DevOps Monitoring Dashboard on AWS

Implementation Guide
# Table of Contents

Welcome ........................................................................................................................................... 1
Cost .................................................................................................................................................. 2
Architecture overview ....................................................................................................................... 4
Solution components ......................................................................................................................... 5
Amazon EventBridge Events rule ....................................................................................................... 5
Amazon Kinesis Data Firehose .......................................................................................................... 5
AWS Lambda .................................................................................................................................... 5
Amazon Simple Storage Service (Amazon S3) .................................................................................. 6
AWS Glue and Amazon Athena ......................................................................................................... 6
Amazon QuickSight .......................................................................................................................... 6
Amazon CloudWatch synthetics canary and Amazon CloudWatch alarm ....................................... 6
Amazon CloudWatch alarm for AWS CodePipeline ......................................................................... 6
Amazon API Gateway ...................................................................................................................... 7
Security .......................................................................................................................................... 8
AWS IAM roles ............................................................................................................................... 8
Amazon S3 ...................................................................................................................................... 8
Deployment considerations .............................................................................................................. 9
Regional deployment ...................................................................................................................... 9
AWS CI/CD pipeline deployment ...................................................................................................... 9
Amazon QuickSight deployment ...................................................................................................... 9
Amazon CloudWatch alarm for Amazon CloudWatch Synthetics canary deployment .................. 9
Amazon CloudWatch alarm for AWS CodePipeline deployment .................................................... 9
AWS CloudFormation template ....................................................................................................... 10
Automated deployment .................................................................................................................... 11
Prerequisites .................................................................................................................................. 11
Deployment overview ...................................................................................................................... 11
Step 1: Launch the stack ................................................................................................................... 11
Step 2: Configure Amazon QuickSight ............................................................................................. 14
Step 3: Set up GitHub webhook ....................................................................................................... 15
Resources ........................................................................................................................................ 16
Update the stack .............................................................................................................................. 17
DevOps metrics list ........................................................................................................................... 18
Code change volume metrics .......................................................................................................... 18
Mean time to recover metrics .......................................................................................................... 18
Change failure rate metrics ............................................................................................................ 18
Deployment metrics ........................................................................................................................ 18
Build metrics .................................................................................................................................. 18
Pipeline metrics ............................................................................................................................... 19
GitHub activity metrics ................................................................................................................... 19
Retrieve the Amazon QuickSight Principal ARN ............................................................................ 20
Set up Amazon CloudWatch synthetics canary and Amazon CloudWatch alarm .......................... 21
Set up Amazon CloudWatch alarm for AWS CodePipeline ............................................................. 24
Database schema information ......................................................................................................... 25
Amazon QuickSight dashboards visuals .......................................................................................... 28
Code change volume dashboard ....................................................................................................... 28
Mean time to recover dashboard ..................................................................................................... 28
Change failure rate dashboard ......................................................................................................... 29
Deployment dashboards ................................................................................................................... 30
Build dashboards ............................................................................................................................. 31
Build activity dashboard .................................................................................................................. 31
Build resource utilization dashboard ............................................................................................... 34
Pipeline dashboard .......................................................................................................................... 37
GitHub activity dashboard ................................................................................................................. 39
Build visualizations with Amazon Athena and Tableau .................................................................. 42
Running queries and work with query results and output files in Amazon Athena ......................... 43
Uninstall the solution ..................................................................................................................... 44
   Using the AWS Management Console .................................................................................... 44
   Using AWS Command Line Interface .................................................................................... 44
Collection of operational metrics ............................................................................................... 45
Source code .................................................................................................................................. 47
Revisions ....................................................................................................................................... 48
Contributors ................................................................................................................................. 49
Notices ......................................................................................................................................... 50
AWS glossary ............................................................................................................................... 51
Ingesting, analyzing, and visualizing metrics with DevOps Monitoring Dashboard on AWS

Publication date: March 2021 (last update (p. 48): April 2022)

Collecting performance and operational metrics in your continuous integration/continuous delivery (CI/CD) pipeline is important in order 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 (p. 18). These metrics are presented in Amazon QuickSight dashboards for visualization. For more information about data visualization, refer to Amazon QuickSight dashboards visuals (p. 28).

