



Implementation Guide

DevOps Monitoring Dashboard on AWS



DevOps Monitoring Dashboard on AWS: 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

Solution overview	1
Features and benefits	2
Use cases	3
Concepts and definitions	4
Architecture overview	6
Architecture diagram	6
AWS Well-Architected design considerations	7
Operational excellence	7
Security	8
Reliability	8
Performance efficiency	8
Cost optimization	9
Sustainability	9
Architecture details	10
Amazon EventBridge events rule	10
Amazon Kinesis Data Firehose	10
AWS Lambda	11
Amazon S3	11
AWS Glue and Amazon Athena	11
Amazon QuickSight	12
CloudWatch Synthetics canary and CloudWatch alarm	12
Amazon CloudWatch alarm for AWS CodePipeline	12
Amazon API Gateway	13
AWS services in this solution	13
Plan your deployment	15
Cost	15
Example 1: Turn on QuickSight without GitHub	15
Example 2: Turn on QuickSight, GitHub with Secrets Manager	16
Security	17
IAM roles	17
Amazon S3	17
AWS CI/CD pipeline deployment	17
Amazon QuickSight deployment	18
Amazon CloudWatch alarm for Amazon CloudWatch Synthetics canary deployment	18

Amazon CloudWatch alarm for AWS CodePipeline deployment	18
Multi-account multi-Region deployment	18
Supported AWS Regions	19
Quotas	19
Quotas for AWS services in this solution	19
AWS CloudFormation quotas	19
Deploy the solution	20
AWS CloudFormation templates	20
Prerequisites	21
Deployment overview	22
Step 1: Retrieve the Amazon QuickSight Principal ARN	23
Step 2: Launch the stack	23
Step 3: Configure Amazon QuickSight	28
Step 4: Set up GitHub webhook	30
Monitor the solution with Service Catalog AppRegistry	32
Activate CloudWatch Application Insights	32
Confirm cost tags associated with the solution	34
Activate cost allocation tags associated with the solution	34
AWS Cost Explorer	35
Update the solution	36
Uninstall the solution	37
Using the AWS Management Console	37
Using AWS Command Line Interface	37
Troubleshooting	38
Contact AWS Support	38
Create case	38
How can we help?	38
Additional information	38
Help us resolve your case faster	39
Solve now or contact us	39
Use the solution	40
Set up Amazon CloudWatch synthetics canary and Amazon CloudWatch alarm	40
Set up Amazon CloudWatch alarm for AWS CodePipeline	43
Set up multi-account multi-Region data ingestion	44
Running queries and work with query results and output files in Amazon Athena	48
Build visualizations with Amazon Athena and Tableau	49

DevOps metrics list	50
Code change volume metrics	50
Mean time to recover metrics	51
Change failure rate metrics	51
Deployment metrics	51
Build metrics	51
Pipeline metrics	52
GitHub activity metrics	52
Database schema information	52
Amazon QuickSight dashboards visuals	58
Code change volume dashboard	58
Mean time to recover dashboard	59
Change failure rate dashboard	60
Deployment dashboards	61
Build dashboards	62
GitHub activity dashboard	70
Developer Guide	73
Source Code	73
Reference	74
Anonymized data collection	74
Contributors	76
Revisions	77
Notices	79

Ingesting, analyzing, and visualizing metrics with DevOps Monitoring Dashboard on AWS

Publication date: *March 2021* ([last update: March 2024](#))

Collecting performance and operational metrics in your continuous integration/continuous delivery (CI/CD) pipeline is important to measure your return on investment in DevOps automation. These metrics also inform you about how to improve your software delivery process. However, the process of aggregating, analyzing, and visualizing metrics from various components through the pipeline can be complex and time consuming.

The DevOps Monitoring Dashboard on AWS solution automates the process for monitoring and visualizing CI/CD metric following AWS best practices. This solution allows organizations of all sizes to track and measure the activities of their development teams. This helps DevOps leaders measure the impact of their DevOps initiatives and make data-driven decisions to drive continuous improvement in their development teams.

This solution supports ingestion, analysis, and visualization of data from [AWS Developer Tools](#) as well as GitHub repository to calculate key DevOps metrics, such as mean time to recovery (MTTR), change failure rate, deployment, build activity, pipeline activity, and Code Change Volume. For more information about the metrics, refer to [DevOps metrics list](#). These metrics are presented in [Amazon QuickSight](#) dashboards for visualization. For more information about data visualization, refer to [Amazon QuickSight dashboards visuals](#).

You can also use other visualization tools, such as Tableau, to build visualizations from the Amazon Athena database. For more information, refer to [Build visualizations with Amazon Athena and Tableau](#). To directly work with query results and output files in Amazon Athena, refer to [Running queries and work with query results and output files in Amazon Athena](#).

This implementation guide describes architectural considerations and configuration steps for deploying this solution in the Amazon Web Services (AWS) Cloud. This solution's [AWS CloudFormation](#) template launches and configures the AWS services required to deploy the solution using AWS best practices for security, availability, performance efficiency, and cost optimization.

This solution is intended for deployment in an enterprise by IT infrastructure architects, administrators, developers, and DevOps professionals who have practical experience with the AWS Cloud.

Use this navigation table to quickly find answers to these questions:

If you want to . . .	Read . . .
Know the cost for running this solution. The estimated cost for running this solution in the US East (N. Virginia) Region is USD \$44.25 per month for AWS resources (option without GitHub).	Cost
Understand the security considerations for this solution.	Security
Know how to plan for quotas for this solution.	Quotas
Know which AWS Regions support this solution.	Supported AWS Regions
View or download the AWS CloudFormation template included in this solution to automatically deploy the infrastructure resources (the “stack”) for this solution.	AWS CloudFormation template

Features and benefits

The DevOps Monitoring Dashboard on AWS solution provides the following features:

Automate near real-time streaming analytics of AWS CI/CD metrics

This solution automates ingestion and analysis of operational metrics in your AWS CI/CD pipeline consisting of AWS CodeCommit, CodeBuild, CodeDeploy, and CodePipeline. It calculates key DevOps metrics including mean time to recovery (MTTR), change failure rate, deployment, build activity, pipeline activity, and code change volume. With these metrics you can track and measure activities and health of your CI/CD environment.

Support cross-account cross-Region data ingestion

This solution streams data from multiple AWS accounts and AWS Regions into a central S3 bucket located in a single monitoring account, where data is aggregated and analyzed.

Automate visualization with Amazon QuickSight dashboard

This solution launches a set of pre-built Amazon QuickSight dashboards in a single monitoring account to visualize the solution's analyses. This makes it easy for you to gain quick insights into your CI/CD activities without having to build your own dashboard.

Integrate with GitHub

This solution is integrated with GitHub by ingesting GitHub events using an Amazon API endpoint and a GitHub webhook. It tracks and measures GitHub activities in near real-time. Currently it supports push events.

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

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

Use cases

The following are example use cases for using this solution. You can customize this solution in innovative ways that aren't limited to this list.

Automate monitoring CI/CD metrics in AWS cloud

Monitoring operational metrics in your AWS CI/CD pipeline is important to improve your software delivery process and measure your return on investment in DevOps. However, the process of aggregating, analyzing, and visualizing metrics from various components through the pipeline cross-account cross Region can be complex and time consuming. To help you, this solution automatically deploys resources needed for data ingestion, analysis, and visualization into your account with a CloudFormation template. This way, you don't need to build your own solution and can get started faster.

Automate monitoring operational metrics in GitHub

This solution streams GitHub events into its monitoring account in near real-time. You can turn on GitHub feature to monitor GitHub activities such as push events, which are used to calculate commits and pushes by author, repository, and branch.

Leverage the source code for applying customization or building your DevOps monitoring dashboard

This solution provides an example for how to use Amazon QuickSight and other services to build DevOps monitoring dashboard on the AWS Cloud. Its [open source code in GitHub](#) makes it convenient for you to apply customizations or build your own dashboard that fit your needs.

Concepts and definitions

This section describes key concepts and defines terminology specific to this solution:

DevOps

DevOps (a portmanteau of “development” and “operations”) is a combination of practices and tools designed to increase an organization’s ability to deliver applications and services faster than traditional software development processes.

CI/CD

CI/CD stand for continuous integration and continuous delivery. In very simple terms, CI is a modern software development practice in which incremental code changes are made frequently and reliably. Automated build-and-test steps invoked by CI ensure that code changes being merged into the repository are reliable. The code is then delivered quickly and seamlessly as part of the CD process. In the software world, the CI/CD pipeline refers to the automation that enables incremental code changes from developers’ desktops to be delivered quickly and reliably to production.

AWS developer tools

AWS provides a set of tools like software development kits (SDKs), code editors, and continuous integration and delivery (CI/CD) services for DevOps software development. In the solution, the developer tools specifically include AWS CodeCommit, CodeBuild, CodeDeploy and CodePipeline.

MTTR

MTTR (Mean time to recovery) is a metric that calculates average time it takes to recover from a product or system failure. This includes the full time of the outage—from the time the system or

product fails to the time that it becomes fully operational again. This solution calculates two types of MTTR: Synthetics canary and AWS CodePipeline.

Change failure rate

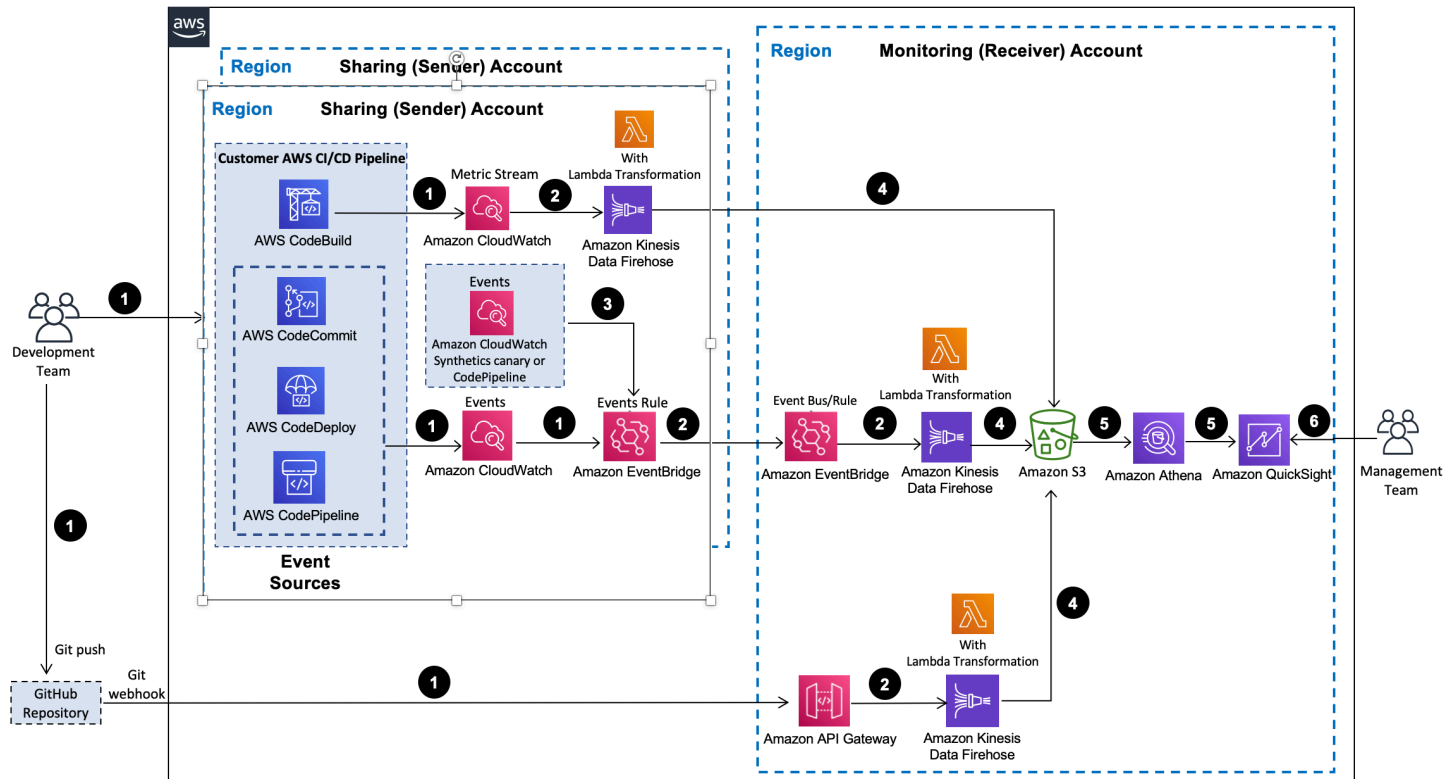
Change Failure Rate (CFR) is a metric that determines the changes that lead to failures after they reach production or are released to end-users, and it's expressed in percentages. To calculate CFR, you have to divide the number of failed deployments in production by the total number of product deployments. This solution calculates CFR of AWS CodeDeploy.

For a general reference of AWS terms, see the [AWS glossary](#) in the AWS General Reference.

Architecture overview

Architecture diagram

Deploying this solution with the default parameters deploys the following components in your AWS account.



DevOps Monitoring Dashboard on AWS architecture

This solution runs the following workflow:

1. A developer initiates an activity in an AWS CI/CD pipeline, such as pushing a code change to [AWS CodeCommit](#) or deploying an application using [AWS CodeDeploy](#). These activities create events. If a multi-account multi-Region feature is activated, the events can be generated from multiple AWS accounts and multiple AWS Regions. For development using GitHub repository, git push events are generated.
2. An [Amazon EventBridge](#) events rule detects the events based on predefined event patterns and then sends the event data to an [Amazon Data Firehose](#) delivery stream. One event rule is created per event source. For activities in [AWS CodeBuild](#), a CloudWatch metric stream is set up to capture CloudWatch metrics and deliver them to a Firehose delivery stream in the monitoring

account. For GitHub push events, an Amazon API endpoint is created to post these events and deliver them to a Firehose delivery stream.

3. An Amazon EventBridge events rule is also created to capture events from an Amazon CloudWatch alarm that monitors the status of an Amazon CloudWatch synthetics canary or [Amazon CodePipeline](#), if you have set up the alarm for the canary or pipeline respectively in your account. This alarm is needed to gather data for calculating MTTR metrics.
4. Amazon Data Firehose uses an AWS Lambda function for data transformation. The Lambda function extracts relevant data to each metric and sends it to a central Amazon S3 bucket in the monitoring account for downstream processing.
5. The data in Amazon S3 is linked to an Amazon Athena database, which runs queries against this data and returns query results to [Amazon QuickSight](#).
6. Amazon QuickSight obtains the query results and builds dashboard visualizations for your management team.

AWS Well-Architected design considerations

This solution uses the best practices from the [AWS Well-Architected Framework](#), which helps customers design and operate reliable, secure, efficient, and cost-effective workloads in the cloud.

This section describes how the design principles and best practices of the Well-Architected Framework benefit this solution.

Operational excellence

This section describes how we architected this solution using the principles and best practices of the [operational excellence pillar](#).

