS OpsWorks 堆栈实现食谱

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

说明书基础知识为您介绍了各说明书和配方。该部分中的示例在设计上很简单，适用于任何支持 Chef 的实例，包括 AWS OpsWorks Stacks 实例。要为 AWS OpsWorks Stacks 实现更复杂的食谱，你通常需要充分利用 AWS OpsWorks Stacks 环境，它在很多方面与标准 Chef 不同。

本主题介绍为 AWS OpsWorks 堆栈实例实现配方的基础知识。

> **Note**
>
> 如果您不熟悉如何实施说明书，应从说明书基础知识入手。

主题

- 在 AWS OpsWorks Stacks Linux 实例上运行配方
- 在 Windows 实例上运行配方
- 运行 Windows PowerShell 脚本

- 在 Vagrant 上模拟堆栈配置和部署属性
- 使用堆栈配置和部署属性值
- 在 Linux 实例上使用外部说明书：Berkshelf
- 使用 SDK for Ruby：从 Amazon S3 下载文件
- 安装 Windows 软件
- 覆盖内置属性
- 覆盖内置模板

在 AWS OpsWorks Stacks Linux 实例上运行配方

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS：Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

Test Kitchen and Vagrant 提供了一种简单而有效的方法来实现食谱，但是要验证食谱的食谱能否在生产中正常运行，你必须在 Stacks 实例 AWS OpsWorks 上运行它们。本主题介绍如何在 AWS OpsWorks Stacks Linux 实例上安装自定义说明书并运行简单的配方。本主题还提供了一些高效修复配方错误的提示。

有关如何在 Windows 实例上运行配方的说明，请参阅在 Windows 实例上运行配方。

主题

- 创建和运行配方
- 自动执行配方
- 排查并修复配方问题

创建和运行配方

首先，您需要创建一个堆栈。下面简要总结了如何为本示例创建堆栈。有关更多信息，请参阅 创建新堆栈。

创建 堆栈

1. 打开 AWS OpsWorks Stacks 控制台，然后单击 Add Stack (添加堆栈)。

2. 指定以下设置，接受其他设置的默认值，然后单击 Add Stack。

   - 姓名 — OpsTest
   - 默认 SSH 密钥 - Amazon EC2 密钥对

   如果您需要创建 Amazon EC2 密钥对，请参阅 Amazon EC2 密钥对。请注意，该密钥对必须属于与实例相同的 Amazon Web Services Region。本示例使用了默认的美国西部（俄勒冈州）区域。
3. 单击 Add a layer 并将采用以下设置的自定义层添加到堆栈。

   - 姓名 — OpsTest
   - 短名称 - opstest

   实际上任何层类型均适用于 Linux 堆栈，但本示例不需要由其他层类型安装的任何程序包，因此自定义层是最简单的方法。
4. 向层添加全天候实例 (采用默认设置) 并启动该实例。

当实例正在启动时(通常需要几分钟)，您可以创建说明书。本示例将使用的配方是在条件逻辑中的配方的基础上稍作修改得来的，该配方将创建一个名称取决于平台的数据目录。

设置说明书

1. 在 opsworks_cookbooks 中创建一个名为 opstest 的目录并导航到该目录。
2. 创建一个包含以下内容的 metadata.rb 文件，并将该文件保存到 opstest。

```
name "opstest"
version "0.1.0"
```

3. 在 recipes 中创建 opstest 目录。
4. 创建包含以下配方的 default.rb 文件，并将该文件保存到 recipes 目录。

```
Chef::Log.info("******Creating a data directory.******")

data_dir = value_for_platform(
  "centos" => { "default" => "/srv/www/shared" },
  "ubuntu" => { "default" => "/srv/www/data" },
  "default" => "/srv/www/config"
```

```
)

directory data_dir do
  mode 0755
  owner 'root'
  group 'root'
  recursive true
  action :create
end
```

请注意，配方会记录消息，但它是通过调用 `Chef::Log.info` 来实现的。在这个例子中，你没有使用 Test Kitchen，所以这个`log`方法不是很有用。 `Chef::Log.info`将消息放入 Chef 日志，你可以在 Chef 运行完成后阅读该日志。 AWS OpsWorks 如后面所述，Stacks 提供了一种查看这些日志的简便方法。

> **Note**
>
> Chef 日志通常包含大量例程和相对无关的信息。“*”字符会将消息文本括起来，使其更容易被发现。

5. 创建 `opsworks_cookbooks` 的 `.zip` 存档。要在 AWS OpsWorks Stacks 实例上安装您的食谱，您必须将其存储在存储库中，并向 AWS OpsWorks Stacks 提供将食谱下载到该实例所需的信息。您可以将说明书存储在任一受支持的存储库类型中。本实例会将包含说明书的存档文件存储在 Amazon S3 存储桶中。有关说明书存储库的更多信息，请参阅说明书存储库。

   > **Note**
   >
   > 为简单起见，本示例仅存档整个 `opsworks_cookbooks` 目录。但是，这意味着 AWS OpsWorks Stacks 会将所有食谱下载`opsworks_cookbooks`到实例中，尽管你只会使用其中一本。要仅安装示例说明书，请创建另一个父目录并将 `opstest` 移动到该目录。然后创建该父目录的 `.zip` 存档，并用它代替 `opsworks_cookbooks.zip`。
   > 发送到 Amazon S3 存储桶的内容可能包含客户内容。有关删除敏感数据的更多信息，请参阅如何清空 S3 存储桶？或如何删除 S3 存储桶？。

6. 将存档上传到 Amazon S3 存储桶，公开存档，然后记录存档的 URL。

您现在可以安装说明书并运行配方。

运行配方

1. 编辑堆栈以启用自定义说明书，然后指定以下设置。

   - 存储库类型 - S3 存档
   - 存储库 URL -您之前记录的说明书存档 URL

   对其他设置使用默认值，然后单击 Save 更新堆栈配置。
2. 运行“Update Custom Cookbooks”堆栈命令，这会将当前版本的自定义说明书安装到堆栈的实例上。如果您的说明书存在早期版本，此命令会覆盖该版本。
3. 通过在 Recipes to execute 设置为 **`opstest::default`** 的情况下运行 Execute Recipes 堆栈命令来执行配方。此命令将使用一个包含 `opstest::default` 的运行列表来启动 Chef 运行。

配方成功运行后，您可以对其进行验证。

验证 opstest

1. 第一步是检查 Chef 日志。单击 opstest1 实例 Log 列中的 show，显示日志。向下滚动，您将在接近底部的位置看到您的日志。

```
...
[2014-07-31T17:01:45+00:00] INFO: Storing updated cookbooks/opsworks_cleanup/
attributes/customize.rb in the cache.
[2014-07-31T17:01:45+00:00] INFO: Storing updated cookbooks/opsworks_cleanup/
metadata.rb in the cache.
[2014-07-31T17:01:46+00:00] INFO: ******Creating a data directory.******
[2014-07-31T17:01:46+00:00] INFO: Processing template[/etc/hosts] action create
 (opsworks_stack_state_sync::hosts line 3)
...
```

2. 使用 SSH 登录实例并列出 `/srv/www/` 的内容。

如果您执行了所有步骤，您将看到 `/srv/www/config` 而不是您需要的 `/srv/www/shared` 目录。以下部分提供了一些快速修复此类错误的指南。

自动执行配方

Execute Recipes 命令是一种测试自定义配方的便捷方式，这正是大多数此类示例中都使用它的原因。但是，实际上，您通常会在实例生命周期的标准时刻运行配方，例如在实例完成启动后或部署应用程序时。 AWS OpsWorks Stacks 支持每个层的一组生命周期事件：设置、配置、部署、取消部署和关闭，从而简化实例上的运行配方。通过将配方分配给相应的生命周期事件，可以让 AWS OpsWorks Stacks 在图层的实例上自动运行配方。

您通常会在实例完成启动后立即创建目录，与设置事件对应。下面展示了如何使用之前在示例中创建的同一堆栈在设置时运行示例配方。您可以对其他事件使用相同的过程。

在设置时自动运行配方

1. 在导航窗格中选择 “图层”，然后选择图 OpsTest 层的 “食谱” 链接旁边的铅笔图标。
2. 将 **`opstest::default`** 添加到层的 Setup (设置) 配方，单击 + 将前者添加到层，然后选择 Save (保存) 保存配置。
3. 选择 Instances，将另一个实例添加到层，然后启动该实例。

   该实例应名为 `opstest2`。启动完成后， AWS OpsWorks Stacks 将运行。`opstest::default`
4. 在 `opstest2` 实例处于在线状态后，请验证 `/srv/www/shared` 是否存在。

> **Note**
>
> 如果您已将配方分配给设置、配置或部署事件，您也可以使用堆栈命令 (设置和配置) 或部署命令 (部署) 来触发事件，手动运行。请注意，如果您向某个事件分配了多个配方，那么这些命令将运行所有这些配方。

排查并修复配方问题

如果您没有得到预期结果，或者您的配方甚至都没有成功运行，那么问题排查通常从检查 Chef 日志开始。它包含运行的详细说明，并包含来自您的配方的任何内联日志消息。如果您的配方只是失败了，日志就会特别有用。当出现这种情况时，Chef 会记录错误 (包括堆栈跟踪)。

如果配方像在本示例中一样成功了，则 Chef 日志通常不会有太大帮助。在这种情况下，您只需更仔细地观察配方即可找到问题所在，特别是前几行：

```
Chef::Log.info("******Creating a data directory.******")
```

```
data_dir = value_for_platform(
  "centos" => { "default" => "/srv/www/shared" },
  "ubuntu" => { "default" => "/srv/www/data" },
  "default" => "/srv/www/config"
)
...
```

当您在 Vagrant 上测试配方时，CentOS 是 Amazon Linux 的合理替代品，但现在您正在实际的 Amazon Linux 实例上运行。Amazon Linux 的平台值为 `amazon`，该值未包含在 `value_for_platform` 调用中，因此配方将默认创建 `/srv/www/config`。有关问题排查的更多信息，请参阅调试和故障排除指南。

既然您已经确定问题，您需要更新配方并验证修复。您可以返回到原始的源文件，更新 `default.rb`，将新存档上传到 Amazon S3 或执行其他操作。但是，这个过程可能有点枯燥和费时。下面所介绍的这种方法在速度上要快得多，这种方法对类似于下例中错误的简单配方错误特别有用：在实例上编辑配方。

在实例上编辑配方

1. 使用 SSH 登录实例，然后运行 `sudo su` 以提升您的权限。您需要根权限才能访问说明书目录。
2. AWS OpsWorks Stacks 会将你的食谱存储在里面`/opt/aws/opsworks/current/site-cookbooks`，所以请导航到。`/opt/aws/opsworks/current/site-cookbooks/opstest/recipes`

   > **Note**
   >
   > AWS OpsWorks Stacks 还会将你的食谱副本存储在里面。`/opt/aws/opsworks/current/merged-cookbooks`请勿编辑该说明书。当你执行食谱时， AWS OpsWorks Stacks 会将食谱从复制`.../site-cookbooks`到`.../merged-cookbooks`，因此你所做的任何更改都`.../merged-cookbooks`将被覆盖。

3. 在实例上使用文本编辑器编辑 `default.rb`，并将 `centos` 替换为 `amazon`。现在，您的配方应该类似于以下形式。

   ```
   Chef::Log.info("******Creating a data directory.******")

   data_dir = value_for_platform(
     "amazon" => { "default" => "/srv/www/shared" },
   ```

```
  "ubuntu" => { "default" => "/srv/www/data" },
  "default" => "/srv/www/config"
)
...
```

要验证修复情况，请再次运行 Execute Recipe 堆栈命令来执行配方。现在，实例应该具有 `/srv/www/shared` 目录。如果您需要进一步更改配方，则可以按所需的频率运行 Execute Recipe；您不需要在每次运行命令时停止并重新启动实例。当您对配方正常工作感到满意后，请勿忘记更新您的源说明书中的代码。

> **Note**
>
> 如果您已将配方分配给生命周期事件，因此 AWS OpsWorks Stacks 会自动运行该食谱，则可以随时使用 Execute Recipe 来重新运行配方。您还可以使用 AWS OpsWorks Stacks 控制台手动触发相应的事件，在不重启实例的情况下根据需要多次重新运行配方。但是，这种方法会运行事件的所有配方。提醒一下：
>
> - 使用堆栈命令可触发设置事件或配置事件。
> - 使用部署命令可触发部署事件或取消部署事件。

在 Windows 实例上运行配方

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

本主题基本上是在 Linux 实例上运行配方的简化版本，旨在向您展示如何在 Windows 堆栈上运行配方。我们建议您先浏览在 Linux 实例上运行配方，当中信息更为详细，且大部分内容都与任一类型的操作系统有关。

有关如何在 AWS OpsWorks Stacks Linux 实例上运行配方的说明，请参阅在 Linux 实例上运行配方。

主题

- 启用 RDP 访问

- 创建和运行配方
- 自动执行配方

启用 RDP 访问

在开始之前，如果您尚未执行此操作，则必须设置一个安全组，该安全组包含一条允许对您的实例进行 RDP 访问的入站规则。您在创建堆栈时将需要该组。

当您在某个区域中创建第一个堆栈时， AWS OpsWorks Stacks 会创建一组安全组。它们包括一个名为类似的名字`AWS-OpsWorks-RDP-Server`， AWS OpsWorks 堆栈将其连接到所有 Windows 实例以允许 RDP 访问。但是，默认情况下，此安全组没有任何规则，因此您必须添加入站规则以允许 RDP 访问您的实例。

允许 RDP 访问

1. 打开 Amazon S3 控制台，将其设置为堆栈的区域，然后从导航窗格中选择安全组。
2. 选择 AWS OpsWorks-RDP-Server，选择入站选项卡，然后选择编辑。
3. 使用以下设置添加一条规则：
   - 类型 - RDP
   - 源 -允许的源 IP 地址。

     通常您会允许来自您的 IP 地址或指定 IP 地址范围（一般是公司的 IP 地址范围）的入站 RDP 请求。

> **Note**
>
> 如下文所述，您还必须编辑用户权限以便为普通用户授予 RDP 访问权限。

有关更多信息，请参阅 使用 RDP 登录。

创建和运行配方

下面简要总结了如何为本示例创建堆栈。有关更多信息，请参阅 创建新堆栈。

创建堆栈

1. 打开 AWS OpsWorks Stacks 控制台，然后选择 Add Stack (添加堆栈)。指定以下设置，接受其他设置的默认值，然后选择 Add Stack。

   - 姓名 — WindowsRecipeTest
   - 区域-美国西部（俄勒冈州）

     本示例在任何区域都有效，但我们建议将美国西部（俄勒冈州）用于教程。
   - 默认操作系统 - Microsoft Windows Server 2012 R2
2. 选择 Add a layer 并将采用以下设置的自定义层添加到堆栈。

   - 姓名 — RecipeTest
   - 短名称 - recipetest
3. 向 RecipeTest图层@@ 添加具有默认设置的全天候实例并启动它。

   AWS OpsWorks 堆栈会自动分配`AWS-OpsWorks-RDP-Server`给此实例，从而允许授权用户登录该实例。
4. 选择 Permissions、Edit，然后选择 SSH/RDP 和 sudo/admin。除了 `AWS-OpsWorks-RDP-Server` 安全组之外，普通用户还需要此授权才能登录实例。

   > **Note**
   >
   > 您也可以以管理员身份登录，但需要采用另一个登录过程。有关更多信息，请参阅 使用 RDP 登录。

当实例正在启动时(通常需要几分钟)，您可以创建说明书。本示例的配方将创建一个数据目录，该配方基本上来自于示例 3：创建目录，只是针对 Windows 进行了修改。

> **Note**
>
> 在为 AWS OpsWorks Stacks Windows 实例实现食谱时，你使用的目录结构与为 Stacks Lin AWS OpsWorks ux 实例实现食谱时使用的目录结构略有不同。有关更多信息，请参阅 说明书存储库。

设置说明书

1. 创建一个名为 windowstest 的目录并导航到该目录。
2. 创建一个包含以下内容的 metadata.rb 文件，并将该文件保存到 windowstest。

```
name "windowstest"
version "0.1.0"
```

3. 在 recipes 中创建 windowstest 目录。
4. 创建包含以下配方的 default.rb 文件，并将该文件保存到 recipes 目录。

```
Chef::Log.info("******Creating a data directory.******")

directory 'C:\data' do
  rights :full_control, 'instance_name\username'
  inherits false
  action :create
end
```

将 *username* 替换为您的用户名。

5. 将说明书放在存储库中。

要在 AWS OpsWorks Stacks 实例上安装您的食谱，您必须将其存储在存储库中，并向 AWS OpsWorks Stacks 提供将食谱下载到该实例所需的信息。您可以将 Windows 说明书作为存档文件存储在 S3 存储桶或 Git 存储库中。本示例使用了 S3 存储桶，因此您必须创建 windowstest 目录的 .zip 存档。有关说明书存储库的更多信息，请参阅说明书存储库。

6. 将存档上传到 S3 存储桶，公开存档，然后记录存档的 URL。您还可以使用私有存档，但公有存档对本示例已经足够了，并且使用起来更容易一点。

发送到 Amazon S3 存储桶的内容可能包含客户内容。有关删除敏感数据的更多信息，请参阅如何清空 S3 存储桶？或如何删除 S3 存储桶？。

您现在可以安装说明书并运行配方。

运行配方

1. 编辑堆栈以启用自定义说明书，然后指定以下设置。

- 存储库类型 - S3 存档
- 存储库 URL -您之前记录的说明书存档 URL

接受其他设置的默认值，然后选择 Save 更新堆栈配置。

2. 运行“Update Custom Cookbooks”堆栈命令，这会将当前版本的自定义说明书安装到堆栈的实例 (包括联机实例) 上。如果您的说明书存在早期版本，此命令会覆盖该版本。
3. 完成“Update Custom Cookbooks”命令后，通过在 Recipes to execute 设置为 **`windowstest::default`** 的情况下运行 Execute Recipes 堆栈命令执行配方。此命令将启动 Chef 运行并显示一个包含配方的运行列表。

配方成功运行后，您可以对其进行验证。

验证 windowstest

1. 检查 Chef 日志。选择 opstest1 实例 Log 列中的 show，显示日志。向下滚动，您将在接近底部的位置看到您的日志。

```
...
[2014-07-31T17:01:45+00:00] INFO: Storing updated cookbooks/opsworks_cleanup/
attributes/customize.rb in the cache.
[2014-07-31T17:01:45+00:00] INFO: Storing updated cookbooks/opsworks_cleanup/
metadata.rb in the cache.
[2014-07-31T17:01:46+00:00] INFO: ******Creating a data directory.******
[2014-07-31T17:01:46+00:00] INFO: Processing template[/etc/hosts] action create
 (opsworks_stack_state_sync::hosts line 3)
...
```

2. 选择 Instances，选择实例的 Actions 列中的 rdp，然后请求一个具有合适到期时间的 RDP 密码。复制 DNS 名称、用户名和密码。您随后可以将该信息用于 RDP 客户端 (如 Windows 远程桌面连接客户端) 以登录实例并验证 `c:\data` 是否存在。有关更多信息，请参阅 使用 RDP 登录。

> **Note**
>
> 如果您的配方无法正常运行，请参阅排查并修复配方问题以获取故障排除提示；其中大多数提示也适用于 Windows 实例。如果你想通过编辑实例上的配方来测试你的修复程序，请在`C:\chef\cookbooks`目录中查找你的食谱， AWS OpsWorks Stacks 会安装自定义食谱。

自动执行配方

Execute Recipes 命令是一种测试自定义配方的便捷方式，这正是大多数此类示例中都使用它的原因。但是，实际上，您通常会在实例生命周期的标准时刻运行配方，例如在实例完成启动后或部署应用程序时。 AWS OpsWorks Stacks 支持每个层的一组生命周期事件：设置、配置、部署、取消部署和关闭，从而简化实例上的运行配方。通过将配方分配给相应的生命周期事件，可以让 AWS OpsWorks Stacks 在图层的实例上自动运行配方。

您通常会在实例完成启动后立即创建目录，与设置事件对应。下面展示了如何使用之前在示例中创建的同一堆栈在设置时运行示例配方。您可以对其他事件使用相同的过程。

在设置时自动运行配方

1. 在导航窗格中选择 “图层”，然后选择图 RecipeTest 层的 “食谱” 链接旁边的铅笔图标。
2. 将 **`windowstest::default`** 添加到层的 Setup (设置) 配方，选择 + 将前者添加到层，然后选择 Save (保存) 保存配置。
3. 选择 Instances，将另一个实例添加到层，然后启动该实例。

   该实例应名为 `recipetest2`。启动完成后， AWS OpsWorks Stacks 将运行。`windowstest::default`
4. 在 `recipetest2` 实例处于在线状态后，请验证 `c:\data` 是否存在。

> **Note**
>
> 如果您已将配方分配给设置、配置或部署事件，您也可以使用堆栈命令 (设置和配置) 或部署命令 (部署) 来触发事件，手动运行。请注意，如果您向某个事件分配了多个配方，那么这些命令将运行所有这些配方。

运行 Windows PowerShell 脚本

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 这些示例假定您已完成在 Windows 实例上运行配方示例。如果没有，您应该先完成该示例。具体而言，它介绍了如何启用对实例的 RDP 访问。

让配方在 Windows 实例上执行任务（尤其是没有相应 Chef 资源的任务）的一种方法是让配方运行 Windows 脚本。 PowerShell 本节介绍如何使用 Windows PowerShell 脚本安装 Windows 功能，从而向您介绍基础知识。

该`powershell_script`资源在实例上运行 Windows PowerShell cmdlet。以下示例使用 In stal WindowsFeature l-cmdlet 在实例上安装 XPS 查看器。

下面简要总结了如何为本示例创建堆栈。有关更多信息，请参阅 创建新堆栈。

创建堆栈

1. 打开 AWS OpsWorks Stacks 控制台，然后选择 Add Stack (添加堆栈)。指定以下设置，接受其他设置的默认值，然后单击 Add Stack。
   - 姓名 — PowerShellTest
   - 区域-美国西部（俄勒冈州）

     本示例在任何区域都有效，但我们建议将美国西部（俄勒冈州）用于教程。
   - 默认操作系统 - Microsoft Windows Server 2012 R2
2. 选择 Add a layer 并将采用以下设置的自定义层添加到堆栈。
   - 姓名 — PowerShell
   - 短名称 - powershell
3. 向 PowerShell 图层@@ 添加一个具有默认设置的全天候实例，然后启动它。

4. 选择 Permissions、Edit，然后选择 SSH/RDP 和 sudo/admin。除了 AWS-OpsWorks-RDP-Server 安全组之外，您还需要此授权才能以普通用户身份登录实例。

当实例正在启动时(通常需要几分钟)，您可以创建说明书。本示例的配方将创建一个数据目录，该配方基本上来自于示例 3：创建目录，只是针对 Windows 进行了修改。

设置说明书

1. 创建一个名为 powershell 的目录并导航到该目录。
2. 创建一个包含以下内容的 metadata.rb 文件，并将该文件保存到 windowstest。

```
name "powershell"
version "0.1.0"
```

3. 在 recipes 目录中创建一个 powershell 目录。
4. 创建包含以下配方的 default.rb 文件，并将该文件保存到 recipes 目录。

```
Chef::Log.info("******Installing XPS.******")

powershell_script "Install XPS Viewer" do
  code <<-EOH
    Install-WindowsFeature XPS-Viewer
  EOH
  guard_interpreter :powershell_script
  not_if "(Get-WindowsFeature -Name XPS-Viewer).installed"
end
```

   - powershell_script 资源将运行 cmdlet 来安装 XPS 查看器。

     本示例仅运行一个 cmdlet，但 code 块可以包含任意数量的命令行。

   - 该guard_interpreter属性指示 Chef 使用 64 位版本的 Windows PowerShell。
   - not_if guard 属性可确保 Chef 不会在已安装该功能的情况下再次安装该功能。

5. 创建 powershell 目录的 .zip 存档。
6. 将存档上传到 Amazon S3 存储桶，公开存档，然后记录存档的 URL。您还可以使用私有存档，但公有存档对本示例已经足够了，并且使用起来更容易一点。

发送到 Amazon S3 存储桶的内容可能包含客户内容。有关删除敏感数据的更多信息，请参阅如何清空 S3 存储桶？或如何删除 S3 存储桶？。

您现在可以安装说明书并运行配方。

运行配方

1. 编辑堆栈以启用自定义说明书，然后指定以下设置。

   - 存储库类型 - S3 存档
   - 存储库 URL -您之前记录的说明书存档 URL

   接受其他设置的默认值，然后选择 Save 更新堆栈配置。
2. 运行 Update Custom Cookbooks 堆栈命令，这会将当前版本的自定义说明书安装到实例上。
3. 完成 Update Custom Cookbooks 命令后，通过在 Recipes to execute 设置为 **`powershell::default`** 的情况下运行 Execute Recipes 堆栈命令来执行配方。

> **Note**
>
> 为了方便起见，此示例使用了 Execute Recipes，但通常会 AWS OpsWorks 让 Stacks 通过将配方分配给相应的生命周期事件来自动运行配方。您可以通过手动触发事件来运行此类配方。您可以使用堆栈命令触发设置和配置事件，使用部署命令触发部署和取消部署事件。

配方成功运行后，您可以对其进行验证。

验证 powershell 配方

1. 检查 Chef 日志。单击 powershell1 实例 Log 列中的 show，显示日志。向下滚动，您将在接近底部的位置看到您的日志。

```
...
[2015-04-27T18:12:09+00:00] INFO: Storing updated cookbooks/powershell/metadata.rb
 in the cache.
[2015-04-27T18:12:09+00:00] INFO: ******Installing XPS.******
```

```
[2015-04-27T18:12:09+00:00] INFO: Processing powershell_script[Install XPS Viewer]
 action run (powershell::default line 3)
[2015-04-27T18:12:09+00:00] INFO: Processing powershell_script[Guard resource]
 action run (dynamically defined)
[2015-04-27T18:12:42+00:00] INFO: powershell_script[Install XPS Viewer] ran
 successfully
...
```

2. 使用 RDP 登录实例并打开 Start 菜单。XPS 查看器应与 Windows Accessories 一起列出。

在 Vagrant 上模拟堆栈配置和部署属性

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 本主题仅适用于 Linux 实例。Test Kitchen 尚不支持 Windows，因此你将在 AWS OpsWorks Stacks 实例上运行所有 Windows 示例。

AWS OpsWorks 对于每个生命周期事件，Stacks 会将堆栈配置和部署属性添加到堆栈中每个实例的节点对象。这些属性可提供堆栈配置 (包括每个层及其联机实例的配置、每个已部署的应用程序的配置等) 的快照。由于这些属性位于节点对象中，因此任何配方都可以访问它们； AWS OpsWorks Stacks 实例的大多数配方都使用其中一个或多个属性。

在Vagrant盒子中运行的实例不是由 AWS OpsWorks Stacks管理的，因此默认情况下，其节点对象不包含任何堆栈配置和部署属性。但是，您可以向 Test Kitchen 环境添加一组合适的属性。然后，Test Kitchen 将属性添加到实例的节点对象中，您的配方可以像在 AWS OpsWorks Stacks 实例上一样访问这些属性。

本主题介绍如何获取合适的堆栈配置和部署属性的一个副本，在实例上设置这些属性，然后访问它们。

> **Note**
>
> 如果您使用 Test Kitchen 来对您的配方运行测试，fauxhai 将提供一种替代方法来模拟堆栈配置和部署 JSON。

设置说明书

1. 创建 opsworks_cookbooks 的名为 printjson 的子目录并导航到该子目录。
2. 按照示例 1：安装软件包中所述初始化并配置 Test Kitchen。
3. 将两个子目录添加到 printjson：recipes 和 environments。

您可以通过向具有合适的定义的说明书添加属性文件来模拟堆栈配置和部署属性，但更好的方法是使用 Test Kitchen 环境。有两种基本方法：

- 将属性定义添加到 .kitchen.yml。

  如果您只有几个属性，那么这种方法最有用。有关更多信息，请参阅 kitchen.yml。
- 在一个环境文件中定义属性并在 .kitchen.yml 中引用该文件。

  此方法通常更适合堆栈配置和部署属性，因为环境文件已经采用 JSON 格式。你可以从合适的 AWS OpsWorks Stacks 实例中获取 JSON 格式的属性的副本，然后将其粘贴进去即可。所有示例都使用一个环境文件。

为您的说明书创建堆栈配置和部署属性的最简单的方法是创建一个已适当配置的堆栈并从作为 JSON 的实例复制生成的属性。要保持您的 Test Kitchen 环境文件的可管理性，您可以随后编辑该 JSON，使其仅具有您的配方需要的属性。本章中的示例基于Chef 11 Linux 堆栈入门中的堆栈，该堆栈是一个简单的 PHP 应用程序服务器堆栈，包含一个负载均衡器、多个 PHP 应用程序服务器和一台 MySQL 数据库服务器。

创建堆栈配置和部署 JSON

1. 按照 MyStack 中所述进行创建Chef 11 Linux 堆栈入门，包括部署 SimplephPapp。如果您愿意，可以省略 第 4 步：横向扩展 MyStack 中需要的第二个 PHP App Server 实例；这些示例不会使用这些属性。
2. 如果您尚未执行此操作，请启动 php-app1 实例，然后使用 SSH 登录。
3. 在终端窗口中，运行以下代理 cli 命令：

```
sudo opsworks-agent-cli get_json
```

此命令会将实例的最新堆栈配置和部署属性以 JSON 格式输出到终端窗口。

4. 将 JSON 复制到 `.json` 文件并将其保存到您工作站中方便的位置。具体细节取决于您的 SSH 客户端。例如，如果您在 Windows 上使用了 PuTTY，则可以运行 `Copy All to Clipboard` 命令，这会将终端窗口中的所有文本复制到 Windows 剪贴板。之后，您可以将内容粘贴到 `.json` 文件并编辑该文件以删除无关的文本。
5. 根据需要编辑 MyStack JSON。堆栈配置和部署属性有很多，说明书通常仅使用其中的一小部分。为了保持您的环境文件的可管理性，您可以对 JSON 进行编辑，使其保留原来的结构但仅包含您的说明书实际使用的属性。

   此示例使用经过大量编辑的 MyStack JSON 版本，其中仅包含两个`['opsworks']['stack']`属性，`['id]`和`['name']`。创建 MyStack JSON 的编辑版本，如下所示：

```
{
  "opsworks": {
    "stack": {
      "name": "MyStack",
      "id": "42dfd151-6766-4f1c-9940-ba79e5220b58",
    },
  },
}
```

要让此 JSON 进入实例的节点对象，您需要将其添加到 Test Kitchen 环境。

将堆栈配置和部署属性添加到 Test Kitchen 环境

1. 创建一个包含以下内容的名为 `test.json` 的环境文件并将该文件保存到说明书的 `environments` 文件夹。

```
{
  "default_attributes": {
    "opsworks" : {
      "stack" : {
        "name" : "MyStack",
        "id" : "42dfd151-6766-4f1c-9940-ba79e5220b58"
```

```
      }
    }
  },
  "chef_type" : "environment",
  "json_class" : "Chef::Environment"
}
```

环境文件具有以下元素：

- default_attributes- JSON 格式的默认属性。

  这些属性被添加到具有 default 属性类型的节点对象，该属性类型是所有堆栈配置和部署 JSON 属性使用的类型。本示例使用前面显示的堆栈配置和部署 JSON 的已编辑版本。

- chef_type-将此元素设置为 environment。
- json_class-将此元素设置为 Chef::Environment。

2. 编辑 .kitchen.yml 以定义 Test Kitchen 环境，如下所示。

```
---
driver:
  name: vagrant

provisioner:
  name: chef_solo
  environments_path: ./environments

platforms:
  - name: ubuntu-12.04

suites:
  - name: printjson
    provisioner:
      solo_rb:
        environment: test
    run_list:
      - recipe[printjson::default]
    attributes:
```

将以下元素添加到 kitchen init 创建的默认 .kitchen.yml 来定义环境。

provisioner

添加以下元素。

- name-将此元素设置为 chef_solo。

  要更紧密地复制 AWS OpsWorks Stacks 环境，你可以使用 Chef 客户端本地模式而不是 Chef solo 模式。本地模式是一个 Chef 客户端选项，该选项使用在实例上本地运行的轻量级版本的 Chef 服务器 (Chef Zero) 替代远程服务器。它使您的配方能够在不连接到远程服务器的情况下使用 Chef 服务器功能 (如搜索或数据包)。

- environments_path-包含环境文件的说明书子目录，本示例中为 ./environments。

suites:provisioner

加上 environment 元素设置为环境文件名称的 solo_rb 元素，再减去 .json 扩展名。本示例将 environment 设置为 test。

3. 创建一个包含以下内容的名为 default.rb 的配方文件并将该文件保存到说明书的 recipes 目录。

```
log "Stack name: #{node['opsworks']['stack']['name']}"
log "Stack id: #{node['opsworks']['stack']['id']}"
```

此配方仅记录您添加到环境的两个堆栈配置和部署值。尽管配方在 Virtual Box 中本地运行，但你可以使用与在 AWS OpsWorks Stacks 实例上运行配方时相同的节点语法来引用这些属性。

4. 运行 kitchen converge。您应看到类似于以下内容的日志输出。

```
...
Converging 2 resources
Recipe: printjson::default
  * log[Stack name: MyStack] action write[2014-07-01T23:14:09+00:00] INFO:
 Processing log[Stack name: MyStack] action write (printjson::default line 1)

[2014-07-01T23:14:09+00:00] INFO: Stack name: MyStack

  * log[Stack id: 42dfd151-6766-4f1c-9940-ba79e5220b58] action
 write[2014-07-01T23:14:09+00:00] INFO: Processing log[Stack id:
 42dfd151-6766-4f1c-9940-ba79e5220b58] action write (printjson::default line 2)
```

```
[2014-07-01T23:14:09+00:00] INFO: Stack id: 42dfd151-6766-4f1c-9940-ba79e5220b58

...
```

使用堆栈配置和部署属性值

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

配方通常需要有关堆栈配置或已部署的应用程序的信息。例如，您可能需要堆栈的 IP 地址的列表来创建配置文件，或需要应用程序的部署目录来创建日志目录。 AWS OpsWorks Stacks 没有将这些数据存储在中央服务器上，而是在每个实例的节点对象中为每个生命周期事件安装一组堆栈配置和部署属性。这些属性表示当前堆栈状态，包括已部署的应用程序。配方随后可以从节点对象中获取其需要的数据。

> **Note**
>
> 应用程序有时需要节点对象中的信息，例如堆栈配置和部署属性值。但是，应用程序无法访问节点对象。要向应用程序提供节点对象数据，您可以实施一个配方，该配方从节点对象检索所需的信息并以方便的格式将其放在一个文件中。应用程序随后可以从该文件中读取数据。有关更多信息以及示例，请参阅 传递数据到应用程序。

配方可以从节点对象获取堆栈配置和部署属性值，如下所示。

- 使用属性的完全限定名称直接获取。

  您可以将此方法用于任何 Linux 堆栈，但不能用于 Windows 堆栈。
- 借助 Chef 搜索来获取，您可以使用它在节点对象中查询属性值。

  您可以将此方法用于 Windows 堆栈和 Chef 11.10 Linux 堆栈。

> **Note**
>
> 对于 Linux 堆栈，您可以使用代理 CLI 来获取实例的堆栈配置和部署属性的副本 (JSON 格式)。有关更多信息，请参阅 在 Vagrant 上模拟堆栈配置和部署属性。

主题

直接获取属性值

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 此方法仅适用于 Linux 堆栈。

在 Vagrant 上模拟堆栈配置和部署属性介绍了如何通过使用节点语法直接引用特定属性来获取堆栈配置和部署数据。有时，这也是最佳方法。但是，很多属性是以集合或列表形式定义的，它们的内容和名称可能因堆栈而异，对于特定堆栈也可能会随着时间的推移而改变。例如，`deploy` 属性包含一个应用程序属性列表，这些属性是使用应用程序的短名称命名的。此列表 (包括应用程序属性名称) 通常会因堆栈而异，甚至会因部署而异。

通过枚举列表或集合中的属性来获取所需的数据通常更有用，有时甚至是必须的。例如，假设您想知道堆栈实例的公有 IP 地址。该信息位于 `['opsworks']['layers']` 属性中，该属性被设置为包含堆栈的各个层的一个元素的哈希表 (使用层的短名称命名)。每个层元素都被设置为包含该层的属性的哈希表，其中一个属性是 `['instances']`。该元素反过来又被设置为另一个包含该层的各个实例的一个属性的哈希表 (使用实例的短名称命名)。每个实例属性又被设置为另一个哈希表，该表包含实例属性，包括表示公有 IP 地址的 `['ip']`。如果您很难在脑海中清晰地呈现这种结构，可参考以下过程，当中的 JSON 格式的示例可帮助您理解。

此实例显示了如何从堆栈的层的堆栈配置和部署 JSON 中获取数据。

设置说明书

1. 在 opsworks_cookbooks 中创建一个名为 listip 的目录并导航到该目录。
2. 按照示例 1：安装软件包中所述初始化并配置 Test Kitchen。
3. 将以下两个目录添加到 listip：recipes 和 environments。
4. 创建包含相关属性的 MyStack 配置和部署属性的已编辑 JSON 版本。它应该类似于以下内容。

```
{
  "opsworks": {
    "layers": {
      "php-app": {
        "name": "PHP App Server",
        "id": "efd36017-ec42-4423-b655-53e4d3710652",
        "instances": {
          "php-app1": {
            "ip": "192.0.2.0"
          }
        }
      },
      "db-master": {
        "name": "MySQL",
        "id": "2d8e0b9a-0d29-43b7-8476-a9b2591a7251",
        "instances": {
          "db-master1": {
            "ip": "192.0.2.5"
          }
        }
      },
      "lb": {
        "name": "HAProxy",
        "id": "d5c4dda9-2888-4b22-b1ea-6d44c7841193",
        "instances": {
          "lb1": {
            "ip": "192.0.2.10"
          }
        }
      }
    }
  }
```

```
}
```

5. 创建名为 `test.json` 的环境文件，将示例 JSON 粘贴到 `default_attributes`，并将该文件保存到说明书的 `environments` 文件夹。该文件应类似于以下内容 (为简洁起见，大多数示例 JSON 用省略号表示)。

```
{
  "default_attributes" : {
    "opsworks": {
      "layers": {
        ...
      }
    }
  },
  "chef_type" : "environment",
  "json_class" : "Chef::Environment"
}
```

6. 使用以下内容替换 `.kitchen.yml` 中的文本。

```
---
driver:
  name: vagrant

provisioner:
  name: chef_zero
  environments_path: ./environment

platforms:
  - name: ubuntu-12.04

suites:
  - name: listip
    provisioner:
      client_rb:
        environment: test
    run_list:
      - recipe[listip::default]
    attributes:
```

设置说明书后，您可以使用以下配方来记录层 ID。

```
node['opsworks']['layers'].each do |layer, layerdata|
  log "#{layerdata['name']} : #{layerdata['id']}"
end
```

该配方将枚举 ['opsworks']['layers'] 中的层并记录每个层的名称和 ID。

运行层 ID 日志记录配方

1. 使用示例配方创建一个名为 default.rb 的文件，并将该文件保存到 recipes 目录。
2. 运行 kitchen converge。

输出的相关部分应该类似于以下内容。

```
Recipe: listip::default
  * log[PHP App Server : efd36017-ec42-4423-b655-53e4d3710652] action
 write[2014-07-17T22:56:19+00:00] INFO: Processing log[PHP App Server : efd36017-
ec42-4423-b655-53e4d3710652] action write (listip::default line 4)
[2014-07-17T22:56:19+00:00] INFO: PHP App Server : efd36017-ec42-4423-b655-53e4d3710652



  * log[MySQL : 2d8e0b9a-0d29-43b7-8476-a9b2591a7251] action
 write[2014-07-17T22:56:19+00:00] INFO: Processing log[MySQL : 2d8e0b9a-0d29-43b7-8476-
a9b2591a7251] action write (listip::default line 4)
[2014-07-17T22:56:19+00:00] INFO: MySQL : 2d8e0b9a-0d29-43b7-8476-a9b2591a7251


  * log[HAProxy : d5c4dda9-2888-4b22-b1ea-6d44c7841193] action
 write[2014-07-17T22:56:19+00:00] INFO: Processing log[HAProxy : d5c4dda9-2888-4b22-
b1ea-6d44c7841193] action write (listip::default line 4)
[2014-07-17T22:56:19+00:00] INFO: HAProxy : d5c4dda9-2888-4b22-b1ea-6d44c7841193
```

要列出实例的 IP 地址，您需要一个类似于下方所示的嵌套循环。

```
node['opsworks']['layers'].each do |layer, layerdata|
```

```
  log "#{layerdata['name']} : #{layerdata['id']}"
  layerdata['instances'].each do |instance, instancedata|
    log "Public IP: #{instancedata['ip']}"
  end
end
```

内部循环将遍历每个层的实例并记录 IP 地址。

运行实例 IP 日志记录配方

1. 将 default.rb 中的代码替换为示例配方。
2. 运行 kitchen converge 以执行该配方。

输出的相关部分应该类似于以下内容。

```
  * log[PHP App Server : efd36017-ec42-4423-b655-53e4d3710652] action
 write[2014-07-17T23:09:34+00:00] INFO: Processing log[PHP App Server : efd36017-
ec42-4423-b655-53e4d3710652] action write (listip::default line 2)
[2014-07-17T23:09:34+00:00] INFO: PHP App Server : efd36017-ec42-4423-b655-53e4d3710652



  * log[Public IP: 192.0.2.0] action write[2014-07-17T23:09:34+00:00] INFO: Processing
 log[Public IP: 192.0.2.0] action write (listip::default line 4)
[2014-07-17T23:09:34+00:00] INFO: Public IP: 192.0.2.0


  * log[MySQL : 2d8e0b9a-0d29-43b7-8476-a9b2591a7251] action
 write[2014-07-17T23:09:34+00:00] INFO: Processing log[MySQL : 2d8e0b9a-0d29-43b7-8476-
a9b2591a7251] action write (listip::default line 2)
[2014-07-17T23:09:34+00:00] INFO: MySQL : 2d8e0b9a-0d29-43b7-8476-a9b2591a7251


  * log[Public IP: 192.0.2.5] action write[2014-07-17T23:09:34+00:00] INFO: Processing
 log[Public IP: 192.0.2.5] action write (listip::default line 4)
[2014-07-17T23:09:34+00:00] INFO: Public IP: 192.0.2.5


  * log[HAProxy : d5c4dda9-2888-4b22-b1ea-6d44c7841193] action
 write[2014-07-17T23:09:34+00:00] INFO: Processing log[HAProxy : d5c4dda9-2888-4b22-
b1ea-6d44c7841193] action write (listip::default line 2)
```

```
[2014-07-17T23:09:34+00:00] INFO: HAProxy : d5c4dda9-2888-4b22-b1ea-6d44c7841193


  * log[Public IP: 192.0.2.10] action write[2014-07-17T23:09:34+00:00] INFO: Processing
 log[Public IP: 192.0.2.10] action write (listip::default line 4)
[2014-07-17T23:09:34+00:00] INFO: Public IP: 192.0.2.10
```

完成后，请运行 `kitchen destroy`；下一个主题将使用新说明书。

> **Note**
>
> 枚举堆栈配置和部署 JSON 集合的最常见的原因之一是获取特定的已部署应用程序 (例如其部署目录) 的数据。有关示例，请参阅Deploy 配方。

使用 Chef 搜索获取属性值

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 此方法可用于 Windows 堆栈和 Chef 11.10 Linux 堆栈。

直接从节点对象获取堆栈配置和部署属性值可能很复杂，并且不能用于 Windows 堆栈。替代方法是使用 Chef 搜索查询相关的属性。如果你熟悉 Chef 服务器，你会发现 Chef 搜索与 AWS OpsWorks Stacks 的工作方式略有不同。由于 AWS OpsWorks Stacks 在本地模式下使用 chef-client，因此 Chef 搜索依赖于名为 chef-zero 的 Chef 服务器的本地版本，因此搜索对存储在实例节点对象中的本地数据而不是远程服务器上的数据进行搜索。

实际上，将搜索限制在本地存储的数据上通常并不重要，因为 AWS OpsWorks Stacks 实例上的节点对象包含堆栈配置和部署属性。它们包含食谱通常从Chef服务器获取的大部分（如果不是全部）数据，并且使用相同的名称，因此您通常可以在 AWS OpsWorks Stacks实例上使用为Chef服务器编写的搜索代码，而无需修改。有关更多信息，请参阅 使用 Chef 搜索。

下面显示了搜索查询的基本结构：

```
result = search(:search_index, "key:pattern")
```

- 搜索索引指定了查询应用于的属性并确定了返回的对象的类型。
- 键指定了属性名称。
- 该模式指定要检索的属性的值。

  您可查询特定属性值或使用通配符查询一系列值。
- 结果将显示一张列表，当中包含满足查询条件的对象，每个列表均为包含多个相关属性的哈希表。

  例如，如果您使用 node 搜索索引，则查询将返回一个实例对象列表 (每个满足查询的实例对应一个实例对象)。每个对象均为一个哈希表，其中包含一组定义实例配置的属性 (如主机名和 IP 地址)。

例如，以下查询使用 node 搜索索引，此索引是适用于堆栈的实例 (Chef 术语中的节点) 的标准 Chef 索引。它使用主机名 `myhost` 搜索实例。

```
result = search(:node, "hostname:myhost")
```

搜索将返回主机名为 `myhost` 的实例对象的列表。例如，如果您需要第一个实例的操作系统，它将由 `result[0][:os]` 表示。如果查询返回多个对象，则可枚举它们以检索必需信息。

如何在配方中使用搜索的详细信息取决于您使用的是 Linux 还是 Windows 堆栈。下列主题提供了这两种堆栈类型的示例。

主题

- 对 Linux 堆栈使用搜索
- 对 Windows 堆栈使用搜索

对 Linux 堆栈使用搜索

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

此示例基于包含单台 PHP 应用程序服务器的 Linux 堆栈。它使用 Chef 搜索来获取服务器的公有 IP 地址，然后将该地址放入 /tmp 目录下的文件中。实际上，它将从节点对象获取与直接获取属性值 相同的信息，但代码简单得多，并且不依赖堆栈配置和部署属性结构的详细信息。

下面简单总结了为此示例创建堆栈的方式。有关更多信息，请参阅 创建新堆栈。

> **Note**
>
> 如果你之前没有在 AWS OpsWorks Stacks 实例上运行过自定义配方，你应该先看一下这个在 Linux 实例上运行配方例子。

创建堆栈

1. 打开 AWS OpsWorks Stacks 控制台，然后单击 Add Stack (添加堆栈)。
2. 指定以下设置，接受其他设置的默认值，然后单击 Add Stack。

   - 名称 - SearchJSON
   - 默认 SSH 密钥 - Amazon EC2 密钥对

   如果您需要创建 Amazon EC2 密钥对，请参阅 Amazon EC2 密钥对。请注意，该密钥对必须属于与实例相同的 Amazon Web Services Region。此示例使用美国西部（俄勒冈州）区域。
3. 单击 Add a layer，然后将采用默认设置的 PHP App Server 层添加到堆栈。
4. 向层添加全天候实例 (采用默认设置) 并启动该实例。

设置说明书

1. 在 opsworks_cookbooks 中创建一个名为 searchjson 的目录并导航到该目录。
2. 创建一个包含以下内容的 metadata.rb 文件，并将该文件保存到 opstest。

   ```
   name "searchjson"
   version "0.1.0"
   ```
3. 在 recipes 中创建 searchjson 目录。
4. 创建包含以下配方的 default.rb 文件，并将该文件保存到 recipes 目录。

```
phpserver = search(:node, "layers:php-app").first
Chef::Log.info("**********The public IP address is: '#{phpserver[:ip]}'**********")

file "/tmp/ip_addresses" do
  content "#{phpserver[:ip]}"
  mode 0644
  action :create
end
```

Linux 堆栈仅支持 node 搜索索引。配方使用此索引获取 php-app 层中实例的列表。由于已知层只有一个实例，配方只需将第一个实例分配给 phpserver。如果层具有多个实例，您可枚举这些实例以检索必需信息。每个列表项均为一个包含一组实例属性的哈希表。ip 属性设置为实例的公有 IP 地址，因此您可在后续配方代码中将此地址表示为 phpserver[:ip]。

在将消息添加到 Chef 日志后，配方将使用 file 资源创建一个名为 ip_addresses 的文件。content 属性设置为 phpserver[:ip] 的字符串表示形式。当 Chef 创建 ip_addresses 时，它会将此字符串添加到该文件。

5. 为 opsworks_cookbooks 创建 .zip 存档，将存档上传到 Amazon S3 存储桶，公开存档并记录存档的 URL。有关说明书存储库的更多信息，请参阅说明书存储库。

   发送到 Amazon S3 存储桶的内容可能包含客户内容。有关删除敏感数据的更多信息，请参阅如何清空 S3 存储桶？或如何删除 S3 存储桶？。

您现在可以安装说明书并运行配方。

运行配方

1. 编辑堆栈以启用自定义说明书，然后指定以下设置。

   - 存储库类型 -Http 存档
   - 存储库 URL -您之前记录的说明书存档 URL

   对其他设置使用默认值，然后单击 Save 更新堆栈配置。

2. 编辑自定义图层配置并分配searchjson::default给图层的 Setup 事件。 AWS OpsWorks 堆栈将在实例启动后或您明确触发安装事件后运行配方。
3. 运行“Update Custom Cookbooks”堆栈命令，这会将当前版本的自定义说明书存储库安装到堆栈的实例上。如果存在早期版本的存储库，此命令会将其覆盖。

4. 通过运行 Setup 堆栈命令在实例上触发设置事件并运行 `searchjson::default` 来执行配方。使 Running command setup page 保持打开状态。

配方成功运行后，可以对其进行验证。

验证 searchjson

1. 第一步是查看 Chef 日志以了解最新的设置事件。在 Running command setup page 上，单击 php-app1 实例 Log 列中的 show，显示日志。向下滚动以查找您的日志消息 (接近中间位置)，该日志消息与以下内容类似。

```
...
[2014-09-05T17:08:41+00:00] WARN: Previous
 bash[logdir_existence_and_restart_apache2]: ...
[2014-09-05T17:08:41+00:00] WARN: Current
 bash[logdir_existence_and_restart_apache2]: ...
[2014-09-05T17:08:41+00:00] INFO: **********The public IP address is:
 '192.0.2.0'**********
[2014-09-05T17:08:41+00:00] INFO: Processing directory[/etc/sysctl.d] action create
 (opsworks_initial_setup::sysctl line 1)
...
```

2. 使用 SSH 登录实例并列出 `/tmp` 的内容，该内容应包含一个名为 `ip_addresses` 的文件 (其中包含 IP 地址)。

对 Windows 堆栈使用搜索

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

AWS OpsWorks Stacks 提供了两个在 Windows 堆栈上使用搜索的选项。

- node 搜索索引，可用于查询一组标准 Chef 属性。

如果您有带有搜索代码的现有配方，则它们通常无需修改即可在 AWS OpsWorks Stacks 堆栈上运行。node

- 一组附加搜索索引，可用于查询 AWS OpsWorks Stacks 特定的属性组以及一些标准属性。

  在 Window AWS OpsWorks s 堆栈上使用特定于堆栈的搜索索引中讨论了这些索引。

建议使用 node 检索标准信息 (如主机名或 IP 地址)。这种方法将使您的配方与标准 Chef 实践保持一致。使用 AWS OpsWorks 堆栈搜索索引检索特定于 AWS OpsWorks 堆栈的信息。

主题

- 对 Windows 堆栈使用节点搜索索引
- 在 Window AWS OpsWorks s 堆栈上使用特定于堆栈的搜索索引

对 Windows 堆栈使用节点搜索索引

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 此示例假定您已完成在 Windows 实例上运行配方示例。如果没有，您应该先完成该示例。具体而言，它介绍了如何启用对实例的 RDP 访问。

此示例基于一个具有一个自定义层和一个实例的 Windows 堆栈。它使用带 node 搜索索引的 Chef 搜索来获取服务器的公用 IP 地址，并将该地址放入 `C:\tmp` 目录中的文件中。下面简单总结了为此示例创建堆栈的方式。有关更多信息，请参阅 创建新堆栈。

创建堆栈

1. 打开 AWS OpsWorks Stacks 控制台，然后选择 Add Stack (添加堆栈)。
2. 指定以下设置，接受其他设置的默认值，然后选择 Add Stack。

- 姓名 — NodeSearch
- 区域-美国西部（俄勒冈州）

  本示例在任何区域都有效，但我们建议将美国西部（俄勒冈州）用于教程。
- 默认操作系统 - Microsoft Windows Server 2012 R2

3. 选择 Add a layer 并将采用以下设置的自定义层添加到堆栈。
   - 名称 - IPTest
   - 短名称 - iptest
4. 向 IPTest 层添加全天候 t2.micro 实例 (采用默认设置) 并启动此实例。它将被命名为 iptest1。

   AWS OpsWorks 堆栈会自动分配AWS-OpsWorks-RDP-Server给此实例，从而允许授权用户登录该实例。
5. 选择 Permissions、Edit，然后选择 SSH/RDP 和 sudo/admin。除了 AWS-OpsWorks-RDP-Server 安全组之外，普通用户还需要此授权才能登录实例。

> **Note**
>
> 您还可以管理员身份登录，但这需要其他过程。有关更多信息，请参阅 使用 RDP 登录。

设置说明书

1. 创建一个名为 nodesearch 的目录并导航到该目录。
2. 创建一个包含以下内容的 metadata.rb 文件，并将该文件保存到 opstest。

```
name "nodesearch"
version "0.1.0"
```

3. 在 recipes 中创建 nodesearch 目录。
4. 创建包含以下配方的 default.rb 文件，并将该文件保存到 recipes 目录。

```
directory 'C:\tmp' do
  rights :full_control, 'Everyone'
  recursive true
```

```
  action :create
end

windowsserver = search(:node, "hostname:iptest*").first
Chef::Log.info("**********The public IP address is:
 '#{windowsserver[:ipaddress]}'**********")

file 'C:\tmp\addresses.txt' do
  content "#{windowsserver[:ipaddress]}"
  rights :full_control, 'Everyone'
  action :create
end
```

此配方会执行以下操作：

1. 使用目录资源为此文件创建一个 `C:\tmp` 目录。

   有关此资源的更多信息，请参阅示例 3：创建目录。

2. 使用带 node 搜索索引的 Chef 搜索来获取主机名以 `iptest` 开头的节点 (实例) 的列表。

   如果您使用默认主题 (通过向层的短名称追加整数来创建主机名)，此查询将返回 IPTest 层中的每个实例。在此示例中，已知层只有一个实例，因此配方只需将第一个实例分配给 `windowsserver`。对于多个实例，您可获取完整列表，然后枚举这些实例。

3. 对于此运行，将包含 IP 地址的消息添加到 Chef 日志。

   `windowsserver` 对象是一个 `ipaddress` 属性设置为实例的公有 IP 地址的哈希表，因此您可在后续配方代码中将此地址表示为 `windowsserver[:ipaddress]`。配方将对应字符串插入消息并将消息添加到 Chef 日志中。

4. 使用 `file` 资源创建一个名为 `C:\tmp\addresses.txt` 的带此 IP 地址的文件。

   资源的 `content` 属性指定要添加到文件中的内容 (在此示例中为公有 IP 地址)。

5. 为 `nodesearch` 创建 `.zip` 存档，将存档上传到 S3 存储桶，公开存档并记录存档的 URL。

   发送到 Amazon S3 存储桶的内容可能包含客户内容。有关删除敏感数据的更多信息，请参阅如何清空 S3 存储桶？或如何删除 S3 存储桶？。

您现在可以安装说明书并运行配方。

安装说明书并运行配方

1. 编辑堆栈以启用自定义说明书，然后指定以下设置。

   - 存储库类型 - S3 存档
   - 存储库 URL -您之前记录的说明书存档 URL

   接受其他设置的默认值，然后选择 Save 更新堆栈配置。
2. 运行“Update Custom Cookbooks”堆栈命令，这会将当前版本的自定义说明书安装到堆栈的实例 (包括联机实例) 上。如果您的说明书存在早期版本，此命令会覆盖该版本。
3. 完成 Update Custom Cookbooks 命令后，通过在 Recipes to execute 设置为 **`nodesearch::default`** 的情况下运行 Execute Recipes 堆栈命令来执行配方。此命令将启动 Chef 运行并显示一个包含配方的运行列表。使 execute_recipes 页面保持打开状态。

配方成功运行后，可以对其进行验证。

验证 nodesearch

1. 查看 Chef 日志以了解最新的 execute_recipes 事件。在 Running command execute_recipes page 上，选择 iptest1 实例 Log 列中的 show，显示日志。向下滚动以查找您的日志消息 (接近底部位置)，该日志消息与以下内容类似。

```
...
[2015-05-13T18:55:47+00:00] INFO: Storing updated cookbooks/nodesearch/recipes/
default.rb in the cache.
[2015-05-13T18:55:47+00:00] INFO: Storing updated cookbooks/nodesearch/metadata.rb
 in the cache.
[2015-05-13T18:55:47+00:00] INFO: **********The public IP address is:
 '192.0.0.1'**********
[2015-05-13T18:55:47+00:00] INFO: Processing directory[C:\tmp] action create
 (nodesearch::default line 1)
[2015-05-13T18:55:47+00:00] INFO: Processing file[C:\tmp\addresses.txt] action
 create (nodesearch::default line 10)
...
```

2. 使用 RDP 登录实例并检查 `C:\tmp\addresses.txt` 的内容。

在 Window AWS OpsWorks s 堆栈上使用特定于堆栈的搜索索引

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 此示例假定您已完成在 Windows 实例上运行配方示例。如果没有，您应该先完成该示例。具体而言，它介绍了如何启用对实例的 RDP 访问。

AWS OpsWorks 除了以下搜索索引外，Stacks 还提供以下搜索索引：node

- `aws_opsworks_stack`-堆栈配置。
- `aws_opsworks_layer`-堆栈的层配置。
- `aws_opsworks_instance`-堆栈的实例配置。
- `aws_opsworks_app`-堆栈的应用程序配置。
- `aws_opsworks_user`-堆栈的用户配置。
- `aws_opsworks_rds_db_instance`-已注册 RDS 实例的连接信息。

这些索引包括一些标准的 Chef 属性，但主要用于检索特定于 AWS OpsWorks Stacks 的属性。例如，`aws_opsworks_instance` 包含一个提供实例状态 (如 `status`) 的 `online` 属性。

> **Note**
>
> 建议的做法是，在能够确保配方与标准 Chef 用法保持一致的情况下使用 node。有关示例，请参阅对 Windows 堆栈使用节点搜索索引。

此示例说明如何使用 AWS OpsWorks 堆栈索引来检索 AWS OpsWorks 堆栈特定属性的值。它基于一个具有自定义层 (带一个实例) 的简单 Windows 堆栈。它使用 Chef 搜索来获取实例的 AWS OpsWorks Stacks ID 并将结果放入 Chef 日志中。

下面简要总结了如何为本示例创建堆栈。有关更多信息，请参阅 创建新堆栈。

创建堆栈

1. 打开 AWS OpsWorks Stacks 控制台，然后选择 + Stack (+ 堆栈)。指定以下设置，接受其他设置的默认值，然后选择 Add Stack。

   - 名称 - IDSearch
   - 区域-美国西部（俄勒冈州）

     本示例在任何区域都有效，但我们建议将美国西部（俄勒冈州）用于教程。
   - 默认操作系统 - Microsoft Windows Server 2012 R2
2. 选择 Add a layer 并将采用以下设置的自定义层添加到堆栈。

   - 名称 - IDCheck
   - 短名称 - idcheck
3. 向 IDCheck 层添加全天候 t2.micro 实例 (采用默认设置) 并启动此实例。它将被命名为 iptest1。

   AWS OpsWorks 堆栈会自动分配`AWS-OpsWorks-RDP-Server`给此实例。 启用 RDP 访问说明如何向该安全组添加允许授权用户登录实例的入站规则。
4. 选择 Permissions、Edit，然后选择 SSH/RDP 和 sudo/admin。除了 `AWS-OpsWorks-RDP-Server` 安全组之外，普通用户还需要此授权才能登录实例。

> **Note**
>
> 您也可以以管理员身份登录，但需要采用另一个登录过程。有关更多信息，请参阅 使用 RDP 登录。

设置说明书

1. 创建一个名为 `idcheck` 的目录并导航到该目录。
2. 创建一个包含以下内容的 `metadata.rb` 文件，并将该文件保存到 `opstest`。

```
name "idcheck"
version "0.1.0"
```

3. 在 recipes 中创建一个 idcheck 目录，并将包含以下配方的 default.rb 文件添加到该目录。

```
windowsserver = search(:aws_opsworks_instance, "hostname:idcheck*").first
Chef::Log.info("**********The public IP address is:
 '#{windowsserver[:instance_id]}'**********")
```

此配方使用包含 aws_opsworks_instance 搜索索引的 Chef 搜索来获取堆栈中每个主机名以 idcheck 开头的实例的实例属性。如果您使用默认主题 (通过向层的短名称追加整数来创建主机名)，此查询将返回 IDCheck 层中的每个实例。在此示例中，已知层只有一个实例，因此配方只需将第一个实例分配给 windowsserver。对于多个实例，您可获取完整列表，然后枚举这些实例。

此配方利用堆栈中只有一个实例具有此主机名的事实，因此第一个结果是正确结果。如果您的堆栈具有多个实例，则搜索其他属性可能返回多个结果。有关实例属性的列表，请参阅实例数据包 (aws_opsworks_instance)。

实例属性基本上是一个哈希表，实例的 AWS OpsWorks 堆栈 ID 已分配给该instance_id属性，因此您可以将 ID 引用为windowsserver[:instance_id]。配方将对应字符串插入消息并将消息添加到 Chef 日志中。

4. 创建 ipaddress 说明书的 .zip 存档，将存档上传到 Amazon S3 存储桶并记录存档的 URL。有关说明书存储库的更多信息，请参阅说明书存储库。

发送到 Amazon S3 存储桶的内容可能包含客户内容。有关删除敏感数据的更多信息，请参阅如何清空 S3 存储桶？或如何删除 S3 存储桶？。

您现在可以安装说明书并运行配方。

安装说明书并运行配方

1. 编辑堆栈以启用自定义说明书，然后指定以下设置。

   - 存储库类型 - S3 存档
   - 存储库 URL -您之前记录的说明书存档 URL

接受其他设置的默认值，然后选择 Save 更新堆栈配置。

2. 运行“Update Custom Cookbooks”堆栈命令，这会将当前版本的自定义说明书安装到堆栈的实例 (包括联机实例) 上。如果您的说明书存在早期版本，此命令会覆盖该版本。
3. 完成“Update Custom Cookbooks”命令后，通过在 Recipes to execute 设置为 **`idcheck::default`** 的情况下运行 Execute Recipes 堆栈命令执行配方。此命令将启动 Chef 运行并显示一个包含配方的运行列表。使 execute_recipes 页面保持打开状态。

配方成功运行后，您可通过查看 Chef 日志中的最新 execute_recipes 事件来验证它。在 Running command execute_recipes page 上，选择 iptest1 实例 Log 列中的 show，显示日志。向下滚动以查找您的日志消息 (接近底部位置)，该日志消息与以下内容类似。

```
...
[2015-05-13T20:03:47+00:00] INFO: Storing updated cookbooks/nodesearch/recipes/
default.rb in the cache.
[2015-05-13T20:03:47+00:00] INFO: Storing updated cookbooks/nodesearch/metadata.rb in
 the cache.
[2015-05-13T20:03:47+00:00] INFO: **********The instance ID is: 'i-8703b570'**********
[2015-05-13T20:03:47+00:00] INFO: Chef Run complete in 0.312518 seconds
...
```

在 Linux 实例上使用外部说明书：Berkshelf

Important

该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

Note

Berkshelf 仅适用于 Chef 11.10 Linux 堆栈。

在开始实施说明书之前，请查看 Chef 社区说明书，此类说明书包含 Chef 社区成员针对多种用途创建的说明书。这些食谱中有许多无需修改即可与 AWS OpsWorks Stacks 一起使用，因此您可以利用它们来完成某些任务，而不必自己实现所有代码。

要在实例上使用外部说明书，您需要通过某种方式来安装说明书和管理任何依赖项。首选方法是实施支持名为 Berkshelf 的依存关系管理器的说明书。Berkshelf 适用于亚马逊 EC2 实例，包括 AWS OpsWorks Stacks 实例，但它也专为与 Test Kitchen 和 Vagrant 配合使用而设计。但是，Vagrant上的用法与 AWS OpsWorks Stacks的用法略有不同，因此本主题包括两个平台的示例。有关如何使用 Berkshelf 的更多信息，请参阅 Berkshelf。

主题

- 将 Berkshelf 用于 Test Kitchen 和 Vagrant
- 将 Berkshelf 与堆栈一起使用 AWS OpsWorks

将 Berkshelf 用于 Test Kitchen 和 Vagrant

此示例演示如何使用 Berkshelf 安装入门社区说明书并执行其配方，这将在实例的主目录中安装简短文本文件。

安装 Berkshelf 并初始化说明书

1. 在工作站上，按照如下所示安装 Berkshelf gem。

   ```
   gem install berkshelf
   ```

   根据您的工作站，此命令可能需要 `sudo`，或者您也可以使用 Ruby 环境管理器 (如 RVM)。要验证 Berkshelf 是否已安装成功，请运行 `berks --version`。

2. 此主题的说明书名为 external_cookbook。您可使用 Berkshelf 创建已初始化的说明书而不是之前的主题采用的手动方法。为此，请导航到 `opsworks_cookbooks` 目录并运行以下命令。

   ```
   berks cookbook external_cookbook
   ```

   此命令将创建 `external_cookbook` 目录和多个标准 Chef 和 Test Kitchen 子目录，包括 `recipes` 和 `test`。此命令还将创建大量标准文件的默认版本，包括：

   - `metadata.rb`
   - Vagrant、Test Kitchen 和 Berkshelf 的配置文件
   - `default.rb` 目录中的空 `recipes` 配方

> **Note**
>
> 您无需运行 `kitchen init`；`berks cookbook` 命令将处理这些任务。

3. 运行 `kitchen converge`。新创建的说明书此时不执行任何相关操作，而是执行聚合操作。

> **Note**
>
> 您还可使用 `berks init` 初始化现有说明书以使用 Berkshelf。

要使用 Berkshelf 管理说明书的外部依赖项，说明书的根目录必须包含 `Berksfile` (它是一个指定 Berkshelf 应如何管理依赖项的配置文件)。如果您使用 `berks cookbook` 创建 `external_cookbook` 说明书，则将使用下列内容创建一个 `Berksfile`。

```
source "https://supermarket.chef.io"
metadata
```

此文件具有以下声明：

- `source`-说明书源的 URL。

  一个 Berksfile 可以包含任意数量的 `source` 声明，每个声明指定依赖说明书的默认源。如果您没有显式指定说明书的源，则 Berkshelf 将在默认存储库中查找同名说明书。默认 Berksfile 包含一个指定社区说明书存储库的 `source` 属性。该存储库包含入门说明书，因此您可将此行保持不变。
- `metadata`-指示 Berkshelf 包含说明书的 `metadata.rb` 文件中声明的说明书依赖项。

  您也可通过包含 `cookbook` 属性来在 Berksfile 中声明依赖说明书，如下文所述。

可通过两种方法声明说明书依赖项：

- 在 Berksfile 中包括 `cookbook` 声明。

  这是 AWS OpsWorks Stacks 使用的方法。例如，要指定此示例中使用的入门说明书，请在 Berksfile 中包含 `cookbook "getting-started"`。随后，Berkshelf 将在默认存储库中查找带此

名称的说明书。您也可以使用 cookbook 显式指定说明书源，甚至指定特定版本。有关更多信息，请参阅 Berkshelf。

- 在 Berksfile 中包含 metadata 声明并在 metadata.rb 中声明依赖项。

  此声明指示 Berkshelf 包含 metadata.rb 中声明的说明书依赖项。例如，要声明入门依赖项，请将 depends 'getting-started' 声明添加到说明书的 metadata.rb 文件。

为了与 AWS OpsWorks Stacks 保持一致，此示例使用了第一种方法。

安装入门说明书

1. 编辑默认 Berksfile 以将 metadata 声明替换为 cookbook 的 getting-started 声明。内容应与以下内容类似。

```
source "https://supermarket.chef.io"

cookbook 'getting-started'
```

2. 运行 berks install 以从社区说明书存储库中将入门说明书下载到您的工作站的 Berkshelf 目录 (通常为 ~/.berkshelf)。此目录通常称为 Berkshelf。在 Berkshelf 的 cookbooks 目录中查找，您应看到入门说明书的目录，其名称类似于 getting-started-0.4.0。
3. 将 external_cookbook::default 运行列表中的 .kitchen.yml 替换为 getting-started::default。此示例不运行来自 external_cookbook 的任何配方，它只是一种使用入门说明书的方式。现在 .kitchen.yml 文件应该呈现以下状态。

```
---
driver:
  name: vagrant

provisioner:
  name: chef_solo

platforms:
  - name: ubuntu-12.04

suites:
  - name: default
    run_list:
```

```
      - recipe[getting-started::default]
    attributes:
```

4. 运行 `kitchen converge`，然后使用 `kitchen login` 登录实例。登录目录应包含名为 `chef-getting-started.txt` 的文件，其中部分内容与以下内容类似：

```
Welcome to Chef!

This is Chef version 11.12.8.
Running on ubuntu.
Version 12.04.
```

   Test Kitchen 将在实例的 `/tmp/kitchen/cookbooks` 目录中安装说明书。如果您列出该目录的内容，则将看到两个说明书：external_cookbook 和 getting-started。

5. 运行 `kitchen destroy` 以关闭实例。下一个示例使用 AWS OpsWorks 堆栈实例。

将 Berkshelf 与堆栈一起使用 AWS OpsWorks

AWS OpsWorks Stacks 可以选择支持 Berkshelf for Chef 11.10 堆栈。要将 Berkshelf 用于您的堆栈，您必须执行下列操作。

- 为堆栈启用 Berkshelf。

  AWS OpsWorks 然后，堆栈会处理在堆栈实例上安装 Berkshelf 的细节。
- 将 Berksfile 添加到您的说明书存储库的根目录。

  Berksfile 应包含所有依赖说明书的 `source` 和 `cookbook` 声明。

当 AWS OpsWorks Stacks 在实例上安装您的自定义食谱存储库时，它会使用 Berkshelf 来安装存储库的 Berksfile 中声明的依赖食谱。有关更多信息，请参阅 使用 Berkshelf。

此示例说明如何使用 Berkshelf 在 Stacks 实例上安装入门社区食谱。 AWS OpsWorks 它还将安装某个版本的 createfile 自定义说明书，这将在指定目录中创建一个文件。有关 createfile 工作原理的更多信息，请参阅从说明书安装文件。

> **Note**
>
> 如果这是你第一次在 Stac AWS OpsWorks ks 堆栈上安装自定义食谱，你应该先看一下这个例子。在 Linux 实例上运行配方

首先创建堆栈，如以下内容所汇总。有关更多信息，请参阅 创建新堆栈。

创建堆栈

1. 打开 AWS OpsWorks Stacks 控制台，然后单击 Add Stack (添加堆栈)。
2. 指定以下设置，接受其他设置的默认值，然后单击 Add Stack。

   - 姓名 — BerksTest
   - 默认 SSH 密钥 - Amazon EC2 密钥对

   如果您需要创建 Amazon EC2 密钥对，请参阅 Amazon EC2 密钥对。请注意，该密钥对必须属于与实例相同的 Amazon Web Services Region。本示例使用了默认的美国西部（俄勒冈州）区域。
3. 单击 Add a layer 并将采用以下设置的自定义层添加到堆栈。

   - 姓名 — BerksTest
   - 短名称 - berkstest

   在此示例中，您实际上可使用任何层类型。但是，此示例不需要其他层安装的任何程序包，因此自定义层是最简单的方法。
4. 使用默认设置向 BerksTest 图层@@ 添加全天候实例，但暂时不要启动。

使用 AWS OpsWorks Stacks，食谱必须位于具有标准目录结构的远程存储库中。然后，您将下载信息提供给 AWS OpsWorks Stacks，堆栈会在启动时自动将存储库下载到堆栈的每个实例。为简单起见，本示例的存储库是一个公共的 Amazon S3 档案，但是 AWS OpsWorks Stacks 还支持 HTTP 档案、Git 存储库和 Subversion 存储库。有关更多信息，请参阅 说明书存储库。

发送到 Amazon S3 存储桶的内容可能包含客户内容。有关删除敏感数据的更多信息，请参阅如何清空 S3 存储桶？或如何删除 S3 存储桶？。

创建说明书存储库

1. 在您的 opsworks_cookbooks 目录中，创建名为 berkstest_cookbooks 的目录。如果您愿意，可在任何方便位置创建此目录，因为您会将它上传到存储库。
2. 将名为 Berksfile 的文件添加到具有下列内容的 berkstest_cookbooks。

```
source "https://supermarket.chef.io"

cookbook 'getting-started'
```

   此文件将声明入门说明书依赖项，并指示 Berkshelf 从社区说明书站点下载此依赖项。
3. 将 createfile 目录添加到包含以下内容的 berkstest_cookbooks 中。
   - 包含下列内容的 metadata.rb 文件。

```
name "createfile"
version "0.1.0"
```

   - 一个 files/default 目录，其中包含具有以下内容的 example_data.json 文件。

```
{
  "my_name" : "myname",
  "your_name" : "yourname",
  "a_number" : 42,
  "a_boolean" : true
}
```

     该文件的名称和内容是任意的。配方会将此文件复制到指定位置。
   - 一个 recipes 目录，其中包含具有以下配方代码的 default.rb 文件。

```
directory "/srv/www/shared" do
  mode 0755
  owner 'root'
  group 'root'
  recursive true
  action :create
```

```
end

cookbook_file "/srv/www/shared/example_data.json" do
  source "example_data.json"
  mode 0644
  action :create_if_missing
end
```

此配方创建 /srv/www/shared 并将 example_data.json 从说明书的 files 目录复制到此目录。

4. 为 berkstest_cookbooks 创建 .zip 存档，将存档上传到 Amazon S3 存储桶，公开存档并记录存档的 URL。

您现在可以安装说明书并运行配方。

安装说明书并运行配方

1. 编辑堆栈以启用自定义说明书，然后指定以下设置。

   - 存储库类型 -Http 存档
   - 存储库 URL -您之前记录的说明书存档 URL
   - 管理 Berkshelf)- 是

   前两个设置为 AWS OpsWorks Stacks 提供了将食谱存储库下载到您的实例所需的信息。最后一个设置启用 Berkshelf 支持，这会将入门说明书下载到实例。接受其他设置的默认值，然后单击 Save 更新堆栈配置。

2. 编辑 BerksTest 图层，将以下配方添加到该图层的设置生命周期事件中。

   - getting-started::default
   - createfile::default

3. 启动实例。Setup 事件在实例完成启动后发生。 AWS OpsWorks 然后，Stacks 安装食谱存储库，使用 Berkshelf 下载入门食谱，然后运行该层的设置和部署食谱，包括和。getting-started::default createfile::default

4. 在实例处于联机状态后，使用 SSH 登录。您应该看到以下内容

   - /srv/www/shared 应包含 example_data.json。
   - /root 应包含 chef-getting-started.txt。

AWS OpsWorks Stacks 以 root 用户身份运行配方，因此入门会将文件安装在`/root`目录中，而不是您的主目录中。

使用 SDK for Ruby：从 Amazon S3 下载文件

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

无法使用 Chef 资源处理某些任务 (如与 Amazon Web Service 进行交互)。例如，有时远程存储文件并将配方下载到实例更可取。您可使用 remote_file 资源从远程服务器下载文件。但是，如果您要将文件存储在 Amazon S3 存储桶中，则仅当 ACL 允许此操作时，`remote_file` 才能下载这些文件。

配方可使用AWS SDK for Ruby访问大多数 Amazon Web Service。本主题说明如何使用 SDK for Ruby; 从 S3 存储桶下载文件。

> **Note**
>
> 有关如何使用AWS SDK for Ruby处理加密和解密的更多信息，请参阅 AWS::S3::S3Object。发送到 Amazon S3 存储桶的内容可能包含客户内容。有关删除敏感数据的更多信息，请参阅如何清空 S3 存储桶？或如何删除 S3 存储桶？。

主题

- 在 Vagrant 实例上使用 SDK for Ruby
- 在 AWS OpsWorks Stacks Linux 实例上使用适用于 Ruby 的 SDK
- 在 AWS OpsWorks Stacks Windows 实例上使用适用于 Ruby 的 SDK

在 Vagrant 实例上使用 SDK for Ruby

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

本主题介绍在 Vagrant 实例上运行的配方如何使用AWS SDK for Ruby从 Amazon S3 下载文件。在开始之前，您必须先拥有一组 AWS 证书（访问密钥和私有访问密钥），允许配方访问 Amazon S3。

> **Important**
>
> 强烈建议您不要将根账户凭证用于此目的。而是使用适当的策略创建用户并为该配方提供这些凭证。
>
> 注意不要将证书（甚至是 IAM 用户证书）放在可公开访问的位置，例如将包含证书的文件上传到公共存储库或 Bitbucket 存储库。 GitHub 这样做会暴露您的凭证，并可能危及您的账户安全。
>
> 在 EC2Amazon 实例上运行的配方可使用更好的方法 (IAM; 角色)，如 在 AWS OpsWorks Stacks Linux 实例上使用适用于 Ruby 的 SDK 中所述。
>
> 发送到 Amazon S3 存储桶的内容可能包含客户内容。有关删除敏感数据的更多信息，请参阅如何清空 S3 存储桶？或如何删除 S3 存储桶？。

如果您还没有适当的 用户，可创建一个此类用户，如下所示。有关更多信息，请参阅什么是 IAM。

> **Warning**
>
> IAM 用户拥有长期证书，这会带来安全风险。为帮助减轻这种风险，我们建议仅向这些用户提供执行任务所需的权限，并在不再需要这些用户时将其移除。

若要创建 IAM 用户

1. 登录 AWS Management Console 并打开 IAM 控制台，网址为 https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 Users 并在必要时选择 Add users 以创建新的管理用户。

3. 在 设置权限 页面上，选择 直接附加策略。
4. 在 Permissions policies 搜索框中键入 **S3** 以显示 Amazon S3 策略。

   选择亚马逊 S3 ReadOnlyAccess。如果您愿意，可以指定授予更广泛权限的策略，例如 AmazonS3 FullAccess，但标准做法是仅授予所需的权限。在此示例中，配方将仅下载一个文件，因此只读访问权就够用了。
5. 选择下一步。
6. 选择 Create user。
7. 为您的用户创建访问密钥。有关创建访问密钥的更多信息，请参阅 IAM 用户指南 中的管理 IAM 用户的访问密钥。

接下来，您必须提供要下载的文件。此示例假定您将名为 `myfile.txt` 的文件放入新创建的名为 `cookbook_bucket` 的 S3 存储桶中。

提供要下载的文件

1. 使用以下文本创建一个名为 `myfile.txt` 的文件并将该文件保存在工作站上的方便位置。

   ```
   This is the file that you just downloaded from Amazon S3.
   ```
2. 在 Amazon S3 控制台上，在 `cookbook_bucket`Standard 区域中创建一个名为 的存储桶并将 `myfile.txt` 上传到此存储桶。

按如下所示设置说明书。

设置说明书

1. 在 `opsworks_cookbooks` 中创建一个名为 `s3bucket` 的目录并导航到该目录。
2. 按照示例 1：安装软件包中所述初始化并配置 Test Kitchen。
3. 使用以下内容替换 `.kitchen.yml` 中的文本。

   ```
   ---
   driver:
     name: vagrant

   provisioner:
     name: chef_solo
   ```

```
  environments_path: ./environments

platforms:
  - name: ubuntu-14.04

suites:
  - name: s3bucket
    provisioner:
      solo_rb:
        environment: test
    run_list:
      - recipe[s3bucket::default]
    attributes:
```

4. 将以下两个目录添加到 s3bucket：recipes 和 environments。
5. 使用以下 default_attributes 部分创建名为 test.json 的环境文件，将 access_key 和 secret_key 值替换为您用户的对应密钥。将文件保存到说明书的 environments 文件夹。

```
{
  "default_attributes" : {
    "cookbooks_101" : {
      "access_key": "AKIAIOSFODNN7EXAMPLE",
      "secret_key" : "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"
    }
  },
  "chef_type" : "environment",
  "json_class" : "Chef::Environment"
}
```

可通过多种方法为实例上运行的配方提供凭证。关键注意事项是限制意外公开密钥和危及账户安全的机会。为此，建议不要在代码中使用显式密钥值。此示例将密钥值放入节点对象中，这将通过使用节点语法而不是公开文本值来允许配方引用密钥值。您必须拥有根特权才能访问节点对象，这限制了公开密钥的可能性。有关更多信息，请参阅管理 AWS 访问密钥的最佳实践。

> **Note**
>
> 请注意，此示例使用嵌套属性，并将 cookbooks_101 作为第一个元素。如果节点对象中有其他 access_key 或 secret_key 属性，此实践将限制名称冲突的可能性。

以下配方从 myfile.text 存储桶下载 cookbook_bucket。

```
gem_package "aws-sdk ~> 3" do
  action :install
end

ruby_block "download-object" do
  block do
    require 'aws-sdk'

    s3 = Aws::S3::Client.new(
          :access_key_id => "#{node['cookbooks_101']['access_key']}",
          :secret_access_key => "#{node['cookbooks_101']['secret_key']}")

    myfile = s3.bucket['cookbook_bucket'].objects['myfile.txt']
    Dir.chdir("/tmp")
    File.open("myfile.txt", "w") do |f|
      f.write(myfile.read)
      f.close
    end
  end
  action :run
end
```

此配方的第一部分安装 SDK for Ruby (gem 程序包)。gem_package 资源安装将由配方或其他应用程序使用的 gem。

> **Note**
>
> 您的实例通常有两个 Ruby 实例（通常版本不同）。一个是由 Chef 客户端使用的专用实例。另一个由实例上运行的应用程序和配方使用。安装 gem 程序包时务必了解此区别，因为提供了两种用于安装 gem 的资源 (gem_package 和 chef_gem)。如果应用程序或配方使用 gem 程序包，则使用 gem_package 安装它。chef_gem 仅适用于由 Chef 客户端使用的 gem 程序包。

其余配方为 ruby_block 资源，其中包含下载文件的 Ruby 代码。您可能会认为，由于配方是一个 Ruby 应用程序，因此您可以直接将代码放入配方。但是，Chef 运行将在执行任何资源之前编译所有代码。如果您将示例代码直接放入配方中，则 Ruby 在执行 `require 'aws-sdk'` 资源之前会尝试解析 gem_package 语句。由于 SDK for Ruby 尚未安装，因此编译将失败。

在执行资源之前不会编译 ruby_block 资源中的代码。在此示例中，将在 ruby_block 资源安装完 SDK for Ruby;之后执行 gem_package 资源，因此代码将成功运行。

ruby_block 中的代码将按如下所示运行。

1. 创建新的 Aws::S3 对象，该对象提供服务接口。

   访问密钥和私有密钥是通过引用存储在节点对象中的值来指定的。
2. 调用 S3 对象的 bucket.objects 关联，这将返回名为 myfile 的表示 myfile.txt 的 Aws::S3::Object 对象。
3. 使用 Dir.chdir 将工作目录设置为 /tmp。
4. 打开名为 myfile.txt 的文件，将 myfile 的内容写入此文件，然后关闭此文件。

运行配方

1. 使用示例配方创建一个名为 default.rb 的文件，并将该文件保存到 recipes 目录。
2. 运行 kitchen converge。
3. 运行 kitchen login 以登录实例，然后运行 ls /tmp。您应该会看到 myfile.txt 以及几个 Test Kitchen 文件和目录。