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 (p. 42). 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 (p. 43).

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.
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. We recommend creating a budget through AWS Cost Explorer to help manage costs.

The following tables are example cost breakdowns for running this solution in the US East (N. Virginia) Region (excludes free tier). Prices are subject to change. For full details, refer to the pricing page for each AWS service used in this solution.

Example 1: Turn on QuickSight without GitHub

<table>
<thead>
<tr>
<th>AWS Service</th>
<th>Dimensions/Month</th>
<th>Cost/Month</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon Athena</td>
<td>100 queries, 10 GB data scanned per query</td>
<td>~$5.00</td>
</tr>
<tr>
<td>Amazon Kinesis Data Firehose</td>
<td>100 GB</td>
<td>~$2.90</td>
</tr>
<tr>
<td>Amazon Simple Storage Service (Amazon S3)</td>
<td>100 GB</td>
<td>~$2.30</td>
</tr>
<tr>
<td>Amazon QuickSight</td>
<td>1 author, 10 readers, access 2 times per month for each reader</td>
<td>~$24.00</td>
</tr>
<tr>
<td><strong>Total:</strong></td>
<td></td>
<td>~$34.20</td>
</tr>
</tbody>
</table>

Example 2: Turn on QuickSight, GitHub with Secrets Manager

<table>
<thead>
<tr>
<th>AWS Service</th>
<th>Dimensions/Month</th>
<th>Cost/Month</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon Athena</td>
<td>100 queries, 10 GB data scanned per query</td>
<td>~$5.00</td>
</tr>
<tr>
<td>Amazon Kinesis Data Firehose</td>
<td>100 GB</td>
<td>~$2.90</td>
</tr>
<tr>
<td>Amazon Simple Storage Service (Amazon S3)</td>
<td>100 GB</td>
<td>~$2.30</td>
</tr>
<tr>
<td>Amazon QuickSight</td>
<td>1 author, 10 readers, access 2 times per month for each reader</td>
<td>~$24.00</td>
</tr>
<tr>
<td>Amazon API Gateway</td>
<td>1 million requests</td>
<td>~$1.0</td>
</tr>
<tr>
<td>AWS Secrets Manager</td>
<td>1 secret, 1 million API calls</td>
<td>~5.40</td>
</tr>
<tr>
<td><strong>Total:</strong></td>
<td></td>
<td>~$40.60</td>
</tr>
</tbody>
</table>

**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.
Architecture overview

Deploying this solution with the default parameters builds the following environment in the AWS Cloud.

Figure 1: 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. 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 Kinesis 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 Kinesis Data Firehose delivery stream. For GitHub push events, an Amazon API endpoint is created to post these events and deliver them to a Kinesis Data 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, if you have set up the canary and alarm in your account. This alarm is needed to gather data for calculating Mean Time to Recover (MTTR) metrics.

4. Amazon Kinesis Data Firehose uses an AWS Lambda function for data transformation. The Lambda function extracts relevant data to each metric and sends it to an Amazon S3 bucket 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.
Solution components

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 execution 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 two Amazon Kinesis Data Firehose delivery streams to process raw data from data sources. The 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 Lambdas: These functions perform Lambda transformation within Amazon Kinesis Data Firehose. They parse raw data from data sources, extracts relevant data, and returns it back to Firehouse delivery stream for downstream operation.
- Query Runner Lambda: This function runs Amazon Athena queries to add Athena partitions and create views at solution deployment.
- Athena Partition Lambda: This function runs on a daily schedule to add a new daily partition to the Amazon Athena table.
- QuickSight Custom Resource Lambda: This function creates Amazon QuickSight resources at solution deployment.
Solution Helper Custom Resource Lambda: This function generates UUID for each solution deployment.