- The solution pushes metrics to [Amazon CloudWatch](#) to provide observability into the infrastructure, Lambda functions, Kinesis data firehose, API Gateway, AWS S3 buckets, and the rest of the solution components.
- The development, testing, and publishing of the solution is done through an AWS CI/CD pipeline that helps developers achieve high quality results consistently.
- The solution installation is done through a simple one-click deployment of AWS Cloudformation template. This will provision all the required resources in your account. To update or delete the solution, you only need to update or delete the template, which requires little time.

Security

This section describes how we architected this solution using the principles and best practices of the [security pillar](#).

- All inter-service communications use IAM roles.
- All multi-account communications use IAM roles.
- All roles used by the solution follow least-privilege access. In other words, they only contain minimum permissions required so that the service can function properly.
- All data storage including Amazon S3 buckets have encryption at rest.
- External GitHub requests are authenticated using webhook secret token and source IP.

Reliability

This section describes how we architected this solution using the principles and best practices of the [reliability pillar](#).

- Using AWS Serverless Services wherever possible (for example, Lambda, Kinesis Data Firehose, API Gateway, Amazon S3, and Athena) to ensure high availability and recovery from service failure.
- Automated tests are performed on the solution to detect and fix errors quickly.
- Using AWS Lambda functions for data processing. Data is stored in Amazon S3 that persists in multiple Availability Zones (AZs) by default.

Performance efficiency

This section describes how we architected this solution using the principles and best practices of the [performance efficiency pillar](#).

- The solution uses serverless architecture.
- The solution optimizes database performance by partitioning and compressing data, using Parquet columnar format, and building Athena views and limiting query lookback time period to reduce amount of data scan.
- The solution is automatically tested and deployed every day. Our solution architects and subject matter experts review the solution for areas to experiment and improve.

Cost optimization

This section describes how we architected this solution using the principles and best practices of the [cost optimization pillar](#).

- The solution uses serverless architecture, and customers pay only for what they use.
- The compute layer defaults to Lambda, which uses a pay-per-use model.
- Athena database and queries are optimized to reduce amount of data scan hence reducing cost.

Sustainability

This section describes how we architected this solution using the principles and best practices of the [sustainability pillar](#).

- The solution uses managed and serverless services to minimize the environmental impact of the backend services.
- The solution's serverless design is aimed at reducing carbon footprint compared to the footprint of continually operating on-premises servers.

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.

Amazon EventBridge events rule

This solution creates one [Amazon EventBridge](#) events rule for each data source as follows:

- **CodeCommit events rule** - This rule is invoked by [AWS CodeCommit](#) events that match a predefined event pattern for code pushes. It routes the events to a target Amazon Kinesis Data Firehose delivery stream for processing.
- **CodeDeploy events rule** - This rule is invoked by [AWS CodeDeploy](#) events that match a predefined event pattern for code deployment state changes. It routes the events to a target Amazon Kinesis Data Firehose delivery stream for processing.
- **CodePipeline events rule** - This rule is invoked by [AWS CodePipeline](#) events that match a predefined event pattern for changes in CodePipeline action run states. This rule routes the events to a target Kinesis Data Firehose delivery stream for processing.
- **Canary events rule** - This rule is invoked by [Amazon CloudWatch Alarm](#) events that match a predefined event pattern for an alarm linked to an [Amazon CloudWatch Synthetics Canary](#), which monitors your endpoints and APIs. It routes the events to a target Amazon Kinesis Data Firehose delivery stream for processing.
- **Athena partitions events rule** - This rule runs on a daily schedule to invoke an AWS Lambda function to add a new daily partition to an Amazon Athena table.
- **CodePipeline alarm events rule** - This rule is invoked by [Amazon CloudWatch Alarm](#) events that match a predefined event pattern for an alarm monitoring the state (FAILED or SUCCEEDED) of an [AWS](#) CodePipeline. It routes the events to a target Kinesis Data Firehose delivery stream for processing.

Amazon Kinesis Data Firehose

This solution creates three [Amazon Kinesis Data Firehose](#) delivery streams to process raw data from data sources . The Data Firehose delivery streams call an AWS Lambda function to transform source records before delivering it to an Amazon S3 bucket.

The output records from Kinesis Data Firehose delivery stream are converted into parquet format for performance optimization and cost reduction. Server-side encryption for source records is turned on to protect data in transit and Amazon S3 encryption is turned on to protect data in destination.

AWS Lambda

This solution creates the following [AWS Lambda](#) functions:

- **Event parser** - These functions perform Lambda transformation within Amazon Kinesis Data Firehose. They parse raw data from data sources, extract relevant data, and return it back to Firehose delivery stream for downstream operation.
- **Query runner** - This function runs Amazon Athena queries to add Athena partitions and create views at solution deployment.
- **Athena Partition** - This function runs on a daily schedule to add a new daily partition to the Amazon Athena table.
- **QuickSight custom resource** - This function creates Amazon QuickSight resources at solution deployment.
- **Solution helper custom resource** - This function generates UUID for each solution deployment.

Amazon S3

This solution creates the Amazon S3 `aws-devops-metrics-<random-ID>` metrics bucket to store metrics output from Amazon Kinesis Firehose delivery stream. The data is stored in a partitioned folder structure `s3://aws-devops-metrics-<random-ID>/DevopsEvents/created_at=yyyy-mm-dd/` and `s3://aws-devops-metrics-<random-ID>/CodeBuildEvents/created_at=yyyy-mm-dd/`) where `created_at` is the partition key. This solution also creates an S3 `aws-devops-metrics-logging-<random-ID>` logging bucket to store access logs for the metrics bucket.

AWS Glue and Amazon Athena

This solution creates an [AWS Glue](#) and [Amazon Athena](#) database, which consists of two primary tables as the entry point to data in the [Amazon S3 metrics bucket](#) and a few views with each

containing a subset of the data from the primary table. There is one view for each metric. This solution also creates a custom Athena workgroup for all query runs and cost management. For more information, refer to [Database schema information](#).

Amazon QuickSight

This solution uses Amazon QuickSight for data visualization. You must create an Amazon QuickSight enterprise admin user if you don't already have one. To create a user, refer to [Managing users in Amazon QuickSight enterprise edition](#) in the *Amazon QuickSight User Guide*.

This solution deploys all required Amazon QuickSight resources, such as data source, datasets, analysis, and dashboards into your account. If you don't provide an Amazon QuickSight enterprise admin user, this solution will not deploy Amazon QuickSight resources in your account. For more information, refer to [Amazon QuickSight dashboards visuals](#).

CloudWatch Synthetics canary and CloudWatch alarm

This solution uses Amazon CloudWatch Synthetics canary and CloudWatch alarm to collect data needed for calculating mean time to recovery (MTTR) metrics. Synthetics canaries are configurable scripts that run on a schedule to monitor your endpoints and APIs. The CloudWatch alarm is invoked when a canary job state changes (FAILED or SUCCEEDED).

When the canary job recovers to its success state from a previously failed state, an [Amazon EventBridge](#) events rule is invoked, which in turn routes events to an Amazon Kinesis Data Firehose delivery stream for downstream processing. You can create your own canary and alarm or use the [canary-alarm.template](#) included in this solution. For more information, refer to [Set up Amazon CloudWatch Synthetics canary and Amazon CloudWatch alarm](#).

Amazon CloudWatch alarm for AWS CodePipeline

This solution uses Amazon CloudWatch Alarm to collect data needed for calculating MTTR metrics for AWS CodePipeline. The CloudWatch alarm is invoked when a pipeline run state changes (FAILED or SUCCEEDED). When the pipeline recovers to its SUCCEEDED state from a previously FAILED state, an [Amazon EventBridge](#) events rule is invoked, which in turn routes events to a Kinesis Data Firehose delivery stream for downstream processing. For more information about creating the alarm, refer to [Set up Amazon CloudWatch alarm for AWS CodePipeline](#).

Amazon API Gateway

This solution uses an API endpoint to post GitHub push events from a GitHub webhook. It is integrated with an Amazon Kinesis Data Firehose delivery stream to process the events.

AWS services in this solution

AWS service	Description
Amazon EventBridge	Core. Creates one EventBridge events rule for each data source to capture events.
Amazon Data Firehose	Core. Delivers data to Amazon S3 buckets.
AWS Lambda	Core. Deploys multiple Lambda functions to transform raw data, run queries and create QuickSight dashboard.
Amazon S3	Core. Object storage service offering industry-leading scalability, data availability, security, and performance.
Amazon CloudWatch	Core. Stores transformed data.
AWS Glue	Core. Creates catalog for Athena database.
Amazon Athena	Core. Creates Athena views, queries and work groups to support QuickSight dashboard visualization.
Amazon QuickSight	Core. Creates dashboard to visualized data.
Amazon API Gateway	Optional. Creates an API endpoint that receives GitHub events.
AWS Secrets Manager	Optional. Creates a secret token that is used to authenticate GitHub requests.

AWS service	Description
AWS Identity and Access Management	Supporting. Creates IAM roles and permissions used by AWS services to perform operations.
AWS Systems Manager	Supporting. Provides application-level resource monitoring and visualization of resource operations and cost data.

Plan your deployment

This section describes the [cost](#), [Security](#), [Regions](#), and [quota considerations](#) for planning your deployment.

Cost

You are responsible for the cost of the AWS services used while running this solution. The total cost for running this solution depends on the amount of data ingested, stored, and processed, the amount of data scanned by Amazon Athena queries, and the number of Amazon QuickSight readers and authors, along with their access time to dashboards. As of this revision, the cost for running this solution with the default settings in the US East (N. Virginia) Region (excludes free tier) with QuickSight is approximately **\$44.25 a month** and **\$50.65 a month** with GitHub and Secrets Manager. These costs are for the resources shown in the following tables.

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

Example 1: Turn on QuickSight without GitHub

AWS service	Dimensions/Month	Cost/Month [USD]
Amazon Athena	100 queries, 10 GB data scanned per query	~\$5.00
Amazon Data Firehose	100 GB	~\$2.90
Amazon Simple Storage Service (Amazon S3)	100 GB	~\$2.30
Amazon QuickSight	1 author, 10 readers, access 2 times per month for each reader	~\$24.00
Amazon CloudWatch Metric Streams	3,000,000 metric updates. \$0.003 per 1,000 metric updates	~\$9.00

AWS service	Dimensions/Month	Cost/Month [USD]
AWS Lambda	128 MB: 12 functions, total of 1M invocations and average 500 millisecond duration per Lambda run	~\$1.05
Total:		~\$44.25

Example 2: Turn on QuickSight, GitHub with Secrets Manager

AWS service	Dimensions/Month	Cost/Month [USD]
Amazon Athena	100 queries, 10 GB data scanned per query	~\$5.00
Amazon Data Firehose	100 GB	~\$2.90
Amazon Simple Storage Service (Amazon S3)	100 GB	~\$2.30
Amazon QuickSight	1 author, 10 readers, access 2 times per month for each reader	~\$24.00
Amazon CloudWatch Metric Streams	3,000,000 metric updates. \$0.003 per 1,000 metric updates	~\$9.00
AWS Lambda	128 MB: 12 functions, total of 1M invocations and average 500 millisecond duration per Lambda run	~\$1.05
Amazon API Gateway	1 million requests	~\$1.00
AWS Secrets Manager	1 secret, 1 million API calls	~\$5.40

AWS service	Dimensions/Month	Cost/Month [USD]
Total:		~\$50.65

Note

This solution implements data partition and parquet data storage for performance optimization and cost reduction. When running your own queries, we recommend that you use the `created_at(timestamp)` partition key. For more information, refer to [Performance tuning in Athena](#) in the *Amazon Athena User Guide*.

Security

When you build systems on AWS infrastructure, security responsibilities are shared between you and AWS. This [shared responsibility 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, visit [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 Regional resources.

Amazon S3

All Amazon S3 buckets are encrypted with [SSE-S3 managed encryption](#). None of the Amazon S3 buckets are available publicly. The Amazon S3 buckets are configured with the retention policy set to **Retain**.

AWS CI/CD pipeline deployment

This solution must be launched in the same Region and account where your AWS CI/CD pipeline is deployed. Refer to [Set Up a CI/CD Pipeline on AWS](#) if you do not currently have a pipeline set up on AWS.

Amazon QuickSight deployment

This solution requires Amazon QuickSight resources to be deployed in an Amazon QuickSight Enterprise edition account in the same Region. If you plan to use the Amazon QuickSight dashboard feature, you must subscribe to Amazon QuickSight Enterprise edition in the account where you deploy the solution. Refer to [Signing Up for An Amazon QuickSight Subscription](#) if you do not have an Amazon QuickSight Enterprise account set up. Ensure that you have the QuickSight Principal ARN, as you will need it later when you deploy the solution. For information, refer to [Retrieve the Amazon QuickSight Principal ARN](#).

Amazon CloudWatch alarm for Amazon CloudWatch Synthetics canary deployment

A REST application can be monitored with an Amazon CloudWatch Synthetics canary job. The solution provides an additional [canary-alarm.template](#) separate from the main CloudFormation template to provision a CloudWatch alarm and other resources to collect data required for calculating MTTR metric for applications. For more information, refer to [Set up Amazon CloudWatch Synthetics canary and Amazon CloudWatch alarm](#).

Amazon CloudWatch alarm for AWS CodePipeline deployment

An Amazon CloudWatch alarm is used to monitor the state (FAILED or SUCCEEDED) of an AWS CodePipeline. The solution provides an additional [pipeline-alarm.template](#) separate from the main CloudFormation template to provision a CloudWatch alarm and other resources to collect data required for calculating MTTR metrics for pipelines. For more information, refer to [Set up Amazon CloudWatch Alarm for AWS CodePipeline](#).

Multi-account multi-Region deployment

Data can be sent from multiple AWS accounts and Regions to the monitoring account. The solution provides an additional [sharing-account-stack.template](#) separate from the main CloudFormation template to provision [Amazon EventBridge](#) events rules and other resources required to collect data from sharing accounts. For more information, refer to the [Set up multi-account multi-Region data ingestion](#) section.

Supported AWS Regions

Depending on the template input parameters values you define, this solution requires different resources. [These resources](#) might not be available in all AWS Regions. Therefore, you must launch this solution in an AWS Region where these services are available. For the most current availability of AWS services by Region, see the [AWS Regional Services List](#).

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, see [AWS CloudFormation quotas](#) in the *AWS CloudFormation User's Guide*.

Deploy the solution

This solution uses [AWS CloudFormation templates and stacks](#) to automate its deployment. The CloudFormation templates specify 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

This solution uses AWS CloudFormation to automate the deployment of the DevOps Monitoring Dashboard on AWS solution in the AWS Cloud. It includes the following CloudFormation templates, which you can download before deployment.

[View template](#)

aws-devops-monitoring-dashboard.template: Use this template to launch the solution. The default configuration deploys Amazon EventBridge events rules, AWS Lambda functions, Amazon Simple Storage Service (Amazon S3) buckets, Amazon Data Firehose, AWS Glue and Amazon Athena databases, and Amazon QuickSight resources (optional) and all associated resources. You can also customize the template to meet your specific needs. Launch this template in the monitoring account.

