JON BONSO AND KENNETH SAMONTE

AWS CERTIFIED

DEVOPS ENGINEER PROFESSIONAL

Tutorials Dojo
Study Guide and Cheat Sheets

3. Amazon CloudWatch Events
4. Amazon CloudWatch Alarms
5. AWS CodePipeline
6. AWS CodeDeploy
7. AWS CodeBuild
8. AWS CodeCommit
9. AWS Config
10. AWS Systems Manager
11. Amazon ECS
12. Amazon Elastic Beanstalk
13. AWS CloudTrail
14. AWS OpsWorks
15. AWS Trusted Advisor

The FAQs provide a good summary for each service, however, the AWS documentation contains more detailed information that you'll need to study. These details will be the deciding factor in determining the correct choice from the incorrect choices in your exam. To supplement your review of the services, we recommend that you take a look at Tutorials Dojo's AWS Cheat Sheets. Their contents are well-written and straight to the point, which will help reduce the time spent going through FAQs and documentations.

Common Exam Scenarios

Scenario	Solution
Software Development and Lifecycle (SDLC) Automation	
An Elastic Beanstalk application must not have any downtime during deployment and requires an easy rollback to the previous version if an issue occurs.	Set up Blue/Green deployment, deploy a new version on a separate environment then swap environment URLs on Elastic Beanstalk.
A new version of an AWS Lambda application is ready to be deployed and the deployment should not cause any downtime. A quick rollback to the previous Lambda version must be available.	Publish a new version of the Lambda function. After testing, use the production Lambda Alias to point to this new version.
In an AWS Lambda application deployment, only 10% of the incoming traffic should be routed to the new version to verify the changes before eventually allowing all production traffic.	Set up Canary deployment for AWS Lambda. Create a Lambda Alias pointed to the new Version. Set Weighted Alias value for this Alias as 10%.
An application hosted in Amazon EC2 instances behind an Application Load Balancer. You must	Launch the application in Amazon EC2 that runs the new version with an Application Load

provide a safe way to upgrade the version on Production and allow easy rollback to the previous version.	Balancer (ALB) in front. Use Route 53 to change the ALB A-record Alias to the new ALB URL. Rollback by changing the A-record Alias to the old ALB.
An AWS OpsWorks application needs to safely deploy its new version on the production environment. You are tasked to prepare a rollback process in case of unexpected behavior.	Clone the OpsWorks Stack. Test it with the new URL of the cloned environment. Update the Route 53 record to point to the new version.
A development team needs full access to AWS CodeCommit but they should not be able to create/delete repositories.	Assign the developers with the AWSCodeCommitPowerUser IAM policy
During the deployment, you need to run custom actions before deploying the new version of the application using AWS CodeDeploy.	Add lifecycle hook action BeforeAllowTraffic
You need to run custom verification actions after the new version is deployed using AWS CodeDeploy.	Add lifecycle hook action AfterAllowTraffic
You need to set up AWS CodeBuild to automatically run after a pull request has been successfully merged using AWS CodeCommit	Create CloudWatch Events rule to detect pull requests and action set to trigger CodeBuild Project. Use AWS Lambda to update the pull request with the result of the project Build
You need to use AWS CodeBuild to create artifact and automatically deploy the new application version	Set CodeBuild to save artifact to S3 bucket. Use CodePipeline to deploy using CodeDeploy and set the build artifact from the CodeBuild output.
You need to upload the AWS CodeBuild artifact to Amazon S3	S3 bucket needs to have versioning and encryption enabled.
You need to review AWS CodeBuild Logs and have an alarm notification for build results on Slack	Send AWS CodeBuild logs to CloudWatch Log group. Create CloudWatch Events rule to detect the result of your build and target a Lambda function to send results to the Slack channel (or SNS notification)
Need to get a Slack notification for the status of the application deployments on AWS CodeDeploy	Create CloudWatch Events rule to detect the result of CodeDeploy job and target a notification

	to AWS SNS or a Lambda function to send results to Slack channel
Need to run an AWS CodePipeline every day for updating the development progress status	Create CloudWatch Events rule to run on schedule every day and set a target to the AWS CodePipeline ARN
Automate deployment of a Lambda function and test for only 10% of traffic for 10 minutes before allowing 100% traffic flow.	Use CodeDeploy and select deployment configuration CodeDeployDefault.LambdaCanary10Percent10Minutes
Deployment of Elastic Beanstalk application with absolutely no downtime. The solution must maintain full compute capacity during deployment to avoid service degradation.	Choose the “Rolling with additional Batch” deployment policy in Elastic Beanstalk
Deployment of Elastic Beanstalk application where the new version must not be mixed with the current version.	Choose the “Immutable deployments” deployment policy in Elastic Beanstalk
Configuration Management and Infrastructure-as-Code	
The resources on the parent CloudFormation stack needs to be referenced by other nested CloudFormation stacks	Use Export on the Output field of the main CloudFormation stack and use Fn::ImportValue function to import the value on the other stacks
On which part of the CloudFormation template should you define the artifact zip file on the S3 bucket?	The artifact file is defined on the AWS::Lambda::Function code resource block
Need to define the AWS Lambda function inline in the CloudFormation template	On the AWS::Lambda::Function code resource block, the inline function must be enclosed inside the ZipFile section.
Use CloudFormation to update Auto Scaling Group and only terminate the old instances when the newly launched instances become fully operational	Set AutoScalingReplacingUpdate : WillReplace property to TRUE to have CloudFormation retain the old ASG until the instances on the new ASG are healthy.
You need to scale-down the EC2 instances at night when there is low traffic using OpsWorks.	Create *Time-based* instances for automatic scaling of predictable workload.

Can't install an agent on on-premises servers but need to collect information for migration	Deploy the Agentless Discovery Connector VM on your on-premises data center to collect information.
Syntax for CloudFormation with an Amazon ECS cluster with ALB	Use the AWS::ECS::Service element for the ECS Cluster, AWS::ECS::TaskDefinition element for the ECS Task Definitions and the AWS::ElasticLoadBalancingV2::LoadBalancer element for the ALB.
Monitoring and Logging	
Need to centralize audit and collect configuration setting on all regions of multiple accounts	Setup an Aggregator on AWS Config.
Consolidate CloudTrail log files from multiple AWS accounts	Create a central S3 bucket with bucket policy to grant cross-account permission. Set this as destination bucket on the CloudTrail of the other AWS accounts.
Ensure that CloudTrail logs on the S3 bucket are protected and cannot be tampered with.	Enable Log File Validation on CloudTrail settings
Need to collect/investigate application logs from EC2 or on-premises server	Install CloudWatch Logs Agent to send the logs to CloudWatch Logs for storage and viewing.
Need to review logs from running ECS Fargate tasks	Enable awslogs log driver on the Task Definition and add the required logConfiguration parameter.
Need to run real-time analysis for collected application logs	Send logs to CloudWatch Logs, create a Lambda subscription filter, Elasticsearch subscription filter, or Kinesis stream filter.
Need to be automatically notified if you are reaching the limit of running EC2 instances or limit of Auto Scaling Groups	Track service limits with Trusted Advisor on CloudWatch Alarms using the ServiceLimitUsage metric.
Policies and Standards Automation	
Need to secure the buildspec.yml file which contains the AWS keys and database password stored in plaintext.	Store these values as encrypted parameter on SSM Parameter Store

Using default IAM policies for AWSCodeCommitPowerUser but must be limited to a specific repository only	Attach additional policy with Deny rule and custom condition if it does not match the specific repository or branch
You need to secure an S3 bucket by ensuring that only HTTPS requests are allowed for compliance purposes.	Create an S3 bucket policy that Deny if checks for condition aws:SecureTransport is **false**
Need to store a secret, database password, or variable, in the most cost-effective solution	Store the variable on SSM Parameter Store and enable encryption
Need to generate a secret password and have it rotated automatically at regular intervals	Store the secret on AWS Secrets Manager and enable key rotation.
Several team members, with designated roles, need to be granted permission to use AWS resources	Assign AWS managed policies on the IAM accounts such as, ReadOnlyAccess, AdministratorAccess, PowerUserAccess
Apply latest patches on EC2 and automatically create an AMI	Use Systems Manager automation to execute an Automation Document that installs OS patches and creates a new AMI.
Need to have a secure SSH connection to EC2 instances and have a record of all commands executed during the session	Install SSM Agent on EC2 and use SSM Session Manager for the SSH access. Send the session logs to S3 bucket or CloudWatch Logs for auditing and review.
Ensure that the managed EC2 instances have the correct application version and patches installed.	Use SSM Inventory to have a visibility of your managed instances and identify their current configurations.
Apply custom patch baseline from a custom repository and schedule patches to managed instances	Use SSM Patch Manager to define a custom patch baseline and schedule the application patches using SSM Maintenance Windows
Incident and Event Response	
Need to get a notification if somebody deletes files in your S3 bucket	Setup Amazon S3 Event Notifications to get notifications based on specified S3 events on a particular bucket.
Need to be notified when an RDS Multi-AZ failover happens	Setup Amazon RDS Event Notifications to detect specific events on RDS.

Get a notification if somebody uploaded IAM access keys on any public GitHub repositories	Create a CloudWatch Events rule for the AWS_RISK_CREDENTIALS_EXPOSED event from AWS Health Service. Use AWS Step Functions to automatically delete the IAM key.
Get notified on Slack when your EC2 instance is having an AWS-initiated maintenance event	Create a CloudWatch Events rule for the AWS Health Service to detect EC2 Events. Target a Lambda function that will send a notification to the Slack channel
Get notified of any AWS maintenance or events that may impact your EC2 or RDS instances	Create a CloudWatch Events rule for detecting any events on AWS Health Service and send a message to an SNS topic or invoke a Lambda function.
Monitor scaling events of your Amazon EC2 Auto Scaling Group such as launching or terminating an EC2 instance.	Use Amazon EventBridge or CloudWatch Events for monitoring the Auto Scaling Service and monitor the EC2 Instance-Launch Successful and EC2 Instance-Terminate Successful events.
View object-level actions of S3 buckets such as upload or deletion of object in CloudTrail	Set up Data events on your CloudTrail trail to record object-level API activity on your S3 buckets.
Execute a custom action if a specific CodePipeline stage has a FAILED status	Create CloudWatch Event rule to detect failed state on the CodePipeline service, and set a target to SNS topic for notification or invoke a Lambda function to perform custom action.
Automatically rollback a deployment in AWS CodeDeploy when the number of healthy instances is lower than the minimum requirement.	On CodeDeploy, create a deployment alarm that is integrated with Amazon CloudWatch. Track the MinimumHealthyHosts metric for the threshold of EC2 instances and trigger the rollback if the alarm is breached.
Need to complete QA testing before deploying a new version to the production environment	Add a Manual approval step on AWS CodePipeline, and instruct the QA team to approve the step before the pipeline can resume the deployment.
Get notified for OpsWorks auto-healing events	Create a CloudWatch Events rule for the OpsWorks Service to track the auto-healing events

Automatically Run CodeBuild Tests After a Developer Creates a CodeCommit Pull Request

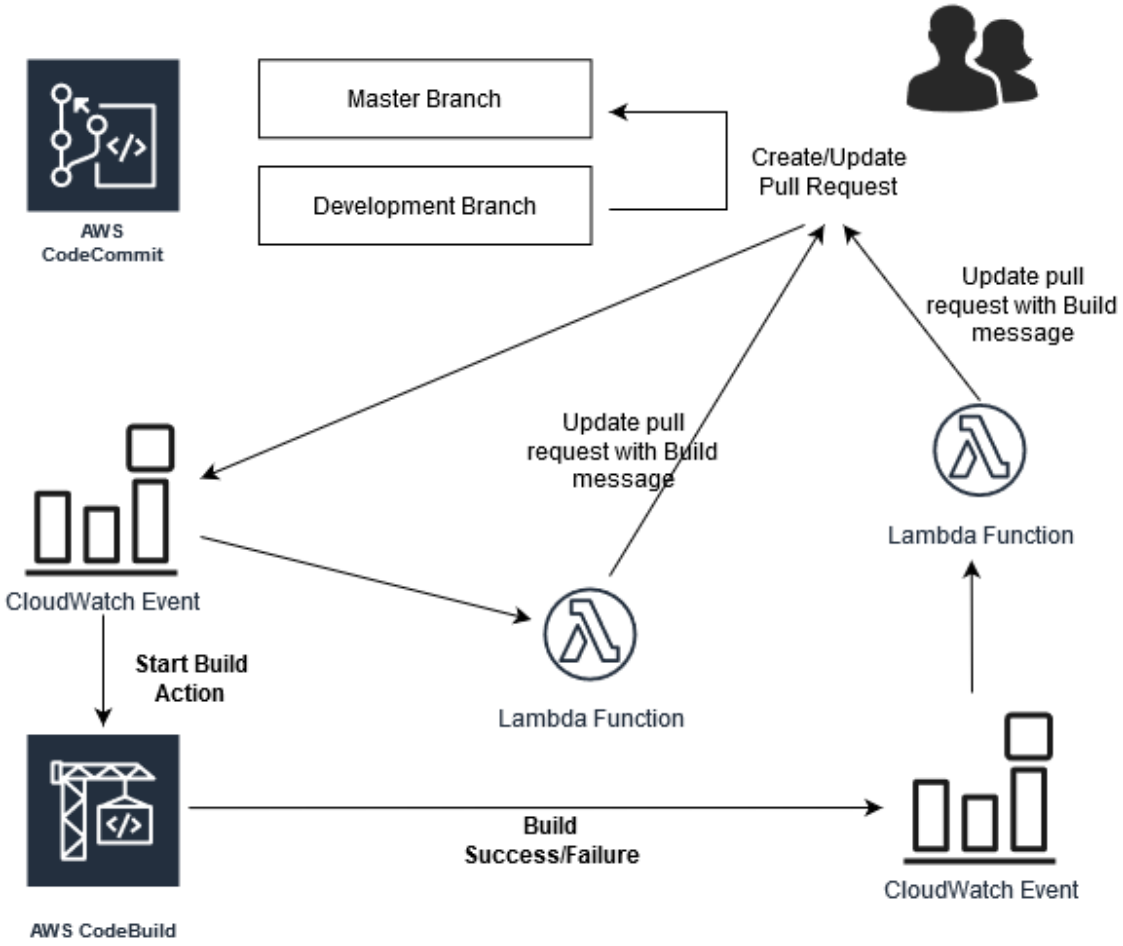
AWS CodeCommit allows developers to create multiple branches, which can be used for developing new features or fixing application bugs. These changes can then be merged on the master branch, which is usually used for the production release. To merge the changes to the master branch on CodeCommit, the developers create a pull request. But these code changes need to be validated in order to make sure that the changes integrate properly with the current code base.

