JON BONSO AND KENNETH SAMONTE

AWS CERTIFIED

DEVOPS ENGINEER PROFESSIONAL

Tutorials Dojo
Study Guide and Cheat Sheets

TABLE OF CONTENTS

INTRODUCTION

As more companies build their DevOps practices, there will always be a growing demand for certified IT Professionals that can do agile software development, configuration management, task automation, and continuous integration/continuous delivery (CI/CD). This Study Guide and Cheat Sheets eBook for AWS Certified DevOps Engineer Professional aims to equip you with the necessary knowledge and practical skill sets needed to pass the latest version of the AWS Certified DevOps Engineer – Professional exam.

This eBook contains the essential concepts, exam domains, exam tips, sample questions, cheat sheets, and other relevant information about the AWS Certified DevOps Engineer – Professional exam. This study guide begins with the presentation of the exam structure, giving you an insight into the question types, exam domains, scoring scheme, and the list of benefits you'll receive once you pass the exam. We used the official AWS exam guide to structure the contents of this guide, where each section discusses a particular exam domain. Various DevOps concepts, related AWS services, and technical implementations are covered to provide you an idea of what to expect on the actual exam.

DevOps Exam Notes:

Don't forget to read the boxed "**exam tips**" (like this one) scattered throughout the eBook as these are the key concepts that you will likely encounter on your test. After covering the six domains, we have added a bonus section containing a curated list of AWS Cheat Sheets to fast track your review. The last part of this guide includes a collection of articles that compares two or more similar AWS services to supplement your knowledge.

The AWS Certified DevOps Engineer - Professional certification exam is a difficult test to pass; therefore, anyone who wants to take it must allocate ample time for review. The exam registration cost is not cheap, which is why we spent considerable time and effort to ensure that this study guide provides you with the essential and relevant knowledge to increase your chances of passing the DevOps exam.

*** **Note:** This eBook is meant to be just a supplementary resource when preparing for the exam. We highly recommend working on hands-on sessions and practice exams to further expand your knowledge and improve your test taking skills.*

AWS CERTIFIED DEVOPS ENGINEER PROFESSIONAL EXAM OVERVIEW

The AWS Certified DevOps Engineer - Professional certification exam validates various skills which are necessary to become a full-fledged DevOps engineer. The exam will check your capability in implementing and managing continuous delivery systems and methodologies on AWS. Automating security controls, validating compliance and optimizing the governance processes are also included in the test.

Exam Details

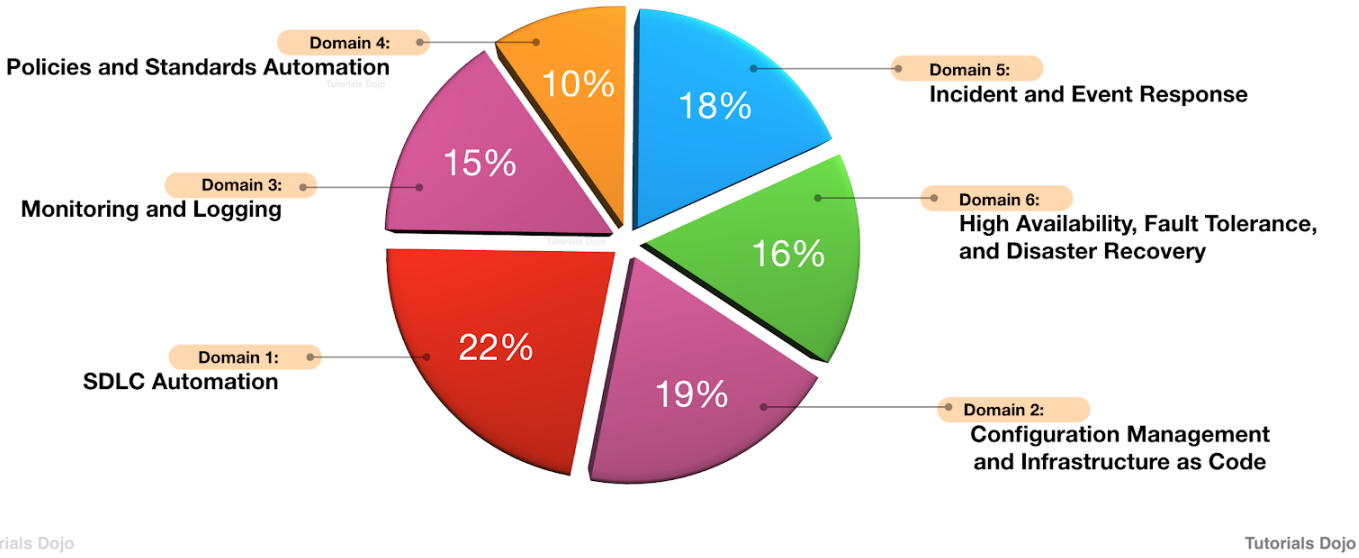
The AWS Certified DevOps Engineer Professional certification is intended for IT Professionals who perform a Solutions Architect or DevOps role and have substantial years of hands-on experience designing available, cost-efficient, fault-tolerant, and scalable distributed systems on the AWS platform. It is composed of scenario-based questions that can either be in multiple-choice or multiple response formats. The first question type has one correct answer and three incorrect responses, while the latter has two or more correct responses out of five or more options. You can take the exam from a local testing center or online from the comforts of your home.

Exam Code:	DOP-C01
Release Date:	February 2019
Prerequisites:	None
No. of Questions:	75
Score Range:	100 - 1000
Cost:	300 USD (Practice exam: 40 USD)
Passing Score:	750/1000
Time Limit:	3 hours (180 minutes)
Format:	Scenario-based. Multiple choice/multiple answers.
Delivery Method:	Testing center or online proctored exam.

Don't be confused if you see in your Pearson Vue booking that the duration is 190 minutes since they included an additional 10 minutes for reading the Non-Disclosure Agreement (NDA) at the start of the exam and the survey at the end of it. If you booked via PSI, the exam duration time that you will see is 180 minutes.

Exam Domains

The AWS Certified DevOps Engineer Professional (DOP-C01) exam has 6 different domains, each with corresponding weight and topic coverage. The exam domains are as follows: SDLC Automation (22%), Configuration Management and Infrastructure as Code (19%), Monitoring and Logging (15%), Policies and Standards Automation (10%), Incident and Event Response (18%) and lastly, High Availability, Fault Tolerance, and Disaster Recovery (16%):



Domain 1: SDLC Automation (22%)

1.1 Apply concepts required to automate a CI/CD pipeline
1.2 Determine source control strategies and how to implement them
1.3 Apply concepts required to automate and integrate testing
1.4 Apply concepts required to build and manage artifacts securely
1.5 Determine deployment/delivery strategies (e.g., A/B, Blue/green, Canary, Red/black) and how to implement them using AWS Services

Domain 2: Configuration Management and Infrastructure as Code (19%)

2.1 Determine deployment services based on deployment needs
2.2 Determine application and infrastructure deployment models based on business needs
2.3 Apply security concepts in the automation of resource provisioning
2.4 Determine how to implement lifecycle hooks on a deployment
2.5 Apply concepts required to manage systems using AWS configuration management tools and services

Domain 3: Monitoring and Logging (15%)

- 3.1 Determine how to set up the aggregation, storage, and analysis of logs and metrics
- 3.2 Apply concepts required to automate monitoring and event management of an environment
- 3.3 Apply concepts required to audit, log, and monitor operating systems, infrastructures, and applications
- 3.4 Determine how to implement tagging and other metadata strategies

Domain 4: Policies and Standards Automation (10%)

- 4.1 Apply concepts required to enforce standards for logging, metrics, monitoring, testing, and security
- 4.2 Determine how to optimize cost through automation
- 4.3 Apply concepts required to implement governance strategies

Domain 5: Incident and Event Response (18%)

- 5.1 Troubleshoot issues and determine how to restore operations
- 5.2 Determine how to automate event management and alerting
- 5.3 Apply concepts required to implement automated healing
- 5.4 Apply concepts required to set up event-driven automated actions

Domain 6: High Availability, Fault Tolerance, and Disaster Recovery (16%)

- 6.1 Determine appropriate use of multi-AZ versus multi-region architectures
- 6.2 Determine how to implement high availability, scalability, and fault tolerance
- 6.3 Determine the right services based on business needs (e.g., RTO/RPO, cost)
- 6.4 Determine how to design and automate disaster recovery strategies
- 6.5 Evaluate a deployment for points of failure

Exam Scoring System

You can get a score from 100 to 1,000 with a minimum passing score of **750** when you take the DevOps Engineer Professional exam. AWS uses a scaled scoring model to equate scores across multiple exam types that may have different difficulty levels. The complete score report will be sent to you by email after a few days. Right after you completed the actual exam, you'll immediately see a pass or fail notification on the testing screen. A *"Congratulations! You have successfully passed..."* message will be shown if you passed the exam.

Individuals who unfortunately do not pass the AWS exam must wait 14 days before they are allowed to retake the exam. Fortunately, there is no hard limit on exam attempts until you pass the exam. Take note that on each attempt, the full registration price of the AWS exam must be paid.

Within 5 business days of completing your exam, your AWS Certification Account will have a record of your complete exam results. The score report contains a table of your performance at each section/domain, which indicates whether you met the competency level required for these domains or not. AWS is using a compensatory scoring model, which means that you do not necessarily need to pass each and every individual section, only the overall examination. Each section has a specific score weighting that translates to the number of questions; hence, some sections have more questions than others. The Score Performance table highlights your strengths and weaknesses that you need to improve on.

Exam Benefits

If you successfully passed any AWS exam, you will be eligible for the following benefits:

- **Exam Discount** - You'll get a 50% discount voucher that you can apply for your recertification or any other exam you plan to pursue. To access your discount voucher code, go to the "Benefits" section of your AWS Certification Account, and apply the voucher when you register for your next exam.

- **Free Practice Exam** - To help you prepare for your next exam, AWS provides another voucher that you can use to take any official AWS practice exam for free. You can access your voucher code from the "Benefits" section of your AWS Certification Account.

- **AWS Certified Store** - All AWS certified professionals will be given access to exclusive AWS Certified merchandise. You can get your store access from the "Benefits" section of your AWS Certification Account.

- **Certification Digital Badges** - You can showcase your achievements to your colleagues and employers with digital badges on your email signatures, Linkedin profile, or on your social media accounts. You can also show your Digital Badge to gain exclusive access to Certification Lounges at AWS re:Invent, regional Appreciation Receptions, and select AWS Summit events. To view your badges, simply go to the "Digital Badges" section of your AWS Certification Account.

- **Eligibility to join AWS IQ -** With the AWS IQ program, you can monetize your AWS skills online by providing hands-on assistance to customers around the globe. AWS IQ will help you stay sharp and be well-versed on various AWS technologies. You can work at the comforts of your home and decide when or where you want to work. Interested individuals must be based in the US, have an Associate, Professional, or Specialty AWS Certification and be over 18 of age.

You can visit the official AWS Certification FAQ page to view the frequently asked questions about getting AWS Certified and other information about the AWS Certification: https://aws.amazon.com/certification/faqs/**.**

AWS CERTIFIED DEVOPS ENGINEER PROFESSIONAL EXAM - STUDY GUIDE AND TIPS

This certification is the pinnacle of your DevOps career in AWS. The AWS Certified DevOps Engineer Professional (or AWS DevOps Pro) is the advanced certification of both AWS SysOps Administrator Associate and AWS Developer Associate. This is similar to how the AWS Solutions Architect Professional role is a more advanced version of the AWS Solutions Architect Associate.

Generally, AWS recommends that you first take (and pass) both AWS SysOps Administrator Associate and AWS Developer Associate certification exams before taking on this certification. Previously, it was a prerequisite that you obtain the associate level certifications before you are allowed to go for the professional level. Last October 2018, AWS removed this ruling to provide customers a more flexible approach to the certifications.

Study Materials

The FREE AWS Exam Readiness course, official AWS sample questions, Whitepapers, FAQs, AWS Documentation, Re:Invent videos, forums, labs, AWS cheat sheets, practice tests, and personal experiences are what you will need to pass the exam. Since the DevOps Pro is one of the most difficult AWS certification exams out there, you have to prepare yourself with every study material you can get your hands on. If you need a review on the fundamentals of AWS DevOps, then do check out our review guides for the AWS SysOps Administrator Associate and AWS Developer Associate certification exams. Also, visit this AWS exam blueprint to learn more details about your certification exam.

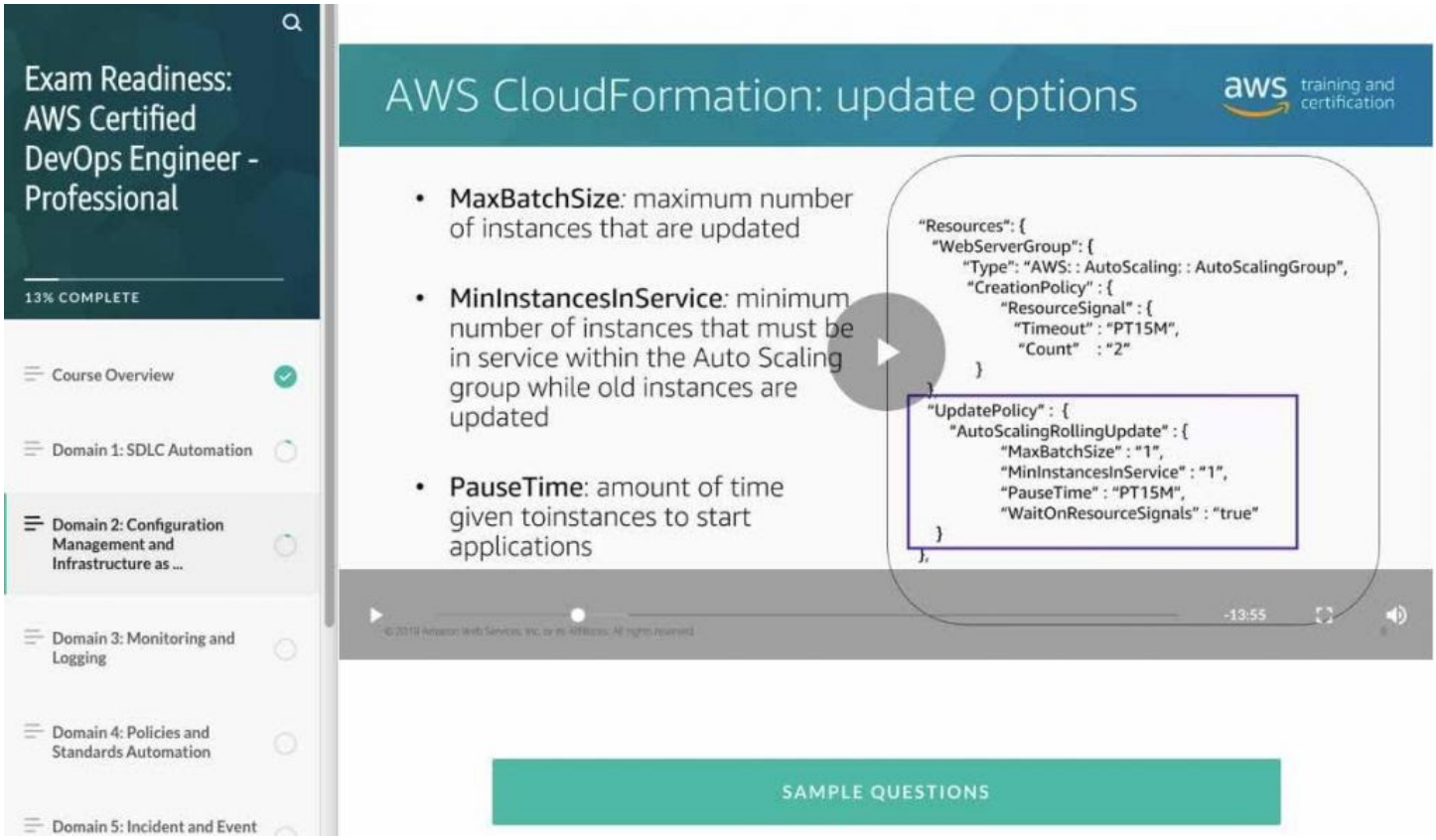
For virtual classes, you can attend the **DevOps Engineering on AWS** and **Systems Operations on AWS** classes since they will teach you concepts and practices that are expected to be in your exam.

For whitepapers, focus on the following:

1. Running Containerized Microservices on AWS
2. Microservices on AWS
3. Infrastructure as Code
4. Introduction to DevOps
5. Practicing Continuous Integration and Continuous Delivery on AWS
6. Jenkins on AWS
7. Blue/Green Deployments on AWS whitepaper
8. Import Windows Server to Amazon EC2 with PowerShell
9. Development and Test on AWS

Almost all online training you need can be found on the AWS web page. One digital course that you should check out is the Exam Readiness: AWS Certified DevOps Engineer – Professional course. This digital course

contains lectures on the different domains of your exam, and they also provide a short quiz right after each lecture to validate what you have just learned.



Lastly, do not forget to study the AWS CLI, SDKs, and APIs. Since the DevOps Pro is also an advanced certification for Developer Associate, you need to have knowledge of programming and scripting in AWS. Go through the AWS documentation to review the syntax of CloudFormation template, Serverless Application Model template, CodeBuild buildspec, CodeDeploy appspec, and IAM Policy.

Also check out this article: **Top 5 FREE AWS Review Materials**.

AWS Services to Focus On

Since this exam is a professional level one, you should already have a deep understanding of the AWS services listed under our SysOps Administrator Associate and Developer Associate review guides. In addition, you should familiarize yourself with the following services since they commonly come up in the DevOps Pro exam:

1. AWS CloudFormation
2. AWS Lambda

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

High Availability, Fault Tolerance, and Disaster Recovery	
Need to ensure that both the application and the database are running in the event that one Availability Zone becomes unavailable.	Deploy your application on multiple Availability Zones and set up your Amazon RDS database to use Multi-AZ Deployments.
In the event of an AWS Region outage, you have to make sure that both your application and database will still be running to avoid any service outages.	Create a copy of your deployment on the backup AWS region. Set up an RDS Read-Replica on the backup region.
Automatically switch traffic to the backup region when your primary AWS region fails	Set up Route 53 Failover routing policy with health check enabled on your primary region endpoint.
Need to ensure the availability of a legacy application running on a single EC2 instance	Set up an Auto Scaling Group with MinSize=1 and MaxSize=1 configuration to set a fixed count and ensure that it will be replaced when the instance becomes unhealthy
Ensure that every EC2 instance on an Auto Scaling group downloads the latest code first before being attached to a load balancer	Create an Auto Scaling Lifecycle hook and configure the Pending:Wait hook with the action to download all necessary packages.
Ensure that all EC2 instances on an Auto Scaling group upload all log files in the S3 bucket before being terminated.	Use the Auto Scaling Lifecycle and configure the Terminating:Wait hook with the action to upload all logs to the S3 bucket.

Validate Your Knowledge

After your review, you should take some practice tests to measure your preparedness for the real exam. AWS offers a sample practice test for free which you can find here. You can also opt to buy the longer AWS sample practice test at aws.training, and use the discount coupon you received from any previously taken certification exams. Be aware though that the sample practice tests do not mimic the difficulty of the real DevOps Pro exam.

Therefore, we highly encourage using other mock exams such as our very own **AWS Certified DevOps Engineer Professional Practice Exam** course which contains high-quality questions with complete explanations on correct and incorrect answers, visual images and diagrams, YouTube videos as needed, and also contains reference links to official AWS documentation as well as our cheat sheets and study guides. You can also pair

our practice exams with our **AWS Certified DevOps Engineer Professional Exam Study Guide eBook** to further help in your exam preparations.

Sample Practice Test Questions:

Question 1

An application is hosted in an Auto Scaling group of Amazon EC2 instances with public IP addresses in a public subnet. The instances are configured with a user data script that fetches and installs the required system dependencies of the application from the Internet upon launch. A change was recently introduced to prohibit any Internet access from these instances to improve the security but after its implementation, the instances could not get the external dependencies anymore. Upon investigation, all instances are properly running but the hosted application is not starting up completely due to the incomplete installation.

Which of the following is the MOST secure solution to solve this issue and also ensure that the instances do not have public Internet access?

1. Download all of the external application dependencies from the public Internet and then store them in an S3 bucket. Set up a VPC endpoint for the S3 bucket and then assign an IAM instance profile to the instances in order to allow them to fetch the required dependencies from the bucket.

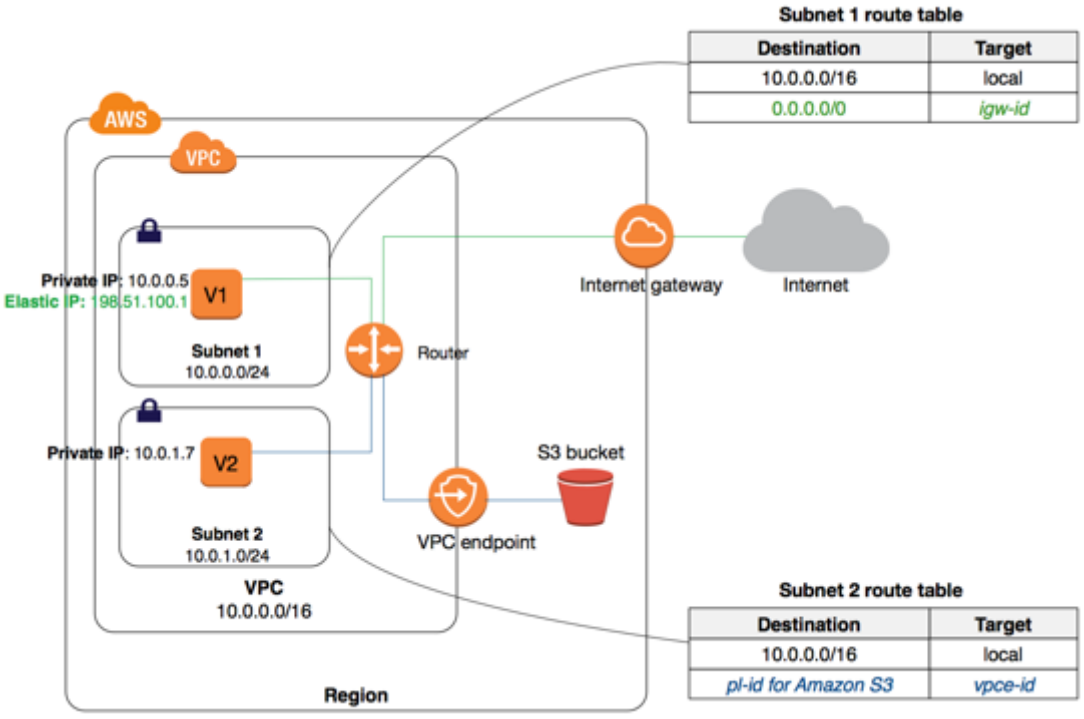
2. Deploy the Amazon EC2 instances in a private subnet and associate Elastic IP addresses on each of them. Run a custom shell script to disassociate the Elastic IP addresses after the application has been successfully installed and is running properly.
3. Use a NAT gateway to disallow any traffic to the VPC which originated from the public Internet. Deploy the Amazon EC2 instances to a private subnet then set the subnet's route table to use the NAT gateway as its default route.
4. Set up a brand new security group for the Amazon EC2 instances. Use a whitelist configuration to only allow outbound traffic to the site where all of the application dependencies are hosted. Delete the security group rule once the installation is complete. Use AWS Config to monitor the compliance.

Correct Answer: 1

A VPC endpoint enables you to privately connect your VPC to supported AWS services and VPC endpoint services powered by PrivateLink without requiring an Internet gateway, NAT device, VPN connection, or AWS Direct Connect connection. Instances in your VPC do not require public IP addresses to communicate with resources in the service. Traffic between your VPC and the other service does not leave the Amazon network.

Endpoints are virtual devices. They are horizontally scaled, redundant, and highly available VPC components that allow communication between instances in your VPC and services without imposing availability risks or bandwidth constraints on your network traffic.

There are two types of VPC endpoints: *interface endpoints* and *gateway endpoints*. You can create the type of VPC endpoint required by the supported service. S3 and DynamoDB are using Gateway endpoints while most of the services are using Interface endpoints.



You can use an S3 bucket to store the required dependencies and then set up a VPC Endpoint to allow your EC2 instances to access the data without having to traverse the public Internet.

Hence, the correct answer is the option that says: **Download all of the external application dependencies from the public Internet and then store them to an S3 bucket. Set up a VPC endpoint for the S3 bucket and then assign an IAM instance profile to the instances in order to allow them to fetch the required dependencies from the bucket.**

The option that says: **Deploy the Amazon EC2 instances in a private subnet and associate Elastic IP addresses on each of them. Run a custom shell script to disassociate the Elastic IP addresses after the application has been successfully installed and is running properly** is incorrect because it is possible that the custom shell script may fail and the disassociation of the Elastic IP addresses might not be fully implemented which will allow the EC2 instances to access the Internet.

The option that says: **Use a NAT gateway to disallow any traffic to the VPC which originated from the public Internet. Deploy the Amazon EC2 instances to a private subnet then set the subnet's route table to use the NAT gateway as its default route** is incorrect because although a NAT Gateway can safeguard the instances from any incoming traffic that were initiated from the Internet, it still permits them to send outgoing requests externally.

The option that says: **Set up a brand new security group for the Amazon EC2 instances. Use a whitelist configuration to only allow outbound traffic to the site where all of the application dependencies are hosted.**

Delete the security group rule once the installation is complete. Use AWS Config to monitor the compliance is incorrect because this solution has a high operational overhead since the actions are done manually. This is susceptible to human error such as in the event that the DevOps team forgets to delete the security group. The use of AWS Config will just monitor and inform you about the security violation but it won't do anything to remediate the issue.

References:
https://docs.aws.amazon.com/vpc/latest/userguide/vpc-endpoints.html
https://docs.aws.amazon.com/vpc/latest/userguide/vpce-gateway.html

Check out this Amazon VPC Cheat Sheet:
https://tutorialsdojo.com/amazon-vpc/

Question 2

Due to the growth of its regional e-commerce website, the company has decided to expand its operations globally in the coming months ahead. The REST API web services of the app is currently running in an Auto Scaling group of EC2 instances across multiple Availability Zones behind an Application Load Balancer. For its database tier, the website is using a single Amazon Aurora MySQL database instance in the AWS Region where the company is based. The company wants to consolidate and store the data of its offerings into a single data source for its product catalog across all regions. For data privacy compliance, they need to ensure that the personal information of their users, as well as their purchases and financial data, are kept in their respective regions.

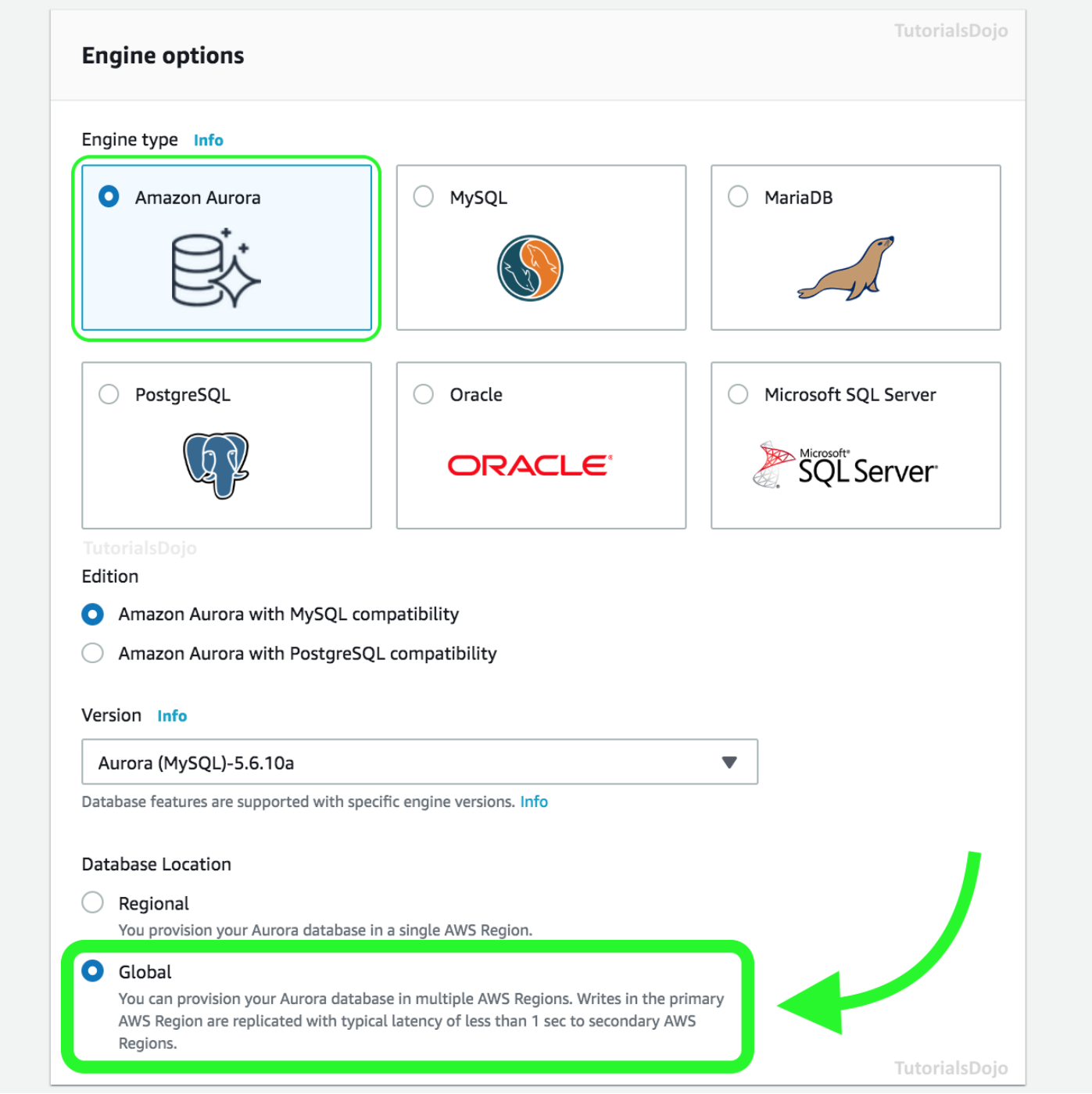
Which of the following options can meet the above requirements and entails the LEAST amount of change to the application?

1. Set up a new Amazon Redshift database to store the product catalog. Launch a new set of Amazon DynamoDB tables to store the personal information and financial data of their customers.
2. Set up a DynamoDB global table to store the product catalog data of the e-commerce website. Use regional DynamoDB tables for storing the personal information and financial data of their customers.
3. Set up multiple read replicas in your Amazon Aurora cluster to store the product catalog data. Launch an additional local Amazon Aurora instances in each AWS Region for storing the personal information and financial data of their customers.
4. Set up multiple read replicas in your Amazon Aurora cluster to store the product catalog data. Launch a new DynamoDB global table for storing the personal information and financial data of their customers.

Correct Answer: 3

An Aurora global database consists of one primary AWS Region where your data is mastered, and one read-only, secondary AWS Region. Aurora replicates data to the secondary AWS Region with typical latency of

under a second. You issue write operations directly to the primary DB instance in the primary AWS Region. An Aurora global database uses dedicated infrastructure to replicate your data, leaving database resources available entirely to serve application workloads. Applications with a worldwide footprint can use reader instances in the secondary AWS Region for low latency reads. In the unlikely event your database becomes degraded or isolated in an AWS region, you can promote the secondary AWS Region to take full read-write workloads in under a minute.



The Aurora cluster in the primary AWS Region where your data is mastered performs both read and write operations. The cluster in the secondary region enables low-latency reads. You can scale up the secondary cluster independently by adding one or more DB instances (Aurora Replicas) to serve read-only workloads. For disaster recovery, you can remove and promote the secondary cluster to allow full read and write operations.

Only the primary cluster performs write operations. Clients that perform write operations connect to the DB cluster endpoint of the primary cluster.

Hence, the correct answer is: **Set up multiple read replicas in your Amazon Aurora cluster to store the product catalog data. Launch an additional local Amazon Aurora instances in each AWS Region for storing the personal information and financial data of their customers.**

The option that says: **Set up a new Amazon Redshift database to store the product catalog. Launch a new set of Amazon DynamoDB tables to store the personal information and financial data of their customers** is incorrect because this solution entails a significant overhead of refactoring your application to use Redshift instead of Aurora. Moreover, Redshift is primarily used as a data warehouse solution and not suitable for OLTP or e-commerce websites.

The option that says: **Set up a DynamoDB global table to store the product catalog data of the e-commerce website. Use regional DynamoDB tables for storing the personal information and financial data of their customers** is incorrect because although the use of Global and Regional DynamoDB is acceptable, this solution still entails a lot of changes to the application. There is no assurance that the application can work with a NoSQL database and even so, you have to implement a series of code changes in order for this solution to work.

The option that says: **Set up multiple read replicas in your Amazon Aurora cluster to store the product catalog data. Launch a new DynamoDB global table for storing the personal information and financial data of their customers** is incorrect because although the use of Read Replicas is appropriate, this solution still requires you to do a lot of code changes since you will use a different database to store your regional data.

References:
https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/aurora-global-database.html#aurora-global-database.advantages
https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/AuroraMySQL.Replication.CrossRegion.html

Check out this Amazon Aurora Cheat Sheet:
https://tutorialsdojo.com/amazon-aurora/

Click **here** for more **AWS Certified DevOps Engineer Professional practice exam questions**.

More AWS reviewers can be found **here**:

At this point, you should already be very knowledgeable on the following domains:

1. CI/CD, Application Development and Automation
2. Configuration Management and Infrastructure as Code
3. Security, Monitoring and Logging
4. Incident Mitigation and Event Response
5. Implementing High Availability, Fault Tolerance, and Disaster Recovery

Additional Training Materials: A Few Video Courses on Udemy

There are a few AWS Certified DevOps Engineer - Professional video courses on Udemy that you can check out as well, which can complement your exam preparations especially if you are the type of person who can learn better through visual courses instead of reading long whitepapers:

1. AWS Certified DevOps Engineer - Professional by Zeal Vora

As an AWS DevOps practitioner, you shoulder a lot of roles and responsibilities. Many professionals in the industry have attained proficiency through continuous practice and producing results of value. Therefore, you should properly review all the concepts and details that you need to learn so that you can also achieve what others have achieved.

The day before your exam, be sure to double-check the schedule, location and the items to bring for your exam. During the exam itself, you have 180 minutes to answer all questions and recheck your answers. Be sure to manage your time wisely. It will also be very beneficial for you to review your notes before you go in to refresh your memory. The AWS DevOps Pro certification is very tough to pass, and the choices for each question can be very misleading if you do not read them carefully. Be sure to understand what is being asked in the questions, and what options are offered to you. With that, we wish you all the best in your exam!

Domain 1: Software Development Life Cycle (SDLC) Automation

Overview

The first domain of the AWS Certified DevOps Engineer Professional exam checks your preparedness on how well you understand the integration between the AWS services necessary for code development and deployment such as AWS CodeCommit, AWS CodeBuild, AWS CodeDeploy and AWS CodePipeline. You will also need a working knowledge on how they compliment each other for software development as well as integrations with CloudWatch Events, Amazon S3 and AWS Lambda. A big part of a normal work day for a DevOps Engineer deals with the software development life cycle. Roughly 22% of questions in the actual DevOps exam revolves around these topics.

This domain will challenge your know-how in doing the following:

- Apply concepts required to automate a CI/CD pipeline
- Determine source control strategies and how to implement them
- Apply concepts required to automate and integrate testing
- Apply concepts required to build and manage artifacts securely
- Determine deployment/delivery strategies (e.g., A/B, Blue/green, Canary, Red/black) and how to implement them using AWS Services

In this chapter, we will cover all of the related topics for SDLC automation in AWS that will likely show up in your DevOps Professional exam.

What is DevOps?

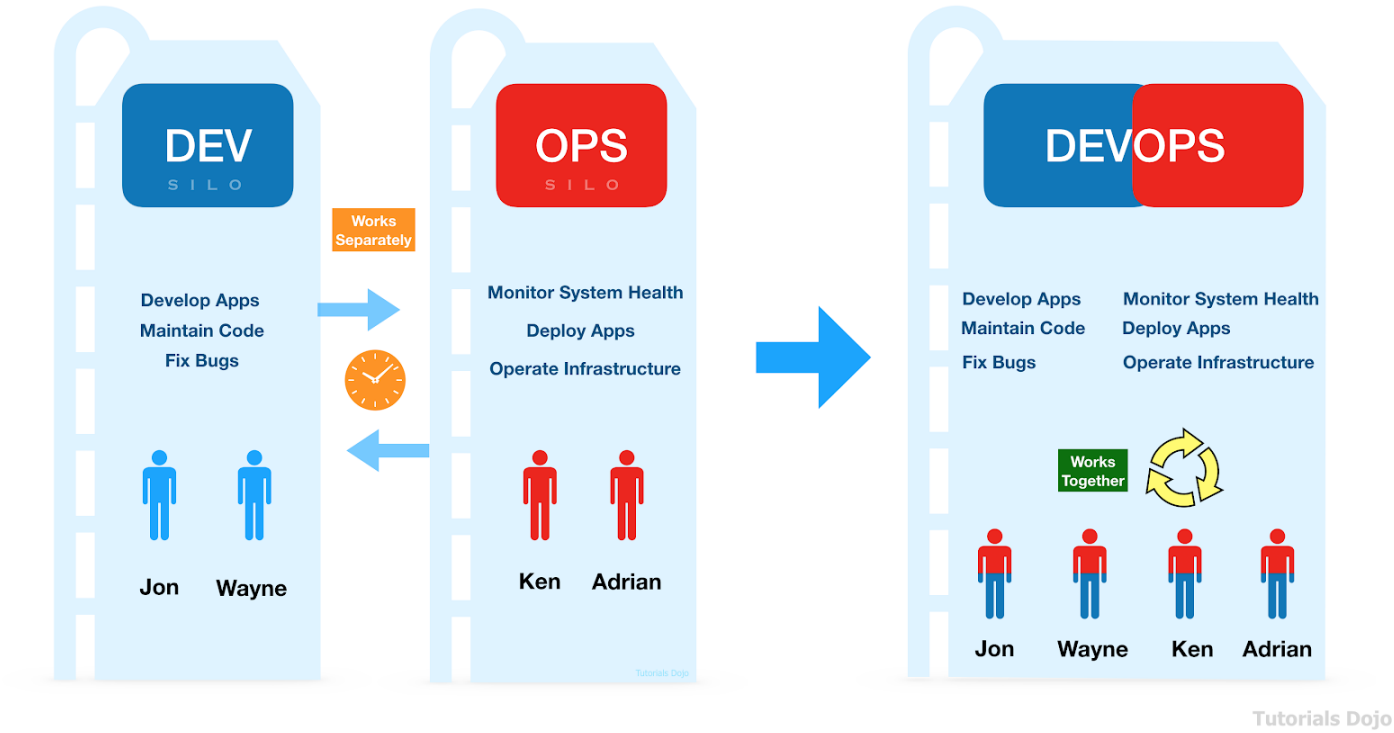
Do you ever wonder what DevOps is and why it is so popular in the IT Industry today? Is it a tool, a process, a corporate culture, or a combination of all of these? Why are companies giving competitive salaries for this kind of role?

If you typed the word “DevOps” in any job market website today, you would see many available positions that require knowledge on both programming and infrastructure management. You will usually see an advertisement looking for a candidate who knows how to program in Python or any other language. The requirements include being capable of managing servers with configuration management tools such as Ansible, Chef, or Puppet as well as the provisioning of entire cloud environments using Infrastructure-as-code tools like Terraform or CloudFormation. The salary range offered for these positions is remarkably high too!

Traditional IT companies have a dedicated Development (Dev) team that builds enterprise applications and an Operations (Ops) team that handles the servers and network infrastructure. These two teams are often siloed or isolated from each other. While the Dev team writes the software code, the Ops team prepares the server, database, and other infrastructure needed to run the soon-to-be-released application. In this setup, the developers are entirely oblivious to what the system operators are doing and vice versa. A lot of time is wasted waiting for the Dev Team to fix minor bugs while developing new features and for the Ops team to provision, deploy and scale the needed server resources. When bugs and incompatibility issues are detected in the development cycle, the Ops team waits for the Dev team to address the issue since it is strictly the job of Developers to fix it. Same is true when there are issues during deployments when the Ops are not familiar with the application and make wrong assumptions which can cause further delays in the deployment targets. Due to this lack of coordination, it impacts both the business and its customers. This is where DevOps comes in!

DevOps is not just the combination of Development (Dev) and Operations (Ops). DevOps is the fusion of practices, processes, tools, and corporate culture that expedite the organization’s ability to deliver applications and services at a higher velocity, faster than traditional software development processes. It’s not merely a tool or a process that your team adopts, but a synergy of values, corporate structure, and internal processes to attain the digital transformation of the business enterprise. It tears down the traditional and isolated silos of the Development, Operations, IT Security, and other teams, enabling collaboration and improving overall business performance. With DevOps, Developers are empowered to directly influence the deployment life cycle and the IT Operations folks have the ability to report and fix possible bugs or incompatibilities in the application.

DevOps is not just a framework, rather, it’s a cultural approach and a mindset combining operations and development skills, and delivering a product (or service) from inception to retirement. Company executives also play a crucial role in allocating budgets and adopting this new status quo within their respective organizations.



With the advent of Cloud Computing, companies can easily unify their software development and system operation processes. AWS enables organizations to rapidly build, deliver, and manage their products, following DevOps practices with just a click of a button. The efficiency of provisioning new resources, managing infrastructure, deploying application code, automating software release processes, and many other tasks in AWS contribute to the overall productivity and business profitability. Because of this massive benefit, companies are willing to pay competitive remuneration for their DevOps Engineers, especially those who are AWS Certified.

A brief history of the DevOps Exam in AWS

In 2013, Amazon Web Services (AWS) began the Global Certification Program to validate the technical skills and knowledge for building secure and reliable cloud-based applications using the AWS platform. The first-ever certification launched by Amazon is the AWS Certified Solutions Architect – Associate, followed by SysOps Administrator and Developer Associate. A year later, AWS released the first Professional-level certification: AWS Certified Solutions Architect - Professional, and in February 2015, they released the AWS Certified DevOps Engineer Professional.

The AWS DevOps Engineer Professional certification enables technology professionals to showcase their DevOps skills, and it allows companies to identify top candidates to lead their internal DevOps initiatives. It validates your technical expertise in provisioning, managing, and operating distributed application systems on the AWS Cloud platform. It tests your ability to implement and manage Continuous Integration/Continuous Delivery (CI/CD) systems and methodologies on AWS following industry's best practices, as well as to automate security controls and handle governance processes and meet compliance. The exam also covers core topics such as Software Development Lifecycle (SDLC) automation, monitoring, logging, configuration management, and incident/event response.

As Amazon Web Services continue to evolve, new and updated versions of the AWS certification exams are released regularly to reflect the service changes and to include new knowledge areas. After four years since its initial release, an updated version of the AWS Certified DevOps Engineer - Professional certification was launched in February 2019 with an exam code of DOP-C01.

Why Automate?

Automation is at the heart of every DevOps engineer. Automation is a key highlight of DevOps practice. There is a saying in the DevOps community to “Automate Everything” and it starts from code inception, releasing to production, application retirement, and everything in between. Eliminating repetitive tasks, reducing toil, and minimizing manual work are the key aspects that you want to solve through automation. Automation in DevOps fosters speed, greater accuracy, consistency, reliability, and rapid delivery.

Here are some major benefits of automation:

- **Speed –** to innovate your product faster and adapt to changing markets trends. Team members are empowered to make changes quickly as needed either in the development side or the operational side.
- **Rapid delivery –** increase the pace of your releases by automating your entire deployment pipeline. This is the concept of “fail-fast, iterate faster” in which you companies are incentivized to release minor changes as often as possible which keeps them on top of competitors.
- **Reliability –** continuous integration and continuous delivery processes allows you to reliably and consistently deliver your product to end-users. This also reduces human error as the automation rarely makes mistakes like humans do.
- **Scale** - infrastructure as code helps you manage your environments in a repeatable and more efficient manner and scale easily as needed. It gives you a robust system to manage your infrastructure no matter how big or small it is.
- **Improved collaboration –** reduce inefficiencies when collaborating with teams. Automation allows the easier integration of development, testing and deployment processes. It facilitates faster collaboration between Dev and Ops, which results in an improved turnaround time for bug fixing, deployment, etc.
- **Security -** reduces risk through integrated security testing tools, automating adoption of compliance requirements. It allows you to declare and script your security compliance requirement and make sure they are applied to needed resources in your environments.

Types of Blue Green Deployment via ELB, Route 53, Elastic Beanstalk

AWS Elastic Beanstalk Blue/Green

Elastic Beanstalk, by default, performs an in-place update when you deploy a newer version of your application. This can cause a short downtime since your application will be stopped while Elastic Beanstalk performs the application update.

Blue/Green deployments allow you to deploy without application downtime.

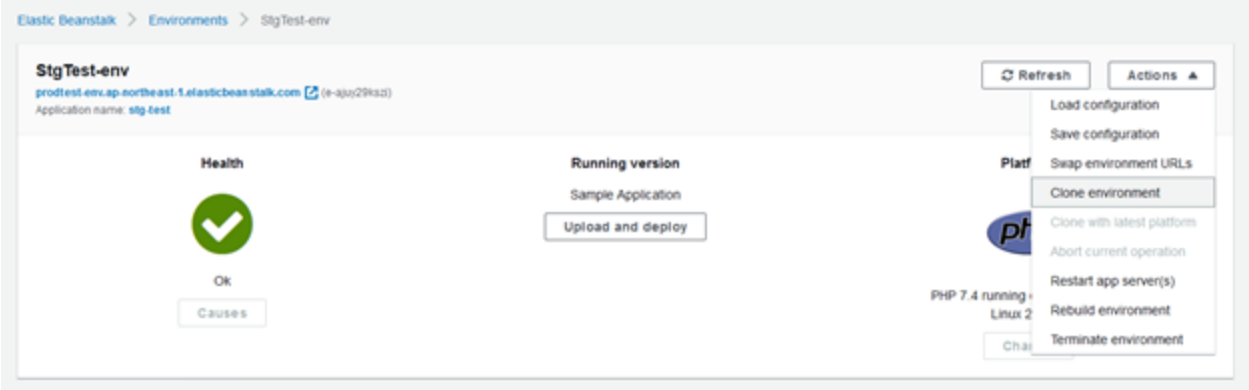
DevOps Exam Notes:

Remember these key points on when to use blue/green deployments:

- No downtime during deployment because you are deploying the newer version on a separate environment
- CNAMEs of the environment URLs are swapped to redirect traffic to the newer version.
- Route 53 will swap the CNAMEs of the application endpoints.
- Fast deployment time and quick rollback since both old and new versions are running at the same time, you just have to swap back the URLs if you need to rollback.
- Useful if your newer version is incompatible with the current platform version of your application. (ex. Jumping from major versions of NodeJS, Python, Ruby, PHP, etc.)
- Your RDS Database instance should be on a separate stack because the data will not transfer to your second environment. You should decouple your database from the web server stack.

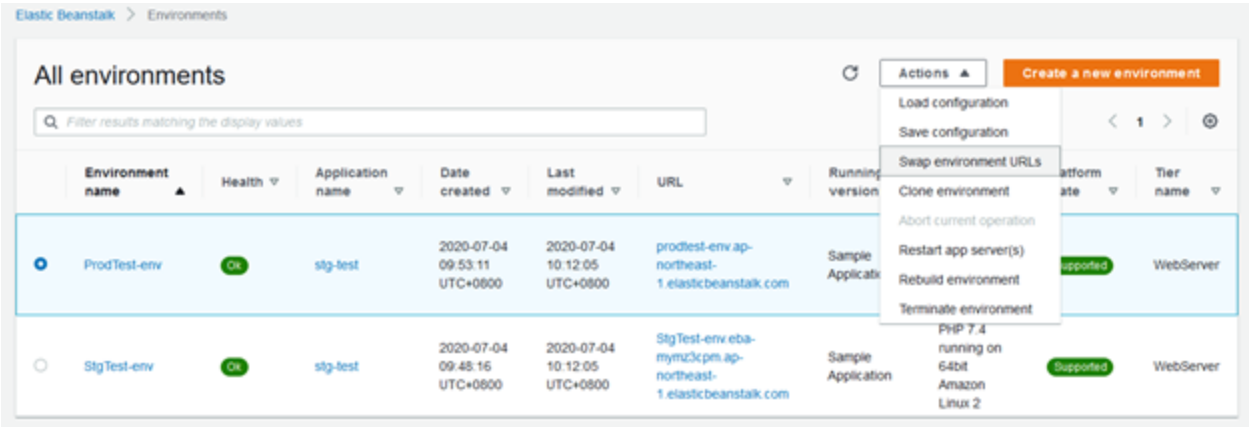
To implement a Blue/Green deployment for your Elastic Beanstalk application, you can perform the following steps:

1. Create another environment on which you will deploy the newer version of your application. You can clone your current environment for easier creation.

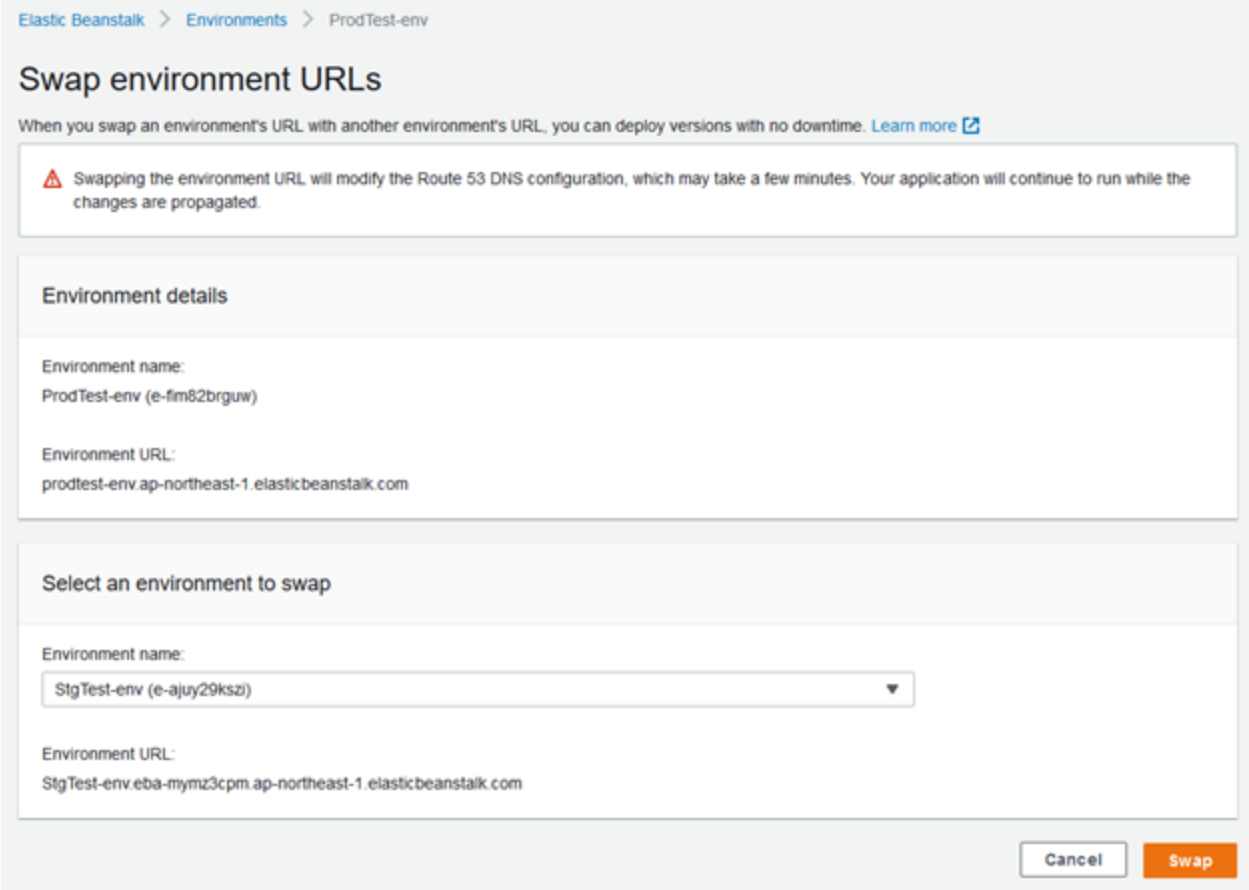


2. Once the new environment is ready, deploy a new version of your application. Perform your tests on the URL endpoint of your new environment.

3. After testing, select your Production environment, click Actions > Swap environment URLs.



4. On the Swap Environment URLs page, select the newer environment and click Swap to apply the changes.


Elastic Beanstalk
Environments
ProdTest-env
Swap environment URLs
When you swap an environment's URL with another environment's URL, you can deploy versions with no downtime. Learn more
Swapping the environment URL will modify the Route 53 DNS configuration, which may take a few minutes. Your application will continue to run while the changes are propagated.
Environment details
Environment name:
ProdTest-env (e-fim82brguw)
Environment URL:
prodtest-env.ap-northeast-1.elasticbeanstalk.com
Select an environment to swap
Environment name:
StgTest-env (e-ajuy29kszi)
Environment URL:
StgTest-env.eba-mymz3cpm.ap-northeast-1.elasticbeanstalk.com
Cancel
Swap

AWS Lambda Blue/Green

You can also implement Blue/Green deployments on your Lambda functions. The concept is the same as in Elastic beanstalk blue/green deployment i.e. you will need to create two versions of your Lambda function and use function Aliases to swap the traffic flow.

Lambda versions – lets you publish a new version of a function that you can test without affecting the current application accessed by users. You can create multiple versions as needed for your testing environments. The ARN of Lambda version is the same as the ARN of the Lambda function with added version suffix.

arn:aws:lambda:aws-region:acct-id:function:helloworld:$LATEST

Lambda aliases – Aliases are merely pointers to specific Lambda versions. You can't select a Lambda alias and edit the function. You need to select the LATEST version if you want to edit the function. Aliases are helpful for blue/green deployments because it allows you to use a fixed ARN and point it to a particular Lambda version that you want to deploy.

DevOps Exam Notes:

Remember the difference between Lambda $LATEST, Lambda Versions and Lambda Aliases:

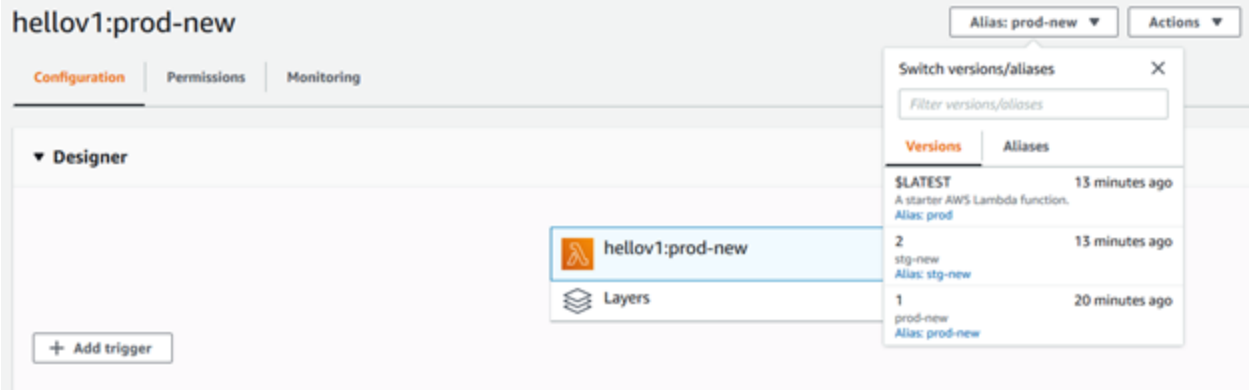
$LATEST - this is the latest version of your Lambda function. You can freely edit this version.

Lambda Version - fixed version of your function. You can't edit this directly.

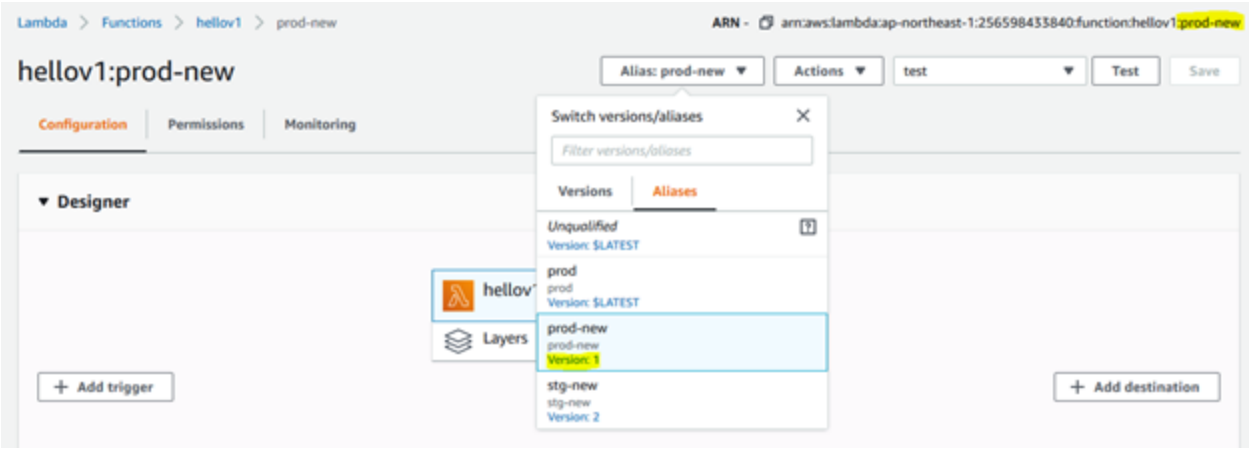
Lambda Alias - a pointer to a specific Lambda version. You can perform blue/green deployment with Aliases by pointing to a newer version.

The following steps will show how blue/green deployment can be done on Lambda functions.

1. The current version of your Lambda function is deployed on Version 1. Create another version and make your changes, this will be Version 2.



2. Create an Alias that will point to the current production version. Use this alias as your fixed production ARN.



3. Create another Alias that you will use for your newer version. Perform your testing and validation on this newer version. Once testing is complete, edit the production alias to point to the newer version. Traffic will now instantly be shifted from the previous version to the newer version.

Sources:
https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/using-features.CNAMESwap.html
https://docs.aws.amazon.com/lambda/latest/dg/configuration-aliases.html
https://docs.aws.amazon.com/lambda/latest/dg/configuration-versions.html

AWS Lambda Function Alias Traffic Shifting

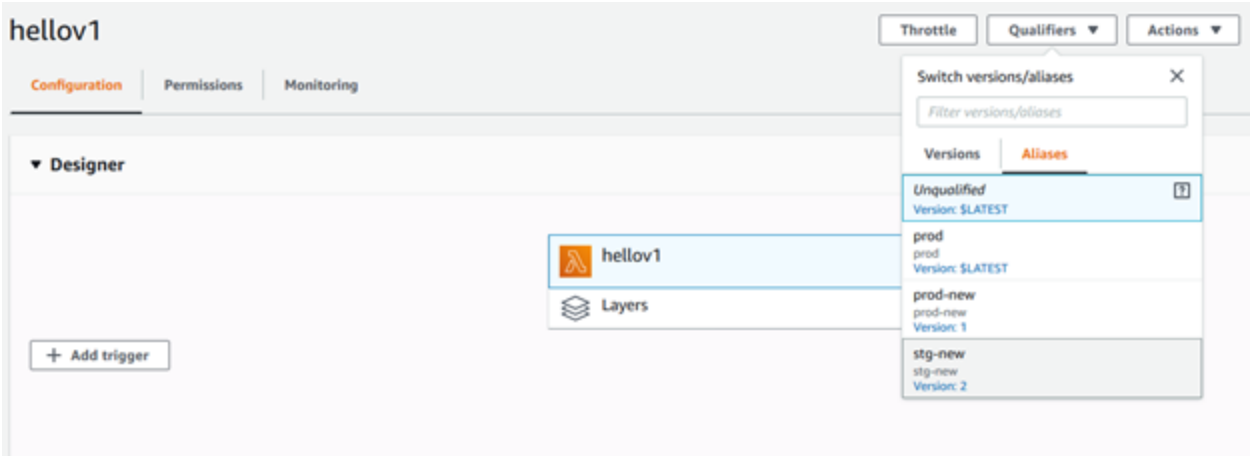
On EC2 instances, you can perform a canary deployment by deploying a newer application version on a single EC2 instance and analyzing the production traffic flowing through it. If you are satisfied with it, you'll proceed to deploy the newer version on all EC2 instances. However, for deployments on your Lambda functions, you can't use a canary deployment since you don't deploy your application directly on EC2 instances.

DevOps Exam Notes:

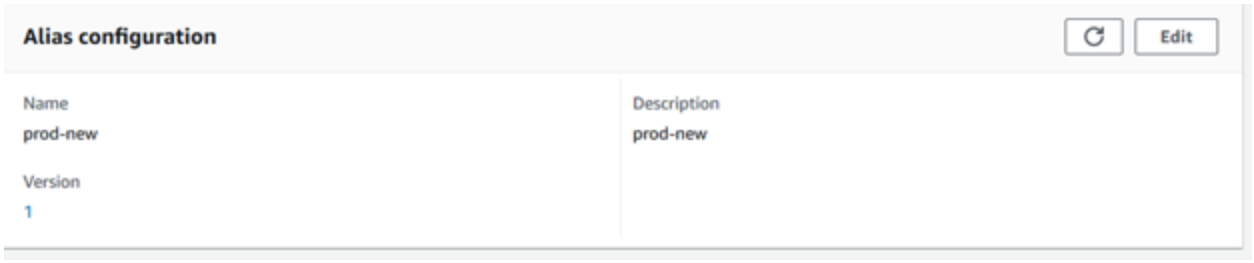
To provide similar functionality as a canary deployment, AWS Lambda gives you the ability to use Function Aliases to shift the percentage of traffic from one version to another. Essentially, you will create an Alias that points to the current version of the Lambda function, then use a weighted alias to define a newer version of the Lambda function. You can then define the weight (percent of traffic) that you want to forward to this version. After validation, you can completely shift traffic to the newer version.

You can consult the previous section (Types of Blue/Green deployment - AWS Lambda) on how to create AWS Lambda Versions and Aliases. Here's an example of how to control that percentage of traffic flowing to different Lambda functions using function alias. This is similar to the way a canary deployment works.

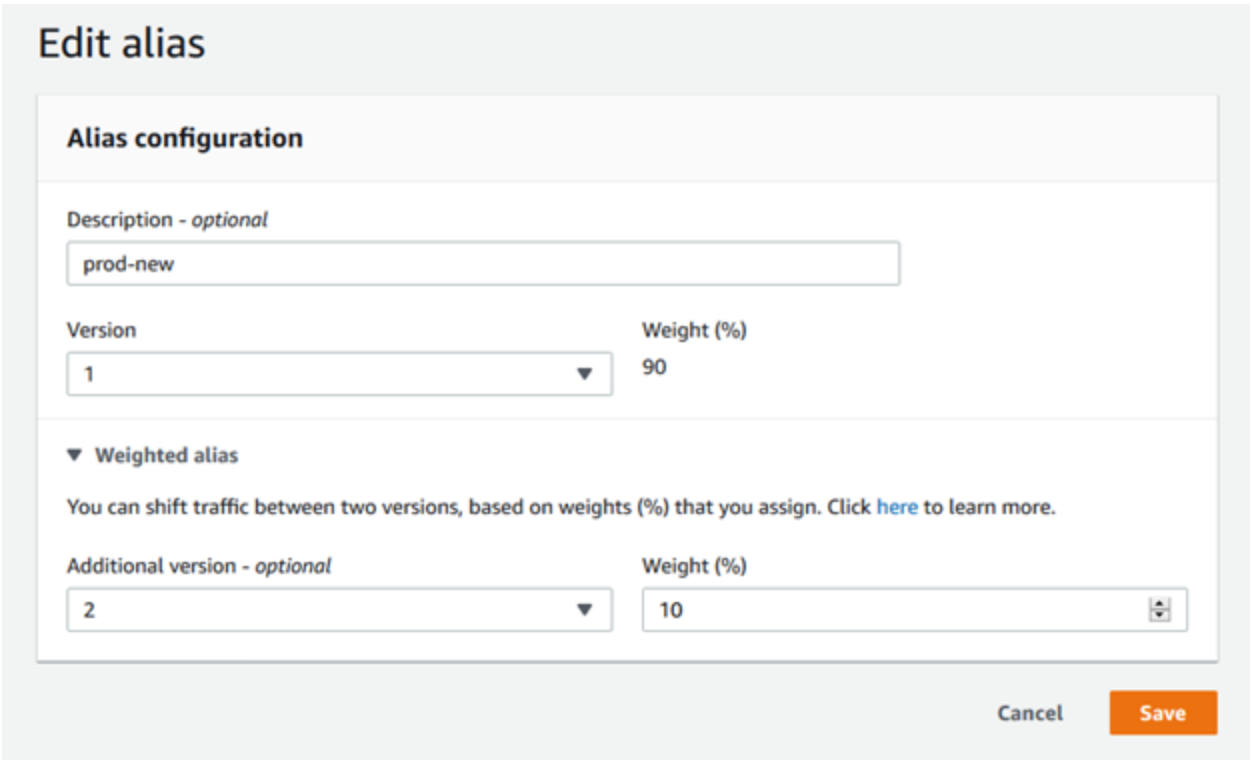
1. Select the function alias pointing to the current production version.



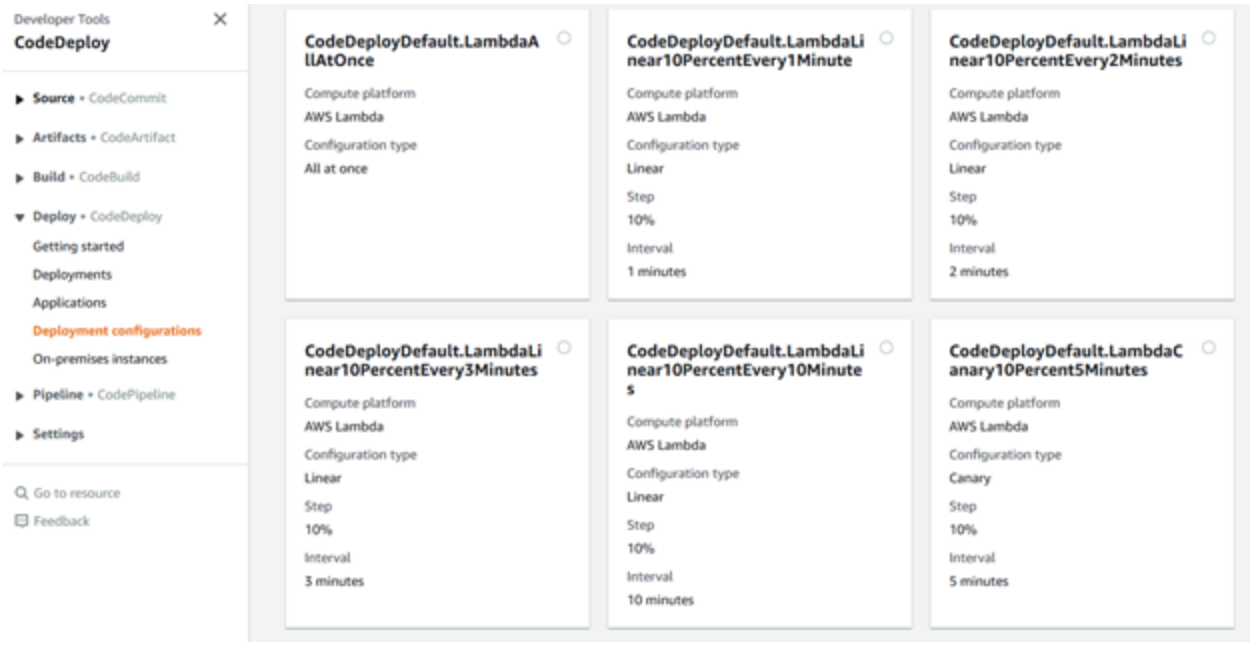
2. Edit the alias configuration.