[View template](#)

sharing-account-stack.template: Use this template to provision Amazon EventBridge events rules and all associated resources required to collect data from sharing accounts. Launch this template in a sharing account, when you are using the multiple-account feature.

[View template](#)

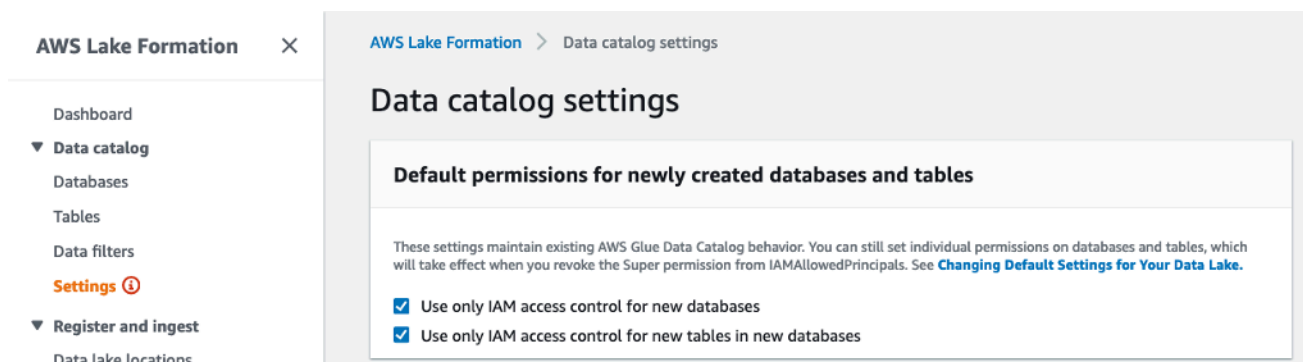
pipeline-alarm.template: Use this supplemental template to provision the CloudWatch alarm for CodePipeline and all associated resources to collect data required for calculating MTTR metrics for pipelines.

[View template](#)

canary-alarm.template: Use this supplemental template to provision the CloudWatch alarm for Synthetics canary and all associated resources to collect data required for calculating MTTR metrics for canaries.

Prerequisites

1. You must have AWS CI/CD pipeline installed in your account. This consists of AWS CodeCommit, AWS CodeBuild, AWS CodeDeploy, and AWS CodePipeline. Refer to [Set Up a CI/CD Pipeline on AWS](#) if you do not currently have a pipeline set up on AWS.
2. If you plan to use the Amazon QuickSight dashboard feature, you must subscribe to Amazon QuickSight Enterprise edition in the account where you deploy the solution. Refer to [Signing Up for An Amazon QuickSight Subscription](#) if you do not have a QuickSight Enterprise account set up. Ensure that you have the QuickSight Principal ARN, as you will need it later when deploy the solution. For more information, refer to [Retrieve the Amazon QuickSight Principal ARN](#). Also, ensure that your QuickSight account has permission to access Amazon Athena. You can choose to skip the Amazon S3 bucket configuration when you set up the Amazon Athena permission.
3. If your AWS account has [AWS Lake Formation](#) configured and your Glue legacy roles were changed, you may encounter a Glue permission issue when deploying the solution. To avoid this issue, you must turn on **Use only IAM access control for new databases** and **Use only IAM access control for new tables in databases** in Lake Formation data catalog settings. For more information, refer to [Change the default permission model in Lake Formation](#). Skip this step if you do not use Lake Formation or did not change Glue legacy roles for Lake Formation.



Lake Formation - Data catalog settings

Deployment process overview

Follow the step-by-step instructions in this section to configure and deploy the solution into your account.

Before you launch the solution, review the [cost,architecture](#), [security network](#), and other considerations discussed earlier in this guide.

Time to deploy: Approximately 10 minutes

[Step 1: Retrieve the Amazon QuickSight Principal ARN](#)

- Launch the AWS CloudFormation template into your AWS account.
- Review the template parameters and enter or adjust the default values as needed.

[Step 2: Launch the stack](#)

- Launch the AWS CloudFormation template into your AWS account.
- Review the template parameters and enter or adjust the default values as needed.

[Step 3: Configure Amazon QuickSight](#)

- After the stack is successfully deployed, set up Amazon QuickSight for data visualization.

[Step 4: Set up GitHub webhook](#)

- Create your webhook to subscribe to GitHub events.

Important

This solution includes an option to send anonymized data 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 Policy](#).

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 the [Anonymized data collection](#) section of this guide.

Step 1: Retrieve the Amazon QuickSight Principal ARN

If you want to deploy Amazon QuickSight resources, you must retrieve the Amazon QuickSight Principal ARN before deploying this solution. To retrieve the Amazon QuickSight User Principal ARN, you must have access to a shell or terminal with the AWS CLI installed. For installation instructions, refer to [What Is the AWS Command Line Interface](#) in the *AWS CLI User Guide*. Optionally, you can use the [AWS CloudShell](#) service to run AWS CLI commands.

Running the following `list-users` command returns the list of users with their corresponding QuickSight User ARNs.

```
aws quicksight list-users --region <aws-region> --aws-account-id <account-id> --  
namespace <namespace-name>
```

The following example shows a valid ARN:

```
arn:aws:quicksight:<aws-region>:<account-id>:user/<namespace-name>/<quicksight-user-  
name>
```

The default namespace-name is `default`. For example, `arn:aws:quicksight:us-east-1:111111111111:user/default/myquicksightuser`. Choose a user who has permissions to create Amazon QuickSight resources in that account and AWS Region, such as a QuickSight admin user.

If you do not have an Amazon QuickSight Enterprise account, refer to [Signing up for an Amazon QuickSight subscription](#) to set up your account and then retrieve the Principal account ARN.

Step 2: Launch the stack

Follow the step-by-step instructions in this section to configure and deploy the solution into your account. Before you launch the stack, you must complete the [prerequisites](#).

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 you used in this solution.

1. Sign in to the [AWS Management Console](#) and select the button to launch the `aws-devops-monitoring-dashboard` AWS CloudFormation template.



A blue rectangular button with rounded corners containing the text "Launch solution" in white, bold font.

You can also [download the template](#) as a starting point for your own implementation.

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.

Note

This solution uses services such as Amazon Athena, Amazon Data Firehose and Amazon QuickSight (optional), which are not currently available in all AWS Regions. You must launch this solution in an AWS Region where these services are available. For the most current availability by Region, refer to the [AWS Regional Services List](#).

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 quotas](#) in the *AWS Identity and Access Management User Guide*.
5. Under **Parameters**, review the parameters for this solution template and modify them as necessary. This solution uses the following default values:

Parameter	Default	Description
Metrics Configuration		

Parameter	Default	Description
Athena Query Data Duration (Days)	90	Enter a lookback duration (days) that Athena query will use to retrieve data. By default, Athena query retrieves data within the last 90 days. We recommend that you limit the duration for performance optimization and cost reduction.
AWS CodeCommit Repository List	ALL	<p>List of the names of AWS CodeCommit repositories that will be monitored. Must be single-quoted and comma separated. For example: 'MyRepository1' , 'MyRepository2'</p> <p>To monitor all the repositories, leave default ALL value.</p>
S3 Configuration		
S3 Transition Days	365	Enter the number of days after which you would like to transition Amazon S3 objects to Amazon S3 Glacier storage class. By default objects are transitioned to Amazon S3 Glacier 365 days (one year) after creation.
QuickSight Configuration		
Amazon QuickSight Principal ARN	<Optional Input>	Provide an Amazon QuickSight admin user ARN to automatically create QuickSight resources. Amazon QuickSight Enterprise edition must be activated for the account. For example: <code>arn:aws:quicksight:AWSRegion:AWSAccountId:user/default/QuickSightUserName</code> . To deactivate QuickSight dashboards creation, do not enter a value. For more information, refer to Prerequisites Step 2 .
GitHub Configuration		

Parameter	Default	Description
Use GitHub Repository	No	Select Yes if GitHub is used, otherwise leave it as No.
Webhook Secret Token	<Optional Input>	Enter a random string with high entropy to authenticate access to webhooks in GitHub. If a webhook request header contains a matching secret, IP address authentication is bypassed. The string cannot contain commas (,) backward slashes (\), or quotes ("). We recommend using a secret token to secure your GitHub webhook. To turn off secret authentication, leave it blank. If you enter a secret, you must enter the same secret in your GitHub webhook configuration to avoid failure. For more information, refer to Setting up a webhook . Ignore this field if you are not using GitHub.
Allowed IP Addresses	192.30.252.0/22 , 185.199.108.0/22 , 140.82.112.0/20 , 143.55.64.0/20	Enter a comma-separated list of allowed IPV4 CIDR blocks. By default, GitHub IP ranges are used. Note that GitHub changes their IP addresses from time to time so we recommend regular monitoring of their API. If API secret is used, IP address authentication is bypassed. Ignore this field if you are not using GitHub.
Multi-Account Configuration		
Principal Type	None	To turn on the multi-account feature, select AWS Account Number or AWS Organization ID as the principal type of the sharing accounts that data comes from. Leave it as None to turn off the multi-account feature.

Parameter	Default	Description
List of AWS Accounts or AWS Organization IDs	<Optional Input>	If you selected List of AWS Accounts, enter a comma-separated list of AWS account numbers, for example, 111111111111 ,222222222222 . If you selected List of AWS Organization IDs, enter a comma-separated AWS Organization IDs, for example, o-xxxxxxxxxx ,o-yyyyyyyyyy . Refer to Viewing the details of an organization from the management account for instructions about how to find the Organization Id . Leave it blank if you don't use the multi-account feature.

Tag Configuration

Tag Configuration for filtering on CodeCommit Repositories	<Optional Input>	Enter a semicolon-separated list of tags, using a comma as a separator between the tag key and value, for example, env , prod ; anotherKey , anotherValue . Omitting a value will result in a filter that captures all values for that tag. This tag is used in an Athena query to find resources with the matching tag, and is used as a data filter in QuickSight dashboard.
Tag Configuration for filtering on CodeBuild Projects	<Optional Input>	Enter a semicolon-separated list of tags, using a comma as a separator between the tag key and value, for example, env , prod ; anotherKey , anotherValue . Omitting a value will result in a filter that captures all values for that tag. This tag is used in an Athena query to find resources with the matching tag, and is used as a data filter in QuickSight dashboard.

Parameter	Default	Description
Tag Configuration for filtering on CodePipeline Projects	<Optional Input>	Enter a semicolon-separated list of tags, using a comma as a separator between the tag key and value, for example, env,prod;anotherKey,anotherValue . Omitting a value will result in a filter that captures all values for that tag. This tag is used in Athena query to find resources with the matching tag and is used as data filter in QuickSight dashboard.

- Choose **Next**.
- On the **Configure stack options** page, choose **Next**.
- 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.
- 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 10 minutes.

Note

If you provided an Amazon QuickSight Principal ARN, this solution launches a nested stack to create QuickSight resources into the account you provided. If you selected Yes to GitHub repository, this solution launches a nested stack to create an Amazon API Gateway and other AWS resources required for GitHub integration into the account you provided.

Step 3: Configure Amazon QuickSight

This solution uses Amazon QuickSight for data visualization. Follow these instructions to configure permissions, and view datasets, analysis, and dashboards in Amazon QuickSight.

Note

You can also set up your own visualization tools, such as Tableau. For more information, refer to [Build visualizations with Amazon Athena and Tableau](#).

1. After the stack successfully deploys, go to the **Outputs** tab of the stack and make a note of the values for **QSAalysisURL**, **QSDashboardURL**, and **DevOpsMetricsS3Bucket**.
2. Sign in to the AWS Management Console and navigate to Amazon QuickSight.
3. Change the Region in the URL to match the Region where you deployed the solution. For example, if the solution was deployed in the us-east-1 Region, the QuickSight URL will mirror the following path: `https://us-east-1.quicksight.aws.amazon.com/sn`.
4. Select your username on the upper right corner, then choose **Manage QuickSight**.
5. From the left navigation menu, select **Security & permissions**.
6. Under **QuickSight access to AWS Services**, choose **Add or remove**.
7. Select **IAM**, **Amazon S3**, and **Amazon Athena**. If these options are already selected, uncheck and recheck the options.
8. Choose **Amazon S3**, choose the **Details** link.
9. Choose **Select S3 buckets**.
10. Select the bucket name for **DevOpsMetricsS3Bucket**, and check the check box under **Write permission for Athena Workgroup** for the bucket.
11. Select **Finish**, then choose **Update**.
12. From the **Output** tab of the stack, select **QSAalysisURL** and **QSDashboardURL** to open dashboards and analyses. You can also navigate to them in the Amazon QuickSight console. This solution creates one analysis, one dashboard, and multiple datasets. This solution creates Amazon QuickSight resources that are prefixed with the stack name. For example, `<stack-name>-analysis`. Refer to [Amazon QuickSight dashboards visuals](#) for sample dashboards visuals. You might receive a **No Data** message if the Amazon S3 metrics bucket is empty after launching this solution. You can refresh the pages to view data and visuals after this solution finishes processing data and sends metrics to Amazon S3.

Note

This solution creates Amazon QuickSight datasets that use Direct Query to query Amazon S3 buckets for data. For better performance, use SPICE. Refer to [Using SPICE Data in an Analysis](#) in the *Amazon QuickSight User Guide* for more information about configuring and using SPICE.

Step 4: Set up GitHub webhook

This solution integrates with GitHub repository via webhooks, which subscribe to certain events from GitHub. When one of those events is invoked, GitHub sends an HTTP POST payload to the webhook's configured URL. Currently the solution collects push events to calculate GitHub activity metrics such as pushes and commits by authors and repositories.

To create a webhook, go to [Setting up a webhook](#). Use the following configurations for your webhook, and leave the rest of the configurations to default values.

- **Payload URL:** Enter the API endpoint created by the solution. The endpoint can be found on the **Outputs** tab of the solution's main CloudFormation stack that you deployed in your AWS account. It should look like `https://api-id.execute-api.us-east-1.amazonaws.com/prod/git`.

Stack info	Events	Resources	Outputs	Parameters	Template	Change sets
Outputs (5)						
<input type="text" value="Search outputs"/>						
Key	Value	Description				
APIEndpoint	https://.execute-api.us-east-1.amazonaws.com/prod/git	Amazon API Endpoint to receive GitHub events for AWS DevOps Monitoring Dashboard Solution				

API endpoint to receive GitHub events

- **Content type:** `application/json`
- **Secret:** If you entered a secret at CloudFormation stack deployment, enter the same secret here. Otherwise leave it blank. Whenever you change the secret in your stack, you must make the same change in webhook. Mismatching secrets between the solution and webhook will lead to request authentication failure.

- **Event types:** Just the push event.

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 description states: 'Service Catalog application to track and manage all your resources for the solution'. 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 note to monitor application health with Amazon CloudWatch. The 'Cost' section includes a 'View all' button and a note to view resource costs per application using AWS Cost Explorer. A 'Cost (USD)' section is partially visible below.