To validate your changes on the code base, you can run an AWS CodeBuild project to the build and test your pull request and based on the result, decide on whether to accept the merging of the code or reject the pull request.

DevOps Exam Notes:

You can automate the validation of AWS CodeCommit pull requests with AWS CodeBuild and AWS Lambda with the help of CloudWatch Events. Basically, CloudWatch Events will detect the pull requests on your CodeCommit repository and then trigger the AWS CodeBuild project with the Lambda function updating the comments on the pull request. The results of the build are also detected by AWS CloudWatch Events and will trigger the Lambda function to update the pull request with the results.

The following diagram shows an example workflow on how you can automate the validation of a pull request with AWS CodeCommit, AWS CodeBuild, CloudWatch Event, and AWS Lambda.



1. The AWS CodeCommit repository contains two branches, the master branch that contains approved code, and the development branch, where changes to the code are developed.

2. Push the new code on the AWS CodeCommit Development branch.

3. Create a pull request to merge their changes to the master branch.

4. Create a CloudWatch Events rule to detect the pull request and have it trigger an AWS Lambda function that will post an automated comment to the pull request that indicates that a build to test the changes is about to begin.

5. With the same CloudWatch Events rule, have it trigger an AWS CodeBuild project that will build and validate the changes.

6. Create another CloudWatch Events rule to detect the output of the build. Have it trigger another Lambda function that posts an automated comment to the pull request with the results of the build and a link to the build logs.

Based on this automated testing, the developer who opened the pull request can update the code to address any build failures, and then update the pull request with those changes. The validation workflow will run again and will produce the updated results.

Once the pull request is successfully validated, you can accept the pull request to merge the changes to the master branch.

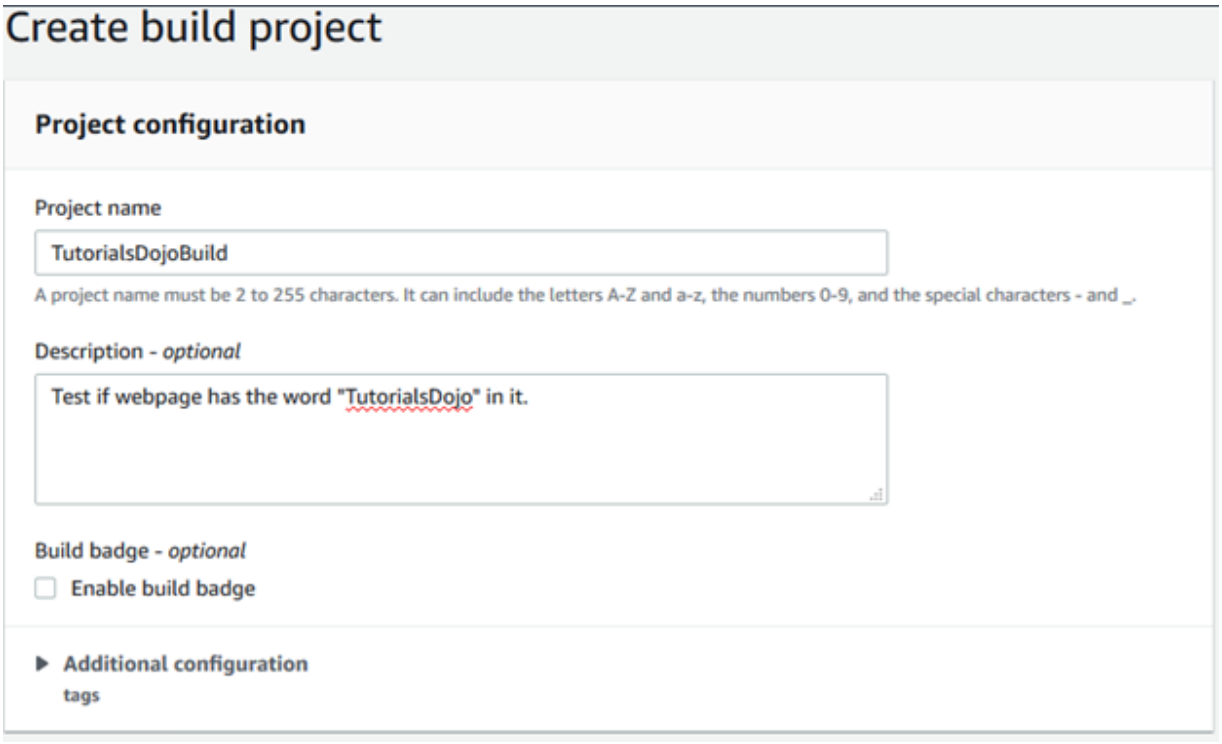
Sources:
https://docs.aws.amazon.com/codebuild/latest/userguide/how-to-create-pipeline.html
https://aws.amazon.com/blogs/devops/validating-aws-codecommit-pull-requests-with-aws-codebuild-and-aws-lambda/

CodeBuild with CloudWatch Logs, Metrics, and Alarms

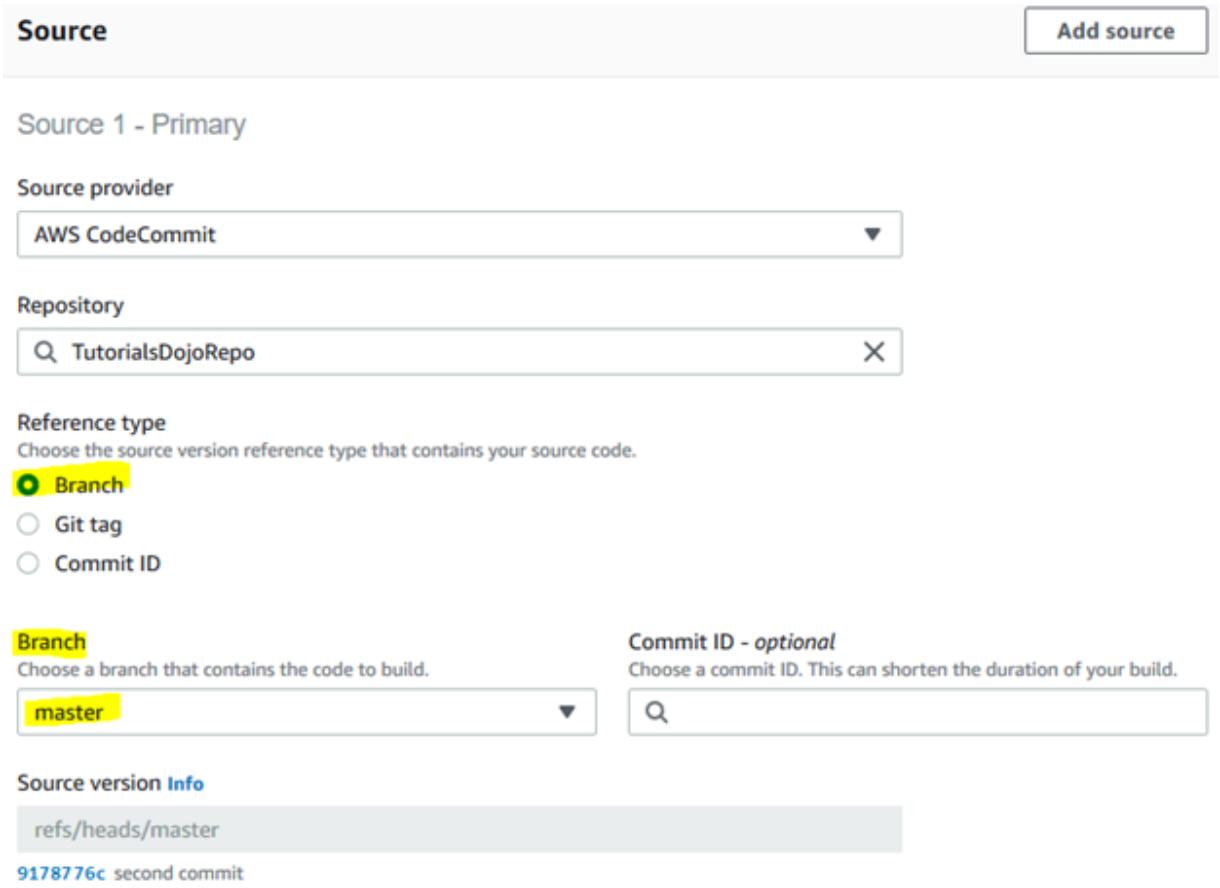
AWS Codebuild allows you to compile, build, run tests, and produce an artifact of your code. This can be integrated into your CI/CD process. For example, you are using AWS CodeCommit as version control for your source code repository. After a developer pushes new code to the Development branch, you can have an automatic trigger for CodeBuild to run your project build, test your application, and then upload the output artifact to S3. The artifact file on S3 can then be used by deployment tools such as AWS CodeDeploy to have it deployed on your EC2 instances, ECS, or Lambda functions.

Here are the steps on how to create a CodeBuild build project and save the artifact on an S3 bucket. We will also demonstrate how CodeBuild integrates with AWS CloudWatch.

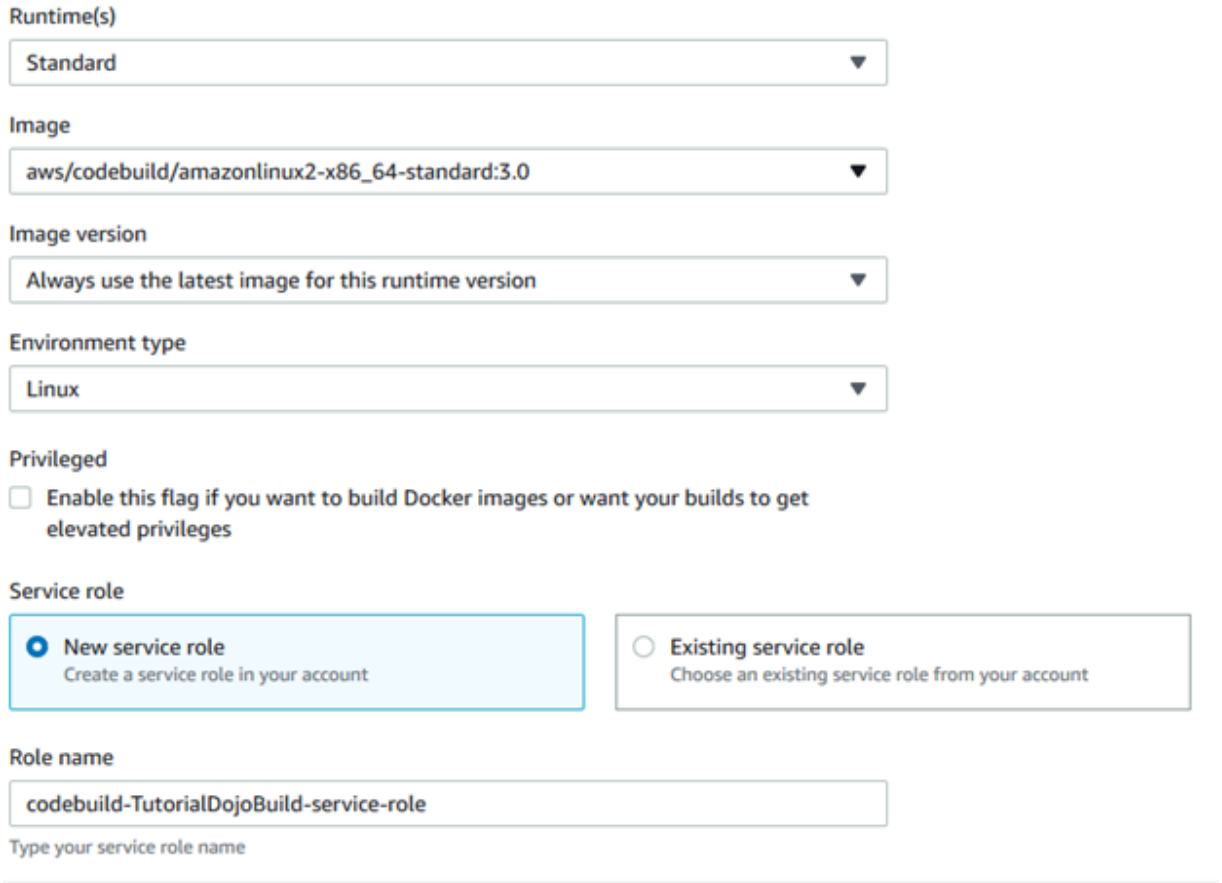
1. Go to AWS Code Build > Build Projects and click Create Build Project. Input your details for this project.



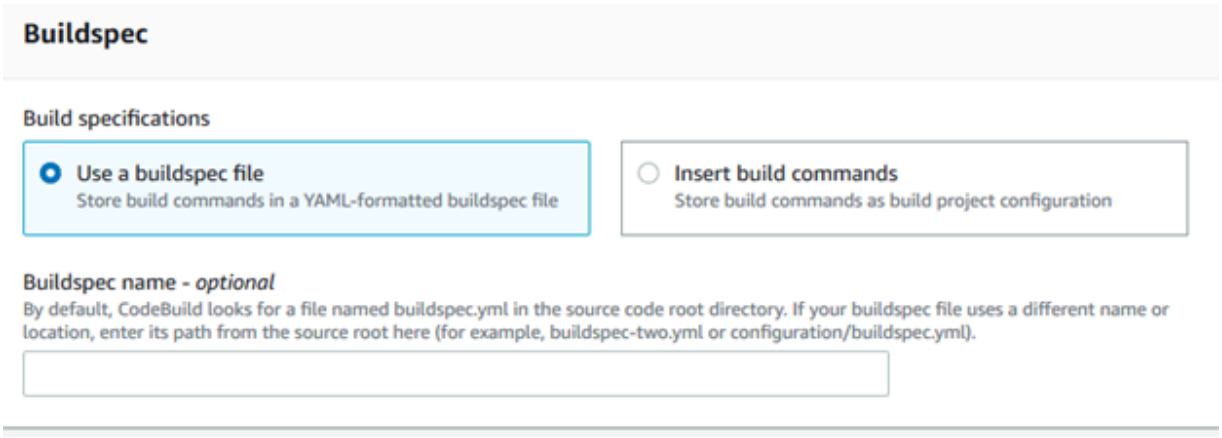
2. CodeBuild supports several sources for your application code, including Amazon S3, AWS CodeCommit, GitHub, Bitbucket, etc. Using CodeBuild, select your source repository and branch.