   ```
   vagrant@s3bucket-ubuntu-1204:~$ ls /tmp
   install.sh  kitchen  myfile.txt  stderr
   ```

   您还可运行 cat /tmp/myfile.txt 验证文件内容是否正确。

完成后，运行 kitchen destroy 以终止实例。

在 AWS OpsWorks Stacks Linux 实例上使用适用于 Ruby 的 SDK

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

本主题介绍如何在 AWS OpsWorks Stacks Linux 实例上使用适用于 Ruby 的软件开发工具包从 Amazon S3 存储桶下载文件。 AWS OpsWorks Stacks 会在每个 Linux 实例上自动安装适用于 Ruby 的 SDK。但是，当您创建服务的客户端对象时，必须为其他服务提供一组合适的 AWS 凭证 `AWS::S3.new` 或等效内容。

发送到 Amazon S3 存储桶的内容可能包含客户内容。有关删除敏感数据的更多信息，请参阅如何清空 S3 存储桶？或如何删除 S3 存储桶？。

在 Vagrant 实例上使用 SDK for Ruby说明如何通过将凭证存储在节点对象中并在配方代码中引用属性来减小公开凭证的风险。当您在 Amazon EC2 实例上运行配方时，更好的选择是 IAM 角色。

IAM 角色的工作原理与 IAM 用户的很类似。它具有授权使用各种 Amazon Web Service 的附加策略。但是，您将角色分配给 Amazon EC2 实例而不是个体。随后，此实例上运行的应用程序将获取附加策略所授予的权限。使用角色，凭证绝不会显示在您的代码中，甚至不会间接显示。本主题介绍如何使用 IAM 角色在 Amazon EC2 实例上运行 在 Vagrant 实例上使用 SDK for Ruby 中的配方。

您可使用 kitchen-ec2 驱动程序来运行此配方与 Test Kitchen，如示例 9：使用 Amazon EC2 实例中所述。但是，在 Amazon EC2 实例上安装适用于 Ruby 的 SDK 有些复杂，对于 AWS OpsWorks 堆栈，您无需担心。默认情况下，所有 AWS OpsWorks Stacks Linux 实例都安装了适用于 Ruby 的 SDK。因此，为简单起见，该示例使用了 AWS OpsWorks Stacks 实例。

第一步是设置 IAM 角色。此示例采用最简单的方法，即使用 AWS OpsWorks Stacks 在创建第一个堆栈时创建的 Amazon EC2 角色。它被命名为 `aws-opsworks-ec2-role`。但是， AWS OpsWorks Stacks 不会为该角色附加策略，因此默认情况下它不授予任何权限。

您必须将 `AmazonS3ReadOnlyAccess` 策略附加到 `aws-opsworks-ec2-role` 角色才能授予相应的权限。有关如何将策略附加到 IAM 实体的更多信息，请参阅IAM 用户指南中的添加 IAM 身份权限（控制台）。

在创建或更新堆栈时指定角色。使用自定义层设置堆栈，如在 Linux 实例上运行配方中所述，增加了一步。在添加堆栈页面上，确认默认 IAM 实例配置文件设置为aws-opsworks-ec双角色。 AWS OpsWorks 然后，堆栈会将该角色分配给堆栈的所有实例。

设置说明书的过程与在 Linux 实例上运行配方使用的过程类似。下面是一个简短摘要；您应参阅该示例以了解详细信息。

设置说明书

1. 创建一个名为 `s3bucket_ops` 的目录并导航到该目录。
2. 创建一个包含以下内容的 `metadata.rb` 文件，并将该文件保存到 `s3bucket_ops`。

```
name "s3bucket_ops"
version "0.1.0"
```

3. 在 recipes 中创建 s3bucket_ops 目录。
4. 创建包含以下配方的 default.rb 文件，并将该文件保存到 recipes 目录。

```
Chef::Log.info("******Downloading a file from Amazon S3.******")

ruby_block "download-object" do
  block do
    require 'aws-sdk'

    s3 = AWS::S3.new

    myfile = s3.buckets['cookbook_bucket'].objects['myfile.txt']
    Dir.chdir("/tmp")
    File.open("myfile.txt", "w") do |f|
      f.syswrite(myfile.read)
      f.close
    end
  end
  action :run
end
```

5. 创建 .zip 的 s3bucket_ops 存档，然后将该存档上传到 Amazon S3 存储桶。为简单起见，请公开存档，然后记录存档的 URL 以供将来使用。您也可以将说明书存储在私有 Amazon S3 存档或几个其他存储库类型中。有关更多信息，请参阅 说明书存储库。

此配方类似于上一示例使用的配方，但存在以下差别。

- 由于 AWS OpsWorks Stacks 已经安装了适用于 Ruby 的 SDK，因此该chef_gem资源已被删除。
- 配方不会将任何凭证传递到 AWS::S3.new。

  凭证将基于实例的角色自动分配到应用程序。
- 配方使用 Chef::Log.info 将消息添加到 Chef 日志。

按下面所示为本示例创建堆栈。您也可以使用现有 Windows 堆栈。只需更新说明书即可，如下文所述。

创建 堆栈

1. 打开 AWS OpsWorks Stacks 控制台，然后单击 Add Stack (添加堆栈)。
2. 指定以下设置，接受其他设置的默认值，然后单击 Add Stack。

   - 名称 - RubySDK
   - 默认 SSH 密钥 - Amazon EC2 密钥对

   如果您需要创建 Amazon EC2 密钥对，请参阅 Amazon EC2 密钥对。请注意，该密钥对必须属于与实例相同的 Amazon Web Services Region。本示例使用了默认的美国西部（俄勒冈州）区域。
3. 单击 Add a layer 并将采用以下设置的自定义层添加到堆栈。

   - 名称 - S3Download
   - 短名称 - s3download

   实际上任何层类型均适用于 Linux 堆栈，但本示例不需要由其他层类型安装的任何程序包，因此自定义层是最简单的方法。
4. 向层添加全天候实例 (采用默认设置) 并启动该实例。

您现在可以安装并运行配方

运行配方

1. 编辑堆栈以启用自定义说明书，然后指定以下设置。

   - 存储库类型 -Http 存档
   - 存储库 URL -您之前记录的说明书存档 URL。

   对其他设置使用默认值，然后单击 Save 更新堆栈配置。
2. 运行“Update Custom Cookbooks”堆栈命令，这会将当前版本的自定义说明书安装到堆栈的实例上。如果您的说明书存在早期版本，此命令会覆盖该版本。

3. 通过在 Recipes to execute 设置为 **`s3bucket_ops::default`** 的情况下运行 Execute Recipes 堆栈命令来执行配方。此命令将使用一个包含 `s3bucket_ops::default` 的运行列表来启动 Chef 运行。

> **Note**
>
> 通常，您可以让 AWS OpsWorks Stacks 自动运行您的配方，方法是将它们分配给相应的生命周期事件。您可以通过手动触发事件来运行此类配方。您可以使用堆栈命令触发设置和配置事件，使用部署命令触发部署和取消部署事件。

配方成功运行后，您可以对其进行验证。

验证 s3bucket_ops

1. 第一步是检查 Chef 日志。您的堆栈应有一个名为 opstest1 的实例。在 Instances 页面上，单击实例 Log 列中的 show，显示 Chef 日志。向下滚动以在底部附近查找您的日志消息。

```
...
[2014-07-31T17:01:45+00:00] INFO: Storing updated cookbooks/opsworks_cleanup/
attributes/customize.rb in the cache.
[2014-07-31T17:01:45+00:00] INFO: Storing updated cookbooks/opsworks_cleanup/
metadata.rb in the cache.
[2014-07-31T17:01:46+00:00] INFO: ******Downloading a file from Amazon S3.******
[2014-07-31T17:01:46+00:00] INFO: Processing template[/etc/hosts] action create
 (opsworks_stack_state_sync::hosts line 3)
...
```

2. 使用 SSH 登录实例并列出 `/tmp` 的内容。

在 AWS OpsWorks Stacks Windows 实例上使用适用于 Ruby 的 SDK

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 此示例假定您已完成在 Windows 实例上运行配方示例。如果没有，您应该先完成该示例。具体而言，它介绍了如何启用对实例的 RDP 访问。
> 发送到 Amazon S3 存储桶的内容可能包含客户内容。有关删除敏感数据的更多信息，请参阅如何清空 S3 存储桶？或如何删除 S3 存储桶？。

本主题介绍如何在 AWS OpsWorks Stacks Windows 实例AWS SDK for Ruby上使用从 S3 存储桶下载文件。

如果 Ruby 应用程序需要访问 AWS 资源，您必须向其提供一组具有适当权限的 AWS 凭证。对于配方，提供 AWS 证书的最佳选择是使用 AWS Identity and Access Management (I AM) 角色。IAM 角色的工作方式与 IAM 用户非常相似，它有一个附加的策略，用于授予使用各种 AWS 服务的权限。但是，您将角色分配给 Amazon Elastic Compute Cloud (Amazon EC2) 实例而不是个别用户。随后，此实例上运行的应用程序将获取附加策略所授予的权限。使用角色，凭证绝不会显示在您的代码中，甚至不会间接显示。

第一步是设置 IAM 角色。此示例采用最简单的方法，即使用 AWS OpsWorks Stacks 在创建第一个堆栈时创建的 Amazon EC2 角色。它被命名为 `aws-opsworks-ec2-role`。但是， AWS OpsWorks Stacks 不会为该角色附加策略，因此默认情况下它不授予任何权限。

您必须将 `AmazonS3ReadOnlyAccess` 策略附加到 `aws-opsworks-ec2-role` 角色才能授予相应的权限。有关如何将策略附加到 IAM 实体的更多信息，请参阅IAM 用户指南中的添加 IAM 身份权限（控制台）。

在创建或更新堆栈时指定角色。使用自定义层设置堆栈，如在 Windows 实例上运行配方中所述，增加了一步。在添加堆栈页面上，确认默认 IAM 实例配置文件设置为aws-opsworks-ec双角色。 AWS OpsWorks 然后，堆栈会将该角色分配给堆栈的所有实例。

设置说明书的过程与在 Linux 实例上运行配方使用的过程类似。以下是简短摘要；请参阅该示例以了解详细信息。

设置说明书

1. 创建一个名为 `s3bucket_ops` 的目录并导航到该目录。
2. 创建一个包含以下内容的 `metadata.rb` 文件，并将该文件保存到 `s3bucket_ops`。

```
name "s3download"
```

```
version "0.1.0"
```

3. 在 recipes 中创建 s3download 目录。
4. 创建包含以下配方的 default.rb 文件，并将该文件保存到 recipes 目录。将 *windows-cookbooks* 替换为您将用于存储要下载的文件的 S3 存储桶的名称。

```
Chef::Log.info("******Downloading an object from S3******")

chef_gem "aws-sdk-s3" do
  compile_time false
  action :install
end

ruby_block "download-object" do
  block do
    require 'aws-sdk-s3'

    Aws.use_bundled_cert!

    s3_client = Aws::S3::Client.new(region:'us-west-2')

    s3_client.get_object(bucket: 'windows-cookbooks',
                         key: 'myfile.txt',
                         response_target: '/chef/myfile.txt')
  end
  action :run
end
```

5. 创建 s3download的 .zip 存档，然后将该文件上传到 S3 存储桶。公开该文件并记录 URL 以供将来使用。
6. 创建一个名为 myfile.txt 的文本文件，然后将其上传到 S3 存储桶。这是您的配方将下载的文件，让您可以使用任何方便的存储桶。

该配方执行以下任务。

1: 安装 SDK for Ruby v2。

该示例使用 SDK for Ruby 来下载对象。但是， AWS OpsWorks Stacks 不会在 Windows 实例上安装此 SDK，因此配方的第一部分使用chef_gem资源来处理该任务。您可以使用此资源安装 Gem 以供 Chef 使用，其中包括配方。

2: 下载文件。

该配方的第三部分使用 ruby_block 资源运行 SDK for Ruby v2 代码，以将 myfile.txt 从名为 *windows-cookbooks* 的 S3 存储桶下载到实例的 /chef 目录。将 *windows-cookbooks* 更改为包含 myfile.txt 的存储桶的名称。

> **Note**
>
> 配方是一种 Ruby 应用程序，因此您可以将 Ruby 代码放在配方的正文中；它不一定要位于 ruby_block 资源中。但是，Chef 会首先在配方的正文中执行 Ruby 代码，然后按顺序执行每个资源。在本示例中，如果您将下载代码放在配方的正文中，则此操作会失败，因为它依赖于，并且安装开发工具包的 chef_gem 资源尚未执行。ruby_block 资源中的代码会在资源执行时执行，并且这将发生在 chef_gem 资源安装 SDK for Ruby 之后。

按下面所示为本示例创建堆栈。您也可以使用现有 Windows 堆栈。只需更新说明书即可，如下文所述。

创建堆栈

1. 打开 AWS OpsWorks Stacks 控制台，然后选择 Add Stack (添加堆栈)。指定以下设置，接受其他设置的默认值，然后选择 Add Stack。
   - 名称 - S3Download
   - 区域-美国西部（俄勒冈州）

     本示例在任何区域都有效，但我们建议将美国西部（俄勒冈州）用于教程。
   - 默认操作系统 - Microsoft Windows Server 2012 R2
2. 选择 Add a layer 并将采用以下设置的自定义层添加到堆栈。
   - 名称 - S3Download
   - 短名称 - s3download
3. 向 S3Download 层添加全天候实例 (采用默认设置) 并启动该实例。

您现在可以安装并运行配方

运行配方

1. 编辑堆栈以启用自定义说明书，然后指定以下设置。

   - 存储库类型 - S3 存档 。
   - 存储库 URL -您之前记录的说明书存档 URL。

   接受其他设置的默认值，然后选择 Save 更新堆栈配置。
2. 运行“Update Custom Cookbooks”堆栈命令，这会将最新版本的自定义说明书安装到堆栈的联机实例上。如果您的说明书存在早期版本，此命令会覆盖该版本。
3. 通过在 Recipes to execute 设置为 **`s3download::default`** 的情况下运行 Execute Recipes 堆栈命令来执行配方。此命令将使用一个包含 `s3download::default` 的运行列表来启动 Chef 运行。

   > **Note**
   >
   > 通常，您可以让 AWS OpsWorks Stacks 自动运行您的配方，方法是将它们分配给相应的生命周期事件。您也可以通过手动触发事件来运行此类配方。您可以使用堆栈命令触发设置和配置事件，使用部署命令触发部署和取消部署事件。

配方成功运行后，您可以对其进行验证。

验证 s3download

1. 第一步是检查 Chef 日志。您的堆栈应有一个名为 s3download1 的实例。在 Instances 页面上，选择实例 Log 列中的 show，显示 Chef 日志。向下滚动以在底部附近查找您的日志消息。

```
...
[2015-05-01T21:11:04+00:00] INFO: Loading cookbooks [s3download@0.0.0]
[2015-05-01T21:11:04+00:00] INFO: Storing updated cookbooks/s3download/recipes/
default.rb in the cache.
[2015-05-01T21:11:04+00:00] INFO: ******Downloading an object from S3******
[2015-05-01T21:11:04+00:00] INFO: Processing chef_gem[aws-sdk] action install
 (s3download::default line 3)
[2015-05-01T21:11:05+00:00] INFO: Processing ruby_block[download-object] action run
 (s3download::default line 8)
...
```

2. 使用 RDP 登录实例并检查 `c:\chef` 的内容。

安装 Windows 软件

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 这些示例假定您已完成在 Windows 实例上运行配方示例。如果没有，您应该先完成该示例。具体而言，它介绍了如何启用对实例的 RDP 访问。

Windows 实例是从 Windows Server 2012 R2 标准版开始引入的，因此，您通常需要安装一些软件。详细信息取决于软件的类型。

- Windows 功能是可选的系统组件 (包括 .NET 框架和 Internet Information Services (IIS))，您可将这些组件下载到您的实例。
- 第三方软件通常附带有安装程序包 (如 MSI 文件)，您必须将其下载到实例然后运行。

  某些 Microsoft 软件也附带有安装程序包。

本节将介绍如何实施说明书以安装 Windows 功能和程序包。它还将介绍 Chef Windows 说明书，其中包含用于简化 Windows 实例的配方的实施的资源和帮助程序函数。

主题

- 安装 Windows 功能：IIS
- 在 Windows 实例上安装程序包

安装 Windows 功能：IIS

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

Windows 功能是一组可选的系统组件，包括 .NET 框架和 Internet Information Services (IIS)。本主题介绍如何实施说明书来安装常用功能 Internet Information Services (IIS)。

> **Note**
>
> 安装程序包说明了如何安装附带有必须下载到实例并运行的安装程序包 (如 MSI 文件) 的软件。IIS 说明书

在 Windows 实例上运行配方说明了如何使用 `powershell_script` 资源安装 Windows 功能。本示例展示了一种替代方法：使用 Chef Windows 说明书的 `windows_feature` 资源。此说明书包含一系列使用部署映像服务和管理在 Windows 上执行各种任务 (包括功能安装) 的资源。

> **Note**
>
> Chef 还有一个可用于管理 IIS 的 IIS 说明书。有关更多信息，请参阅 IIS 说明书。

设置说明书

1. 前往 Windows 食谱 GitHub 存储库并下载`windows`食谱。

   本示例假定您将以 .zip 文件的形式下载 `windows` 存储库，但您也可以克隆该存储库 (如果您愿意)。
2. 前往 chef_handler 食谱 GitHub 存储库并下载食谱。`chef-handler`

   `windows` 说明书依赖于 `chef_handler`；您不会直接使用它。本示例假定您将以 .zip 文件的形式下载 `chef_handler` 存储库，但您也可以克隆该存储库 (如果您愿意)。
3. 将 `windows` 和 `chef_handler` 说明书分别提取到您的说明书目录中的名为 `windows` 和 `chef_handler` 的子目录中。

4. 在您的名为 `install-iis` 的说明书目录中创建一个目录并导航到该目录。
5. 将包含以下内容的 `metadata.rb` 文件添加到 `install-iis`。

```
name "install-iis"
version "0.1.0"

depends "windows"
```

`depends` 指令允许您在配方中使用 `windows` 说明书资源。

6. 将 `recipes` 目录添加到 `install-iis` 并将名为 `default.rb` 的文件添加到包含以下配方代码的目录。

```
%w{ IIS-WebServerRole IIS-WebServer }.each do |feature|
  windows_feature feature do
    action :install
  end
end

service 'w3svc' do
  action [:start, :enable]
end
```

配方将使用 `windows` 说明书的 `windows_feature` 资源安装以下内容：

1. IIS Web 服务器角色。
2. IIS Web 服务器。

配方随后使用 `service` 资源启动并启用 IIS 服务 (W3SVC)。

> **Note**
>
> 有关可用 Windows 功能的完整列表，请使用 RDP 登录实例，打开命令提示窗口，然后运行以下命令。请注意，该列表非常长。
>
> ```
> dism /online /Get-Features
> ```

7. 创建包含 `install-iis`、`chef_handler` 和 `windows` 说明书的 `.zip` 存档，并将存档上传到 S3 存储桶。公开该存档并记录 URL 以供将来使用。本示例假定该存档命名为 `install-iis.zip`。有关更多信息，请参阅 说明书存储库。

   发送到 Amazon S3 存储桶的内容可能包含客户内容。有关删除敏感数据的更多信息，请参阅如何清空 S3 存储桶？或如何删除 S3 存储桶？。

按下面所示为本示例创建堆栈。您也可以使用现有 Windows 堆栈。只需更新说明书即可，如下文所述。

创建堆栈

1. 打开 AWS OpsWorks Stacks 控制台，然后选择 Add Stack (添加堆栈)。指定以下设置，接受其他设置的默认值，然后选择 Add Stack。

   - 名称 - InstallIIS
   - 区域-美国西部（俄勒冈州）

     本示例在任何区域都有效，但我们建议将美国西部（俄勒冈州）用于教程。
   - 默认操作系统 - Microsoft Windows Server 2012 R2
2. 选择 Add a layer 并将采用以下设置的自定义层添加到堆栈。

   - 名称 - IIS
   - 短名称 - iis
3. 向 IIS 层添加全天候实例 (采用默认设置) 并启动该实例。

您现在可以安装说明书并运行配方

安装说明书并运行配方

1. 编辑堆栈以启用自定义说明书，然后指定以下设置。

   - 存储库类型 - S3 存档
   - 存储库 URL -您之前记录的说明书存档 URL。

   接受其他设置的默认值，然后选择 Save 更新堆栈配置。

2. 运行 Update Custom Cookbooks 堆栈命令，这会将最新版本的自定义说明书安装到堆栈的联机实例上。如果您的说明书存在早期版本，此命令会覆盖该版本。
3. 通过在 Recipes to execute 设置为 **`install-iis::default`** 的情况下运行 Execute Recipes 堆栈命令来执行配方。此命令将启动 Chef 运行，它将运行指定的配方。

> **Note**
>
> 为了方便起见，此示例使用了 Execute Recipes，但通常会 AWS OpsWorks 让 Stacks 通过将配方分配给相应的生命周期事件来自动运行配方。您可以通过手动触发事件来运行此类配方。您可以使用堆栈命令触发设置和配置事件，使用部署命令触发部署和取消部署事件。

4. 要确认安装，请使用 RDP 连接到该实例并打开 Windows 资源管理器。文件系统现在应有一个 `C:\inetpub` 目录。如果您查看管理工具控制面板应用程序中的服务列表，则会发现 IIS 在底部附近。但是，它将被命名为“World Wide Web Publishing Service”，而不是“IIS”。

在 Windows 实例上安装程序包

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 此示例假定您已完成在 Windows 实例上运行配方示例。如果没有，您应该先完成该示例。具体而言，它介绍了如何启用对实例的 RDP 访问。

如果您的程序附带有安装程序包 (如 MSI)，您必须将该文件下载到实例，然后运行它。本示例说明如何实施说明书来安装 MSI 程序包 (Python 运行时)，包括如何定义关联的环境变量。有关如何安装 Windows 功能 (如 IIS) 的更多信息，请参阅安装 Windows 功能：IIS。

设置说明书

1. 创建一个名为 `installpython` 的目录并导航到该目录。

2. 将包含以下内容的 metadata.rb 文件添加到 installpython。

```
name "installpython"
version "0.1.0"
```

3. 将 recipes 和 files 目录添加到 installpython，并将 default 目录添加到文件。
4. 从适用于 Windows 的 Python 版本将 Python 程序包下载到说明书的 files\default 目录。本示例将安装 3.5.0a3 的 Windows x86- 版本，该版本使用一个名为 python-3.4.3.amd64.msipython-64. 的 MSI 安装程序。
5. 使用以下配方代码将名为 default.rb 的文件添加到 recipes 目录。

```
directory 'C:\tmp' do
  rights :full_control, 'Everyone'
  recursive true
  action :create
end

cookbook_file 'C:\tmp\python-3.4.3.amd64.msi' do
  source "python-3.4.3.amd64.msi"
  rights :full_control, 'Everyone'
  action :create
end

windows_package 'python' do
  source 'C:\tmp\python-3.4.3.amd64.msi'
  action :install
end

env "PATH" do
  value 'c:\python34'
  delim ";"
  action :modify
end
```

此配方会执行以下操作：

1. 使用目录资源创建一个 C:\tmp 目录。

   有关此资源的更多信息，请参阅示例 3：创建目录。

2. 使用 cookbook_file 资源将安装程序从说明书的 `files\default` 目录复制到 `C:\tmp`。

   有关此资源的更多信息，请参阅从说明书安装文件。

3. 使用 windows_package 资源运行 MSI 安装程序，这会将 Python 安装到 `c:\python34`。

   安装程序将创建所需的目录并安装文件，但不会修改系统的 PATH 环境变量。

4. 使用 env 资源将 `c:\python34` 添加到系统路径。

   您可使用 env 资源定义环境变量。在本例中，配方允许您通过将 `c:\python34` 添加到该路径来从命令行轻松运行 Python 脚本。

   - 资源名称指定环境变量的名称，本例中为 PATH。
   - `value` 属性指定该变量的值，本例中为 `c:\\python34` (您需要转义 \ 字符)。
   - `:modify` 操作在该变量的当前值前加上指定的值。
   - `delim` 属性指定一个将新值与现有值分开的分隔符，本例中为 `;`。

6. 创建 `.zip` 的 `installpython` 存档，将该存档上传到 S3 存储桶，然后将其公开。记录该存档的 URL 以供将来使用。有关更多信息，请参阅 说明书存储库。

   发送到 Amazon S3 存储桶的内容可能包含客户内容。有关删除敏感数据的更多信息，请参阅如何清空 S3 存储桶？或如何删除 S3 存储桶？。

按下面所示为本示例创建堆栈。您也可以使用现有 Windows 堆栈。只需更新说明书即可，如下文所述。

创建堆栈

1. 打开 AWS OpsWorks Stacks 控制台，然后选择 Add Stack (添加堆栈)。指定以下设置，接受其他设置的默认值，然后选择 Add Stack。

   - 姓名 — InstallPython
   - 区域-美国西部（俄勒冈州）

     本示例在任何区域都有效，但我们建议将美国西部（俄勒冈州）用于教程。

   - 默认操作系统 - Microsoft Windows Server 2012 R2

2. 选择 Add a layer 并将采用以下设置的自定义层添加到堆栈。

   - 名称 - Python
   - 短名称 - python

3. 向 Python 层添加全天候实例 (采用默认设置) 并启动该实例。

在实例处于联机状态后，您可以安装说明书并运行配方

安装说明书并运行配方

1. 编辑堆栈以启用自定义说明书，然后指定以下设置。

   - 存储库类型 - S3 存档 。
   - 存储库 URL -您之前记录的说明书存档 URL。

   接受其他设置的默认值，然后选择 Save 更新堆栈配置。

2. 运行 Update Custom Cookbooks 堆栈命令，这会将最新版本的自定义说明书安装到堆栈的联机实例上。如果您的说明书存在早期版本，此命令会覆盖该版本。
3. 通过在 Recipes to execute 设置为 **`installpython::default`** 的情况下运行 Execute Recipes 堆栈命令来执行配方。此命令将使用一个包含 `installpython::default` 的运行列表来启动 Chef 运行。

   > **Note**
   >
   > 为了方便起见，此示例使用了 Execute Recipes，但通常会 AWS OpsWorks 让 Stacks 通过将配方分配给相应的生命周期事件来自动运行配方。您可以通过手动触发事件来运行此类配方。您可以使用堆栈命令触发设置和配置事件，使用部署命令触发部署和取消部署事件。

4. 要确认安装，请使用 RDP 连接到该实例并打开 Windows 资源管理器。

   - 文件系统现在应有一个 `C:\Python34` 目录。
   - 如果您从命令行运行 `path`，则它应该如下所示：`PATH=c:\python34;C:\Windows\system32;...`
   - 如果您从命令行运行 `python --version`，则它应返回 `Python 3.4.3`。

覆盖内置属性

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 本主题仅适用于 Linux 堆栈。您无法在 Windows 堆栈上覆盖内置属性。

AWS OpsWorks Stacks 会在每个实例上安装一组内置食谱。其中的许多内置说明书都支持内置层，而且其属性文件定义了各种默认系统和应用程序设置，如 Apache 服务器配置设置。通过将这些设置放在属性文件中，您可以采用以下任一方式覆盖对应的内置属性，从而自定义多种配置设置：

- 在自定义 JSON 中定义属性。

  此方法的优点是简单灵活。但是，您必须手动输入自定义 JSON，因此没有可靠的方法来管理属性定义。
- 在 `customize.rb` 属性文件中实施自定义说明书并定义属性。

  此方法比使用自定义 JSON 的灵活性更小，但更可靠，因为您可以将自定义说明书置于源代码管理下。

本主题介绍如何使用自定义说明书属性文件覆盖内置属性，例如使用 Apache 服务器。有关如何使用自定义 JSON 覆盖属性的更多信息，请参阅使用自定义 JSON。有关如何覆盖属性的一般性讨论，请参阅覆盖属性。

> **Note**
>
> 覆盖属性是自定义配置设置的首选方法，但设置并不总是由属性表示。在出现这种情况时，您通常可以通过覆盖内置配方用于创建配置文件的模板来自定义配置文件。有关示例，请参阅覆盖内置模板。

内置属性通常表示 Setup 配方用于创建配置文件的模板文件中的值。例如，一种 apache2 Setup 配方 default.rb 使用 apache2.conf.erb 模板创建 Apache 服务器的主配置文件 httpd.conf (Amazon Linux) 或 apache2.conf (Ubuntu)。下面是来自模板文件的摘录：

```
...
#
# MaxKeepAliveRequests: The maximum number of requests to allow
# during a persistent connection. Set to 0 to allow an unlimited amount.
# We recommend you leave this number high, for maximum performance.
#
MaxKeepAliveRequests <%= node[:apache][:keepaliverequests] %>
#
# KeepAliveTimeout: Number of seconds to wait for the next request from the
# same client on the same connection.
#
KeepAliveTimeout <%= node[:apache][:keepalivetimeout] %>
##
## Server-Pool Size Regulation (MPM specific)
##

...
```

在本示例中，KeepAliveTimeout 设置是 [:apache][:keepalivetimeout] 属性的值。此属性的默认值在 apache2 说明书的 apache.rb 属性文件中定义，如以下摘录所示：

```
...
# General settings
default[:apache][:listen_ports] = [ '80','443' ]
default[:apache][:contact] = 'ops@example.com'
default[:apache][:log_level] = 'info'
default[:apache][:timeout] = 120
default[:apache][:keepalive] = 'Off'
default[:apache][:keepaliverequests] = 100
default[:apache][:keepalivetimeout] = 3
...
```

> **Note**
>
> 有关常用内置属性的更多信息，请参阅内置说明书属性。

为了支持覆盖内置属性，所有内置说明书都要包含一个 customize.rb 属性文件，该文件将通过 include_attribute 指令集成到所有模块中。内置说明书的 customize.rb 文件未包含任何属性定义且对内置属性没有影响。要覆盖内置属性，您应使用与内置说明书相同的名称创建自定义说明书，然后将自定义属性定义放在同样名为 customize.rb 的属性文件中。该文件优先于内置版本且包含在任何相关模块中。如果您在 customize.rb 中定义了任何内置属性，则它们会覆盖对应的内置属性。

本示例说明如何覆盖内置 [:apache][:keepalivetimeout] 属性以将其值设置为 5 而不是 3。您可以对任何内置属性使用类似方法。但是，请注意您覆盖的属性。例如，在 opsworks 命名空间中覆盖属性可能会导致一些内置配方出现问题。

> **Important**
>
> 请勿通过修改内置属性文件本身的副本来覆盖内置属性。例如，您可以 将 apache.rb 的副本放在您的自定义说明书的 apache2/attributes 文件夹中并修改此文件的某些设置。但是，此文件优先于内置版本，而且内置配方现在将使用您的 apache.rb 版本。如果 AWS OpsWorks Stacks 稍后修改了内置apache.rb文件，则除非您手动更新版本，否则配方将不会获得新值。通过使用customize.rb，您只能覆盖指定的属性；内置配方会继续自动获取您尚未覆盖的每个属性的 up-to-date 值。

要开始，请创建一个自定义说明书。

创建说明书

1. 在您的 opsworks_cookbooks 目录中，创建名为 apache2 的说明书目录并导航到该目录。

   要覆盖内置属性，自定义说明书必须具有与内置说明书相同的名称，本例中为 apache2。
2. 在 apache2 目录中创建一个 attributes 目录。
3. 将一个名为 customize.rb 的文件添加到 attributes 目录并使用该文件定义要覆盖的内置说明书属性。在本示例中，该文件应包含以下内容：

   ```
   normal[:apache][:keepalivetimeout] = 5
   ```

> **Important**
>
> 要覆盖内置属性，自定义属性必须为 `normal` 类型或更高类型且具有与对应的内置属性完全相同的节点名称。`normal` 类型可确保自定义属性优先于内置属性，后者都是 `default` 类型。有关更多信息，请参阅 属性优先顺序。

4. 创建 `opsworks_cookbooks` 的名为 `opsworks_cookbooks.zip` 的 `.zip` 存档，并将该存档上传到 Amazon Simple Storage Service (Amazon S3) 存储桶。为简单起见，请将此文件设置为公用。记录 URL 以供将来使用。您也可以将说明书存储在私有 Amazon S3 存档或其他存储库类型中。有关更多信息，请参阅 说明书存储库。

   发送到 Amazon S3 存储桶的内容可能包含客户内容。有关删除敏感数据的更多信息，请参阅如何清空 S3 存储桶？或如何删除 S3 存储桶？。

要使用自定义属性，请创建堆栈并安装说明书。

使用自定义属性

1. 打开 AWS OpsWorks Stacks 控制台，然后选择 Add Stack (添加堆栈)。
2. 指定以下标准设置。

   - 姓名 — ApacheConfig
   - 区域-美国西部（俄勒冈州）

     虽然您可将堆栈放在任何区域中，但在教程中，建议您放在美国西部（俄勒冈州）中。
   - 默认 SSH 密钥 - EC2 密钥对

     如果您需要创建 EC2 密钥对，请参阅 Amazon EC2 密钥对。请注意，该密钥对必须属于与堆栈相同的 Amazon Web Services Region。

   选择 Advanced>>，将 Use custom Chef cookbooks 设置为 Yes，然后指定以下设置。

   - 存储库类型 -Http 存档
   - 存储库 URL -您之前记录的说明书存档 URL。

   接受其他设置的默认值，然后选择 Add Stack 创建堆栈。

> **Note**
>
> 本示例使用默认操作系统 Amazon Linux。如果您愿意，则可以使用 Ubuntu。唯一区别是：在 Ubuntu 系统上，内置 Setup 配方会生成一个具有相同设置的名为 `apache2.conf` 的配置文件并将该文件放在 `/etc/apache2` 目录中。

3. 选择 Add a layer，然后将采用默认设置的 ava App Server 层添加到堆栈。
4. 向层添加全天候实例 (采用默认设置) 并启动该实例。

   本示例中，一个 t2.micro 实例已足够。
5. 在实例处于联机状态后，使用 SSH 连接到它。`httpd.conf` 文件位于 `/etc/httpd/conf` 目录中。如果您检查该文件，则应看到您的自定义 `KeepAliveTimeout` 设置。设置的其余部分将具有内置 `apache.rb` 文件中的默认值。`httpd.conf` 的相关部分应类似于以下内容：

```
...
#
# KeepAliveTimeout: Number of seconds to wait for the next request from the
# same client on the same connection.
#
KeepAliveTimeout 5
...
```

覆盖内置模板

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 本主题仅适用于 Linux 堆栈。您无法在 Windows 堆栈上覆盖内置模板。

AWS OpsWorks Stacks 内置配方使用模板在实例上创建文件，主要是服务器（例如 Apache）的配置文件。例如，apache2 配方使用 apache2.conf.erb 模板创建 Apache 服务器的主配置文件 httpd.conf (Amazon Linux) 或 apache2.conf (Ubuntu)。

这些模板中的大部分配置设置由属性表示，因此自定义配置文件的首选方法是覆盖相应的内置属性。有关示例，请参阅覆盖内置属性。但是，如果您要自定义的设置不是由内置属性表示的，或根本不在模板中，则您必须覆盖模板本身。本主题介绍如何覆盖内置模板以指定自定义 Apache 配置设置。

您可以通过将 ErrorDocument 设置添加到 httpd.conf 文件来提供对 Apache 的自定义错误响应。apache2.conf.erb 仅包含一些注释掉的示例，如下所示：

```
...
#
# Customizable error responses come in three flavors:
# 1) plain text 2) local redirects 3) external redirects
#
# Some examples:
#ErrorDocument 500 "The server made a boo boo."
#ErrorDocument 404 /missing.html
#ErrorDocument 404 "/cgi-bin/missing_handler.pl"
#ErrorDocument 402 http://www.example.com/subscription_info.html
...
```

由于这些设置是硬编码注释，因此您无法通过覆盖属性来指定自定义值；您必须覆盖模板本身。但是，与属性不同，您无法通过任何方法覆盖模板文件的特定部分。您必须使用与内置版本相同的名称创建自定义说明书，将模板文件复制到同一子目录，然后根据需要修改文件。本主题说明如何覆盖 apache2.conf.erb 以提供对错误 500 的自定义响应。有关覆盖模板的一般性讨论，请参阅使用自定义模板。

> **Important**
>
> 当您覆盖内置模板时，内置配方将使用您的模板的自定义版本而不是内置版本。如果 AWS OpsWorks Stacks 更新内置模板，则自定义模板将不同步，可能无法正常工作。 AWS OpsWorks Stacks 不经常进行此类更改，当模板发生更改时， AWS OpsWorks Stacks 会列出更改，并允许您选择升级到新版本。建议您监控 AWS OpsWorks Stacks 存储库的更改并根据需要手动更新您的自定义模板。请注意，该存储库对于每个支持的 Chef 版本都有一个单独的分支，因此，请确保您位于正常的分支中。

要开始，请创建一个自定义说明书。

创建说明书

1. 在 opsworks_cookbooks 目录中，创建名为 apache2 的说明书目录，然后导航到该目录。要覆盖内置模板，自定义说明书必须具有与内置说明书相同的名称，本例中为 apache2。

   > **Note**
   >
   > 如果您已完成覆盖内置属性演练，则可在本例中使用相同的 apache2 说明书，然后跳过步骤 2。

2. 创建一个包含以下内容的 metadata.rb 文件，并将该文件保存到 apache2 目录。

   ```
   name "apache2"
   version "0.1.0"
   ```

3. 在 apache2 目录中创建一个 templates/default 目录。

   > **Note**
   >
   > templates/default 目录适用于 Amazon Linux 实例，这些实例使用默认 apache2.conf.erb 模板。Ubuntu 14.04 实例使用特定于操作系统的 apache2.conf.erb 模板，该模板位于 templates/ubuntu-14.04 目录中。如果您希望自定义项也应用于 Ubuntu 14.04 实例，则还必须覆盖该模板。

4. 将内置apache2.conf.erb模板复制到您的 templates/default 目录。打开模板文件，取消注释 ErrorDocument 500 行，然后提供自定义错误消息，如下所示：

   ```
   ...
   ErrorDocument 500 "A custom error message."
   #ErrorDocument 404 /missing.html
   ...
   ```

5. 创建 opsworks_cookbooks 的名为 opsworks_cookbooks.zip 的 .zip 存档，并将文件上传到 Amazon Simple Storage Service (Amazon S3) 存储桶。为简单起见，请公开该存档。记录该存档的 URL 以供将来使用。您也可以将说明书存储在私有 Amazon S3 存档或其他存储库类型中。有关更多信息，请参阅 说明书存储库。

发送到 Amazon S3 存储桶的内容可能包含客户内容。有关删除敏感数据的更多信息，请参阅如何清空 S3 存储桶？或如何删除 S3 存储桶？。

> **Note**
>
> 为简单起见，本示例向该模板添加了一个硬编码的错误消息。要更改它，您必须修改该模板并重新安装说明书。要为自己提供更大的灵活性，您可以在自定义说明书的 属性文件中为错误字符串定义一个默认自定义属性customize.rb，然后将该属性的值分配到 ErrorDocument 500。例如，如果您将属性命名为 [:apache][:custom][:error500]，则 apache2.conf.erb 中的对应行将类似于以下内容：
>
> ```
> ...
> ErrorDocument 500 <%= node[:apache][:custom][:error500] %>
> #ErrorDocument 404 /missing.html
> ...
> ```
>
> 您随后可以通过覆盖 [:apache][:custom][:error500] 来随时更改自定义错误消息。如果您使用自定义 JSON 覆盖属性，则您甚至不需要接触说明书。

要使用自定义模板，请创建一个堆栈并安装说明书。

使用自定义模板

1. 打开 AWS OpsWorks Stacks 控制台，然后选择 Add Stack (添加堆栈)。
2. 指定以下标准设置：

   - 姓名 — ApacheTemplate
   - 区域-美国西部（俄勒冈州）
   - 默认 SSH 密钥 — Amazon Elastic Compute Cloud（Amazon EC2）密钥对

     如果您需要创建 Amazon EC2 密钥对，请参阅 Amazon EC2 密钥对。请注意，该密钥对必须属于与实例相同的 Amazon Web Services Region。

   选择 Advanced>>，然后选择 Use custom Chef cookbooks 指定以下设置：

- 存储库类型 -Http 存档
- 存储库 URL -您之前记录的说明书存档 URL。

接受其他设置的默认值，然后选择 Add Stack 创建堆栈。

3. 选择 Add a layer，然后将采用默认设置的 Java App Server 层添加到堆栈。
4. 向层添加全天候实例 (采用默认设置) 并启动该实例。

   本示例中，一个 t2.micro 实例已足够。
5. 在实例处于联机状态后，使用 SSH 连接到它。`httpd.conf` 文件位于 `/etc/httpd/conf` 目录中。该文件应包含您的自定义 `ErrorDocument` 设置，后者将类似于以下内容：

```
...
# Some examples:
ErrorDocument 500 "A custom error message."
#ErrorDocument 404 /missing.html
#ErrorDocument 404 "/cgi-bin/missing_handler.pl"
#ErrorDocument 402 http://www.example.com/subscription_info.html
...
```

负载均衡一个层

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

AWS OpsWorks Stacks 提供了两个负载平衡选项，即 Ela stic Lo ad Balancing 和 HAProxy，它们通常用于在应用服务器层的实例之间平衡负载。本主题介绍这两个选项各自的优点和限制，帮助您决定在向一个层添加负载均衡时应该选择哪个选项。在某些情况下，最佳方法是同时使用这两个选项。

SSL 终止

嵌入式 HAProxy 层并不能处理 SSL 终止，您必须在服务器上终止 SSL。这种方法的好处是，通信内容在抵达服务器之前都是加密的。不过，服务器必须处理解密工作，这会增加服务器负载。另外，您必须将您的 SSL 证书放到应用服务器上，这样用户访问起来才更容易。

借助 Elastic Load Balancing，您可以在负载均衡器上终止 SSL。这会减少您应用服务器上的负载，但负载均衡器和服务器之间的通信不会经过加密。Elastic Load Balancing 还使您可以在服务器上终止 SSL，但是设置起来有点儿复杂。

扩展

如果输入通信量超过 HAProxy 负载均衡器的处理能力，则您必须手动提高其处理能力。

Elastic Load Balancing 会自动扩展以处理传入流量。要确保 Elastic Load Balancing 负载均衡器一上线就有足够能力处理预期的负载，您可以对其进行预热。

负载均衡器故障

如果托管您的 HAProxy 服务器的实例发生故障，则会使您的整个站点处于离线状态，直至您重新启动该实例。

Elastic Load Balancing 比 HAProxy 更具抗故障能力。例如，它会在已注册了 EC2 实例的各个可用区中配置负载均衡节点。如果在某区的服务中断，其他节点会继续处理输入通信量。有关更多信息，请参阅 Elastic Load Balancing Concepts。

空闲超时

如果服务器空闲时间超过指定的空闲超时值，这两个负载均衡器都会终止连接。

- HAProxy -空闲超时值没有上限。
- Elastic Load Balancing -空闲超时默认值为 60 秒，最高可达 3600 秒（60 分钟）。

Elastic Load Balancing 空闲时间限制对大多数目的来说是充足的。如果您需要更长时间的空闲超时，我们建议使用 HAProxy。例如：

- 长期运行的 HTTP 连接，可用于推送通知。
- 管理界面，可用于执行可能需要 60 分钟以上时间的任务。

基于 URL 的映射

您可能想要让负载均衡器基于输入请求的 URL 将该请求传递到特定的服务器。例如，假定您有十台应用服务器，这些服务器支持电子商务应用。其中八台服务器处理目录事项，剩余两台处理支付

事项。您希望根据请求 URL 将所有与支付相关的 HTTP 请求定向到支付服务器。在这种情况下，您将所有包含“支付”或“结账”的 URL 定向到其中一个支付服务器。

借助 HAProxy，您可以使用基于 URL 映射，将包含指定字符串的 URL 定向到特定服务器。要在 AWS OpsWorks Stacks 中使用基于 URL 的映射，必须通过覆盖内置食谱中的`haproxy-default.erb`模板来创建自定义 HAProxy 配置文件。`haproxy`有关更多信息，请参阅 HAProxy Configuration Manual 和使用自定义模板。您不可以对 HTTPS 请求使用基于 URL 的映射。HTTPS 请求是经过加密的，所以 HAProxy 无法检查请求 URL。

Elastic Load Balancing 对 URL 映射的支持有限。有关更多信息，请参阅 Elastic Load Balancing 的侦听器配置。

建议：我们建议使用 Elastic Load Balancing 来保持负载均衡，除非您有仅可由 HAProxy 处理的要求。在这种情况下，最佳方法可能是将 Elastic Load Balancing 用作可将输入通信量分发给一组 HAProxy 服务器的前端负载均衡器，从而将它与 HAProxy 这两者结合起来使用。要实现此目的，应按照以下步骤进行：

- 在您的各个堆栈的可用区设置 HAProxy 实例，从而将请求分发给这些区的应用服务器。
- 将 HAProxy 实例分配给 Elastic Load Balancing 负载均衡器，该负载均衡器然后会输入请求分发给 HAProxy 负载均衡器。

借助这种方法，您可以使用 HAProxy 的基于 URL 的映射，将不同类型的请求分发给适当的应用服务器。不过，如果其中一个 HAProxy 服务器离线，站点将继续正常运行，因为 Elastic Load Balancing 负载均衡器会自动将输入通信量分发给正常运转的 HAProxy 服务器。请注意，您必须将 Elastic Load Balancing 用作前端负载均衡器；HAProxy 服务器无法将请求分发给其他 HAProxy 服务器。

从 Chef 服务器迁移到 AWS OpsWorks 堆栈

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

由于 AWS OpsWorks Stacks 基于 Chef，因此从 Chef Server 迁移到 AWS OpsWorks Stacks 相对简单。本主题提供了修改 Chef Server 代码以用于 AWS OpsWorks Stacks 的指南。

> **Note**
>
> 我们不建议使用版本低于 11.10 的 Chef 迁移到堆栈，这些版本基于 chef-solo，不支持搜索或数据包。

主题

将角色映射到层

Chef Server 使用角色来表示和管理具有相同用途和配置的实例，例如每个实例都托管了一个 Java 应用程序服务器的一组实例。AWS OpsWorks Stacks 层的作用与 Chef 角色的基本相同。一个层是一个蓝图，可用于创建一组具有相同配置、已安装软件包、应用程序部署过程等的 Amazon Elastic Compute Cloud (Amazon EC2) 实例。

AWS OpsWorks 堆栈包括一组适用于多种应用服务器的内置层、一个 HAProxy 负载均衡器、一个 MySQL 数据库主服务器和 Ganglia 监控主服务器。例如，内置的 Java App Server 层是用于创建托管 Tomcat 服务器的实例的蓝图。

要迁移到 AWS OpsWorks Stacks，您需要将每个角色与提供同等功能的层相关联。对于一些角色，您可能只需要使用其中一个内置层。其他角色可能需要进行不同程度的自定义。首先需要检查内置层的功能，包括与各层关联的配方，以查看是否有某个层至少提供了您的角色的一些功能。有关内置层的更多信息，请参阅图层和AWS OpsWorks 堆栈图层参考。要查看内置配方，请参阅 AWS OpsWorks Stacks 公共 GitHub存储库。

接下来如何继续取决于层与各个角色之间的匹配程度，如下所示。

一个内置层支持角色的所有功能

如果需要，您可以进行最少的自定义来直接使用内置层。例如，如果某个角色支持 Tomcat 服务器，则 Java App Server 层的配方可能已经处理了角色的所有任务，可能还有一些适当的自定义。例如，您可以通过覆盖相应的属性或模板，让层的内置配方使用自定义 Tomcat 或 Apache 配置设置。

一个内置层支持部分而非全部角色功能。

您也许可以通过扩展层来使用内置层。这通常涉及到实施自定义配方来支持缺少的功能，以及将配方分配到层的生命周期事件中。例如，假设您的角色在托管 Tomcat 服务器的同一个实例上安装了 Redis 服务器。您可以实施自定义的配方，在该层的实例上安装 Redis 并将配方分配到该层的“Setup”事件，从而扩展 Java App Server 层以匹配角色的功能。

没有内置层能够为角色功能提供充分支持

实施自定义层。例如，假设您的角色支持 MongoDB 数据库服务器，但任何内置层不支持该服务器。您可以通过实施配方来安装所需的软件包、配置服务器等并将配方分配到自定义层的生命周期事件，从而提供该支持。通常，您可以至少将角色的一些配方用于此目的。有关如何实施自定义层的更多信息，请参阅创建自定义 Tomcat 服务器层。

使用数据包

Chef Server 允许您通过数据包将用户定义的数据传递到配方。

- 您将数据与说明书保存在一起，Chef 将它安装在各个实例上。
- 您可以将加密的数据包用于敏感数据，例如密码。

AWS OpsWorks Stacks 支持数据包；配方可以使用与 Chef Server 完全相同的代码来检索数据。但是，该支持具有以下限制和区别：

- 只有 Chef 11.10 Linux 和更高版本的堆栈上支持数据包。

  运行较早版本 Chef 的 Windows 堆栈和 Linux 堆栈不支持数据包。
- 您不能将数据包存储在说明书存储库中。

  相反，您使用自定义 JSON 来管理数据包的数据。
- AWS OpsWorks 堆栈不支持加密的数据包。

  如果您需要以加密形式存储敏感数据，例如密码或证书，我们建议将其存储在私有 S3 存储桶中。然后，您可以创建一个使用适用于 Ruby 的 Amazon SDK 来检索数据的自定义配方。有关示例，请参阅使用适用于 Ruby 的 SDK。

有关更多信息，请参阅 使用数据包。

使用 Chef 搜索

Chef Server 将 IP 地址和角色配置等堆栈配置信息存储在服务器上。食谱使用 Chef 搜索来检索这些数据。 AWS OpsWorks Stacks 使用的方法略有不同。例如，Chef 11.10 Linux 堆栈基于 Chef Client 本地模式，这是一个在实例上本地运行 Chef Server 的轻量级版本 (通常称为 Chef Zero) 的 Chef Client 选项。Chef Zero 支持搜索存储在实例的节点对象中的数据。

Stack AWS OpsWorks s 不会将堆栈数据存储在远程服务器上，而是为每个生命周期事件的每个实例的节点对象添加一组堆栈配置和部署属性。这些属性表示堆栈配置的快照。它们使用与 Chef Server 相同的语法，提供配方从服务器检索所需的大部分数据。

你通常不需要修改食谱中依赖于搜索的 Stacks 代码。 AWS OpsWorks 由于 Chef 搜索对节点对象（包括堆栈配置和部署属性）进行操作，因此 AWS OpsWorks Stacks 中的搜索查询通常与 Chef Server 中的搜索查询完全相同。

主要的例外是由于堆栈配置和部署属性仅包含 AWS OpsWorks Stacks 在实例上安装属性时知道的数据。如果您在特定实例上本地创建或修改属性，则这些更改不会传播回 AWS OpsWorks 堆栈，也不会合并到安装在其他实例上的堆栈配置和部署属性中。您可以使用搜索来仅检索该实例上的属性值。有关更多信息，请参阅 使用 Chef 搜索。

为了与 Chef Server 兼容， AWS OpsWorks Stacks 向节点对象添加了一组`role`属性，每个属性都包含堆栈的一个层属性。如果您的配方使用 `roles` 作为搜索关键字，则无需更改搜索代码。该查询自动返回对应层的数据。例如，以下查询均返回 `php-app` 层的属性。

```
phpserver = search(:node, "layers:php-app").first
```

```
phpserver = search(:node, "roles:php-app").first
```

管理说明书和配方

AWS OpsWorks Stacks 和 Chef Server 处理食谱和食谱的方式略有不同。对于 Chef Server：

- 您提供所有说明书，可以自行实施，也可以使用社区说明书。
- 您在服务器上存储说明书。
- 您可以手动或定期执行配方。

使用 AWS OpsWorks 堆栈：

- AWS OpsWorks Stacks 为每个内置图层提供一本或多本食谱。这些说明书处理标准任务，例如安装和配置内置层的软件以及部署应用程序。

  要处理并非由内置说明书执行的任务，您需要添加自定义说明书到堆栈或使用社区说明书。
- 您可以将 AWS OpsWorks Stacks 食谱存储在远程存储库中，例如 S3 存储桶或 Git 存储库。

  有关更多信息，请参阅 存储说明书。
- 您可以手动执行配方，但通常会让 AWS OpsWorks Stacks 为您执行配方，以响应在实例生命周期关键时刻发生的一系列生命周期事件。

  有关更多信息，请参阅 执行配方。
- AWS OpsWorks Stacks 仅支持 Chef 11.10 堆栈上的 Berkshelf。如果您使用 Berkshelf 来管理自己的说明书依赖项，则不能使用运行 Chef 11.4 或更低版本的堆栈。

  有关更多信息，请参阅 使用 Berkshelf。

主题

存储说明书

使用 Chef Server，您可以将说明书存储在服务器上并将其从服务器部署到实例。使用 AWS OpsWorks Stacks，您可以将食谱存储在存储库中，即 S3 或 HTTP 存档或 Git 或 Subversion 存储库。安装食谱时，您可以指定 AWS OpsWorks Stacks 将代码从存储库下载到堆栈实例所需的信息。

要从 Chef Server 迁移，您必须将说明书放在这些存储库之一中。有关如何构造说明书存储库结构的信息，请参阅说明书存储库。

执行配方

在 AWS OpsWorks Stacks 中，每个层都有一组生命周期事件（设置、配置、部署、取消部署和关闭），每个事件都发生在实例生命周期的关键时刻。要执行自定义配方，您通常需要将其分配到相应层上的相应事件。发生事件时， AWS OpsWorks Stacks 运行关联的配方。例如，“Setup”事件在实例完成启动后发生，因此您通常可以将执行诸如安装和配置软件包并启动服务等任务的配方分配到此事件。

您可以使用 Execute Recipes 堆栈命令手动执行配方。此命令对于开发和测试非常有用，不过您还可以将其用于执行不映射到生命周期事件的配方。您还可以使用 Execute Recipes 命令手动触发设置和配置事件。

除了 AWS OpsWorks Stacks 控制台之外，您还可以使用 AWS CLI 或软件开发工具包来执行配方。这些工具支持所有 AWS OpsWorks Stacks API 操作，不过比 API 更容易使用。使用 create-deployment CLI 命令可触发生命周期事件，这将运行所有关联的配方。您还可以使用此命令来执行一个或多个配方而不触发事件。等效的开发工具包代码取决于特定语言，不过通常类似于 CLI 命令。

以下示例介绍了使用 `create-deployment` CLI 命令自动完成应用程序部署的两种方式。

- 通过添加具有单个实例的自定义层到您的堆栈，定期部署您的应用程序。

  将自定义设置配方添加到在实例上创建 `cron` 作业的层，以按照指定计划运行命令。有关如何使用配方创建 `cron` 作业的示例，请参阅在 Linux 实例上运行 Cron 作业。
- 将任务添加到使用 `create-deployment` CLI 命令部署应用程序的连续集成管道。

使用 Chef 环境

AWS OpsWorks Stacks 不支持 Chef 环境；`node.chef_environment`总是会返回`_default`。

AWS OpsWorks 堆栈图层参考

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

AWS OpsWorks Stacks 部署的每个实例都必须是至少一个层的成员，该层定义了实例在堆栈中的角色，并控制设置和配置实例、安装软件包、部署应用程序等的细节。有关如何使用 AWS OpsWorks 堆栈创建和管理图层的更多信息，请参阅图层。

每个图层描述都包含 AWS OpsWorks Stacks 为图层的每个生命周期事件运行的内置配方列表。这些配方存储在 https://github.com/aws/opsworks-cookbooks。请注意，这些列表仅包含那些由 AWS OpsWorks Stacks 直接运行的食谱。这些配方有时会运行列表中未列出的从属配方。要查看特定事件的完整配方列表 (包括从属配方和自定义配方)，请检查相应的生命周期事件的 Chef 日志中的运行列表。

主题

HAProxy 层参考

Important

该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

Note

此层仅适用于基于 Linux 的堆栈。

HAProxy 层使用 HAProxy (一种可靠的高性能 TCP/HTTP 负载均衡器) 来为基于 TCP 和 HTTP 的应用程序提供高可用性负载均衡和代理服务。它特别适用于必须在负载非常高的情况下进行爬行且需要持久性或第 7 层处理的网站。

HAProxy 监控流量并显示网页上关联实例的统计信息和状态。默认情况下，URI 是 http://*DNSName*/haproxy?stats，其中 *DNSName* 是 实例的 DNS 名称。

短名称：lb

兼容性：HAProxy 层与以下层兼容：自定义、db-master 和 memcached。

开放端口：HAProxy 允许公开访问端口 22 (SSH)、80 (HTTP) 和 443 (HTTPS)。

自动分配弹性 IP 地址：默认情况下开启

默认 EBS 卷：否

默认安全组：AWS OpsWorks--LB-Server

配置：要配置 HAProxy 层，您必须指定以下内容：

- 状态检查 URI (默认情况下为：http://*DNSName*/)。
- 统计 URI (默认情况下为：http://*DNSName*/haproxy?stats)。
- 统计密码 (可选)。
- 状态检查方法 (可选)。默认情况下，HAProxy 使用 HTTP OPTIONS 方法。您还可以指定 GET 或 HEAD。
- 启用统计数据 (可选)。
- 端口。默认情况下， AWS OpsWorks Stacks 将 HAProxy 配置为同时处理 HTTP 和 HTTPS 流量。您可以通过覆盖 Chef 配置模板 `haproxy.cfg.erb`，将 HAProxy 配置为仅处理其中一种或另一种流量。

Setup 配方：

- opsworks_initial_setup
- ssh_host_keys
- ssh_users
- mysql::client
- dependencies
- ebs
- opsworks_ganglia::client
- haproxy

Configure 配方：

- opsworks_ganglia::configure-client
- ssh_users

- agent_version
- haproxy::configure

Deploy 配方：

- deploy::default
- haproxy::configure

Shutdown 配方：

- opsworks_shutdown::default
- haproxy::stop

安装:

- AWS OpsWorks Stacks 使用实例的软件包安装程序将 HAProxy 安装到其默认位置。
- 您必须设置 syslog，以将日志文件指向指定位置。有关更多信息，请参阅 HAProxy。

HAProxy AWS OpsWorks 堆栈层

> **Note**
>
> 该层仅适用于 Chef 11 和更早的基于 Linux 的堆栈。

AWS OpsWorks Stacks HAProxy 层是一个 AWS OpsWorks 堆栈层，它为托管 HAProxy 服务器的实例提供了蓝图，这是一种可靠的高性能 TCP/ HTTP 负载平衡。一个小型实例通常足以处理全部应用程序服务器流量。

> **Note**
>
> 堆栈限于单个区域。要跨多个区域分发应用程序，必须为每个区域创建一个单独的堆栈。

创建 HAProxy 层

1. 在导航窗格中，单击 Layers (层)。

2. 在 Layers 页面上，单击 Add a Layer 或 + Layer。对于 Layer type，选择 HAProxy。

该层具有以下配置设置，且全部为可选。

HAProxy statistics

该层是否收集并显示统计信息。默认值是 Yes。

Statistics URL

统计信息页面的 URL 路径。完整的 URL 是 http://*dnsName StatisticsPath*，其中 *dnsName* ###### *DNS* ##。默认*StatisticsPath*值是 /haproxy？ 统计数据，对应于：http://ec2-54-245-151-7.us-west-2.compute.amazonaws.com/haproxy?stats。

Statistics user name

统计信息页面的用户名，必须提供该用户名才能查看统计信息页面。默认值为 “Opsworks”。

Statistics password

统计信息页面的密码，必须提供该密码才能查看统计信息页面。默认值是一个随机生成的字符串。

Health check URL

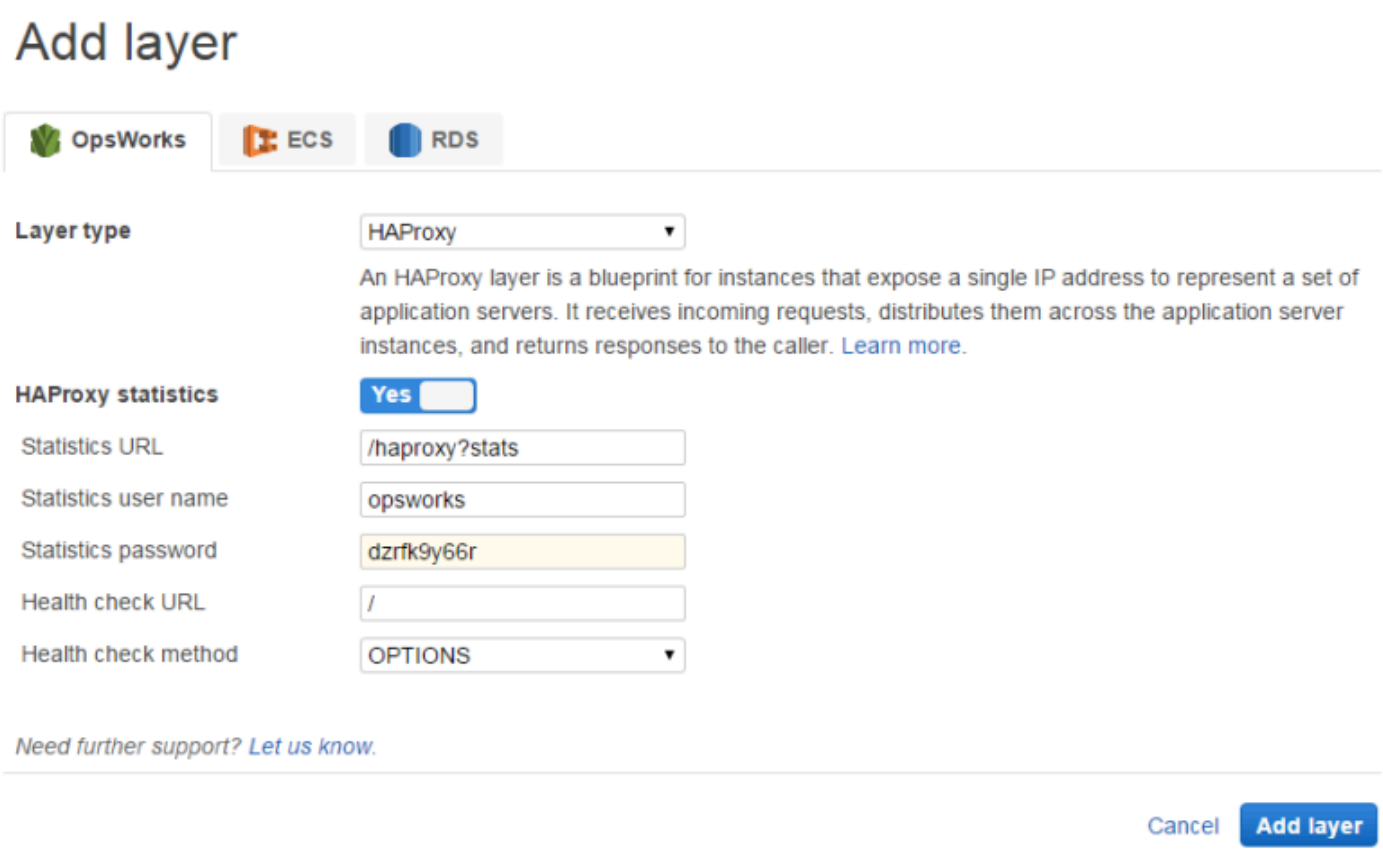
运行状况检查 URL 的后缀。HAProxy 使用此 URL 定期在每个应用程序服务器实例上调用 HTTP 方法以判断实例是否正常运行。如果某个实例的运行状况检查未通过，则 HAProxy 停止将流量路由至该实例，直到通过手动或自动修复方式重启该实例。该 URL 后缀的默认值是 “/”，对应服务器实例的主页：http://*DNSName*/。

Health check method

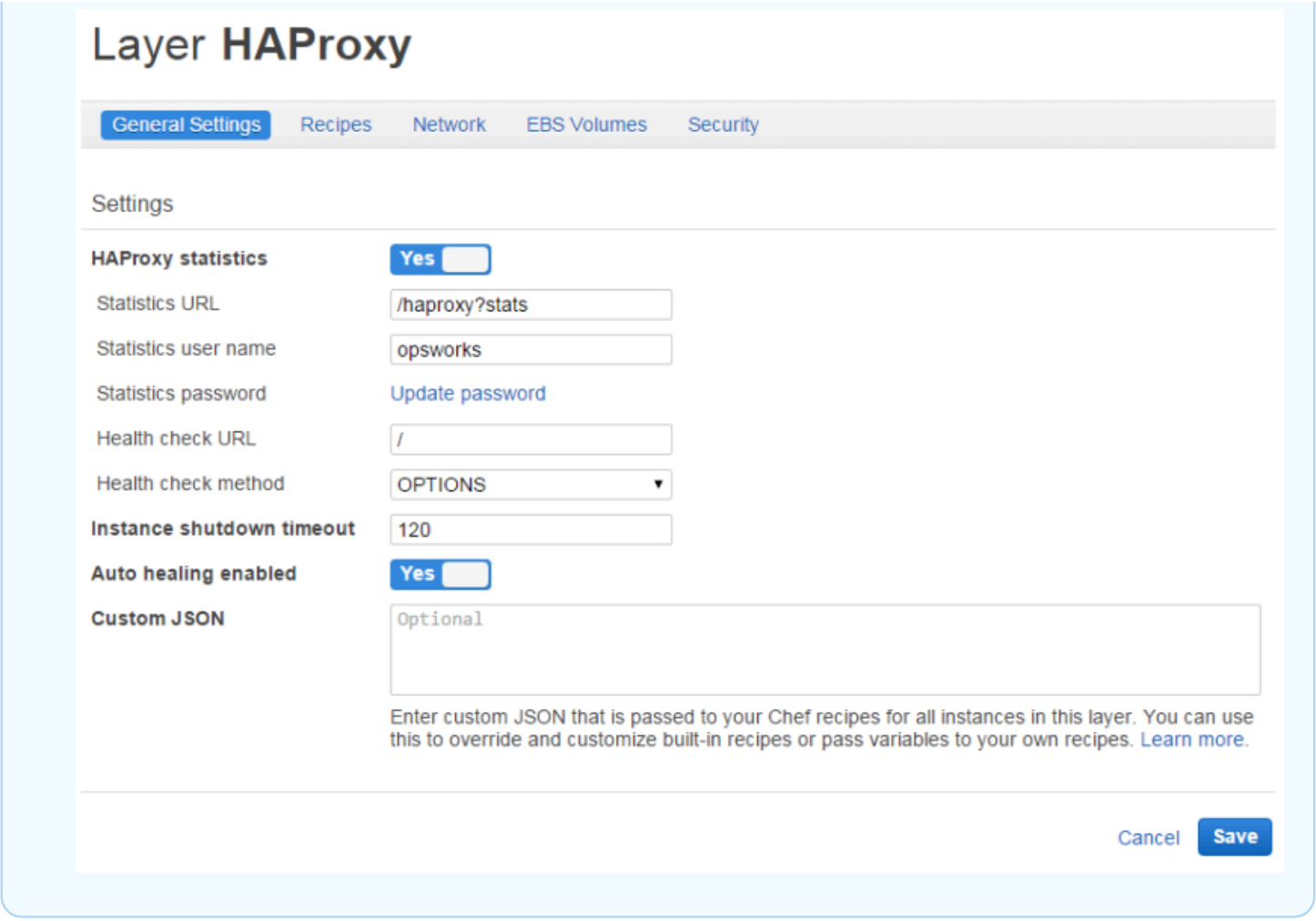
用于检查实例是否在正常运行的 HTTP 方法。默认值为 OPTIONS，您也可以指定 GET 或 HEAD。有关更多信息，请参阅 httpchk。

自定义安全组

如果您选择不自动将内置 AWS OpsWorks Stacks 安全组与您的图层关联，则会显示此设置。您必须指定要将哪一安全组与层关联起来。确保该组具有正确的设置以便允许层之间的流量。有关更多信息，请参阅 创建新堆栈。



> **Note**
>
> 记录密码以备后用； AWS OpsWorks Stacks 不允许您在创建图层后查看密码。不过，您可以通过转到该层的 Edit 页面，然后单击 General Settings 选项卡上的 Update password 来更新密码。



HAProxy 层的工作原理

默认情况下，HAProxy 执行以下操作：

- 侦听 HTTP 和 HTTPS 端口上的请求。

  您可以通过重写 Chef 配置模板 `haproxy.cfg.erb`，将 HAProxy 配置为仅侦听 HTTP 或 HTTPS 端口。
- 将传入流量路由到属于任何应用程序服务器层成员的实例。

  默认情况下， AWS OpsWorks Stacks 将 HAProxy 配置为将流量分配给属于任何应用程序服务器层的实例。例如，您可以有这样一个堆栈，它同时包含 Rails App Server 和 PHP App Server 层，而 HAProxy 主节点会将流量分配给这两个层中的实例。可以使用自定义配方来配置默认路由。
- 跨多个可用区路由流量。

如果一个可用区出现故障，负载均衡器则将传入流量路由到其他区域的实例，这样您的应用程序可以继续运行而不会中断。为此，我们建议您将您的应用程序服务器分散在多个可用区。

- 定期在每个应用程序服务器实例上运行指定的运行状况检查方法以评估其状况。

  如果该方法未在指定的超时时间内返回，则假定该实例已失败，HAProxy 将停止将请求路由到该实例。 AWS OpsWorks 堆栈还提供了一种自动替换失败实例的方法。有关更多信息，请参阅 使用自动修复。您可以在创建层时更改运行状况检查方法。

- 收集统计信息并在网页上显示 (可选)。

> **Important**
>
> 要让采用默认 OPTIONS 方法的运行状况检查正确工作，您的应用程序必须返回状态代码 2xx 或 3xx。

默认情况下，当您将实例添加到 HAProxy 层时， AWS OpsWorks Stacks 会为其分配一个弹性 IP 地址来代表该应用程序，该地址向全世界公开。因为 HAProxy 实例的 Elastic IP 地址是应用程序唯一公开的 URL，所以您不必为底层应用程序服务器实例创建和管理公有域名。您可以通过转至 Instances 页并检查实例的公有 IP 地址来获得该地址，如下图所示。地址后面跟 (EIP) 表明是弹性 IP 地址。有关弹性 IP 地址的更多信息，请参阅弹性 IP 地址 (EIP)。

当您停止 HAProxy 实例时， AWS OpsWorks Stacks 会保留弹性 IP 地址，并在您重启实例时将其重新分配给该实例。如果删除 HAProxy 实例，则默认情况下， AWS OpsWorks Stacks 会删除该实例的 IP 地址。要保留该地址，请清除 Delete instance's Elastic IP 选项，如下图所示。

该选项会影响当您在层中添加新实例以替换被删除的实例时发生的情况：

- 如果您保留了已删除实例的弹性 IP 地址， AWS OpsWorks Stacks 会将该地址分配给新实例。
- 否则， AWS OpsWorks Stacks 会为该实例分配一个新的弹性 IP 地址，您必须更新 DNS 注册商设置以映射到新地址。

当应用程序服务器实例上线或离线 (不论是手动还是作为自动扩展或自动修复的结果)，负载均衡器配置都必须更新以将流量路由到当前的联机实例集。该任务由层的内置配方自动处理：

- 当新实例上线时， AWS OpsWorks Stacks 会触发配置生命周期事件。HAProxy 层的内置 Configure 配方更新负载均衡器配置，以便它也将请求分发到任何新的应用程序服务器实例。
- 当实例脱机或实例未通过运行状况检查时， AWS OpsWorks Stacks 还会触发配置生命周期事件。HAProxy Configure 配方更新负载均衡器配置以将流量仅路由到其余上线实例。

最后，您还可以将自定义域和 HAProxy 层结合使用。有关更多信息，请参阅 使用自定义域。

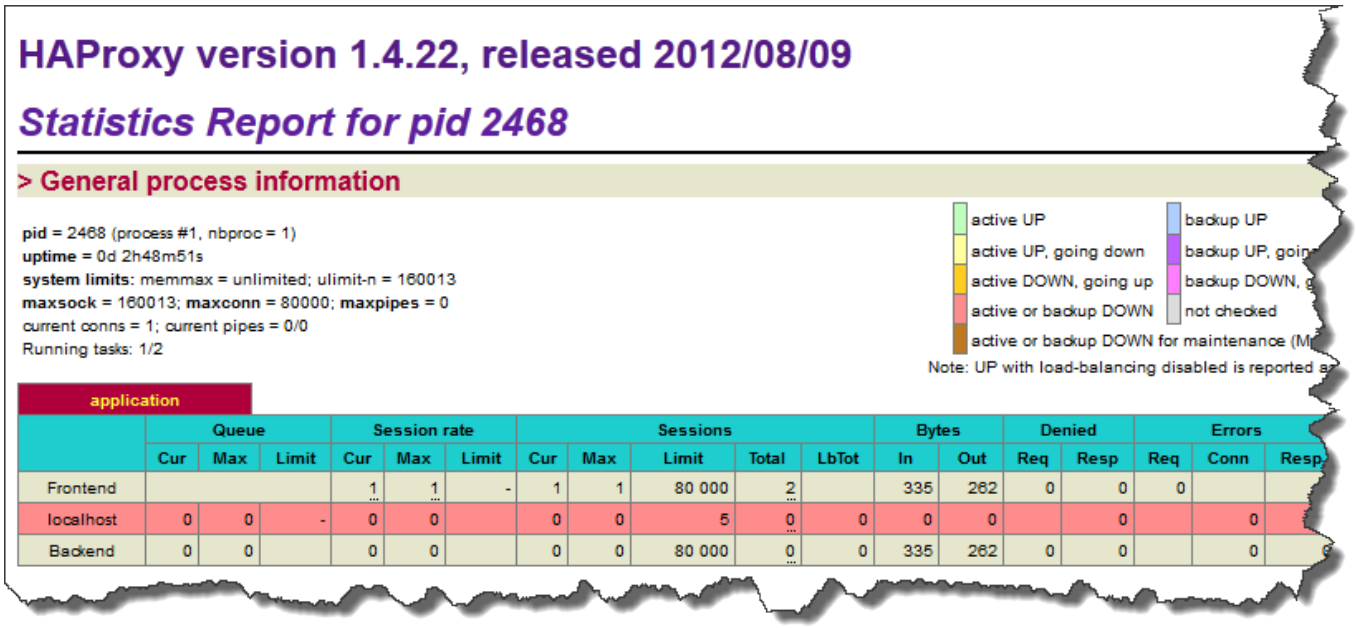
统计信息页面

如果您启用了统计信息页面，则 HAProxy 会在指定 URL 处显示一个包含多种指标的页面。

查看 HAProxy 统计数据

1. 从 &lb; 实例的 Details 页面获得其 Public DNS 名称并复制该名称。
2. 在 层 页面上，单击 HAProxy 以打开层的详细信息页面。

3. 从层的详细信息中获取统计信息 URL 并将其附加到公有 DNS 名称后。例如：**`http://ec2-54-245-102-172.us-west-2.compute.amazonaws.com/haproxy?stats`**。
4. 将来自上一步的 URL 粘贴到浏览器中，用您在创建层时指定的用户名和密码打开统计信息页面。



HAProxy version 1.4.22, released 2012/08/09

Statistics Report for pid 2468

> General process information

pid = 2468 (process #1, nbproc = 1)
uptime = 0d 2h48m51s
system limits: memmax = unlimited; ulimit-n = 160013
maxsock = 160013; **maxconn** = 80000; **maxpipes** = 0
current conns = 1; current pipes = 0/0
Running tasks: 1/2

active UP | backup UP
active UP, going down | backup UP, goin
active DOWN, going up | backup DOWN, g
active or backup DOWN | not checked
active or backup DOWN for maintenance (M
Note: UP with load-balancing disabled is reported a

application

	Queue			Session rate			Sessions					Bytes		Denied		Errors		
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp
Frontend				1	1	-	1	1	80 000	2		335	262	0	0	0		
localhost	0	0	-	0	0		0	0	5	0	0	0	0		0		0	
Backend	0	0		0	0		0	0	80 000	0	0	335	262	0	0		0	0

MySQL 层参考

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 此层仅适用于基于 Linux 的堆栈。

MySQL 层支持 MySQL，这是一种广泛使用的关系数据库管理系统。 AWS OpsWorks Stacks 会安装最新的可用版本，具体取决于操作系统。如果您添加 MySQL 实例，则系统会为应用程序服务器层提供所需的访问信息。您必须写入自定义 Chef 配方，以设置 master-master 或 masterslave 配置。

短名称：db-master

兼容性：MySQL 层与以下层兼容：自定义、lb、memcached、monitoring-master、nodejs-app、php-app、rails-app 和 web。

开放端口：MySQL 层允许公开访问端口 22(SSH) 和堆栈的 Web 服务器、自定义服务器以及 Rails、PHP 和 Node.js 应用程序服务器的所有端口。

自动分配弹性 IP 地址：默认情况下关闭

默认 EBS 卷：是，在 `/vol/mysql` 中

默认安全组：AWS--db-Master OpsWorks-Server

配置：要配置 MySQL 层，您必须指定以下内容：

- 根用户密码
- MySQL 引擎

Setup 配方：

- opsworks_initial_setup
- ssh_host_keys
- ssh_users
- mysql::client
- dependencies
- ebs
- opsworks_ganglia::client
- mysql::server
- dependencies
- deploy::mysql

Configure 配方：

- opsworks_ganglia::configure-client
- ssh_users
- agent_version
- deploy::mysql

Deploy 配方：

- deploy::default
- deploy::mysql

Shutdown 配方：

- opsworks_shutdown::default
- mysql::stop

安装:

- AWS OpsWorks Stacks 使用实例的包安装程序将 MySQL 及其日志文件安装到其默认位置。有关更多信息，请参阅 MySQL Documentation。

MySQL OpsWorks 层

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 该层仅适用于 Chef 11 和更早的基于 Linux 的堆栈。

MySQL OpsWorks 层为充当 MyS QL 数据库主服务器的 Amazon EC2 实例提供了蓝图。内置配方为已部署到应用程序服务器层的每个应用程序创建数据库。例如，如果您部署 PHP 应用程序“myapp”，则该配方会创建“myapp”数据库。

MySQL 层具有以下配置设置。

MySQL root user password

(必填项) 根用户密码。

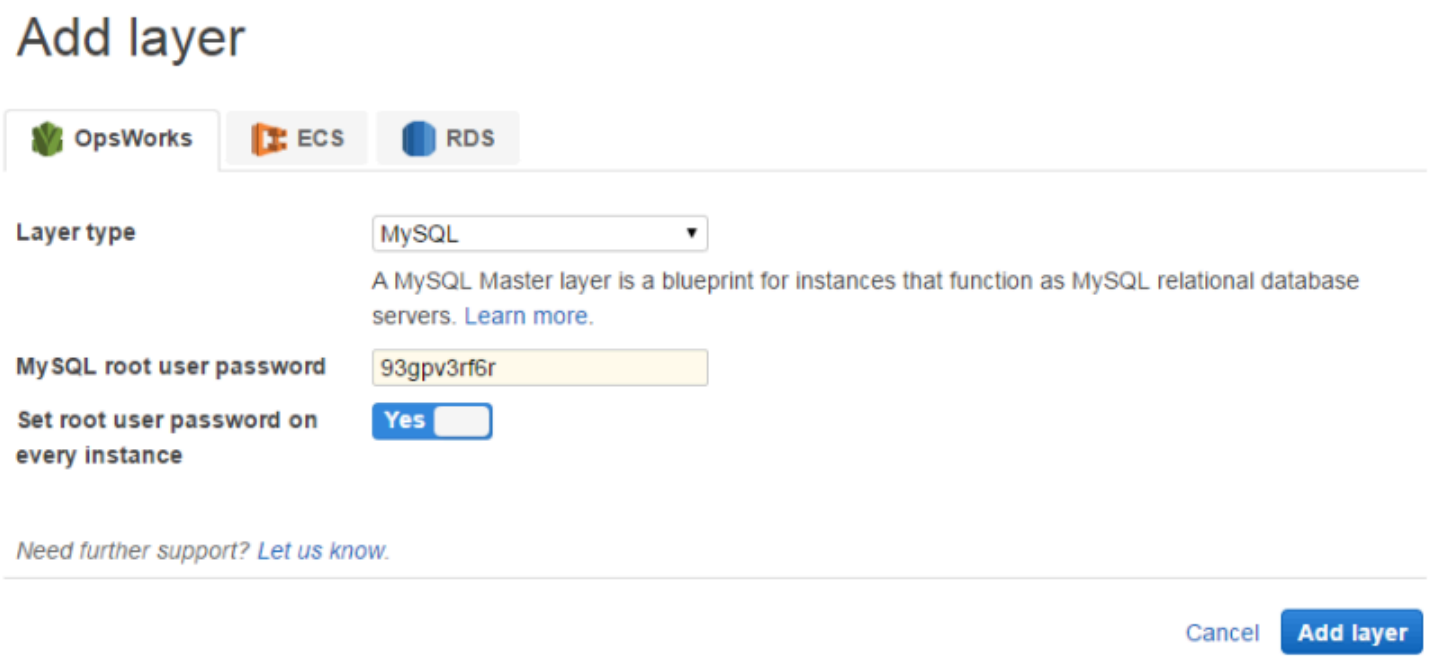
Set root user password on every instance

(可选) 堆栈中每个实例上安装的堆栈配置和部署属性中是否包含根用户密码。默认设置为 Yes。

如果将此值设置为 “否”，则 AWS OpsWorks Stacks 仅将 root 密码传递给应用程序服务器实例。

自定义安全组

(可选) 要与该层关联的自定义安全组。有关更多信息，请参阅 创建新堆栈。



您可以向该层添加一个或多个实例，每个实例都表示一个单独的 MySQL 数据库主实例。然后，您可以将实例附加到应用程序，这会在该应用程序的应用程序服务器上安装必要的连接信息。然后，应用程序可以使用连接信息来连接到实例的数据库服务器。

应用程序服务器层参考

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

AWS OpsWorks Stacks 支持几种不同的应用程序和静态网页服务器。

主题

AWS Flow (Ruby) 层参考

Important

该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

Note

此层仅适用于基于 Linux 的堆栈。

AWS Flow (Ruby) 层为托管 Amazon Simple Workflow Service 活动和工作流工作线程的实例提供蓝图。

短名： aws-flow-ruby

兼容性：AWS Flow (Ruby) 层与 PHP App Server、MySQL、Memcached、Ganglia 和自定义层兼容。

开放端口：无。

IAM 角色： aws-opsworks-ec2-role-with-swf 是 AWS OpsWorks Stacks 根据要求为您创建的标准 AWS Flow (Ruby) 角色。

自动分配弹性 IP 地址：默认情况下关闭

默认 EBS 卷：否

默认安全组：AWS-AWS-flow-Ruby OpsWorks-Server

Setup 配方：

- opsworks_initial_setup
- ssh_host_keys
- ssh_users
- mysql::client
- dependencies
- ebs
- opsworks_ganglia::client
- opsworks_aws_flow_ruby::setup

Configure 配方：

- opsworks_ganglia::configure-client
- ssh_users
- mysql::client
- agent_version
- opsworks_aws_flow_ruby::configure

Deploy 配方：

- deploy::default
- 部署:: aws-flow-ruby

UnDeploy 配方：

- 部署:: aws-flow-ruby-undeploy

Shutdown 配方：

- opsworks_shutdown::default

Java App Server 层参考

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 此层仅适用于基于 Linux 的堆栈。

Java App Server 层支持 Apache Tomcat 7.0 应用程序服务器。

短名称：java-app

兼容性：Java App Server 层与以下层兼容：自定义、db-master 和 memcached。

开放端口：Java App Server 层允许公开访问端口 22 (SSH)、80 (HTTP)、443 (HTTPS) 和负载均衡器的所有端口。

自动分配弹性 IP 地址：默认情况下关闭

默认 EBS 卷：否

默认安全组：AWS--Java-App OpsWorks-Server

Setup 配方：

- opsworks_initial_setup
- ssh_host_keys
- ssh_users
- mysql::client
- dependencies
- ebs
- opsworks_ganglia::client