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 monitoring dashboard. The navigation bar includes tabs for Overview, Resources, Provisioning, Compliance, Monitoring (selected), OpsItems, Logs, Runbooks, and Cost. The main content area is titled "Application Insights (0) Info" and includes a toggle for "View Ignored Problems", an "Actions" dropdown, and an "Add an application" button. Below this is a search bar with the placeholder "Find problems", a "Last 7 days" filter, a refresh button, and pagination controls. A table header is visible with columns: Problem su..., Status, Severity, Source, Start time, and Insights. A message states "Advanced monitoring is not enabled" and provides instructions on how to onboard an application. A prominent button labeled "Auto-configure Application Insights" is displayed at the bottom of the message box.

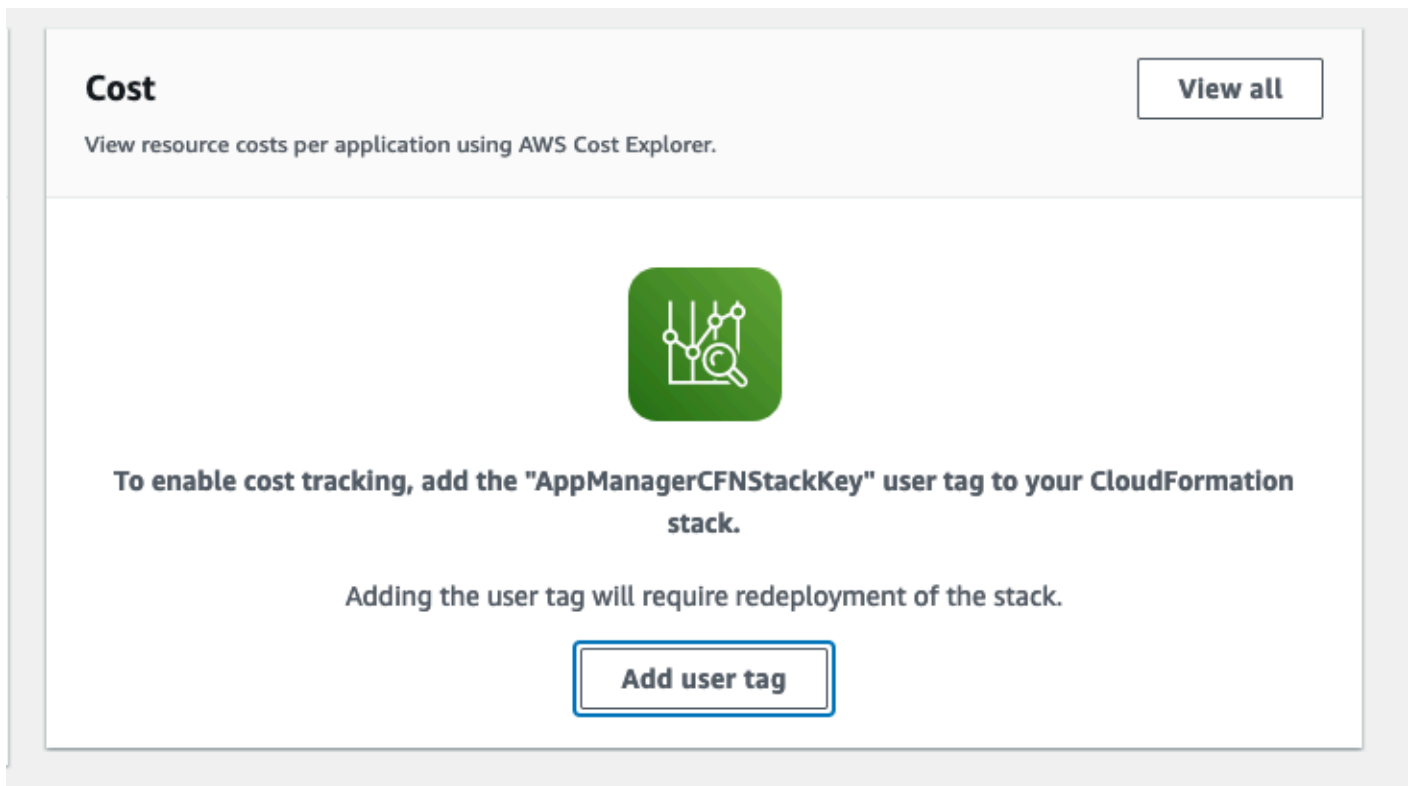
Monitoring for your applications is now activated and the following status box appears:

This screenshot shows the same AWS Application Insights monitoring dashboard as the previous one, but with a success message. The navigation bar and top controls remain the same. The message box, which was previously empty, now contains a green checkmark icon and the text: "Application monitoring has been successfully enabled. It will take some time to display any results. Please use the refresh button to view results." The "Auto-configure Application Insights" button is no longer visible.

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 `aws-devops-monitoring-dashboard` CloudFormation stack to get the latest version of the solution's framework.

1. Sign in to the AWS CloudFormation Console, select your existing DevOps Monitoring Dashboard on AWS 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 for the stack.
 - 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 AWS Identity and Access Management (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 10 minutes depending on the options chosen.

Uninstall the solution

You can uninstall the DevOps Monitoring Dashboard on AWS solution from the AWS Management Console or by using the AWS Command Line Interface (AWS CLI). You must manually delete the Amazon Simple Storage Service (Amazon S3) buckets created by this solution. To protect customer data, AWS Solutions Implementations do not automatically delete these resources in case you need to retain stored data.

Note

The Amazon S3 buckets are configured with the retention policy set to **Retain**. You must manually delete them.

Using the AWS Management Console

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

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*. Optionally, you can use the [AWS CloudShell](#) service to run AWS CLI commands. After confirming that the AWS CLI is available, run the following command.

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

Note

The Amazon S3 buckets are configured with the retention policy set to **Retain**. You must manually delete them.

Troubleshooting

If you need help with this solution, contact AWS Support to open a support case for this solution.

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**.

Use the solution

This section provides detailed instructions to use the solution after you deploy the solution.

Set up Amazon CloudWatch synthetics canary and Amazon CloudWatch alarm

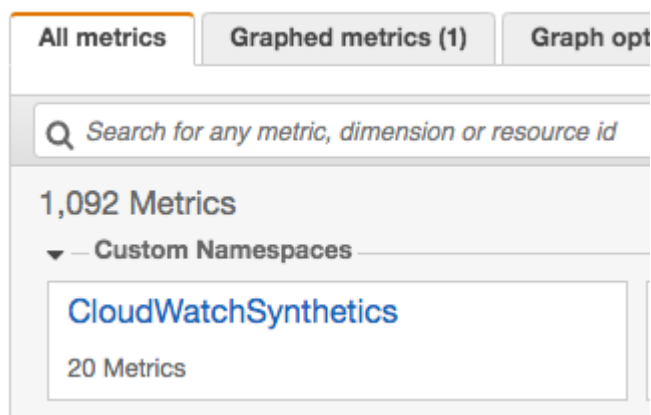
You can use the solution to monitor the MTTR metrics of your REST API. To do so, you can choose one of the following two ways to set up [Amazon CloudWatch Synthetics canary](#) and its [Amazon CloudWatch Alarm](#). The goal is to create an Amazon CloudWatch alarm that monitors the state (success or failure) of a new or existing canary job. Whenever the canary job state changes, it invokes the alarm. This will generate events that are needed for calculating MTTR metrics.

1. (Recommended) Automated setup:

- This solution provides a [canary-alarm.template](#) that you can deploy to create an Amazon CloudWatch alarm and/or canary into your AWS account where you deployed the main AWS CloudFormation template.

2. Manual setup

- If you don't have a canary, sign in to the AWS Management Console and create a canary. To create a canary, refer to [Creating a canary](#) in the *Amazon CloudWatch User guide* to create one. If you have already created one, skip to the next step.
- To create an alarm, refer to [Create a CloudWatch Alarm Based on Static Threshold](#) in the *Amazon CloudWatch User Guide*. When you reach the select metrics step, make sure you select CloudWatchSynthetics metrics, your canary and SuccessPercent metric as shown below.
- Select **CloudWatchSynthetics** metrics for the alarm.



Select CloudWatchSynthetics metrics

- From the **All metrics** tab, select **By Canary** and then select your canary and **SuccessPercent** metric name.

The screenshot shows the AWS CloudWatch Synthetics interface. At the top, there are tabs for 'All metrics', 'Graphed metrics (1)', 'Graph options', and 'Source'. Below these, the breadcrumb navigation is 'All > CloudWatchSynthetics > By Canary'. A search bar contains the text 'Search for any metric, dimension or resource id' and a 'Graph search' button. The main content is a table with two columns: 'CanaryName (20)' and 'Metric Name'. The table lists several metrics for the 'mycanary' canary, with 'SuccessPercent' selected. At the bottom right, there are 'Cancel' and 'Select metric' buttons.

CanaryName (20)	Metric Name
<input type="checkbox"/> mycanary	Failed requests
<input type="checkbox"/> mycanary	Duration
<input checked="" type="checkbox"/> mycanary	SuccessPercent
<input type="checkbox"/> mycanary	2xx
<input type="checkbox"/> mycanary	Error
<input type="checkbox"/> mycanary	4xx

Select your canary and SuccessPercent metric

- Name the alarm `S00143-[my-application-name]-[my-repository-name]-MTTR`. For example, `S00143-[MyDemoApplication]-[MyDemoRepo]-MTTR`. `S00143` is the solution ID. Application name is the name of the application that your canary monitors and repository name is the name of the repository where the source code for your application resides. This solution uses alarm name to determine if an alarm is used for MTTR metrics and what application and repository are associated with the metrics.
- Under **Conditions** of the alarm, leave threshold type as **Static** and choose whenever **SuccessPercent** is **Lower** than **100** or enter a threshold value that fits your use case.
- For an example of the alarm, refer to the following figure.

Metric

Edit

Graph
This alarm will trigger when the blue line goes below the red line for 1 datapoints within 5 minutes.

Percent

101

101

100

99.5

99

23:00 00:00 01:00

● SuccessPercent

Namespace
CloudWatchSynthetics

Metric name

CanaryName

Statistic

Period

Conditions

Threshold type

Static
Use a value as a threshold

Anomaly detection
Use a band as a threshold

Whenever SuccessPercent is...
Define the alarm condition.

Greater
> threshold

Greater/Equal
>= threshold

Lower/Equal
<= threshold

Lower
< threshold

than...
Define the threshold value.

Must be a number

Alarm Example


Set up Amazon CloudWatch alarm for AWS CodePipeline

You can use the solution to monitor the MTTR metric of your AWS CodePipeline. To do so, deploy the [pipeline-alarm.template](#) into the AWS account where you deployed the main AWS CloudFormation template for the solution. This will create an Amazon CloudWatch alarm that monitors the state (FAILED or SUCCEEDED) of your pipeline. Whenever the pipeline state changes, it invokes the alarm. This will generate events that are needed for calculating MTTR metrics.

Parameter	Default	Description
CodePipeline Name	<i><requires input></i>	The name of the CodePipeline that you want to monitor. This name becomes part of the CloudWatch alarm name.
Create a new log group	YES	Whether or not to create a new log group. If you are deploying this stack for the first time, you should select YES. If the log group for the pipeline already exists from a previous deployment, select NO.
Log Group Name	<i><requires input></i>	Name of the log group (new or existing) to write the CloudWatch metrics into. If you have selected YES to create a new log group, you can enter an arbitrary log group name here. If you choose to use an existing log group, enter the name of the existing log group.

Parameter	Default	Description
Repository Name	<requires input>	The name of the repository which is the source of your CodePipeline. This name becomes part of the CloudWatch alarm name.

Once the stack is deployed successfully, go to the **Resources** tab of the CloudFormation stack to view the newly created alarm.

CodePipelineAlarmAppAlarm C62B9C08	SO0143-[Demo-pipeline]- [Demo-Repository]-MTTR 	AWS::CloudWatch::Alarm
---------------------------------------	---	------------------------

CloudWatch alarm

Set up multi-account multi-Region data ingestion

You can use this solution to monitor DevOps metrics from multiple AWS accounts and Regions in a monitoring account. To do so, follow these steps to set up resources and permissions required for streaming data multi-account multi-Region into a central S3 bucket in the monitoring account.

1. [Deploy the solution's main AWS CloudFormation template into a monitoring account](#), where the data from different accounts are gathered and analyzed. You must enter the AWS account numbers or organization IDs of other accounts in the main template.
2. Deploy the [sharing-account-stack.template](#) into other accounts where the data is generated.

Parameter	Default	Description
Monitoring Account Configuration		
ARN of the custom event bus in the monitoring account	<requires input>	ARN of the custom Amazon EventBridge event bus in the monitoring account where the events are sent. To find the ARN, sign in to the AWS

Parameter	Default	Description
		<p>CloudFormation console in the monitoring account, select the solution's main CloudFormation stack you deployed, open the Outputs tab, then copy the value for CustomEventBusArn, for example, <code>arn:aws:events:Region:Account:event-bus/EventBusName</code></p>
<p>ARN of the DevOps metrics S3 bucket in the monitoring account</p>	<p><i><requires input></i></p>	<p>Enter the ARN of the S3 bucket in the monitoring account where DevOps metrics are stored. To find the ARN, sign in to the AWS CloudFormation console in the monitoring account, select the solution's main CloudFormation stack you deployed, open the Outputs tab, then copy the value for DevOpsMetricsS3Bucket, for example, <code>arn:aws:s3:::aws-devops-metrics-xxxxxx</code>.</p>
<p>AWS account number of the monitoring account</p>	<p><i><requires input></i></p>	<p>Enter the AWS account number of the monitoring account where the solution's main template is deployed to receive data from other accounts.</p>

Parameter	Default	Description
AWS region of the monitoring account where the solution's main template is deployed	<i><requires input></i>	Enter the AWS region of the monitoring account where the solution's main template is deployed to receive data from other account, for example, <code>us-east-1</code> .
Tag Configuration		
Tag Configuration for filtering on CodeCommit Repositories	<i><requires input></i>	Enter a semicolon-separated list of tags, using comma as a separator between the tag key and value, for example, <code>env,prod;anotherKey,anotherValue</code> . This tag is used in an Athena query to find repositories with the matching tag and is used as data filter in the QuickSight dashboard. Omitting a tag value will result in a filter that captures all values for that tag key. Only repositories matching the combination of all tags will be captured. Leave it blank if you do not use the tag feature.

Parameter	Default	Description
Tag Configuration for filtering on CodeBuild Projects	<i><requires input></i>	Enter a semicolon-separated list of tags, using a comma as a separator between the tag key and value, for example, <code>env,prod;anotherKey,anotherValue</code> . This tag is used in an Athena query to find build projects with the matching tag and is used as a data filter in QuickSight dashboard. Omitting a tag value will result in a filter that captures all values for that tag key. Only build projects matching the combination of all tags will be captured. Leave it blank if you do not use the tag feature.