3. On the weighted alias section, select the newer version of your Lambda function and assign the percentage of traffic to shift to the newer version. You can repeat this step multiple times if you want to slowly shift traffic from the older version to the newer version.



If you are using AWS CodeDeploy to deploy your Lambda functions, CodeDeploy uses Aliases to shift traffic to the newer version. As you can see on the options on the deployment configurations, CodeDeploy can automate this gradual traffic shifting for your Lambda Functions.

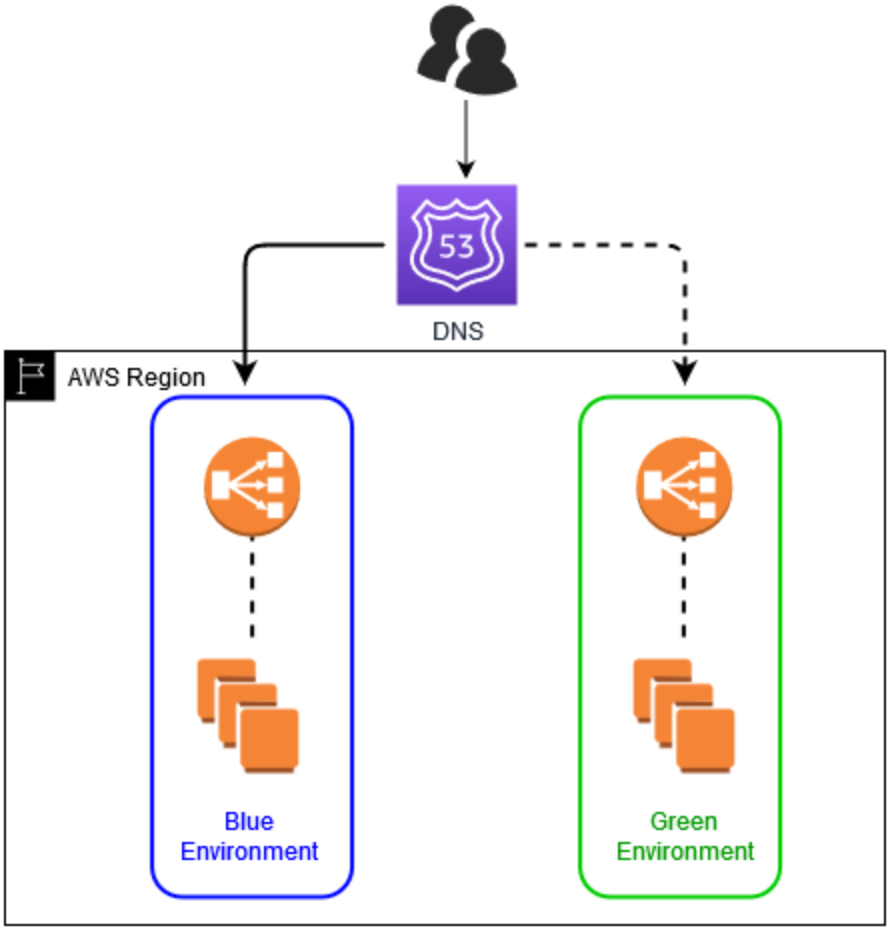


See the CodeDeploy - Linear, Canary and All-at-Once (Pre-defined Deployments) topic for more discussion on these deployment strategies.

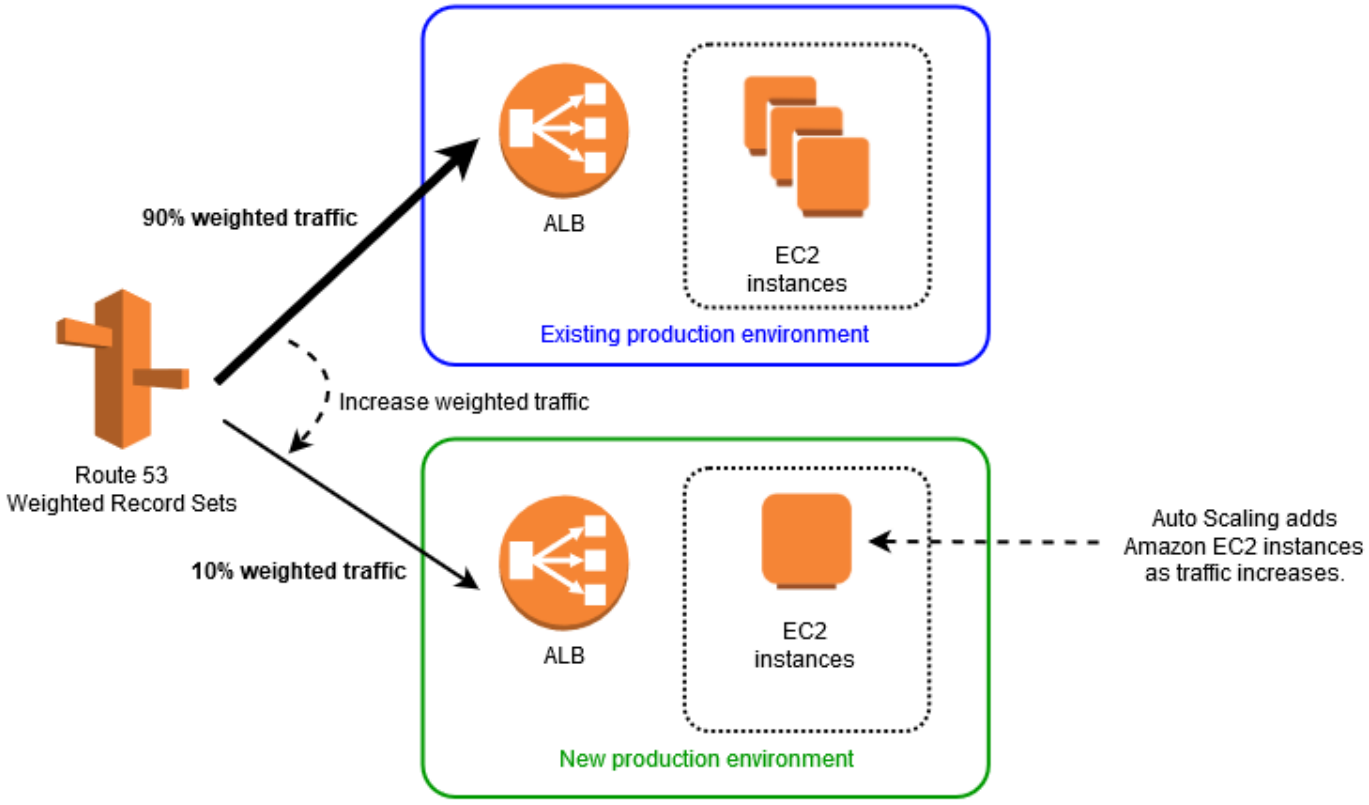
Source:
https://aws.amazon.com/blogs/compute/implementing-canary-deployments-of-aws-lambda-functions-with-alias-traffic-shifting/

Basic Blue/Green Deployment using Route 53

Blue/green deployment on the AWS platform provides a safer way to upgrade production software. This deployment usually involves two environments, the production environment (blue) and the new updated environment (green).



Once the new version is deployed on the green environment, you can validate the new software before going live. Then, you start shifting traffic away from the blue environment and sending it to the green one. Normally, you'd use Route 53 weighted routing policy because it gives you an easy way to push incremental traffic to the green environment or revert traffic back to the blue environment in case of issues. If you want to, you can switch the traffic immediately by updating the production Route 53 record to point to the green endpoint. Users will not see that you changed the endpoint since from their perspective, the production URL is the same.



You can also shift a small portion (like 10%) of traffic on the Green environment by using a weighted routing policy on Route 53. This way, you can test live traffic on the new environment, analyze the new logs, and then you can easily revert to the original environment if you find any problems. This process is also called a canary deployment.

Source:
https://aws.amazon.com/blogs/startups/upgrades-without-tears-part-2-bluegreen-deployment-step-by-step-on-aws/

Using a Cloned Stack in OpsWorks for Implementing Blue/Green Deployment

In AWS OpsWorks, you can create stacks which represent logical groupings of your AWS resources such as EC2 instances group, database group, or load balancer group. These stacks can be made of one or more layers in which the layer represents how the actual resource should be created.

DevOps Exam Notes:

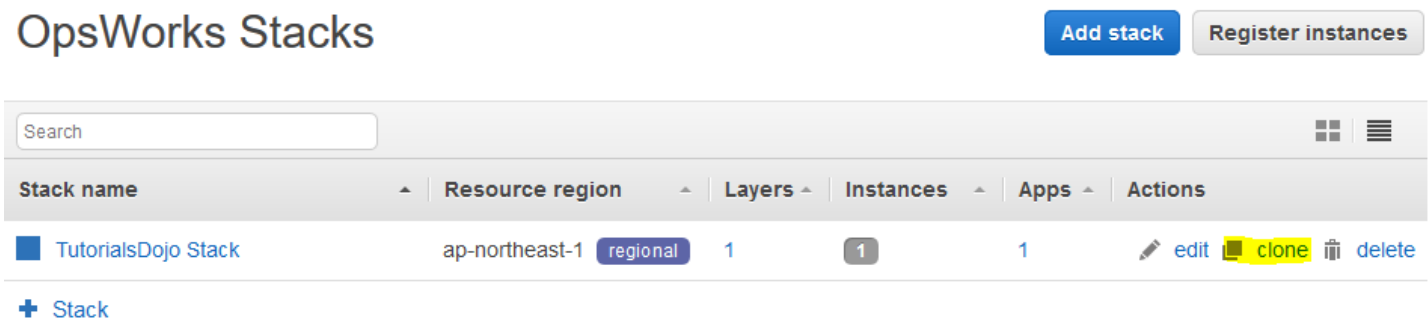
You can implement a blue/green deployment strategy with OpsWorks stacks, which allows you to run another version of your application – blue and green environment. A blue-green deployment strategy is one common way to efficiently use separate stacks to deploy an application update to production.

In a nutshell, you will clone your current OpsWorks stack and then deploy a new version on the cloned stack. Then you will use Route 53 to point the users to the new stack URL.

Here's the setup for a blue/green deployment on OpsWorks Stacks:

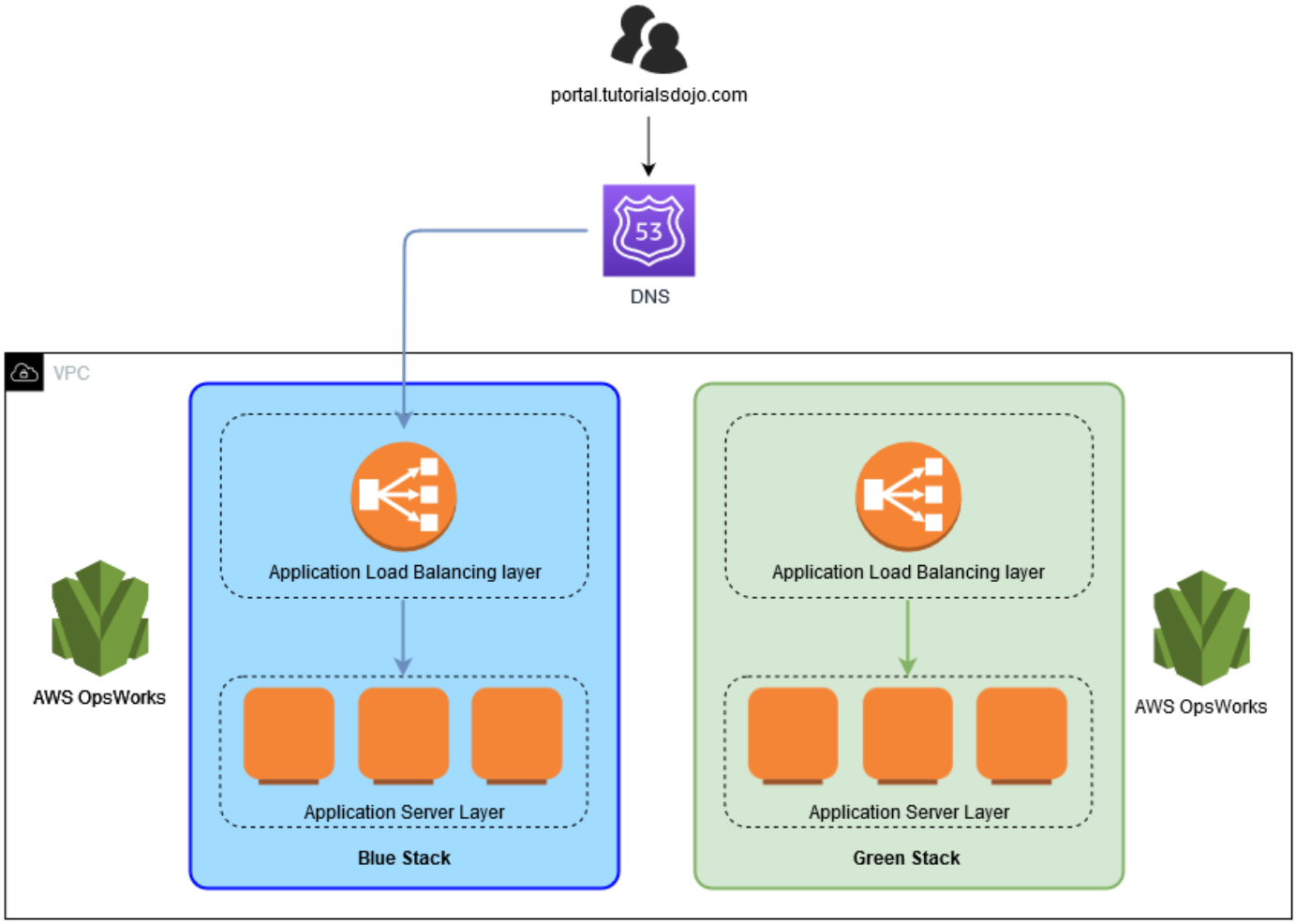
- The blue environment is the production stack, which hosts the current application.
- The green environment is the staging stack, which hosts the updated application.
- When you are ready to deploy the updated app to production, you switch user traffic from the blue stack to the green stack, which becomes the new production stack. You then retire the old blue stack.

Here's how to do it in OpsWorks. First, you have to clone your current production OpsWorks stack (blue environment). On the AWS OpsWorks Stacks dashboard, in the **Actions** column of the row for the stack that you want to clone, choose **clone**, which opens the **Clone stack** page.

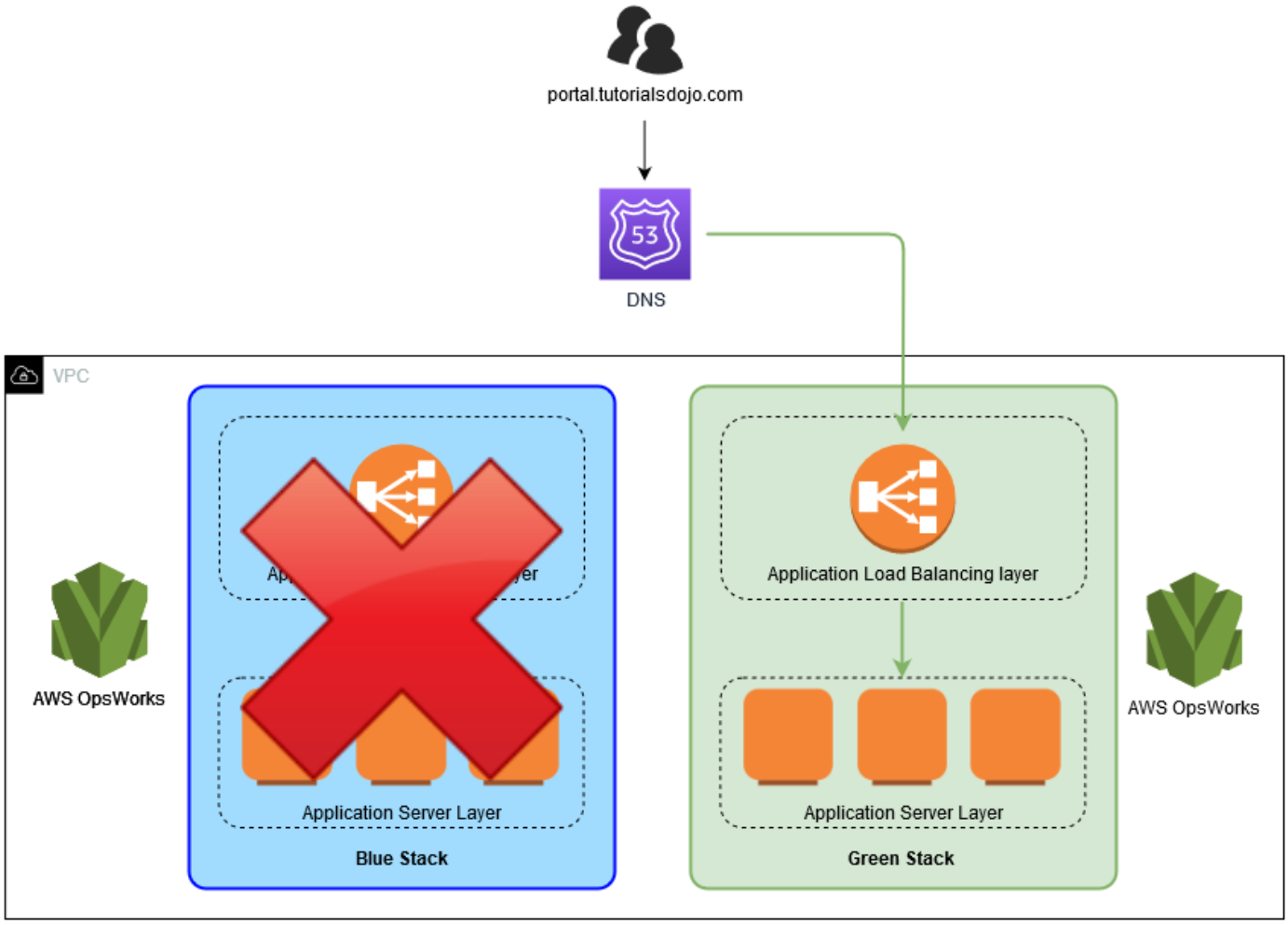


AWS OpsWorks Stacks creates a new stack that consists of the source stack's layers and optionally, its apps and permissions. The layers have the same configuration as the originals, subject to any modifications that you

made. When cloning, you can choose to deploy a newer version of the application or use a newer AMI for the EC2 instances on the application server layer.



You should run tests by now on the green environment to validate that the new application version is working as expected. When it's time to promote the green environment/stack into production, update DNS records to point to the green environment/stack's load balancer DNS name. You can also do this DNS flip gradually by using the Amazon Route 53 weighted routing policy.



Traffic will now be redirected to the new OpsWorks stack and you can tear down the older stack.

Sources:
https://d1.awsstatic.com/whitepapers/AWS_Blue_Green_Deployments.pdf
https://docs.aws.amazon.com/opsworks/latest/userguide/best-deploy.html
https://docs.aws.amazon.com/opsworks/latest/userguide/workingstacks-cloning.html

AWSCodeCommitFullAccess, AWSCodeCommitPowerUser, AWSCodeCommitReadOnly - Permissions

AWS CodeCommit can be used to collaborate code deployment with several teams. Access to AWS CodeCommit requires credentials. And those credentials need to have specific permissions on the level of access allowed for each individual or team.

For example, you have three teams – Admin team (merge requests and approves production deployments, create/delete repositories), Development team (handles code development on their respective branches), and Reviewers (reviews code changes on the repository).

Devops Exam Notes:

AWS has predefined policies for common use cases such as these groups. Going to the exam, you need to know the key differences between each policy.

- **AWSCodeCommitFullAccess** - full Admin policy for CodeCommit
- **AWSCodeCommitPowerUser** - users can't create or delete CodeCommit repositories
- **AWSCodeCommitReadOnly** - read-only access for users

AWSCodeCommitFullAccess – Grants full access to CodeCommit. Apply this policy only to administrative-level users to whom you want to grant full control over CodeCommit repositories and related resources in your AWS account, including the ability to delete repositories. You can assign this to your Admin Team. This also allows the group to create and manage CloudWatch Events for the repositories, which is helpful if the Admin group wants to know who pushes code on the branches.

AWSCodeCommitPowerUser – Allows users access to all of the functionality of CodeCommit and repository-related resources, except that it does not allow them to delete CodeCommit repositories nor create or delete repository-related resources in other AWS services, such as Amazon CloudWatch Events. You can apply this policy to the Development Team so each member can work on their tasks and freely push code to their respective branches without worrying that they may accidentally delete important repositories or branches.

AWSCodeCommitReadOnly – Grants read-only access to CodeCommit and repository-related resources in other AWS services, as well as the ability to create and manage their own CodeCommit-related resources (such as Git credentials and SSH keys for their IAM user to use when accessing repositories). You can apply this to the Reviewers Team so that they can view review changes on the repositories but not make any changes to its contents.

Sources:
https://docs.aws.amazon.com/codecommit/latest/userguide/auth-and-access-control.html#managing-access-resources
https://docs.aws.amazon.com/codecommit/latest/userguide/auth-and-access-control-iam-identity-based-access-control.html

Lifecycle Event Hook Availability (CodeDeploy Concept)

AWS CodeDeploy gives you several lifecycle event hooks to perform actions on different stages of the deployment. You can run scripts on your desired stage to run specific actions needed for that stage.

For example, you can use the **BeforeInstall** lifecycle event hook to create a backup of your current version before CodeDeploy installs the new version on your instances. You can use the **BeforeAllowTraffic** lifecycle event hook to perform some tasks or run scripts on the instances before they are registered on the load balancer.

Another example is when using blue/green deployment and you want to run validation scripts after the new instances have been registered on the load balancer. You want to validate the new version before you remove the old version instances. For this scenario, you will use the **AfterAllowTraffic** lifecycle event hook.

The stages are available depending on which deployment method you have chosen such as in-place deployment or blue-green deployments.

The following table lists the lifecycle event hooks available for each deployment and rollback scenario.

Lifecycle event name	In-place deployment[1]	Blue/green deployment: Original instances	Blue/green deployment: Replacement instances	Blue/green deployment rollback: Original instances	Blue/green deployment rollback: Replacement instances
ApplicationStop	✓		✓		
DownloadBundle[2]	✓		✓		
BeforeInstall	✓		✓		
Install[2]	✓		✓		
AfterInstall	✓		✓		
ApplicationStart	✓		✓		
ValidateService	✓		✓		
BeforeBlockTraffic	✓	✓			✓
BlockTraffic[2]	✓	✓			✓
AfterBlockTraffic	✓	✓			✓

BeforeAllowTraffic	✓		✓	✓	
AllowTraffic[2]	✓		✓	✓	
AfterAllowTraffic	✓		✓	✓	

[1]Also applies to the rollback of an in-place deployment.

[2] Reserved for CodeDeploy operations. Cannot be used to run scripts.

Here's a summary of the Lifecycle event hooks and what actions are performed on that stage.

ApplicationStop – This deployment lifecycle event occurs even before the application revision is downloaded.

DownloadBundle – During this deployment lifecycle event, the CodeDeploy agent copies the application revision files to a temporary location.

BeforeInstall – You can use this deployment lifecycle event for preinstall tasks, such as decrypting files and creating a backup of the current version.

Install – During this deployment lifecycle event, the CodeDeploy agent copies the revision files from the temporary location to the final destination folder.

AfterInstall – You can use this deployment lifecycle event for tasks such as configuring your application or changing file permissions.

ApplicationStart – You typically use this deployment lifecycle event to restart services that were stopped during ApplicationStop.

ValidateService – This is the last deployment lifecycle event. It is used to verify that the deployment was completed successfully.

BeforeBlockTraffic – You can use this deployment lifecycle event to run tasks on instances before they are deregistered from a load balancer.

BlockTraffic – During this deployment lifecycle event, internet traffic is blocked from accessing instances that are currently serving traffic.

AfterBlockTraffic – You can use this deployment lifecycle event to run tasks on instances after they are deregistered from a load balancer.

BeforeAllowTraffic – You can use this deployment lifecycle event to run tasks on instances before they are registered with a load balancer.

AllowTraffic – During this deployment lifecycle event, internet traffic is allowed to access instances after a deployment.

AfterAllowTraffic – You can use this deployment lifecycle event to run tasks on instances after they are registered with a load balancer.

Going to the exam, you don't have to remember all stages but you need to know the important lifecycle hooks such as **BeforeInstall**, **BeforeAllowTraffic**, and **AfterAllowTraffic**.

Here's an example structure of a "hooks" section on your YAML file:

```
hooks:
   deployment-lifecycle-event-name:
      - location: script-location
        timeout: timeout-in-seconds
        runas: user-name
```

Source:
https://docs.aws.amazon.com/codedeploy/latest/userguide/reference-appspec-file-structure-hooks.html#reference-appspec-file-structure-hooks-availability

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

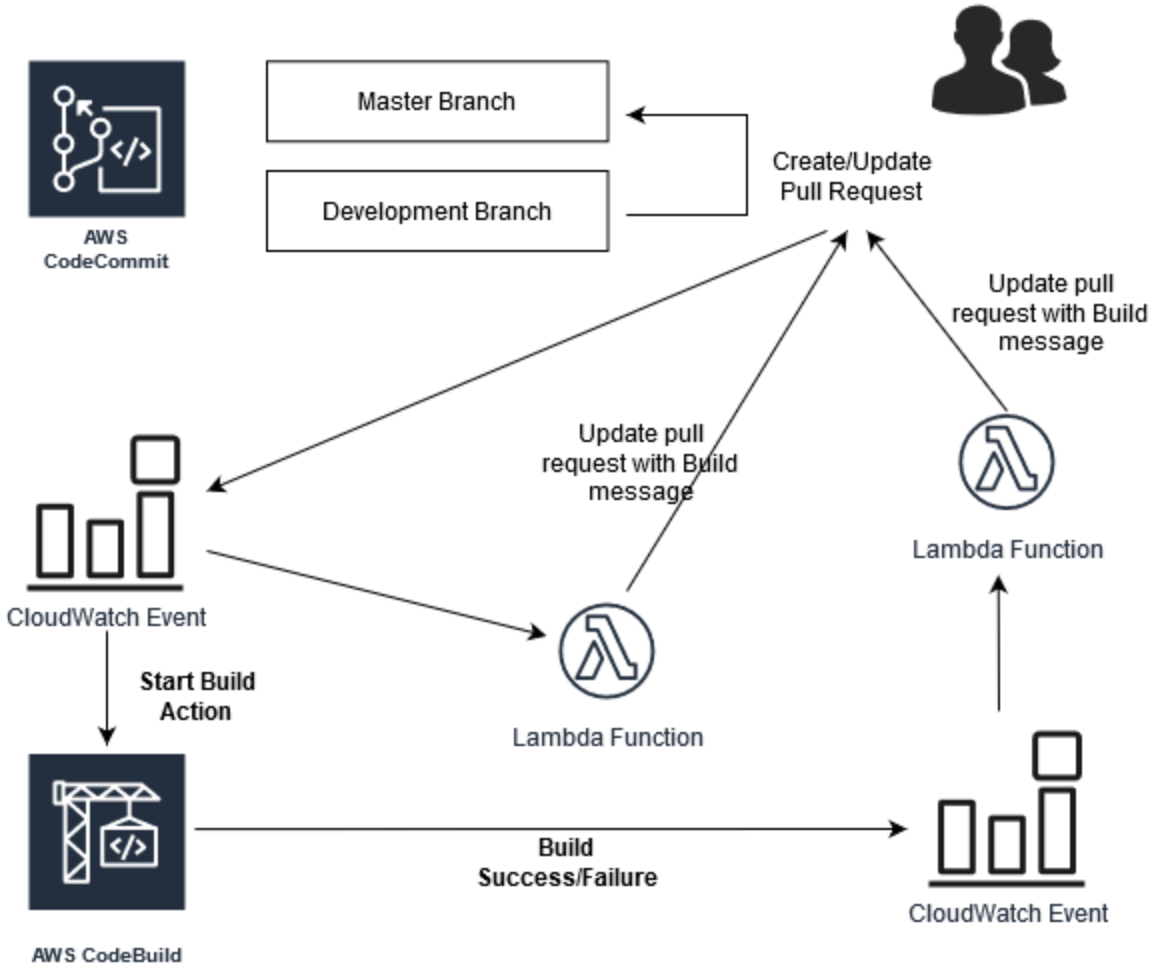
AWS CodeCommit allows developers to create multiple branches, which can be used for developing new features or fixing application bugs. These changes can then be merged on the master branch, which is usually used for the production release. To merge the changes to the master branch on CodeCommit, the 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/

Managing Artifacts in AWS CodeBuild and CodePipeline

In AWS CodeBuild, you can run build projects that will build and test your code. After the build process you have the option to store the artifact on an AWS S3 bucket which can then be used by AWS CodeDeploy to deploy on your instances. You can create an AWS Pipeline for the whole process to be automated – from building, testing, validation, up to deployment of your artifact.

Here's the reference snippet of code on your CodeBuild buildspec.yml file showing how to declare the output artifact file.

```
phases:
  install:
      runtime-versions:
      java: corretto11
  pre_build:
      commands:
      - echo Nothing to do in the pre_build phase...
  build:
      commands:
      - echo Build started on `date`
      - mvn install
  post_build:
      commands:
      - echo Build completed on `date`
artifacts:
  files:
      - target/CodeDeploySample.zip
```

The **buildspec.yml** file should be in the same folder as your source code. The source code will be built based on the contents of the **buildspec.yml** file and the output will be sent to the S3 bucket that you have specified on the build project.

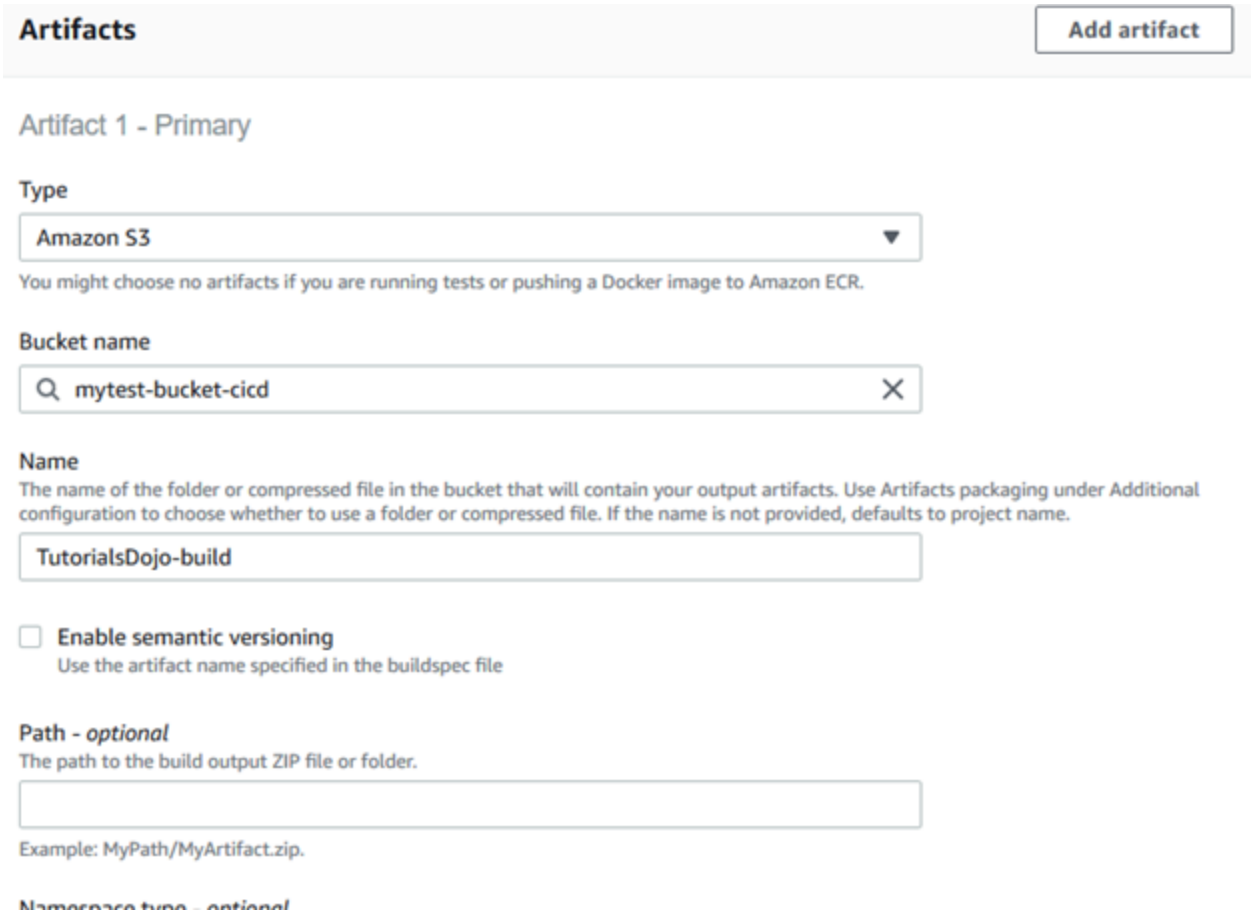
This is an example of a JSON file that you can use when creating a CodeBuild project. Notice that the input and output buckets are specified.

```
{

  "name": "sample-codedeploy-project",
  "source": {
      "type": "S3",
      "location": "my-codepipeline-website-bucket/CodeDeploySample.zip"
  },
```

```
  "artifacts": {
      "type": "S3",
      "location": "my-codepipeline-website-bucket",
      "packaging": "ZIP",
      "name": "CodeDeployOutputArtifact.zip"
  },
  "environment": {
      "type": "LINUX_CONTAINER",
      "image": "aws/codebuild/standard:4.0",
      "computeType": "BUILD_GENERAL1_SMALL"
  },
  "serviceRole": "arn:aws:iam::account-ID:role/role-name",
  "encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID"
}
```

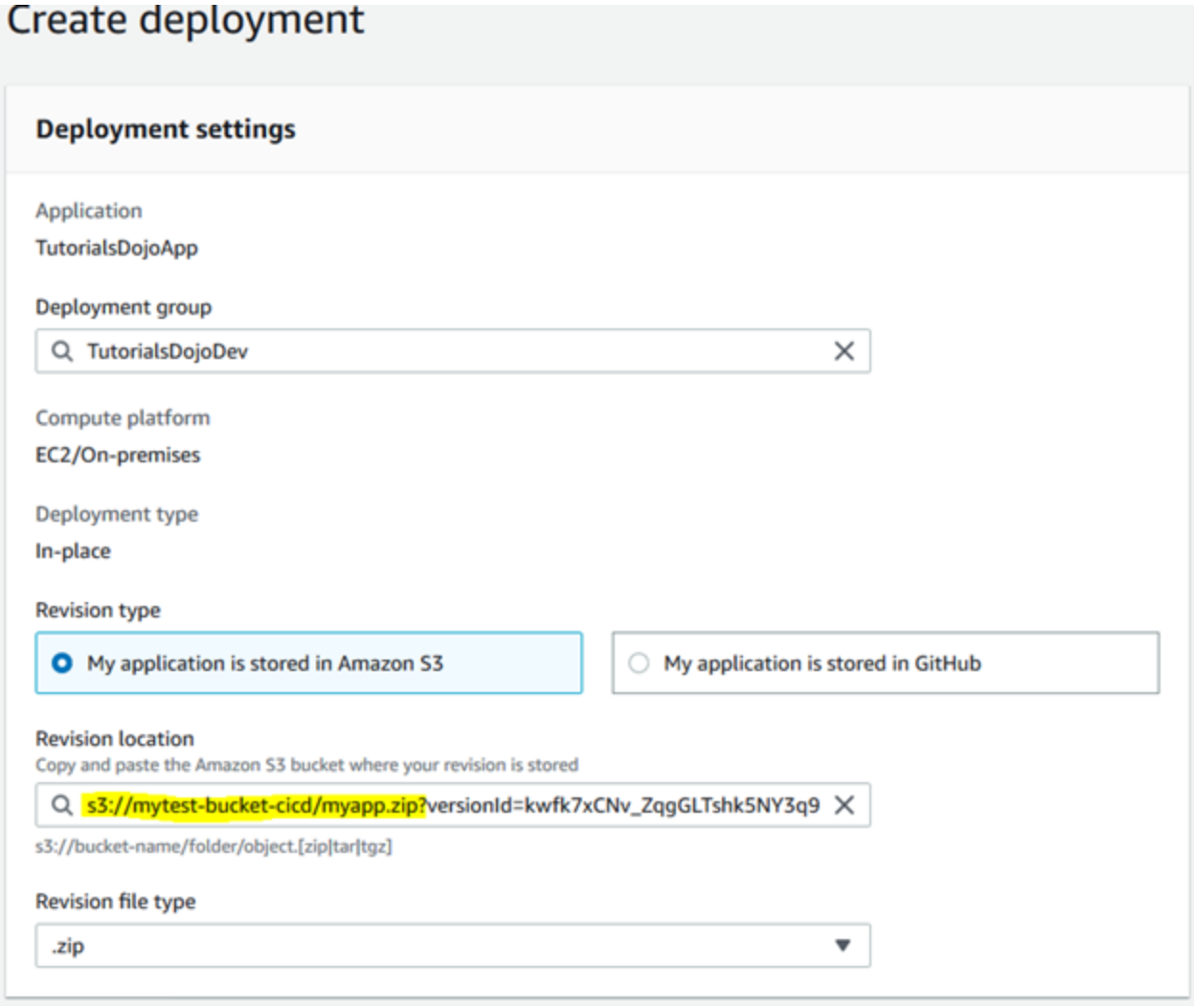
If you are using the AWS CodeBuild, this is where you specify the S3 bucket for the artifact.



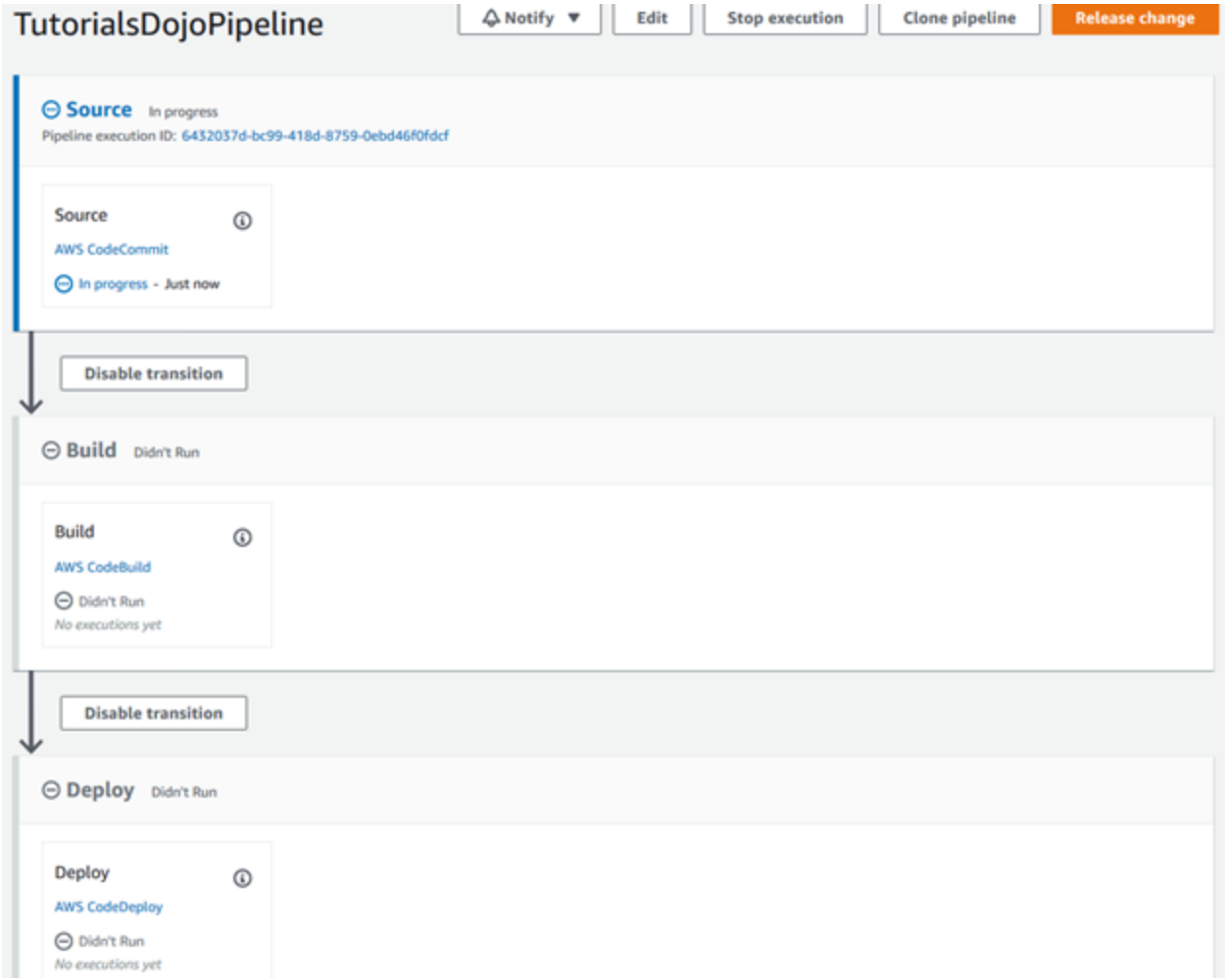
DevOps Exam Notes:

After the build runs, you can use the output artifact to be deployed by AWS CodeDeploy manually. Or using CodePipeline, you can create another stage on which CodeDeploy will automatically pick up this artifact and run the deployment on your desired instances.

Here's how to specify the artifact filename on CodeDeploy.

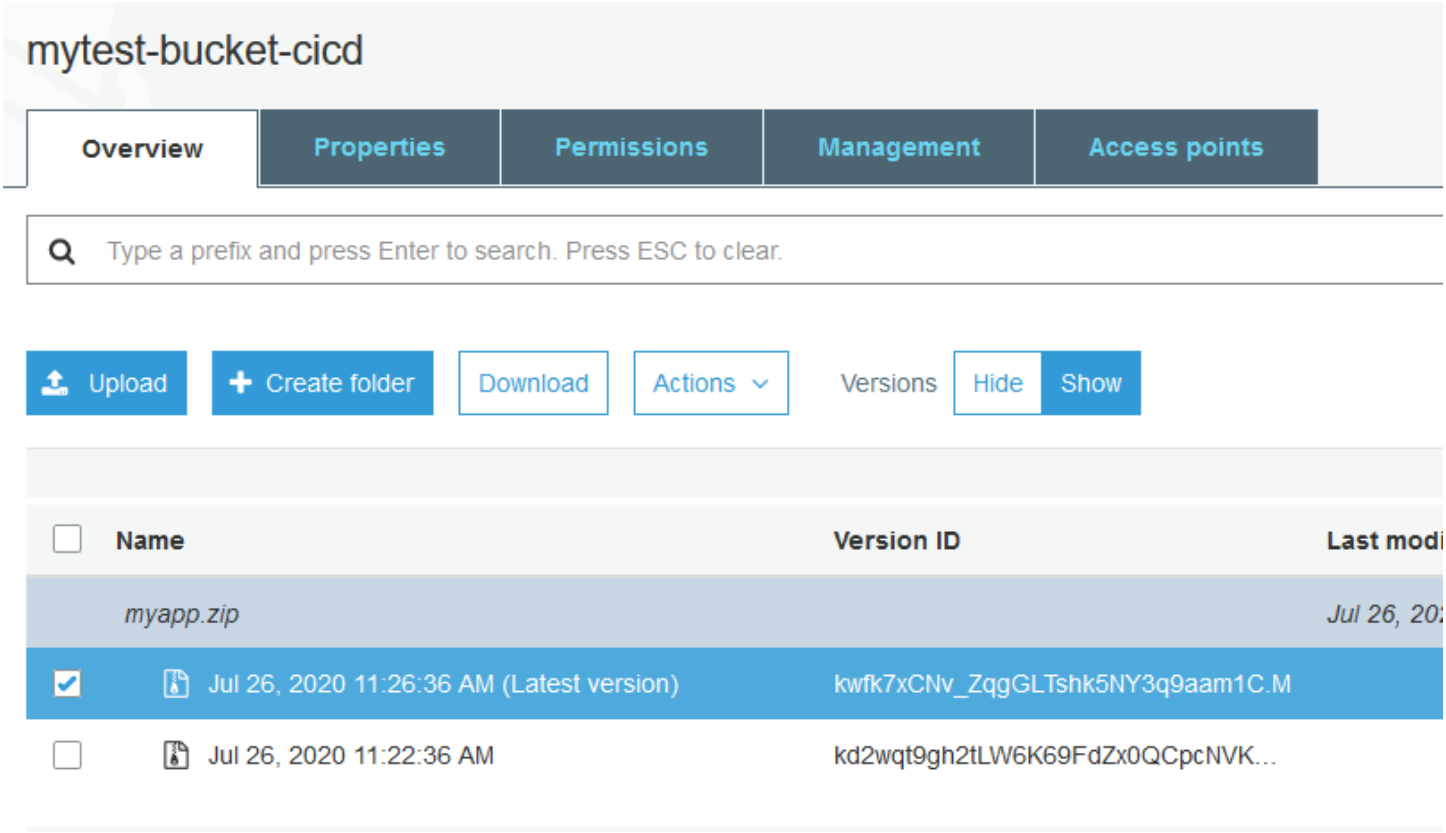


Here's how it should look on your CodePipeline run with multiple stages.



S3 Versioning and Encryption

By default, all the artifacts that you upload to an S3 bucket from AWS CodeBuild are encrypted. The default encryption is AWS S3 Server-side encryption using AES-256.



DevOps Exam Notes:

If you are using CodePipeline and you reference an S3 object artifact (like the output from CodeDeploy), you need to have versioning enabled. When you create your source bucket, make sure that you enable versioning on the bucket first. When you specify the S3 object name on your artifact parameter, you can specify the specific version ID that you want to deploy.

Also, remember that when you use the console to create or edit your pipeline, CodePipeline creates a CloudWatch Events rule that starts your pipeline when a change occurs in the S3 source bucket or when the CodeBuild stage completes and successfully uploads the artifact to S3.

Sources:
https://docs.aws.amazon.com/codepipeline/latest/userguide/tutorials-s3deploy.html
https://docs.aws.amazon.com/codebuild/latest/APIReference/API_ProjectArtifacts.html
https://docs.aws.amazon.com/codebuild/latest/userguide/getting-started-output-console.html
https://docs.aws.amazon.com/codepipeline/latest/userguide/action-reference-S3.html

DynamoDB – Fetch vs Projected Attributes

A projection is the set of attributes that are copied (projected) from a table into a secondary index. These are in addition to the primary key attributes and index key attributes, which are automatically projected.

For example, we can see this base table (GameScores) with partition key (UserID) and sort key (GameTitle).

GameScores

UserId	GameTitle	TopScore	TopScoreDateTime	Wins	Losses
"101"	"Galaxy Invaders"	5842	"2015-09-15:17:24:31"	21	72
"101"	"Meteor Blasters"	1000	"2015-10-22:23:18:01"	12	3
"101"	"Starship X"	24	"2015-08-31:13:14:21"	4	9
"102"	"Alien Adventure"	192	"2015-07-12:11:07:56"	32	192
"102"	"Galaxy Invaders"	0	"2015-09-18:07:33:42"	0	5
"103"	"Attack Ships"	3	"2015-10-19:01:13:24"	1	8
"103"	"Galaxy Invaders"	2317	"2015-09-11:06:53:00"	40	3
"103"	"Meteor Blasters"	723	"2015-10-19:01:13:24"	22	12
"103"	"Starship X"	42	"2015-07-11:06:53:00"	4	19

You can create a global secondary index (GameTitleIndex) with a new partition key (GameTitle) and sort key (TopScore). The base table's primary key attributes are always projected into an index, so the UserID attribute is also present. This improves searching when not using the primary keys of the base table.

GameTitleIndex

GameTitle	*TopScore*	*UserId*	*Wins*	*Losses*
"Alien Adventure"	*192*	*"102"*	*32*	*192*
"Attack Ships"	*3*	*"103"*	*1*	*8*
"Galaxy Invaders"	*0*	*"102"*	*0*	*5*
"Galaxy Invaders"	*2317*	*"103"*	*40*	*3*
"Galaxy Invaders"	*5842*	*"101"*	*21*	*72*
"Meteor Blasters"	*723*	*"103"*	*22*	*12*
"Meteor Blasters"	*1000*	*"101"*	*12*	*3*
"Starship X"	*24*	*"101"*	*4*	*9*
"Starship X"	*42*	*"103"*	*4*	*19*

When you query an index, Amazon DynamoDB can access any attribute in the projection as if those attributes were in a table of their own. When you create a secondary index, you need to specify the attributes that will be projected into the index. DynamoDB provides three different options for this:

- **KEYS_ONLY** – Each item in the index consists only of the table partition key and sort key values, plus the index key values. The KEYS_ONLY option results in the smallest possible secondary index.
- **INCLUDE** – In addition to the attributes described in KEYS_ONLY, the secondary index will include other non-key attributes that you specify.
- **ALL** – The secondary index includes all of the attributes from the source table. Because all of the table data are duplicated in the index, an ALL projection results in the largest possible secondary index.

You can project other base table attributes into the index if you want. When you query the index, DynamoDB can retrieve these projected attributes efficiently. However, global secondary index queries cannot fetch attributes from the base table. For example, if you query GameTitleIndex as shown in the diagram above, the query could not access any non-key attributes other than TopScore (although the key attributes GameTitle and UserID would automatically be projected).

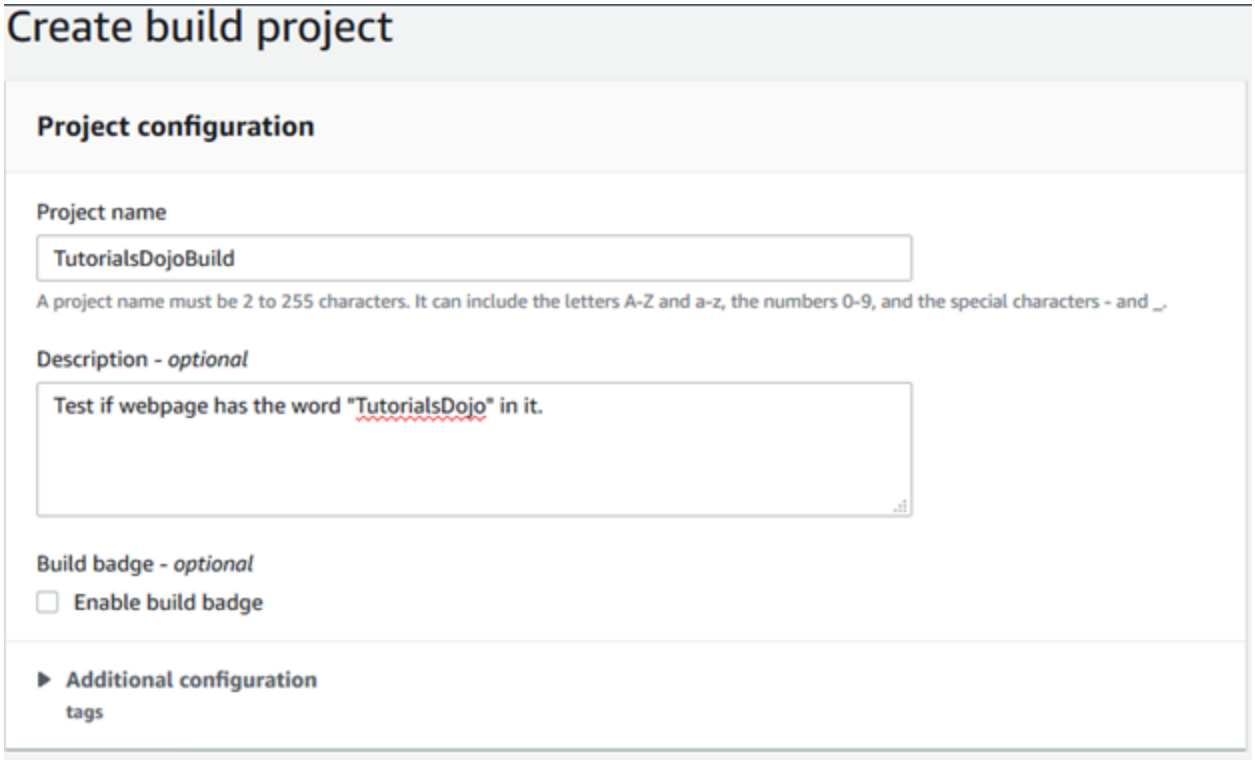
Sources:
https://docs.aws.amazon.com/amazondynamodb/latest/APIReference/API_Projection.html
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GSI.html#GSI.Projections

CodeBuild with CloudWatch Logs, Metrics, and Alarms

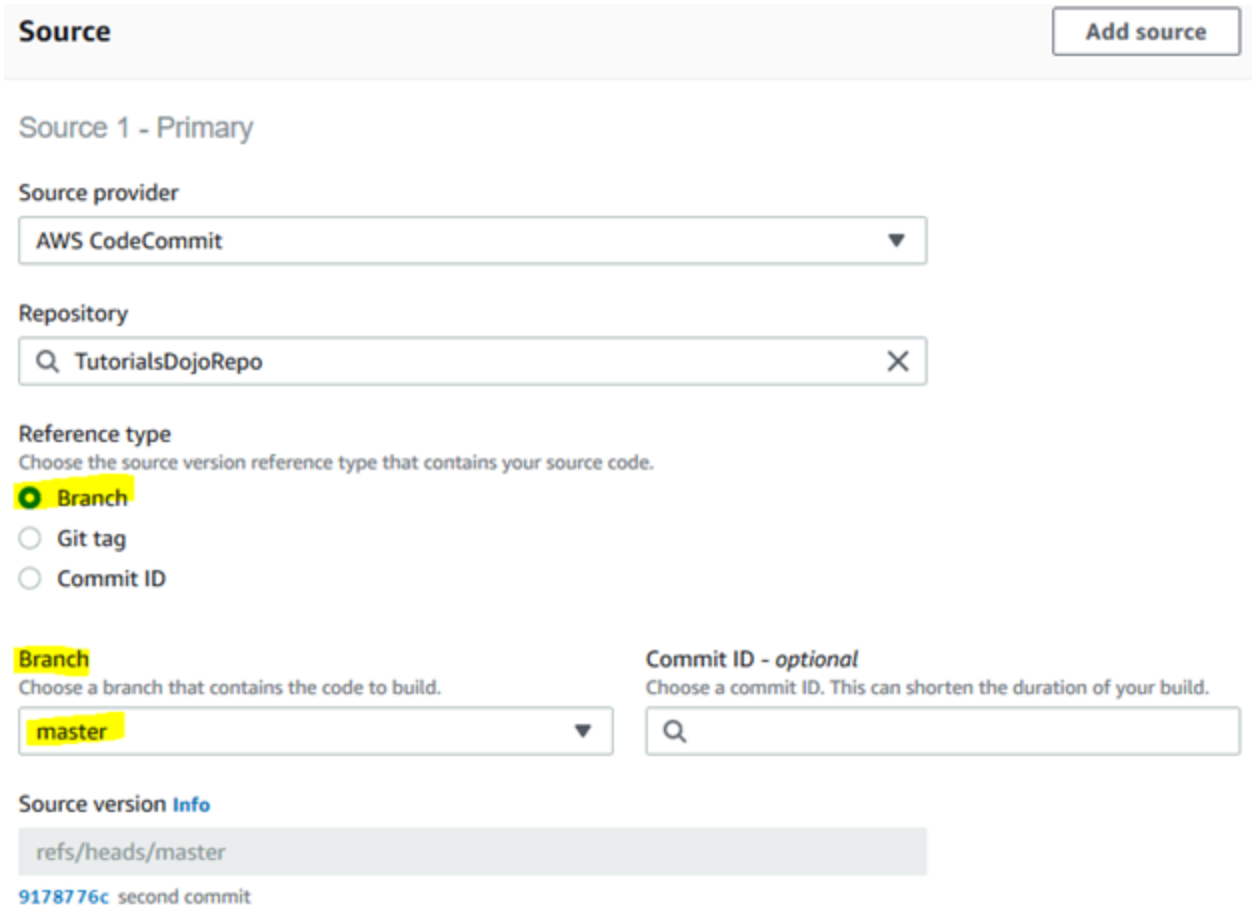
AWS Codebuild allows you to compile, build, run tests, and produce an artifact of your code. This can be integrated into your CI/CD process. For example, you are using AWS CodeCommit as version control for your source code repository. After a developer pushes new code to the Development branch, you can 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.

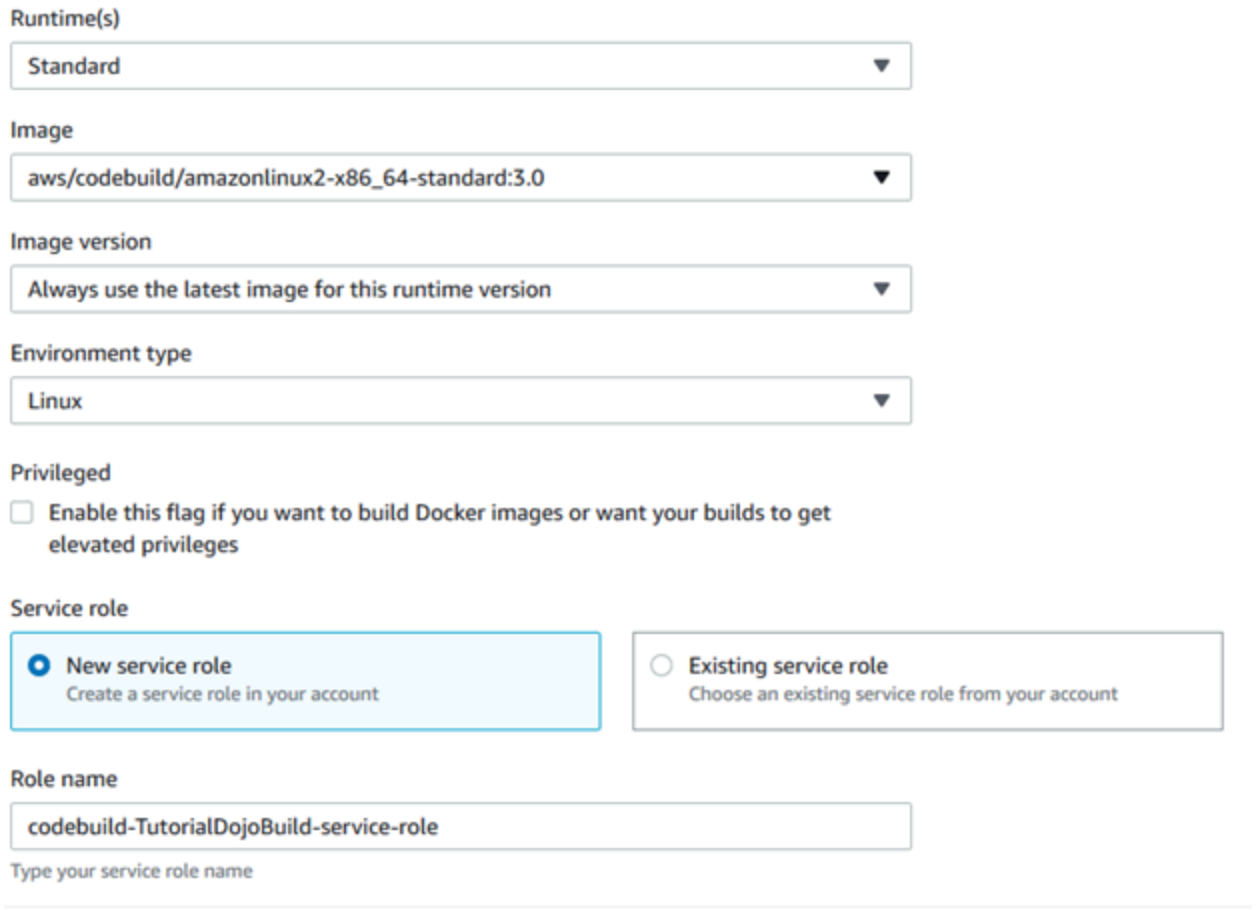
1. Go to AWS Code Build > Build Projects and click Create Build Project. Input your details for this project.



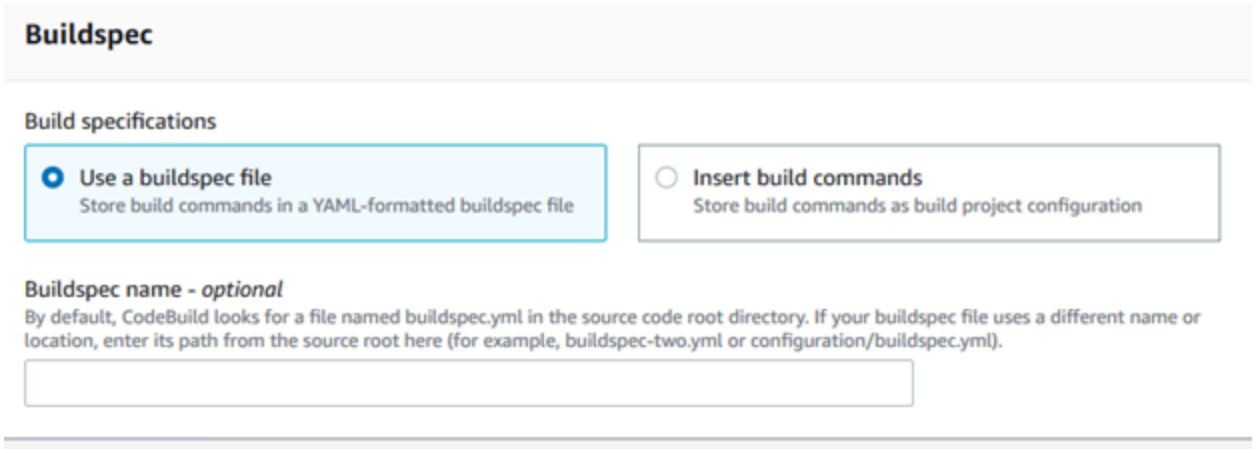
2. CodeBuild supports several sources for your application code, including Amazon S3, AWS CodeCommit, GitHub, Bitbucket, etc. Using CodeBuild, select your source repository and branch.



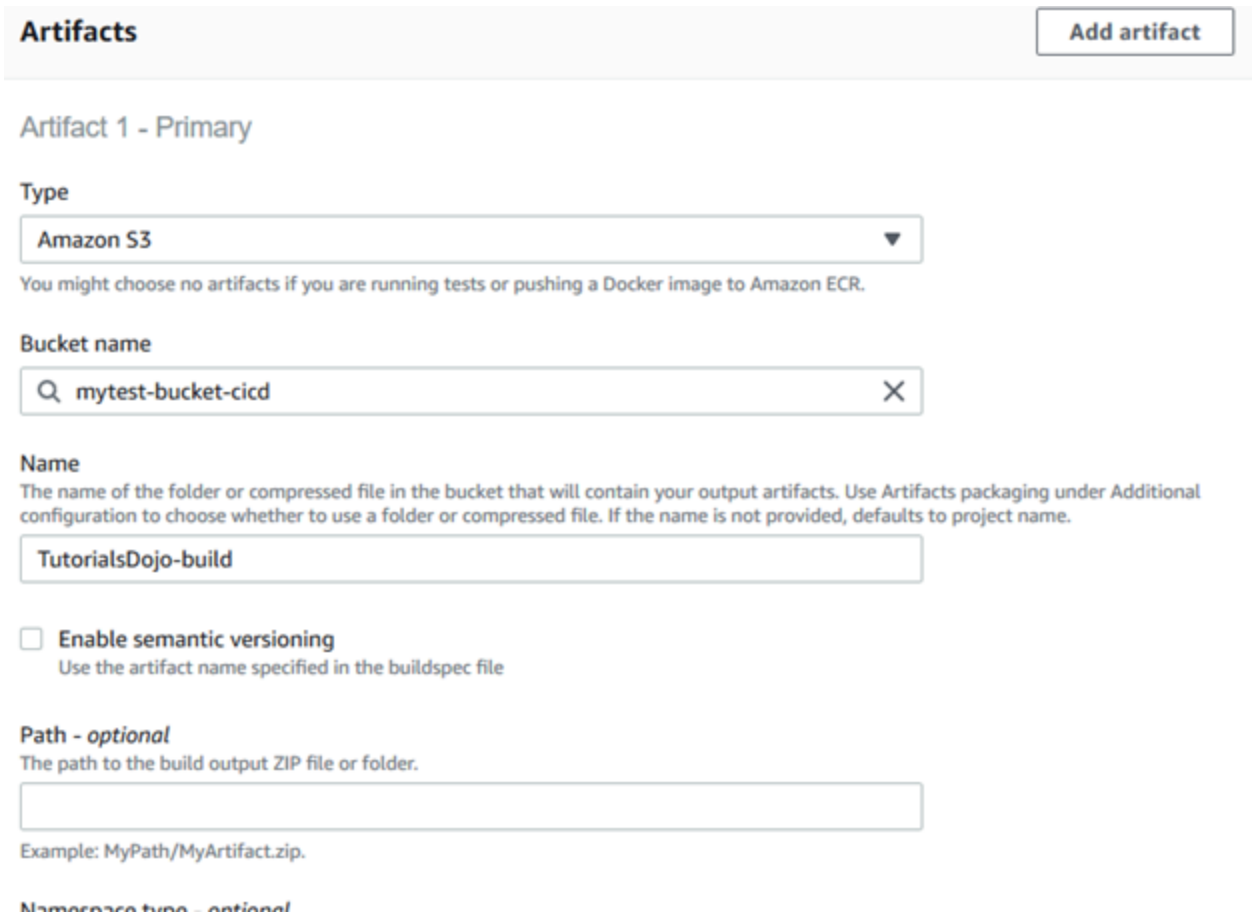
3. Use the Amazon Linux runtime since we'll build this for the Amazon Linux AMI.



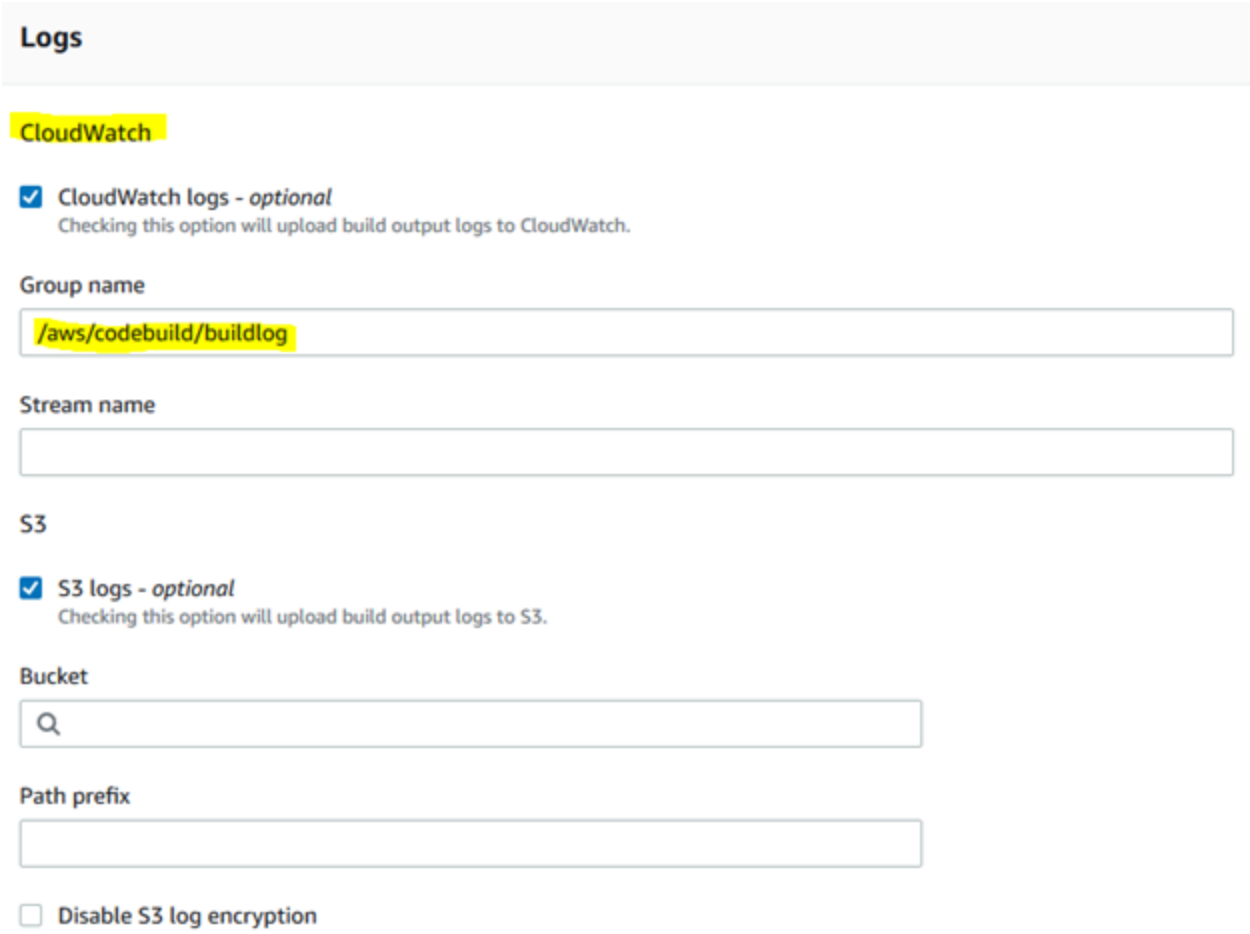
4. By default, the build specification filename is “buildspec.yml”. This should be in the root folder of your application code.