Amazon Simple Storage Service (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 executions and cost management. For more information, refer to Database schema information (p. 25).

Amazon QuickSight

This solution uses Amazon QuickSight for data visualization. You must create an Amazon QuickSight enterprise admin user account if you don’t already have one. To create a user account, 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 account, this solution will not deploy Amazon QuickSight resources in your account. For more information, refer to Amazon QuickSight dahsboards visuals (p. 28).

Amazon CloudWatch synthetics canary and Amazon CloudWatch alarm

This solution uses Amazon CloudWatch synthetics canary and Amazon CloudWatch Alarm to collect data needed for calculating Mean Time to Recover (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 (failure or success). 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 (p. 21).

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 execution 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 (p. 24).

**Amazon API Gateway**

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

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

AWS IAM roles

AWS Identity and Access Management (IAM) roles allow you to assign granular access policies and permissions to services and users in 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.
Deployment considerations

Regional deployment

This solution uses services such as Amazon Athena, Amazon Kinesis 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.

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 (p. 20).

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 (p. 21).

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 (p. 24).
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 template, which you can download before deployment.

**AWS CloudFormation 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 Kinesis Data Firehose, AWS Glue and Amazon Athena databases, and Amazon QuickSight resources (optional). You can also customize the template to meet your specific needs.
Automated deployment

Before you launch the solution, review the considerations discussed in this guide. Follow the step-by-step instructions in this section to configure and deploy the solution into your account.

**Time to deploy the template:** Approximately 10 mins

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 (p. 20). 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.

Deployment overview

**Step 1. Launch the Stack (p. 11)**

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

**Step 2. Configure Amazon QuickSight (p. 14)**

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

Step 1: Launch the stack

This automated AWS CloudFormation template deploys the DevOps Monitoring Dashboard on AWS solution in the AWS Cloud. Before you launch the stack, you must complete the prerequisites (p. 11).

**Important**

You are responsible for the cost of the AWS services used while running this solution. For more details, refer to the **Cost** (p. 2) 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.
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 Kinesis 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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Metrics Configuration</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Athena Query Data Duration (Days)</td>
<td>90</td>
<td>Enter a duration (days) that Athena query uses to retrieve data. By default, Athena query retrieves data within the last 90 days. We recommend that you to limit the duration for performance optimization and cost reduction.</td>
</tr>
<tr>
<td>AWS CodeCommit Repository List</td>
<td>ALL</td>
<td>List of the names of AWS CodeCommit repositories that will be monitored. Must be single-quoted and comma separated. For example: 'MyRepository1', 'MyRepository2' To monitor all the repositories, leave default ALL value.</td>
</tr>
<tr>
<td><strong>S3 Configuration</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S3 Transition Days</td>
<td>365</td>
<td>Enter the number of days after which you would like to transition Amazon S3 objects to Amazon S3 Glacier storage</td>
</tr>
</tbody>
</table>
### DevOps Monitoring Dashboard on AWS Implementation Guide

#### Step 1: Launch the stack

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td></td>
<td>By default objects are transitioned to Amazon S3 Glacier 365 days (one year) after creation.</td>
</tr>
<tr>
<td>Amazon QuickSight Principal ARN</td>
<td>&lt;Optional Input&gt;</td>
<td>Provide an Amazon QuickSight admin user ARN to automatically create QuickSight resources. Amazon QuickSight Enterprise edition must be activated for the account. For example: arn:aws:quicksight:AWSRegion:AWSAccountId:default/QuickSightUserName. To deactivate QuickSight dashboards creation, do not enter a value. For more information, refer to Prerequisites Step 2 (p. 11).</td>
</tr>
<tr>
<td>Use GitHub Repository</td>
<td>No</td>
<td>Select Yes if GitHub is used, otherwise leave it as No.</td>
</tr>
<tr>
<td>Webhook Secret Token</td>
<td>&lt;Optional Input&gt;</td>
<td>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 (&quot;). 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.</td>
</tr>
</tbody>
</table>
DevOps Monitoring Dashboard
on AWS Implementation Guide

Step 2: 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 (p. 42).