Parameter	Default	Description
Tag Configuration for filtering on CodePipeline Projects	<i><requires input></i>	Enter a semicolon-separated list of tags, using a comma as a separator between the tag key and value, for example, <code>env,prod;anotherKey,anotherValue</code> . This tag is used in an Athena query to find pipelines with the matching tag and is used as a data filter in QuickSight dashboard. Omitting a tag value will result in a filter that captures all values for that tag key. Only pipelines matching the combination of all tags will be captured. Leave it blank if you do not use the tag feature.

Running queries and work with query results and output files in Amazon Athena

You can [run SQL queries using Amazon Athena](#) to directly query the table and views created by the solution. For details about these table and views, refer to [Database schema information](#) and [Build visualizations with Amazon Athena and Tableau](#). Query results are stored as CSV files in the metrics S3 bucket under the prefix, `athena_query_output`. You can also download query result files directly from the Amazon Athena console.

Note

This solution implements data partition and parquet data storage for performance optimization and cost reduction. When running your own queries, we recommend that

you use the `created_at` (timestamp) partition key. For more information, refer to [Performance tuning in Athena](#) in the *Amazon Athena User Guide*.

Build visualizations with Amazon Athena and Tableau

You can build visualizations using Tableau and Amazon Athena for the views created by this solution. By default, the solution deploys a set of pre-built Amazon QuickSight dashboards into your account. You can also choose other BI tools, such as Tableau, to build your own visualization. For more information, refer to [Building AWS Data Lake visualizations with Amazon Athena and Tableau](#). The following database information can be used to build database connection:

- Athena database name: `aws_devops_metrics_db_so0143`
- You can build visualizations for the following views:
 - `code_change_activity_view`: This view contains data related to code pushes to AWS CodeCommit.
 - `code_deployment_detail_view`: This view contains data related to code deployments using AWS CodeDeploy.
 - `code_build_detail_view`: This view contains data related to code builds generated by AWS CodeBuild.
 - `code_pipeline_detail_view`: This view contains data related to code builds generated by AWS CodePipeline.
 - `recovery_time_detail_view`: This view contains Amazon CloudWatch Alarm data related to MTTR metrics. The **`duration_minutes`** column shows how long it takes to restore a service from its failure to success state at one time.
 - `github_change_activity_view`: This view contains data related to GitHub code change activities generated by changes made to GitHub repositories.
- Table:

Note

Do not directly build visualizations for the following tables as they contain a large amount of unfiltered data. Instead, you must build visualizations for the views that contain a subset of data filtered for specific metrics.

- `aws_devops_metrics_table`: This table is the entry point to most of data in the Amazon S3 metrics bucket (`s3://YourS3MetricsBucket/DevOpsEvents/`). It is the base table for all the views except for `code_build_detail_view`. Do not directly build visualizations for this table. You should build visualizations for the views.
- `aws_codebuild_metrics_table`: This table is the entry point to CodeBuild data in the Amazon S3 metrics bucket (`s3://YourS3MetricsBucket/CodeBuildEvents/`). It is the base table for `code_build_detail_view`. Do not directly build visualizations for this table. You should build visualizations for the view.
- `aws_github_metrics_table`: This table is the entry point to GitHub data in the Amazon [S3 metrics bucket](#) (`s3://YourS3MetricsBucket/GitHubEvents/`). It is the base table for `github_change_activity_view`.
- `tagged_codecommit_table`: This table contains the AWS CodeCommit repositories tagged by the user-specified tags. It points to JSON files in the Amazon [S3 metrics bucket](#) (`s3://YourS3MetricsBucket/TaggedResources/CodeCommit`), which is used to join `aws_devops_metrics_table` to get repositories with matching tags.
- `tagged_codepipeline_table`: This table contains the AWS CodePipeline pipelines tagged by the user specified tags. It points to JSON files in the Amazon [S3 metrics bucket](#) (`s3://YourS3MetricsBucket/TaggedResources/CodePipeline`), which is used to join `aws_devops_metrics_table` to get pipelines with matching tags.
- `tagged_codebuild_table`: This table contains the AWS CodeBuild projects tagged by the user-specified tags. It points to JSON files in the Amazon [S3 metrics bucket](#) (`s3://YourS3MetricsBucket/TaggedResources/CodeBuild`), which is used to join `aws_codebuild_metrics_table` to get build projects with matching tags.

For more information about the database schema, refer to [Database schema information](#).

DevOps metrics list

Code change volume metrics

The code change volume metrics indicate the code change frequency of developers in a source control, such as AWS CodeCommit. These metrics give DevOps leaders better visibility into the coding activities of their development teams. They can answer questions, such as who makes the most number of code changes and which repositories are the most active over time. The underlying

data for these metrics are AWS CodeCommit events. To view an example dashboard for these metrics, refer to [Code change volume dashboards](#).

Mean time to recover metrics

MTTR metrics present outage minutes and the average time it takes to restore an application from a failed state. These metrics help DevOps leaders correlate change activity to system stability, track problematic applications and identify opportunities to improve the stability of applications. Currently MTTR metrics track two types of applications: REST API and AWS CodePipeline. The underlying data for these metrics are Amazon CloudWatch alarm events for Synthetics canary and CodePipeline respectively. To view an example dashboard for these metrics, refer to [Mean time to recover dashboards](#).

Change failure rate metrics

The change failure rate metrics indicate how often deployment failures occur for an application. These metrics help DevOps leaders track the code quality of their development teams and drive improvements to reduce change failure rate over time. The underlying data for this metric are AWS CodeDeploy events. To view an example dashboard for these metrics, refer to [Change failure rate dashboards](#).

Deployment metrics

Deployment metrics present data about application deployment, such as deployment state (failure or success) and frequency. These metrics help DevOps leaders track the frequency and quality of their continuous software release to end-users. The underlying data for this metric are AWS CodeDeploy events. To view an example dashboard for these metrics, refer to [Deployment dashboards](#).

Build metrics

Build metrics present data about CodeBuild activities, such as build duration, build state (failure or success) and frequency, along with resource utilization metrics for CPU, memory, and storage utilization. These metrics help DevOps leaders track the frequency and quality of their code build process. These metrics can indicate which build projects or phases take the longest time to run, which build projects are the most active over time, and which build projects fail the most. The underlying data for these metrics are AWS CodeBuild metrics. To view an example dashboard for these metrics, refer to [Build dashboards](#).

Note

Resource utilization metrics are not available for builds shorter than one minute and they are not supported in all of the Regions where AWS CodeBuild is supported. For a complete list of the supported Regions, refer to [Monitoring CodeBuild resource utilization metrics](#) in the *AWS CodeBuild User Guide*.

Pipeline metrics

The Pipeline metrics present data about CodePipeline, such as pipeline execution state (failure, success, or other), execution duration and frequency along with state at stage and action level. These metrics give DevOps leaders better visibility into the pipeline activities of their development teams. These metrics can indicate which pipelines fail the most, which pipelines take the longest time to run, and which pipelines are the most active over time. The underlying data for these metrics are AWS CodePipeline events. To view an example dashboard for these metrics, refer to [Pipeline dashboards](#).

GitHub activity metrics

The GitHub activity metrics present code change volumes in GitHub, such as the number and frequency of pushes and commits by authors and repositories. These metrics give DevOps leaders better visibility into the coding activities of their development teams. They can answer questions, such as who makes the greatest number of pushes or commits and which repositories are the most active over time. The underlying data for these metrics are push events from GitHub. To view an example dashboard for these metrics, refer to [GitHub activity dashboards](#).

Database schema information

The following diagrams displays a high-level database schema structure for the tables and views created in AWS Glue and Amazon Athena database. The data model is not normalized and includes redundant attributes for reporting performance.

The `aws_devops_metrics_table` is a primary table for data related to [AWS Developer Tools](#) (not including AWS CodeBuild) in the Amazon [S3 metrics bucket](#). The **detail** column in the table uses a struct data type and contains data for different metrics. Several views are built based on this table. Each view contains only a subset of the base table's data for a specific metrics, such as code

change activity and code deployment. A view's data is mainly extracted from the **detail** column in the table for those metrics.

The `aws_codebuild_metrics_table` is a primary table that points to AWS CodeBuild data in the Amazon S3 metrics bucket. It is the base table for the code build view.

The `aws_github_metrics_table` is a primary table for GitHub data in the Amazon [S3 metrics bucket](#). It is the base table for the GitHub change activity view.

The `tagged_codecommit_table` contains the data for the AWS CodeCommit repositories tagged by the user-specified tags in the Amazon [S3 metrics bucket](#). It is used to join `aws_devops_metrics_table` to get the tag information for the CodeCommit change activity view.

The `tagged_codepipeline_table` contains the data for the AWS CodePipeline pipelines tagged by the user-specified tags in the Amazon [S3 metrics bucket](#). It is used to join `aws_devops_metrics_table` to get the tag information for the CodePipeline view.

The `tagged_codebuild_table` contains the data for the AWS CodeBuild projects tagged by the user-specified tags in the Amazon [S3 metrics bucket](#). It is used to join `aws_codebuild_metrics_table` to get the tag information for the CodeBuild view.



CodeCommit metrics database schema structure



CodePipeline metrics database schema structure



CodeBuild metrics database schema structure



CodeDeploy and MTTR metrics database schema structure

aws_github_metrics_table		github_change_activity_view	
repository_name	string	repository_name	string
branch_name	string	branch_name	string
author_name	string	author_name	string
event_name	string	event_name	string
commit_id	array(string)	commit_count	int
time	timestamp	time	timestamp
created_at	timestamp	created_at	timestamp

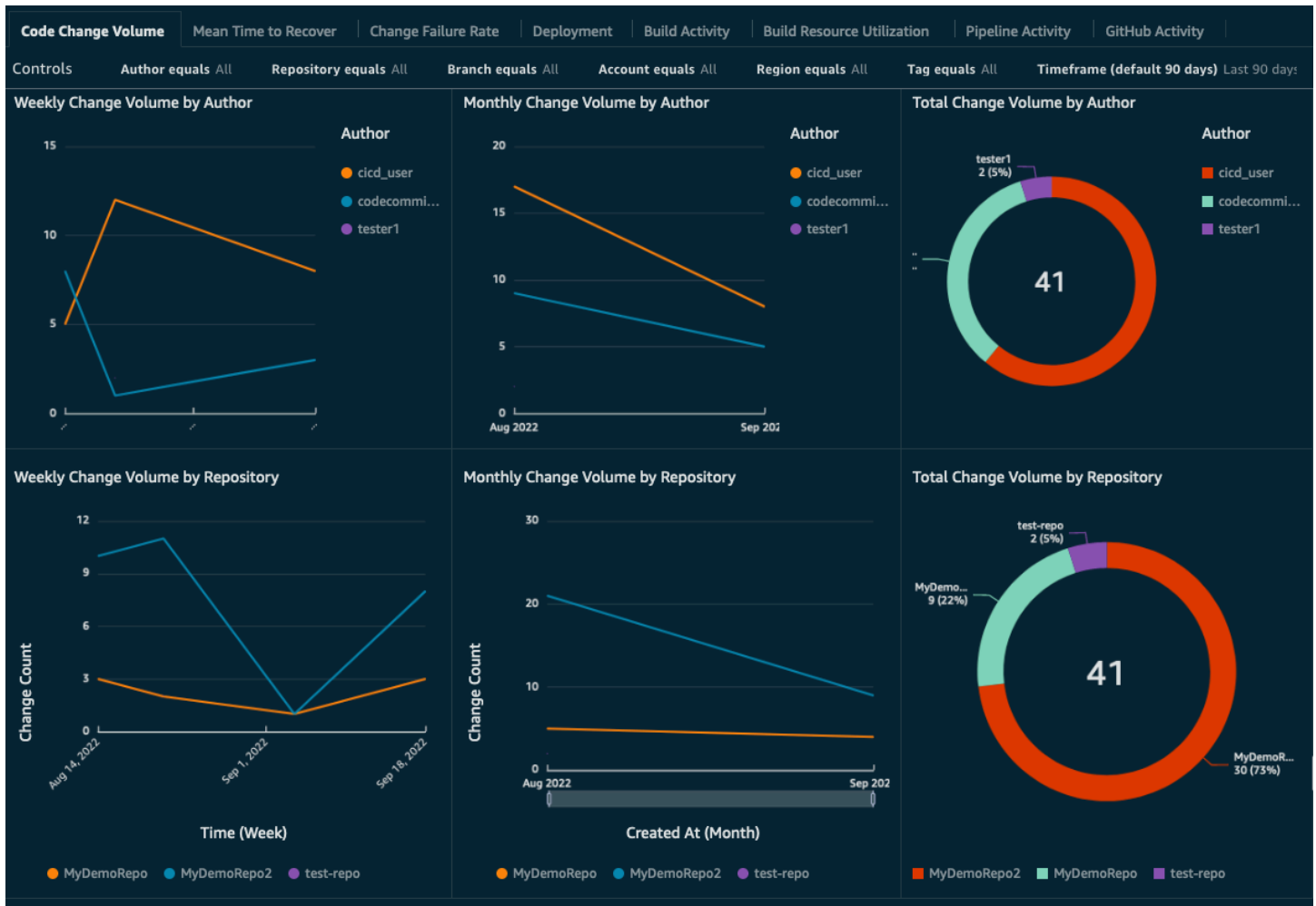
GitHub change activity metrics database schema structure

Amazon QuickSight dashboard visuals

The following dashboards are examples of dashboard visuals that this solution deploys with Amazon QuickSight. These examples use a dark midnight theme, but your dashboards may use a different theme.