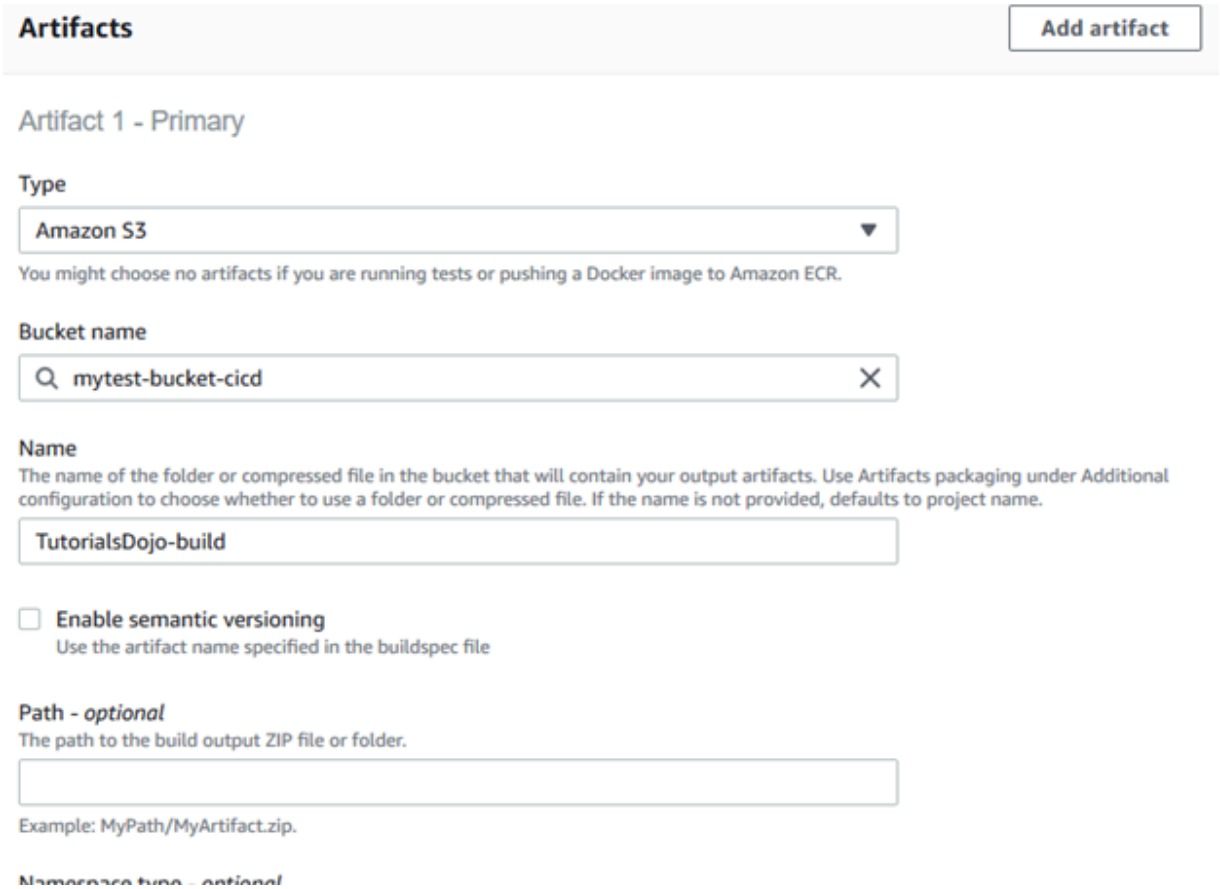
3. Use the Amazon Linux runtime since we'll build this for the Amazon Linux AMI.



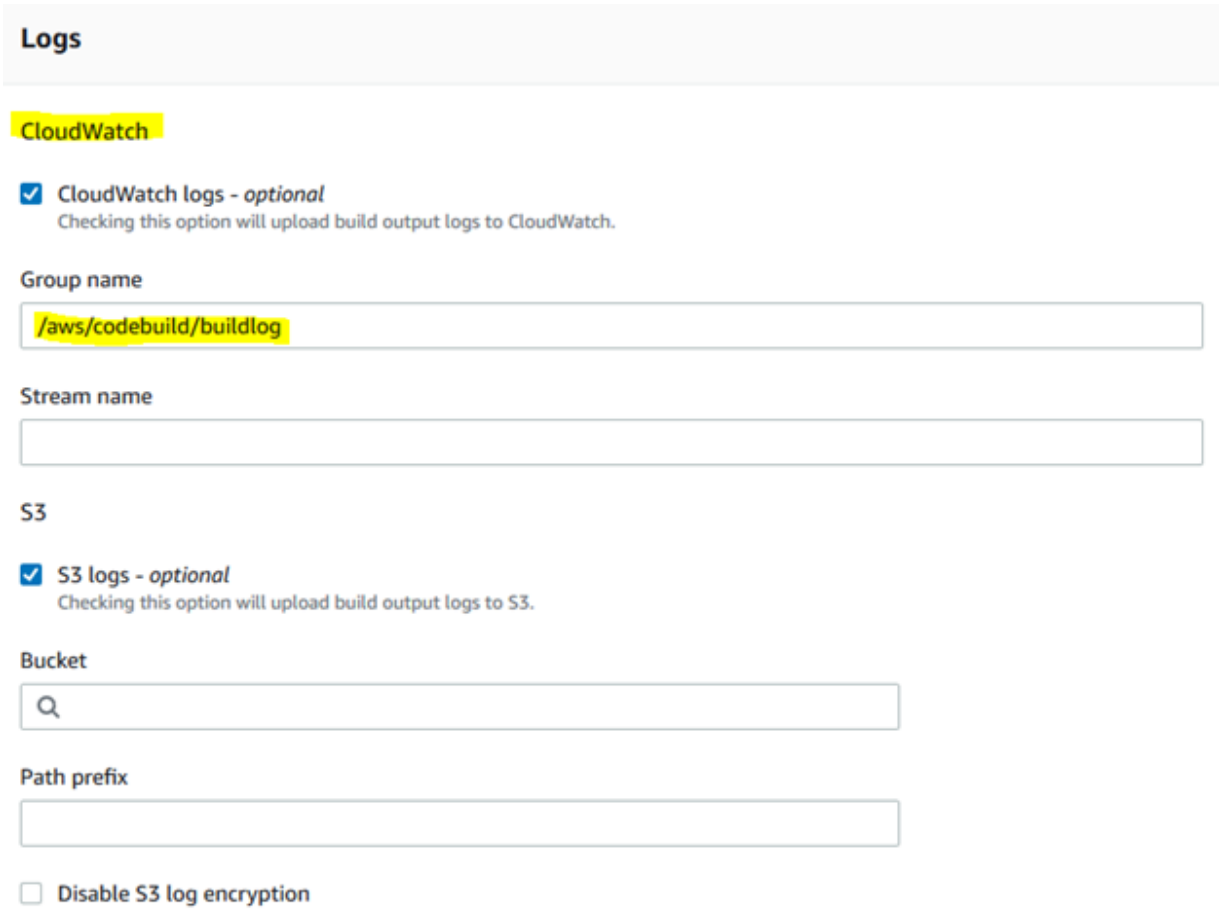
4. By default, the build specification filename is “buildspec.yml”. This should be in the root folder of your application code.



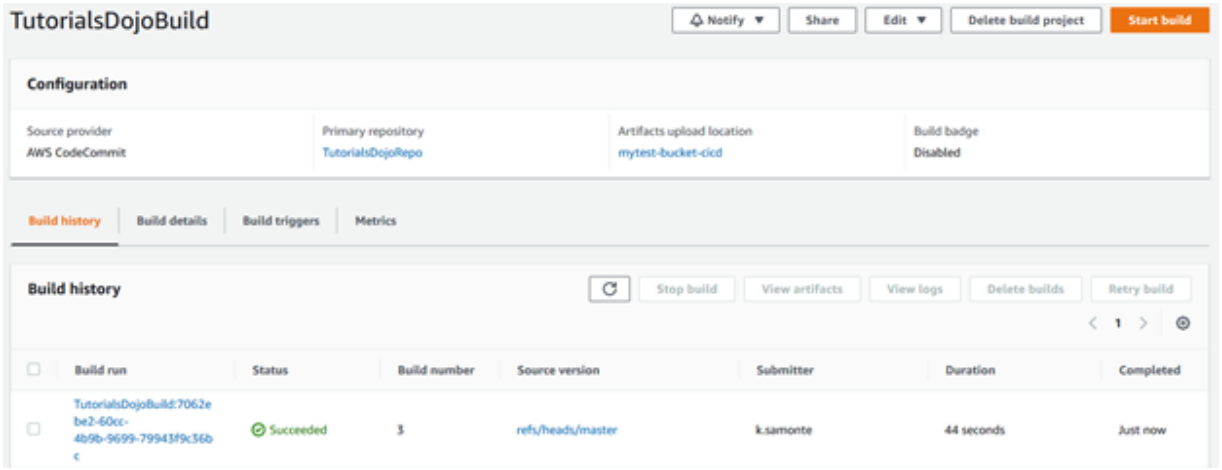
5. Specify which S3 bucket you are going to send the output artifact of your build.



6. On the logs part, you can have the option to send the build logs to CloudWatch Log group or send to an S3 bucket. The AWS CloudWatch log group or S3 bucket must exist first before you specify it here.

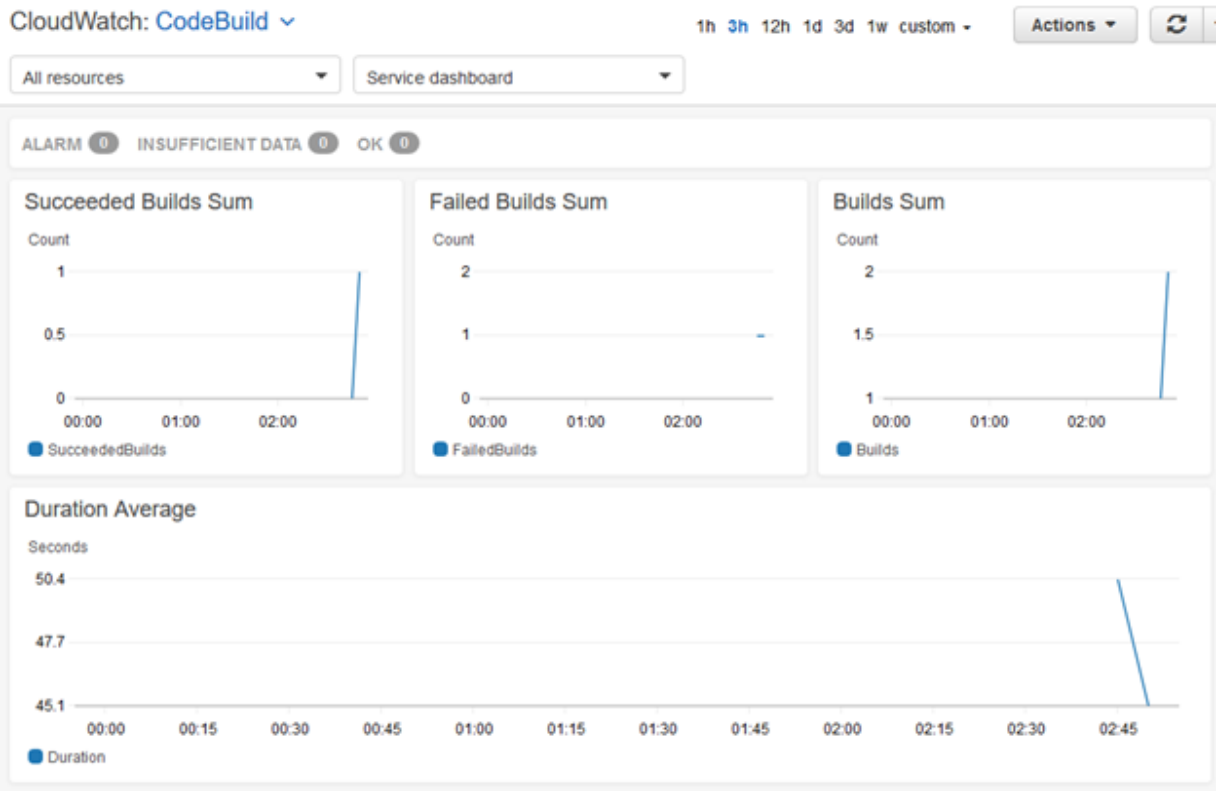


7. Click “Create build project”. Select your project and click the “Start build” button.

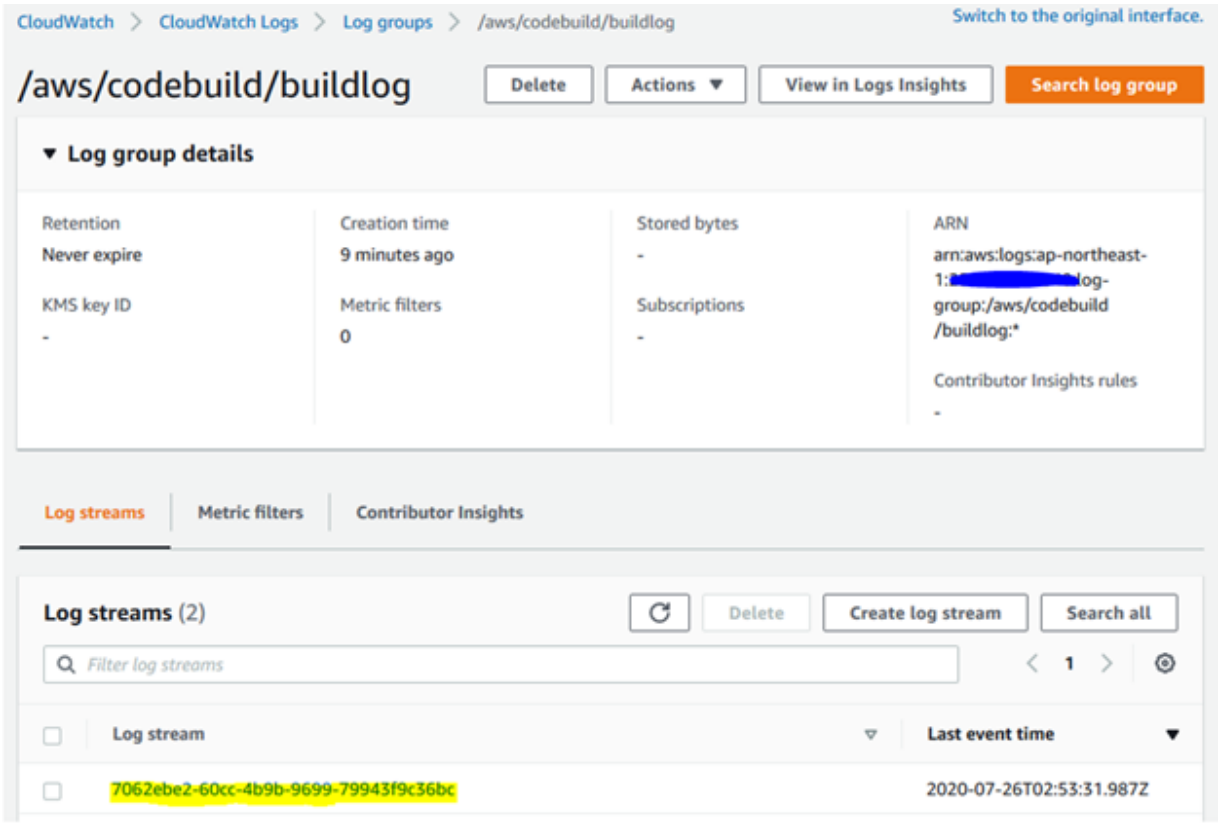


8. After the build, you should see the artifact on the S3 bucket. You will also see the build logs of your project if you click on the Build run ID of your project.