- opsworks_java::setup

Configure 配方：

- opsworks_ganglia::configure-client
- ssh_users
- agent_version
- opsworks_java::configure

Deploy 配方：

- deploy::default
- deploy::java

UnDeploy 配方：

- deploy::java-undeploy

Shutdown 配方：

- opsworks_shutdown::default
- deploy::java-stop

安装:

- Tomcat 安装到 `/usr/share/tomcat7`。
- 有关如何生成日志文件的更多信息，请参阅 Logging in Tomcat。

Node.js App Server 层参考

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 此层仅适用于基于 Linux 的堆栈。

Node.js App Server 层支持 Node.js 应用程序服务器，它是一个用于实施高度可扩展的网络应用程序服务器的平台。程序是使用事件驱动的异步 I/O 编写的 JavaScript，以最大限度地减少开销并最大限度地提高可扩展性。

短名称：nodejs-app

兼容性：Node.js App Server 层与以下层兼容：自定义、db-master、memcached 和 monitoring-master。

开放端口：Node.js App Server 层允许公开访问端口 22 (SSH)、80 (HTTP)、443 (HTTPS) 和负载均衡器的所有端口。

自动分配弹性 IP 地址：默认情况下关闭

默认 EBS 卷：否

默认安全组：AWS--nodejs-App OpsWorks-Server

Setup 配方：

- opsworks_initial_setup
- ssh_host_keys
- ssh_users
- mysql::client
- dependencies
- ebs
- opsworks_ganglia::client
- opsworks_nodejs
- opsworks_nodejs::npm

Configure 配方：

- opsworks_ganglia::configure-client

- ssh_users
- agent_version
- opsworks_nodejs::configure

Deploy 配方：

- deploy::default
- opsworks_nodejs
- opsworks_nodejs::npm
- deploy::nodejs

UnDeploy 配方：

- deploy::nodejs-undeploy

Shutdown 配方：

- opsworks_shutdown::default
- deploy::nodejs-stop

安装:

- Node.js 安装到 `/usr/local/bin/node`。
- 有关如何生成日志文件的更多信息，请参阅在 Nodejitsu 网站上如何登录 node.js。

Node.js 应用程序配置：

- Node.js 运行的主文件必须命名为 `server.js`，并位于已部署的应用程序的根目录中。
- 必须将 Node.js 应用程序设置为侦听端口 80 (或端口 443，如果适用)。

> **Note**
>
> 运行 Express 的 Node.js 应用程序通常使用以下代码来设置侦听端口，其中 `process.env.PORT` 表示默认端口并解析为 80：

```
app.set('port', process.env.PORT || 3000);
```

对于 AWS OpsWorks 堆栈，您必须明确指定端口 80，如下所示：

```
app.set('port', 80);
```

PHP App Server 层参考

Important

该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

Note

此层仅适用于基于 Linux 的堆栈。

PHP App Server 层通过使用带有 mod_php 的 Apache2 来支持 PHP 应用程序服务器。

短名称：php-app

兼容性：PHP App Server 层与以下层兼容：自定义、db-master、memcached、monitoring-master 和 rails-app。

开放端口：PHP App Server 层允许公开访问端口 22 (SSH)、80 (HTTP)、443 (HTTPS) 和负载均衡器的所有端口。

自动分配弹性 IP 地址：默认情况下关闭

默认 EBS 卷：否

默认安全组：AWS--php-App OpsWorks-Server

Setup 配方：

- opsworks_initial_setup

- ssh_host_keys
- ssh_users
- mysql::client
- dependencies
- ebs
- opsworks_ganglia::client
- mysql::client
- dependencies
- mod_php5_apache2

Configure 配方：

- opsworks_ganglia::configure-client
- ssh_users
- agent_version
- mod_php5_apache2::php
- php::configure

Deploy 配方：

- deploy::default
- deploy::php

UnDeploy 配方：

- deploy::php-undeploy

Shutdown 配方：

- opsworks_shutdown::default
- apache2::stop

安装：

- AWS OpsWorks Stacks 使用实例的软件包安装程序将 Apache2、mod_php 和相关的日志文件安装到其默认位置。有关安装的更多信息，请参阅 Apache。有关日志记录的更多信息，请参阅 Log Files。

Rails App Server 层参考

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 此层仅适用于基于 Linux 的堆栈。

Rails App Server 层支持 Ruby on Rails 应用程序服务器。

短名称：rails-app

兼容性：Rails App Server 层与以下层兼容：自定义、db-master、memcached、monitoring-master 和 php-app。

Ports：Rails App Server 层允许公开访问端口 22 (SSH)、80 (HTTP)、443 (HTTPS) 和负载均衡器的所有端口。

自动分配弹性 IP 地址：默认情况下关闭

默认 EBS 卷：否

默认安全组：AWS-Rails-App OpsWorks-Server

配置：要配置 Rails App Server 层，您必须指定以下内容：

- Ruby 版本
- Rails 堆栈
- Rubygems 版本
- 是否安装和管理 Bundler

- Bundler 版本

Setup 配方：

- opsworks_initial_setup
- ssh_host_keys
- ssh_users
- mysql::client
- dependencies
- ebs
- opsworks_ganglia::client
- apache2 apache2::mod_deflate
- passenger_apache2
- passenger_apache2::mod_rails
- passenger_apache2::rails

Configure 配方：

- opsworks_ganglia::configure-client
- ssh_users
- agent_version
- rails::configure

Deploy 配方：

- deploy::default
- deploy::rails

UnDeploy 配方：

- deploy::rails-undeploy

Shutdown 配方：

- opsworks_shutdown::default
- apache2::stop

安装:

- AWS OpsWorks Stacks 使用实例的软件包安装程序将包含 mod_passerger、mod_rails 和相关日志文件的 Apache2 安装到其默认位置。有关安装的更多信息，请参阅 Phusion Passenger。有关日志记录的更多信息，请参阅 Log Files。

静态 Web 服务器层参考

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 此层仅适用于基于 Linux 的堆栈。

静态 Web 服务器层提供静态 HTML 页面，其中可能包含客户端代码， JavaScript例如。它基于 Nginx，这是一种开源 HTTP、反向代理和电子邮件代理服务器。

短名称：web

兼容性：Static Web Server 层与以下层兼容：自定义、db-master 和 memcached。

开放端口：Static Web Server 层允许公开访问端口 22 (SSH)、80 (HTTP)、443 (HTTPS) 和负载均衡器的所有端口。

自动分配弹性 IP 地址：默认情况下关闭

默认 EBS 卷：否

默认安全组：AWS--We b OpsWorks-Server

Setup 配方：

- opsworks_initial_setup
- ssh_host_keys
- ssh_users
- mysql::client
- dependencies
- ebs
- opsworks_ganglia::client
- nginx

Configure 配方：

- opsworks_ganglia::configure-client
- ssh_users
- agent_version

Deploy 配方：

- deploy::default
- deploy::web

UnDeploy 配方：

- deploy::web-undeploy

Shutdown 配方：

- opsworks_shutdown::default
- nginx::stop

安装:

- Nginx 安装到 `/usr/sbin/nginx`。

- Nginx 日志文件采用 `/var/log/nginx` 格式。

应用程序服务器层

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 这些层仅适用于 Chef 11 和更早的基于 Linux 的堆栈。

AWS OpsWorks Stacks 支持几种不同的应用程序服务器，其中 “应用程序” 包括静态网页。每种类型的服务器都有单独的 AWS OpsWorks Stacks 层，其内置配方用于在每个层的实例上安装应用程序服务器和任何相关软件包、部署应用程序等。例如，Java App Server 层将安装多个程序包 (包括 Apache、Tomcat 和 OpenJDK)，并将 Java 应用程序部署到层的每个实例中。

以下是使用应用程序服务器层的基本步骤：

1. 创建一种可用的 App Server 层类型。
2. 将一个或多个实例添加到该层。
3. 创建应用程序并将应用程序部署到实例中。有关更多信息，请参阅 应用程序。
4. (可选) 如果层有多个实例，您可以添加一个负载均衡器，以在这些实例之间分配传入流量。有关更多信息，请参阅 HAProxy AWS OpsWorks 堆栈层。

主题

- AWS Flow (Ruby) 层
- Java 应用服务器 AWS OpsWorks 堆栈层
- Node.js App Server AWS OpsWorks 堆栈层
- PHP 应用服务器 AWS OpsWorks 堆栈层
- Rails 应用程序服务器 AWS OpsWorks 堆栈层

- 静态 Web 服务器 AWS OpsWorks 堆栈层

AWS Flow (Ruby) 层

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 此层仅适用于基于 Linux 的堆栈。

AWS Flow (Ruby) 层是一个 AWS OpsWorks 堆栈层，它为托管 A mazon SWF 活动和工作流程工作人员的实例提供蓝图。工作线程通过使用适用于 Ruby 的 AWS Flow Framework 进行实施，AWS Flow Framework 是一个编程框架，可简化实施分布式的异步应用程序同时提供 Amazon SWF 的所有好处的过程。对于实施应用程序以满足广泛的场景 (包括业务流程、媒体编码、长时间运行的任务和后台处理) 来说，这是理想的做法。

AWS Flow (Ruby) 层包含以下配置设置。

RubyGems 版本

框架的 Gem 版本。

Bundler 版本

Bundler 版本。

EC2 实例配置文件

层的实例要使用的用户定义的 Amazon EC2 实例配置文件。此配置文件必须授予在层的实例上运行的应用程序访问 Amazon EC2 的权限。

如果您的账户没有合适的配置文件，则可以选择具有 SWF 访问权限的新建配置文件让 AWS OpsWorks Stacks 更新其配置文件，或者您可以使用 IAM 控制台自行更新。然后，您可以为所有后继

的 AWS Flow 层使用更新后的配置文件。以下是关于如何使用 IAM; 控制台创建配置文件的简短说明。有关更多信息，请参阅 Amazon Simple Workflow Service 中的 Identity and Access Management。

为 AWS Flow (Ruby) 实例创建配置文件

1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在导航窗格中选择 Policies，然后选择 Create Policy 创建一个新的客户管理的策略。
3. 对于服务，选择 SNS。
4. 在操作中，选择所有 SWF 操作(swf: *)。
5. 对于 (Amazon 资源名称 (ARN)，一个指定工作线程可访问哪些 Amazon SWF 域的 ARN。选择 **`All resources`** 以提供对所有域的访问权。
6. 选择下一步。
7. （可选）输入标签来标识策略。
8. 选择下一步。
9. 完成后，选择 Create policy。
10. 在导航窗格中，选择 Roles，然后选择 Create role。
11. 指定角色名称，然后选择 Next Step。当创建角色后，您便无法更改角色名称。
12. 选择 Amazon Web Service，然后选择 EC2。
13. 选择下一步。
14. 从权限策略列表中，选择您之前创建的策略。
15. 选择下一步。
16. 输入角色名称，然后选择创建角色。当创建角色后，您便无法更改角色名称。
17. 在 AWS OpsWorks 堆栈中创建 AWS Flow (Ruby) 层时，请指定此配置文件。

Java 应用服务器 AWS OpsWorks 堆栈层

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 此层仅适用于基于 Linux 的堆栈。

Java App Server 层是一个 AWS OpsWorks 堆栈层，它为充当 Java 应用程序服务器的实例提供蓝图。该层基于 Apache Tomcat 7.0 和 Ope n JDK 7。 AWS OpsWorks Stacks 还会安装 Java 连接器库，该库允许 Java 应用程序使用 JDBC `DataSource` 对象连接到后端数据存储。

安装：Tomcat 安装在 `/usr/share/tomcat7` 中。

Add Layer 页面提供以下配置选项：

Java VM 选项

您可以使用此设置来指定自定义 Java VM 选项；没有默认选项。例如，一组常见的选项是 `-Djava.awt.headless=true -Xmx128m -XX:+UseConcMarkSweepGC`。如果您使用 Java VM 选项，请确保传递了一组有效的选项； AWS OpsWorks Stacks 不会验证字符串。如果您试图传递无效的选项，Tomcat 服务器通常会无法启动，这会导致设置失败。如果出现这种情况，您可以检查实例的设置 Chef 日志以了解详细信息。有关如何查看和解释 Chef 日志的更多信息，请参阅Chef 日志。

自定义安全组

如果您选择不自动将内置 AWS OpsWorks Stacks 安全组与您的图层关联，则会显示此设置。您必须指定要将哪一安全组与层关联起来。有关更多信息，请参阅 创建新堆栈。

Elastic Load Balancer

您可以将 Elastic Load Balancing 负载均衡器连接到层的实例。有关更多信息，请参阅 Elastic Load Balancing 层。

您可以通过使用自定义 JSON 或自定义属性文件指定其他配置设置。有关更多信息，请参阅 自定义配置。

> **Important**
>
> 如果您的 Java 应用程序使用的是 SSL，我们建议您禁用 SSLv3 (如果可能的话)，以应对 CVE-2014-3566 中介绍的漏洞。有关更多信息，请参阅 为 Apache 服务器禁用 SSLv3。

主题

为 Apache 服务器禁用 SSLv3

要禁用 SSLv3，您必须修改 Apache 服务器的 `ssl.conf` 文件的 `SSLProtocol` 设置。为此，您必须覆盖内置 apache2 说明书的 `ssl.conf.erb` 模板文件，Java App Server 层的 Setup 配方将使用该文件创建 `ssl.conf`。具体细节取决于您为层的实例指定的操作系统。下面总结了对于 Amazon Linux 和 Ubuntu 系统所需进行的修改。对于 Red Hat Enterprise Linux (RHEL) 系统，将会自动禁用 SSLv3。有关如何覆盖内置模板的更多信息，请参阅使用自定义模板。

Amazon Linux

这些操作系统的 `ssl.conf.erb` 文件位于 apache2 说明书的 `apache2/templates/default/mods` 目录中。下面显示了内置文件的相关部分。

```
...
#SSLCipherSuite ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL

# enable only secure protocols: SSLv3 and TLSv1.2, but not SSLv2
SSLProtocol all -SSLv2
</IfModule>
```

覆盖 `ssl.conf.erb` 并修改 `SSLProtocol` 设置，如下所示。

```
...
#SSLCipherSuite ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL

# enable only secure protocols: SSLv3 and TLSv1.2, but not SSLv2
SSLProtocol all -SSLv3 -SSLv2
</IfModule>
```

Ubuntu 14.04 LTS

此操作系统的 `ssl.conf.erb` 文件位于 apache2 说明书的 `apache2/templates/ubuntu-14.04/mods` 目录中。下面显示了内置文件的相关部分。

```
...
# The protocols to enable.
# Available values: all, SSLv3, TLSv1.2
# SSL v2 is no longer supported
SSLProtocol all
...
```

将此设置更改为以下内容。

```
...
# The protocols to enable.
# Available values: all, SSLv3, TLSv1.2
# SSL v2 is no longer supported
SSLProtocol all -SSLv3 -SSLv2
...
```

自定义配置

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

AWS OpsWorks Stacks 将其他配置设置作为内置属性公开，这些属性都在命名空间中。`opsworks_java`您可以使用自定义 JSON 或自定义属性文件以覆盖内置属性并指定自定义值。例如，JVM 和 Tomcat 版本由内置 `jvm_version` 和 `java_app_server_version` 属性表示，两者均被设置为 7。您可以使用自定义 JSON 或自定义属性文件将两个属性都设置为 6，或仅将其中一个属性设置为 6。以下示例使用自定义 JSON 将这两个属性都设置为 6：

```
{
  "opsworks_java": {
    "jvm_version": 6,
    "java_app_server_version" : 6
  }
```

```
}
```

有关更多信息，请参阅 使用自定义 JSON。

自定义配置的另一个示例是通过覆盖 `use_custom_pkg_location`、`custom_pkg_location_url_debian` 和 `custom_pkg_location_url_rhel` 属性安装自定义 JDK。

> **Note**
>
> 如果您覆盖内置说明书，您将需要自己更新这些组件。

有关属性以及如何覆盖属性的更多信息，请参阅覆盖属性。要查看内置属性的列表，请参阅opsworks_java 属性。

部署 Java 应用程序

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

以下主题介绍了如何将应用程序部署到 Java App Server 层的实例中。示例针对的都是 JSP 应用程序，但安装其他类型的 Java 应用程序时，您可以使用基本相同的安装过程。

您可以从任何受支持的存储库中部署 JSP 页面。如果你想部署 WAR 文件，请注意， AWS OpsWorks Stacks 会自动提取从 Amazon S3 或 HTTP 存档中部署的 WAR 文件，而不是从 Git 或 Subversion 存储库中提取部署的 WAR 文件。如果您想要为 WAR 文件使用 Git 或 Subversion，您可以执行以下操作之一：

- 将提取的存档存储在存储库中。
- 将 WAR 文件存储到存储库中，并使用 Chef 部署挂钩提取存档，如下例中所述。

您可以在四个部署阶段的任一阶段使用 Chef 部署挂钩在实例上运行用户提供的 Ruby 应用程序。应用程序名称决定着阶段。以下是一个名为 `before_migrate.rb` 的 Ruby 应用程序的示例，该应用程

序可从 Git 或 Subversion 存储库中提取已部署的 WAR 文件。名称将应用程序与 Checkout 部署挂钩关联起来，以便它在部署操作开始时 (检查代码之后，迁移之前) 运行。有关如何使用此示例的更多信息，请参阅使用 Chef 部署挂钩。

```
::Dir.glob(::File.join(release_path, '*.war')) do |archive_file|
  execute "unzip_#{archive_file}" do
    command "unzip #{archive_file}"
    cwd release_path
  end
end
```

Note

当您为 JSP 应用程序部署更新时，Tomcat 可能无法识别该更新，而是继续运行现有的应用程序版本。在某些情况下，比如当您将您的应用程序作为仅包含一个 JSP 页面的 .zip 文件时，可能会发生这种情况。要确保 Tomcat 将运行最近部署的版本，项目的根目录应包含放置了 `web.xml` 文件的 WEB-INF 目录。`web.xml` 文件可包含各种内容，但以下内容足以确保 Tomcat 识别更新并运行当前部署的应用程序版本。您无需为每个更新更改版本。Tomcat 将识别更新，即使版本未发生更改。

```
<context-param>
  <param-name>appVersion</param-name>
  <param-value>0.1</param-value>
</context-param>
```

主题

- 部署 JSP 应用程序
- 部署 JSP 应用程序附带后端数据库

部署 JSP 应用程序

要部署 JSP 应用程序，指定名称和存储库信息。您也可以选择性地指定域和 SSL 设置。有关如何创建应用程序的更多信息，请参阅添加应用程序。以下过程介绍了如何在公有 Amazon S3 存档中创建和部署一个简单的 JSP 页面。有关如何使用其他存储库类型 (包括私有 Amazon S3 存档) 的信息，请参阅应用程序源。

以下示例显示了 JSP 页面，该页面仅显示某些系统信息。

```
<%@ page import="java.net.InetAddress" %>
<html>
<body>
<%
    java.util.Date date = new java.util.Date();
    InetAddress inetAddress = InetAddress.getLocalHost();
%>
The time is
<%
    out.println( date );
    out.println("<br>Your server's hostname is "+inetAddress.getHostName());
%>
<br>
</body>
</html>
```

> **Note**
>
> 以下过程假定您已熟悉关于创建堆栈、将实例添加到层等的基础知识。如果你不熟悉 AWS OpsWorks Stacks，你应该先看看Chef 11 Linux 堆栈入门。

从 Amazon S3 存档部署 JSP 页面

1. 创建带有 Java App Server 层的堆栈，将一个全天候实例添加到层中，然后启动实例。
2. 将代码复制到名为 `simplejsp.jsp` 的文件中，将该文件放到名为 `simplejsp` 的文件夹中，然后创建该文件夹的 `.zip` 存档。名称是任意的；您可以按照您的意愿使用任何文件或文件夹名称。您也可以使用其他类型的存档，包括 gzip、bzip2、tarball 或 Java WAR 文件。请注意， AWS OpsWorks Stacks 不支持未压缩的压缩包。要部署多个 JSP 页面，将这些页面放到相同的存档中。
3. 将该存档上传到 Amazon S3 存储桶，并将该文件设置为公有。复制该文件的 URL 以备后用。有关如何创建存储桶和上传文件的更多信息，请转至 Amazon Simple Storage Service 入门指南。
4. 将一个应用程序添加到堆栈并指定以下设置：
   - 名称 – `SimpleJSP`
   - App type - `Java`

- Repository type – `Http Archive`
- Repository URL -存档文件的 Amazon S3 URL。

对其余设置使用默认值，然后单击 Add App 创建应用程序。

5. 将应用程序部署到 Java App Server 实例。

现在，您可以转到应用程序的 URL 并查看应用程序。如果您未指定域，您可以使用实例的公有 IP 地址或其公有 DNS 名称构建一个 URL。要获取实例的公有 IP 地址或公有 DNS 名称，请转到 AWS OpsWorks Stacks 控制台，然后在 “实例” 页面上单击该实例的名称以打开其详细信息页面。

网址的其余部分取决于应用程序的短名称，该名称是 AWS OpsWorks Stacks 根据您在创建应用程序时指定的应用程序名称生成的小写名称。例如，SimpleJSP 的短名称是 simplejsp。您可以从应用程序的详细信息页面中获得应用程序的短名称。

- 如果短名称为 `root`，您可以使用 `http://`*`public_DNS`*`/`*`appname`*`.jsp` 或 `http://`*`public_IP`*`/`*`appname`*`.jsp`。
- 否则，您可以使用 `http://`*`public_DNS`*`/`*`app_shortname`*`/`*`appname`*`.jsp` 或 `http://`*`public_IP`*`/`*`app_shortname`*`/`*`appname`*`.jsp`。

如果您已经为应用程序指定了域，则 URL 为 `http://`*`domain`*`/`*`appname`*`.jsp`。

此示例中的 URL 将如下所示：`http://192.0.2.0/simplejsp/simplejsp.jsp`。

如果您希望将多个应用程序部署到同一个实例，则您不应当使用 `root` 作为短名称。这可能会导致 URL 冲突，从而妨碍应用程序正常运行。而应当为每个应用程序分配不同的域名。

部署 JSP 应用程序附带后端数据库

JSP 页面可以使用 JDBC `DataSource` 对象连接到一个后端数据库。您可以通过上一部分中的步骤创建和部署此类应用程序，另外再执行一个设置连接的步骤即可。

以下 JSP 页面介绍如何连接到 `DataSource` 对象。

```
<html>
  <head>
    <title>DB Access</title>
  </head>
  <body>
```

```
<%@ page language="java" import="java.sql.*,javax.naming.*,javax.sql.*" %>
<%
  StringBuffer output = new StringBuffer();
  DataSource ds = null;
  Connection con = null;
  Statement stmt = null;
  ResultSet rs = null;
  try {
    Context initCtx = new InitialContext();
    ds = (DataSource) initCtx.lookup("java:comp/env/jdbc/mydb");
    con = ds.getConnection();
    output.append("Databases found:<br>");
    stmt = con.createStatement();
    rs = stmt.executeQuery("show databases");
    while (rs.next()) {
      output.append(rs.getString(1));
      output.append("<br>");
    }
  }
  catch (Exception e) {
    output.append("Exception: ");
    output.append(e.getMessage());
    output.append("<br>");
  }
  finally {
    try {
      if (rs != null) {
        rs.close();
      }
      if (stmt != null) {
        stmt.close();
      }
      if (con != null) {
        con.close();
      }
    }
    catch (Exception e) {
      output.append("Exception (during close of connection): ");
      output.append(e.getMessage());
      output.append("<br>");
    }
  }
%>
<%= output.toString() %>
```

```
  </body>
</html>
```

AWS OpsWorks Stacks 创建和初始化DataSource对象，将其绑定到逻辑名称，然后在 Java 命名和目录接口 (JNDI) 命名服务中注册该名称。完整的逻辑名称为 `java:comp/env/user-assigned-name`。您必须通过将自定义 JSON 属性添加到堆栈配置和部署属性定义 `['opsworks_java']['datasources']` 属性，来指定名称中用户分配的部分，如下所述。

部署可连接到 MySQL 数据库的 JSP 页面

1. 创建带有 Java App Server 层的堆栈，将一个全天候实例添加到每个层中，然后启动实例。
2. 将一个数据库层添加到堆栈中。具体细节取决于您使用的数据库。

   要为示例使用 MySQL 实例，添加 MySQL 层到堆栈，添加全天候实例到该层，然后启动实例。

   要为示例使用 Amazon RDS (MySQL) 实例：

   - 为实例指定 MySQL 数据库引擎。
   - ***# AWS OpsWorks-db-Master-Server#security_group_id## AWS-Java-App-Server#security_group_id########OpsWorks##*** AWS OpsWorks 当您在该地区创建第一个堆栈时，Stacks 会为您创建这些安全组。
   - 创建一个名为 `simplejspdb` 的数据库。
   - 确保主用户名和密码不包含 & 或其他可能会导致 Tomcat 错误的符号。

     尤其是在启动期间，Tomcat 必须解析 Web 应用程序上下文文件，该文件是一个 XML 文件，其中包含主密码和用户名。如果任一字符串包含一个 & 字符，XML 分析器就会将其视为格式不正确的 XML 实体，并引发解析异常，从而阻止 Tomcat 启动。有关 Web 应用程序上下文文件的更多信息，请参阅tomcat::context。
   - 将一个 MySQL 驱动程序添加到 Java App Server 层中。
   - 向您的堆栈注册 RDS 实例。

   有关如何使用带有 AWS OpsWorks 堆栈的 Amazon RDS 实例的更多信息，请参阅Amazon RDS 服务层。
3. 将示例代码复制到名为 `simplejspdb.jsp` 的文件中，将该文件放到名为 `simplejspdb` 的文件夹中，然后创建该文件夹的 `.zip` 存档。名称是任意的；您可以按照您的意愿使用任何文件或文件夹名称。您还可以使用其他类型的存档，包括 gzip、bzip2 或 tarball。要部署多个 JSP 页面，将

这些页面放到相同的存档中。有关如何部署其他存储库类型中的应用程序的信息，请参阅应用程序源。

4. 将该存档上传到 Amazon S3 存储桶，并将该文件设置为公有。复制该文件的 URL 以备后用。有关如何创建存储桶和上传文件的更多信息，请转至 Amazon Simple Storage Service 入门指南。
5. 将一个应用程序添加到堆栈并指定以下设置：

   - 名称 – `SimpleJSPDB`
   - App type - `Java`
   - 数据源类型 — OpsWorks（对于 MySQL 实例）或 RDS（对于 Amazon RDS 实例）。
   - Database instance -您之前创建的 MySQL 实例，名称通常为 db-master1(mysql)；或者是名为 ***DB_instance_name*** (mysql) 的Amazon RDS 实例。
   - 数据库名称 – `simplejspdb`。
   - Repository type – `Http Archive`
   - Repository URL -存档文件的 Amazon S3 URL。

   对其余设置使用默认值，然后单击 Add App 创建应用程序。
6. 将以下自定义 JSON 属性添加到堆栈配置属性，其中 simplejspdb 是应用程序的短名称。

```
{
  "opsworks_java": {
    "datasources": {
      "simplejspdb": "jdbc/mydb"
    }
  }
}
```

   AWS OpsWorks Stacks 使用此映射生成包含必要数据库信息的上下文文件。

   有关如何将自定义 JSON 属性添加到堆栈配置属性的更多信息，请参阅使用自定义 JSON。
7. 将应用程序部署到 Java App Server 实例。

现在，您可以使用应用程序的 URL 来查看应用程序。有关如何构建 URL 的介绍，请参阅部署 JSP 应用程序。

此示例中的 URL 将如下所示：`http://192.0.2.0/simplejspdb/simplejspdb.jsp`。

> **Note**
>
> `datasources` 属性可以包含多个属性。每个属性都以应用程序短名称命名，并被设置为逻辑名称中适当的用户分配的部分。如果您有多个应用程序，您可以使用单独的逻辑名称，这需要一个类似于以下内容的自定义 JSON。

```
{
  "opsworks_java": {
    "datasources": {
      "myjavaapp": "jdbc/myappdb",
      "simplejsp": "jdbc/myjspdb",
      ...
    }
  }
}
```

Node.js App Server AWS OpsWorks 堆栈层

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 此层仅适用于基于 Linux 的堆栈。

Node.js App Server 层是一个 AWS OpsWorks Stacks 层，它为充当 Node.js 应用程序服务器的实例提供蓝图。 AWS OpsWorks Stacks 还会安装 Express，因此该层的实例同时支持标准和 Express 应用程序。

安装：Node.js 安装在 `/usr/local/bin/node` 中。

Add Layer 页面提供以下配置选项：

Node.js 版本

有关当前支持的版本的列表，请参阅AWS OpsWorks 堆栈操作系统。

自定义安全组

如果您选择不自动将内置 AWS OpsWorks Stacks 安全组与您的图层关联，则会显示此设置。您必须指定要将哪一安全组与层关联起来。有关更多信息，请参阅 创建新堆栈。

Elastic Load Balancer

您可以将 Elastic Load Balancing 负载均衡器连接到层的实例。

> **Important**
>
> 如果您的 Node.js 应用程序使用的是 SSL，我们建议您禁用 SSLv3 (如果可能的话)，以应对 CVE-2015-8027 中介绍的漏洞。为此，您必须将 Node.js version 设置为 `0.12.9`。

部署 Node.js 应用程序

有关如何为 AWS OpsWorks Stacks 实施简单的 Node.js 应用程序并将其部署到一个堆栈的详细演练，请参阅创建您的第一个 Node.js 堆栈。一般而言， AWS OpsWorks Stacks 的 Node.js 应用程序应满足以下条件：

- 主文件必须命名为 `server.js` 且驻留在部署的应用程序的根目录下。
- Express 应用程序必须将 `package.json` 文件放在应用程序的根目录下。
- 默认情况下，应用程序必须侦听端口 80 (HTTP) 或端口 443 (HTTPS)。

  可以在其他端口上进行侦听，但是 Node.js App Server 层的内置安全组 AWS OpsWorks-nodejs-App-Server 仅允许进入端口 80、443 和 22 (SSH) 的入站用户流量。要允许流入其他端口的入站用户流量，请使用适当的入站规则创建安全组并将其分配给 Node.js App Server 层。请勿通过编辑内置安全组来修改入站规则。每次创建堆栈时，Stac AWS OpsWorks ks 都会使用标准设置覆盖内置安全组，因此您所做的任何更改都将丢失。

> **Note**
>
> AWS OpsWorks Stacks 将 PORT 环境变量设置为 80（默认）或 443（如果您启用 SSL），因此您可以使用以下代码来侦听请求。

```
app.listen(process.env.PORT);
```

如果您将 Node.js 应用程序配置为支持 SSL，则必须指定密钥和证书。 AWS OpsWorks Stacks 将每个应用服务器实例的数据作为单独的文件放在/srv/www/*app_shortname*/shared/config目录中，如下所示。

- ssl.crt- SSL 证书。
- ssl.key- SSL 密钥。
- ssl.ca-链证书（如果您已经指定了一个）。

您的应用程序可以从这些文件中获取 SSL 密钥和证书。

PHP 应用服务器 AWS OpsWorks 堆栈层

Important

该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

Note

此层仅适用于基于 Linux 的堆栈。

PHP App Server 层是一个 AWS OpsWorks 堆栈层，它为充当 PHP 应用程序服务器的实例提供蓝图。PHP App Server 层基于包含 mod_php 的 Apache2，没有标准配置选项。PHP 和 Apache 版本取决于您为层的实例指定了哪一操作系统。

操作系统	PHP 版本	Apache 版本
Amazon Linux 2018.03	5.3	2.2

操作系统	PHP 版本	Apache 版本
Amazon Linux 2017.09	5.3	2.2
Amazon Linux 2017.03	5.3	2.2
Amazon Linux 2016.09	5.3	2.2
Amazon Linux 2016.03	5.3	2.2
Amazon Linux 2015.09	5.3	2.2
Amazon Linux 2015.03	5.3	2.2
Amazon Linux 2014.09	5.3	2.2
Ubuntu 14.04 LTS	5.5	2.4

安装： AWS OpsWorks Stacks 使用实例的软件包安装程序mod_php在默认位置安装 Apache2。有关安装的更多信息，请参阅 Apache。

Add Layer 页面提供以下配置选项：

自定义安全组

如果您选择不自动将内置 AWS OpsWorks Stacks 安全组与您的图层关联，则会显示此设置。您必须指定要将哪一安全组与层关联起来。有关更多信息，请参阅 创建新堆栈。

Elastic Load Balancer

您可以将 Elastic Load Balancing 负载均衡器连接到层的实例。

您可以使用自定义 JSON 或自定义属性文件来修改某些 Apache 配置设置。有关更多信息，请参阅 覆盖属性。要查看可以覆盖的 Apache 属性的列表，请参阅apache2 属性。

有关如何部署 PHP 应用程序 (包括如何将应用程序连接到后端数据库) 的示例，请参阅Chef 11 Linux 堆栈入门。

> **Important**
>
> 如果您的 PHP 应用程序使用的是 SSL，我们建议您禁用 SSLv3 (如果可能的话)，以应对 CVE-2014-3566 中介绍的漏洞。为此，您必须修改 Apache 服务器的 SSLProtocol 文件中的 ssl.conf 设置。有关如何修改此设置的更多信息，请参阅为 Apache 服务器禁用 SSLv3。

Rails 应用程序服务器 AWS OpsWorks 堆栈层

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 此层仅适用于基于 Linux 的堆栈。

Rails App Server 层是一个 AWS OpsWorks 堆栈层，它为充当 Rails 应用程序服务器的实例提供蓝图。

安装： AWS OpsWorks Stacks 使用实例的软件包安装程序将服务器软件包安装到其默认位置。有关 Apache/Passenger 安装的更多信息，请参阅 Phusion Passenger。有关日志记录的更多信息，请参阅 Log Files。有关 Nginx/Unicorn 安装的更多信息，请参阅 Unicorn。

Add Layer 页面提供以下配置选项，所有这些选项都是可选的。

Ruby 版本

您的应用程序将使用的 Ruby 版本。默认值为 2.3。

您还可以通过覆盖 [:opsworks][:ruby_version] 属性来指定您的首选 Ruby 版本。

> **Note**
>
> AWS OpsWorks Stacks 会安装一个单独的 Ruby 软件包，供配方和实例代理使用。有关更多信息，请参阅 Ruby 版本。

Rails 堆栈

默认 Rails 堆栈为包含 Phusion Passenger 的 Apache2。您还可以使用包含 Unicorn 的 Nginx。

> **Note**
>
> 如果您使用 Nginx 和 Unicorn，则必须将 unicorn gem 添加到您的应用程序的 Gemfile 中，如以下示例中所示：
>
> ```
> source 'https://rubygems.org'
> gem 'rails', '3.2.15'
> ...
> # Use unicorn as the app server
> gem 'unicorn'
> ...
> ```

Passenger 版本

如果您已经指定了 Apache2/Passenger，则您必须指定 Passenger 版本。默认值是 5.0.28。

Rubygems 版本

默认 Rubygems 版本是 2.5.1

安装并托管 Bundler

允许您选择是否安装并托管 Bundler。默认值是 Yes。

Bundler 版本

默认 Bundler 版本是 1.12.5。

自定义安全组

如果您选择不自动将内置 AWS OpsWorks Stacks 安全组与您的图层关联，则会显示此设置。您必须指定要将哪一安全组与层关联起来。有关更多信息，请参阅 创建新堆栈。

Elastic Load Balancer

您可以将 Elastic Load Balancing 负载均衡器连接到层的实例。

您可以使用自定义 JSON 或自定义属性文件来修改某些配置设置。有关更多信息，请参阅 覆盖属性。要查看可以覆盖的 Apache、Nginx、Phusion Passenger 和 Unicorn 属性的列表，请参阅内置说明书属性。

> **Important**
>
> 如果您的 Ruby on Rails 应用程序使用的是 SSL，我们建议您禁用 SSLv3 (如果可能的话)，以应对 CVE-2014-3566 中介绍的漏洞。有关更多信息，请参阅 对 Rails 服务器禁用 SSLv3。

主题

- 对 Rails 服务器禁用 SSLv3
- 连接到数据库
- 部署 Ruby on Rails 应用程序

对 Rails 服务器禁用 SSLv3

要对 Rails 服务器禁用 SSLv3，请将层的 Ruby Version (Ruby 版本) 设置更新为 2.1 或更高版本，这样就会安装 Ruby 2.1.4 或更高版本作为应用程序要使用的版本。

- 将层的 Ruby Version (Ruby 版本) 设置更新为 2.1 或更高版本。
- 为您的 Rails 堆栈更新配置文件，如下所示。

带 Phusion Passenger 的 Apache

更新 Apache 服务器的 `SSLProtocol` 文件中的 `ssl.conf` 设置，如为 Apache 服务器禁用 SSLv3中所述。

带 Unicorn 的 Nginx

在 Nginx 服务器的 `ssl_protocols` 文件中添加一个显式 `nginx.conf` 指令。要禁用 SSLv3，覆盖内置 nginx 说明书的 `nginx.conf.erb` 模板文件，Rails App Server 层的 Setup 配方使用该文件来创建 `nginx.conf`，并添加以下指令：

```
ssl_protocols TLSv1.2;
```

有关如何配置 nginx.conf 的更多信息，请参阅 Configuring HTTPS servers。有关如何覆盖内置模板的更多信息，请参阅使用自定义模板。

连接到数据库

部署应用程序时， AWS OpsWorks Stacks 会使用来自应用程序deploy属性的信息创建一个新database.yml文件。如果您将 MySQL 或 Amazon RDS 实例附加到应用程序， AWS OpsWorks Stacks 会将连接信息添加到deploy属性中，从而database.yml自动包含正确的连接数据。

如果应用程序没有附加数据库，则默认情况下， AWS OpsWorks Stacks 不会向deploy属性添加任何连接信息，也不会创建database.yml。如果您希望使用不同的数据库，您可以使用自定义 JSON 将数据库属性以及连接信息添加到应用程序的 deploy 属性。属性全部位于下方["deploy"]["*appshortname*"]["database"]，其中 *appshortn* ame 是应用程序的短名称， AWS OpsWorks Stacks 根据应用程序名称生成。您在自定义 JSON 中指定的值将覆盖任何默认设置。有关更多信息，请参阅 添加应用程序。

AWS OpsWorks 堆栈将以下[:...][:database]属性值合并到。database.yml所需的属性取决于特定的数据库，但必须有host属性，否则 AWS OpsWorks 堆栈将无法创建database.yml。

- [:adapter] (String)-数据库适配器，如 mysql。
- [:database]（字符串）-数据库名称。
- [:encoding]（字符串）-编码，这通常被设置为 utf8。
- [:host]（字符串）-主机 URL，如 railsexample.cdlqlk5uwd0k.us-west-2.rds.amazonaws.com。
- [:reconnect]（布尔值）-如果连接不再存在，应用程序是否重新连接。
- [:password]（字符串）-数据库密码。
- [:port] (数字)。-数据库的端口号。使用此属性覆盖默认端口号，该属性由适配器设置。
- [:username]（字符串）-数据库用户名称。

以下示例显示了其短名称为 myapp 的应用程序的自定义 JSON。

```
{
  "deploy" : {
```

```
    "myapp" : {
      "database" : {
        "adapter" : "adapter",
        "database" : "databasename",
        "host" : "host",
        "password" : "password",
        "port" : portnumber
        "reconnect" : true/false,
        "username" : "username"
      }
    }
  }
}
```

有关如何指定自定义 JSON 的信息，请参阅使用自定义 JSON。要查看用于创建 `database.yml` (`database.yml.erb`) 的模板，请转至内置说明书存储库。

部署 Ruby on Rails 应用程序

您可以在任一受支持的存储库中部署 Ruby on Rails 应用程序。下面的内容介绍了如何将示例 Ruby on Rails 应用程序部署到运行 Apache/Passenger Rails 堆栈的服务器中。示例代码存储在公共存储 GitHub库中，但其他支持的存储库的基本过程相同。有关如何创建和部署应用程序的更多信息，请参阅应用程序。要查看包含大量评论的示例代码，请访问 https://github.com/awslabs/ opsworks-demo-rails-photo-share-app。

从 GitHub 存储库部署 Ruby on Rails 应用程序

1. 创建包含 Rails App Server 层的堆栈 (Apache/Passenger 作为 Rails 堆栈)，将一个全天候实例添加到层，然后启动实例。
2. 当实例处于联机状态后，将一个应用程序添加到堆栈，并指定以下设置：

   - Name - 您喜欢的任何名称；示例中使用的是 `PhotoPoll`。

     AWS OpsWorks Stacks 使用此名称进行显示，并生成一个短名称供内部使用，并在堆栈配置和部署属性中标识应用程序。例如， PhotoPoll 简称是 photopoll。

   - App type - Ruby on Rails。
   - Rails environment - 可用的环境由应用程序决定。

     示例应用程序有三个：**`development`**、**`test`** 和 **`production`**。在此示例中，将环境设置为 **`development`**。查看每个环境的描述的示例代码。

- Repository type -任何受支持的存储库类型。在此示例中指定 `Git`
- Repository URL - 将从其中部署代码的存储库。

  在此示例中，将 URL 设置为 **`git://github.com/awslabs/opsworks-demo-rails-photo-share-app`**。

对其余设置使用默认值，然后单击 Add App 创建应用程序。

3. 将应用程序部署到 Rails App Server 实例。
4. 部署完成后，转到 实例 页面，然后单击 Rails App Server 实例的公有 IP 地址。您将看到以下内容：

静态 Web 服务器 AWS OpsWorks 堆栈层

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 此层仅适用于基于 Linux 的堆栈。

静态 Web 服务器层是一个 AWS OpsWorks 堆栈层，它为提供静态 HTML 页面的实例提供模板，其中可能包括客户端脚本。此层基于 Nginx。

安装：Nginx 安装在 `/usr/sbin/nginx` 中。

Add Layer 页面提供以下配置选项：

自定义安全组

如果您选择不自动将内置 AWS OpsWorks Stacks 安全组与您的图层关联，则会显示此设置。您必须指定要将哪一安全组与层关联起来。有关更多信息，请参阅 创建新堆栈。

Elastic Load Balancer

您可以将 Elastic Load Balancing 负载均衡器连接到层的实例。

您可以使用自定义 JSON 或自定义属性文件来修改某些 Nginx 配置设置。有关更多信息，请参阅 覆盖属性。要查看可以覆盖的 Apache 属性的列表，请参阅nginx 属性。

> **Important**
>
> 如果您的 Web 应用程序使用的是 SSL，我们建议您禁用 SSLv3 (如果可能的话)，以应对 CVE-2014-3566 中介绍的漏洞。
> 要禁用 SSLv3，您必须修改 Nginx 服务器的 `nginx.conf` 文件。为此，覆盖内置 nginx 说明书的 `nginx.conf.erb` 模板文件，Rails App Server 层的 Setup 配方使用该文件来创建 `nginx.conf`，并添加以下指令：
>
> ```
> ssl_protocols TLSv1.2;
> ```
>
> 有关如何配置 `nginx.conf` 的更多信息，请参阅 Configuring HTTPS servers。有关如何覆盖内置模板的更多信息，请参阅使用自定义模板。

ECS 集群层参考

> **Note**
>
> 此层仅适用于基于 Linux 的堆栈。

ECS 集群层代表 Amazon Elastic Container Service (Amazon ECS) 集群，可简化集群管理。

短名称：ecs-cluster

兼容性：Amazon ECS 服务层仅与自定义层兼容

开放端口：ECS Cluster 允许公开访问端口 22 (SSH)

自动分配弹性 IP 地址：默认情况下关闭

默认 EBS 卷：否

默认安全组：AWS--EC S OpsWorks-Cluster

配置：要配置 ECS Cluster 层，您必须指定以下内容：

- 是否为容器实例分配公有 IP 地址或弹性 IP 地址
- 容器实例的实例配置文件

Setup 配方：

- opsworks_initial_setup
- ssh_host_keys
- ssh_users
- mysql::client
- dependencies
- ebs
- opsworks_ganglia::client
- opsworks_ecs::setup

Configure 配方：

- opsworks_ganglia::configure-client
- ssh_users
- mysql::client
- agent_version
- opsworks_ecs::configure

Deploy 配方：

- deploy::default
- opsworks_ecs::deploy

UnDeploy 配方：

- opsworks_ecs::undeploy

Shutdown 配方：

- opsworks_shutdown::default
- opsworks_ecs::shutdown

安装:

- AWS OpsWorks Stacks 使用实例的软件包安装程序将 Docker 安装到其默认位置
- Setup 事件的 Chef 日志记录 Amazon ECS 代理是否已成功安装。否则，AWS OpsWorks Stacks 提供的日志不包含 Amazon ECS 错误日志信息。有关如何处理 Amazon ECS 错误的更多信息，请参阅 Amazon ECS 故障排除。

自定义层参考

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

如果标准层无法满足您的需求，您可以创建自定义层。一个堆栈可以拥有多个自定义层。默认情况下，自定义层会运行有限的几个支持基本功能的标准配方。然后，您可以通过为每个适当的生命周期事件实施一组自定义 Chef 配方来设置和配置层的软件等等，从而设施层的主要功能。自定义食谱在每个事件的标准 AWS OpsWorks 堆栈配方之后运行。

短名称：由用户定义；堆栈中的每个自定义层都必须拥有不同的短名称

开放端口：默认情况下，自定义服务器层允许公开访问端口 22(SSH)、80 (HTTP)、443 (HTTPS) 以及堆栈的 Rails 和 PHP 应用程序服务器层的所有端口

自动分配弹性 IP 地址：默认情况下关闭

默认 EBS 卷：否

默认安全组：AWS OpsWorks-自定义服务器

兼容性：自定义层与以下层兼容：自定义、db-master、lb、memcached、monitoring-master、nodejs-app、php-app、rails-app 和 web

配置：要配置自定义层，您必须指定以下内容：

- 层的名称
- 层的短名称，它用于标识 Chef 配方中的层且只能使用字母 a-z 和数字

对于 Linux 堆栈，自定义层使用以下配方。

Setup 配方：

- opsworks_initial_setup
- ssh_host_keys
- ssh_users
- mysql::client
- dependencies
- ebs
- opsworks_ganglia::client

Configure 配方：

- opsworks_ganglia::configure-client
- ssh_users
- agent_version

Deploy 配方：

- deploy::default

Shutdown 配方：

- opsworks_shutdown::default

其他层参考

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

AWS OpsWorks 堆栈还支持以下图层。

主题

- Ganglia 层参考
- Memcached 层参考

Ganglia 层参考

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 此层仅适用于基于 Linux 的堆栈。

Ganglia 层支持 Ganglia，这是一种用于管理实例指标的存储和可视化的分布式监控系统。它旨在与分层实例拓扑一起使用，这种拓扑使它非常适用于实例组。Ganglia 拥有两个基本组件：

- 低开销客户端，它安装在堆栈中的每个实例上，并向主机发送指标。
- 主机，它从客户端收集指标并将其存储在 Amazon EBS 卷中。它还会在网页上显示指标。

AWS OpsWorks Stacks 在其管理的每个实例上都有一个 Ganglia 监控代理。当您在堆栈中添加 Ganglia 层并启动它时，每个实例上的 Ganglia 代理都会将指标报告给 Ganglia 实例。要使用 Ganglia，请在堆栈中添加带有一个实例的 Ganglia 层。您可以通过登录位于主机的 IP 地址的 Ganglia 后端来访问数据。您可以通过编写 Chef 配方来提供更多指标定义。

短名称：monitoring-master

兼容性：Ganglia 层与以下层兼容：自定义、db-master、memcached、php-app、rails-app。

开放端口：负载均衡器允许公开访问端口 22 (SSH)、80 (HTTP) 和 443 (HTTPS)。

自动分配弹性 IP 地址：默认情况下关闭

默认 EBS 卷：是，在 `/vol/ganglia` 中

默认安全组：AWS-监控 OpsWorks-主服务器

配置：要配置 Ganglia 层，您必须指定以下内容：

- 提供对监控图形的访问权限的 URI。默认值是 http://*DNSName*/ganglia，其中 *DNSName* 是 Ganglia 实例的 DNS 名称。
- 控制对监控统计数据的访问权限的用户名和密码。

Setup 配方：

- opsworks_initial_setup
- ssh_host_keys

- ssh_users
- mysql::client
- dependencies
- ebs
- opsworks_ganglia::client
- opsworks_ganglia::server

Configure 配方：

- opsworks_ganglia::configure-client
- ssh_users
- agent_version
- opsworks_ganglia::configure-server

Deploy 配方：

- deploy::default
- opsworks_ganglia::configure-server
- opsworks_ganglia::deploy

Shutdown 配方：

- opsworks_shutdown::default
- apache2::stop

安装:

- Ganglia 客户端安装在 `/etc/ganglia` 下。
- Ganglia Web 前端安装在 `/usr/share/ganglia-webfrontend` 下。
- Ganglia logtailer 安装在 `/usr/share/ganglia-logtailer` 下。

Memcached 层参考

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 此层仅适用于基于 Linux 的堆栈。

Memcached 是一种适用于任意数据的分布式内存缓存系统。它在内存中将字符串和对象作为关键字和值进行缓存，以减少必须读取外部数据源的次数，从而加快网站运行速度。

要在堆栈中使用 Memcached，请创建 Memcached 层并添加一个或多个实例，使其充当 Memcached 服务器。这些实例会自动安装 Memcached，而堆栈的其他实例则可以访问并使用 Memcached 服务器。如果您使用 Rails App Server 层， AWS OpsWorks Stacks 会自动将`memcached.yml`配置文件放置在该层中每个实例的配置目录中。您可以从此文件中获得 Memcached 服务器和端口号。

短名称：memcached

兼容性：Memcached 层与以下层兼容：自定义、db-master、lb、monitoring-master、nodejs-app、php-app、rails-app 和 web。

开放端口：Memcached 层允许公开访问端口 22(SSH) 和堆栈的 Web 服务器、自定义服务器以及 Rails、PHP 和 Node.js 应用程序服务器的所有端口。

自动分配弹性 IP 地址：默认情况下关闭

默认 EBS 卷：否

默认安全组：AWS--Memcac hed OpsWorks-Server

要配置 Memcached 层，您必须指定缓存大小 (以 MB 为单位)。

Setup 配方：

- opsworks_initial_setup
- ssh_host_keys

- ssh_users
- mysql::client
- dependencies
- ebs
- opsworks_ganglia::client
- memcached

Configure 配方：

- opsworks_ganglia::configure-client
- ssh_users
- agent_version

Deploy 配方：

- deploy::default

Shutdown 配方：

- opsworks_shutdown::default
- memcached::stop

安装:

- AWS OpsWorks Stacks 使用实例的软件包安装程序将 Memcached 及其日志文件安装到其默认位置。

其他层

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 这些层仅适用于 Chef 11 和更早的基于 Linux 的堆栈。

AWS OpsWorks Stacks 还支持 Ganglia 和 Memcached 图层。

主题

- Ganglia 层
- Memcached

Ganglia 层

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 该层仅适用于 Chef 11 和更早的基于 Linux 的堆栈。

AWS OpsWorks Stacks 会将您的所有实例和容量指标发送到 Amazon CloudWatch，这样您就可以轻松查看图表和设置警报，从而帮助您进行故障排除并根据资源状态自动采取行动。您还可以将 Ganglia AWS OpsWorks Stacks 层用于其他应用程序监控选项，例如存储所选指标。

Ganglia 层是实例的蓝图，通过使用 Ganglia 分布式监控来监控您的堆栈。堆栈通常只有一个 Ganglia 实例。该 Ganglia 层包括以下可选配置设置：

Ganglia URL

统计数据的 URL 路径。完整的 URL 是 http://*DNSNameURLPath*，其中 *DNSName* 是关联实例的 DNS 名称。默认 *URLPath* 值是“/ganglia”，它与如下内容对应：http://ec2-54-245-151-7.us-west-2.compute./ganglia。

Ganglia user name

统计信息网页的用户名称。当您查看此页面时，必须提供用户名称。默认值为 “Opsworks”。

Ganglia password

用于控制访问统计信息网页的密码。当您查看此页面时，必须提供该密码。默认值是一个随机生成的字符串。

> **Note**
>
> 记录密码以备后用； AWS OpsWorks Stacks 不允许您在创建图层后查看密码。不过，您可以通过转到层的“Edit”页面并单击 Update password 来更新密码。

自定义安全组

如果您选择不自动将内置 AWS OpsWorks Stacks 安全组与您的图层关联，则会显示此设置。您必须指定要将哪一安全组与层关联起来。有关更多信息，请参阅 创建新堆栈。

Elastic Load Balancer

您可以将 Elastic Load Balancing 负载均衡器连接到层的实例。

> **Important**
>
> 如果您的堆栈包含 Ganglia 层，我们建议您尽可能禁用该层的 SSLv3，以解决 CVE-2014-3566 中所述的漏洞。为此，您必须覆盖 Apache 服务器的 `ssl.conf.erb` 模板来修改 `SSLProtocol` 设置。有关更多信息，请参阅 为 Apache 服务器禁用 SSLv3。

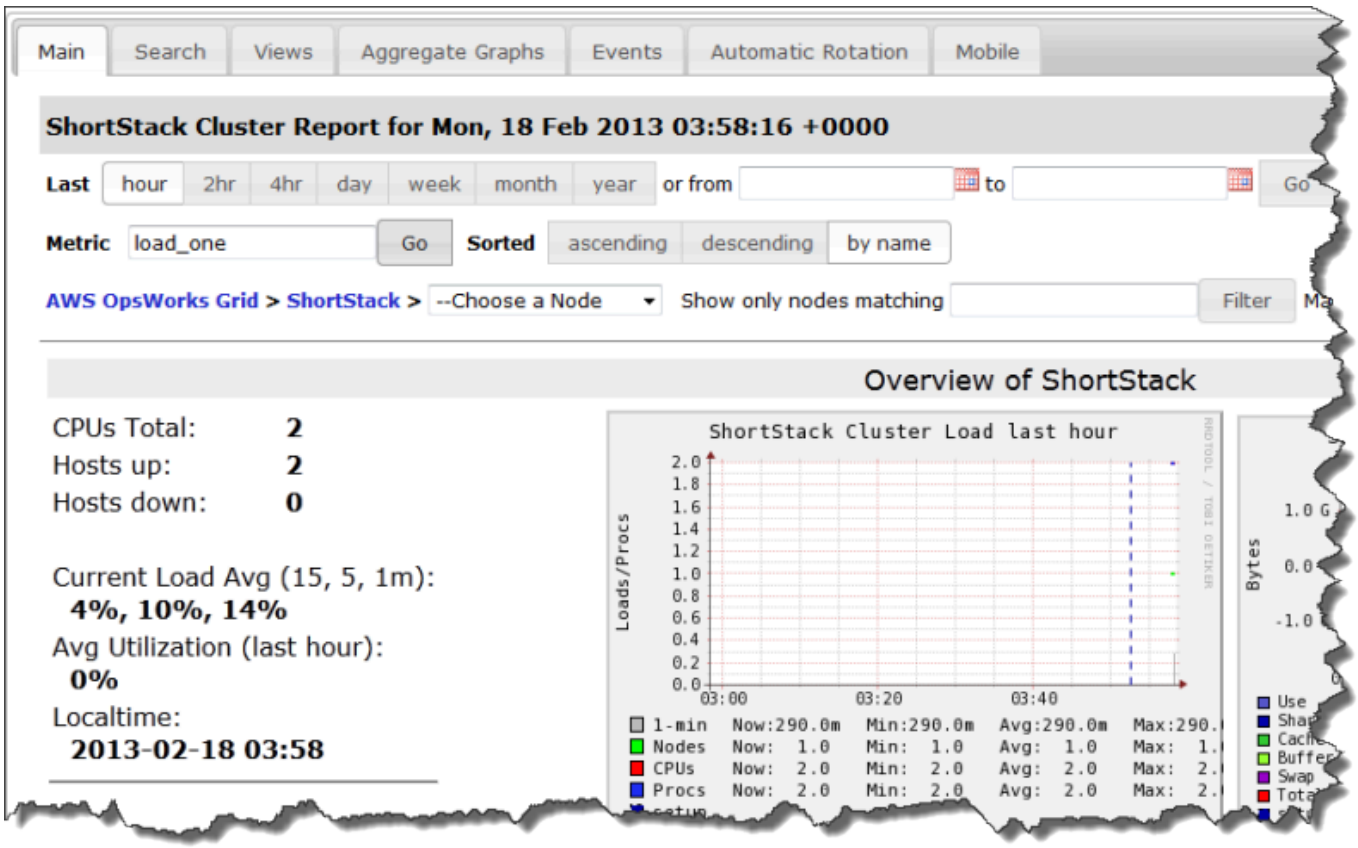
查看 Ganglia 统计信息

AWS OpsWorks Stacks 配方会在每个实例上安装一个低开销的 Ganglia 客户端。如果您的堆栈包含 Ganglia 层，在实例联机后，Ganglia 客户端将自动开始向 Ganglia 报告。Ganglia 使用此客户端数据计算各种统计信息，并在其统计信息网页上以图形化的方式显示结果。

查看 Ganglia 统计信息

1. 在 层 页面上，单击 Ganglia 以打开层的详细信息页面。

2. 在导航窗格中，单击 Instances。在 Ganglia 下，单击实例名称。
3. 复制实例的 Public DNS 名称。
4. 使用 DNS 名称来构建统计信息的 URL，这将类似于：http://ec2-54-245-151-7.us-west-2.compute.amazonaws.com/ganglia。
5. 将完整的 URL 粘贴到您的浏览器中，导航到该页面，然后输入 Ganglia 用户名称和密码以显示页面。下面是一个示例。



Memcached

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 此层仅适用于 Chef 11 及更低版本的基于 Linux 的堆栈。

Memcached 层是一个 AWS OpsWorks 堆栈层，它为充当 Memcached 服务器的实例提供蓝图，Memcached 服务器是一种用于存储任意数据的分布式内存缓存系统。该 Memcached 层包含以下配置设置。

Allocated memory (MB)

(可选) 每个层实例的缓存内存量 (MB)。默认值为 512 MB。

自定义安全组

如果您选择不自动将内置 AWS OpsWorks Stacks 安全组与您的图层关联，则会显示此设置。您必须指定要将哪一安全组与层关联起来。有关更多信息，请参阅 创建新堆栈。

说明书组件

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

一个说明书通常包含以下基本组件：

- 属性文件包含一组表示要由配方和模板使用的值的属性。
- 模板文件是配方用于创建其他文件 (如配置文件) 的模板。

  模板文件通常允许您通过覆盖属性（此操作可在不接触到说明书的情况下完成）来修改配置文件而不是重写配置文件。标准做法是：只要您希望更改某个实例上的配置文件 (甚至是略微更改)，您就应使用模板文件。
- 配方文件是用于定义配置系统所需的一切内容的 Ruby 应用程序，包括创建和配置文件夹、安装和配置程序包、启动服务等。

说明书不必包含全部这三个组件。进行自定义的更简单方法只需属性或模板文件。此外，说明书可有选择地包含其他文件类型，如定义或规范。

本部分介绍了三个标准说明书组件。有关更多信息，特别是有关如何实施配方的信息，请参阅 Opscode。

主题

- Attributes
- 模板
- 配方

Attributes

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

配方和模板取决于各种值，如配置设置。您可以创建一个包含表示每个值的属性的属性文件，而不是直接在配方或模板中对这些值进行硬编码。然后，在您的配方或模板中使用这些属性而不是显式值。使用属性的优点是，您可以覆盖属性值而不用接触到说明书。因此，您应始终使用属性来定义以下类型的值：

- 因堆栈不同或随时间可能发生变化的值，如用户名。

  如果对这类值进行硬编码，您必须在每次需要更改值时更改相应的配方或模板。通过使用属性定义这些值，您可以针对每个堆栈使用相同的说明书，而且只需覆盖相应的属性。
- 敏感值，如密码或私有密钥。

  将显式敏感值放置于您的说明书中可能会增加暴露的风险。相反，应使用虚拟值定义属性，然后覆盖它们以设置实际值。覆盖这类属性的最佳方法是使用自定义 JSON。有关更多信息，请参阅 使用自定义 JSON。

有关属性以及如何覆盖它们的更多信息，请参阅覆盖属性。

以下示例是示例属性文件的一部分。

```
...
default["apache"]["listen_ports"] = [ '80','443' ]
default["apache"]["contact"] = 'ops@example.com'
default["apache"]["timeout"] = 120
default["apache"]["keepalive"] = 'Off'
default["apache"]["keepaliverequests"] = 100
default["apache"]["keepalivetimeout"] = 3
default["apache"]["prefork"]["startservers"] = 16
default["apache"]["prefork"]["minspareservers"] = 16
default["apache"]["prefork"]["maxspareservers"] = 32
default["apache"]["prefork"]["serverlimit"] = 400
default["apache"]["prefork"]["maxclients"] = 400
default["apache"]["prefork"]["maxrequestsperchild"] = 10000
...
```

AWS OpsWorks Stacks 使用以下语法定义属性：

```
node.type["attribute"]["subattribute"]["..."]=value
```

您也可以使用冒号 (:)，如下所示：

```
node.type[:attribute][:subattribute][:...]=value
```

一个属性定义包含以下部分：

node.

node. 前缀是可选的且通常会被省略，如示例中所示。

type

该类型决定该属性是否可以被覆盖。 AWS OpsWorks 堆栈属性通常使用以下类型之一：

- default 是最常用类型，因为它允许覆盖属性。
- normal定义一个属性，该属性会覆盖标准 AWS OpsWorks 堆栈属性值之一。

> **Note**
>
> Chef 支持其他类型，这些类型对于 AWS OpsWorks Stacks 来说不是必需的，但可能对你的项目有用。有关更多信息，请参阅关于属性。

attribute name

属性名称使用标准 Chef 节点语法：`[:attribute][:subattribute][...]`。您可以为您的属性使用您喜欢的任何名称。不过，如覆盖属性中所讨论的，自定义说明书属性与来自堆栈配置的属性和部署属性，以及 Chef 的 Ohai 工具一起，合并到实例的节点对象中。常用配置名称 (如 port 或 user) 可能会显示在各种说明书中。

为避免名称冲突，通常的做法是创建至少包含两个元素的限定属性名称，如示例中所示。第一个元素应是唯一的且通常基于产品名称，如 Apache。它后接一个或多个标识特定值的子属性，如 `[:user]` 或 `[:port]`。您可以根据项目的需要使用任意数量的子属性。

value

一个属性可设置为以下类型的值：

- 字符串，如 `default[:apache][:keepalive] = 'Off'`。
- 数字 (不带引号)，如 `default[:apache][:timeout] = 120`。
- 布尔值，可以是 `true` 或 `false` (不带引号)。
- 值列表，如 `default[:apache][:listen_ports] = [ '80','443' ]`

属性文件是一个 Ruby 应用程序，因此您还可以使用节点语法和逻辑运算符来基于其他属性分配值。有关如何定义属性的更多信息，请参阅关于属性。有关工作属性文件的示例，请参阅 AWS OpsWorks Stacks 内置食谱，网址为 https://github.com/aws/opsworks-cookbooks。

模板

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

您通过创建配置文件并将它放置在合适的目录中来配置许多程序包。您可以在说明书中包含一个配置文件并将它复制到合适的目录中，但更为灵活的方法是让您的配方从一个模板创建配置文件。模板的一个优势是，您可以使用属性来定义模板的值。例如，这允许您通过使用自定义 JSON 覆盖相应的属性值来修改配置文件，而不用接触到说明书。

模板与关联的文件具有基本相同的内容和结构。下面给出了一个示例文件 `httpd.conf`。

```
ServerRoot "<%= node[:apache][:dir] %>"
<% if node[:platform] == "debian" || node[:platform] == "ubuntu" -%>
  LockFile /var/lock/apache2/accept.lock
<% else -%>
   LockFile logs/accept.lock
<% end -%>
PidFile <%= node[:apache][:pid_file] %>
Timeout <%= node[:apache][:timeout] %>
KeepAlive <%= node[:apache][:keepalive] %>
MaxKeepAliveRequests <%= node[:apache][:keepaliverequests] %>
KeepAliveTimeout <%= node[:apache][:keepalivetimeout] %>
<IfModule mpm_prefork_module>
    StartServers          <%= node[:apache][:prefork][:startservers] %>
    MinSpareServers       <%= node[:apache][:prefork][:minspareservers] %>
    MaxSpareServers       <%= node[:apache][:prefork][:maxspareservers] %>
    ServerLimit           <%= node[:apache][:prefork][:serverlimit] %>
    MaxClients            <%= node[:apache][:prefork][:maxclients] %>
    MaxRequestsPerChild   <%= node[:apache][:prefork][:maxrequestsperchild] %>
</IfModule>
...
```

以下示例是一个为 Ubuntu 实例生成的 `httpd.conf` 文件：

```
ServerRoot "/etc/httpd"
LockFile logs/accept.lock
PidFile /var/run/httpd/httpd.pid
Timeout 120
KeepAlive Off
MaxKeepAliveRequests 100
KeepAliveTimeout 3
<IfModule mpm_prefork_module>
    StartServers          16
    MinSpareServers       16
    MaxSpareServers       32
```

```
    ServerLimit           400
    MaxClients            400
    MaxRequestsPerChild   10000
</IfModule>
...
```

此模板的许多文本只是简单地从模板复制到 `httpd.conf` 文件。但是，`<%= ... %>` 内容的处理方式如下所示：

- Chef 将 `<%= node[:attribute][:sub_attribute][:...]%>` 替换为属性的值。

  例如，`StartServers <%= node[:apache][:prefork][:startservers] %>` 变为 `httpd.conf` 中的 `StartServers 16`。
- 您可以使用 `<%if-%>`, `<%else-%>`, and `<%end-%>` 来有条件地选择一个值。

  该示例根据平台为 `accept.lock` 设置了一个不同的文件路径。

> **Note**
>
> 您并非仅限于使用说明书的属性文件中的属性。您可以使用实例的节点对象中的任何属性。例如，由名为 Ohai 的 Chef 工具生成且合并到节点对象中。有关属性的更多信息，请参阅覆盖属性。

有关模板的更多信息，包括如何合并 Ruby 代码，请参阅关于模板。

配方

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

配方是用于定义系统配置的 Ruby 应用程序。它们可安装程序包、从模板创建配置文件、执行 shell 命令、创建文件和目录等。当实例发生生命周期事件时，你通常会让 AWS OpsWorks Stacks 自动执行配方，但你也可以使用 E xecute Recipes 堆栈命令随时显式运行它们。有关更多信息，请参阅关于配方。

配方通常主要包含一系列资源，每种资源均表示系统的某方面的所需状态。每种资源包含一组属性，用于定义所需状态并指定要采取的操作。Chef 将每种资源与执行操作的相应提供商进行关联。有关更多信息，请参阅资源和提供商参考。

package 资源可帮助您管理 Linux 实例上的软件包。下面的示例将安装 Apache 程序包。

```
...
package 'apache2' do
  case node[:platform]
  when 'centos','redhat','fedora','amazon'
    package_name 'httpd'
  when 'debian','ubuntu'
    package_name 'apache2'
  end
  action :install
end
...
```

Chef 针对平台使用合适的程序包提供商。通常只向资源属性分配一个值，但您可以使用 Ruby 逻辑运算执行条件分配。该示例使用一个 case 运算符，该运算符使用 node[:platform] 识别实例的操作系统并相应地设置 package_name 属性。您可以通过使用标准 Chef 节点语法将属性插入配方中，然后 Chef 会将它替换为关联的值。您可以使用节点对象中的任何属性，而不仅仅是说明书的属性。

在确定合适的程序包名称后，代码段会以用于安装程序包的 install 操作结束。此资源的其他操作包括 upgrade 和 remove。有关更多信息，请参阅程序包。

将复杂的安装和配置任务拆分为一个或多个子任务 (每个子任务作为单独的配方执行) 并且在适当的时间让您的主配方运行这些子任务通常很有用。以下示例显示了沿袭前一个示例的代码行：

```
include_recipe 'apache2::service'
```

要让配方执行子配方，请使用 include_recipe 关键字并后跟配方名称。使用标准 Chef *CookbookName*::*RecipeName* 语法标识配方，其中 *RecipeName* 省略了 .rb 扩展名。

> **Note**
>
> include_recipe 语句在主配方中的此位置有效地执行配方。但是，实际发生的情况是，Chef 在执行主配方前会将每个 include_recipe 语句替换为指定配方的代码。

directory 资源表示一个目录，如用于包含程序包的文件的目录。以下 default.rb 资源创建 Linux 日志目录。

```
directory node[:apache][:log_dir] do
    mode 0755
    action :create
end
```

此日志目录在说明书的一个属性文件中进行定义。资源将目录的模式指定为 0755 并使用 create 操作来创建目录。有关更多信息，请参阅目录。您还可以将此资源与 Windows 实例一起使用。

execute 资源表示命令，如 shell 命令或脚本。以下示例生成 module.load 文件。

```
execute 'generate-module-list' do
  if node[:kernel][:machine] == 'x86_64'
    libdir = 'lib64'
  else
    libdir = 'lib'
  end
  command "/usr/local/bin/apache2_module_conf_generate.pl /usr/#{libdir}/httpd/
modules /etc/httpd/mods-available"
  action :run
end
```

资源首先确定 CPU 类型。[:kernel][:machine] 是 Chef 生成的用来表示各种系统属性的另一种自动属性（在此例中为 CPU 类型）。然后，它指定命令 (Perl 脚本) 并使用 run 操作运行该脚本，从而生成 module.load 文件。有关更多信息，请参阅 execute。

template 资源表示要从说明书的模板文件之一生成的一种文件，通常是配置文件。以下示例从在模板中所讨论的 apache2.conf.erb 模板中创建 httpd.conf 配置文件。

```
template 'apache2.conf' do
  case node[:platform]
  when 'centos','redhat','fedora','amazon'
    path "#{node[:apache][:dir]}/conf/httpd.conf"
  when 'debian','ubuntu'
    path "#{node[:apache][:dir]}/apache2.conf"
  end
  source 'apache2.conf.erb'
```

```
  owner 'root'
  group 'root'
  mode 0644
  notifies :restart, resources(:service => 'apache2')
end
```

资源将基于实例的操作系统确定生成的文件的名称和位置。然后，它将 `apache2.conf.erb` 指定为要用于生成文件的模板并设置文件的所有者、组和模式。它运行 `notify` 操作来通知代表 Apache 服务器的 `service` 资源重新启动该服务器。有关更多信息，请参阅模板。

堆栈配置和部署属性：Linux

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

本主题包含最常用的堆栈配置和部署属性及其关联的节点语法。本主题是围绕 Linux 堆栈使用的堆栈配置命名空间结构组织的。请注意，同样的属性名称有时候会用于不同的目的，并且会在不同的命名空间中出现。例如，`id` 可能是指堆栈 ID、层 ID、应用程序 ID 等，因此您需要完全限定名称来使用属性值。将此数据可视化的一个简便方法是将其作为 JSON 对象。有关示例，请参阅 堆栈配置和部署属性。

> **Note**
>
> 在 Linux 实例上， AWS OpsWorks Stacks 除了将数据添加到节点对象之外，还会在每个实例上安装此 JSON 对象。您可使用代理 CLI 的 `get_json` 命令来检索它。

主题

- opsworks 属性
- opsworks_custom_cookbooks 属性
- dependencies 属性
- ganglia 属性
- mysql 属性
- passenger 属性

- opsworks_bundler 属性
- deploy 属性
- 其他顶级属性

opsworks 属性

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

`opsworks` 元素 (有时称为 `opsworks` 命名空间) 包含定义基本堆栈配置的一组属性。

> **Important**
>
> 建议不要覆盖 `opsworks` 命名空间中的属性值。这样做可能会导致内置配方失败。

主题

- applications
- instance 属性
- layers 属性
- rails_stack 属性
- stack 属性
- 其他顶级 opsworks 属性

applications

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

包含一个嵌入式对象列表，每个对象对应于堆栈的一个应用程序。每个嵌入式对象包含说明应用程序配置的以下属性。

> **Note**
>
> 这些属性的常规节点语法如下所示，其中 *i* 指定实例的从零开始的列表索引。
>
> ```
> node["opsworks"]["applications"]["i"]["attribute_name"]
> ```

application_type

应用程序的类型 (字符串)。可能值如下所示：

- php：PHP 应用程序
- rails：Ruby on Rails 应用程序
- java：Java 应用程序
- nodejs：Node.js 应用程序
- web：静态 HTML 页面
- other：所有其他应用程序类型

```
node["opsworks"]["applications"]["i"]["application_type"]
```

name

用户定义的显示名称，如 "SimplePHP" (字符串)。

```
node["opsworks"]["applications"]["i"]["name"]
```

slug_name

一个短名称，它是一个全小写的名称"simplephp"，例如由 OpsWorks 应用程序的名称（字符串）生成的名称。

```
node["opsworks"]["applications"]["i"]["slug_name"]
```

instance 属性

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

`instance` 属性包含指定此实例的配置的一组属性。

架构	availability_zone	backends
aws_instance_id	hostname	id
instance_type	ip	层
private_dns_name	private_ip	public_dns_name
region		

架构

实例的架构，如 "i386" (字符串)。

```
node["opsworks"]["instance"]["architecture"]
```

availability_zone

实例的可用区，如 "us-west-2a" (字符串)。

```
node["opsworks"]["instance"]["availability_zone"]
```

backends

后端 Web 进程的数量 (字符串)。例如，它可确定 HAProxy 将转发到 Rails 后端的并行连接的数量。默认值取决于实例的内存和内核数。

```
node["opsworks"]["instance"]["backends"]
```

aws_instance_id

EC2 实例 ID (字符串)。

```
node["opsworks"]["instance"]["aws_instance_id"]
```

hostname

主机名，如 "php-app1" (字符串)。

```
node["opsworks"]["instance"]["hostname"]
```

id

实例 ID，这是 AWS OpsWorks 堆栈生成的 GUID，用于唯一标识实例（字符串）。

```
node["opsworks"]["instance"]["id"]
```

instance_type

实例类型，如 "c1.medium" (字符串)。

```
node["opsworks"]["instance"]["instance_type"]
```

ip

公有 IP 地址 (字符串)。

```
node["opsworks"]["instance"]["ip"]
```

层

实例的层的列表，这些层由其短名称标识，如 "lb" 或 "db-master" (字符串列表)。

```
node["opsworks"]["instance"]["layers"]
```

private_dns_name

私有 DNS 名称 (字符串)。

```
node["opsworks"]["instance"]["private_dns_name"]
```

private_ip

私有 IP 地址 (字符串)。

```
node["opsworks"]["instance"]["private_ip"]
```

public_dns_name

公有 DNS 名称 (字符串)。

```
node["opsworks"]["instance"]["public_dns_name"]
```

region

Amazon Web Services Region，如 "us-west-2" (字符串)。

```
node["opsworks"]["instance"]["region"]
```

layers 属性

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

layers 属性包含一组层属性，每个属性对应于堆栈的一个层，这些属性是用层的短名称命名的，如 php-app。一个堆栈最多可以有一个内置层，后者的短名称如下所示：

- db-master: MySQL 层
- java-app: Java App Server 层
- lb: HAProxy 层
- monitoring-master: Ganglia 层
- memcached: 内存缓存层

- `nodejs-app`: Node.js App Server 层
- `php-app`: PHP App Server 层
- `rails-app`: Rails App Server 层
- `web`：静态 Web 服务器层

一个堆栈可包含任意数量的自定义层，这些层具有用户定义的短名称。

每个层属性包含以下属性：

- id
- 实例
- name

id

图层 ID，由图层生成 OpsWorks 并唯一标识图层（字符串）的 GUID。

```
node["opsworks"]["layers"]["layershortname"]["id"]
```

instances

`instances` 元素包含一组实例属性，每个实例对应于层的一个联机属性。这些属性是用实例的主机名命名的，如 `php-app1`。

> **Note**
>
> `instances` 元素仅包含创建特定堆栈配置和部署属性后处于联机状态的实例。

每个实例元素包含以下属性：

availability_zone	aws_instance_id	backends
booted_at	created_at	elastic_ip
instance_type	ip	private_ip

public_dns_name	private_dns_name	region
status		

availability_zone

可用区，如 "us-west-2a" (字符串)。

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]
["availability_zone"]
```

aws_instance_id

EC2 实例 ID (字符串)。

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]
["aws_instance_id"]
```

backends

后端 Web 进程的数量 (数字)。例如，它可确定 HAProxy 将转发到 Rails 后端的并行连接的数量。默认值取决于实例的内存和内核数。

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]
["backends"]
```

booted_at

EC2 实例的启动时间，使用 UTC yyyy-mm-ddd thh: mm: ss+HH: mm 格式（字符串）。例如，"2013-10-01T08:35:22+00:00" 对应于 2013 年 10 月 10 日 8:35:22，无时区偏移。有关更多信息，请参阅 ISO 8601。

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]
["booted_at"]
```

created_at

创建 EC2 实例的时间，使用 UTC yyyy-mm-ddd thh: mm: ss+HH: mm 格式（字符串）。例如，"2013-10-01T08:35:22+00:00" 对应于 2013 年 10 月 10 日 8:35:22，无时区偏移。有关更多信息，请参阅 ISO 8601。

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]
["created_at"]
```

elastic_ip

弹性 IP 地址，在实例没有 IP 地址时设置为 null (字符串)。

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]
["elastic_ip"]
```

instance_type

实例类型，如 "c1.medium" (字符串)。

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]
["instance_type"]
```

ip

公有 IP 地址 (字符串)。

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]
["ip"]
```

private_ip

私有 IP 地址 (字符串)。

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]
["private_ip"]
```

public_dns_name

公有 DNS 名称 (字符串)。

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]
["public_dns_name"]
```

private_dns_name

私有 DNS 名称 (字符串)。

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]
["private_dns_name"]
```

region

Amazon Web Services Region，如 "us-west-2" (字符串)。

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]
["region"]
```

status

状态 (字符串)。可能值如下所示：

- "requested"
- "booting"
- "running_setup"
- "online"
- "setup_failed"
- "start_failed"
- "terminating"
- "terminated"
- "stopped"
- "connection_lost"

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]
["status"]
```

name

层的名称，用于表示控制台中的层 (字符串)。它可以是用户定义的，并且不一定是唯一的。

```
node["opsworks"]["layers"]["layershortname"]["name"]
```

rails_stack 属性

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

name

指定 rails 堆栈，并将其设置为 "apache_passenger" 或 "nginx_unicorn" (字符串)。

```
node["opsworks"]["rails_stack"]["name"]
```

recipe

关联的配方，取决于您使用的是 Passenger 还是 Unicorn (字符串)：

- Unicorn：`"unicorn::rails"`
- Passenger: `"passenger_apache2::rails"`

```
node["opsworks"]["rails_stack"]["recipe"]
```

restart_command

重新启动命令，取决于您使用的是 Passenger 还是 Unicorn (字符串)：

- Unicorn：`"../../shared/scripts/unicorn clean-restart"`
- Passenger: `"touch tmp/restart.txt"`

服务

服务名称，取决于您使用的是 Passenger 还是 Unicorn (字符串)：

- Unicorn：`"unicorn"`
- Passenger: `"apache2"`

```
node["opsworks"]["rails_stack"]["service"]
```

stack 属性

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

`stack` 属性指定堆栈配置的某些方面 (如服务层配置)。

- elb-load-balancers
- id
- name
- rds_instances
- vpc_id

elb-load-balancers

包含一个嵌入式对象列表，每个对象对应于堆栈中的一个 Elastic Load Balancing 负载均衡器。每个嵌入式对象包含说明负载均衡器配置的以下属性。

> **Note**
>
> 这些属性的常规节点语法如下所示，其中 *i* 指定实例的从零开始的列表索引。
>
> ```
> node["opsworks"]["stack"]["elb-load-balancers"]["i"]["attribute_name"]
> ```

dns_name

负载均衡器的 DNS 名称 (字符串)。

```
node["opsworks"]["stack"]["elb-load-balancers"]["i"]["dns_name"]
```

name

负载均衡器的名称 (字符串)。

```
node["opsworks"]["stack"]["elb-load-balancers"]["i"]["name"]
```

layer_id

负载均衡器附加到的层的 ID (字符串)。

```
node["opsworks"]["stack"]["elb-load-balancers"]["i"]["layer_id"]
```

id

堆栈 ID (字符串)。

```
node["opsworks"]["stack"]["id"]
```

name

堆栈名称 (字符串)。

```
node["opsworks"]["stack"]["name"]
```

rds_instances

包含一个嵌入式对象列表，每个对象对应于注册到堆栈的一个 Amazon RDS 实例。每个嵌入式对象包含定义实例配置的一组属性。您在使用 Amazon RDS 控制台或 API 创建实例时指定这些值。在实例创建后，您还可使用 Amazon RDS 控制台或 API 编辑其中一些设置。有关更多信息，请参阅 Amazon RDS 文档。

> **Note**
>
> 这些属性的常规节点语法如下所示，其中 *i* 指定实例的从零开始的列表索引。
>
> ```
> node["opsworks"]["stack"]["rds_instances"]["i"]["attribute_name"]
> ```

如果您的堆栈具有多个 Amazon RDS 实例，那么下面是介绍如何在配方中使用特定实例的一个示例。

```
if my_rds = node["opsworks"]["stack"]["rds_instances"].select{|rds_instance|
 rds_instance["db_instance_identifier"] == 'db_id' }.first
  template "/etc/rds.conf" do
    source "rds.conf.erb"
    variables :address => my_rds["address"]
  end
end
```

address	allocated_storage	arn
auto_minor_version_upgrade	availability_zone	backup_retention_period
db_instance_class	db_instance_identifier	db_instance_status
db_name	db_parameter_groups	db_security_groups
db_user	engine	instance_create_time
license_model	multi_az	option_group_memberships
port	preferred_backup_window	preferred_maintenance_window
publicly_accessible	read_replica_db_instance_identifiers	region
status_infos	vpc_security_groups	

address

实例 URL，如 `opsinstance.ccdvt3hwog1a.us-west-2.rds.amazonaws.com` (字符串)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["address"]
```

allocated_storage

分配的存储空间，以 GB 为单位 (数字)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["allocated_storage"]
```

arn

实例的 ARN (字符串)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["arn"]
```

auto_minor_version_upgrade

是否自动应用次要版本升级 (布尔值)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["auto_minor_version_upgrade"]
```

availability_zone

实例的可用区，如 `us-west-2a` (字符串)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["availability_zone"]
```

backup_retention_period

备份保留期，以天为单位 (数字)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["backup_retention_period"]
```

db_instance_class

数据库实例类，如 `db.m1.small` (字符串)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["db_instance_class"]
```

db_instance_identifier

用户定义的数据库实例标识符 (字符串)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["db_instance_identifier"]
```

db_instance_status

实例的状态 (字符串)。有关更多信息，请参阅数据库实例。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["db_instance_status"]
```

db_name

用户定义的数据库名称 (字符串)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["db_name"]
```

db_parameter_groups

实例的数据库参数组，它包含一个嵌入式对象列表，每个对象对应于一个参数组。有关更多信息，请参阅使用数据库参数组。每个对象包含以下属性：

db_parameter_group_name

组名称 (字符串)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["db_parameter_groups"][j"]
["db_parameter_group_name"]
```

parameter_apply_status

应用状态 (字符串)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["db_parameter_groups"][j"]
["parameter_apply_status"]
```

db_security_groups

实例的数据库安全组，它包含一个嵌入式对象列表，每个对象对应于一个安全组。有关更多信息，请参阅使用数据库安全组。每个对象包含以下属性

db_security_group_name

安全组名称 (字符串)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["db_security_groups"][j"]
["db_security_group_name"]
```

status

状态 (字符串)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["db_security_groups"][j"]
["status"]
```

db_user

用户定义的主用户名 (字符串)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["db_user"]
```

engine

数据库引擎，如 `mysql(5.6.13)` (字符串)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["engine"]
```

instance_create_time

实例创建时间，如 `2014-04-15T16:13:34Z` (字符串)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["instance_create_time"]
```

license_model

实例的许可证模型，如 `general-public-license` (字符串)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["license_model"]
```

multi_az

是否启用了多可用区部署 (布尔值)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["multi_az"]
```

option_group_memberships

实例的选项组成员资格，它包含一个嵌入式对象列表，每个对象对应于一个选项组。有关更多信息，请参阅使用选项组。每个对象包含以下属性：

option_group_name

组的名称 (字符串)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["option_group_memberships"]
[j"]["option_group_name"]
```

status

组的状态 (字符串)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["option_group_memberships"]
[j"]["status"]
```

port

数据库服务器的端口 (数字)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["port"]
```

preferred_backup_window

首选每日备份时段，如 `06:26-06:56` (字符串)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["preferred_backup_window"]
```

preferred_maintenance_window

首选每周维护时段，如 `thu:07:13-thu:07:43` (字符串)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["preferred_maintenance_window"]
```

publicly_accessible

数据库是否可公开访问 (布尔值)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["publicly_accessible"]
```

read_replica_db_instance_identifiers

只读副本实例标识符的列表 (字符串列表)。有关更多信息，请参阅使用只读副本。

```
node["opsworks"]["stack"]["rds_instances"]["i"]
["read_replica_db_instance_identifiers"]
```

region

Amazon Web Services Region，如 `us-west-2` (字符串)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["region"]
```

status_infos

状态信息的列表 (字符串列表)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["status_infos"]
```

vpc_security_groups

VPC 安全组的列表 (字符串列表)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["vpc_security_groups"]
```

vpc_id

VPC id (字符串)。如果实例不在 VPC 中，则此值为 `null`。