Code change volume dashboard

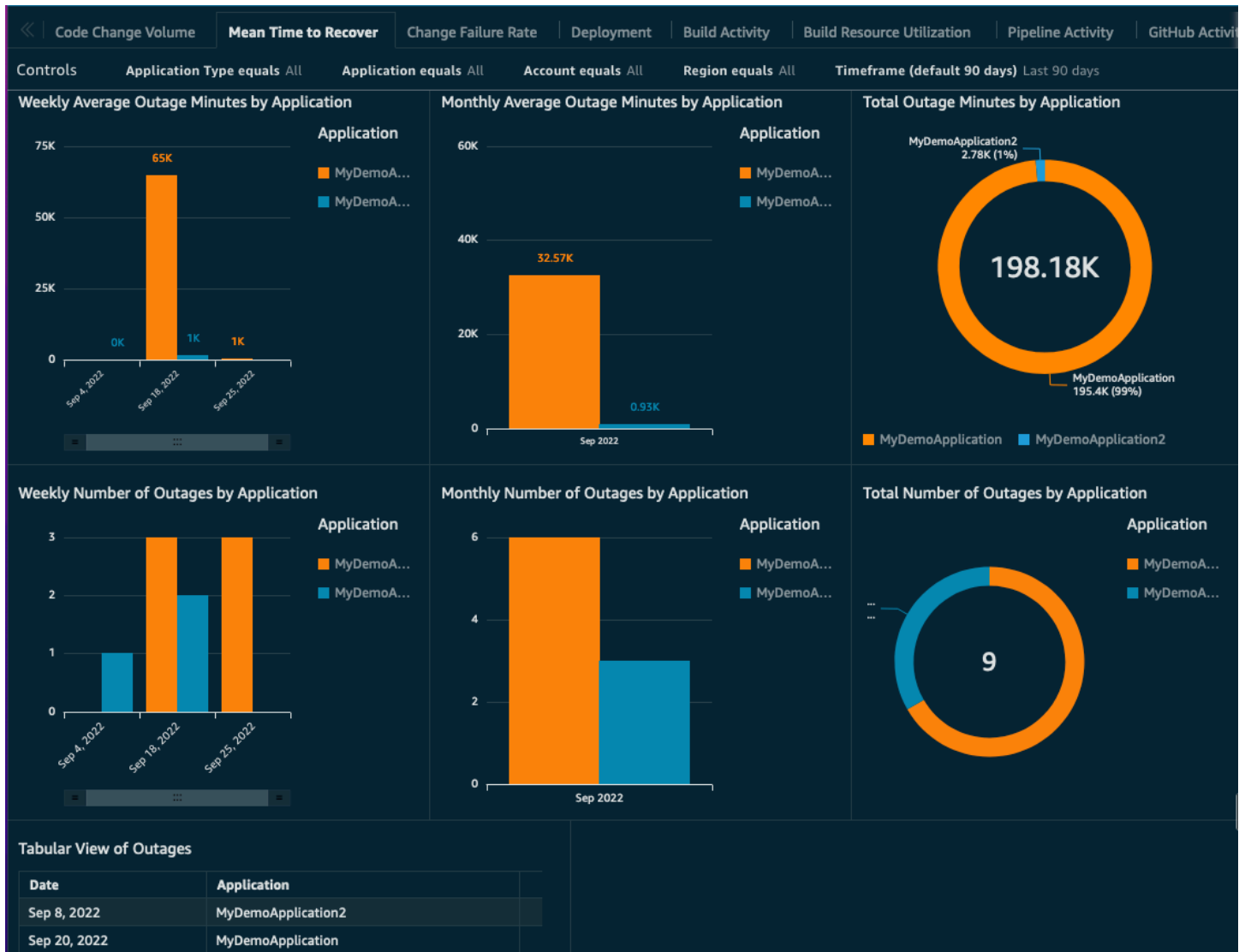
This dashboard displays the number of code changes made by author and repository. It provides a weekly, monthly and aggregated view of the metrics by author and repository. You can filter data by author, repository, branch, AWS account, AWS Region, tag, or time period (default to last 90 days) using the custom filter as needed. For more information about the metrics, refer to [Code change volume metrics](#).



Code change volume dashboard

Mean time to recover dashboard

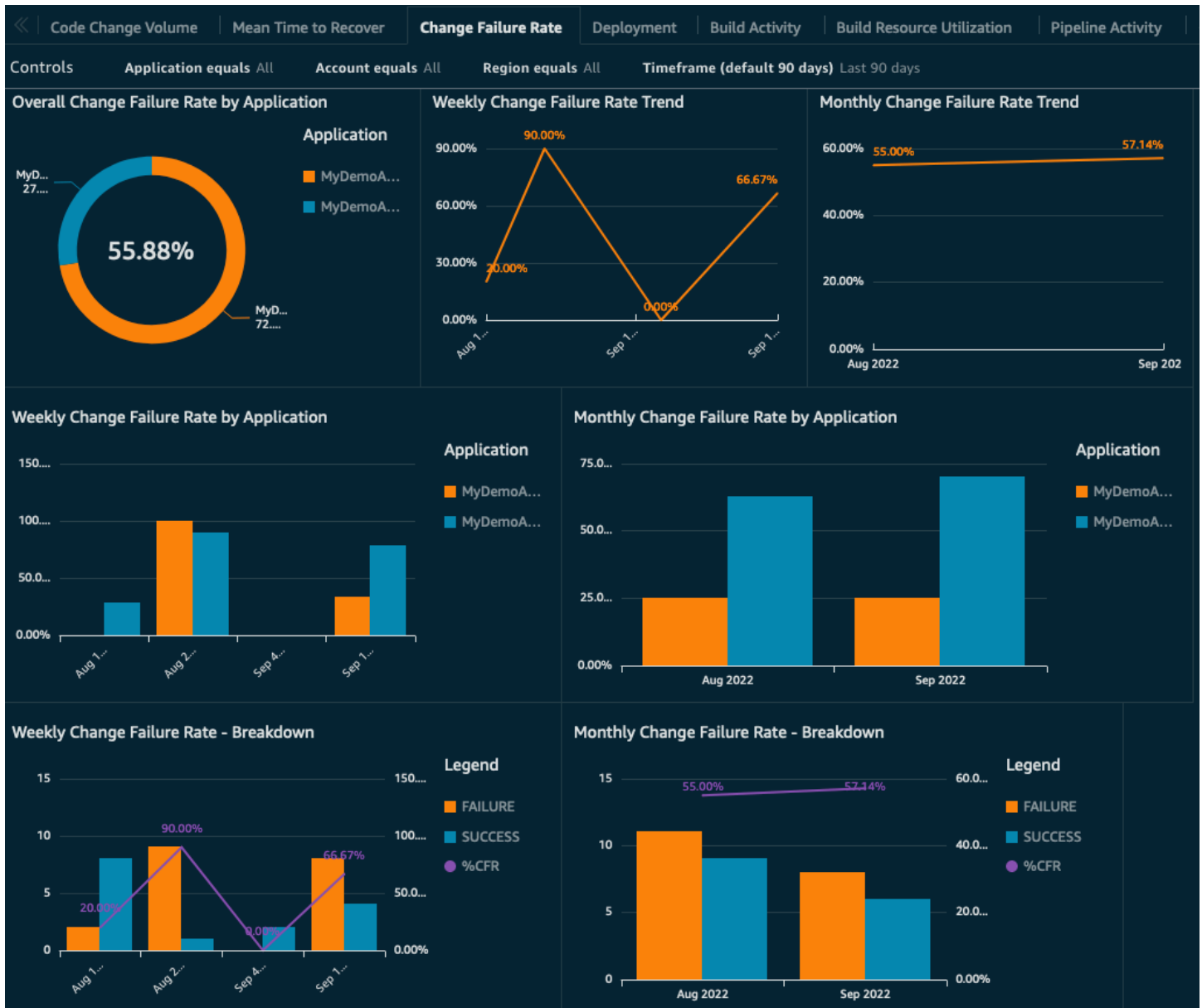
This dashboard displays outage minutes by application and the average time it takes to restore an application from a failure to success state. It provides a weekly, monthly and aggregated view of the metrics by application. You can filter data by application type, application name, AWS account, AWS Region, or time period (default to last 90 days) using the custom filter as needed. For more information about the metrics, refer to [Mean time to recover metrics](#).



Mean time to recover dashboard

Change failure rate dashboard

This dashboard displays the frequency of deployment failures per application by measuring the ratio of unsuccessful to total deployments. It provides a weekly, monthly and aggregated view of the metrics by application. You can filter metrics by application, AWS account, AWS Region, or time period (default to last 90 days) using the custom filter. For more information about the metrics, refer to [Change failure rate metrics](#).



Change failure rate dashboard

Deployment dashboards

This dashboard displays the deployment frequency and state (success/failure) by application. It provides a weekly, monthly and aggregated view of the metrics by application. You can filter metrics by application, AWS account, AWS Region, or time period (default to last 90 days) using the custom filter. For more information about the metrics, refer to [Deployment metrics](#).

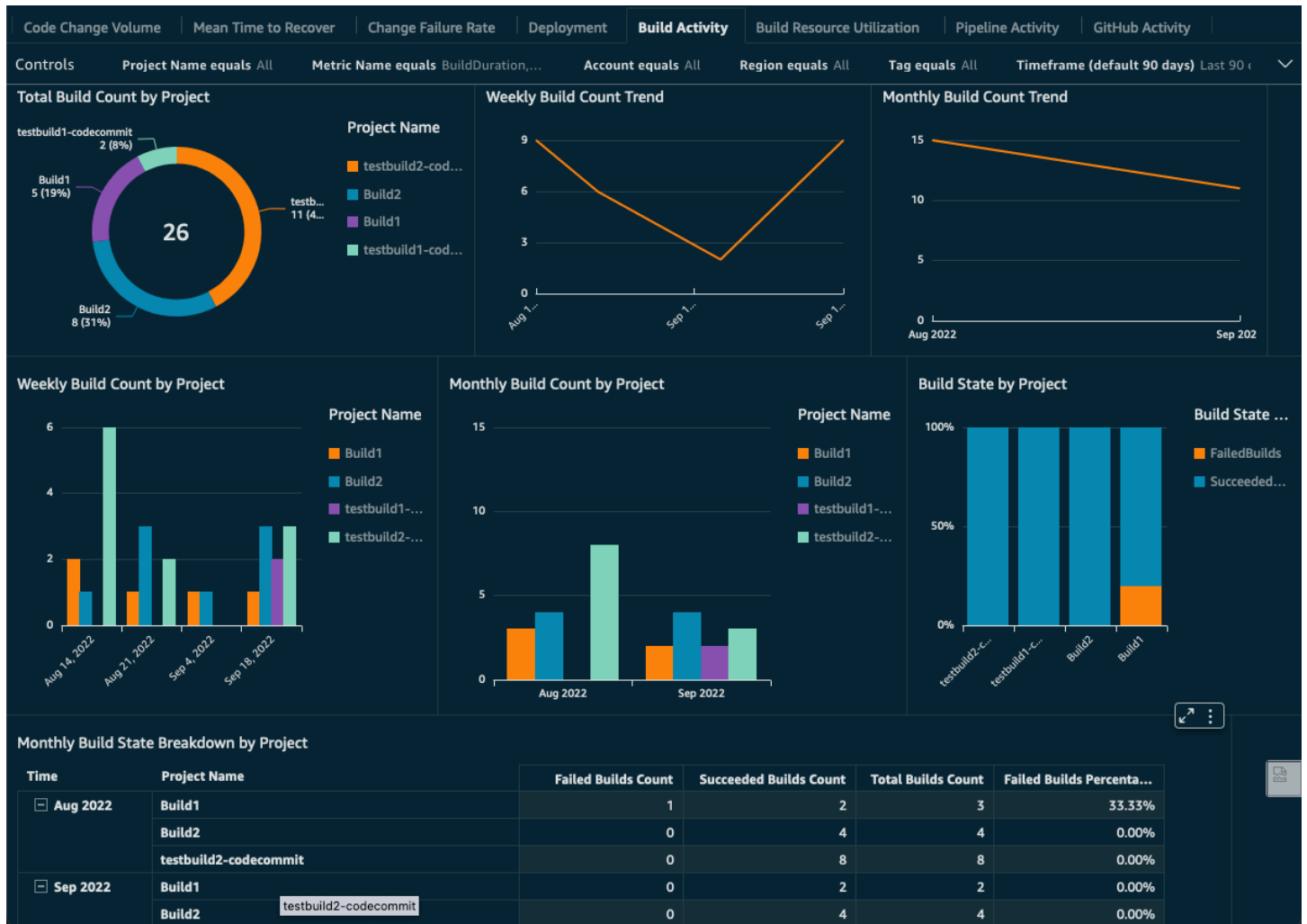


Deployment dashboard

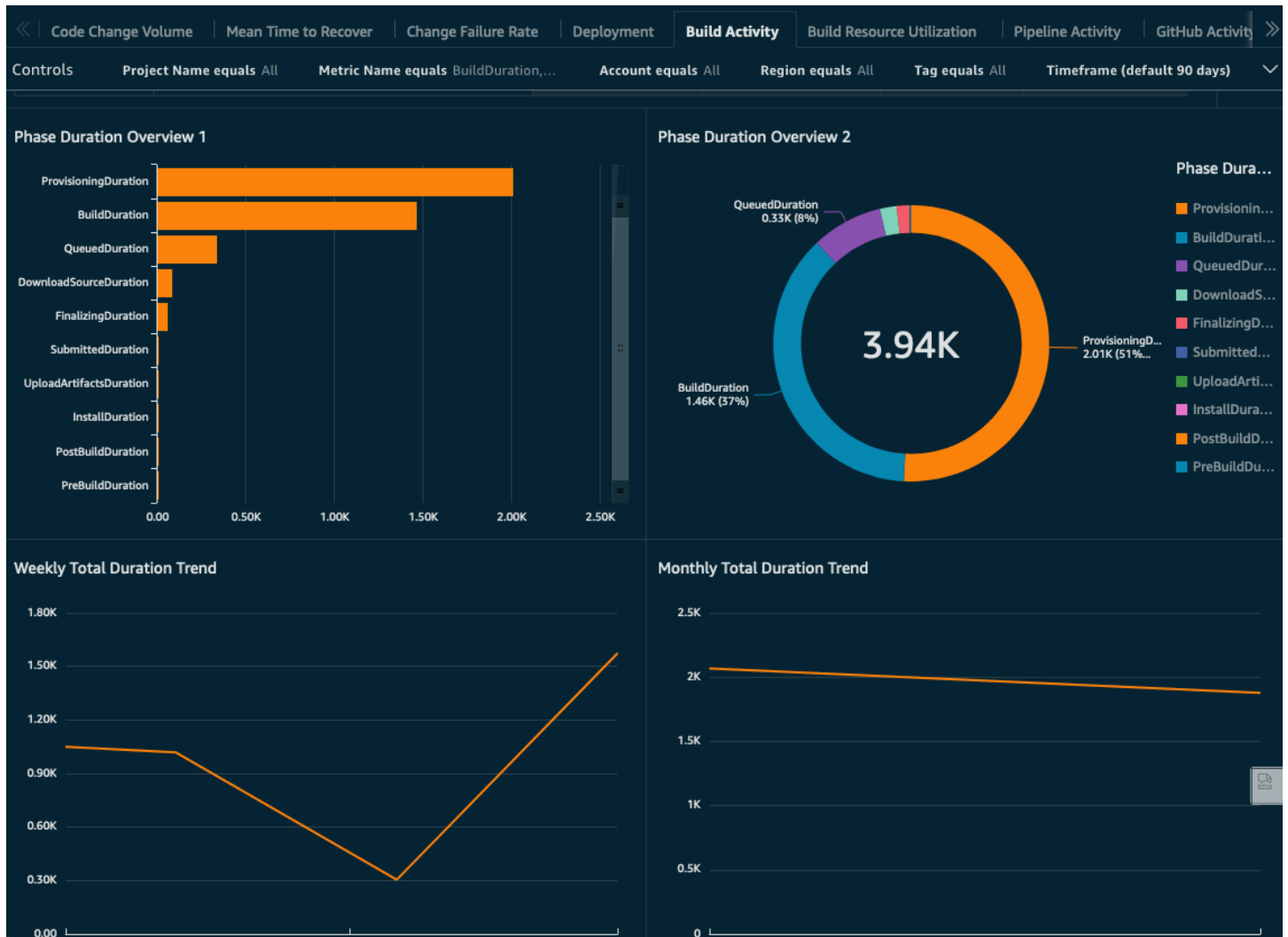
Build dashboards

Build activity dashboard

This dashboard displays the code build frequency, duration and state (success/failure) by project. It provides a weekly, monthly, and aggregated view of the metrics by project. You can filter metrics by project, metric name (for example, FailedBuilds, SucceededBuilds, and others), AWS account, AWS Region, tag, or time period (default to last 90 days) using the custom filter. For more information about the metrics, refer to [Build metrics](#).



