



Implementation Guide

# Application Monitoring with Amazon CloudWatch



# Application Monitoring with Amazon CloudWatch : Implementation Guide

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

---

# Table of Contents

<b>Solution overview</b> .....	<b>1</b>
Features and benefits .....	2
<b>Architecture overview</b> .....	<b>3</b>
<b>Architecture details</b> .....	<b>7</b>
Amazon CloudWatch agent configuration .....	7
Web server configuration .....	7
Amazon CloudWatch dashboard .....	8
AWS services in this solution .....	8
<b>Plan your deployment</b> .....	<b>10</b>
Cost .....	10
Sample cost tables .....	10
Security .....	11
IAM roles .....	11
Security groups .....	13
Supported AWS Regions .....	13
Quotas .....	14
Quotas for AWS services in this solution .....	14
AWS CloudFormation quotas .....	14
AWS Systems Manager Parameter Store quotas .....	15
Amazon CloudWatch Dashboard quotas .....	15
<b>Deploy the solution</b> .....	<b>16</b>
AWS CloudFormation templates .....	16
Prerequisites .....	18
Deployment process overview .....	18
Step 1. Launch the stack .....	19
Step 2. Tag the demo instance .....	22
Step 3. (Optional) Onboard your Apache EC2 instances .....	23
<b>Monitor the solution with Service Catalog AppRegistry</b> .....	<b>24</b>
Activate CloudWatch Application Insights .....	24
Confirm cost tags associated with the solution .....	26
Activate cost allocation tags associated with the solution .....	26
AWS Cost Explorer .....	27
<b>Update the solution</b> .....	<b>28</b>
<b>Apache key performance indicators</b> .....	<b>29</b>

<b>NGINX key performance indicators</b> .....	<b>30</b>
<b>Puma key performance indicators</b> .....	<b>31</b>
<b>Configuring your workload instances</b> .....	<b>32</b>
Configuring your Apache EC2 instance .....	32
Amazon CloudWatch agent configuration for Apache .....	32
httpd.conf for Apache web server .....	33
Configuring your NGINX EC2 instance .....	34
Amazon CloudWatch agent configuration for Nginx .....	34
nginx.conf for NGINX web server .....	35
Configuring your Puma EC2 instance .....	36
Amazon CloudWatch agent configuration for Puma .....	36
Configurations for Puma web server .....	38
<b>Add or remove EC2 instance tags</b> .....	<b>40</b>
<b>Extending to more workloads</b> .....	<b>41</b>
<b>Troubleshooting</b> .....	<b>42</b>
Common errors .....	43
My instance doesn't show up on the dashboard. ....	43
My EC2 Apache access logs appear on the CloudWatch Logs but they are not showing on the dashboard .....	43
My instances continue to show up on the dashboard even after being terminated. ....	44
Contact AWS Support .....	45
Create case .....	45
How can we help? .....	45
Additional information .....	45
Help us resolve your case faster .....	46
Solve now or contact us .....	46
<b>Uninstall the solution</b> .....	<b>47</b>
Using the AWS Management Console .....	47
Delete CloudWatch Log groups .....	47
Using AWS Command Line Interface .....	48
<b>Developer guide</b> .....	<b>49</b>
Source code .....	49
<b>Reference</b> .....	<b>50</b>
Anonymized data collection .....	50
Contributors .....	51
<b>Revisions</b> .....	<b>52</b>

---

**Notices** ..... **54**

# Automate setting up an Amazon CloudWatch dashboard

Publication date: April 2021 ([last update: June 2024](#))

The Application Monitoring with Amazon CloudWatch solution automates the process of setting up Amazon CloudWatch dashboards for your Apache, NGINX, and Puma workloads running on Amazon EC2. Using the solution, you can reduce the time it takes to get started with monitoring key performance metrics and logs for your web servers running on AWS. This solution provides a preconfigured dashboard, so you can analyze web traffic patterns, determine whether to scale servers up or out, and detect bottlenecks and other performance problems for the workloads.

This implementation guide provides an overview of the Application Monitoring with Amazon CloudWatch solution, its reference architecture and components, considerations for planning the deployment, configuration steps for deploying this solution to the Amazon Web Services (AWS) Cloud. The solution is intended for DevOps engineers, developers, site reliability engineers (SREs), and IT managers who are responsible for workload and performance monitoring.

You can use this navigation table to quickly find answers to these questions:

## If you want to ...

## Read ...

Know the cost for running this solution.

[Cost](#)

Understand the security considerations for this solution.

[Security](#)

Know how to plan for quotas for this solution.

[Quotas](#)

Know which AWS Regions this solution supports.

[Supported AWS Regions](#)

View or download the AWS CloudFormation templates included in this solution to automatically deploy the infrastructure resources (the “stack”) for this solution.

[AWS CloudFormation templates](#)

Access the source code and optionally use the AWS Cloud Development Kit (AWS CDK) to deploy the solution.

[GitHub repository](#)

## Features and benefits

This solution is built on [Amazon CloudWatch](#) and uses several CloudWatch features, such as metrics, metrics explorer, logs insights, and dashboards to automate the dashboard set up process, and provide you with a centralized view of your workload performance. This solution also provides demo resources, such as a demo EC2 instance, so that you can preview how this solution works.

This solution provides the following features:

- Preconfigured dashboard for Apache, NGINX, and Puma workloads.
- Automated process to set up an Amazon CloudWatch dashboard.
- Tagging mechanism to add or remove instances from the dashboard.
- Amazon CloudWatch agent configuration files for specific workloads.

### **Integration with AWS Service Catalog AppRegistry and Application Manager, a capability of Systems Manager**

This solution includes a Service Catalog AppRegistry resource to register the solution's CloudFormation template and its underlying resources as an application in both both [AWS Service Catalog AppRegistry](#) and [AWS Systems Manager Application Manager](#). With this integration, you can centrally manage the solution's resources.

## Architecture overview

This solution is a reference architecture that automates the setup of Amazon CloudWatch dashboards to monitor key performance metrics and logs from your web servers. This solution deploys a preconfigured dashboard for Apache, NGINX, and Puma workloads.

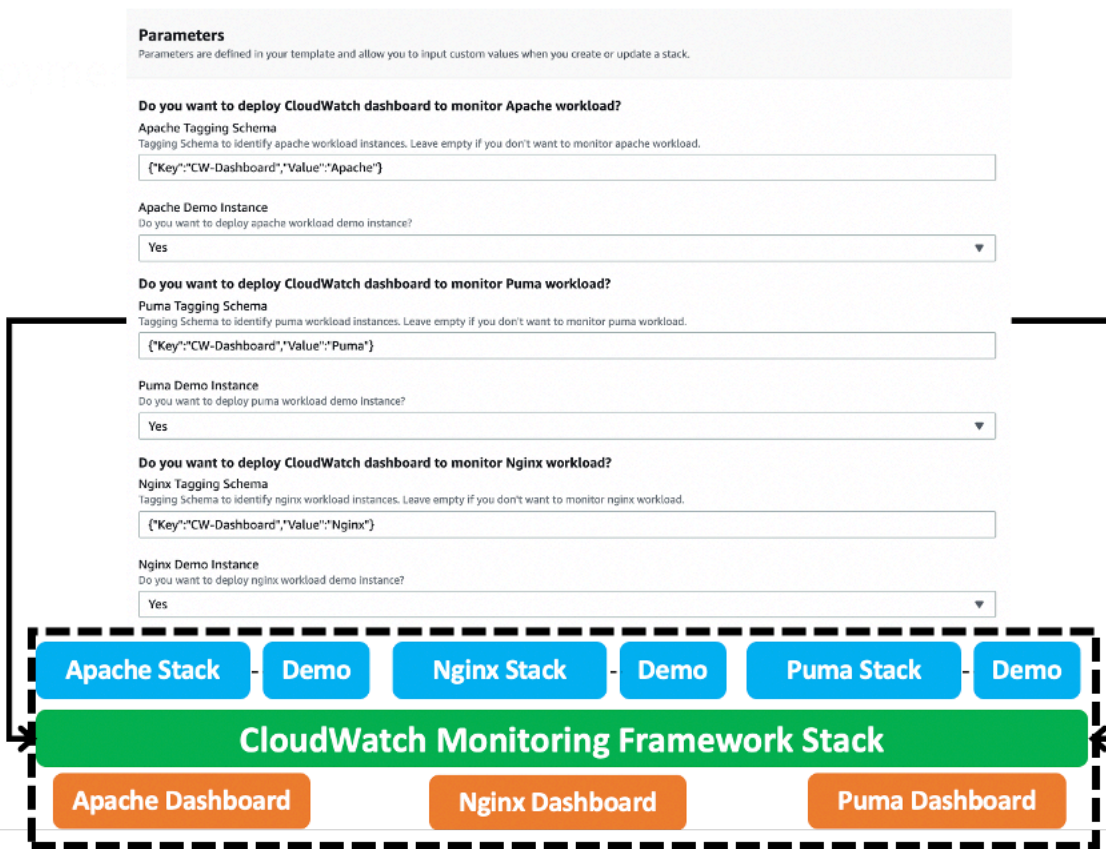
This solution uses tagging mechanisms to add or remove instances from the monitoring dashboard. When you deploy the solution, you provide the tag (key, value pair) as a parameter to identify your instances. An AWS Lambda function runs on cron schedule to search for instances with the provided tag. The Lambda function maintains the list of instances in an AWS Systems Manager Parameter Store. Whenever the instance list on Systems Manager Parameter Store gets updated, it invokes an Amazon CloudWatch Events rule. This rule uses Lambda to update the deployed dashboard with metrics and logs for the new instance list.