```
node["opsworks"]["stack"]["vpc_id"]
```

其他顶级 opsworks 属性

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

本节包含没有子属性的 `opsworks` 属性。

活动

与属性关联的活动，如 `deploy` (字符串)。

```
node["opsworks"]["activity"]
```

agent_version

实例 OpsWorks 代理的版本（字符串）。

```
node["opsworks"]["agent_version"]
```

deploy_chef_provider

Chef 部署提供程序，其影响已部署应用程序的目录结构 (字符串)。您可以将此属性设置为以下之一：

- `Branch`
- `Revision`
- `Timestamped` (默认值)

```
node["opsworks"]["deploy_chef_provider"]
```

ruby_stack

Ruby 堆栈 (字符串)。默认设置为企业版 (`ruby_enterprise`)。对于 MRI 版本，请将此属性设置为 `ruby`。

```
node["opsworks"]["ruby_stack"]
```

ruby_version

应用程序将使用的 Ruby 版本 (字符串)。您可使用此属性仅指定主要和次要版本。您必须使用适当的 [“ruby”] 属性来指定修补程序版本。有关如何指定版本的更多信息 (包括示例)，请参阅 Ruby 版本。有关 AWS OpsWorks Stacks 如何确定 Ruby 版本的完整详细信息，请参阅内置属性文件 ruby.rb。

```
node["opsworks"]["ruby_version"]
```

run_cookbook_tests

是否对你的 Chef 11.4 食谱minitest-chef-handler进行测试（布尔值）。

```
node["opsworks"]["run_cookbook_tests"]
```

sent_at

此命令发送到实例的时间 (数字)。

```
node["opsworks"]["sent_at"]
```

部署

如果这些属性与部署活动关联，deployment 将设置为部署 ID，即由 AWS OpsWorks Stacks 生成的唯一地标识部署的 GUID（字符串）。否则，该属性设置为 null。

```
node["opsworks"]["deployment"]
```

opsworks_custom_cookbooks 属性

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

包含指定堆栈的自定义说明书的属性。

已启用

是否启用自定义说明书 (布尔值)。

```
node["opsworks_custom_cookbooks"]["enabled"]
```

recipes

要为此命令执行的配方的列表 (包括自定义配方)，使用 *cookbookname*::*recipename* 格式 (字符串列表)。

```
node["opsworks_custom_cookbooks"]["recipes"]
```

dependencies 属性

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

包含几个与 update_dependencies 堆栈命令相关的属性。

gem_binary

Gems 二进制文件的位置 (字符串)。

upgrade_debs

是否升级 Debs 程序包 (布尔值)。

update_debs

是否更新 Debs 程序包 (布尔值)。

ganglia 属性

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

包含一个 web 属性，该属性包含几个指定如何访问 Ganglia 统计信息网页的属性：

password

访问统计信息页所需的密码 (字符串)。

```
node["ganglia"]["web"]["password"]
```

url

统计信息页的 URL 路径，如 "/ganglia" (字符串)。完整的 URL 为 http://*DNSNameURLPath*，其中 *DNSName* 是关联实例的 DNS 名称。

```
node["ganglia"]["web"]["url"]
```

用户

访问统计信息页所需的用户名 (字符串)。

```
node["ganglia"]["web"]["user"]
```

mysql 属性

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

包含一组指定 MySQL 数据库服务器配置的属性。

客户端

客户端 IP 地址的列表 (字符串列表)。

```
node["mysql"]["clients"]
```

server_root_password

根密码 (字符串)。

```
node["mysql"]["server_root_password"]
```

passenger 属性

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

包含一组指定 Phusion Passenger 配置的属性。

gem_bin

RubyGems 二进制文件的位置，例如"/usr/local/bin/gem"（字符串）。

```
node["passenger"]["gem_bin"]
```

max_pool_size

最大池大小 (数字)。

```
node["passenger"]["max_pool_size"]
```

ruby_bin

Ruby 二进制文件的位置，例如 "/usr/local/bin/ruby"。

```
node["passenger"]["ruby_bin"]
```

版本

Passenger 版本 (字符串)。

```
node["passenger"]["version"]
```

opsworks_bundler 属性

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

包含指定捆绑程序支持的元素。

manage_package

是否安装和管理捆绑程序 (布尔值)。

```
node["opsworks_bundler"]["manage_package"]
```

版本

捆绑程序版本 (字符串)。

```
node["opsworks_bundler"]["version"]
```

deploy 属性

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

如果此类属性与部署事件或“执行配方”堆栈命令关联，则 deploy 属性包含已部署的每个应用程序的属性 (由应用程序的短名称命名)。每个应用程序属性包含以下属性：

应用程序	application_type	auto_bundle_on_deploy
database	deploy_to	域
document_root	environment_variables	组
keep_releases	memcached	migrate
mounted_at	purge_before_symlink	rails_env
restart_command	scm	ssl_certificate
ssl_certificate_ca	ssl_certificate_key	ssl_support
堆栈	symlink_before_migrate	symlinks
用户		

应用程序

应用程序的缩略名，如 "simplephp" (字符串)。

```
node["deploy"]["appshortname"]["application"]
```

application_type

应用程序类型 (字符串)。可能值如下所示：

- `java`：Java 应用程序
- `nodejs`：Node.js 应用程序
- `php`：PHP 应用程序
- `rails`：Ruby on Rails 应用程序
- `web`：静态 HTML 页面
- `other`：所有其他应用程序类型

```
node["deploy"]["appshortname"]["application_type"]
```

auto_bundle_on_deploy

对于 Rails 应用程序，是否在部署期间执行捆绑程序 (布尔值)。

```
node["deploy"]["appshortname"]["auto_bundle_on_deploy"]
```

database

包含连接应用程序数据库所需的信息。如果应用程序附加了数据库层， AWS OpsWorks Stacks 会自动为这些属性分配适当的值。

adapter

数据库适配器，如 `mysql` (字符串)。

```
node["deploy"]["appshortname"]["database"]["adapter"]
```

database

数据库名称，这通常是应用程序的缩略名，如 `"simplephp"` (字符串)。

```
node["deploy"]["appshortname"]["database"]["database"]
```

data_source_provider

数据源：mysql 或 rds (字符串)。

```
node["deploy"]["appshortname"]["database"]["data_source_provider"]
```

host

数据库主机的 IP 地址 (字符串)。

```
node["deploy"]["appshortname"]["database"]["host"]
```

password

数据库密码 (字符串)。

```
node["deploy"]["appshortname"]["database"]["password"]
```

port

数据库端口 (数字)。

```
node["deploy"]["appshortname"]["database"]["port"]
```

reconnect

对于 Rails 应用程序，确定在连接不再存在时是否应重新连接应用程序 (布尔值)。

```
node["deploy"]["appshortname"]["database"]["reconnect"]
```

username

用户名称 (字符串)。

```
node["deploy"]["appshortname"]["database"]["username"]
```

deploy_to

应用程序要部署到的位置，例如 "/srv/www/simplephp" (字符串)。

```
node["deploy"]["appshortname"]["deploy_to"]
```

域

应用程序的域的列表 (字符串列表)。

```
node["deploy"]["appshortname"]["domains"]
```

document_root

文档根 (如果您指定非默认根) 或 null (如果您使用默认根) (字符串)。

```
node["deploy"]["appshortname"]["document_root"]
```

environment_variables

由最多 20 个属性组成的集合，这些属性表示已为应用程序定义的用户指定的应用程序环境变量。有关如何定义应用程序的环境变量的更多信息，请参阅添加应用程序。每个属性名称都设置为一个环境变量名称，并且其对应值设置为变量的值，因此您可使用以下语法引用特定值。

```
node["deploy"]["appshortname"]["environment_variables"]["variable_name"]
```

组

应用程序的组 (字符串)。

```
node["deploy"]["appshortname"]["group"]
```

keep_releases

AWS OpsWorks Stacks 将存储的应用程序部署数量（数字）。此属性控制您可回滚应用程序的次数。默认情况下，它设置为全局值 deploy_keep_releases (其默认值为 5)。您可覆盖 keep_releases 以指定特定应用程序的已存储部署的数量。

```
node["deploy"]["appshortname"]["keep_releases"]
```

memcached

包含定义 memcached 配置的两个属性。

host

Memcached 服务器实例的 IP 地址 (字符串)。

```
node["deploy"]["appshortname"]["memcached"]["host"]
```

port

memcached 服务器正在侦听的端口 (数字)。

```
node["deploy"]["appshortname"]["memcached"]["port"]
```

migrate

对于 Rails 应用程序，确定是否运行迁移 (布尔值)。

```
node["deploy"]["appshortname"]["migrate"]
```

mounted_at

应用程序的安装点 (如果您指定非默认安装点) 或 null (如果您使用默认安装点) (字符串)。

```
node["deploy"]["appshortname"]["mounted_at"]
```

purge_before_symlink

对于 Rails 应用程序，在创建符号链接之前要清除的一组路径 (字符串列表)。

```
node["deploy"]["appshortname"]["purge_before_symlink"]
```

rails_env

对于 Rails App Server 实例，为 rails 环境，如 "production" (字符串)。

```
node["deploy"]["appshortname"]["rails_env"]
```

restart_command

要在应用程序重新启动时运行的命令，如 "echo 'restarting app'"。

```
node["deploy"]["appshortname"]["restart_command"]
```

scm

包含一组属性，用于指定用于从其源代码控制存储库部署应用程序的信息。 OpsWorks这些属性因存储库类型而异。

password

密码 (对于私有存储库) 和 null (对于公共存储库) (字符串)。对于私有 Amazon S3 存储桶，该属性设置为私有密钥。

```
node["deploy"]["appshortname"]["scm"]["password"]
```

存储库

存储库 URL，如 "git://github.com/amazonwebservices/opsworks-demo-php-simple-app.git" (字符串)。

```
node["deploy"]["appshortname"]["scm"]["repository"]
```

revision

如果存储库具有多个分支，该属性将指定应用程序的分支或版本，如 "version1" (字符串)。否则，其设置为 null。

```
node["deploy"]["appshortname"]["scm"]["revision"]
```

scm_type

存储库类型 (字符串)。可能值如下所示：

- "git"：Git 存储库
- "svn"：Subversion 存储库
- "s3"：一个 Amazon S3 存储桶
- "archive"：HTTP 存档
- "other"：另一个存储库类型

```
node["deploy"]["appshortname"]["scm"]["scm_type"]
```

ssh_key

部署 SSH 密钥 (对于访问私有 Git 存储库) 和 null (对于公共存储库) (字符串)。

```
node["deploy"]["appshortname"]["scm"]["ssh_key"]
```

用户

用户名 (对于私有存储库) 和 null (对于公共存储库) (字符串)。对于私有 Amazon S3 存储桶，该属性设置为访问密钥。

```
node["deploy"]["appshortname"]["scm"]["user"]
```

ssl_certificate

应用程序的 SSL 证书 (如果您启用了 SSL 支持) 或 null (如果您未启用 SSL 支持) (字符串)。

```
node["deploy"]["appshortname"]["ssl_certificate"]
```

ssl_certificate_ca

如果启用 SSL，则为指定中间证书颁发机构密钥或客户端身份验证的属性 (字符串)。

```
node["deploy"]["appshortname"]["ssl_certificate_ca"]
```

ssl_certificate_key

应用程序的 SSL 私有密钥 (如果您启用了 SSL 支持) 或 null (如果您未启用 SSL 支持) (字符串)。

```
node["deploy"]["appshortname"]["ssl_certificate_key"]
```

ssl_support

是否支持 SSL (布尔值)。

```
node["deploy"]["appshortname"]["ssl_support"]
```

堆栈

包含一个布尔值属性 `needs_reload`，其指定是否在部署期间重新加载应用程序服务器。

```
node["deploy"]["appshortname"]["stack"]["needs_reload"]
```

symlink_before_migrate

对于 Rails 应用程序，包含要在运行迁移之前作为 "*link*":"*target*" 对创建的符号链接。

```
node["deploy"]["appshortname"]["symlink_before_migrate"]
```

symlinks

包含部署的作为 "*link*":"*target*" 对的符号链接。

```
node["deploy"]["appshortname"]["symlinks"]
```

用户

应用程序的用户 (字符串)。

```
node["deploy"]["appshortname"]["user"]
```

其他顶级属性

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

本节包含没有子属性的顶级堆栈配置属性。

rails 属性

包含一个指定服务器的最大池大小的 max_pool_size 属性 (数字)。属性值由 AWS OpsWorks Stacks 设置并取决于实例类型，但您可以使用自定义 JSON 或自定义属性文件来覆盖该值。

```
node["rails"]["max_pool_size"]
```

recipes 属性

由此活动运行的内置配方的列表，使用 "*cookbookname*::*recipename*" 格式 (字符串列表)。

```
node["recipes"]
```

opsworks_rubygems 属性

包含指定版本的版本元素（字符串）。 RubyGems

```
node["opsworks_rubygems"]["version"]
```

languages 属性

包含每种已安装语言的属性，该属性根据语言进行命名，如 ruby。该属性是包含一个属性 (如 ruby_bin) 的对象，用于指定安装文件夹，如 "/usr/bin/ruby" (字符串)。

ssh_users 属性

包含一组属性，每个属性描述已获得 SSH 权限的用户之一。每个属性都以用户的 Unix ID 命名。AWS OpsWorks Stacks 会为 2000-4000 范围内的每个用户生成一个唯一 ID（例如）"2001"，并在每个实例上创建一个具有该 ID 的用户。由于 AWS OpsWorks 保留了 2000-4000 的范围，因此您在之外创建的用户 AWS OpsWorks（例如，通过使用食谱食谱或 AWS OpsWorks 从 IAM 导入用户）可以拥有被 AWS OpsWorks 其他用户的 Stacks 覆盖的 UID。最佳做法是，在 AWS OpsWorks Stacks 控制台中创建用户并管理他们的访问权限。如果您确实在 AWS OpsWorks 堆栈之外创建用户，请使用大于 4000 的 *UnixID* 值。

每个属性包含以下属性：

email

用户的电子邮件地址 (字符串)。

```
node["ssh_users"]["UnixID"]["email"]
```

public_key

用户的公有 SSH 密钥 (字符串)。

```
node["ssh_users"]["UnixID"]["public_key"]
```

sudoer

用户是否具有 sudo 权限 (布尔值)。

```
node["ssh_users"]["UnixID"]["sudoer"]
```

name

用户名称 (字符串)。

```
node["ssh_users"]["UnixID"]["name"]
```

内置说明书属性

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 大多数这些属性仅在 Linux 堆栈上可用。

大多数内置配方有一个或多个属性文件，用于定义各种设置。您可以在自定义配方中访问这些设置，并使用自定义 JSON 覆盖它们。您通常需要访问或覆盖用于控制 AWS OpsWorks Stacks 支持的各种服务器技术配置的属性。本部分汇总了这些属性。可在 https://github.com/aws/opsworks-cookbooks.git 上获得完整的属性文件以及关联的配方和模板。

> **Note**
>
> 所有内置配方属性均为 `default` 类型。

主题

- apache2 属性
- deploy 属性
- haproxy 属性
- memcached 属性
- mysql 属性
- nginx 属性
- opsworks_berkshelf 属性
- opsworks_java 属性
- passenger_apache2 属性
- ruby 属性
- unicorn 属性

apache2 属性

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 这些属性仅在 Linux 堆栈上可用。

apache2 属性指定 Apache HTTP 服务器配置。有关更多信息，请参阅 Apache 核心功能。有关如何覆盖内置属性以指定自定义值的更多信息，请参阅 覆盖属性。

binary	contact	deflate_types
dir	document_root	组
hide_info_headers	icondir	init_script

keepalive	keepaliverequests	keepalivetimeout
lib_dir	libexecdir	listen_ports
log_dir	logrotate 属性	pid_file
prefork 属性	serversignature	servertokens
timeout	traceenable	用户
版本	worker 属性	

binary

Apache 二进制文件的位置 (字符串)。默认值为 '/usr/sbin/httpd'。

```
node[:apache][:binary]
```

contact

电子邮件联系人 (字符串)。默认值为虚拟地址 'ops@example.com'。

```
node[:apache][:contact]
```

deflate_types

指示 mod_deflate 对指定 Mime 类型启用压缩，前提是这些类型受浏览器支持 (字符串列表)。默认值如下所示：

```
['application/javascript',
 'application/json',
 'application/x-javascript',
 'application/xhtml+xml',
 'application/xml',
 'application/xml+rss',
 'text/css',
 'text/html',
 'text/javascript',
 'text/plain',
```

```
'text/xml']
```

> **Warning**
>
> 压缩可能会带来安全风险。要完全禁用压缩，请按如下方式设置此属性：
>
> ```
> node[:apache][:deflate_types] = []
> ```

```
node[:apache][:deflate_types]
```

dir

服务器的根目录 (字符串)。默认值如下所示：

- Amazon Linux 和 Red Hat Enterprise Linux (RHEL)：`'/etc/httpd'`
- Ubuntu: `'/etc/apache2'`

```
node[:apache][:dir]
```

document_root

文档根目录 (字符串)。默认值如下所示：

- Amazon Linux 和 RHEL: `'/var/www/html'`
- Ubuntu: `'/var/www'`

```
node[:apache][:document_root]
```

组

组名称 (字符串)。默认值如下所示：

- Amazon Linux 和 RHEL: `'apache'`
- Ubuntu: `'www-data'`

```
node[:apache][:group]
```

hide_info_headers

是否忽略 HTTP 标头中的版本和模块信息 (`'true'`/`'false'`) (字符串)。默认值为 `'true'`。

```
node[:apache][:hide_info_headers]
```

icondir

图标目录 (字符串)。默认值如下所示：

- Amazon Linux 和 RHEL: `'/var/www/icons/'`
- Ubuntu: `'/usr/share/apache2/icons'`

```
node[:apache][:icondir]
```

init_script

初始化脚本 (字符串)。默认值如下所示：

- Amazon Linux 和 RHEL: `'/etc/init.d/httpd'`
- Ubuntu: `'/etc/init.d/apache2'`

```
node[:apache][:init_script]
```

keepalive

是否启用保持活动连接 (字符串)。可能的值为 `'On'` 和 `'Off'` (字符串)。默认值为 `'Off'`。

```
node[:apache][:keepalive]
```

keepaliverequests

Apache 将同时处理的最大保持活动请求数 (数字)。默认值为 `100`。

```
node[:apache][:keepaliverequests]
```

keepalivetimeout

Apache 在关闭连接前等待请求的时间 (数字)。默认值为 `3`。

```
node[:apache][:keepalivetimeout]
```

lib_dir

包含对象代码库的目录 (字符串)。默认值如下所示：

- Amazon Linux (x86)：'/usr/lib/httpd'
- Amazon Linux (x64) 和 RHEL：'/usr/lib64/httpd'
- Ubuntu: '/usr/lib/apache2'

```
node[:apache][:lib_dir]
```

libexecdir

包含程序可执行文件的目录 (字符串)。默认值如下所示：

- Amazon Linux (x86)：'/usr/lib/httpd/modules'
- Amazon Linux (x64) 和 RHEL：'/usr/lib64/httpd/modules'
- Ubuntu: '/usr/lib/apache2/modules'

```
node[:apache][:libexecdir]
```

listen_ports

服务器侦听的端口的列表 (字符串列表)。默认值为 ['80','443']。

```
node[:apache][:listen_ports]
```

log_dir

日志目录 (字符串)。默认值如下所示：

- Amazon Linux 和 RHEL: '/var/log/httpd'
- Ubuntu: '/var/log/apache2'

```
node[:apache][:log_dir]
```

logrotate 属性

这些属性指定如何轮换日志文件。

delaycompress

是否延迟压缩关闭的日志文件，直到下一个轮换周期开始 ('true'/'false') (字符串)。默认值为 'true'。

```
node[:apache][:logrotate][:delaycompress]
```

组

日志文件所在的组 (字符串)。默认值为 'adm'。

```
node[:apache][:logrotate][:group]
```

mode

日志文件的模式 (字符串)。默认值为 '640'。

```
node[:apache][:logrotate][:mode]
```

owner

日志文件的所有者 (字符串)。默认值为 'root'。

```
node[:apache][:logrotate][:owner]
```

rotate

删除关闭的日志文件之前的轮换周期数 (字符串)。默认值为 '30'。

```
node[:apache][:logrotate][:rotate]
```

schedule

轮换计划 (字符串)。可能值如下所示：

- 'daily'
- 'weekly'
- 'monthly'

默认值为 'daily'。

```
node[:apache][:logrotate][:schedule]
```

pid_file

包含守护程序进程 ID 的文件 (字符串)。默认值如下所示：

- Amazon Linux 和 RHEL: `'/var/run/httpd/httpd.pid'`
- Ubuntu: `'/var/run/apache2.pid'`

```
node[:apache][:pid_file]
```

prefork 属性

这些属性指定预派生配置。

maxclients

将提供的最大并行请求数 (数字)。默认值为 `400`。

> **Note**
>
> 此属性仅适用于运行 Amazon Linux 或 RHEL 的实例。如果您的实例运行的是 Ubuntu 14.04 LTS，请使用 maxrequestworkers。

```
node[:apache][:prefork][:maxclients]
```

maxrequestsperchild

子服务器进程处理的最大请求数 (数字)。默认值为 `10000`。

```
node[:apache][:prefork][:maxrequestsperchild]
```

maxrequestworkers

将提供的最大并行请求数 (数字)。默认值为 `400`。

> **Note**
>
> 此属性仅适用于运行 Ubuntu 14.04 LTS 的实例。如果您的实例运行的是 Amazon Linux 或 RHEL，请使用 maxclients 。

```
node[:apache][:prefork][:maxrequestworkers]
```

maxspareservers

最大空闲子服务器进程数 (数字)。默认值为 32。

```
node[:apache][:prefork][:maxspareservers]
```

minspareservers

最小空闲子服务器进程数 (数字)。默认值为 16。

```
node[:apache][:prefork][:minspareservers]
```

serverlimit

可配置的最大进程数 (数字)。默认值为 400。

```
node[:apache][:prefork][:serverlimit]
```

startservers

在启动时创建的子服务器进程数 (数字)。默认值为 16。

```
node[:apache][:prefork][:startservers]
```

serversignature

指定是否为服务器生成的文档配置尾部页脚以及如何配置 (字符串)。可能的值为 'On'、'Off' 和 'Email'。默认值为 'Off'。

```
node[:apache][:serversignature]
```

servertokens

指定响应标头应包含哪种类型的服务器版本信息 (字符串)：

- 'Full'：完整信息。例如，服务器：Apache/2.4.2 (Unix) PHP/4.2.2 /1.2 MyMod
- 'Prod'：产品名称。例如，Server: Apache

- `'Major'`：主要版本。例如，Server: Apache/2
- `'Minor'`：主要版本和次要版本。例如，Server: Apache/2.4
- `'Min'`：最低版本。例如，Server: Apache/2.4.2
- `'OS'`：带操作系统的版本。例如，Server: Apache/2.4.2 (Unix)

默认值为 `'Prod'`。

```
node[:apache][:servertokens]
```

timeout

Apache 等待 I/O 的时长 (数字)。默认值为 `120`。

```
node[:apache][:timeout]
```

traceenable

是否启用 TRACE 请求 (字符串)。可能的值为 `'On'` 和 `'Off'`。默认值为 `'Off'`。

```
node[:apache][:traceenable]
```

用户

用户名称 (字符串)。默认值如下所示：

- Amazon Linux 和 RHEL: `'apache'`
- Ubuntu: `'www-data'`

```
node[:apache][:user]
```

版本

Apache 版本 (字符串)。默认值如下所示：

- Amazon Linux: `2.2`
- Ubuntu 14.04 LTS: `2.4`
- RHEL: `2.4`

```
node[:apache][:version]
```

worker 属性

这些属性指定工作进程配置。

startservers

在启动时创建的子服务器进程数 (数字)。默认值为 4。

```
node[:apache][:worker][:startservers]
```

maxclients

将提供的最大并行请求数 (数字)。默认值为 1024。

```
node[:apache][:worker][:maxclients]
```

maxsparethreads

最大空闲线程数 (数字)。默认值为 192。

```
node[:apache][:worker][:maxsparethreads]
```

minsparethreads

最小空闲线程数 (数字)。默认值为 64。

```
node[:apache][:worker][:minsparethreads]
```

threadsperchild

每个子进程的线程数 (数字)。默认值为 64。

```
node[:apache][:worker][:threadsperchild]
```

maxrequestsperchild

子服务器进程处理的最大请求数 (数字)。默认值为 10000。

```
node[:apache][:worker][:maxrequestsperchild]
```

deploy 属性

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

内置部署说明书的 `deploy.rb` 属性文件在 `opsworks` 命名空间中定义了以下属性。有关部署目录的更多信息，请参阅Deploy 配方。有关如何覆盖内置属性以指定自定义值的更多信息，请参阅 覆盖属性。

deploy_keep_releases

AWS OpsWorks Stacks 将存储的应用程序部署数量的全局设置（数字）。默认值是 5。该值可控制您能够回滚应用程序的次数。

```
node[:opsworks][:deploy_keep_releases]
```

组

(仅限 Linux) 应用程序部署目录的 `group` 设置 (字符串)。该默认值取决于实例的操作系统：

- 对于 Ubuntu 实例，默认值为 `www-data`。
- 对于属于使用 Nginx 和 Unicorn 的 Rails App Server 层的成员 Amazon Linux 或 RHEL 实例，默认值为 `nginx`。
- 对于所有其他 Amazon Linux 或 RHEL 实例，默认值为 `apache`。

```
node[:opsworks][:deploy_user][:group]
```

用户

(仅限 Linux) 应用程序部署目录的 `user` 设置 (字符串)。默认值为 `deploy`。

```
node[:opsworks][:deploy_user][:user]
```

haproxy 属性

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 这些属性仅在 Linux 堆栈上可用。

haproxy 属性指定 HAProxy 服务器配置。有关更多信息，请参阅 HAProxy 文档。有关如何覆盖内置属性以指定自定义值的更多信息，请参阅 覆盖属性。

balance	check_interval	client_timeout
connect_timeout	default_max_connections	global_max_connections
health_check_method	health_check_url	queue_timeout
http_request_timeout	maxcon_factor_nodejs_app	maxcon_factor_nodejs_app_ssl
maxcon_factor_php_app	maxcon_factor_php_app_ssl	maxcon_factor_rails_app
maxcon_factor_rails_app_ssl	maxcon_factor_static	maxcon_factor_static_ssl
重试	server_timeout	stats_url
stats_user		

balance

负载均衡器用来选择服务器的算法 (字符串)。默认值为 `'roundrobin'`。其他选项是：

- 'static-rr'

- 'leastconn'
- 'source'
- 'uri'
- 'url_param'
- 'hdr(name)'
- 'rdp-cookie'
- 'rdp-cookie(name)'

有关这些参数的更多信息，请参阅 balance。

```
node[:haproxy][:balance]
```

check_interval

运行状况检查时间间隔 (字符串)。默认值为 '10s'。

```
node[:haproxy][:check_interval]
```

client_timeout

客户端可以处于非活动状态的最长时间 (字符串)。默认值为 '60s'。

```
node[:haproxy][:client_timeout]
```

connect_timeout

HAProxy 等待服务器连接尝试成功的最长时间 (字符串)。默认值为 '10s'。

```
node[:haproxy][:connect_timeout]
```

default_max_connections

默认的最大连接数 (字符串)。默认值为 '80000'。

```
node[:haproxy][:default_max_connections]
```

global_max_connections

最大连接数 (字符串)。默认值为 '80000'。

```
node[:haproxy][:global_max_connections]
```

health_check_method

运行状况检查方法 (字符串)。默认值为 'OPTIONS'。

```
node[:haproxy][:health_check_method]
```

health_check_url

用于检查服务器运行状况的 URL 路径 (字符串)。默认值为 '/'。

```
node[:haproxy][:health_check_url ]
```

queue_timeout

等待免费连接的最长时间 (字符串)。默认值为 '120s'。

```
node[:haproxy][:queue_timeout]
```

http_request_timeout

HAProxy 等待完整 HTTP 请求的最长时间 (字符串)。默认值为 '30s'。

```
node[:haproxy][:http_request_timeout]
```

重试

服务器连接失败后的重试次数 (字符串)。默认值为 '3'。

```
node[:haproxy][:retries]
```

server_timeout

客户端可以处于非活动状态的最长时间 (字符串)。默认值为 '60s'。

```
node[:haproxy][:server_timeout]
```

stats_url

统计信息页面的 URL 路径 (字符串)。默认值为 '/haproxy?stats'。

```
node[:haproxy][:stats_url]
```

stats_user

统计信息页面用户名称 (字符串)。默认值为 'opsworks'。

```
node[:haproxy][:stats_user]
```

maxcon 属性表示负载因子乘数，可用于计算 HAProxy 允许后端具有的最大连接数。例如，假设您在一个backend值为 4 的小实例上有一个 Rails 应用程序服务器，这意味着 AWS OpsWorks Stacks 将为该实例配置四个 Rails 进程。如果您使用默认的 maxcon_factor_rails_app 值 7，HAProxy 将处理与 Rails 服务器的 28 (4* 7) 个连接。

maxcon_factor_nodejs_app

Node.js 应用程序服务器的 maxcon 因子 (数字)。默认值为 10。

```
node[:haproxy][:maxcon_factor_nodejs_app]
```

maxcon_factor_nodejs_app_ssl

使用 SSL 的 Node.js 应用程序服务器的 maxcon 因子 (数字)。默认值为 10。

```
node[:haproxy][:maxcon_factor_nodejs_app_ssl]
```

maxcon_factor_php_app

PHP 应用程序服务器的 maxcon 因子 (数字)。默认值为 10。

```
node[:haproxy][:maxcon_factor_php_app]
```

maxcon_factor_php_app_ssl

使用 SSL 的 PHP 应用程序服务器的 maxcon 因子 (数字)。默认值为 10。

```
node[:haproxy][:maxcon_factor_php_app_ssl]
```

maxcon_factor_rails_app

Rails 应用程序服务器的 maxcon 因子 (数字)。默认值为 7。

```
node[:haproxy][:maxcon_factor_rails_app]
```

maxcon_factor_rails_app_ssl

使用 SSL 的 Rails 应用程序服务器的 maxcon 因子 (数字)。默认值为 7。

```
node[:haproxy][:maxcon_factor_rails_app_ssl]
```

maxcon_factor_static

静态 Web 服务器的 maxcon 因子 (数字)。默认值为 15。

```
node[:haproxy][:maxcon_factor_static]
```

maxcon_factor_static_ssl

使用 SSL 的静态 Web 服务器的 maxcon 因子 (数字)。默认值为 15。

```
node[:haproxy][:maxcon_factor_static_ssl]
```

memcached 属性

Important

该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

Note

这些属性仅在 Linux 堆栈上可用。

memcached 属性指定 Memcached 服务器配置。有关如何覆盖内置属性以指定自定义值的更多信息，请参阅 覆盖属性。

memory	max_connections	pid_file

port	start_command	stop_command
用户		

memory

要使用的最大内存 (MB) (数字)。默认值为 512。

```
node[:memcached][:memory]
```

max_connections

最大连接数 (字符串)。默认值为 '4096'。

```
node[:memcached][:max_connections]
```

pid_file

包含守护程序进程 ID 的文件 (字符串)。默认值为 'var/run/memcached.pid'。

```
node[:memcached][:pid_file]
```

port

要侦听的端口 (数字)。默认值为 11211。

```
node[:memcached][:port]
```

start_command

开始命令 (字符串)。默认值为 '/etc/init.d/memcached start'。

```
node[:memcached][:start_command]
```

stop_command

停止命令 (字符串)。默认值为 '/etc/init.d/memcached stop'。

```
node[:memcached][:stop_command]
```

用户

用户 (字符串)。默认值为 'nobody'。

```
node[:memcached][:user]
```

mysql 属性

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 这些属性仅在 Linux 堆栈上可用。

mysql 属性指定 MySQL 主配置。有关更多信息，请参阅服务器系统变量。有关如何覆盖内置属性以指定自定义值的更多信息，请参阅 覆盖属性。

basedir	bind_address	客户端
conf_dir	confd_dir	datadir
grants_path	mysql_bin	mysqladmin_bin
pid_file	port	root_group
server_root_password	球座	tunable 属性

basedir

基目录 (字符串)。默认值为 '/usr'。

```
node[:mysql][:basedir]
```

bind_address

MySQL 侦听的地址 (字符串)。默认值为 '0.0.0.0'。

```
node[:mysql][:bind_address]
```

客户端

客户端列表 (字符串列表)。

```
node[:mysql][:clients]
```

conf_dir

包含配置文件的目录 (字符串)。默认值如下所示：

- Amazon Linux 和 RHEL: '/etc'
- Ubuntu: '/etc/mysql'

```
node[:mysql][:conf_dir]
```

confd_dir

包含额外配置文件的目录 (字符串)。默认值为 '/etc/mysql/conf.d'。

```
node[:mysql][:confd_dir]
```

datadir

数据目录 (字符串)。默认值为 '/var/lib/mysql'。

```
node[:mysql][:datadir]
```

grants_path

授权表位置 (字符串)。默认值为 '/etc/mysql_grants.sql'。

```
node[:mysql][:grants_path]
```

mysql_bin

mysql 二进制文件位置 (字符串)。默认值为 '/usr/bin/mysql'。

```
node[:mysql][:mysql_bin]
```

mysqladmin_bin

mysqladmin 位置 (字符串)。默认值为 '/usr/bin/mysqladmin'。

```
node[:mysql][:mysqladmin_bin]
```

pid_file

包含守护程序进程 ID 的文件 (字符串)。默认值为 '/var/run/mysqld/mysqld.pid'。

```
node[:mysql][:pid_file]
```

port

服务器侦听的端口 (数字)。默认值为 3306。

```
node[:mysql][:port]
```

root_group

根组 (字符串)。默认值为 'root'。

```
node[:mysql][:root_group]
```

server_root_password

服务器的根密码 (字符串)。默认值将随机生成。

```
node[:mysql][:server_root_password]
```

球座

套接字文件的位置 (字符串)。默认值为 '/var/lib/mysql/mysql.sock'。默认值如下所示：

- Amazon Linux 和 RHEL: '/var/lib/mysql/mysql.sock'

- Ubuntu: `'/var/run/mysqld/mysqld.sock'`

```
node[:mysql][:socket]
```

tunable 属性

tunable 属性用于优化性能。

back_log	innodb_additional_mem_pool_size	innodb_buffer_pool_size
innodb_flush_log_at_trx_commit	innodb_lock_wait_timeout	key_buffer
log_slow_queries	long_query_time	max_allowed_packet
max_connections	max_heap_table_size	net_read_timeout
net_write_timeout	query_cache_limit	query_cache_size
query_cache_type	thread_cache_size	thread_stack
wait_timeout	table_cache	

back_log

最大未完成请求数 (字符串)。默认值为 `'128'`。

```
node[:mysql][:tunable][:back_log]
```

innodb_additional_mem_pool_size

Innodb 用于存储内部数据结构的池的大小 (字符串)。默认值为 `'20M'`。

```
node[:mysql][:tunable][:innodb_additional_mem_pool_size]
```

innodb_buffer_pool_size

Innodb 缓冲池的大小 (字符串)。属性值由 AWS OpsWorks Stacks 设置并取决于实例类型，但您可以使用自定义 JSON 或自定义属性文件来覆盖该值。

```
node[:mysql][:tunable][:innodb_buffer_pool_size]
```

innodb_flush_log_at_trx_commit

Innodb 刷新日志缓冲区的频率 (字符串)。默认值为 '2'。有关更多信息，请参阅 innodb_flush_log_at_trx_commit。

```
node[:mysql][:tunable][:innodb_flush_log_at_trx_commit]
```

innodb_lock_wait_timeout

Innodb 事务等待行锁的最长时间 (秒) (字符串)。默认值为 '50'。

```
node[:mysql][:tunable][:innodb_lock_wait_timeout]
```

key_buffer

索引缓冲区大小 (字符串)。默认值为 '250M'。

```
node[:mysql][:tunable][:key_buffer]
```

log_slow_queries

慢速查询日志文件的位置 (字符串)。默认值为 '/var/log/mysql/mysql-slow.log'。

```
node[:mysql][:tunable][:log_slow_queries]
```

long_query_time

将查询指定为长时间运行查询所需的时间 (秒) (字符串)。默认值为 '1'。

```
node[:mysql][:tunable][:long_query_time]
```

max_allowed_packet

允许的最大数据包大小 (字符串)。默认值为 '32M'。

```
node[:mysql][:tunable][:max_allowed_packet]
```

max_connections

最大并发客户端连接数 (字符串)。默认值为 '2048'。

```
node[:mysql][:tunable][:max_connections]
```

max_heap_table_size

用户创建的 MEMORY 表的最大大小 (字符串)。默认值为 '32M'。

```
node[:mysql][:tunable][:max_heap_table_size]
```

net_read_timeout

等待来自连接的更多数据的时长 (秒) (字符串)。默认值为 '30'。

```
node[:mysql][:tunable][:net_read_timeout]
```

net_write_timeout

等待数据块写入连接的时长 (秒) (字符串)。默认值为 '30'。

```
node[:mysql][:tunable][:net_write_timeout]
```

query_cache_limit

单个缓存查询的最大大小 (字符串)。默认值为 '2M'。

```
node[:mysql][:tunable][:query_cache_limit]
```

query_cache_size

查询缓存大小 (字符串)。默认值为 '128M'。

```
node[:mysql][:tunable][:query_cache_size]
```

query_cache_type

查询缓存类型 (字符串)。可能的值如下所示：

- '0'：不缓存或检索缓存数据。

- '1'：缓存不以 SELECT SQL_NO_CACHE 开头的语句。
- '2'：缓存以 SELECT SQL_CACHE 开头的语句。

默认值为 '1'。

```
node[:mysql][:tunable][:query_cache_type]
```

thread_cache_size

缓存以便重复使用的客户端线程数 (字符串)。默认值为 '8'。

```
node[:mysql][:tunable][:thread_cache_size]
```

thread_stack

每个线程的堆栈大小 (字符串)。默认值为 '192K'。

```
node[:mysql][:tunable][:thread_stack]
```

wait_timeout

等待非交互式连接的时长 (秒)。默认值为 '180' (字符串)。

```
node[:mysql][:tunable][:wait_timeout]
```

table_cache

打开的表的数量 (字符串)。默认值为 '2048'。

```
node[:mysql][:tunable][:table_cache]
```

nginx 属性

Important

该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 这些属性仅在 Linux 堆栈上可用。

nginx 属性指定 Nginx 配置。有关更多信息，请参阅指令索引。有关如何覆盖内置属性以指定自定义值的更多信息，请参阅 覆盖属性。

binary	dir	gzip
gzip_comp_level	gzip_disable	gzip_http_version
gzip_proxied	gzip_static	gzip_types
gzip_vary	keepalive	keepalive_timeout
log_dir	用户	server_names_hash_bucket_size
worker_processes	worker_connections	

binary

Nginx 二进制文件的位置 (字符串)。默认值为 `'/usr/sbin/nginx'`。

```
node[:nginx][:binary]
```

dir

诸如配置文件之类的文件的位置 (字符串)。默认值为 `'/etc/nginx'`。

```
node[:nginx][:dir]
```

gzip

是否启用 gzip 压缩 (字符串)。可能的值为 `'on'` 和 `'off'`。默认值为 `'on'`。

> **Warning**
>
> 压缩可能会带来安全风险。要完全禁用压缩，请按如下方式设置此属性：

```
node[:nginx][:gzip] = 'off'
```

```
node[:nginx][:gzip]
```

gzip_comp_level

压缩级别，范围为 1 至 9，其中 1 对应于最低程度的压缩 (字符串)。默认值为 '2'。

```
node[:nginx][:gzip_comp_level]
```

gzip_disable

对指定的用户代理禁用 gzip 压缩 (字符串)。该值是一个正则表达式，默认值为 'MSIE [1-6].(?!.*SV1)'。

```
node[:nginx][:gzip_disable]
```

gzip_http_version

对指定的 HTTP 版本启用 gzip 压缩 (字符串)。默认值为 '1.0'。

```
node[:nginx][:gzip_http_version]
```

gzip_proxied

是否压缩代理请求的响应以及如何压缩，可以使用以下任一值 (字符串)：

- 'off'：不压缩代理请求
- 'expired'：如果 Expire 标头阻止缓存，则压缩
- 'no-cache'：如果 Cache-Control 标头设置为“no-cache”，则压缩
- 'no-store'：如果 Cache-Control 标头设置为“no-store”，则压缩
- 'private'：如果 Cache-Control 标头设置为“private”，则压缩
- 'no_last_modified'：如果未设置 Last-Modified，则压缩
- 'no_etag'：如果请求缺少 ETag 标头，则压缩
- 'auth'：如果请求包含 Authorization 标头，则压缩
- 'any'：压缩所有代理请求

默认值为 'any'。

```
node[:nginx][:gzip_proxied]
```

gzip_static

是否启用 gzip 静态模块 (字符串)。可能的值为 'on' 和 'off'。默认值为 'on'。

```
node[:nginx][:gzip_static]
```

gzip_types

要压缩的 MIME 类型的列表 (字符串列表)。默认值为 ['text/plain', 'text/html', 'text/css', 'application/x-javascript', 'text/xml', 'application/xml', 'application/xml+rss', 'text/javascript']。

```
node[:nginx][:gzip_types]
```

gzip_vary

是否启用 Vary:Accept-Encoding 响应标头 (字符串)。可能的值为 'on' 和 'off'。默认值为 'on'。

```
node[:nginx][:gzip_vary]
```

keepalive

是否启用保持活动连接 (字符串)。可能的值为 'on' 和 'off'。默认值为 'on'。

```
node[:nginx][:keepalive]
```

keepalive_timeout

保持活动的连接处于打开状态的最长时间 (秒) (数字)。默认值为 65。

```
node[:nginx][:keepalive_timeout]
```

log_dir

日志文件的位置 (字符串)。默认值为 '/var/log/nginx'。

```
node[:nginx][:log_dir]
```

用户

用户 (字符串)。默认值如下所示：

- Amazon Linux 和 RHEL: `'www-data'`
- Ubuntu: `'nginx'`

```
node[:nginx][:user]
```

server_names_hash_bucket_size

服务器名称哈希表的存储桶大小，该值可设置为 32、64 或 128 (数字)。默认值为 64。

```
node[:nginx][:server_names_hash_bucket_size]
```

worker_processes

工作进程数 (数字)。默认值为 10。

```
node[:nginx][:worker_processes]
```

worker_connections

最大工作线程连接数 (数字)。默认值为 1024。最大客户端数设置为 `worker_processes * worker_connections`。

```
node[:nginx][:worker_connections]
```

opsworks_berkshelf 属性

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 这些属性仅在 Linux 堆栈上可用。

opsworks_berkshelf 属性指定 Berkshelf 配置。有关更多信息，请参阅 Berkshelf。有关如何覆盖内置属性以指定自定义值的更多信息，请参阅 覆盖属性。

调试

是否要在 Chef 日志中包含 Berkshelf 调试信息 (布尔值)。默认值为 `false`。

```
node['opsworks_berkshelf]['debug']
```

opsworks_java 属性

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 这些属性仅在 Linux 堆栈上可用。

opsworks_java 属性指定 Tomcat 服务器配置。有关更多信息，请参阅 Apache Tomcat 配置参考。有关如何覆盖内置属性以指定自定义值的更多信息，请参阅 覆盖属性。

datasources	java_app_server_version	java_shared_lib_dir
jvm_pkg 属性	custom_pkg_location_url_debian	java_home_basedir
custom_pkg_location_url_rhel	use_custom_pkg_location	jvm_options

jvm_version	tomcat 属性	

datasources

用于定义 JNDI 资源名称的一组属性 (字符串)。有关如何使用此属性的更多信息，请参阅 部署 JSP 应用程序附带后端数据库。默认值为空的哈希，可以填充应用程序短名称与 JNDI 名称之间的自定义映射。有关更多信息，请参阅 部署 JSP 应用程序附带后端数据库。

```
node['opsworks_java']['datasources']
```

java_app_server_version

Java 应用程序服务器版本 (数字)。默认值为 7。您可覆盖此属性以指定版本 6。如果您安装了非默认 JDK，此属性将被忽略。

```
node['opsworks_java']['java_app_server_version']
```

java_shared_lib_dir

Java 共享库的目录 (字符串)。默认值为 `/usr/share/java`。

```
node['opsworks_java']['java_shared_lib_dir']
```

jvm_pkg 属性

一组属性，您可以覆盖该属性以安装非默认 JDK。

use_custom_pkg_location

是否要安装自定义 JDK，以替代 OpenJDK (布尔值)。默认值为 `false`。

```
node['opsworks_java']['jvm_pkg']['use_custom_pkg_location']
```

custom_pkg_location_url_debian

要在 Ubuntu 实例上安装的 JDK 软件包的位置 (字符串)。默认值为 `'http://aws.amazon.com/'`，这只是初始化值，没有适当的意义。如果您想要安装非默认 JDK，则必须覆盖此属性，并将其设置为适当的 URL。

```
node['opsworks_java']['jvm_pkg']['custom_pkg_location_url_debian']
```

custom_pkg_location_url_rhel

要在 Amazon Linux 和 RHEL 实例上安装的 JDK 软件包的位置 (字符串)。默认值为 'http://aws.amazon.com/'，这只是初始化值，没有适当的意义。如果您想要安装非默认 JDK，则必须覆盖此属性，并将其设置为适当的 URL。

```
node['opsworks_java']['jvm_pkg']['custom_pkg_location_url_rhel']
```

java_home_basedir

JDK 软件包会被提取到的目录 (字符串)。默认值为 /usr/local。您不需要为 RPM 软件包指定此设置；它们包括完整的目录结构。

```
node['opsworks_java']['jvm_pkg']['java_home_basedir']
```

jvm_options

JVM 命令行选项，这些选项允许您指定一些设置，例如堆大小 (字符串)。常见的选项集为 -Djava.awt.headless=true -Xmx128m -XX:+UseConcMarkSweepGC。默认值为无选项。

```
node['opsworks_java']['jvm_options']
```

jvm_version

OpenJDK 版本 (数字)。默认值为 7。您可覆盖此属性以指定 OpenJDK 版本 6。如果您安装了非默认 JDK，此属性将被忽略。

```
node['opsworks_java']['jvm_version']
```

tomcat 属性

一组属性，您可以覆盖此属性以安装默认的 Tomcat 配置。

ajp_port	apache_tomcat_bind_mod	apache_tomcat_bind_path
auto_deploy	connection_timeout	mysql_connector_jar

port	secure_port	shutdown_port
threadpool_max_threads	threadpool_min_spare_threads	unpack_wars
uri_encoding	use_ssl_connector	use_threadpool
userdatabase_pathname		

ajp_port

AJP 端口 (数字)。默认值为 8009。

```
node['opsworks_java']['tomcat]['ajp_port']
```

apache_tomcat_bind_mod

代理模块 (字符串)。默认值为 proxy_http。您可覆盖此属性以指定 AJP 代理模块 proxy_ajp。

```
node['opsworks_java']['tomcat]['apache_tomcat_bind_mod']
```

apache_tomcat_bind_path

Apache-Tomcat 绑定路径 (字符串)。默认值为 /。您不应覆盖此属性；更改绑定路径可能会导致应用程序停止工作。

```
node['opsworks_java']['tomcat]['apache_tomcat_bind_path']
```

auto_deploy

是否自动部署 (布尔值)。默认值为 true。

```
node['opsworks_java']['tomcat]['auto_deploy']
```

connection_timeout

连接超时 (毫秒) (数字)。默认值为 20000 (20 秒)。

```
node['opsworks_java']['tomcat]['connection_timeout']
```

mysql_connector_jar

MySQL 连接器库的 JAR 文件 (字符串)。默认值为 `mysql-connector-java.jar`。

```
node['opsworks_java']['tomcat]['mysql_connector_jar']
```

port

标准端口 (数字)。默认值为 `8080`。

```
node['opsworks_java']['tomcat]['port']
```

secure_port

安全端口 (数字)。默认值为 `8443`。

```
node['opsworks_java']['tomcat]['secure_port']
```

shutdown_port

关闭端口 (数字)。默认值为 `8005`。

```
node['opsworks_java']['tomcat]['shutdown_port']
```

threadpool_max_threads

线程池中的最大线程数 (数字)。默认值为 `150`。

```
node['opsworks_java']['tomcat]['threadpool_max_threads']
```

threadpool_min_spare_threads

线程池中的最少备用线程数 (数字)。默认值为 `4`。

```
node['opsworks_java']['tomcat]['threadpool_min_spare_threads']
```

unpack_wars

是否解压缩 WAR 文件 (布尔值)。默认值为 `true`。

```
node['opsworks_java']['tomcat]['unpack_wars']
```

uri_encoding

URI 编码 (字符串)。默认值为 `UTF-8`。

```
node['opsworks_java']['tomcat]['uri_encoding']
```

use_ssl_connector

是否使用 SSL 连接器 (布尔值)。默认值为 `false`。

```
node['opsworks_java']['tomcat]['use_ssl_connector']
```

use_threadpool

是否使用线程池 (布尔值)。默认值为 `false`。

```
node['opsworks_java']['tomcat]['use_threadpool']
```

userdatabase_pathname

用户数据库路径名称 (字符串)。默认值为 `conf/tomcat-users.xml`。

```
node['opsworks_java']['tomcat]['userdatabase_pathname']
```

passenger_apache2 属性

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 这些属性仅在 Linux 堆栈上可用。

`passenger_apache2` 属性指定 Phusion Passenger 配置。有关更多信息，请参阅 Phusion Passenger 用户指南，Apache 版本。有关如何覆盖内置属性以指定自定义值的更多信息，请参阅 覆盖属性。

friendly_error_pages	gem_bin	gems_path
high_performance_mode	root_path	max_instances_per_app
max_pool_size	max_requests	module_path
pool_idle_time	rails_app_spawner_idle_time	rails_framework_spawner_idle_time
rails_spawn_method	ruby_bin	ruby_wrapper_bin
stat_throttle_rate	版本	

friendly_error_pages

如果应用程序无法启动，是否显示友好的错误页面 (字符串)。此属性可设置为“on”或“off”；默认值为“off”。

```
node[:passenger][:friendly_error_pages]
```

gem_bin

Gem 二进制文件的位置 (字符串)。默认值为 `'/usr/local/bin/gem'`。

```
node[:passenger][:gem_bin]
```

gems_path

Gem 路径 (字符串)。默认值取决于 Ruby 版本。例如：

- Ruby 版本 1.8：`'/usr/local/lib/ruby/gems/1.8/gems'`

- Ruby 版本 1.9：`'/usr/local/lib/ruby/gems/1.9.1/gems'`

```
node[:passenger][:gems_path]
```

high_performance_mode

是否使用 Passenger 的高性能模式 (字符串)。可能的值为 `'on'` 和 `'off'`。默认值为 `'off'`。

```
node[:passenger][:high_performance_mode ]
```

root_path

Passenger 根目录 (字符串)。默认值取决于 Ruby 和 Passenger 版本。在 Chef 语法中，该值为 `"#{node[:passenger][:gems_path]}/passenger-#{passenger[:version]}"`。

```
node[:passenger][:root_path]
```

max_instances_per_app

每个应用程序的最大应用程序进程数 (数字)。默认值为 `0`。有关更多信息，请参阅PassengerMaxInstancesPerApp。

```
node[:passenger][:max_instances_per_app]
```

max_pool_size

最大应用程序处理器数 (数字)。默认值为 8。有关更多信息，请参阅PassengerMaxPoolSize。

```
node[:passenger][:max_pool_size]
```

max_requests

最大请求数 (数字)。默认值为 `0`。

```
node[:passenger][:max_requests]
```

module_path

模块路径 (字符串)。默认值如下所示：

- Amazon Linux 和 RHEL: `"#{node['apache']['libexecdir']}/mod_passenger.so"`

- Ubuntu: "#{passenger[:root_path]}/ext/apache2/mod_passenger.so"

```
node[:passenger][:module_path]
```

pool_idle_time

应用程序进程可以保持空闲状态的最长时间 (秒) (数字)。默认值为 14400 (4 小时)。有关更多信息，请参阅PassengerPoolIdleTime。

```
node[:passenger][:pool_idle_time]
```

rails_app_spawner_idle_time

Rails 应用程序生成器的最长空闲时间 (数字)。如果此属性设置为零，则应用程序生成器不会超时。默认值为 0。有关更多信息，请参阅生成方法说明。

```
node[:passenger][:rails_app_spawner_idle_time]
```

rails_framework_spawner_idle_time

Rails 框架生成器的最长空闲时间 (数字)。如果此属性设置为零，则框架生成器不会超时。默认值为 0。有关更多信息，请参阅生成方法说明。

```
node[:passenger][:rails_framework_spawner_idle_time]
```

rails_spawn_method

Rails 生成方法 (字符串)。默认值为 'smart-lv2'。有关更多信息，请参阅生成方法说明。

```
node[:passenger][:rails_spawn_method]
```

ruby_bin

Ruby 二进制文件的位置 (字符串)。默认值为 '/usr/local/bin/ruby'。

```
node[:passenger][:ruby_bin]
```

ruby_wrapper_bin

Ruby 包装程序脚本的位置 (字符串)。默认值为 '/usr/local/bin/ruby_gc_wrapper.sh'。

```
node[:passenger][:ruby_wrapper_bin]
```

stat_throttle_rate

Passenger 执行文件系统检查的速率 (数字)。默认值为 5，这意味着检查最多每 5 秒执行一次。有关更多信息，请参阅PassengerStatThrottleRate 。

```
node[:passenger][:stat_throttle_rate]
```

版本

版本 (字符串)。默认值为 '3.0.9'。

```
node[:passenger][:version]
```

ruby 属性

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 这些属性仅在 Linux 堆栈上可用。

ruby 属性指定应用程序使用的 Ruby 版本。请注意，在 Ruby 2.1 中引入语义版本控制会改变该属性的用法。有关如何指定版本的更多信息 (包括示例)，请参阅 Ruby 版本。有关 AWS OpsWorks Stacks 如何确定 Ruby 版本的完整详细信息，请参阅内置属性文件 r uby.rb。有关如何覆盖内置属性以指定自定义值的更多信息，请参阅 覆盖属性。

full_version

完整的版本号 (字符串)。您不应覆盖此属性，而应使用 [:opsworks][:ruby_version] 和相应的修补版本属性来指定版本。

```
[:ruby][:full_version]
```

major_version

主要版本号 (字符串)。您不应覆盖此属性，而应使用 [:opsworks][:ruby_version] 来指定主要版本。

```
[:ruby][:major_version]
```

minor_version

次要版本号 (字符串)。您不应覆盖此属性，而应使用 [:opsworks][:ruby_version] 来指定次要版本。

```
[:ruby][:minor_version]
```

patch

修补程序级别 (字符串)。此属性对 Ruby 版本 2.0.0 及更低版本有效。对于更高版本的 Ruby，请使用 `patch_version` 属性。

```
[:ruby][:patch]
```

修补程序版本号必须以 p 为前缀。例如，您可以使用以下自定义 JSON 来指定修补程序级别 484。

```
{
  "ruby":{"patch":"p484"}
}
```

patch_version

修补程序版本号 (字符串)。此属性对 Ruby 版本 2.1 及更高版本有效。对于更低版本的 Ruby，请使用 `patch` 属性。

```
[:ruby][:patch_version]
```

pkgrelease

软件包版本号 (字符串)。

```
[:ruby][:pkgrelease]
```

unicorn 属性

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 这些属性仅在 Linux 堆栈上可用。

`unicorn` 属性指定 Unicorn 配置。有关更多信息，请参阅 Unicorn::Configurator。有关如何覆盖内置属性以指定自定义值的更多信息，请参阅 覆盖属性。

accept_filter	backlog	delay
tcp_nodelay	tcp_nopush	preload_app
timeout	tries	版本
worker_processes		

accept_filter

接受筛选条件，`'httpready'` 或 `'dataready'` (字符串)。默认值为 `'httpready'`。

```
node[:unicorn][:accept_filter]
```

backlog

队列可容纳的最大请求数 (数字)。默认值为 `1024`。

```
node[:unicorn][:backlog]
```

delay

等待重新尝试绑定套接字的时长 (秒) (数字)。默认值为 `0.5`。

```
node[:unicorn][:delay]
```

preload_app

在派生工作进程之前，是否要预加载应用程序 (布尔值)。默认值为 `true`。

```
node[:unicorn][:preload_app]
```

tcp_nodelay

是否要对 TCP 套接字禁用 Nagle 算法 (布尔值)。默认值为 `true`。

```
node[:unicorn][:tcp_nodelay]
```

tcp_nopush

是否启用 TCP_CORK (布尔值)。默认值为 `false`。

```
node[:unicorn][:tcp_nopush]
```

timeout

允许工作线程用于每个请求的最长时间 (秒) (数字)。超出超时值的工作线程都将终止。默认值为 `60`。

```
node[:unicorn][:timeout]
```

tries

重新尝试绑定到套接字的最大次数 (数字)。默认值为 `5`。

```
node[:unicorn][:tries]
```

版本

Unicorn 版本 (字符串)。默认值为 `'4.7.0'`。

```
node[:unicorn][:version]
```

worker_processes

工作进程数 (数字)。如果存在默认值，则为 `max_pool_size`，否则为 `4`。

```
node[:unicorn][:worker_processes]
```

为 Linux 进行 Chef 11.10 和早期版本的故障排除

Important

该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

Note

有关其他故障排除信息，请参阅 调试和故障排除指南。

适用于 Linux 的 Chef 11.10 和早期版本的 Chef 日志

AWS OpsWorks 堆栈将每个实例的 Chef 日志存储在其`/var/lib/aws/opsworks/chef`目录中。您需要 sudo 权限来访问此目录。每个运行的日志位于名为 *YYYY*-*MM*-*DD*-*HH*-*MM*-*SS*-*NN*.log 的文件中。

有关更多信息，请参阅下列内容：

- 使用控制台查看 Chef 日志
- 使用 CLI 或 API 查看 Chef 日志
- 解释 Chef 日志
- 常见 Chef 日志错误

将 AWS OpsWorks 堆栈与其他 AWS 服务一起使用

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

您可以让在 Stac AWS OpsWorks ks 堆栈中运行的应用程序服务器使用各种 AWS 服务，这些服务并未直接与 AWS OpsWorks Stacks 集成。例如，您可以让您的应用程序服务器将 Amazon RDS 用作后端数据库。您可以通过使用以下一般模式来访问此类服务：

1. 通过使用 Amazon Web Services Console、API 或 CLI 创建和配置 Amazon Web Service，并记录应用程序访问服务将需要的任何所需配置数据，如主机名或端口。
2. 创建一个或多个自定义配方来配置应用程序，使其可以访问服务。

   该配方包含您在运行这些配方之前使用自定义 JSON 定义的堆栈配置和部署 JSON 属性中的配置数据。
3. 将自定义配方分配给应用程序服务器层上的 Deploy 生命周期事件。
4. 创建可为配置数据属性分配适当值的自定义 JSON 对象，并将该对象添加到您的堆栈配置和部署 JSON 中。
5. 将应用程序部署到堆栈。

   部署运行自定义配方，这些自定义配方使用您在自定义 JSON 中定义的配置数据值来配置应用程序，使其可以访问服务。

本节介绍如何让 AWS OpsWorks Stacks 应用程序服务器访问各种 AWS 服务。本文假设您已熟悉 Chef 说明书，并了解配方如何使用堆栈和配置 JSON 属性来配置应用程序 (通常是通过创建配置文件来实现)。如果您对上述内容并不熟悉，则您应当先阅读说明书和诀窍和自定义堆栈 AWS OpsWorks。

主题

使用后端数据存储

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

应用程序服务器堆栈通常包括用于提供后端数据存储的数据库服务器。 AWS OpsWorks Stacks 通过 MySQL 层为 MySQL 服务器提供集成支持，并通过亚马逊关系数据库服务 (Amazon RDS) 层为多种类型的数据库服务器提供集成支持。但是，您可以轻松自定义一个堆栈来让应用程序服务器使用其他数据库服务器，如 Amazon DynamoDB 或 MongoDB。本主题描述连接应用程序服务器与 AWS 数据库服务器的基本过程。这里使用来自 Chef 11 Linux 堆栈入门中的堆栈和应用程序展示如何手动将 PHP 应用程序服务器连接到 RDS 数据库。尽管这是一个基于 Linux 堆栈的示例，但基本原理同样适用于 Windows 堆栈。有关如何将 MongoDB 数据库服务器整合到堆栈中的示例，请参阅使用部署 MongoDB。 OpsWorks

> **Note**
>
> 为方便起见，本主题以 Amazon RDS 为例。但是，如果您希望将 Amazon RDS 数据库用于您的堆栈，则使用 Amazon RDS 层更为容易。

主题

如何设置数据库连接

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

可以使用自定义配方来设置应用程序服务器与其后端数据库之间的连接。该配方按要求配置应用程序服务器，通常是通过创建配置文件。该配方从 Stacks 在每个实例上安装的堆栈配置和部署属性中的一组属性中获取连接数据， AWS OpsWorks 例如主机和数据库名称。

例如，的Chef 11 Linux 堆栈入门第 2 步基于一个名为 MyStack PHP App Server 和 MySQL 的堆栈，每个层都有一个实例。您在 PHP 应用服务器实例上部署了一个名为 SimplePHPApp 的应用程序，该应用程序使用 MySQL 实例上的数据库作为后端数据存储。当您部署应用程序时， AWS OpsWorks Stacks 安装包含数据库连接信息的堆栈配置和部署属性。下例显示数据库连接属性，表示为 JSON：

```
{
  ...
  "deploy": {
    "simplephpapp": {
      ...
      "database": {
        "reconnect": true,
        "password": null,
        "username": "root",
        "host": null,
        "database": "simplephpapp"
        ...
      },
      ...
    }
  }
}
```

属性值由 AWS OpsWorks Stacks 提供，要么是生成的，要么基于用户提供的信息。

要允许 SimplePHPApp 访问数据存储，必须将名为 `appsetup.rb` 的自定义配方分配给 PHP 应用程序服务器层的部署生命周期事件，以此来设置 PHP 应用程序服务器与 MySQL 数据库之间的连接。当你部署 SimplephPapp 时， AWS OpsWorks Stacks 会运行`appsetup.rb`，它会创建一个名为`db-connect.php`的配置文件来设置连接，如以下摘录所示。

```
node[:deploy].each do |app_name, deploy|
  ...
  template "#{deploy[:deploy_to]}/current/db-connect.php" do
    source "db-connect.php.erb"
```

```
    mode 0660
    group deploy[:group]

    if platform?("ubuntu")
      owner "www-data"
    elsif platform?("amazon")
      owner "apache"
    end

    variables(
      :host =>     (deploy[:database][:host] rescue nil),
      :user =>     (deploy[:database][:username] rescue nil),
      :password => (deploy[:database][:password] rescue nil),
      :db =>       (deploy[:database][:database] rescue nil),
      :table =>    (node[:phpapp][:dbtable] rescue nil)
    )
    ...
  end
end
```

表示连接特征的变量，如 host、user 等被设置为来自部署 JSON 的 [:deploy][:app_name][:database] 属性的相应值。为简单起见，示例假定您已创建了一个名为 urler 的表，所以在说明书的属性文件中该表名表示为 [:phpapp][:dbtable]。

该配方实际上可以将 PHP 应用程序服务器与任何 MySQL 数据库服务器连接，而不仅仅是 MySQL 层的成员。要使用其他 MySQL 服务器，您只需将[:database]属性设置为适合您的服务器的值，您可以使用自定义 JSON 来完成此操作。 AWS OpsWorks 然后，堆栈将这些属性和值合并到堆栈配置和部署属性中，并appsetup.rb使用它们来创建用于设置连接的模板。有关重写堆栈配置和部署 JSON 的更多信息，请参阅覆盖属性。

如何将应用程序服务器实例连接到 Amazon RDS

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

本节介绍如何进行 MyStack 自定义Chef 11 Linux 堆栈入门，让 PHP 应用程序服务器连接到 RDS 实例。

主题

创建 Amazon RDS MySQL 数据库

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

现在，您已经准备好使用 Amazon RDS 控制台的“启动数据库实例向导”创建一个 RDS 数据库作为示例。以下过程是对基本详细信息的简要概括。有关如何创建数据库的详细描述，请参阅 Amazon RDS 入门。

创建 Amazon RDS 数据库

1. 如果这是您第一次创建 RDS 数据库，请单击 Get Started Now。否则，在导航窗格中单击 RDS Dashboard，然后单击 Launch a DB Instance。
2. 选择 MySQL Community Edition 作为数据库实例。
3. 对于 Do you plan to use this database for production purposes? (您是否计划将此数据库用于生产目的?)，选择 No, this instance... (否，此实例...)，这个选项对此示例已足够。若要用于生产用途，则可能需要选择 Yes, use Multi-AZ Deployment...。单击 Next Step (下一步)。
4. 在 Specify DB Details 页面上，指定以下设置：

   - DB Instance Class：db.t2.micro
   - Multi-AZ Deployment：No
   - 分配的存储空间：**5** GB
   - 数据库实例标识符：**`rdsexample`**
   - 主用户名：**`opsworksuser`**
   - Master Password：指定一个合适的密码并将它记录下来以供将来使用。

   接受其他选项的默认设置，然后单击 Next Step。

5. 在 Configure Advanced Settings 页面上，指定以下设置：

   - 在 Network & Security 部分中，对于 VPC Security Group(s)，选择 phpsecgroup (VPC)
   - 在 Database Options (数据库选项) 部分中，对于 Database Name (数据库名称)，键入 **rdsexampledb**。
   - 在 Backup 部分中，针对本演练，将 Backup Retention Period 设置为 0。

   接受其他选项的默认设置，然后单击 Launch DB Instance。
6. 选择 View Your DB Instances 以查看数据库实例列表。
7. 在列表中选择 rdsexample 实例，然后单击箭头显示该实例的终端节点和其他详细信息。记下终端节点以备以后使用。它类似于 `rdsexample.c6c8mntzhgv0.us-west-2.rds.amazonaws.com:3306`。只记下 DNS 名称；不需要端口号。
8. 使用 MySQL Workbench 等工具以及以下 SQL 命令在 `urler` 数据库中创建一个名为 `rdsexampledb` 的表：

```
CREATE TABLE urler(id INT UNSIGNED NOT NULL AUTO_INCREMENT,author VARCHAR(63) NOT
 NULL,message TEXT,PRIMARY KEY (id))
```

自定义堆栈以连接到 RDS 数据库

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

一旦创建了用作 PHP 应用程序服务器后端数据库的 RDS 实例，就可以 MyStack从中Chef 11 Linux 堆栈入门进行自定义。

将 PHP 应用程序服务器连接到 RDS 数据库

1. 打开 AWS OpsWorks Stacks 控制台，创建一个包含一个实例的 PHP App Server 层的堆栈，然后部署 SimplephPapp，如中所述。Chef 11 Linux 堆栈入门该堆栈采用版本 1 的 SimplePHPApp，不使用数据库连接。
2. 更新堆栈配置以使用包括 `appsetup.rb` 配方、相关模板和属性文件的自定义说明书。

1. 将 Use custom Chef cookbooks 设置为 Yes。
2. 将 Repository type 设置为 Git，并将 Repository URL 设置为 `git://github.com/amazonwebservices/opsworks-example-cookbooks.git`。

3. 将以下代码添加到堆栈的 Custom Chef JSON 框中，以将 RDS 连接数据分配给 `appsetup.rb` 用来创建配置文件的 `[:database]` 属性。

```
{
  "deploy": {
    "simplephpapp": {
      "database": {
        "username": "opsworksuser",
        "password": "your_password",
        "database": "rdsexampledb",
        "host": "rds_endpoint",
        "adapter": "mysql"
      }
    }
  }
}
```

使用以下属性值：

- username：在创建 RDS 实例时指定的主用户名。

  此示例使用 `opsworksuser`。
- password：创建 RDS 实例时指定的主密码。

  填写您指定的密码。
- database：在创建 RDS 实例时创建的数据库。

  此示例使用 `rdsexampledb`。
- host：您在上一部分中创建实例时从 RDS 控制台获得的 RDS 实例终端节点。不要包括端口号。
- adapter：适配器。

本示例的 RDS 实例使用 MySQL，所以 adapter 设置为 `mysql`。与其他属性不同，adapter 不由 `appsetup.rb` 使用，而是被 PHP 应用程序服务器层的内置配置配方用来创建不同的配置文件。

4. 编辑 SimplePHPApp 配置来指定使用后端数据库的 SimplePHPApp 版本，如下所示：

   - Document root：将此选项设置为 `web`。
   - Branch/Revision：将此选项设置为 `version2`。

   其余选项保持不变。

5. 编辑 PHP 应用程序服务器层来设置数据库连接，方法是将 添加到层的 Deploy 配方中。
6. 部署新的 SimplePHPApp 版本。
7. 部署 SimplePHPApp 后，通过转至 Instances 页面并单击 php-app1 实例的公有 IP 地址来运行该应用程序。您应在浏览器中看到以下页面，您可以在其中输入文本，然后将其存储在数据库中。

> **Note**
>
> 如果您的堆栈有 MySQL 层，则 AWS OpsWorks Stacks 会自动将相应的连接数据分配给 `[:database]` 属性。但是，如果您将自定义 JSON 分配给定义不同 `[:database]` 值的堆栈，则它们会覆盖默认值。因为 `[:deploy]` 属性安装在每个实例上，所以任何依赖

[:database] 属性的配方都将使用自定义连接数据而非 MySQL 层数据。如果您想让一个特定应用程序服务器层使用自定义连接数据，则将自定义 JSON 分配给该层的 Deploy 事件，并限制部署只到该层。有关如何使用部署属性的更多信息，请参阅部署应用程序。有关重写 AWS OpsWorks Stacks 内置属性的更多信息，请参阅覆盖属性。

使用 ElastiCache Redis 作为内存中的键值存储

Important

该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

Note

本主题基于 Linux 堆栈，但是 Windows 堆栈也可以使用 Amazon ElastiCache (ElastiCache)。有关如何使用 ElastiCache Windows 实例的示例，请参阅ElastiCache 作为 ASP.NET 会话存储。

通常，您可以通过使用缓存服务器提供小型数据项 (如字符串) 的内存中键值存储，以提高应用程序服务器性能。Amazon ElastiCache 是一项 AWS 服务，可以使用 Memcached 或 Redis 缓存引擎轻松为您的应用程序服务器提供缓存支持。 AWS OpsWorks Stacks 提供了对 Memcached 的内置支持。但是，如果 Redis 更适合您的需求，则可以自定义堆栈，以便您的应用程序服务器使用 ElastiCache Redis。

本主题以 Rails 应用程序服务器为例，向您介绍 ElastiCache 为 Linux 堆栈提供 Redis 缓存支持的基本过程。假设您已经有一个相应的 Ruby on Rails 应用程序。有关更多信息 ElastiCache，请参阅 Amazon 是什么 ElastiCache？ 。

主题

- 步骤 4：为 LifeCycle活动分配配方
- 步骤 5：将访问信息添加到堆栈配置 JSON
- 步骤 6：部署和运行应用程序

步骤 1：创建 ElastiCache Redis 集群

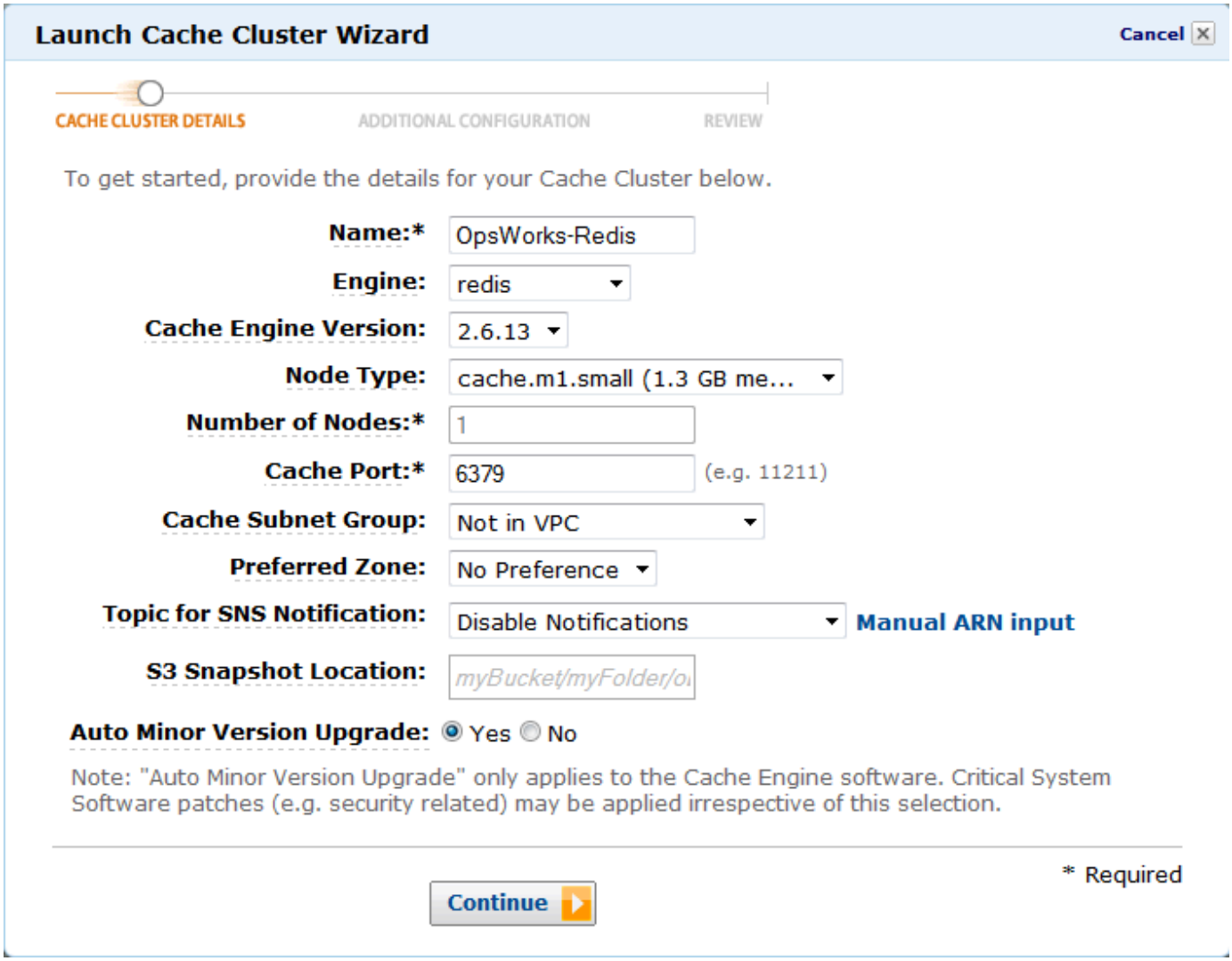
> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS：Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

您必须先使用 ElastiCache 控制台、API 或 CLI 创建 Amazon ElastiCache Redis 集群。下面介绍了如何使用控制台创建集群。

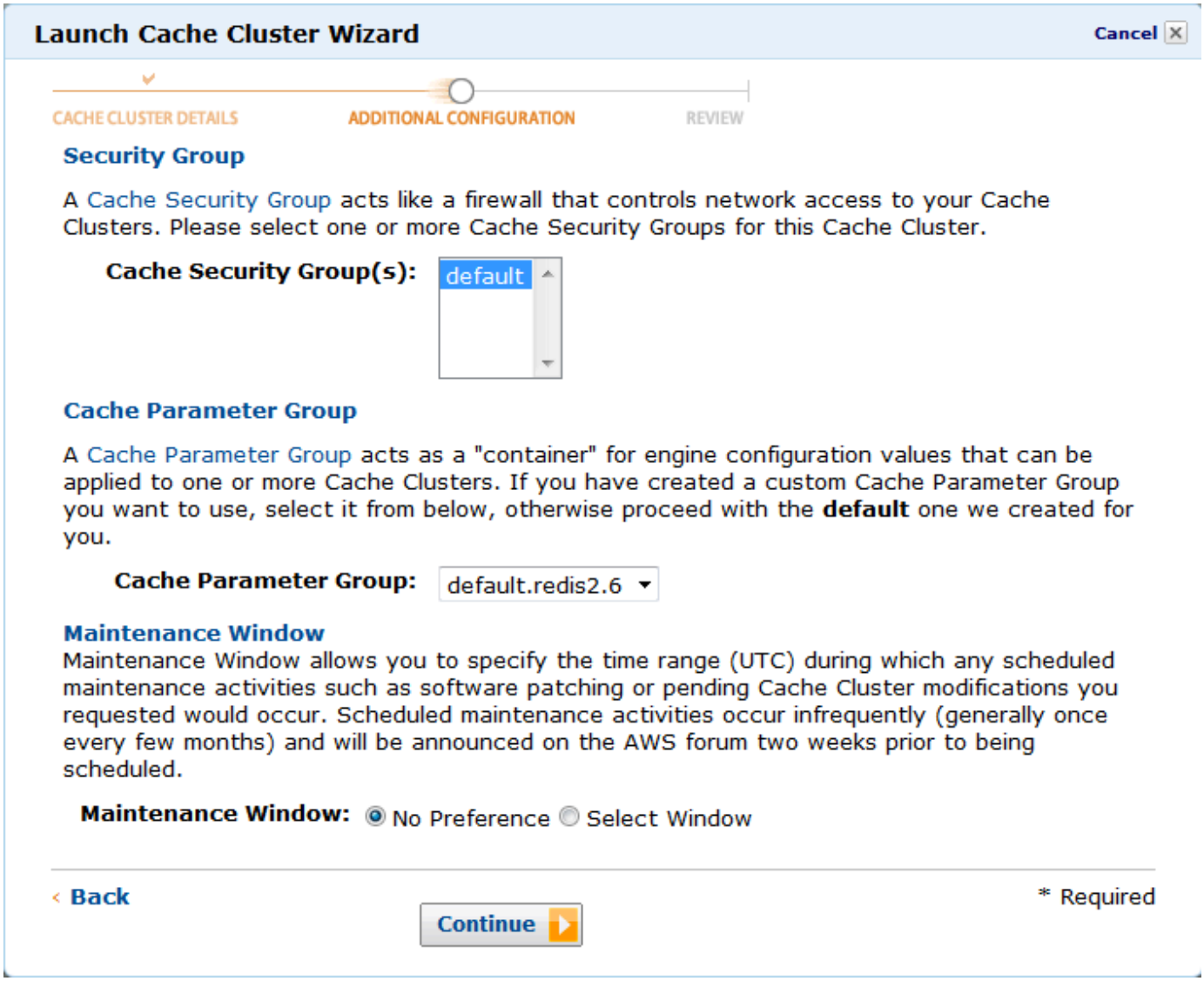
创建 ElastiCache Redis 集群

1. 转到ElastiCache控制台并单击 “启动缓存集群” 以启动缓存集群向导。
2. 在“Cache Cluster Details”页面上，执行以下操作：
   - 将 Name 设置为您的缓存服务器名称。

     此示例使用 OpsWorks-Redis。
   - 将 Engine 设置为 redis。
   - 将 Topic for SNS Notification 设置为 Disable Notifications。
   - 接受其他设置的默认值并单击 Continue。



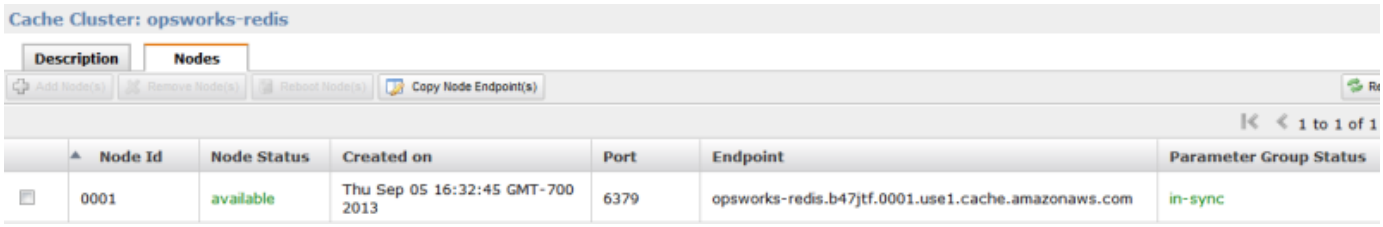
3. 在 Additional Configuration 页面上，接受默认值并单击 Continue。



4. 单击 Launch Cache Cluster 以创建集群。

> **Important**
>
> 对此示例来说，默认缓存安全组已足够，但在实际生产使用中，您应创建一个适合您的环境的缓存安全组。有关更多信息，请参阅管理缓存安全组。

5. 在集群开始后，单击该名称以打开详细信息页面，然后单击 Nodes 选项卡。记录集群的 Port 和 Endpoint 值以供将来使用。



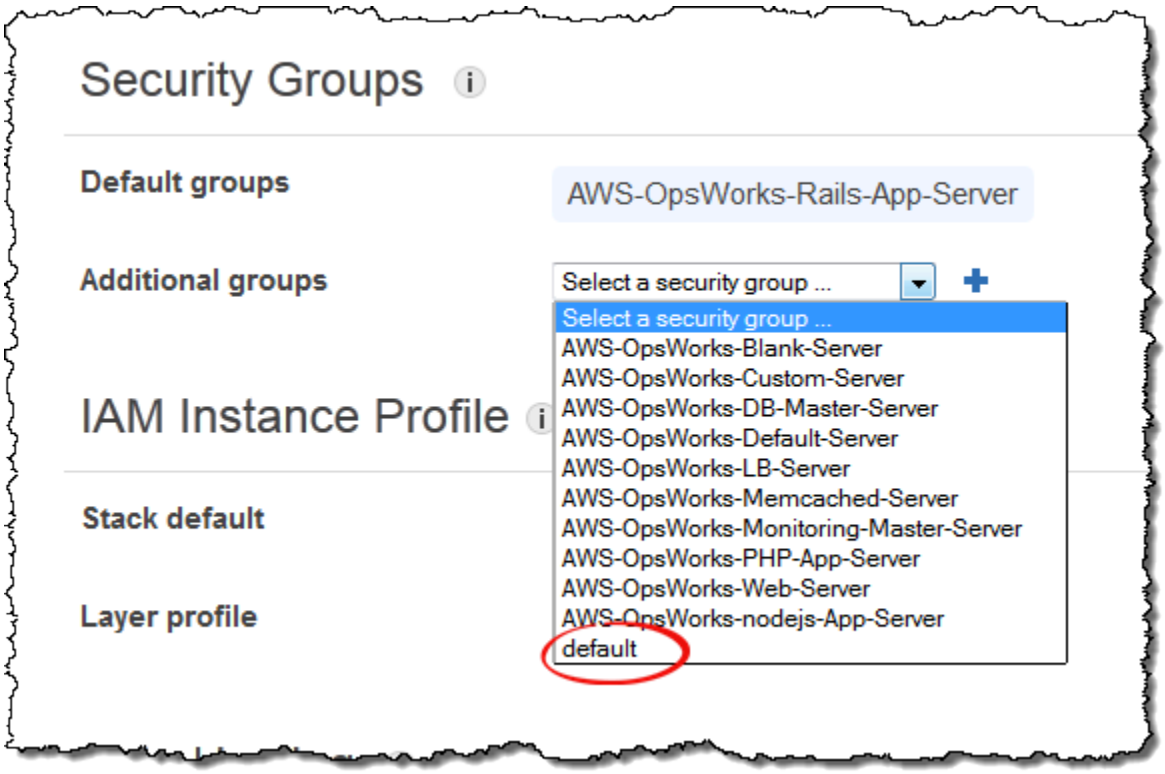
步骤 2：设置 Rails 堆栈

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

除了创建支持 Rails App Server 层的堆栈，您还必须配置该层的安全组，以便 Rails 服务器能够与 Redis 服务器正确通信。

设置堆栈

1. 创建一个为本示例命名为 **`RedisStack`** 的新堆栈，然后添加一个 Rails App Server 层。您可以为两者使用默认设置。有关更多信息，请参阅 创建新堆栈 和 创建图 OpsWorks 层 。
2. 在 Layers 页面上，对于 Rails App Server，单击 Security，然后单击 Edit。
3. 转到安全组部分，将 ElastiCache集群的安全组添加到其他组中。在本示例中，选择 default 安全组，单击 + 以将其添加到该层，然后单击 Save 以保存新配置。



4. 将一个实例添加到 Rails App Server 层，并启动该实例。有关如何添加和启动实例的更多信息，请参阅将实例添加到层。

步骤 3：创建和部署自定义说明书

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

目前，堆栈的功能尚不完备；您需要允许您的应用程序访问 Redis 服务器。最灵活的方法是在应用程序的 `config` 子文件夹中放置一个包含访问信息的 YAML 文件。这样，应用程序即可从该文件中获取信息。使用此方法，您可以更改连接信息，而无需重新编写和重新部署应用程序。在此示例中，文件应命名`redis.yml`并包含 ElastiCache 集群的主机名和端口，如下所示：