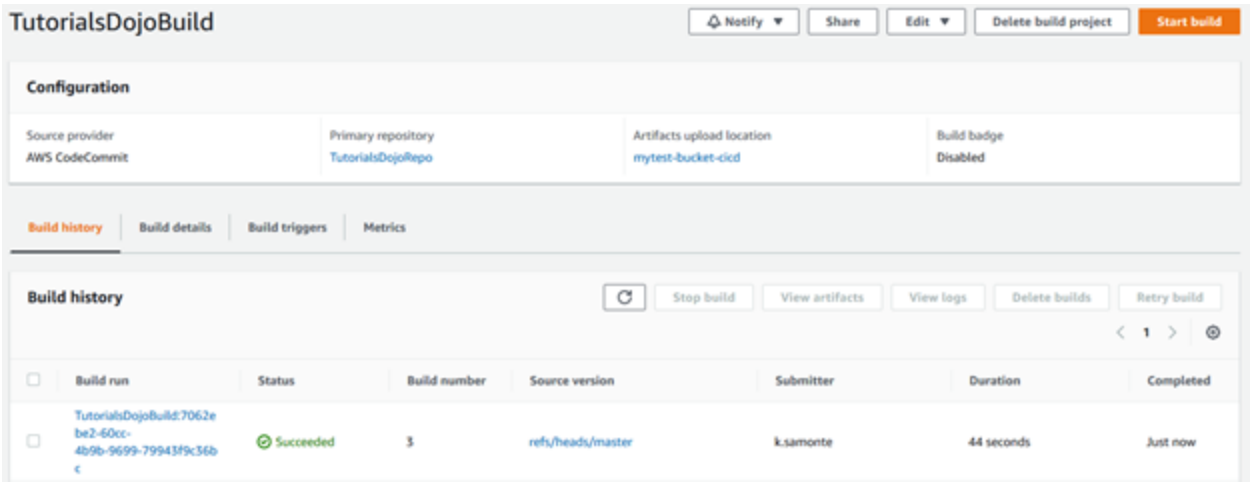
5. Specify which S3 bucket you are going to send the output artifact of your build.



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.

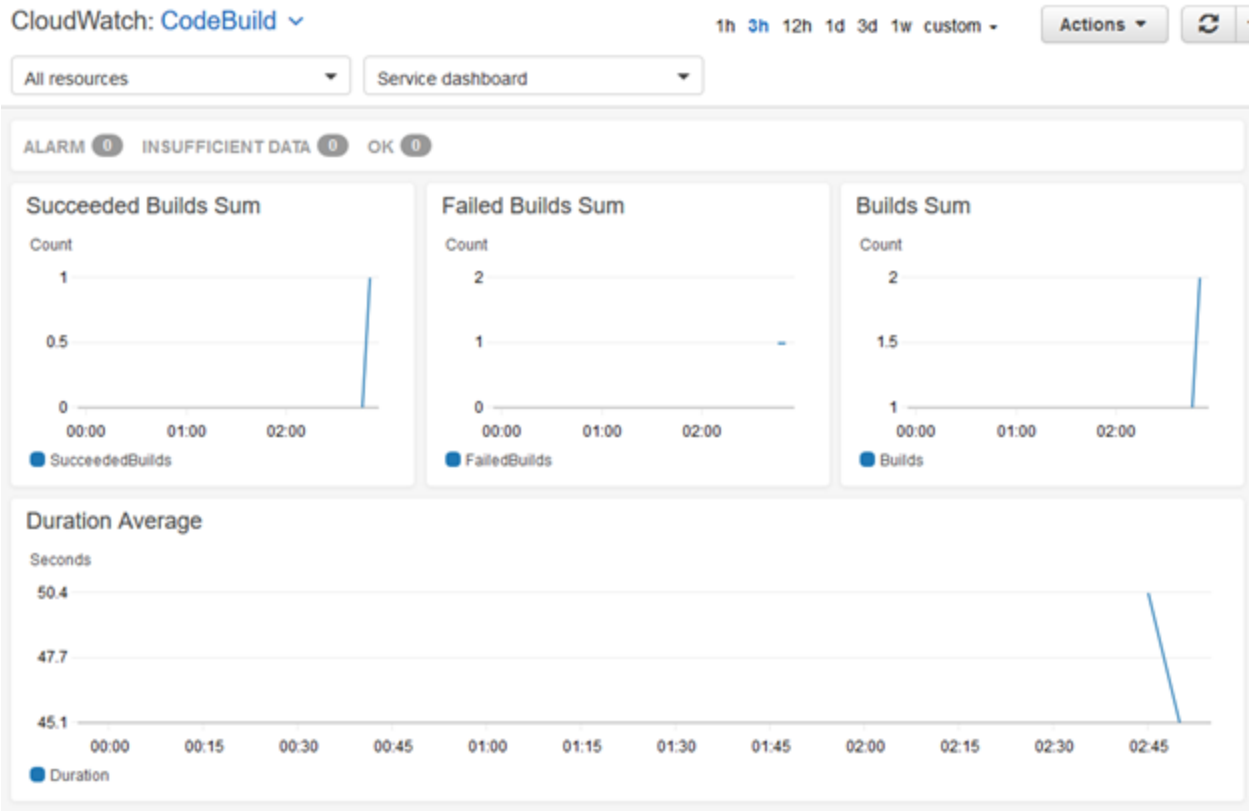


7. Click “Create build project”. Select your project and click the “Start build” button.

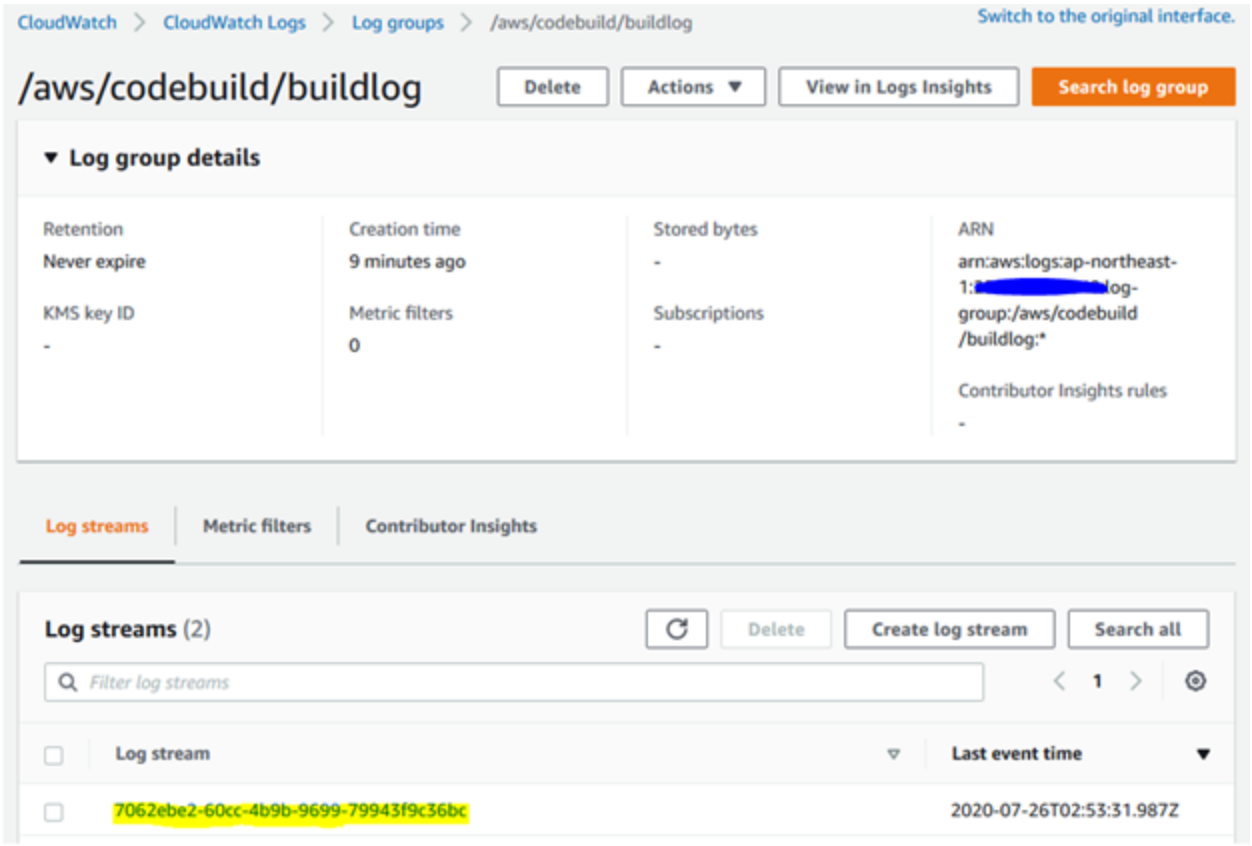


8. After the build, you should see the artifact on the S3 bucket. You will also see the build logs of your project if you click on the Build run ID of your project.

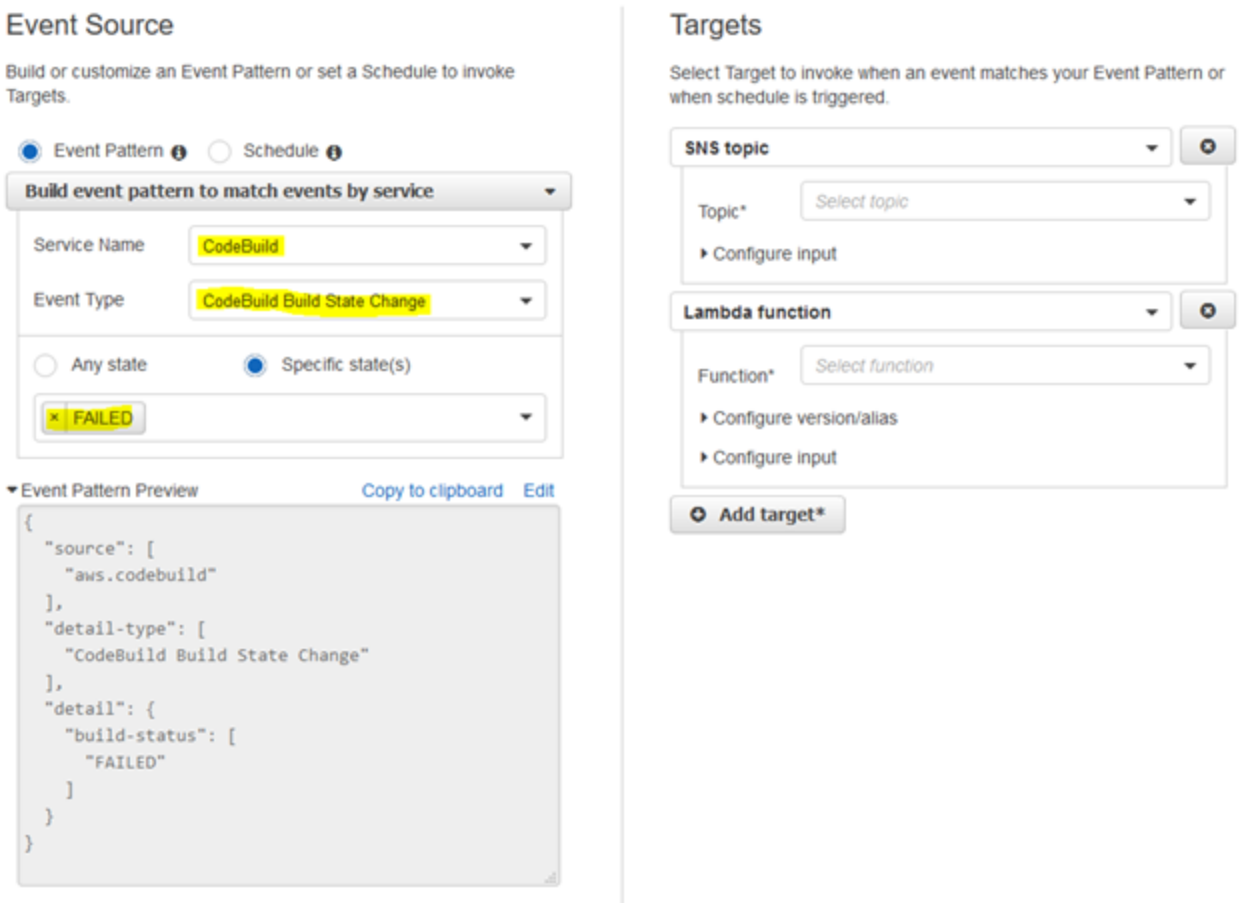
9. AWS CodeBuild is also integrated on CloudWatch Logs. When you go to CloudWatch Dashboard, you can select the pre-defined CodeBuild Dashboard where you will see the metrics such as Successful builds, Failed builds, etc.



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

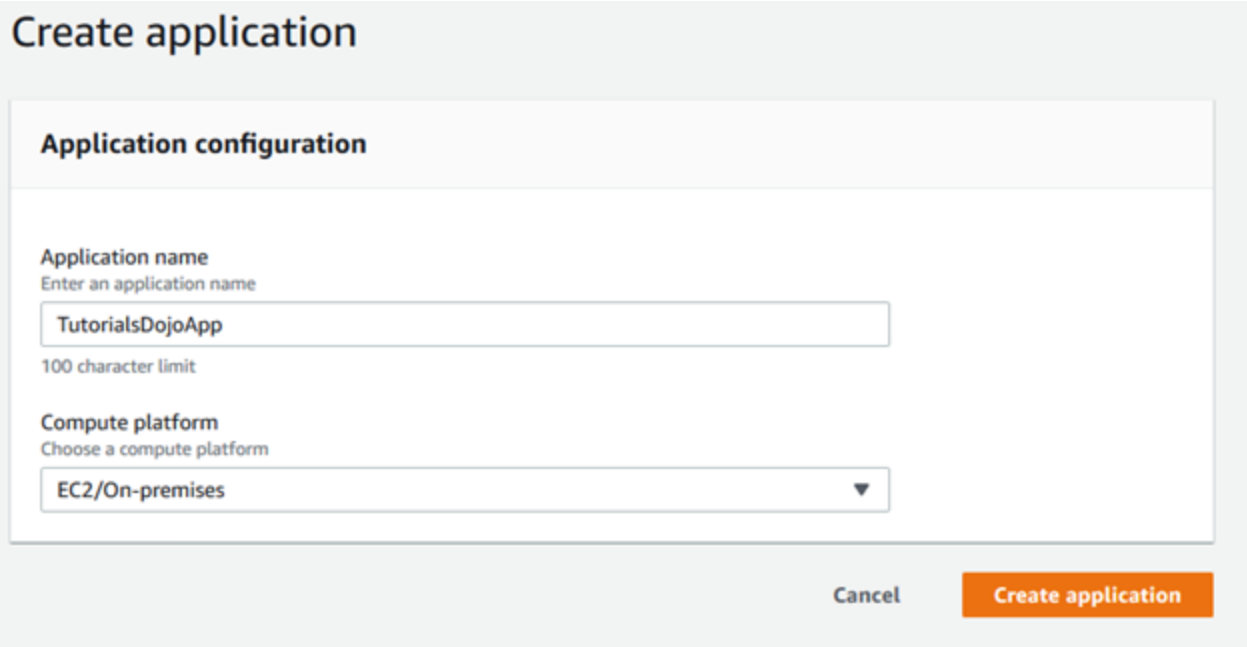
CodeDeploy with CloudWatch Logs, Metrics, and Alarms

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.

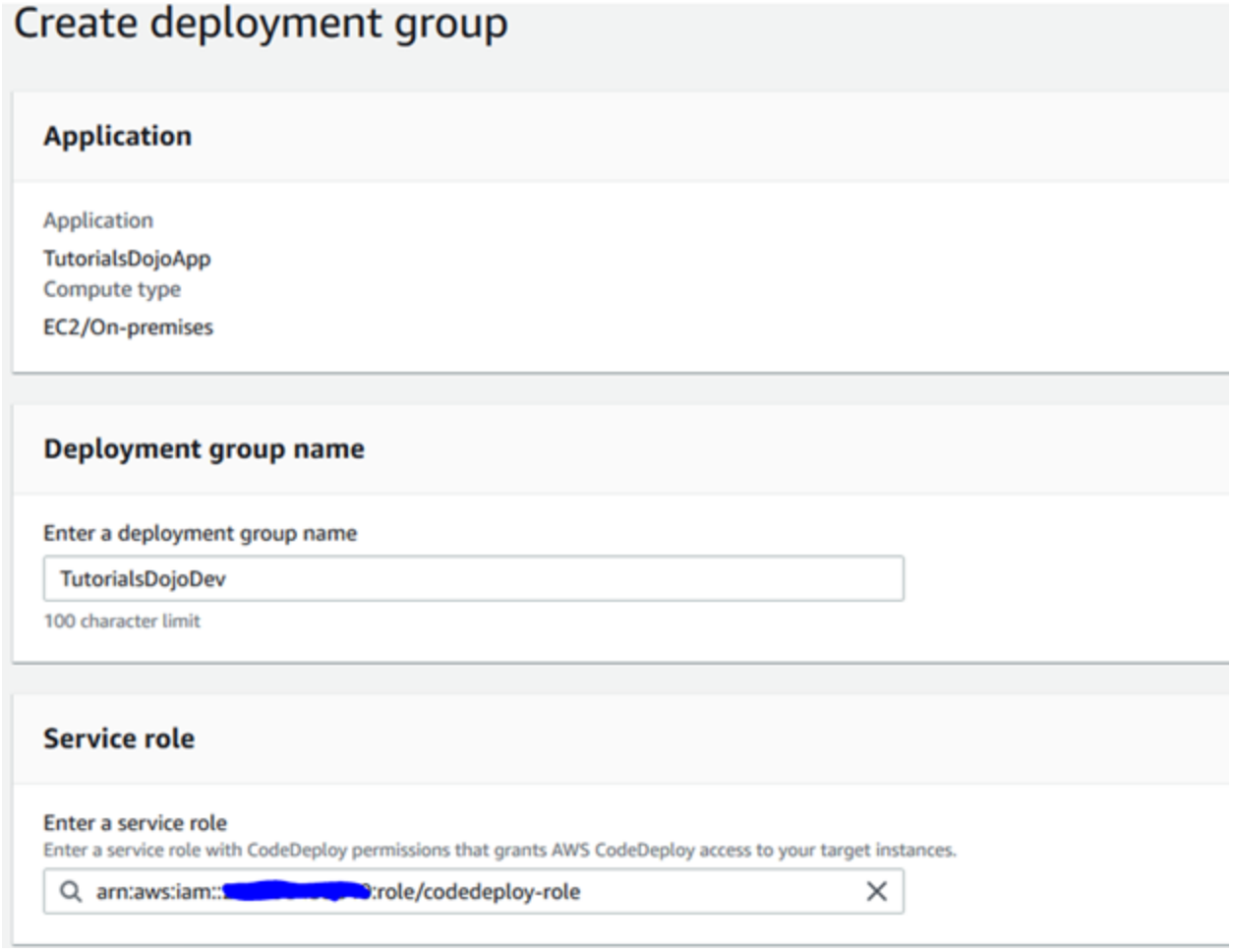
For example, after you have produced from AWS CodeBuild, you can have CodeDeploy fetch the artifact from the AWS S3 bucket and then deploy it to your AWS instances. Please note that the target instances need to have the AWS CodeDeploy agent installed on them for this to be successful and have proper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

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

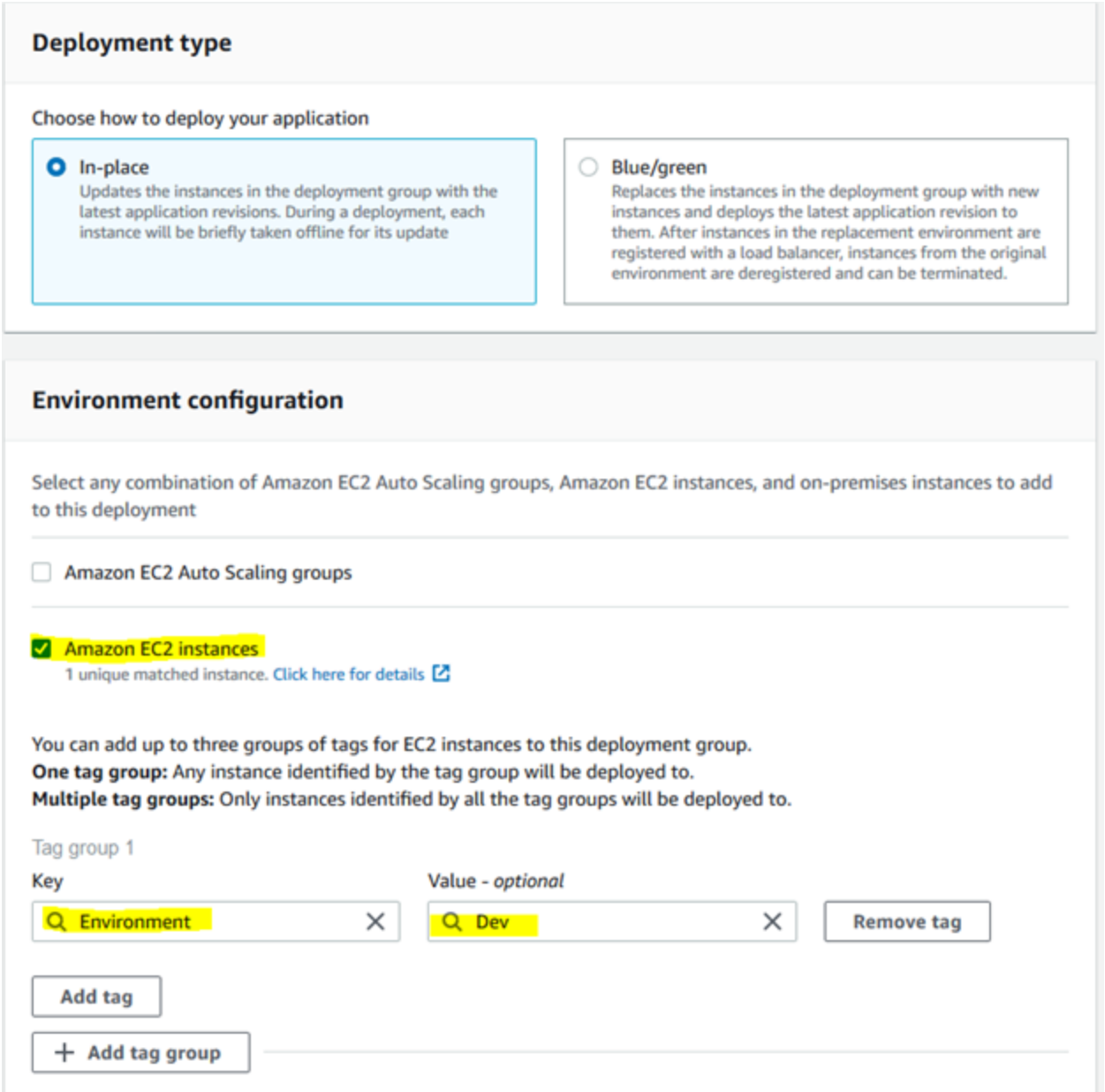
1. Go to AWS CodeDeploy > Applications and click “Create Application”. Input details of your application and which platform it is running.



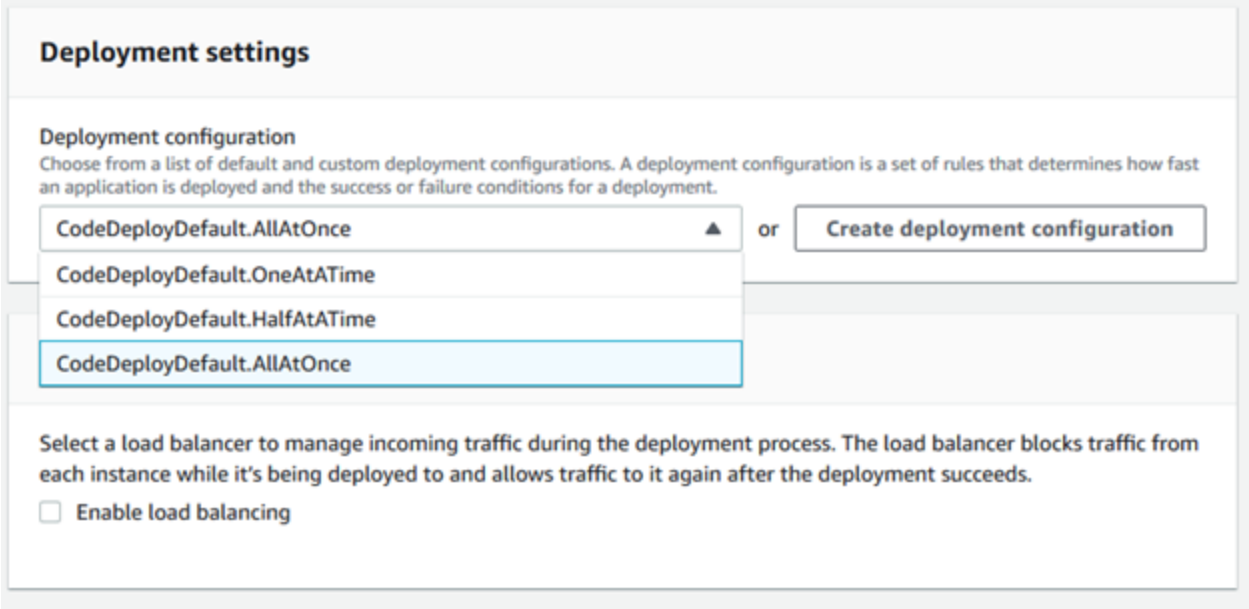
2. Select your application and create a Deployment Group. CodeDeploy needs to have an IAM permission to access your targets as well as read the AWS S3 bucket containing the artifact to be deployed.



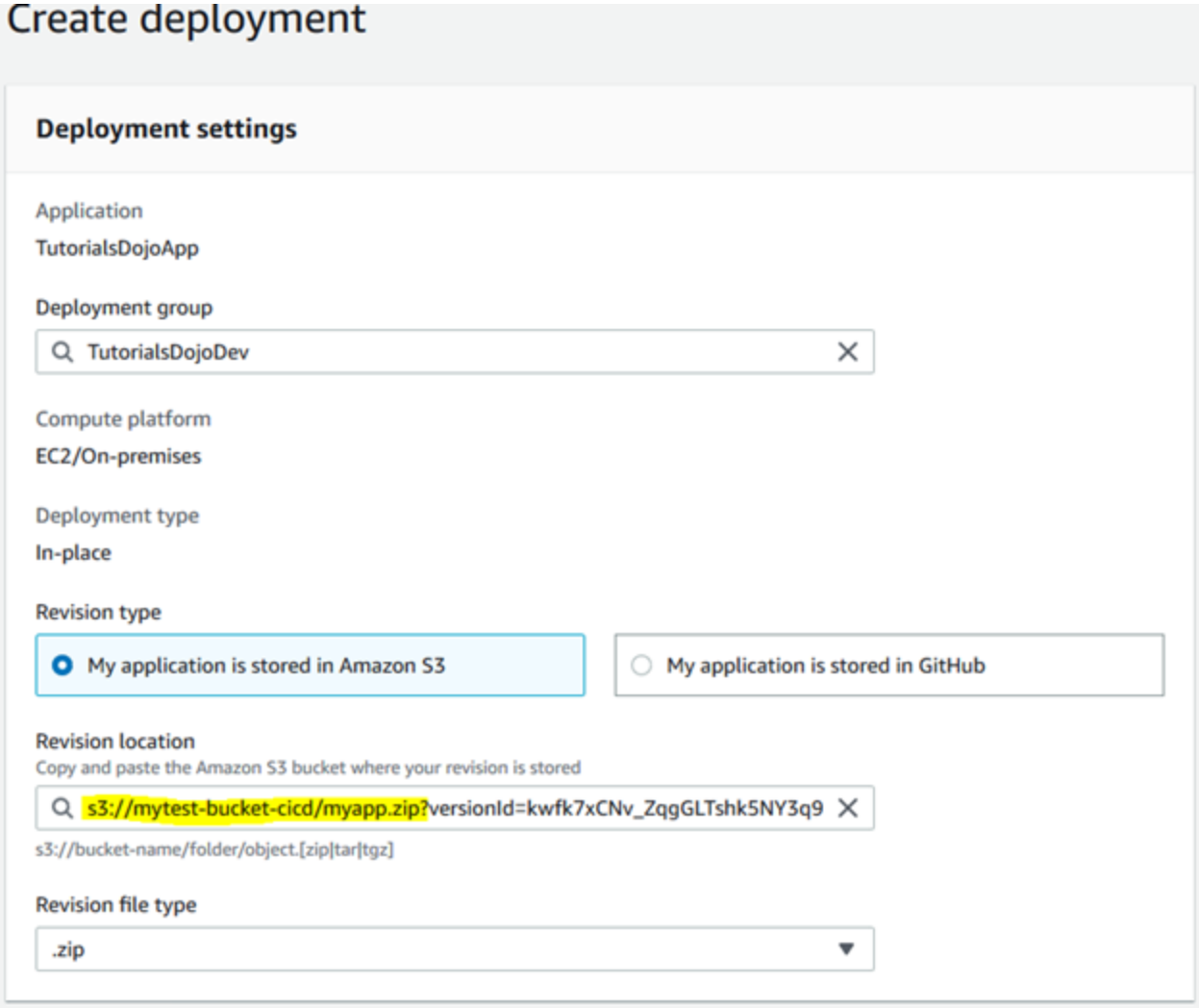
3. Select your deployment type. You can use In-place deployment if you already have the instances running. Specify the Tags of those EC2 instances, for example **Environment:Dev**. CodeDeploy will use this as an identifier for your target instances.



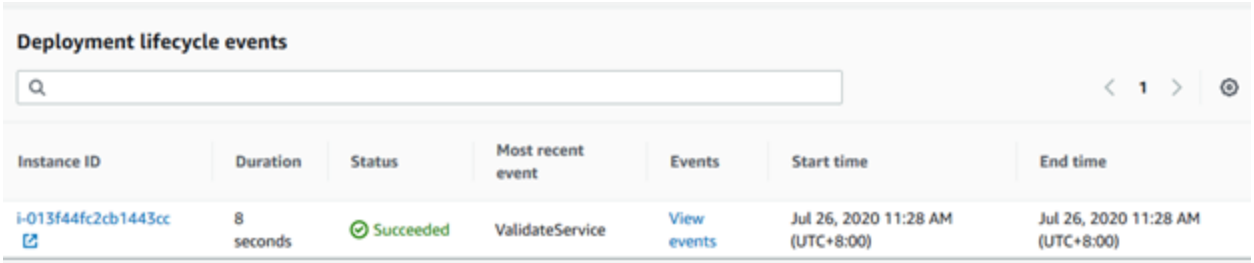
4. Select how you want the new code to be deployed, such as All-at-once, or one-at-a-time, etc. These deployment settings will be discussed on the succeeding sections.



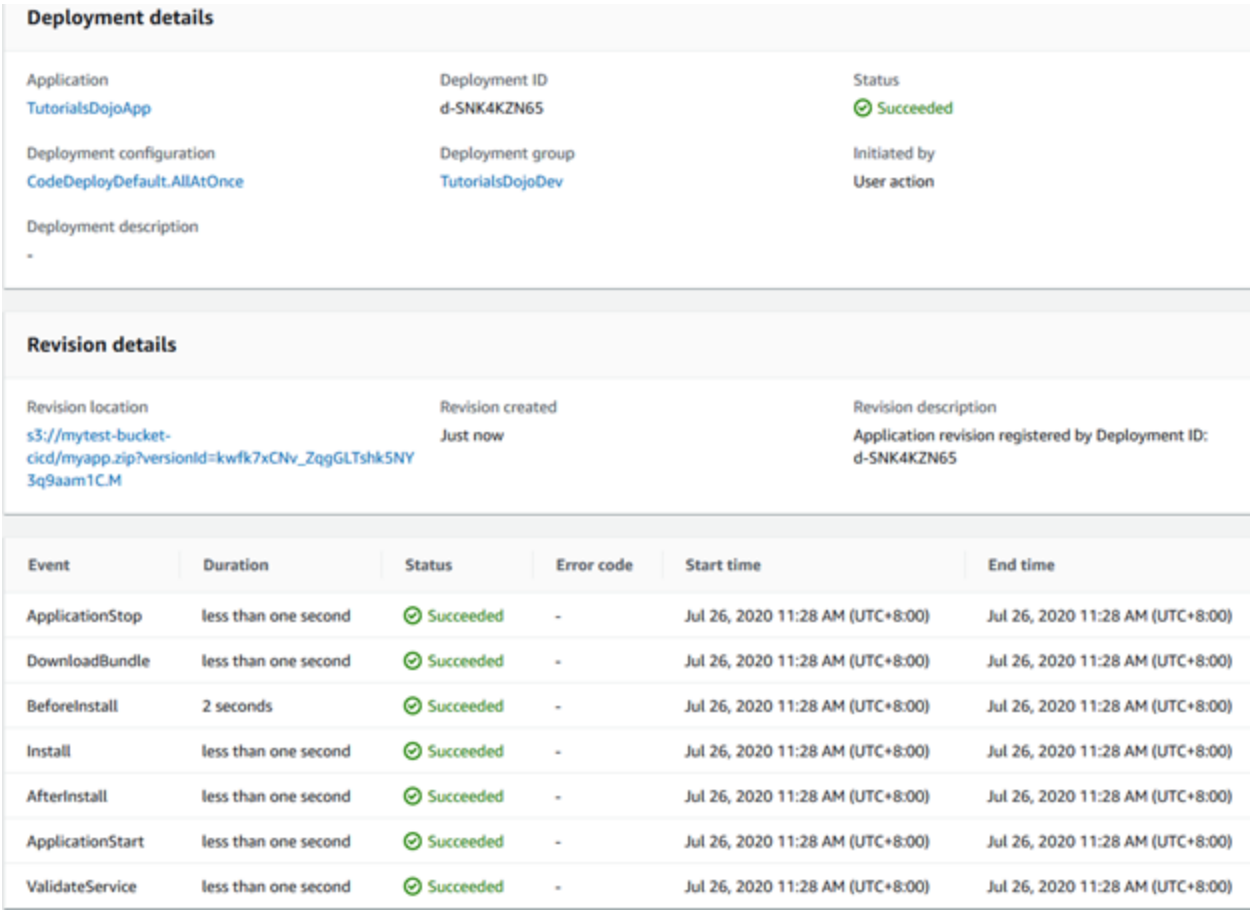
5. Now on your application Deployment group, create a Deployment. Input details for the artifact source. On the S3 bucket, you can specify the versionID of the artifact file.



6. After you click “Create deployment”, the deployment of the artifact on your EC2 instances will begin. You should see that the deployment will succeed.



7. You can click on “View events” to see the stages of the deployment process. Here you can view the deployment events as well as the status of the deployment lifecycle hooks you have defined. See the Lifecycle Event hooks section of this book for the details regarding each event.



Deployment details

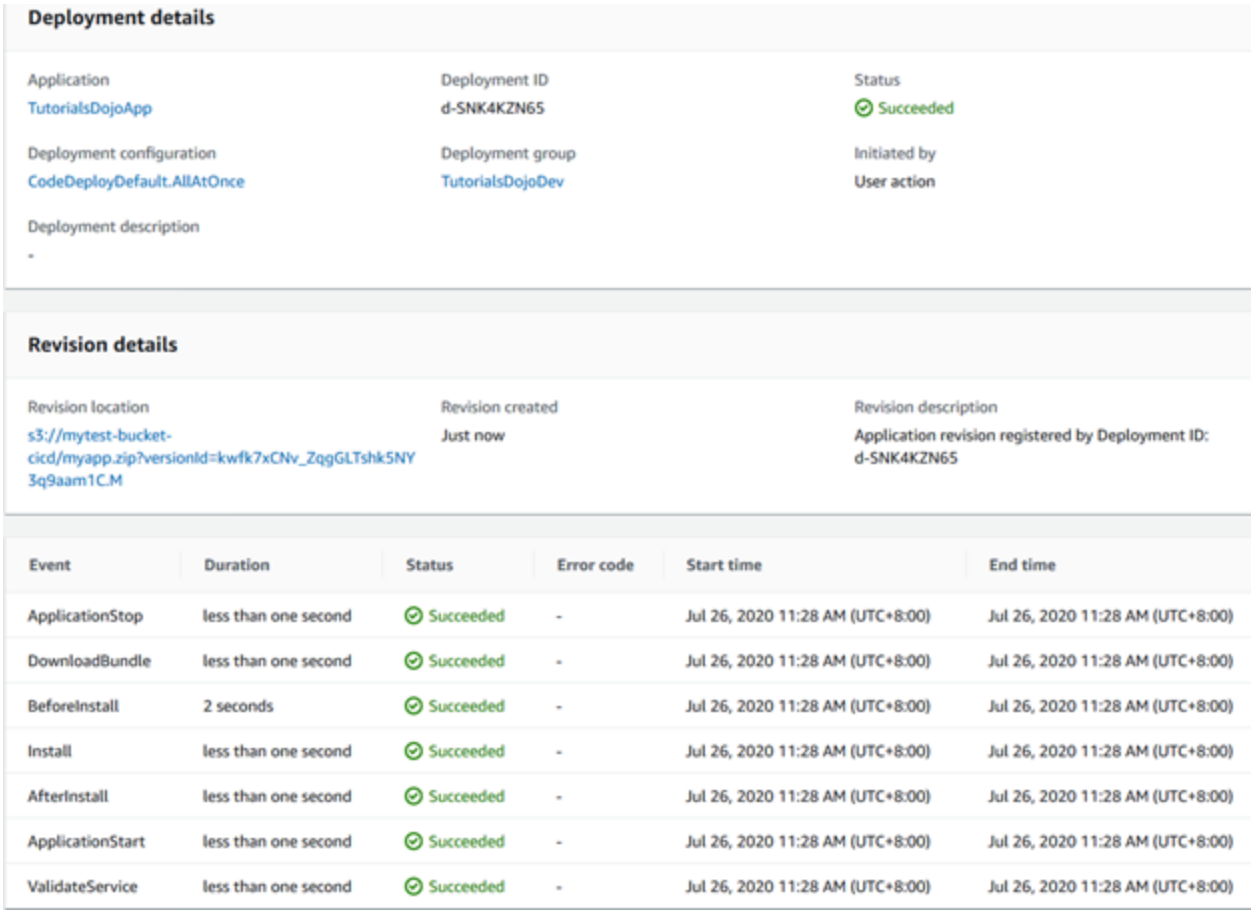
Application	Deployment ID	Status
TutorialsDojoApp	d-SNK4KZN65	Succeeded
Deployment configuration	**Deployment group**	**Initiated by**
CodeDeployDefault.AllAtOnce	TutorialsDojoDev	User action
Deployment description		
-		

Revision details

Revision location	Revision created	Revision description
s3://mytest-bucket-cicd/myapp.zip?versionId=kwfk7xCNv_ZqgGLTshk5NY3q9aam1C.M	Just now	Application revision registered by Deployment ID: d-SNK4KZN65

Event	Duration	Status	Error code	Start time	End time
ApplicationStop	less than one second	Succeeded	-	Jul 26, 2020 11:28 AM (UTC+8:00)	Jul 26, 2020 11:28 AM (UTC+8:00)
DownloadBundle	less than one second	Succeeded	-	Jul 26, 2020 11:28 AM (UTC+8:00)	Jul 26, 2020 11:28 AM (UTC+8:00)
BeforeInstall	2 seconds	Succeeded	-	Jul 26, 2020 11:28 AM (UTC+8:00)	Jul 26, 2020 11:28 AM (UTC+8:00)
Install	less than one second	Succeeded	-	Jul 26, 2020 11:28 AM (UTC+8:00)	Jul 26, 2020 11:28 AM (UTC+8:00)
AfterInstall	less than one second	Succeeded	-	Jul 26, 2020 11:28 AM (UTC+8:00)	Jul 26, 2020 11:28 AM (UTC+8:00)
ApplicationStart	less than one second	Succeeded	-	Jul 26, 2020 11:28 AM (UTC+8:00)	Jul 26, 2020 11:28 AM (UTC+8:00)
ValidateService	less than one second	Succeeded	-	Jul 26, 2020 11:28 AM (UTC+8:00)	Jul 26, 2020 11:28 AM (UTC+8:00)

8. CodePipeline is integrated with AWS CloudWatch rule. You can create a CloudWatch Events rule that will detect CodeDeploy status changes, such as a Successful or Failed deployment. Then have it invoke a Lambda function to perform a custom action such as sending a notification on a Slack channel or setting an SNS topic to send an email to you about the status of your deployment.



Deployment details

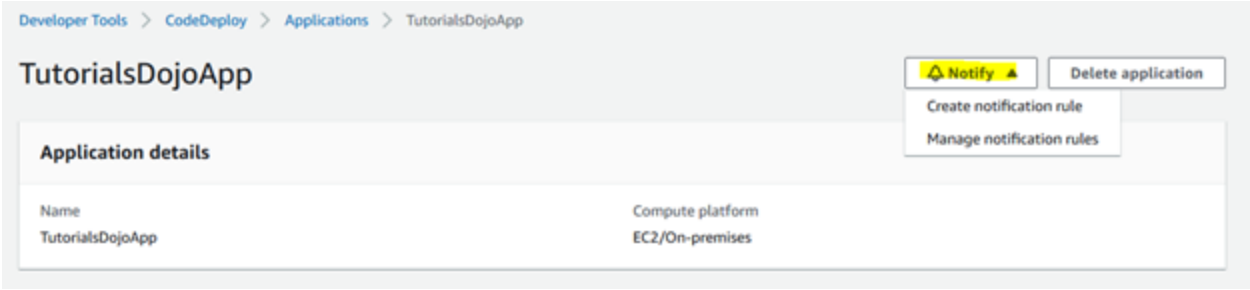
Application	Deployment ID	Status
TutorialsDojoApp	d-SNK4KZN65	Succeeded
Deployment configuration	**Deployment group**	**Initiated by**
CodeDeployDefault.AllAtOnce	TutorialsDojoDev	User action
Deployment description		
-		

Revision details

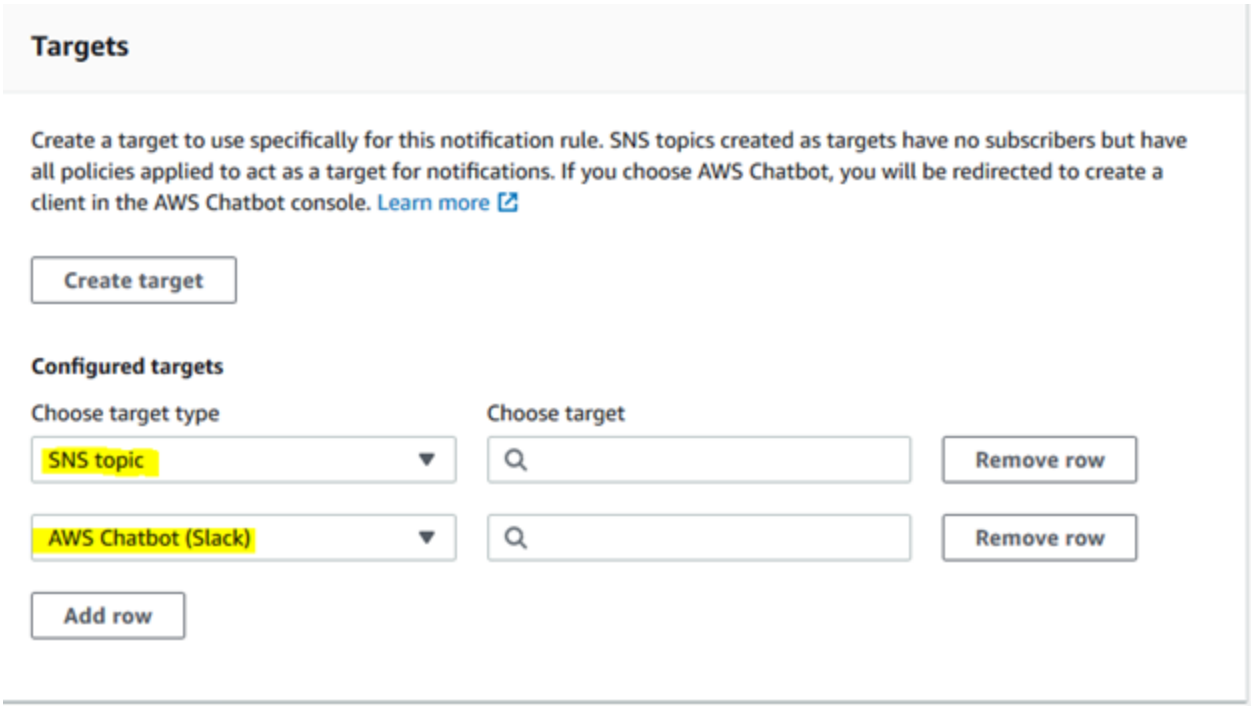
Revision location	Revision created	Revision description
s3://mytest-bucket-cicd/myapp.zip?versionId=kwfk7xCNv_ZqgGLTshk5NY3q9aam1C.M	Just now	Application revision registered by Deployment ID: d-SNK4KZN65

Event	Duration	Status	Error code	Start time	End time
ApplicationStop	less than one second	Succeeded	-	Jul 26, 2020 11:28 AM (UTC+8:00)	Jul 26, 2020 11:28 AM (UTC+8:00)
DownloadBundle	less than one second	Succeeded	-	Jul 26, 2020 11:28 AM (UTC+8:00)	Jul 26, 2020 11:28 AM (UTC+8:00)
BeforeInstall	2 seconds	Succeeded	-	Jul 26, 2020 11:28 AM (UTC+8:00)	Jul 26, 2020 11:28 AM (UTC+8:00)
Install	less than one second	Succeeded	-	Jul 26, 2020 11:28 AM (UTC+8:00)	Jul 26, 2020 11:28 AM (UTC+8:00)
AfterInstall	less than one second	Succeeded	-	Jul 26, 2020 11:28 AM (UTC+8:00)	Jul 26, 2020 11:28 AM (UTC+8:00)
ApplicationStart	less than one second	Succeeded	-	Jul 26, 2020 11:28 AM (UTC+8:00)	Jul 26, 2020 11:28 AM (UTC+8:00)
ValidateService	less than one second	Succeeded	-	Jul 26, 2020 11:28 AM (UTC+8:00)	Jul 26, 2020 11:28 AM (UTC+8:00)

9. CodeDeploy also has built-in notification triggers. Click “Notify” on your application.



10. For events happening on your deployment, you can have targets much like AWS CloudWatch Events, however, this is limited to only an SNS Topic or AWS chatbot to Slack.



DevOps Exam Notes:

AWS CodePipeline is integrated with AWS CloudWatch Events. You can create a CloudWatch Events rule that will detect CodeDeploy status changes, such as a Successful or Failed deployment. With the rule targets, you invoke a Lambda function to perform a custom action i.e. set an SNS topic to send an email to you about the status of your deployment.

CodeDeploy also has built-in notification triggers to notify you of your deployment status, however, this is limited to only an SNS Topic or AWS Chatbot to Slack.

CodeDeploy Supports ECS and Lambda Deployments

Aside from EC2 instances, AWS CodeDeploy also supports deployment for ECS instances and Lambda deployments. The general steps for deployment are still the same - create an **Application**, create a **deployment group** for your instances, and create a **deployment** for your application.

However, for ECS, the artifact is a Docker image which can be stored from AWS ECR or from DockerHub. For Lambda deployments, the artifact will come from a zip file from an S3 bucket. Be sure to have the proper filename of your Lambda handler file for successful code deployments.

Deployments for ECS also support deployment configuration such as all-at-once, one-at-a-time, half-of-the-time. These deployment configurations will be discussed on a separate section. Lambda on the other hand supports a percentage of traffic shifting such as linear or canary deployment. This is also discussed in a separate section.

Sources:
https://docs.aws.amazon.com/codedeploy/latest/userguide/monitoring-create-alarms.html
https://docs.aws.amazon.com/codedeploy/latest/userguide/monitoring.html
https://docs.aws.amazon.com/codedeploy/latest/userguide/deployments-create-console-ecs.html
https://docs.aws.amazon.com/codedeploy/latest/userguide/deployments-create-console-lambda.html

CodePipeline and CloudWatch Events Integration

AWS CodePipeline is a continuous delivery service that helps you automate your release pipelines for fast and reliable application and infrastructure updates. For example, you can use CodePipeline to have specific stages that build, test, and deploy your code to your target instances.

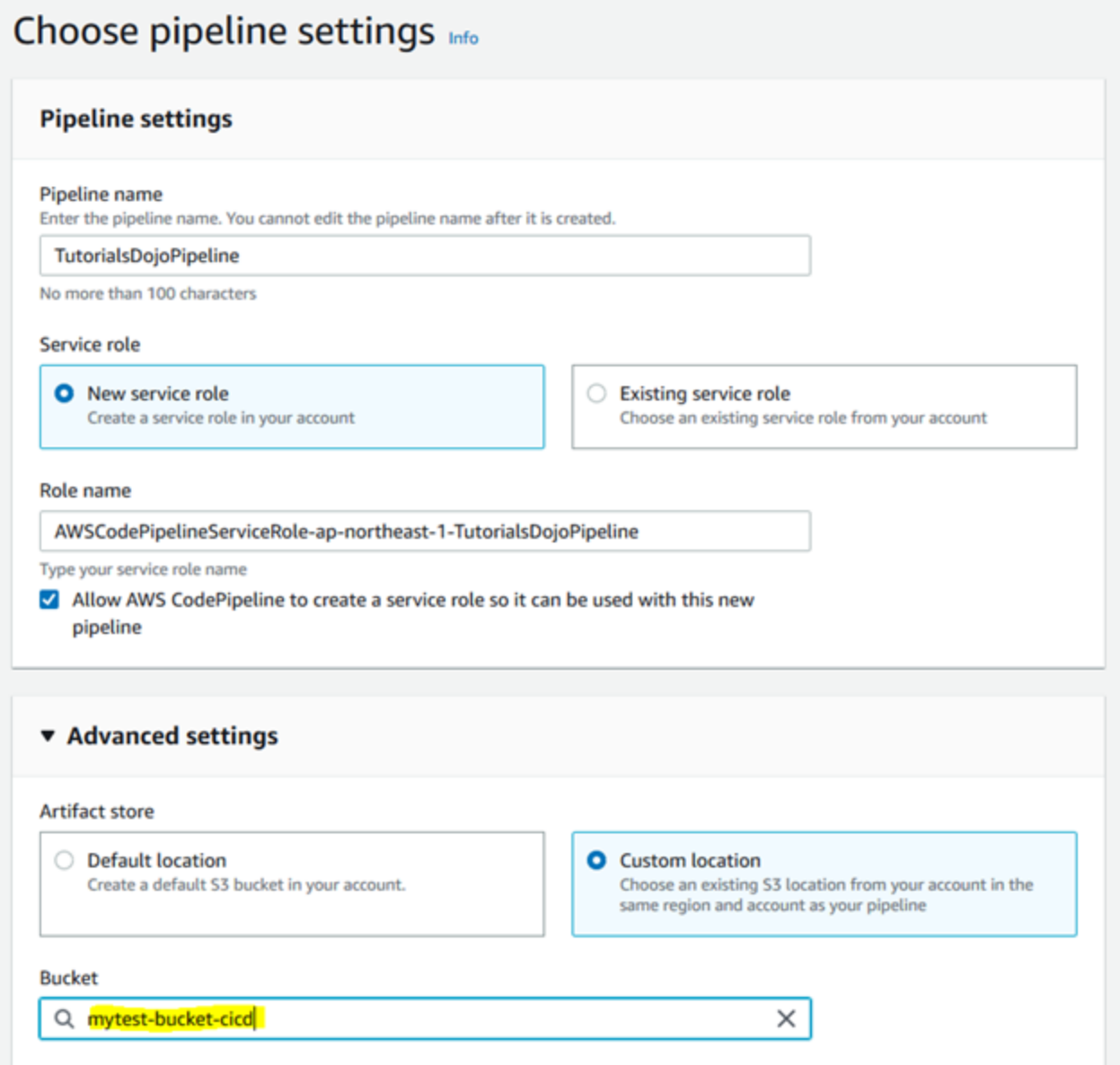
You can have CodePipeline trigger a full deployment pipeline when your developer pushes code to your CodeCommit repository, which will then start the series of pipeline stages:

- Pull source code from the CodeCommit repository.
- Run the CodeBuild project to build and test the artifact file. Upload the artifact to AWS S3 bucket.
- Trigger CodeDeploy deployment to fetch the artifact and deploy it to your instances.

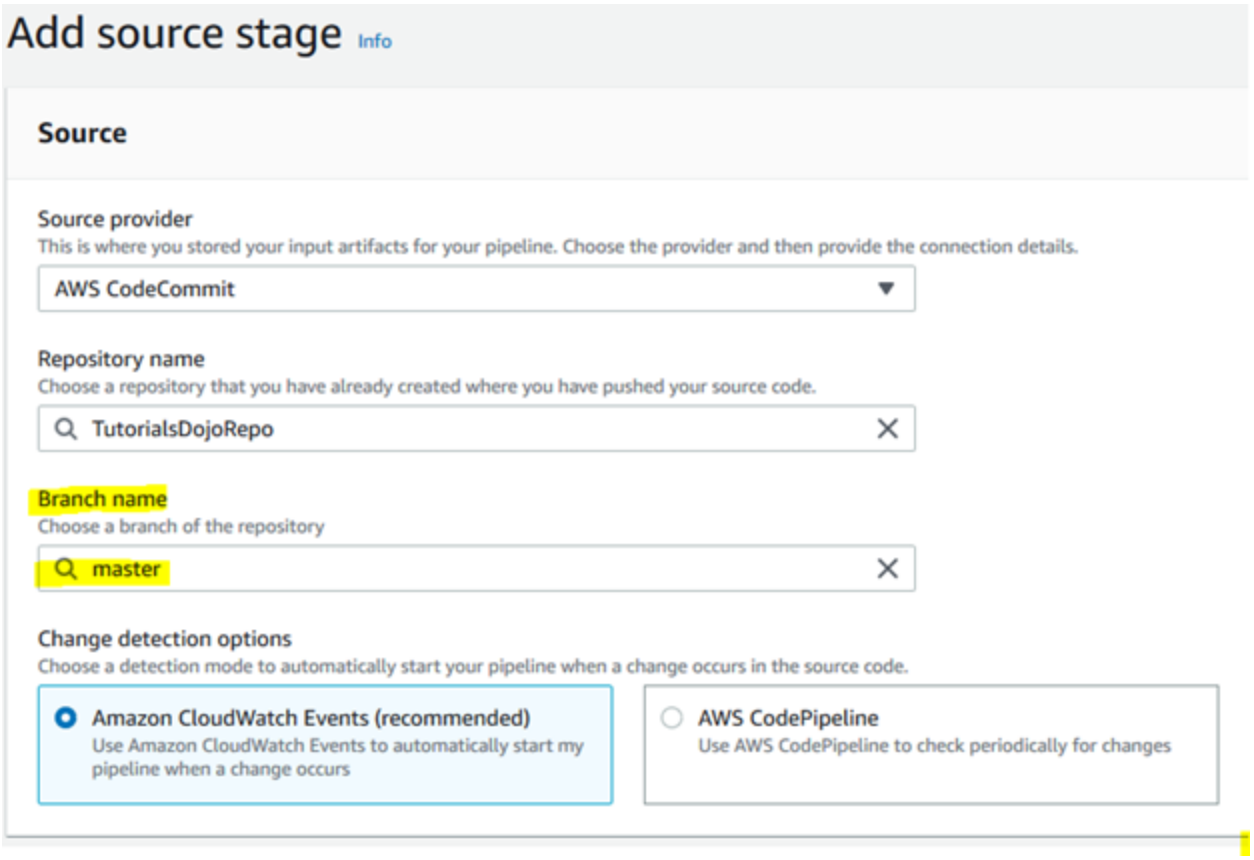
This whole cascading process is triggered only from a single CodeCommit repository push event.

Here are the steps to create a pipeline on AWS CodePipeline as well as its integration with AWS CloudWatch.

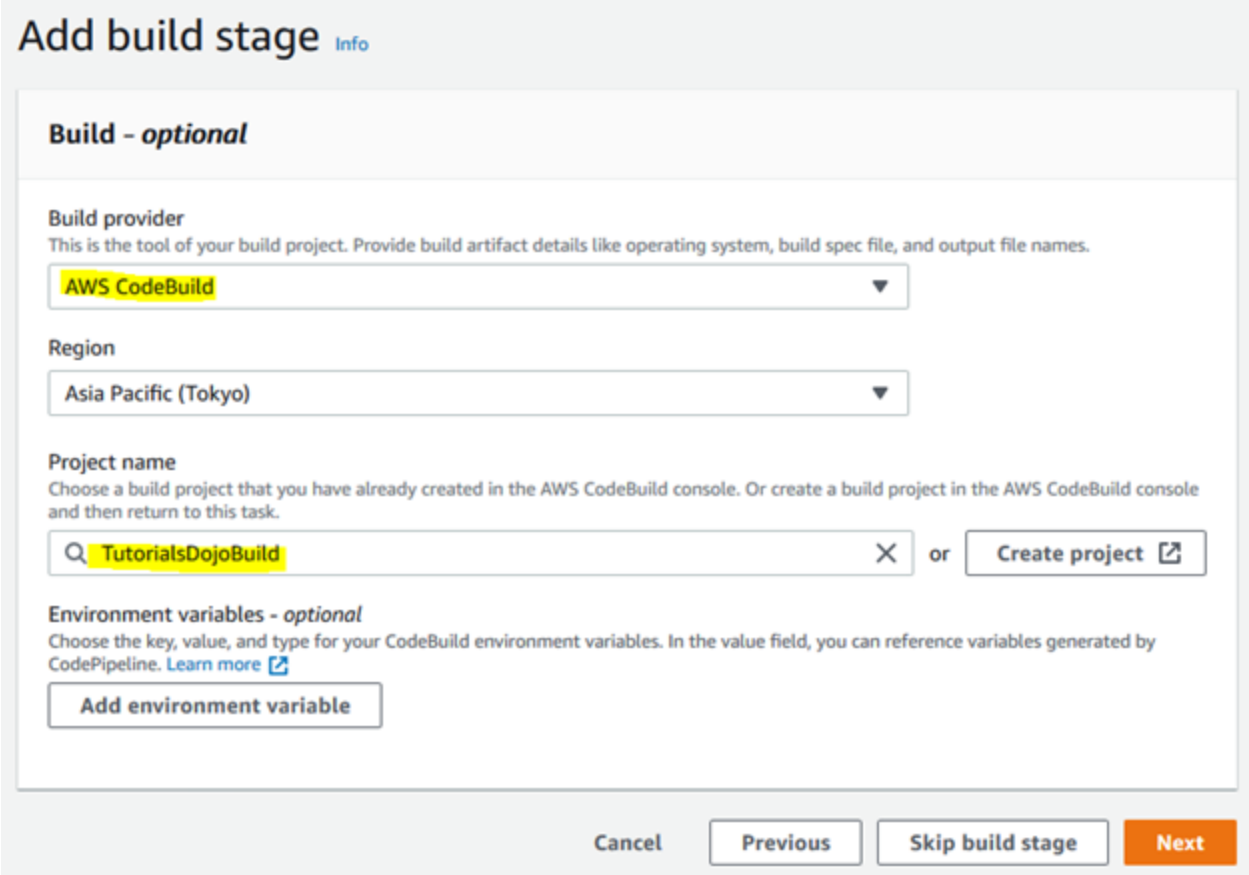
1. Go to AWS CodePipeline > Pipeline and click Create pipeline. Input the name for the service for CodePipeline and the S3 bucket that holds the artifacts for this pipeline.



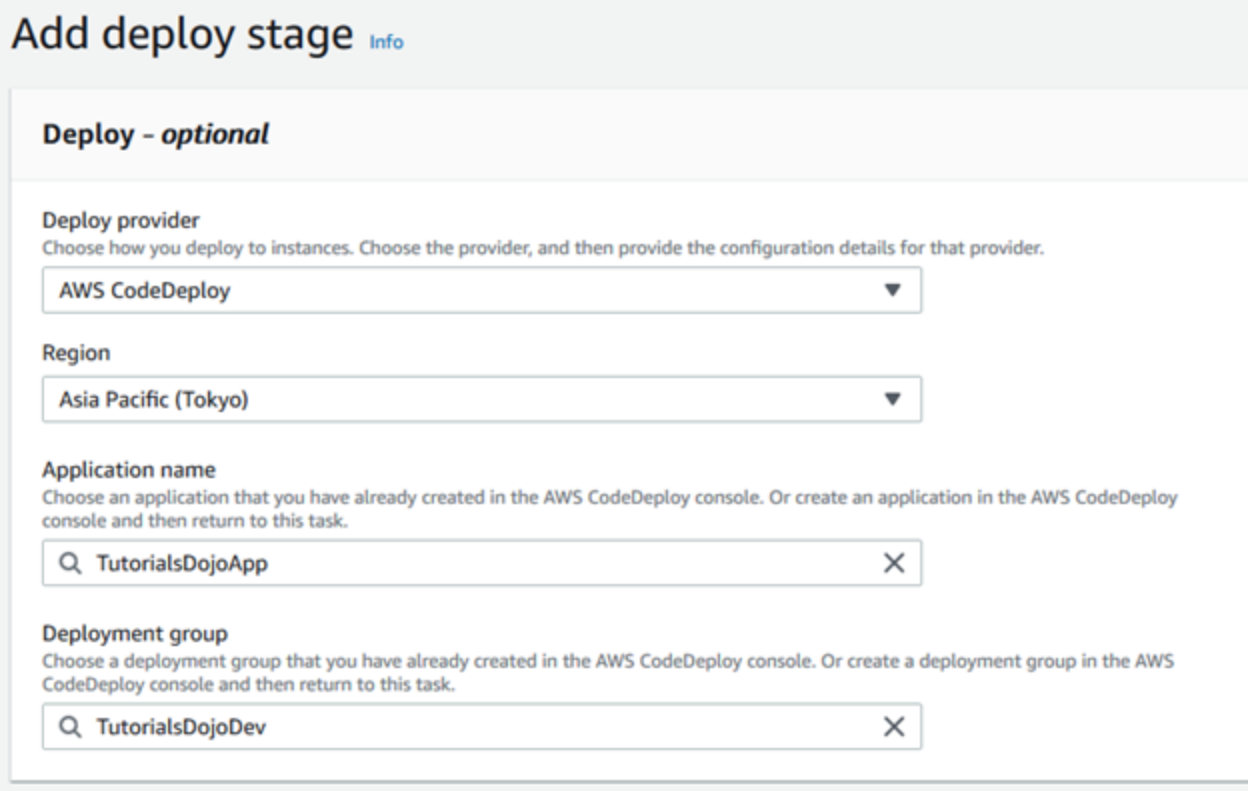
2. Add a source stage such as CodeCommit repository and branch on which code version you want to deploy.



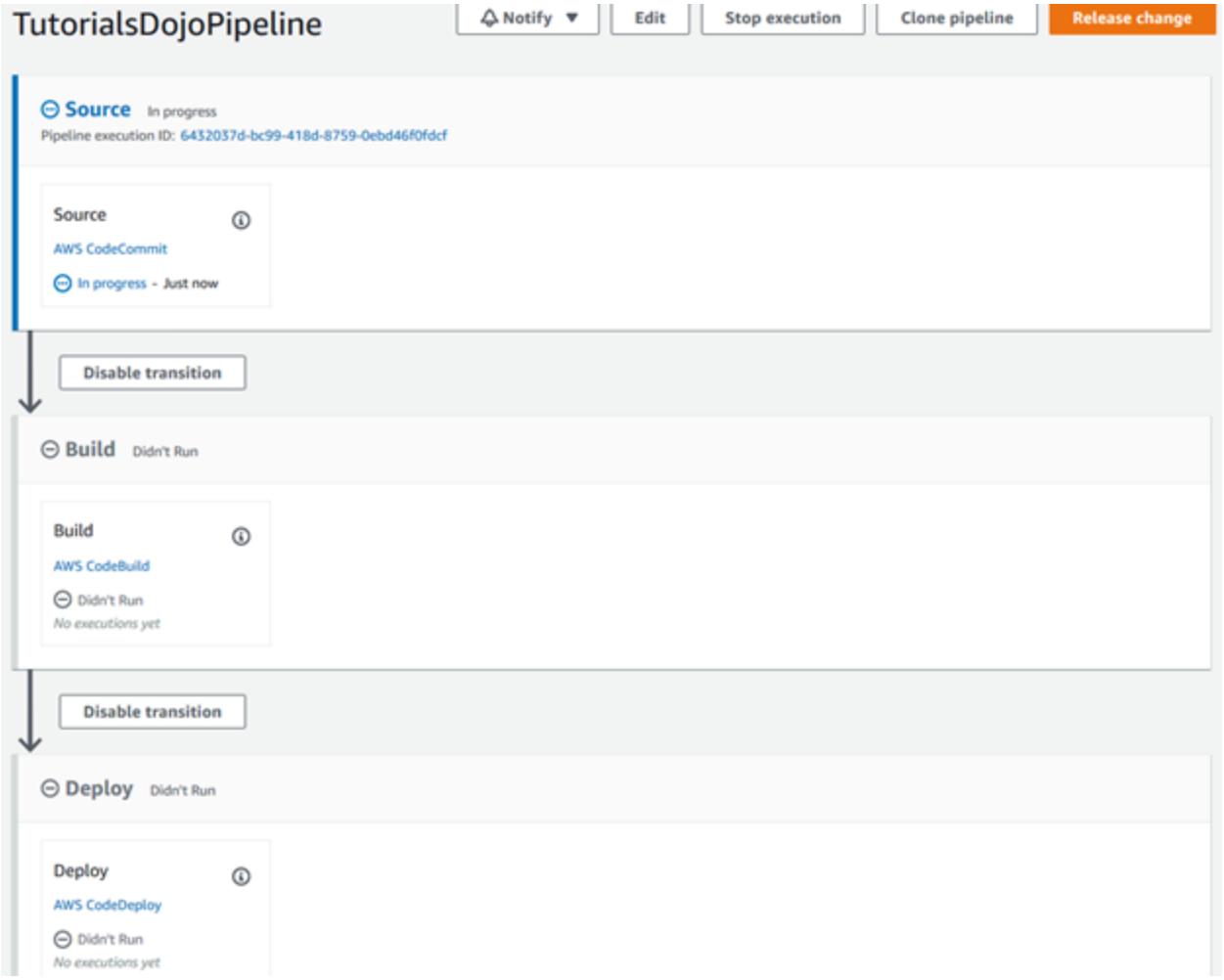
3. Create a build stage such as CodeBuild that will build and test the artifact for you. You must have an existing CodeBuild build project for this.