### Note

Before you add designated tags to your desired workload instance, ensure that you configure your instances correctly. It is important to refer to the CloudWatch agent and web server config files when configuring your instances. Failing to do so will result in missing metrics and logs from dashboard. For more information, refer to [Configuring your Apache EC2 instance](#), [Configuring your NGINX EC2 instance](#), or [Configuring your Puma EC2 instance](#).

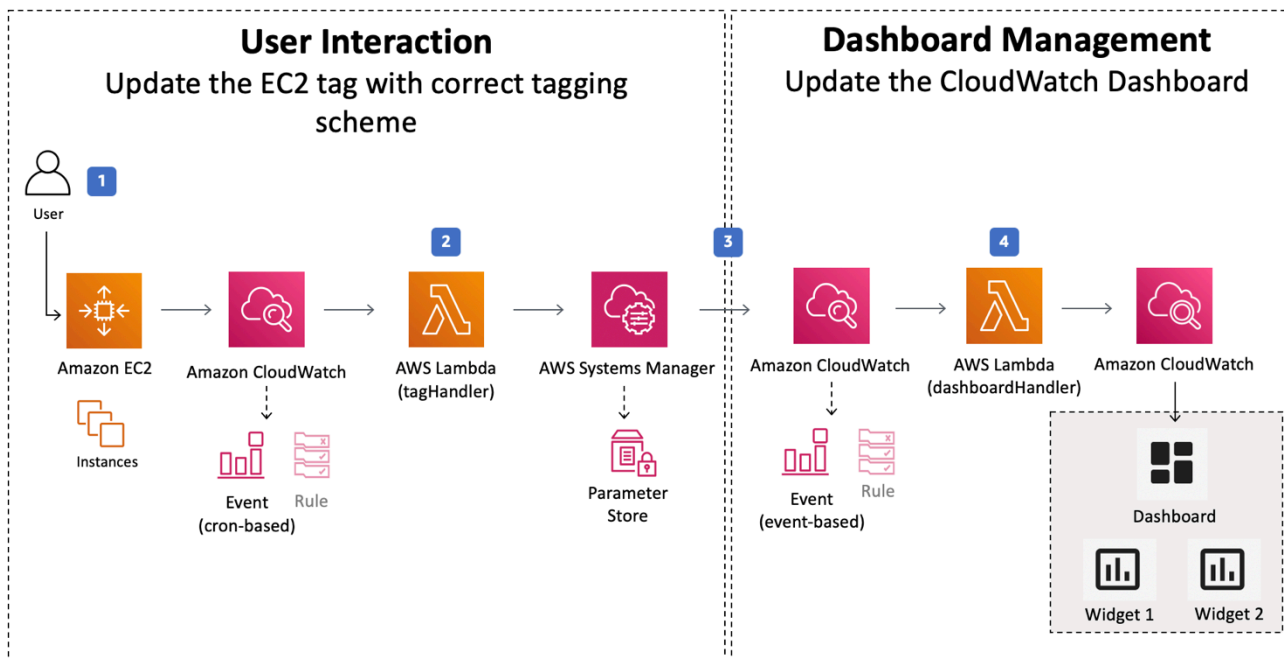
As illustrated in the figure, the deployed solution provides demo web server instances and Amazon CloudWatch dashboards to monitor your workload. As of September 2022, this solution only supports Apache, NGINX, and Puma. However, you can customize this solution to work with other workloads. For more information on how to extend this solution to other workloads, refer to the [README.md](#) file in the GitHub repository.





## Deployment diagram

Deploying this solution with the default parameters builds the following environment in the AWS Cloud. The architecture can be broken down into two workflows: User Interaction and Dashboard Management.



## Application Monitoring with Amazon CloudWatch architecture

### User Interaction Workflow:


The AWS CloudFormation template deploys the [Amazon CloudWatch Events](#) rule, [AWS Lambda](#) function, and [AWS Systems Manager Parameter Store](#) necessary to capture the workload instances in your account.

1. Users update the tag on their instances with the tagging schema provided as input during the stack deployment.
2. The CloudWatch Events rule invokes the `tagHandler` Lambda function on cron schedule. The Lambda function primarily performs two tasks:
  - It uses the `ec2 describe-tags` API call to get instances with the desired tag value (same as step 1).
  - It compares the fetched instance list with the SSM Parameter Store instance list and updates the parameter as needed.

### Dashboard Management Workflow:

The AWS CloudFormation template deploys the Amazon CloudWatch Events rule and AWS Lambda function necessary to configure workload specific Amazon CloudWatch dashboard in your account.

3. When the Systems Manager Parameter Store gets updated, the event is captured using CloudWatch Events rule to invoke the `dashboardHandler` Lambda function.
4. The `dashboardHandler` Lambda function updates the deployed dashboard widgets with logs and metrics for the updated instance list.

 **Note**

AWS CloudFormation resources are created from AWS Cloud Development Kit (AWS CDK) constructs.

## Architecture details

This section describes the components and AWS services that make up this solution and the architecture details on how these components work together.

The solution does not configure your Apache, NGINX, or Puma EC2 instances. You must verify that your EC2 instance is sending workload related metrics and logs in the appropriate format to Amazon CloudWatch. Failing to do so can result in missing values from the dashboard. For more information, refer to [Configuring your Apache EC2 instance](#), [Configuring your NGINX EC2 instance](#), or [Configuring your Puma EC2 instance](#).

## Amazon CloudWatch agent configuration

The solution uses multiple CloudWatch agent configuration files to gather workload metrics and logs. For more information, refer to [Multiple CloudWatch Agent Configuration Files](#) in the *Amazon CloudWatch User Guide*. The following reference CloudWatch agent configuration files provide more information about collecting metrics and logs for your workload instances.

- **Base infrastructure config file** - CloudWatch agent configuration for collecting infrastructure related metrics and logs. For more information, refer to the file in the [GitHub repository](#).
- **Apache config file** - CloudWatch agent configuration for collecting apache host and system related metrics and logs. For more information, refer to the file in the [GitHub repository](#).
- **NGINX config file** - CloudWatch agent configuration for collecting NGINX host and system related metrics and logs. For more information, refer to the file in the [GitHub repository](#).
- **Puma config file** - CloudWatch agent configuration for collecting puma host and system related metrics and logs. For more information, refer to the file in the [GitHub repository](#).

For a complete list of the monitored key performance indicators, refer to [Apache key performance indicators](#), [NGINX key performance indicators](#), and [Puma key performance indicators](#).

## Web server configuration

The solution provides sample web server config files for reference when configuring your web servers. This file provides information on how to collect error and access logs using the appropriate format. You must verify that your web server log files are pushed to Amazon CloudWatch using the appropriate format. For more information, refer to the following files in the GitHub repository.

- [Sample Apache web server config](#)
- [Sample NGINX web server config](#)
- [Sample Puma web server config](#)

## Amazon CloudWatch dashboard

The solution deploys an opinionated dashboard(s) to monitor your Apache, NGINX, and Puma workloads running on Amazon EC2 instances. The dashboard is a collection of logs, metrics, and widgets to monitor some common performance indicators. For more information about metrics and logs being monitored, refer to the [Apache key performance indicators](#), [NGINX key performance indicators](#), or [Puma key performance indicators](#).

You can customize the dashboard. For more information on how to customize metrics for your workload, refer to the Custom-Metrics section in the [README.md](#) file.

## AWS services in this solution

The following AWS services are used in this solution:

AWS service	Description
<a href="#">AWS CloudFormation</a> Core.	Deploys the templates in this solution.
<a href="#">Amazon CloudWatch</a> Core.	Deploys CloudWatch Event rules to periodically invoke the <code>tagHandler</code> Lambda function to check tagged instances and to invoke the <code>dashboardHandler</code> Lambda function when the value in the parameter store changes. Deploys the dashboard for workloads.
<a href="#">AWS Identity and Access Management</a> (IAM) Core.	Deploys roles and policies to provide Lambda functions access to CloudWatch and Amazon EC2.
<a href="#">AWS Lambda</a> Core.	Deploys <code>tagHandler</code> and <code>dashboardHandler</code> Lambda functions for workloads.

AWS service	Description
<a href="#">AWS Systems Manager Parameter Store</a> <b>Core.</b>	Contains the instance IDs of the tagged resources.
<a href="#">Amazon EC2</a> <b>Supporting.</b>	Deploys the sample workloads included in this solution.

# Plan your deployment

This section covers cost, security, quotas, AWS regions and other considerations for planning your deployment.

## Cost

You are responsible for the cost of the AWS services used while running this solution. As of this revision, the cost for running this solution for Apache, NGINIX, and Puma workloads with the default settings in the US East (N. Virginia) Region is approximately **\$3.16 per month per workload**.

This cost includes estimated charges for Amazon CloudWatch, AWS Lambda, and AWS Systems Manager Parameter Store.

Refer to the pricing webpage for each AWS service used in this solution.

We recommend creating a [budget](#) through [AWS Cost Explorer](#) to help manage costs. Prices are subject to change. For full details, refer to the pricing webpage for each AWS service used in this solution.