```
host: cache-cluster-hostname
port: cache-cluster-port
```

你可以手动将这个文件复制到你的服务器上，但更好的方法是实现一个 Chef 配方来生成文件，然后让 AWS OpsWorks Stacks 在每台服务器上运行这个配方。Chef 配方是专门的 Ruby 应用程序，AWS OpsWorks Stacks 使用它在实例上执行任务，例如安装软件包或创建配置文件。配方打包在说明书中，说明书可以包含多个配方和相关的文件，例如配置文件的模板。食谱放在存储库中，例如 GitHub，并且必须具有标准的目录结构。如果您还没有自定义说明书存储库，请参阅说明书存储库，以了解如何进行设置的信息。

在本示例中，将使用以下内容，向您的说明书存储库中添加一个名为 `redis-config` 的说明书：

```
my_cookbook_repository
  redis-config
    recipes
      generate.rb
    templates
      default
        redis.yml.erb
```

`recipes` 文件夹包含一个名为 `generate.rb` 的配方，该配方从 `redis.yml.erb` 生成应用程序的配置文件，如下所示：

```
node[:deploy].each do |app_name, deploy_config|
  # determine root folder of new app deployment
  app_root = "#{deploy_config[:deploy_to]}/current"

  # use template 'redis.yml.erb' to generate 'config/redis.yml'
  template "#{app_root}/config/redis.yml" do
    source "redis.yml.erb"
    cookbook "redis-config"

    # set mode, group and owner of generated file
    mode "0660"
    group deploy_config[:group]
    owner deploy_config[:user]

    # define variable “@redis” to be used in the ERB template
    variables(
      :redis => deploy_config[:redis] || {}
    )

    # only generate a file if there is Redis configuration
```

```
    not_if do
      deploy_config[:redis].blank?
    end
  end
end
```

配方取决于来自 St ac AWS OpsWorks ks 堆栈配置和部署 JSON 对象的数据，该对象安装在每个实例上，包含有关堆栈和任何已部署应用程序的详细信息。该对象的 deploy 节点具有以下结构：

```
{
   ...
  "deploy": {
    "app1": {
      "application" : "short_name",
      ...
    }
    "app2": {
      ...
    }
    ...
  }
}
```

此部署节点包含一组嵌入式 JSON 对象 (每个已部署应用程序对应一个对象，且对象以应用程序的短名称命名)。每个应用程序对象包含一组定义应用程序配置的属性，例如文档根和应用程序类型。关于 deploy 属性的列表，请参阅 deploy 属性。配方可以使用 Chef 属性语法来表示堆栈配置和部署 JSON 值。例如，[:deploy][:app1][:application] 表示 app1 应用程序的短名称。

对于 [:deploy] 中的每个应用程序，该配方执行关联的代码块，其中 deploy_config 表示应用程序属性。该配方首先将 app_root 设置为应用程序的根目录 [:deploy][:*app_name*][:deploy_to]/current。然后，它使用 Chef 模板资源从 redis.yml.erb 生成配置文件，并将该文件放在 app_root/config 中。

配置文件通常是从模板创建的，而且许多 (即便不是大多数) 设置是由 Chef 属性定义。借助这些属性，您可以使用自定义 JSON 更改设置 (如下文所述)，而不是重写模板文件。redis.yml.erb 模板包含以下内容：

```
host: <%= @redis[:host] %>
```

```
port: <%= @redis[:port] || 6379 %>
```

<%... %> 元素是表示属性值的占位符。

- `<%= @redis[:host] %>` 表示 `redis[:host]` 的值，即缓存集群的主机名。
- `<%= @redis[:port] || 6379 %>` 表示 `redis[:port]` 的值，如果未定义该属性，则表示默认端口值 6379。

`template` 资源的工作方式如下所示：

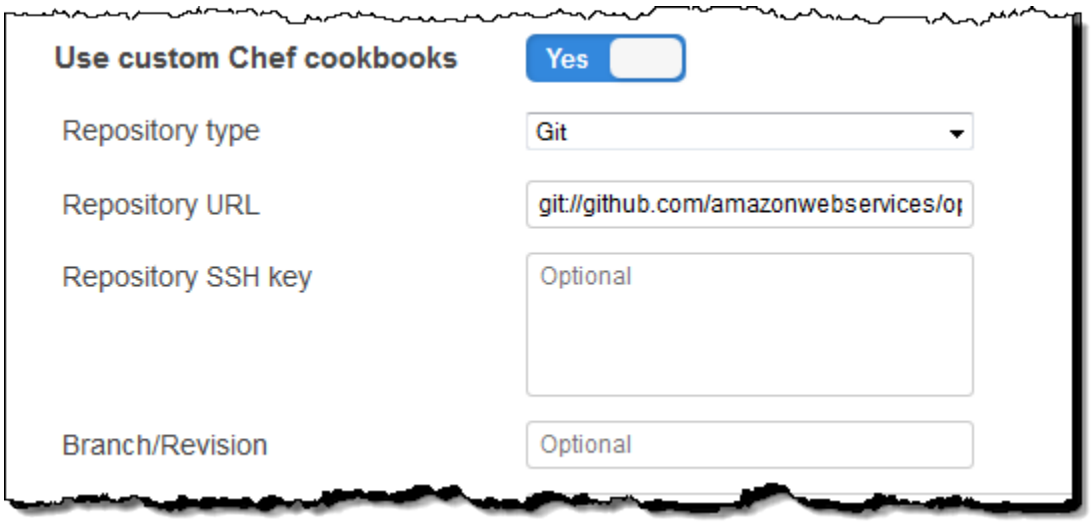
- `source` 和 `cookbook` 分别指定模板和说明书名称。
- `mode`、`group` 和 `owner` 为配置文件提供与应用程序相同的访问权限。
- `variables` 部分将模板中使用的 `@redis` 变量设置为应用程序的 `[:redis]` 属性值。

  通过使用自定义 JSON 来设置 `[:redis]` 属性的值，如下文所述；它不是标准应用程序属性之一。
- `not_if` 指令可确保该配方在已存在配置文件的情况下不会再生成配置文件。

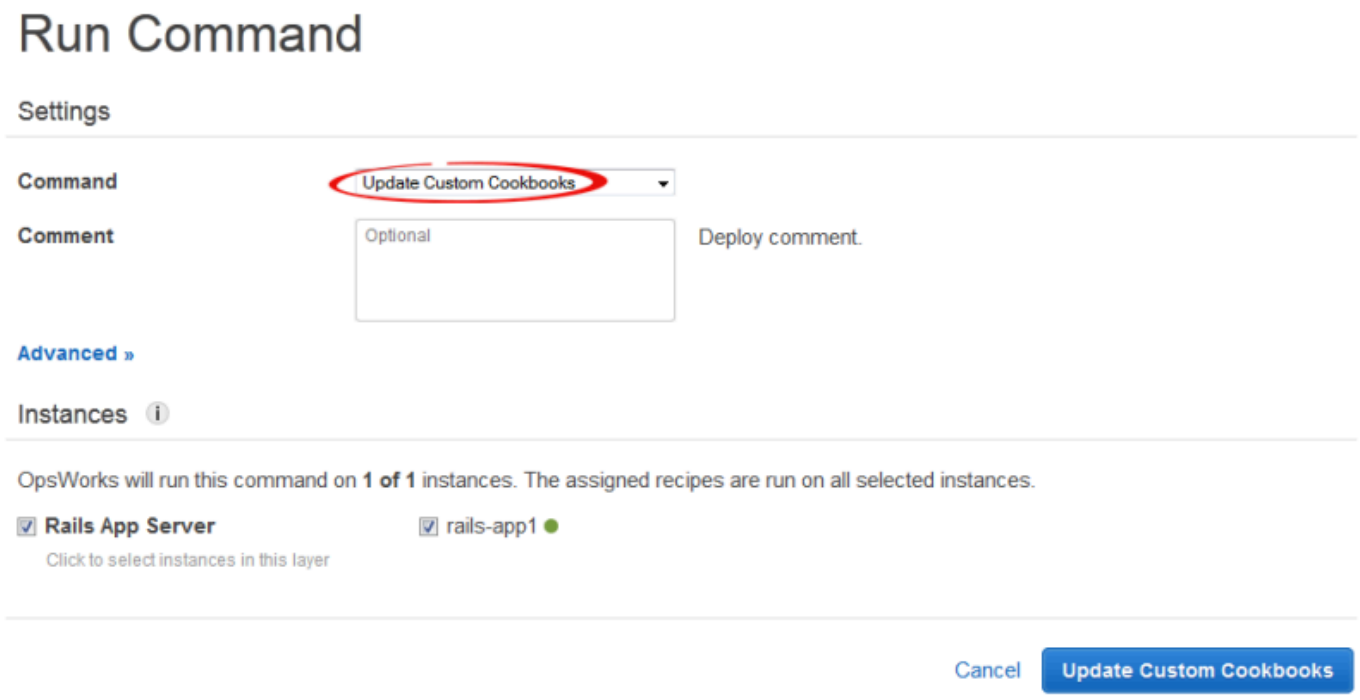
编写说明书后，您必须将其部署到每个实例的说明书缓存。此操作并不运行该配方，它只是在堆栈的实例上安装新说明书。一般情况下，可通过将配方分配到某个层的生命周期事件来运行该配方，如下文所述。

部署自定义说明书

1. 在 “堆栈 AWS OpsWorks 堆栈” 页面上，单击 “堆栈设置”，然后单击 “编辑”。
2. 在 Configuration Management 部分中，将 Use custom Chef cookbooks 设置为 Yes，输入说明书存储库信息，然后单击 Save 以更新堆栈配置。



3. 在 Stack 页面上，单击 Run Command，选择 Update Custom Cookbooks 堆栈命令，然后单击 Update Custom Cookbooks 以将新说明书安装到实例的说明书缓存中。



如果您修改您的说明书，只需再次运行 Update Custom Cookbooks 来安装更新版本。有关此过程的更多信息，请参阅安装自定义说明书。

步骤 4：为 LifeCycle活动分配配方

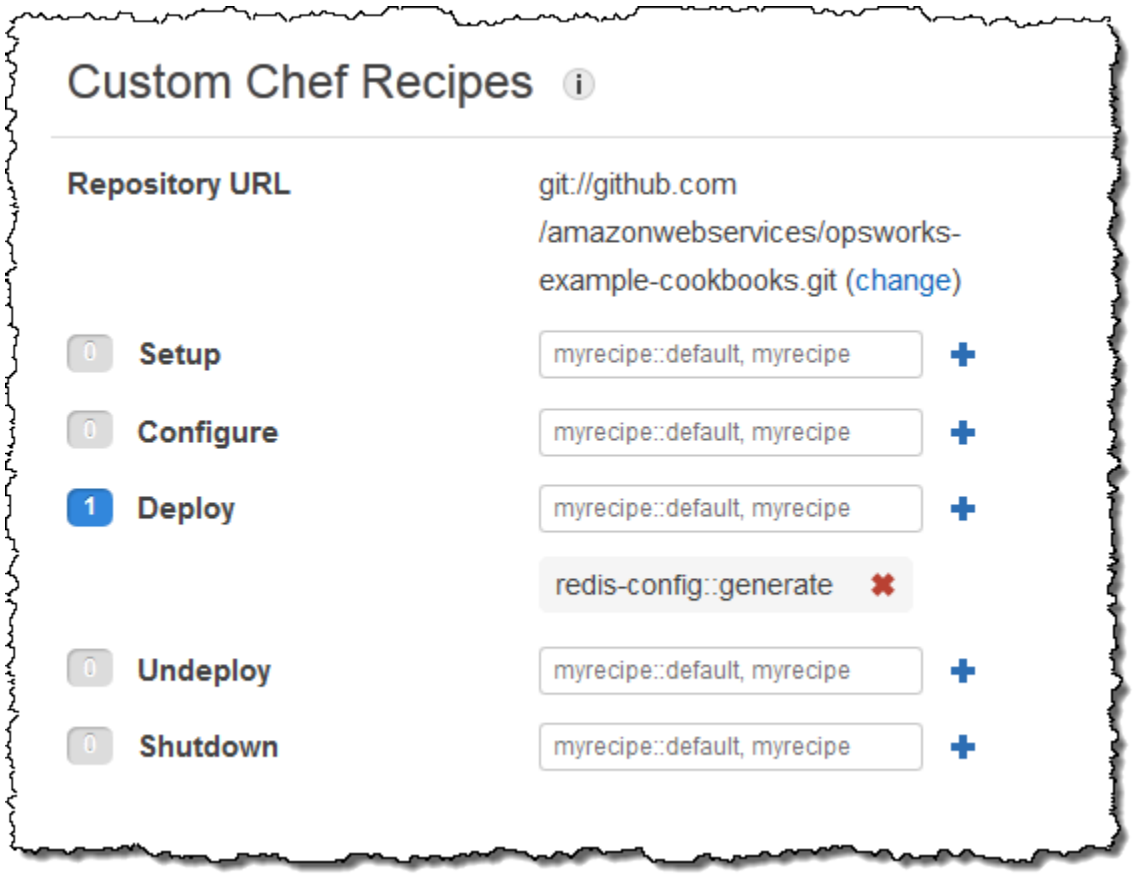
> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

你可以手动运行自定义配方，但最好的方法通常是让 AWS OpsWorks Stacks 自动运行它们。每层都有一组内置配方，分配到 Setup、Configure、Deploy、Undeploy 和 Shutdown 这五个生命周期事件。实例上每次发生某个事件时， AWS OpsWorks Stacks 为实例的每一层运行关联的配方来处理对应的任务。例如，当实例完成启动时， AWS OpsWorks Stacks 会触发安装事件。此事件运行关联层的 Setup 配方，这通常处理诸如安装和配置软件包之类的任务。

通过将配方分配给相应的生命周期事件，可以让 AWS OpsWorks Stacks 在图层的实例上运行自定义配方。在本示例中，您应该将`generate.rb`配方分配给 Rails App Server 层的 Deploy 事件。 AWS OpsWorks 然后，Stacks 将在启动期间、安装配方完成后以及每次部署应用程序时在层的实例上运行它。有关更多信息，请参阅 自动运行配方。

将配方分配给 Rails App Server 层的 Deploy 事件

1. 在 AWS OpsWorks Stacks L ay ers 页面上，对于 Rails App Server，单击 “食谱”，然后单击 “编辑”...
2. 在 Custom Chef Recipes 下，将完全限定的配方名称添加到 Deploy 事件，然后单击 +。完全限定的配方名称采用 *`cookbookname`*`::`*`recipename`* 格式，其中 *`recipename`* 不包含 `.rb` 扩展名。在本例中，完全限定的名称为 `redis-config::generate`。然后单击 Save 以更新层配置。



步骤 5：将访问信息添加到堆栈配置 JSON

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

`generate.rb` 配方依赖于表示 Redis 服务器主机名和端口的一对堆栈配置和部署 JSON 属性。尽管这些属性是标准`[:deploy]`命名空间的一部分，但它们不是由 AWS OpsWorks Stacks 自动定义的。而是由您通过将自定义 JSON 对象添加到堆栈来定义属性及其值。以下示例显示了本例所述的自定义 JSON。

将访问信息添加到堆栈配置和部署 JSON

1. 在“堆栈 AWS OpsWorks 堆栈”页面上，单击“堆栈设置”，然后单击“编辑”。

2. 在 Configuration Management 部分中，将访问信息添加到 Custom Chef JSON 框中。它应类似于以下示例，但需要进行以下修改：

   - 将 `elasticache_redis_example` 替换为应用程序的短名称。
   - 将`host`和`port`值替换为您在中创建 ElastiCache 的 Redis 服务器实例的步骤 1：创建 ElastiCache Redis 集群值。

```
{
  "deploy": {
     "elasticache_redis_example": {
       "redis": {
         "host": "mycluster.XXXXXXXXX.amazonaws.com",
         "port": "6379"
       }
     }
  }
}
```

这种方法的优点是，您可以随时更改端口或主机值，而无需触摸自定义食谱。 AWS OpsWorks Stacks 将自定义 JSON 合并到内置 JSON 中，并将其安装在堆栈的实例上，用于所有后续生命周期事件。然后，应用程序可通过使用 Chef 节点语法来访问属性值，如步骤 3：创建和部署自定义说明书中所述。下次部署应用程序时， AWS OpsWorks Stacks 将安装包含新定义的堆栈配置和部署 JSON，并且 `generate.rb` 将创建一个具有更新的主机和端口值的配置文件。

> **Note**
>
> [:deploy] 会自动为每个部署的应用程序包含一个属性，因此 [:deploy] [elasticache_redis_example] 已在堆栈和配置 JSON 中。但是，[:deploy] [elasticache_redis_example]不包含[:redis]属性，使用自定义 JSON 定义它们会指示 AWS OpsWorks Stacks 将这些属性添加到[:deploy] [elasticache_redis_example]。您还可以使用自定义 JSON 覆盖现有属性。有关更多信息，请参阅 覆盖属性。

步骤 6：部署和运行应用程序

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

本示例假定您拥有使用 Redis 的 Ruby on Rails 应用程序。要访问配置文件，您可以将 `redis` gem 添加到 Gemfile，并按如下所示在 `config/initializers/redis.rb` 中创建 Rails 初始化程序：

```
REDIS_CONFIG = YAML::load_file(Rails.root.join('config', 'redis.yml'))
$redis = Redis.new(:host => REDIS_CONFIG['host'], :port => REDIS_CONFIG['port'])
```

然后，创建应用程序以表示您的应用程序，并将其部署到 Rails App Server 层的实例，这将更新应用程序代码并运行 `generate.rb` 以生成配置文件。运行应用程序时，它将使用 ElastiCache Redis 实例作为其内存中的键值存储。

使用 Amazon S3 存储桶

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

应用程序通常使用 Amazon Simple Storage Service (Amazon S3)存储桶来存储图像或其他媒体文件等大型项目。尽管 AWS OpsWorks Stacks 不提供对 Amazon S3 的集成支持，但您可以轻松地自定义堆栈以允许您的应用程序使用 Amazon S3 存储。本主题向您演示了向应用程序提供 Amazon S3 访问权限的基本过程，使用带有 PHP 应用程序服务器的 Linux 堆栈为例。基本原则同样适用于 Windows 堆栈。

发送到 Amazon S3 存储桶的内容可能包含客户内容。有关删除敏感数据的更多信息，请参阅如何清空 S3 存储桶？或如何删除 S3 存储桶？。

主题

步骤 1：创建 Amazon S3 存储桶

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS：Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

您必须先创建 Amazon S3 存储桶。您可以使用 Amazon S3 控制台、API 或 CLI 来直接完成此操作，不过创建资源更简单的方法通常是使用 AWS CloudFormation 模板。以下模板为此示例创建 Amazon S3 存储桶，并设置具有 IAM 角色的实例配置文件，授予对存储桶的无限制访问权限。然后，您可以使用层设置来将实例配置文件附加到堆栈的应用程序服务器实例，这会允许应用程序访问存储桶，如下文所述。实例配置文件的实效性不限制为 Amazon S3，对于集成各种 Amazon Web Service 非常有价值。

```
{
   "AWSTemplateFormatVersion" : "2010-09-09",
   "Resources" : {
```

```
      "AppServerRootRole": {
         "Type": "AWS::IAM::Role",
         "Properties": {
            "AssumeRolePolicyDocument": {
               "Statement": [ {
                  "Effect": "Allow",
                  "Principal": {
                     "Service": [ "ec2.amazonaws.com" ]
                  },
                  "Action": [ "sts:AssumeRole" ]
               } ]
            },
            "Path": "/"
         }
      },
      "AppServerRolePolicies": {
         "Type": "AWS::IAM::Policy",
         "Properties": {
            "PolicyName": "AppServerS3Perms",
            "PolicyDocument": {
               "Statement": [ {
                  "Effect": "Allow",
                  "Action": "s3:*",
                  "Resource": { "Fn::Join" : ["", [ "arn:aws:s3:::", { "Ref" :
 "AppBucket" } , "/*" ]
                  ] }
               } ]
            },
            "Roles": [ { "Ref": "AppServerRootRole" } ]
         }
      },
      "AppServerInstanceProfile": {
         "Type": "AWS::IAM::InstanceProfile",
         "Properties": {
            "Path": "/",
            "Roles": [ { "Ref": "AppServerRootRole" } ]
         }
      },
     "AppBucket" : {
      "Type" : "AWS::S3::Bucket"
      }
   },
   "Outputs" : {
       "BucketName" : {
```

```
            "Value" : { "Ref" : "AppBucket" }
        },
        "InstanceProfileName" : {
            "Value" : { "Ref" : "AppServerInstanceProfile" }
        }
    }
}
```

在您启动模板时，将发生几件事：

- `AWS::S3::Bucket` 资源创建 Amazon S3 存储桶。
- `AWS::IAM::InstanceProfile` 资源创建将分配到应用程序服务器实例的实例配置文件。
- `AWS::IAM::Role` 资源创建实例配置文件的角色。
- `AWS::IAM::Policy` 资源将角色的权限设置为允许不受限制地访问 Amazon S3 存储桶。
- 在您启动模板之后，`Outputs` 部分在 AWS CloudFormation 控制台中显示存储桶和实例配置文件名称。

  您需要这些值来设置堆栈和应用程序。

有关如何创建 AWS CloudFormation 模板的更多信息，请参阅学习模板基础知识。

创建 Amazon S3 存储桶

1. 将示例模板复制到系统上的文本文件。

   此示例假定文件名为 `appserver.template`。
2. 打开 AWS CloudFormation 控制台并选择创建堆栈。
3. 在 Stack Name 框中，输入堆栈名称。

   此示例假定名称为 **`AppServer`**。
4. 请依次选择 Upload template file (上传模板文件)、Browse (浏览)、您在步骤 1 中创建的 `appserver.template` 文件以及 Next Step (下一步)。
5. 在 Specify Parameters (指定参数) 页面上，选择 I acknowledge that this template may create IAM resources (我确认，此模板可创建 IAM 资源)，然后在向导的各页面上选择 Next Step (下一步)，直至您完成。选择创建。
6. AppServer堆栈达到 CREATE_COMPLETE 状态后，将其选中并选择 “输出” 选项卡。

   您可能需要刷新几次来更新状态。

7. 在 “输出” 选项卡上，记录BucketName和InstanceProfileName值以备后用。

> **Note**
>
> AWS CloudFormation 使用术语堆栈来指根据模板创建的资源集合；它与堆栈 AWS OpsWorks 堆栈不同。

步骤 2：创建 PHP App Server 堆栈

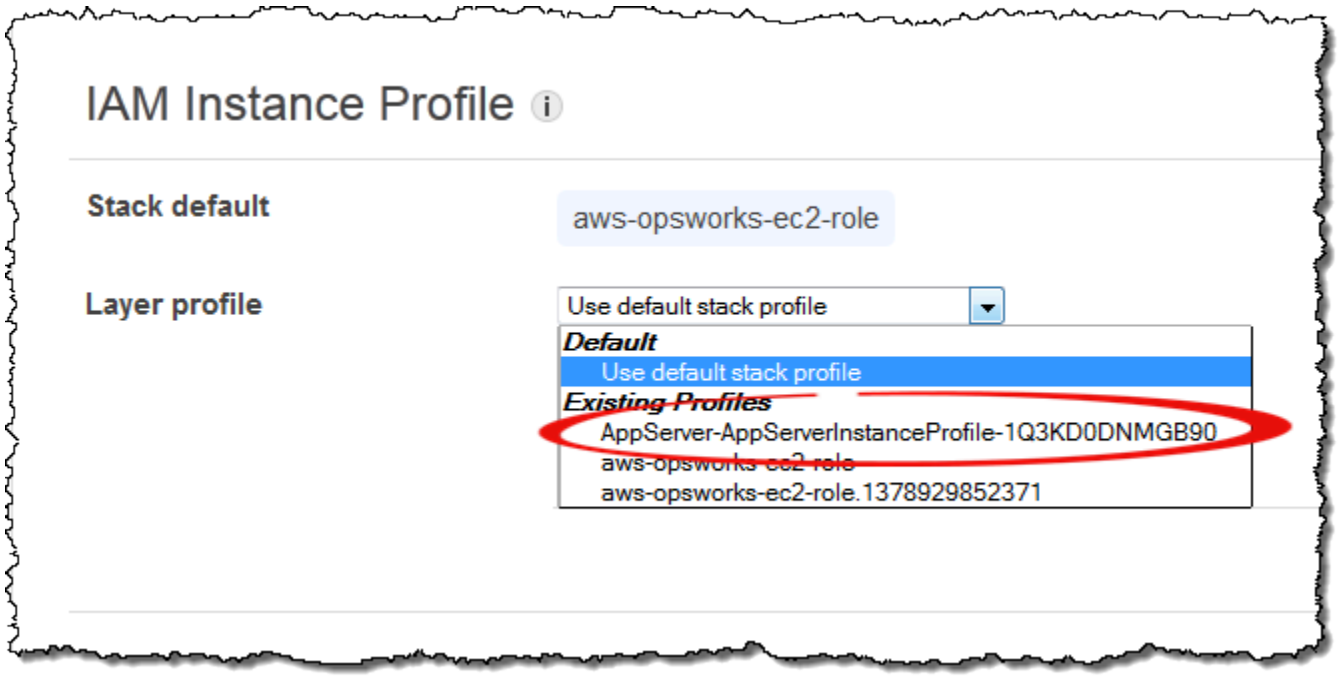
> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

堆栈包含两层， PHP App Server 和 MySQL，每一层有一个实例。应用程序在 Amazon S3 存储桶上存储照片，但使用 MySQL 实例作为后端数据存储，用于存储每张照片的元数据。

发送到 Amazon S3 存储桶的内容可能包含客户内容。有关删除敏感数据的更多信息，请参阅如何清空 S3 存储桶？或如何删除 S3 存储桶？。

要创建 堆栈，请执行以下操作：

1. 创建一个为本示例命名为 **`PhotoSite`** 的新堆栈，然后添加一个 PHP App Server 层。您可以为两者使用默认设置。有关更多信息，请参阅 创建新堆栈 和 创建图 OpsWorks 层 。
2. 在 层 页面上，对于 PHP App Server，选择 安全性，然后选择 编辑。
3. 在层配置文件部分，选择您在启动 AppServer AWS CloudFormation 堆栈后之前记录的实例配置文件名称。会是这样的`AppServer-AppServerInstanceProfile-1Q3KD0DNMGB90`。 AWS OpsWorks Stacks 会将此配置文件分配给该层的所有 Amazon EC2 实例，从而向在该层实例上运行的应用程序授予访问您的 Amazon S3 存储桶的权限。



4. 将一个实例添加到 PHP App Server 层，并启动该实例。有关如何添加和启动实例的更多信息，请参阅将实例添加到层。
5. 添加 MySQL 层到堆栈，添加实例，然后启动它。您可以同时为层和实例使用默认设置。特别是，MySQL 实例不需要访问 Amazon S3 存储桶，因此它可以使用默认选择的标准 AWS OpsWorks 堆栈实例配置文件。

步骤 3：创建和部署自定义说明书

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

堆栈尚未就绪：

- 您的应用程序需要一些信息来访问 MySQL 数据库服务器和 Amazon S3 存储桶，例如数据库主机名和 Amazon S3 存储桶名称。
- 您需要在 MySQL 数据库服务器中设置数据库，并创建表来保存照片的元数据。

你可以手动处理这些任务，但更好的方法是实现 Chef 配方，让 AWS OpsWorks Stacks 在相应的实例上自动运行配方。Chef 配方是专门的 Ruby 应用程序， AWS OpsWorks Stacks 使用它在实例上执行任务，例如安装软件包或创建配置文件。它们打包在说明书 中，说明书可以包含多个配方和相关的文件，例如配置文件的模板。食谱放在存储库中，例如 GitHub，并且必须具有标准的目录结构。如果您还没有自定义说明书存储库，请参阅说明书存储库，以了解如何进行设置的信息。

在此示例中，食谱已为您实现并存储在公共存储 GitHub 库中。说明书包含两个配方 `appsetup.rb` 和 `dbsetup.rb`，以及一个模板文件 `db-connect.php.erb`。

`appsetup.rb` 配方将创建配置文件，其中包含应用程序访问数据库和 Amazon S3 存储桶所需信息。它基本上是 将应用程序连接到数据库 中所述的 `appsetup.rb` 配方经过略微修改的版本。主要差别在于传递到模板的变量，这代表了访问信息。

前四个属性定义数据库连接设置，由 AWS OpsWorks 堆栈在创建 MySQL 实例时自动定义。

这些变量与原始配方中的变量之间有两个主要差别：

- 与原始配方一样，`table` 变量表示由 `dbsetup.rb` 创建的数据库表的名称，并设置为在说明书的属性文件中定义的属性的值。

  但是，该属性有不同的名称：`[:photoapp][:dbtable]`。
- `s3bucket` 变量特定于此示例，设置为表示 Amazon S3 存储桶名称的属性 `[:photobucket]` 的值。

  `[:photobucket]` 使用自定义 JSON 定义，如下文所述。有关属性的更多信息，请参阅Attributes。

有关属性的更多信息，请参阅Attributes。

`dbsetup.rb` 配方设置数据库表用于保存各个照片的元数据。它基本上是 `dbsetup.rb` 中所述的 设置数据库 配方经过略微修改的版本；有关详细说明，请参阅该主题。

此示例与原始配方之间的唯一差别是数据库架构，它有三列，包含 ID、URL 和各照片的标题，存储在 Amazon S3 存储桶中。

食谱已经实现了，所以你所需要做的就是将 photoapp 食谱部署到每个实例的食谱缓存中。 AWS OpsWorks 然后，当相应的生命周期事件发生时，Stacks 会运行缓存的配方，如下所述。

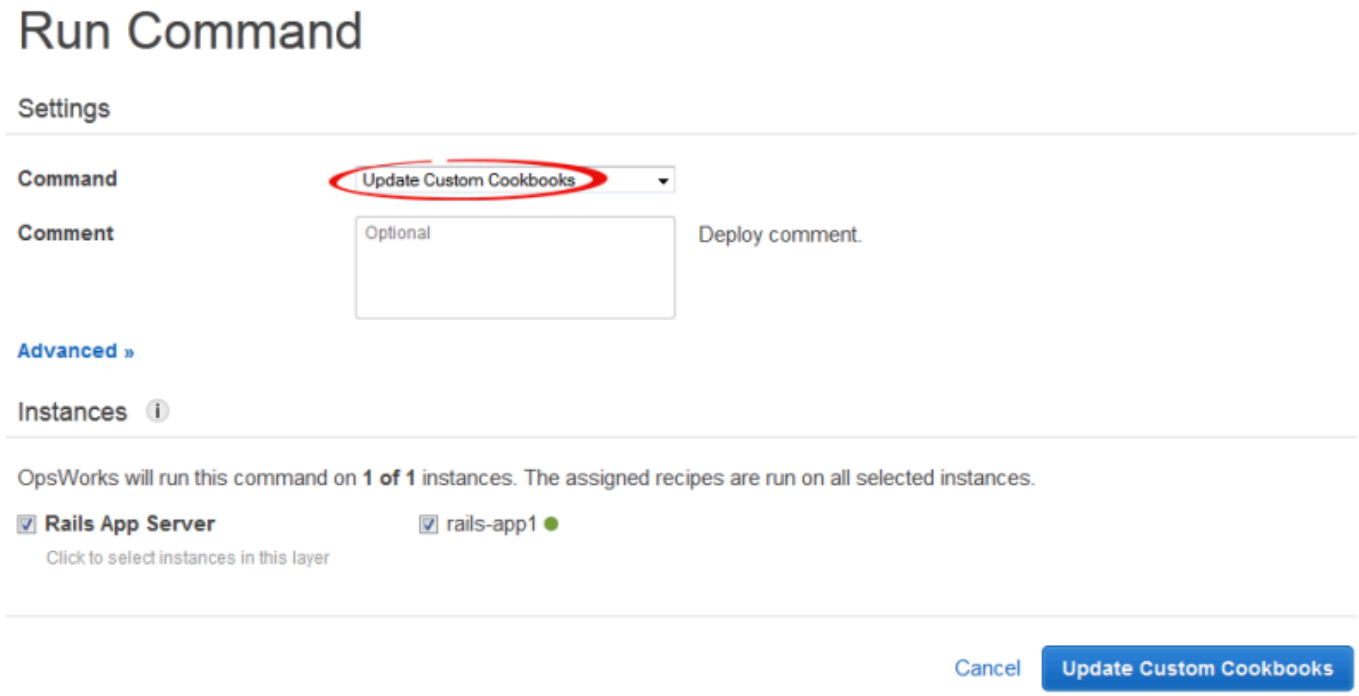
部署 photoapp 说明书

1. 在 “堆栈 AWS OpsWorks 堆栈” 页面上，选择 “堆栈设置”，然后选择 “编辑”。

2. 在 Configuration Management 部分中：

   - 将 Use custom Chef cookbooks 设置为 Yes。
   - 将 Repository type 设置为 Git。
   - 将 Repository URL (存储库 URL) 设置为 **`git://github.com/amazonwebservices/opsworks-example-cookbooks.git`**。

3. 在 Stack (堆栈) 页面上，依次选择 Run Command (运行命令)、Update Custom Cookbooks (更新自定义说明书) 堆栈命令以及 Update Custom Cookbooks (更新自定义说明书)，以将新说明书安装到实例说明书缓存中。



第 4 步：为 LifeCycle 活动分配食谱

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

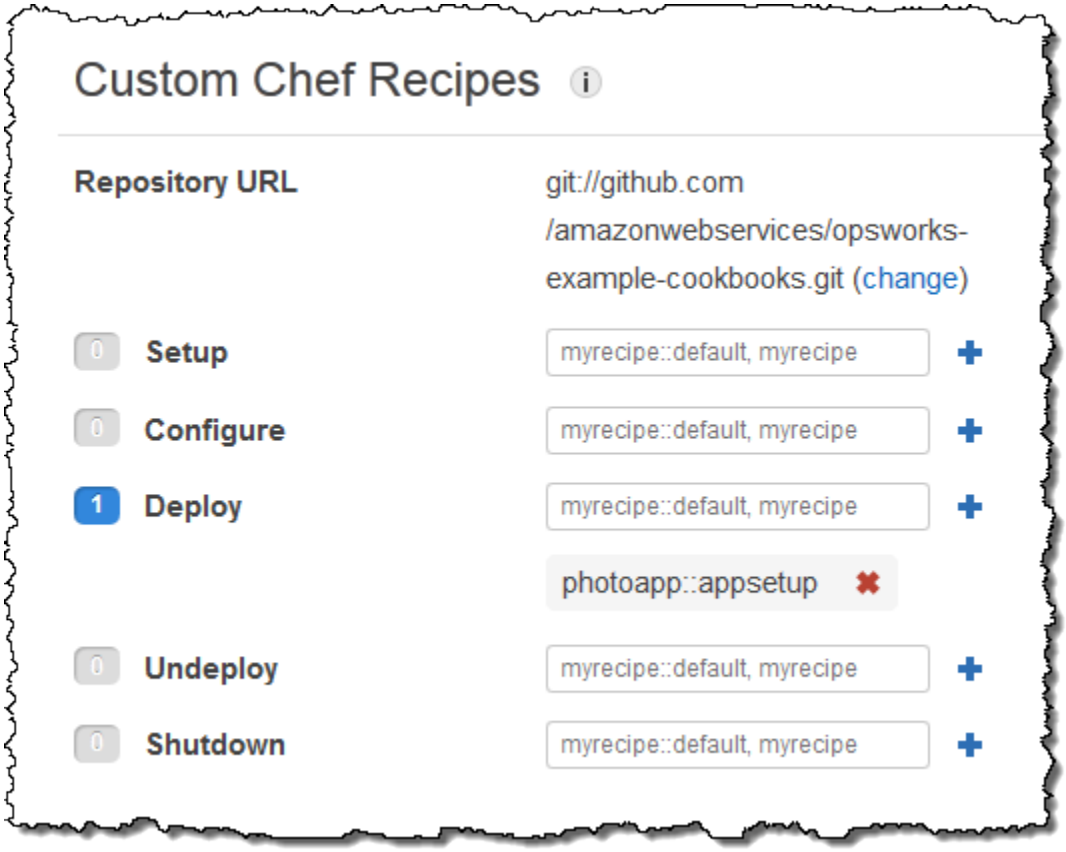
你可以手动运行自定义配方，但最好的方法通常是让 AWS OpsWorks Stacks 自动运行它们。每层都有一组内置配方，分配到 Setup、Configure、Deploy、Undeploy 和 Shutdown 这五个生命周期事件。

每次实例上发生事件时， AWS OpsWorks Stacks 都会为实例的每个层运行关联配方，这些配方处理所需的任务。例如，当实例完成启动时， AWS OpsWorks Stacks 会触发安装事件来运行安装配方，该配方通常处理安装和配置软件包等任务。

通过将每个配方分配给相应的生命周期事件，可以让 AWS OpsWorks Stacks 在图层的实例上运行自定义配方。 AWS OpsWorks 图层的内置配方完成后，堆栈将运行任何自定义配方。在本示例中，分配`appsetup.rb`给 PHP 应用服务器层的 Deploy 事件和 `dbsetup.rb` MySQL 层的 Deploy 事件。 AWS OpsWorks 然后，堆栈将在启动期间、内置安装配方完成后、每次部署应用程序时、构建的 Deploy 配方完成后，在关联层的实例上运行配方。有关更多信息，请参阅 自动运行配方。

将自定义配方分配到层的部署事件

1. 在 AWS OpsWorks Stacks L ay ers 页面上，对于 PHP App Server，选择食谱，然后选择编辑。
2. 在 Custom Chef Recipes (自定义 Chef 配方) 下，将配方名称添加到 Deploy 事件，然后选择 +。该名称必须为 Chef *`cookbookname`*`::`*`recipename`* 格式，其中 *`recipename`* 不包括 `.rb` 扩展名。对于此示例，您输入 `photoapp::appsetup`。然后选择 Save (保存) 以更新层配置。



3. 在 层 页面上，选择 MySQL 层的 操作 列中的 编辑。
4. 将 `photoapp::dbsetup` 添加到层的部署事件，然后保存新配置。

步骤 5：将访问信息添加到堆栈配置和部署属性

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

`appsetup.rb`配方取决于堆栈 AWS OpsWorks 堆栈配置和部署属性的数据，这些数据安装在每个实例上，包含有关堆栈和任何已部署应用程序的详细信息。对象的 `deploy` 属性具有以下结构，为便利起见显示为 JSON：

```
{
   ...
  "deploy": {
    "app1": {
      "application" : "short_name",
      ...
    }
    "app2": {
      ...
    }
    ...
  }
}
```

部署节点包含各个已部署应用程序的属性，使用应用程序的短名称命名。每个应用程序属性包含一组定义应用程序配置的属性，例如文档根和应用程序类型。对于 `deploy` 属性的列表，请参阅 deploy 属性。您可以使用 Chef 属性语法显示配方中的堆栈配置和部署属性值。例如，`[:deploy][:app1][:application]` 表示 app1 应用程序的短名称。

自定义配方依赖于代表数据库和 Amazon S3 访问信息的多个堆栈配置和部署属性：

- 数据库连接属性（例如`[:deploy][:database][:host]`）由 AWS OpsWorks Stacks 在创建 MySQL 层时定义。
- 表名称属性 `[:photoapp][:dbtable]` 在自定义说明书的属性文件中定义，并设置为 `foto`。

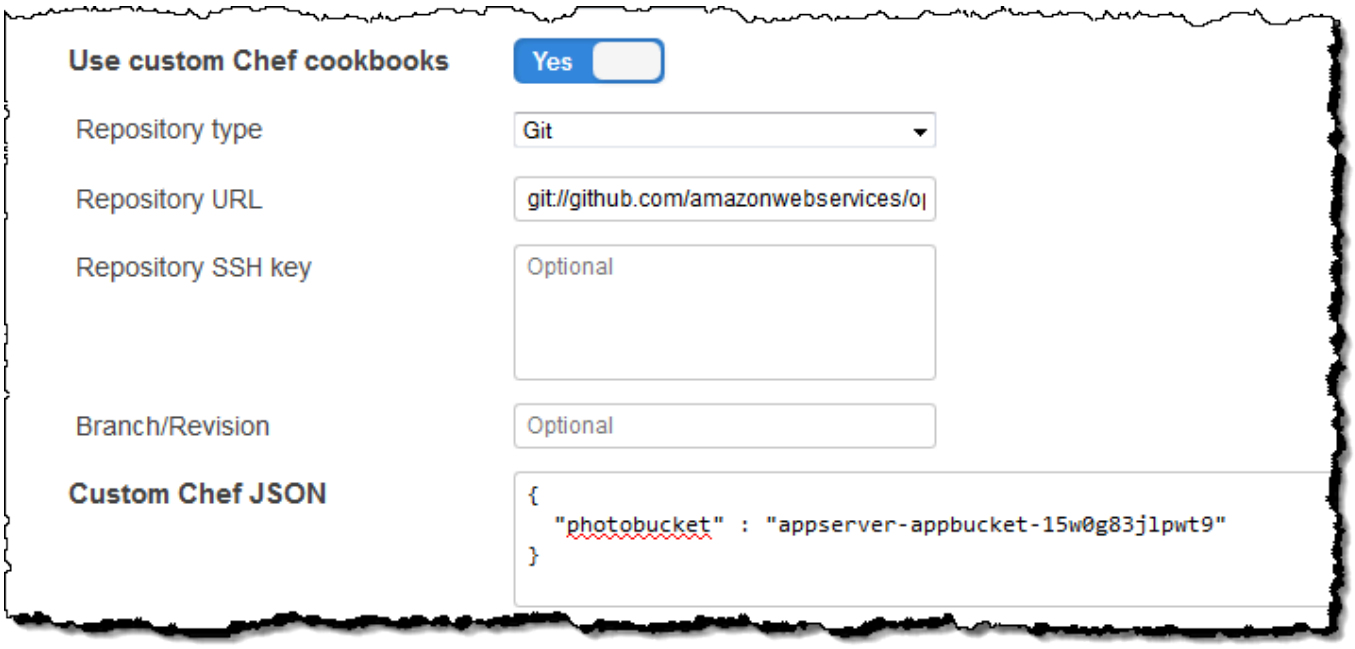
- 您必须使用自定义 JSON 将属性添加到堆栈配置和部署属性，以此来定义存储桶名称属性 `[:photobucket]`。

定义 Amazon S3 存储桶名称属性

1. 在堆栈 AWS OpsWorks 堆栈页面上，选择堆栈设置，然后选择编辑。
2. 在 Configuration Management 部分中，将访问信息添加到 Custom Chef JSON 框中。它应与以下内容类似：

```
{
  "photobucket" : "yourbucketname"
}
```

使用您在 *##########* yourbucketname步骤 1：创建 Amazon S3 存储桶。



AWS OpsWorks 堆栈将自定义 JSON 合并到堆栈配置和部署属性中，然后再将其安装到堆栈的实例上；然后`appsetup.rb`可以从该`[:photobucket]`属性中获取存储桶名称。如果要更改存储桶，您并不需要接触配方；只需覆盖属性来提供新存储桶名称。

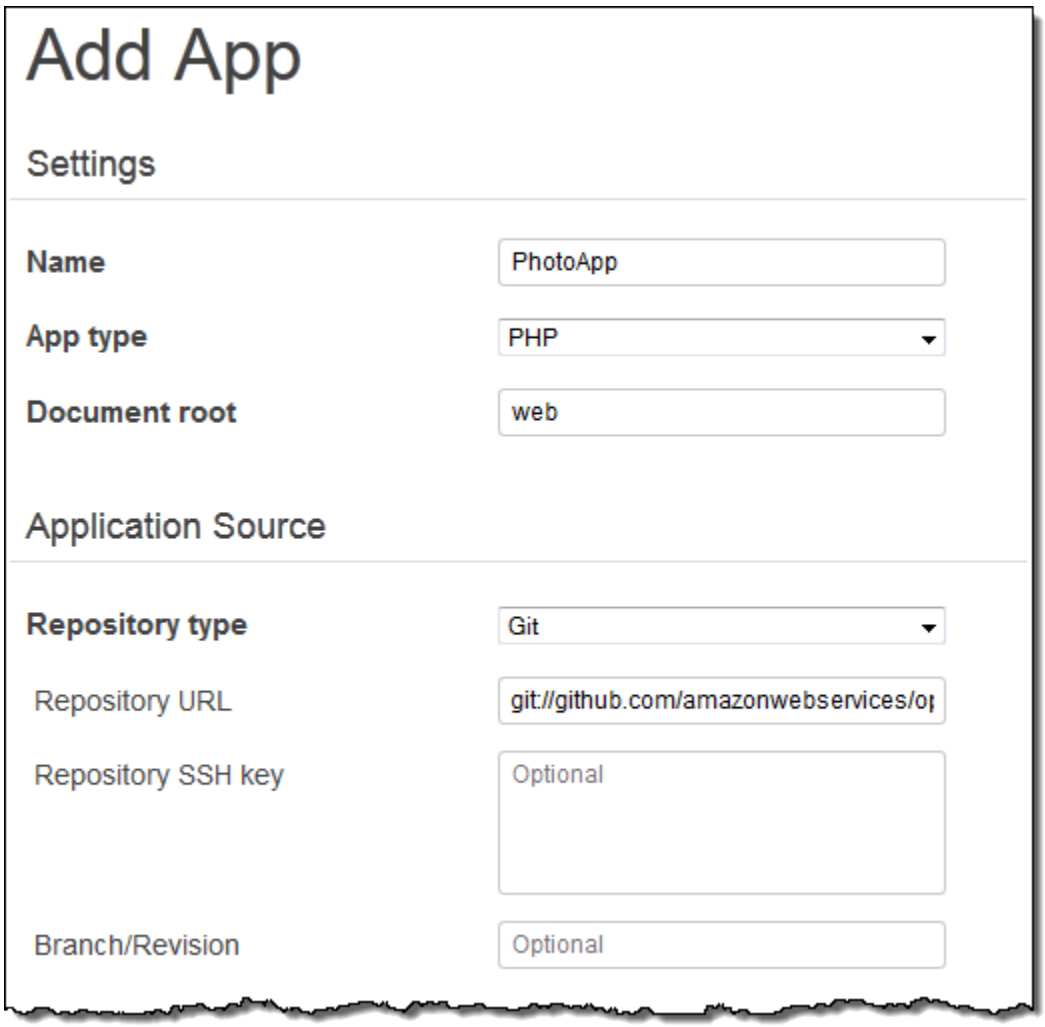
步骤 6：部署并运行 PhotoApp

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

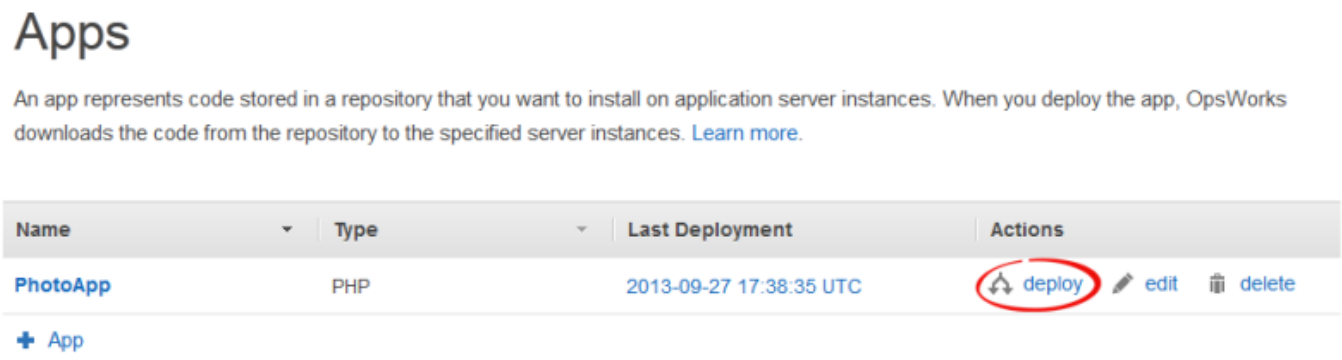
在本示例中，该应用程序也已为您实现并存储在公共存储 GitHub 库中。您只需将应用程序添加到堆栈，将其部署到应用程序服务器，然后运行它。

将应用程序添加到堆栈并将其部署到应用程序服务器

1. 打开 Apps (应用程序) 页面，然后选择 Add an app (添加应用程序)。
2. 在 Add App 页面上，执行以下操作：
   - 将名称设置为 **PhotoApp**。
   - 将 App type 设置为 PHP。
   - 将 Document root (文档根目录) 设置为 **web**。
   - 将 Repository type 设置为 Git。
   - 将 Repository URL (存储库 URL) 设置为 **git://github.com/awslabs/opsworks-demo-php-photo-share-app.git**。
   - 选择 Add App (添加应用程序) 以接受其他设置的默认值。

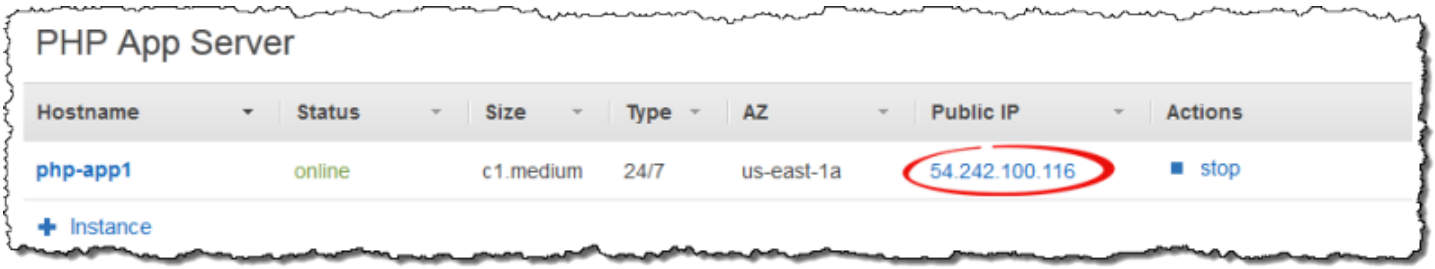


3. 在应用程序页面上，在 PhotoApp 应用程序的操作列中选择部署。



4. 接受默认值，然后选择 Deploy (部署) 以将应用程序部署到服务器。

要运行 PhotoApp，请转到实例页面并选择 PHP App Server 实例的公有 IP 地址。



您应看到以下用户界面。选择 添加照片以在 Amazon S3 存储桶上存储照片，并在后端数据存储中存储元数据。

AWS CodePipeline 与 AWS OpsWorks 堆栈一起使用

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

AWS CodePipeline允许您创建持续交付管道，以跟踪来自诸如 CodeCommit亚马逊简单存储服务 (Amazon S3) Simple Service 或之类的源代码更改。GitHub你可以使用 CodePipeline 在 Chef 11.10、Chef 12 和 Chef 12.2 AWS OpsWorks 堆栈上自动将 Chef 食谱和应用程序代码发布到 Stacks。本节中的示例介绍如何创建和使用其中的简单管道 CodePipeline 作为在 AWS OpsWorks Stacks 层上运行的代码的部署工具。

> **Note**
>
> CodePipeline 而且部署到 Chef 11.4 及更早版本的 AWS OpsWorks 堆栈时不支持堆栈集成。

主题

- AWS CodePipeline 带 AWS OpsWorks 堆栈-Chef 12 个堆栈
- AWS CodePipeline 带 AWS OpsWorks 堆栈-Chef 11 Stacks

AWS CodePipeline 带 AWS OpsWorks 堆栈-Chef 12 个堆栈

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

AWS CodePipeline允许您创建持续交付管道，以跟踪来自诸如 CodeCommit亚马逊简单存储服务 (Amazon S3) Simple Service 或之类的源代码更改。GitHub本主题中的示例介绍如何创建和使用其中的简单管道 CodePipeline 作为在 AWS OpsWorks Stacks 层上运行的代码的部署工具。在此示例中，您为一个简单的 Node.js 应用程序创建了一个管道，然后指示 AWS OpsWorks Stacks 在 Chef 12 堆栈中某个层的所有实例（在本例中为单个实例）上运行该应用程序。

> **Note**
>
> 本主题介绍了如何使用管道来运行和更新 Chef 12 堆栈中的应用程序。有关如何使用管道来运行和更新 Chef 11.10 堆栈中的应用程序的信息，请参阅 AWS CodePipeline 带 AWS OpsWorks 堆栈-Chef 11 Stacks。发送到 Amazon S3 存储桶的内容可能包含客户内容。有关删除敏感数据的更多信息，请参阅如何清空 S3 存储桶？或如何删除 S3 存储桶？。

主题

- 步骤 7（可选）：更新应用程序代码以查看自动 CodePipeline 重新部署您的应用程序
- 步骤 8（可选）：清除资源

先决条件

在开始本演练之前，请确保您具有管理员权限来执行以下所有任务。您可以是已应用AdministratorAccess策略的群组的成员，也可以是拥有下表所示权限和策略的群组的成员。作为最佳安全做法，您应加入有权执行以下任务的组，而不是为各个用户分配所需权限。

有关在 IAM 中创建安全组并为该组分配权限的更多信息，请参阅创建 IAM 用户组。有关管理 AWS OpsWorks 堆栈权限的更多信息，请参阅最佳实践：管理权限。

权限	推荐的附加到组的策略
在堆栈中创建和编辑堆栈、图层和实 AWS OpsWorks 例。	AWSOpsWorks_FullAccess
在 AWS CloudFormation中创建、编辑和运行模板。	AmazonCloudFormationFullAccess
创建、编辑和访问 Amazon S3 存储桶。	亚马逊 3 FullAccess
在中创建、编辑和运行管道 CodePipeline，尤其是使用 AWS OpsWorks Stacks 作为提供者的管道。	AWSCodePipeline_FullAccess

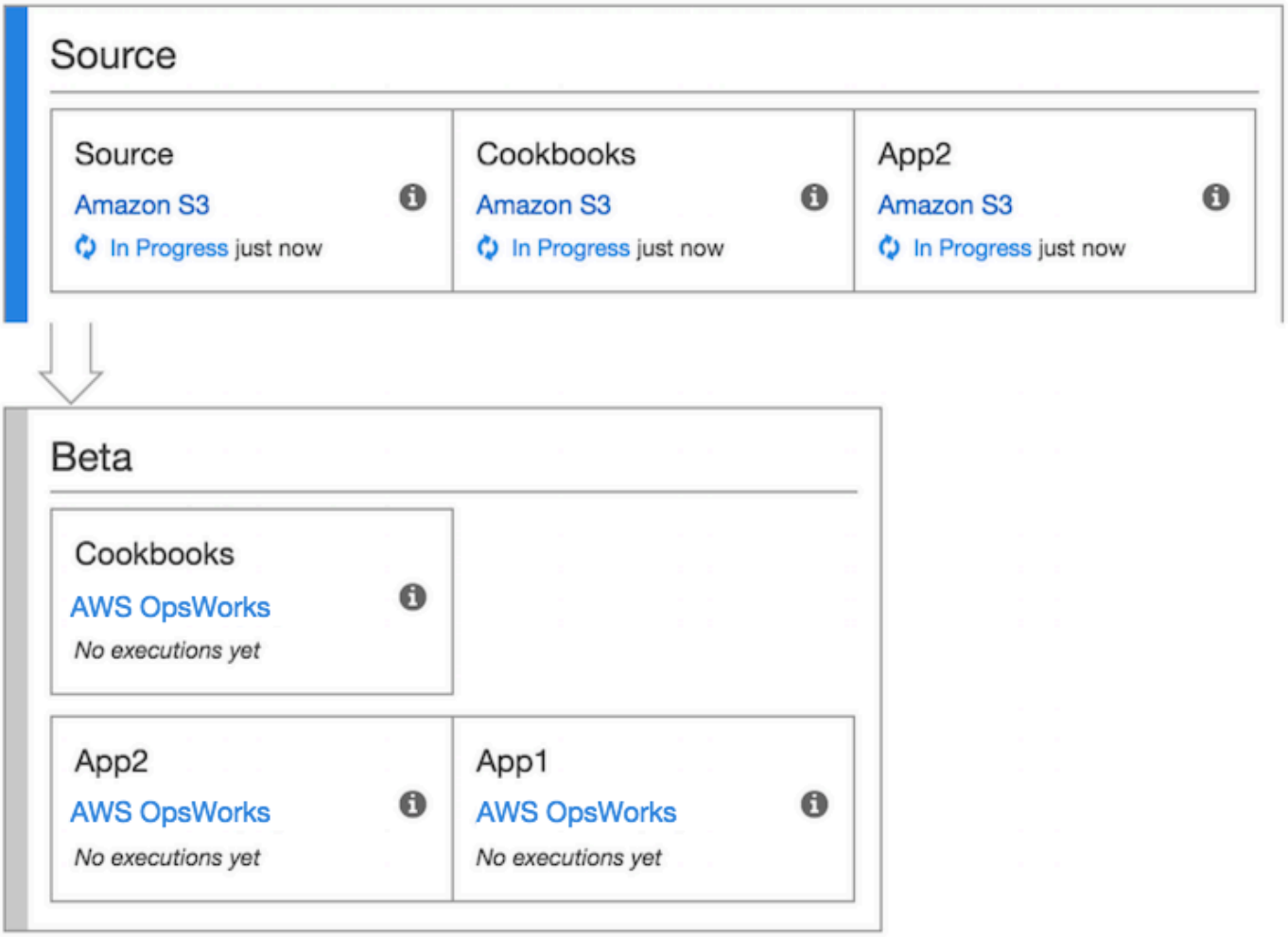
您还必须拥有 Amazon EC2 密钥对。在本演练中运行创建示例堆栈、层和实例的 AWS CloudFormation 模板时，系统将提示您提供此 key pair 的名称。有关在 Amazon EC2 控制台中获取密钥对的更多信息，请参阅 Amazon EC2 文档中的创建密钥对。主题必须位于美国东部（弗吉尼亚州北部）区域。如果您在该区域已有一个密钥对，则可使用现有密钥对。

其他受支持的场景

本演练会创建包含一个 Source 和一个 Deploy 阶段的简单管道。但是，您可以创建使用 AWS OpsWorks Stacks 作为提供者的更复杂的管道。以下是受支持的管道和场景的示例：

- 您可以编辑管道，以将 Chef 说明书添加到 Source 阶段，以及将已更新说明书的关联目标添加到 Deploy 阶段。在这种情况下，您添加一项 Deploy 操作，以便在更改源时触发对说明书的更新。更新的说明书将在应用程序前面部署。

- 您可以创建包含自定义食谱和多个应用程序的复杂管道，然后部署到 AWS OpsWorks 堆栈堆栈。该管道可跟踪对应用程序和说明书源的更改，并在您做出更改后重新部署。下面提供了一个类似的复杂管道的示例：



有关使用的更多信息 CodePipeline，请参阅《CodePipeline 用户指南》。

步骤 1：在堆栈中创建堆栈、层和实 AWS OpsWorks 例

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

要使用 AWS OpsWorks Stacks 作为管道的部署提供商，必须首先在层中拥有堆栈、层和至少一个实例。尽管您可以按照 Linux 堆栈入门或 Windows 堆栈入门中的说明在堆栈中创建堆栈，但为了节省时间，本示例使用 AWS CloudFormation 模板创建基于 Linux 的 Chef 12 堆栈、层和实例。 AWS OpsWorks 通过本模板创建的实例运行 Amazon Linux 2016.03，实例类型为 `c3.large`。虽然模板不会配置您的堆栈以便使用自定义说明书，但您将在稍后的演练中执行此操作。

> **Important**
>
> AWS CloudFormation 模板的存储和运行必须与您稍后将应用程序上传到的 Amazon S3 存储桶所在的区域以及您稍后在其中创建管道的区域相同 CodePipeline。目前，仅 CodePipeline 支持美国东部（弗吉尼亚北部）区域 (us-ea AWS OpsWorks st-1) 的堆栈提供商。本演练中的全部资源都应在美国东部（弗吉尼亚州北部）区域中创建。
>
> 如果堆栈创建失败，您可能即将达到您账户允许的最大 IAM 角色数。如果您的账户无法启动实例类型为 `c3.large` 的实例，堆栈创建也可能会失败。例如，如果您使用的是 AWS Free Tier，您可能会收到一条错误，如 `Root device type: must be included in EBS`。如果您的账户对允许您创建的实例类型有限制，例如 AWS 免费套餐施加的限制，请尝试将模板实例块中的`InstanceType`参数值更改为您的账户可以使用的实例类型。

使用创建堆栈、层和实例 AWS CloudFormation

1. 将以下 AWS CloudFormation 模板复制到新的纯文本文档中。将文件保存到本地计算机上方便的位置，然后将其命名为 NewOpsWorksStack.templat e 或其他便于使用的名称。

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Mappings": {
    "Region2Principal": {
      "us-east-1": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "us-west-2": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "us-west-1": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
```

```
      "eu-west-1": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "ap-southeast-1": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "ap-northeast-1": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "ap-northeast-2": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "ap-southeast-2": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "sa-east-1": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "cn-north-1": {
        "EC2Principal": "ec2.amazonaws.com.cn",
        "OpsWorksPrincipal": "opsworks.amazonaws.com.cn"
      },
      "eu-central-1": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      }
    }
  },
  "Parameters": {
    "EC2KeyPairName": {
   "Type": "String",
   "Description": "The name of an existing EC2 key pair that lets you use SSH to
connect to the OpsWorks instance."
  }
  },
  "Resources": {
"CPOpsDeploySecGroup": {
   "Type": "AWS::EC2::SecurityGroup",
```

```
  "Properties": {
    "GroupDescription" : "Lets you manage OpsWorks instances to which you deploy
apps with CodePipeline"
  }
},
"CPOpsDeploySecGroupIngressHTTP": {
  "Type": "AWS::EC2::SecurityGroupIngress",
  "Properties" : {
    "IpProtocol" : "tcp",
       "FromPort" : "80",
       "ToPort" : "80",
       "CidrIp" : "0.0.0.0/0",
 "GroupId": {
   "Fn::GetAtt": [
     "CPOpsDeploySecGroup", "GroupId"
   ]
 }
     }
},
"CPOpsDeploySecGroupIngressSSH": {
  "Type": "AWS::EC2::SecurityGroupIngress",
  "Properties" : {
    "IpProtocol" : "tcp",
       "FromPort" : "22",
       "ToPort" : "22",
       "CidrIp" : "0.0.0.0/0",
 "GroupId": {
   "Fn::GetAtt": [
     "CPOpsDeploySecGroup", "GroupId"
   ]
 }
  }
},
"MyStack": {
     "Type": "AWS::OpsWorks::Stack",
     "Properties": {
       "Name": {
         "Ref": "AWS::StackName"
       },
       "ServiceRoleArn": {
         "Fn::GetAtt": [
           "OpsWorksServiceRole",
           "Arn"
         ]
```

```
      },
"ConfigurationManager" : { "Name": "Chef","Version": "12" },
"DefaultOs": "Amazon Linux 2016.03",
      "DefaultInstanceProfileArn": {
        "Fn::GetAtt": [
          "OpsWorksInstanceProfile",
          "Arn"
        ]
      },
"UseCustomCookbooks": "false"
    }
  },
  "MyLayer": {
    "Type": "AWS::OpsWorks::Layer",
    "Properties": {
      "StackId": {
        "Ref": "MyStack"
      },
      "Name": "Node.js App Server",
"Type": "custom",
      "Shortname": "app1",
"EnableAutoHealing": "true",
      "AutoAssignElasticIps": "false",
      "AutoAssignPublicIps": "true",
"CustomSecurityGroupIds": [
  {
    "Fn::GetAtt": [
            "CPOpsDeploySecGroup", "GroupId"
    ]
  }
 ]
    },
    "DependsOn": [
      "MyStack",
      "CPOpsDeploySecGroup"
    ]
  },
  "OpsWorksServiceRole": {
    "Type": "AWS::IAM::Role",
    "Properties": {
      "AssumeRolePolicyDocument": {
        "Statement": [
          {
            "Effect": "Allow",
```

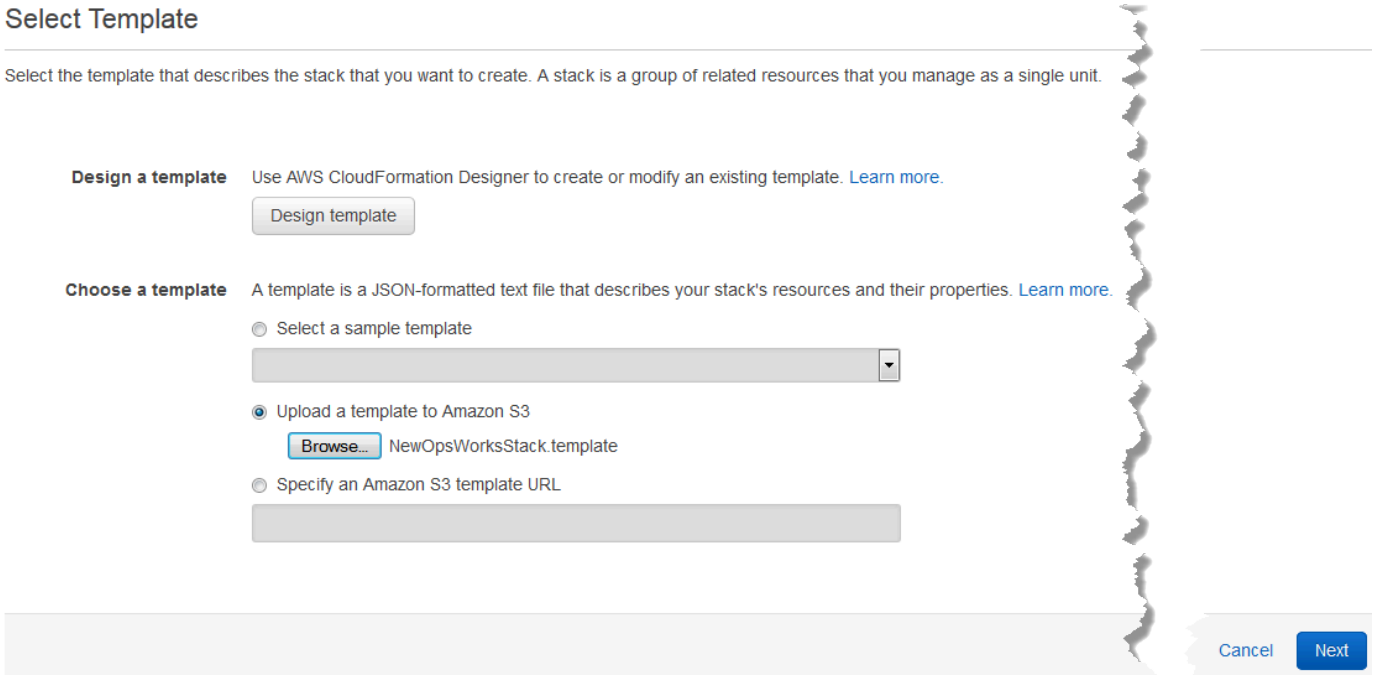
```
            "Principal": {
              "Service": [
                {
                  "Fn::FindInMap": [
                    "Region2Principal",
                    {
                      "Ref": "AWS::Region"
                    },
                    "OpsWorksPrincipal"
                  ]
                }
              ]
            },
            "Action": [
              "sts:AssumeRole"
            ]
          }
        ]
      },
      "Path": "/",
      "Policies": [
        {
          "PolicyName": "opsworks-service",
          "PolicyDocument": {
            "Statement": [
              {
                "Effect": "Allow",
                "Action": [
                  "ec2:*",
                  "iam:PassRole",
                  "cloudwatch:GetMetricStatistics",
                  "elasticloadbalancing:*"
                ],
                "Resource": "*"
              }
            ]
          }
        }
      ]
    }
  },
  "OpsWorksInstanceProfile": {
    "Type": "AWS::IAM::InstanceProfile",
    "Properties": {
```

```
        "Path": "/",
        "Roles": [
          {
            "Ref": "OpsWorksInstanceRole"
          }
        ]
      }
    },
    "OpsWorksInstanceRole": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "AssumeRolePolicyDocument": {
          "Statement": [
            {
              "Effect": "Allow",
              "Principal": {
                "Service": [
                  {
                    "Fn::FindInMap": [
                      "Region2Principal",
                      {
                        "Ref": "AWS::Region"
                      },
                      "EC2Principal"
                    ]
                  }
                ]
              },
              "Action": [
                "sts:AssumeRole"
              ]
            }
          ]
        },
        "Path": "/",
  "Policies": [
          {
            "PolicyName": "s3-get",
            "PolicyDocument": {
              "Version": "2012-10-17",
              "Statement": [
                {
                  "Effect": "Allow",
                  "Action": [
```

```
                    "s3:GetObject"
                  ],
                  "Resource": "*"
                }
              ]
            }
          }
        ]
      }
    },
    "myinstance": {
      "Type": "AWS::OpsWorks::Instance",
      "Properties": {
        "LayerIds": [
          {
            "Ref": "MyLayer"
          }
        ],
        "StackId": {
          "Ref": "MyStack"
        },
        "InstanceType": "c3.large",
        "SshKeyName": {
    "Ref": "EC2KeyPairName"
  }
      }
    }
  },
  "Outputs": {
    "StackId": {
      "Description": "Stack ID for the newly created AWS OpsWorks stack",
      "Value": {
        "Ref": "MyStack"
      }
    }
  }
}
```

2. 登录 AWS Management Console 并打开 AWS CloudFormation 控制台，网址为 https://console.aws.amazon.com/cloudformation。
3. 在 AWS CloudFormation 主页上，选择创建堆栈。
4. 在 Select Template 页面上的 Choose a template 区域中，选择 Upload a template to Amazon S3，然后选择 Browse。

5. 浏览到您在步骤 1 中保存的 AWS CloudFormation 模板，然后选择 “打开”。在 Select Template 页面上，选择 Next。

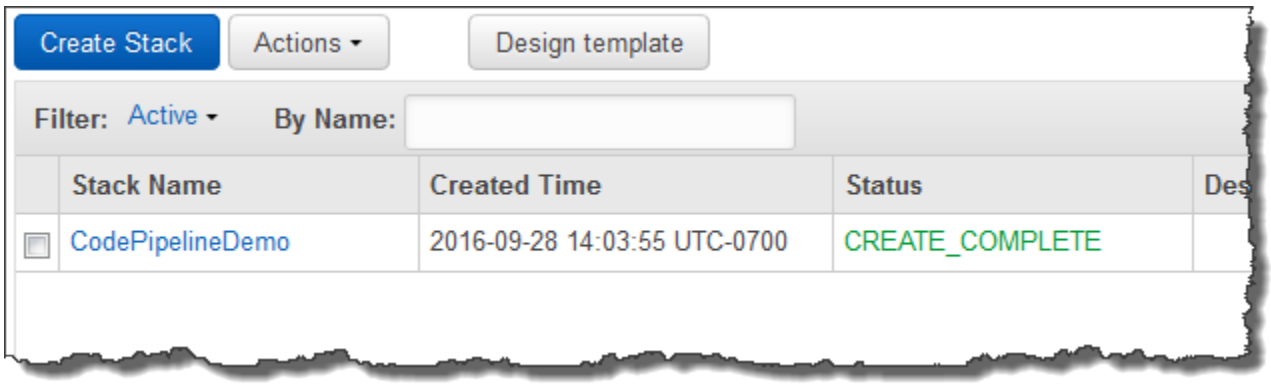


6. 在 “指定详细信息” 页面上，命名堆CodePipelineDemo栈或您的账户独有的任意堆栈名称。如果您选择了不同的堆栈名称，请更改整个演练中该堆栈的名称。
7. 在 “参数” 区域中，提供您要在创建 AWS OpsWorks 堆栈实例后用于访问堆栈实例的 EC2 密钥对的名称。选择下一步。
8. 在选项页面上，选择下一步。(此演练不必执行此页面上的设置。)
9. 您在本演练中使用的 AWS CloudFormation 模板创建 IAM 角色、实例配置文件和实例。

> **Important**
>
> 在选择 “创建” 之前，请选择 “成本” 以估算使用此模板创建资源可能产生的费用。 AWS

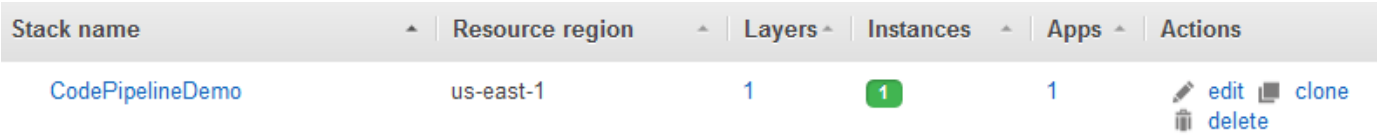
   如果创建 IAM 资源是可以接受的，请选中 “我确认此模板可能会 AWS CloudFormation 导致创建 IAM 资源” 复选框，然后选择 “创建”。如果创建 IAM 资源不可行，您将无法继续此过程。

10. 在 AWS CloudFormation 仪表板上，您可以查看堆栈的创建进度。请在 Status 列显示 CREATE_COMPLETE 之后再继续下一步。



验证堆栈中的 AWS OpsWorks 堆栈创建

1. 打开 AWS OpsWorks 控制台，网址为 https://console.aws.amazon.com/opsworks/。
2. 在 AWS OpsWorks 堆栈控制面板上，查看您创建的堆栈。



3. 打开堆栈，并查看层和实例。请注意，图层和实例是使用 AWS CloudFormation 模板中提供的名称和其他元数据创建的。您可以配置您的堆栈和层以使用自定义 Chef 说明书和配方。

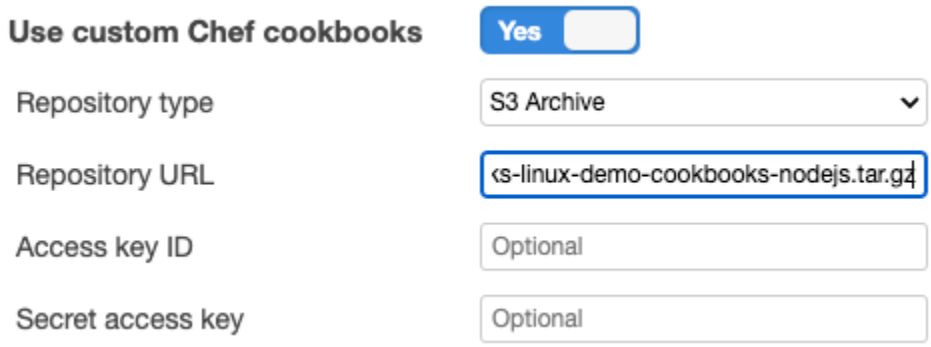
步骤 2：配置您的堆栈和层以使用自定义说明书

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

Stacks 中的 Chef 12 AWS OpsWorks 堆栈需要你自己或社区创建的食谱才能构建自定义应用程序层。在本演练中，您可以指向其中包含一组 Chef 说明书和 Chef 配方的存储库。这些配方在您的实例上安装 Node.js 程序包及其依赖项。您将使用其他 Chef 配方来部署将在 第 4 步：将您的应用程序添加到 AWS OpsWorks Stacks 中准备的 Node.js 应用程序。每次部署应用程序的新版本时，您在此步骤中指定的 Chef 配方都会运行 CodePipeline。

1. 在 AWS OpsWorks 堆栈控制台中，打开您在中步骤 1：在堆栈中创建堆栈、层和实 AWS OpsWorks 例创建的堆栈。选择 Stack Settings，然后选择 Edit。

2. 将 Use custom Chef cookbooks 设置为 Yes。这将显示相关自定义说明书设置。
3. 从 Repository type 下拉列表中选择 S3 Archive。要同时使用 CodePipeline 和 AWS OpsWorks，您的食谱来源必须是 S3。
4. 对于 Repository URL，请指定 **`https://s3.amazonaws.com/opsworks-demo-assets/opsworks-linux-demo-cookbooks-nodejs.tar.gz`**。您的设置与以下内容类似。



5. 选择保存。
6. 在导航窗格中，选择层。
7. 为您在中创建的层选择 Settings步骤 1：在堆栈中创建堆栈、层和实 AWS OpsWorks 例。
8. 在 General Settings 选项卡上，请确保层名称是 Node.js App Server，并且层短名称是 app1。选择 Recipes。
9. 在 Recipes (配方) 选项卡上，将 **`nodejs_demo`** 指定为想要在 Deploy (部署) 生命周期事件期间运行的配方。选择保存。
10. 在 “安全” 选项卡上，从 “安全组” 下拉列表中选择 AWS OpsWorks-Webapp 安全组。
11. 选择保存。

步骤 3：将应用程序代码上传到 Amazon S3 存储桶

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

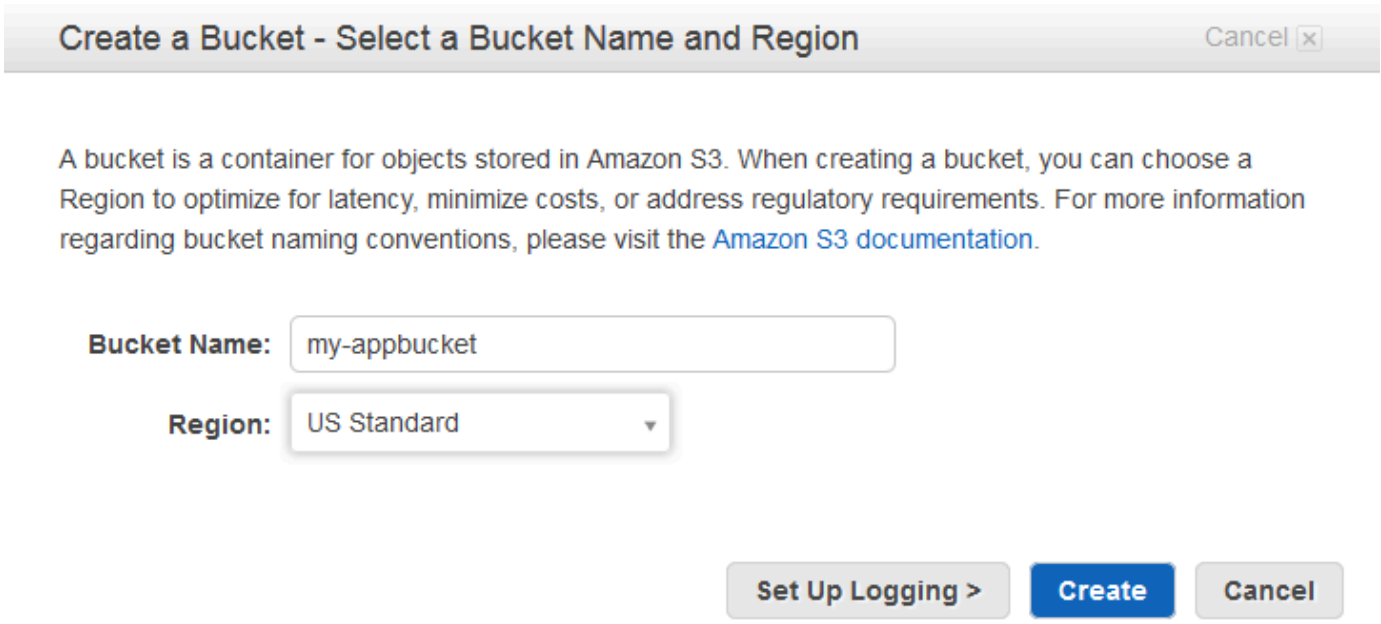
由于您必须提供指向您的代码存储库的链接作为管道设置的一部分，因此，在创建管道之前，应该先准备好代码存储库。在本演练中，您会将 Node.js 应用程序上传到 Amazon S3 存储桶。

尽管 CodePipeline 可以直接使用来自 GitHub 或 CodeCommit 作为源代码的代码，但本演练演示了如何使用 Amazon S3 存储桶。在本演练中，您将示例 Node.js 应用程序上传到您自己的 Amazon S3 存储桶，以便您可以对应用程序进行更改。您在此步骤中创建的 Amazon S3 存储桶可以 CodePipeline 检测应用程序代码的更改并自动部署更改后的应用程序。如果需要，您可以使用现有存储桶。请确保存储桶符合 CodePipeline 文档中简单管道演练（Amazon S3 存储桶）中描述的标准。

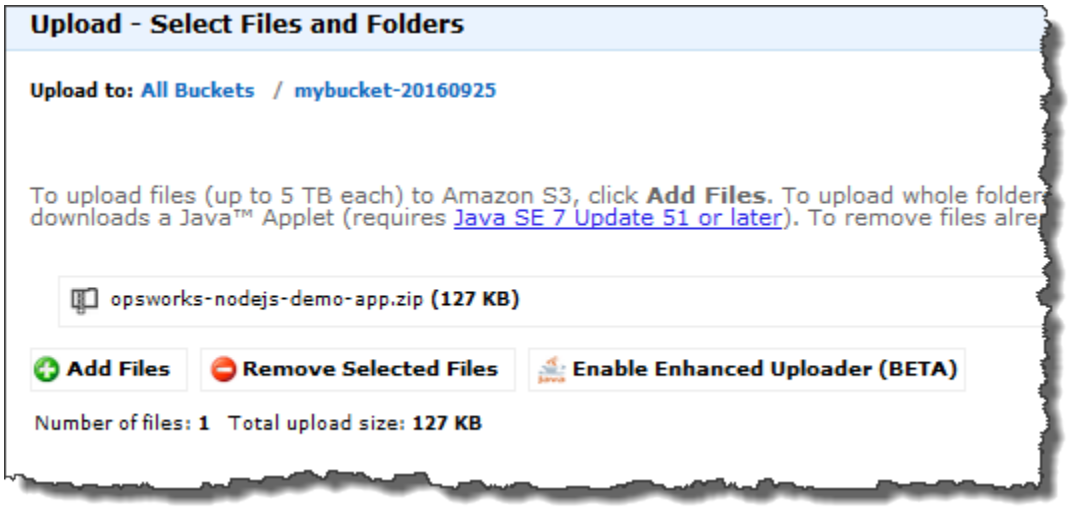
> **Important**
>
> Amazon S3 存储桶必须位于稍后将创建管道的同一区域。目前，仅 CodePipeline 支持美国东部（弗吉尼亚北部）区域 (us-ea AWS OpsWorks st-1) 的堆栈提供商。本演练中的全部资源都应在美国东部（弗吉尼亚州北部）区域中创建。由于 CodePipeline 需要版本控制源，因此还必须对存储桶进行版本控制。有关详细信息，请参阅使用版本控制。