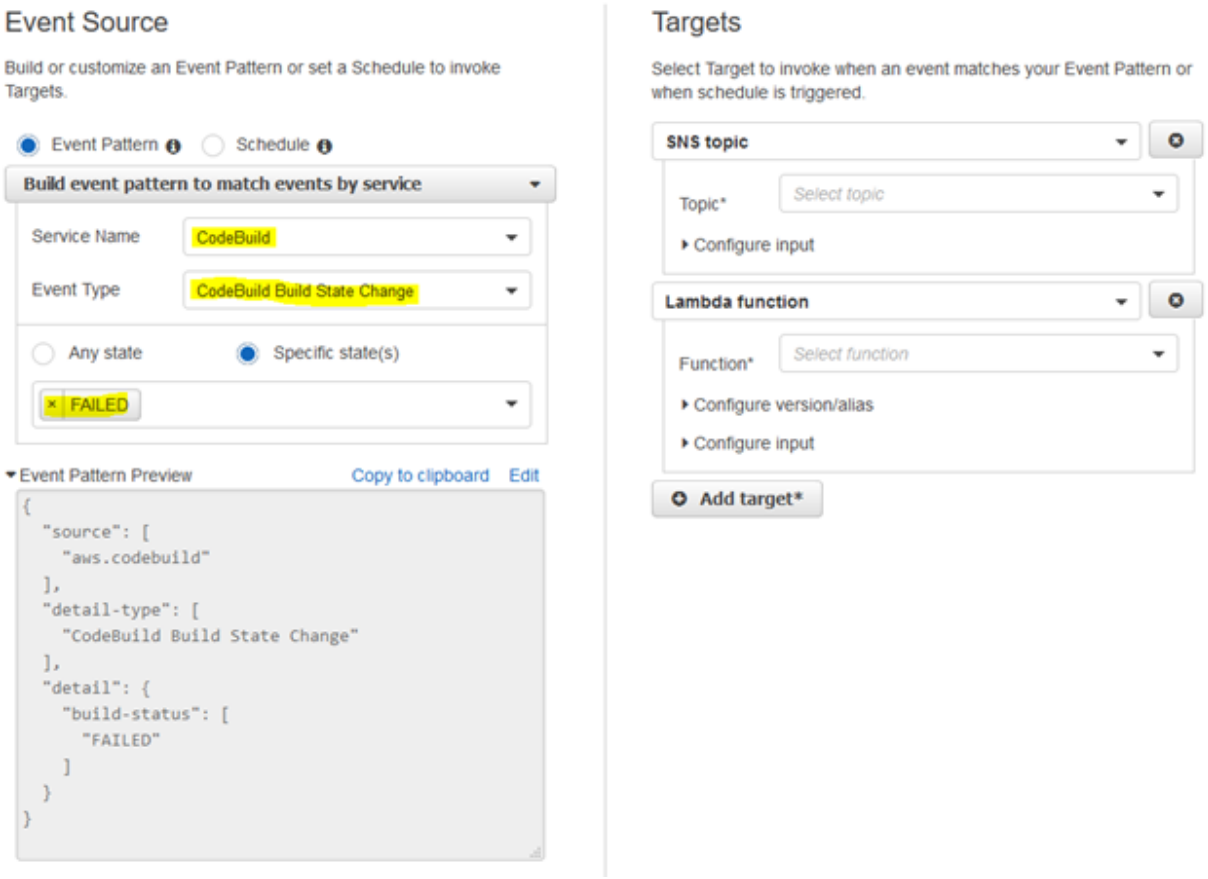
9. AWS CodeBuild is also integrated on CloudWatch Logs. When you go to CloudWatch Dashboard, you can select the pre-defined CodeBuild Dashboard where you will see the metrics such as Successful builds, Failed builds, etc.



10. You should also be able to see the AWS CloudWatch Log Group you created and that CodeBuild logs are delivered to it.



11. Using CloudWatch Event rules, you can also create a rule to detect Successful or Failed builds, and then use the Trigger to invoke a Lambda function to send you a slack notification, or use an SNS Topic target to send you an email notification about the build status.



DevOps Exam Notes:

AWS CodeBuild is integrated with AWS CloudWatch. A predefined dashboard is created on CloudWatch to view metrics regarding project builds on your account. You can send CodeBuild build logs to a CloudWatch log group so you have a central location to review them. You can set filters and alarms on these logs and set up a notification. You can also use CloudWatch Events to detect changes on your build project, such as FAILED builds and have it invoke a Lambda function to send you a message on your Slack channel, or SNS topic to send you an email notification.

Sources:
https://docs.aws.amazon.com/codebuild/latest/userguide/create-project.html
https://docs.aws.amazon.com/codebuild/latest/userguide/create-project-console.html

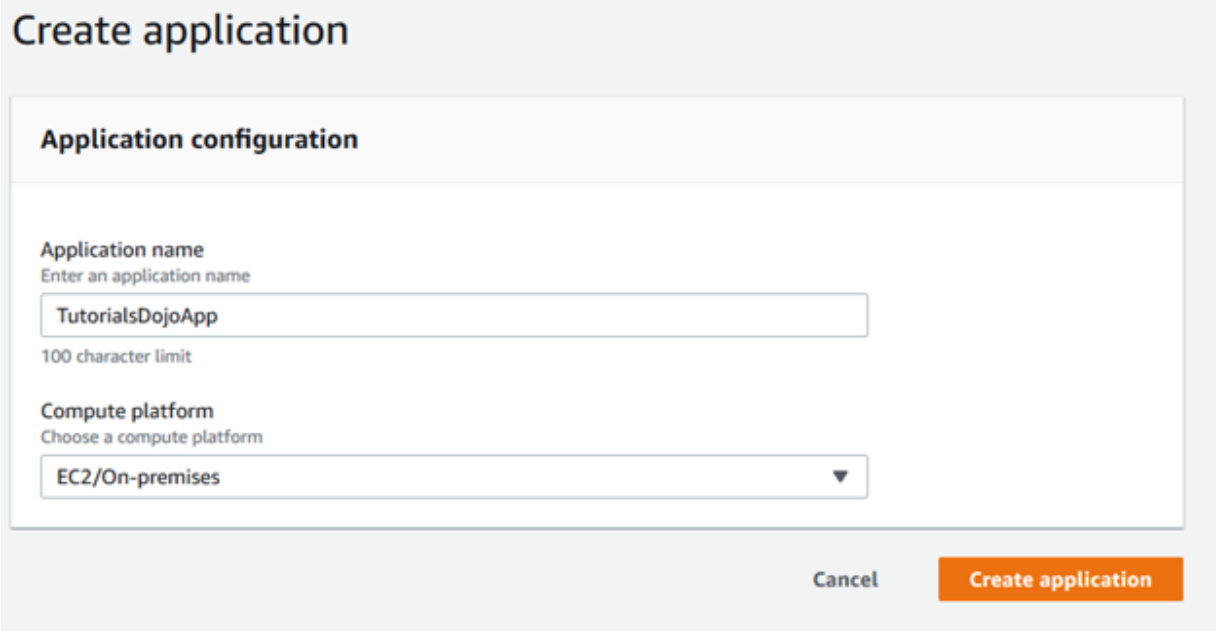
CodeDeploy with CloudWatch Logs, Metrics, and Alarms

AWS CodeDeploy allows you to automate software deployments to a variety of services such as EC2, AWS Fargate, AWS Lambda, and your on-premises servers.

For example, after you have produced from AWS CodeBuild, you can have CodeDeploy fetch the artifact from the AWS S3 bucket and then deploy it to your AWS instances. Please note that the target instances need to have the AWS CodeDeploy agent installed on them for this to be successful and have proper Tags.

Here are the steps to create a deployment on AWS CodeDeploy, including a discussion on how it integrates with AWS CloudWatch.

1. Go to AWS CodeDeploy > Applications and click “Create Application”. Input details of your application and which platform it is running.

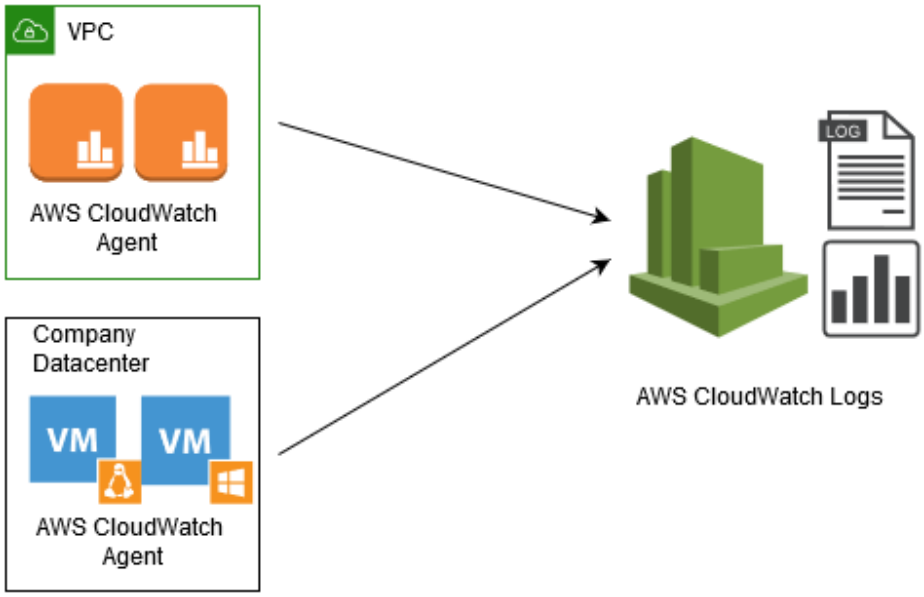


2. Select your application and create a Deployment Group. CodeDeploy needs to have an IAM permission to access your targets as well as read the AWS S3 bucket containing the artifact to be deployed.

Fetching Application Logs from Amazon EC2, ECS and On-premises Servers

Application logs are vital for monitoring, troubleshooting, and regulatory compliance of every enterprise system. Without it, your team will waste a lot of time trying to find the root cause of an issue that can be easily detected by simply checking the logs. These files often live inside the server or a container. Usually, you have to connect to the application server via SSH or RDP before you can view the logs. This manual process seems to be inefficient, especially for high-performance organizations with hybrid network architectures.

Using the Amazon CloudWatch Logs agent, you can collect system metrics and logs from your Amazon EC2 instances and on-premises application servers. Gone are the days of spending several minutes connecting to your server and manually retrieving the application logs. For Linux servers, you don't need to issue a *tail - f* command anymore since you can view the logs on the CloudWatch dashboard on your browser in near real-time. It also collects both system-level and custom metrics from your EC2 instances and on-premises servers, making your monitoring tasks a breeze.



You have to manually download and install the Amazon CloudWatch Logs agent to your EC2 instances or on-premises servers using the command line. Alternatively, you can use AWS Systems Manager to automate the installation process. For your EC2 instances, it is preferable to attach an IAM Role to allow the application to send data to CloudWatch. For your on-premises servers, you have to create a separate IAM User to integrate your server to CloudWatch. Of course, you should first establish a connection between your on-premises data center and VPC using a VPN or a Direct Connect connection. You have to use a named profile in your local server that contains the credentials of the IAM user that you created.

If you are running your containerized application in Amazon Elastic Container Service (ECS), you can view the different logs from your containers in one convenient location by integrating Amazon CloudWatch. You can configure your Docker containers' tasks to send log data to CloudWatch Logs by using the *awslogs* log driver.


```
"logConfiguration": {
    "logDriver": "awslogs",
    "options": {
        "awslogs-group": "awslogs-wordpress",
        "awslogs-region": "us-west-2",
        "awslogs-stream-prefix": "awslogs-example"
    }
```

If your ECS task is using a Fargate launch type, you can enable the *awslogs* log driver and add the required *logConfiguration* parameters to your task definition. For EC2 launch types, you have to ensure that your Amazon ECS container instances have an attached IAM role that contains *logs:CreateLogStream* and *logs:PutLogEvents* permissions. Storing the log files to Amazon CloudWatch prevents your application logs from taking up disk space on your container instances.