1. After the stack successfully deploys, go to the Outputs tab of the stack and make a note of the values for QSAnalysisURL, 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.

### Parameter Table

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allowed IP Addresses</td>
<td>192.30.252.0/22,185.199.108.0/22,140.82.112.0/20,143.55.64.0/20</td>
<td>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.</td>
</tr>
</tbody>
</table>

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

You can view the status of the stack in the AWS CloudFormation console in the Status column. You should receive a CREATE_COMPLETE status in approximately 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.
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 **QSAnalysisURL** 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** (p. 28) 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 3: 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.

  ![API endpoint to receive GitHub events](image)

- **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.
## Resources

**AWS services**

<table>
<thead>
<tr>
<th>AWS CloudFormation</th>
<th>Amazon Kinesis Data Firehose</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS Glue</td>
<td>Amazon Simple Storage Service</td>
</tr>
<tr>
<td>Amazon EventBridge</td>
<td>Amazon QuickSight</td>
</tr>
<tr>
<td>AWS Identity and Access Management</td>
<td>Amazon Athena</td>
</tr>
<tr>
<td>AWS Lambda</td>
<td>Amazon CloudWatch</td>
</tr>
<tr>
<td>AWS Secrets Manager</td>
<td>Amazon API Gateway</td>
</tr>
</tbody>
</table>
# Update the stack

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 (p. 11) 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.
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 (p. 28).

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 (p. 28).

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 (p. 29).

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 (p. 30).

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 (p. 31).
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 (p. 37).

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 (p. 39).
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-1:1111111111111: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.
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.

Figure 3: Select CloudWatchSynthetics metrics
- From the All metrics tab, select By Canary and then select your canary and SuccessPercent metric name.
Figure 4: Select your canary and SuccessPercent metric

- Name the alarm SO0143-[my-application-name]-[my-repository-name]-MTTR. For example, SO0143-[MyDemoApplication]-[MyDemoRepo]-MTTR. SO0143 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 figure 4:
Figure 5: 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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CodePipeline Name</td>
<td></td>
<td>The name of the CodePipeline that you want to monitor. This name will become part of the CloudWatch alarm name.</td>
</tr>
<tr>
<td>Create a new log group</td>
<td>YES</td>
<td>Whether or not to create a new log group.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>Log Group Name</td>
<td></td>
<td>Name of the log group (new or existing) to write the CloudWatch metrics into.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>Repository Name</td>
<td></td>
<td>The name of the repository which is the source of your CodePipeline. This name will become part of the CloudWatch alarm name.</td>
</tr>
</tbody>
</table>

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

Figure 6: CloudWatch alarm
Database schema information

The following diagrams display 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 (p. 6). 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 (p. 6). It is the base table for the GitHub change activity view.
Figure 7: DevOps metrics database schema structure

<table>
<thead>
<tr>
<th><strong>aws_codebuild_metrics_table</strong></th>
<th><strong>code_build_detail_view</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>account</td>
</tr>
<tr>
<td>id</td>
<td>timestamp</td>
</tr>
<tr>
<td>detail_type</td>
<td>region</td>
</tr>
<tr>
<td>source</td>
<td>namespace</td>
</tr>
<tr>
<td>account</td>
<td>metric_name</td>
</tr>
<tr>
<td>time</td>
<td>project_name</td>
</tr>
<tr>
<td>region</td>
<td>build_id</td>
</tr>
<tr>
<td>resources</td>
<td>count</td>
</tr>
<tr>
<td>detail</td>
<td>sum</td>
</tr>
<tr>
<td>detail</td>
<td>max</td>
</tr>
<tr>
<td>detail</td>
<td>min</td>
</tr>
<tr>
<td>created_at</td>
<td>unit</td>
</tr>
</tbody>
</table>