要将应用程序上传到 Amazon S3 存储桶

1. 下载 AWS OpsWorks Stacks 示例的 ZIP 文件，即 Node.js 应用程序，然后将其保存到本地计算机上方便的位置。
2. 通过 https://console.aws.amazon.com/s3/ 打开 Amazon S3 控制台。
3. 选择创建存储桶。
4. 在 Create a Bucket - Select a Bucket Name and Region 页面上，对于 Bucket Name，键入存储桶的唯一名称。存储桶名称在所有 AWS 账户中必须是唯一的，而不仅仅是在您自己的账户中。本演练使用名称 **`my-appbucket`**，但您可以使用 `my-appbucket-`*`yearmonthday`* 让您的存储桶具有唯一的名称。从 Region 下拉列表中，选择 US Standard，然后选择 Create。US Standard 等同于 `us-east-1`。



5. 从 All Buckets (所有存储桶) 列表中选择您创建的存储桶。
6. 在存储桶页面上，选择 Upload (上传)。
7. 在 Upload - Select Files and Folders 页面上，选择 Add files。找到您在第 1 步中保存的 ZIP 文件，选择 Open (打开)，然后选择 Start Upload (开始上传)。



8. 上传完成后，从存储桶中的文件列表中选择 ZIP 文件，然后选择 Properties。
9. 在 Properties 窗格中，复制您的 ZIP 文件的链接，并记下此链接。您将需要存储桶名称和该链接的 ZIP 文件名部分来创建您的管道。

第 4 步：将您的应用程序添加到 AWS OpsWorks Stacks

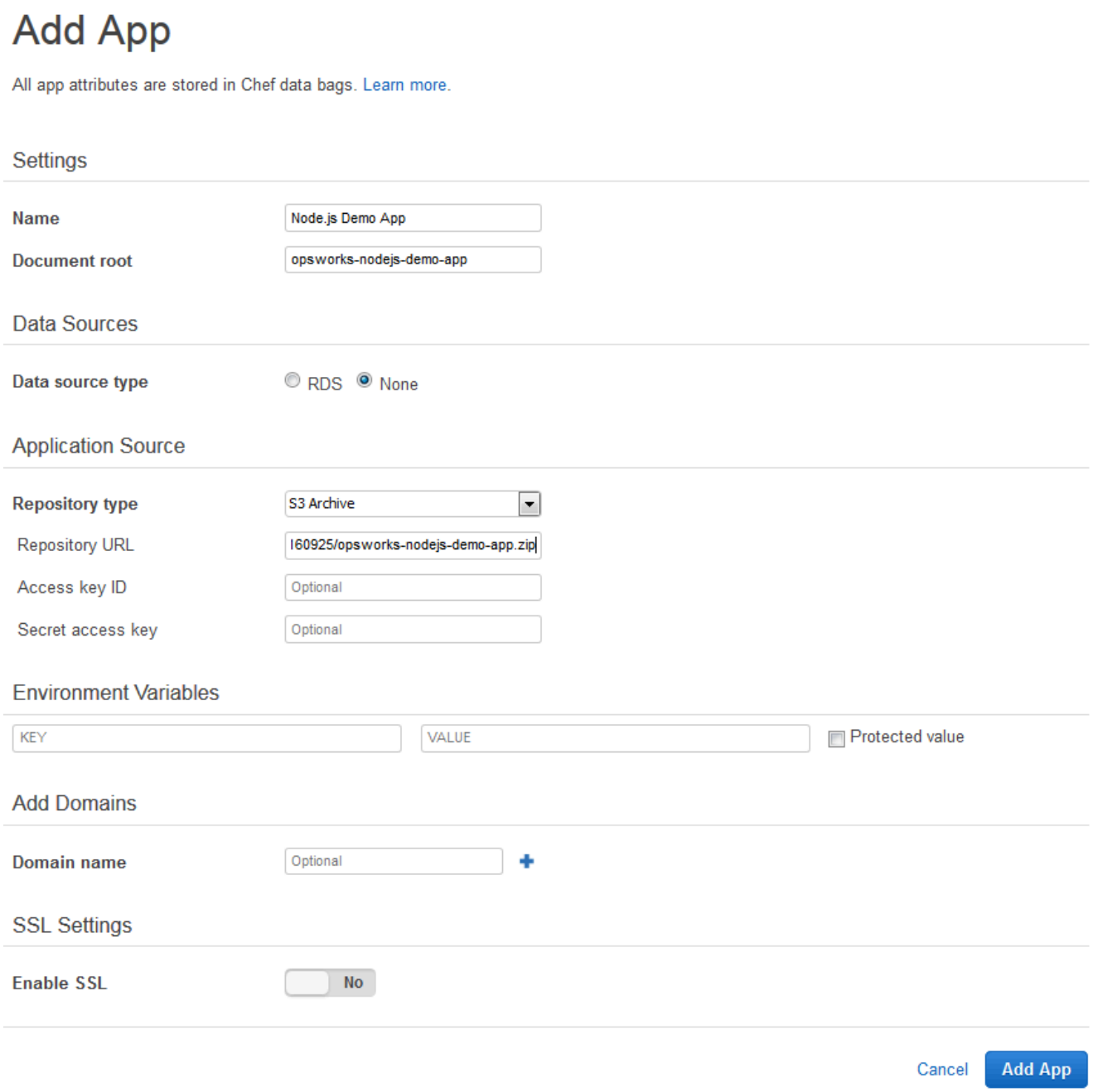
> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

在中创建管道之前 CodePipeline，请将 Node.js 测试应用程序添加到 AWS OpsWorks Stacks。创建管道时，需要选择已添加到 AWS OpsWorks Stacks 的应用程序。

准备好前面步骤 9 中的 Amazon S3 存储桶链接。您需要指向存储测试应用程序的存储桶的链接，才能完成此过程。

将应用程序添加到 AWS OpsWorks Stacks

1. 在 AWS OpsWorks Stacks 控制台中打开 CodePipelineDemo，然后在导航窗格中选择 Apps。
2. 选择 Add app (添加应用程序)。
3. 在 Add App (添加应用程序) 页面上，提供以下信息：
   a. 为应用程序指定名称。此演练使用名称 `Node.js Demo App`。
   b. 对于 Data source type (数据源类型)，选择 None (无)。此应用程序不需要外部数据库或数据源。
   c. 在 Repository type 下拉列表中选择 S3 Archive。
   d. 在 Repository URL 字符串框中，粘贴在步骤 3：将应用程序代码上传到 Amazon S3 存储桶的步骤 9 中复制的 URL。您的表单应类似如下：



4. 您无需更改本表单中的任何其他设置。选择 Add App。
5. 当 Node.js Demo App 应用程序出现在 Apps 页面上的列表中时，继续执行下一个过程，即[步骤 5：在中创建管道 CodePipeline]。

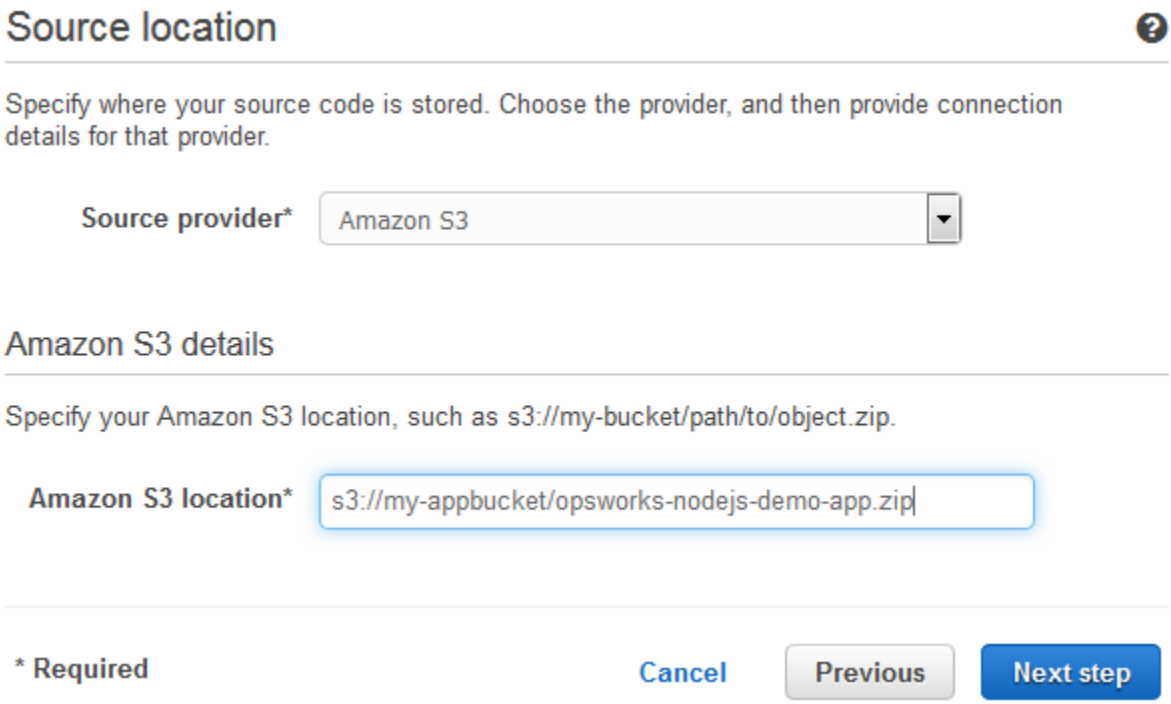
步骤 5：在中创建管道 CodePipeline

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

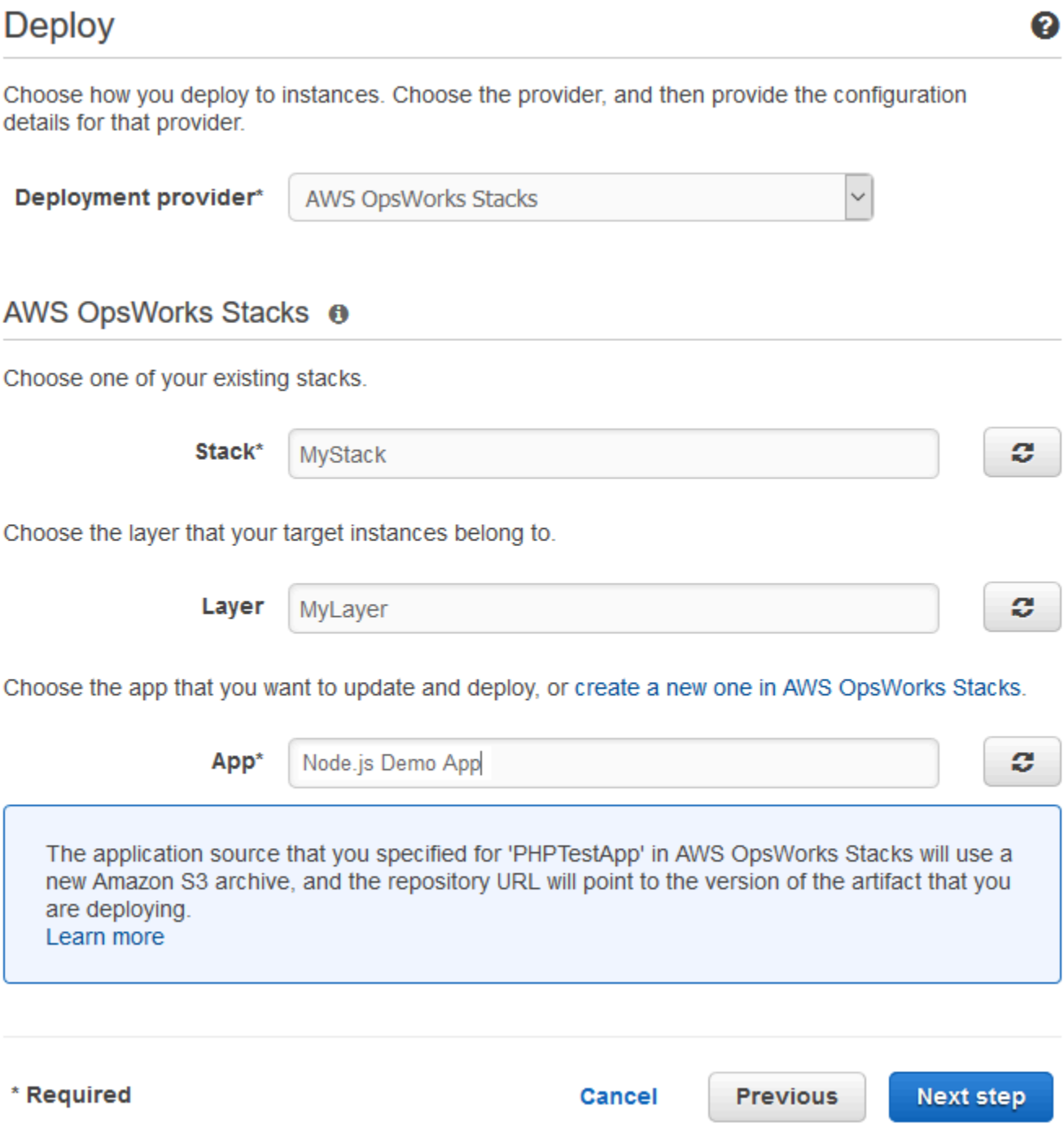
在你有一个包含层的堆栈并在堆栈中配置了至少一个实例之后，创建一个 CodePipeline 以 AWS OpsWorks AWS OpsWorks Stacks 作为提供者的管道，将应用程序或 Chef 食谱部署到你 AWS OpsWorks 的 Stacks 资源中。

要创建管道

1. 打开 CodePipeline 控制台，网址为 https://console.aws.amazon.com/codepipeline/。
2. 选择 Create pipeline（创建管道）。
3. 在 “入门 CodePipeline” 页面上**MyOpsWorksPipeline**，键入您的账户独有的管道名称或任何其他管道名称，然后选择 Ne xt step。
4. 在 Source Location 页面上，从 Source provider 下拉列表中选择 Amazon S3。
5. 在 Amazon S3 详细信息 页面中，键入格式为 **s3://*bucket-name*/*file name*** 的 Amazon S3 存储桶路径。请参阅您在 步骤 3：将应用程序代码上传到 Amazon S3 存储桶 的步骤 9 中记录的链接。在本演练中，路径为 `s3://my-appbucket/opsworks-nodejs-demo-app.zip`。选择下一步。



6. 在 Build 页面上，从下拉列表中选择 No Build，然后选择 Next step。
7. 在 Deploy (部署) 页面上，选择 AWS OpsWorks Stacks 作为部署提供程序。



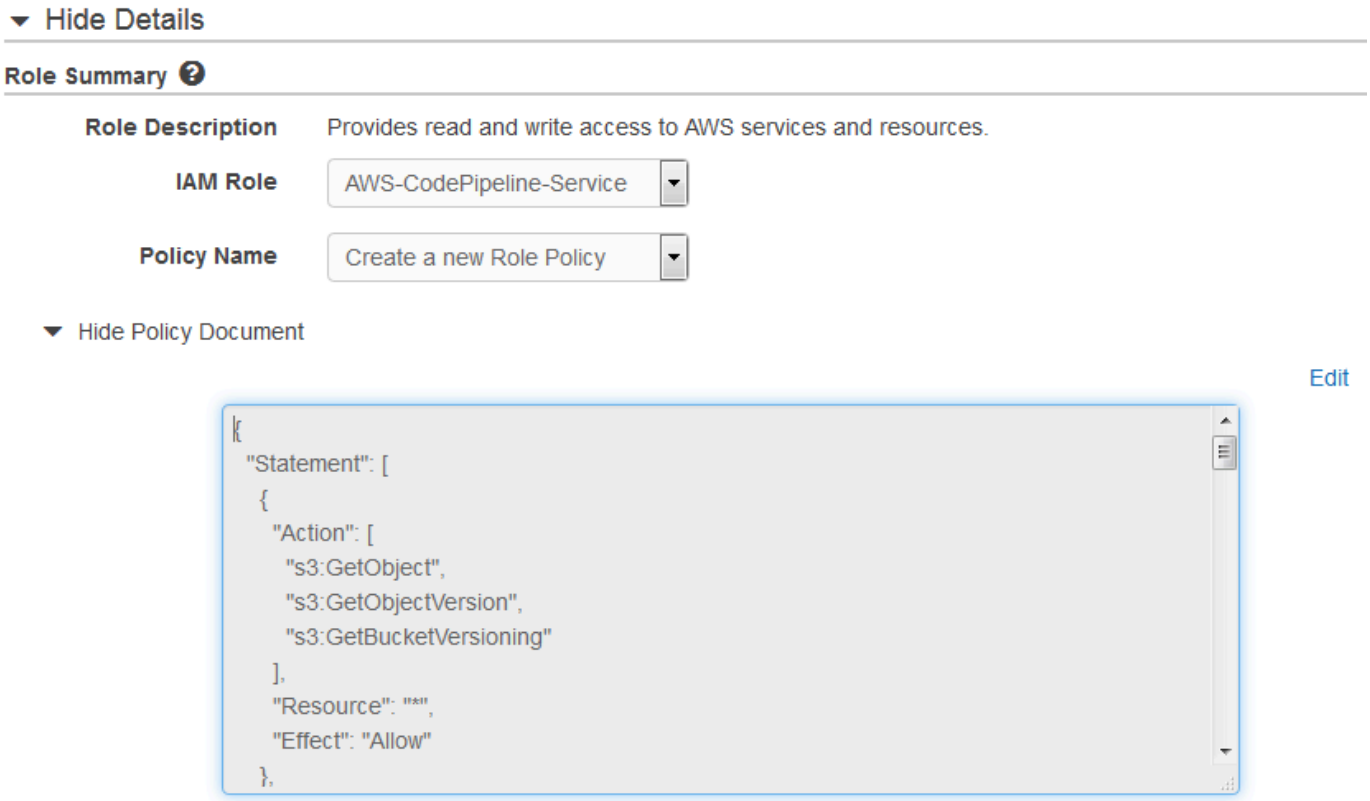
8. 在 Stack 字段中，键入 `CodePipelineDemo` 或者在步骤 1：在堆栈中创建堆栈、层和实 AWS OpsWorks 例中创建的堆栈的名称。
9. 在 Layer 字段中，键入 `Node.js App Server` 或者在步骤 1：在堆栈中创建堆栈、层和实 AWS OpsWorks 例中创建的层的名称。

10. 在应用程序字段中，选择在 步骤 3：将应用程序代码上传到 Amazon S3 存储桶 中上传到 Amazon S3 的应用程序，然后选择 下一步。
11. 在 AWS 服务角色 页面上，选择创建角色。

    系统将打开一个新窗口，其中的 IAM 控制台页面将介绍将为您创建的角色 `AWS-CodePipeline-Service`。从 Policy name 下拉列表中，选择 Create new policy。请确保策略文档具有以下内容。选择 Edit，根据需要更改策略文档。

```
{
    "Statement": [
        {
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:GetBucketVersioning"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": "opsworks:*",
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

    完成对策略文档的更改后，选择 Allow (允许)。您的更改将显示在 IAM 控制台中。



> **Note**
>
> 如果角色创建失败，可能是因为您已经有一个名为 A WS CodePipeline-Servic e 的 IAM 角色。如果您在 2016 年 5 月之前一直在使用 AWS CodePipeline-Servic e 角色，则该角色可能没有使用 AWS OpsWorks 堆栈作为部署提供商的权限。在这种情况下，您必须更新此步骤中所示的策略声明。如果您看到错误消息，请回到此步骤的开始位置，然后选择 Use existing role (使用现有角色) 而非 Create role (创建角色)。如果您使用现有角色，该角色应随附一个包含此步骤中所示权限的策略。有关服务角色及其策略声明的更多信息，请参阅 Edit a Policy for an IAM Service Role。

12. 如果角色创建过程成功，IAM 页面将关闭，并且您将返回 AWS 服务角色页面。选择下一步。
13. 在 Review your pipeline 页面上，验证页面上显示的选择，然后选择 Create pipeline。
14. 当您的管道已准备就绪，该管道应开始查找您的源代码并自动将应用程序部署到您的堆栈。此过程可能耗时数分钟。

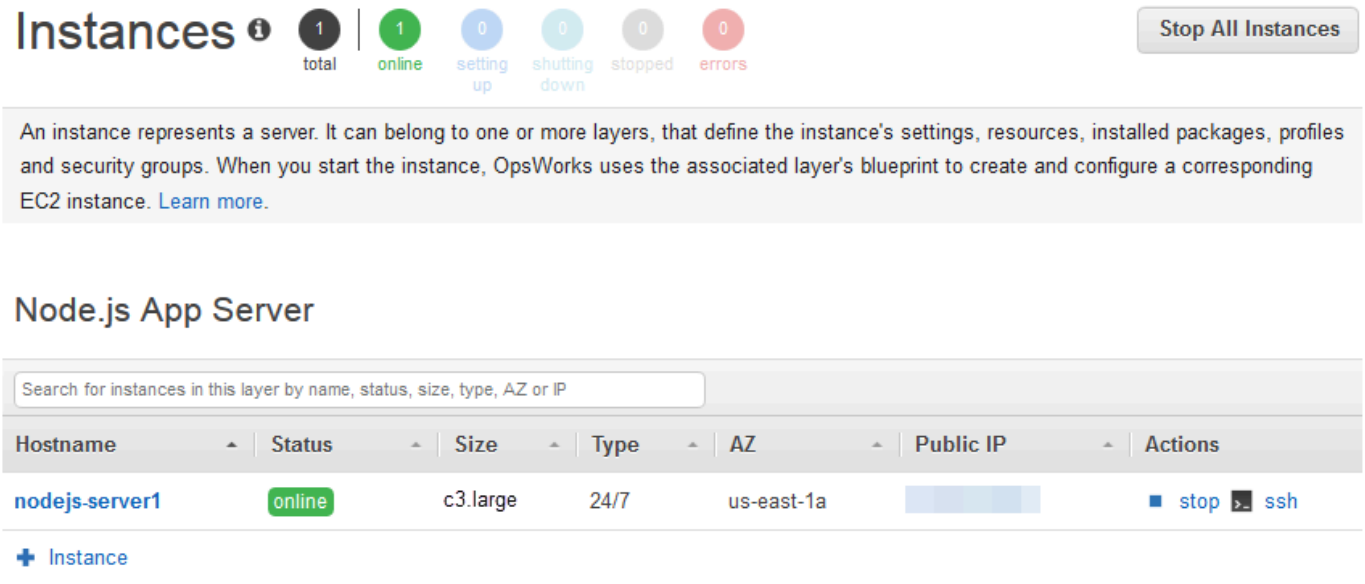
步骤 6：在 AWS OpsWorks Stacks 中验证应用程序部署

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

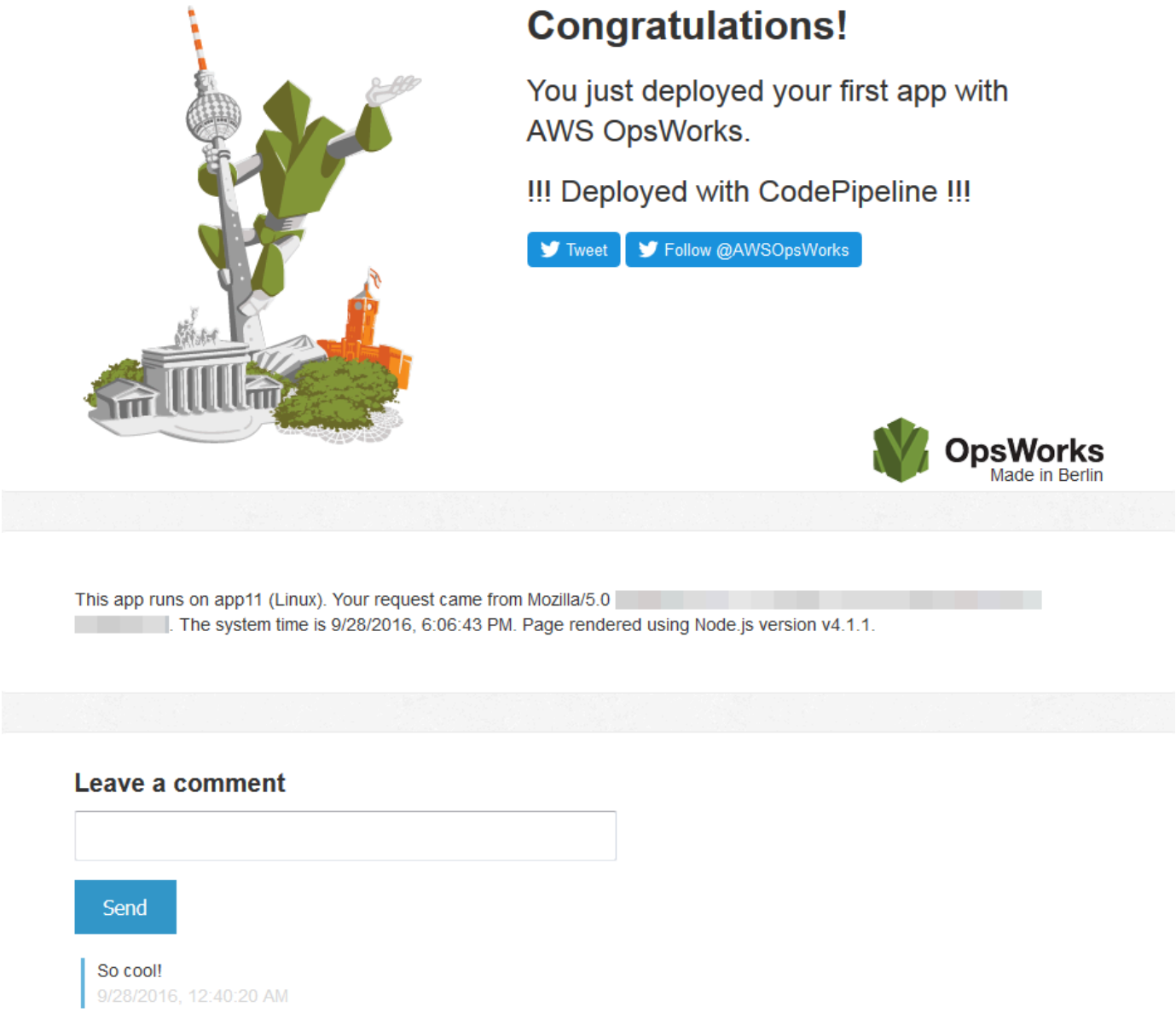
要验证已将 Node.js 应用程序 CodePipeline 部署到您的堆栈，请登录到您在中创建的实例步骤 1：在堆栈中创建堆栈、层和实 AWS OpsWorks 例。您应该可以看到并使用 Node.js Web 应用程序。

验证您的 AWS OpsWorks Stacks 实例中的应用程序部署

1. 打开 AWS OpsWorks 控制台，网址为 https://console.aws.amazon.com/opsworks/。
2. 在 AWS OpsWorks Stacks 控制面板上，选择 CodePipelineDemo，然后选择 Node.js App Server。
3. 在导航窗格中选择 Instances，然后选择您创建的实例的公有 IP 地址以查看 Web 应用程序。



该应用程序将显示在新的浏览器选项卡中。



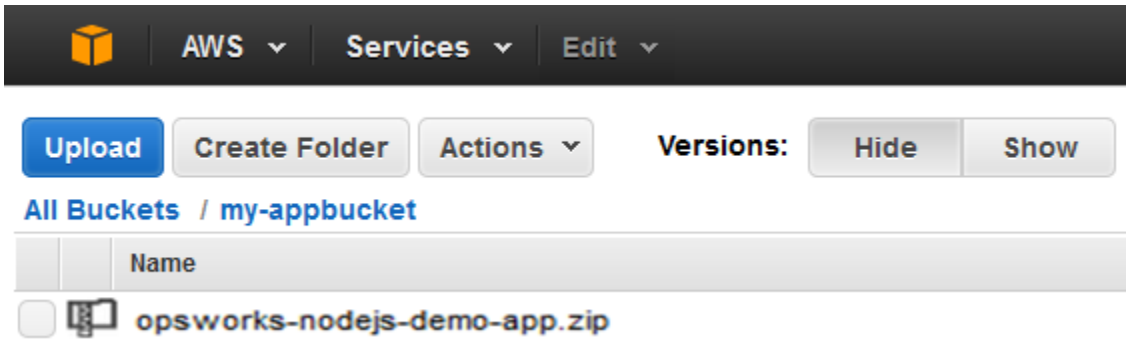
步骤 7（可选）：更新应用程序代码以查看自动 CodePipeline 重新部署您的应用程序

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

当您对使用部署的应用程序或食谱中的代码进行更改时 CodePipeline，更新的构件将自动部署 CodePipeline 到您的目标实例（在本例中为目标堆栈 AWS OpsWorks 堆栈）。本部分向您展示在更新示例 Node.js 应用程序中的代码时的自动重新部署工作。如果您仍在本地存储本演练的应用程序代码，并且在您开始演练后其他任何人都未更改代码，则可跳过此过程的步骤 1 至 4。

编辑示例应用程序中的代码

1. 登录 AWS Management Console 并打开 Amazon S3 控制台，网址为 https://console.aws.amazon.com/s3/。
2. 打开您在其中存储示例 Node.js 应用程序的存储桶。



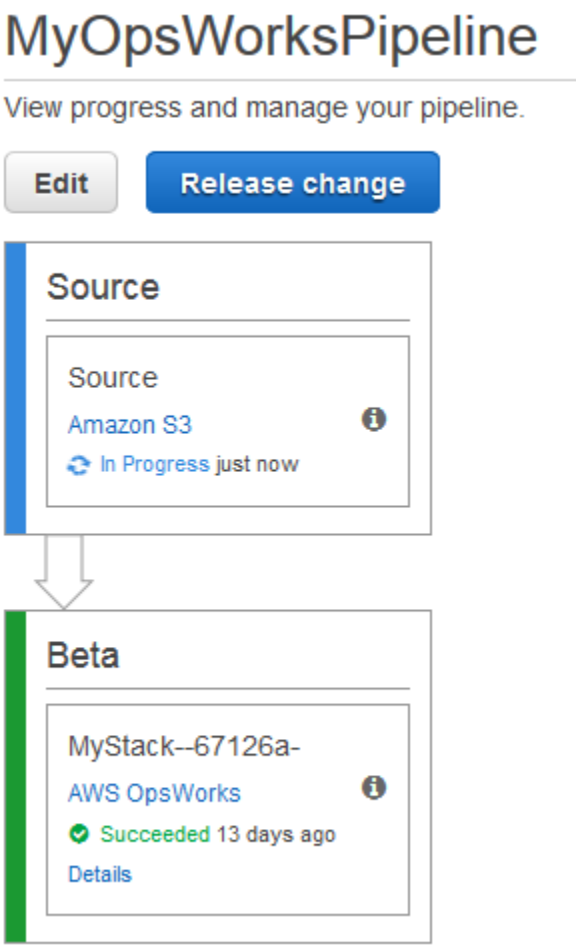
3. 选择包含该应用程序的 ZIP 文件。在 Actions 菜单上选择 Download。
4. 在对话框中，打开上下文 (右键单击) 菜单，选择 Download，然后将 ZIP 文件保存到方便位置。选择 确定。
5. 将 ZIP 文件的内容提取到方便的位置。您可能需要更改提取的文件夹及其子文件夹和内容的权限，才能进行编辑。在 `opsworks-nodejs-demo-app\views` 文件夹中，打开 `header.html` 文件进行编辑。
6. 搜索 `You just deployed your first app with` 这一短语。用 `updated` 一词替换 `deployed`。在下一行中，将 `AWS OpsWorks.` 更改为 `AWS OpsWorks and AWS CodePipeline.`。请勿编辑除文本外的任何内容。

```
<div id="main" role="main">LF
  <div class="container">LF
    <div class="hero-unit">LF
      <div class="robot">LF
        <h1>Congratulations!</h1>LF
        <h2>LF
          You just updated your first app with<br/>LF
          AWS OpsWorks and AWS CodePipeline.LF
        </h2>LF
```

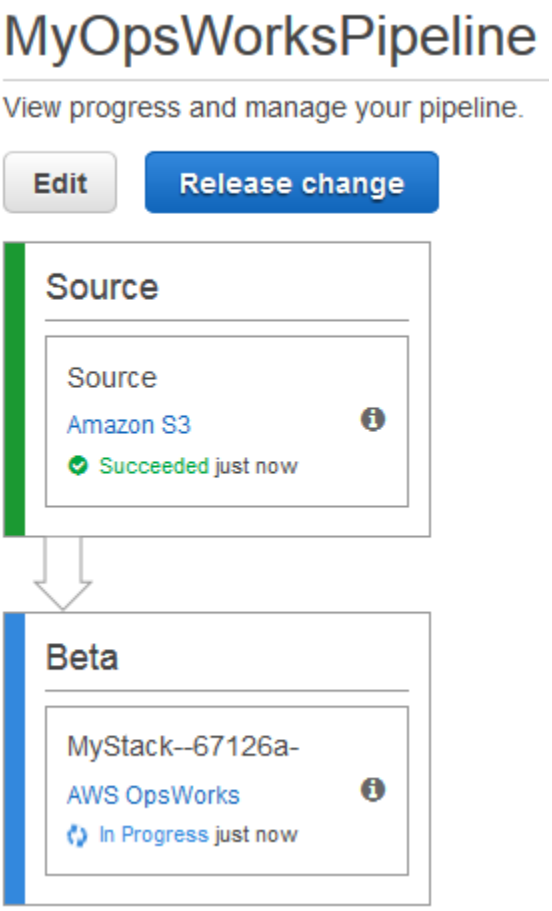
7. 保存并关闭 `header.html`文件。

8. 压缩 `opsworks-nodejs-demo-app` 文件夹，并将 ZIP 文件保存到方便位置。请勿更改 ZIP 文件的名称。
9. 将新的 ZIP 文件上传到 Amazon S3 存储桶。在本演练中，存储桶的名称为 `my-appbucket`。
10. 打开 CodePipeline 控制台，然后打开你的 AWS OpsWorks Stacks 管道 (MyOpsWorksPipeline)。选择 Release Change。

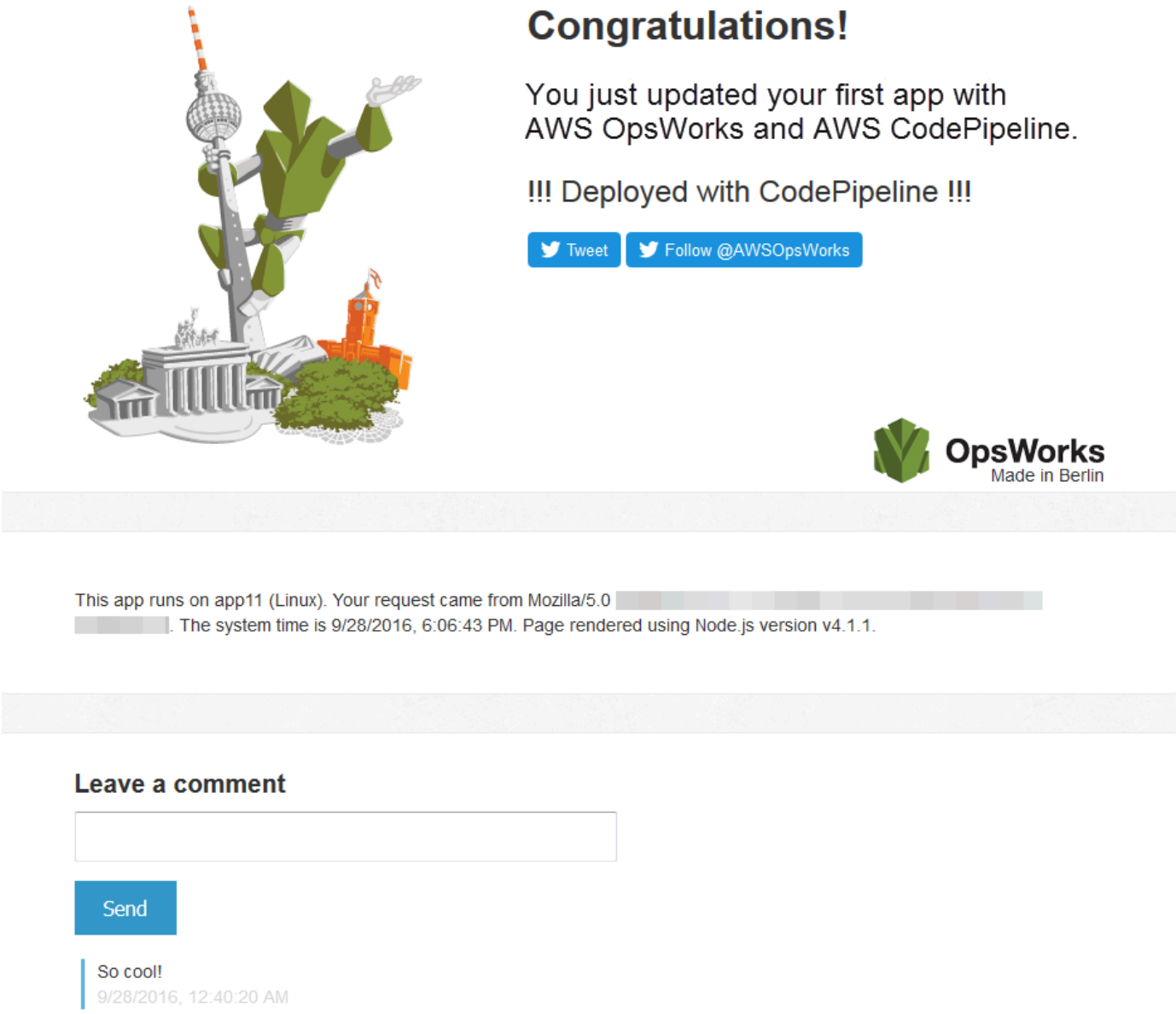
    （您可以等待在您的 Amazon S3 存储桶中检测 CodePipeline 到与应用程序更新版本相比的代码更改。 为了节省时间，本演练指导您只需选择 “发布更改” 即可。 ）
11. CodePipeline 在管道的各个阶段进行观察。首先， CodePipeline 检测对源构件的更改。



CodePipeline 将更新的代码推送到堆栈中的 AWS OpsWorks 堆栈。



12. 在成功完成两个阶段的管道后，请打开 AWS OpsWorks Stacks 中的堆栈。
13. 在堆栈属性页面上，选择 Instances。
14. 在 Public IP (公有 IP) 列中，选择实例的公有 IP 地址以查看更新后应用程序的文本。



步骤 8（可选）：清除资源

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

为了防止向您的 AWS 账户收取不必要的费用，您可以删除用于本演练的 AWS 资源。这些 AWS 资源包括堆栈 AWS OpsWorks 堆栈、IAM 角色和实例配置文件以及您在中 CodePipeline创建的管道。但是，随着您继续了解有关 AWS OpsWorks Stacks和 CodePipeline的更多信息，您可能需要继续使用这些 AWS 资源。如果您愿意，也可以在完成本演练后保留这些资源。

从堆栈中删除应用程序

由于您没有将应用程序作为 AWS CloudFormation 模板的一部分创建或应用，因此在中删除堆栈之前，请先删除 Node.js 测试应用程序 AWS CloudFormation。

1. 在 AWS OpsWorks Stacks 控制台的服务导航窗格中，选择应用程序。
2. 在 Apps 页面上选择 Node.js Demo App，然后在 Actions 中选择 delete。当系统提示您确认时，选择删除。 AWS OpsWorks Stacks 将删除该应用程序。

删除堆栈

由于堆栈是通过运行 AWS CloudFormation 模板创建的，因此可以在 AWS CloudFormation 控制台中删除堆栈，包括该模板创建的层、实例、实例配置文件和安全组。

1. 打开控制 AWS CloudFormation 台。
2. 在 AWS CloudFormation 控制台仪表板中，选择您创建的堆栈。在 Actions (操作) 菜单上，选择 Delete Stack (删除堆栈)。请在提示您进行确认时选择 Yes, Delete (是，删除)。
3. 等待 DELETE_COMPLETE 出现在堆栈的 Status (状态) 列中。

删除管道

1. 打开控制 CodePipeline 台。
2. 在 CodePipeline 仪表板中，选择您为此演练创建的管道。
3. 在管道页面上，选择 Edit。
4. 在 Edit 页面上，选择 Delete。请在提示您进行确认时选择删除。

AWS CodePipeline 带 AWS OpsWorks 堆栈-Chef 11 Stacks

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

AWS CodePipeline允许您创建持续交付管道，以跟踪来自诸如 CodeCommit亚马逊简单存储服务 (Amazon S3) Simple Service 或之类的源代码更改。GitHub本主题中的示例介绍如何创建和使用其中的简单管道 CodePipeline 作为在 AWS OpsWorks Stacks 层上运行的代码的部署工具。在此示例中，您为一个简单的 PHP 应用程序创建了一个管道，然后指示 AWS OpsWorks Stacks 在 Chef 11.10 堆栈中某个层的所有实例（在本例中为单个实例）上运行该应用程序。

> **Note**
>
> 本主题介绍了如何使用管道来运行和更新 Chef 11.10 堆栈中的应用程序。有关如何使用管道来运行和更新 Chef 12 堆栈中的应用程序的信息，请参阅 AWS CodePipeline 带 AWS OpsWorks 堆栈-Chef 12 个堆栈。发送到 Amazon S3 存储桶的内容可能包含客户内容。有关删除敏感数据的更多信息，请参阅如何清空 S3 存储桶？或如何删除 S3 存储桶？。

主题

先决条件

在开始本演练之前，请确保您具有管理员权限来执行以下所有任务。您可以是已应用AdministratorAccess策略的群组的成员，也可以是拥有下表所示权限和策略的群组的成员。作为最佳安全做法，您应加入有权执行以下任务的组，而不是为各个用户分配所需权限。

有关在 IAM 中创建安全组并为该组分配权限的更多信息，请参阅创建 IAM 用户组。有关管理 AWS OpsWorks 堆栈权限的更多信息，请参阅最佳实践：管理权限。

权限	推荐的附加到组的策略
在堆栈中创建和编辑堆栈、图层和实 AWS OpsWorks 例。	AWSOpsWorks_FullAccess
在 AWS CloudFormation中创建、编辑和运行模板。	AmazonCloudFormationFullAccess
创建、编辑和访问 Amazon S3 存储桶。	亚马逊 3 FullAccess
在中创建、编辑和运行管道 CodePipeline，尤其是使用 AWS OpsWorks Stacks 作为提供者的管道。	AWSCodePipeline_FullAccess

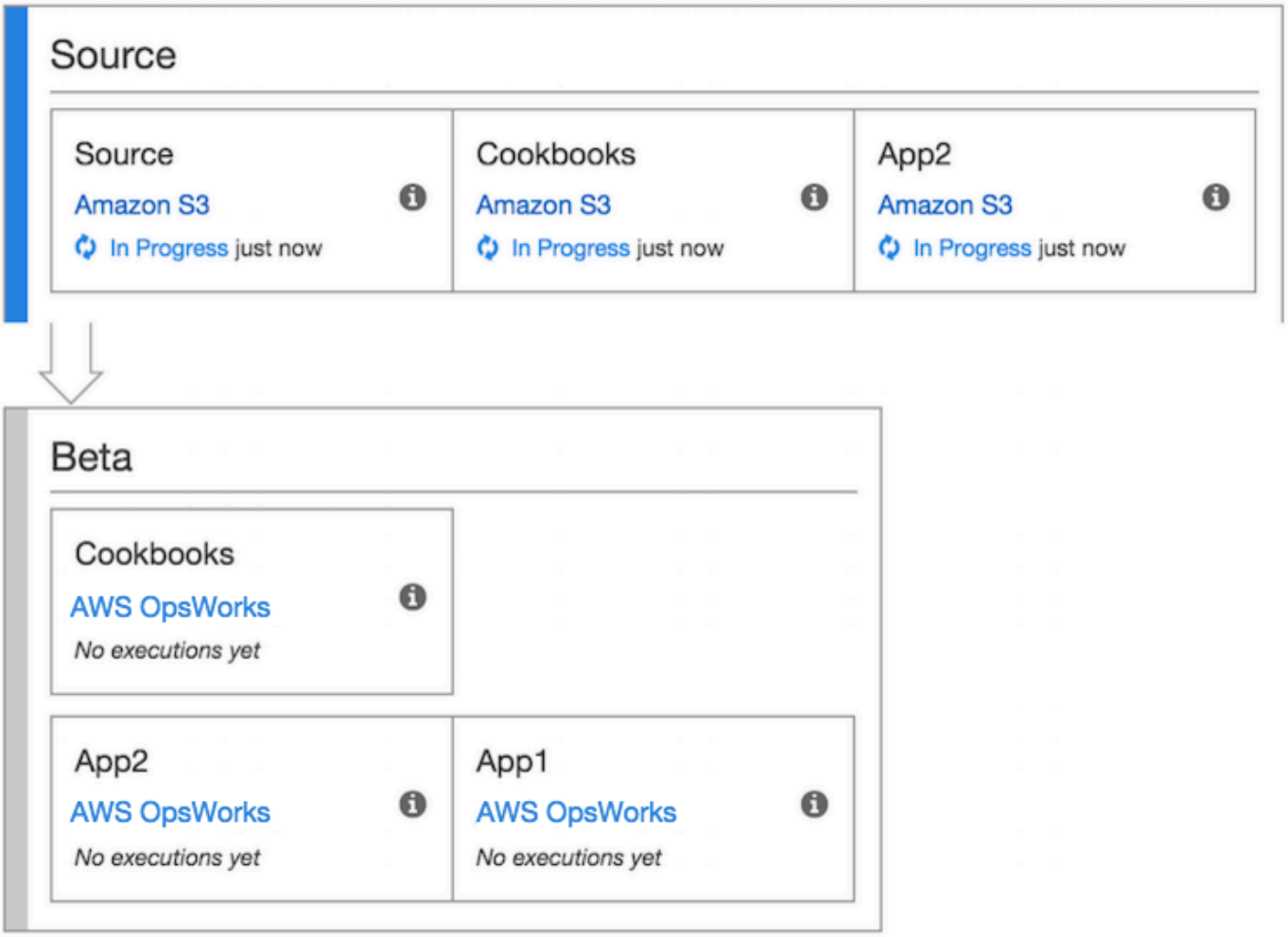
您还必须拥有 Amazon EC2 密钥对。在本演练中运行创建示例堆栈、层和实例的 AWS CloudFormation 模板时，系统将提示您提供此 key pair 的名称。有关在 Amazon EC2 控制台中获取密钥对的更多信息，请参阅 Amazon EC2 文档中的创建密钥对。密钥对应位于美国东部（弗吉尼亚北部）区域。如果您在该区域已有一个密钥对，则可使用现有密钥对。

其他受支持的场景

本演练会创建包含一个 Source 和一个 Deploy 阶段的简单管道。但是，您可以创建使用 AWS OpsWorks Stacks 作为提供者的更复杂的管道。以下是受支持的管道和场景的示例：

- 您可以编辑管道，以将 Chef 说明书添加到 Source 阶段，以及将已更新说明书的关联目标添加到 Deploy 阶段。在这种情况下，您添加一项 Deploy 操作，以便在更改源时触发对说明书的更新。更新的说明书将在应用程序前面部署。

- 您可以创建包含自定义食谱和多个应用程序的复杂管道，然后部署到 AWS OpsWorks Stacks 堆栈。该管道可跟踪对应用程序和说明书源的更改，并在您做出更改后重新部署。下面提供了一个类似的复杂管道的示例：



有关使用的更多信息 CodePipeline，请参阅CodePipeline文档。

步骤 1：在 AWS OpsWorks Stacks 中创建堆栈、层和实例

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

要使用 AWS OpsWorks Stacks 作为管道的部署提供商，必须首先在层中拥有堆栈、层和至少一个实例。尽管您可以按照 Linux 堆栈入门或 Windows 堆栈入门中的说明在堆栈中创建堆栈，但为了节省时间，此示例使用 AWS CloudFormation 模板创建基于 Linux 的 Chef 11.10 堆栈、层和实例。 AWS OpsWorks 通过本模板创建的实例运行 Amazon Linux 2016.03，实例类型为 `c3.large`。

> **Important**
>
> AWS CloudFormation 模板的存储和运行必须与您稍后将应用程序上传到的 Amazon S3 存储桶所在的区域以及您稍后在其中创建管道的区域相同 CodePipeline。目前，仅 CodePipeline 支持美国东部（弗吉尼亚北部）区域 (us-ea AWS OpsWorks st-1) 的堆栈提供商。本演练中的全部资源都应在美国东部（弗吉尼亚州北部）区域中创建。
> 如果堆栈创建失败，您可能即将达到您账户允许的最大 IAM 角色数。如果您的账户无法启动实例类型为 `c3.large` 的实例，堆栈创建也可能会失败。例如，如果您使用的是 AWS Free Tier，您可能会收到一条错误，如 `Root device type: must be included in EBS`。如果您的账户对允许您创建的实例类型有限制，例如 AWS 免费套餐施加的限制，请尝试将模板实例块中的`InstanceType`参数值更改为您的账户可以使用的实例类型。

使用创建堆栈、层和实例 AWS CloudFormation

1. 将以下 AWS CloudFormation 模板复制到新的纯文本文档中。将文件保存到本地计算机上方便的位置，然后将其命名为 NewOpsWorksStack.templat e 或其他便于使用的名称。

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Mappings": {
    "Region2Principal": {
      "us-east-1": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "us-west-2": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "us-west-1": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "eu-west-1": {
```

```
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "ap-southeast-1": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "ap-northeast-1": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "ap-northeast-2": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "ap-southeast-2": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "sa-east-1": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "cn-north-1": {
        "EC2Principal": "ec2.amazonaws.com.cn",
        "OpsWorksPrincipal": "opsworks.amazonaws.com.cn"
      },
      "eu-central-1": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      }
    }
 },
 "Parameters": {
   "EC2KeyPairName": {
  "Type": "String",
  "Description": "The name of an existing EC2 key pair that allows you to use SSH
to connect to the OpsWorks instance."
 }
 },
 "Resources": {
"CPOpsDeploySecGroup": {
  "Type": "AWS::EC2::SecurityGroup",
  "Properties": {
```

```
    "GroupDescription" : "Lets you manage OpsWorks instances deployed to by
CodePipeline"
  }
},
"CPOpsDeploySecGroupIngressHTTP": {
  "Type": "AWS::EC2::SecurityGroupIngress",
  "Properties" : {
    "IpProtocol" : "tcp",
       "FromPort" : "80",
       "ToPort" : "80",
       "CidrIp" : "0.0.0.0/0",
 "GroupId": {
   "Fn::GetAtt": [
     "CPOpsDeploySecGroup", "GroupId"
   ]
 }
    }
},
"CPOpsDeploySecGroupIngressSSH": {
  "Type": "AWS::EC2::SecurityGroupIngress",
  "Properties" : {
    "IpProtocol" : "tcp",
       "FromPort" : "22",
       "ToPort" : "22",
       "CidrIp" : "0.0.0.0/0",
 "GroupId": {
   "Fn::GetAtt": [
     "CPOpsDeploySecGroup", "GroupId"
   ]
 }
  }
},
"MyStack": {
     "Type": "AWS::OpsWorks::Stack",
     "Properties": {
       "Name": {
         "Ref": "AWS::StackName"
       },
       "ServiceRoleArn": {
         "Fn::GetAtt": [
           "OpsWorksServiceRole",
           "Arn"
         ]
       },
```

```
  "ConfigurationManager" : { "Name": "Chef","Version": "11.10" },
  "DefaultOs": "Amazon Linux 2016.03",
        "DefaultInstanceProfileArn": {
          "Fn::GetAtt": [
            "OpsWorksInstanceProfile",
            "Arn"
          ]
        }
      }
    },
    "MyLayer": {
      "Type": "AWS::OpsWorks::Layer",
      "Properties": {
        "StackId": {
          "Ref": "MyStack"
        },
        "Name": "MyLayer",
        "Type": "php-app",
  "Shortname": "mylayer",
        "EnableAutoHealing": "true",
        "AutoAssignElasticIps": "false",
        "AutoAssignPublicIps": "true",
  "CustomSecurityGroupIds": [
    {
      "Fn::GetAtt": [
              "CPOpsDeploySecGroup", "GroupId"
      ]
          }
        ]
      },
      "DependsOn": [
        "MyStack",
        "CPOpsDeploySecGroup"
      ]
    },
    "OpsWorksServiceRole": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "AssumeRolePolicyDocument": {
          "Statement": [
            {
              "Effect": "Allow",
              "Principal": {
                "Service": [
```

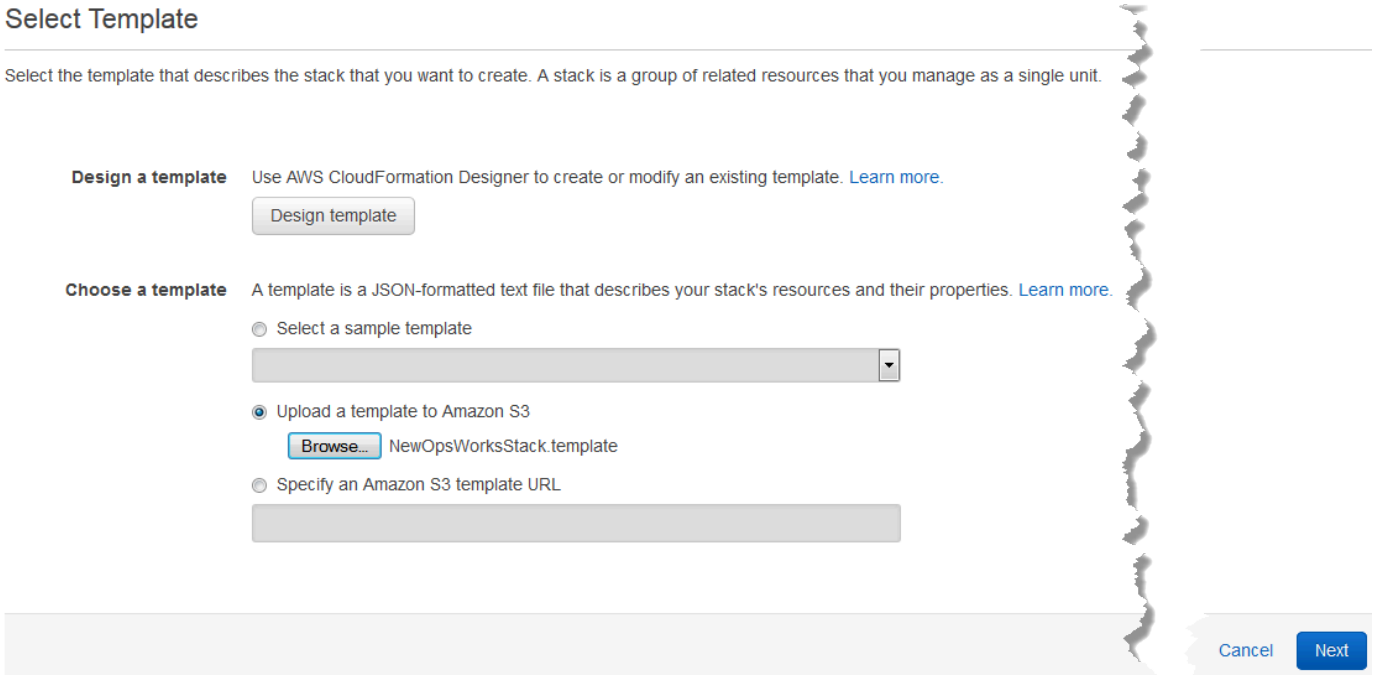
```
                {
                  "Fn::FindInMap": [
                    "Region2Principal",
                    {
                      "Ref": "AWS::Region"
                    },
                    "OpsWorksPrincipal"
                  ]
                }
              ]
            },
            "Action": [
              "sts:AssumeRole"
            ]
          }
        ]
      },
      "Path": "/",
      "Policies": [
        {
          "PolicyName": "opsworks-service",
          "PolicyDocument": {
            "Statement": [
              {
                "Effect": "Allow",
                "Action": [
                  "ec2:*",
                  "iam:PassRole",
                  "cloudwatch:GetMetricStatistics",
                  "elasticloadbalancing:*"
                ],
                "Resource": "*"
              }
            ]
          }
        }
      ]
    }
  },
  "OpsWorksInstanceProfile": {
    "Type": "AWS::IAM::InstanceProfile",
    "Properties": {
      "Path": "/",
      "Roles": [
```

```
          {
            "Ref": "OpsWorksInstanceRole"
          }
        ]
      }
    },
    "OpsWorksInstanceRole": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "AssumeRolePolicyDocument": {
          "Statement": [
            {
              "Effect": "Allow",
              "Principal": {
                "Service": [
                  {
                    "Fn::FindInMap": [
                      "Region2Principal",
                      {
                        "Ref": "AWS::Region"
                      },
                      "EC2Principal"
                    ]
                  }
                ]
              },
              "Action": [
                "sts:AssumeRole"
              ]
            }
          ]
        },
        "Path": "/",
  "Policies": [
          {
            "PolicyName": "s3-get",
            "PolicyDocument": {
              "Version": "2012-10-17",
              "Statement": [
                {
                  "Effect": "Allow",
                  "Action": [
                    "s3:GetObject"
                  ],
```

```
                  "Resource": "*"
                }
              ]
            }
          }
        ]
      }
    },
    "myinstance": {
      "Type": "AWS::OpsWorks::Instance",
      "Properties": {
        "LayerIds": [
          {
            "Ref": "MyLayer"
          }
        ],
        "StackId": {
          "Ref": "MyStack"
        },
        "InstanceType": "c3.large",
        "SshKeyName": {
    "Ref": "EC2KeyPairName"
  }
      }
    }
  },
  "Outputs": {
    "StackId": {
      "Description": "Stack ID for the newly created AWS OpsWorks stack",
      "Value": {
        "Ref": "MyStack"
      }
    }
  }
}
```

2. 登录 AWS Management Console 并打开 AWS CloudFormation 控制台，网址为 https://console.aws.amazon.com/cloudformation。
3. 在 AWS CloudFormation 主页上，选择创建堆栈。
4. 在 Select Template 页面上的 Choose a template 区域中，选择 Upload a template to Amazon S3，然后选择 Browse。

5. 浏览到您在步骤 1 中保存的 AWS CloudFormation 模板，然后选择 “打开”。在 Select Template 页面上，选择 Next。

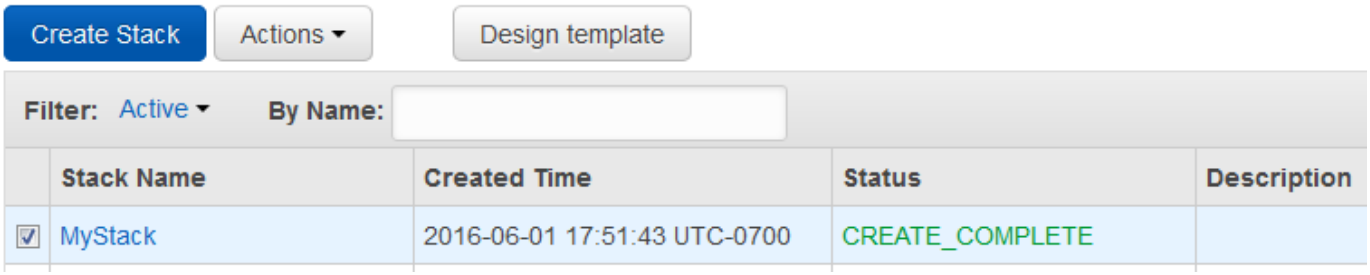


6. 在 “指定详细信息” 页面上，命名堆MyStack栈或您的账户独有的任意堆栈名称。如果您选择了不同的堆栈名称，请更改整个演练中该堆栈的名称。
7. 在 “参数” 区域中，提供您要在创建 AWS OpsWorks 堆栈实例后用于访问堆栈实例的 EC2 密钥对的名称。选择下一步。
8. 在选项页面上，选择下一步。(此演练不必执行此页面上的设置。)
9. 您在本演练中使用的 AWS CloudFormation 模板创建 IAM 角色、实例配置文件和实例。

> **Important**
>
> 在选择 “创建” 之前，请选择 “成本” 以估算使用此模板创建资源可能产生的费用。 AWS

   如果可以创建 IAM 资源，请选中 “我确认此模板可能会导致 AWS CloudFormation 创建 IAM 资源” 复选框，然后选择 “创建”。如果创建 IAM 资源不可行，您将无法继续此过程。

10. 在 AWS CloudFormation 仪表板上，您可以查看堆栈的创建进度。请在 Status 列显示 CREATE_COMPLETE 之后再继续下一步。



验证堆栈中的 AWS OpsWorks 堆栈创建

1. 打开 AWS OpsWorks 控制台，网址为 https://console.aws.amazon.com/opsworks/。
2. 在 AWS OpsWorks 堆栈控制面板上，查看您创建的堆栈。

3. 打开堆栈，并查看层和实例。请注意，图层和实例是使用 AWS CloudFormation 模板中提供的名称和其他元数据创建的。现在，您已可以将应用程序上传到 Amazon S3 存储桶。

步骤 2：将应用程序代码上传到 Amazon S3 存储桶

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

由于您必须提供指向您的代码存储库的链接作为管道设置的一部分，因此，在创建管道之前，应该先准备好代码存储库。在本演练中，您可将 PHP 应用程序上传到一个 Amazon S3 存储桶。

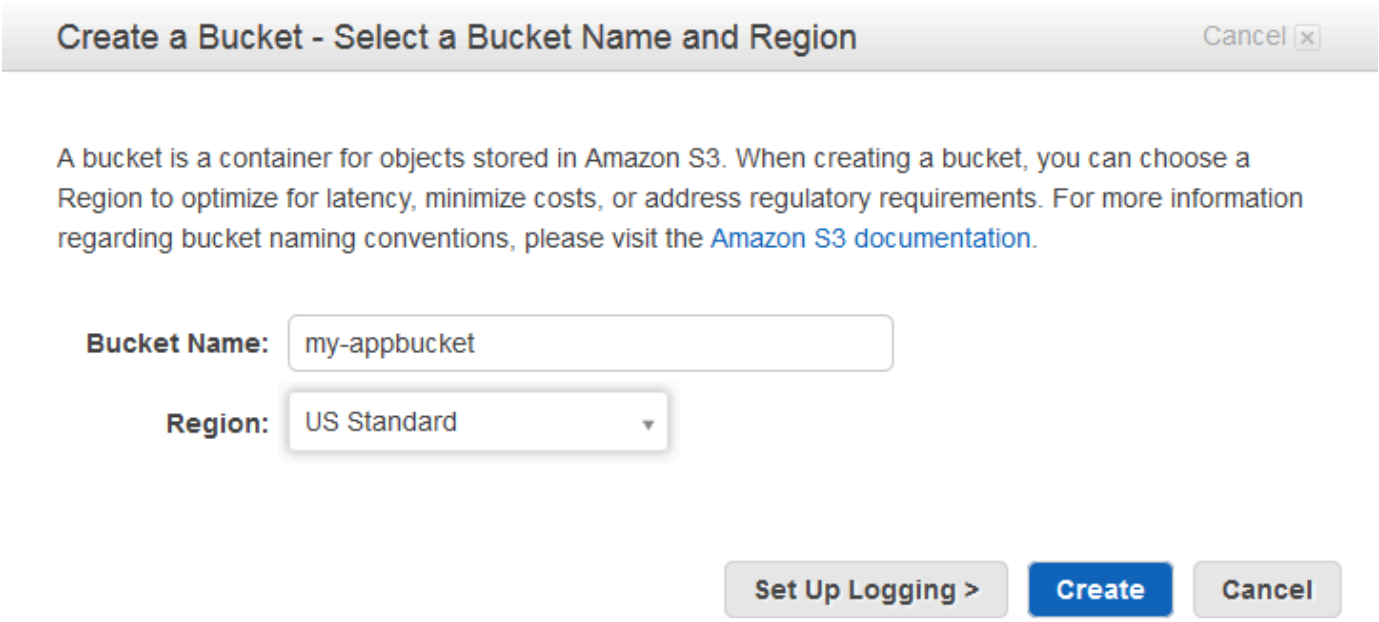
尽管 CodePipeline 可以直接使用来自 GitHub 或 CodeCommit 作为源代码的代码，但本演练演示了如何使用 Amazon S3 存储桶。Amazon S3 存储桶 CodePipeline 允许检测应用程序代码的更改并自动部署更改后的应用程序。如果需要，您可以使用现有存储桶。确保存储桶符合 CodePipeline 文档中简单管道演练（Amazon S3 存储桶）中所述的标准。 CodePipeline

> **Important**
>
> Amazon S3 存储桶必须位于稍后将创建管道的同一区域。目前，仅 CodePipeline 支持美国东部（弗吉尼亚北部）区域 (us-ea AWS OpsWorks st-1) 的堆栈提供商。本演练中的全部资源都

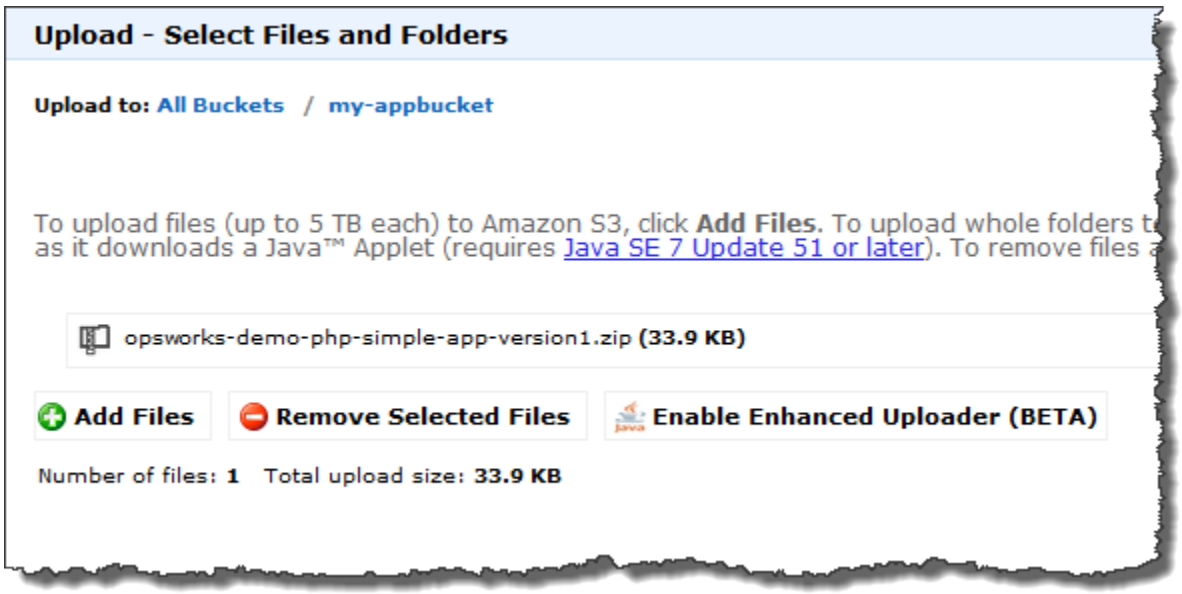
应在美国东部（弗吉尼亚州北部）区域中创建。由于 CodePipeline 需要版本控制源，因此还必须对存储桶进行版本控制。有关详细信息，请参阅使用版本控制。

要将应用程序上传到 Amazon S3 存储桶

1. 从GitHub 网站上下载 AWS OpsWorks Stacks 示例 PHP 应用程序的 ZIP 文件，然后将其保存到本地计算机上方便的位置。
2. 请确保 `index.php` 和 `ASSETS` 文件夹位于已下载的 ZIP 文件的根级别。如果没有，解压缩该文件，并创建一个这些文件都位于根级别的新 ZIP 文件。
3. 通过 https://console.aws.amazon.com/s3/ 打开 Amazon S3 控制台。
4. 选择创建存储桶。
5. 在 Create a Bucket - Select a Bucket Name and Region 页面上，对于 Bucket Name，键入存储桶的唯一名称。存储桶名称在所有 AWS 账户中必须是唯一的，而不仅仅是在您自己的账户中。本演练使用名称 **`my-appbucket`**，但您可以使用 `my-appbucket-`*`yearmonthday`* 让您的存储桶具有唯一的名称。从 Region 下拉列表中，选择 US Standard，然后选择 Create。US Standard 等同于 `us-east-1`。



6. 从 All Buckets 列表中选择您创建的存储桶。
7. 在存储桶页面上，选择 Upload (上传)。
8. 在 Upload - Select Files and Folders 页面上，选择 Add files。找到您在第 1 步中保存的 ZIP 文件，选择 Open (打开)，然后选择 Start Upload (开始上传)。



9. 上传完成后，从存储桶中的文件列表中选择 ZIP 文件，然后选择 Properties。
10. 在 Properties 窗格中，复制您的 ZIP 文件的链接，并记下此链接。您将需要存储桶名称和该链接的 ZIP 文件名部分来创建您的管道。

第 3 步：将您的应用程序添加到 AWS OpsWorks Stacks

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

在中创建管道之前 CodePipeline，请将 PHP 测试应用程序添加到 AWS OpsWorks Stacks。创建管道时，需要选择已添加到 AWS OpsWorks Stacks 的应用程序。

准备好前面步骤 10 中的 Amazon S3 存储桶链接。您需要指向存储测试应用程序的存储桶的链接，才能完成此过程。

将应用程序添加到 AWS OpsWorks Stacks

1. 在 AWS OpsWorks Stacks 控制台中打开 MyStack，然后在导航窗格中选择 Apps。
2. 选择 Add app (添加应用程序)。
3. 在 Add App (添加应用程序) 页面上，提供以下信息：

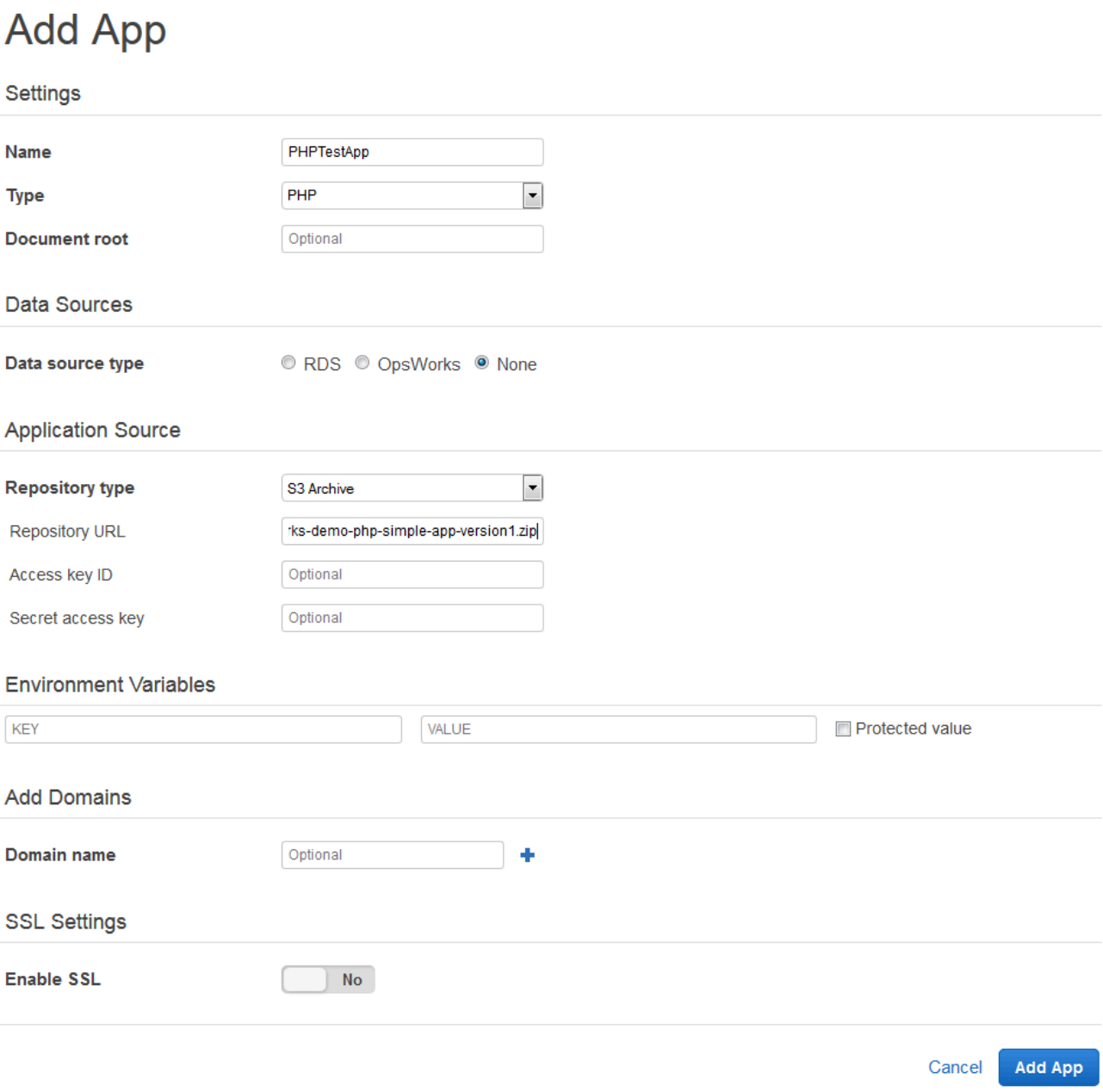
a. 为应用程序指定名称。此演练使用名称 PHPTestApp。

b. 在 Type 下拉列表中，选择 PHP。

c. 对于 Data source type (数据源类型)，选择 None (无)。此应用程序不需要外部数据库或数据源。

d. 在 Repository type 下拉列表中选择 S3 Archive。

e. 在 Repository URL 字符串框中，粘贴在步骤 2：将应用程序代码上传到 Amazon S3 存储桶的步骤 10 中复制的 URL。您的表单应类似如下：



4. 您无需更改本表单中的任何其他设置。选择 Add App。

5. 当 PHP TestApp 应用程序出现在应用程序页面的列表中时，请继续执行下一个过程步骤 4：在中创建管道 CodePipeline。

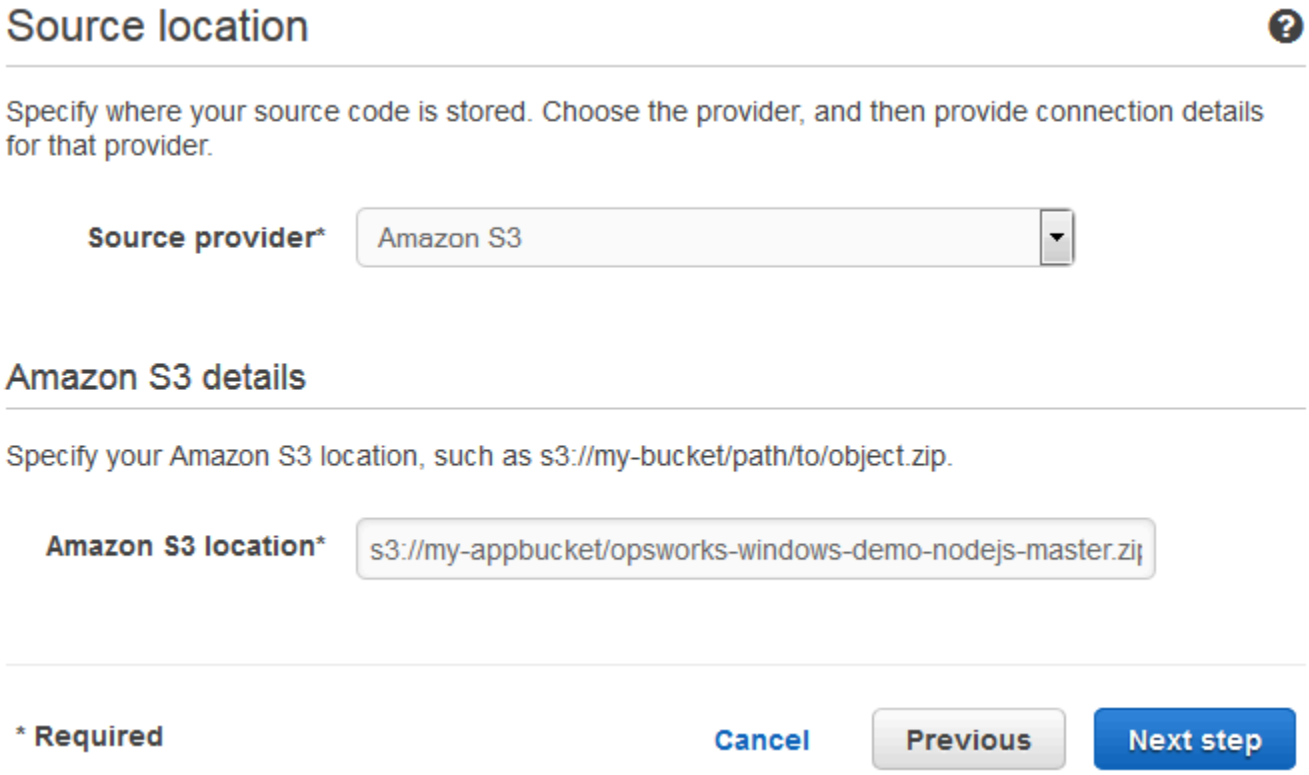
步骤 4：在中创建管道 CodePipeline

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS：Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

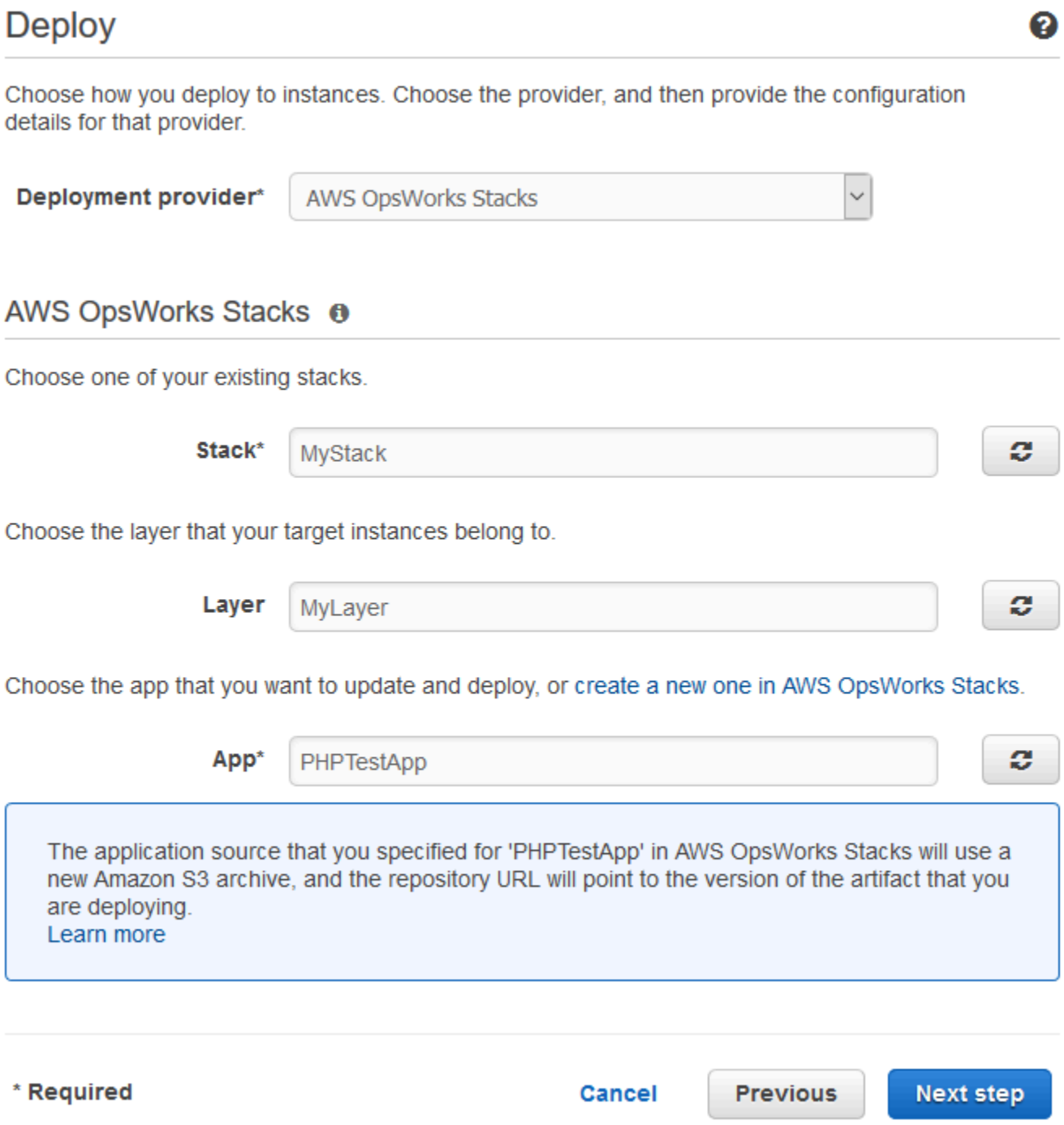
在你有一个包含层的堆栈并在堆栈中配置了至少一个实例之后，创建一个 CodePipeline 以 AWS OpsWorks AWS OpsWorks Stacks 作为提供者的管道，将应用程序或 Chef 食谱部署到你 AWS OpsWorks 的 Stacks 资源中。

要创建管道

1. 打开 CodePipeline 控制台，网址为 https://console.aws.amazon.com/codepipeline/。
2. 选择 Create pipeline（创建管道）。
3. 在 “入门 CodePipeline” 页面上**MyOpsWorksPipeline**，键入您的账户独有的管道名称或任何其他管道名称，然后选择 Ne xt step。
4. 在 Source Location 页面上，从 Source provider 下拉列表中选择 Amazon S3。
5. 在 Amazon S3 详细信息 页面中，键入格式为 **s3://*bucket-name*/*file name*** 的 Amazon S3 存储桶路径。请参阅您在 步骤 2：将应用程序代码上传到 Amazon S3 存储桶 的步骤 10 中记录的链接。在本演练中，路径为 `s3://my-appbucket/opsworks-demo-php-simple-app-version1.zip`。选择下一步。



6. 在 Build 页面上，从下拉列表中选择 No Build，然后选择 Next step。
7. 在 Deploy (部署) 页面上，选择 AWS OpsWorks Stacks 作为部署提供程序。