Sources:
https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/UseCloudWatchUnifiedAgent.html
https://docs.aws.amazon.com/AmazonECS/latest/userguide/using_awslogs.html

CloudWatch Logs Agent to CloudWatch Logs Subscription

You can install CloudWatch Agent on your on-premises instances or EC2 instances to allow them to send detailed metrics to CloudWatch or send application logs to CloudWatch Logs. The logs will be sent to your configured CloudWatch log group for viewing and searching.

DevOps Exam Notes:

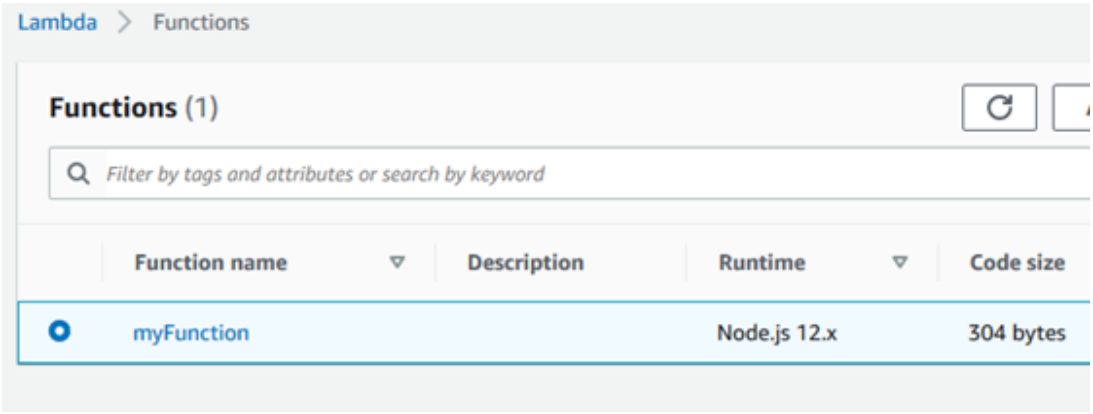
Additionally, you can use CloudWatch Logs subscriptions to get access to a real-time feed of log events from CloudWatch Logs and have it delivered to other services such as an Amazon Kinesis stream, an Amazon Kinesis Data Firehose stream, or AWS Lambda for custom processing, analysis, or loading to other systems. This way, you can perform near real-time analysis of the logs, further log processing using AWS Lambda, or have advanced searching capability using Elasticsearch.

To begin subscribing to log events, create the receiving source such as a Kinesis stream or Lambda function where the events will be delivered.

A subscription filter defines the filter pattern to use to sort out which log events get delivered to your AWS resource, as well as information about where to send matching log events to.

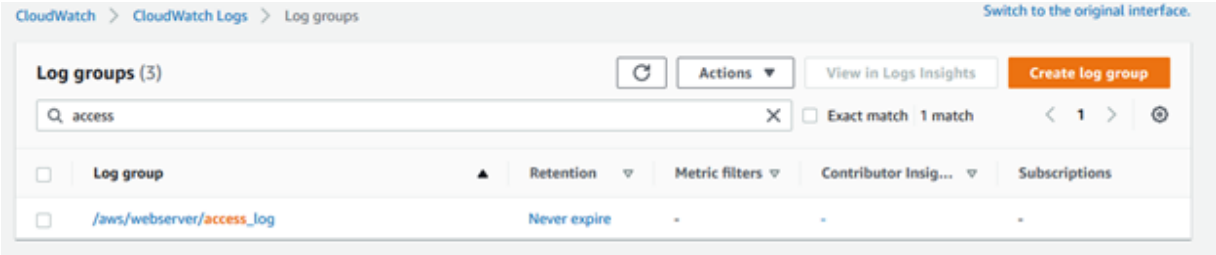
Here are the steps to setup CloudWatch logs subscription:

1. Create a receiving source, such as a Kinesis stream, Elasticsearch cluster, or Lambda function.

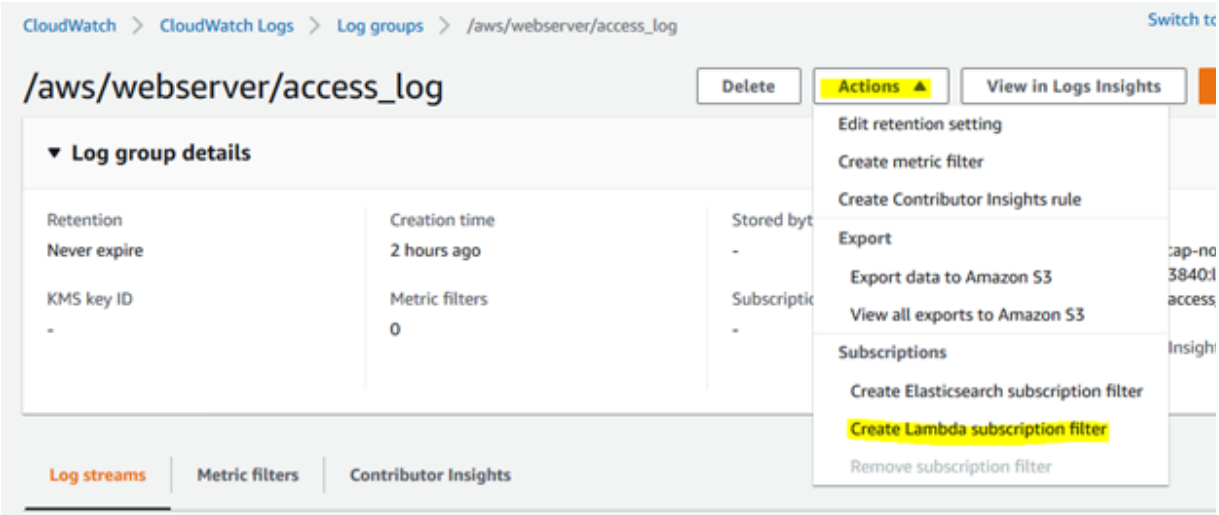


2. Install CloudWatch Unified Agent on the EC2 instance (Linux or Windows) and configure it to send application logs to CloudWatch log group.

3. Create the CloudWatch log group that will access the logs.



4. Create a subscription filter for the log group and select the Lambda function or Elasticsearch cluster that you created. If you need to set a Kinesis Data Firehose stream as the subscription filter, you will need to use AWS CLI as the web console does not support it yet.



Sources:
https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/QuickStartEC2Instance.html
https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/Subscriptions.html
https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/SubscriptionFilters.html

Monitoring Service Limits with Trusted Advisor

AWS Trusted Advisor provides you real-time guidance to help you provision your resources following AWS best practices. It provides recommendations on 5 categories – Cost Optimization, Performance, Security, Fault Tolerance and Service Limits.

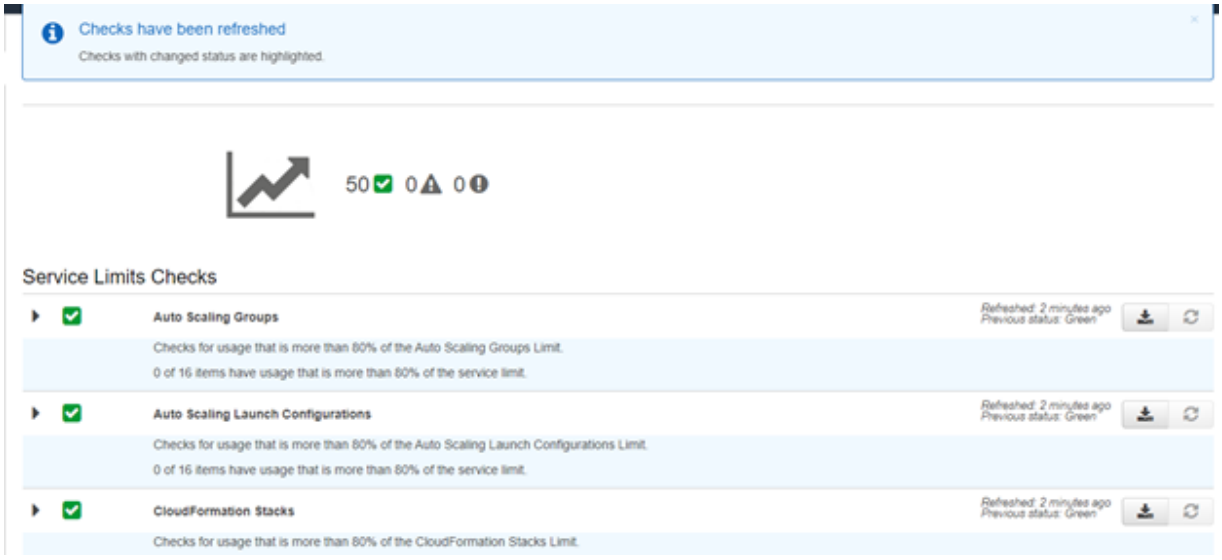
With Trusted Advisor's Service Limit Dashboard, you can view, refresh, and export utilization and limit data on a per-limit basis. These metrics are published on AWS CloudWatch in which you can create custom alarms based on percentage of service utilization against limits, understand the number of resources by each check, or view time-aggregate views of check results across service categories.

DevOps Exam Notes:

You need to understand that service limits are important when managing your AWS resources. In the exam you can be given a scenario in which you have several Auto Scaling groups in your AWS account and you need to make sure that you are not reaching the service limit when you perform your blue/green deployments for your application. You can track service limits with Trusted Advisor and CloudWatch Alarms. The ServiceLimitUsage metric on CloudWatch Alarms is only visible for Business and Enterprise support customers.

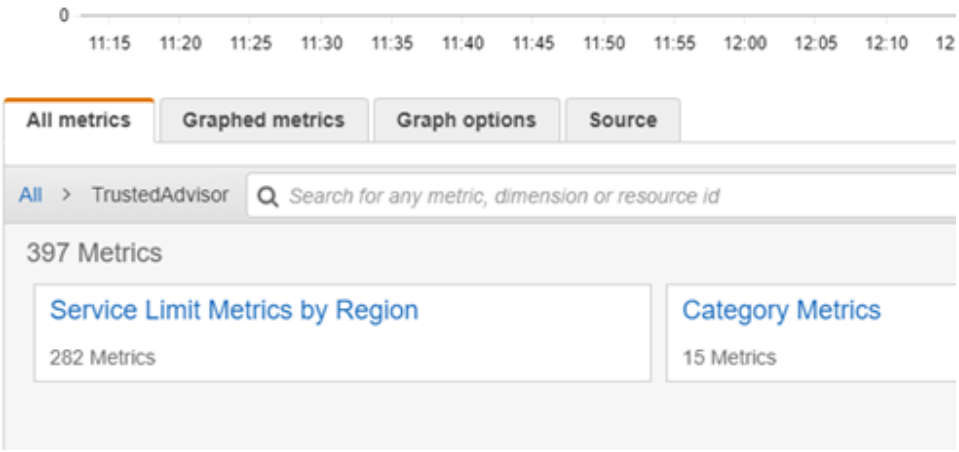
Here's how you can create a CloudWatch Alarm to detect if you are nearing your auto scaling service limit and send a notification so you can request for a service limit increase to AWS support.

1. First, head over to AWS Trusted Advisor > Service Limits and click the refresh button. This will refresh the service limit status for your account.

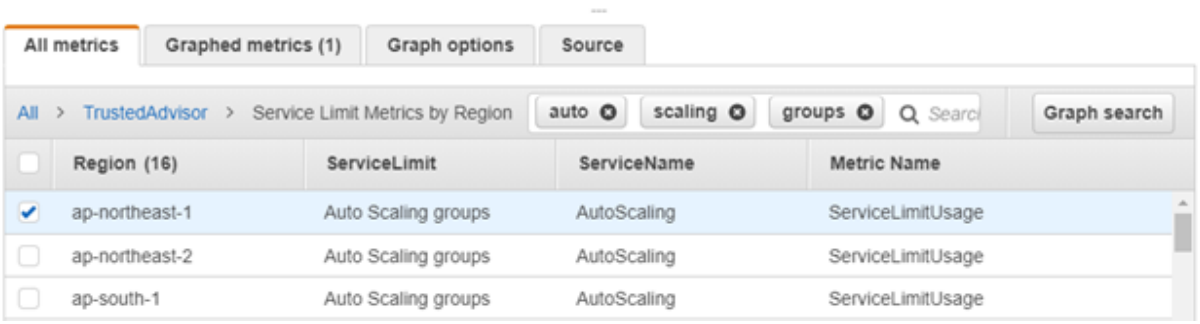


2. Go to CloudWatch > Alarms. Make sure that you are in the North Virginia region. Click the “Create Alarm” button and click “Select Metric”.

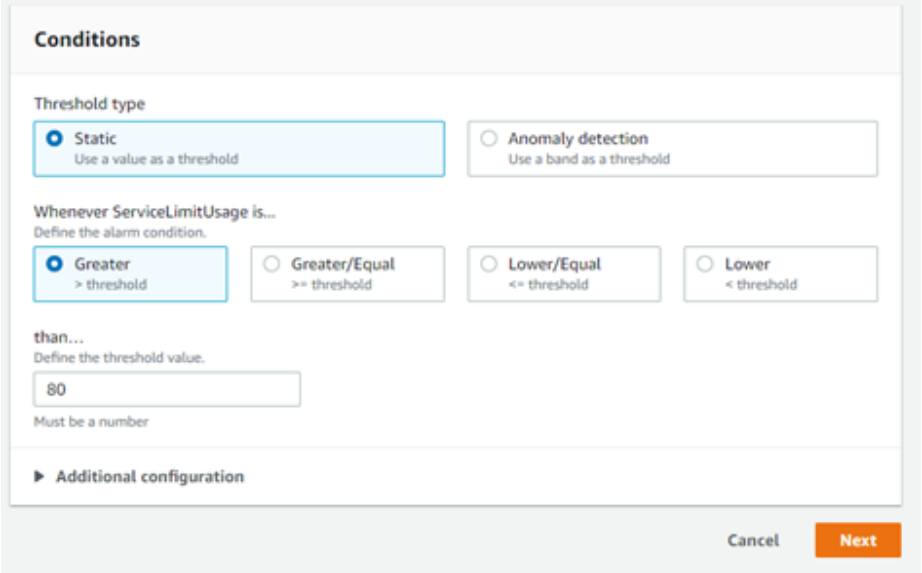
3. In the “All metrics” tab, click “Trusted Advisor” category and you will see “Service Limits by Region”.