## Sample cost tables

AWS Service	Dimension	Total Cost (per month) [USD]
Amazon CloudWatch	1 Dashboard	\$3.00
AWS Lambda	15,000 requests with average billed duration of 500ms and memory allocated 512MB	\$0.06
AWS Systems Manager	1 Advanced parameter storage cost + 10,000 API interactions cost	\$0.10
	<b>Total</b>	<b>\$3.16</b>

If you choose to deploy the demo resources for the supported workloads, this solution launches a sample web server that runs on an Amazon EC2 instance. The Amazon EC2 instance sends metrics and logs to Amazon CloudWatch. If the instance runs for one month and generates 10 GB of log data with 1GB of data analyzed by CloudWatch Logs Insight queries, the estimated cost is approximately **\$13.50 per month**, as shown in the following table.

AWS Service	Dimension	Total Cost (per month) [USD]
Amazon CloudWatch	10GB Logs data with 1GB analyzed with CloudWatch Logs Insights queries	\$5.10
Amazon EC2	t3.micro instance with 8GB EBS (gp2) storage	\$8.40
	Total	<b>\$13.50</b>

Prices are subject to change. For full details, refer to the pricing webpage for each AWS service you will be using in this solution.

## Security

When you build systems on AWS infrastructure, security responsibilities are shared between you and AWS. This [shared model](#) reduces your operational burden because AWS operates, manages, and controls the components including the host operating system, the virtualization layer, and the physical security of the facilities in which the services operate. For more information about AWS security, refer to [AWS Cloud Security](#).

## IAM roles

AWS Identity and Access Management (IAM) roles allow customers to assign granular access policies and permissions to services and users on the AWS Cloud. This solution creates IAM roles that grant the solution's AWS Lambda functions access to create, update, or delete Regional resources.

**Tag Handler Role:** permission to update SSM Parameter Store and run `describeTags` call on EC2 resources.



```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "ssm:PutParameter",
        "ssm:GetParameter"
      ],
      "Resource": "arn:aws:ssm:<region>:<account-id>:parameter/<parameter-name>",
      "Effect": "Allow",
      "Sid": "SSMWrite"
    },
    {
      "Action": "ec2:DescribeTags",
      "Resource": "*",
      "Effect": "Allow",
      "Sid": "EC2Describe"
    }
  ]
}
```

**Dashboard Handler Role:** permissions to get parameter and update or delete CloudWatch dashboard.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "ssm:GetParameter",
      "Resource": "arn:aws:ssm:<region>:<account-id>:parameter/<parameter-name>",
      "Effect": "Allow",
      "Sid": "SSMRead"
    },
    {
      "Action": [
        "cloudwatch:PutDashboard",
        "cloudwatch>DeleteDashboards"
      ],
      "Resource": "arn:aws:cloudwatch::<account-id>:dashboard/<dashboard-name>",
      "Effect": "Allow",
      "Sid": "CWWrite"
    }
  ]
}
```