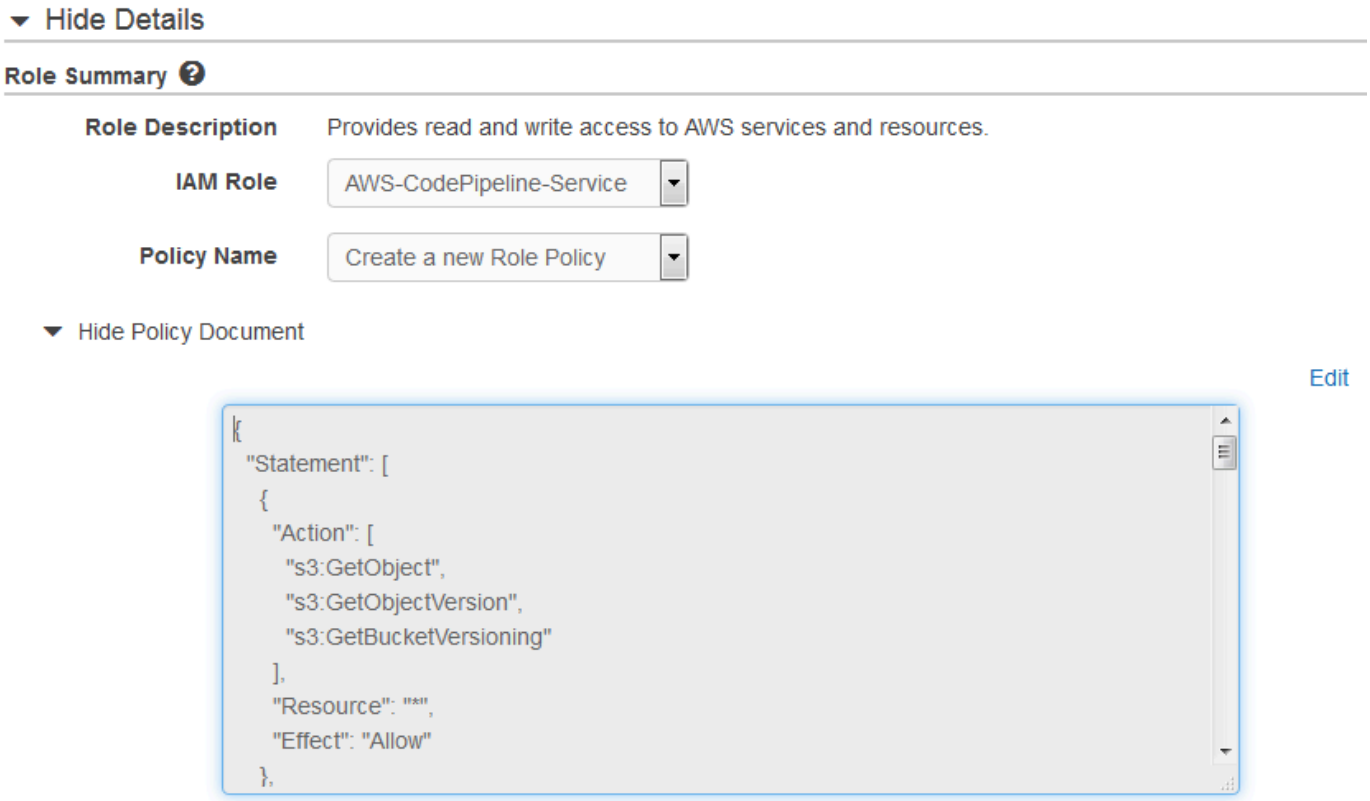
8. 在 Stack 字段中，键入 `MyStack` 或者在步骤 1：在 AWS OpsWorks Stacks 中创建堆栈、层和实例中创建的堆栈的名称。
9. 在 Layer 字段中，键入 `MyLayer` 或者在步骤 1：在 AWS OpsWorks Stacks 中创建堆栈、层和实例中创建的层的名称。

10. 在应用程序字段中，选择在 步骤 2：将应用程序代码上传到 Amazon S3 存储桶 中上传到 Amazon S3 的应用程序，然后选择 下一步。
11. 在 Amazon Web Service 角色 页面上，选择创建角色。

    系统将打开一个新窗口，其中的 IAM 控制台页面将介绍将为您创建的角色 `AWS-CodePipeline-Service`。从 Policy name 下拉列表中，选择 Create new policy。请确保策略文档具有以下内容。选择 Edit，根据需要更改策略文档。

```
{
    "Statement": [
        {
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:GetBucketVersioning"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": "opsworks:*",
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

    完成对策略文档的更改后，选择 Allow (允许)。您的更改将显示在 IAM 控制台中。



> **Note**
>
> 如果角色创建失败，可能是因为您已经有一个名为 A WS CodePipeline-Servic e 的 IAM 角色。如果您在 2016 年 5 月之前一直在使用 AWS CodePipeline-Servic e 角色，则该角色可能无权将 AWS OpsWorks Stacks 用作部署提供商；在这种情况下，您必须更新策略声明，如本步骤所示。如果您看到错误消息，请回到此步骤的开始位置，然后选择 Use existing role (使用现有角色) 而非 Create role (创建角色)。如果您使用现有角色，该角色应随附一个包含此步骤中所示权限的策略。有关服务角色及其策略声明的更多信息，请参阅 Edit a Policy for an IAM Service Role。

12. 如果角色创建过程成功，IAM 页面将关闭，并且您将返回 Amazon Web Service 角色页面。选择下一步。
13. 在 Review your pipeline 页面上，验证页面上显示的选择，然后选择 Create pipeline。

We will create your pipeline with the following resources.

Source Stage

Source provider Amazon S3

Amazon S3 location s3://my-appbucket0/opsworks-demo-php-simple-app-version1.zip

Build Stage

Build provider No Build

Beta Stage

Deployment provider AWS OpsWorks

Stack MyStack

App PHPTestApp

Layer MyLayer

Pipeline settings

Pipeline name MyOpsWorksPipeline

Artifact location s3://codepipeline-us-east-
AWS CodePipeline will use this existing S3 bucket to store artifacts for this pipeline. Depending on the size of your artifacts, you might be charged for storage costs. For more information, see Amazon S3 storage pricing.

Role name AWS-CodePipeline-Service

To save this configuration with these resources, choose Create pipeline.

Would you like to create this pipeline?

Cancel Previous Create pipeline

14. 当您的管道已准备就绪，该管道应开始查找您的源代码并自动将应用程序部署到您的堆栈。此过程可能耗时数分钟。

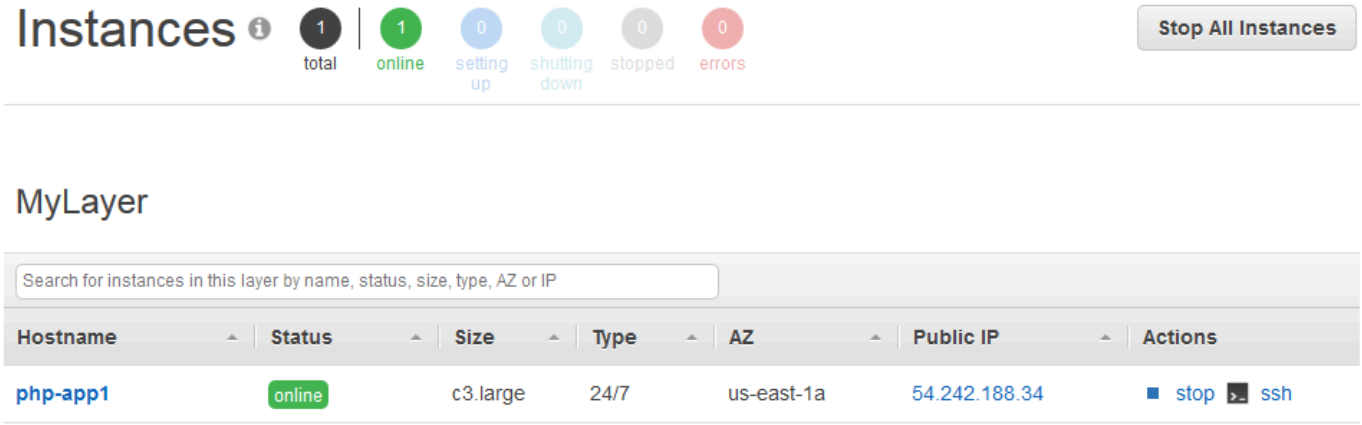
步骤 5：在 AWS OpsWorks Stacks 中验证应用程序部署

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

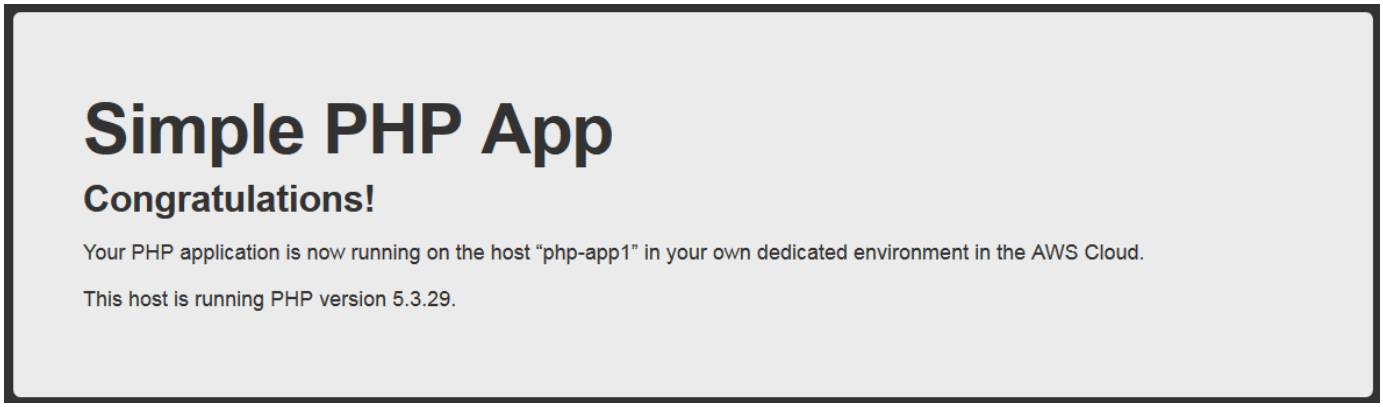
要验证已将 PHP 应用程序 CodePipeline 部署到您的堆栈，请登录到您在中创建的实例步骤 1：在 AWS OpsWorks Stacks 中创建堆栈、层和实例。您应该可以看到并使用 PHP Web 应用程序。

验证您的 AWS OpsWorks Stacks 实例中的应用程序部署

1. 打开 AWS OpsWorks 控制台，网址为 https://console.aws.amazon.com/opsworks/。
2. 在 AWS OpsWorks Stacks 控制面板上，选择 MyStack，然后选择MyLayer。
3. 在导航窗格中选择 Instances，然后选择您创建的实例的公有 IP 地址以查看 Web 应用程序。



该应用程序将显示在新的浏览器选项卡中。



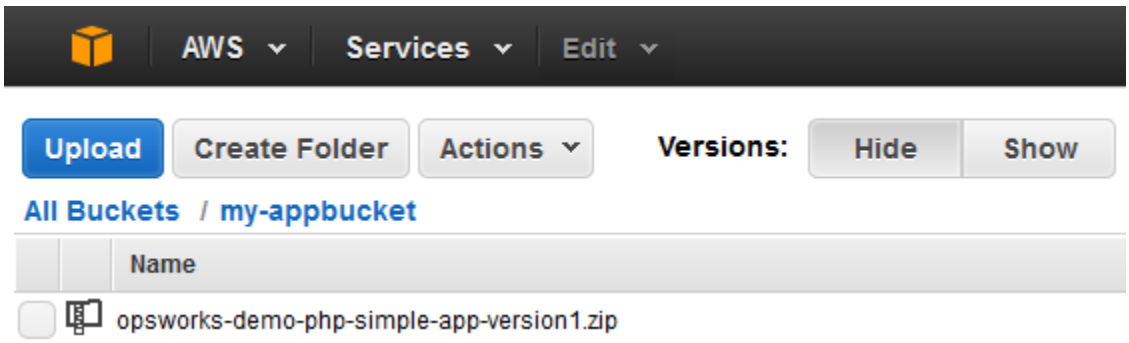
步骤 6（可选）：更新应用程序代码以查看自动 CodePipeline重新部署您的应用程序

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

当您对使用部署的应用程序或食谱中的代码进行更改时 CodePipeline，更新的构件将自动部署 CodePipeline 到您的目标实例（在本例中为目标堆栈 AWS OpsWorks 堆栈）。本部分向您演示了在更新示例 PHP 应用程序中的代码时的自动重新部署工作。

编辑示例应用程序中的代码

1. 登录 AWS Management Console 并打开 Amazon S3 控制台，网址为 https://console.aws.amazon.com/s3/。
2. 打开存储您的示例 PHP 应用程序的存储桶。



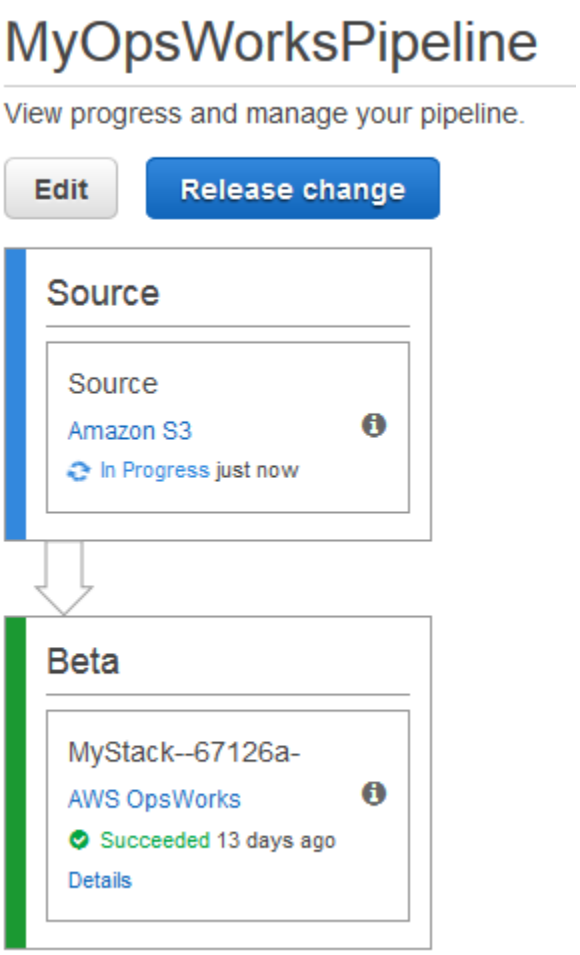
3. 选择包含该应用程序的 ZIP 文件。在 Actions 菜单上选择 Download。

4. 在对话框中，打开上下文 (右键单击) 菜单，选择 Download，然后将 ZIP 文件保存到方便位置。选择 确定。
5. 将 ZIP 文件的内容提取到方便的位置。您可能需要更改提取的文件夹及其子文件夹和内容的权限，才能进行编辑。在 opsworks-demo-php-simple-app-version1 文件夹中，打开 index.php 文件进行编辑。
6. 搜索 Your PHP application is now running 这一短语。将文本 Your PHP application is now running 替换为 You've just deployed your first app to AWS OpsWorks with AWS CodePipeline,。请勿编辑变量。

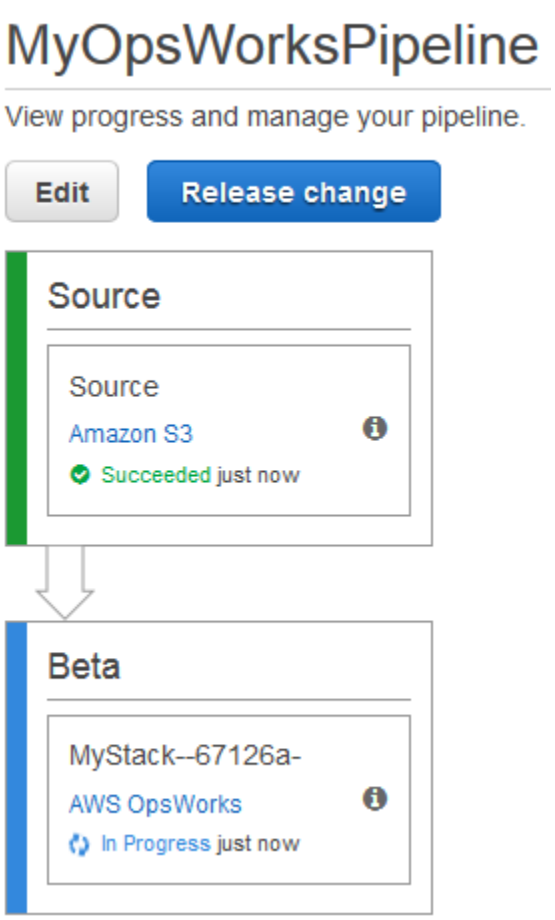
```
<body>
    <div class="container">
        <div class="hero-unit">
            <h1>Simple PHP App</h1>
            <h2>Congratulations!</h2>
            <p>You've just deployed your first app to AWS OpsWorks with AWS CodePipeline,</p>
            <p>on the host &ldquo;<?php echo gethostname(); ?>&rdquo; </p>
            <p>in your own dedicated environment in the AWS Cloud.</p>
            <p>This host is running PHP version <?php echo phpversion(); ?>.</p>
        </div>
    </div>

    <script src="//ajax.googleapis.com/ajax/libs/jquery/1.8.3/jquery.min.js"></script>
    <script src="assets/js/bootstrap.min.js"></script>
</body>
```

7. 保存并关闭 index.php文件。
8. 压缩 opsworks-demo-php-simple-app-version1 文件夹，并将 ZIP 文件保存到方便位置。请勿更改 ZIP 文件的名称。
9. 将新的 ZIP 文件上传到 Amazon S3 存储桶。在本演练中，存储桶的名称为 my-appbucket。
10. 打开 CodePipeline 控制台，然后打开你的 AWS OpsWorks Stacks 管道 (MyOpsWorksPipeline)。选择 Release Change。

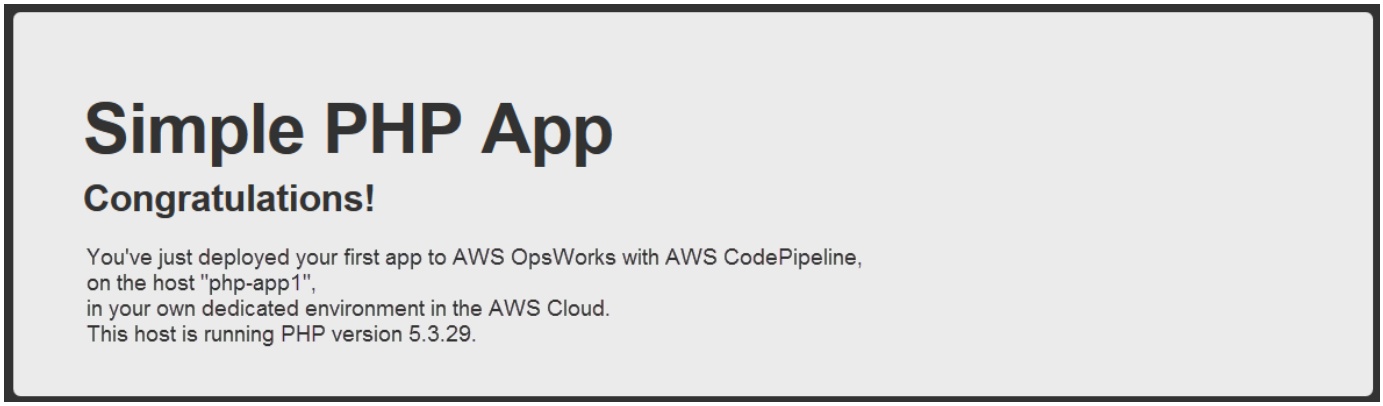
    （您可以等待在您的 Amazon S3 存储桶中检测 CodePipeline 到与应用程序更新版本相比的代码更改。 为了节省时间，本演练指导您只需选择 “发布更改” 即可。 ）
11. CodePipeline 在管道的各个阶段进行观察。首先， CodePipeline 检测对源构件的更改。



CodePipeline 在 Stack AWS OpsWorks s 中将更新的代码推送到堆栈中。



12. 成功完成管道的两个阶段后，在 Stac AWS OpsWorks ks (MyStack) 中打开您的堆栈。
13. 在MyStack属性页面上，选择实例。
14. 在 Public IP (公有 IP) 列中，选择实例的公有 IP 地址以查看更新后应用程序的文本。



步骤 7 (可选)：清除资源

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

为了防止向您的 AWS 账户收取不必要的费用，您可以删除用于本演练的 AWS 资源。这些 AWS 资源包括堆栈 AWS OpsWorks 堆栈、IAM 角色和实例配置文件以及您在中 CodePipeline创建的管道。但是，随着您继续了解有关 AWS OpsWorks Stacks和 CodePipeline的更多信息，您可能需要继续使用这些 AWS 资源。如果您愿意，也可以在完成本演练后保留这些资源。

从堆栈中删除应用程序

由于您没有将应用程序作为 AWS CloudFormation 模板的一部分创建或应用，因此在中删除堆栈之前，请先删除 PHP 测试应用程序 AWS CloudFormation。

1. 在 AWS OpsWorks Stacks 控制台的服务导航窗格中，选择应用程序。
2. 在 “应用程序” 页面上，选择 PHP TestApp，然后在 “操作” 中选择 “删除”。当系统提示您确认时，选择删除。 AWS OpsWorks Stacks 将删除该应用程序。

删除堆栈

由于堆栈是通过运行 AWS CloudFormation 模板创建的，因此可以在 AWS CloudFormation 控制台中删除堆栈，包括该模板创建的层、实例、实例配置文件和安全组。

1. 打开控制 AWS CloudFormation 台。
2. 在 AWS CloudFormation 控制台仪表板中，选择您创建的堆栈（ MyStack ）。在 Actions (操作) 菜单上，选择 Delete Stack (删除堆栈)。请在提示您进行确认时选择 Yes, Delete (是，删除)。
3. 等待 DELETE_COMPLETE 出现在堆栈的 Status (状态) 列中。

删除管道

1. 打开控制 CodePipeline 台。
2. 在 CodePipeline 仪表板中，选择您为此演练创建的管道。

3. 在管道页面上，选择 Edit。
4. 在 Edit 页面上，选择 Delete。请在提示您进行确认时选择删除。

使用 AWS OpsWorks Stacks CLI

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

AWS OpsWorks Stacks 命令行界面 (CLI) 提供的功能与控制台相同，可用于各种任务。 AWS OpsWorks Stacks CLI 是其中的 AWS CLI一部分。有关更多信息，包括如何安装和配置 AWS CLI，请转到什么是 AWS 命令行界面？ 。有关各个命令的完整说明，请转到 AWS OpsWorks Stacks 参考。

> **Note**
>
> 如果您使用的是基于 Windows 的工作站，也可以运行适用于 Windows 的 AWS 工具，通过 PowerShell 命令行执行 AWS OpsWorks 堆栈操作。有关更多信息，请参阅适用于 Windows 的 AWS 工具 PowerShell。

AWS OpsWorks 堆栈命令的常规格式如下：

```
aws opsworks --region us-west-1 opsworks command-name [--argument1 value] [...]
```

如果某个参数值是 JSON 对象，您应对 " 字符进行转义，否则命令可能返回 JSON 无效错误。例如，如果 JSON 对象为 `"{"somekey":"somevalue"}"`，则您应将其格式化为 `"{\"somekey\":\"somevalue\"}"`。另一种替代方法是将 JSON 对象放在文件中，并使用 `file://` 将其包括在命令行内。以下示例使用存储在 appsource.json 中的应用程序源对象创建应用程序。

```
aws opsworks --region us-west-1 create-app --stack-id 8c428b08-a1a1-46ce-a5f8-
feddc43771b8 --name SimpleJSP --type java --app-source file://appsource.json
```

大部分命令返回一个或多个值，打包为 JSON 对象。以下部分包含一些示例。有关各命令返回值的详细说明，请转到 AWS OpsWorks Stacks 参考。

Note

AWS CLI 命令必须指定区域，如示例所示。--region 参数的有效值显示在下表中。要简化 AWS OpsWorks Stacks 命令字符串，请配置 CLI 以指定您的默认区域，这样您就可以省略该`--region`参数。如果您通常在多个区域终端节点上工作，请不要将配置 AWS CLI 为使用默认的区域终端节点。加拿大（中部）地区终端节点 AWS CLI 仅在 API 中可用；它不适用于您在中创建的堆栈。 AWS Management Console有关更多信息，请参阅配置 Amazon Web Services Region。

区域名称	命令代码
美国东部（俄亥俄）区域	us-east-2
美国东部（弗吉尼亚北部）区域	us-east-1
美国西部（加利福尼亚北部）区域	us-west-1
美国西部（俄勒冈州）区域	us-west-2
加拿大（中部）区域	ca-central-1
欧洲（爱尔兰）区域	eu-west-1
欧洲（伦敦）区域	eu-west-2
欧洲（巴黎）区域	eu-west-3
欧洲（法兰克福）区域	eu-central-1
亚太区域（东京）	ap-northeast-1
亚太区域（首尔）	ap-northeast-2
亚太地区 (孟买) 区域	ap-south-1
亚太区域（新加坡）	ap-southeast-1
亚太区域（悉尼）	ap-southeast-2
南美洲（圣保罗）区域	sa-east-1

要使用 CLI 命令，您必须具有适当的权限。有关 AWS OpsWorks Stacks 权限的更多信息，请参阅管理用户权限。要确定特定命令所需的权限，请参阅 AWS OpsWorks Stacks 参考中的命令参考页面。

以下各节介绍如何使用 AWS OpsWorks Stacks CLI 执行各种常见任务。

创建实例 (create-instance)

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

使用 create-instance 命令在指定堆栈上创建实例。

主题

- 使用默认主机名创建实例
- 使用主题化主机名创建实例
- 使用自定义 AMI 创建实例

使用默认主机名创建实例

```
C:\>aws opsworks --region us-west-1 create-instance --stack-id 935450cc-61e0-4b03-
a3e0-160ac817d2bb
    --layer-ids 5c8c272a-f2d5-42e3-8245-5bf3927cb65b --instance-type m1.large --os
 "Amazon Linux"
```

参数如下所示：

- `stack-id`— 你可以从控制台上的堆栈设置页面（查找 ID）或通过调用 desc ribe-stacks 来获取堆栈 OpsWorks ID。
- `layer-ids`— 您可以从控制台上的图层详细信息页面（查找 ID）或通过调用 desc ribe-layers 来获取图层 OpsWorks ID。在此示例中，实例只属于一个层。
- `instance-type` - 定义实例内存、CPU、存储容量以及每小时成本的规范；对于本示例为 `m1.large`。

- os - 实例的操作系统；对于本示例为 Amazon Linux。

命令返回一个包含实例 ID 的 JSON 对象，如下所示：

```
{
    "InstanceId": "5f9adeaa-c94c-42c6-aeef-28a5376002cd"
}
```

此示例创建具有默认主机名的实例，这仅是一个整数。以下部分介绍了如何使用从主题生成的主机名来创建实例。

使用主题化主机名创建实例

您还可以使用主体化主机名来创建实例。您可以在创建堆栈时指定主题。有关更多信息，请参阅创建新堆栈。要创建实例，请先调get-hostname-suggestion用生成名称。例如：

```
C:\>aws opsworks get-hostname-suggestion --region us-west-1 --layer-id 5c8c272a-
f2d5-42e3-8245-5bf3927cb65b
```

如果您指定了默认 `Layer Dependent` 主题，则 `get-hostname-suggestion` 仅附加一位数字到层的短名称。有关更多信息，请参阅 创建新堆栈。

该命令返回生成的主机名。

```
{
    "Hostname": "php-app2",
    "LayerId": "5c8c272a-f2d5-42e3-8245-5bf3927cb65b"
}
```

然后，您可以使用 `hostname` 参数将生成的名称传递到 `create-instance`，如下所示：

```
c:\>aws --region us-west-1 opsworks create-instance --stack-id 935450cc-61e0-4b03-
a3e0-160ac817d2bb
   --layer-ids 5c8c272a-f2d5-42e3-8245-5bf3927cb65b --instance-type m1.large --os
 "Amazon Linux" --hostname "php-app2"
```

使用自定义 AMI 创建实例

以下 create-instance 命令使用自定义 AMI (必须来自堆栈的区域) 创建实例。有关如何为 AWS OpsWorks 堆栈创建自定义 AMI 的更多信息，请参阅使用自定义 AMI。

```
C:\>aws opsworks create-instance --region us-west-1 --stack-id c5ef46ce-3ccd-472c-
a3de-9bec94c6028e
   --layer-ids 6ff8a2ac-c9cc-49cf-9c67-fc852539ade4 --instance-type c3.large --os
 Custom
   --ami-id ami-6c61f104
```

参数如下所示：

- `stack-id`— 你可以从控制台上的堆栈设置页面（查找 ID）或通过调用 desc ribe-stacks 来获取堆栈 OpsWorks ID。
- `layer-ids`— 您可以从控制台上的图层详细信息页面（查找 ID）或通过调用 desc ribe-layers 来获取图层 OpsWorks ID。在此示例中，实例只属于一个层。
- `instance-type` - 该值定义了实例的内存、CPU、存储容量以及每小时成本，并且必须与 AMI 兼容 (对于本示例为 `c3.large`)。
- `os` - 该实例的操作系统，对于自定义 AMI 必须设置为 `Custom`。
- `ami-id` - AMI ID，该 ID 应类似于 `ami-6c61f104`

> **Note**
>
> 在您使用自定义 AMI 时，不支持块设备映射，并且忽略您为 `--block-device-mappings` 选项指定的值。

命令返回一个包含实例 ID 的 JSON 对象，如下所示：

```
{
    "InstanceId": "5f9adeaa-c94c-42c6-aeef-28a5376002cd"
}
```

部署应用程序 (create-deployment)

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

使用 create-deployment 命令将应用程序部署到指定的堆栈。

主题

- 部署应用程序

部署应用程序

```
aws opsworks --region us-west-1 create-deployment --stack-id cfb7e082-ad1d-4599-8e81-
de1c39ab45bf
   --app-id 307be5c8-d55d-47b5-bd6e-7bd417c6c7eb --command "{\"Name\":\"deploy\"}"
```

参数如下所示：

- `stack-id`— 您可以从控制台上的堆栈设置页面（查找 OpsWorks ID）或通过调用获取堆栈 ID `describe-stacks`。
- `app-id`— 您可以从应用程序的详细信息页面（查找 ID）或致电 desc ribe-apps 获取应用程序 OpsWorks ID。
- `command` - 参数获取将命令名称设置为 `deploy` 的 JSON 对象，该命令将指定的应用程序部署到堆栈。

  请注意 JSON 对象中的 " 字符均已转义。否则，命令可能返回 JSON 无效错误。

命令返回一个包含部署 ID 的 JSON 对象，如下所示：

```
{
    "DeploymentId": "5746c781-df7f-4c87-84a7-65a119880560"
```

```
}
```

> **Note**
>
> 之前的示例部署到堆栈中的每个实例。要部署指定的实例子集，请添加 `instance-ids` 参数并列出实例 ID。

列出堆栈的应用程序 (describe-apps)

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

使用 describe-apps 命令列出堆栈的应用程序或者获取有关指定应用程序的详细信息。

```
aws opsworks --region us-west-1 describe-apps --stack-id 38ee91e2-abdc-4208-
a107-0b7168b3cc7a
```

前面的示例返回一个 JSON 对象，其中包含有关每个应用程序的信息。此示例只有一个应用程序。有关各个参数的说明，请参阅 describe-apps。

```
{
  "Apps": [
    {
      "StackId": "38ee91e2-abdc-4208-a107-0b7168b3cc7a",
      "AppSource": {
        "Url": "url",
        "Type": "archive"
      },
      "Name": "SimpleJSP",
      "EnableSsl": false,
      "SslConfiguration": {},
      "AppId": "da1decc1-0dff-43ea-ad7c-bb667cd87c8b",
      "Attributes": {
        "RailsEnv": null,
```

```
        "AutoBundleOnDeploy": "true",
        "DocumentRoot": "ROOT"
      },
      "Shortname": "simplejsp",
      "Type": "other",
      "CreatedAt": "2013-08-01T21:46:54+00:00"
    }
  ]
}
```

列出堆栈的命令 (describe-commands)

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

使用 describe-commands 命令列出堆栈的命令或者获取有关指定命令的详细信息。以下示例获取有关已在指定实例上执行的命令的详细信息。

```
aws opsworks --region us-west-1 describe-commands --instance-id
 8c2673b9-3fe5-420d-9cfa-78d875ee7687
```

该命令返回一个 JSON 对象，其中包含有关各个命令的详细信息。对于此示例，Type 参数标识命令名称、部署或取消部署。有关其他参数的说明，请参阅 describe-commands。

```
{
  "Commands": [
    {
      "Status": "successful",
      "CompletedAt": "2013-07-25T18:57:47+00:00",
      "InstanceId": "8c2673b9-3fe5-420d-9cfa-78d875ee7687",
      "DeploymentId": "6ed0df4c-9ef7-4812-8dac-d54a05be1029",
      "AcknowledgedAt": "2013-07-25T18:57:41+00:00",
      "LogUrl": "https://s3.amazonaws.com/prod_stage-log/logs/008c1a91-
ec59-4d51-971d-3adff54b00cc?AWSAccessKeyId=AIDACKCEVSQ6C2EXAMPLE
 &Expires=1375394373&Signature=HkXil6UuNfxTCC37EPQAa462E1E%3D&response-cache-
control=private&response-content-encoding=gzip&response-content- type=text%2Fplain",
```

```
      "Type": "undeploy",
      "CommandId": "008c1a91-ec59-4d51-971d-3adff54b00cc",
      "CreatedAt": "2013-07-25T18:57:34+00:00",
      "ExitCode": 0
    },
    {
      "Status": "successful",
      "CompletedAt": "2013-07-25T18:55:40+00:00",
      "InstanceId": "8c2673b9-3fe5-420d-9cfa-78d875ee7687",
      "DeploymentId": "19d3121e-d949-4ff2-9f9d-94eac087862a",
      "AcknowledgedAt": "2013-07-25T18:55:32+00:00",
      "LogUrl": "https://s3.amazonaws.com/prod_stage-log/
logs/899d3d64-0384-47b6-a586-33433aad117c?AWSAccessKeyId=AIDACKCEVSQ6C2EXAMPLE
 &Expires=1375394373&Signature=xMsJvtLuUqWmsr8s%2FAjVru0BtRs%3D&response-cache-
control=private&response-content-encoding=gzip&response-conten t-type=text%2Fplain",
      "Type": "deploy",
      "CommandId": "899d3d64-0384-47b6-a586-33433aad117c",
      "CreatedAt": "2013-07-25T18:55:29+00:00",
      "ExitCode": 0
    }
  ]
}
```

列出堆栈的部署 (describe-deployments)

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

使用 describe-deployments 命令列出堆栈的部署或者获取有关指定部署的详细信息。

```
aws opsworks --region us-west-1 describe-deployments --stack-id 38ee91e2-abdc-4208-
a107-0b7168b3cc7a
```

前面的命令返回一个 JSON 对象，其中包含有关指定堆栈的各个部署的详细信息。有关每个参数的说明，请参阅 describe-deployments。

```
{
```

```
  "Deployments": [
      {
          "StackId": "38ee91e2-abdc-4208-a107-0b7168b3cc7a",
          "Status": "successful",
          "CompletedAt": "2013-07-25T18:57:49+00:00",
          "DeploymentId": "6ed0df4c-9ef7-4812-8dac-d54a05be1029",
          "Command": {
              "Args": {},
              "Name": "undeploy"
          },
          "CreatedAt": "2013-07-25T18:57:34+00:00",
          "Duration": 15,
          "InstanceIds": [
              "8c2673b9-3fe5-420d-9cfa-78d875ee7687",
              "9e588a25-35b2-4804-bd43-488f85ebe5b7"
          ]
      },
      {
          "StackId": "38ee91e2-abdc-4208-a107-0b7168b3cc7a",
          "Status": "successful",
          "CompletedAt": "2013-07-25T18:56:41+00:00",
          "IamUserArn": "arn:aws:iam::444455556666:user/example-user",
          "DeploymentId": "19d3121e-d949-4ff2-9f9d-94eac087862a",
          "Command": {
              "Args": {},
              "Name": "deploy"
          },
          "InstanceIds": [
              "8c2673b9-3fe5-420d-9cfa-78d875ee7687",
              "9e588a25-35b2-4804-bd43-488f85ebe5b7"
          ],
          "Duration": 72,
          "CreatedAt": "2013-07-25T18:55:29+00:00"
      }
  ]
}
```

列出堆栈的弹性 IP 地址 (describe-elastic-ips)

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

使用describe-elastic-ips命令列出已在堆栈中注册的弹性 IP 地址或获取有关指定弹性 IP 地址的详细信息。

```
aws opsworks --region us-west-2 describe-elastic-ips --instance-id b62f3e04-e9eb-436c-a91f-d9e9a396b7b0
```

前面的命令返回一个 JSON 对象，其中包含指定实例的各个弹性 IP 地址 (本示例中为一个) 的详细信息。有关每个参数的说明，请参见describe-elastic-ips。

```
{
  "ElasticIps": [
      {
          "Ip": "192.0.2.0",
          "Domain": "standard",
          "Region": "us-west-2"
      }
  ]
}
```

列出堆栈的实例 (describe-instances)

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

使用 describe-instances 命令可列出堆栈的实例或者获取有关指定实例的详细信息。

```
C:\>aws opsworks --region us-west-2 describe-instances --stack-id 38ee91e2-abdc-4208-
a107-0b7168b3cc7a
```

前面的命令返回一个 JSON 对象，其中包含指定堆栈中的每个实例的详细信息。有关各个参数的说明，请参阅 describe-instances。

```
{
  "Instances": [
      {
          "StackId": "38ee91e2-abdc-4208-a107-0b7168b3cc7a",
          "SshHostRsaKeyFingerprint":
 "f4:3b:8e:27:1b:73:98:80:5d:d7:33:e2:b8:c8:8f:de",
          "Status": "stopped",
          "AvailabilityZone": "us-west-2a",
          "SshHostDsaKeyFingerprint":
 "e8:9b:c7:02:18:2a:bd:ab:45:89:21:4e:af:0b:07:ac",
          "InstanceId": "8c2673b9-3fe5-420d-9cfa-78d875ee7687",
          "Os": "Amazon Linux",
          "Hostname": "db-master1",
          "SecurityGroupIds": [],
          "Architecture": "x86_64",
          "RootDeviceType": "instance-store",
          "LayerIds": [
              "41a20847-d594-4325-8447-171821916b73"
          ],
          "InstanceType": "c1.medium",
          "CreatedAt": "2013-07-25T18:11:27+00:00"
      },
      {
          "StackId": "38ee91e2-abdc-4208-a107-0b7168b3cc7a",
          "SshHostRsaKeyFingerprint":
 "ae:3a:85:54:66:f3:ce:98:d9:83:39:1e:10:a9:38:12",
          "Status": "stopped",
          "AvailabilityZone": "us-west-2a",
          "SshHostDsaKeyFingerprint":
 "5b:b9:6f:5b:1c:ec:55:85:f3:45:f1:28:25:1f:de:e4",
          "InstanceId": "9e588a25-35b2-4804-bd43-488f85ebe5b7",
          "Os": "Amazon Linux",
          "Hostname": "tomcustom1",
          "SecurityGroupIds": [],
          "Architecture": "x86_64",
          "RootDeviceType": "instance-store",
          "LayerIds": [
```

```
                "e6cbcd29-d223-40fc-8243-2eb213377440"
            ],
            "InstanceType": "c1.medium",
            "CreatedAt": "2013-07-25T18:15:52+00:00"
        }
    ]
}
```

列出账户的堆栈 (describe-stacks)

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

使用 describe-stacks 命令列出账户的堆栈或者获取有关指定堆栈的详细信息。

```
aws opsworks --region us-west-2 describe-stacks
```

前面的命令返回一个 JSON 对象，其中包含账户中每个堆栈 (在本示例中为两个) 的详细信息。有关各个参数的说明，请参阅 describe-stacks。

```
{
    "Stacks": [
        {
            "ServiceRoleArn": "arn:aws:iam::444455556666:role/aws-opsworks-service-
role",
            "StackId": "aeb7523e-7c8b-49d4-b866-03aae9d4fbcb",
            "DefaultRootDeviceType": "instance-store",
            "Name": "TomStack-sd",
            "ConfigurationManager": {
                "Version": "11.4",
                "Name": "Chef"
            },
            "UseCustomCookbooks": true,
            "CustomJson": "{\n  \"tomcat\": {\n    \"base_version\": 7,\n
 \"java_opts\": \"-Djava.awt.headless=true -Xmx256m\"\n  },\n  \
"datasources\": {\n    \"ROOT\": \"jdbc/mydb\"\n  }\n}",
            "Region": "us-west-2",
```

```
            "DefaultInstanceProfileArn": "arn:aws:iam::444455556666:instance-profile/
aws-opsworks-ec2-role",
            "CustomCookbooksSource": {
                "Url": "git://github.com/example-repo/tomcustom.git",
                "Type": "git"
            },
            "DefaultAvailabilityZone": "us-west-2a",
            "HostnameTheme": "Layer_Dependent",
            "Attributes": {
                "Color": "rgb(45, 114, 184)"
            },
            "DefaultOs": "Amazon Linux",
            "CreatedAt": "2013-08-01T22:53:42+00:00"
        },
        {
            "ServiceRoleArn": "arn:aws:iam::444455556666:role/aws-opsworks-service-
role",
            "StackId": "40738975-da59-4c5b-9789-3e422f2cf099",
            "DefaultRootDeviceType": "instance-store",
            "Name": "MyStack",
            "ConfigurationManager": {
                "Version": "11.4",
                "Name": "Chef"
            },
            "UseCustomCookbooks": false,
            "Region": "us-west-2",
            "DefaultInstanceProfileArn": "arn:aws:iam::444455556666:instance-profile/
aws-opsworks-ec2-role",
            "CustomCookbooksSource": {},
            "DefaultAvailabilityZone": "us-west-2a",
            "HostnameTheme": "Layer_Dependent",
            "Attributes": {
                "Color": "rgb(45, 114, 184)"
            },
            "DefaultOs": "Amazon Linux",
            "CreatedAt": "2013-10-25T19:24:30+00:00"
        }
    ]
}
```

列出堆栈的层 (describe-layers)

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

使用 describe-layers 命令可列出堆栈的层或获取有关指定层的详细信息。

```
aws opsworks --region us-west-2 describe-layers --stack-id 38ee91e2-abdc-4208-
a107-0b7168b3cc7a
```

前面的命令返回一个 JSON 对象，其中包含有关指定堆栈中各个层的详细信息，在本例中为 MySQL 层和自定义层。有关各个参数的说明，请参阅 describe-layers。

```
{
  "Layers": [
      {
          "StackId": "38ee91e2-abdc-4208-a107-0b7168b3cc7a",
          "Type": "db-master",
          "DefaultSecurityGroupNames": [
              "AWS-OpsWorks-DB-Master-Server"
          ],
          "Name": "MySQL",
          "Packages": [],
          "DefaultRecipes": {
              "Undeploy": [],
              "Setup": [
                  "opsworks_initial_setup",
                  "ssh_host_keys",
                  "ssh_users",
                  "mysql::client",
                  "dependencies",
                  "ebs",
                  "opsworks_ganglia::client",
                  "mysql::server",
                  "dependencies",
                  "deploy::mysql"
              ],
```

```
            "Configure": [
                "opsworks_ganglia::configure-client",
                "ssh_users",
                "agent_version",
                "deploy::mysql"
            ],
            "Shutdown": [
                "opsworks_shutdown::default",
                "mysql::stop"
            ],
            "Deploy": [
                "deploy::default",
                "deploy::mysql"
            ]
        },
        "CustomRecipes": {
            "Undeploy": [],
            "Setup": [],
            "Configure": [],
            "Shutdown": [],
            "Deploy": []
        },
        "EnableAutoHealing": false,
        "LayerId": "41a20847-d594-4325-8447-171821916b73",
        "Attributes": {
            "MysqlRootPasswordUbiquitous": "true",
            "RubygemsVersion": null,
            "RailsStack": null,
            "HaproxyHealthCheckMethod": null,
            "RubyVersion": null,
            "BundlerVersion": null,
            "HaproxyStatsPassword": null,
            "PassengerVersion": null,
            "MemcachedMemory": null,
            "EnableHaproxyStats": null,
            "ManageBundler": null,
            "NodejsVersion": null,
            "HaproxyHealthCheckUrl": null,
            "MysqlRootPassword": "*****FILTERED*****",
            "GangliaPassword": null,
            "GangliaUser": null,
            "HaproxyStatsUrl": null,
            "GangliaUrl": null,
            "HaproxyStatsUser": null
```

```
            },
            "Shortname": "db-master",
            "AutoAssignElasticIps": false,
            "CustomSecurityGroupIds": [],
            "CreatedAt": "2013-07-25T18:11:19+00:00",
            "VolumeConfigurations": [
                {
                    "MountPoint": "/vol/mysql",
                    "Size": 10,
                    "NumberOfDisks": 1
                }
            ]
        },
        {
            "StackId": "38ee91e2-abdc-4208-a107-0b7168b3cc7a",
            "Type": "custom",
            "DefaultSecurityGroupNames": [
                "AWS-OpsWorks-Custom-Server"
            ],
            "Name": "TomCustom",
            "Packages": [],
            "DefaultRecipes": {
                "Undeploy": [],
                "Setup": [
                    "opsworks_initial_setup",
                    "ssh_host_keys",
                    "ssh_users",
                    "mysql::client",
                    "dependencies",
                    "ebs",
                    "opsworks_ganglia::client"
                ],
                "Configure": [
                    "opsworks_ganglia::configure-client",
                    "ssh_users",
                    "agent_version"
                ],
                "Shutdown": [
                    "opsworks_shutdown::default"
                ],
                "Deploy": [
                    "deploy::default"
                ]
            },
```

```
            "CustomRecipes": {
                "Undeploy": [],
                "Setup": [
                    "tomcat::setup"
                ],
                "Configure": [
                    "tomcat::configure"
                ],
                "Shutdown": [],
                "Deploy": [
                    "tomcat::deploy"
                ]
            },
            "EnableAutoHealing": true,
            "LayerId": "e6cbcd29-d223-40fc-8243-2eb213377440",
            "Attributes": {
                "MysqlRootPasswordUbiquitous": null,
                "RubygemsVersion": null,
                "RailsStack": null,
                "HaproxyHealthCheckMethod": null,
                "RubyVersion": null,
                "BundlerVersion": null,
                "HaproxyStatsPassword": null,
                "PassengerVersion": null,
                "MemcachedMemory": null,
                "EnableHaproxyStats": null,
                "ManageBundler": null,
                "NodejsVersion": null,
                "HaproxyHealthCheckUrl": null,
                "MysqlRootPassword": null,
                "GangliaPassword": null,
                "GangliaUser": null,
                "HaproxyStatsUrl": null,
                "GangliaUrl": null,
                "HaproxyStatsUser": null
            },
            "Shortname": "tomcustom",
            "AutoAssignElasticIps": false,
            "CustomSecurityGroupIds": [],
            "CreatedAt": "2013-07-25T18:12:53+00:00",
            "VolumeConfigurations": []
        }
    ]
```

```
}
```

执行配方 (create-deployment)

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

使用 create-deployment 命令执行堆栈命令和部署命令。以下示例执行堆栈命令以在指定任务上运行自定义配方。

```
aws opsworks --region us-west-1 create-deployment --stack-id 935450cc-61e0-4b03-
a3e0-160ac817d2bb
    --command "{\"Name\":\"execute_recipes\", \"Args\":{\"recipes\":[\"phpapp::appsetup
\"]}}"
```

command 参数获取以下格式的 JSON 对象：

- Name - 指定命令名称。用于此示例的 execute_recipes 命令在堆栈的实例上执行指定的配方。
- Args - 指定列表及其值参数。此示例有一个参数 recipes，设置为要执行的配方 phpapp::appsetup。

请注意 JSON 对象中的 " 字符均已转义。否则，命令可能返回 JSON 无效错误。

该命令返回部署 ID，您可用它来确定其他 CLI 命令的命令，例如 describe-commands。

```
{
    "DeploymentId": "5cbaa7b9-4e09-4e53-aa1b-314fbd106038"
}
```

安装依赖项 (create-deployment)

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

使用 create-deployment 命令执行堆栈命令和部署命令。以下示例运行 `update_dependencies` 堆栈命令来更新堆栈实例上的依赖项。

```
aws opsworks --region us-west-1 create-deployment --stack-id 935450cc-61e0-4b03-
a3e0-160ac817d2bb
--command "{\"Name\":\"install_dependencies\"}"
```

`command` 参数获取具有 `Name` 参数的 JSON 对象，其值指定命令名称，对于本示例为 `install_dependencies`。请注意 JSON 对象中的 " 字符均已转义。否则，命令可能返回 JSON 无效错误。

该命令返回部署 ID，您可用它来确定其他 CLI 命令的命令，例如 `describe-commands`。

```
{
    "DeploymentId": "aef5b255-8604-4928-81b3-9b0187f962ff"
}
```

更新堆栈配置 (update-stack)

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

使用 update-stack 命令可更新指定堆栈的配置。以下示例更新堆栈以将自定义 JSON 添加到堆栈配置属性。

```
aws opsworks --region us-west-1 update-stack --stack-id 935450cc-61e0-4b03-
a3e0-160ac817d2bb
   --custom-json "{\"somekey\":\"somevalue\"}" --service-role-arn
 arn:aws:iam::444455556666:role/aws-opsworks-service-role
```

请注意 JSON 对象中的 " 字符均已转义。否则，命令可能返回 JSON 无效错误。

> **Note**
>
> 该示例还为堆栈指定服务角色。您必须将 `service-role-arn` 设置为有效的服务角色 ARN，否则操作将失败；此处没有默认值。如果希望，您可以指定堆栈的当前服务角色 ARN，但必须明确指定。

The `update-stack` 命令不返回值。

调试和故障排除指南

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

如果您需要调试配方或排查服务问题，最佳方法通常是按顺序执行以下步骤：

1. 针对您的特定问题查看常见的调试和故障排除问题。
2. 搜索 AWS OpsWorks Stacks 论坛以了解此处是否讨论过该问题。

   该论坛包括许多经验丰富的用户，并由 AWS OpsWorks Stacks团队监控。
3. 有关配方的问题，请参阅调试配方。
4. 请联系 AWS OpsWorks Stacks 支持人员或在 AWS OpsWorks Stacks 论坛上发布您的问题。

下一节将提供调试配方的指南。最后一节将说明常见的调试和故障排除问题及其解决方案。

> **Note**
>
> 每运行一次 Chef 都会生成一个日志，它提供了详细的运行说明，是宝贵的故障排除资源。要在日志中指定详细信息的数量，请将 `Chef::Log.level` 语句添加到指定所需日志级别的自定义配方。默认值为 `:info`。以下示例演示如何将 Chef 日志级别设置为提供最详细的运行说明的 `:debug`。
>
> ```
> Chef::Log.level = :debug
> ```
>
> 有关查看和解释 Chef 日志的更多信息，请参阅 Chef 日志。

主题

调试配方

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

当生命周期事件发生，或者您运行 Execute Recipes 堆栈命令时， AWS OpsWorks Stacks 会向代理发出一个命令，以在指定的实例上开始运行 Chef Solo，从而执行相应的配方，包括您的自定义配方。本部分介绍了一些调试失败的配方的方法。

主题

登录到失败的实例

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

如果配方失败，则实例将以 `setup_failed` 状态结束，而不是联机状态。尽管就 AWS OpsWorks Stacks 而言，该实例未处于联机状态，但 EC2 实例仍在运行，登录以解决问题通常很有用。例如，您可以检查是否已正确安装应用程序或自定义说明书。 AWS OpsWorks Stacks 对 SSH 和 RDP 登录的内置支持仅适用于处于联机状态的实例。但是，如果您已为实例分配了一个 SSH 密钥对，则您仍然可以登录，如下所示：

- Linux 实例：使用 SSH 密钥对的私有密钥即可通过第三方 SSH 客户端进行登录，如 OpenSSH 或 PuTTY。

  您可以使用 EC2 密钥对或您的个人 SSH 密钥对来实现此目的。
- Windows 实例：使用 EC2 密钥对的私有密钥来检索实例的管理员密码。

  使用该密码即可通过您首选的 RDP 客户端进行登录。有关更多信息，请参阅 作为管理员登录。您无法使用个人 SSH 密钥对检索管理员密码。

Chef 日志

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

Chef 日志是您的关键故障排除资源之一，尤其是在调试配方方面。 AWS OpsWorks Stacks 会捕获每个命令的 Chef 日志，并保留实例最近 30 个命令的日志。由于运行处于调试模式，因此该日志包含 Chef 运行的详细描述，包括发送到 `stdout` 和 `stderror` 的文本。如果配方失败，则该日志将包含 Chef 堆栈跟踪。

AWS OpsWorks Stacks 为你提供了几种查看 Chef 日志的方法。拥有该日志信息后，您可用它来调试失败的配方。

> **Note**
>
> 您还可以通过使用 SSH 连接到实例并运行代理 CLI `show_log` 命令，来查看指定日志的结尾。有关更多信息，请参阅 显示 Chef 日志。

主题

使用控制台查看 Chef 日志

查看 Chef 日志最简单的方法是，转到实例的详细信息页面。对于每个事件和执行配方命令，日志章节均包含一个条目。下面显示了实例的 Logs 部分，该实例使用了 configure 和 setup 命令，分别对应于 Configure 和 Setup 生命周期事件。

单击相应命令日志列中的显示，可查看对应的 Chef 日志。如果发生错误， AWS OpsWorks Stacks 会自动打开错误日志，该日志通常位于文件末尾。

使用 CLI 或 API 查看 Chef 日志

您可以使用 AWS OpsWorks Stacks CLI `describe-commands`命令或 `DescribeCommands`API 操作来查看存储在 Amazon S3 存储桶中的日志。以下过程演示了如何使用 CLI 来查看指定实例的任何当前日志文件。使用 `DescribeCommands` 的过程基本类似。

使用 AWS OpsWorks 堆栈查看实例的 Chef 日志

1. 打开实例的详细信息页面并复制其 OpsWorksID 值。
2. 使用该 ID 值运行 `describe-commands` CLI 命令，如下所示：

```
aws opsworks describe-commands --instance-id 67bf0da2-29ed-4217-990c-d895d51812b9
```

该命令返回一个 JSON 对象，其中包含一个 AWS OpsWorks Stacks 在实例上执行的每个命令的嵌入式对象，最新的命令排在最前面。在本示例中，Type 参数包含每个命令嵌入式对象的命令类型、`configure` 命令和 `setup` 命令。

```
{
    "Commands": [
        {
            "Status": "successful",
            "CompletedAt": "2013-10-25T19:38:36+00:00",
            "InstanceId": "67bf0da2-29ed-4217-990c-d895d51812b9",
            "AcknowledgedAt": "2013-10-25T19:38:24+00:00",
            "LogUrl": "https://s3.amazonaws.com/prod_stage-log/logs/
b6c402df-5c23-45b2-a707-ad20b9c5ae40?AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE
&Expires=1382731518&Signature=YkqS5IZN2P4wixjHwoC3aCMbn5s%3D&response-cache-
control=private&response-content-encoding=gzip&response-content-
type=text%2Fplain",
            "Type": "configure",
            "CommandId": "b6c402df-5c23-45b2-a707-ad20b9c5ae40",
            "CreatedAt": "2013-10-25T19:38:11+00:00",
            "ExitCode": 0
        },
        {
            "Status": "successful",
            "CompletedAt": "2013-10-25T19:31:08+00:00",
            "InstanceId": "67bf0da2-29ed-4217-990c-d895d51812b9",
            "AcknowledgedAt": "2013-10-25T19:29:01+00:00",
            "LogUrl": "https://s3.amazonaws.com/prod_stage-log/logs/2a90e862-
f974-42a6-9342-9a4f03468358?AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE
&Expires=1382731518&Signature=cxKYHO8mCCd4MvOyFb6ywebeQtA%3D&response-cache-
control=private&response-content-encoding=gzip&response-content-
type=text%2Fplain",
            "Type": "setup",
            "CommandId": "2a90e862-f974-42a6-9342-9a4f03468358",
            "CreatedAt": "2013-10-25T19:26:01+00:00",
```

```
            "ExitCode": 0
        }
    ]
}
```

3. 将 LogUrl 值复制到您的浏览器以查看日志。

如果实例具有多个命令，您可以向 describe-commands 中添加参数，来筛选响应对象应包含的命令。有关更多信息，请参阅 describe-commands。

在实例上查看 Chef 日志

> **Note**
>
> 该部分中的主题适用于 Chef 12。有关 Chef 11.10 及更低版本的 Chef 日志的位置信息，请参阅为 Linux 进行 Chef 11.10 和早期版本的故障排除。

Linux 实例

AWS OpsWorks 堆栈将每个实例的 Chef 日志存储在其/var/chef/runs目录中。(对于 Linux 实例，此目录还包括相关的数据包，存储为 JSON 格式的文件。) 您需要 sudo 权限来访问此目录。每个运行的日志位于各个运行的子目录内名为 chef.log 的文件中。

AWS OpsWorks Stacks 将其内部日志存储在实例的/var/log/aws/opsworks文件夹中。通常，该信息在故障排除时不是很有帮助。但是，这些日志对 AWS OpsWorks Stacks支持很有用，如果您遇到服务问题，可能会要求您提供这些日志。Linux 日志有时也能提供有用的故障排除数据。

Windows 实例

代理日志

在 Windows 实例上， OpsWorks 日志存储在如下ProgramData路径中。其中的 number 包含时间戳。

```
C:\ProgramData\OpsWorksAgent\var\logs\number
```

> **Note**
>
> 默认情况下，ProgramData 是一个隐藏文件夹。要取消隐藏该文件夹，请导航到 Folder Options。在 View 下，选择该选项来显示隐藏的文件。

以下示例显示的是 Windows 实例上的代理日志。

```
Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
-a---         5/24/2015  11:59 PM     127277 command.20150524.txt
-a---         5/25/2015  11:59 PM     546772 command.20150525.txt
-a---         5/26/2015  11:59 PM     551514 command.20150526.txt
-a---         5/27/2015   9:43 PM     495181 command.20150527.txt
-a---         5/24/2015  11:59 PM      24353 keepalive.20150524.txt
-a---         5/25/2015  11:59 PM     106232 keepalive.20150525.txt
-a---         5/26/2015  11:59 PM     106208 keepalive.20150526.txt
-a---         5/27/2015   8:54 PM      92593 keepalive.20150527.txt
-a---         5/24/2015   7:19 PM       3891 service.20150524.txt
-a---         5/27/2015   8:54 PM       1493 service.20150527.txt
-a---         5/24/2015  11:59 PM     112549 wire.20150524.txt
-a---         5/25/2015  11:59 PM     501501 wire.20150525.txt
-a---         5/26/2015  11:59 PM     499640 wire.20150526.txt
-a---         5/27/2015   8:54 PM     436870 wire.20150527.txt
```

Chef 日志

在 Windows 实例上，Chef 日志存储在 ProgramData 路径下，如下所示。其中的 number 包含时间戳。

```
C:\ProgramData\OpsWorksAgent\var\commands\number
```

> **Note**
>
> 此目录仅包含第一次（OpsWorks 自有）Chef 运行的输出。

以下示例显示了 Windows 实例上 OpsWorks 拥有的 Chef 日志。

```
Mode                 LastWriteTime           Name
```

```
 ----                 -------------           ----
 d----         5/24/2015   7:23 PM
 configure-7ecb5f47-7626-439b-877f-5e7cb40ab8be
 d----         5/26/2015   8:30 PM            configure-8e74223b-d15d-4372-aeea-
a87b428ffc2b
 d----         5/24/2015   6:34 PM            configure-
c3980a1c-3d08-46eb-9bae-63514cee194b
 d----         5/26/2015   8:32 PM            grant_remote_access-70dbf834-1bfa-4fce-
b195-e50e85402f4c
 d----         5/26/2015  10:30 PM            revoke_remote_access-1111fce9-843a-4b27-
b93f-ecc7c5e9e05b
 d----         5/24/2015   7:21 PM            setup-754ec063-8b60-4cd4-
b6d7-0e89d7b7aa78
 d----         5/26/2015   8:27 PM            setup-af5bed36-5afd-4115-
af35-5766f88bc039
 d----         5/24/2015   6:32 PM            setup-d8abeffa-24d4-414b-
bfb1-4ad07319f358
 d----         5/24/2015   7:13 PM            shutdown-c7130435-9b5c-4a95-
be17-6b988fc6cf9a
 d----         5/26/2015   8:25 PM
 sync_remote_users-64c79bdc-1f6f-4517-865b-23d2def4180c
 d----         5/26/2015   8:48 PM
 update_custom_cookbooks-2cc59a94-315b-414d-85eb-2bdea6d76c6a
```

用户 Chef 日志

Chef 运行的日志可在名为 `logfile.txt` 的文件中找到，该文件位于以带编号的 Chef 命令命名的文件夹中，如下图所示。

C:/chef └── runs └── command-12345 ├── attribs.json ├── client.rb └── logfile.txt

解释 Chef 日志

日志的开头主要包含内部 Chef 日志记录。

```
# Logfile created on Thu Oct 17 17:25:12 +0000 2013 by logger.rb/1.2.6
[2013-10-17T17:25:12+00:00] INFO: *** Chef 11.4.4 ***
[2013-10-17T17:25:13+00:00] DEBUG: Building node object for php-app1.localdomain
[2013-10-17T17:25:13+00:00] DEBUG: Extracting run list from JSON attributes provided on
 command line
[2013-10-17T17:25:13+00:00] INFO: Setting the run_list to
 ["opsworks_custom_cookbooks::load", "opsworks_custom_cookbooks::execute"] from JSON
```

```
[2013-10-17T17:25:13+00:00] DEBUG: Applying attributes from json file
[2013-10-17T17:25:13+00:00] DEBUG: Platform is amazon version 2013.03
[2013-10-17T17:25:13+00:00] INFO: Run List is [recipe[opsworks_custom_cookbooks::load],
 recipe[opsworks_custom_cookbooks::execute]]
[2013-10-17T17:25:13+00:00] INFO: Run List expands to [opsworks_custom_cookbooks::load,
 opsworks_custom_cookbooks::execute]
[2013-10-17T17:25:13+00:00] INFO: Starting Chef Run for php-app1.localdomain
[2013-10-17T17:25:13+00:00] INFO: Running start handlers
[2013-10-17T17:25:13+00:00] INFO: Start handlers complete.
[2013-10-17T17:25:13+00:00] DEBUG: No chefignore file found at /opt/aws/opsworks/
releases/20131015111601_209/cookbooks/chefignore no files will be ignored
[2013-10-17T17:25:13+00:00] DEBUG: Cookbooks to compile: ["gem_support", "packages",
 "opsworks_bundler", "opsworks_rubygems", "ruby", "ruby_enterprise", "dependencies",
 "opsworks_commons", "scm_helper", :opsworks_custom_cookbooks]
[2013-10-17T17:25:13+00:00] DEBUG: Loading cookbook gem_support's library file: /
opt/aws/opsworks/releases/20131015111601_209/cookbooks/gem_support/libraries/
current_gem_version.rb
[2013-10-17T17:25:13+00:00] DEBUG: Loading cookbook packages's library file: /opt/aws/
opsworks/releases/20131015111601_209/cookbooks/packages/libraries/packages.rb
[2013-10-17T17:25:13+00:00] DEBUG: Loading cookbook dependencies's library file: /
opt/aws/opsworks/releases/20131015111601_209/cookbooks/dependencies/libraries/
current_gem_version.rb
[2013-10-17T17:25:13+00:00] DEBUG: Loading cookbook opsworks_commons's library file: /
opt/aws/opsworks/releases/20131015111601_209/cookbooks/opsworks_commons/libraries/
activesupport_blank.rb
[2013-10-17T17:25:13+00:00] DEBUG: Loading cookbook opsworks_commons's library file: /
opt/aws/opsworks/releases/20131015111601_209/cookbooks/opsworks_commons/libraries/
monkey_patch_chefgem_resource.rb
...
```

文件的这一部分对 Chef 专家非常有用。请注意，运行列表仅包含两个配方，尽管大多数命令会涉及更多配方。这两个配方负责处理加载和执行所有其他内置和自定义配方的任务。

文件最相关的部分通常在文件末尾。如果运行成功结束，您应看到与以下类似的内容：

```
...
[Tue, 11 Jun 2013 16:00:50 +0000] DEBUG: STDERR:
[Tue, 11 Jun 2013 16:00:50 +0000] DEBUG: ---- End output of /sbin/service mysqld
 restart ----
[Tue, 11 Jun 2013 16:00:50 +0000] DEBUG: Ran /sbin/service mysqld restart returned 0
[Tue, 11 Jun 2013 16:00:50 +0000] INFO: service[mysql]: restarted successfully
[Tue, 11 Jun 2013 16:00:50 +0000] INFO: Chef Run complete in 84.07096 seconds
```

```
[Tue, 11 Jun 2013 16:00:50 +0000] INFO: cleaning the checksum cache
[Tue, 11 Jun 2013 16:00:50 +0000] DEBUG: removing unused checksum cache file /var/chef/
cache/checksums/chef-file--tmp-chef-rendered-template20130611-4899-8wef7e-0
[Tue, 11 Jun 2013 16:00:50 +0000] DEBUG: removing unused checksum cache file /var/chef/
cache/checksums/chef-file--tmp-chef-rendered-template20130611-4899-1xpwyb6-0
[Tue, 11 Jun 2013 16:00:50 +0000] DEBUG: removing unused checksum cache file /var/chef/
cache/checksums/chef-file--etc-monit-conf
[Tue, 11 Jun 2013 16:00:50 +0000] INFO: Running report handlers
[Tue, 11 Jun 2013 16:00:50 +0000] INFO: Report handlers complete
[Tue, 11 Jun 2013 16:00:50 +0000] DEBUG: Exiting
```

> **Note**
>
> 您可以使用代理 CLI 以在运行过程中或运行之后显示日志的结尾。有关更多信息，请参阅 显示 Chef 日志。

如果配方失败，您应查找 ERROR 级别的输出，该输出将包含异常信息，后跟 Chef 堆栈跟踪，如下所示：

```
...
Please report any problems with the /usr/scripts/mysqlbug script!

[  OK  ]
MySQL Daemon failed to start.
Starting mysqld:  [FAILED]STDERR: 130611 15:07:55 [Warning] The syntax '--log-slow-
queries' is deprecated and will be removed in a future release. Please use '--slow-
query-log'/'--slow-query-log-file' instead.
130611 15:07:56 [Warning] The syntax '--log-slow-queries' is deprecated and will be
 removed in a future release. Please use '--slow-query-log'/'--slow-query-log-file'
 instead.
---- End output of /sbin/service mysqld start ----

/opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/mixin/command.rb:184:in `handle_command_failures'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/mixin/command.rb:131:in `run_command'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/provider/service/init.rb:37:in `start_service'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/provider/service.rb:60:in `action_start'
```

```
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/resource.rb:406:in `send'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/resource.rb:406:in `run_action'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/runner.rb:53:in `run_action'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/runner.rb:89:in `converge'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/runner.rb:89:in `each'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/runner.rb:89:in `converge'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/resource_collection.rb:94:in `execute_each_resource'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/resource_collection/stepable_iterator.rb:116:in `call'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/resource_collection/stepable_iterator.rb:116:in
 `call_iterator_block'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/resource_collection/stepable_iterator.rb:85:in `step'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/resource_collection/stepable_iterator.rb:104:in `iterate'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/resource_collection/stepable_iterator.rb:55:in
 `each_with_index'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/resource_collection.rb:92:in `execute_each_resource'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/runner.rb:84:in `converge'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/client.rb:268:in `converge'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/client.rb:158:in `run'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/application/solo.rb:190:in `run_application'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/application/solo.rb:181:in `loop'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/application/solo.rb:181:in `run_application'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/application.rb:62:in `run'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/chef-solo:25
```

```
/opt/aws/opsworks/current/bin/chef-solo:16:in `load'
/opt/aws/opsworks/current/bin/chef-solo:16
```

文件的末尾是 Chef 堆栈跟踪。您还应检查异常信息前面的输出内容，通常包含系统错误 (如 `package not available`)，该信息在确定故障原因时也很有用。在本例中，MySQL 守护程序启动失败。

常见 Chef 日志错误

下面是一些常见的 Chef 日志错误，并介绍了如何解决这些问题。

找不到日志

在 Chef 运行开始时，实例会收到一个预签名 Amazon S3 URL，该 URL 可让您在 Chef 运行完成后在网页上查看日志。由于此 URL 将在两小时后过期，因此，如果 Chef 运行时间超过两个小时，即使 Chef 运行期间没有出现任何问题，日志也不会上传到 Amazon S3 站点。创建日志的命令会成功，但仅可在实例上查看日志，而不能在预签名 URL 上查看。

日志意外结束

如果 Chef 日志意外结束，没有指示成功或显示错误信息，您可能遇到内存不足的状态，从而阻止了 Chef 完成日志。您最好使用更大的实例重新尝试。

缺少说明书或配方

如果 Chef 运行遇到说明书缓存内没有说明书或配方，您将看到与以下类似的内容：

```
DEBUG: Loading Recipe mycookbook::myrecipe via include_recipe
ERROR: Caught exception during execution of custom recipe: mycookbook::myrecipe:
   Cannot find a cookbook named mycookbook; did you forget to add metadata to a
 cookbook?
```

此条目表示说明书缓存中没有 `mycookbook` 说明书。使用 Chef 11.4，如果您没有在 `metadata.rb` 中正确声明依赖关系，也可能遇到此错误。

AWS OpsWorks Stacks 从实例的食谱缓存中运行食谱。它会在实例启动时将说明书从您的存储库下载到此缓存中。但是，如果您随后修改存储库中的食谱， AWS OpsWorks Stacks 不会自动更新在线实例上的缓存。如果您在实例启动后，修改了说明书或添加了新的说明书，请执行以下步骤：

1. 确保您已将更改提交到存储库。
2. 运行 Update Cookbooks 堆栈命令以使用存储库中最新版本更新说明书缓存。

本地命令失败

如果 Chef `execute` 资源无法执行指定的命令，您将看到与以下类似的内容：

```
DEBUG: ---- End output of ./configure --with-config-file-path=/ returned 2
ERROR: execute[PHP: ./configure] (/root/opsworks-agent/site-cookbooks/php-fpm/
recipes/install.rb line 48) had an error:
   ./configure --with-config-file-path=/
```

向上滚动日志，您应该可以看到命令的 `stderr` 和 `stdout` 输出，该信息有助于您确定命令失败的原因。

软件包故障

如果软件包安装失败，您将看到与以下类似的内容：

```
ERROR: package[zend-server-ce-php-5.3] (/root/opsworks-agent/site-cookbooks/
zend_server/recipes/install.rb line 20)
   had an error: apt-get -q -y --force-yes install zend-server-ce-php-5.3=5.0.4+b17
 returned 100, expected 0
```

向上滚动日志，您应该可以看到命令的 STDOUT 和 STDERROR 输出，该信息有助于您确定软件包安装失败的原因。

使用 AWS OpsWorks Stacks Agent CLI

Important

该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

Note

代理 CLI 仅适用于 Linux 实例。

在每个在线实例上， AWS OpsWorks Stacks 都会安装一个代理，该代理与服务进行通信。 AWS OpsWorks Stacks 服务反过来向代理发送命令以执行任务，例如在生命周期事件发生时在实例上启动 Chef 运行。在 Linux 实例中，代理公开了一个对故障排除非常有用的命令行界面 (CLI)。要运行代理 CLI 命令，请使用 SSH 连接到实例。然后，您就可以运行代理 CLI 命令来执行各种任务，包括以下任务：

- 执行配方。
- 显示 Chef 日志。
- 显示堆栈配置和部署 JSON。

有关如何设置与实例的 SSH 连接的更多信息，请参阅使用 SSH 登录。您还必须拥有堆栈的 SSH 和 Sudo 权限。

本部分介绍了如何使用代理 CLI 进行故障排除。有关更多信息和完整的命令参考，请参阅 AWS OpsWorks Stacks Agent CLI。

主题

执行配方

代理 CLI `run_command` 命令指示代理返回一个它之前执行过的命令。对故障排除最有用的命令为 `setup`、`configure`、`deploy` 和 `undeploy`，每个命令对应于一个生命周期事件。这些命令指示代理启动 Chef 运行以执行关联的配方。

> **Note**
>
> `run_command` 命令仅限于执行与指定命令关联的配方组，通常是与生命周期事件关联的配方。您不能用它来执行特定的配方。要执行一个或多个指定的配方，请使用 Execute Recipes 堆栈命令或等效的 CLI 或 API 操作（`create-deployment` 和 `CreateDeployment`）。

`run_command` 命令对调试自定义配方非常有用，尤其是分配到 Setup 和 Configure 生命周期事件的配方，您无法从控制台直接触发这些事件。通过使用 `run_command`，您可以随时依照您的需要来运行特定事件的配方，而无须启动或停止实例。

> **Note**
>
> AWS OpsWorks Stacks 从实例的食谱缓存中运行食谱，而不是食谱存储库。 AWS OpsWorks 实例启动时，Stacks 会将食谱下载到此缓存中，但如果您随后修改食谱，则不会自动更新在线实例上的缓存。如果您在实例启动后修改了您的说明书，请确保运行 Update Cookbooks 堆栈命令以使用存储库中最新版本更新说明书缓存。

代理仅缓存最近的命令。您可以通过运行 `list_commands` 列出这些命令，它会返回一个包含已缓存的命令和命令执行时间的列表。

```
sudo opsworks-agent-cli list_commands
2013-02-26T19:08:26        setup
2013-02-26T19:12:01        configure
2013-02-26T19:12:05        configure
2013-02-26T19:22:12        deploy
```

要重新运行最近的命令，请运行：

```
sudo opsworks-agent-cli run_command
```

要重新运行最近实例的指定命令，请运行：

```
sudo opsworks-agent-cli run_command command
```

例如，要重新运行 Setup 配方，您可以运行以下命令：

```
sudo opsworks-agent-cli run_command setup
```

每个命令都有一个关联的堆栈配置和部署 JSON，表示该命令运行时堆栈和部署的状态。由于这些数据可能会因命令而异，因此，较早实例的命令与最近的命令可能使用略微不同的数据。要重新运行特定实例的命令，请复制 `list_commands` 输出中的时间，并运行以下命令：

```
sudo opsworks-agent-cli run_command time
```

前面的示例均使用默认的 JSON 重新运行命令，即为该命令安装的 JSON。您可以使用任意 JSON 文件重新运行命令，如下所示：

```
sudo opsworks-agent-cli run_command -f /path/to/valid/json.file
```

显示 Chef 日志

代理 CLI show_log 命令显示一个指定的日志。在该命令执行完毕后，您将看到文件结尾。因此，show_log 命令提供了一种查看日志结尾的简便方法，您通常会在这里找到错误信息。您可以向上滚动以查看日志的前面部分。

要显示当前命令的日志，请运行以下命令：

```
sudo opsworks-agent-cli show_log
```

您也可以显示特定命令的日志，但请注意，代理仅缓存最后 30 个命令的日志。您可以通过运行 list_commands，列出某个实例的命令，该命令返回一个包含已缓存的命令和命令执行时间的列表。有关示例，请参阅执行配方。

要显示特定命令最近执行情况的日志，请运行以下命令：

```
sudo opsworks-agent-cli show_log command
```

该命令参数可设置为 setup、configure、deploy、undeploy、start、stop 或 restart。大多数这些命令对应于生命周期事件，指示代理运行关联的配方。

要显示特定命令执行情况的日志，请复制 list_commands 输出中的日期并运行：

```
sudo opsworks-agent-cli show_log date
```

如果命令仍在执行中，show_log 会显示日志的当前状态。

> **Note**
>
> 解决错误和 out-of-memory问题的一种方法是在执行期间跟踪日志，如下所示：show_log
>
> 1. 使用 run_command 来触发相应的生命周期事件。有关更多信息，请参阅 执行配方。
> 2. 重复运行 show_log 以在日志写入时查看日志的结尾部分。
>
> 如果 Chef 运行的内存不足或意外退出，则日志将突然结束。如果配方失败，则日志的结尾将包含一个异常信息和堆栈跟踪。

显示堆栈配置和部署 JSON

配方使用的大部分数据来自堆栈配置和部署 JSON，它定义了一组 Chef 属性，提供堆栈配置、任何部署和用户可添加的可选自定义属性的详细描述。对于每个命令， AWS OpsWorks Stacks 都会安装一个 JSON，该JSON表示命令执行时的堆栈和部署状态。有关更多信息，请参阅 堆栈配置和部署属性。

如果您的自定义配方从堆栈配置和部署 JSON 中获取数据，您可以通过检查该 JSON 来验证数据。显示堆栈配置和部署 JSON 最简单的方式是运行代理 CLI `get_json` 命令，该命令将显示格式化版本的 JSON 对象。下面显示了某些典型输出的前几行内容：

```
{
  "opsworks": {
    "layers": {
      "php-app": {
        "id": "4a2a56c8-f909-4b39-81f8-556536d20648",
        "instances": {
          "php-app2": {
            "elastic_ip": null,
            "region": "us-west-2",
            "booted_at": "2013-02-26T20:41:10+00:00",
            "ip": "10.112.235.192",
            "aws_instance_id": "i-34037f06",
            "availability_zone": "us-west-2a",
            "instance_type": "c1.medium",
            "private_dns_name": "ip-10-252-0-203.us-west-2.compute.internal",
            "private_ip": "10.252.0.203",
            "created_at": "2013-02-26T20:39:39+00:00",
            "status": "online",
            "backends": 8,
            "public_dns_name": "ec2-10-112-235-192.us-west-2.compute.amazonaws.com"
...
```

您可以显示最近的堆栈配置和部署 JSON，如下所示：

```
sudo opsworks-agent-cli get_json
```

通过执行以下命令，您可以显示特定命令最近的堆栈配置和部署 JSON：

```
sudo opsworks-agent-cli get_json command
```

该命令参数可设置为 `setup`、`configure`、`deploy`、`undeploy`、`start`、`stop` 或 `restart`。大多数这些命令对应于生命周期事件，指示代理运行关联的配方。

通过指定命令的日期，您可以显示特定命令执行的堆栈配置和部署 JSON，如下所示：

```
sudo opsworks-agent-cli get_json date
```

使用此命令的最简单方法如下所示：

1. 运行 `list_commands`，该命令会返回一个列表，其中包含已在实例上运行的命令以及每个命令的运行日期。
2. 复制相应命令的日期，并将该日期用作 `get_json` *`date`* 参数。

常见的调试和故障排除问题

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

这部分介绍一些常见的调试和故障排除问题以及它们的解决方案。

主题

- AWS OpsWorks 堆栈故障排除
- 对实例注册进行故障排除

AWS OpsWorks 堆栈故障排除

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

本节包含一些常见的 AWS OpsWorks Stacks 问题及其解决方案。

主题

无法管理实例

问题：您无法继续管理过去可管理的实例。在某些情况下，日志可能会显示与以下内容相似的错误。