4. Add a deploy stage using AWS CodeDeploy. The details such as the Application and Deployment group must exist first before you proceed here.



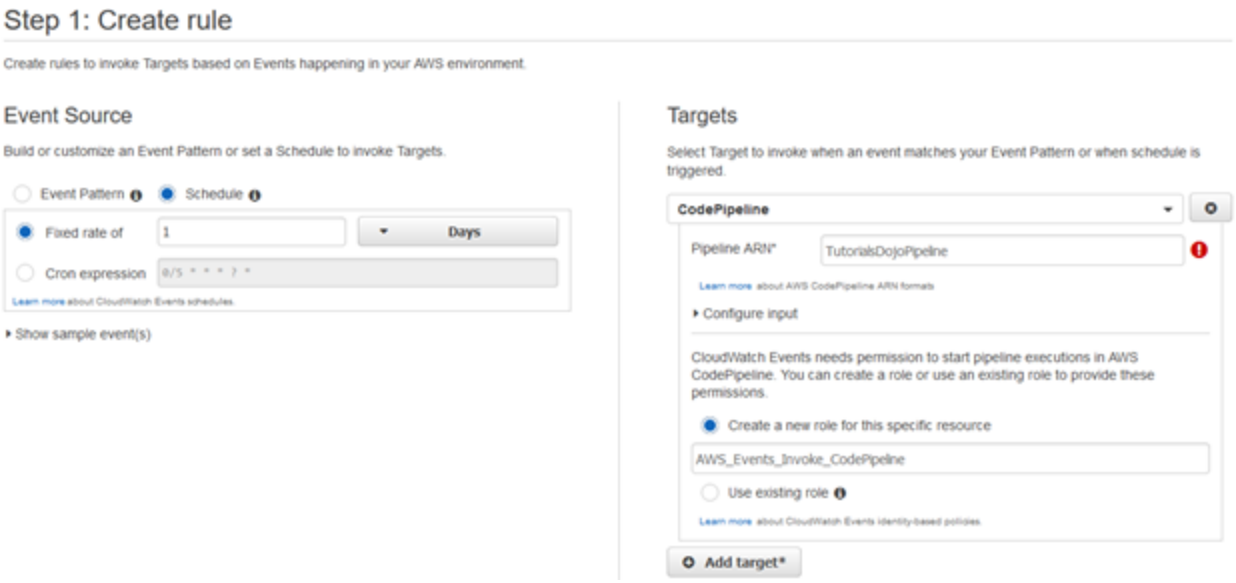
5. After creating the pipeline, you should see your stages, and the pipeline starts the whole process, from CodeCommit, CodeBuild to CodeDeploy. You should be able to see the status of each stage.



Devops Exam Notes:

AWS CodePipeline can be integrated with CloudWatch Events. For example, you have your entire pipeline running on schedule everyday so you will have an updated environment based on the daily progress of development. You can have an AWS CloudWatch Events rule that runs on schedule and have it target an AWS CodePipeline ARN.

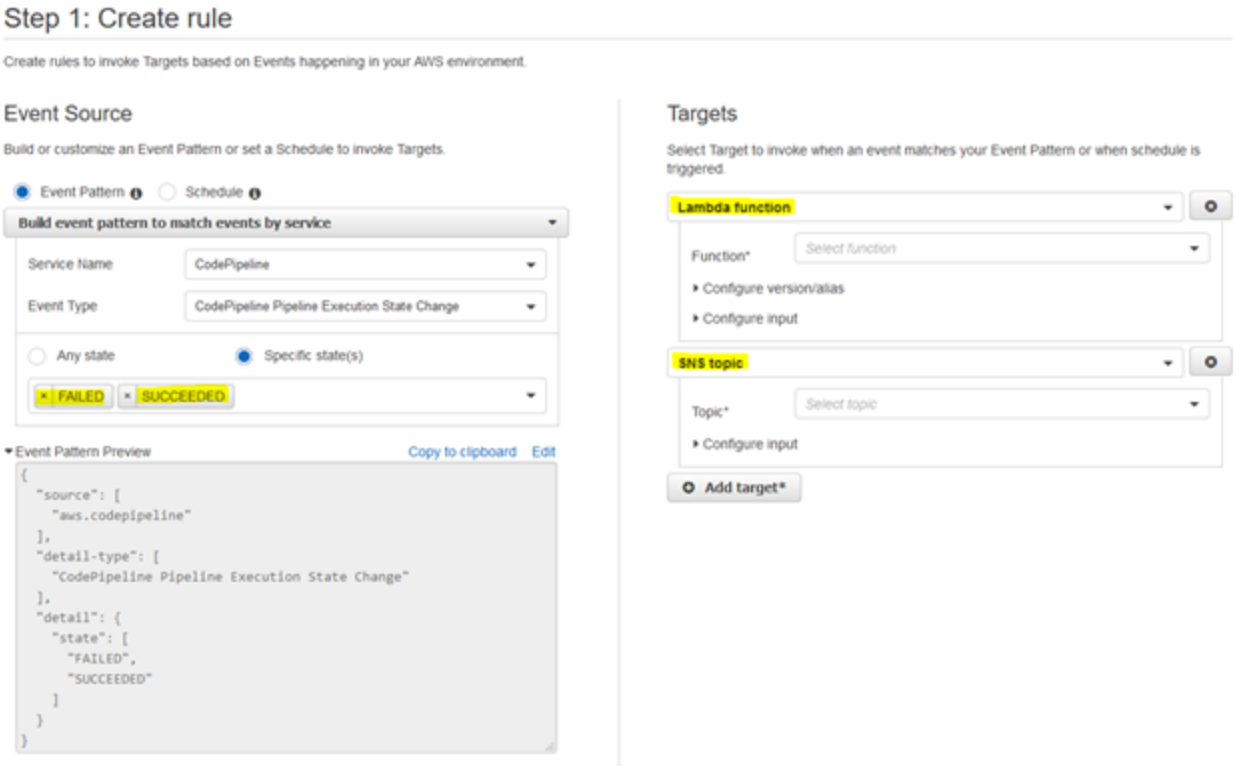
Here's a sample rule on specifying a daily run of a pipeline on AWS CodePipeline:



Devops Exam Notes:

With AWS CloudWatch Events, you can also detect the Status of each stage of the pipeline. You can have a rule for the Pipeline execution state change such as a FAILED stage and have it invoke a Lambda function or an SNS topic to send you an email notification.

Here's a screenshot of a CloudWatch Events rule that targets a Lambda function and an SNS topic.



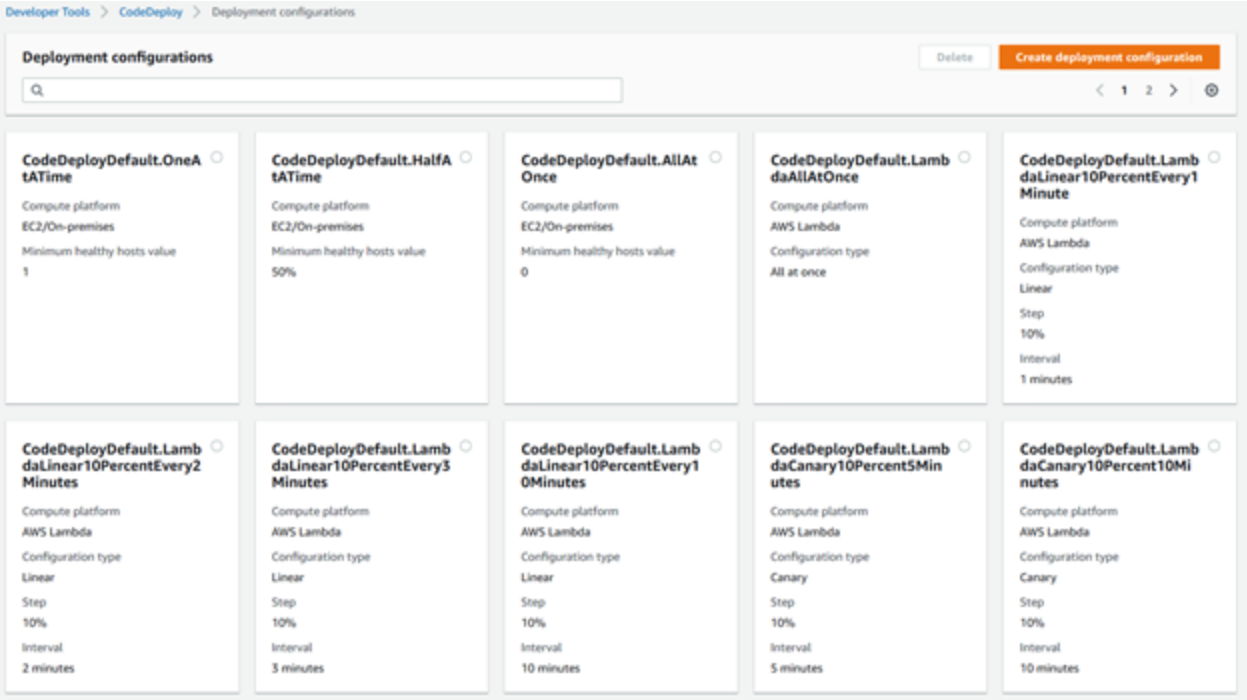
Source:
https://docs.aws.amazon.com/codepipeline/latest/userguide/pipelines-create.html

CodeDeploy - Linear, Canary and All-at-Once (Pre-defined Deployments)

When you deploy a new application version on your target compute platform (EC2 or ECS or Lambda), you can have several options for shifting network traffic to the newer version.

Using CodeDeploy, you can control how many instances are getting updated at any given time during the deployment. This is important because during deployments, the application is stopped while the new version is deployed. You want to make sure that enough instances are online to serve the traffic as well as the ability to roll-back the changes when an error during the deployment occurs.

On the AWS Console, when you click on CodeDeploy > Deployment configurations, you will see the list of AWS defined deployment strategies that you can use for your deployments. You can create your own deployment configuration if you want to, but we'll discuss the most common ones here, which you will likely encounter in the exam.



CodeDeployDefault.AllAtOnce – this is the fastest deployment. The application will stop on all EC2 instances and CodeDeploy will install the newer version on all instances. The application will stop serving traffic during the deployment as all instances are offline.

CodeDeployDefault.OneAtATime – this is the slowest deployment. CodeDeploy will stop only one instance at a time. This will take time to deploy on all instances but the application will remain online since only one instance is offline at any given time.

CodeDeployDefault.HalfAtATime – half, or 50% of the instances will be offline during the deployment, while the other half are online to serve traffic. This is a good balance between a fast and safe deployment.

CodeDeployDefault.LambdaLinear10PercentEvery10Minutes – Deployment for Lambda functions. This deployment will use Aliases on the backend to shift traffic from the old version to the newer version. Ten percent of traffic will be shifted to the newer version every 10 minutes. Deployment will run at 10-minute intervals until 100% of the traffic is shifted to the newer version.

CodeDeployDefault.LambdaCanary10Percent10Minutes – Deployment for Lambda functions. This deployment will use Aliases on the backend to shift traffic from the old version to the newer version. Initially, 10% of the traffic will be shifted to the newer version. This will only last 10 minutes so you can have time to check the application logs. After 10 minutes, the remaining 90% of the traffic will be shifted to the newer version.

Source:
https://docs.aws.amazon.com/codedeploy/latest/userguide/deployment-configurations.html

Elastic Beanstalk - Deployment Policies and Settings

AWS Elastic Beanstalk provides several options for how deployments are processed, including deployment policies (All at once, Rolling, Rolling with additional batch, Immutable, and Traffic splitting) and options that let you configure batch size and health check behavior during deployments. By default, your environment uses all-at-once deployments.

All at once – The quickest deployment method. Suitable if you can accept a short loss of service, and if quick deployments are important to you. With this method, Elastic Beanstalk deploys the new application version to each instance.

Rolling deployments - Elastic Beanstalk splits the environment's Amazon EC2 instances into batches and deploys the new version of the application to one batch at a time. During a rolling deployment, some instances serve requests with the old version of the application, while instances in completed batches serve other requests with the new version.

Rolling deployment with additional batch - launches new batches during the deployment. To maintain full capacity during deployments, you can configure your environment to launch a new batch of instances before taking any instances out of service. When the deployment completes, Elastic Beanstalk terminates the additional batch of instances.

Immutable deployments - perform an immutable update to launch a full set of new instances running the new version of the application in a separate Auto Scaling group, alongside the instances running the old version. Immutable deployments can prevent issues caused by partially completed rolling deployments. If the new instances don't pass health checks, Elastic Beanstalk terminates them, leaving the original instances untouched.

Traffic-splitting deployments - let you perform canary testing as part of your application deployment. In a traffic-splitting deployment, Elastic Beanstalk launches a full set of new instances just like during an immutable deployment. It then forwards a specified percentage of incoming client traffic to the new application version for a specified evaluation period. If the new instances stay healthy, Elastic Beanstalk forwards all traffic to them and terminates the old ones. If the new instances don't pass health checks, or if you choose to abort the deployment, Elastic Beanstalk moves traffic back to the old instances and terminates the new ones.

Here's a summary of the deployment methods, how long each deployment takes, and how rollback is handled.

Method	Impact of failed deployment	Deploy time	Zero downtime	No DNS change	Rollback process	Code deployed to
All at once	Downtime	🕒	No	Yes	Manual redeploy	Existing instances
Rolling	Single batch out of service; any successful batches before failure running new application version	🕒 🕒 †	Yes	Yes	Manual redeploy	Existing instances
Rolling with an additional batch	Minimal if first batch fails; otherwise, similar to **Rolling**	🕒 🕒 🕒 †	Yes	Yes	Manual redeploy	New and existing instances
Immutable	Minimal	🕒 🕒 🕒 🕒	Yes	Yes	Terminate new instances	New instances
Traffic splitting	Percentage of client traffic routed to new version temporarily impacted	🕒 🕒 🕒 🕒 ††	Yes	Yes	Reroute traffic and terminate new instances	New instances
Blue/green	Minimal	🕒 🕒 🕒 🕒	Yes	No	Swap URL	New instances

† *Varies depending on batch size.*

†† *Varies depending on* ***evaluation time*** *option setting.*

Sources:
https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/using-features.rolling-version-deploy.html
https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/using-features.deploy-existing-version.html

Domain 2: Configuration Management and Infrastructure-as-Code

Overview

The second domain of the AWS Certified DevOps Engineer Professional exam focuses on the topics related to configuration management and infrastructure-as-code. You should have working knowledge with the contents of CloudFormation templates and how to use it for deployments, operating with ElasticBeanstalk environments, AWS Lambda deployments, deployments on the Elastic Container Service, and using OpsWorks. To be an effective DevOps Engineer, it is important that you understand these key concepts. Roughly 19% of questions in the actual DevOps exam revolves around these topics.

This domain will challenge your know-how in doing the following:

- Determine deployment services based on deployment needs
- Determine application and infrastructure deployment models based on business needs
- Apply security concepts in the automation of resource provisioning
- Determine how to implement lifecycle hooks on a deployment
- Apply concepts required to manage systems using AWS configuration management tools and services

In this chapter, we will cover all of the related topics for configuration management and infrastructure-as-code in AWS that will likely show up in your DevOps Professional exam.

What is Configuration Management?

Configuration management is the process of standardizing resource configurations and maintaining consistency of your application and server components. Configuration management deals with several areas such as source code repositories, artifact and image repositories, and configuration repositories.

For example, when deploying your application on EC2 instances, you want to make sure that the correct artifact version is deployed and that the required dependencies are installed on the servers. If your new application version requires another dependency package, you should configure all related servers to accommodate this change as well.

If you want to make changes on the OS configuration, for example - an updated logging configuration, you will want to apply it on all running servers as well as to new servers that will be created in the future, plus have the ability to roll back the changes in case you find any errors.

Benefits of configuration management:

Scalability – you don't have to manually configure your servers, such as installation of OS updates, application dependencies, and security compliance configurations. The same process will apply no matter how many servers you have.

Reliability - configuration management offers a reliable way for you to deploy your code. There is central management for your changes and updates so it reduces human errors compared to applying changes manually across your systems.

Disaster Recovery –If you happen to deploy an artifact with bad code or a new config file causing an error, you will have a quick and easy way to rollback since you can go back to the last working version of your configuration.

With proper configuration management tools, you will only have to make changes on your configuration code and it will be applied to all related instances. This process is consistent and scalable in such a way that it's applicable from a range of few instances to several hundreds of instances. Automation is a key component for configuration management so there are several AWS tools available for you.

AWS OpsWorks is a configuration management service that provides managed instances of Chef and Puppet. OpsWorks has three offerings, AWS Opsworks for Chef Automate, AWS OpsWorks for Puppet Enterprise, and AWS OpsWorks Stacks.

AWS OpsWorks for Chef Automate is a fully managed configuration management service that hosts Chef Automate, a suite of automation tools from Chef for configuration management, compliance and security, and continuous deployment.

AWS OpsWorks for Puppet Enterprise is a fully managed configuration management service that hosts Puppet Enterprise, a set of automation tools from Puppet for infrastructure and application management.

AWS OpsWorks Stacks is an application and server management service. With OpsWorks Stacks, you can create a stack containing layers, such as load balancing layer, application layer, and database layer. Within each layer, you can provision Amazon EC2 instances, enable automatic scaling, and configure your instances with Chef recipes using Chef Solo. This allows you to automate tasks such as installing packages and programming languages or frameworks, configuring software, and more.

If you are already using Chef or Puppet for your configuration management, you can easily migrate to AWS with minimal or little changes on your part.

What is Infrastructure-as-Code (IaC)?

Infrastructure-as-code (IaC) takes the concept of configuration management to the next level. Imagine your entire AWS infrastructure and resources described inside a YAML or JSON file. Just like how your application source code outputs an artifact, IaC generates a consistent environment when you apply it.

For example, Infrastructure as Code enables DevOps teams to easily and quickly create test environments that are similar to the production environment. IaC allows you to deliver stability rapidly, consistently, and at scale.

Another example is when you need to create a Disaster Recovery site in another region. With IaC, you can quickly create resources on the new region and be assured that the environment is consistent with the current live environment because everything is defined and described on your JSON or YAML code. You can also save your code to repositories and you can version control it to track changes on your infrastructure. AWS CloudFormation is the main service that you can use if you have codified your infrastructure.

AWS CloudFormation gives developers and systems administrators an easy way to create and manage a collection of related AWS resources, provisioning and updating them in an orderly and predictable fashion. You can use templates to describe the AWS resources and any associated dependencies or runtime parameters required to run your application.

Benefits of using CloudFormation

- **Extensibility** - Using the AWS CloudFormation Registry, you can model, provision, and manage third party application resources alongside AWS resources with AWS CloudFormation.
- **Authoring with JSON/YAML** - allows you to model your entire infrastructure in a text file. You can use JSON or YAML to describe what AWS resources you want to create and configure.
- **Safety controls** - automates the provisioning and updating of your infrastructure in a safe and controlled manner. You can use Rollback Triggers to rollback in case errors are encountered during the update.
- **Preview changes to your environment** - Change Sets allow you to preview how proposed changes to a stack might impact your running resources. For example, whether your changes will delete or replace any critical resources.
- **Dependency management** - automatically manages dependencies between your resources during stack management actions. The sequence of creating, updating, or deleting the dependencies and resources are automatically taken care of.
- **Cross account & cross-region management** - AWS StackSets that lets you provision a common set of AWS resources across multiple accounts and regions with a single CloudFormation template.

Sources:
https://aws.amazon.com/opsworks/
https://aws.amazon.com/cloudformation/

CloudFormation Cross-Stack Reference

CloudFormation allows you to reference resources from one CloudFormation stack and use those resources on another stack. This is called **cross-stack reference**. It allows for a layering of stacks, which is useful for separating your resources based on your services. Instead of putting all resources on one stack, you can create resources from one stack and reference those resources on other CloudFormation stacks.

This also allows you to re-use the same CloudFormation stacks so that you can build faster if you need a new environment with minimal changes.

Example:

- Network stack – contains VPC, public and private subnets, and security groups.
- Web server stack – contains webserver and referencing the public subnets and security groups from the network stack
- Database stack – contains your database server and referencing the private subnets and security groups from the network stack

DevOps Exam Notes:

The requirement for cross stack reference is that you need to export the resources that you want to be referenced by other stacks. Use **Export** on the output Field of your main CloudFormation stack to define the resources that you want to expose to other stacks. On the other stacks, use the **Fn::ImportValue** intrinsic function to import the value that was previously exported.

Here's an example of CloudFormation exporting a subnet and a security group, and referencing it on another CloudFormation stack.

Source:
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/walkthrough-crossstackref.html

Lambda Function Artifact from S3 or CloudFormation Inline

S3 bucket on CloudFormation

Using AWS CloudFormation, you can deploy AWS Lambda functions, which is an easy way to reliably reproduce and version your application deployments.

DevOps Exam Notes:

On your CloudFormation template, the **AWS::Lambda::Function** resource creates a Lambda function. To create a function, you need a deployment package and an execution role. The deployment package contains your function code. The artifact file is defined on the **AWS::Lambda::Function Code** resource.

Here is an example of Node.js lambda function that uses an artifact saved on an S3 bucket on CloudFormation (uses JSON Format).

```
            "AMIIDLookup": {
                  "Type": "AWS::Lambda::Function",
                  "Properties": {
                  "Handler": "index.handler",
                  "Role": {
                        "Fn::GetAtt": [
                            "LambdaExecutionRole",
                              "Arn"
                        ]
                  },
                  "Code": {
                        "S3Bucket": "lambda-functions",
                        "S3Key": "amilookup.zip"
                  },
                  "Runtime": "nodejs12.x",
                  "Timeout": 25,
                  "TracingConfig": {
                        "Mode": "Active"
                  }
                  }
            }
```

Note that Changes to a deployment package in Amazon S3 are not detected automatically during stack updates. To update the function code, change the object key or version in the template. Or you can use a

parameter on your CloudFormation template to input the name of the artifact on S3 that contains the latest version of your code.

If you have a zip file on your local machine, you can use the package command to create a template that generates a template that you can use.

```
aws cloudformation package --template /path_to_template/template.json --s3-bucket mybucket
--output json > packaged-template.json
```

This will save the output on a json template that you can upload on CloudFormation.

Inline Lambda functions in CloudFormation

For Node.js and Python functions, you can specify the function code inline in the template.

> **DevOps Exam Notes:**
>
> Note that you need to use the **AWS::Lambda::Function Code** to define that function and it should be enclosed inside the **ZipFile: |** section to ensure that CloudFormation correctly parses your code.

Here is an example of Node.js lambda function inline on CloudFormation template using the YAML format.

```
AWSTemplateFormatVersion: '2010-09-09'
Description: Lambda function with cfn-response.
Resources:
  primer:
      Type: AWS::Lambda::Function
      Properties:
      Runtime: nodejs12.x
      Role: arn:aws:iam::123456789012:role/lambda-role
      Handler: index.handler
      Code:
      ZipFile: |
            var aws = require('aws-sdk')
            var response = require('cfn-response')
            exports.handler = function(event, context) {
            console.log("REQUEST RECEIVED:\n" + JSON.stringify(event))
            // For Delete requests, immediately send a SUCCESS response.
            if (event.RequestType == "Delete") {
                  response.send(event, context, "SUCCESS")
                  return
            }
```

```
                var responseStatus = "FAILED"
                var responseData = {}
                var functionName = event.ResourceProperties.FunctionName
                var lambda = new aws.Lambda()
                lambda.invoke({ FunctionName: functionName }, function(err,
      invokeResult) {
                      if (err) {
                            responseData = {Error: "Invoke call failed"}
                          console.log(responseData.Error + ":\n", err)
                      }
                      else responseStatus = "SUCCESS"
                      response.send(event, context, responseStatus, responseData)
                })
                }
          Description: Invoke a function during stack creation.
          TracingConfig:
          Mode: Active
```

Sources:
https://docs.aws.amazon.com/lambda/latest/dg/deploying-lambda-apps.html
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/using-cfn-cli-package.html

AutoScalingReplacingUpdate vs AutoScalingRollingUpdate Policy

When using AWS CloudFormation to provision your Auto Scaling groups, you can control how CloudFormation handles updates for your Auto Scaling Group. You need to define the proper **UpdatePolicy** attribute for your ASG depending on your desired behavior during an update.

DevOps Exam Notes:

You can use **AWS::AutoScaling::AutoScalingGroup** resource type on CloudFormation if you want to create an AutoScaling group for your fleet of EC2 instances. Going into the exam, you will need to distinguish the difference between the **AutoScalingReplacingUpdate** and **AutoScalingRollingUpdate** UpdatePolicy attribute, which define how the instances on your group will be updated when you deploy a new revision of your application.

AutoScalingReplacingUpdate - will create a new auto scaling group with new launch configuration. This is more like an immutable type of deployment.

AutoScalingRollingUpdate - will replace the instances on the current auto-scaling group. You can control if instances will be replaced "all-at-once" or use a rolling update by batches. The default behavior is to delete instances first, before creating the new instances.

The **AutoScalingReplacingUpdate** policy specifies how AWS CloudFormation handles replacement updates for an Auto Scaling group. This policy enables you to specify whether AWS CloudFormation replaces an Auto Scaling group with a new one or replaces only the instances in the Auto Scaling group.

```
"UpdatePolicy" : {
  "AutoScalingReplacingUpdate" : {
    "WillReplace" : Boolean
  }
}
```

For example, you can set **AutoScalingReplacingUpdate WillReplace** property to TRUE to have CloudFormation retain the old ASG and the instances it contains. CloudFormation will wait for the successful creation of the new ASG and its instances before it deletes the old ASG. This is helpful when the update fails; CloudFormation can quickly rollback as it will only delete the new ASG. The current ASG and its instances are not affected during the deployment and rollback process.

The **AutoScalingRollingUpdate** policy specifies how AWS CloudFormation handles rolling updates for an Auto Scaling group. Rolling updates enable you to specify whether AWS CloudFormation updates instances that are in an Auto Scaling group in batches or all at once.

```
"UpdatePolicy" : {
  "AutoScalingRollingUpdate" : {
      "MaxBatchSize" : Integer,
      "MinInstancesInService" : Integer,
      "MinSuccessfulInstancesPercent" : Integer,
      "PauseTime" : String,
      "SuspendProcesses" : [ List of processes ],
      "WaitOnResourceSignals" : Boolean
  }
}
```

For example, **AutoScalingRollingUpdate** allows you to specify the **MaxBatchSize** property to set the maximum number of instances that AWS CloudFormation updates at any given time. Or use the **MinInstancesInService** property to ensure that there is a minimum number of instances that are in service while CloudFormation updates the old instances.

Sources:
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-attribute-updatepolicy.html
https://aws.amazon.com/premiumsupport/knowledge-center/auto-scaling-group-rolling-updates/

Time-Based vs Load-Based Instance

With OpsWorks stacks, you can create a layer of auto scaling EC2 instances and a load balancer to host your application and service traffic from the public Internet. You can set up AWS OpsWorks Stacks to start or stop EC2 instances based on the configuration of your instances. You can use this to automatically increase/decrease the size of your EC2 cluster depending on when you need computing power. This automatic scaling is based on two instance types:

1. **Time-based instances -** allow a stack to handle loads that follow a predictable pattern by including instances that run only at certain times or on certain days. This is helpful in situations when you want to start some instances after 6PM to perform nightly backup tasks or stop some instances on weekends when traffic is lower.

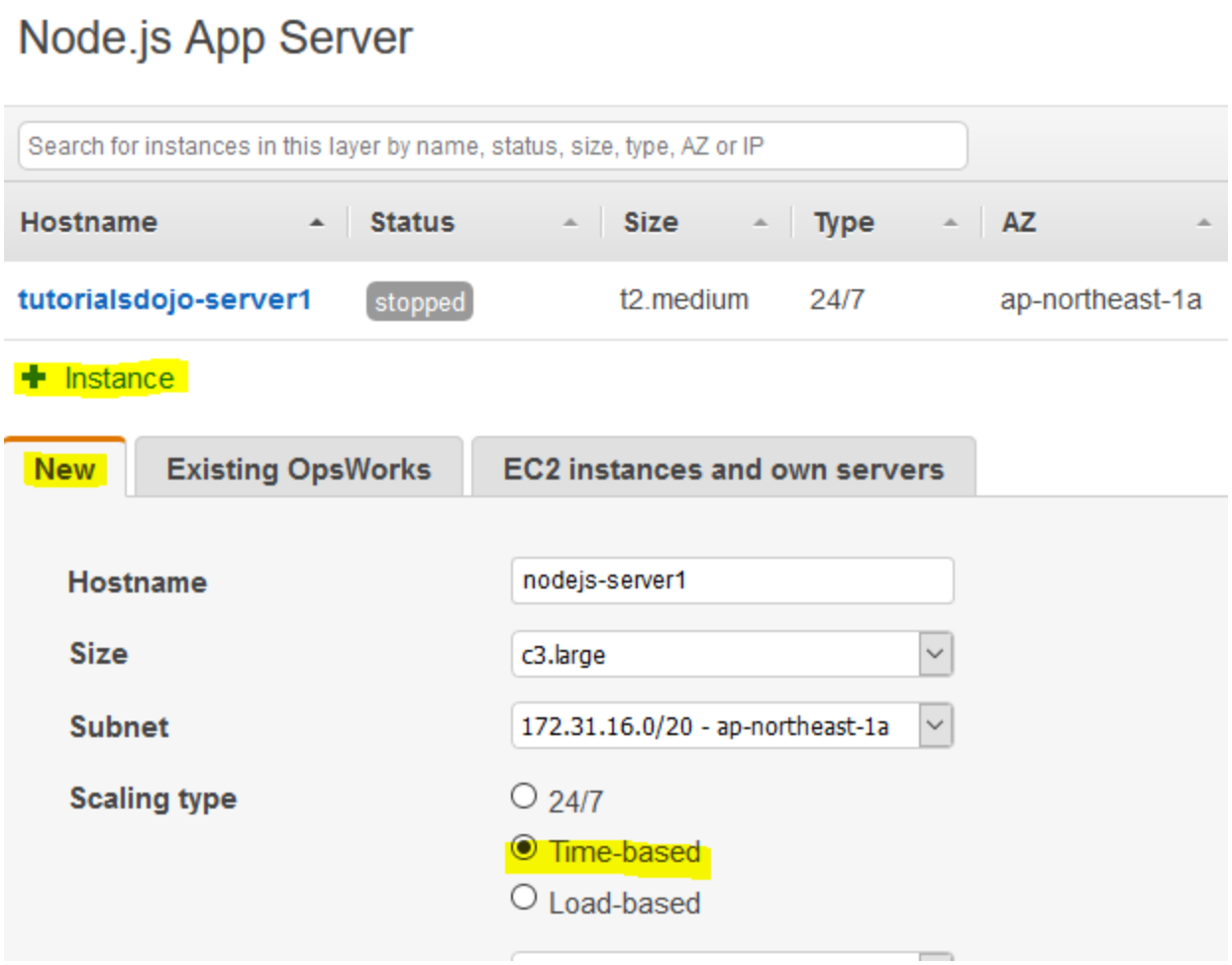
2. **Load-based instances -** allow a stack to handle variable loads by starting additional instances when traffic is high and stopping instances when traffic is low, based on any of several load metrics. This is helpful in situations where you want to start instances when the average CPU utilization exceeds 80% and stop instances when the average CPU load falls below 60%.

A common practice is to use all three instance types together, as follows:

- A set 24/7 instances to handle the base load. You typically just start these instances and let them run continuously.
- A set of time-based instances, which AWS OpsWorks Stacks starts and stops to handle predictable traffic variations.
- A set of load-based instances, which AWS OpsWorks Stacks starts and stops to handle unpredictable traffic variations.

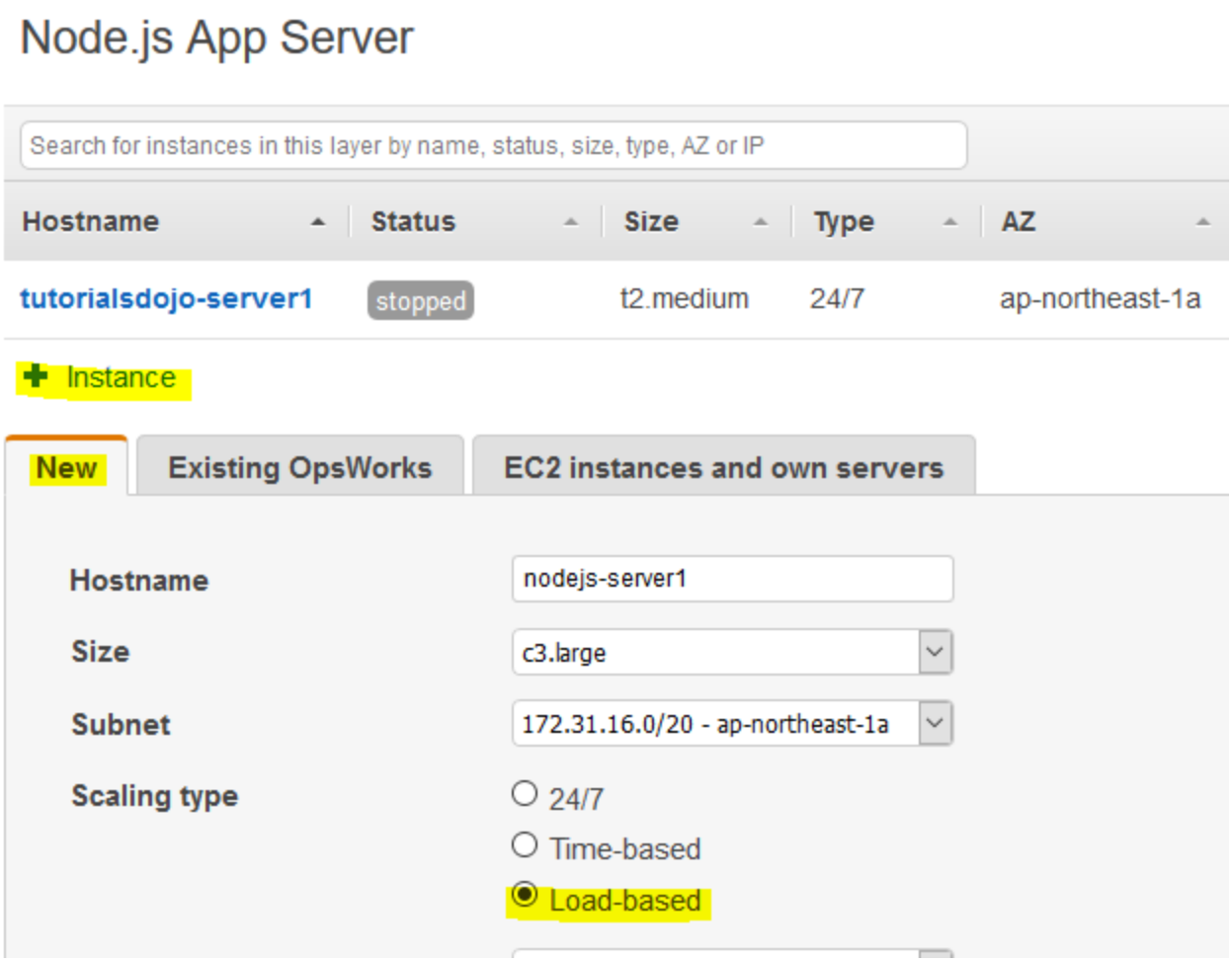
If you want to add a **time-based instance** to a layer follow these steps:

Click the Instances page, click + Instance to add an instance. On the New tab, click Advanced >> and then click time-based.



If you want to add a **load-based instance** to a layer:

On the Instances page, click +Instance to add an instance. Click Advanced >> and then click load-based.



Source:
https://docs.aws.amazon.com/opsworks/latest/userguide/workinginstances-autoscaling.html

Discovery Agent vs Discovery Connector

Application Discovery Agent and Agentless Discovery Connector are helpful tools in the AWS Application Discovery Service to help you plan your migration from on-premises servers and VMs to AWS cloud. Here's a quick description and summary of differences between the two.

The **Application Discovery Agent** is a software that you install on on-premises servers and VMs targeted for discovery and migration. The agent is needed by AWS Discovery Service to help you plan your migration to AWS cloud by collecting usage and configuration data about your on-premises servers. The Agent captures system configuration, system performance, running processes, and details of the network connections between systems.

You can then view the discovered servers, group them into applications, and then track the migration status of each application from the AWS Migration Hub console.

If you can't install the agent on your on-premises servers, AWS Application Discovery Service offers another way of performing discovery through the AWS **Agentless Discovery Connector**. This agentless discovery is performed by deploying an OVA file in VMware vCenter.

The Discovery Connector identifies virtual machines (VMs) and hosts associated with vCenter, collects static configuration data such as Server hostnames, IP addresses, MAC addresses, and disk resource allocations. Additionally, it collects the utilization data for each VM and computes average and peak utilization for metrics such as CPU, RAM, and Disk I/O.

DevOps Exam Notes:

Know the differences between **Application Discovery Agent** and **Agentless Discovery Connector**.

- **Application Discovery Agent** - agent package is installed on on-premises VMs and servers for migration.
- **Agentless Discovery Connector** - standalone VM to be deployed on on-premises data center to collect information for migration.

Sources:
https://docs.aws.amazon.com/application-discovery/latest/userguide/what-is-appdiscovery.html
https://docs.aws.amazon.com/application-discovery/latest/userguide/discovery-agent.html
https://docs.aws.amazon.com/application-discovery/latest/userguide/discovery-connector.html

CloudFormation Template for ECS, Auto Scaling and ALB

Amazon Elastic Container Service (ECS) allows you to manage and run Docker containers on clusters of EC2 instances. You can also configure your ECS to use Fargate launch type which eliminates the need to manage EC2 instances.

With CloudFormation, you can define your ECS clusters and tasks definitions to easily deploy your containers. For high availability of your Docker containers, ECS clusters are usually configured with an auto scaling group behind an application load balancer. These resources can also be declared on your CloudFormation template.

DevOps Exam Notes:

Going on to the exam, be sure to remember the syntax needed to declare your ECS cluster, Auto Scaling group, and application load balancer. The **AWS::ECS::Service** resource creates an ECS cluster and the **AWS::ECS::TaskDefinition** resource creates a task definition for your container. The **AWS::ElasticLoadBalancingV2::LoadBalance**r resource creates an application load balancer and the **AWS::AutoScaling::AutoScalingGroup** resource creates an EC2 auto scaling group.

AWS provides an example template which you can use to deploy a web application in an Amazon ECS container with auto scaling and application load balancer. Here's a snippet of the template with the core resources:

```
{
  "AWSTemplateFormatVersion":"2010-09-09",
  "Resources":{
       "ECSCluster":{
       "Type":"AWS::ECS::Cluster"

       },

……

       "taskdefinition":{
       "Type":"AWS::ECS::TaskDefinition",

       "Properties":{

……

       "ECSALB":{
       "Type":"AWS::ElasticLoadBalancingV2::LoadBalancer",
```

```
        "Properties":{

……

        "ECSAutoScalingGroup":{
        "Type":"AWS::AutoScaling::AutoScalingGroup",
        "Properties":{
        "VPCZoneIdentifier":{
              "Ref":"SubnetId"

        },
```

Sources:
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/quickref-ecs.html
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-ecs-service.html
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-ecs-service-loadbalancers.html

Domain 3: Monitoring and Logging

Overview

The third exam domain of the AWS Certified DevOps Engineer Professional test is all about the process of monitoring and logging resources in AWS. You must learn how to set up advanced monitoring configurations using Amazon CloudWatch, AWS Systems Manager, AWS CloudTrail and other related services. The task of aggregating logs and metrics across multiple AWS accounts and regions is also covered. Roughly 15% of questions in the actual DevOps exam revolves around these topics.

This domain will challenge your know-how in doing the following:

- Determine how to set up the aggregation, storage, and analysis of logs and metrics
- Apply concepts required to automate monitoring and event management of an environment
- Apply concepts required to audit, log, and monitor operating systems, infrastructures, and applications
- Determine how to implement tagging and other metadata strategies

In this chapter, we will cover all of the related topics for monitoring and logging in AWS that will likely show up in your DevOps exam.

AWS Config Multi-Account Multi-Region Data Aggregation

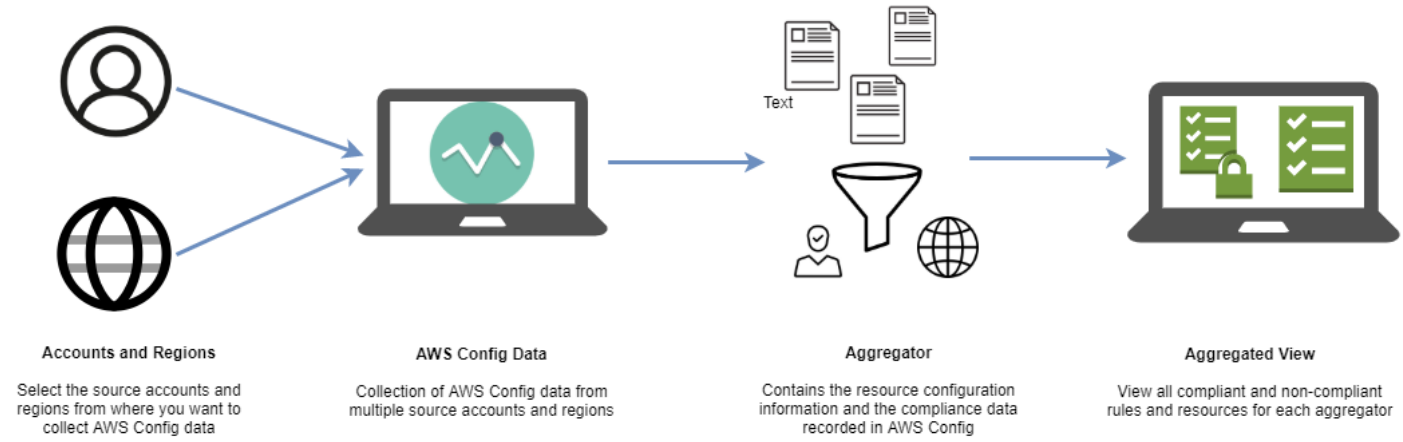
AWS Config enables you to monitor, audit, and evaluate the configurations of your AWS resources. It allows you to track and review the configuration changes of your resources as well as determine your overall compliance against your internal IT policies. You can also use AWS Systems Manager Automation to automatically remediate noncompliant resources that were evaluated by your AWS Config Rules.

This service is a great monitoring and automatic remediation tool for your AWS resources. However, there are certain limitations that you should know about AWS Config. The scope of this service is regional which means that it can only monitor the AWS resources on a specific region. It is usually enabled on a per-region basis on your AWS account. This poses a problem if your organization is using multiple AWS regions and accounts.

You can use the multi-Account, multi-Region data aggregation capability of AWS Config if you want to centralize the auditing and governance of your ubiquitous cloud resources. This functionality reduces the time and overhead required to collect an enterprise-wide view of your compliance status. It provides you with a single, aggregated view of your AWS resources across regions, accounts and even your AWS Organizations. To do this, you have to create an *Aggregator* first and specify the regions where you want to collect data from.

An *Aggregator*, as its name suggests, is a resource of AWS Config that collects or groups data together. It replicates data from the specified source accounts into the aggregator AWS account where the aggregated view will be used. The aggregator account has access to the resource configuration and compliance details for multiple accounts and regions. This is a type of an AWS Config resource that gathers AWS Config configuration and compliance data from the following:

- A single AWS account with resources from multiple AWS Regions.
- Multiple AWS accounts where each account uses multiple AWS Regions.
- The master and member accounts of an organization entity in AWS Organizations.



You can use an aggregator to view the resource configuration and compliance data recorded in AWS Config.

DevOps Exam Notes:

AWS Config is a regional resource. If you want to implement a centralized monitoring system of your resources across various AWS Regions and AWS accounts, you have to set up data aggregation using an *Aggregator* in AWS Config.

Remember that **AWS AppConfig** and **AWS Config** are two different services! The former is a capability of AWS Systems Manager that assists you in managing, storing, and deploying your application configurations to your Amazon EC2 instances at runtime. The latter is a configuration management service that helps you monitor, audit, and evaluate the configurations of your AWS resources.

Consolidating VPC Flow Logs From Multiple Sources

VPC Flow Logs is a feature in AWS that allows you to capture information about the incoming and outgoing IP traffic of the network interfaces in your Amazon VPC. Flow logs can assist you to properly monitor and log the activities in your VPC. It can diagnose overly restrictive security group or network ACL rules, monitor the incoming traffic to your EC2 instances, and determine the flow of traffic to and from the network interfaces. After you've created a flow log, you can retrieve and view its data in the chosen destination.

A flow log data can be published to these destinations:

- Amazon CloudWatch Logs
- Amazon S3

Large companies often have multiple AWS accounts and use multiple VPCs for their cloud architecture. Monitoring the IP traffic flow could be difficult for a complex and extensive network architecture since the scope of the flow logs is only within a single VPC. You can enable flow logs for the VPCs that are peered with

your VPC as long as the peer VPC is in your AWS account. However, VPC Peering is still not enough to build a centralized logging for multi-account environments with different types of network configurations.

Storing all the logs data to Amazon S3 is a strategy that you can adopt to consolidate every flow log from across all VPCs that you own. The flow logs of your VPC can be published to an Amazon S3 bucket that you specify. The collected flow log records for all of the monitored network interfaces are sent to a series of log file objects stored in the S3 bucket. In this way, all of your logs are in one place which lessen the management overhead.

The buckets and the objects in Amazon S3 are private by default and only the bucket owner can access data stored in it. You must grant the proper access and modify the bucket policy to allow the delivery.logs.amazonaws.com service to send and store the logs to the centralized S3 bucket.

You can also use Amazon Athena to easily query the flow log records in the log files stored in the centralized Amazon S3 bucket. Amazon Athena is an interactive query service that simplifies the process of analyzing data in Amazon S3 using standard SQL. The built-in Amazon S3 Select capability can also be used to fetch the logs based on a simple structured query language (SQL) statement, but this is quite limited and can only query a subset of your data. Therefore, using Amazon Athena is the preferred service to analyze the unified data instead of Amazon S3 Select.

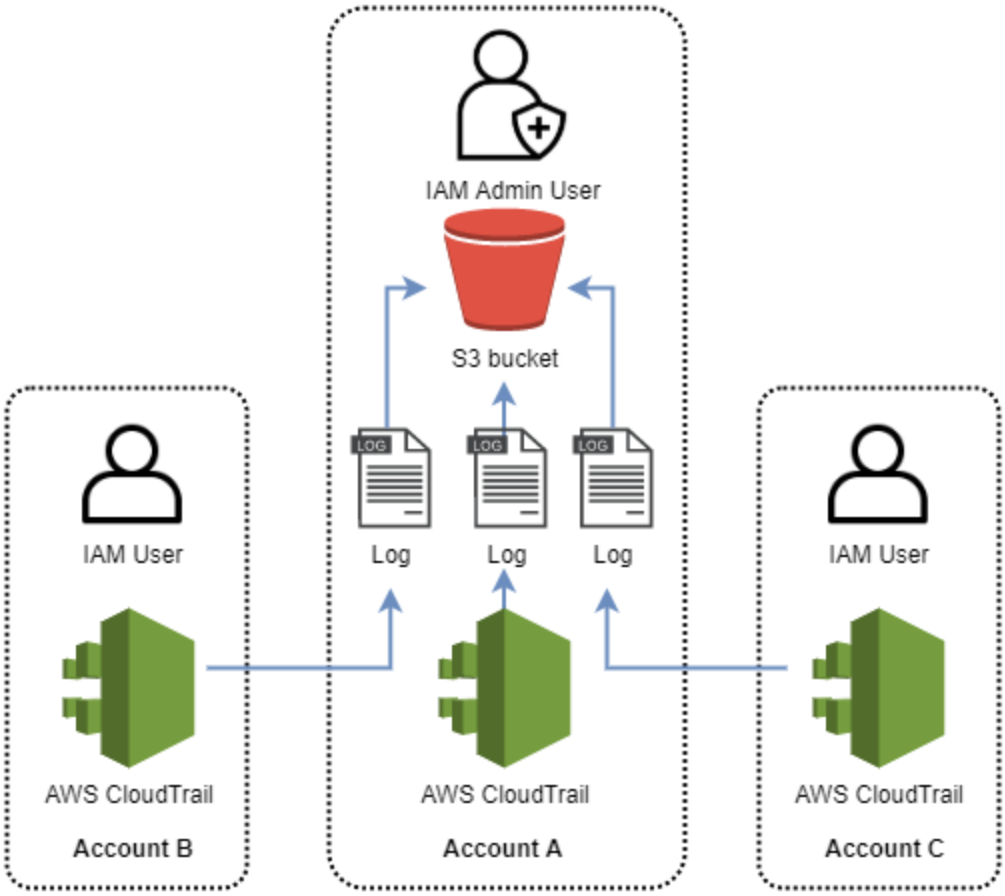
Remember that VPC Flow logs do not capture each and every IP traffic. The following items are not logged:

- Traffic to and from the instance metadata (169.254.169.254)
- Traffic to and from the Amazon Time Sync Service (169.254.169.123)
- Dynamic Host Configuration Protocol (DHCP) traffic.
- For the default VPC router, the traffic to the Reserved IP address is not logged.
- Traffic between a Network Load Balancer (ELB) network interface and an endpoint network interface (ENI).
- Traffic generated by an Amazon EC2 Windows instance for the Windows license activation.
- Traffic generated by the Amazon EC2 instances when they connect to the Amazon DNS server. However, if you use your own BIND DNS server, all traffic to that DNS server is logged by the VPC Flow Logs.

Consolidating CloudTrail Log Files from Multiple Sources

AWS CloudTrail is the primary service used for auditing your AWS resources. It provides an event history of all of your AWS account activity, including the actions taken through the AWS SDKs, AWS CLI, AWS Management Console, and other AWS services. However, it can only track the API calls made on a single AWS account. If your company has multiple AWS accounts, you can consolidate them into a single organizational unit (OU) using AWS Organizations. With this, you can create a trail that will collect all events for all AWS accounts in that organization. This is often referred to as an "organization trail" in AWS CloudTrail. The organization trail logs events for the master account and all member accounts in the organization.

However, some companies have complex cloud architectures that hinder them from using AWS Organizations. Businesses may have two or more external AWS accounts that belong to their subsidiaries or partners. To support this use case, you can configure AWS CloudTrail to send the log files from multiple AWS accounts into a single Amazon S3 bucket for centralized logging.



For example, there are four AWS accounts owned by your company: Manila account, New York account, Singapore account, and London account, that you want to effectively monitor. These AWS accounts are separate business units which handle their own billing. Using AWS Organizations to consolidate billing and trail logs is not applicable due to the organizational constraints. But alternatively, you can configure CloudTrail to deliver log files from all four accounts to an S3 bucket that belongs to a central AWS Account that you specify.

This can be done with these simple steps:

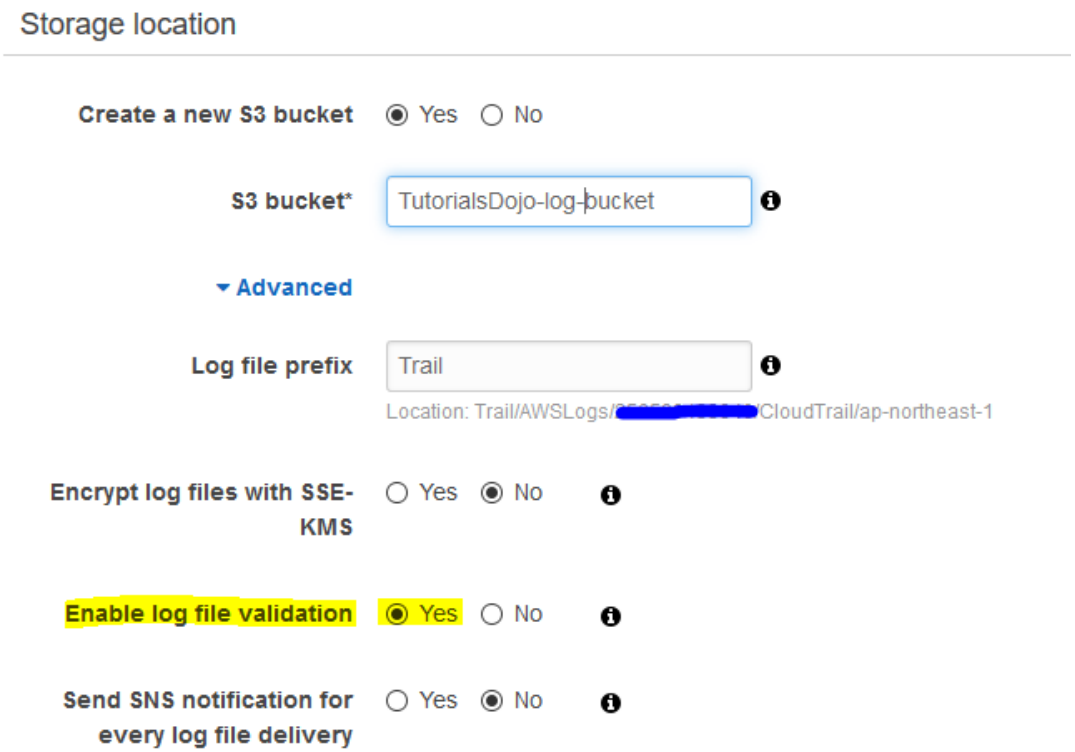
1. Enable AWS CloudTrail in the AWS account where the destination bucket will belong *(e.g. tutorialsdojo-trail S3 bucket in the Manila AWS account)*. You may refer to this as your "central" or top-level account. Make sure that CloudTrail is disabled on the other accounts.
2. Modify the S3 bucket policy on the destination bucket to grant cross-account permissions to AWS CloudTrail.
3. Enable AWS CloudTrail in the other accounts that you want to include. Configure CloudTrail in these AWS accounts to use the same S3 bucket, which belongs to the AWS account that you specified in the first step.

Sources:
https://docs.aws.amazon.com/vpc/latest/userguide/flow-logs-s3.html
https://aws.amazon.com/blogs/architecture/stream-amazon-cloudwatch-logs-to-a-centralized-account-for-audit-and-analysis/

Ensuring the Integrity of the CloudTrail Log Files

The trail logs produced by AWS CloudTrail are invaluable for conducting security and compliance audits, including forensic investigations. It can attest that a particular user credential performed a specific API activity. For IT audit activities, the trail logs in CloudTrail can be used as proof that your AWS infrastructure complies with the specified set of operational guidelines. But what if these logs were modified and deleted? How can you secure and validate the integrity of your trail logs?

To protect your trail data, you can enable the 'log file integrity validation' feature in the CloudTrail via the AWS Management console, CloudTrail API, or AWS CLI. This feature verifies whether a trail log file was modified, removed, or kept unchanged after CloudTrail sent it to the S3 bucket. AWS CloudTrail tracks the changes of each trail log using a separate digest file, which will also be stored in the S3 bucket. This digest file contains the digital signatures and hashes used to validate the integrity of the log files.

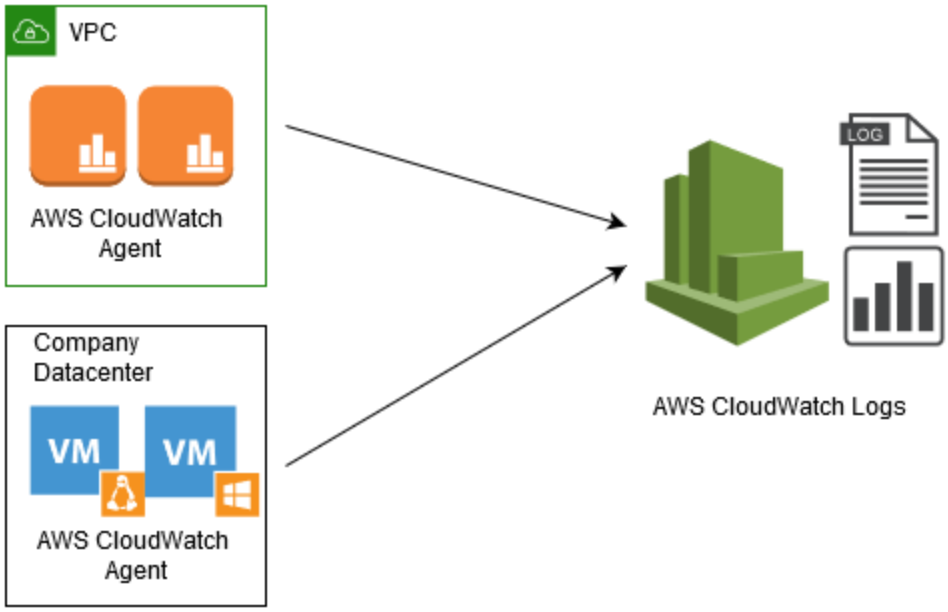


AWS CloudTrail uses SHA-256 for hashing and SHA-256 with RSA for digital signing for log file integrity validation. These industry-standard algorithms make it computationally infeasible to modify and delete the CloudTrail log files. The digest file is signed by AWS CloudTrail using the private key of a public and private key pair. The public key can be used to validate the digest file for a specific AWS Region.

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:

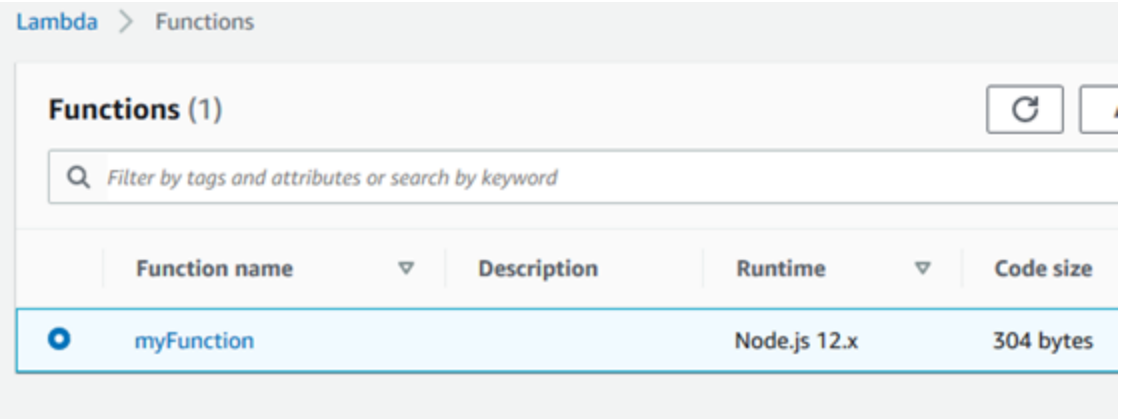
Additionally, you can use CloudWatch Logs subscriptions to get access to a real-time feed of log events from CloudWatch Logs and have it delivered to other services such as an Amazon Kinesis stream, an Amazon Kinesis Data Firehose stream, or AWS Lambda for custom processing, analysis, or loading to other 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.

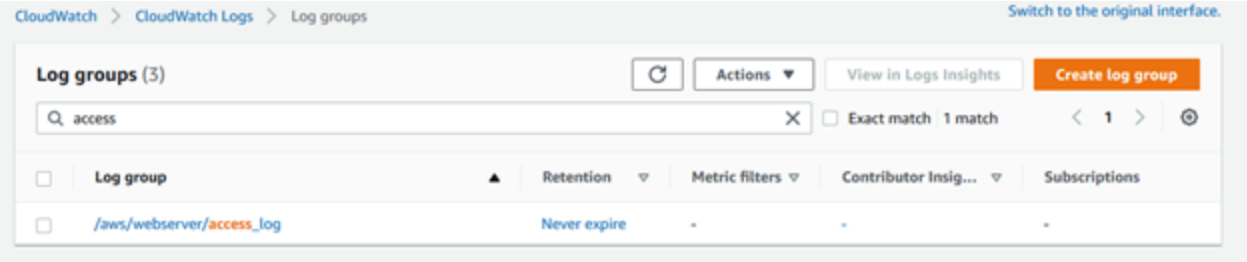
Here are the steps to setup CloudWatch logs subscription:

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

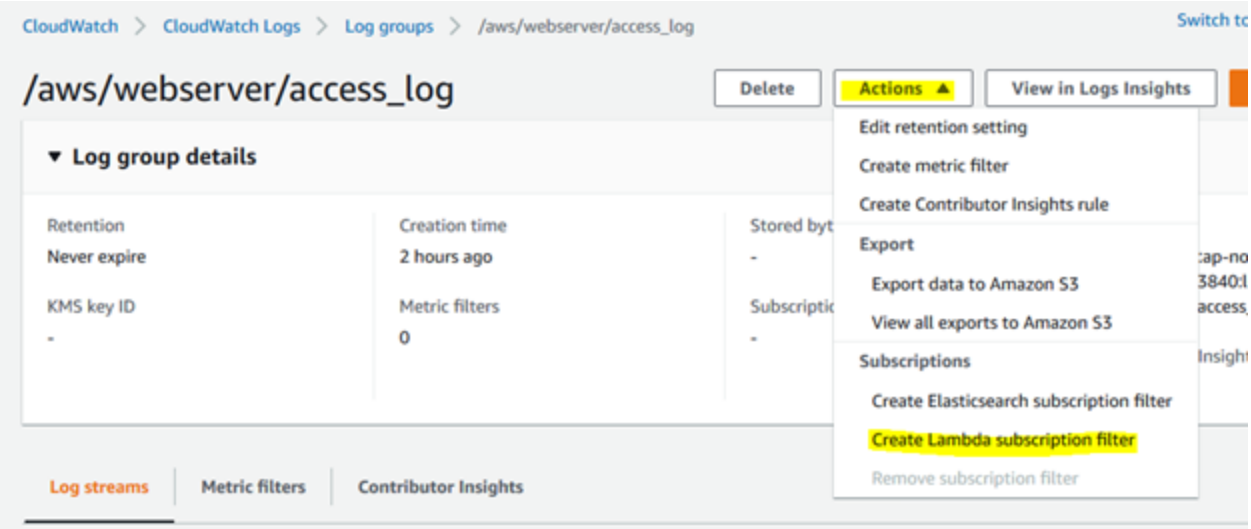


2. Install CloudWatch Unified Agent on the EC2 instance (Linux or Windows) and configure it to send application logs to CloudWatch log group.

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.

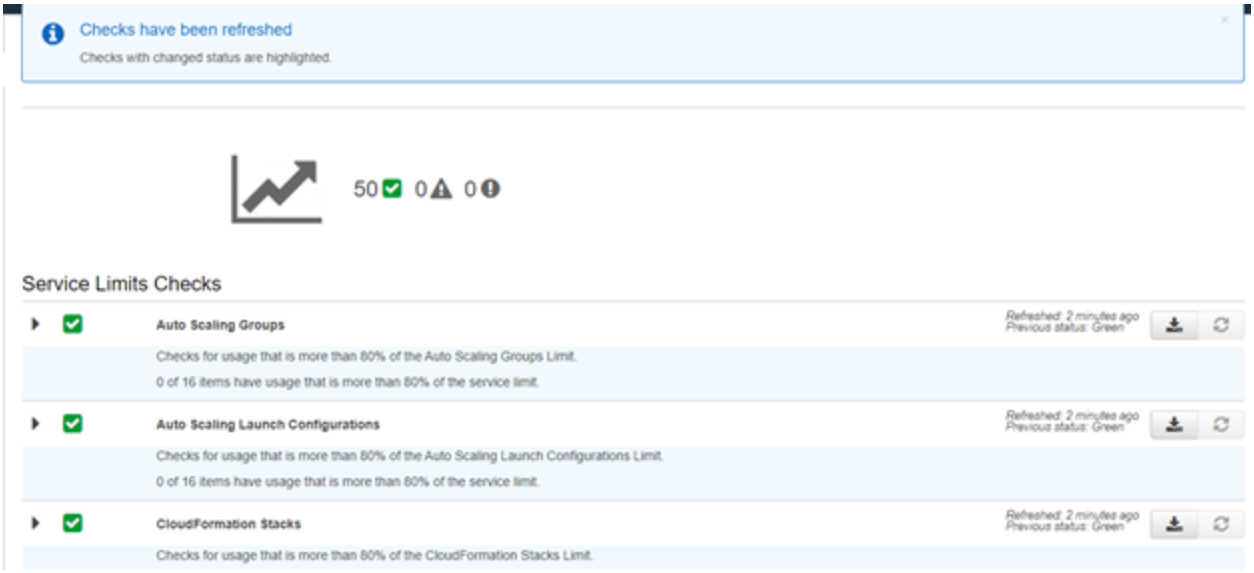
With Trusted Advisor's Service Limit Dashboard, you can view, refresh, and export utilization and limit data on a per-limit basis. These metrics are published on AWS CloudWatch in which you can create custom alarms based on percentage of service utilization against limits, understand the number of resources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

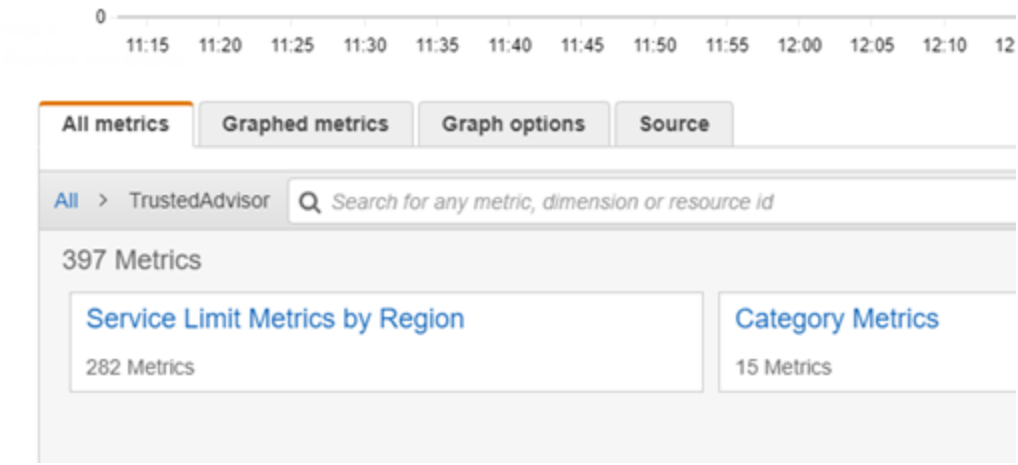
Here's how you can create a CloudWatch Alarm to detect if you are nearing your auto scaling service limit and send a notification so you can request for a service limit increase to AWS support.

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.

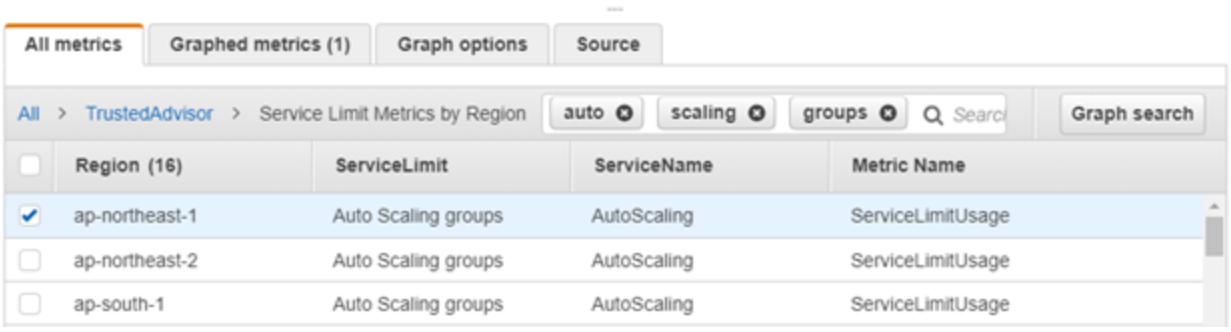


2. Go to CloudWatch > Alarms. Make sure that you are in the North Virginia region. Click the “Create Alarm” button and click “Select Metric”.