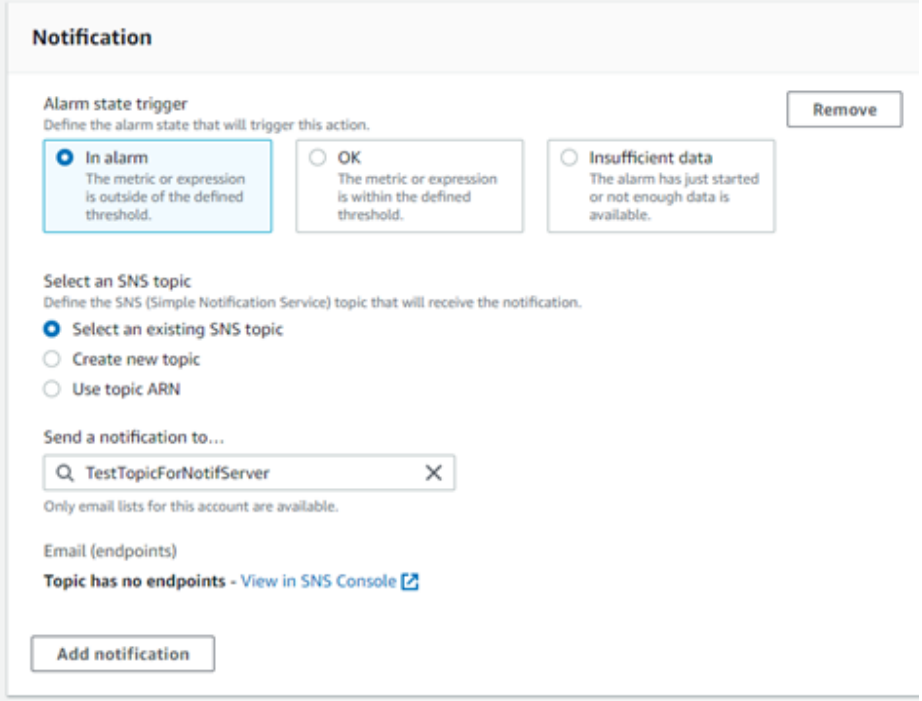
4. Search for Auto Scaling groups on your desired region and click Select Metric.



5. We can set the condition for this Alarm so that when your Auto Scaling group reaches 80 for that particular region, the alarm is triggered.



6. You can then configure an SNS topic to receive a notification for this alarm.



7. Click Next to Preview the alarm and click “Create Alarm” to create the alarm.

Sources:
https://aws.amazon.com/about-aws/whats-new/2017/11/aws-trusted-advisor-adds-service-limit-dashboard-and-cloudwatch-metrics/
https://aws.amazon.com/blogs/mt/monitoring-service-limits-with-trusted-advisor-and-amazon-cloudwatch/

Incident and Event Response Management on AWS

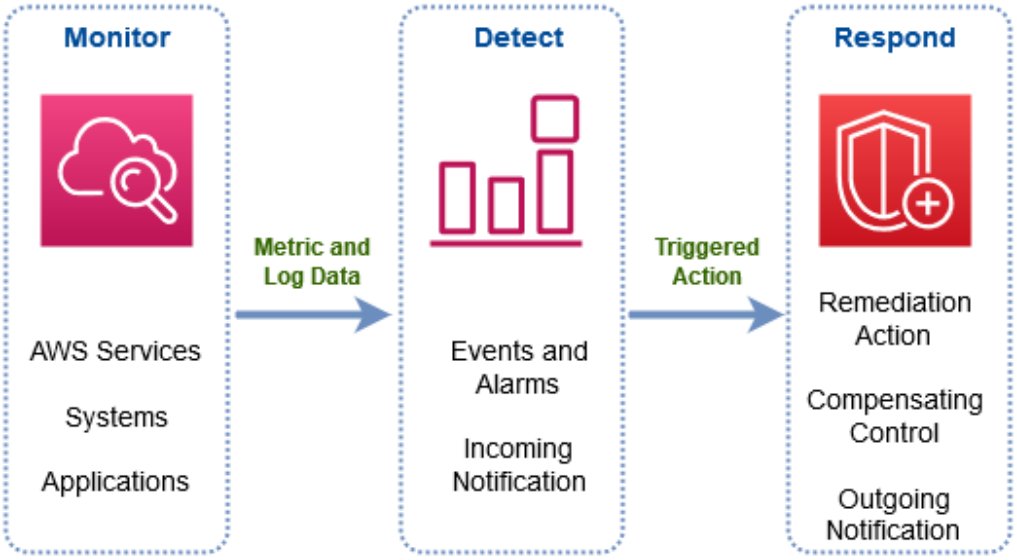
There is always a lot of action in a typical IT environment. The development team builds the new features within a given sprint and rectifies any software defects along the way. The operations team continuously monitors the underlying resources used by the enterprise applications and troubleshoots technical problems throughout the day. Critical incidents might suddenly happen that could adversely affect your production environments and many other system events that could potentially bring your business operations into a screeching halt! Thus, automating the incident and event management of your infrastructure is of utmost importance to maintain maximum efficiency and to reduce unnecessary interruptions.

AWS provides a myriad of services and features to automate manual and repetitive IT tasks. Gone are the days of receiving outraged emails, calls, or tweets from your customers because your production server was down and you're not even aware of it. By using Amazon CloudWatch Events and CloudWatch Alarms, your teams can immediately be notified of any system events or breach of a specified threshold. You can also enable the Auto Healing feature in your AWS OpsWorks Stack or place your EC2 instances in an Auto Scaling group to automatically replace failed instances without manual intervention. Deployment issues can quickly be resolved or prevented through deployment monitoring and automatic rollbacks using AWS CodeDeploy and CloudWatch Events. S3 Events enables you to monitor unauthorized actions in your S3 buckets continuously, and RDS Events keeps you in the know for any failover, configuration change, or backup-related events that affect your database tier. Amazon EventBridge can also track all the changes in your AWS services, your custom applications, and external Software-as-a-Service (SaaS) applications in real-time. These AWS features and services complement your existing security information and event management (SIEM) solutions to manage your entire cloud infrastructure properly.

DevOps Exam Notes:

An event indicates a change in a resource that is routed by an ‘event bus’ to its associated rule. A custom event bus can receive rules from AWS services, custom applications, and SaaS partner services. Both Amazon CloudWatch Events and Amazon EventBridge have the same API and underlying service, but the latter provides more features. Amazon EventBridge is the ideal service to manage your events, but take note that CloudWatch Events is still available and not entirely deprecated.

Automated Remediation Flow



The process of auditing your applications, systems, and infrastructure services in AWS is also simplified as all events and activities are appropriately tracked. Within minutes, you can identify the root cause of a recent production incident by checking the event history in AWS CloudTrail. Real-time logs feed on CloudWatch can be delivered to an Amazon Kinesis stream, an Amazon Kinesis Data Firehose stream, or AWS Lambda for processing, analysis, or integration to other systems through CloudWatch Subscription Filters. Security incidents can be remediated immediately by setting up custom responses to Amazon GuardDuty findings using Amazon CloudWatch Events. In this way, any security vulnerability in your AWS resources, such as an SSH brute force attack on one of your EC2 instances, can immediately be identified.

AWS-Scheduled Maintenance Notification to Slack Channel via CloudWatch Events

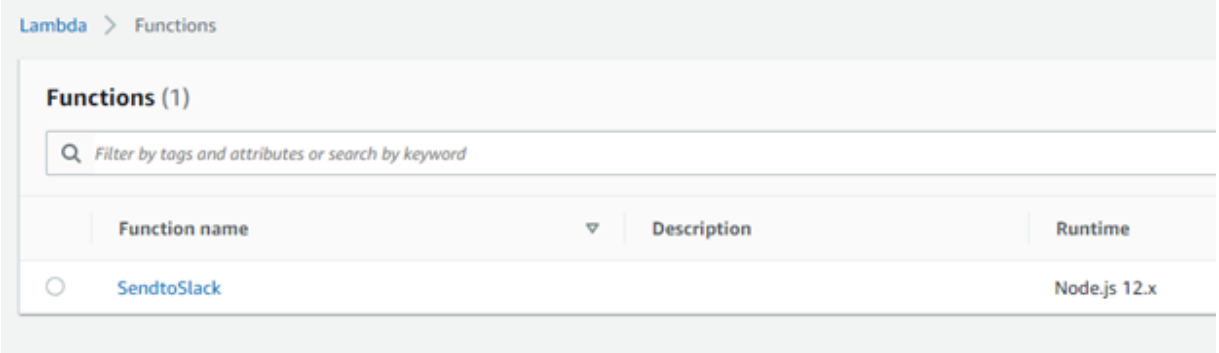
AWS Scheduled maintenance events are listed on AWS Health Dashboard. For example, if AWS needs to perform maintenance on the underlying host of your EC2 instance in which the EC2 instance usually needs to shut down, you will see an event on your AWS Health Dashboard for it. You can use CloudWatch Events to detect these changes and you trigger notifications so you will be notified for these events. You can then perform the needed actions based on the events

You can choose the following types of targets when using CloudWatch Events as a part of your AWS Health workflow:

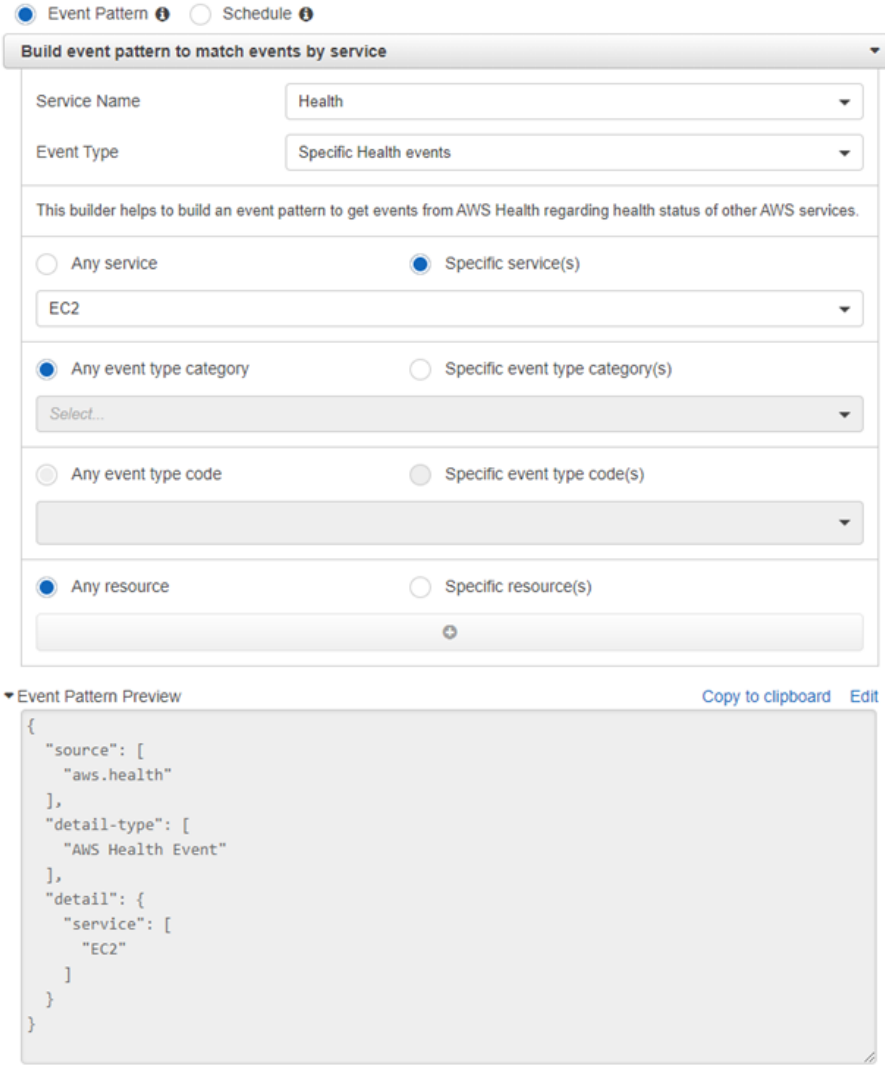
- AWS Lambda functions
- Amazon Kinesis Data Streams
- Amazon Simple Queue Service (Amazon SQS) queues
- Built-in targets (CloudWatch alarm actions)
- Amazon Simple Notification Service (Amazon SNS) topics

For example, you can use a Lambda function to pass a notification to a Slack channel when an AWS Health event occurs. Here are the steps to do this.

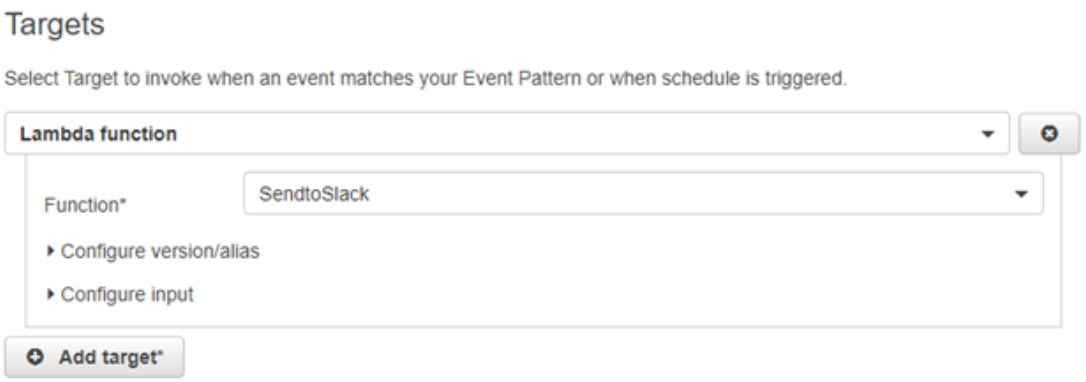
1. Create a Lambda function that will send a message to your Slack Channel. A Nodejs or Python script will suffice. The function will call an API URL for the Slack channel passing along the message.