```
}
```

**DemoEC2:** The demo template provisions an Amazon EC2 instance to demonstrate sample Apache web server. The EC2 IAM profile has permissions to publish CloudWatch Metrics and create logs on CloudWatch log group.

## Security groups

The demo AWS CloudFormation template provisions a security group for the demo instance. This security group only allows http(s) ingress and egress traffic.

### Note

Because the demo Apache web server is publicly accessible on port 80 (http), we **do not** recommend deploying the demo AWS CloudFormation template in a production environment. For production environments, we recommend [creating scalable and load-balanced web server application](#).

## Supported AWS Regions

The solution monitors instances in the deployed Region. You should deploy the solution in the Region where you want to monitor your workload instances.

Region name	Region name
US East (Ohio)	Canada (Central)
US East (N. Virginia)	Europe (Frankfurt)
US West (Northern California)	Europe (Ireland)
US West (Oregon)	Europe (London)
Africa (Cape Town)	Europe (Milan)
Asia Pacific (Hong Kong)	Europe (Paris)
Asia Pacific (Hyderabad)	Europe (Spain)

Region name	Region name
Asia Pacific (Jakarta)	Europe (Stockholm)
Asia Pacific (Melbourne)	Europe (Zurich)
Asia Pacific (Mumbai)	Middle East (Bahrain)
Asia Pacific (Osaka)	Middle East (UAE)
Asia Pacific (Seoul)	South America (São Paulo)
Asia Pacific (Singapore)	AWS GovCloud (US-East)
Asia Pacific (Sydney)	AWS GovCloud (US-West)
Asia Pacific (Tokyo)	

## Quotas

Service quotas, also referred to as limits, are the maximum number of service resources or operations for your AWS account.

### Quotas for AWS services in this solution

Make sure you have sufficient quota for each of the [services implemented in this solution](#). For more information, see [AWS service quotas](#).

Use the following links to go to the page for that service. To view the service quotas for all AWS services in the documentation without switching pages, view the information in the [Service endpoints and quotas](#) page in the PDF instead.

### AWS CloudFormation quotas

Your AWS account has AWS CloudFormation quotas that you should be aware of when [launching the stack](#) in this solution. By understanding these quotas, you can avoid limitation errors that would prevent you from deploying this solution successfully. For more information, refer to AWS CloudFormation quotas in the in the [AWS CloudFormation User's Guide](#).

## AWS Systems Manager Parameter Store quotas

As of December 2022, CloudWatch Monitoring on AWS uses AWS Systems Manager Parameter Store with *Advanced* tier parameters to store instance IDs. The parameters have a maximum size quota of 8KB, which amounts to approximately 400 instance IDs.

## Amazon CloudWatch Dashboard quotas

As of December 2022, Amazon CloudWatch Dashboard widgets have a maximum quota of 500 metrics on a single widget.

Due to limitations, we recommend monitoring **less than 400 instances** on a single dashboard. To monitor more than 400 instances on a single dashboard, you can deploy the `workload.template` multiple times in any Region, with **different dashboard names** and **tag schema** parameter values.

## Deploy the solution

This solution uses [AWS CloudFormation templates and stacks](#) to automate its deployment. The CloudFormation template(s) describes the AWS resources included in this solution and their properties. The CloudFormation stack provisions the resources that are described in the templates.

## AWS CloudFormation templates

You can download the CloudFormation template for this solution before deploying it.

### Important

This solution includes an option to send anonymized operational metrics to AWS. We use this data to better understand how customers use this solution and related services and products. AWS owns the data gathered through this survey. Data collection is subject to the [AWS Privacy Notice](#).



To opt out of this feature, download the template, modify the AWS CloudFormation mapping section, and then use the AWS CloudFormation console to upload your template and deploy the solution. For more information, refer to section [Anonymized data collection](#) of this guide.

[View template](#)

**amazon-cloudwatch-monitoring-framework.template:** Use this template to launch the solution and all associated components. The default configuration deploys infrastructure for currently supported Apache, NGINX and Puma workload, and a demo web server running on an EC2 instance for each workload. The demo instance is pre-configured with the necessary CloudWatch agent and web server configurations.

[View template](#)

**workload.template:** Use this template if you want to deploy infrastructure for your workload without using the framework template. Review the following parameters if you are deploying this template independently.

Parameter	Default	Description
Workload Name	none	<p>Workload that you want to monitor. Supported workload names are <b>Apache, Nginx, Puma</b>.</p> <div data-bbox="1068 472 1510 688" style="border: 1px solid #add8e6; border-radius: 10px; padding: 10px; background-color: #e6f2ff;"> <p> <b>Note</b></p> <p>Workload names are case sensitive.</p> </div>
Tag Schema	none	<p>Tagging schema to identity workload instances.</p> <div data-bbox="1068 856 1510 1029" style="border: 1px solid #add8e6; border-radius: 10px; padding: 10px; background-color: #e6f2ff;"> <p> <b>Note</b></p> <p>Tags are key sensitive.</p> </div>
Dashboard Name	none	<p>Name for the deployed CloudWatch Dashboard. Region will be appended to the name. For example: ApacheDashboard-us-east-1</p>
Access Log Group	none	<p>CloudWatch Log Group where instances push their access logs.</p>
SSM Parameter Name	none	<p>AWS Systems Manager parameter used for maintaining workload instance list.</p>

[apache-demo.template](#): Use this template to deploy demo Apache web server running on EC2 instance which is preconfigured with needed CloudWatch agent and httpd configurations.

[nginx-demo.template](#): Use this template to deploy a demo NGINX web server running on EC2 instance, which is preconfigured with a CloudWatch agent and dependency configurations.

[puma-demo.template](#): Use this template to deploy demo Puma web server running on an EC2 instance, which is preconfigured with a CloudWatch agent and dependency configurations.

### Note

As of September 2022, the `amazon-cloudwatch-monitoring-framework.template` only supports Apache, NGINX, and Puma workloads and it deploys `workload.template` as nested stacks. Optionally, you can deploy `workload.template` as standalone templates as well.

AWS CloudFormation resources are created from AWS Cloud Development Kit (AWS CDK) constructs.

## Prerequisites

Before you launch this solution, review the configuration section and consider the CloudWatch agent and web server configuration files discussed in the [Solution Components](#) section of this guide. Follow the step-by-step instructions in this section to configure and deploy this solution into your account.

### Note

This solution does not configure your EC2 instances. Amazon CloudWatch agent and required configuration files are provided with the solution so that you can refer to them when configuring EC2 instances for your respective workload. For more information, refer to [Configuring your Apache EC2 instance](#), [Configuring your Puma EC2 instance](#), or [Configuring your NGINX instance](#). Additionally, a demo template is provided for you to use for demo and proof of concept purposes.

## Deployment process overview

**Time to deploy:** Approximately five minutes

Use the following steps to deploy this solution on AWS. For detailed instructions, follow the links for each step.

### [Step 1. Launch the stack](#)

- Launch the AWS CloudFormation template for the framework into your AWS account.
- Enter values for required parameters: Tagging Schema and Demo Instance.
- Review the template parameters, and adjust if necessary.

### [Step 2. Tag the demo instance](#)

### [Step 3. Onboard your workload EC2 instances \(Optional\)](#)

## Step 1: Launch the stack

This automated AWS CloudFormation template deploys `amazon-cloudwatch-monitoring-framework.template` for Apache, NGINX, and Puma workloads in the AWS Cloud.

#### Note

You are responsible for the cost of the AWS services used while running this solution. For more details, refer to the [Cost](#) section in this guide, and refer to the pricing webpage for each AWS service used in this solution.

1. Sign in to the AWS Management Console and select the button to launch the framework AWS CloudFormation template.




Launch  
solution

2. The template launches in the US East (N. Virginia) Region by default. To launch the solution in a different AWS Region, use the Region selector in the console navigation bar.
3. On the **Create stack** page, verify that the correct template URL is in the **Amazon S3 URL** text box and choose **Next**.



4. On the **Specify stack details** page, assign a name to your solution stack. For information about naming character limitations, refer to [IAM and STS Limits](#) in the *AWS Identity and Access Management User Guide*.
5. Under **Parameters**, review the parameters for Apache workload. Modify them as necessary. This solution uses the following default values.

Parameter	Default	Description
Apache Tagging Schema	<code>{"Key": "CW-Dashboa rd", "Value": "Apach e"}</code>	Tagging schema to identity apache workload instances.  <div style="border: 1px solid #0070C0; border-radius: 10px; padding: 10px; background-color: #E6F2FF;"> <p><b>Note</b></p> <p>Tags are key sensitive</p> <p>.</p> </div>
Apache Demo Instance	Yes	Demo EC2 instance with needed configurations and Apache workload running.
NGINX Tagging Schema	<code>{"Key": "CW-Dashboa rd", "Value": "Nginx "}</code>	Tagging schema to identity NGINX workload instances.  <div style="border: 1px solid #0070C0; border-radius: 10px; padding: 10px; background-color: #E6F2FF;"> <p><b>Note</b></p> <p>Tags are key sensitive</p> <p>.</p> </div> <p>Leave empty if you don't want to monitor NGINX workload.</p>
NGINX Demo Instance	Yes	Demo EC2 instance with required configurations and NGINX workload running.

Parameter	Default	Description
Puma Tagging Schema	<code>{"Key": "CW-Dashboa rd", "Value": "Puma"}</code>	<p>Tagging schema to identity Puma workload instances.</p> <div data-bbox="1081 352 1510 569" style="border: 1px solid #add8e6; border-radius: 10px; padding: 10px; margin: 10px 0;"> <p> <b>Note</b></p> <p>Tags are key sensitive</p> <p>.</p> </div> <p>Leave empty if you don't want to monitor Puma workload.</p>
Puma Demo Instance	Yes	Demo EC2 instance with required configurations and Puma workload running.

6. Choose **Next**.
7. On the **Configure stack options** page, choose Next.
8. On the **Review** page, review and confirm the settings. Check the box acknowledging that the template will create AWS Identity and Access Management (IAM) resources.
9. Choose **Create stack** to deploy the stack.

You can view the status of the stack in the AWS CloudFormation Console in the **Status** column. You should receive a CREATE\_COMPLETE status in approximately five minutes.

 **Note**

In addition to the primary AWS Lambda functions TagHandler & DashboardHandler this solution includes the solution-helper Lambda function, which runs only during initial configuration or when resources are updated or deleted.

When you run this solution, you will notice these Lambda functions in the AWS console. Only the primary lambda functions are regularly active. However, you must not delete the solution-helper function, as it is necessary to manage associated resources.

## Step 2: Tag the demo instance

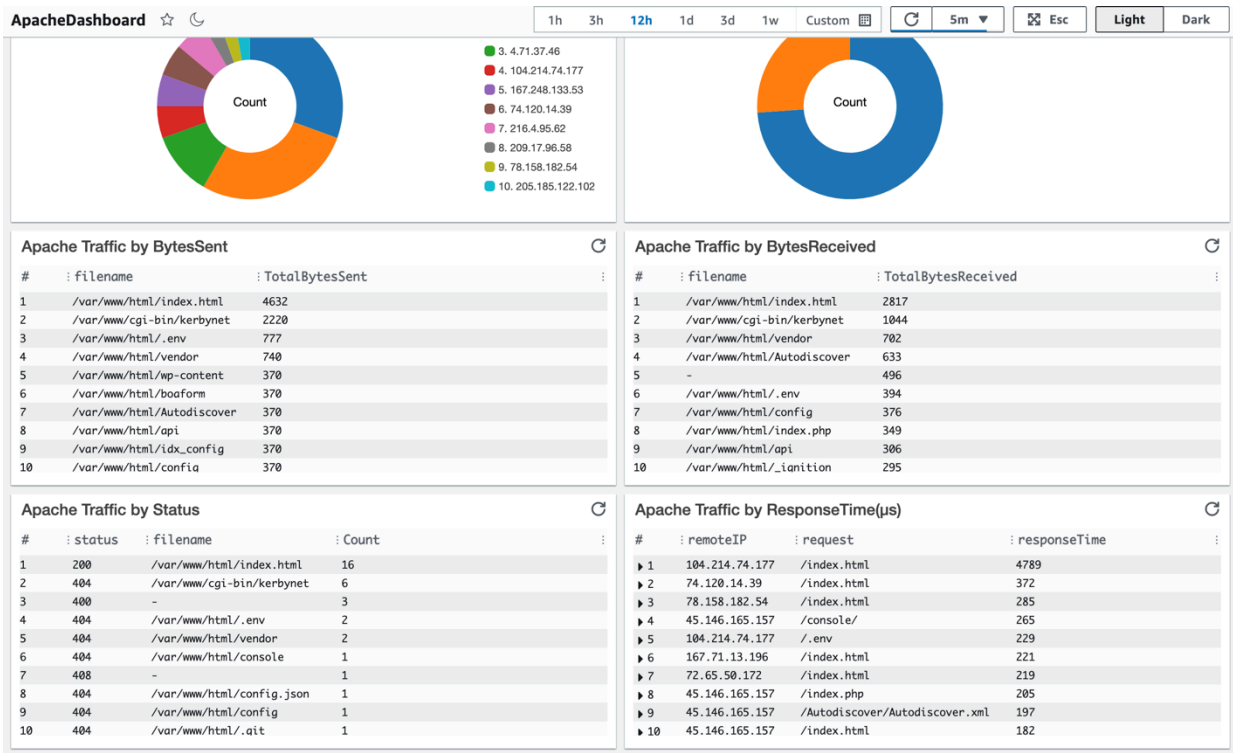
Before adding a tag to the demo instances, verify that the **<Workload> Demo Instance** parameter for the workloads is set to Yes. Once the deployment is complete, you can tag instances and monitor them on the Amazon CloudWatch dashboard. When adding a tag to your demo EC2 Instance, you must use the same tag as the **<Workload> Tagging Schema** parameter value used in your stack deployment.

1. Sign in to the [Amazon EC2 console](#).
2. Select the demo instance `<stack-name>-DemoEC2` deployed with default parameters.
3. Select the **Tags** tab and choose **Manage tags**.
4. Choose **Add tag** to the instance and provide the key-value pair you provided during deployment. The default value for Apache demo instance is shown in the Add tag figure for the Apache workload. Similar tags need to be added for respective workload demo instances.

Key	Value - <i>optional</i>
<input type="text" value="CW-Dashboard"/>	<input type="text" value="Apache"/>
<input type="button" value="Add tag"/>	<input type="button" value="Remove"/>

### Add tag

5. Choose **Save**.
6. The Lambda function runs on cron schedule and it can take up to 5 minutes for the instance to show up on the CloudWatch dashboard.
7. Sign in to the [Amazon CloudWatch dashboard console](#) and select the deployed dashboard. The dashboard should look similar to the example (CloudWatch Dashboard for Apache figure) for Apache. You should see similar dashboards for NGINX and Puma workloads as well.