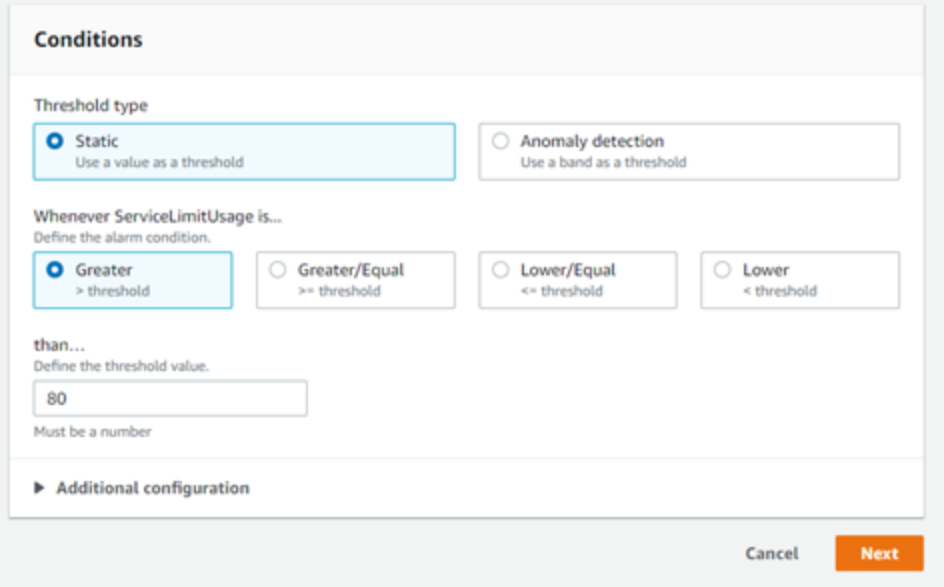
3. In the “All metrics” tab, click “Trusted Advisor” category and you will see “Service Limits by Region”.



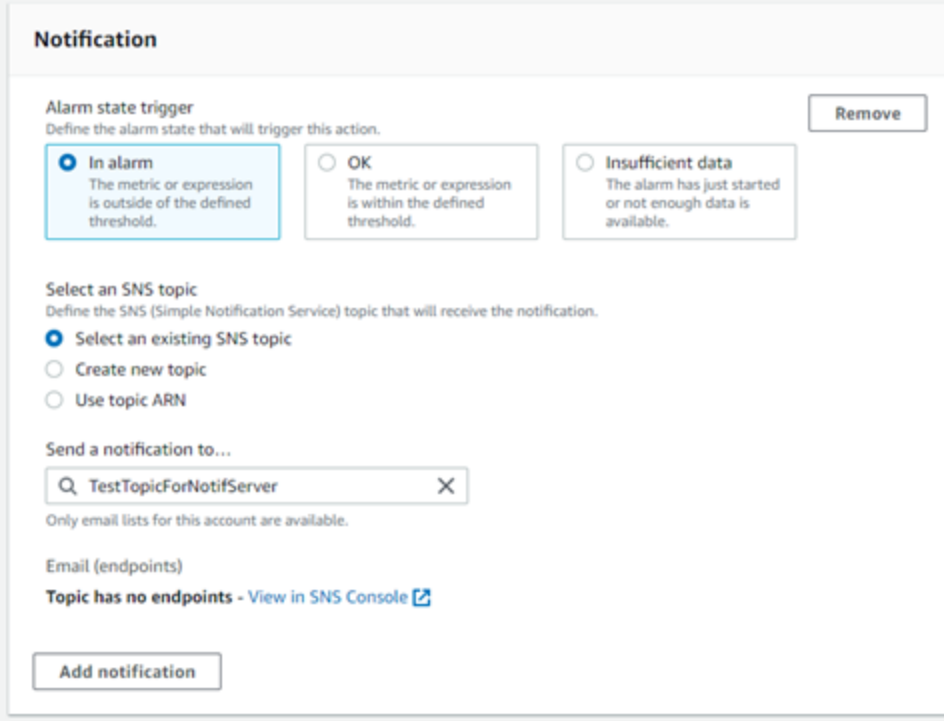
4. Search for Auto Scaling groups on your desired region and click Select Metric.



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/

Domain 4: Policies and Standards Automation

Overview

The fourth exam domain of the AWS Certified DevOps Engineer Professional exam deals with the automation of policies and standards in your architecture to enforce logging, metrics, monitoring, testing, and security. Since it covers policies and governance, the related AWS services that you have to review are AWS Organizations, AWS Config, AWS Service Catalog, and AWS Identity and Access Management (IAM). This domain will test your knowledge on the following:

- Applying concepts required to enforce standards for logging, metrics, monitoring, testing, and security
- Determining how to optimize cost through automation
- Applying concepts required to implement governance strategies

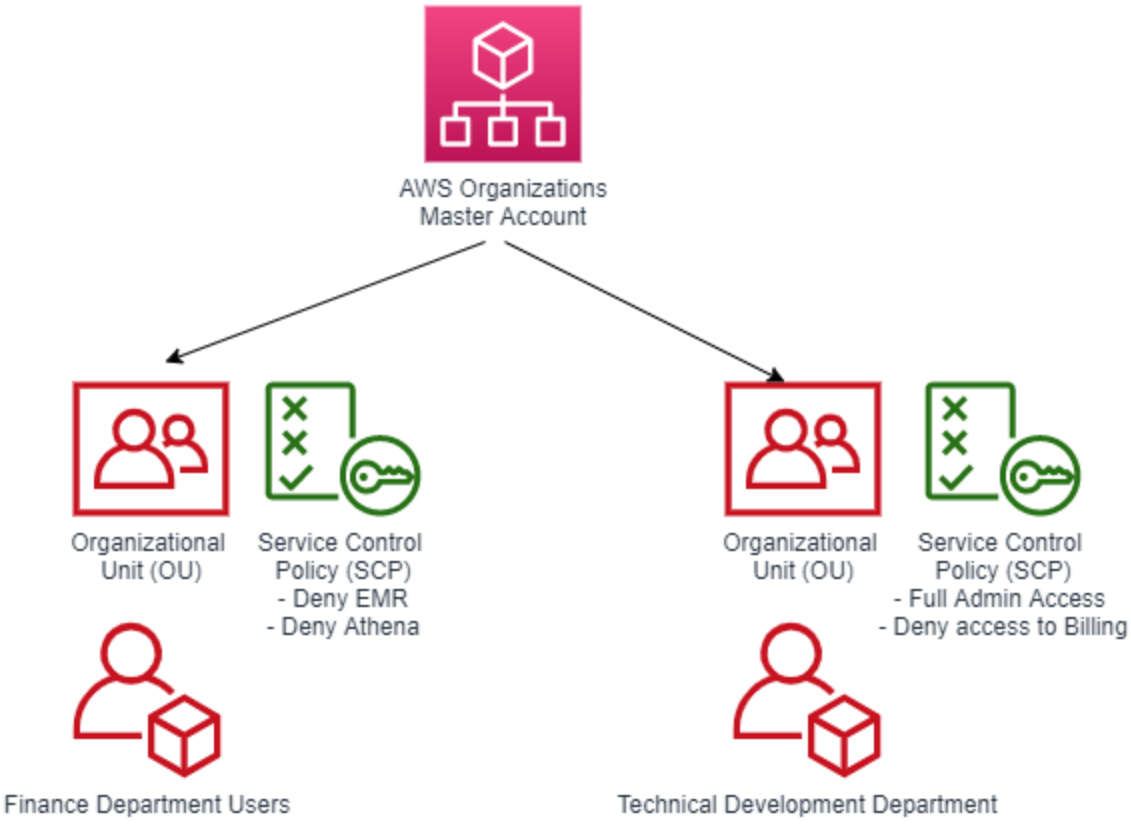
This has the least amount of weight in the exam (10%) thus, you can limit the time you invest in reviewing this section.

Management and Governance on AWS

Every company has its own standard operating procedures and policies that need to be enforced to its infrastructure to ensure efficient operations and comply with industry regulations. For example, healthcare organizations in the US must abide by the privacy and security standards of the Health Insurance Portability and Accountability Act of 1996 (HIPAA). Other companies also follow the data protection and privacy regulations covered in the General Data Protection Regulation (GDPR), which addresses the transfer of personal data outside the European Union (EU) and other European Economic Area (EEA). Failure to conform to these policies entails millions of dollars in fines and potential loss of revenue. Implementing strict security standards and procedures within your organization could also safeguard your business from unnecessary system failures and data leaks.

AWS provides various management and governance services to provision, manage, govern, and operate their cloud environments more effectively. You can use AWS Organizations, AWS Config, AWS Service Catalog, AWS Systems Manager, and other services to enforce operational standards across all your AWS resources and comply with your corporate IT policies.

The **AWS Organizations** service enables you to govern your AWS accounts and resources centrally. It provides consolidated billing, access control, compliance, and security, as well as the ability to share resources across your AWS accounts. You can use Service Control Policies (SCPs) to ensure that only the authorized users can execute actions that meet your policy requirements. Central logging can be implemented to monitor all activities performed across your organization using AWS CloudTrail. You can also aggregate data from all your AWS Config rules to quickly audit your environment for compliance.



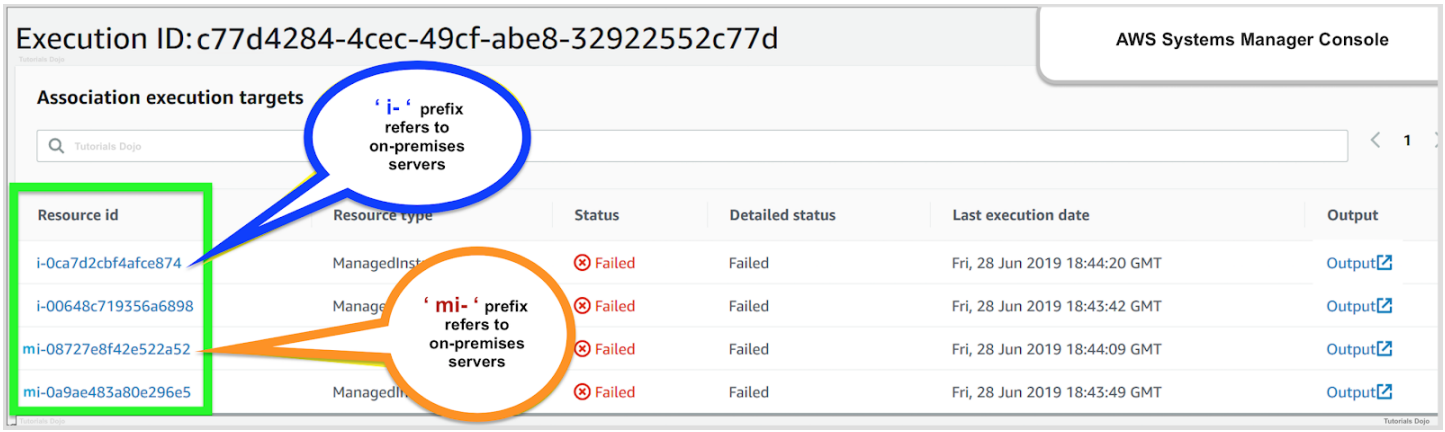
AWS Service Catalog empowers you to set up and centrally manage catalogs of approved IT services that you specify on AWS. You can manage various IT services, referred to as "products" in Service Catalog then group them in a portfolio. In AWS Service Catalog, a product could be a machine image, application server, program, tool, database, or other services that you use for your cloud architecture. AWS Service Catalog assists you in meeting your compliance requirements and enforce granular access control that allows the deployment of only approved IT services to your AWS Cloud.

AWS Config automates the compliance assessment of your internal policies and regulatory standards by giving you visibility on the existing configurations of your various AWS and third-party resources. It continuously assesses changes in your resource configuration and compares them against your specified criteria. You can create rules that detect an EC2 instance running on an unapproved AMI, publicly accessible S3 buckets, and many more. The evaluation can either be triggered periodically or by an actual configuration change of your AWS resource (e.g., CloudTrail was disabled in one of your accounts).

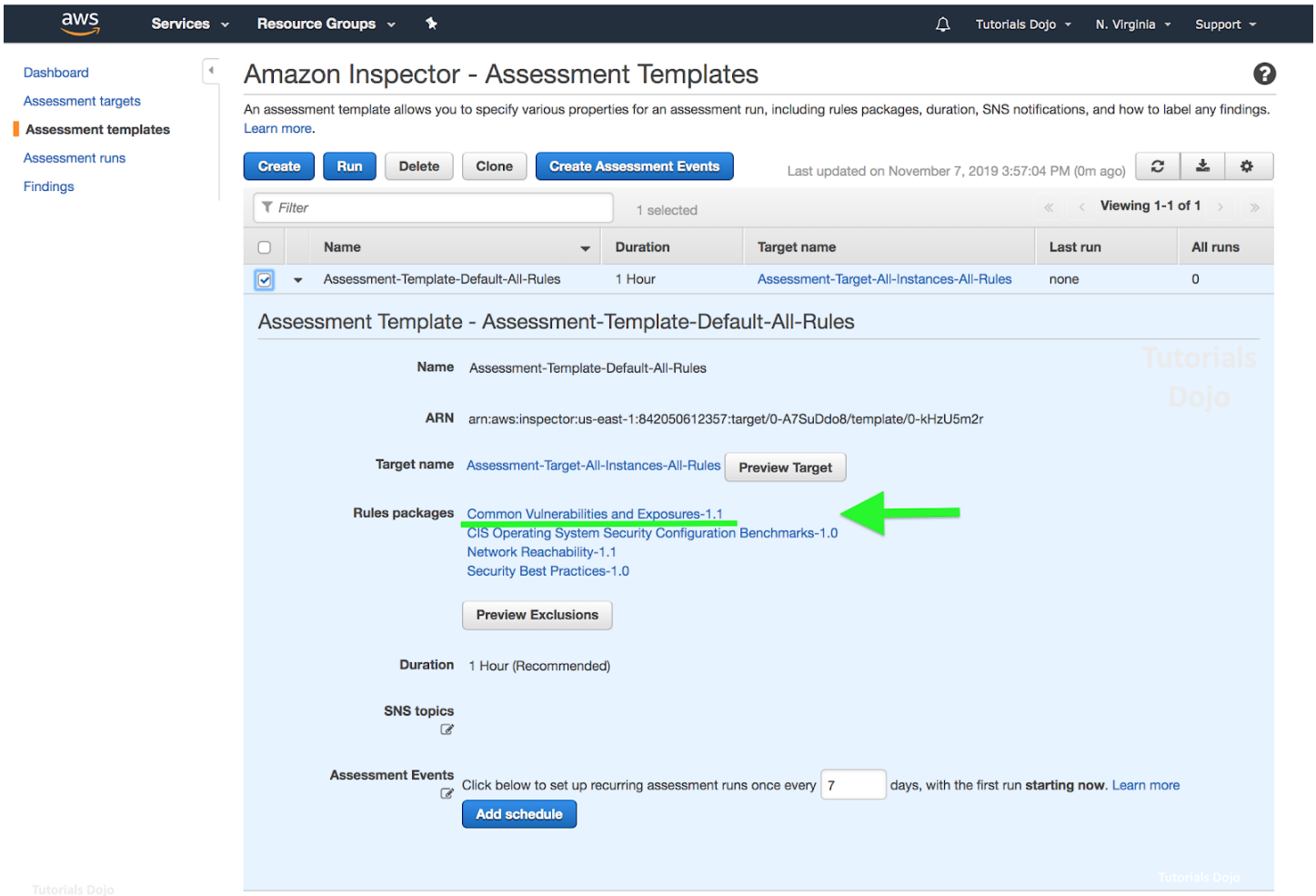
You can integrate AWS Config with CloudWatch Events and AWS Lambda to keep you updated for any resource changes in near real-time and to execute custom actions. Remediating noncompliant AWS resources can be automated by just creating AWS Config Rules and AWS Systems Manager Automation documents.

AWS Systems Manager (SSM) is a suite of services that provides you visibility and control of your cloud and on-premises infrastructure. SSM has several features that you can leverage on such as the Session Manager, State Manager, Patch Manager, Automation, Maintenance Window, Run Command, Parameter Store, and many other sub-modules.

Through the **SSM agent**, the AWS Systems Manager service can manage both Amazon EC2 instances and on-premises servers in which the former is prefixed with an *"i-"* and the latter with *"mi-"* in your AWS Management Console. The **Patch Manager** automates the process of patching the OS of your EC2 instances and on-premises servers using predefined and custom patch baselines. You can set a scheduled maintenance window to execute the patching activities to reduce any operational impact. With the **State Manager**, you can control the configuration details or the "state" of your resources, such as server configurations, virtualized hardware, and firewall settings. You can even associate Ansible playbooks, Chef recipes, PowerShell modules, and other SSM Documents to your resources. The **Systems Manager Parameter Store** provides centralized storage and management of your "parameters" such as passwords, database strings, Amazon Machine Image (AMI) IDs, license codes, environment variables, et cetera. Store the parameter as a *SecureString* datatype to instruct SSM to automatically encrypt it using a customer master key (CMK) in AWS KMS.



Amazon Inspector is an automated security assessment service that allows you to identify security issues and enforce standards to your AWS environment. You can install an Amazon Inspector agent using the Systems Manager Run Command and run security vulnerability assessments throughout your EC2 instances. Agentless network reachability assessments are also possible. You can run Common Vulnerabilities and Exposures (CVE), Center for Internet Security (CIS) Operating System configuration benchmarks, Network Reachability, and other assessments. It can also assess programs in your instances installed via apt, yum, or Microsoft Installer.



Amazon GuardDuty fortifies the security of your AWS architecture with intelligent threat detection and continuous monitoring across your AWS accounts. It aggregates and analyzes the data collected from your AWS CloudTrail, VPC Flow Logs, and DNS logs to detect various threats such as intra-VPC port scanning, cryptocurrency mining, malware, backdoor command, control (C&C) activities, and many other vulnerabilities. You can consolidate your security finding by setting up a master account for GuardDuty and associating other AWS accounts as member accounts. This integration can be done via AWS Organizations or by manually sending an invitation to the target member account.

Amazon Macie helps your organizations comply with the Health Insurance Portability and Accountability Act (HIPAA) and General Data Privacy Regulation (GDPR) regulations by detecting personally identifiable information (PII) in your Amazon S3 buckets. It also comes with native multi-account support to efficiently confirm your data security posture across your multiple AWS accounts from a single Macie administrator account.

AWS Trusted Advisor provides a set of best practice checks and recommendations for your AWS infrastructure covering cost optimization, security, fault tolerance, performance, and service limits. You can also fetch the various Trusted Advisor checks programmatically via web service using the AWS Support API. You can even use a CloudWatch Events rule to monitor your Trusted Advisor checks or create a CloudWatch Alarm to notify you of any status changes of your resources. These integrations to other AWS services make it easy to track underutilized Amazon EC2 instances in your account or detect any exposed IAM access keys on public code repositories such as GitHub.

Sources:
https://aws.amazon.com/blogs/mt/monitor-changes-and-auto-enable-logging-in-aws-cloudtrail/
https://docs.aws.amazon.com/config/latest/developerguide/remediation.html
https://docs.aws.amazon.com/awssupport/latest/user/trustedadvisor.html

AWS CodeBuild Configuration Best Practices

Configuration files are essential in your Continuous Integration / Continuous Delivery (CI/CD) pipeline. It contains vital information on how to build your application and the required permissions, password or other credentials to enable it to connect to its external dependencies such as a database or other AWS services. The sensitive information in these configuration files could pose a security risk. Part of configuration management is to ensure that these files comply with your organization's security standards and policies.

A build specification file (buildspec.yaml) contains the commands and related settings to instruct AWS CodeBuild on how to build your source code. Adding IAM access keys and database passwords in plaintext in your specification file is discouraged as these could be easily seen by unauthorized personnel. A better approach is to create an IAM Role and attach a permissions policy that grants permissions on your resources. You can store passwords and other sensitive credentials in AWS Systems Manager Parameter Store or AWS Secrets Manager. You can also leverage on using AWS Systems Manager Run Command instead of using scp and ssh commands that could potentially expose the SSH keys, IP addresses and root access of your production servers.

Below is an example of a poorly designed `buildspec.yaml` file. What do you notice?

```
version: 0.2
run-as: tutorialsdojo-cebu-admin
env:
  variables:
    AWS_DEFAULT_REGION: "ap-southeast-2"
    AWS_ACCESS_KEY_ID: "AKIAIOSFOD4NTUTSDOJO"
    AWS_SECRET_ACCESS_KEY: "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"
    DATABASE_PASSWORD: "tut0rialsd0j0DBpass"
  git-credential-helper: yes

proxy:
    upload-artifacts: yes
    logs: yes

phases:
  build:
    run-as: tutorialsdojo-cebu-admin
    commands:
      - echo Starting Installation...
      - cd /tmp
      - aws s3 cp s3://tutorialsdojo-db/my.cnf.template my.cnf
      - sed -i '' s/MYSQL_PW/${DATABASE_PASSWORD}/ my.cnf
      - aws s3 cp s3://tutorialsdojo-db/ec2-instance.key ec2-instance.key
      - chmod 600 ec2-instance.key
      - scp -i ec2-instance.key my.cnf ubuntu@10.10.5.243:/etc/my.cnf
      - ssh -i ec2-instance.key ubuntu@10.10.5.243 /etc/init.d/mysqld restart
    finally:
      - echo Installation Complete!

```


Tutorials Dojo

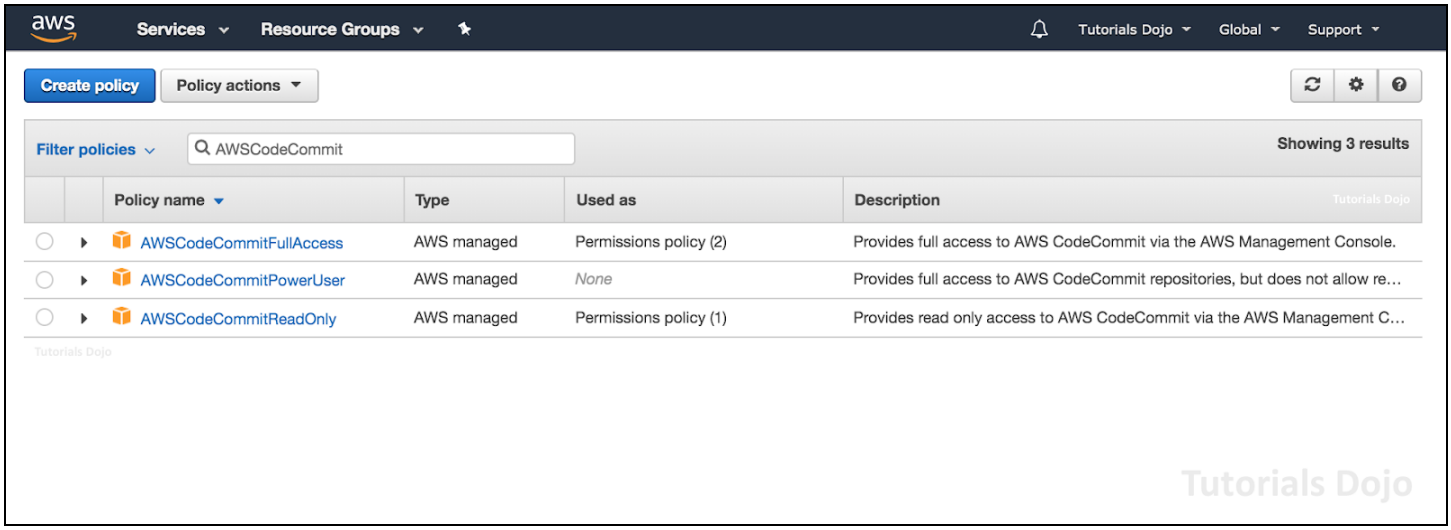
The above configuration file has the following security issues:

- The AWS access and secret keys are stored in the `buildpsec.yaml` file.
- The database credential (`DATABASE_PASSWORD`) is stored as an environment variable in plaintext.
- Contains embedded `scp` and `ssh` commands that expose the SSH keys and server IP addresses.

Source:
https://docs.aws.amazon.com/codebuild/latest/userguide/data-protection.html

AWS CodeCommit Managed Policies with Customized Permissions

AWS provides several AWS CodeCommit managed policies that you can use to provision access to the source code repositories. You can attach AWSCodeCommitFullAccess to your administrators, AWSCodeCommitPowerUser for managers or developers, and AWSCodeCommitReadOnly to auditors or external parties.



The AWSCodeCommitPowerUser policy grants users access to all of the functionality of CodeCommit and repository-related resources. However, it does not allow them to delete CodeCommit repositories or create or delete repository-related resources in other AWS services. Developers who have this policy can directly push their code to the master branch on all CodeCommit repositories without raising a proper pull-request. It falls short on the principle of granting the least privilege as developers could circumvent the standard Git workflow or your standard development process.

Remember that you can't modify these AWS-managed policies. However, you can add a *Deny* rule to an IAM Role to block specific capabilities included in these policies and to further customize the permissions. Notice the following IAM Policy:

The CodeCommitPowerUser managed policy can be affected by another policy that denies several actions, such as pushing code to the master branch. This policy will prevent the developer from pushing, deleting, and merging code to the master branch of the TutorialsDojoManila CodeCommit repository as shown above.

Source:
https://docs.aws.amazon.com/codecommit/latest/userguide/auth-and-access-control-iam-identity-based-access-control.html

S3 Bucket Policy to Only Allow HTTPS Requests

By default, Amazon S3 allows both HTTP and HTTPS requests. As part of security compliance, you may be required to ensure that all data in transit to and from your S3 buckets are encrypted using the HTTPS protocol.

If you have a lot of S3 buckets, you can enable the AWS Config rule "**s3-bucket-ssl-requests-only**", which checks whether S3 buckets have policies that require requests to use Secure Socket Layer (SSL).

DevOps Exam Notes:

To be compliant on the AWS Config rule, your S3 bucket will need to have the proper bucket policy to explicitly deny HTTP requests and only allow HTTPS requests. To determine HTTP or HTTPS requests in a bucket policy, use a condition that checks for the key "aws:SecureTransport". You can set the action to "Deny" any requests if the condition "**aws:SecureTransport**" is **false**.

Here's an example S3 bucket policy to deny HTTP requests on the bucket, thus forcing all connections to be HTTPS-only.:

```
        "Id": "ExamplePolicy",
        "Version": "2012-10-17",
        "Statement": [
            {
            "Sid": "AllowSSLRequestsOnly",
            "Action": "s3:*",
            "Effect": "Deny",
            "Resource": [
              "arn:aws:s3:::exampletutorialsdojobucket",
              "arn:aws:s3:::exampletutorialsdojobucket/*"
            ],
            "Condition": {
               "Bool": {
                "aws:SecureTransport": "false"
            }
            },
            "Principal": "*"
            }
        ]
    }
```

Sources:
https://aws.amazon.com/premiumsupport/knowledge-center/s3-bucket-policy-for-config-rule/
https://docs.aws.amazon.com/config/latest/developerguide/s3-bucket-ssl-requests-only.html

Secrets Manager vs. Systems Manager Parameter Store

AWS Secrets Manager and Systems Manger Parameter Store offer similar functionalities that allow you to centrally manage and secure your secret information that can then be retrieved by your applications and resources in AWS.

Both services offer similar web interfaces on which you can declare key-value pairs for your parameters and secrets. So it is important to know the similarities and differences they have in order to choose the right service for a given situation in the exam.

Here's a summary of features of SSM Parameter Store and AWS Secrets Manager:

	SSM Parameter Store	AWS Secrets Manager
Store values up to 4096 Characters	Yes	Yes
Values can be encrypted with KMS	Yes	Yes
Can be referenced in CloudFormation	Yes	Yes
Built-in password generator		Yes
Automated secret rotation		Yes
Cross-account access		Yes
Additional Cost	Free	Yes

DevOps Exam Notes:

In the exam, the usual situation is when you need to store a database password, you should choose SSM Parameter Store if you don't have to automatically rotate the secret regularly, plus it doesn't cost anything.

As an additional note, Parameter Store is now integrated with Secrets Manager so that you can retrieve Secrets Manager secrets when using other AWS services that already support references to Parameter Store parameters. This is helpful if your application is configured to use Parameter Store APIs, but you want your secrets to be stored in Secrets Manager.

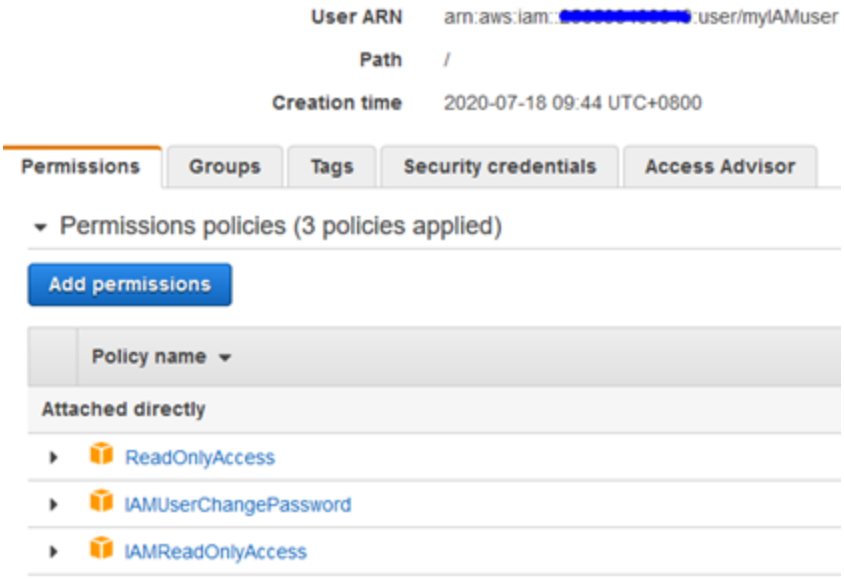
Sources:
https://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-parameter-store.html
https://aws.amazon.com/about-aws/whats-new/2018/07/aws-systems-manager-parameter-store-integrates-with-aws-secrets-manager-and-adds-parameter-version-labeling/

AWS Managed Policy

AWS provides you with three types of policies that you can attach when managing permissions for your IAM Users, Groups or Roles – **AWS managed policy**, **customer managed policy** or an **inline policy**.

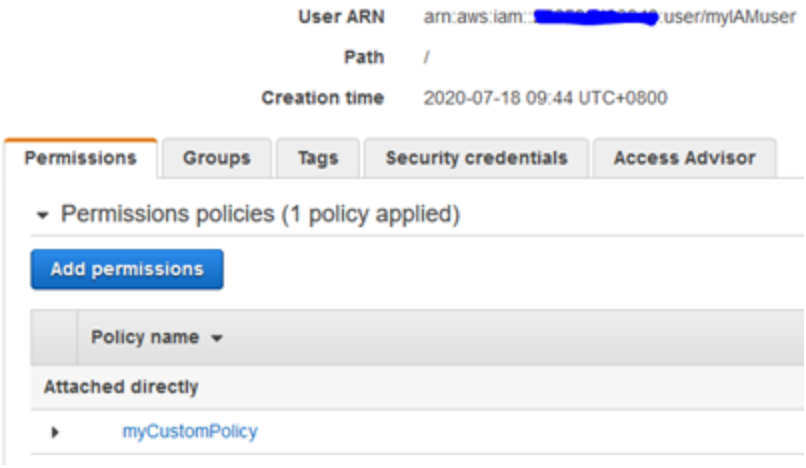
An **AWS managed policy** is a standalone policy that is created and administered by AWS which can be used to provide permissions for many common use cases or specific job functions. You cannot change the permissions defined in AWS managed policies. AWS occasionally updates the permissions defined in an AWS managed policy. For example, arn:aws:iam::aws:policy/IAMReadOnlyAccess is an AWS managed policy as well as, AmazonDynamoDBFullAccess, IAMFullAccess, or AmazonEC2ReadOnlyAccess.

Example of attached AWS managed policies:

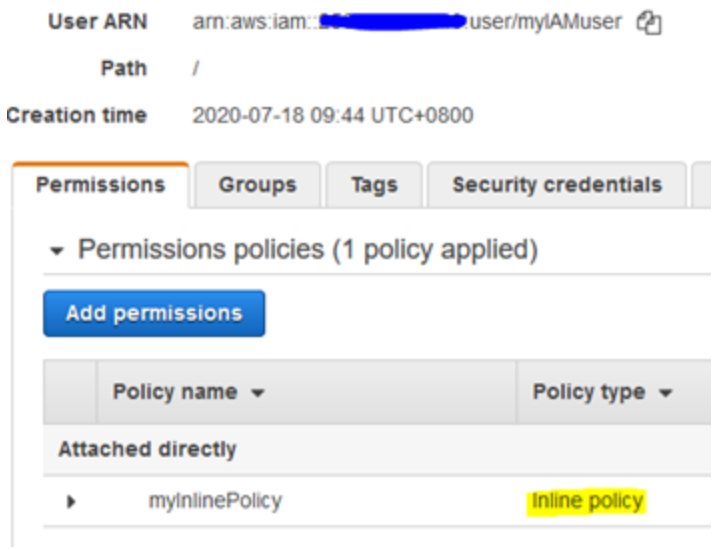


Customer managed policies are standalone policies that you manage on your own AWS Account. You can attach these policies to users, groups, or roles the same way as managed policies. You can copy an AWS managed policy and modify its contents to apply it as a customer managed policy. This gives you much better control of the permissions you grant to your IAM entities.

Example of customer managed policy:



An **inline policy** is a policy that's embedded in an IAM identity (a user, group, or role). When you create an identity in IAM, you can directly attach an inline policy. An inline policy is a strict one-to-one relationship between a policy and the identity that it's applied to. It can't be re-used to be attached to other IAM identities. For example, you want to be sure that the permissions in a policy are not inadvertently assigned to an identity other than the one they're intended for.



Managed Policy vs Inline Policy

Managed Policies are more flexible than Inline policies. Managed policies are recommended for the following reasons:

- **Reusability** - Can be reused for by attaching to other identities
- **Central change management –** one policy change can be applied to all identities.

- **Versioning and rollback** – can create multiple versions and rollback changes if needed.
- **Delegating permissions management –** users can attach/detach policies on their own, while you control the permissions on those policies.
- **Automatic updates for AWS managed policies** – AWS updates managed policies when necessary

Source:
https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_managed-vs-inline.html

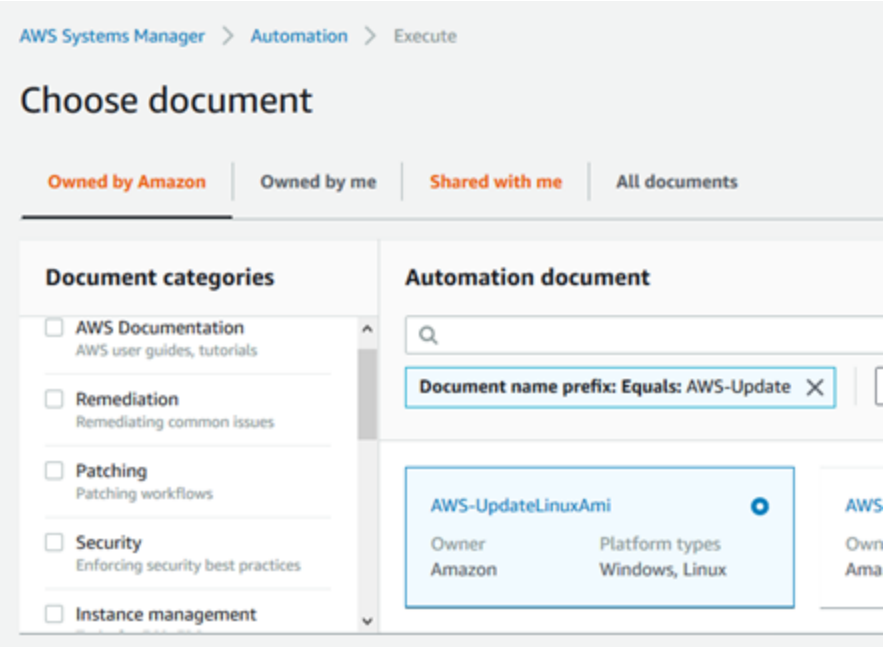
Using Systems Manager Automation to create AMIs

Systems Manager Automation allows you to perform maintenance and deployment tasks for EC2 instances. There are several predefined AWS Automation documents that you provide for common use cases but you can also upload your own Automation documents or share documents to other AWS accounts. Some use cases for SSM Automation are the following:

- Build Automation workflows to configure and manage instances and AWS resources.
- Create custom workflows or use pre-defined workflows maintained by AWS.
- Receive notifications about Automation tasks and workflows by using Amazon CloudWatch Events.
- Monitor Automation progress and execution details by using the Amazon EC2 or the AWS Systems Manager console.

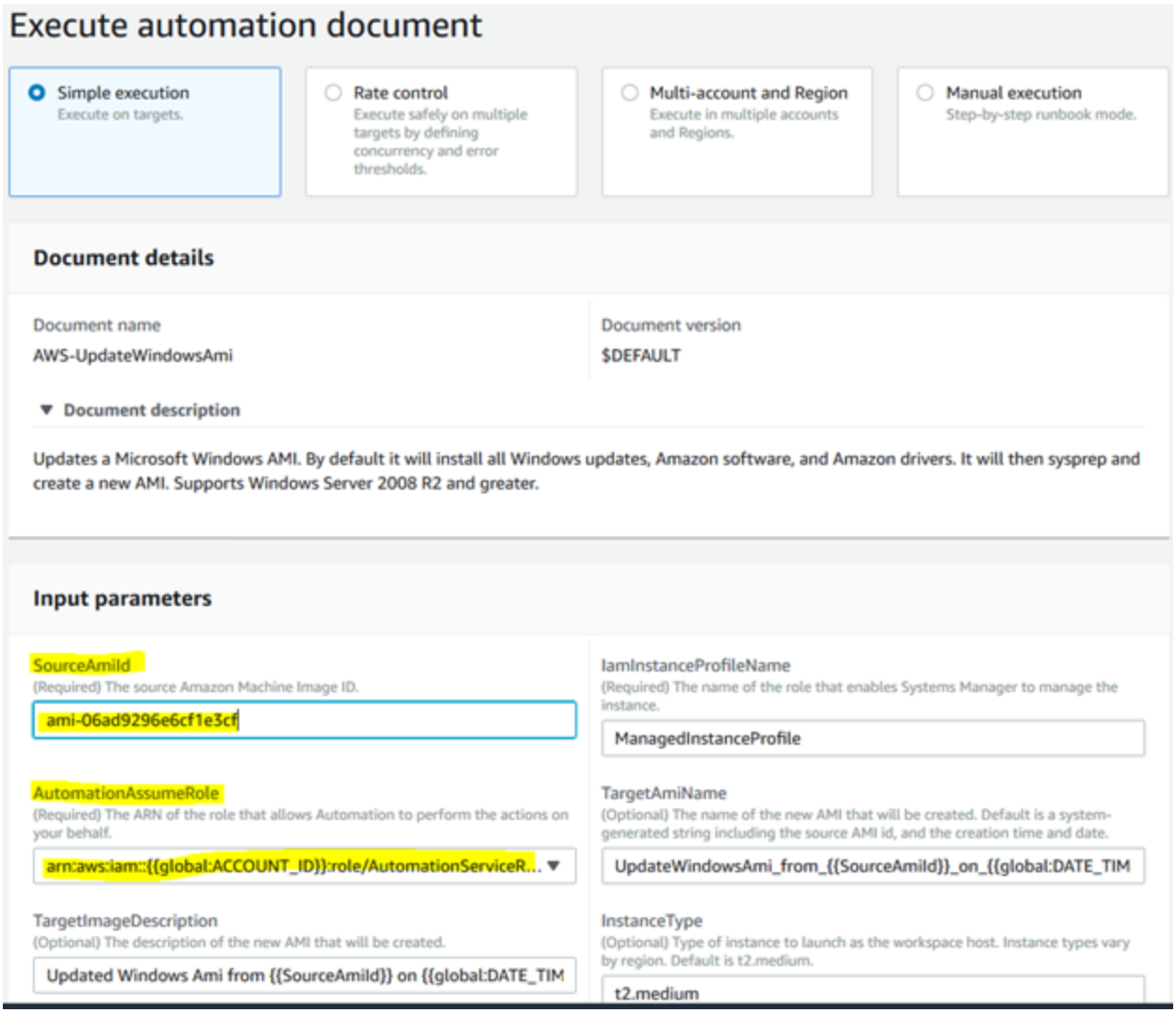
Among the automation workflows is the ability to create updated AMIs. This is helpful if for example you want to apply the latest system patches to your EC2 instances and create an updated AMI so that all new instances that you will create will have the latest patch applied.

On AWS Systems Manager Automation, select “Execute”, and you can choose the Automation Document **AWS-UpdateLinuxAMI**.



The next page will present you with an option to fill out the input parameters for this document. The important part here is the **SourceAmiId**. This value should be your current AMI ID that you want to use as a base and

where the patches will be applied. It's also important to have an **AutomationAssumeRole** present in order to allow SSM Automation to perform the needed actions.



Upon clicking the Execute button, SSM Automation will take the following actions based on the automation document:

- Take the AMI and create a new EC2 instance with it
- Execute the document runbook that apply patches to the OS
- Shutdown the instance
- Create an AMI of the EC2 instance
- Terminate the EC2 instance
- Output the new AMI ID

The output AMI ID can now be stored on SSM Parameter Store or be used by a Launch Configuration. You can also register this Automation task on the SSM Maintenance Windows console page to allow you to set a schedule for this automation to run regularly.

Sources:
https://aws.amazon.com/premiumsupport/knowledge-center/ec2-systems-manager-ami-automation/
https://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-automation.html

AWS SSM Session Manager to Record Sessions on your Instances

AWS Systems Manager includes the Session Manager service which allows you to manage your EC2 instances, on-premises instances, and virtual machines (VMs) through an interactive one-click browser-based shell or through the AWS CLI. With Session Manager, you don't need to open inbound ports or use bastion hosts when you want to have Shell access or RDP access to your EC2 instances.

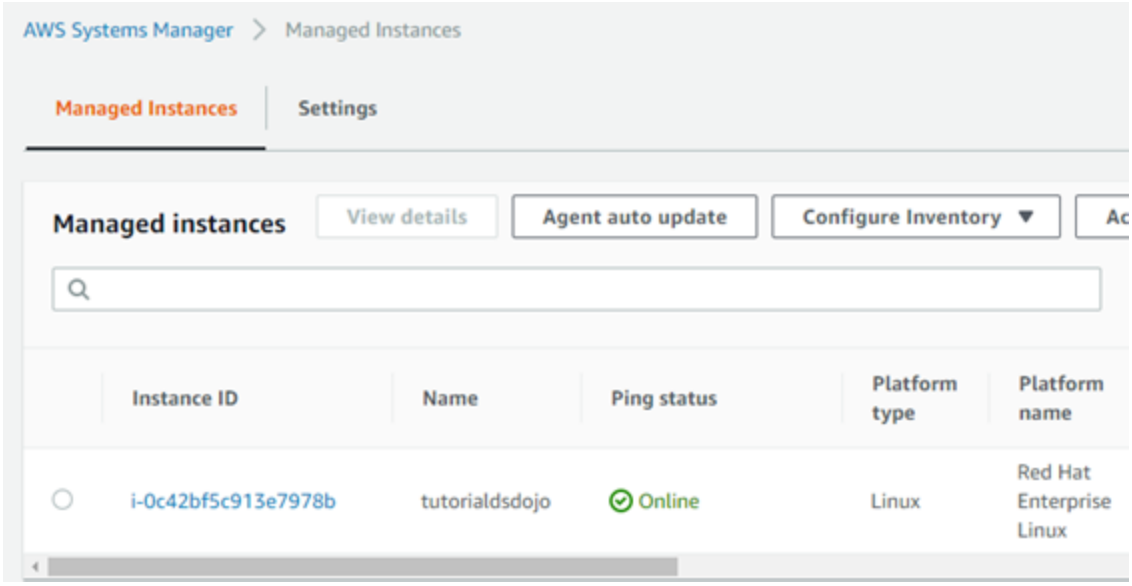
Session Manager also makes it easy to comply with corporate policies that require controlled access to instances, strict security practices, and fully auditable logs with instance access details. For example, you can view all sessions performed by users on the Session Manager History page. Or you can send the whole session log to an S3 bucket or CloudWatch Logs for recording purposes.

DevOps Exam Notes:

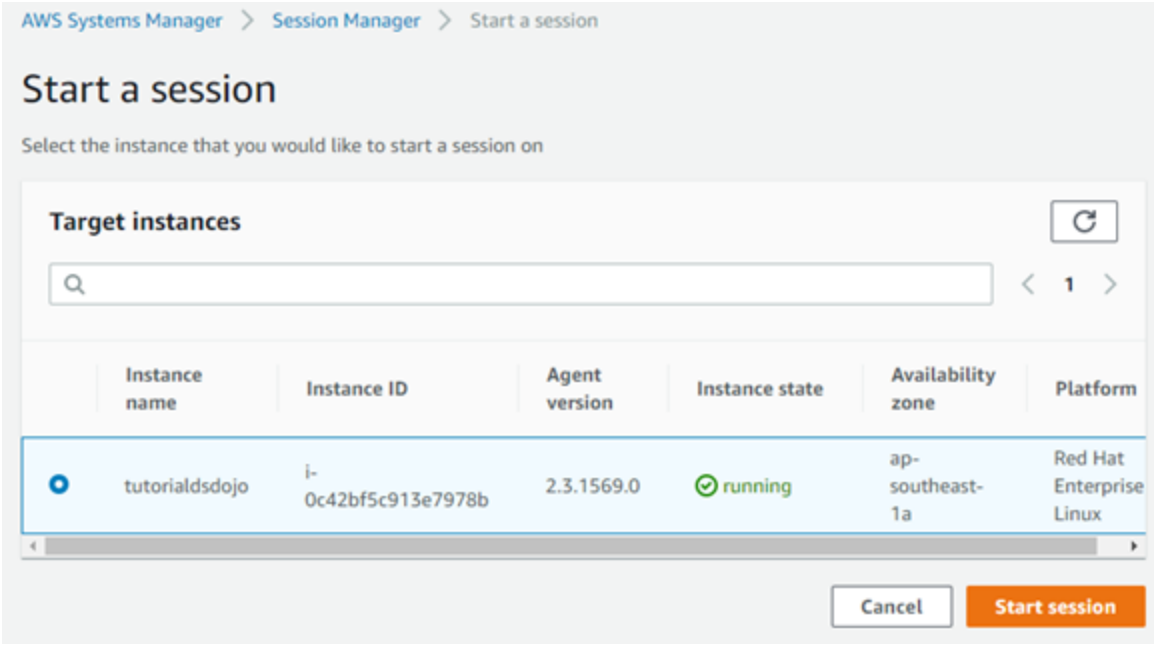
Remember that SSM Session manager provides a centralized location on where users can SSH to EC2 instances or on-premises service configured with the SSM agent. For example, you are required by the company to provide a solution for Developers to SSH on both on-premises instances and EC2 instances and save all sessions to an S3 bucket (or CloudWatch Logs) which will be available for the security team to perform an audit. Once you installed and configured the SSM agent on your instances, they will show up on the Session Manager page.

Here's how you can use Session Manager and send your session logs to an S3 bucket.

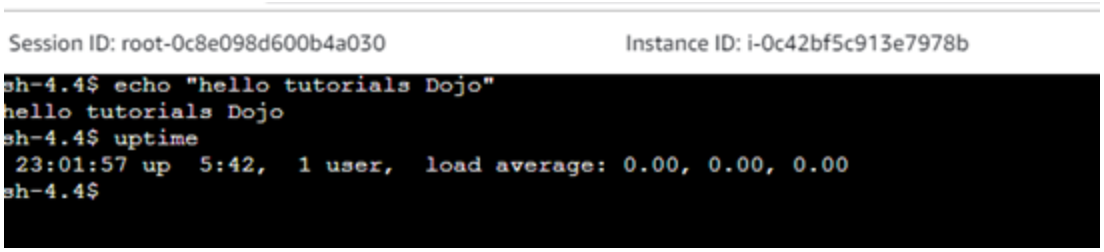
1. Be sure that your instance has the SSM agent installed and is registered on the Managed Instances section of Systems Manager.



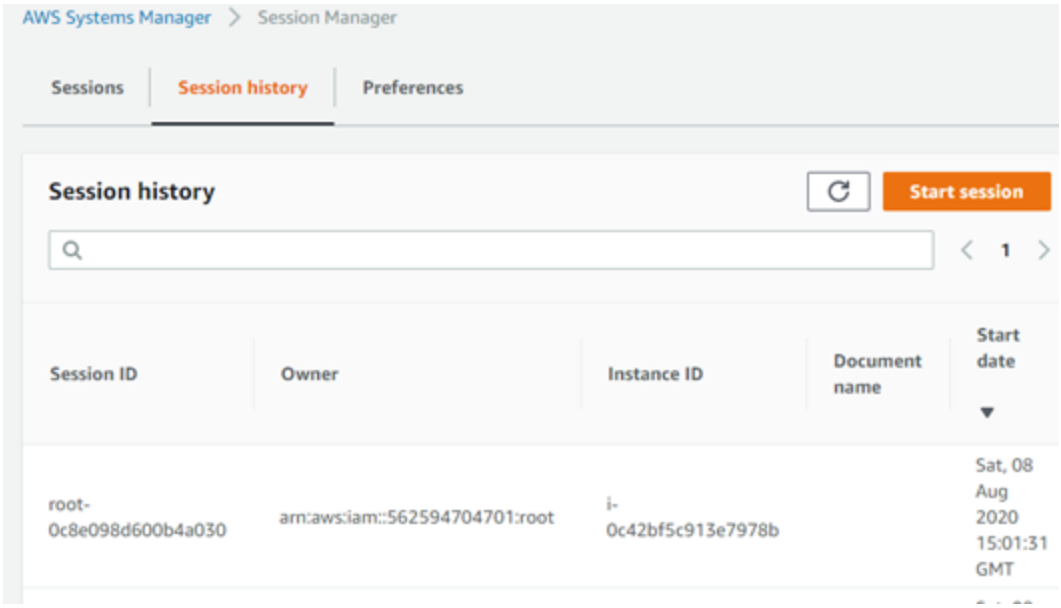
2. Go to Systems Manager > Session Manager and click the “Start session” button. Select your EC2 instance and click “Start Session”.



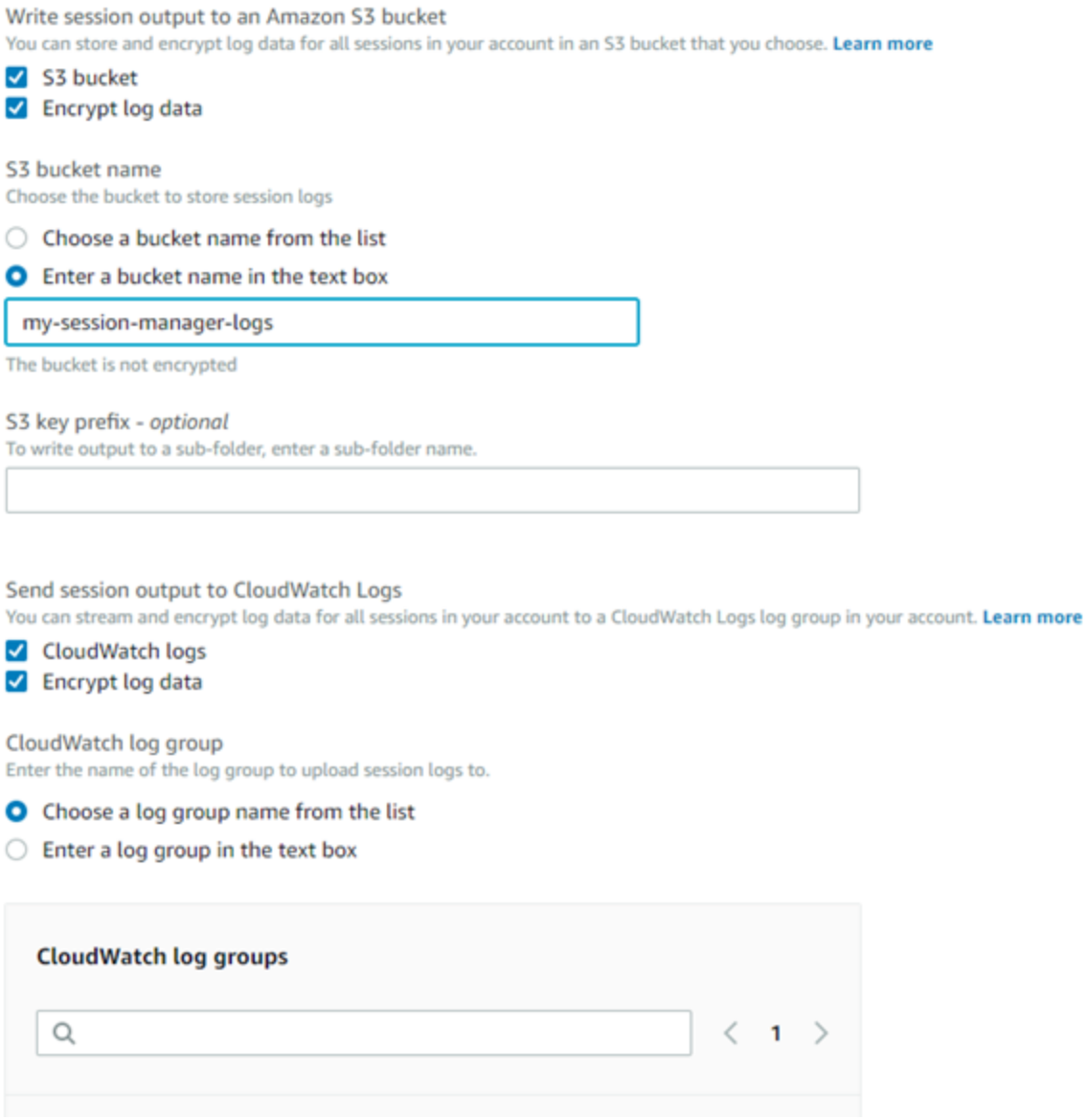
3. You will have a browser tab that provides a Shell access to your instance. You can click the “Terminate” button on the upper-right side once you are done with your session.



4. You can view this session on the Session Manager history page.



5. On the preferences page, you can configure Session Manager to send logs to an S3 bucket or CloudWatch Logs. Click the “Edit” button on the Preferences page.



Sources:
https://docs.aws.amazon.com/systems-manager/latest/userguide/session-manager.html
https://docs.aws.amazon.com/systems-manager/latest/userguide/session-manager-getting-started.html

AWS Systems Manager Inventory

AWS SSM Inventory provides visibility for your Amazon EC2 instances or on-premises servers. You can use SSM Inventory to collect metadata from your managed instances such as installed applications, OS versions, Windows Updates, Network configuration, or running services, etc. You can store this metadata in a central Amazon S3 bucket, and then use built-in tools to query the data and quickly determine which instances are running the software and configurations required by your software policy, and which instances need to be updated.

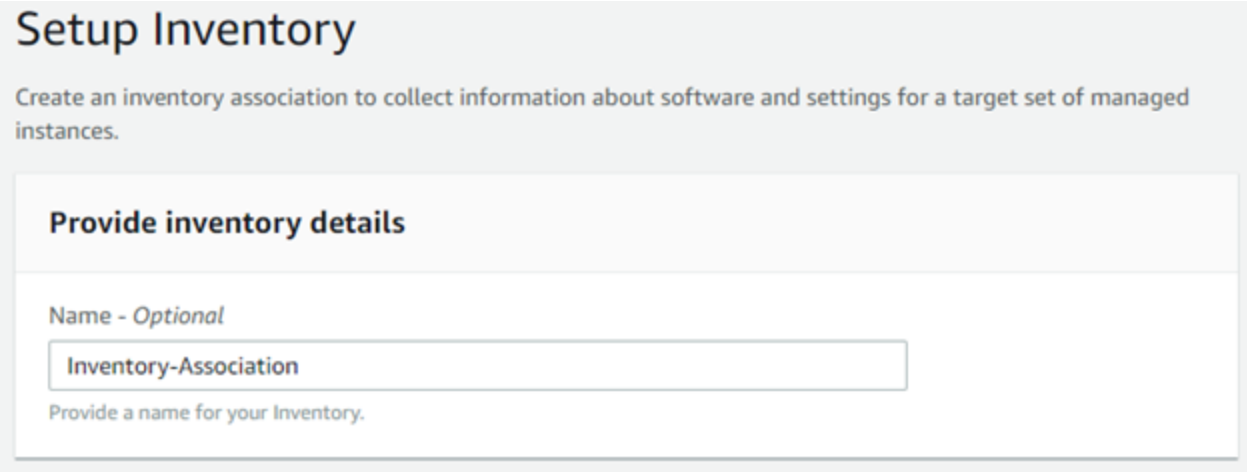
DevOps Exam Notes:

AWS SSM inventory is helpful if you want to make sure that your managed instances have the correct application versions and patches installed. You even filter which instances have outdated system files. On the exam, you need to remember that SSM Inventory is also helpful if you want to identify details of your on-premises instances that you want to migrate to AWS. It will help you view and gather details of those instances to sync the data collected on an S3 bucket. This helps ensure that your new instances on AWS will have similar configurations to those on your current on-premises environment.

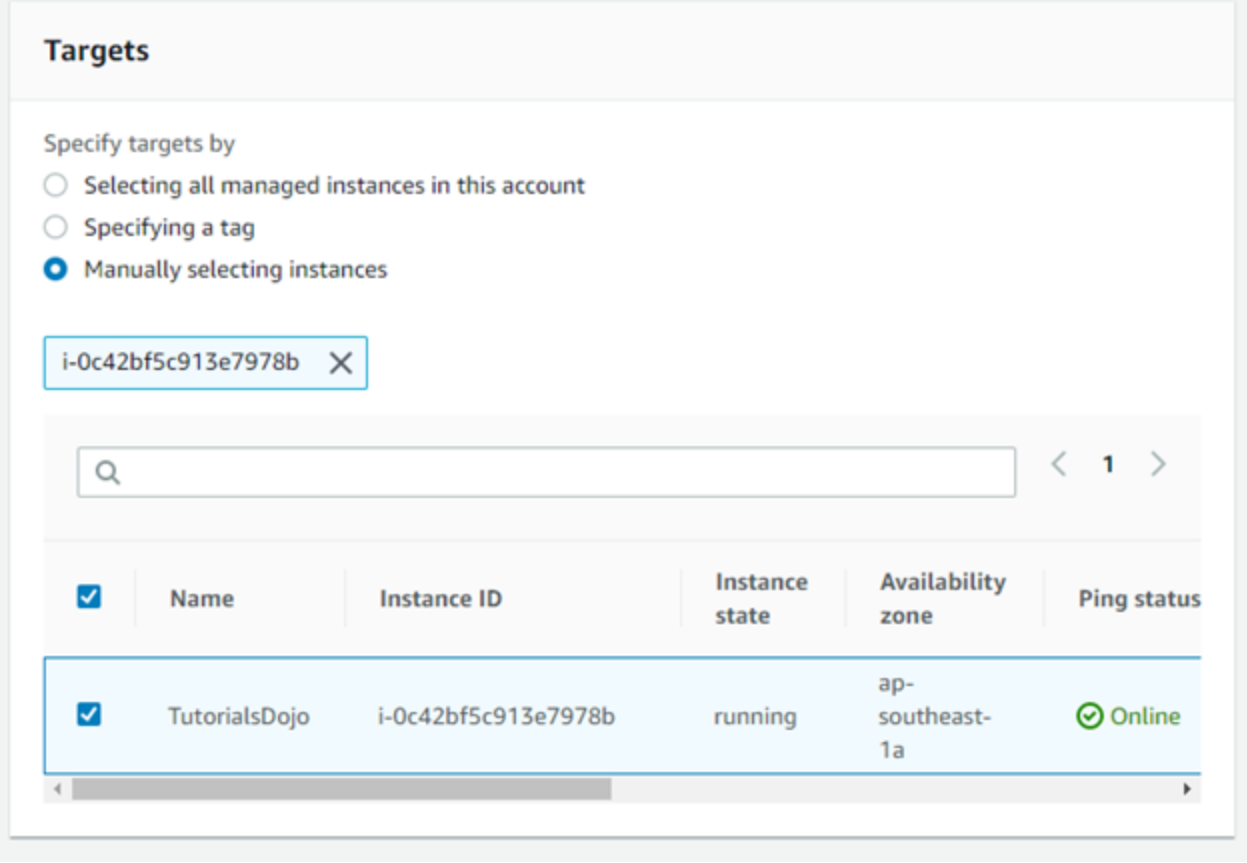
The following steps show how to set up Systems Manager Inventory:

1. Ensure that your instance is registered as a Managed Instance on Systems Manager.

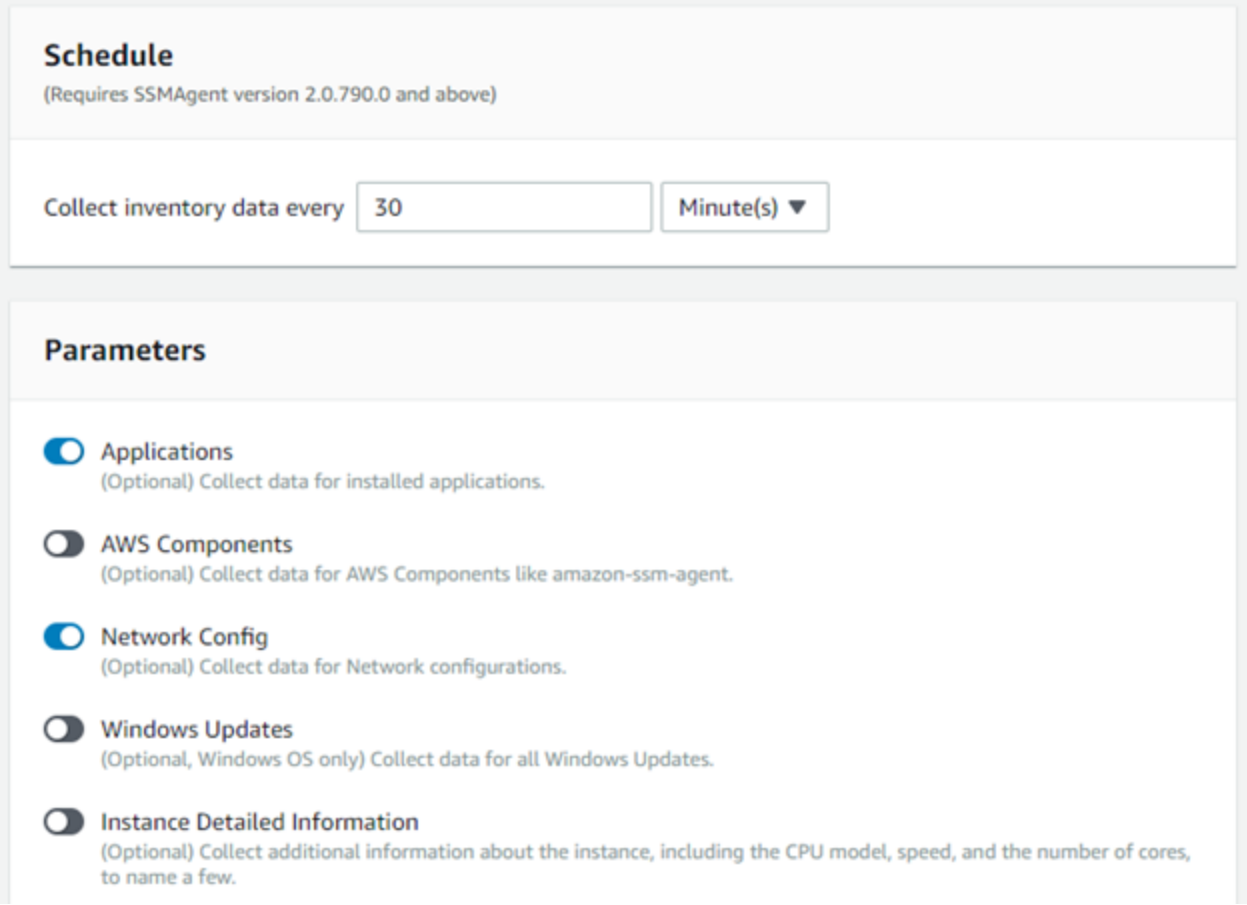
2. Go to Systems Manager > Inventory and click “Setup Inventory”. Set a name for this inventory association.



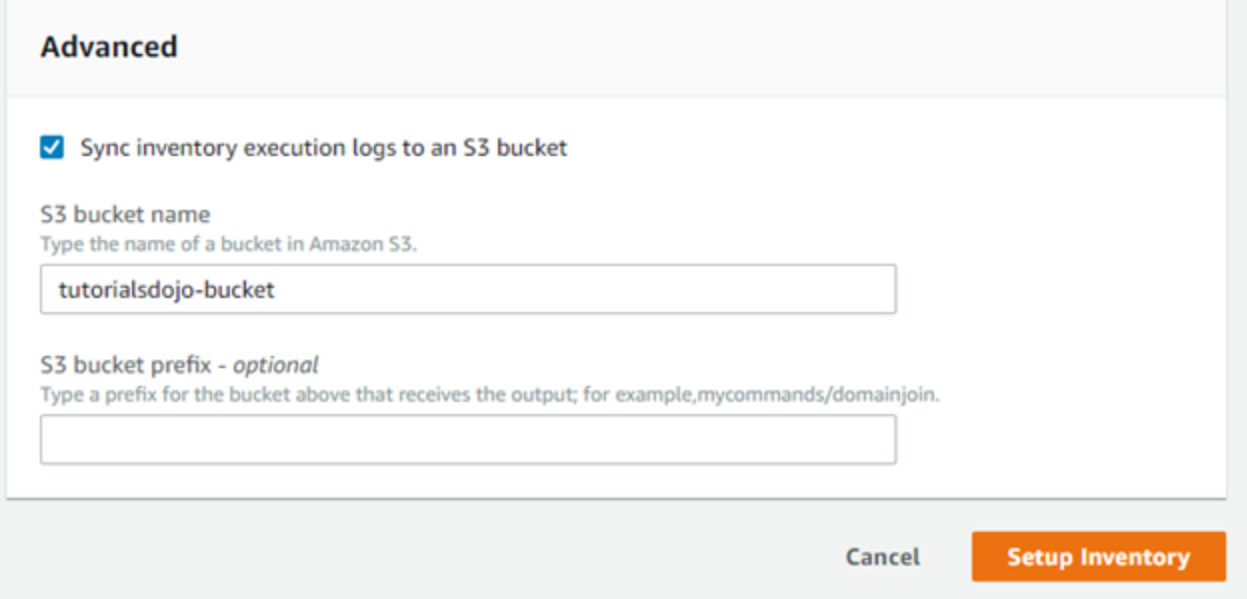
3. Select targets for your inventory. You can choose to select all instances, or use a specific Tag identifier, or manually select your desired instances.



4. You can set the schedule on how often the inventory is updated as well as the parameters it will collect on the instances.



5. You can also specify an S3 bucket on which the inventory will sync the data it collects.



6. After clicking the "Setup Inventory" button, the details of your instances will be shown on the SSM Inventory page.

Sources:
https://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-inventory.html
https://docs.aws.amazon.com/systems-manager/latest/userguide/sysman-inventory-configuring.html

Systems Manager Patch Manager and Maintenance Windows

AWS SSM Patch Manager automates the process of patching managed instances with both security related and other types of updates. Patch Manager supports several OS types such as Windows Server, Amazon Linux, Ubuntu, RHEL, Debian, Centos, SLES, etc. It allows you to select patch baselines which are a list of approved patches that will be applied to your instances. You can choose from AWS pre-defined patch baselines or you can create your own patch baseline if you have your own custom repository.

AWS SSM Maintenance Windows allows you to define schedules for when to perform potentially disruptive actions on your instances such as patching an operating system, or installing software updates. Each maintenance window has a schedule, a duration, a set of registered targets and a set of registered tasks. When you create a Patch Manager task, it will be shown on the Maintenance Windows Tasks page.