2. Go to AWS CloudWatch Events and create a rule.
3. Set a rule for the AWS Health Service, and EC2 Service Event Type.



4. Add a Target for this event to run the Lambda function and save the CloudWatch Events Rule.



Sources:
https://aws.amazon.com/premiumsupport/knowledge-center/cloudwatch-notification-scheduled-events/
https://docs.aws.amazon.com/health/latest/ug/cloudwatch-events-health.html

Using AWS Health API and CloudWatch Events for Monitoring AWS-Scheduled Deployments/Changes

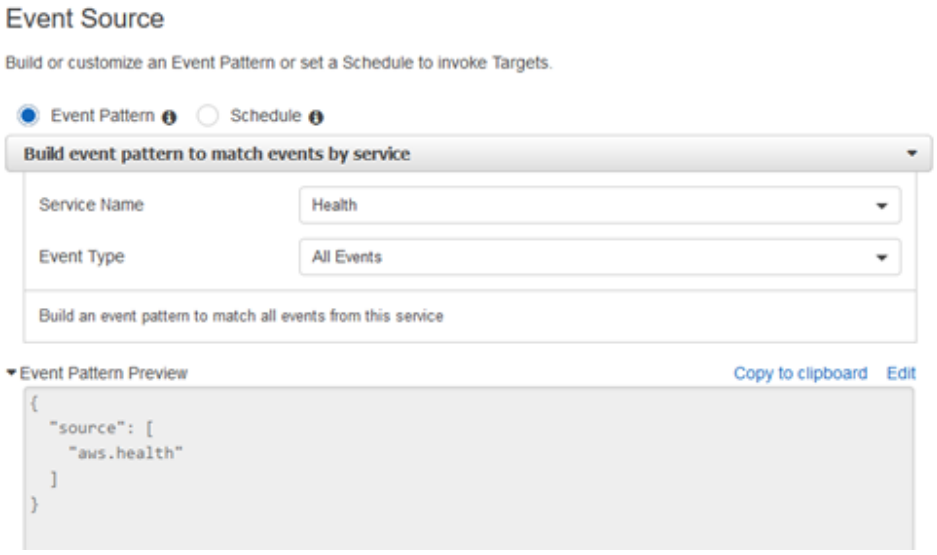
AWS Personal Health Dashboard provides you with alerts and remediation status when AWS is experiencing events that may impact your resources. These events may be scheduled or unscheduled. For example, scheduled events such as changes on your underlying EC2 hosts may shutdown or terminate your instances, or AWS RDS scheduled upgrades that may reboot your RDS instance.

DevOps Exam Notes:

You can monitor these AWS Health events using CloudWatch Events by calling the AWS Health API. Then, you can set a target to an SNS topic to inform you of that Event, or you can trigger a Lambda function to perform a custom action based on the Event.

Here are the steps on to set this up:

1. Go to CloudWatch > Events > Rules and create a rule that will detect AWS Health Events.
2. Select the Services in Event Type if you want to monitor a particular service, or choose All Events to be notified for all events.

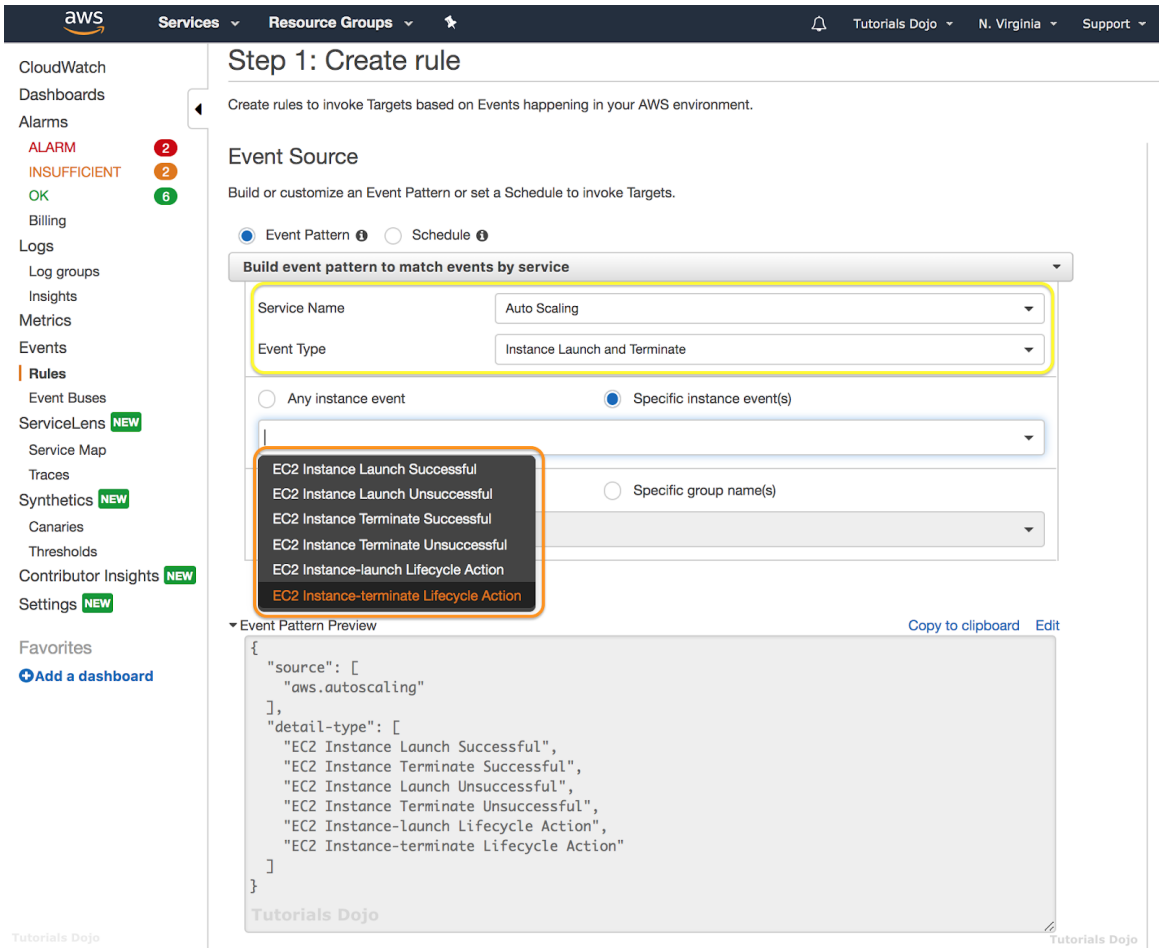


3. You can select a Target such as an SNS Topic if you want to get notified of the event, or Lambda function that can perform a custom action based on your requirements.
4. Click the Configure details to save and activate this CloudWatch Events rule.

Monitoring Amazon EC2 Auto Scaling Events

Auto Scaling provides fault tolerance, high availability, and cost management to your computing resources. It automatically scales in (terminates existing EC2 instances) if the demand is low and scales out (launch new EC2 instances) if the incoming traffic exceeds the specified threshold. AWS offers various ways of monitoring your fleet of Amazon EC2 instances as well responding to Auto Scaling events.

You can either use Amazon EventBridge or Amazon CloudWatch Events to track the Auto Scaling Events and run a corresponding custom action using AWS Lambda. Amazon EventBridge extends the capabilities of Amazon CloudWatch Events to integrate internal and external services. It allows you to track the changes of your Auto Scaling group in near real-time, including your custom applications, Software-as-a-Service (SaaS) partner apps, and other AWS services.



Amazon EventBridge or Amazon CloudWatch Events can be used to send events to the specified target when the following events occur:

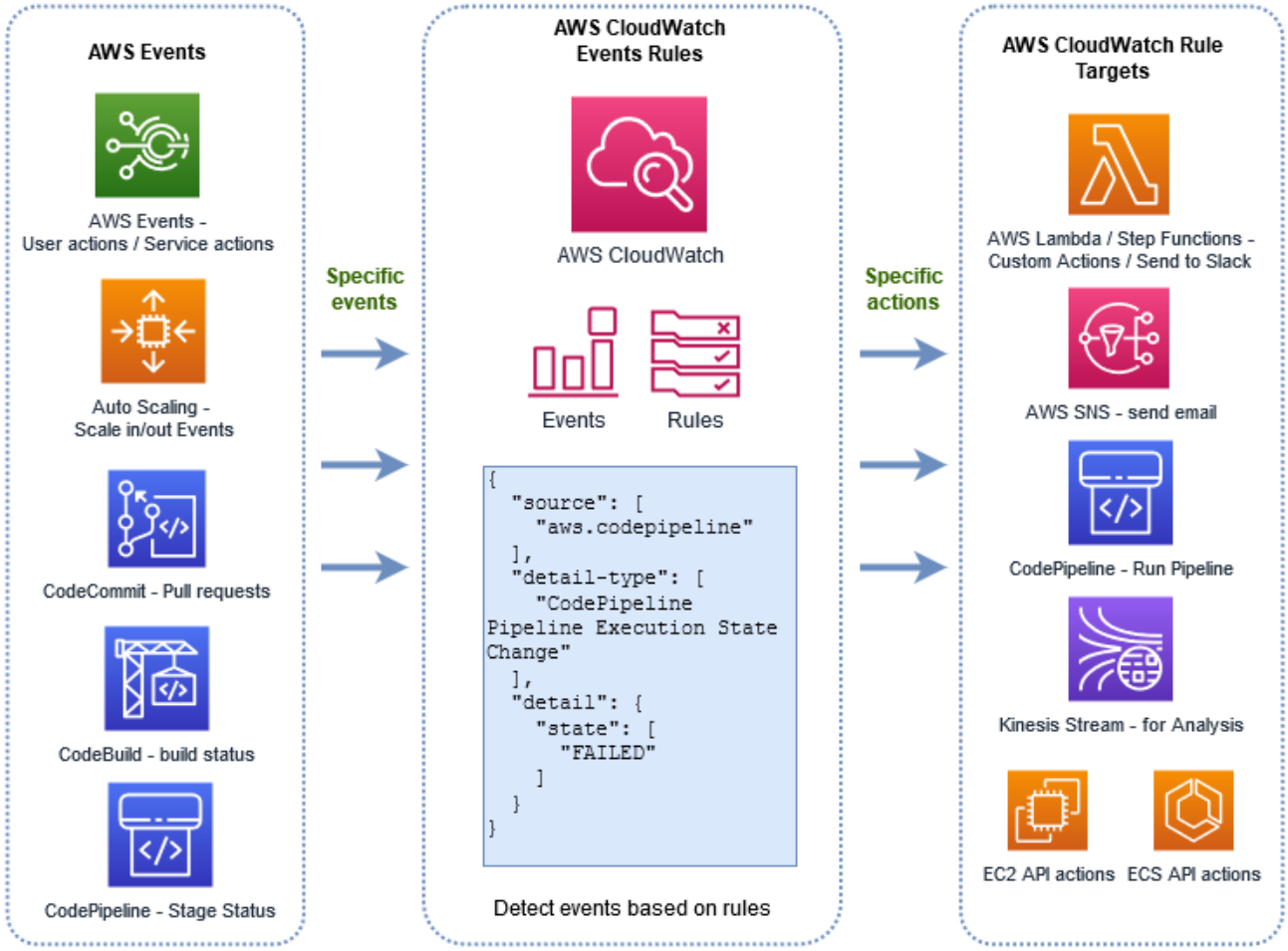
- EC2 Instance-launch Lifecycle Action
- EC2 Instance Launch Successful
- EC2 Instance Launch Unsuccessful
- EC2 Instance-terminate Lifecycle Action
- EC2 Instance Terminate Successful
- EC2 Instance Terminate Unsuccessful

The **`EC2 Instance-launch Lifecycle Action`** is a scale-out event in which the Amazon EC2 Auto Scaling moved an EC2 instance to a **`Pending:Wait`** state due to a lifecycle hook. Conversely, the **`EC2 Instance-terminate Lifecycle Action`** is a scale-in event in which EC2 Auto Scaling updates an instance to a **`Terminating:Wait`** state.

Source:
https://docs.aws.amazon.com/autoscaling/ec2/userguide/cloud-watch-events.html

AWS CodePipeline Event Patterns

You can detect and respond to certain changes of your pipeline state in AWS CodePipeline using Amazon CloudWatch Events. You can use AWS CodePipeline (aws.codepipeline) as the event source of your CloudWatch Events rule and then associate an Amazon SNS topic to send notification or a Lambda function to execute a custom action. CloudWatch Events can automatically detect the state changes of your pipelines, stages, or actions in AWS CodePipeline which improves incident and event management of your CI/CD processes.



You can specify both the state and type of CodePipeline execution that you want to monitor. An item can be in a **STARTED**, **SUCCEEDED**, **RESUMED**, **FAILED**, **CANCELED** or **SUPERSEDED** state. Refer to the table below for the list of available detail types that you can use.