## CloudWatch dashboard for Apache

### Step 3: Onboard your Apache EC2 instances (optional)

Steps 1 and 2 covered how to deploy a demo web server and monitor the workload on a CloudWatch dashboard. To start using this solution for your own use cases, you will need to onboard your EC2 instances on the Amazon CloudWatch dashboard.

To onboard your instances on the dashboard, follow these steps:

1. Ensure that the appropriate logs and metrics are sent to Amazon CloudWatch from your EC2 instance(s). For more information on how to send logs and metrics, refer to [Configuring your Apache EC2 instance](#), [Configuring your NGINX EC2 instance](#), or [Configuring your Puma instance](#).
2. After configuring your EC2 instances, tag your instances with the same tag you provided in the **<workload> tagging schema** parameter. You can add or remove instances from the dashboard by adding or removing the tag from your instances. For more information, refer to [Add or remove EC2 tag](#).

# Monitor the solution with Service Catalog AppRegistry

This solution includes a Service Catalog AppRegistry resource to register the CloudFormation template and underlying resources as an application in both [Service Catalog AppRegistry](#) and [AWS Systems Manager Application Manager](#).

AWS Systems Manager Application Manager gives you an application-level view into this solution and its resources so that you can:

- Monitor its resources, costs for the deployed resources across stacks and AWS accounts, and logs associated with this solution from a central location.
- View operations data for the resources of this solution (such as deployment status, CloudWatch alarms, resource configurations, and operational issues) in the context of an application.

The following figure depicts an example of the application view for the solution stack in Application Manager.

The screenshot displays the AWS Systems Manager Application Manager console. On the left, a sidebar shows a list of components under 'Components (2)', including 'AWS-Systems-Manager-Application-Manager' and 'AWS-Systems-Manager-A'. The main content area is titled 'AWS-Systems-Manager-Application-Manager' and features a 'Start runbook' button. Below the title is the 'Application information' section, which includes fields for 'Application type' (AWS-AppRegistry), 'Name' (AWS-Systems-Manager-Application-Manager), and 'Application monitoring' (Not enabled). A 'View in AppRegistry' link is also present. Below this is a navigation bar with tabs for Overview, Resources, Instances, Compliance, Monitoring, OpsItems, Logs, Runbooks, and Cost. The 'Overview' tab is active, showing 'Insights and Alarms' and 'Cost' sections. The 'Insights and Alarms' section includes a 'View all' button and a description: 'Monitor your application health with Amazon CloudWatch.' The 'Cost' section includes a 'View all' button and a description: 'View resource costs per application using AWS Cost Explorer.' Below the 'Cost' section, there is a 'Cost (USD)' field.

## Solution stack in Application Manager

## Activate CloudWatch Application Insights

1. Sign in to the [Systems Manager console](#).

2. In the navigation pane, choose **Application Manager**.
3. In **Applications**, search for the application name for this solution and select it.

The application name will have App Registry in the **Application Source** column, and will have a combination of the solution name, Region, account ID, or stack name.

4. In the **Components** tree, choose the application stack you want to activate.
5. In the **Monitoring** tab, in **Application Insights**, select **Auto-configure Application Insights**.

The screenshot shows the AWS Application Insights interface. At the top, there is a navigation bar with tabs: Overview, Resources, Provisioning, Compliance, Monitoring (selected), OpsItems, Logs, Runbooks, and Cost. Below the navigation bar, the main content area is titled "Application Insights (0) Info". There is a toggle for "View Ignored Problems" (currently off), an "Actions" dropdown menu, and an "Add an application" button. Below this, there is a search bar with the placeholder text "Find problems", a "Last 7 days" filter, a refresh button, and navigation arrows. A table header is visible with columns: Problem su..., Status, Severity, Source, Start time, and Insights. The main content area displays a message: "Advanced monitoring is not enabled". Below this message, there is explanatory text: "When you onboard your first application, a service-linked role (SLR) is created in your account. The SLR is predefined by CloudWatch Application Insights and includes the permissions the service requires to monitor AWS services on your behalf." At the bottom of this section, there is a button labeled "Auto-configure Application Insights".

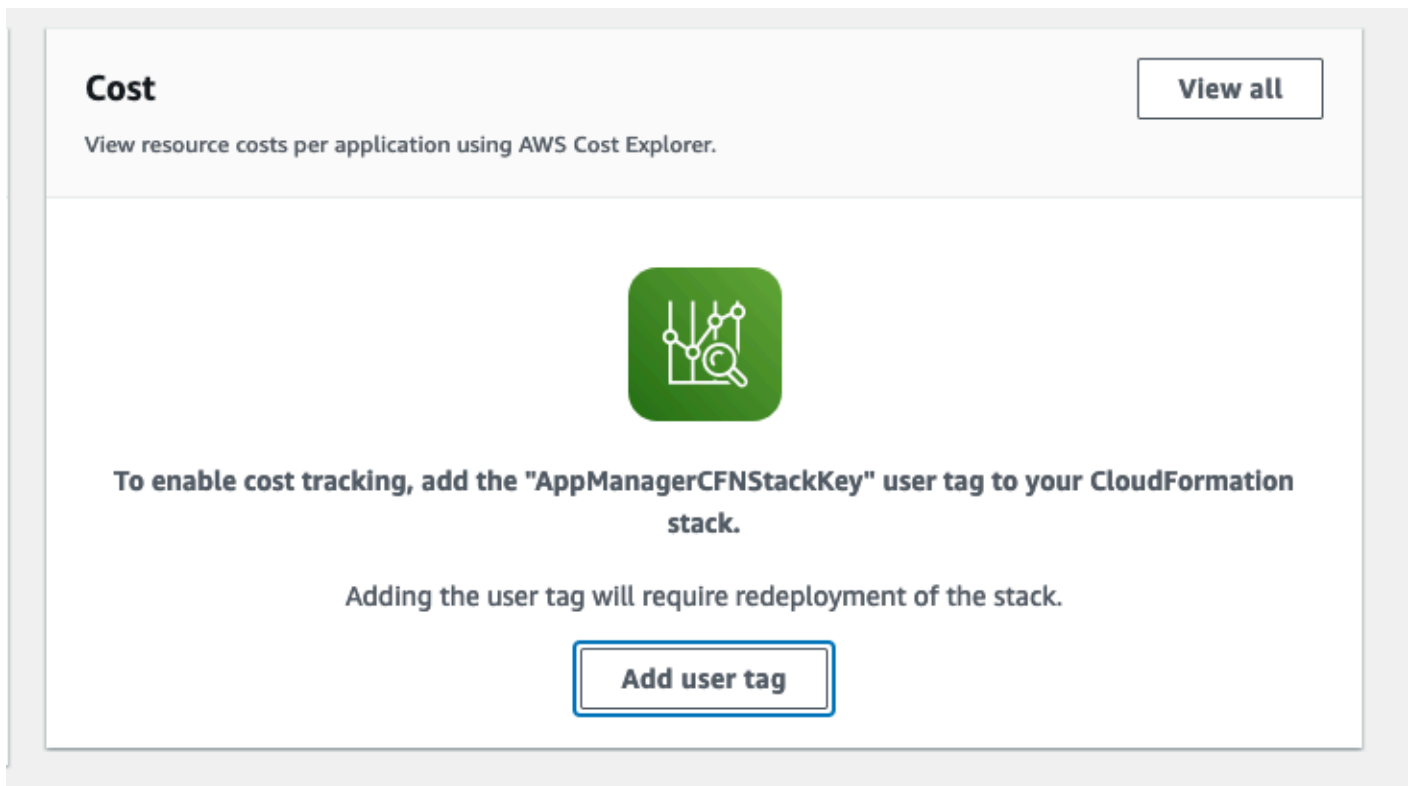
Monitoring for your applications is now activated and the following status box appears:

The screenshot shows the AWS Application Insights interface, similar to the previous one, but with a success message displayed in a green-bordered box. The message reads: "Application monitoring has been successfully enabled. It will take some time to display any results. Please use the refresh button to view results." The rest of the interface, including the navigation bar, search bar, and table header, remains the same.

## Confirm cost tags associated with the solution

After you activate cost allocation tags associated with the solution, you must confirm the cost allocation tags to see the costs for this solution. To confirm cost allocation tags:

1. Sign in to the [Systems Manager console](#).
2. In the navigation pane, choose **Application Manager**.
3. In **Applications**, choose the application name for this solution and select it.
4. In the **Overview** tab, in **Cost**, select **Add user tag**.



5. On the **Add user tag** page, enter `confirm`, then select **Add user tag**.

The activation process can take up to 24 hours to complete and the tag data to appear.

## Activate cost allocation tags associated with the solution

After you confirm the cost tags associated with this solution, you must activate the cost allocation tags to see the costs for this solution. The cost allocation tags can only be activated from the management account for the organization.

To activate cost allocation tags:

1. Sign in to the [AWS Billing and Cost Management and Cost Management console](#).
2. In the navigation pane, select **Cost Allocation Tags**.
3. On the **Cost allocation tags** page, filter for the AppManagerCFNStackKey tag, then select the tag from the results shown.
4. Choose **Activate**.

## AWS Cost Explorer

You can see the overview of the costs associated with the application and application components within the Application Manager console through integration with AWS Cost Explorer. Cost Explorer helps you manage costs by providing a view of your AWS resource costs and usage over time.

1. Sign in to the [AWS Cost Management console](#).
2. In the navigation menu, select **Cost Explorer** to view the solution's costs and usage over time.



## Update the solution

If you have previously deployed the solution, follow this procedure to update the CloudWatch Monitoring on AWS CloudFormation stack to get the latest version of the solution's framework.

1. Sign in to the [AWS CloudFormation console](#), select your existing amazon-cloudwatch-monitoring-framework CloudFormation stack, and select **Update**.
2. Select **Replace current template**.
3. Under **Specify template**:
  - a. Select **Amazon S3 URL**.
  - b. Copy the link of the [latest template](#).
  - c. Paste the link in the **Amazon S3 URL** box.
  - d. Verify that the correct template URL shows in the **Amazon S3 URL** text box, and choose **Next**. Choose **Next** again.
4. Under **Parameters**, review the parameters for the template and modify them as necessary. Refer to [Step 1. Launch the Stack](#) for details about the parameters.
5. Choose **Next**.
6. On the **Configure stack options** page, choose **Next**.
7. On the **Review** page, review and confirm the settings. Check the box acknowledging that the template might create IAM resources.
8. Choose **View change set** and verify the changes.
9. Choose **Update stack** to deploy the stack.

You can view the status of the stack in the AWS CloudFormation console in the **Status** column. You should see a status of UPDATE\_COMPLETE in approximately five minutes.

# Apache key performance indicators

This solution deploys a dashboard for Apache, which allows you to monitor the following key performance indicators by default:

- Apache traffic patterns.
  - Apache Traffic by RemoteIP: Top 10 Remote IPs
  - Apache Traffic by Host: Top 10 Hosts
- Apache throughput and latency metrics to report on the health of individual servers and detect bottlenecks or other performance problems.
  - Apache Traffic by ResponseTime: The response time for requests
  - Apache Traffic by BytesSent: Bytes sent in response to requests
  - Apache Traffic by BytesReceived: Bytes received in request payloads
- Apache Host-level resource metrics to determine whether to scale servers up or out.
  - Host mem\_used\_percent: The percentage of memory currently in use
  - Host tcp\_established: The number of TCP connections established
  - Apache CPU: CPU utilized by httpd process
- Apache status to identify misconfigured files or other underlying issues.
  - Apache Traffic by Status: HTTP status for requests

# NGINX key performance indicators

This solution deploys a dashboard for NGINX, which allows you to monitor the following key performance indicators by default:

- NGINX traffic patterns.
  - NGINX Traffic by RemoteIP: Top 10 Remote IPs
  - NGINX Traffic by Host: Top 10 Hosts
- NGINX throughput and latency metrics to report health of individual servers.
  - NGINX Traffic by ResponseTime: The response time for requests
  - NGINX Traffic by BytesSent: Bytes sent in response to requests
  - NGINX Traffic by BytesReceived: Bytes received in request payloads
- NGINX host-level resource metrics to determine scaling up or down.
  - Host mem\_used\_percent: The percentage of memory currently in use
  - Host tcp\_established: The number of TCP connections established
  - Total Requests: Total number of web server requests
  - Total Active Connections: Current active web server-client connections
  - NGINX CPU: CPU utilized by NGINX process
- NGINX status to identify issues.
  - NGINX Traffic by Status: HTTP status for requests

# Puma key performance indicators

This solution deploys a dashboard for Puma, which allows you to monitor the following key performance indicators by default:

- Puma traffic patterns.
  - Puma Traffic by RemoteIP: Top 10 Remote IPs
  - Puma Traffic by Host: Top 10 Hosts
- Puma throughput and latency metrics to report on the health of individual servers and detect bottlenecks or other performance problems.
  - Puma Traffic by ResponseTime: The response time for requests
  - Puma Traffic by BytesSent: Bytes sent in response to requests
  - Puma Traffic by BytesReceived: Bytes received in request payloads
- Puma Host-level resource metrics to determine whether to scale servers up or out.
  - Host mem\_used\_percent: The percentage of memory currently in use
  - Host tcp\_established: The number of TCP connections established
  - Puma CPU: CPU utilized by bundle process
- Puma status to identify misconfigured files or other underlying issues.
  - Puma Traffic by Status: HTTP status for requests
- Puma server metrics to identify application performance on server.
  - Pool Capacity: The number of maximum concurrent connections possible
  - Max Threads Count: The number of maximum threads allowed at any time
  - Running Threads Count: The number of threads currently running
  - Workers Count: The number of workers currently running
  - Requests Count: The number of requests made to Puma web server
- Custom Puma metrics supported by StatsD to measure application performance.
  - Page Load Time: Time elapsed to load page
  - Page Hit Count: The number of times page accessed

## Configuring your workload instances

The dashboard deployed by the solution assumes that your instances are configured correctly and send the appropriate metrics and logs to Amazon CloudWatch. If your EC2 instances do not send metrics and logs in the appropriate format, they will not appear on the dashboard.

The demo instances created by the workload demo templates are pre-configured with these configurations and can be used for reference purpose. Refer to [AWS CloudFormation templates](#) for details on demo templates.

**Note:** You can use AWS Systems Manager *run* command to configure your instances (highlighted below in Configuring EC2 instance sections). For more information, refer to [AWS Systems Manager Run Command](#) in the *AWS Systems Manager User Guide*.

## Configuring your Apache EC2 instance

For more information about metrics and logs, refer to [Apache key performance indicators](#).

### Amazon CloudWatch agent configuration for Apache

Refer to the sample configuration files `apache.json` and `linux_cw_infra.json` to ensure that your EC2 instance CloudWatch agent is sending metrics and logs in the appropriate format.

The following example code is from the [apache.json](#) file. This code identifies the appropriate process-level metrics being sent to Amazon CloudWatch.