DevOps Exam Notes:

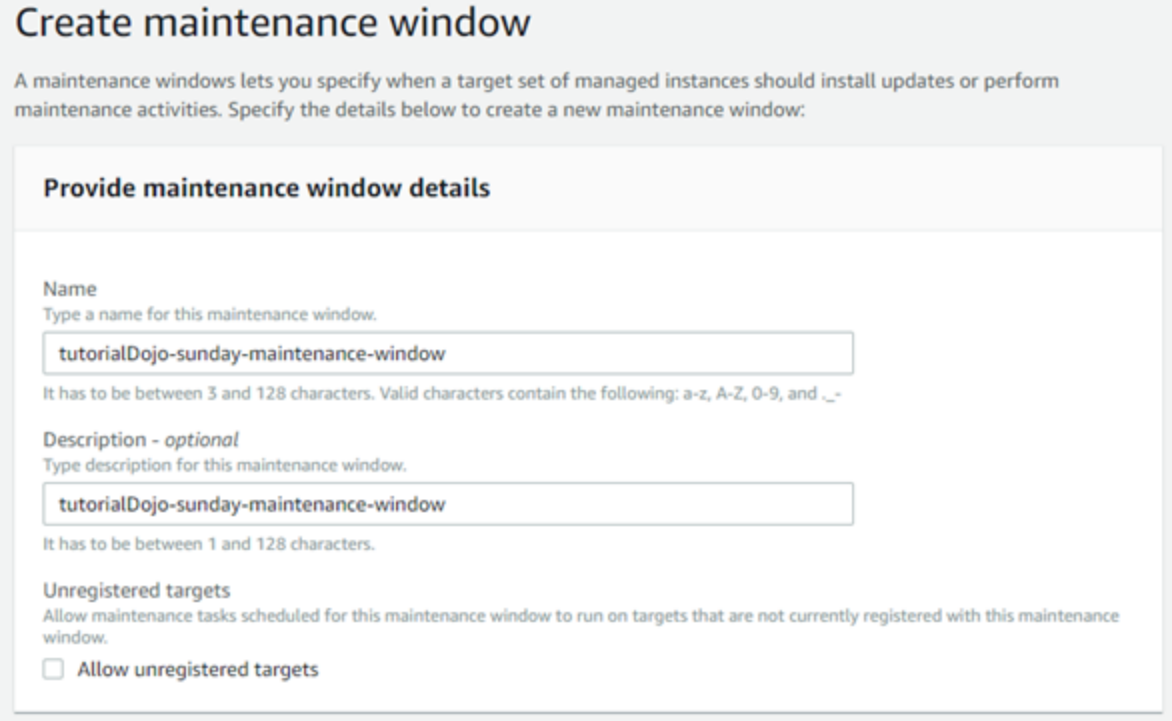
On the exam, you need to remember that Patch Manager allows you to define custom patch baselines. This is helpful if your company has a custom repository of software updates that needs to be installed on your managed instances. You can schedule the application of the patches by creating a Maintenance Window and assigning your Patching task to that window.

Below are the steps to set up Maintenance Windows and Patch Manager for your managed instances.

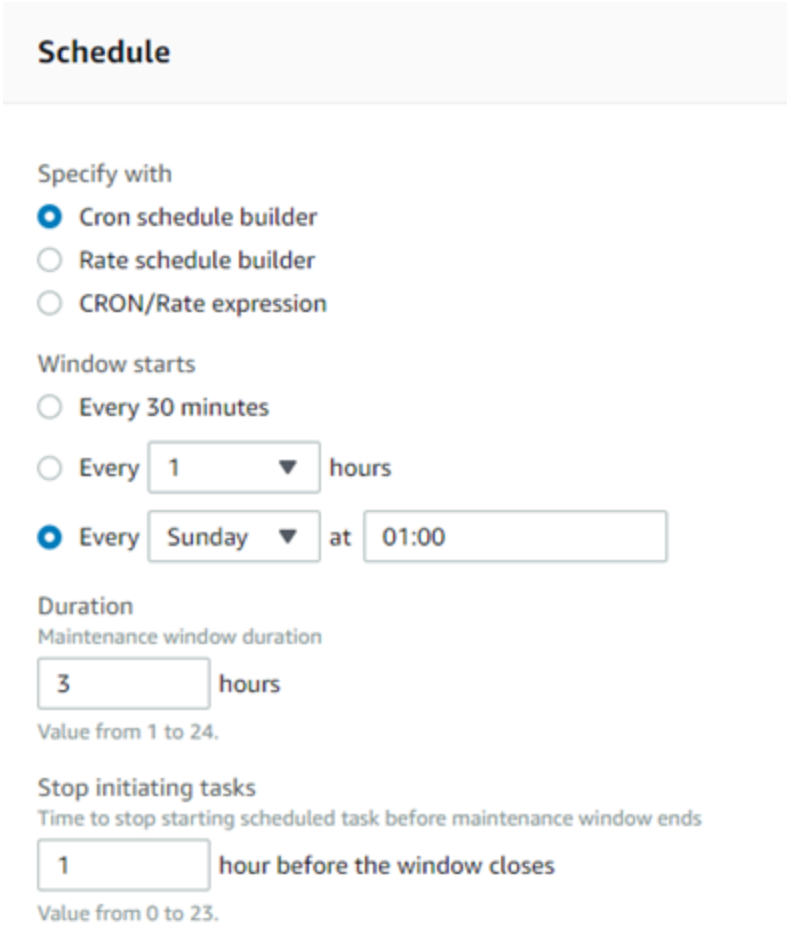
1. Ensure that your managed instances are registered on Systems Manager.

2. Go to Systems Manager > Maintenance Windows and click “Create Maintenance Window”.

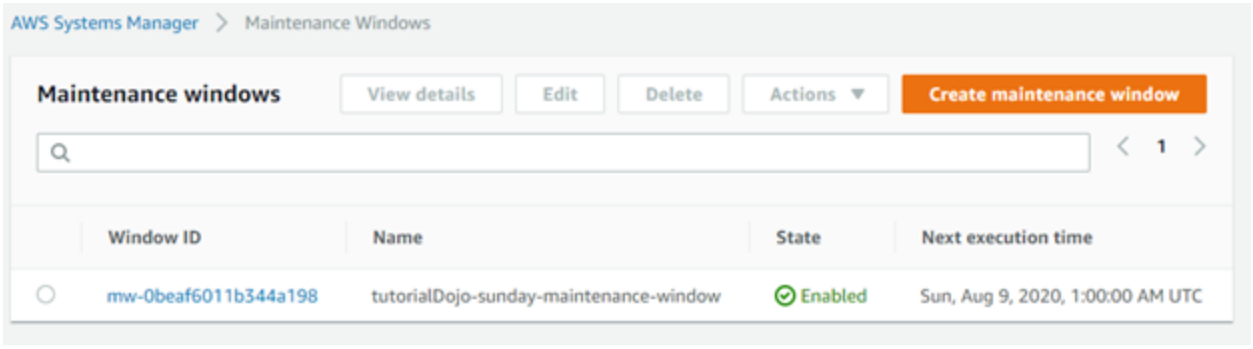
3. Provide a name and description for your Maintenance window.



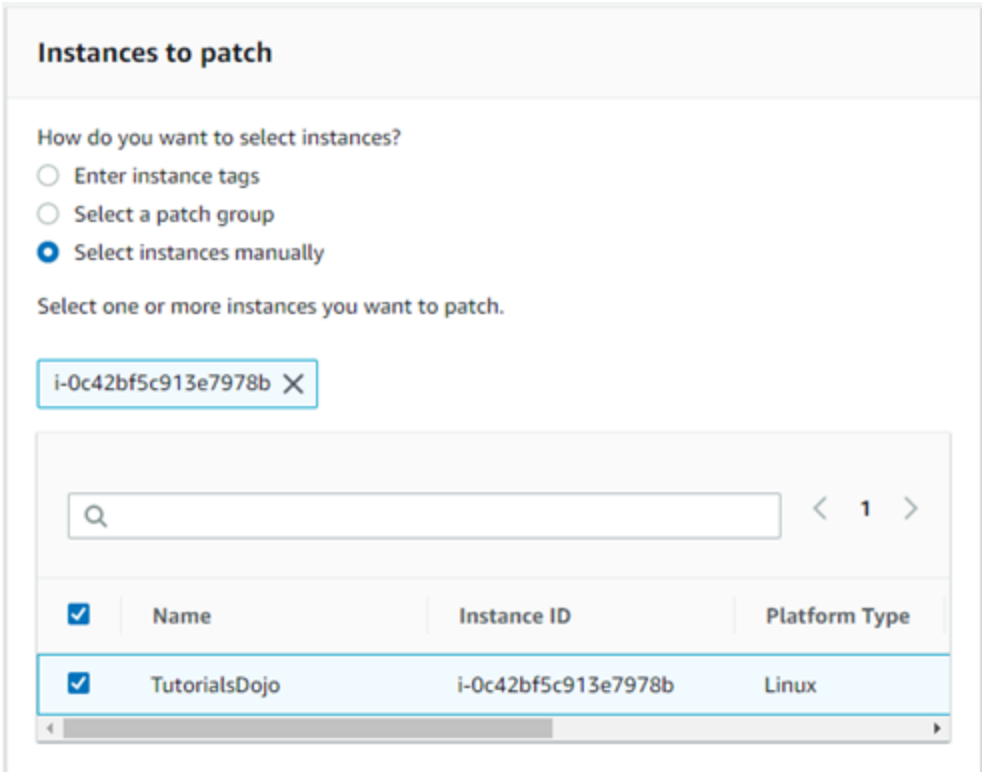
4. Provide a schedule for this maintenance. We'll set this to run every Sunday morning at 1:00 AM with a 3-hour window time. And one hour before the window closes, SSM will stop initiating tasks.



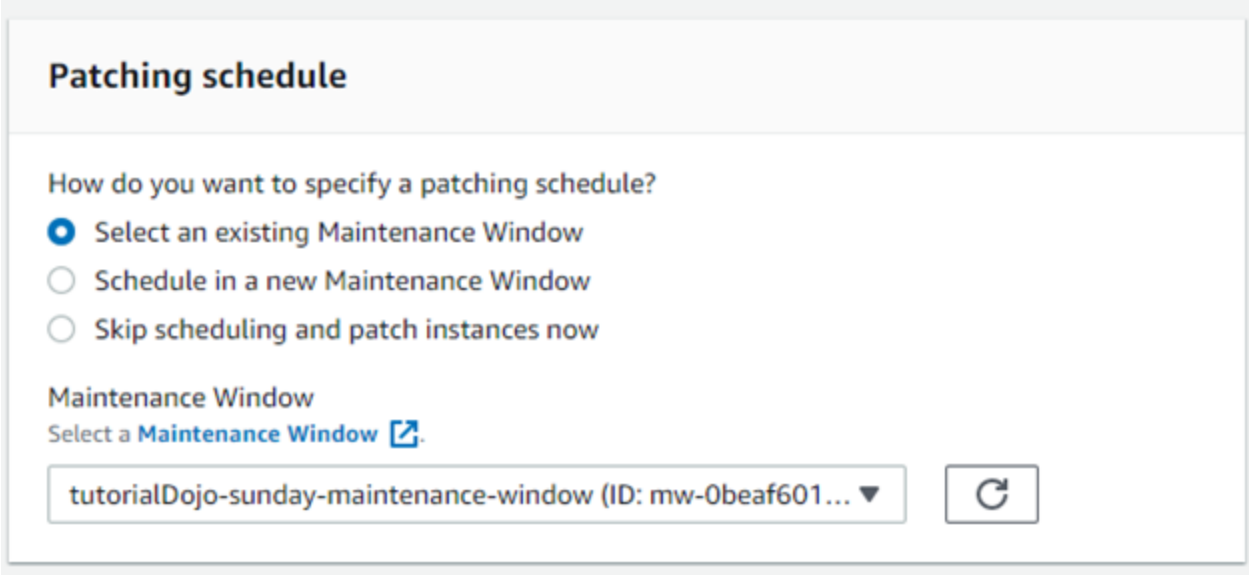
5. Click “Create maintenance window”. You will then see your created maintenance window.



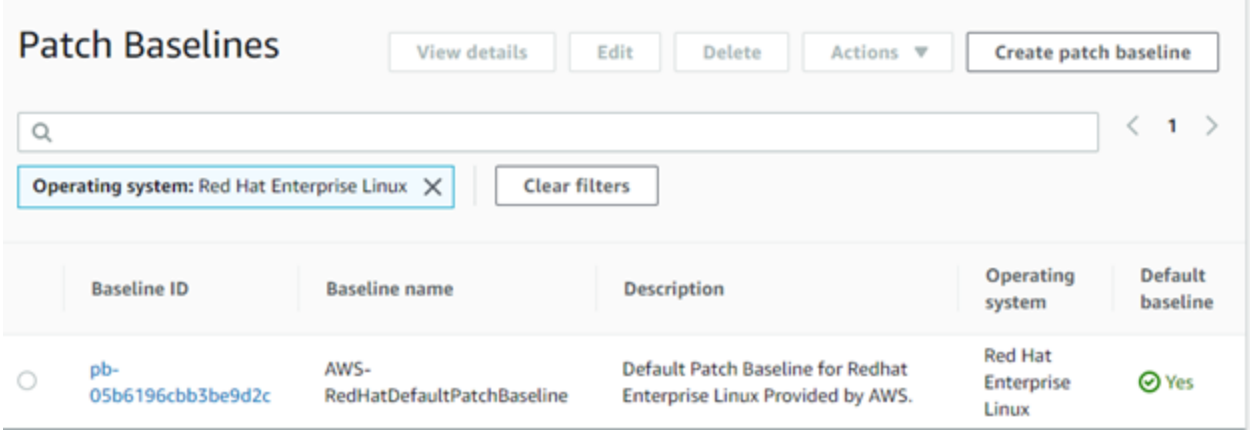
6. Now go to Patch Manager and click “Configure patching”. Select your instances to patch. You can select by instance tags, patch groups, or by selecting the instances manually.



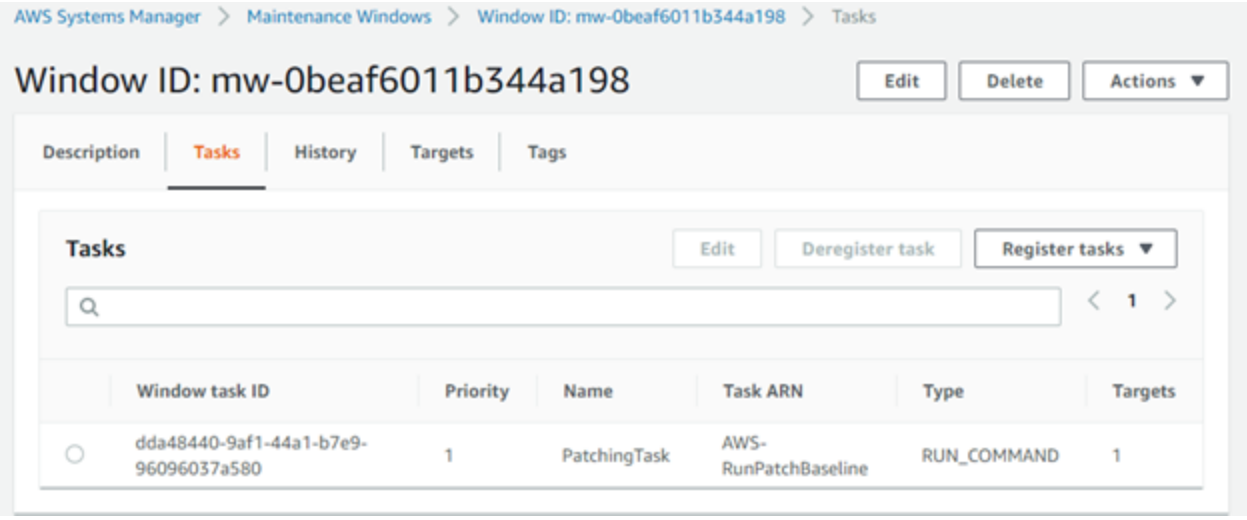
7. Select your Patching schedule then select the previously created maintenance window.



8. Click “Configure patching” to create this patching task. You should see the “AWS-RedHatDefaultPatchBaseline” for your instance as the default baseline.



9. Go to the Maintenance Windows page again and select your maintenance window. Click the Tasks section and you should see your configured Patching task.



10. SSM Maintenance Windows will now run this task on the specified window time.

Sources:
https://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-patch.html
https://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-maintenance.html

Domain 5: Incident and Event Response

Overview

The fifth exam domain of the AWS Certified DevOps Engineer Professional test focuses on incident and event management in the AWS infrastructure. It has the third biggest percentage representing approximately 18% of the overall test.

This domain will test you knowledge and skills on the following:

- Troubleshoot issues and determine how to restore operations
- Determine how to automate event management and alerting
- Apply concepts required to implement automated healing
- Apply concepts required to set up event-driven automated actions

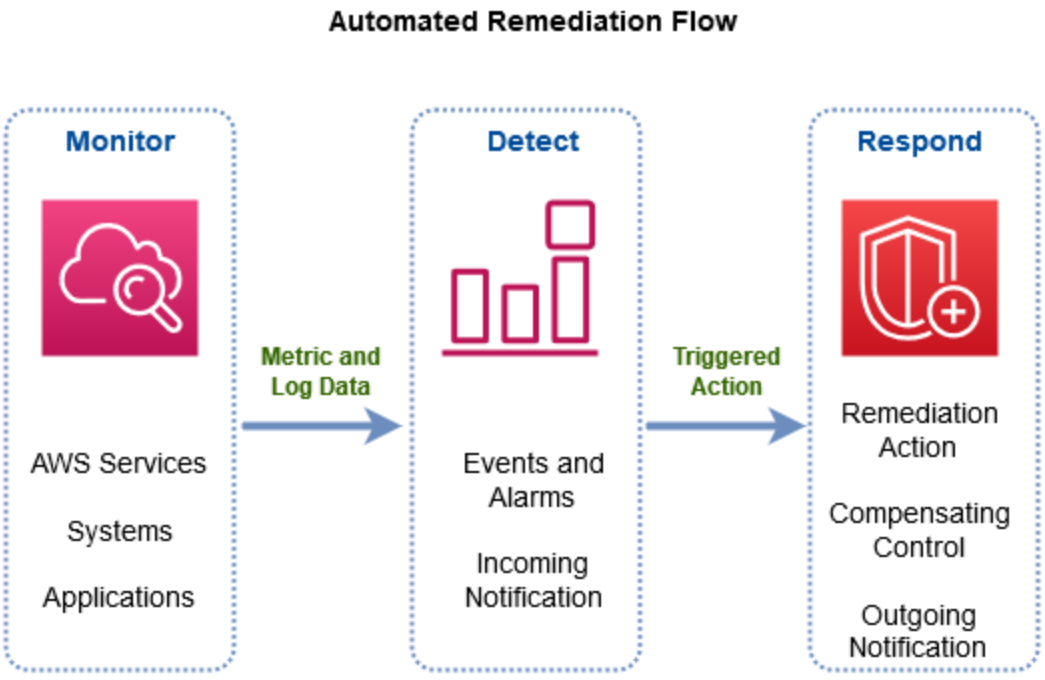
Incident and Event Response Management on AWS

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.



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.

Amazon S3 Event Notifications

The Amazon Simple Storage System (Amazon S3) service provides scalable and durable object storage for storing a variety of data. Companies can store static content, media files, data archives, and even confidential information on their S3 buckets. Many AWS Services also use it for storing CloudTrail logs, CloudFormation templates, EBS Snapshots, ELB access logs, and many others. You can also utilize Amazon S3 as a data lake for Amazon Redshift Spectrum, AWS Glue, Amazon Athena, or Amazon EMR. With this vast amount of stored information, organizations must ensure that their Amazon S3 buckets are secured and monitored for unexpected changes.

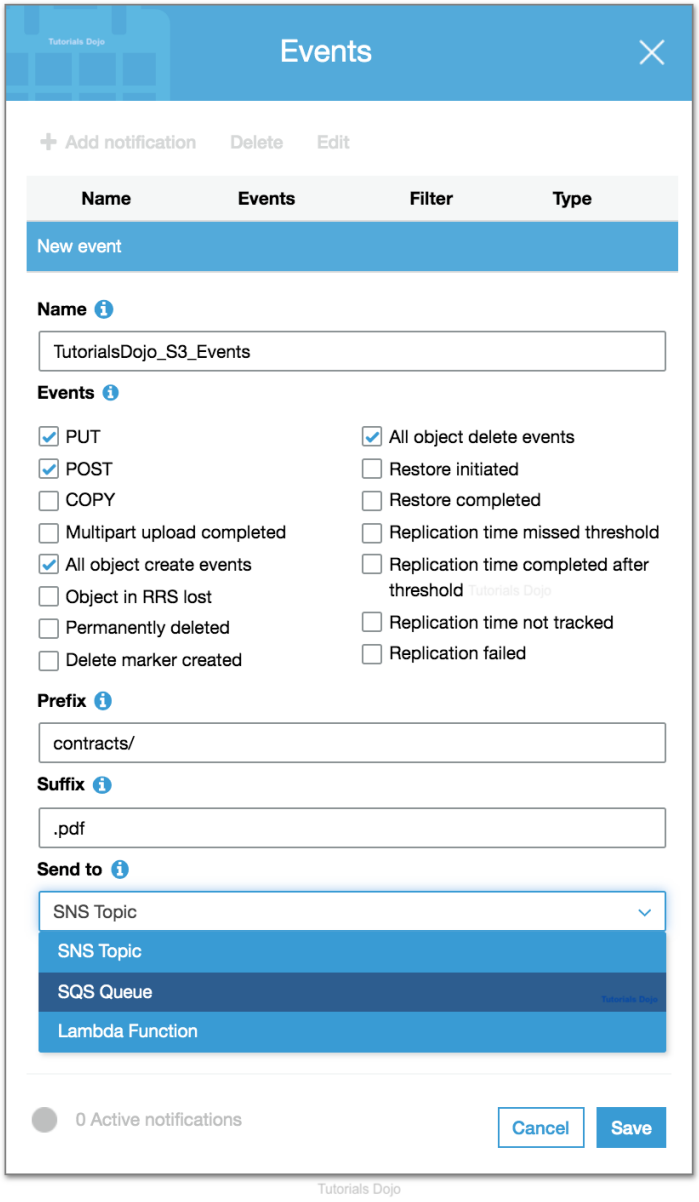
The Amazon S3 Event Notification feature enables teams to be notified when specific events happen in their S3 buckets. You can choose the particular S3 Events that you want to monitor, and Amazon S3 will publish the notifications to your desired destinations. This feature enables you to have more visibility on your data and promptly remediate any potential data leaks.

Amazon S3 can publish notifications for these events:

- Object creation
- Object deletion
- Object restoration from the S3 Glacier storage class
- Reduced Redundancy Storage (RRS) object lost events
- Objects eligible for replication using Amazon S3 Replication Time Control

You can configure Amazon S3 to publish events to these destinations:

- Amazon SNS Topic
- Amazon SQS) queue
- AWS Lambda

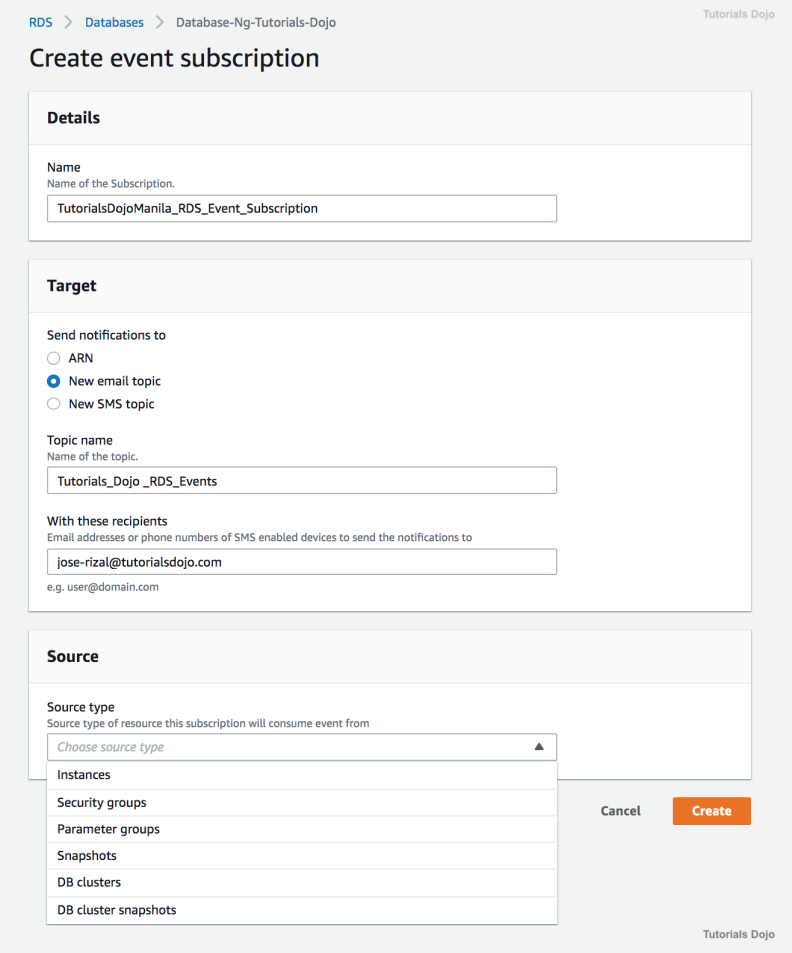


The S3 event notifications are usually transmitted within seconds and are designed to be delivered at least once. You can enable object versioning on your S3 bucket to ensure that an event notification is always sent for every successful object upload. With versioning, every successful write operation will produce a new version of your S3 object and send the corresponding event notification. Versioning averts the event notification issue where only one notification is sent when multiple operations are executed to a non-versioned object.

Source:
https://docs.aws.amazon.com/AmazonS3/latest/user-guide/enable-event-notifications.html

Amazon RDS Event Notifications

A database is a critical component of any enterprise system; therefore, it should be appropriately monitored for any potential issues. Like Amazon S3, there is also an event notification feature in Amazon RDS that notifies you of occurrences in your database resources. You can configure RDS to send you updates if an RDS DB instance has been created, restarted, or deleted. The RDS Event notifications can also detect low storage, configuration changes, Multi-AZ failover events, and many more.



Amazon RDS produces numerous events in specific categories that you can subscribe to using various tools such as the AWS CLI, Amazon RDS Console, or the RDS API. Each event category can refer to the parameter group, snapshot, or security group of your DB instance. Moreover, you can automatically process your RDS event notifications by using an AWS Lambda function or set an alarm threshold that tracks specific metrics by creating a CloudWatch Alarm.

Source:
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_Events.html

AWS_RISK_CREDENTIALS_EXPOSED Event

Step Functions makes it easy to develop and orchestrate components of operational response automation using visual workflows and it can be integrated with CloudWatch Events rule.

For example, your Development team is storing code on GitHub, and the developers sometimes forget to remove their personal IAM keys on their code before pushing to the repository. Since an exposed IAM key is a security issue, you want to make sure that you are notified of this event and the security issue is automatically remediated.

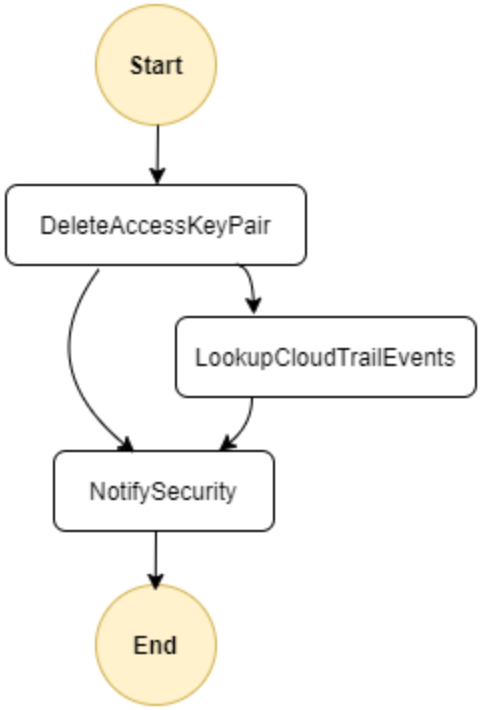
DevOps Exam Notes:

AWS can monitor popular code repository sites for IAM access keys that have been publicly exposed. Upon detection of an exposed IAM access key, AWS Health generates an **AWS_RISK_CREDENTIALS_EXPOSED** event in the AWS account related to the exposed key.

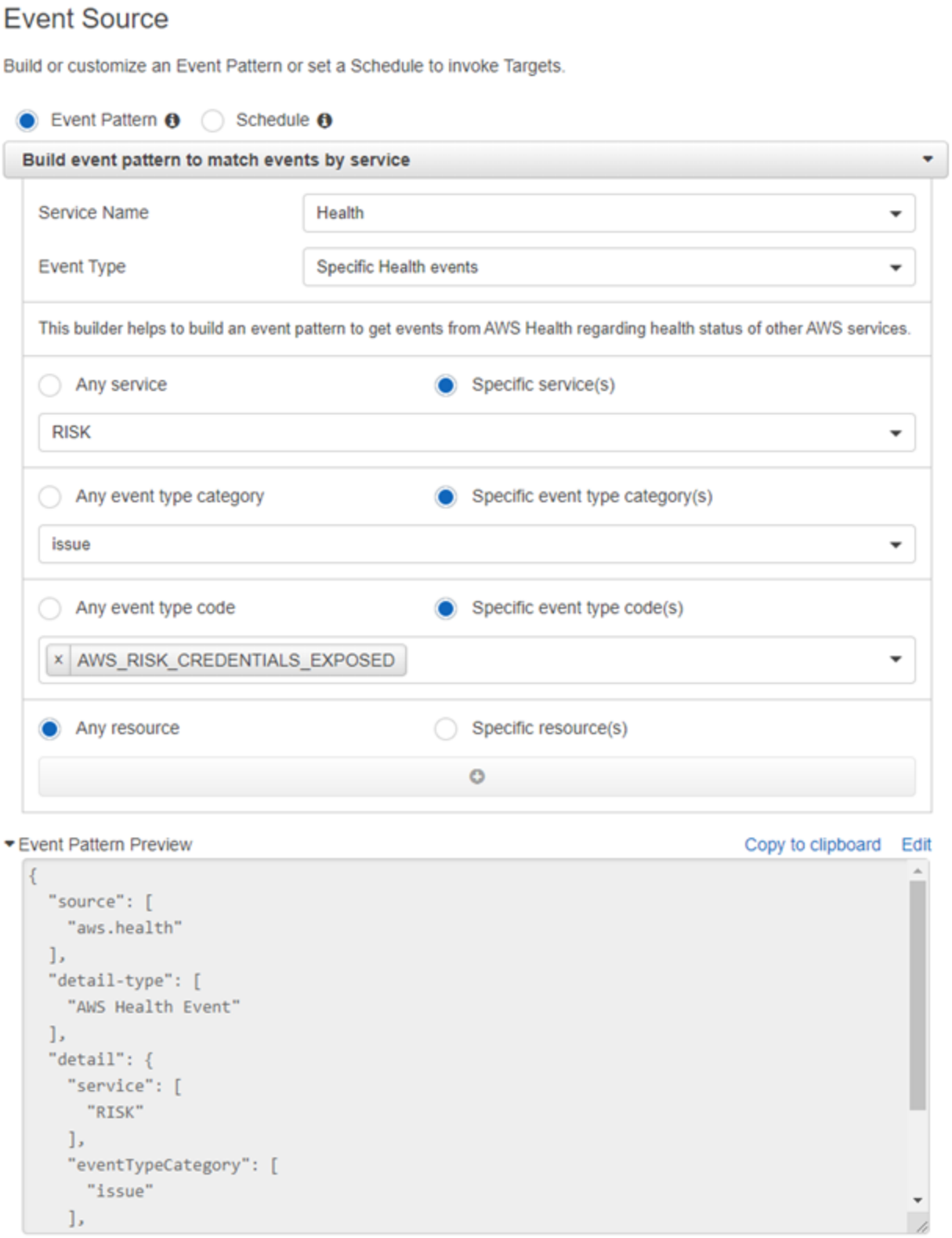
A configured CloudWatch Events rule detects this **RISK** service event and invokes a **Step Functions state machine**. The state machine then orchestrates the automated workflow that deletes the exposed IAM access key, summarizes the recent API activity for the exposed key, and sends the summary message to an Amazon SNS topic to notify the subscribers.

Here are the steps for this setup:

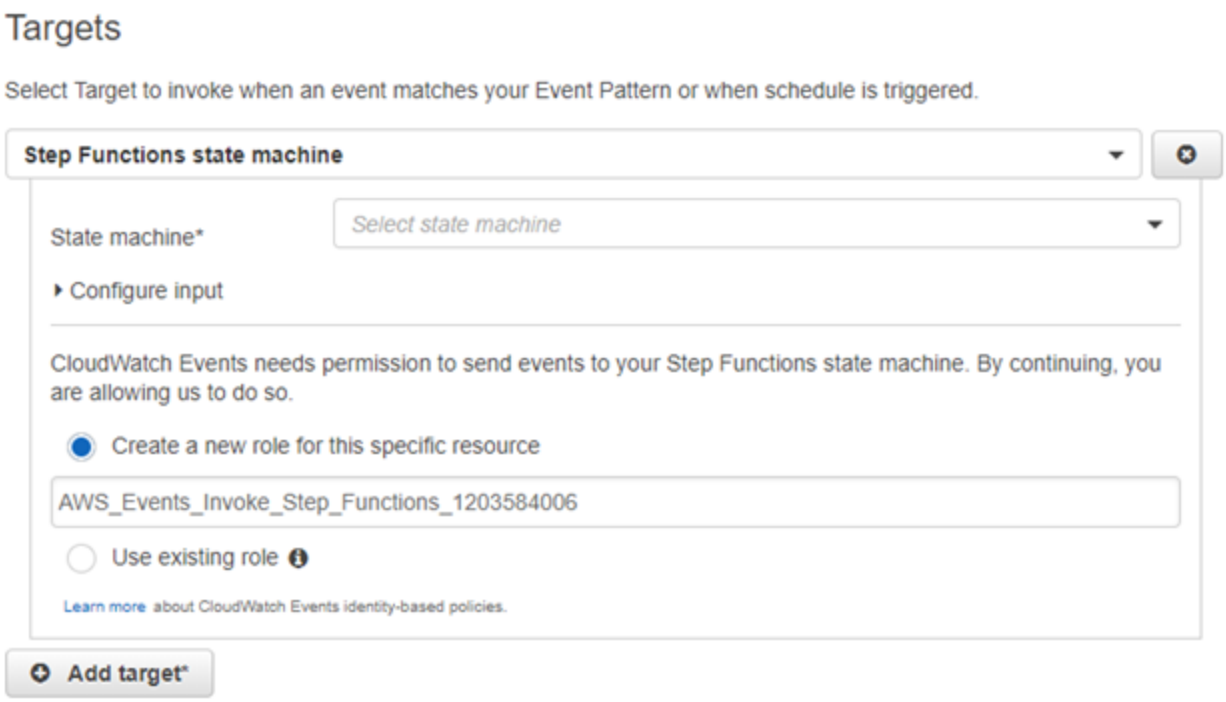
1. Create a State function state machine that does the following:
    a. Deletes the exposed IAM access key (to ensure that the exposed key can no longer be used)
    b. Summarizes the recent API activity for the exposed key from AWS CloudTrail (to know what changes were made using the exposed key)
    c. Send a summary message to an Amazon SNS topic to notify the subscribers.



2. Create a CloudWatch Events rule for detecting the **AWS_RISK_CREDENTIALS_EXPOSED** event from the Personal Health Dashboard Service.


Event Source
Build or customize an Event Pattern or set a Schedule to invoke Targets.
Event Pattern
Schedule
Build event pattern to match events by service
Service Name
Health
Event Type
Specific Health events
This builder helps to build an event pattern to get events from AWS Health regarding health status of other AWS services.
Any service
Specific service(s)
RISK
Any event type category
Specific event type category(s)
issue
Any event type code
Specific event type code(s)
AWS_RISK_CREDENTIALS_EXPOSED
Any resource
Specific resource(s)
Event Pattern Preview
Copy to clipboard
Edit
{
"source": [
"aws.health"
],
"detail-type": [
"AWS Health Event"
],
"detail": {
"service": [
"RISK"
],
"eventTypeCategory": [
"issue"
],

3. Set the Step function as Target for this CloudWatch Events rule.



Source:
https://aws.amazon.com/blogs/compute/automate-your-it-operations-using-aws-step-functions-and-amazon-cloudwatch-events/

AWS-Scheduled Maintenance Notification to Slack Channel via CloudWatch Events

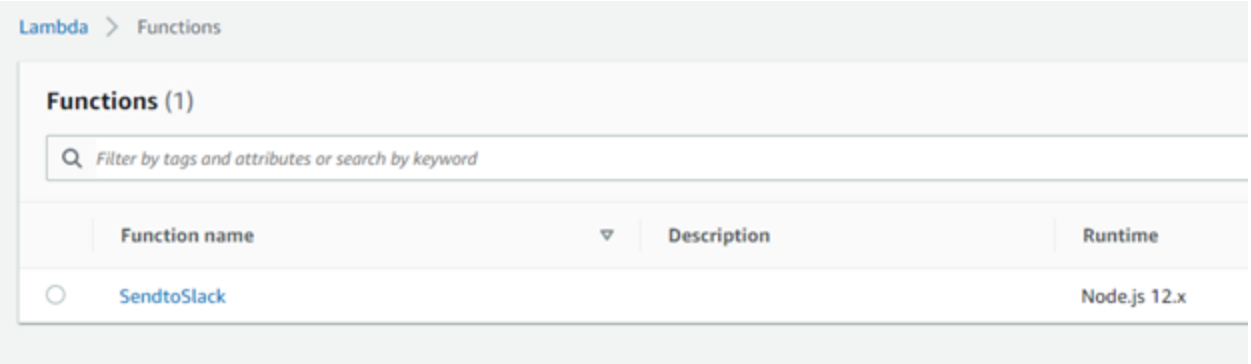
AWS Scheduled maintenance events are listed on AWS Health Dashboard. For example, if AWS needs to perform maintenance on the underlying host of your EC2 instance in which the EC2 instance usually needs to shut down, you will see an event on your AWS Health Dashboard for it. You can use CloudWatch Events 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:

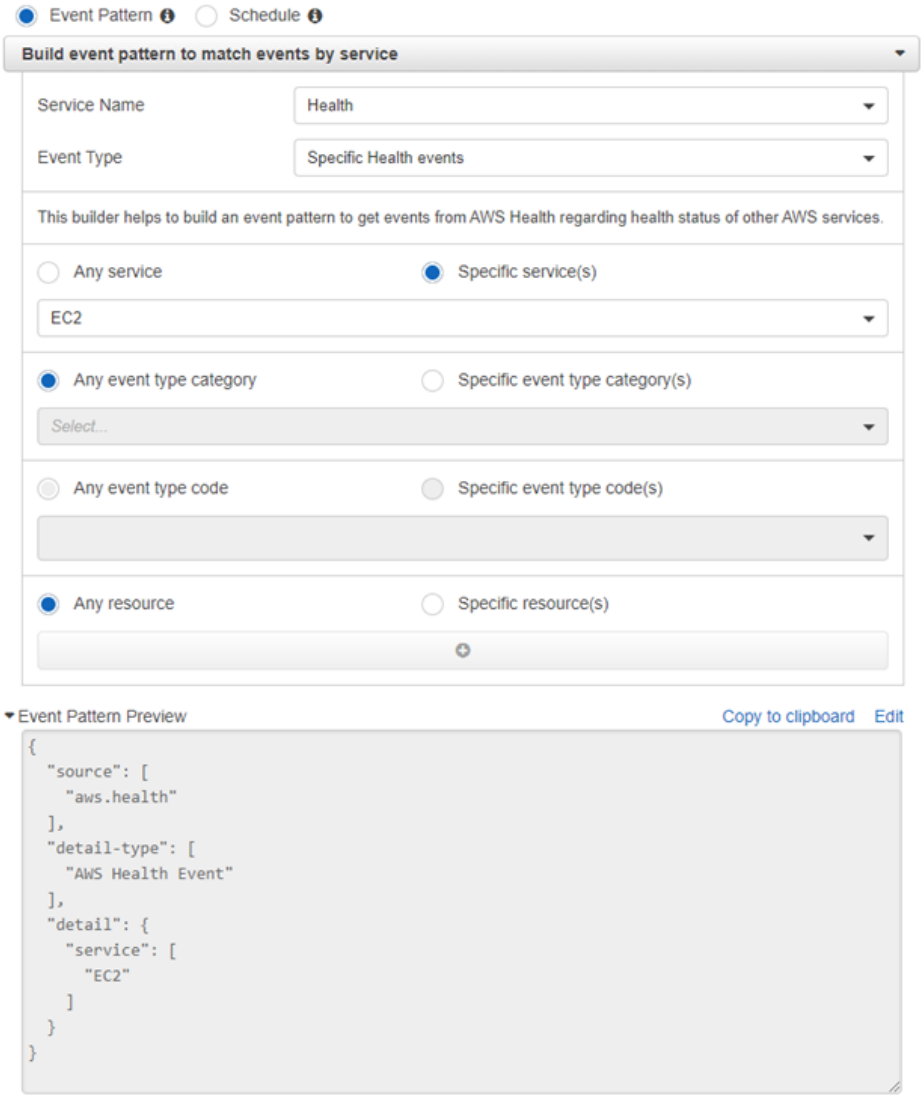
- AWS Lambda functions
- Amazon Kinesis Data Streams
- Amazon Simple Queue Service (Amazon SQS) queues
- Built-in targets (CloudWatch alarm actions)
- Amazon Simple Notification Service (Amazon SNS) topics

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.

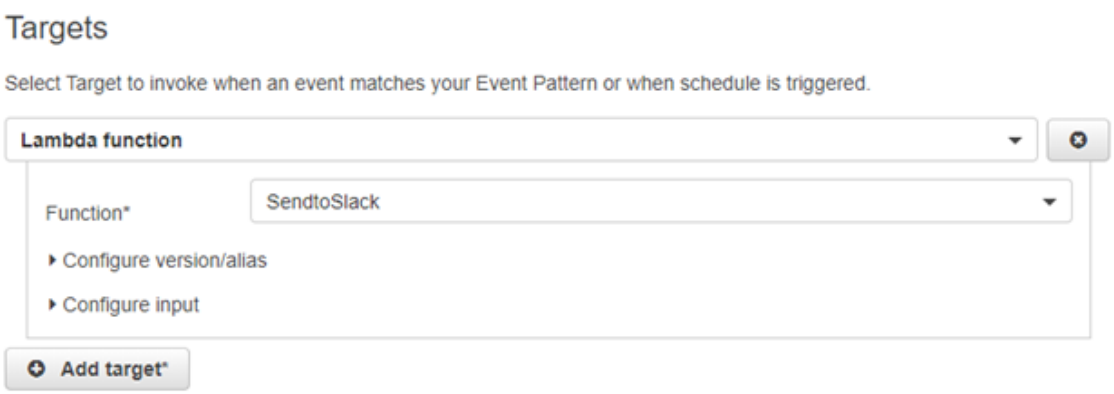
1. Create a Lambda function that will send a message to your Slack Channel. A Nodejs or Python script will suffice. The function will call an API URL for the Slack channel passing along the message.



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

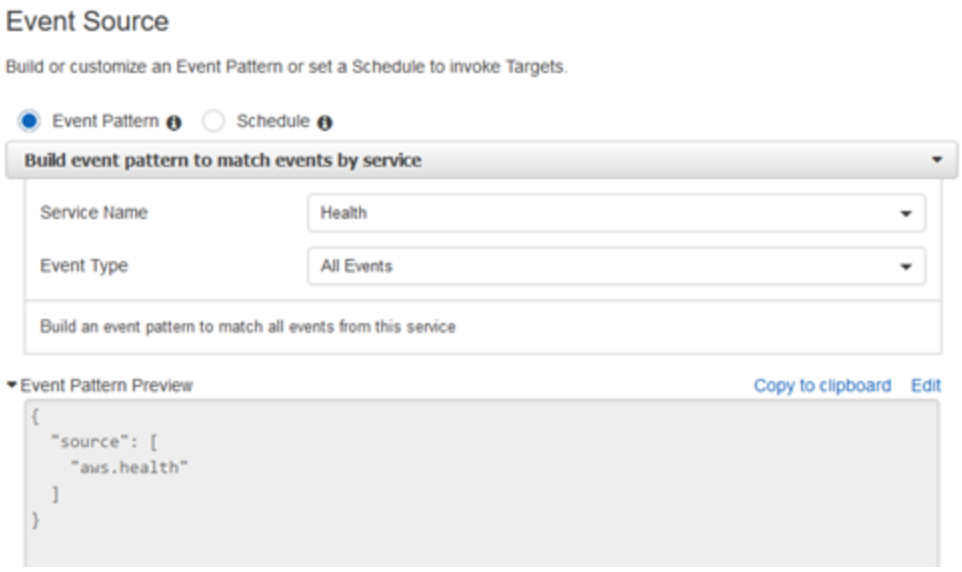
AWS Personal Health Dashboard provides you with alerts and remediation status when AWS is experiencing events that may impact your resources. These events may be scheduled or unscheduled. For example, scheduled events such as changes on your underlying EC2 hosts may shutdown or terminate your instances,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

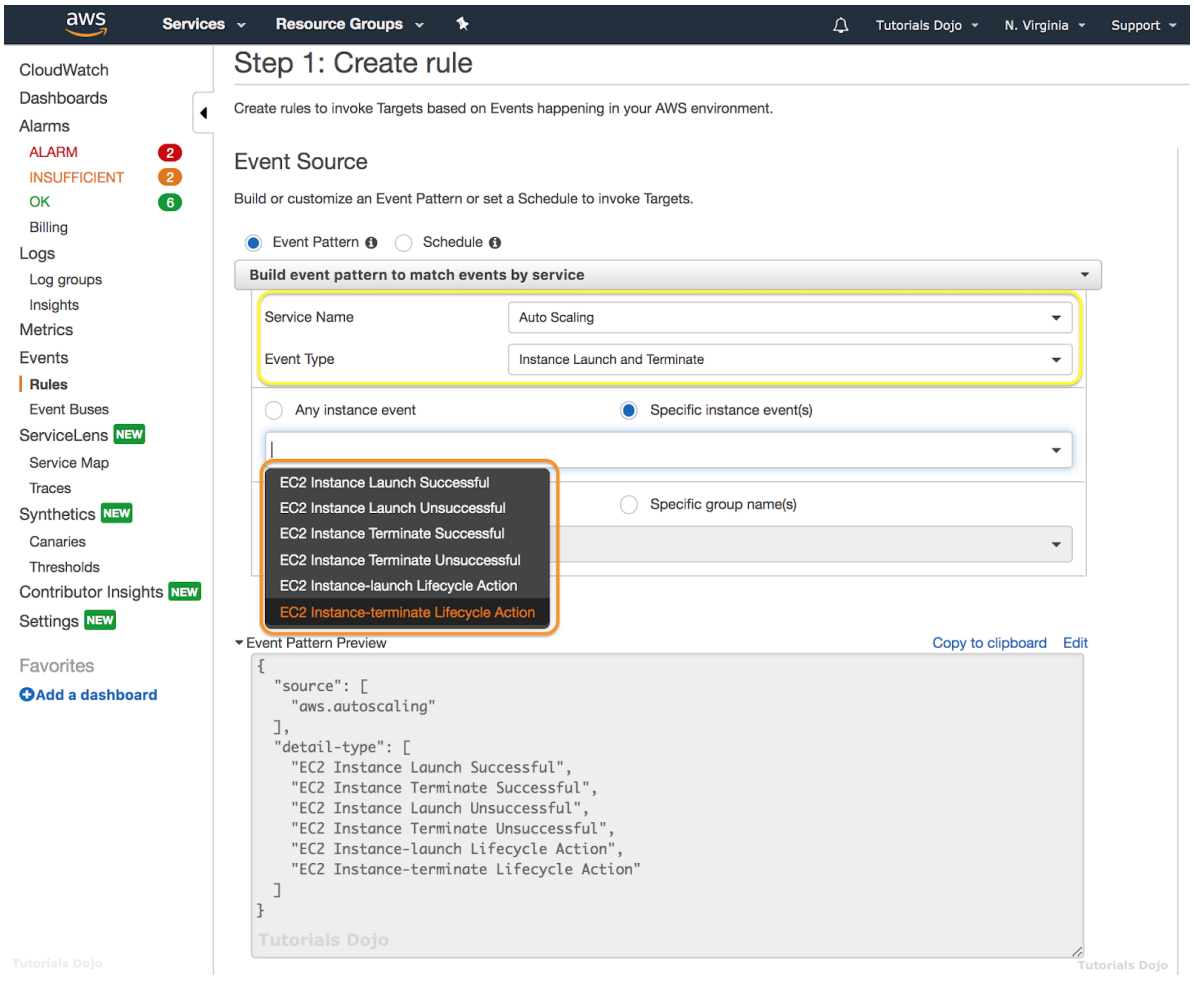


3. You can select a Target such as an SNS Topic if you want to get notified of the event, or Lambda function that can perform a custom action based on your requirements.
4. Click the Configure details to save and activate this CloudWatch Events rule.

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

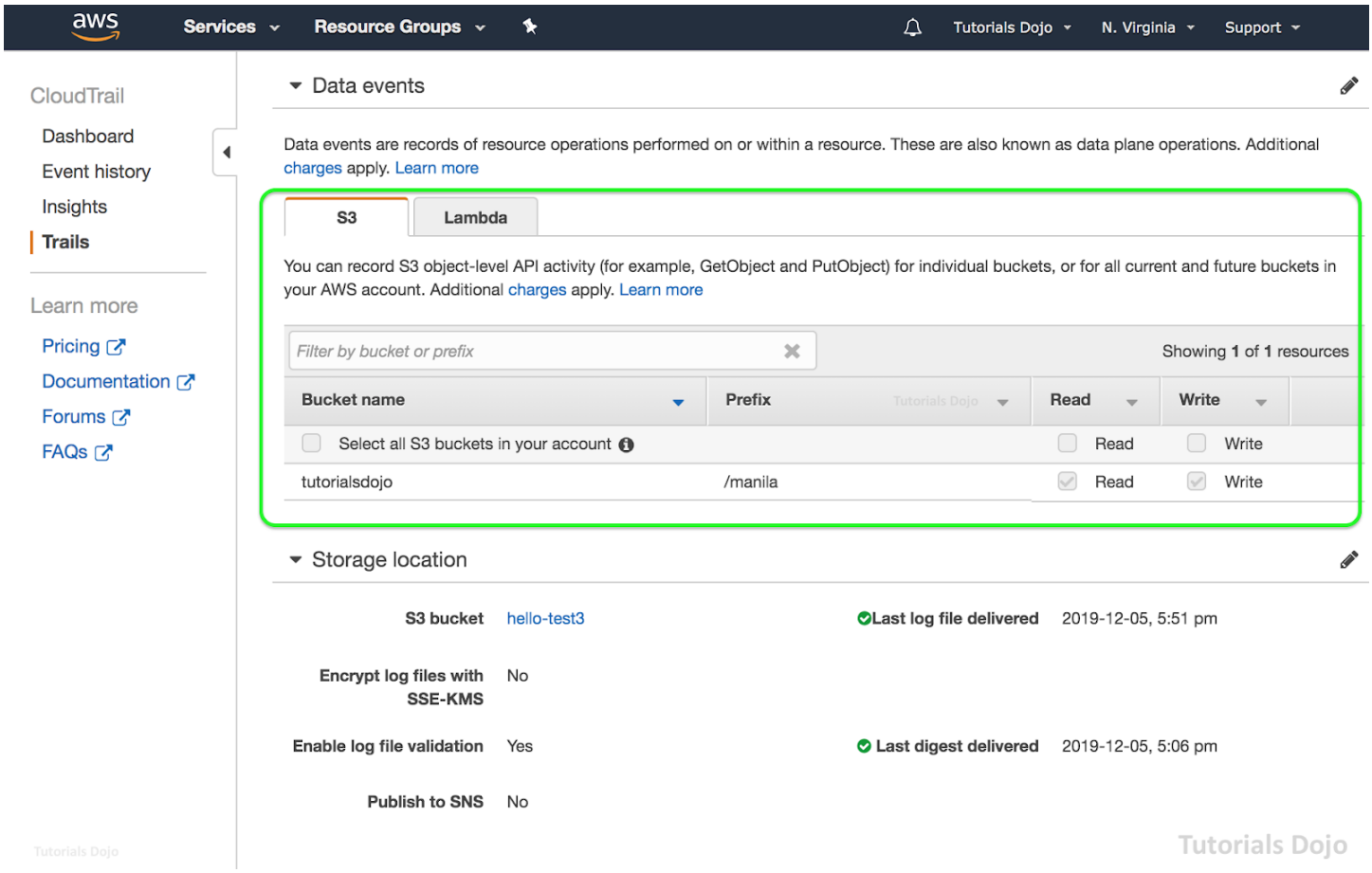
The **`EC2 Instance-launch Lifecycle Action`** is a scale-out event in which the Amazon EC2 Auto Scaling moved an EC2 instance to a **`Pending:Wait`** state due to a lifecycle hook. Conversely, the **`EC2 Instance-terminate Lifecycle Action`** is a scale-in event in which EC2 Auto Scaling updates an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

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

Monitoring Amazon S3 Data Events in AWS CloudTrail

AWS CloudTrail only logs bucket-level actions in your Amazon S3 buckets by default. If you want to record all object-level API activity in your S3 bucket, you can set up data events in CloudTrail. You also have the option to configure data event logging for your individual Lambda functions.

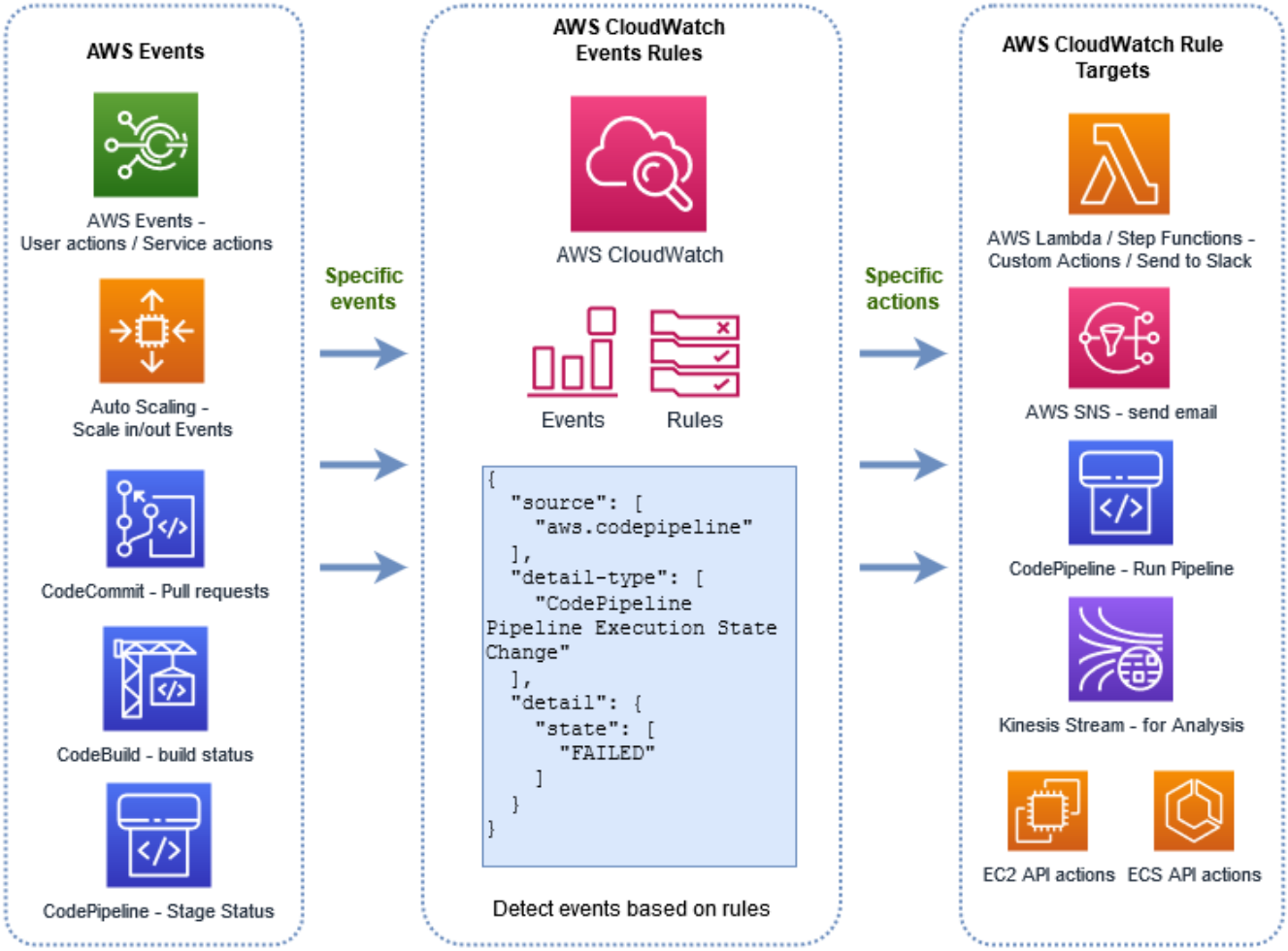
Every data event or log entry in CloudTrail contains essential information about who generated the request to your S3 buckets. This capability allows you to determine whether the S3 request was made by another AWS service, including the IAM User and temporary security credentials used. Amazon S3 Data Events duly records all S3 object changes and updates in your production S3 buckets.



Source:
https://docs.aws.amazon.com/AmazonS3/latest/dev/cloudtrail-logging.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"]
  }
}
```

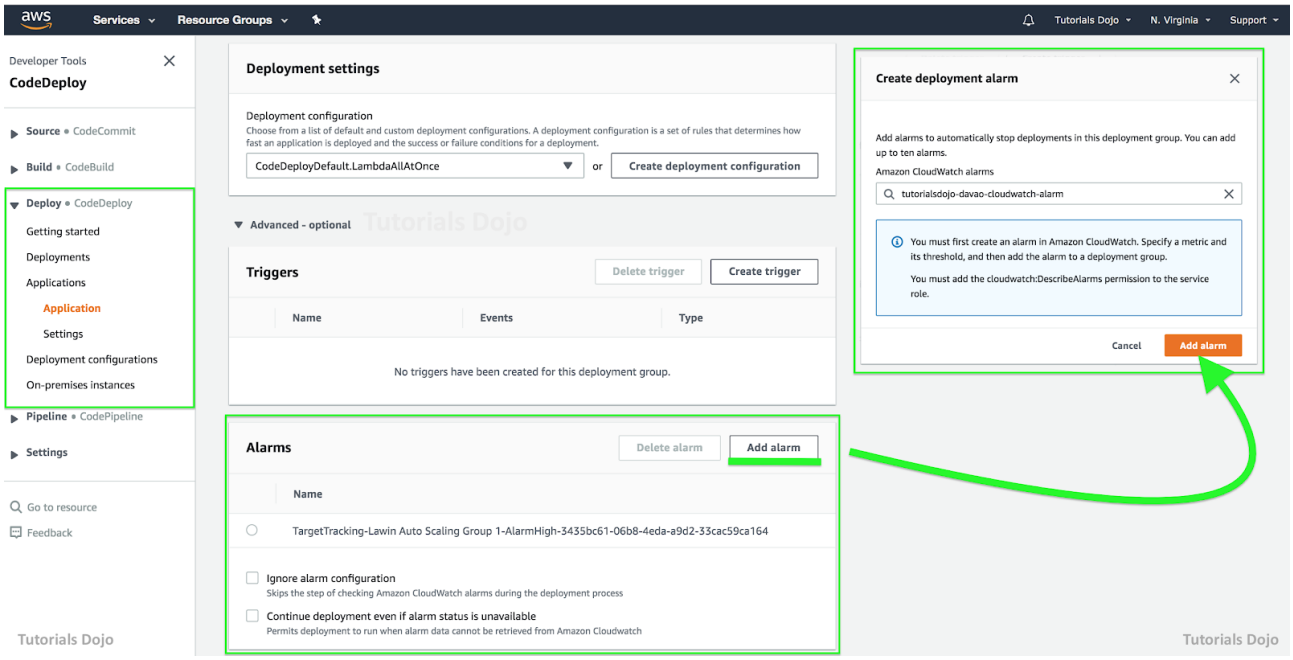
Source:
https://docs.aws.amazon.com/codepipeline/latest/userguide/detect-state-changes-cloudwatch-events.html

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.

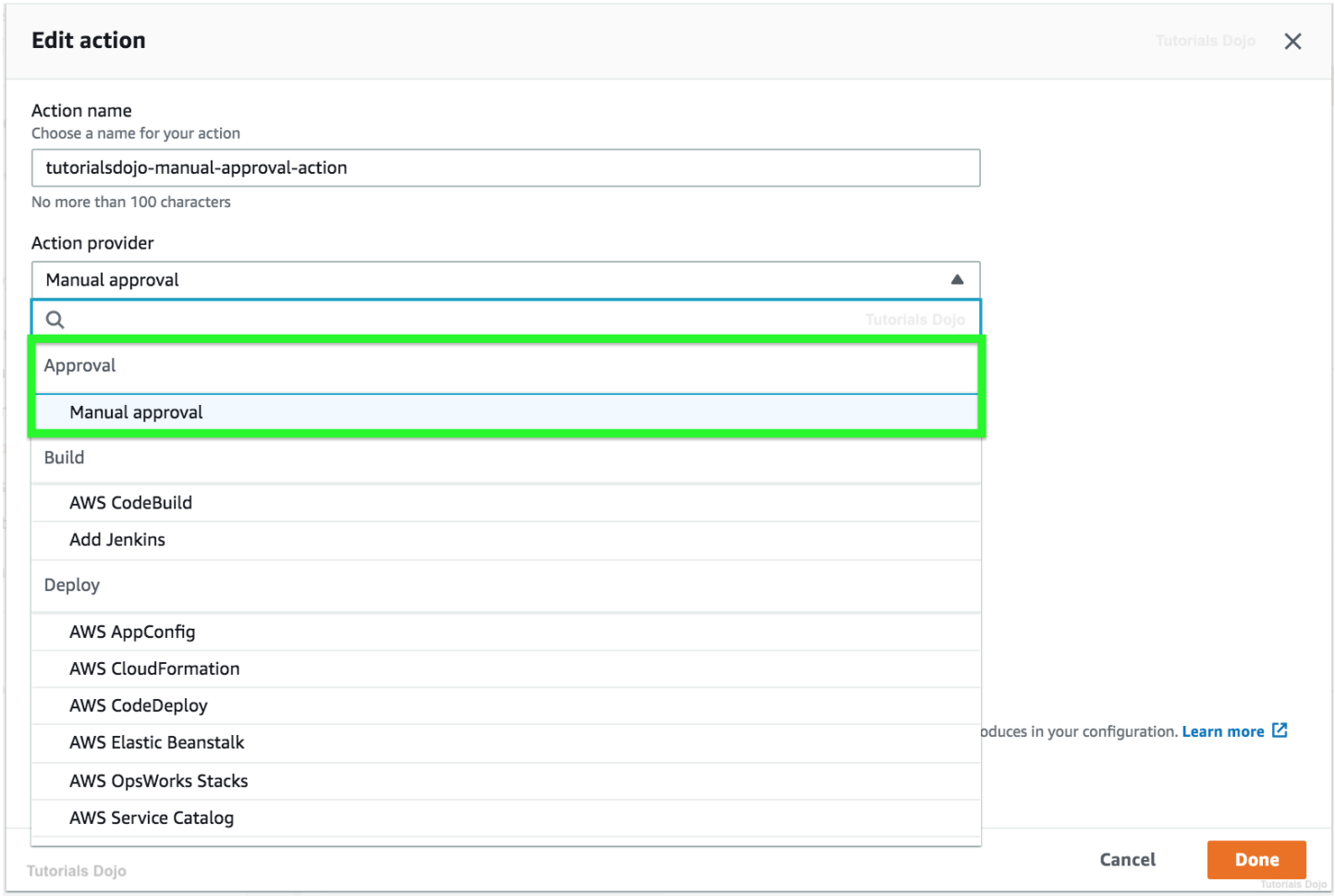


Source:
https://docs.aws.amazon.com/codedeploy/latest/userguide/monitoring-create-alarms.html

Orchestrating Events in AWS CodePipeline

In AWS CodePipeline, you can control your pipeline's actions and stages to optimize the performance of your CI/CD processes. The **`runOrder`** and **`PollForSourceChanges`** parameters can assist you in orchestrating the various activities in your pipeline. For example, you have a serverless application with independent AWS Lambda functions, and you want to expedite the pipeline execution time. You can modify your CodePipeline configuration to execute actions for each Lambda function in parallel by specifying the same **`runOrder`** value. The **`PollForSourceChanges`** parameter automates the pipeline creation by tracking source changes from CodeCommit, Amazon S3, or Github. In this way, you will have more control over the various stages of your pipeline.

You can also add a manual approval action in CodePipeline before any change is deployed to your production environment. This provides a final checkpoint for your release process after all your unit and integration tests were successfully completed.



By adding a manual approval step in your deployment, you can:

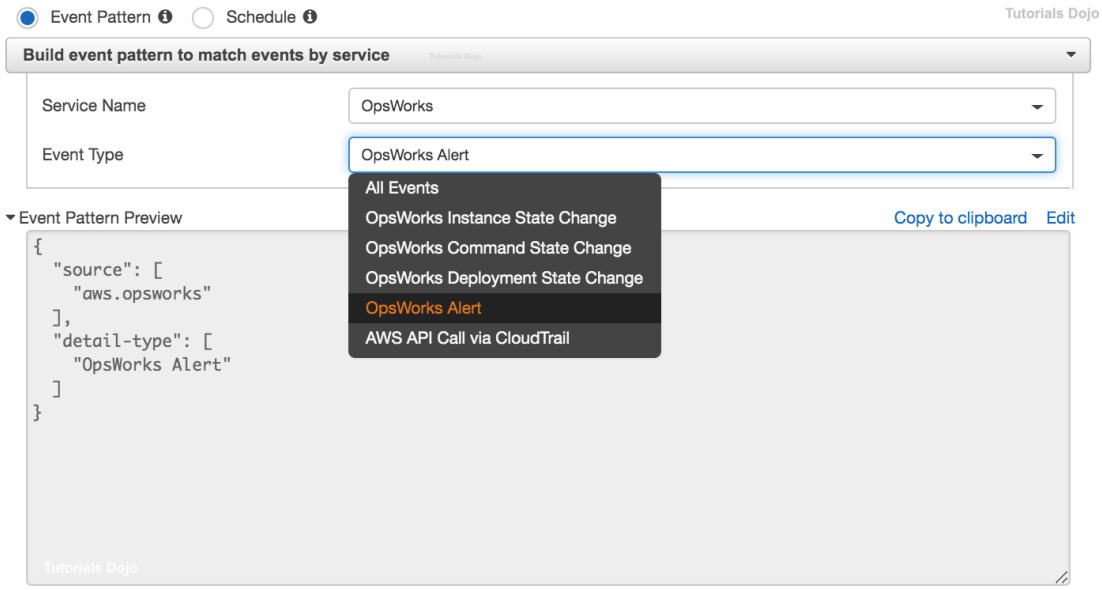
- Perform a code review before a revision is pushed to the next stage of a pipeline.
- Perform manual quality assurance testing on the latest version of a feature or application
- Review any updated feature or application before it is published to the production environment

AWS CodePipeline offers a way to publish approval notifications for your release managers and other authorized staff. A pipeline can be configured with a manual approval action to push a message to an Amazon SNS topic when the approval action was invoked. Amazon SNS delivers the message to every endpoint subscribed to the SNS topic that you specified. Amazon SNS lets the approvers know, via email or SMS, that a new update is ready to be deployed. You can also forward these notifications to SQS queues or HTTP/HTTPS endpoints and execute a custom action using a Lambda function.

Sources:
https://docs.aws.amazon.com/codepipeline/latest/userguide/reference-pipeline-structure.html#action-requirements
https://docs.aws.amazon.com/codepipeline/latest/userguide/approvals.html

Monitoring OpsWorks Auto-Healing Events

If you are using AWS OpsWorks Stacks resources, you can set up rules in Amazon CloudWatch Events to notify you for any changes in your application stack and execute certain actions that you specify. Any changes in the state of your instance, command or deployment in OpsWorks stack can be properly monitored, including service alerts.



You can also prevent OpsWorks Stacks from automatically healing the EC2 instances that it manages by declaring **`aws.opsworks`** as a source and setting **`auto-healing`** in the **`initiated_by`** detail parameter in your CloudWatch Events rule. This rule is helpful in situations where AWS OpsWorks Stacks erroneously restarts instances that it determines to be unhealthy, even if the EC2 health checks are successfully passing. You can refer to the event pattern below to enable this capability:

```
{
  "source": [
    "aws.opsworks"
  ],
  "detail": {
    "initiated_by": [
      "auto-healing"
    ]
  }
}
```

Source:
https://aws.amazon.com/premiumsupport/knowledge-center/opsworks-unexpected-start-instance/

Domain 6: High Availability, Fault Tolerance, and Disaster Recovery

Overview

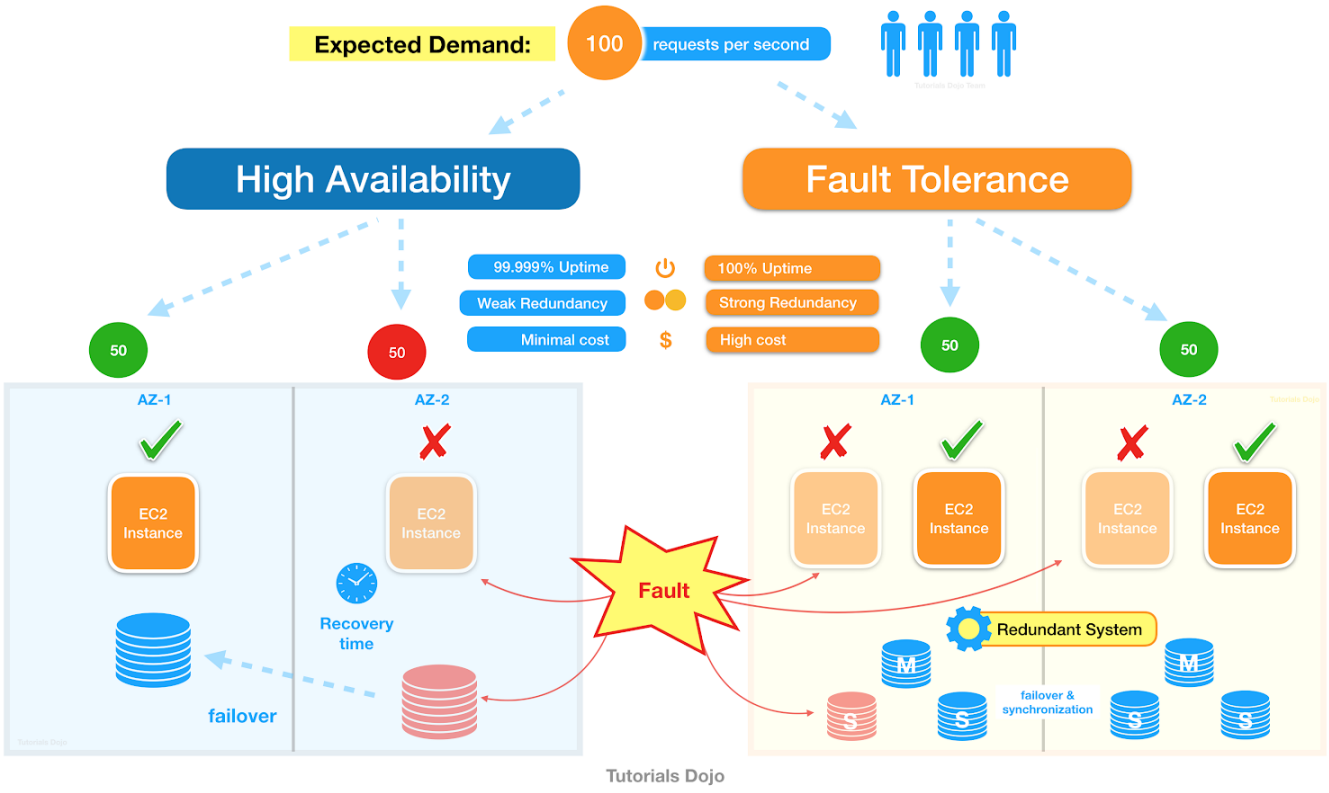
The sixth and final exam domain of the AWS Certified DevOps Engineer Professional test covers High Availability, Fault Tolerance, and Disaster Recovery. This domain will verify your capability in implementing the following:

- Determine appropriate use of multi-AZ versus multi-region architectures
- Determine how to implement high availability, scalability, and fault tolerance
- Determine the right services based on business needs (e.g., RTO/RPO, cost)
- Determine how to design and automate disaster recovery strategies
- Evaluate a deployment for points of failure

High Availability vs. Fault Tolerance

Many IT Professionals nowadays do not have the right understanding of High Availability and Fault Tolerance, even those who are already in the IT industry for a long time. These two concepts are quite similar to each other, and sometimes, people use these two terms interchangeably. But are they theoretically the same?

Both High Availability and Fault Tolerance have the same objective of ensuring that your application runs all the time without any system degradation. However, these concepts have unique attributes that differentiate them from each other. These two differ in cost, design, redundancy level, and behavior on component faults or failures.



High Availability aims for your application to run 99.999% of the time. Its design ensures that the entire system can quickly recover if one of its components crashed. It has an ample number of redundant resources to allow failover to another resource if the other one fails. This concept accepts that a failure will occur but provides a way for the system to recover fast.

Fault Tolerance, on the other hand, has the goal of keeping your application running with zero downtime. It has a more complex design, and higher redundancy to sustain any fault in one of its components. Think of it as an upgraded version of High Availability. As its name implies, it can *tolerate* any component fault to avoid any performance impact, data loss, or system crashes by having redundant resources beyond what is typically needed. The caveat for implementing a fault-tolerant system is its cost as companies have to shoulder the capital and operating expenses for running its required numerous resources.

A system can be highly available but not fault-tolerant, and it can be both. If an application is said to be fault-tolerant then it is also considered highly available. However, there are situations in which a highly available application is not considered fault-tolerant.

There are various services, features, and techniques in AWS that you can use to implement a highly available and fault-tolerant architecture. You can ensure high availability by deploying your application to multiple Availability Zones or several AWS Regions. Auto Scaling can dynamically scale your systems depending on the incoming demand, and an active-active or active-passive failover policy can be implemented in Route 53 to reduce downtime. Amazon RDS offers Automated Snapshots, Read Replica, and Multi-AZ Deployments to strengthen your database tier to remove single point of failure in your system. Alternatively, you can opt to use the Amazon Aurora Global database or DynamoDB Global tables for your globally-accessible applications. You can also leverage on the self-healing capabilities of AWS services to achieve fault-tolerance.

Multi-AZ vs Multi-Region Architecture

The AWS Global Architecture is composed of an extensive worldwide network of Local Zones, Availability Zones (AZs), Regions, Regional Edge Caches, and Edge locations to enable organizations to scale and go global in minutes. Cloud computing has eliminated the boundaries of deploying and running applications in limited and costly on-premises data centers. Both small and large companies can now serve their customers more efficiently and cost-effectively using AWS. Businesses can opt to deploy non-prod environments in a single Availability Zone to save on cost or design a Multi-AZ architecture for its mission-critical applications. Deploying clusters of AWS resources across multiple Availability Zones can withstand an outage in one or more zones in your primary AWS Region. You can also design a Multi-Region architecture to serve your global users and comply with your disaster recovery requirements.

You can adopt these types of architectures both on your application and database tier. AWS Auto Scaling helps your web servers handle sudden or expected demand in your application. For your database, you can deploy one or more Read Replicas to offload the surge of incoming load from the primary database instance. For Multi-AZ, the OS patching or DB Instance scaling operations are applied first on the standby instance before triggering the automatic failover to reduce interruptions and downtime.

	MULTI-AZ DEPLOYMENTS	MULTI-REGION DEPLOYMENTS	READ REPLICAS
OBJECTIVE	High availability	Disaster recovery and local performance	Scalability
REPLICATION TYPE	Non-Aurora: Synchronous replication; Aurora: Asynchronous replication	Asynchronous replication	Asynchronous replication
OPERABILITY	Non-Aurora: Only the primary instance is active; Aurora: All instances are active	All regions are accessible and can be used for reads	All read replicas are accessible and can be used for readscaling
BACKUPS	Non-Aurora: Automated backups are taken from standby; Aurora: Automated backups are taken from shared storage layer	Automated backups can be taken in each region	No backups configured by default
SCOPE	Always span at least two Availability Zones within a single region	Each region can have a Multi-AZ deployment	Can be within an Availability Zone, Cross-AZ, or Cross-Region
DB UPGRADE	Non-Aurora: Database engine version upgrades happen on primary; Aurora: All instances are updated together	Non-Aurora: Database engine version upgrade is independent in each region; Aurora: All instances are updated together	Non-Aurora: Database engine version upgrade is independent from source instance; Aurora: All instances are updated together
FAILOVER PROCESS	Automatic failover to standby (non-Aurora) or read replica (Aurora) when a problem is detected	Aurora allows promotion of a secondary region to be the master	Can be manually promoted to a standalone database instance (non-Aurora) or to be the primary instance (Aurora)

TUTORIALS DOJO TUTORIALS DOJO

Disaster Recovery Objectives

There are two types of Objectives in Disaster Recovery:

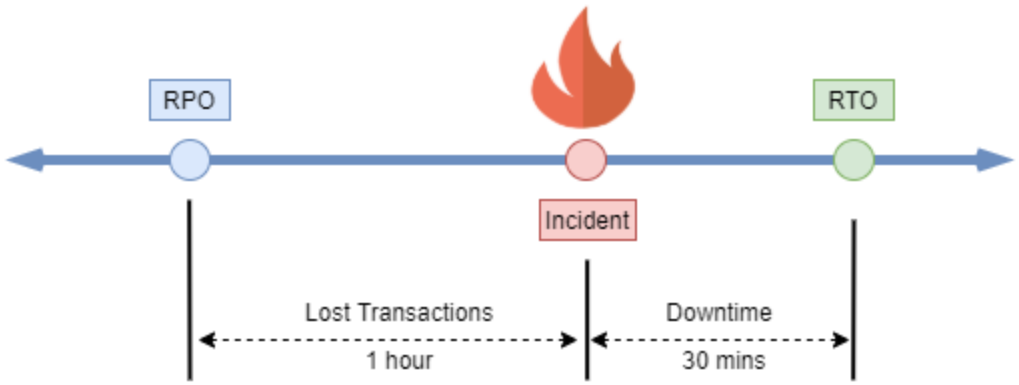
1. RTO or Recovery Time Objective
2. RPO or Recovery Point Objective

Basically, RTO refers to the time and RPO is all about the data point. RTO is the time it takes after a disruption to restore a business process to its service level, as defined by the operational level agreement (OLA).

For example, if a disaster occurs at 12 noon and the RTO is 3 hours, the Disaster Recovery process should restore service to the acceptable service level on or before 3 PM.

RPO, on the other hand, is the acceptable amount of data loss measured in time. For example, if a disaster occurs at 12 noon and the RPO is one hour, the system should recover all data that was in the system before 11 AM. The acceptable data loss should only be one hour, between 11:00 AM and 12:00 PM (noon). If you cannot recover a transaction or data made before 11:00 AM , say 10:30 or 9:30 then this means that you have failed your RPO.

RTO refers to the amount of time for the system to recover from an outage. RPO is the specific point, or state, of your data store that needs to be recoverable.



Amazon Route 53 Routing Policies

Most large organizations often have complex network structures to support their hybrid cloud architecture, distributed systems, and global users. They have several on-premises networks integrated with AWS Direct Connect or AWS VPN to connect to their AWS cloud resources across multiple Availability Zones and AWS Regions. To ensure business continuity, companies implement a disaster recovery plan that fails over the production traffic from the primary environment to the disaster recovery (DR) site.

Amazon Route 53 is a global Domain Name System (DNS) service that allows you to route traffic across various AWS Regions and external systems outside of AWS. It provides a variety of routing policies that you can implement to meet your required use cases and automatically monitor the state and performance of your applications, servers, and other resources using health checks. You can combine two or more routing policies to comply with your company's strict RTO and RPO requirements. It helps simplify the process of setting up an active-passive or active-active failover for your disaster recovery plan by intelligently routing traffic from your primary resources to the secondary resources based on the rules you specify.

Your globally distributed resources can either be considered active or passive. It's active if it accepts live production traffic and passive if it is just on standby, which will only be activated during a failover event. You can set up an active-active failover to improve your systems' fault tolerance and performance. By having several active environments, you can ensure the high availability and resiliency of your global applications. To set up an active-active failover, you can use a single or a combination of routing policies such as latency, geolocation, geoproximity, and others to configure Route 53 to respond to a DNS query using any healthy record.

Below are the different types of Amazon Route 53 routing policies that you can use in your architecture:

- **Simple -** This routing policy is commonly used for a single resource that performs a straight-forward function for your domain records. For example, you can use this policy to route traffic from tutorialsdojo.com apex domain to an NGINX web server running on an Amazon EC2 instance.
- **Failover** – As the name implies, you can use this policy to set up an active-passive failover for your network architecture.
- **Geolocation** – Amazon Route 53 can detect the geographical location where the DNS queries originated. This routing policy lets you choose the specific resources that serve incoming traffic based on your users' geographic location. Say, you might want all user traffic from North America routed to an Application Load Balancer in the Singapore region. It works by mapping the IP addresses to geographical areas using the Extension Mechanisms for DNS version 0 (EDNS0).
- **Geoproximity** – This one is similar to the Geolocation routing policy except that it uses the Traffic Flow feature of Route 53 and has an added capability of shifting more or less traffic to your AWS services in one geographical location using a bias. It concentrates on the proximity of the resource in a given geographic area rather than its exact location.

- **Latency** – You can improve the application performance for the benefit of your global users by serving their requests from the AWS Region that provides the lowest latency. This routing policy is suitable for organizations that have resources in multiple AWS Regions.
- **Multivalue Answer** – Unlike the *Simple* routing policy, this type can route traffic to numerous resources in response to DNS queries with up to eight active records selected randomly. This policy is perfect if you are configuring an active-active failover for your network.
- **Weighted** – This policy allows you to route traffic to multiple resources in proportions that you specify. It acts as a load balancer that routes requests to a record based on the relative percentage of traffic or weight that you specified.

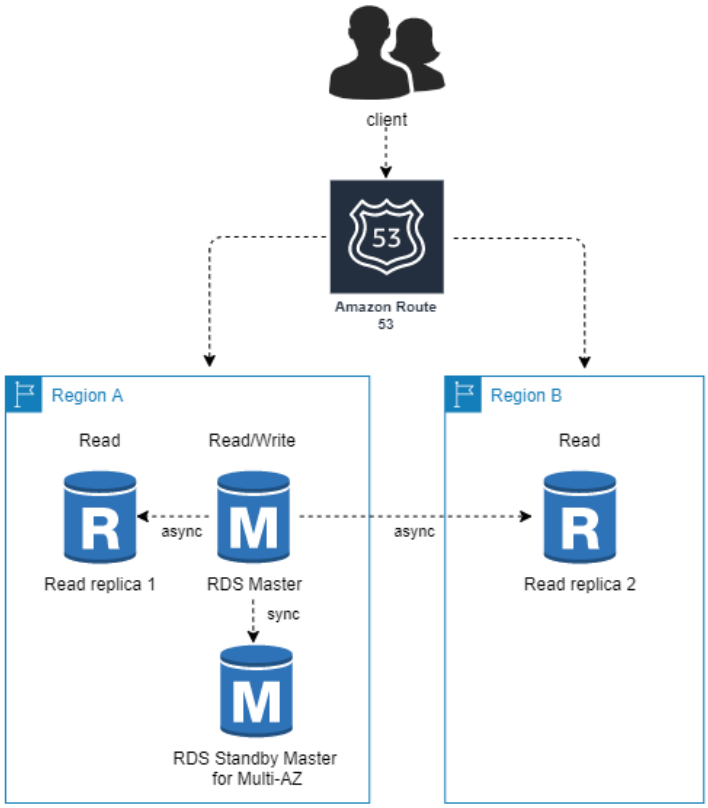
To monitor the system status or health, you can use Amazon Route 53 health checks to properly execute automated tasks to ensure the availability of your system. Health checks can also track the status of another health check or an Amazon CloudWatch Alarm.

Sources:
https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/routing-policy.html
https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/disaster-recovery-resiliency.html
https://tools.ietf.org/html/rfc2671

Amazon RDS Disaster Recovery Strategies

AWS provides a plethora of database services and database engines that you can use for your applications. If you need a relational database type, you can use Amazon Aurora, Amazon RDS, or Amazon Redshift. For key-value data store, you can use an Amazon DynamoDB table that you can upgrade to DynamoDB Global to support your users worldwide. Amazon ElastiCache is perfect for in-memory data stores where you can use a Redis or Memcached engine. You can choose many other database services for document, wide column, graph, time series, and ledger type databases.

Amazon Aurora is a fully managed MySQL and PostgreSQL-compatible relational database that provides high performance, availability, and scalability to your applications. Since it is a fully managed service, Amazon handles all of the underlying resources in your Aurora database and ensures that your cluster is highly available to meet your disaster recovery objectives and achieves fault tolerance. Aurora is excellent, but it has certain limitations, which compels companies to choose Amazon RDS as their database tier. Aurora does not use a native MySQL or PostgreSQL engine like RDS and can't directly run Oracle and Microsoft SQL Server databases unless you migrate them using the AWS Database Migration Service (AWS DMS) and AWS Schema Conversion Tool (AWS SCT). These constraints are the reasons why thousands of companies are still using Amazon RDS in their cloud architecture.



Amazon RDS is a managed database service. Unlike its "fully managed" counterparts, AWS does not entirely 'manage' or control all of the components of an Amazon RDS database compared to what it does for Amazon Aurora. If you launched an RDS database, you are responsible for making it highly scalable and highly available by deploying Read Replicas or using Multi-AZ Deployments configurations. You can also improve the data durability of your database-tier by taking automated or manual snapshots in RDS. For disaster recovery planning, you can set up a disaster recovery (DR) site to another AWS Region if the primary region becomes unavailable.

	DISASTER RECOVERY		COST	SCOPE
	RTO	RPO		
AUTOMATED BACKUPS	GOOD	BETTER	LOW	SINGLE REGION
MANUAL SNAPSHOTS	BETTER	GOOD	MEDIUM	CROSS-REGION
READ REPLICAS	BEST	BEST	HIGH	CROSS-REGION

An RDS Read Replica is mainly used to vertically scale your application by offloading the read requests from your primary DB instance. But do you know that it is tremendously useful for disaster recovery too? It uses asynchronous replication to mirror all the changes from the primary instance to the replica, located on the same or a different AWS Region. In contrast, the Multi-AZ Deployments configuration uses synchronous replication to keep its standby instance up-to-date. As its name implies, the standby instance is just on *standby*, meaning it neither accepts read nor write requests. This standby instance can only run on the same AWS Region, unlike a Read Replica with a cross-region capability. These unique attributes enable the Read Replica to provide the best RTO and RPO for your disaster recovery plan. You can deploy a Read Replica of your RDS database to another AWS Region to expedite the application failover if the primary region becomes unavailable without having to wait for hours to migrate and launch the automated/manual RDS snapshots to the other region.

You should also know the difference between automated backups, manual snapshots, and Read Replicas for your Business Continuity Plan (BCP). Amazon RDS has a built-in automated backups feature that regularly takes snapshots of your database and stores it on an Amazon S3 bucket that is owned and managed by AWS. The retention period of these backups vary between 0 and 35 days. It provides a low-cost DR solution for your database-tier but is only limited to a single AWS Region. Manual snapshots are the ones that you manually take

yourself, hence the name. In contrast with the automated backups, the S3 bucket where the snapshots are stored is owned by you, which means that you can control its retention period and deploy cross-region snapshots. Since you manage your own RDS snapshots, you can move these across AWS Regions using a shell script or a Lambda function run by CloudWatch Events regularly.

The advantage of using Read Replicas over automated backups and manual snapshots is its near real-time synchronous replication. To put it into perspective, the replication time of the primary DB instance to the replica instance is less than a second! Compare that to the time required to move an RDS snapshot across another region and waiting for it to start up. Hence, Read Replicas provide the fastest RTO and the best RPO for your architecture. The only setback is its high cost since you have to run your replica continuously.

Sources:
https://aws.amazon.com/blogs/database/implementing-a-disaster-recovery-strategy-with-amazon-rds/
https://d0.awsstatic.com/whitepapers/Backup_and_Recovery_Approaches_Using_AWS.pdf
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_CopySnapshot.html

Auto Scaling Group with MinSize = 1 and MaxSize = 1

With Auto Scaling, you can set the number of EC2 instances that you need depending on the traffic of your application. However, there will be scenarios on the exam where you will need to set a fixed number of instances.

For example, you have a legacy application hosted on an EC2 instance. The application does not support working on a cluster of EC2 instances so it needs to run on a single EC2 instance and you need to make sure that the application is available and will heal itself even if that EC2 instance crashes.

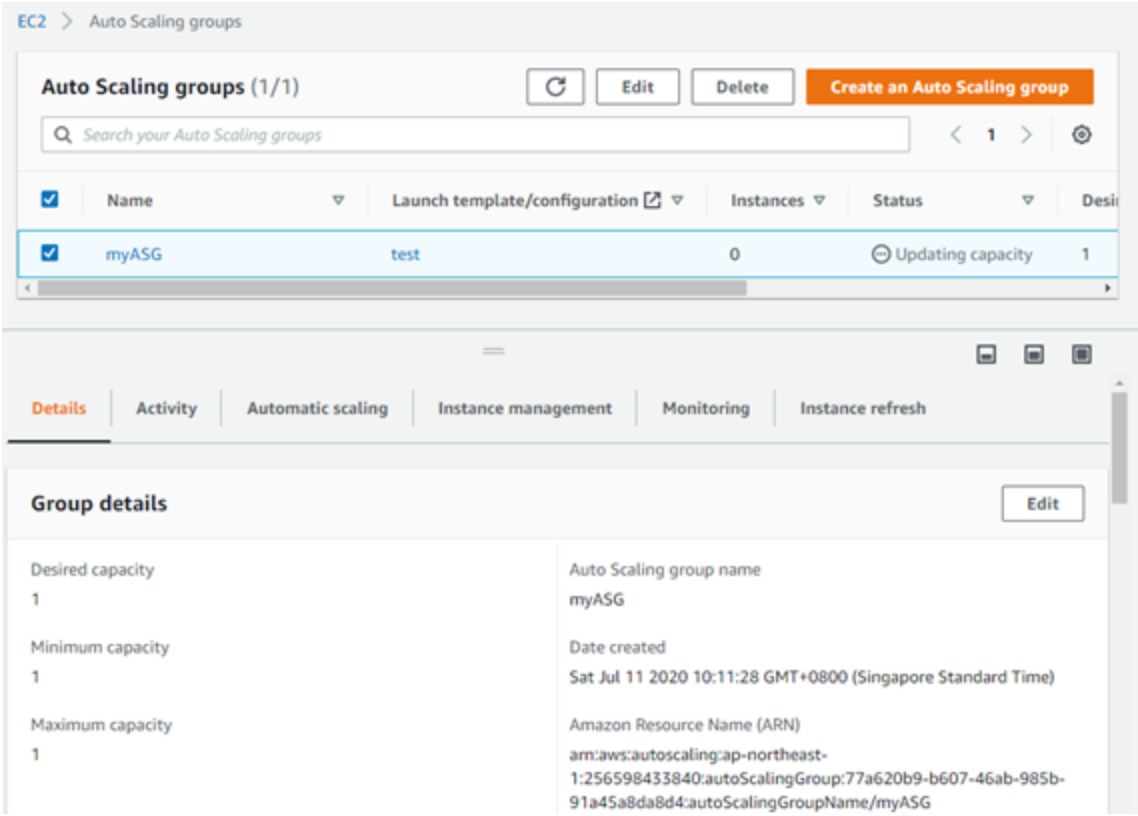
In this scenario, you will create an Auto Scaling Group using the EC2 AMI in the Launch configuration and set the size of the group to Min 1 and Max 1. This ensures that only instances of the application are running. Auto Scaling will perform health checks for the EC2 instances periodically. If the EC2 instance fails the health check, Auto Scaling will replace the instance.

Hence, it will always be available and self-healing. This makes your application fault-tolerant.

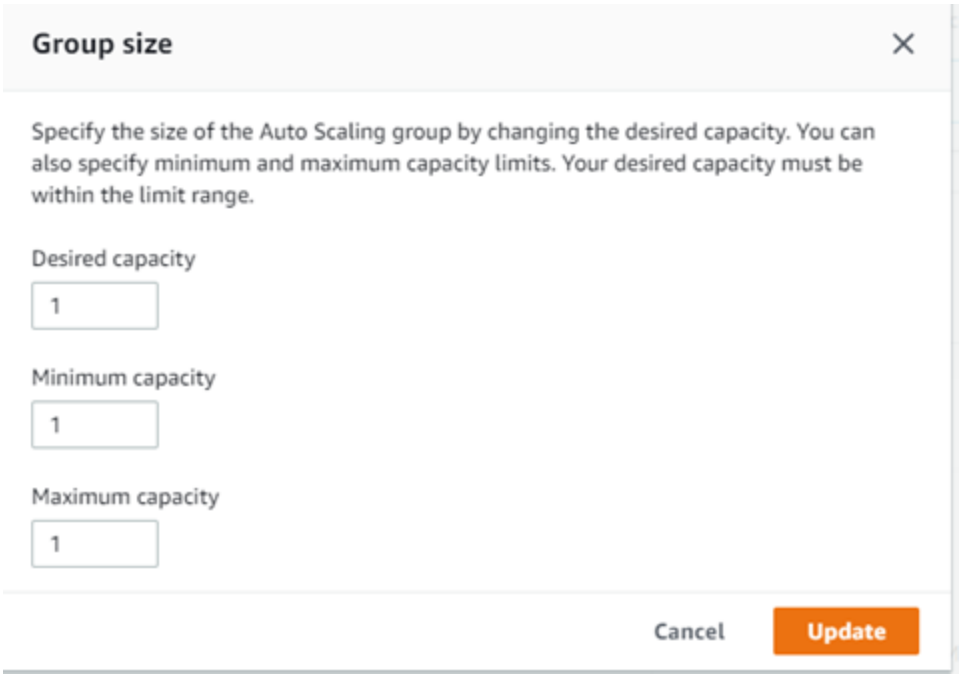
To set the MinSize and MaxSize of the Auto Scaling group:

1. Go to the EC2 Console page, on the left pane, choose Auto Scaling Groups.

2. Select the check box next to your Auto Scaling group. The bottom pane will show information on the selected Auto Scaling group.



3. On the Details tab, view or change the current settings for minimum, maximum, and desired capacity. Set the Desired capacity to 1, Minimum capacity to 1 and Maximum capacity to 1. Make sure that you don't have any Automatic Scaling policies configured for this group.



Sources:
https://docs.aws.amazon.com/autoscaling/ec2/userguide/as-maintain-instance-levels.html
https://docs.aws.amazon.com/autoscaling/ec2/userguide/asg-capacity-limits.html

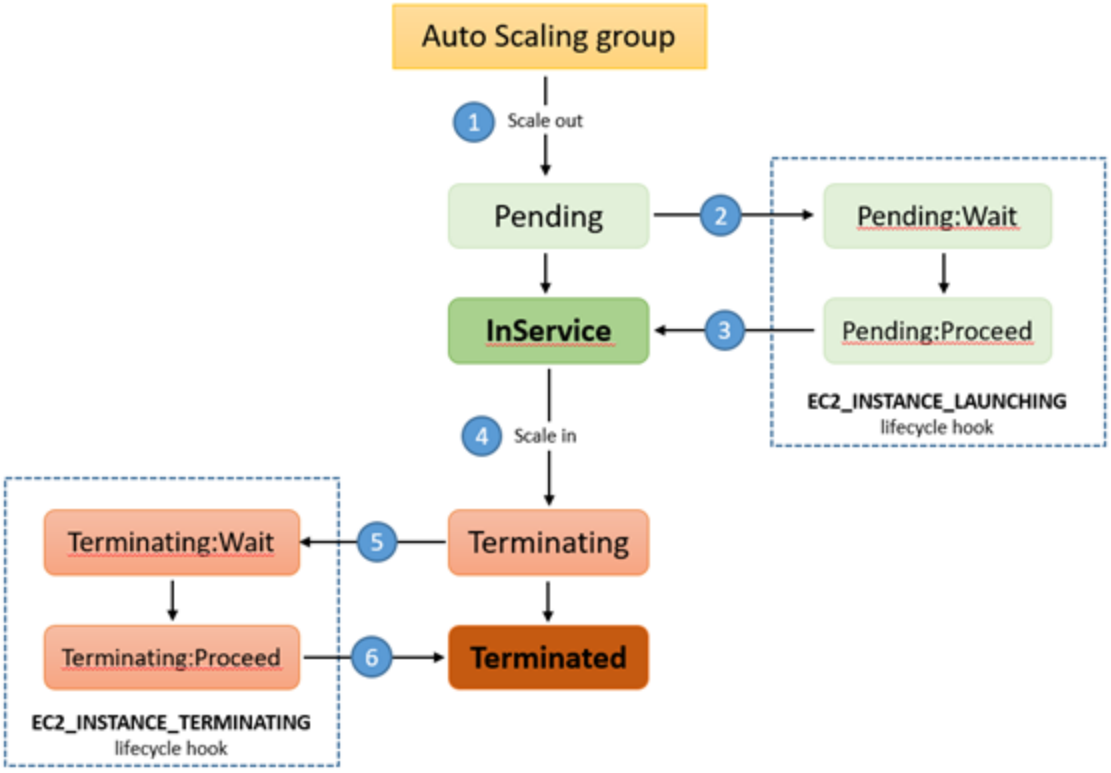
Auto Scaling Lifecycle Hooks

As your Auto Scaling group scales out or scales in your EC2 instances, you may want to perform custom actions before they start accepting traffic or before they get terminated. Auto Scaling Lifecycle Hooks allows you to perform custom actions during these stages. A lifecycle hook puts your EC2 instance into a wait state (`Pending:Wait` or `Terminating:Wait`) until your custom action has been performed or when the timeout period ends. The EC2 instance stays in a wait state for one hour by default, and then the Auto Scaling group resumes the launch or terminate process (`Pending:Proceed` or `Terminating:Proceed`).

For example, during the scale-out event of your ASG, you want to make sure that new EC2 instances download the latest code base from the repository and that your EC2 user data has completed before it starts accepting traffic. You can use the Pending:Wait hook. This way, the new instances will be fully ready and will quickly pass the load balancer health check when they are added as targets.

Another example: during the scale-in event of your ASG, suppose your instances upload data logs to S3 every minute. You may want to pause the instance termination for a certain amount of time to allow the EC2 to upload all data logs before it gets completely terminated.

The following diagram shows the transitions between the EC2 instance states with lifecycle hooks.



DevOps Exam Notes:

During the paused state (either launch or terminate), you can do more than just run custom scripts or wait for timeouts. CloudWatch Events receives the scaling action and you can define a CloudWatch Events Target to invoke a Lambda function that can perform custom actions, have it send a notification to your email via SNS, or trigger an SSM Run Command or SSM Automation to perform specific EC2 related tasks.

Sources:
https://docs.aws.amazon.com/autoscaling/ec2/userguide/lifecycle-hooks.html
https://docs.aws.amazon.com/cli/latest/reference/autoscaling/put-lifecycle-hook.html

AWS CHEAT SHEETS

AWS Compute Services

Amazon Elastic Compute Cloud (EC2)

- A Linux-based/Windows-based/Mac-based virtual server that you can provision.

Features

- Server environments called **instances**.
- Package OS and additional installations in a reusable template called **Amazon Machine Images**
- Secure login information for your instances using **key pairs**
- Storage volumes for temporary data that are deleted when you STOP or TERMINATE your instance, known as **instance store volumes.**
- Persistent storage volumes for your data using **Elastic Block Store volumes** (see aws storage services).
- Multiple physical locations for deploying your resources, such as instances and EBS volumes, known as **regions** and **Availability Zones** (see AWS overview)**.**
- A firewall that enables you to specify the protocols, ports, and source IP ranges that can reach your instances using **security groups** (see aws networking and content delivery).
- Static IPv4 addresses for dynamic cloud computing, known as **Elastic IP addresses** (see aws networking and content delivery).
- Metadata, known as **tags**, that you can create and assign to your EC2 resources
- Virtual networks you can create that are logically isolated from the rest of the AWS cloud, and that you can optionally connect to your own network, known as **virtual private clouds** or **VPC**s (see aws networking and content delivery).
- Add a script that will be run on instance boot called **user-data**.
- **Host Recovery for Amazon EC2** automatically restarts your instances on a new host in the event of an unexpected hardware failure on a Dedicated Host.

Instance states

- To prevent accidental termination, enable termination protection.

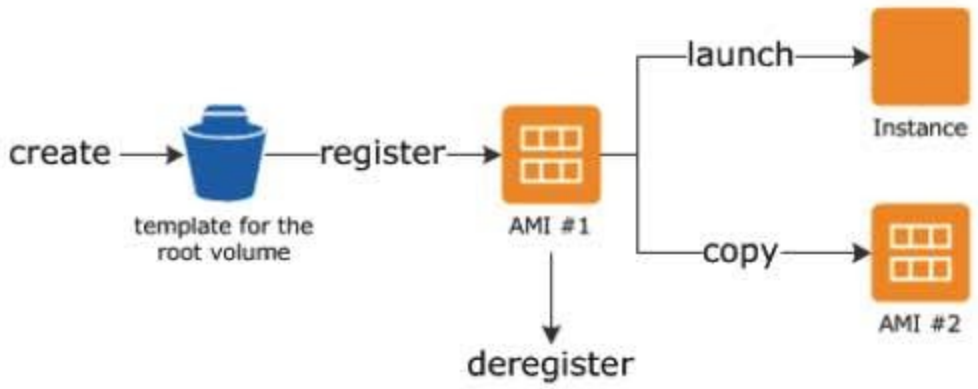
Root Device Volumes

- The root device volume contains the image used to boot the instance.
- Instance Store-backed Instances
    - Any data on the instance store volumes is deleted when the instance is terminated (instance store-backed instances do not support the Stop action) or if it fails (such as if an underlying drive has issues).

- Amazon EBS-backed Instances
  - An Amazon EBS-backed instance can be stopped and later restarted without affecting data stored in the attached volumes.
  - By default, the root device volume for an AMI backed by Amazon EBS is deleted when the instance terminates.

AMI

- Includes the following:
  - A template for the root volume for the instance (OS, application server, and applications)
  - Launch permissions that control which AWS accounts can use the AMI to launch instances
  - A block device mapping that specifies the volumes to attach to the instance when it's launched



- You can copy AMIs to different regions.

Pricing

- On-Demand - pay for the instances that you use by the second, with no long-term commitments or upfront payments.
- Reserved - make a low, one-time, up-front payment for an instance, reserve it for a *one*- or *three*-year term, and pay a significantly lower hourly rate for these instances.
- Spot - request unused EC2 instances, which can lower your costs significantly. Spot Instances are available at up to a 90% discount compared to On-Demand prices.

Security

- Use IAM to control access to your instances (see AWS Security and Identity Service).
  - IAM policies

    - 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**

Amazon Elastic Container Registry (ECR)

- A managed AWS Docker registry service. Stores your Docker images that you can use to deploy on your EC2, ECS, or Fargate deployments.

Features

- ECR supports Docker Registry HTTP API V2 allowing you to use Docker CLI commands or your preferred Docker tools in maintaining your existing development workflow.
- You can transfer your container images to and from Amazon ECR via HTTPS.

Components

- **Registry**
  - A registry is provided to each AWS account; you can create image repositories in your registry and store images in them.
  - The URL for your default registry is https://*aws_account_id*.dkr.ecr.*region*.amazonaws.com.
- **Authorization token**
  - Your Docker client needs to authenticate to ECR registries as an AWS user before it can push and pull images. The AWS CLI *get-login* command provides you with authentication credentials to pass to Docker.
- **Repository**
  - An image repository contains your Docker images.
  - **ECR lifecycle policies** enable you to specify the lifecycle management of images in a repository.
- **Repository policy**
  - You can control access to your repositories and the images within them with repository policies.
- **Image**
  - You can push and pull Docker images to your repositories. You can use these images locally on your development system, or you can use them in ECS task definitions.
  - You can replicate images in your private repositories across AWS regions.

Amazon Elastic Container Service (ECS)

- A container management service to run, stop, and manage Docker containers on a cluster.
- ECS can be used to create a consistent deployment and build experience, manage, and scale batch and **Extract-Transform-Load** (ETL) workloads, and build sophisticated application architectures on a microservices model.

Features

- You can create ECS clusters within a new or existing VPC.
- After a cluster is up and running, you can define task definitions and services that specify which Docker container images to run across your clusters.

Components

- Containers and Images
  - Your application components must be architected to run in **containers** — containing everything that your software application needs to run: code, runtime, system tools, system libraries, etc.
  - Containers are created from a read-only template called an **image**.
- Task Components
  - **Task definitions** specify various parameters for your application. It is a text file, in JSON format, that describes one or more containers, up to a maximum of ten, that form your application.
  - Task definitions are split into separate parts:
    - Task family - the name of the task, and each family can have multiple revisions.
    - IAM task role - specifies the permissions that containers in the task should have.
    - Network mode - determines how the networking is configured for your containers.
    - Container definitions - specify which image to use, how much CPU and memory the container are allocated, and many more options.
- Tasks and Scheduling
  - A **task** is the instantiation of a task definition within a cluster. After you have created a task definition for your application, you can specify the number of tasks that will run on your cluster.
    - Each task that uses the Fargate launch type has its own isolation boundary and does not share the underlying kernel, CPU resources, memory resources, or elastic network interface with another task.
  - You can upload a new version of your application task definition, and the ECS scheduler automatically starts new containers using the updated image and stop containers running the previous version.
- Clusters
  - When you run tasks using ECS, you place them in a **cluster**, which is a logical grouping of resources.
  - Clusters can contain tasks using both the Fargate and EC2 launch types.

    - 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

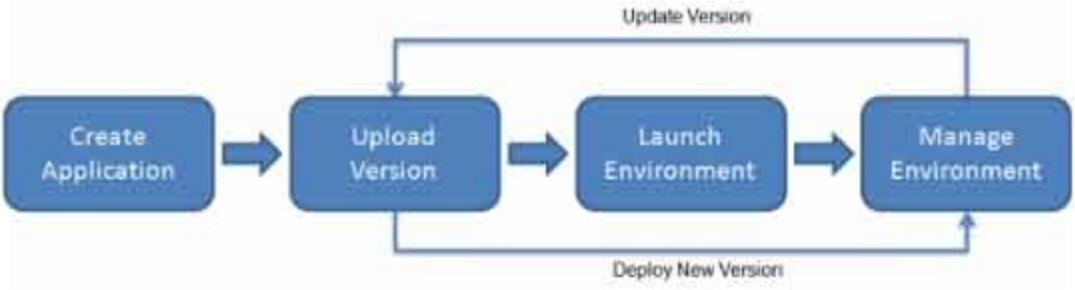
- Fargate task definitions require that the network mode is set to *awsvpc*. The *awsvpc* network mode provides each task with its own elastic network interface.
- Fargate task definitions only support the *awslogs* log driver for the log configuration. This configures your Fargate tasks to send log 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 Elastic Beanstalk

- Allows you to quickly deploy and manage applications in the AWS Cloud without worrying about the infrastructure that runs those applications.
- Elastic Beanstalk automatically handles the details of capacity provisioning, load balancing, scaling, and application health monitoring for your applications.
- Elastic Beanstalk supports Docker containers.
- Elastic Beanstalk Workflow



- Your application's domain name is in the format: *subdomain.region*.elasticbeanstalk.com

Elastic Beanstalk Concepts

- **Application** - a logical collection of Elastic Beanstalk components, including environments, versions, and environment configurations. It is conceptually similar to a folder.
- **Application Version** - refers to a specific, labeled iteration of deployable code for a web application. An application version points to an Amazon S3 object that contains the deployable code. Applications can have many versions and each application version is unique.
- **Environment** - a version that is deployed on to AWS resources. Each environment runs only a single application version at a time, however you can run the same version or different versions in many environments at the same time.
- **Environment Tier** - determines whether Elastic Beanstalk provisions resources to support an application that handles HTTP requests or an application that pulls tasks from a queue. An application that serves HTTP requests runs in a **web server environment**. An environment that pulls tasks from an Amazon SQS queue runs in a **worker environment**.
- **Environment Configuration** - identifies a collection of parameters and settings that define how an environment and its associated resources behave.
- **Configuration Template** - a starting point for creating unique environment configurations.
- There is a limit to the number of application versions you can have. You can avoid hitting the limit by applying an *application version lifecycle policy* to your applications to tell Elastic Beanstalk to delete

application versions that are old, or to delete application versions when the total number of versions for an application exceeds a specified number.

AWS Lambda

- A serverless compute service. Function-as-a-Service.
- Lambda executes your code only when needed and scales automatically.
- Lambda functions are stateless - no affinity to the underlying infrastructure.
- You choose the amount of memory you want to allocate to your functions and AWS Lambda allocates proportional CPU power, network bandwidth, and disk I/O.

Components of a Lambda Application

- **Function** – a script or program that runs in Lambda. Lambda passes invocation events to your function. The function processes an event and returns a response.
- **Runtimes** – Lambda runtimes allow functions in different languages to run in the same base execution environment. The runtime sits in-between the Lambda service and your function code, relaying invocation events, context information, and responses between the two.
- **Layers** – Lambda layers are a distribution mechanism for libraries, custom runtimes, and other function dependencies. Layers let you manage your in-development function code independently from the unchanging code and resources that it uses.
- **Event source** – an AWS service or a custom service that triggers your function and executes its logic.
- **Downstream resources** – an AWS service that your Lambda function calls once it is triggered.
- **Log streams** – While Lambda automatically monitors your function invocations and reports metrics to CloudWatch, you can annotate your function code with custom logging statements that allow you to analyze the execution flow and performance of your Lambda function.
- AWS Serverless Application Model

Lambda Functions

- You upload your application code in the form of one or more *Lambda functions*. Lambda stores code in Amazon S3 and encrypts it at rest.
- To create a Lambda function, you first package your code and dependencies in a deployment package. Then, you upload the deployment package to create your Lambda function.
- After your Lambda function is in production, Lambda automatically monitors functions on your behalf, reporting metrics through Amazon CloudWatch.
- Configure **basic function settings** including the description, memory usage, execution timeout, and role that the function will use to execute your code.
- **Environment variables** are always encrypted at rest, and can be encrypted in transit as well.
- **Versions and aliases** are secondary resources that you can create to manage function deployment and invocation.
- A **layer** is a ZIP archive that contains libraries, a custom runtime, or other dependencies. Use layers to manage your function's dependencies independently and keep your deployment package small.

AWS Serverless Application Model (SAM)

- An open-source framework for building serverless applications.
- It provides shorthand syntax to express functions, APIs, databases, and event source mappings.
- You create a **JSON** or **YAML** configuration template to model your applications.
- During deployment, SAM transforms and expands the SAM syntax into **AWS CloudFormation syntax**. Any resource that you can declare in an AWS CloudFormation template you can also declare in an AWS SAM template.
- The **SAM CLI** provides a Lambda-like execution environment that lets you locally build, test, and debug applications defined by SAM templates. You can also use the SAM CLI to deploy your applications to AWS.
- You can use AWS SAM to build serverless applications that use **any runtime supported by AWS Lambda**. You can also use SAM CLI to locally debug Lambda functions written in Node.js, Java, Python, and Go.
- Commonly used SAM CLI commands
    - The **sam init** command generates pre-configured AWS SAM templates.
    - The **sam local c**ommand supports local invocation and testing of your Lambda functions and SAM-based serverless applications by executing your function code locally in a Lambda-like execution environment.
    - The **sam package** and **sam deploy c**ommands let you bundle your application code and dependencies into a “deployment package" and then deploy your serverless application to the AWS Cloud.
    - The **sam logs** command enables you to fetch, tail, and filter logs for Lambda functions.
    - The output of the **sam publish** command includes a link to the AWS Serverless Application Repository directly to your application.
    - Use **sam validate** to validate your SAM template.

AWS Storage Services

Amazon EBS

- **Block level storage** volumes for use with EC2 instances.
- Well-suited for use as the primary storage for file systems, databases, or for any applications that require fine granular updates and access to raw, unformatted, block-level storage.
- Well-suited to both database-style applications (random reads and writes), and to throughput-intensive applications (long, continuous reads and writes).

Amazon EFS

- A fully-managed **file storage service** that makes it easy to set up and scale file storage in the Amazon Cloud.

Features

- The service manages all the file storage infrastructure for you, avoiding the complexity of deploying, patching, and maintaining complex file system configurations.
- EFS supports the Network File System version 4 protocol.
- Multiple Amazon EC2 instances can access an EFS file system at the same time, providing a common data source for workloads and applications running on more than one instance or server.
- Moving your EFS file data can be managed simply with AWS DataSync - a managed data transfer service that makes it faster and simpler to move data between on-premises storage and Amazon EFS.

Amazon S3

- S3 stores data as objects within **buckets**.
- An **object** consists of a file and optionally any metadata that describes that file.
- A **key** is the unique identifier for an object within a bucket.
- Storage capacity is virtually unlimited.
- Good for storing static web content or media. Can be used to host static websites.

Buckets

- For each bucket, you can:
    - Control access to it (create, delete, and list objects in the bucket)

    - View access logs for it and its objects
    - Choose the geographical region where to store the bucket and its contents.
- **Bucket name** must be a unique DNS-compliant name.
    - The name must be unique across all existing bucket names in Amazon S3.
    - After you create the bucket you cannot change the name.
    - The bucket name is visible in the URL that points to the objects that you're going to put in your bucket.
- You can host static websites by configuring your bucket for website hosting.

Security

- Policies contain the following:
    - **Resources** – buckets and objects
    - **Actions** – set of operations
    - **Effect** – can be either allow or deny. Need to explicitly grant allow to a resource.
    - **Principal** – the account, service or user who is allowed access to the actions and resources in the statement.
- Resource Based Policies
    - Bucket Policies
        - Provides **centralized access control** to buckets and objects based on a variety of conditions, including S3 operations, requesters, resources, and aspects of the request (e.g., IP address).
        - Can either **add or deny permissions** across all (or a subset) of objects within a bucket.
        - IAM users need additional permissions from root account to perform bucket operations.
        - Bucket policies are limited to 20 KB in size.
    - User Policies
    - AWS IAM (see AWS Security and Identity Services)
        - IAM User Access Keys
        - Temporary Security Credentials
- Versioning
    - Use versioning to keep multiple versions of an object in one bucket.
    - Versioning protects you from the consequences of unintended overwrites and deletions.
    - You can also use versioning to archive objects so you have access to previous versions.
    - Since versioning is disabled by default, you need to EXPLICITLY enable it.
- Encryption
    - Server-side Encryption using
        - **Amazon S3-Managed Keys (SSE-S3)**
        - **AWS KMS-Managed Keys (SSE-KMS)**
        - **Customer-Provided Keys (SSE-C)**
    - Client-side Encryption using
        - AWS KMS-managed customer master key

        - client-side master key
- MFA Delete
    - MFA delete grants additional authentication for either of the following operations:
        - Change the versioning state of your bucket
        - Permanently delete an object version
- Cross-Account Access
    - You can provide another AWS account access to an object that is stored in an Amazon Simple Storage Service (Amazon S3) bucket. These are the methods on how to grant cross-account access to objects that are stored in your own Amazon S3 bucket:
- Monitoring
    - Automated monitoring tools to watch S3:
        - Amazon 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.
        - AWS CloudTrail Log Monitoring – Share log files between accounts, monitor CloudTrail log files in real time by sending them to CloudWatch Logs, write log processing applications in Java, and validate that your log files have not changed after delivery by CloudTrail.
    - Only certain S3 events are recorded on CloudTrail Event History. For Object level events you **enable server access logging for an S3 bucket**
- S3 Events Notification
    - To enable notifications, add a *notification configuration* identifying the events to be published, and the destinations where to send the event notifications.
    - Can publish following events:
        - A new object created event
        - An object removal event
        - A Reduced Redundancy Storage (RRS) object lost event
    - Supports the following destinations for your events:
        - Amazon Simple Notification Service (Amazon SNS) topic
        - Amazon Simple Queue Service (Amazon SQS) queue
        - AWS Lambda

Amazon S3 Bucket Policies for VPC Endpoints

What are VPC endpoints?

A VPC endpoint is what you use to privately connect your VPC to supported AWS services, such as Amazon S3. It adds a gateway entry in your VPC's route table so that communication between your AWS resources, such as Amazon EC2 instances, and your S3 bucket pass through the gateway instead of the public internet. As a result, VPC endpoint is a regional service. You should create the endpoint in the same region as the VPC you want to link it to.

VPC endpoints are best used when you have compliance requirements or sensitive information stored in S3 that should not leave the Amazon network. A VPC endpoint is also a better option for private network connections in AWS, as compared to using a VPN solution or a NAT solution since it is easier to setup and offers you more network bandwidth at your disposal.

AWS Database Services

Amazon Aurora

- A fully managed relational database engine that's compatible with MySQL and PostgreSQL.
- With some workloads, Aurora can deliver up to five times the throughput of MySQL and up to three times the throughput of PostgreSQL.
- Aurora includes a high-performance storage subsystem. The underlying storage grows automatically as needed, up to 64 terabytes. The minimum storage is 10GB.
- **DB Clusters**
    - An Aurora DB cluster consists of one or more DB instances and a cluster volume that manages the data for those DB instances.
    - An Aurora cluster volume is a virtual database storage volume that spans multiple AZs, with each AZ having a copy of the DB cluster data.
    - Cluster Types:
        - Primary DB instance – Supports read and write operations, and performs all of the data modifications to the cluster volume. Each Aurora DB cluster has one primary DB instance.
        - Aurora Replica – Connects to the same storage volume as the primary DB instance and supports only read operations. Each Aurora DB cluster can have up to 15 Aurora Replicas in addition to the primary DB instance. Aurora automatically fails over to an Aurora Replica in case the primary DB instance becomes unavailable. You can specify the failover priority for Aurora Replicas. Aurora Replicas can also offload read workloads from the primary DB instance.
- **Aurora Multi Master**
    - The feature is available on Aurora MySQL 5.6
    - Allows you to create multiple read-write instances of your Aurora database across multiple Availability Zones, which enables uptime-sensitive applications to achieve continuous write availability through instance failure.
    - In the event of instance or Availability Zone failures, Aurora Multi-Master enables the Aurora database to maintain read and write availability with zero application downtime. There is no need for database failovers to resume write operations.
- **Monitoring**
    - Subscribe to Amazon RDS events to be notified when changes occur with a DB instance, DB cluster, DB cluster snapshot, DB parameter group, or DB security group.
    - Database log files
    - RDS Enhanced Monitoring — Look at metrics in real time for the operating system.
    - RDS Performance Insights monitors your Amazon RDS DB instance load so that you can analyze and troubleshoot your database performance.
    - Use CloudWatch Metrics, Alarms and Logs

Amazon DynamoDB

- NoSQL database service that provides fast and predictable performance with seamless scalability.
- Offers encryption at rest.
- You can create database tables that can store and retrieve any amount of data, and serve any level of request traffic.
- You can scale up or scale down your tables' throughput capacity without downtime or performance degradation, and use the AWS Management Console to monitor resource utilization and performance metrics.

Core Components

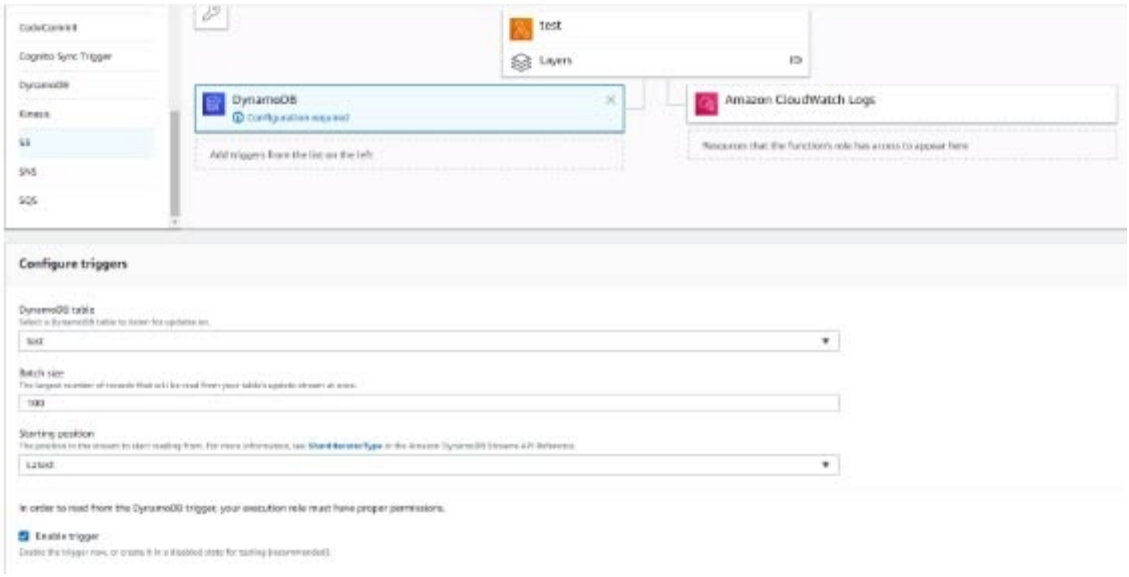
- **Tables** - a collection of items
  - DynamoDB stores data in a table, which is a collection of data.
  - Are schemaless.
  - There is an initial limit of 256 tables per region.
- **Items** - a collection of attributes
  - DynamoDB uses **primary keys** to uniquely identify each item in a table and **secondary indexes** to provide more querying flexibility.
  - Each table contains zero or more items.
- **Attributes** - a fundamental data element
  - DynamoDB supports nested attributes up to 32 levels deep.
- **Primary Key** - uniquely identifies each item in the table, so that no two items can have the same key. Must be scalar.
  - **Partition key** - a simple primary key, composed of one attribute.
  - **Partition key and sort key** (*composite primary key*) - composed of two attributes.
  - DynamoDB uses the partition key value as input to an internal hash function. The output from the hash function determines the partition in which the item will be stored. All items with the same partition key are stored together, in sorted order by sort key value. If no sort key is used, no two items can have the same partition key value.
- **Secondary Indexes** - lets you query the data in the table using an alternate key, in addition to queries against the primary key.
  - You can create one or more secondary indexes on a table.
  - Two kinds of indexes:
    - **Global secondary index** – An index with a partition key and sort key that can be different from those on the table.
    - **Local secondary index** – An index that has the same partition key as the table, but a different sort key.
  - You can define up to 20 global secondary indexes and 5 local secondary indexes per table.

DynamoDB Accelerator (DAX)

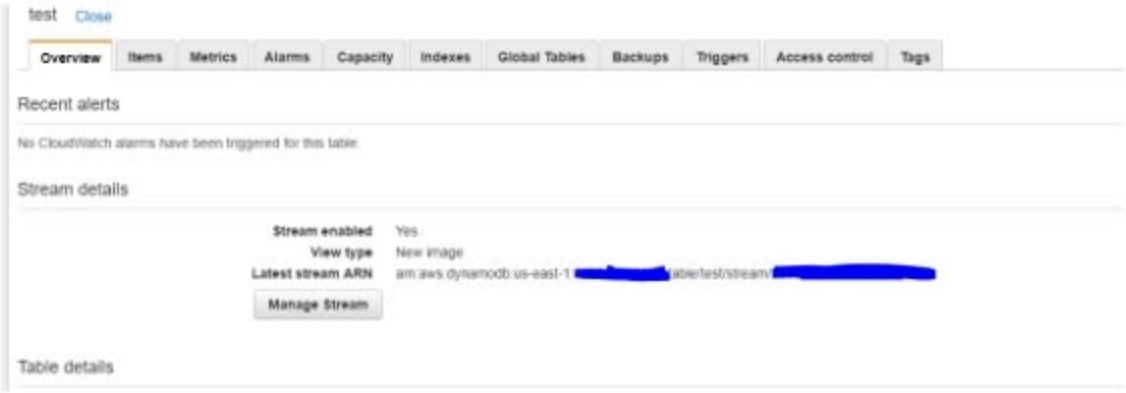
- DAX is a fully managed, highly available, in-memory cache for DynamoDB.
- **DynamoDB Accelerator (DAX)** delivers microsecond response times for accessing eventually consistent data.
- It requires only minimal functional changes to use DAX with an existing application since it is API-compatible with DynamoDB.
- For read-heavy or bursty workloads, DAX provides increased throughput and potential cost savings by reducing the need to overprovision read capacity units.
- DAX lets you scale on-demand.
- DAX is fully managed. You no longer need to do hardware or software provisioning, setup and configuration, software patching, operating a reliable, distributed cache cluster, or replicating data over multiple instances as you scale.

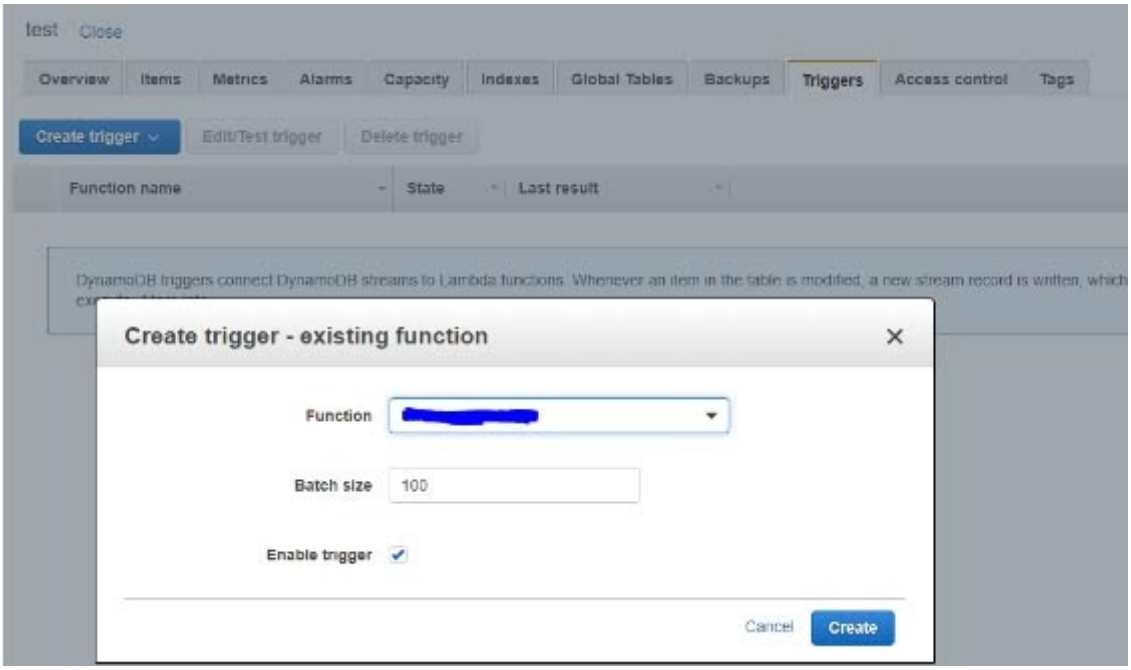
Lambda Integration With Amazon DynamoDB Streams

- Amazon DynamoDB is integrated with AWS Lambda so that you can create triggers, which are pieces of code that automatically respond to events in DynamoDB Streams. With triggers, you can build applications that react to data modifications in DynamoDB tables.

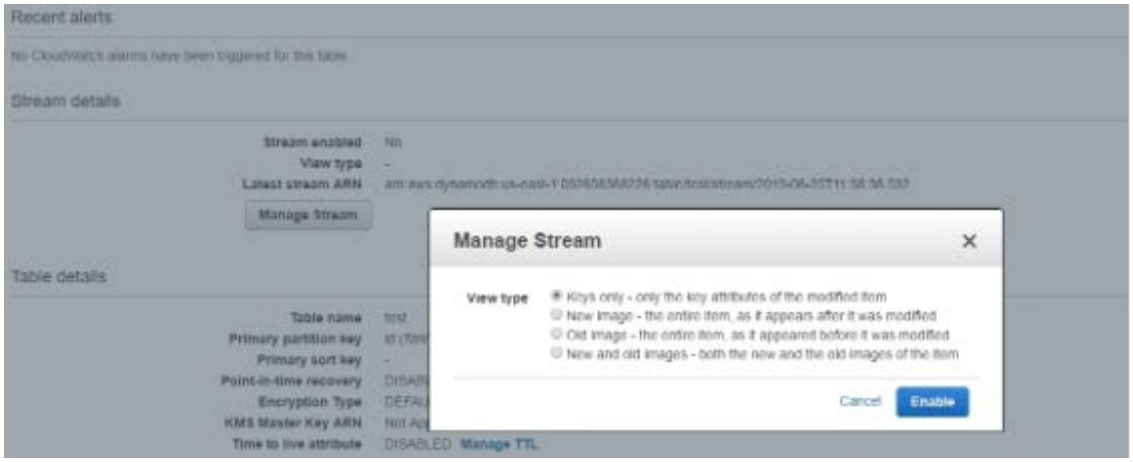


- After you enable DynamoDB Streams on a table, associate the DynamoDB table with a Lambda function. AWS Lambda polls the stream and invokes your Lambda function synchronously when it detects new stream records.





- Configure the StreamSpecification you want for your DynamoDB Streams:
    - StreamEnabled (Boolean) - indicates whether DynamoDB Streams is enabled (true) or disabled (false) on the table.
    - StreamViewType (string) - when an item in the table is modified, StreamViewType determines what information is written to the stream for this table. Valid values for StreamViewType are:
        - KEYS_ONLY - Only the key attributes of the modified items are written to the stream.
        - NEW_IMAGE - The entire item, as it appears after it was modified, is written to the stream.
        - OLD_IMAGE - The entire item, as it appeared before it was modified, is written to the stream.
        - NEW_AND_OLD_IMAGES - Both the new and the old item images of the items are written to the stream.



Sources:
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Streams.Lambda.html
https://docs.aws.amazon.com/amazondynamodb/latest/APIReference/API_StreamSpecification.html

Amazon RDS

- Supports **Aurora**, **MySQL, MariaDB, PostgreSQL, Oracle, Microsoft SQL Server**.
- You can get high availability with a primary instance and a synchronous secondary instance that you can fail over to when problems occur. You can also use MySQL, MariaDB, or PostgreSQL Read Replicas to increase read scaling.
- You can select the computation and memory capacity of a DB instance, determined by its **DB instance class**. If your needs change over time, you can change DB instances.
- Each DB instance has minimum and maximum storage requirements depending on the storage type and the database engine it supports.
- You can run your DB instance in several AZs, an option called a **Multi-AZ deployment**. Amazon automatically provisions and maintains a secondary standby DB instance in a different AZ. Your primary DB instance is synchronously replicated across AZs to the secondary instance to provide data redundancy, failover support, eliminate I/O freezes, and minimize latency spikes during system backups.

Security

- Security Groups
    - **DB Security Groups** - controls access to a DB instance that is not in a VPC. By default, network access is turned off to a DB instance. This SG is for the EC2-Classic platform.
    - **VPC Security Groups** - controls access to a DB instance inside a VPC. This SG is for the EC2-VPC platform.
    - **EC2 Security Groups** - controls access to an EC2 instance and can be used with a DB instance.
- A *resource owner* is the AWS account that created a resource. That is, the resource owner is the AWS account of the *principal entity* (the root account, an IAM user, or an IAM role) that authenticates the request that creates the resource.
- A *permissions policy* describes who has access to what. Policies attached to an IAM identity are *identity-based policies* (IAM policies) and policies attached to a resource are *resource-based policies*. Amazon RDS supports only identity-based policies (IAM policies).
- MySQL and PostgreSQL both support **IAM database authentication**.

High Availability using Multi-AZ

- Multi-AZ deployments for **Oracle, PostgreSQL, MySQL, and MariaDB** DB instances use **Amazon's failover technology**. **SQL Server DB** instances use **SQL Server Mirroring**.
- The primary DB instance switches over automatically to the standby replica if any of the following conditions occur:
    - An Availability Zone outage
    - The primary DB instance fails
    - The DB instance's server type is changed

  - The operating system of the DB instance is undergoing software patching
  - A manual failover of the DB instance was initiated using **Reboot with failover**

Read Replicas

- Updates made to the source DB instance are asynchronously copied to the Read Replica.
- You can reduce the load on your source DB instance by routing read queries from your applications to the Read Replica.
- You can elastically scale out beyond the capacity constraints of a single DB instance for read-heavy database workloads.
- You can create a Read Replica that has a different storage type from the source DB instance.

Backups and Restores

- Your DB instance must be in the **ACTIVE state** for automated backups to occur. Automated backups and automated snapshots don't occur while a copy is executing in the same region for the same DB instance.
- The first snapshot of a DB instance contains the data for the full DB instance. Subsequent snapshots of the same DB instance are incremental.
- The default backup retention period is one day if you create the DB instance using the RDS API or the AWS CLI, or seven days if you used the AWS Console.
- Manual snapshot limits are limited to 100 per region.
- You can copy a snapshot within the same AWS Region, you can copy a snapshot across AWS Regions, and you can copy a snapshot across AWS accounts.
- When you restore a DB instance to a point in time, the default DB parameter and default DB security group is applied to the new DB instance.

AWS Networking & Content Delivery

Amazon API Gateway

- Enables developers to create, publish, maintain, monitor, and secure APIs at any scale.
- Allows creating, deploying, and managing a RESTful API to expose backend HTTP endpoints, Lambda functions, or other AWS services.
- Together with Lambda, API Gateway forms the app-facing part of the AWS serverless infrastructure.
- **Concepts**
    - API deployment - a point-in-time snapshot of your API Gateway API resources and methods. To be available for clients to use, the deployment must be associated with one or more API stages.
    - API endpoints - host names APIs in API Gateway, which are deployed to a specific region and of the format: rest-api-id.execute-api.region.amazonaws.com
    - API key - An alphanumeric string that API Gateway uses to identify an app developer who uses your API.
    - API stage - A logical reference to a lifecycle state of your API. API stages are identified by API ID and stage name.
    - Model - Data schema specifying the data structure of a request or response payload.
    - Private API - An API that is exposed through interface VPC endpoints and isolated from the public internet
    - Private integration - An API Gateway integration type for a client to access resources inside a customer's VPC through a private API endpoint without exposing the resources to the public internet.
    - Proxy integration - You can set up a proxy integration as an HTTP proxy integration type or a Lambda proxy integration type.
        - For the HTTP proxy integration, API Gateway passes the entire request and response between the frontend and an HTTP backend.
        - For the Lambda proxy integration, API Gateway sends the entire request as an input to a backend Lambda function.
    - Usage plan - Provides selected API clients with access to one or more deployed APIs. You can use a usage plan to configure throttling and quota limits, which are enforced on individual client API keys.
- **Features**
    - API Gateway can execute Lambda code in your account, start Step Functions state machines, or make calls to Elastic Beanstalk, EC2, or web services outside of AWS with publicly accessible HTTP endpoints.
    - API Gateway helps you define plans that meter and restrict third-party developer access to your APIs.
    - API Gateway helps you manage traffic to your backend systems by allowing you to set throttling rules based on the number of requests per second for each HTTP method in your APIs.

    - You can set up a cache with customizable keys and time-to-live in seconds for your API data to avoid hitting your backend services for each request.
    - API Gateway lets you run multiple versions of the same API simultaneously with API Lifecycle.
    - After you build, test, and deploy your APIs, you can package them in an API Gateway usage plan and sell the plan as a Software as a Service (SaaS) product through AWS Marketplace.
    - API Gateway offers the ability to create, update, and delete documentation associated with each portion of your API, such as methods and resources.
    - Amazon API Gateway offers general availability of HTTP APIs, which gives you the ability to route requests to private ELBs and IP-based services registered in AWS CloudMap such as ECS tasks. Previously, HTTP APIs enabled customers to only build APIs for their serverless applications or to proxy requests to HTTP endpoints.
- All of the APIs created expose HTTPS endpoints only. API Gateway does not support unencrypted (HTTP) endpoints.

Amazon Route 53

- A highly available and scalable Domain Name System (DNS) web service used for domain registration, DNS routing, and health checking.

Routing Internet Traffic to your Website or Web Application

- Use the Route 53 console to register a domain name and configure Route 53 to route internet traffic to your website or web application.
- After you register your domain name, Route 53 automatically creates a **public hosted zone** that has the same name as the domain.
- To route traffic to your resources, you create **records**, also known as *resource record sets*, in your hosted zone.
- You can create special Route 53 records, called **alias records**, that route traffic to S3 buckets, CloudFront distributions, and other AWS resources.
- Each record includes information about how you want to route traffic for your domain, such as:
    - Name - name of the record corresponds with the domain name or subdomain name that you want Route 53 to route traffic for.
    - Type - determines the type of resource that you want traffic to be routed to.
    - Value

Route 53 Health Checks

    - Create a health check and specify values that define how you want the health check to work, such as:
        - The IP address or domain name of the endpoint that you want Route 53 to monitor.
        - The protocol that you want Route 53 to use to perform the check: HTTP, HTTPS, or TCP.
        - The **request interval** you want Route 53 to send a request to the endpoint.
        - How many consecutive times the endpoint must fail to respond to requests before Route 53 considers it unhealthy. This is the **failure threshold**.
    - You can configure a health check to check the health of one or more other health checks.
    - You can configure a health check to check the status of a CloudWatch alarm so that you can be notified on the basis of a broad range of criteria.

Know the following Concepts

- Domain Registration Concepts - domain name, domain registrar, domain registry, domain reseller, top-level domain
- DNS Concepts
    - **Alias record** - a type of record that you can create to route traffic to AWS resources.

    - **Hosted zone** - a container for records, which includes information about how to route traffic for a domain and all of its subdomains.
    - **Name servers** - servers in the DNS that help to translate domain names into the IP addresses that computers use to communicate with one another.
    - **Record** (DNS record) - an object in a hosted zone that you use to define how you want to route traffic for the domain or a subdomain.
    - **Routing policy -** policy on how to redirect users based on configured routing policy
    - **Subdomain** - name below the zone apex. Example: portal.tutorialsdojo.com
    - Time to live (TTL) - time that the DNS record is cached by querying servers.
- Health Checking Concepts
    - **DNS failover** - a method for routing traffic away from unhealthy resources and to healthy resources.
    - Endpoint - the URL or endpoint on which the health check will be performed.
    - Health check - the metric on which to determine if an endpoint is healthy or not.

Records

- Create records in a hosted zone. Records define where you want to route traffic for each domain name or subdomain name. The name of each record in a hosted zone must end with the name of the hosted zone.
- Alias Records
    - Route 53 **alias records** provide a Route 53–specific extension to DNS functionality. Alias records let you route traffic to selected AWS resources. They also let you route traffic from one record in a hosted zone to another record.
    - You can create an alias record at the top node of a DNS namespace, also known as the zone apex.
- CNAME Record
    - You cannot create an alias record at the top node (zone apex) of a DNS namespace using a CNAME record.

AWS Elastic Load Balancing (ELB)

- Distributes incoming application or network traffic across multiple targets, such as **EC2 instances**, **containers (ECS)**, **Lambda functions,** and **IP addresses**, in multiple Availability Zones.
- When you create a load balancer, you must specify one public subnet from at least two Availability Zones. You can specify only one public subnet per Availability Zone.

General features

- Accepts incoming traffic from clients and routes requests to its registered targets.
- Monitors the health of its registered targets and routes traffic only to healthy targets.
- **Cross Zone Load Balancing** - when enabled, each load balancer node distributes traffic across the registered targets in all enabled AZs.

Three Types of Load Balancers

- **Application Load Balancer**
- **Network Load Balancer**
- **Classic load Balancer**
- **Slow Start Mode** gives targets time to warm up before the load balancer sends them a full share of requests.
- **Sticky sessions** route requests to the same target in a target group. You enable sticky sessions at the target group level. You can also set the duration for the stickiness of the load balancer-generated cookie, in seconds. Useful if you have stateful applications.
- **Health checks** verify the status of your targets. The statuses for a registered target are:

AWS Security & Identity Services

Amazon GuardDuty

- An intelligent threat detection service. It analyzes billions of events across your AWS accounts from AWS CloudTrail (AWS user and API activity in your accounts), Amazon VPC Flow Logs (network traffic data), and DNS Logs (name query patterns).
- GuardDuty is a regional service.
- Threat detection categories
    - **Reconnaissance** -- Activity suggesting reconnaissance by an attacker, such as unusual API activity, intra-VPC port scanning, unusual patterns of failed login requests, or unblocked port probing from a known bad IP.
    - **Instance compromise** -- Activity indicating an instance compromise, such as cryptocurrency mining, backdoor command and control activity, malware using domain generation algorithms, outbound denial of service activity, unusually high volume of network traffic, unusual network protocols, outbound instance communication with a known malicious IP, temporary Amazon EC2 credentials used by an external IP address, and data exfiltration using DNS.
    - **Account compromise** -- Common patterns indicative of account compromise include API calls from an unusual geolocation or anonymizing proxy, attempts to disable AWS CloudTrail logging, changes that weaken the account password policy, unusual instance or infrastructure launches, infrastructure deployments in an unusual region, and API calls from known malicious IP addresses.
- Amazon GuardDuty provides three severity levels (Low, Medium, and High) to allow you to prioritize response to potential threats.
- CloudTrail Event Source
    - Currently, GuardDuty only analyzes CloudTrail management events. (Read about types of CloudTrail trails for more information)
    - GuardDuty processes all CloudTrail events that come into a region, including global events that CloudTrail sends to all regions, such as AWS IAM, AWS STS, Amazon CloudFront, and Route 53.
- VPC Flow Logs Event Source
    - VPC Flow Logs capture information about the IP traffic going to and from Amazon EC2 network interfaces in your VPC.
- DNS Logs Event Source
    - If you use AWS DNS resolvers for your EC2 instances (the default setting), then GuardDuty can access and process your request and response DNS logs through the internal AWS DNS resolvers. Using other DNS resolvers will not provide GuardDuty access to its DNS logs.
- GuardDuty vs Macie
    - Amazon GuardDuty provides broad protection of your AWS accounts, workloads, and data by helping to identify threats such as attacker reconnaissance, instance compromise, and account compromise. Amazon Macie helps you protect your data in Amazon S3 by helping you classify

what data you have, the value that data has to the business, and the behavior associated with access to that data.

Amazon Inspector

- An automated security assessment service that helps you test the network accessibility of your EC2 instances and the security state of your applications running on the instances.
- Inspector uses IAM *service-linked roles*.

Features

- Inspector provides an engine that analyzes system and resource configuration and monitors activity to determine what an assessment target looks like, how it behaves, and its dependent components. The combination of this telemetry provides a complete picture of the assessment target and its potential security or compliance issues.
- Inspector incorporates a built-in library of rules and reports. These include checks against best practices, common compliance standards and vulnerabilities.
- Automate security vulnerability assessments throughout your development and deployment pipeline or against static production systems.
- Inspector is an API-driven service that uses an optional *agent*, making it easy to deploy, manage, and automate.

Concepts

- **Inspector Agent** - A software agent that you can install on all EC2 instances that are included in the assessment target, the security of which you want to evaluate with Inspector.
- **Assessment run** - The process of discovering potential security issues through the analysis of your assessment target's configuration and behavior against specified rules packages.
- **Assessment target** - A collection of AWS resources that work together as a unit to help you accomplish your business goals. Inspector assessment targets can consist only of EC2 instances.
- **Assessment template** - A configuration that is used during your assessment run, which includes
    - Rules packages against which you want Inspector to evaluate your assessment target,
    - The duration of the assessment run,
    - Amazon SNS topics to which you want Inspector to send notifications about assessment run states and findings,
    - Inspector-specific attributes (key-value pairs) that you can assign to findings generated by the assessment run that uses this assessment template.
    - After you create an assessment template, you can't modify it.
- **Finding** - A potential security issue discovered during the assessment run of the specified target.
- **Rule** - A security check performed during an assessment run. When a rule detects a potential security issue, Inspector generates a finding that describes the issue.
- **Rules package** - A collection of rules that corresponds to a security goal that you might have.
- **Telemetry** - EC2 instance data collected by Inspector during an assessment run and passed to the Inspector service for analysis.

- The telemetry data generated by the Inspector Agent during assessment runs is formatted in JSON files and delivered in near-real-time over TLS to Inspector, where it is encrypted with a per-assessment-run, ephemeral KMS-derived key and securely stored in an S3 bucket dedicated for the service.

Assessment Reports

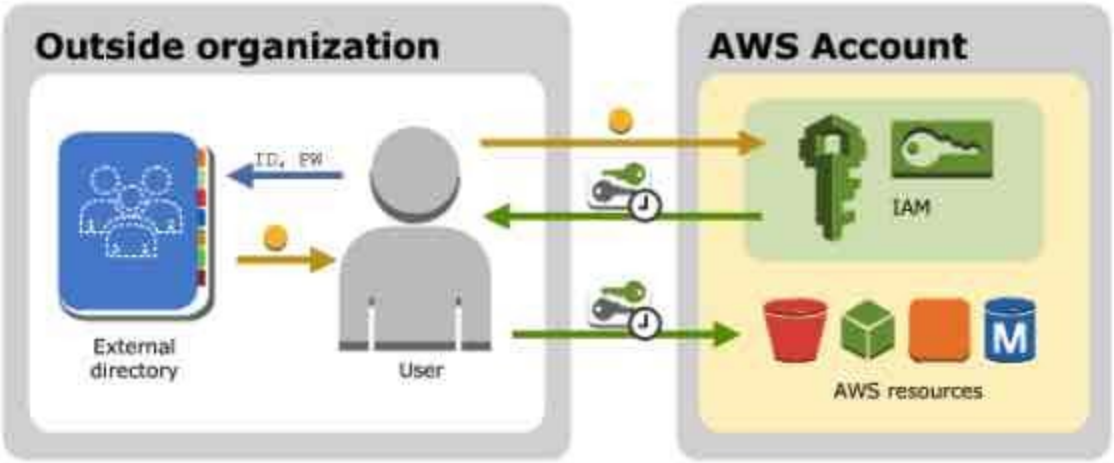
- A document that details what is tested in the assessment run, and the results of the assessment.
- You can view the following types of assessment reports:
    - **Findings report** - this report contains the following information:
        - Executive summary of the assessment
        - EC2 instances evaluated during the assessment run
        - Rules packages included in the assessment run
        - Detailed information about each finding, including all EC2 instances that had the finding
    - **Full report** - this report contains all the information that is included in a findings report, and additionally provides the list of rules that passed on all instances in the assessment target.

Amazon Macie

- A security service that uses machine learning to automatically discover, classify, and protect sensitive data in AWS. Macie recognizes sensitive data such as personally identifiable information (PII) or intellectual property.
- Amazon Macie allows you to achieve the following:
    - Identify and protect various data types, including PII, PHI, regulatory documents, API keys, and secret keys
    - Verify compliance with automated logs that allow for instant auditing
    - Identify changes to policies and access control lists
    - Observe changes in user behavior and receive actionable alerts
    - Receive notifications when data and account credentials leave protected zones
    - Detect when large quantities of business-critical documents are shared internally and externally
- **Concepts**
    - An **Alert** is a notification about a potential security issue that Macie discovers. Alerts appear on the Macie console and provide a comprehensive narrative about all activity that occurred over the last 24 hours.
        - Basic alerts – Alerts that are generated by the security checks that Macie performs. There are two types of basic alerts in Macie:
            - Managed (curated by Macie) basic alerts that you can't modify. You can only enable or disable the existing managed basic alerts.
            - Custom basic alerts that you can create and modify to your exact specifications.