Figure 8: CodeBuild metrics database schema structure

<table>
<thead>
<tr>
<th><strong>aws_github_metrics_table</strong></th>
<th><strong>github_change_activity_view</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>repository_name</td>
<td>repository_name</td>
</tr>
<tr>
<td>branch_name</td>
<td>branch_name</td>
</tr>
<tr>
<td>author_name</td>
<td>author_name</td>
</tr>
<tr>
<td>event_name</td>
<td>event_name</td>
</tr>
<tr>
<td>commit_id</td>
<td>commit_count</td>
</tr>
<tr>
<td>time</td>
<td>time</td>
</tr>
<tr>
<td>created_at</td>
<td>created_at</td>
</tr>
</tbody>
</table>

Figure 9: 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 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 (p. 18).

![Code change volume dashboard](image)

**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, name, 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 (p. 18).

Figure 11: 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 or time period (default to last 90 days) using the custom filter. For more information about the metrics, refer to Change failure rate metrics (p. 18).
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 or time period (default to last 90 days) using the custom filter. For more information about the metrics, refer to Deployment metrics (p. 18).
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 (FailedBuilds, SucceededBuilds, and BuildDuration), or time period (default to last 90 days) using the custom filter. For more information about the metrics, refer to Build metrics (p. 18).
Figure 14: Build activity dashboard - 1
Figure 15: Build activity dashboard - 2
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, metric name (CPUUtilized, MemoryUtilized), 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 (p. 18).
Figure 17: Build resource utilization dashboard - 1
Figure 18: Build resource utilization dashboard - 2
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, or time period (default to last 90 days) using the custom filter. For more information about the metrics, refer to Pipeline metrics (p. 19).
Figure 20: Pipeline dashboard - 1
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 (p. 19).
Figure 22: GitHub activity dashboard - 1
Figure 23: GitHub activity dashboard - 2
Build visualizations with Amazon Athena and Tableau

You can build visualizations using Tableau and Amazon Athena for the views created by this solution. 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 it may contain more data than needed. Views contain only a subset of refined data from the tables, one view per metric. You can build visualizations using the views for respective metrics as needed.
  - `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 (p. 6) (`s3://YourS3MetricsBucket/GitHubEvents/`). It is the base table for `github_change_activity_view`.

For more information about the database schema, refer to Database schema information (p. 25).
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 (p. 25) and Build visualizations with Amazon Athena and Tableau (p. 42). 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.
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.
Collection of operational metrics

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

- **Solution ID**: The AWS solution identifier
- **Unique ID (UUID)**: Randomly generated, unique identifier for each deployment of the solution template
- **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 GitHub webhook and authorize request – Yes or No

AWS owns the data gathered through 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:

   ```json
   "AnonymousData" : {
   "SendAnonymousUsageData" : { "Data" : "Yes" } 
   },
   ```

   to:

   ```json
   "AnonymousData" : {
   "SendAnonymousUsageData" : { "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 (p. 11)** in the Automated deployment section of this 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 (CDK). Refer to the README.md file for additional information.
# Revisions

<table>
<thead>
<tr>
<th>Date</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>March 2021</td>
<td>Initial release</td>
</tr>
<tr>
<td>June 2021</td>
<td>Release version 1.1: Added support for AWS CodeBuild and AWS CodePipeline metrics. For additional information, refer to the CHANGELOG.md file.</td>
</tr>
<tr>
<td>April 2022</td>
<td>Release version 1.5: 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.</td>
</tr>
</tbody>
</table>
Contributors

- Aijun Peng
- Mike O’Brien
- Nikhil Reddy
- Aaron Schuetter
- Thiemo Belmega
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.
AWS glossary

For the latest AWS terminology, see the AWS glossary in the AWS General Reference.