```
"metrics_collected": {
  "procstat": [
    {
      "exe": "httpd",
      "measurement": [
        "cpu_usage",
        "memory_rss",
        "memory_vms",
        "read_bytes",
        "write_bytes",
        "read_count",
        "write_count"
      ]
    }
  ]
}
```

```
    ]
  }
]
```

Similarly, the following example code is also from the `apache.json` and it identifies the web server access log file path and corresponding CloudWatch log group name. The CloudWatch log group name should be consistent with [Access Log Group parameter value in the workload template](#).

```
"logs": {
  "logs_collected": {
    "files": {
      "collect_list": [
        {
          "file_path": "/var/log/www/error/*",
          "log_group_name": "/cw-monitoring-framework/apache/error",
          "log_stream_name": "{instance_id}"
        },
        {
          "file_path": "/var/log/www/access/*",
          "log_group_name": "/cw-monitoring-framework/apache/access",
          "log_stream_name": "{instance_id}"
        }
      ]
    }
  }
}
```

## httpd.conf for Apache web server

The following example code shows the appropriate log format for the Apache access log in the [httpd.conf](#) file to support CloudWatch dashboard queries. You can modify this file to customize log data that you want to display on your Apache dashboard. Additionally, the data points and schema in this file must be consistent with ApacheLogWidgets in the [widget export file](#).

```

<IfModule log_config_module>
#
# The following directives define some format nicknames for use with
# a CustomLog directive (see below).
#
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %b" common

<IfModule logio_module>
# You need to enable mod_logio.c to use %I and %O
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\" %I %O" combinedio
LogFormat "{ \"time\": \"%Y-%m-%d %T.%t.%{msec_frac}tZ\", \"process\": \"%D\", \"filename\": \"%f\", \"remoteIP\": \"%a\", \"host\": \"%V\"",
</IfModule>

```

```

{
  type: "log",
  x: 0,
  y: 0,
  width: 12,
  height: 6,
  properties: {
    view: "pie",
    title: "Apache Traffic by RemoteIP",
    region: process.env.AWS_REGION as string,
    query: "SOURCE \"${process.env.ACCESS_LOG_GROUP}\" | %filter%\n stats count [remoteIP] as Count by remoteIP\n sort
  },
},

```

```

{
  type: "log",
  x: 12,
  y: 0,
  width: 12,
  height: 6,
  properties: {
    view: "pie",
    title: "Apache Traffic by Host",
    region: process.env.AWS_REGION as string,
    query: "SOURCE \"${process.env.ACCESS_LOG_GROUP}\" | %filter%\n stats count [host] as Count by host\n sort Count
  },
},

```

## httpd.conf

# Configuring your NGINX EC2 instance

For more information about metrics and logs, refer to [NGINX key performance indicators](#).

## Amazon CloudWatch agent configuration for NGINX

Refer to the sample configuration files, `nginx.json`, and `linux_cw_infra.json` to ensure that your EC2 instance CloudWatch agent is sending metrics and logs in the appropriate format.

The following example code is from the [nginx.json](#) file. This code identifies the appropriate process-level metrics being sent to Amazon CloudWatch.

```

"metrics_collected": {
  "procstat": [
    {
      "exe": "nginx",
      "measurement": [
        "cpu_usage",
        "memory_rss",
        "memory_vms",
        "read_bytes",
        "write_bytes",
        "read_count",
        "write_count"
      ]
    }
  ]
}

```

```
    }  
  ]  
}
```

Similarly, the following example code is also from the `nginx.json`. It identifies the web server access log file path and corresponding CloudWatch log group name. The CloudWatch log group name should be consistent with [Access Log Group parameter value in the workload template](#).

```
{  
  "logs": {  
    "logs_collected": {  
      "files": {  
        "collect_list": [  
          {  
            "file_path": "/var/log/nginx/access.log",  
            "log_group_name": "/cw-monitoring-framework/nginx/access",  
            "log_stream_name": "{instance_id}"  
          },  
          {  
            "file_path": "/var/log/nginx/error.log",  
            "log_group_name": "/cw-monitoring-framework/nginx/error",  
            "log_stream_name": "{instance_id}"  
          }  
        ]  
      }  
    }  
  }  
}
```

## nginx.conf for NGINX web server

The following example code shows the appropriate log format for the NGINX access log in the [nginx.conf](#) file, to support CloudWatch dashboard queries. You can modify this file to customize log data that you want to display on your NGINX dashboard. Additionally, the data points and schema in this file must be consistent with `NginxLogWidgets` in the [widget export file](#).