Build activity dashboard - 1



Build activity dashboard - 2

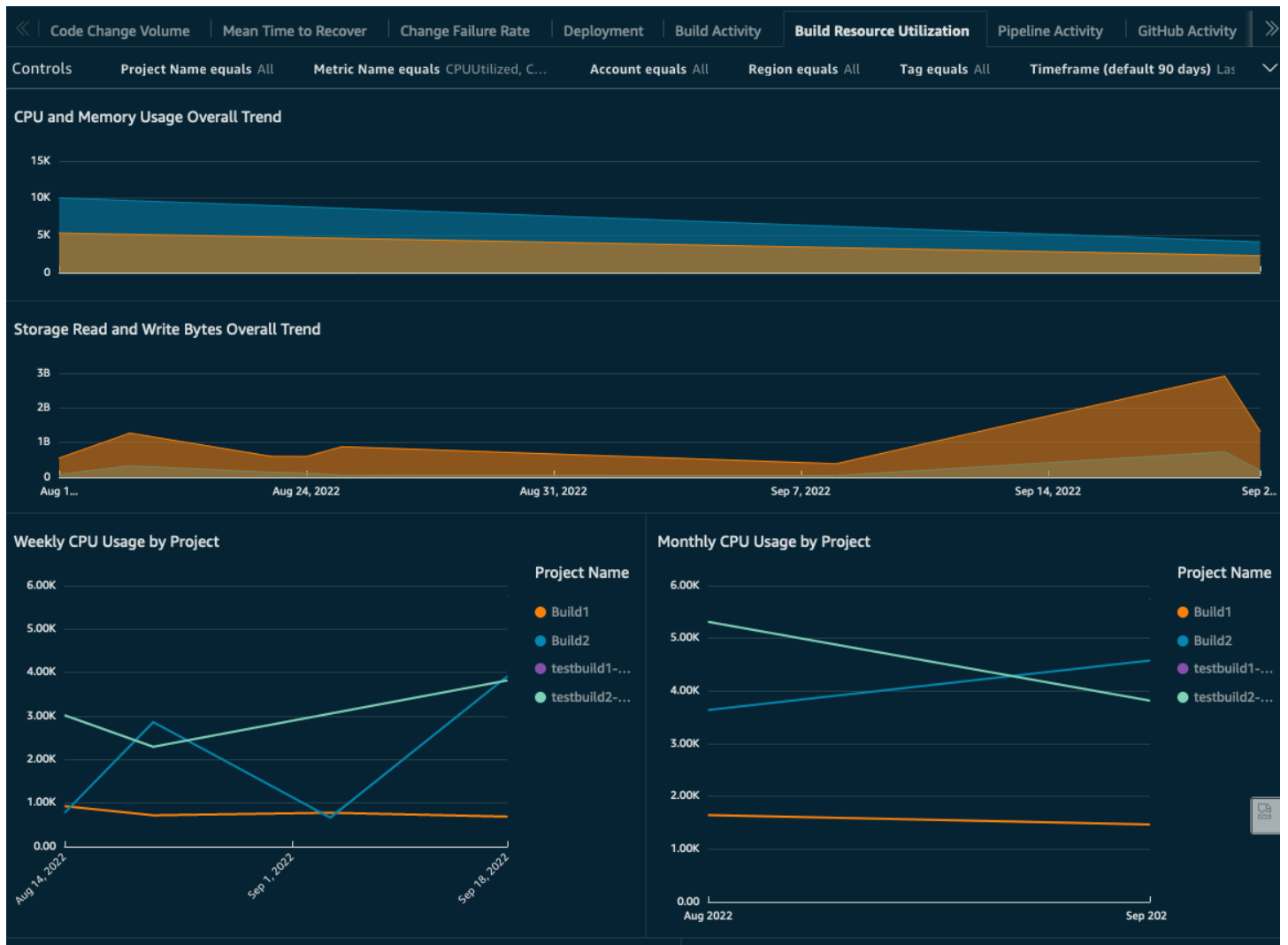


Build activity dashboard - 3

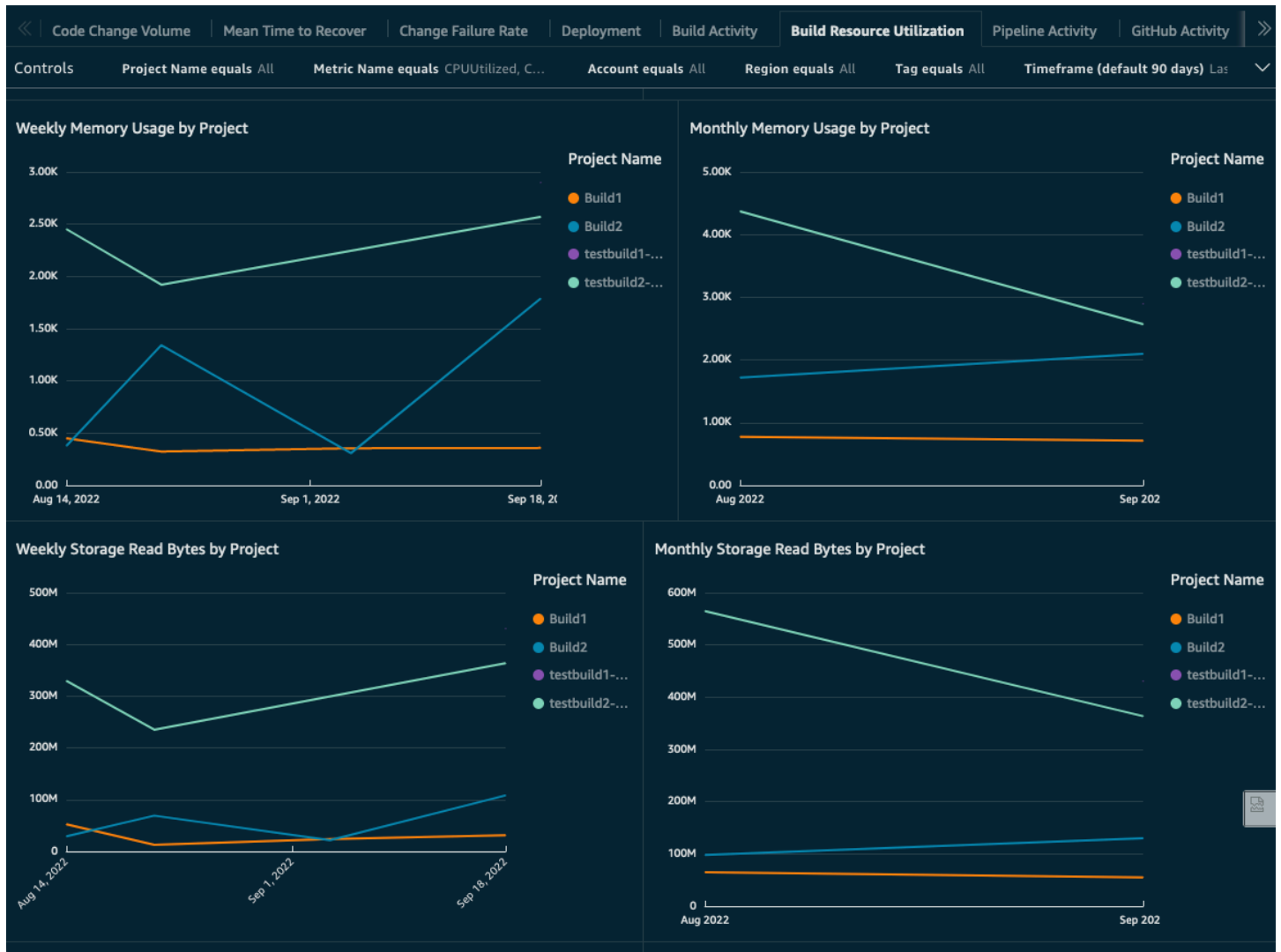
Build resource utilization dashboard

This dashboard displays code build resource utilization metrics for CPU, memory, and storage utilization by project and build. It provides a weekly, monthly, and aggregated view of the metrics by projects and build. You can filter metrics by project name, metric name (for example, CPUUtilized, MemoryUtilized and others), AWS account, AWS Region, tag, or time period (default to last 90 days) using the custom filter. Resource utilization metrics are not available for builds shorter than one minute and they are not supported in all the AWS Regions where AWS CodeBuild is supported. For a complete list of the supported Regions, refer to [Monitoring CodeBuild resource](#)

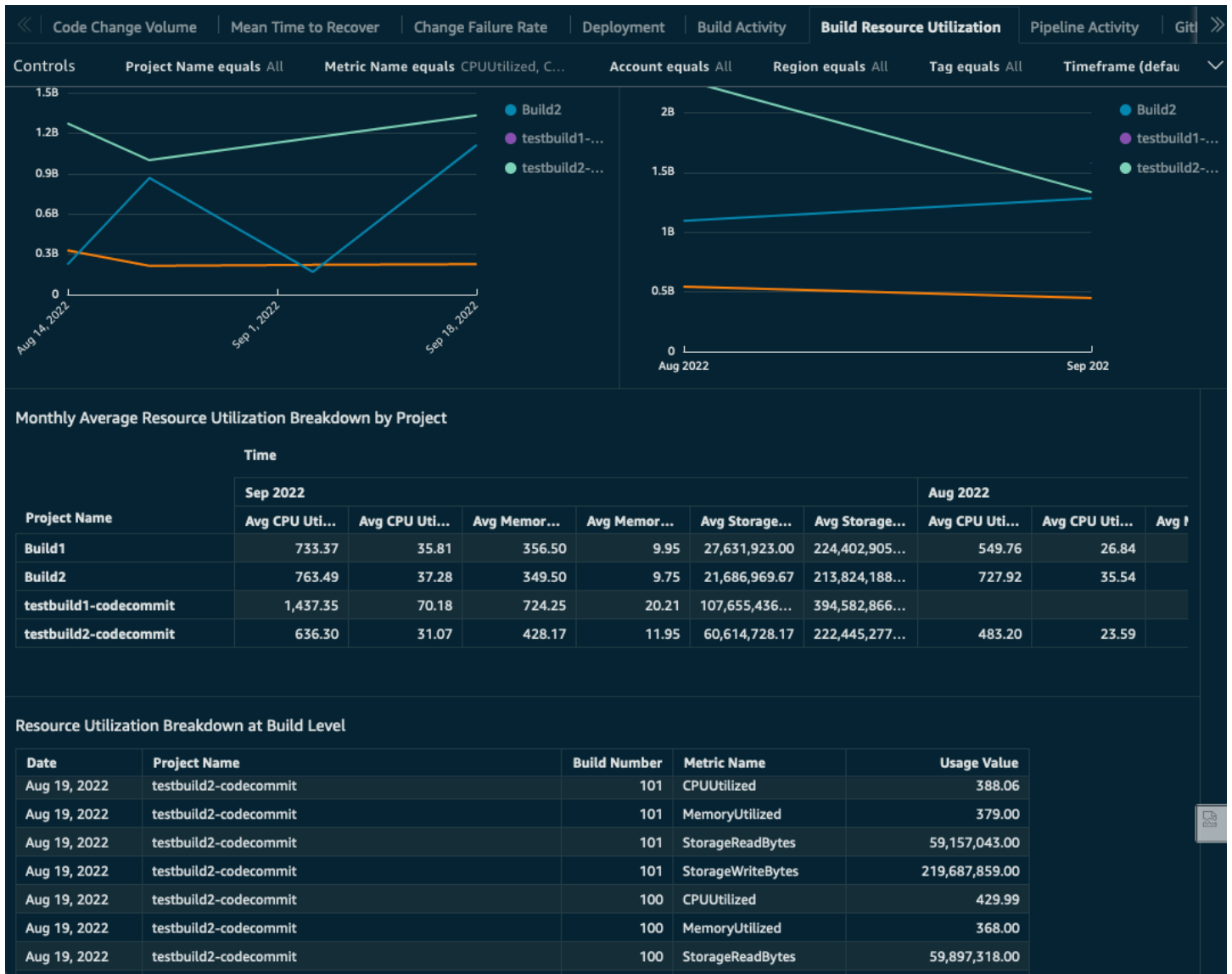
[utilization metrics](#) in the AWS CodeBuild User Guide. For more information about the metrics, refer to [Build metrics](#).



Build resource utilization dashboard - 1



Build resource utilization dashboard - 2



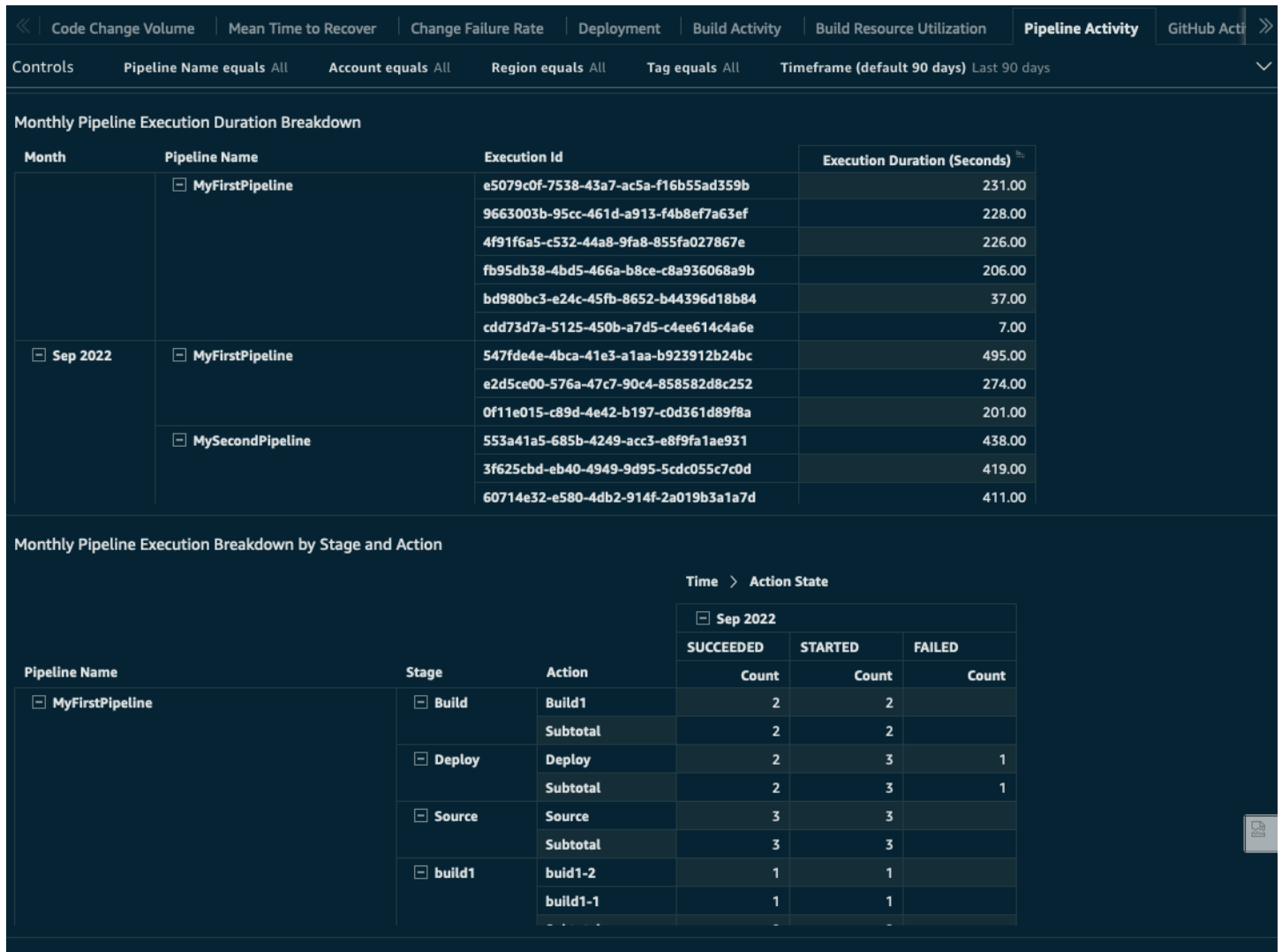
Build resource utilization dashboard - 3

Pipeline dashboard

This dashboard displays pipeline execution state (failure, success, and others), execution duration and frequency in addition to the state at stage and action level. It provides a weekly, monthly, and aggregated view of the metrics by pipeline. You can filter metrics by pipeline, AWS account, AWS Region, tag, or time period (default to last 90 days) using the custom filter. For more information about the metrics, refer to [Pipeline metrics](#).