```
Aws::CharlieInstanceService::Errors::UnrecognizedClientException - The security token
 included in the request is invalid.
```

原因：如果编辑或删除实例所依赖的 AWS OpsWorks 以外的资源，则会发生此错误。下面是一些关于可中断与实例的通信的资源变更的示例。

- 与该实例关联的 IAM 用户或角色在 AWS OpsWorks 堆栈之外被意外删除。这会导致安装在实例上的 AWS OpsWorks 代理与 AWS OpsWorks Stacks 服务之间的通信失败。在实例的整个生命周期中，都需要与实例关联的 用户。
- 在实例处于离线状态时编辑卷或存储配置可能会导致实例不可管理。
- 手动向弹性负载均衡添加 EC2 实例。 AWS OpsWorks 每次实例进入或离开在线状态时，都会重新配置分配的 Elastic Load Balancing 负载均衡器。 AWS OpsWorks 仅将其知道的实例视为有效成

员；在其他进程之 AWS OpsWorks外或由其他进程添加的实例将被删除。其他所有实例都将被删除。

解决方案：不要删除您的实例所依赖的 IAM 用户或角色。如果可以，仅当从属实例运行时编辑卷或存储配置。 AWS OpsWorks 用于管理实例的负载均衡器或 EIP 成员资格。 AWS OpsWorks 当您注册实例时，为了防止出现意外删除用户后无法管理注册的实例的情况，可将 `--use-instance-profile` 参数添加到您的 `register` 命令，以使用实例的内置实例配置文件。

Chef 运行后，实例不启动

问题：在 Chef 11.10 或配置为使用自定义说明书的较早版本的堆栈上，当 Chef 运行 (使用社区说明书) 后，实例不启动。日志消息可表明配方无法编译 (“配方编译错误”)，或无法负载，因为它们找不到依赖项。

原因：最可能的原因是自定义或社区说明书不支持您的堆栈使用的 Chef 版本。一些常见的社区说明书 (如 `apt` 和 `build-essential`) 与 Chef 11.10 之间存在已知的兼容性问题。

解决方案：在开启了 “使用自定义 Chef 食谱” 设置的 Stacks AWS OpsWorks 堆栈上，自定义或社区食谱必须始终支持堆栈使用的 Chef 版本。将社区说明书固定到一个与在您的堆栈设置中配置的 Chef 版本兼容的版本 (即，将说明书版本号设置为特定版本)。要查找受支持的社区说明书版本，请查看无法编译的说明书的变更日志，并仅使用您的堆栈将支持的说明书的最新版本。要固定说明书版本，请在您的自定义说明书存储库的 Berksfile 中指定精确的版本号。例如，`cookbook 'build-essential', '= 3.2.0'`。

层的所有实例都没有通过 Elastic Load Balancing 状态检查

问题：您将 Elastic Load Balancing 负载均衡器连接到某个应用程序服务器层，但所有实例都没有通过状态检查。

原因：当您创建 Elastic Load Balancing 负载均衡器时，您必须指定负载均衡器调用的 ping 路径，以确定实例的状态是否良好。请确保指定适用于您的应用程序的 ping 路径，默认值是 /index.html。如果您的应用程序不包括 `index.html`，您必须指定适当的路径，否则将无法通过状态检查。例如，Chef 11 Linux 堆栈入门 中使用的 SimplePHPApp 应用程序不使用 index.html；这些服务器的适当的 ping 路径是 /。

解决方案：编辑负载均衡器的 ping 路径。有关更多信息，请参阅 Elastic Load Balancing

无法与 Elastic Load Balancing 负载均衡器通信

问题：您创建一个 Elastic Load Balancing 负载均衡器并将它连接到一个应用程序服务器层，但当您单击负载均衡器的 DNS 名称或 IP 地址以运行该应用程序时，您得到以下错误：“The remote server is not responding (远程服务器未响应)”。

原因：如果您的堆栈正在默认 VPC 中运行，则当您在区域中创建 Elastic Load Balancing 负载均衡器时，您必须指定安全组。该安全组必须拥有入口规则，这些规则允许来自您的 IP 地址的入站流量。如果您指定 default VPC security group，则默认入口规则不接受任何入站流量。

解决方案：编辑安全组入口规则，以接受来自适当 IP 地址的入站流量。

1. 在 Amazon EC2 控制台的导航窗格中，单击 Security Groups。
2. 选择负载均衡器的安全组。
3. 单击 Inbound 选项卡上的 Edit。
4. 添加入口规则，并将 Source 设置为适当的 CIDR。

   例如，指定 Anywhere 会将 CIDR 设置为 0.0.0.0/0，这会指示负载均衡器接受来自任何 IP 地址的传入流量。

导入的本地实例重启后无法完成卷设置

问题：您重新启动已导入到 AWS OpsWorks 堆栈中的 EC2 实例， AWS OpsWorks Stacks 控制台将实例状态显示为失败。Chef 11 或 Chef 12 实例上都会发生此问题。

原因：AWS OpsWorks Stacks 在设置过程中可能无法将卷连接到您的实例。一种可能的原因是：当您运行 `setup` 命令时， AWS OpsWorks Stacks 覆盖了您的实例上的卷配置。

解决方案：打开实例的 Details 页面，检查 Volumes 区域中的卷配置。请注意，您只有在实例处于 stopped 状态时才能更改您的卷配置。请确保每个卷都拥有指定的挂载点和名称。在重启实例之前，请确认您在 AWS OpsWorks Stacks 的配置中提供了正确的挂载点。

重启后 EBS 卷不重新连接

问题：您使用 Amazon EC2 控制台将 Amazon EBS 卷连接到实例，但是当您重启实例时，卷无法连接。

原因： AWS OpsWorks 堆栈只能重新连接它知道的 Amazon EBS 卷，这些卷仅限于以下内容：

- 由 AWS OpsWorks Stacks 创建的卷。

- 您账户中的卷，您已明确使用 Resources 页面向堆栈注册这些卷。

解决方案：只能使用 AWS OpsWorks 堆栈控制台、API 或 CLI 来管理您的 Amazon EBS 卷。如果您想对一个堆栈使用您账户的 Amazon EBS 卷之一，请使用该堆栈的 资源 页面注册该卷并将其附加到实例。有关更多信息，请参阅 资源管理。

无法删除 AWS OpsWorks Stacks 安全组

问题：删除堆栈后，仍有许多 AWS OpsWorks 堆栈安全组无法删除。

原因：必须按照特定的顺序删除安全组。

解决方案：首先，确保任何实例都未使用安全组。然后，按照下面的顺序删除以下任何安全组 (如果存在)。

1. AWS-OpsWorks-空白服务器
2. AWS OpsWorks-监控主服务器
3. AWS-db OpsWorks-Master-Server
4. AWS--Memcached OpsWorks-Server
5. AWS-OpsWorks-自定义服务器
6. AWS-nodej OpsWorks s-App-Server
7. AWS--php-App OpsWorks-Server
8. AWS-Rails-App OpsWorks-Server
9. AWS OpsWorks-网络服务器
10. AWS-OpsWorks-默认服务器
11. AWS OpsWorks--LB-Server

意外删除了 AWS OpsWorks Stacks 安全组

问题：您删除了其中一个 AWS OpsWorks Stacks 安全组，需要重新创建它。

原因：这些安全组通常是无意中被删除的。

解决方案：重新创建的组必须与原始组完全相同，包括组名称的大写字母也要相同。首选的方法不是手动重新创建组，而是让 AWS OpsWorks Stacks 为您执行此任务。只需在同一 AWS 区域和 VPC（如果有）中创建一个新堆栈， AWS OpsWorks Stacks 就会自动重新创建所有内置安全组，包括您删除的安全组。如果您不再需要该堆栈，则随后您可以删除它；安全组将保留。

Chef 日志突然终止

问题：Chef 日志突然终止；日志结尾未指示运行成功，也未显示异常情况和堆栈跟踪。

原因：此行为通常是由于内存不足造成的。

解决方案：创建一个较大的实例，然后使用代理 CLI `run_command` 命令再次运行配方。有关更多信息，请参阅 执行配方。

说明书未更新

问题：您更新了说明书，但是堆栈的实例仍然运行旧的配方。

原因： AWS OpsWorks Stacks 在每个实例上缓存食谱，并从缓存而不是存储库中运行食谱。当您启动新实例时， AWS OpsWorks Stacks 会将您的食谱从存储库下载到实例的缓存中。但是，如果您随后修改您的自定义说明书，则 AWS OpsWorks Stacks 不会自动更新联机实例的缓存。

解决方案：运行 U pdate Cookbooks 堆栈命令，明确指示 AWS OpsWorks Stacks 更新在线实例的食谱缓存。

实例停滞在启动状态

问题：当您重启实例或自动修复自动重启实例时，启动操作停滞在 `booting` 状态。

原因：引发此问题的一种可能原因是 VPC 配置，包括默认 VPC。实例必须始终能够与 AWS OpsWorks Stacks 服务、Amazon S3 以及软件包、食谱和应用程序存储库进行通信。例如，如果您从默认 VPC 中移除默认网关，则这些实例将失去与 AWS OpsWorks Stacks 服务的连接。由于 AWS OpsWorks Stacks 无法再与实例代理通信，因此它会将实例视为失败并且 auto 对其进行修复。但是，如果没有连接， AWS OpsWorks Stacks 就无法在治疗过的实例上安装实例代理。如果没有代理， AWS OpsWorks Stacks 就无法在实例上运行安装配方，因此启动操作无法超出 “启动” 状态。

解决方案：修改您的 VPC 配置，以便实例拥有所需的连接。

实例意外重启

问题：已经停止的实例意外重启。

原因 1：如果您为实例启用了自动修复功能，则 AWS OpsWorks Stacks 会定期对关联的 Amazon EC2 实例执行状态检查，并重启状态不佳的实例。如果您使用 Amazon EC2 控制台、API 或 CLI 停止或终止 AWS OpsWorks 堆栈管理的实例，则不会通知 AWS OpsWorks 堆栈。它会认为已停止的实例状态不佳，并自动重启该实例。

解决方案：仅使用 AWS OpsWorks Stacks 控制台、API 或 CLI 管理您的实例。如果您使用 AWS OpsWorks Stacks 停止或删除实例，则该实例不会重启。有关更多信息，请参阅 手动启动、停止和重启全天候实例 和 删除 AWS OpsWorks 堆栈实例。

原因 2：实例可能会因为各种原因而失败。如果您启用了自动修复， AWS OpsWorks Stacks 会自动重启失败的实例。

解决方案：这是正常操作；除非您不希望 AWS OpsWorks Stacks 重启失败的实例，否则无需执行任何操作。在这种情况下，您应当禁用自动修复功能。

正在对实例运行 **opsworks-agent** 进程

问题：正在对您的实例运行多个 opsworks-agent 进程。例如：

```
aws 24543 0.0 1.3 172360 53332 ? S Feb24 0:29 opsworks-agent: master 24543
aws 24545 0.1 2.0 208932 79224 ? S Feb24 22:02 opsworks-agent: keep_alive of master
 24543
aws 24557 0.0 2.0 209012 79412 ? S Feb24 8:04 opsworks-agent: statistics of master
 24543
aws 24559 0.0 2.2 216604 86992 ? S Feb24 4:14 opsworks-agent: process_command of master
 24
```

原因：这些是执行代理的常规操作所需的合法进程。它们执行诸如处理部署以及将保持活动状态的消息发送回服务等任务。

解决方案：这是常规行为。不要停止这些进程；这样会影响代理的操作。

意外的 execute_recipes 命令

问题：实例的详细信息页面上的 Logs 部分包含意外的 execute_recipes 命令。意外的 execute_recipes 命令还可能显示在 Stack 和 Deployments 页面上。

原因：此问题通常是由权限变更导致的。当您更改用户或组的 SSH 或 sudo 权限时， AWS OpsWorks Stacks 会运行execute_recipes以更新堆栈的实例，还会触发配置事件。出现 execute_recipes 命令的另一个原因是 AWS OpsWorks Stacks 更新了实例代理。

解决方案：这是常规操作；无需执行其任何他操作。

要了解 execute_recipes 命令执行了哪些操作，请转到 Deployments 页面并单击该命令的时间戳。这会打开该命令的详细信息页面，上面列出了已经运行的关键配方。例如，以下详细信息页面对应的是运行了 ssh_users 以更新 SSH 权限的 execute_recipes 命令。

要查看所有详细信息，单击该命令的 Log 列中的 show 以显示关联的 Chef 日志。在日志中搜索**Run List**。 AWS OpsWorks 堆栈维护配方将出炉。`OpsWorks Custom Run List`例如，下面是上文中的屏幕截图中显示的 `execute_recipes` 命令的运行列表，其中显示了与该命令相关联的每个配方。

```
[2014-02-21T17:16:30+00:00] INFO: OpsWorks Custom Run List:
 ["opsworks_stack_state_sync",
  "ssh_users", "test_suite", "opsworks_cleanup"]
```

对实例注册进行故障排除

这部分包含一些常见的实例注册问题及其解决方案。

> **Note**
>
> 如果您遇到了注册问题，请使用 `--debug` 参数运行 `register`，其中提供了更多调试信息。

主题

- EC2User 未获得授权执行：.....
- 凭证应当局限于有效区域

EC2User 未获得授权执行：.....

问题：`register` 命令返回如下内容：

```
A client error (AccessDenied) occurred when calling the CreateGroup operation:
```

```
User: arn:aws:iam::123456789012:user/ImportEC2User is not authorized to
perform: iam:CreateGroup on resource:
arn:aws:iam::123456789012:group/AWS/OpsWorks/OpsWorks-b583ce55-1d01-4695-b3e5-
ee19257d1911
```

原因： `register` 命令运行时使用的凭证未授予所需的权限。该用户的策略必须允许执行 `iam:CreateGroup` 等操作。

解决方案：向 `register` 提供拥有所需权限的 IAM 用户凭证。有关更多信息，请参阅 安装和配置 AWS CLI。

凭证应当局限于有效区域

问题：`register` 命令返回以下内容：

```
A client error (InvalidSignatureException) occurred when calling the
DescribeStacks operation: Credential should be scoped to a valid region, not 'cn-
north-1'.
```

原因：该命令的区域必须是有效的 AWS OpsWorks Stacks 区域。有关受支持的区域的列表，请参阅“区域支持”。此错误通常是由以下原因之一导致的：

- 堆栈位于其他区域，而您将堆栈的区域分配给了该命令的 `--region` 参数。

  您无需指定堆栈区域；Stac AWS OpsWorks ks 会根据堆栈 ID 自动确定堆栈区域。
- 您省略了隐式指定默认区域的 `--region` 参数，但 AWS OpsWorks Stacks 不支持您的默认区域。

解决方案：明确设置`--region`为支持的 AWS OpsWorks 堆栈区域，或者编辑您的 AWS CLI `config`文件以将默认区域更改为支持的 AWS OpsWorks 堆栈区域。有关更多信息，请参阅配置 AWS 命令行界面。

AWS OpsWorks Stacks Agent CLI

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

> **Note**
>
> 此功能仅适用于 Linux 实例。

AWS OpsWorks Stacks 在每个实例上安装的代理会公开命令行界面 (CLI)。如果您使用 SSH 登录到实例，就可以使用该 CLI 完成以下工作：

- 访问 Chef run 的日志文件。
- 访问 AWS OpsWorks 堆栈命令。
- 手动运行 Chef 配方。
- 查看实例报告。
- 查看代理报告。
- 查看一组有限的堆栈配置和部署属性。

> **Important**
>
> 只能以根用户身份或使用 sudo 运行代理 CLI 命令。

基本的命令语法是：

```
sudo opsworks-agent-cli [--help] [command [activity] [date]]
```

四个参数如下：

help

(可选) 单独使用时显示可用命令的简短概要。与一条命令结合使用时，help 显示该命令的说明。

command

(可选) 代理 CLI 命令，必须设置为以下值之一：

- agent_report
- get_json
- instance_report

- list_commands
- run_command
- show_log
- stack_state

活动

（可选）用作某些命令的参数来指定特定的 AWS OpsWorks Stacks 活动：`setup`、`configure`、`deploy`、`undeploy`、`start`、`stop` 或 `restart`。

date

(可选) 用作某些命令的参数来指定特定的 AWS OpsWorks Stacks 命令执行。通过将日期设置为以 t *`yyyy-mm-ddhh: mm: ss`* 格式执行命令的时间戳（包括单引号）来指定命令的执行。例如，对于 2013 年 2 月 5 日星期二 10:31:55，使用：`'2013-02-05T10:31:55'`。要确定何时执行特定 AWS OpsWorks Stacks 命令，请运行 list_commands。

> **Note**
>
> 如果代理多次执行同一 AWS OpsWorks Stacks 活动，则可以通过指定活动及其执行时间来选择特定的执行。如果您指定了活动而省略了时间，则代理 CLI 命令作用于活动的最近一次执行。如果同时省略这两个参数，则代理 CLI 命令作用于最近的活动。

下列部分描述命令以及其相关参数。为简便起见，语法部分省略了可选的 `--help` 选项，该选项可与任何命令结合使用。

主题

- agent_report
- get_json
- instance_report
- list_commands
- run_command
- show_log
- stack_state

agent_report

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

返回代理报告。

```
sudo opsworks-agent-cli agent_report
```

以下输出示例来自最近运行了配置活动的实例。

```
$ sudo opsworks-agent-cli agent_report

AWS OpsWorks Instance Agent State Report:

  Last activity was a "configure" on 2015-12-01 18:19:23 UTC
  Agent Status: The AWS OpsWorks agent is running as PID 30998
  Agent Version: 4004-20151201152533, up to date
```

get_json

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

以 JSON 对象形式返回有关 Chef run 的信息。

```
sudo opsworks-agent-cli get_json [activity] [date] [-i | --internal | --no-i | --no-
internal]
```

默认情况下，`get_json` 显示最近 Chef run 的客户提供信息。使用以下选项来指定一组特定的信息。

活动

显示与最近指定的活动关联的 Chef run 的信息。要获取有效活动的列表，请运行 list_commands。

date

显示与针对指定时间戳而执行的活动关联的 Chef run 的信息。要获取有效时间戳的列表，请运行 list_commands。

-i、--internal

显示 AWS OpsWorks Stacks 内部在 Chef 运行时使用的信息。

--no-i、--no-internal

显式显示 Chef run 的客户提供信息。如果未另行指定，这就是默认值。

> **Note**
>
> 对于 Chef 12 Linux 实例，运行此命令将返回有效的信息，例如实例的堆栈配置和部署属性。但是，要获得更完整的信息，请参阅 AWS OpsWorks Stacks 在实例上创建的 Chef 数据包。有关更多信息，请参阅 AWS OpsWorks 堆栈数据包参考。

以下输出示例显示最近配置活动的最近 Chef run 的客户提供信息。

```
$ sudo opsworks-agent-cli get_json configure

{
  "run_list": [
    "recipe[opsworks_cookbook_demo::configure]"
  ]
}
```

以下输出示例显示了 AWS OpsWorks Stacks 在内部使用的信息，用于在指定时间戳内执行的 Chef 运行。

```
$ sudo opsworks-agent-cli get_json 2015-12-01T18:20:24 -i

{
  "aws_opsworks_agent": {
    "version": "4004-20151201152533",
```

```
    "valid_client_activities": [
      "reboot",
      "stop",
      "deploy",
      "grant_remote_access",
      "revoke_remote_access",
      "update_agent",
      "setup",
      "configure",
      "update_dependencies",
      "install_dependencies",
      "update_custom_cookbooks",
      "execute_recipes",
      "sync_remote_users"
    ],
    "command": {
      "type": "configure",
      "args": {
        "app_ids": [

        ]
      },
      "sent_at": "2015-12-01T18:19:23+00:00",
      "command_id": "5c2113f3-c6d5-40eb-bcfa-77da2885eeEX",
      "iam_user_arn": null,
      "instance_id": "cfdaa716-42fe-4e3b-9762-fef184ddd8EX"
    },
    "resources": {
      "apps": [

      ],
      "layers": [
        {
          "layer_id": "93f50d83-1e73-45c4-840a-0d4f07cda1EX",
          "name": "MyCookbooksDemoLayer",
          "packages": [

          ],
          "shortname": "cookbooks-demo",
          "type": "custom",
          "volume_configurations": [

          ]
        }
```

```
      ],
      "instances": [
        {
          "ami_id": "ami-d93622EX",
          "architecture": "x86_64",
          "auto_scaling_type": null,
          "availability_zone": "us-west-2a",
          "created_at": "2015-11-18T00:21:05+00:00",
          "ebs_optimized": false,
          "ec2_instance_id": "i-a480e960",
          "elastic_ip": null,
          "hostname": "cookbooks-demo1",
          "instance_id": "cfdaa716-42fe-4e3b-9762-fef184ddd8EX",
          "instance_type": "c3.large",
          "layer_ids": [
            "93f50d83-1e73-45c4-840a-0d4f07cda1EX"
          ],
          "os": "Amazon Linux 2015.09",
          "private_dns": "ip-192-0-2-0.us-west-2.compute.internal",
          "private_ip": "10.122.69.33",
          "public_dns": "ec2-203-0-113-0.us-west-2.compute.amazonaws.com",
          "public_ip": "192.0.2.0",
          "root_device_type": "ebs",
          "root_device_volume_id": "vol-f6f7e8EX",
          "ssh_host_dsa_key_fingerprint": "f2:...:15",
          "ssh_host_dsa_key_public": "ssh-dss AAAAB3Nz...a8vMbqA=",
          "ssh_host_rsa_key_fingerprint": "0a:...:96",
          "ssh_host_rsa_key_public": "ssh-rsa AAAAB3Nz...yhPanvo7",
          "status": "online",
          "subnet_id": null,
          "virtualization_type": "paravirtual",
          "infrastructure_class": "ec2",
          "ssh_host_dsa_key_private": "-----BEGIN DSA PRIVATE KEY-----
\nMIIDVwIB...g5OtgQ==\n-----END DSA PRIVATE KEY-----\n",
          "ssh_host_rsa_key_private": "-----BEGIN RSA PRIVATE KEY-----
\nMIIEowIB...78kprtIw\n-----END RSA PRIVATE KEY-----\n"
        }
      ],
      "users": [

      ],
      "elastic_load_balancers": [

      ],
```

```
      "rds_db_instances": [

      ],
      "stack": {
        "arn": "arn:aws:opsworks:us-west-2:80398EXAMPLE:stack/040c3def-b2b4-4489-bb1b-
e08425886fEX/",
        "custom_cookbooks_source": {
          "type": "s3",
          "url": "https://s3.amazonaws.com/opsworks-demo-bucket/opsworks-cookbook-
demo.tar.gz",
          "username": "AKIAJUQN...WG644EXA",
          "password": "O5v+4Zz+...rcKbFTJu",
          "ssh_key": null,
          "revision": null
        },
        "name": "MyCookbooksDemoStack",
        "region": "us-west-2",
        "stack_id": "040c3def-b2b4-4489-bb1b-e08425886fEX",
        "use_custom_cookbooks": true,
        "vpc_id": null
      },
      "ecs_clusters": [

      ],
      "volumes": [

      ]
    },
    "chef": {
      "customer_recipes": [
        "opsworks_cookbook_demo::configure"
      ],
      "customer_json": "e30=\n",
      "customer_data_bags": "e30=\n"
    }
  }
}
```

instance_report

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

返回扩展的实例报告。

```
sudo opsworks-agent-cli instance_report
```

以下输出示例来自一个实例。

```
$ sudo opsworks-agent-cli instance_report

AWS OpsWorks Instance Agent State Report:

  Last activity was a "configure" on 2015-12-01 18:19:23 UTC
  Agent Status: The AWS OpsWorks agent is running as PID 30998
  Agent Version: 4004-20151201152533, up to date
  OpsWorks Stack: MyCookbooksDemoStack
  OpsWorks Layers: MyCookbooksDemoLayer
  OpsWorks Instance: cookbooks-demo1
  EC2 Instance ID: i-a480e9EX
  EC2 Instance Type: c3.large
  Architecture: x86_64
  Total Memory: 3.84 Gb
  CPU: 2x  Intel(R) Xeon(R) CPU E5-2680 v2 @ 2.80GHz

Location:

  EC2 Region: us-west-2
  EC2 Availability Zone: us-west-2a

Networking:

  Public IP: 192.0.2.0
  Private IP: 198.51.100.0
```

list_commands

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

列出在此实例上执行的每个活动的时间。您可以将这些时间用于其他代理 CLI 命令来指定特定的执行。

```
sudo opsworks-agent-cli list_commands  [activity] [date]
```

以下输出示例来自运行了配置、设置和更新自定义说明书活动的实例。

```
$ sudo opsworks-agent-cli list_commands

2015-11-24T21:00:28        update_custom_cookbooks
2015-12-01T18:19:09        setup
2015-12-01T18:20:24        configure
```

run_command

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

运行 AWS OpsWorks Stacks 命令，该命令是一个 JSON 文件，其中包含一个 Chef 运行列表，其中包含执行 AWS OpsWorks 堆栈活动所需的信息（设置、配置、部署等）。run_command 命令会生成一个日志条目，可以通过运行 show_log 来查看。此选项仅用于开发目的，因此 AWS OpsWorks Stacks 不会跟踪更改。

```
sudo opsworks-agent-cli run_command [activity] [date] [/path/to/valid/json.file]
```

默认情况下，run_command运行最新的 AWS OpsWorks 堆栈命令。使用以下选项指定特定的命令。

活动

运行指定的 AWS OpsWorks Stacks 命令：setupconfigure、deploy、、undeploy、startstop、或restart。

date

运行在指定时间戳执行的 AWS OpsWorks 命令。要获取有效时间戳的列表，请运行 list_commands。

file

运行指定命令的 JSON 文件。要获取命令的文件路径，请运行 get_json。

以下输出示例来自运行了配置命令的实例。

```
$ sudo opsworks-agent-cli run_command configure

[2015-12-02 16:52:53]  INFO [opsworks-agent(21970)]: About to re-run 'configure' from
 2015-12-01T18:20:24
...
[2015-12-02 16:53:02]  INFO [opsworks-agent(21970)]: Finished Chef run with exitcode 0
```

show_log

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

返回命令的日志文件。

```
sudo opsworks-agent-cli show_log [activity] [date]
```

默认情况下，show_log 跟踪最近的日志文件。使用以下选项指定特定的命令。

活动

显示指定活动的日志文件。

date

显示在指定时间戳执行的活动的日志文件。要获取有效时间戳的列表，请运行 list_commands。

以下输出示例显示了最新的日志。

```
$ sudo opsworks-agent-cli show_log

[2015-12-02T16:52:59+00:00] INFO: Storing updated cookbooks/opsworks_cookbook_demo/
opsworks-cookbook-demo.tar.gz in the cache.
...
[2015-12-02T16:52:59+00:00] INFO: Report handlers complete
```

stack_state

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

显示 AWS OpsWorks Stacks 内部在最近运行 Chef 时使用的信息。

```
opsworks-agent-cli stack_state
```

> **Note**
>
> 对于 Chef 12 Linux 实例，运行此命令将返回有效的信息，例如实例的堆栈配置和部署属性。但是，要获得更完整的信息，请参阅 AWS OpsWorks Stacks 在实例上创建的 Chef 数据包。有关更多信息，请参阅 AWS OpsWorks 堆栈数据包参考。

以下输出示例来自一个实例。

```
$ sudo opsworks-agent-cli stack_state
```

```
{
  "last_command": {
    "sent_at": "2015-12-01T18:19:23+00:00",
    "activity": "configure"
  },
  "instance": {
    "ami_id": "ami-d93622EX",
    "architecture": "x86_64",
    "auto_scaling_type": null,
    "availability_zone": "us-west-2a",
    "created_at": "2015-11-18T00:21:05+00:00",
    "ebs_optimized": false,
    "ec2_instance_id": "i-a480e9EX",
    "elastic_ip": null,
    "hostname": "cookbooks-demo1",
    "instance_id": "cfdaa716-42fe-4e3b-9762-fef184ddd8EX",
    "instance_type": "c3.large",
    "layer_ids": [
      "93f50d83-1e73-45c4-840a-0d4f07cda1EX"
    ],
    "os": "Amazon Linux 2015.09",
    "private_dns": "ip-192-0-2-0.us-west-2.compute.internal",
    "private_ip": "10.122.69.33",
    "public_dns": "ec2-203-0-113-0.us-west-2.compute.amazonaws.com",
    "public_ip": "192.0.2.0",
    "root_device_type": "ebs",
    "root_device_volume_id": "vol-f6f7e8EX",
    "ssh_host_dsa_key_fingerprint": "f2:...:15",
    "ssh_host_dsa_key_public": "ssh-dss AAAAB3Nz...a8vMbqA=",
    "ssh_host_rsa_key_fingerprint": "0a:...:96",
    "ssh_host_rsa_key_public": "ssh-rsa AAAAB3Nz...yhPanvo7",
    "status": "online",
    "subnet_id": null,
    "virtualization_type": "paravirtual",
    "infrastructure_class": "ec2",
    "ssh_host_dsa_key_private": "-----BEGIN DSA PRIVATE KEY-----\nMIIDVwIB...g5OtgQ==
\n-----END DSA PRIVATE KEY-----\n",
    "ssh_host_rsa_key_private": "-----BEGIN RSA PRIVATE KEY-----\nMIIEowIB...78kprtIw
\n-----END RSA PRIVATE KEY-----\n"
  },
  "layers": [
    {
      "layer_id": "93f50d83-1e73-45c4-840a-0d4f07cda1EX",
```

```
      "name": "MyCookbooksDemoLayer",
      "packages": [

      ],
      "shortname": "cookbooks-demo",
      "type": "custom",
      "volume_configurations": [

      ]
    }
  ],
  "applications": null,
  "stack": {
    "arn": "arn:aws:opsworks:us-west-2:80398EXAMPLE:stack/040c3def-b2b4-4489-bb1b-
e08425886fEX/",
    "custom_cookbooks_source": {
      "type": "s3",
      "url": "https://s3.amazonaws.com/opsworks-demo-bucket/opsworks-cookbook-
demo.tar.gz",
      "username": "AKIAJUQN...WG644EXA",
      "password": "O5v+4Zz+...rcKbFTJu",
      "ssh_key": null,
      "revision": null
    },
    "name": "MyCookbooksDemoStack",
    "region": "us-west-2",
    "stack_id": "040c3def-b2b4-4489-bb1b-e08425886fEX",
    "use_custom_cookbooks": true,
    "vpc_id": null
  },
  "agent": {
    "valid_activities": [
      "reboot",
      "stop",
      "deploy",
      "grant_remote_access",
      "revoke_remote_access",
      "update_agent",
      "setup",
      "configure",
      "update_dependencies",
      "install_dependencies",
      "update_custom_cookbooks",
      "execute_recipes",
```

```
      "sync_remote_users"
    ]
  }
}
```

AWS OpsWorks 堆栈数据包参考

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

AWS OpsWorks Stacks 为食谱提供了各种各样的设置作为厨师数据包的内容。本参考列出了此数据包内容。

数据包是一个 Chef 概念。数据包是一种全局变量，在实例上存储为 JSON 数据；可从 Chef 访问该 JSON 数据。例如，数据袋可以存储全局变量，例如应用程序的源 URL、实例的主机名和关联堆栈的 VPC 标识符。 AWS OpsWorks 堆栈将其数据包存储在每个堆栈的实例上。在 Linux 实例上， AWS OpsWorks Stacks 将数据包存储在`/var/chef/runs/`*`run-ID`*`/data_bags`目录中。在 Windows 实例上，它将数据包存储在 *`drive`*`:\chef\runs\`*`run-id`*`\data_bags` 目录中。在这两种情况下，*`run-ID`* 都是 AWS OpsWorks Stacks 分配给在实例上运行的每个 Chef 的唯一 ID。这些目录包含一组数据包（子目录）。每个数据包均包含零个或多个数据包项目，这些项目是 JSON 格式的文件，其中包含一系列数据包内容。

> **Note**
>
> AWS OpsWorks 堆栈不支持加密的数据包。要以加密形式存储敏感数据，例如密码或证书，我们建议将其存储在私有 S3 存储桶中。然后，您可以创建一个使用适用于 Ruby 的 Amazon SDK 来检索数据的自定义配方。有关示例，请参阅使用适用于 Ruby 的 SDK。

数据包内容可包括以下任一类型：

- 字符串内容，遵循标准 Ruby 语法，可使用单引号或双引号，但其中包含某些特殊字符的字符串必须用双引号引起来。有关更多信息，请转到 Ruby 文档站点。
- 布尔值内容，即 `true` 或 `false` (无引号)。

- 数字内容，可以是整数或小数，如 4 或 2.5 (无引号)。
- 列表内容，格式为用方括号括起以逗号分隔的值（无引号），例如 ['80', '443']
- JSON 对象，其中包含额外数据包内容，如 "my-app": {"elastic_ip": null,...}。

Chef 配方可直接或通过 Chef 搜索来访问数据包、数据包项目和数据包内容。下面介绍了如何使用这两种访问方法 (虽然 Chef 搜索是首选)。

要通过 Chef 搜索访问数据包，请使用搜索方法，指定所需的搜索索引。 AWS OpsWorks Stacks 提供以下搜索索引：

- aws_opsworks_app，表示为堆栈部署的一组应用程序。
- aws_opsworks_command，表示在堆栈上运行的一组命令。
- aws_opsworks_ecs_cluster，表示堆栈的一组 Amazon Elastic Container Service (Amazon ECS) 集群实例。
- aws_opsworks_elastic_load_balancer，表示堆栈的一组 Elastic Load Balancing 负载均衡器。
- aws_opsworks_instance，表示堆栈的一组实例。
- aws_opsworks_layer，表示堆栈的一组层。
- aws_opsworks_rds_db_instance，它表示堆栈的一组 Amazon Relational Database Service (Amazon RDS) 实例。
- aws_opsworks_stack，表示一个堆栈。
- aws_opsworks_user，表示堆栈的一组用户。

获知搜索索引名称后，就可以访问该搜索索引的数据包内容。例如，以下配方代码使用 aws_opsworks_app 搜索索引来获取 aws_opsworks_app 数据包 (aws_opsworks_app 目录) 中第一个数据包项目 (第一个 JSON 文件) 的内容。然后，该代码会将两个消息写入 Chef 日志，一个消息包含应用程序的短名称数据包内容 (JSON 文件中的字符串)，另一个消息包含应用程序的源 URL 数据包内容 (JSON 文件中的另一个字符串)：

```
app = search("aws_opsworks_app").first
Chef::Log.info("********** The app's short name is '#{app['shortname']}' **********")
Chef::Log.info("********** The app's URL is '#{app['app_source']['url']}' **********")
```

其中，['shortname'] 和 ['app_source']['url'] 在相应的 JSON 文件中指定以下数据包内容：

```
{
  ...
  "shortname": "mylinuxdemoapp",
  ...
  "app_source": {
    ...
    "url": "https://s3.amazonaws.com/opsworks-demo-assets/opsworks-linux-demo-
nodejs.tar.gz",
  },
  ...
}
```

有关您可以搜索的数据包内容列表，请参阅本节中的参考主题。

您也可以循环访问数据包中的一组数据包项目。例如，以下配方代码类似于上一个示例；在有多个数据包项目时，该代码可循环访问数据包中的每个数据包项目：

```
search("aws_opsworks_app").each do |app|
  Chef::Log.info("********** The app's short name is '#{app['shortname']}' **********")
  Chef::Log.info("********** The app's URL is '#{app['app_source']['url']}'
 **********")
end
```

如果您知道存在特定的数据包内容，则可以使用以下语法来查找相应的数据包项目：

```
search("search_index", "key:value").first
```

例如，以下配方代码使用 aws_opsworks_app 搜索索引，以查找包含应用程序短名称 mylinuxdemoapp 的数据包项目。然后，它使用该数据包项目的内容，将包含相应的应用程序短名称和源 URL 的消息写入 Chef 日志：

```
app = search("aws_opsworks_app", "shortname:mylinuxdemoapp").first
Chef::Log.info("********** For the app with the short name '#{app['shortname']}', the
 app's URL is '#{app['app_source']['url']}' **********")
```

仅针对 aws_opsworks_instance 搜索索引，您可以指定 self:true 以表示正在对其执行配方的实例。以下配方代码使用相应的数据袋项的内容向 Chef 日志写一条消息，其中包含相应实例的 AWS OpsWorks Stacks 生成的 ID 和操作系统：

```
instance = search("aws_opsworks_instance", "self:true").first
```

```
Chef::Log.info("********** For instance '#{instance['instance_id']}', the instance's
 operating system is '#{instance['os']}' **********")
```

您可以直接访问数据包、数据包项目和数据包内容，而不使用 Chef 搜索进行访问。为此，可使用 data_bag 和 data_bag_item 方法分别访问数据包和数据包项目。例如，以下配方代码执行的操作与上述示例相同，只是该代码直接访问单个数据包项目，然后访问多数据包项目 (当有多个项目时)：

```
# Syntax: data_bag_item("the data bag name", "the file name in the data bag without the
 file extension")
app = data_bag_item("aws_opsworks_app", "mylinuxdemoapp")
Chef::Log.info("********** The app's short name is '#{app['shortname']}' **********")
Chef::Log.info("********** The app's URL is '#{app['app_source']['url']}' **********")

data_bag("aws_opsworks_app").each do |data_bag_item|
  app = data_bag_item("aws_opsworks_app", data_bag_item)
  Chef::Log.info("********** The app's short name is '#{app['shortname']}' **********")
  Chef::Log.info("********** The app's URL is '#{app['app_source']['url']}'
 **********")
end
```

在这两种方法中，我们建议您使用 Chef 搜索。本指南中的所有相关示例均演示的是这种方法。

主题

- 应用程序数据包 (aws_opsworks_app)
- 命令数据包 (aws_opsworks_command)
- Amazon ECS 集群数据包 (aws_opsworks_ecs_cluster)
- Elastic Load Balancing 数据包 (aws_opsworks_elastic_load_balancer)
- 实例数据包 (aws_opsworks_instance)
- 层数据包 (aws_opsworks_layer)
- Amazon RDS 数据包 (aws_opsworks_rds_db_instance)
- 堆栈数据包 (aws_opsworks_stack)
- 用户数据包 (aws_opsworks_user)

应用程序数据包 (aws_opsworks_app)

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

对于 Deploy 事件或 Execute Recipes 堆栈命令，表示应用程序的设置。

以下示例演示了如何使用 Chef 搜索来先后搜索单个数据包项目和多个数据包项目，以将包含应用程序的短名称和源 URL 的消息写入 Chef 日志：

```
app = search("aws_opsworks_app").first
Chef::Log.info("********** The app's short name is '#{app['shortname']}' **********")
Chef::Log.info("********** The app's URL is '#{app['app_source']['url']}' **********")

search("aws_opsworks_app").each do |app|
  Chef::Log.info("********** The app's short name is '#{app['shortname']}' **********")
  Chef::Log.info("********** The app's URL is '#{app['app_source']['url']}'
 **********")
end
```

app_id	app_source	data_sources
部署	attributes	域
enable_ssl	environment	name
shortname	ssl_configuration	type

app_id

应用程序 ID (字符串)。标识应用程序的 GUID。

app_source

一组内容，用于指定 AWS OpsWorks Stacks 用于从其源代码控制存储库部署应用程序的信息。该内容因存储库类型而异。

password

对于私有存储库为密码，对于公共存储库为 "null" (字符串)。对于私有 S3 存储桶，此内容设置为私有密钥。

revision

如果存储库具有多个分支，该内容将指定应用程序的分支或版本，如 "version1" (字符串)。否则，其设置为 "null"。

ssh_key

访问私有 Git 存储库时为部署 SSH 密钥，对于公共存储库为 "null" (字符串)。

type

应用程序的源位置 (字符串)。有效值包括：

- "archive"
- "git"
- "other"
- "s3"

url

应用程序源代码所在的位置 (字符串)。

用户

对于私有存储库为用户名称，对于公共存储库为 "null" (字符串)。对于私有 S3 存储桶，该内容设置为访问密钥。

attributes

一组描述目录结构和应用程序内容的内容。

document_root

文档树的根目录。定义文档根目录的路径 (或应用程序主页的位置，如 home_html)，该路径相对于您的部署目录。除非指定此属性，否则 document_root 默认为 public。document_root 的值只能以 a-z、A-Z、0-9、_ (下划线) 或 - (连字符) 字符开头。

data_sources

连接应用程序数据库所需的信息。如果应用程序附加了数据库层， AWS OpsWorks Stacks 会自动为此内容分配适当的值。

data_sources 的值为数组，这些数组是通过整型偏移量进行访问，而不是通过密钥进行访问。例如，要访问应用程序的第一个数据源，请使用 `app[:data_sources][0][:type]`。

database_name

数据库名称，这通常是应用程序的短名称 (字符串)。

type

数据库实例的类型，通常为 `"RdsDbInstance"` (字符串)。

arn

数据库实例的 Amazon 资源名称 (ARN) (字符串)。

部署

是否应部署应用程序（布尔值）。`true` 表示应在 Deploy 生命周期事件中部署应用程序。在 Setup 生命周期事件中，所有应用程序的此内容均为 `true`。要确定应在实例上部署哪些应用程序，请查看该实例所属的层。

域

应用程序域的列表 (字符串列表)。

enable_ssl

是否启用 SSL 支持 (布尔值)。

environment

用户指定的已为应用程序定义的环境变量的集合。有关如何定义应用程序的环境变量的更多信息，请参阅添加应用程序。每个内容名称均设置为环境变量名称，且相应的值设置为变量的值。

name

应用程序的名称，用于显示目的 (字符串)。

shortname

应用程序的短名称，由 AWS OpsWorks Stacks 根据名称（字符串）生成。该短名称在内部由配方使用；用作安装应用程序文件的目录的名称。

ssl_configuration

证书

如果您启用了 SSL 支持，即为应用程序的 SSL 证书；否则，为 `"null"` (字符串)。

chain

如果启用 SSL，则为指定中间证书颁发机构密钥或客户端身份验证的内容 (字符串)。

private_key

如果您启用了 SSL 支持，即为应用程序的 SSL 私有密钥；否则，为 "null" (字符串)。

type

应用程序的类型，对于 Chef 12 Linux 和 Chef 12.2 Windows 堆栈，该类型始终设置为 "other" (字符串)。

命令数据包 (aws_opsworks_command)

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

表示 AWS OpsWorks Stacks 在一个或多个实例上运行的命令的设置。

以下示例演示了如何使用 Chef 搜索来先后搜索单个数据包项目和多个数据包项目，以将包含命令类型和发送时间的消息写入 Chef 日志：

```
command = search("aws_opsworks_command").first
Chef::Log.info("********** The command's type is '#{command['type']}' **********")
Chef::Log.info("********** The command was sent at '#{command['sent_at']}' **********")

search("aws_opsworks_command").each do |command|
  Chef::Log.info("********** The command's type is '#{command['type']}' **********")
  Chef::Log.info("********** The command was sent at '#{command['sent_at']}'
 **********")
end
```

args	command_id	iam_user_arn
instance_id	sent_at	type

args

命令的参数 (字符串)。

command_id

命令的随机唯一标识符，由 AWS OpsWorks Stacks（字符串）分配。

iam_user_arn

如果命令是由客户创建的，即为创建命令的用户的 Amazon 资源名称 (ARN) (字符串)。

instance_id

运行命令的实例的标识符 (字符串)。

sent_at

AWS OpsWorks Stacks 运行命令的时间戳（字符串）。

type

命令的类型 (字符串)。有效值包括：

- "configure"
- "deploy"
- "deregister"
- "execute_recipes"
- "grant_remote_access"
- "install_dependencies"
- "restart"
- "revoke_remote_access"
- "rollback"
- "setup"
- "shutdown"
- "start"
- "stop"
- "sync_remote_users"
- "undeploy"
- "update_agent"

- "update_custom_cookbooks"
- "update_dependencies"

Amazon ECS 集群数据包 (aws_opsworks_ecs_cluster)

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

表示 Amazon ECS 集群的设置。

以下示例演示了如何使用 Chef 搜索来先后搜索单个数据包项目和多个数据包项目，以将包含 Amazon ECS 集群名称和 Amazon 资源名称 (ARN) 的消息写入 Chef 日志：

```
ecs_cluster = search("aws_opsworks_ecs_cluster").first
Chef::Log.info("********** The ECS cluster's name is
 '#{ecs_cluster['ecs_cluster_name']}' **********")
Chef::Log.info("********** The ECS cluster's ARN is '#{ecs_cluster['ecs_cluster_arn']}'
 **********")

search("aws_opsworks_ecs_cluster").each do |ecs_cluster|
  Chef::Log.info("********** The ECS cluster's name is
 '#{ecs_cluster['ecs_cluster_name']}' **********")
  Chef::Log.info("********** The ECS cluster's ARN is
 '#{ecs_cluster['ecs_cluster_arn']}' **********")
end
```

ecs_cluster_arn	ecs_cluster_name

ecs_cluster_arn

集群的 Amazon 资源名称 (ARN) (字符串)。

ecs_cluster_name

集群的名称 (字符串)。

Elastic Load Balancing 数据包（aws_opsworks_elastic_load_balancer）

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

表示 Elastic Load Balancing 负载均衡器的设置。

以下示例演示了如何使用 Chef 搜索来先后搜索单个数据包项目和多个数据包项目，以将包含 Elastic Load Balancing 负载均衡器名称和 DNS 名称的消息写入 Chef 日志：

```
elastic_load_balancer = search("aws_opsworks_elastic_load_balancer").first
Chef::Log.info("********** The ELB's name is
 '#{elastic_load_balancer['elastic_load_balancer_name']}' **********")
Chef::Log.info("********** The ELB's DNS name is '#{elastic_load_balancer['dns_name']}'
 **********")

search("aws_opsworks_elastic_load_balancer").each do |elastic_load_balancer|
  Chef::Log.info("********** The ELB's name is
 '#{elastic_load_balancer['elastic_load_balancer_name']}' **********")
  Chef::Log.info("********** The ELB's DNS name is
 '#{elastic_load_balancer['dns_name']}' **********")
end
```

elastic_load_balancer_name	dns_name	layer_id

elastic_load_balancer_name

负载均衡器的名称 (字符串)。

dns_name

负载均衡器的 DNS 名称 (字符串)。

layer_id

负载均衡器分配到的层的 AWS OpsWorks 堆栈 ID（字符串）。

实例数据包 (aws_opsworks_instance)

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

表示实例的设置。

以下示例演示了如何使用 Chef 搜索来先后搜索单个数据包项目和多个数据包项目，以将包含实例主机名和 ID 的消息写入 Chef 日志：

```
instance = search("aws_opsworks_instance").first
Chef::Log.info("********** The instance's hostname is '#{instance['hostname']}'
 **********")
Chef::Log.info("********** The instance's ID is '#{instance['instance_id']}'
 **********")

search("aws_opsworks_instance").each do |instance|
  Chef::Log.info("********** The instance's hostname is '#{instance['hostname']}'
 **********")
  Chef::Log.info("********** The instance's ID is '#{instance['instance_id']}'
 **********")
end
```

以下示例演示了关于使用 Chef 搜索来搜索多个数据包项目以查找包含指定 Amazon EC2 实例 ID 的数据包项目的不同方法。然后，该示例会使用数据包项目的内容将包含相应实例的公有 IP 地址的消息写入 Chef 日志：

```
instance = search("aws_opsworks_instance", "ec2_instance_id:i-12345678").first
Chef::Log.info("********** For instance '#{instance['ec2_instance_id']}', the
 instance's public IP address is '#{instance['public_ip']}' **********")

search("aws_opsworks_instance").each do |instance|
  if instance['ec2_instance_id'] == 'i-12345678'
    Chef::Log.info("********** For instance '#{instance['ec2_instance_id']}', the
 instance's public IP address is '#{instance['public_ip']}' **********")
  end
```

```
end
```

以下示例演示了如何使用 Chef 搜索的 `self:true` 以查找数据包项目，其中包含有关正在对其执行配方的实例的信息。然后，该示例使用数据袋项的内容向 Chef 日志写一条消息，其中包含相应实例的 AWS OpsWorks Stacks 生成的 ID 和实例的公有 IP 地址：

```
instance = search("aws_opsworks_instance", "self:true").first
Chef::Log.info("********** For instance '#{instance['instance_id']}', the instance's
 public IP address is '#{instance['public_ip']}' **********")
```

ami_id	架构	auto_scaling_type
availability_zone	created_at	ebs_optimized
ec2_instance_id	elastic_ip	hostname
instance_id	instance_type	layer_ids
os	private_dns	private_ip
public_dns	public_ip	root_device_type
root_device_volume_id	self	ssh_host_dsa_key_fingerprint
ssh_host_dsa_key_private	ssh_host_dsa_key_public	ssh_host_rsa_key_fingerprint
ssh_host_rsa_key_private	ssh_host_rsa_key_public	status
subnet_id	virtualization_type	

ami_id

实例的 AMI (Amazon 系统映像) ID (字符串)。

架构

实例的架构，始终设置为 "x86_64" (字符串)。

auto_scaling_type

实例的扩展类型：`null`、`timer` 或 `load` (字符串)。

availability_zone

实例的可用区 (AZ)，如 "us-west-2a" (字符串)。

created_at

创建实例的时间，采用 UTC "*yyyy*-*mm*-*ddd*T*hh*:*mm*:*ss*+*hh*:*mm*" 格式 (字符串)。例如，"2013-10-01T08:35:22+00:00" 对应于 2013 年 10 月 10 日 8:35:22，无时区偏移。有关更多信息，请参阅 ISO 8601。

ebs_optimized

实例是否经过 EBS 优化 (布尔值)。

ec2_instance_id

EC2 实例 ID (字符串)。

elastic_ip

弹性 IP 地址；如果实例没有弹性 IP 地址，则设置为 "null" (字符串)。

hostname

主机名，如 "demo1" (字符串)。

instance_id

实例 ID，这是 AWS OpsWorks 堆栈生成的 GUID，用于唯一标识实例（字符串）。

instance_type

实例类型，如 "c1.medium" (字符串)。

layer_ids

实例层的列表，由其唯一 ID 标识；例如，307ut64c-c7e4-40cc-52f0-67d5k1f9992c。

os

实例的操作系统 (字符串)。有效值包括：

- "Amazon Linux 2"
- "Amazon Linux 2018.03"
- "Amazon Linux 2017.09"
- "Amazon Linux 2017.03"

- "Amazon Linux 2016.09"
- "Custom"
- "Microsoft Windows Server 2022 Base"
- "Microsoft Windows Server 2022 with SQL Server Express"
- "Microsoft Windows Server 2022 with SQL Server Standard"
- "Microsoft Windows Server 2022 with SQL Server Web"
- "Microsoft Windows Server 2019 Base"
- "Microsoft Windows Server 2019 with SQL Server Express"
- "Microsoft Windows Server 2019 with SQL Server Standard"
- "Microsoft Windows Server 2019 with SQL Server Web"
- "CentOS 7"
- "Red Hat Enterprise Linux 7"
- "Ubuntu 20.04 LTS"
- "Ubuntu 18.04 LTS"
- "Ubuntu 16.04 LTS"
- "Ubuntu 14.04 LTS"

private_dns

私有 DNS 名称 (字符串)。

private_ip

私有 IP 地址 (字符串)。

public_dns

公有 DNS 名称 (字符串)。

public_ip

公有 IP 地址 (字符串)。

root_device_type

根设备类型 (字符串)。有效值包括：

- "ebs
- "instance-store"

root_device_volume_id

根设备的卷 ID (字符串)。

self

如果此数据包项目包含有关正在对其执行配方的实例的信息，则为 `true`；否则，为 `false` (布尔值)。此值仅适用于配方，不能通过 AWS OpsWorks Stacks API 获得。

ssh_host_dsa_key_fingerprint

较短序列的字节，用于标识较长 DSA 公有密钥 (字符串)。

ssh_host_dsa_key_private

DSA 生成的私有密钥，用于对实例进行 SSH 身份验证 (字符串)。

ssh_host_dsa_key_public

DSA 生成的公有密钥，用于对实例进行 SSH 身份验证 (字符串)。

ssh_host_rsa_key_fingerprint

较短序列的字节，用于标识较长 RSA 公有密钥 (字符串)。

ssh_host_rsa_key_private

RSA 生成的私有密钥，用于对实例进行 SSH 身份验证 (字符串)。

ssh_host_rsa_key_public

RSA 生成的公有密钥，用于对实例进行 SSH 身份验证 (字符串)。

status

实例的状态 (字符串)。有效值包括：

- `"requested"`
- `"booting"`
- `"running_setup"`
- `"online"`
- `"setup_failed"`
- `"start_failed"`
- `"terminating"`
- `"terminated"`
- `"stopped"`

- "connection_lost"

subnet_id

实例的子网 ID (字符串)。

virtualization_type

实例的虚拟化类型 (字符串)。

层数据包 (aws_opsworks_layer)

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

表示层的设置。

以下示例演示了如何使用 Chef 搜索来先后搜索单个数据包项目和多个数据包项目，以将包含层的名称和短名称的消息写入 Chef 日志：

```
layer = search("aws_opsworks_layer").first
Chef::Log.info("********** The layer's name is '#{layer['name']}' **********")
Chef::Log.info("********** The layer's shortname is '#{layer['shortname']}'
 **********")

search("aws_opsworks_layer").each do |layer|
  Chef::Log.info("********** The layer's name is '#{layer['name']}' **********")
  Chef::Log.info("********** The layer's shortname is '#{layer['shortname']}'
 **********")
end
```

ecs_cluster_arn	layer_id	name
软件包	shortname	type
volume_configurations		

ecs_cluster_arn

如果层分配了一个 Amazon ECS 集群，即为 Amazon ECS 集群的 Amazon 资源名称 (ARN)（字符串）。

encrypted

如果 EBS 卷已加密，则为 `true` ；否则为 `false` (布尔值)。

layer_id

图层 ID，这是由 AWS OpsWorks Stacks 生成的 GUID，用于唯一标识图层（字符串）。

name

层的名称，用于表示控制台中的层 (字符串)。它可以由用户定义，并且不必唯一。

软件包

要安装的软件包的列表 (字符串列表)。

shortname

由用户定义的层的短名称 (字符串)。

type

层的类型，对于 Chef 12 Linux 和 Chef 12.2 Windows，该类型始终设置为 `"custom"` (字符串)。

volume_configurations

Amazon EBS 卷配置的列表。

iops

卷支持的每秒 I/O 操作数。

mount_point

卷的挂载点目录。

number_of_disks

卷中的磁盘数。

raid_level

卷的 RAID 配置级别。

size

卷大小 (GiB)。

volume_type

卷的类型：通用型、磁介质、预配置 IOPS、吞吐量优化型 HDD 或冷 HDD。

Amazon RDS 数据包 (aws_opsworks_rds_db_instance)

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

一组数据包内容，用于指定 Amazon Relational Database Service (Amazon RDS) 实例的配置，如下所示：

address	db_instance_identifier	db_password
db_user	engine	rds_db_instance_arn
region		

以下示例演示了如何使用 Chef 搜索来先后搜索单个数据包项目和多个数据包项目，以将包含 Amazon RDS 实例地址和数据库引擎类型的消息写入 Chef 日志：

```
rds_db_instance = search("aws_opsworks_rds_db_instance").first
Chef::Log.info("********** The RDS instance's address is
 '#{rds_db_instance['address']}' **********")
Chef::Log.info("********** The RDS instance's database engine type is
 '#{rds_db_instance['engine']}' **********")

search("aws_opsworks_rds_db_instance").each do |rds_db_instance|
  Chef::Log.info("********** The RDS instance's address is
 '#{rds_db_instance['address']}' **********")
  Chef::Log.info("********** The RDS instance's database engine type is
 '#{rds_db_instance['engine']}' **********")
```

```
end
```

address

实例的 DNS 名称。

port

实例的端口。

db_instance_identifier

实例的 ID。

db_password

实例的主密码。

db_user

实例的主用户名称。

engine

实例的数据库引擎，如 `mysql`。

rds_db_instance_arn

实例的 Amazon 资源名称 (ARN)。

region

实例的 Amazon Web Services Region，如 `us-west-2`。

堆栈数据包 (aws_opsworks_stack)

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

表示堆栈的设置。

以下示例演示了如何使用 Chef 搜索将包含堆栈名称和说明书源 URL 的消息写入 Chef 日志：

```
stack = search("aws_opsworks_stack").first
Chef::Log.info("********** The stack's name is '#{stack['name']}' **********")
Chef::Log.info("********** The stack's cookbook URL is
 '#{stack['custom_cookbooks_source']['url']}' **********")
```

arn	custom_cookbooks_source	name
region	stack_id	use_custom_cookbooks
vpc_id		

arn

堆栈的 Amazon 资源名称 (ARN) (字符串)。

custom_cookbooks_source

一组指定自定义说明书源存储库的内容。

type

存储库类型 (字符串)。有效值包括：

- `"archive"`
- `"git"`
- `"s3"`

url

存储库 URL，如 `"git://github.com/amazonwebservices/opsworks-demo-php-simple-app.git"` (字符串)。

username

对于私有存储库为用户名称，对于公共存储库为 `null` (字符串)。对于私有 Amazon Simple Storage Service (Amazon S3) 存储桶，该内容设置为访问密钥。

password

对于私有存储库为密码，对于公共存储库为 `null` (字符串)。对于私有 S3 存储桶，此内容设置为私有密钥。

ssh_key

访问私有 Git 存储库时为部署 SSH 密钥，对于公共存储库为 null (字符串)。

revision

如果存储库具有多个分支，该内容将指定应用程序的分支或版本，如 "version1" (字符串)。否则，其设置为 null。

name

堆栈名称 (字符串)。

region

堆栈的 Amazon Web Services Region (字符串)。

stack_id

用于标识堆栈的 GUID (字符串)。

use_custom_cookbooks

是否启用自定义说明书 (布尔值)。

vpc_id

如果堆栈在 VPC 中运行，则为 VPC ID (字符串)。

用户数据包 (aws_opsworks_user)

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

表示用户的设置。

以下示例演示了如何使用 Chef 搜索来先后搜索单个数据包项目和多个数据包项目，以将包含用户的用户名称和 Amazon 资源名称 (ARN) 的消息写入 Chef 日志：

```
user = search("aws_opsworks_user").first
Chef::Log.info("********** The user's user name is '#{user['username']}' **********")
Chef::Log.info("********** The user's user ARN is '#{user['iam_user_arn']}'
 **********")

# Or...

search("aws_opsworks_user").each do |user|
  Chef::Log.info("********** The user's user name is '#{user['username']}' **********")
  Chef::Log.info("********** The user's user ARN is '#{user['iam_user_arn']}'
 **********")
end
```

administrator_privileges	iam_user_arn	remote_access
ssh_public_key	unix_user_id	username

administrator_privileges

用户是否具有管理员权限 (布尔值)。

iam_user_arn

用户的 Amazon 资源名称 (ARN) (字符串)。

remote_access

用户是否可以使用 RDP 登录实例 (布尔值)。

ssh_public_key

用户的公钥，由 AWS OpsWorks Stacks 控制台或 API 提供（字符串）。

unix_user_id

用户的 Unix ID (数字)。

username

用户名称 (字符串)。

OpsWorks 代理变更

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

Chef 12 代理版本

下表描述了 AWS OpsWorks Stacks 在其管理的实例上安装的 Chef 12 代理的重要更改。

代理版本	描述	发行日期
4042	• 此代理版本仅包含细微的更改，没有增加新功能	2023 年 2 月 7 日
4041	• 此代理版本仅包含细微的更改，没有增加新功能 • 更新 Amazon CA 证书	2023 年 1 月 27 日
4040	• 此代理版本仅包含细微的更改，没有增加新功能	2022 年 7 月 22 日
4039	• 修复 Ubuntu AMI 的 ECS 集成	2020 年 4 月 30 日
4038	• 修复了在 DST 更改期间发送实例统计信息时的错误 • 在代理下载和安装过程中采用 `no_proxy` 环境变量	2020 年 3 月 5 日
4037	• 增加了在无区域的情况下使用 Sigv4 对 S3 URL 请求进行签名的支持 • 删除了使用 Sigv2 对 S3 请求进行签名的支持	2019 年 6 月 4 日
4035	• 在 ECS 设置期间修复错误 • 更改实例类型后修复重复的 `fstab` 条目	2019 年 5 月 8 日
4033	• 增加了对 Ubuntu 18.04 的支持 • 修复 Amazon Linux 2 中的代理安装错误	2018 年 11 月 26 日

代理版本	描述	发行日期
4032	• 添加了对 Amazon Linux 2 的支持	2018 年 10 月 24 日
4031	• 添加了对 Amazon Linux 2018.03 的支持 • 支持在另一个账户上托管的公有 S3 存档	2018 年 8 月 15 日
4030	• 修复了 c5d 实例的卷处理	2018 年 5 月 31 日
4029	• 在 Ubuntu 14.04 上安装 `nvme-cli` • 修复了 c5、m5 实例上的卷装载 • 在重新启动时始终保留主机名	2018 年 5 月 2 日
4028	• 修复了 CentOS 的 `monit` 配置	2018 年 3 月 20 日
4027	• 支持在 Ubuntu 14.04 上装载 NVMe 卷（`nvme-cli` 必须手动安装） • 不需要卷的 `name` 属性	2018 年 2 月 17 日
4026	• 使用 EBS 卷 ID 装载基于 NVMe 的 EBS 卷 • 修复了 i3 实例上的 EBS 卷装载 • 修复了 c5、m5 实例上装载的 EBS 卷的顺序	2018 年 1 月 31 日
4025	• NVMe 设备的修复处理	2017 年 12 月 13 日
4024	• 添加了对 Amazon Linux 2017.09 的支持	2017 年 12 月 5 日
4023	• 添加对 CloudWatch 日志集成的支持	2017 年 4 月 2 日
4022	• 将 Chef 客户端版本更新到 12.18.31	2017 年 2 月 1 日
4021	• 改善代理处理	2016 年 12 月 16 日
4020	• 将 Chef 客户端版本更新到 12.16.42	2016 年 12 月 8 日

代理版本	描述	发行日期
4019	• 提供代理安装期间的代理变量 • Red Hat Enterprise Linux 7 现在使用 `systemd` 而不是 `monit` • 请勿在 Red Hat Enterprise Linux 7 上设置 EPEL • 使用 `flock` 而不是 `lockrun.c` 来进行流程锁定 • 在检查 `ps -p1` 时避免异常输出 `systemd`	2016 年 10 月 19 日
4018	• 将 Chef 客户端版本更新到 12.13.37 • 添加了对 Amazon Linux 2016.09 的支持	2016 年 8 月 25 日
4017	• 将 Chef 客户端版本更新到 12.12.15	2016 年 8 月 10 日
4016	• 修复了未使用 `monit` 的系统上的卸载代理问题	2016 年 6 月 23 日
4015	• 修复了 Amazon Linux 2016.03 的 ECS 设置问题	2016 年 6 月 17 日
4011	• 将 Chef 客户端版本更新到 12.10.24 • 改善了日志上传处理	2016 年 5 月 19 日
4008	• 添加了对 Amazon Linux 2016.03 的支持 • 为捆绑安装添加了超时设置 • 将 xfs 添加到 /etc/filesystems (如果存在)	2016 年 3 月 16 日
4007	• 将 Chef 客户端版本更新到 12.7.2 • 改进了本地实例 (在 AWS 外部托管的服务器) 的错误处理 • 改善了与最新 chef-sugar 的兼容性 • 为部署重试存档下载	2016 年 3 月 4 日
4006	• 将 Chef 客户端版本更新到 12.6.0 • 请勿在代理安装时安装 libxml2-devel/libxml2-dev 和 libxslt-devel/libxslt-dev 程序包	2016 年 1 月 21 日

代理版本	描述	发行日期
4005	• 通过始终在 ec2 基础设施的 ohai 中启用 ec2 数据修复了 ec2 导入问题	2015 年 12 月 17 日
4004	• AWS OpsWorks 堆栈支持 Chef 12 Linux-Chef Client 12.5.1	2015 年 12 月 3 日

Chef 11.10 代理版本

下表描述了 AWS OpsWorks Stacks 在其管理的实例上安装的 Chef 11.10 代理的重要更改。

代理版本	描述	发行日期
3456	• 此代理版本仅包含细微的更改，没有增加新功能 • 更新 Amazon CA 证书	2023 年 1 月 27 日
3455	• 此代理版本仅包含细微的更改，没有增加新功能	2022 年 11 月 1 日
3454	• 修复 Ubuntu AMI 的 ECS 集成	2020 年 4 月 28 日
3453	• 修复了在 DST 更改期间发送实例统计信息时的错误 • 修复了 RHEL7 设置中丢失软件包的错误 • 在代理下载和安装过程中采用 `no_proxy` 环境变量	2020 年 3 月 5 日
3452	• 不要在 Amazon S3 虚拟路径 URL 中包含区域（如果该区域为 `us-east-1` ） • 将内部说明书提取并上传到阶段区域特定的存储桶 • 修复了 Chef 11.10 的 `fstab` 条目 • 删除了 S3 的 Sigv2 使用，并在请求中获取存储桶的区域	2019 年 8 月 13 日
3451	• 增加了对 Ruby 2.6.1 的支持	2019 年 3 月 20 日
3450	• 修复默认的 EBS 属性 • 修复亚马逊 Linux 2 的 CloudWatchLogs 代理安装问题	2018 年 12 月 3 日

代理版本	描述	发行日期
	• 修复高于 2.6.14 版的 rubygem 版本的 Bundler 安装 • 修复公有 S3 存档支持	
3449	• 修复了 c5d 实例的卷处理 • 修复了 NVMe 设备实例上的 RAID 阵列支持	2018 年 6 月 5 日
3448	• 将 Ruby 的默认 2.3 版本升级到 2.3.7 • 修复了 Ubuntu 14.04 实例上的基于 NVMe 的实例上的 EBS 卷装载 • 支持在另一个账户上托管的公有 Amazon S3 存档 • 修复了 Red Hat Enterprise Linux 实例上的 `opsworks-agent` 启动问题	2018 年 5 月 8 日
3447	• 使用 EBS 卷 ID 装载基于 NVMe 的 EBS 卷 • 修复了 i3 实例上的 EBS 卷装载 • 修复了 c5、m5 上装载的 EBS 卷的顺序 • 将 Ruby 的默认 2.3 版本更新到 2.3.6	2018 年 1 月 31 日
3446	• NVMe 设备的修复处理 • 将 Ruby 的默认 2.3 版本更新到 2.3.5	2017 年 12 月 14 日
3445	• 添加了对 Amazon Linux 2017.09 的支持 • 将 Ruby 的默认 2.2 版本更新到 2.2.8	2017 年 10 月 31 日
3444	• 添加对 CloudWatch 日志的支持	2017 年 4 月 1 日
3443	• 改善代理处理	2016 年 12 月 15 日
3442	• 将 Ruby 的默认 2.3 版本更新到 2.3.3 • 将 Ruby 的默认 2.2 版本更新到 2.2.6	2016 年 12 月 6 日
3441	• 提供代理安装期间的代理变量	2016 年 10 月 21 日

代理版本	描述	发行日期
3440	• 添加了对 Amazon Linux 2016.09 的支持	2016 年 9 月 13 日
3439	• 次要更改；没有新功能	2016 年 7 月 29 日
3438	• 增加了对 Ruby 2.3.1 的支持 • 使用来自 IAM 实例配置文件的凭证改进了实例注册 • 删除了 `s3curl.pl` 剩余项 • 修复了 Amazon Linux 2016.03 的 ECS 设置问题	2016 年 6 月 17 日
3437	• 将 Ruby 的默认 2.2 版本更新到 2.2.5	2016 年 5 月 4 日
3436	• 更新了 Red Hat Enterprise Linux 的 EPEL URL。重要提示：如果没有此更改，Red Hat Enterprise Linux 实例无法启动。	2016 年 4 月 18 日
3435	• 将 Ruby 的默认 2.1 版本更新到 2.1.9 • 改进了 Amazon S3 和存档部署的处理	2016 年 4 月 6 日
3434	• 添加了对 Amazon Linux 2016.03 的支持 • 重试程序包安装	2016 年 3 月 16 日
3433	• 对本地实例（托管在外部的服务器 AWS）进行了一些改进 • 改善了与最新 的兼容性`chef-sugar` • 为部署重试存档下载 • 修复了 Ruby gem 安装 URL	2016 年 2 月 27 日

代理版本	描述	发行日期
3432	• 改进了存储桶名称中特殊字符的处理 • 将 s3_file 更新至版本 2.6.6 • 跳过无指定装载点的卷装载 • 始终重启 unicorn 而不是停止和启动，以防止部署过程中出现停机 • 始终更新 setup 命令的自定义说明书 • 创建 RAID 阵列后，在重启时更新 initramfs 以防止设备映射问题	2016 年 1 月 20 日
3431	• 修复了 Rails 层中的 passenger 和 unicorn gem 安装问题 • 将 Ruby 的默认 2.0、2.1 和 2.2 版本更新到 2.0.0p648、2.1.8 和 2.2.4 • 允许在自定义 JSON 中设置 postgres 程序包名称 • 将 Node.js 默认版本更新到 0.12.9	2015 年 12 月 22 日
3430	• 次要更改；没有新功能	2015 年 11 月 25 日
3429	• 改进 OpsWorks 代理守护进程（关闭 stdout/stderr） • 提高了 s3_file 资源的稳健性（重试，捕获异常）	2015 年 11 月 18 日
3428	• 增加了基于 Gemfile 的 postgres 适配器检测，修复了 https://github.com/aws/opsworks-cookbooks/issues/136	2016 年 6 月 17 日
3427	• 修复了在代理中检索凭证的问题 • 将 Ruby 的默认 2.0、2.1 和 2.2 版本更新到 2.0.0p647、2.1.7 和 2.2.3	2015 年 9 月 11 日

代理版本	描述	发行日期
3426	• 将 aws-sdk 更新到 1.65.0 • 通过将 s3curl 替换为 s3_file cookbook，改进了从 Amazon S3 下载的操作 • 将 Node.js 默认版本更改为 0.12.7 • 为 Node.js 应用程序增加了日志记录。在 shared/log 目录中记录和轮换 STDOUT 和 STDERR • 明确自定义说明书子模块结算更新 • 增加了针对 https://github.com/aws/opsworks-cookbooks/issues/213 的解决方法，将检查该问题以确保在创建 deploy 目录之前已进行绑定装载	2015 年 8 月 27 日
3425	• 对 Amazon Linux 和 Ubuntu 的 ECS 支持	2015 年 7 月 27 日
3424	• 次要更改；没有新功能	2015 年 7 月 9 日
3422	• 完全支持 Red Hat Enterprise Linux 7 • 使 /etc/hosts 生成操作在出错时更容易恢复	2015 年 6 月 29 日
3421	• 可为 Red Hat Enterprise Linux 7 覆盖数据库程序包名称的选项 • 更新了 monit systemd 配置，防止 systemd 发送 kill 信号来处理 monit 监控的进程	2015 年 6 月 11 日

AWS OpsWorks 堆栈资源

> **Important**
>
> 该 AWS OpsWorks Stacks 服务于 2024 年 5 月 26 日终止，新客户和现有客户均已禁用。我们强烈建议客户尽快将其工作负载迁移到其他解决方案。如果您对迁移有疑问，请通过 re AWS : Post 或通过 Pre mium Su AWS pp ort 与 AWS Support 团队联系。

下列相关资源在您使用此服务的过程中会有所帮助。

参考指南、工具和支持资源

AWS OpsWorks Stacks 和 Amazon Web Services 提供了几个很有用的指南、论坛、联系信息和其他资源。

- AWS OpsWorks Stacks API 参考——有关 AWS OpsWorks 堆栈操作和数据类型的描述、语法和用法示例，包括常用参数和错误代码。
- AWS OpsWorks Stacks 技术常见问题解答 — 开发者询问的有关此产品的热门问题。
- AWS OpsWorks Stacks 发布说明-对当前版本的高度概述。本文档对所有新功能、更正和已发现的问题做了具体说明。
- 适用于 AWS 的工具 PowerShell — 一组 Windows PowerShell cmdlet，用于在环境 AWS SDK for .NET 中公开的功能。 PowerShell
- AWS 命令行界面-用于访问 Amazon Web Service 的统一命令行语法。AWS CLI 使用单一设置过程来启用对所有受支持服务的访问。
- AWS OpsWorks Stacks 命令行参考 — 在命令行提示符下使用的 AWS OpsWorks 堆栈专用命令。

- 课程和研讨会 — 指向基于角色的课程和专业课程的链接，以及自定进度的实验室，可帮助您提高 AWS 技能并获得实践经验。
- AWS 开发者中心 — 浏览教程、下载工具并了解 AWS 开发者活动。
- AWS 开发者工具-指向开发者工具、SDK、IDE 工具包和命令行工具的链接，用于开发和管理 AWS 应用程序。
- 入门资源中心 — 了解如何设置你的 AWS 账户、加入 AWS 社区和启动你的第一个应用程序。
- 动手教 step-by-step 程 — 按照教程启动您的第一个应用程序 AWS。

- AWS 白皮书 — 由 AWS 解决方案架构师或其他技术专家撰写的技术 AWS 白皮书完整列表的链接，这些白皮书涵盖架构、安全和经济学等主题。
- AWS Support 中心 — 创建和管理 AWS Support 案例的中心。还包括指向其他有用资源的链接，例如论坛、技术常见问题解答、服务运行状况和 AWS Trusted Advisor。
- AWS Support— 提供有关 AWS Support快速响应支持渠道信息的主要网页 one-on-one，该渠道可帮助您在云中构建和运行应用程序。
- 联系我们 – 用于查询有关 AWS 账单、账户、事件、滥用和其他问题的中央联系点。
- AWS 网站条款 — 有关我们的版权和商标、您的帐户、许可证和网站访问权限以及其他主题的详细信息。

AWS 软件开发套件

Amazon Web Services 提供软件开发套件，用于从几种不同的编程语言访问 AWS OpsWorks 堆栈。软件开发工具包库自动执行许多常见任务，如以加密方式对服务请求进行签名、重试请求或处理错误响应。

- AWS SDK for Java-设置和其他文档
- AWS SDK for .NET-设置和其他文档
- 适用于 PHP 的 AWS 开发工具包-文档
- AWS SDK for Ruby-文档
- 其他文档
- AWS SDK for Python (Boto)-设置和其他文档

开源软件

AWS OpsWorks Stacks 包括各种开源软件包，这些软件包受各自的许可证的约束。有关更多信息，请参阅下列内容：

- 对于 Chef 12 Linux 实例，打开实例上 `/opt/aws/opsworks/current` 目录中的 `THIRD_PARTY_LICENSES` 文件。
- 对于适用于 Linux 的 Chef 11.10 及更早版本，请下载 OpsWorks Linux 代理归因文档 PD F。

AWS OpsWorks 文档历史记录

变更	说明	日期
AWS OpsWorks Stacks 的更新	现在，您可以使用 “就地分离” 工具将您的 OpsWorks 实例与 OpsWorks Stacks 服务分离，请参阅本指南中的使用就地 AWS OpsWorks Stacks 分离工具。	2024 年 4 月 11 日
AWS OpsWorks Stacks 的更新	现在，您可以使用迁移脚本将 AWS OpsWorks 堆栈迁移到 AWS Systems Manager 应用程序管理器。有关更多信息，请参阅本指南中的将AWS OpsWorks Stacks 应用程序迁移到 AWS Systems Manager 应用程序管理器。	2022 年 12 月 22 日
AWS OpsWorks for Chef Automate 和的更新 AWS OpsWorks for Puppet Enterprise	现在提供了故障排除程序，其中描述了如果您的服务器 AWS OpsWorks for Chef Automate 或 OpsWorks Puppet Enterprise 服务器的系统维护失败，您可以采取哪些措施。有关更多信息，请参阅本指南中该部分内容： Chef Automate 服务器的系统维护失败或 Puppet Enterprise 服务器的系统维护失败。	2022 年 9 月 29 日
AWS OpsWorks for Chef Automate 和的更新 AWS OpsWorks for Puppet Enterprise	如果您的 AWS OpsWorks for Chef Automate 或适用于 Puppet Enterprise OpsWorks 的服务器进入`Connection lost`状态，则现在可以使用故	2022 年 3 月 23 日

	障排除程序。有关更多信息，请参阅本指南中该部分内容：Chef Automate 服务器处于 Connection lost 状态或 Puppet Enterprise 服务器处于 Connection lost 状态。	
AWS OpsWorks Stacks 的更新	作为安全最佳实践，您现在可以在信任关系策略中添加aws:SourceArn 或aws:SourceAccount 条件密钥（或两者兼而有之），从而允许 AWS OpsWorks Stacks 访问其他 AWS 服务中执行任务。有关更多信息，请参阅本指南中的防止出现在 AWS OpsWorks Stacks 中出现跨服务混淆代理。	2022 年 3 月 4 日
AWS OpsWorks for Chef Automate 和的更新 AWS OpsWorks for Puppet Enterprise	作为安全最佳实践，您现在可以在信任关系策略中添加aws:SourceArn 或aws:SourceAccount 条件密钥（或两者兼而有之），这些策略允许 AWS OpsWorks for Chef Automate 和 OpsWorks 允许 Puppet Enterprise 访问其他 AWS 服务中的任务。有关更多信息，请参阅本指南中的防止跨服务混淆代理。	2022 年 1 月 10 日

AWS OpsWorks for Chef Automate 和的更新 AWS OpsWorks for Puppet Enterprise	AWS OpsWorks for Chef Automate OpsWorks 而且 Puppet Enterprise 已经更新了托管策略 AWSOpsWorksCMServiceRole AWSOpsWorksCMInstanceProfileRole ，现在将机密存储在。AWS Secrets Manager	2021 年 5 月 3 日
更新到 AWS OpsWorks for Puppet Enterprise	你在控制台中创建的 Puppe OpsWorks t Enterprise 服务器的引擎版本现在是 2019.8.5。通过使用 API，您可以在创建 Puppet Enterprise 服务器时指定版本2019或2017。DescribeServers API 现在会在结果中返回一个名为 PUPPET_API_CRL 的属性。此属性包含供内部使用的证书吊销列表。	2021 年 4 月 28 日
AWS OpsWorks 堆栈使用新的托管策略	AWS OpsWorks Stacks 已更改托管策略，其中包括在 AWS OpsWorks 堆栈中执行所有操作的权限。新政策是AWSOpsWorks_FullAccess。有关每个示例策略权限的更多信息，请参阅 Example policies。	2021 年 2 月 19 日
将 AWS OpsWorks Stacks 堆栈从 EC2-Classic 迁移到 VPC	添加了描述如何将 AWS OpsWorks Stacks 堆栈从 EC2-Classic 迁移到 VPC 的文档。	2020 年 9 月 29 日

为和重新生成入门套件 AWS OpsWorks for Chef AutomateAWS OpsWorks for Puppet Enterprise	添加了描述如何为 AWS OpsWorks for Chef Automate 或 AWS OpsWorks for Puppet Enterprise 服务器重新生成入门套件的文档。	2020 年 7 月 29 日
AWS OpsWorks for Puppet Enterprise 允许您创建使用自定义域、证书和私钥的服务器	现在，您可以使用自定义域、证书和私钥创建 OpsWorks 适用于 Puppet Enterprise 的服务器。您可以通过使用现有服务器的备份创建一个服务器，来将现有的 Puppet Enterprise 服务器更新为使用自定义域。	2020 年 4 月 17 日
AWS OpsWorks for Chef AutomateAWS OpsWorks for Puppet Enterprise 现在支持在控制台中添加标签	现在，您可以使用或向 AWS OpsWorks for Chef Automate 服务器、 AWS OpsWorks for Puppet Enterprise 主服务器或服务器备份添加标签 AWS CLI。 AWS Management Console 有关更多信息，请参阅： 使用标签 (Chef) 或使用标签 (Puppet)。	2020 年 2 月 26 日
AWS OpsWorks for Chef Automate 简化了现有的 Chef Automate 1 服务器升级到 Chef Automate	你可以通过在控制 AWS OpsWorks for Chef Automate 台服务器的详细信息页面上选择 “开始升级” 或运行 **`StartMaintenance`** API 操作，将运行 Chef Automate 1 的符合条件的服务器升级到 Chef Automate 2。有关更多信息，请参阅将 AWS OpsWorks for Chef Automate 服务器升级到 Chef Automate 2。	2020 年 1 月 24 日

AWS OpsWorks for Chef Automate 和 AWS OpsWorks for Puppet Enterprise	本指南中增加了有关 AWS OpsWorks CM（AWS OpsWorks for Chef Automate 和 AWS OpsWorks for Puppet Enterprise）中安全性的新章节。	2019 年 12 月 23 日
AWS OpsWorks for Chef Automate 并 AWS OpsWorks for Puppet Enterprise 支持标记	现在，您可以使用向 AWS OpsWorks for Chef Automate 服务器或 AWS OpsWorks for Puppet Enterprise 主服务器或服务器备份添加标签 AWS CLI。 AWS OpsWorks CM 现在支持基于标签的授权。	2019 年 12 月 18 日
AWS OpsWorks for Chef Automate 允许您创建使用自定义域、证书和私钥的服务器	现在，您可以使用自定义域、证书和私钥创建一个 AWS OpsWorks for Chef Automate 2.0 服务器。您可以通过使用现有服务器的备份创建一个服务器，来将现有的 Chef Automate 2.0 服务器更新为使用自定义域。	2019 年 10 月 22 日
AWS OpsWorks Stacks 现在支持 Ruby 2.6.1	AWS OpsWorks Stacks 支持 Chef 11.10 堆栈中的 Rails App Server 层上的 Ruby 2.6.1。	2019 年 5 月 2 日
AWS OpsWorks for Chef Automate 现在支持厨师自动化 2.0	新的 AWS OpsWorks for Chef Automate 服务器将运行Chef Automate 2.0，其中包括Chef 的更新 InSpec、合规性扫描和报告方面的新功能以及Chef Infra。	2019 年 4 月 30 日

AWS OpsWorks for Chef Automate 和 AWS OpsWorks for Puppet Enterprise	现在 AWS CloudFormation ，您可以使用创建 AWS OpsWorks for Chef Automate 服务器或 AWS OpsWorks for Puppet Enterprise 主服务器。	2019 年 1 月 24 日
AWS OpsWorks 堆栈	AWS OpsWorks 堆栈现在支持在 Chef 12 堆栈中运行 Ubuntu 18.04 LTS 的实例。	2018 年 12 月 18 日
AWS f OpsWorks or Puppet En	添加了与使用的控制存储库建立基于 SSH 的连接的过程。CodeCommit	2018 年 12 月 3 日
AWS OpsWorks 堆栈	AWS OpsWorks 堆栈现在支持在 Chef 12 堆栈中运行 Amazon Linux 2 的实例。	2018 年 11 月 15 日
AWS OpsWorks 堆栈	AWS OpsWorks Stacks 现在支持在 Chef 11.10 堆栈中运行 Amazon Linux 2018.03 的实例。	2018 年 10 月 23 日
AWS OpsWorks 堆栈	AWS OpsWorks Stacks 现在支持在 Chef 12 堆栈中运行 Amazon Linux 2018.03 的实例。	2018 年 8 月 23 日
AWS OpsWorks for Chef Automate 还有 Pupp OpsWorks et Enterp	OpsWorks for Puppet Enterprise 已升级到 PE 2018.1.2。 AWS OpsWorks for Chef Automate 已升级到 Chef Automate 1.8.68。	2018 年 6 月 29 日

- AWS OpsWorks for Chef Automate OpsWorks 适用于 Puppet Enterprise API 版本：2016-11-01
- AWS OpsWorks Stacks API 版本：2016-03-08
- 最新文档更新：2024-04-11

早期更新

下表描述了 2018 年 6 月之前每次发布 AWS OpsWorks 用户指南 时进行的重要更改。

描述	日期
AWS OpsWorks 基于 Windows 的堆栈的 Stacks Chef 版本已升级到 12.22；Ruby 版本现在是 2.3.6。	2018 年 4 月 19 日
使用创建 AWS OpsWorks for Chef Automate 服务器或为 Puppet Enterprise 主服务器创建服务器的新过程。 OpsWorks AWS CLI	2018 年 3 月 23 日
Chef Automate 版本更新至 1.8；Chef Compliance 设置简化，新增 `opsworks-audit` 说明书。	2018 年 3 月 5 日
在 Amazon Ev AWS OpsWorks ents 中增加了对堆栈 CloudWatch 事件的支持。	2018 年 2 月 20 日
增加了对 AWS OpsWorks 堆栈中新 EBS 卷类型的支持，以及新的 API。 DescribeOperatingSystems	2018 年 1 月 25 日
OpsWorks 适用于 Puppet Enterprise， AWS OpsWorks for Chef Automate 现在支持在创建服务器时选择多个安全组。	2018 年 1 月 18 日
增加了对欧洲（巴黎）区域 AWS OpsWorks 堆栈的支持。	2017 年 12 月 19 日
在另外六个区域增加了对 Puppet Enterprise AWS OpsWorks for Chef Automate 和 OpsWorks 对 Puppet Enterprise 服务器的支持，并在中添加了为 Puppet Enterprise 服务器创建备份的过程 AWS OpsWorks for Chef Automate OpsWorks AWS Management Console	2017 年 12 月 18 日
OpsWorks 为 Puppet Enterprise 服务和文档添加了新内容。	2017 年 11 月 16 日
在 Stacks 中增加了对亚马逊 Linux 2017.09 的 AWS OpsWorks 支持。	2017 年 11 月 7 日
添加了对厨师合规性的支持 AWS OpsWorks for Chef Automate。	2017 年 10 月 25 日
添加了对亚马逊 Linux 2017.09 的支持。 AWS OpsWorks for Chef Automate	2017 年 10 月 9 日

描述	日期
在本 AWS OpsWorks for Chef Automate 章中添加了系统维护主题。	2017 年 28 月 7 日
添加了对 AWS OpsWorks 堆栈中标签的支持。	2017 年 6 月 6 日
添加了与 CloudWatch 日志的集成。	2017 年 4 月 10 日
添加了新的 AWS OpsWorks for Chef Automate 服务和文档。	2016 年 12 月 1 日
增加了对美国东部（俄亥俄州）区域终端节点的支持。	2016 年 10 月 12 日
增加了对运行 Amazon Linux 2016.09 操作系统的堆栈和实例的支持。	2016 年 9 月 30 日
增加了对 亚太地区（首尔） 区域 和 9 个额外的区域终端节点的支持。	2016 年 8 月 15 日
增加了对内置层中的 Node.js 0.12.15 和 Ruby 2.3 的支持。	2016 年 7 月 6 日
添加了 对亚太地区（孟买）区域的支持。	2016 年 6 月 28 日
增加了对运行 CentOS 7 操作系统的堆栈和实例的支持。	2016 年 6 月 22 日
添加了演练描述 CodePipeline 和 AWS OpsWorks Stacks 集成。	2016 年 6 月 2 日
增加了对运行 Ubuntu 16.04 LTS 操作系统的堆栈和实例的支持。	2016 年 6 月 1 日
增加了 Chef 12 Linux 支持和相关文档。	2015 年 12 月 3 日
增加了 Node.js 入门演练。	2015 年 7 月 14 日
向 Cookbooks 101 添加了两个新的说明书示例。	2015 年 7 月 14 日
增加了对代理版本管理的支持。	2015 年 6 月 23 日
增加了对管理代理版本的支持。	2015 年 6 月 24 日
增加了对自定义 Windows AMI 的支持。	2015 年 6 月 22 日
新增了三个最佳实践主题。	2015 年 6 月 11 日
增加了对 Windows 堆栈的支持。	2015 年 5 月 18 日

描述	日期
增加了“最佳实践”章节。	2014 年 12 月 15 日
增加了对 Elastic Load Balancing 连接耗尽和自定义 Shutdown 超时的支持。	2014 年 12 月 15 日
增加了对注册在 AWS OpsWorks Stacks 之外创建的实例的支持。	2014 年 12 月 9 日
添加了对 Amazon SWF 的支持。	2014 年 9 月 4 日
增加了对将环境变量与应用程序关联的支持并扩展了 Cookbooks 101 的内容。	2014 年 7 月 16 日
增加了 Cookbooks 101 (一个实施说明书的教程简介)。	2014 年 7 月 16 日
添加了对的支持 CloudTrail。	2014 年 6 月 4 日
添加了对 Amazon RDS 的支持。	2014 年 14 月 5 日
增加了对 Chef 11.10 和 Berkshelf 的支持。	2014 年 3 月 27 日
增加了对 Amazon EBS PIOPS 卷的支持。	2013 年 12 月 16 日
增加了基于资源的权限。	2013 年 12 月 5 日
增加了资源管理。	2013 年 10 月 7 日
增加了对 VPC 的支持。	2013 年 8 月 29 日
增加了对自定义 AMI 和 Chef 11.4 的支持。	2013 年 7 月 24 日
增加了对每个实例的多个层的控制台支持。	2013 年 7 月 1 日
增加了对亚马逊 EBS 支持的实例、Elastic Load Balancing 和亚马逊 CloudWatch监控的支持。	2013 年 5 月 14 日
AWS OpsWorks Stacks 用户指南的初始版本。	2013 年 2 月 18 日

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。