```

http {
    log_format json_combined
    '{
        "time": "$time_iso8601",
        "process": "$pid",
        "filename": "$request_filename",
        "remoteIP": "$remote_addr",
        "method": "$request_method",
        "request": "$request_uri",
        "status": "$status",
        "responseTime": "$request_time",
        "referer": "$http_referer",
        "userAgent": "$http_user_agent",
        "bytesSent": "$bytes_sent",
        "bytesReceived": "$request_length",
        "host": "$host",
        "connection_requests": "$connection_requests",
        "connection_active": "$connections_active",
        "connection_read": "$connections_reading",
        "connection_write": "$connections_writing",
        "connection_wait": "$connections_waiting"
    }';
    access_log /var/log/nginx/access.log json_combined;
}

```

```

// Log widgets - cloudwatch log insights queries
// =====
export const NginxLogWidgets: ILogWidget[] = [
  /**
   * @description remote IP widget for dashboard
   */
  {
    type: "log",
    x: 0,
    y: 0,
    width: 12,
    height: 6,
    properties: {
      view: "pie",
      title: "Nginx Traffic by RemoteIP",
      region: process.env.AWS_REGION as string,
      query: `SOURCE "${process.env.ACCESS_LOG_GROUP}" | %%filter%%\n| stats count(remoteIP) as Count by rem
    },
  },
  /**
   * @description host widget for dashboard
   */
  {
    type: "log",
    x: 12,
    y: 0,
    width: 12,
    height: 6,
    properties: {
      view: "pie",
      title: "Nginx Traffic by Host",
      region: process.env.AWS_REGION as string,
      query: `SOURCE "${process.env.ACCESS_LOG_GROUP}" | %%filter%%\n| stats count(host) as Count by hc
    },
  },
]

```

## nginx.conf

# Configuring your Puma EC2 instance

For more information about metrics and logs, refer to [Puma key performance indicators](#).

## Amazon CloudWatch agent configuration for Puma

Refer to the sample configuration files `puma.json` and `linux_cw_infra.json` to ensure that your EC2 instance CloudWatch agent is sending metrics and logs in the appropriate format.

The following example code is from the `puma.json` file. This code identifies the appropriate `statsd` and process-level metrics being sent to Amazon CloudWatch.

```

"metrics_collected": {
  "statsd": {
    "service_address": ":8125",
    "metrics_collection_interval": 10,
    "metrics_aggregation_interval": 10
  },
  "procstat": [
    {
      "exe": "ruby",
      "measurement": [

```

```

        "cpu_usage",
        "memory_rss",
        "memory_vms",
        "read_bytes",
        "write_bytes",
        "read_count",
        "write_count",
        "pid_count",
        "realtime_priority",
        "nice_priority"
    ]
}
]
}

```

Similarly, the following example code is also from the `puma.json`. It identifies the web server access log file path and corresponding CloudWatch log group name. The CloudWatch log group name should be consistent with [Access Log Group parameter value in the workload template](#).

```

"logs": {
  "logs_collected": {
    "files": {
      "collect_list": [
        {
          "file_path": "/root/sample-app/log/puma.stderr.log",
          "log_group_name": "/cw-monitoring-framework/puma/stderror",
          "log_stream_name": "{instance_id}"
        },
        {
          "file_path": "/root/sample-app/log/puma.stdout.log",
          "log_group_name": "/cw-monitoring-framework/puma/out",
          "log_stream_name": "{instance_id}"
        },
        {
          "file_path": "/root/sample-app/log/production.log",
          "log_group_name": "/cw-monitoring-framework/puma/access",
          "log_stream_name": "{instance_id}"
        }
      ]
    }
  }
}
}

```

## Configurations for Puma web server

Following configuration files are used to bootstrap the demo Puma instance as well.

- [puma.rb](#): The following code example shows how to configure `stdout` and `stderr` logs on the Puma web server. The log file paths should be consistent with the file paths in [CloudWatch agent configurations](#). Additionally, `statsd` plugin needs to be turned on as well.

```
#define location of stdout and stderr logs
stdout_redirect "/root/sample-app/log/puma.stdout.log", "/root/sample-app/log/
puma.stderr.log", true

# Enable collection and sending of StatsD metrics
plugin :statsd
```

- [application\\_controller.rb](#): The following code example shows the appropriate log format in the `application_controller.rb` file to support CloudWatch dashboard queries. Additionally, the data points and schema in this file must be consistent with `PumaLogWidgets` in the [widget export file](#).

```
def generate_cloudwatch_log_json
  start_request_time = Time.now
  yield
  total_request_time = Time.now - start_request_time
  logging_hash = {
    time: Time.now,
    request_uuid: request.uuid,
    method: request.method,
    remoteIP: request.ip,
    host: Socket.gethostname,
    parameters: request.parameters,
    bytesSent: request.content_length,
    bytesReceived: response.body.size,
    filename: request.original_fullpath,
    request: request.original_url,
    responseTime: total_request_time,
    user_agent: request.user_agent
  }
}

// Log widgets - cloudwatch log insights queries
// Log widgets - cloudwatch log insights queries
export const PumaLogWidgets: ILogWidget[] = [
  // @description remote IP widget for dashboard
  {
    type: "log",
    x: 0,
    y: 0,
    width: 12,
    height: 6,
    properties: {
      view: "pie",
      title: "Puma Traffic by RemoteIP",
      region: process.env.AWS_REGION as string,
      query: `SOURCE="${process.env.ACCESS_LOG_GROUP}" | %filter%\n stats count(remoteIP) as Count by remoteIP\`
    }
  },
  // @description host widget for dashboard
  {
    type: "log",
    x: 12,
    y: 0,
    width: 12,
    height: 6,
    properties: {
      view: "pie",
      title: "Puma Traffic by Host",
      region: process.env.AWS_REGION as string,
      query: `SOURCE="${process.env.ACCESS_LOG_GROUP}" | %filter%\n stats count(host) as Count by host\`
    }
  },
]
```

### application\_controller.rb

- [production.rb](#): The following code example shows how to configure the log format by commenting out request id tags and prepending the logs. This turns on full support for the CloudWatch dashboard queries.

```
# Prepend all log lines with the following tags.
# comment out line 55 in order to populate desired log metrics on CloudWatch
dashboard
```

```
# config.log_tags = [ :request_id ]
```

- [index.html.erb](#): Default home page for the Puma web server.
- [routes.rb](#): Directs to the index page of the sample Rails application preinstalled on the Puma demo instance.
- [statsd\\_instrument.rb](#): Initial configuration for statsd module.
- [page\\_controller.rb](#): Uses statsd instrument gem to add any additional statsd supported metrics
- [supervisord.conf](#): Supervisor helps manage statsd processes and auto restart statsd processes in case of failure

## Add or remove EC2 instance tags

To add or remove EC2 instances from the dashboard, you can add or remove the pre-determined tag at any point. These changes update on the dashboard in approximately five minutes.

To add an Amazon EC2 tag:

1. Sign in to the [Amazon EC2 console](#).
2. Select the instance you want to onboard to the dashboard
3. Select the **Tags** tab. Choose **Manage tags**.
4. Choose **Add tag** to the instance and provide the key-value pair you provided during deployment. If you used the default value during deployment, it would look as follows for the Apache workload.

Key	Value - <i>optional</i>	
<input type="text" value="CW-Dashboard"/>	<input type="text" value="Apache"/>	<input type="button" value="Remove"/>
<input type="button" value="Add tag"/>		

### ***Add Amazon EC2 tag***

5. Choose **Save**.

To remove an Amazon EC2 tag:

1. Sign in to the [Amazon EC2 console](#).
2. Select the instance you want to onboard to the dashboard
3. Select the **Tags** tab. Choose **Manage tags**.
4. Identify the workload specific tag, choose **Remove**.
5. Choose **Save**.

## Extending to more workloads

This solution currently only supports Apache workloads. To extend this solution for other workloads, refer to the [README.md](#) file in the GitHub repository.

# Troubleshooting

Use the following common error scenarios to troubleshoot the issue. Before troubleshooting, we recommend enabling **debug** mode.

This solution logs error, warning, informational, and debugging messages for the AWS Lambda functions. To choose the type of messages to log, locate the applicable function in the AWS Lambda console and change the LOG\_LEVEL environment variable to the applicable type of message.

Level	Description	
ERROR	Logs will include information on anything that causes an operation to fail.	
WARNING	Logs will include information on anything that can potentially cause inconsistencies in the function, but might not necessarily cause the operation to fail. Logs will also include ERROR messages	
INFO	Logs will include high-level information about how the function is operating. Logs will also include ERROR and WARNING messages.	
DEBUG	Logs will include information that might be helpful when debugging a problem with the function. Logs will also include ERROR, WARNING, and INFO messages.	

If these instructions don't address your issue, see the [Contact AWS Support](#) section for instructions on opening an AWS Support case for this solution.

## Common errors

### My instance doesn't show up on the dashboard.

**Issue:** The instance does not show up on the CloudWatch dashboard despite adding the same tag that is provided during stack deployment

**Reason:** There could be multiple issues:

1. The Lambda function runs at interval of five minutes by default, so after tagging the instance, wait a few minutes for it to reflect on the dashboard.
2. The tag is a case-sensitive key-value pair. Ensure the tag added to the instance matches the parameter input, provided during stack deployment.
3. Ensure the CloudWatch agent and `httpd.config` files on your EC2 instance are configured correctly. For more details, refer to [Configuring your Apache EC2 instance](#) or [Configuring your Puma EC2 instances](#).

### My EC2 Apache, NGINX, or Puma access logs appear on the CloudWatch Logs but they are not showing on the dashboard

**Issue:** You have set up CloudWatch agent configurations on your EC2 instance and can see your instance Apache/NGINX/Puma access logs on CloudWatch logs, but the widgets do not show data points on the dashboard.

**Reason:** The CloudWatch log group name for access logs might be different in the CloudWatch agent config file than in `dashboardHandler` Lambda environment variable.

**Resolution:**

1. Validate the file path and log group name for Apache access logs in your CloudWatch agent config file.