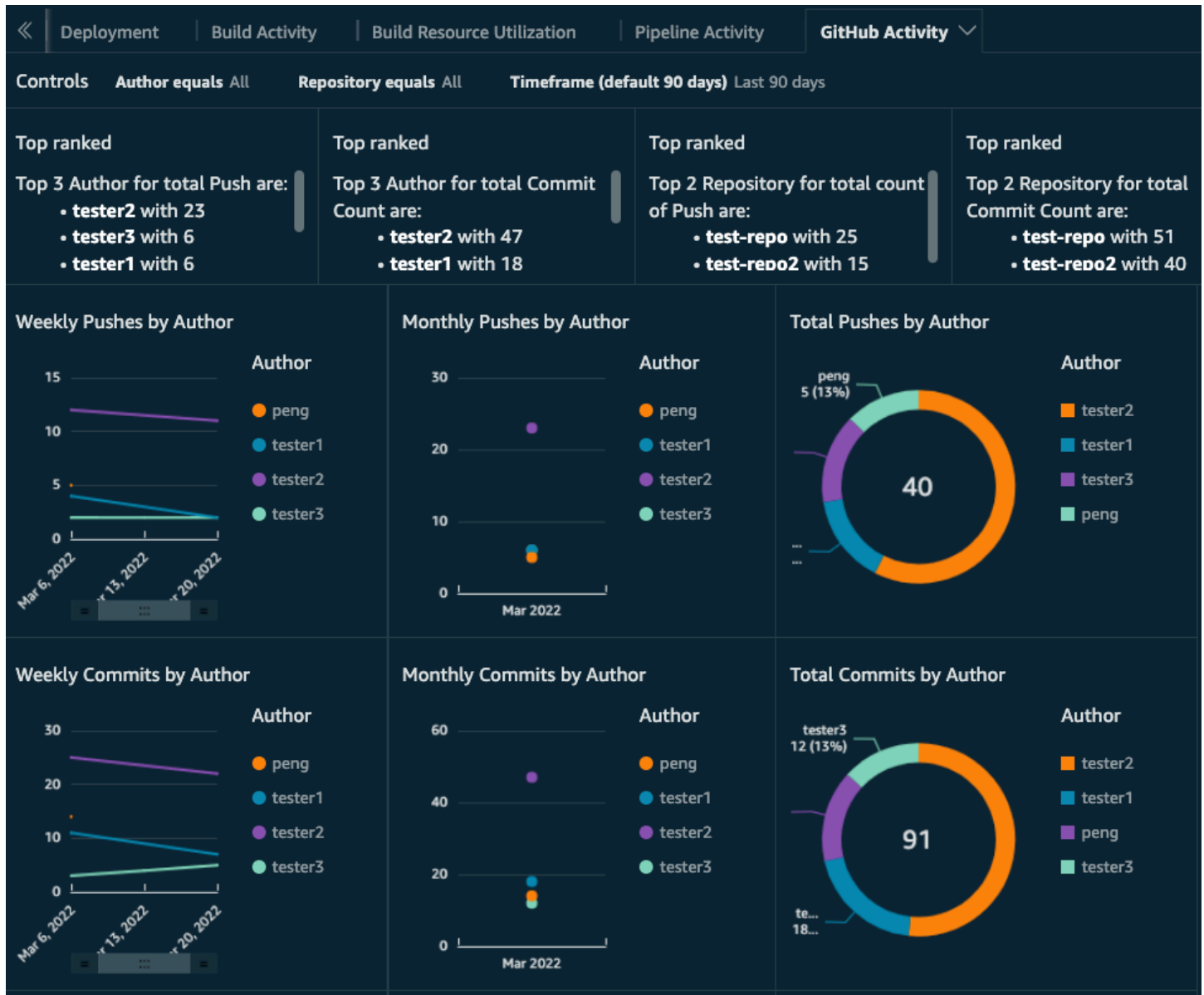
Pipeline dashboard - 1



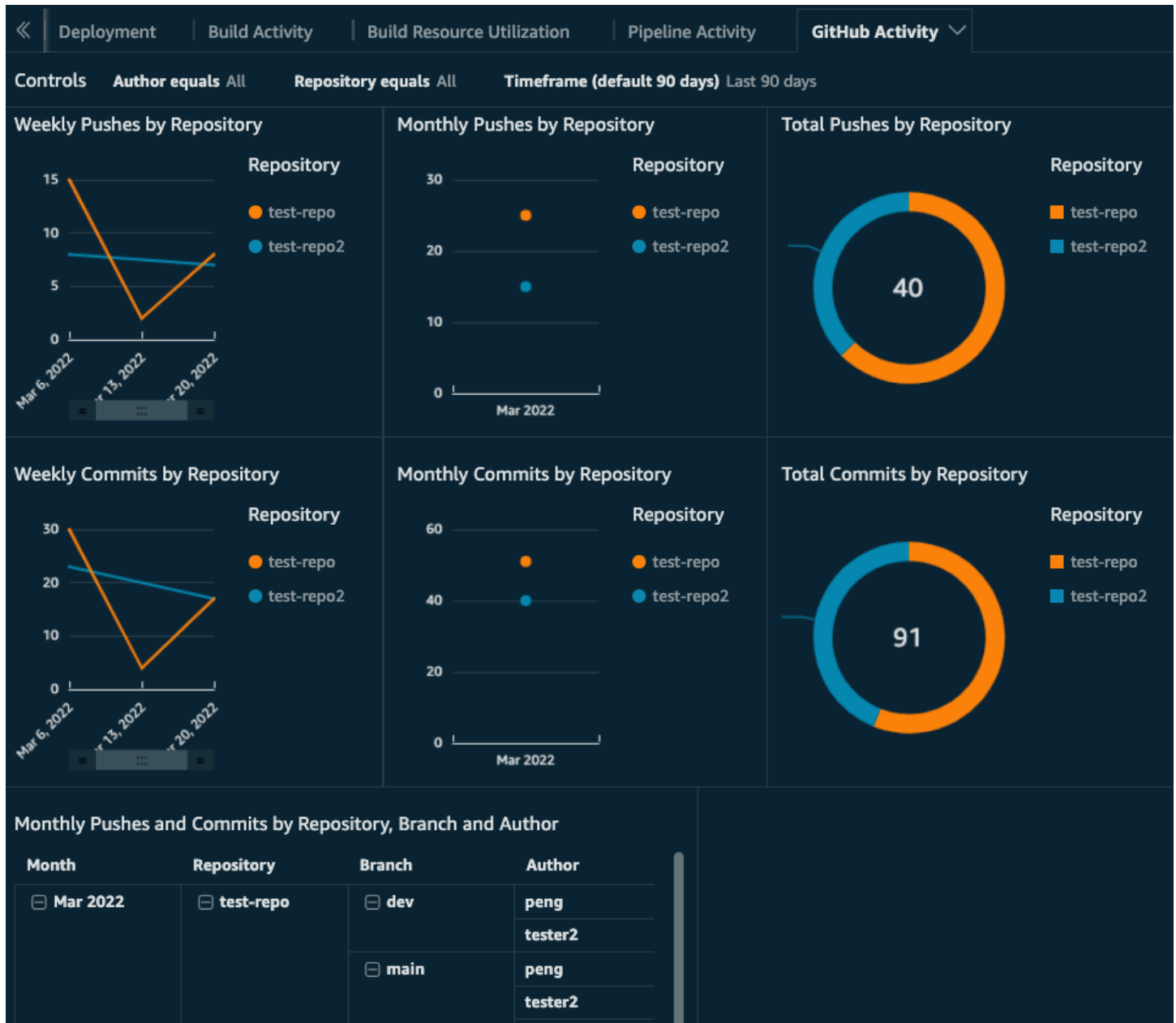
Pipeline dashboard - 2

GitHub activity dashboard

This dashboard displays the number of code changes (pushes and commits) made by author and repository. It provides a weekly, monthly, and aggregated view of the metrics by author and repository. You can filter data by author, repository, or time period (default to last 90 days) using the custom filter as needed. For more information about the metrics, refer to [GitHub activity metrics](#).



GitHub activity dashboard - 1



GitHub activity dashboard - 2

Developer Guide

Source Code

Visit our [GitHub Repository](#) to download the templates and scripts for this solution, and to share your customizations with others. The CloudFormation templates are generated using the [AWS Cloud Development Kit \(AWS CDK\)](#). Refer to the [README.md](#) file for additional information.

Reference

This section includes information about an optional feature for collecting unique metrics for this solution and a list of builders who contributed to this solution.

Anonymized data collection

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. 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 Live Streaming on AWS deployment
- **Timestamp** - - Data-collection timestamp
- **Data**- Nested structure containing the following information:
 - **Region** - The AWS Region in which the solution is deployed.
 - **Version** - The version of the deployed solution.
 - **RequestType** - Stack action: Create, Update, or Delete.
 - **DataType** - Sender of the metrics such as Lambda function.
 - **AthenaQueryExecutionCount** - Number of successful Athena queries run by the solution (mainly QuickSight).
 - **QuickSightDeployed** -Yes or No. Customer configuration at stack deployment.
 - **AthenaQueryDataRetrievalDuration** - The duration in which Athena query retrieves data. By default Athena fetches data within the past 90 days. Customer configuration at stack deployment.
 - **Repository** - `all` or `customer list` . Indicates if a customer chooses to track all repositories or enter a list of selected repositories. Customer configuration at stack deployment.
 - **S3TransitionDays** - The number of days after which Amazon S3 objects are transitioned to Amazon S3 Glacier storage class. Customer configuration at stack deployment.
 - **UseGitHubRepository** - Whether or not GitHub repository is used – Yes or No.
 - **UseWebhookSecret** - Whether or not a secret token is used to secure webhook and authorize request: Yes or No.

- **UseMultiAccount** -Whether or not the multi-account feature is turned on – Yes or No.
- **PrincipalType** -The principal type of other accounts when the multi-account feature is turned on – AWS Account Number or AWS Organization IDs.
- **PrincipalCount** - Count of AAWS Account Number or AWS Organization ID when the multi-account feature is turned on.
- **CodeCommitTagsCount**- Number of tags for CodeCommit repositories.
- **CodeBuildTagsCount** - Number of tags for CodeBuild projects.
- **CodePipelineTagsCount** - Number of tags for CodePipeline pipelines.
- **Stack** - sharing or monitoring. Indicates if the CloudFormation stack is the main stack deployed in a monitoring account, or the sharing account stack deployed in a sharing account.

AWS owns the data gathered though this survey. Data collection is subject to the [AWS Privacy Policy](#). To opt out of this feature, complete the following steps before launching the AWS CloudFormation template.

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:

```
AnonymizedData:  
  SendAnonymizedData:  
    Data: Yes
```

to:

```
AnonymizedData:  
  SendAnonymizedData:  
    Data: 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**, choose **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

- Aijun Peng
- Mike O'Brien
- Nikhil Reddy
- Aaron Schuetter
- Thiemo Belmega
- Max Granat
- Eric Van Wieren
- George Lenz

Revisions

Date	Change
March 2021	Initial release
June 2021	Release version 1.1.0: Added support for AWS CodeBuild and AWS CodePipeline metrics. For additional information, refer to the CHANGELOG.md file.
April 2022	Release version 1.5.0: Added GitHub integration - GitHub activity metrics for push events. Added mean time to recovery (MTTR) metric for CodePipeline. For additional information, refer to the CHANGELOG.md file.
October 2022	Release version 1.8.0: Added multi-account multi-Region data ingestion and tag filter for AWS CodeCommit, CodeBuild and CodePipeline. For additional information, refer to the CHANGELOG.md file.
December 2022	Release version 1.8.1: Added application registry integration for monitoring the solution, and upgraded node version to 16. For additional information, refer to the CHANGELOG.md file.
January 2023	Release version 1.8.2: Upgraded dependencies to mitigate CVE-2022-46175, CVE-2022-23491. For additional information, refer to the CHANGELOG.md file.
April 2023	Release version 1.8.3: Mitigated impact caused by new default settings for S3 Object Ownership (ACLs disabled) for all new S3

Date	Change
	buckets. For additional information, refer to the CHANGELOG.md file.
August 2023	Release v1.8.4: Refactored the code to reduce complexity and upgraded dependencies. For additional information, refer to the CHANGELOG.md file.
October 2023	Release v1.8.5: Updated package versions to resolve security vulnerabilities. For additional information, refer to the CHANGELOG.md file.
November 2023	Documentation update: Added Confirm cost tags associated with the solution to the Monitoring the solution with AWS Service Catalog AppRegistry section.
December 2023	Release v1.8.6: Upgraded package versions to resolve security vulnerabilities. Upgraded Lambda runtime node and Python versions. For additional information, refer to the CHANGELOG.md file.
January 2024	Release v1.8.7: Fixed handling of AWS SDK v3 exceptions. For additional information, refer to the CHANGELOG.md file.
March 2024	Release v1.8.8: Upgraded synthetics canary runtime, upgraded CDK and updated solutions constructs and dependencies. For additional information, refer to the CHANGELOG.md file.

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.

DevOps Monitoring Dashboard on AWS is licensed under the terms of the of the Apache License Version 2.0 available at [The Apache Software Foundation](https://www.apache.org/licenses/LICENSE-2.0).