Entity	Detail Type
Pipeline	**CodePipeline Pipeline Execution State Change**
Stage	**CodePipeline Stage Execution State Change**
Action	**CodePipeline Action Execution State Change**

You can use this sample event pattern to capture failed **deploy** and **build** actions across all your pipelines in AWS CodePipeline:

```
{
  "source": [
    "aws.codepipeline"
  ],
  "detail-type": [
    "CodePipeline Action Execution State Change"
  ],
  "detail": {
    "state": [
      "FAILED"
    ],
    "type": {
      "category": ["Deploy", "Build"]
    }
  }
}
```

The sample event pattern below captures all **`rejected`** or **`failed`** approval actions across all the pipelines:

```
{
  "source": [
    "aws.codepipeline"
  ],
  "detail-type": [
    "CodePipeline Action Execution State Change"
  ],
  "detail": {
    "state": [
      "FAILED"
    ],
    "type": {
      "category": ["Approval"]
    }
  }
}
```

The following sample event pattern tracks all the events from the specified pipelines (**`TD-Pipeline-Manila`**, **`TD-Pipeline-Frankfurt`** and **`TD-Pipeline-New-York`**)

```
{
"source": [
    "aws.codepipeline"
  ],
  "detail-type": [
    "CodePipeline Pipeline Execution State Change",
    "CodePipeline Action Execution State Change",
    "CodePipeline Stage Execution State Change"
  ],
  "detail": {
    "pipeline": ["TD-Pipeline-Manila",
                 "TD-Pipeline-Frankfurt",
                 "TD-Pipeline-New-York"]
  }
}
```

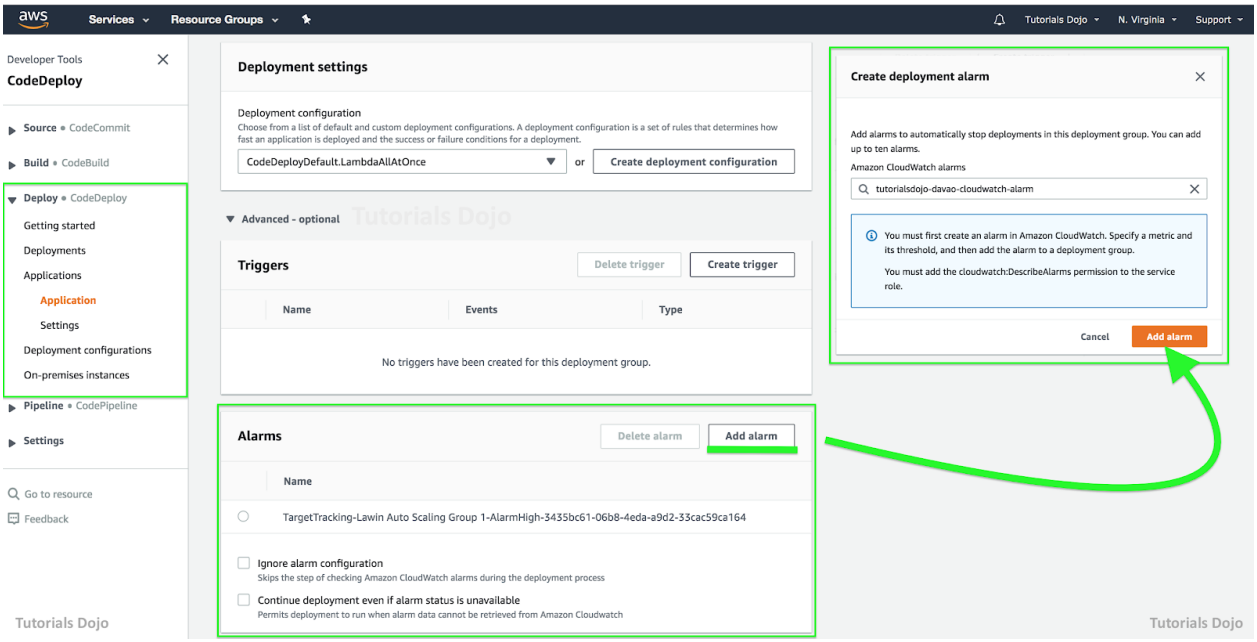
Source:
https://docs.aws.amazon.com/codepipeline/latest/userguide/detect-state-changes-cloudwatch-events.html

Monitoring Deployments in AWS CodeDeploy with CloudWatch Events and Alarms

Amazon CloudWatch Events helps you detect any changes in the state of an instance or a deployment in AWS CodeDeploy. You can also send notifications, collect state information, rectify issues, initiate events, or execute other actions using CloudWatch Alarms. This type of monitoring is useful if you want to be notified via Slack (or in other channels) whenever your deployments fail or push deployment data to a Kinesis stream for real-time status monitoring. If you integrated Amazon CloudWatch Events in your AWS CodeDeploy operations, you can specify the following as targets to monitor your deployments:

- AWS Lambda functions
- Kinesis streams
- Amazon SQS queues
- Built-in targets (CloudWatch alarm actions)
- Amazon SNS topics

Integrating AWS CodeDeploy and CloudWatch Alarms provides you an automated way to roll back your release when your deployment fails or if certain thresholds are not met. You can easily track the minimum number of healthy instances (`MinimumHealthyHosts`) that should be available at any time during the deployment. The `HOST_COUNT` or `FLEET_PERCENT` deployment configuration parameters can also be utilized to monitor the absolute number or just the relative percentage of healthy hosts respectively.



Source:
https://docs.aws.amazon.com/codedeploy/latest/userguide/monitoring-create-alarms.html

    - IAM roles
- Restrict access by only allowing trusted hosts or networks to access ports on your instance.
- A **security group** acts as a virtual firewall that controls the traffic for one or more instances.
    - Evaluates all the rules from all the security groups that are associated with an instance to decide whether to allow traffic or not.
    - By default, security groups allow **all outbound traffic**.
    - Security group rules are **always permissive**; you can't create rules that deny access.
    - Security groups are **stateful**
- You can replicate the network traffic from an EC2 instance within your Amazon VPC and forward that traffic to security and monitoring appliances for content inspection, threat monitoring, and troubleshooting.

Networking

- An **Elastic IP address** is a static IPv4 address designed for dynamic cloud computing. With it, you can mask the failure of an instance or software by rapidly remapping the address to another instance in your account.
- If you have not enabled auto-assign public IP address for your instance, you need to associate an Elastic IP address with your instance to enable communication with the internet.
- An elastic **network interface** is a logical networking component in a VPC that represents a virtual network card, which directs traffic to your instance
- Scale with **EC2 Scaling Groups** and distribute traffic among instances using **Elastic Load Balancer**.

Monitoring

- EC2 items to monitor
    - CPU utilization, Network utilization, Disk performance, Disk Reads/Writes using EC2 metrics
    - Memory utilization, disk swap utilization, disk space utilization, page file utilization, log collection using a monitoring agent/CloudWatch Logs
- Automated monitoring tools include:
    - System Status Checks - monitor the AWS systems required to use your instance to ensure they are working properly. These checks detect problems with your instance that require AWS involvement to repair.
    - Instance Status Checks - monitor the software and network configuration of your individual instance. These checks detect problems that require your involvement to repair.
    - Amazon CloudWatch Alarms - watch a single metric over a time period you specify, and perform one or more actions based on the value of the metric relative to a given threshold over a number of time periods.
    - Amazon CloudWatch Events - automate your AWS services and respond automatically to system events.

    - Amazon CloudWatch Logs - monitor, store, and access your log files from Amazon EC2 instances, AWS CloudTrail, or other sources.
- Monitor your EC2 instances with CloudWatch. By default, EC2 sends metric data to CloudWatch in 5-minute periods.
- You can also enable detailed monitoring to collect data in 1-minute periods.

Instance Metadata and User Data

- **Instance metadata** is data about your instance that you can use to configure or manage the running instance.
- View all categories of instance metadata from within a running instance at **http://169.254.169.254/latest/meta-data/**
- Retrieve user data from within a running instance at **http://169.254.169.254/latest/user-data**

    - When using the Fargate launch type with tasks within your cluster, ECS manages your cluster resources.
    - Enabling managed Amazon ECS cluster auto scaling allows ECS to manage the scale-in and scale-out actions of the Auto Scaling group.
- Services
    - ECS allows you to run and maintain a specified number of instances of a task definition simultaneously in a cluster.
    - In addition to maintaining the desired count of tasks in your service, you can optionally run your service behind a load balancer.
    - There are two deployment strategies in ECS:
        - **Rolling Update**
            - This involves the service scheduler replacing the current running version of the container with the latest version.

          Blue/Green Deployment with AWS CodeDeploy
            - This deployment type allows you to verify a new deployment of a service before sending production traffic to it.
            - The service must be configured to use either an Application Load Balancer or Network Load Balancer.
- Container Agent (AWS ECS Agent)
    - The **container agent** runs on each infrastructure resource within an ECS cluster.
    - It sends information about the resource's current running tasks and resource utilization to ECS, and starts and stops tasks whenever it receives a request from ECS.
    - Container agent is only supported on Amazon EC2 instances.

AWS Fargate

- You can use Fargate with ECS to run containers without having to manage servers or clusters of EC2 instances.
- You no longer have to provision, configure, or scale clusters of virtual machines to run containers.
- Fargate only supports container images hosted on Elastic Container Registry (ECR) or Docker Hub.

Task Definitions for Fargate Launch Type

- Fargate task definitions require that the network mode is set to *awsvpc*. The *awsvpc* network mode provides each task with its own elastic network interface.
- Fargate task definitions only support the *awslogs* log driver for the log configuration. This configures your Fargate tasks to send log information to Amazon CloudWatch Logs.
- Task storage is **ephemeral**. After a Fargate task stops, the storage is deleted.

Monitoring

- You can configure your container instances to send log information to CloudWatch Logs. This enables you to view different logs from your container instances in one convenient location.
- With CloudWatch Alarms, watch a single metric over a time period that you specify, and perform one or more actions based on the value of the metric relative to a given threshold over a number of time periods.
- Share log files between accounts, monitor CloudTrail log files in real time by sending them to CloudWatch Logs.

AWS Management Tools

Amazon CloudWatch

- Monitoring tool for your AWS resources and applications.
- Display metrics and create alarms that watch the metrics and send notifications or automatically make changes to the resources you are monitoring when a threshold is breached.
- CloudWatch is basically a metrics repository. An AWS service, such as Amazon EC2, puts metrics into the repository and you retrieve statistics based on those metrics. If you put your own custom metrics into the repository, you can retrieve statistics on these metrics as well.
- CloudWatch does not aggregate data across regions. Therefore, metrics are completely separate between regions.
- **CloudWatch Concepts**
    - **Namespaces** - a container for CloudWatch metrics.
        - There is no default namespace.
        - The AWS namespaces use the following naming convention: AWS/*service*.
    - **Metrics** - represents a time-ordered set of data points that are published to CloudWatch.
        - Exists only in the region in which they are created.
        - By default, several services provide free metrics for resources. You can also enable **detailed monitoring**, or publish your own application metrics.
        - **Metric math** enables you to query multiple CloudWatch metrics and use math expressions to create new time series based on these metrics.
        - **Important note for EC2 metrics:** CloudWatch does not collect memory utilization and disk space usage metrics right from the get go. You need to install CloudWatch Agent in your instances first to retrieve these metrics.
    - **Dimensions** - a name/value pair that uniquely identifies a metric.
        - You can assign up to 10 dimensions to a metric.
        - Whenever you add a unique dimension to one of your metrics, you are creating a new variation of that metric.
    - **Statistics** - metric data aggregations over specified periods of time.
        - Each statistic has a unit of measure. Metric data points that specify a unit of measure are aggregated separately.
        - You can specify a unit when you create a custom metric. If you do not specify a unit, CloudWatch uses *None* as the unit.
        - A *period* is the length of time associated with a specific CloudWatch statistic. The default value is 60 seconds.
        - CloudWatch aggregates statistics according to the period length that you specify when retrieving statistics.
        - For large datasets, you can insert a pre-aggregated dataset called a *statistic set*.

- **Alarms** - watches a single metric over a specified time period, and performs one or more specified actions, based on the value of the metric relative to a threshold over time.
    - You can create an alarm for monitoring CPU usage and load balancer latency, for managing instances, and for billing alarms.
    - When an alarm is on a dashboard, it turns red when it is in the *ALARM* state.
    - Alarms invoke actions for sustained state changes only.
    - Alarm States
        - **OK**—The metric or expression is within the defined threshold.
        - **ALARM**—The metric or expression is outside of the defined threshold.
        - **INSUFFICIENT_DATA**—The alarm has just started, the metric is not available, or not enough data is available for the metric to determine the alarm state.
    - You can also monitor your estimated AWS charges by using Amazon CloudWatch Alarms. However, take note that you can only track the estimated AWS charges in CloudWatch and not the actual utilization of your resources. Remember that you can only set coverage targets for your reserved EC2 instances in AWS Budgets or Cost Explorer, but not in CloudWatch.
    - When you create an alarm, you specify three settings:
        - **Period** is the length of time to evaluate the metric or expression to create each individual data point for an alarm. It is expressed in seconds.
        - **Evaluation Period** is the number of the most recent periods, or data points, to evaluate when determining alarm state.
        - **Datapoints to Alarm** is the number of data points within the evaluation period that must be breaching to cause the alarm to go to the ALARM state. The breaching data points do not have to be consecutive, they just must all be within the last number of data points equal to **Evaluation Period**.

CloudWatch Dashboard

- Customizable home pages in the CloudWatch console that you can use to monitor your resources in a single view, even those spread across different regions.
- There is no limit on the number of CloudWatch dashboards you can create.
- All dashboards are **global**, not region-specific.
- You can add, remove, resize, move, edit or rename a graph. You can metrics manually in a graph.

CloudWatch Events

- Deliver near real-time stream of system events that describe changes in AWS resources.
- Events respond to these operational changes and take corrective action as necessary, by sending messages to respond to the environment, activating functions, making changes, and capturing state information.
- Concepts
    - **Events** - indicates a change in your AWS environment.
    - **Targets** - processes events.
    - **Rules** - matches incoming events and routes them to targets for processing.

CloudWatch Logs

- Features

- Monitor logs from EC2 instances in real-time
- Monitor CloudTrail logged events
- By default, logs are kept indefinitely and never expire
- Archive log data
- Log Route 53 DNS queries

CloudWatch Agent

- Collect more logs and system-level metrics from EC2 instances and your on-premises servers.
- Needs to be installed.

Comparison of AWS Services

AWS CloudTrail vs Amazon CloudWatch

- CloudWatch is a monitoring service for AWS resources and applications. CloudTrail is a web service that records API activity in your AWS account. They are both useful monitoring tools in AWS.
- By default, CloudWatch offers free basic monitoring for your resources, such as EC2 instances, EBS volumes, and RDS DB instances. CloudTrail is also enabled by default when you create your AWS account.
- With CloudWatch, you can collect and track metrics, collect and monitor log files, and set alarms. CloudTrail, on the other hand, logs information on who made a request, the services used, the actions performed, parameters for the actions, and the response elements returned by the AWS service. CloudTrail Logs are then stored in an S3 bucket or a CloudWatch Logs log group that you specify.
- You can enable detailed monitoring from your AWS resources to send metric data to CloudWatch more frequently, with an additional cost.
- CloudTrail delivers one free copy of management event logs for each AWS region. Management events include management operations performed on resources in your AWS account, such as when a user logs in to your account. Logging data events are charged. Data events include resource operations performed on or within the resource itself, such as S3 object-level API activity or Lambda function execution activity.
- CloudTrail helps you ensure compliance and regulatory standards.
- CloudWatch Logs reports on application logs, while CloudTrail Logs provide you specific information on what occurred in your AWS account.
- CloudWatch Events is a near real time stream of system events describing changes to your AWS resources. CloudTrail focuses more on AWS API calls made in your AWS account.
- Typically, CloudTrail delivers an event within 15 minutes of the API call. CloudWatch delivers metric data in 5 minutes periods for basic monitoring and 1 minute periods for detailed monitoring. The CloudWatch Logs Agent will send log data every five seconds by default.