```
"logs": {
  "logs_collected": {
    "files": {
```



```

    "collect_list": [
      {
        "file_path": "/var/log/www/error/*",
        "log_group_name": "/cw-monitoring-framework/apache/error",
        "log_stream_name": "{instance_id}"
      },
      {
        "file_path": "/var/log/www/access/*",
        "log_group_name": "/cw-monitoring-framework/apache/access",
        "log_stream_name": "{instance_id}"
      }
    ]
  }
},

```

2. Check the environment variable in `dashboardHandler` and ensure that it matches to the log group name from step 1.

#### Environment variables (10)

The environment variables below are encrypted at rest with the default Lambda service key.

Key	Value
ACCESS_LOG_GROUP	/cw-monitoring-framework/apache/access
CUSTOM_SDK_USER_AGENT	AwsSolution/SO0158/v1.0.0
DASHBOARD_NAME	ApacheDashboard-eu-central-1
LOG_LEVEL	debug
METRICS_ENDPOINT	https://metrics.awssolutionsbuilder.com/generic
SEND_METRIC	Yes
SSM_PARAMETER	/cw-monitoring-framework/ApacheInstances
START_TIME	-PT12H
TAG_SCHEMA	{"Key": "CW-Dashboard", "Value": "Apache"}
WORKLOAD	Apache

### Access log group

## My instances continue to show up on the dashboard even after being terminated.

**Issue:** Instance(s) show(s) up on the dashboard even though they are in terminated state.

**Reason:** The CloudWatch dashboard widgets take some time to update and remove terminated instances

## Resolution:

1. Allow the dashboard widgets time to update and reflect the change. The widgets should eventually be consistent and not show terminated instances.
2. Remove the added dashboard tag ([Add/remove EC2 tag](#)) from the workload instance(s) before terminating them.

## Contact AWS Support

If you have [AWS Developer Support](#), [AWS Business Support](#), or [AWS Enterprise Support](#), you can use the Support Center to get expert assistance with this solution. The following sections provide instructions.

### Create case

1. Sign in to [Support Center](#).
2. Choose **Create case**.

### How can we help?

1. Choose **Technical**.
2. For **Service**, select **Solutions**.
3. For **Category**, select **Other Solutions**.
4. For **Severity**, select the option that best matches your use case.
5. When you enter the **Service**, **Category**, and **Severity**, the interface populates links to common troubleshooting questions. If you can't resolve your question with these links, choose **Next step: Additional information**.

### Additional information

1. For **Subject**, enter text summarizing your question or issue.
2. For **Description**, describe the issue in detail.
3. Choose **Attach files**.
4. Attach the information that AWS Support needs to process the request.

## Help us resolve your case faster

1. Enter the requested information.
2. Choose **Next step: Solve now or contact us**.

## Solve now or contact us

1. Review the **Solve now** solutions.
2. If you can't resolve your issue with these solutions, choose **Contact us**, enter the requested information, and choose **Submit**.

# Uninstall the solution

To uninstall the CloudWatch Monitoring on AWS solution, delete the AWS CloudFormation stack. This will delete all the resources created by the template except for the CloudWatch Log groups. You must manually delete CloudWatch Log groups.

Use the following procedure to uninstall the solution.

## Using the AWS Management Console

1. Sign in to the [AWS CloudFormation console](#).
2. Select this solution's installation stack.
3. Choose **Delete**.

## Delete CloudWatch Log groups

CloudWatch Log groups are retained since they may contain data. To delete these log groups, use the steps below:

1. Sign in to the [AWS CloudWatch Logs console](#).
2. Identify the log groups that are no longer needed. You may find similar log groups in your console.

```
/aws/lambda/XXDashboardHandlerXX  
/aws/lambda/XXhelperProviderframeworkXX  
/aws/lambda/XXTagHandlerXX  
/aws/lambda/XXApach-HelperXX  
/cw-monitoring-framework/apache/*  
/cw-monitoring-framework/nginx/*  
/cw-monitoring-framework/puma/*  
  
linux/var/log/boot  
linux/var/log/cron  
linux/var/log/messages  
linux/var/log/secure
```

3. Select the log groups.

4. Choose **Actions** and choose **delete log group(s)**.

## Using AWS Command Line Interface

Determine whether the AWS Command Line Interface (AWS CLI) is available in your environment. For installation instructions, refer to [What Is the AWS Command Line Interface](#) in the *AWS CLI User Guide*. After confirming that the AWS CLI is available, run the following command.

```
$ aws cloudformation delete-stack --stack-name <installation-stack-name>
```

## Developer guide

This section provides the source code for the solution.

### Source code

Visit our [GitHub repository](#) to download the source files for this solution and to share your customizations with others. The Application Monitoring with Amazon CloudWatch templates are generated using the [AWS Cloud Development Kit \(AWS CDK\)](#). Refer to the [README.md](#) file for additional information.

# Reference

## Anonymized data collection

This solution includes an option to send anonymous operational metrics to AWS. We use this data to better understand how customers use this solution and related services and products. When invoked, the following information is collected and sent to AWS:

- **Solution ID** - The AWS solution identifier
- **Unique ID (UUID)** - Randomly generated, unique identifier for each CloudWatch Monitoring on AWS deployment
- **Timestamp**- Data-collection timestamp
- **Launch Data** - Relevant launch data for the solution

Example data:

```
Event: SolutionLaunched/SolutionDeleted  
Version: v1.1.0
```

AWS owns the data gathered through this survey. Data collection is subject to the [AWS Privacy Notice](#).

To opt out of this feature, modify the AWS CloudFormation template mapping section as follows:

1. Download the [AWS CloudFormation](#) template to your local hard drive.
2. Open the AWS CloudFormation template with a text editor.
3. Modify the AWS CloudFormation template mapping section from:

```
"Mappings":{  
  "StackMap":{  
    "Metric":{  
      "SendAnonymizedMetric":"Yes"  
    },  
  },  
}
```

to:

```
"Mappings":{
  "StackMap":{
    "Metric":{
      "SendAnonymizedMetric":"No"
    },

```

4. Sign in to the [AWS CloudFormation console](#).
5. Select **Create stack**.
6. On the **Create stack** page, **Specify template** section, select **Upload a template file**.
7. Under **Upload a template file**, select **Choose file** and select the edited template from your local drive.
8. Choose **Next** and follow the steps in [Launch the stack](#) in the [Deploy the solution](#) section of this guide.

## Contributors

- Abe Wubshet
- Aaron Schuetter
- Austin Park
- Garvit Singh
- Varun Jasti
- Omair Qazi
- Doug Toppin



# Revisions

Date	Change
April 2021	Initial release
November 2021	Release v1.1.0: Added support for NGINX and Puma workloads. For more information, refer to the <a href="#">CHANGELOG.md</a> file in the GitHub repository
September 2022	Release v1.1.1: Updated library dependencies. For more information, refer to the <a href="#">CHANGELOG.md</a> file in the GitHub repository
December 2022	Release v1.2.0: Added support for AppRegistry integration and various updates. For more information, refer to the <a href="#">CHANGELOG.md</a> file in the GitHub repository.
January 2023	Release v1.2.1: Updated library dependencies. Revised implementation guide structure for ease of readability and navigation. For more information, refer to the <a href="#">CHANGELOG.md</a> file in the GitHub repository.
June 2023	Release v1.2.2: Updated library dependencies. For more information, refer to the <a href="#">CHANGELOG.md</a> file in the GitHub repository.
October 2023	Release v1.2.3: Updated Lambda runtime to Nodejs 18. Updated library dependencies. For more information, refer to the <a href="#">CHANGELOG.md</a> file in the GitHub repository.
October 2023	Release v1.2.4: Updated package versions to resolve security vulnerabilities. For more

Date	Change
	information, refer to the <a href="#">CHANGELOG.md</a> file in the GitHub repository.
November 2023	Documentation update: Added <a href="#">Confirm cost tags associated with the solution</a> to the Monitoring the solution with AWS Service Catalog AppRegistry section.
June 2024	Release v1.2.5: Updated package versions to resolve security vulnerabilities. For more information, refer to the <a href="#">CHANGELOG.md</a> file in the GitHub repository.

# Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents AWS current product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers, or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. AWS responsibilities and liabilities to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

CloudWatch Monitoring on AWS is licensed under the terms of the Apache License Version 2.0 available at [The Apache Software Foundation](#).