        - Predictive alerts – Automatic alerts based on activity in your AWS infrastructure that deviates from the established normal activity baseline. More specifically, Macie continuously monitors IAM user and role activity in your AWS infrastructure and builds a model of the normal behavior. It then looks for deviations from that normal baseline, and when it detects such activity, it generates automatic predictive alerts.
    - **Data source** is the origin or location of a set of data.
        - AWS CloudTrail event logs and errors, including Amazon S3 object-level API activity. You can't modify existing or add new CloudTrail events to the list that Macie manages. You can enable or disable the supported CloudTrail events, thus instructing Macie to either include or exclude them in its data security process.
        - Amazon S3 objects. You can integrate Macie with your S3 buckets and/or specify S3 prefixes
    - **User**, in the context of Macie, a user is the AWS Identity and Access Management (IAM) identity that makes the request.

AWS Identity & Access Management (IAM)

- Control who is authenticated (signed in) and authorized (has permissions) to use resources.
- AWS account root user is a single sign-in identity that has complete access to all AWS services and resources in the account.
- **Features**
  - You can grant other people permission to administer and use resources in your AWS account without having to share your password or access key.
  - You can grant different permissions to different people for different resources.
  - You can use IAM features to securely provide credentials for applications that run on EC2 instances which provide permissions for your applications to access other AWS resources.
  - You can add two-factor authentication to your account and to individual users for extra security.
  - You can allow users to use identity federation to get temporary access to your AWS account.
  - You receive AWS CloudTrail log records that include information about IAM identities who made requests for resources in your account.
  - You use an access key (an access key ID and secret access key) to make programmatic requests to AWS. An Access Key ID and Secret Access Key can only be uniquely generated once and must be regenerated if lost.
  - You can generate and download a credential report that lists all users on your AWS account. The report also shows the status of passwords, access keys, and MFA devices.
- **Users**
  - IAM Users
    - Instead of sharing your root user credentials with others, you can create individual IAM users within your account that correspond to users in your organization. IAM users are not separate accounts; they are users within your account.
    - Each user can have its own password for access to the AWS Management Console. You can also create an individual access key for each user so that the user can make programmatic requests to work with resources in your account.
    - By default, a brand new IAM user has NO permissions to do anything.
    - Users are global entities.
  - Federated Users
    - If the users in your organization already have a way to be authenticated, you can federate those user identities into AWS.



- IAM Groups
  - An IAM group is a collection of IAM users.
  - You can organize IAM users into IAM groups and attach access control policies to a group.
  - A user can belong to multiple groups.
  - Groups cannot belong to other groups.
  - Groups do not have security credentials, and cannot access web services directly.
- IAM Role
  - A role does not have any credentials associated with it.
  - An IAM user can assume a role to temporarily take on different permissions for a specific task. A role can be assigned to a federated user who signs in by using an external identity provider instead of IAM.
  - AWS service role is a role that a service assumes to perform actions in your account on your behalf. This service role must include all the permissions required for the service to access the AWS resources that it needs.
    - AWS service role for an EC2 instance is a special type of service role that a service assumes to launch an EC2 instance that runs your application. This role is assigned to the EC2 instance when it is launched.
    - AWS service-linked role is a unique type of service role that is linked directly to an AWS service. Service-linked roles are predefined by the service and include all the permissions that the service requires to call other AWS services on your behalf.
  - An instance profile is a container for an IAM role that you can use to pass role information to an EC2 instance when the instance starts.
- Users or groups can have multiple policies attached to them that grant different permissions.

When to Create IAM User	When to Create an IAM Role
You created an AWS account and you're the only person who works in your account.	You're creating an application that runs on an Amazon EC2 instance and that application makes requests to AWS.
Other people in your group need to work in your AWS account, and your group is using no other identity mechanism.	You're creating an app that runs on a mobile phone and that makes requests to AWS.
You want to use the command-line interface to work with AWS.	Users in your comapany are authenticated in your corporate network and want to be able to use AWS without having no sign in again (federate into AWS)

Tutorials Dojo

AWS Key Management Service

- A managed service that enables you to easily encrypt your data. KMS provides a highly available key storage, management, and auditing solution for you to encrypt data within your own applications and control the encryption of stored data across AWS services.

Features

- KMS is integrated with CloudTrail, which provides you the ability to audit who used which keys, on which resources, and when.
- Customer master keys (CMKs) are used to control access to data encryption keys that encrypt and decrypt your data.
- You can choose to have KMS automatically rotate master keys created within KMS once per year without the need to re-encrypt data that has already been encrypted with your master key.

Concepts

- **Customer Master Keys (CMKs)** - You can use a CMK to encrypt and decrypt up to 4 KB of data. Typically, you use CMKs to generate, encrypt, and decrypt the data keys that you use outside of KMS to encrypt your data. Master keys are 256-bits in length.
- There are three types of CMKs:

Type of CMK	Can view	Can manage	Used only for my AWS account
Customer managed CMK	Yes	Yes	Yes
AWS managed CMK	Yes	No	Yes
AWS owned CMK	No	No	No

    - ***Customer managed CMKs*** are CMKs that you create, own, and manage. You have full control over these CMKs, including establishing and maintaining their key policies, IAM policies, and grants, enabling and disabling them, rotating their cryptographic material, adding tags, creating aliases that refer to the CMK, and scheduling the CMKs for deletion.
    - ***AWS managed CMKs*** are CMKs in your account that are created, managed, and used on your behalf by an AWS service that integrates with KMS. You can view the AWS managed CMKs in your account, view their key policies, and audit their use in CloudTrail logs. However, you cannot

manage these CMKs or change their permissions. And, you cannot use AWS managed CMKs in cryptographic operations directly; the service that creates them uses them on your behalf.

- ***AWS owned CMKs*** are not in your AWS account. They are part of a collection of CMKs that AWS owns and manages for use in multiple AWS accounts. AWS services can use AWS owned CMKs to protect your data. You cannot view, manage, or use AWS owned CMKs, or audit their use.

AWS Secrets Manager

- A secret management service that enables you to easily rotate, manage, and retrieve database credentials, API keys, and other secrets throughout their lifecycle.
- **Features**
    - AWS Secrets Manager encrypts secrets at rest using encryption keys that you own and store in AWS Key Management Service [customer managed keys]. When you retrieve a secret, Secrets Manager decrypts the secret and transmits it securely over TLS to your local environment.
    - You can rotate secrets on a schedule or on demand by using the Secrets Manager console, AWS SDK, or AWS CLI.
    - Secrets Manager natively supports rotating credentials for databases hosted on Amazon RDS and Amazon DocumentDB and clusters hosted on Amazon Redshift.
    - You can extend Secrets Manager to rotate other secrets, such as credentials for Oracle databases hosted on EC2 or OAuth refresh tokens, by using custom AWS Lambda functions.
- A secret consists of a set of credentials (username and password), and the connection details used to access a secured service.
- A secret can contain versions:
    - Although you typically only have one version of the secret active at a time, multiple versions can exist while you rotate a secret on the database or service. Whenever you change the secret, Secrets Manager creates a new version.
    - Each version holds a copy of the encrypted secret value.
    - Each version can have one or more staging labels attached identifying the stage of the secret rotation cycle.
- Supported Secrets
    - Database credentials, on-premises resource credentials, SaaS application credentials, third-party API keys, and SSH keys.
    - You can also store JSON documents.
- To retrieve secrets, you simply replace secrets in plain text in your applications with code to pull in those secrets programmatically using the Secrets Manager APIs.
- Secrets can be cached on the client side, and updated only during a secret rotation.
- During the secret rotation process, Secrets Manager tracks the older credentials, as well as the new credentials you want to start using, until the rotation completes. It tracks these different versions by using staging labels.

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.

AWS Auto Scaling

- Configure automatic scaling for the AWS resources quickly through a scaling plan that uses dynamic scaling and predictive scaling.
- Optimize for availability, for cost, or a balance of both.
- Scaling in means decreasing the size of a group while scaling out means increasing the size of a group.
- Useful for:
    - Cyclical traffic such as high use of resources during regular business hours and low use of resources overnight
    - On and off traffic patterns, such as batch processing, testing, or periodic analysis
    - Variable traffic patterns, such as software for marketing campaigns with periods of spiky growth
- **Features**
    - Launch or terminate EC2 instances in an Auto Scaling group.
    - Launch or terminate instances from an EC2 Spot Fleet request, or automatically replace instances that get interrupted for price or capacity reasons.
    - Adjust the ECS service desired count up or down in response to load variations.
    - Use Dynamic Scaling to add and remove capacity for resources to maintain resource utilization at the specified target value.
    - Use Predictive Scaling to forecast your future load demands by analyzing your historical records for a metric. It also allows you to schedule scaling actions that proactively add and remove resource capacity to reflect the load forecast, and control maximum capacity behavior. Only available for EC2 Auto Scaling groups.
    - You can suspend and resume any of your AWS Application Auto Scaling actions.
- **Amazon EC2 Auto Scaling**
    - Ensuring you have the correct number of EC2 instances available to handle your application load using Auto Scaling Groups.
    - An Auto Scaling group contains a collection of EC2 instances that share similar characteristics and are treated as a logical grouping for the purposes of instance scaling and management.
    - You specify the minimum, maximum and desired number of instances in each Auto Scaling group.
    - Key Components

Groups	Your EC2 instances are organized into groups so that they are treated as a logical unit for scaling and management. When you create a group, you can specify its minimum, maximum, and desired number of EC2 instances.

Launch configurations	Your group uses a launch configuration as a template for its EC2 instances. When you create a launch configuration, you can specify information such as the AMI ID, instance type, key pair, security groups, and block device mapping for your instances.
Scaling options	How to scale your Auto Scaling groups.

    - Scaling Options
        - Scale to maintain current instance levels at all times
        - Manual Scaling
        - Scale based on a schedule
        - Scale based on a demand
    - The cooldown period is a configurable setting that helps ensure to not launch or terminate additional instances before previous scaling activities take effect.
        - EC2 Auto Scaling supports cooldown periods when using simple scaling policies, but not when using target tracking policies, step scaling policies, or scheduled scaling.
    - Amazon EC2 Auto Scaling marks an instance as unhealthy if the instance is in a state other than running, the system status is impaired, or Elastic Load Balancing reports that the instance failed the health checks.
    - Termination of Instances
        - When you configure automatic scale in, you must decide which instances should terminate first and set up a termination policy. You can also use instance protection to prevent specific instances from being terminated during automatic scale in.
        - Default Termination Policy
        - Custom Termination Policies
            - OldestInstance - Terminate the oldest instance in the group.
            - NewestInstance - Terminate the newest instance in the group.
            - OldestLaunchConfiguration - Terminate instances that have the oldest launch configuration.
            - ClosestToNextInstanceHour - Terminate instances that are closest to the next billing hour.

  You can create launch templates that specifies instance configuration information when you launch EC2 instances, and allows you to have multiple versions of a template.
  A launch configuration is an instance configuration template that an Auto Scaling group uses to launch EC2 instances, and you specify information for the instances.
    - You can specify your launch configuration with multiple Auto Scaling groups.
    - You can only specify one launch configuration for an Auto Scaling group at a time, and you can't modify a launch configuration after you've created it.
- **Monitoring**
    - Health checks - identifies any instances that are unhealthy

        - Amazon EC2 status checks (default)
        - Elastic Load Balancing health checks
        - Custom health checks.
    - Auto scaling does not perform health checks on instances in the standby state. Standby state can be used for performing updates/changes/troubleshooting without health checks being performed or replacement instances being launched.
    - CloudWatch metrics - enables you to retrieve statistics about Auto Scaling-published data points as an ordered set of time-series data, known as metrics. You can use these metrics to verify that your system is performing as expected.
    - CloudWatch Events - Auto Scaling can submit events to CloudWatch Events when your Auto Scaling groups launch or terminate instances, or when a lifecycle action occurs.
    - SNS notifications - Auto Scaling can send Amazon SNS notifications when your Auto Scaling groups launch or terminate instances.
    - CloudTrail logs - enables you to keep track of the calls made to the Auto Scaling API by or on behalf of your AWS account, and stores the information in log files in an S3 bucket that you specify.

AWS CloudFormation

- A service that gives developers and businesses an easy way to create a collection of related AWS resources and provision them in an orderly and predictable fashion.

Features

- CloudFormation allows you to model your entire infrastructure in a text file called a **template**. You can use JSON or YAML to describe what AWS resources you want to create and configure. If you want to design visually, you can use *AWS CloudFormation Designer*.
- CloudFormation automates the provisioning and updating of your infrastructure in a safe and controlled manner. You can use **Rollback Triggers** to specify the CloudWatch alarm that CloudFormation should monitor during the stack creation and update process. If any of the alarms are breached, CloudFormation rolls back the entire stack operation to a previously deployed state.
- CloudFormation enables you to build custom extensions to your stack template using AWS Lambda.

Concepts

- **Templates**
  - A JSON or YAML formatted text file.
  - CloudFormation uses these templates as blueprints for building your AWS resources.
- **Stacks**
  - Manage related resources as a single unit.
  - All the resources in a stack are defined by the stack's CloudFormation template.
- **Change Sets**
  - Before updating your stack and making changes to your resources, you can generate a change set, which is a summary of your proposed changes.
  - Change sets allow you to see how your changes might impact your running resources, especially for critical resources, before implementing them.
- With AWS CloudFormation and AWS CodePipeline, you can use continuous delivery to automatically build and test changes to your CloudFormation templates before promoting them to production stacks.
- *CloudFormation artifacts* can include a stack template file, a template configuration file, or both. AWS CodePipeline uses these artifacts to work with CloudFormation stacks and change sets.
  - **Stack Template File** - defines the resources that CloudFormation provisions and configures. You can use YAML or JSON-formatted templates.
  - **Template Configuration File** - a JSON-formatted text file that can specify template parameter values, a stack policy, and tags. Use these configuration files to specify parameter values or a stack policy for a stack.

Stacks

- If a resource cannot be created, CloudFormation rolls the stack back and automatically deletes any resources that were created. If a resource cannot be deleted, any remaining resources are retained until the stack can be successfully deleted.
- Stack update methods
    - Direct update
    - Creating and executing change sets
- **Drift detection** enables you to detect whether a stack's actual configuration differs, or has drifted, from its expected configuration. Use CloudFormation to detect drift on an entire stack, or on individual resources within the stack.
    - A resource is considered to have drifted if any if its actual property values differ from the expected property values.
    - A stack is considered to have drifted if one or more of its resources have drifted.
- To share information between stacks, export a stack's output values. Other stacks that are in the same AWS account and region can import the exported values.
- You can nest stacks.

Templates

- Templates include several major sections. The Resources section is the only required section.
- **CloudFormation Designer** is a graphic tool for creating, viewing, and modifying CloudFormation templates. You can diagram your template resources using a drag-and-drop interface, and then edit their details using the integrated JSON and YAML editor.
- Custom resources enable you to write custom provisioning logic in templates that CloudFormation runs anytime you create, update (if you change the custom resource), or delete stacks.
- Template macros enable you to perform custom processing on templates, from simple actions like find-and-replace operations to extensive transformations of entire templates.

StackSets

- CloudFormation StackSets allow you to roll out CloudFormation stacks over multiple AWS accounts and in multiple Regions with just a couple of clicks. StackSets is commonly used together with AWS Organizations to centrally deploy and manage services in different accounts.
- Administrator and target accounts - An *administrator account* is the AWS account in which you create stack sets. A stack set is managed by signing in to the AWS administrator account in which it was created. A *target account* is the account into which you create, update, or delete one or more stacks in your stack set.
- Stack sets - A *stack set* lets you create stacks in AWS accounts across regions by using a single CloudFormation template. All the resources included in each stack are defined by the stack set's CloudFormation template. A stack set is a regional resource.
- Stack instances - A *stack instance* is a reference to a stack in a target account within a region. A stack instance can exist without a stack, and can be associated with only one stack set.

- Stack set operations - Create stack set, update stack set, delete stacks, and delete stack set.
- Tags - You can add tags during stack set creation and update operations by specifying key and value pairs.
- Drift detection identifies unmanaged changes or changes made to stacks outside of CloudFormation. When CloudFormation performs drift detection on a stack set, it performs drift detection on the stack associated with each stack instance in the stack set. If the current state of a resource varies from its expected state, that resource is considered to have drifted.
- If one or more resources in a stack has drifted then the stack itself is considered to have drifted, and the stack instances that the stack is associated with is considered to have drifted as well.
- If one or more stack instances in a stack set has drifted, the stack set itself is considered to have drifted.

AWS CloudTrail

- Actions taken by a user, role, or an AWS service in the AWS Management Console, AWS Command Line Interface, and AWS SDKs and APIs are recorded as events.
- CloudTrail is enabled on your AWS account when you create it.
- CloudTrail focuses on auditing API activity.
- **Trails**
  - Create a CloudTrail trail to archive, analyze, and respond to changes in your AWS resources.
  - Types
    - A trail that applies to all regions - CloudTrail records events in each region and delivers the CloudTrail event log files to an S3 bucket that you specify. This is the default option when you create a trail in the CloudTrail console.
    - A trail that applies to one region - CloudTrail records the events in the region that you specify only. This is the default option when you create a trail using the AWS CLI or the CloudTrail API.
  - You can create an organization trail that will log all events for all AWS accounts in an organization created by AWS Organizations. Organization trails must be created in the management account.
  - By default, CloudTrail event log files are encrypted using Amazon S3 server-side encryption. You can also choose to encrypt your log files with an AWS Key Management Service key.
  - You can store your log files in your S3 bucket for as long as you want, and also define S3 lifecycle rules to archive or delete log files automatically.
  - If you want notifications about log file delivery and validation, you can set up Amazon SNS notifications.
  - CloudTrail publishes log files about every five minutes.
- **Events**
  - The record of an activity in an AWS account. This activity can be an action taken by a user, role, or service that is monitorable by CloudTrail.
  - Types
    - Management events
      - Logged by default
      - Management events provide insight into management operations performed on resources in your AWS account, also known as control plane operations.
    - Data events
      - Not logged by default
      - Data events provide insight into the resource operations performed on or in a resource, also known as data plane operations.
      - Data events are often high-volume activities.
- **Monitoring**
  - Use CloudWatch Logs to monitor log data. CloudTrail events that are sent to CloudWatch Logs can trigger alarms according to the metric filters you define.

- To determine whether a log file was modified, deleted, or unchanged after CloudTrail delivered it, you can use CloudTrail log file integrity validation.

AWS Config

- A fully managed service that provides you with an AWS resource inventory, configuration history, and configuration change notifications to enable security and governance.

Features

- Multi-account, multi-region data aggregation gives you an enterprise-wide view of your ***Config rule*** compliance status, and you can associate your AWS organization to quickly add your accounts.
- Provides you pre-built rules to evaluate your AWS resource configurations and configuration changes, or create your own custom rules in AWS Lambda that define your internal best practices and guidelines for resource configurations.
- **Config records** details of changes to your AWS resources to provide you with a configuration history, and automatically deliver it to an S3 bucket you specify.
- Receive a notification whenever a resource is created, modified, or deleted.
- Config enables you to record software configuration changes within your EC2 instances and servers running on-premises, as well as servers and Virtual Machines in environments provided by other cloud providers. You gain visibility into:
    - operating system configurations
    - system-level updates
    - installed applications
    - network configuration
- Config can provide you with a **configuration snapshot** - a point-in-time capture of all your resources and their configurations.

Concepts

Configuration History

- A collection of the configuration items for a given resource over any time period, containing information such as when the resource was first created, how the resource has been configured over the last month, etc.
- Config automatically delivers a configuration history file for each resource type that is being recorded to an S3 bucket that you specify.
- A configuration history file is sent every six hours for each resource type that Config records.

Configuration item

- A record of the configuration of a resource in your AWS account. Config creates a configuration item whenever it detects a change to a resource type that it is recording.
- The components of a configuration item include metadata, attributes, relationships, current configuration, and related events.

Configuration Recorder

- Stores the configurations of the supported resources in your account as configuration items.

    - By default, the configuration recorder records all supported resources in the region where Config is running. You can create a customized configuration recorder that records only the resource types that you specify.
    - You can also have Config record supported types of *global resources* which are IAM users, groups, roles, and customer managed policies.

  Configuration Item
    - The configuration of a resource at a given point-in-time. A CI consists of 5 sections:
        - Basic information about the resource that is common across different resource types.
        - Configuration data specific to the resource.
        - Map of relationships with other resources.
        - CloudTrail event IDs that are related to this state.
        - Metadata that helps you identify information about the CI, such as the version of this CI, and when this CI was captured.

  Resource Relationship
    - Config discovers AWS resources in your account and then creates a map of relationships between AWS resources.

  Config rule
    - Represents your desired configuration settings for specific AWS resources or for an entire AWS account.
    - Provides customizable, predefined rules. If a resource violates a rule, Config flags the resource and the rule as noncompliant, and notifies you through Amazon SNS.
    - Evaluates your resources either **in response to configuration changes** or **periodically**.
- Multi-Account Multi-Region Data Aggregation
    - An aggregator collects configuration and compliance data from the following:
        - Multiple accounts and multiple regions.
        - Single account and multiple regions.
        - An organization in AWS Organizations and all the accounts in that organization.

Monitoring

- Use Amazon SNS to send you notifications every time a supported AWS resource is created, updated, or otherwise modified as a result of user API activity.
- Use Amazon CloudWatch Events to detect and react to changes in the status of AWS Config events.
- Use AWS CloudTrail to capture API calls to Config.

AWS Health

- Provides ongoing visibility into the state of your AWS resources, services, and accounts.
- The service delivers alerts and notifications triggered by changes in the health of AWS resources.
- The **Personal Health Dashboard**, powered by the AWS Health API, is available to all customers. The dashboard requires no setup, and it is ready to use for authenticated AWS users. The Personal Health Dashboard organizes issues in three groups:
    - Open issues - restricted to issues whose start time is within the last seven days.
    - Scheduled changes - contains items that are ongoing or upcoming.
    - Other notifications - restricted to issues whose start time is within the last seven days.
- You can centrally aggregate your AWS Health events from all accounts in your AWS Organization. The AWS Health Organizational View provides centralized and real-time access to all AWS Health events posted to individual accounts in your organization, including operational issues, scheduled maintenance, and account notifications.

AWS OpsWorks

- A configuration management service that helps you configure and operate applications in a cloud enterprise by using **Puppet** or **Chef**.
- AWS OpsWorks Stacks and AWS OpsWorks for Chef Automate (1 and 2) let you use Chef cookbooks and solutions for configuration management, while OpsWorks for Puppet Enterprise lets you configure a Puppet Enterprise master server in AWS.

OpsWorks for Puppet Enterprise

- Provides a fully-managed Puppet master, a suite of automation tools that enable you to inspect, deliver, operate, and future-proof your applications, and access to a user interface that lets you view information about your nodes and Puppet activities.
- Uses puppet-agent software.
- **Features**
    - AWS manages the Puppet master server running on an EC2 instance. You retain control over the underlying resources running your Puppet master.
    - You can choose the weekly maintenance window during which OpsWorks for Puppet Enterprise will automatically install updates.
    - Monitors the health of your Puppet master during update windows and automatically rolls back changes if issues are detected.
    - You can configure automatic backups for your Puppet master and store them in an S3 bucket in your account.
    - You can register new nodes to your Puppet master by inserting a user-data script, provided in the *OpsWorks for Puppet Enterprise StarterKit*, into your Auto Scaling groups.
    - Puppet uses SSL and a certification approval process when communicating to ensure that the Puppet master responds only to requests made by trusted users.
- Deleting a server also deletes its events, logs, and any modules that were stored on the server. Supporting resources are also deleted, along with all automated backups.

OpsWorks for Chef Automate

- Lets you create AWS-managed Chef servers that include Chef Automate premium features, and use the Chef DK and other Chef tooling to manage them.
- Uses chef-client.
- **Features**
    - You can use Chef to manage both Amazon EC2 instances and on-premises servers running Linux or Windows.
    - You receive the full Chef Automate platform which includes premium features that you can use with Chef server, like Chef Workflow, Chef Visibility, and Chef Compliance.

    - You provision a managed Chef server running on an EC2 instance in your account. You retain control over the underlying resources running your Chef server and you can use Knife to SSH into your Chef server instance at any time.
    - You can set a weekly maintenance window during which OpsWorks for Chef Automate will automatically install updates.
    - You can configure automatic backups for your Chef server and is stored in an S3 bucket.
    - You can register new nodes to your Chef server by inserting user-data code snippets provided by OpsWorks for Chef Automate into your Auto Scaling groups.
    - Chef uses SSL to ensure that the Chef server responds only to requests made by trusted users. The Chef server and Chef client use bidirectional validation of identity when communicating with each other.
- Deleting a server also deletes its events, logs, and any cookbooks that were stored on the server. Supporting resources are deleted also, along with all automated backups.

OpsWorks Stacks

- Provides a simple and flexible way to create and manage stacks and applications.
- You can create stacks that help you manage cloud resources in specialized groups called **layers**. A layer represents a set of EC2 instances that serve a particular purpose, such as serving applications or hosting a database server. Layers depend on Chef recipes to handle tasks such as installing packages on instances, deploying apps, and running scripts.
- **Features**
    - You can deploy EC2 instances from template configurations, including EBS volume creation.
    - You can configure the software on your instances on-demand or automatically based on lifecycle events, from bootstrapping the base OS image into a working server to modifying running services to reflect changes.
    - OpsWorks Stacks can auto heal your stack. If an instance fails in your stack, OpsWorks Stacks can replace it with a new one.
    - You can adapt the number of running instances to match your load, with time-based or load-based auto scaling.
    - You can use OpsWorks Stacks to configure and manage both Linux and Windows EC2 instances.
    - You can use AWS OpsWorks Stacks to deploy, manage, and scale your application on any Linux server such as EC2 instances or servers running in your own data center.
- **Instance Types**
    - **24/7 instances** are started manually and run until you stop them.
    - **Time-based instances** are run by OpsWorks Stacks on a specified daily and weekly schedule. They allow your stack to automatically adjust the number of instances to accommodate predictable usage patterns.

    - **Load-based instances** are automatically started and stopped by OpsWorks Stacks, based on specified load metrics, such as CPU utilization. They allow your stack to automatically adjust the number of instances to accommodate variations in incoming traffic.
        - Load-based instances are available only for Linux-based stacks.
- **Lifecycle Events**
    - You can run recipes manually, but OpsWorks Stacks also lets you automate the process by supporting a set of five lifecycle events:
        - **Setup** occurs on a new instance after it successfully boots.
        - **Configure** occurs on all of the stack's instances when an instance enters or leaves the online state.
        - **Deploy** occurs when you deploy an app.
        - **Undeploy** occurs when you delete an app.
        - **Shutdown** occurs when you stop an instance.

AWS Systems Manager

- Allows you to centralize operational data from multiple AWS services and automate tasks across your AWS resources.

Features

- Create logical groups of resources such as applications, different layers of an application stack, or production versus development environments.
- You can select a resource group and view its recent API activity, resource configuration changes, related notifications, operational alerts, software inventory, and patch compliance status.
- Collects information about your instances and the software installed on them.
- Allows you to safely automate common and repetitive IT operations and management tasks across AWS resources.
- Provides a browser-based interactive shell and CLI for managing Windows and Linux EC2 instances, without the need to open inbound ports, manage SSH keys, or use bastion hosts. Administrators can grant and revoke access to instances through a central location by using IAM policies.
- Helps ensure that your software is up-to-date and meets your compliance policies.
- Lets you schedule windows of time to run administrative and maintenance tasks across your instances.

SSM Agent is the tool that processes Systems Manager requests and configures your machine as specified in the request. SSM Agent must be installed on each instance you want to use with Systems Manager. On newer AMIs and instance types, SSM Agent is installed by default. On older versions, you must install it manually.

Capabilities

- **Automation**
    - Allows you to safely automate common and repetitive IT operations and management tasks across AWS resources
    - A **step** is defined as an initiated action performed in the Automation execution on a per-target basis. You can execute the entire Systems Manager automation document in one action or choose to execute one step at a time.
    - Concepts
        - **Automation document** - defines the Automation workflow.
        - **Automation action** - the Automation workflow includes one or more steps. Each step is associated with a particular action or plugin. The action determines the inputs, behavior, and outputs of the step.
        - **Automation queue** - if you attempt to run more than 25 Automations simultaneously, Systems Manager adds the additional executions to a queue and displays a status of *Pending*. When an Automation reaches a terminal state, the first execution in the queue starts.

    - You can schedule Systems Manager automation document execution.
- **Resource Groups**
    - A collection of AWS resources that are all in the same AWS region, and that match criteria provided in a query.
    - Use Systems Manager tools such as *Automation* to simplify management tasks on your groups of resources. You can also use groups as the basis for viewing monitoring and configuration *insights* in Systems Manager.
- **Built-in Insights**
    - Show detailed information about a single, selected resource group.
    - Includes recent API calls through CloudTrail, recent configuration changes through Config, Instance software inventory listings, instance patch compliance views, and instance configuration compliance views.
- **Systems Manager Activation**
    - Enable hybrid and cross-cloud management. You can register any server, whether physical or virtual to be managed by Systems Manager.
- **Inventory Manager**
    - Automates the process of collecting software inventory from managed instances.
    - You specify the type of metadata to collect, the instances from where the metadata should be collected, and a schedule for metadata collection.
- **Configuration Compliance**
    - Scans your fleet of managed instances for patch compliance and configuration inconsistencies.
    - View compliance history and change tracking for Patch Manager patching data and State Manager associations by using AWS Config.
    - Customize Systems Manager Compliance to create your own compliance types.
- **Run Command**
    - Remotely and securely manage the configuration of your managed instances at scale.
    - **Managed Instances** - any EC2 instance or on-premises server or virtual machine in your hybrid environment that is configured for Systems Manager.
- **Session Manager**
    - Manage your EC2 instances through an interactive one-click browser-based shell or through the AWS CLI.
    - Makes it easy to comply with corporate policies that require controlled access to instances, strict security practices, and fully auditable logs with instance access details, while still providing end users with simple one-click cross-platform access to your Amazon EC2 instances.
    - You can use AWS Systems Manager Session Manager to tunnel SSH (Secure Shell) and SCP (Secure Copy) traffic between a client and a server.
- **Distributor**
    - Lets you package your own software or find AWS-provided agent software packages to install on Systems Manager managed instances.
    - After you create a package in Distributor, which creates an Systems Manager document, you can install the package in one of the following ways.

        - One time by using Systems Manager Run Command.
        - On a schedule by using Systems Manager State Manager.
- **Patch Manager**
    - Automate the process of patching your managed instances.
    - Enables you to scan instances for missing patches and apply missing patches individually or to large groups of instances by using EC2 instance tags.
    - For security patches, Patch Manager uses *patch baselines* that include rules for auto-approving patches within days of their release, as well as a list of approved and rejected patches.
    - You can use AWS Systems Manager Patch Manager to select and apply Microsoft application patches automatically across your Amazon EC2 or on-premises instances.
    - AWS Systems Manager Patch Manager includes common vulnerability identifiers (CVE ID). CVE IDs can help you identify security vulnerabilities within your fleet and recommend patches.
- **Maintenance Window**
    - Set up recurring schedules for managed instances to execute administrative tasks like installing patches and updates without interrupting business-critical operations.
    - Supports running four types of tasks:
        - Systems Manager Run Command commands
        - Systems Manager Automation workflows
        - AWS Lambda functions
        - AWS Step Functions tasks
- **Systems Manager Document (SSM)**
    - Defines the actions that Systems Manager performs.
    - Types of SSM Documents

Type	Use with	Details
Command document	Run Command, State Manager	Run Command uses command documents to execute commands. State Manager uses command documents to apply a configuration. These actions can be run on one or more targets at any point during the lifecycle of an instance.
Policy document	State Manager	Policy documents enforce a policy on your targets. If the policy document is removed, the policy action no longer happens.
Automation document	Automation	Use automation documents when performing common maintenance and deployment tasks such as creating or updating an AMI.

Package document	Distributor	In Distributor, a package is represented by a Systems Manager document. A package document includes attached ZIP archive files that contain software or assets to install on managed instances. Creating a package in Distributor creates the package document.

- Can be in JSON or YAML.
- You can create and save different versions of documents. You can then specify a default version for each document.
- If you want to customize the steps and actions in a document, you can create your own.
- You can tag your documents to help you quickly identify one or more documents based on the tags you've assigned to them.

State Manager

- A service that automates the process of keeping your EC2 and hybrid infrastructure in a state that you define.
- A *State Manager association* is a configuration that is assigned to your managed instances. The configuration defines the state that you want to maintain on your instances. The association also specifies actions to take when applying the configuration.

Parameter Store

- Provides secure, hierarchical storage for configuration data and secrets management.
- You can store values as plain text or encrypted data with *SecureString*.
- Parameters work with Systems Manager capabilities such as Run Command, State Manager, and Automation.

OpsCenter

- OpsCenter helps you view, investigate, and resolve operational issues related to your environment from a central location.
- OpsCenter complements existing case management systems by enabling integrations via Amazon Simple Notification Service (SNS) and public AWS SDKs. By aggregating information from AWS Config, AWS CloudTrail logs, resource descriptions, and Amazon CloudWatch Events, OpsCenter helps you reduce the mean time to resolution (MTTR) of incidents, alarms, and operational tasks.

AWS Trusted Advisor

- Trusted Advisor analyzes your AWS environment and provides best practice recommendations in five categories:
    - Cost Optimization
    - Performance
    - Security
    - Fault Tolerance
    - Service Limits
- Access to the seven core Trusted Advisor checks are available to all AWS users.
- Access to the full set of Trusted Advisor checks are available to Business and Enterprise Support plans.

AWS Analytics Services

Amazon Elasticsearch (ES)

- Amazon ES lets you search, analyze, and visualize your data **in real-time**. This service manages the capacity, scaling, patching, and administration of your Elasticsearch clusters for you, while still giving you direct access to the Elasticsearch APIs.
- The service offers open-source Elasticsearch APIs, managed Kibana, and integrations with Logstash and other AWS Services. This combination is often coined as the **ELK Stack**.
- **Concepts**
    - An Amazon ES **domain** is synonymous with an Elasticsearch cluster. Domains are clusters with the settings, instance types, instance counts, and storage resources that you specify.
    - You can create multiple Elasticsearch indices within the same domain. Elasticsearch automatically distributes the indices and any associated replicas between the instances allocated to the domain.
    - Amazon ES uses a **blue/green deployment process** when updating domains. Blue/green typically refers to the practice of running two production environments, one live and one idle, and switching the two as you make software changes.
- **Data Ingestion**
    - Easily ingest structured and unstructured data into your Amazon Elasticsearch domain with **Logstash**, an open-source data pipeline that helps you process logs and other event data.
    - You can also ingest data into your Amazon Elasticsearch domain using Amazon Kinesis Firehose, AWS IoT, or Amazon CloudWatch Logs.
    - You can get faster and better insights into your data using **Kibana**, an open-source analytics and visualization platform. Kibana is automatically deployed with your Amazon Elasticsearch Service domain.
    - You can load streaming data from the following sources using AWS Lambda event handlers:
        - Amazon S3
        - Amazon Kinesis Data Streams and Data Firehose
        - Amazon DynamoDB
        - Amazon CloudWatch
        - AWS IoT
    - Amazon ES exposes three Elasticsearch logs through CloudWatch Logs:
        - error logs
        - search slow logs - These logs help fine tune the performance of any kind of search operation on Elasticsearch.
        - index slow logs - These logs provide insights into the indexing process and can be used to fine-tune the index setup.
    - Kibana and Logstash
        - Kibana is a popular open source visualization tool designed to work with Elasticsearch.

- The URL is *elasticsearch-domain-endpoint*/_plugin/kibana/.
- You can configure your own Kibana instance aside from using the default provided Kibana.
- Amazon ES uses **Amazon Cognito** to offer username and password protection for Kibana. (Optional feature)
- Logstash provides a convenient way to use the bulk API to upload data into your Amazon ES domain with the S3 plugin. The service also supports all other standard Logstash input plugins that are provided by Elasticsearch.
- Amazon ES also supports two Logstash output plugins:
  - standard Elasticsearch plugin
  - *logstash-output-amazon-es* plugin, which signs and exports Logstash events to Amazon ES.

Amazon Kinesis

- Makes it easy to collect, process, and analyze real-time, streaming data.
- Kinesis can ingest real-time data such as video, audio, application logs, website clickstreams, and IoT telemetry data for machine learning, analytics, and other applications.

Kinesis Data Stream

- A massively scalable, highly durable data ingestion and processing service optimized for streaming data. You can configure hundreds of thousands of data producers to continuously put data into a Kinesis data stream.
- **Concepts**
    - **Data Producer** - An application that typically emits data records as they are generated to a Kinesis data stream. Data producers assign partition keys to records. Partition keys ultimately determine which shard ingests the data record for a data stream.
    - **Data Consumer** - A distributed Kinesis application or AWS service retrieving data from all shards in a stream as it is generated. Most data consumers are retrieving the most recent data in a shard, enabling real-time analytics or handling of data.
    - **Data Stream** - A logical grouping of shards. There are no bounds on the number of shards within a data stream. A data stream will retain data for **24 hours, or up to 7 days** when extended retention is enabled.
    - **Shard** - The base throughput unit of a Kinesis data stream.
        - A shard is an append-only log and a unit of streaming capability. A shard contains an ordered sequence of records ordered by arrival time.
        - Add or remove shards from your stream dynamically as your data throughput changes.
        - One shard can ingest up to 1000 data records per second, or 1MB/sec. Add more shards to increase your ingestion capability.
        - When consumers use **enhanced fan-out**, one shard provides 1MB/sec data input and 2MB/sec data output for each data consumer registered to use enhanced fan-out.
        - When consumers do **not** use **enhanced fan-out**, a shard provides 1MB/sec of input and 2MB/sec of data output, and this output is shared with any consumer not using enhanced fan-out.
    - **Data Record**
        - A record is the unit of data stored in a Kinesis stream. A record is composed of a sequence number, partition key, and data blob.
        - A data blob is the data of interest your data producer adds to a stream. The maximum size of a data blob is 1 MB.
    - **Partition Key**
        - A partition key is typically a meaningful identifier, such as a user ID or timestamp. It is specified by your data producer while putting data into a Kinesis data stream, and useful

for consumers as they can use the partition key to replay or build a history associated with the partition key.
        - The partition key is also used to segregate and route data records to different shards of a stream.
    - **Sequence Number**
        - A sequence number is a unique identifier for each data record. Sequence number is assigned by Kinesis Data Streams when a data producer calls *PutRecord* or *PutRecords* API to add data to a Kinesis data stream.

Kinesis Data Firehose

- The easiest way to load streaming data into data stores and analytics tools.
- It is a fully managed service that automatically scales to match the throughput of your data.
- It can also batch, compress, and encrypt the data before loading it.
- **Features**
    - It can capture, transform, and load streaming data into S3, Redshift, Elasticsearch Service, and Splunk, enabling near real-time analytics with existing business intelligence tools and dashboards being used today.
    - Once launched, your delivery streams automatically scale up and down to handle gigabytes per second or more of input data rate, and maintain data latency at levels you specify for the stream.
    - Kinesis Data Firehose can convert the format of incoming data from JSON to Parquet or ORC formats before storing the data in S3.
    - You can configure Kinesis Data Firehose to prepare your streaming data before it is loaded to data stores. Kinesis Data Firehose provides pre-built Lambda blueprints for converting common data sources such as Apache logs and system logs to JSON and CSV formats. You can use these pre-built blueprints without any change, or customize them further, or write your own custom functions.
- **Concepts**
    - **Kinesis Data Firehose Delivery Stream** - The underlying entity of Kinesis Data Firehose. You use Kinesis Data Firehose by creating a Kinesis Data Firehose delivery stream and then sending data to it.
    - **Record** - The data of interest that your data producer sends to a Kinesis Data Firehose delivery stream. A record can be as large as 1,000 KB.
    - **Data Producer** - Producers send records to Kinesis Data Firehose delivery streams.
    - **Buffer Size and Buffer Interval** - Kinesis Data Firehose buffers incoming streaming data to a certain size or for a certain period of time before delivering it to destinations. Buffer Size is in MBs and Buffer Interval is in seconds.
- **Stream Sources**
    - You can send data to your Kinesis Data Firehose Delivery stream using different types of sources:

        - a Kinesis data stream,
        - the Kinesis Agent,
        - or the Kinesis Data Firehose API using the AWS SDK.
    - You can also use CloudWatch Logs, CloudWatch Events, or AWS IoT as your data source.
    - Some AWS services can only send messages and events to a Kinesis Data Firehose delivery stream that is in the same Region.
- **Data Delivery and Transformation**
    - Kinesis Data Firehose can invoke your Lambda function to transform incoming source data and deliver the transformed data to destinations.
    - Kinesis Data Firehose buffers incoming data up to 3 MB by default.
    - If your Lambda function invocation fails because of a network timeout or because you've reached the Lambda invocation limit, Kinesis Data Firehose retries the invocation three times by default.
    - Kinesis Data Firehose can convert the format of your input data from JSON to Apache Parquet or Apache ORC before storing the data in S3. Parquet and ORC are columnar data formats that save space and enable faster queries compared to row-oriented formats like JSON.
    - Data delivery format:
        - **For data delivery to S3**, Kinesis Data Firehose concatenates multiple incoming records based on buffering configuration of your delivery stream. It then delivers the records to S3 as an S3 object.
        - **For data delivery to Redshift**, Kinesis Data Firehose first delivers incoming data to your S3 bucket in the format described earlier. Kinesis Data Firehose then issues an Redshift *COPY* command to load the data from your S3 bucket to your Redshift cluster.
        - **For data delivery to ElasticSearch**, Kinesis Data Firehose buffers incoming records based on buffering configuration of your delivery stream. It then generates an Elasticsearch bulk request to index multiple records to your Elasticsearch cluster.
        - **For data delivery to Splunk**, Kinesis Data Firehose concatenates the bytes that you send.

Kinesis Data Analytics

- Analyze streaming data, gain actionable insights, and respond to your business and customer needs in real time. You can quickly build SQL queries and Java applications using built-in templates and operators for common processing functions to organize, transform, aggregate, and analyze data at any scale.
- **General Features**
    - Kinesis Data Analytics is **serverless** and takes care of everything required to continuously run your application.
    - Kinesis Data Analytics elastically scales applications to keep up with any volume of data in the incoming data stream.
    - Kinesis Data Analytics delivers sub-second processing latencies so you can generate real-time alerts, dashboards, and actionable insights.

- An **application** is the primary resource in Kinesis Data Analytics. Kinesis data analytics applications continuously read and process streaming data in real time.
    - You write application code using SQL to process the incoming streaming data and produce output. Then, Kinesis Data Analytics writes the output to a configured destination.
    - You can also process and analyze streaming data using Java.
- **Components**
    - Input is the streaming source for your application. In the input configuration, you map the streaming source to an in-application data stream(s).
    - **Application code** is a series of SQL statements that process input and produce output.
    - You can create one or more in-application streams to store the **output**. You can then optionally configure an application output to persist data from specific in-application streams to an external destination.
- An **in-application data stream** is an entity that continuously stores data in your application for you to perform processing.

AWS Developer Tools

AWS CodeBuild

- A fully managed **continuous integration service** that compiles source code, runs tests, and produces software packages that are ready to deploy.
- **Concepts**
  - A **build project** defines how CodeBuild will run a build. It includes information such as where to get the source code, which build environment to use, the build commands to run, and where to store the build output.
  - A **build environment** is the combination of operating system, programming language runtime, and tools used by CodeBuild to run a build.
  - The **build specification** is a YAML file that lets you choose the commands to run at each phase of the build and other settings. Without a build spec, CodeBuild cannot successfully convert your build input into build output or locate the build output artifact in the build environment to upload to your output bucket.
    - If you include a build spec as part of the source code, by default, the build spec file must be named buildspec.yml and placed in the root of your source directory.
  - A collection of input files is called **build input artifacts or build input** and a deployable version of a source code is called **build output artifact or build output**.
- **Features**
  - AWS CodeBuild runs your builds in preconfigured build environments that contain the operating system, programming language runtime, and build tools (such as Apache Maven, Gradle, npm) required to complete the task. You just specify your source code's location and select settings for your build, such as the build environment to use and the build commands to run during a build.
  - AWS CodeBuild builds your code and stores the artifacts into an Amazon S3 bucket, or you can use a build command to upload them to an artifact repository.
  - AWS CodeBuild provides build environments for
    - Java
    - Python
    - Node.js
    - Ruby
    - Go
    - Android
    - .NET Core for Linux
    - Docker
  - You can define the specific commands that you want AWS CodeBuild to perform, such as installing build tool packages, running unit tests, and packaging your code.

    - You can integrate CodeBuild into existing CI/CD workflows using its source integrations, build commands, or Jenkins integration.
    - CodeBuild can connect to AWS CodeCommit, S3, GitHub, and GitHub Enterprise and Bitbucket to pull source code for builds.
    - CodeBuild allows you to use Docker images stored in another AWS account as your build environment, by granting resource level permissions.
    - It now allows you to access Docker images from any private registry as the build environment. Previously, you could only use Docker images from public DockerHub or Amazon ECR in CodeBuild.
    - You can access your past build results through the console, CloudWatch, or the API. The results include outcome (success or failure), build duration, output artifact location, and log location.
    - You can automate your release process by using **AWS CodePipeline** to test your code and run your builds with CodeBuild.
- **Steps in a Build Process**
    - CodeBuild will create a temporary compute container of the class defined in the build project
    - CodeBuild loads it with the specified runtime environment
    - CodeBuild downloads the source code
    - CodeBuild executes the commands configured in the project
    - CodeBuild uploads the generated artifact to an S3 bucket
    - Then it destroys the compute container
- Build Duration is calculated in minutes, from the time you submit your build until your build is terminated, rounded up to the nearest minute.
- You can save time when your project builds by using a cache. A build project can use one of two types of caching:
    - Amazon S3 - stores the cache in an Amazon S3 bucket that is available across multiple build hosts. This is a good option for small intermediate build artifacts that are more expensive to build than to download. Not the best option for large build artifacts because they can take a long time to transfer over your network, which can affect build performance.
    - Local - stores a cache locally on a build host that is available to that build host only. This is a good option for large intermediate build artifacts because the cache is immediately available on the build host. Build performance is not impacted by network transfer time.
    - If you use a local cache, you must choose one or more of three cache modes:
        - source cache
        - Docker layer cache
        - custom cache.

AWS CodeCommit

- A **fully-managed source control** service that hosts secure Git-based repositories, similar to Github.
- You can create your own code repository and use Git commands to interact with your own repository and other repositories.
- You can store and version any kind of file, including application assets such as images and libraries alongside your code.
- The AWS CodeCommit Console lets you visualize your code, pull requests, commits, branches, tags and other settings.
- **Concepts**
    - An *active user* is any unique AWS identity (IAM user/role, federated user, or root account) that accesses AWS CodeCommit repositories during the month. AWS identities that are created through your use of other AWS Services, such as AWS CodeBuild and AWS CodePipeline, as well as servers accessing CodeCommit using a unique AWS identity, count as active users.
    - A **repository** is the fundamental version control object in CodeCommit. It's where you securely store code and files for your project. It also stores your project history, from the first commit through the latest changes.
    - A **file** is a version-controlled, self-contained piece of information available to you and other users of the repository and branch where the file is stored.
    - A **pull request** allows you and other repository users to review, comment on, and merge code changes from one branch to another.
    - An **approval rule** is used to designate a number of users who will approve a pull request before it is merged into your branch.
    - A **commit** is a snapshot of the contents and changes to the contents of your repository. This includes information like who committed the change, the date and time of the commit, and the changes made as part of the commit.
    - In Git, **branches** are simply pointers or references to a commit. You can use branches to separate work on a new or different version of files without impacting work in other branches. You can use branches to develop new features, store a specific version of your project from a particular commit, etc.
- **Repository Features**
    - You can share your repository with other users.
    - If you add AWS tags to repositories, you can set up notifications so that repository users receive email about events, such as another user commenting on code.
    - You can create triggers for your repository so that code pushes or other events trigger actions, such as emails or code functions.
    - To copy a remote repository to your local computer, use the command ‘git clone’
    - To connect to the repository after the name is changed, users must use the ‘git remote set-url’ command and specify the new URL to use.

  - To push changes from the local repo to the CodeCommit repository, run 'git push *remote-name branch-name*'.
  - To pull changes to the local repo from the CodeCommit repository, run 'git pull *remote-name branch-name*'.
  - You can create up to 10 triggers for Amazon SNS or AWS Lambda for each CodeCommit repository.
  - You can push your files to two different repositories at the same time.
- **Pull Requests**
  - Pull requests require two branches: a source branch that contains the code you want reviewed, and a destination branch, where you merge the reviewed code.
  - Create pull requests to let other users see and review your code changes before you merge them into another branch.
  - Create approval rules for your pull requests to ensure the quality of your code by requiring users to approve the pull request before the code can be merged into the destination branch. You can specify the number of users who must approve a pull request. You can also specify an approval pool of users for the rule.
  - To review the changes on files included in a pull request and resolve merge conflicts, you use the CodeCommit console, the 'git diff' command, or a diff tool.
  - After the changes have been reviewed and all approval rules on the pull request have been satisfied, you can merge a pull request using the AWS Console, AWS CLI or with the 'git merge' command.
  - You can close a pull request without merging it with your code.
- **Commit and Branch Features**
  - If using the AWS CLI, you can use the 'create-commit' command to create a new commit for your branch.
  - If using the AWS CLI, you can use 'create-branch' command to create a new branch for your repository..
  - You can also use Git commands to manage your commits and branches.
  - Create a new commit to a branch using 'git commit -m *message*'.
  - Create a branch in your local repo by running the git checkout -b *new-branch-name* command.
- **Migration from Git repositories to CodeCommit**
  - You can migrate a Git repository to a CodeCommit repository in a number of ways: by cloning it, mirroring it, or migrating all or just some of the branches.
  - You can also migrate your local repository in your machine to CodeCommit.
- **Monitoring**
  - CodeCommit uses AWS IAM to control and monitor who can access your data as well as how, when, and where they can access it.
  - CodeCommit helps you monitor your repositories via AWS CloudTrail and AWS CloudWatch.

- You can use Amazon SNS to receive notifications for events impacting your repositories. Each notification will include a status message as well as a link to the resources whose event generated that notification.

AWS CodeDeploy

- A fully managed deployment service that automates software deployments to a variety of compute services such as Amazon EC2, AWS Fargate, AWS Lambda, and your on-premises servers.
- **Concepts**
  - An Application is a name that uniquely identifies the application you want to deploy. CodeDeploy uses this name, which functions as a container, to ensure the correct combination of revision, deployment configuration, and deployment group are referenced during a deployment.
  - Compute platform is the platform on which CodeDeploy deploys an application (EC2, ECS, Lambda, On-premises servers).
  - Deployment configuration is a set of deployment rules and deployment success and failure conditions used by CodeDeploy during a deployment.
  - Deployment group contains individually tagged instances, Amazon EC2 instances in Amazon EC2 Auto Scaling groups, or both.
    1. In an Amazon ECS deployment, a deployment group specifies the Amazon ECS service, load balancer, optional test listener, and two target groups. It also specifies when to reroute traffic to the replacement task set and when to terminate the original task set and ECS application after a successful deployment.
    2. In an AWS Lambda deployment, a deployment group defines a set of CodeDeploy configurations for future deployments of an AWS Lambda function.
    3. In an EC2/On-Premises deployment, a deployment group is a set of individual instances targeted for a deployment.
       - In an in-place deployment, the instances in the deployment group are updated with the latest application revision.
       - In a blue/green deployment, traffic is rerouted from one set of instances to another by deregistering the original instances from a load balancer and registering a replacement set of instances that typically has the latest application revision already installed.
  - A deployment goes through a set of predefined phases called deployment lifecycle events. A deployment lifecycle event gives you an opportunity to run code as part of the deployment.
    1. ApplicationStop
    2. DownloadBundle
    3. BeforeInstall
    4. Install
    5. AfterInstall
    6. ApplicationStart
    7. ValidateService
  - Features
    - CodeDeploy protects your application from downtime during deployments through rolling updates and deployment health tracking.
    - AWS CodeDeploy tracks and stores the recent history of your deployments.

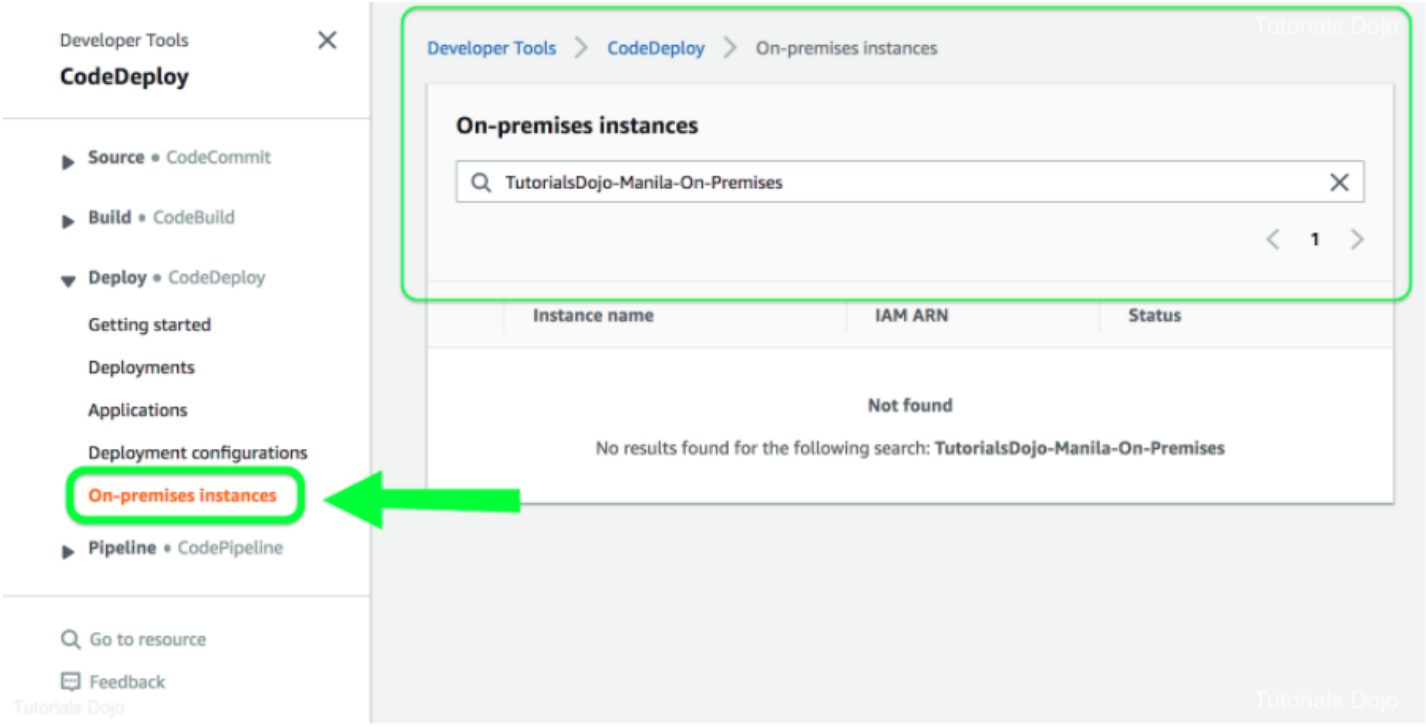
        - CodeDeploy is platform and language agnostic.
        - CodeDeploy uses a file and command-based install model, which enables it to deploy any application and reuse existing setup code. The same setup code can be used to consistently deploy and test updates across your environment release stages for your servers or containers.
        - CodeDeploy integrates with Amazon Auto Scaling, which allows you to scale EC2 capacity according to conditions you define such as traffic spikes. Notifications are then sent to AWS CodeDeploy to initiate an application deployment onto new instances before they are placed behind an Elastic Load Balancing load balancer.
        - When using AWS CodeDeploy with on-premises servers, make sure that they can connect to AWS public endpoints.
        - AWS CodeDeploy offers two types of deployments:
            - With in-place deployments, the application on each instance in the deployment group is stopped, the latest application revision is installed, and the new version of the application is started and validated. Only deployments that use the EC2/On-Premises compute platform can use in-place deployments.
            - With blue/green deployments, once the new version of your application is tested and declared ready, CodeDeploy can shift the traffic from your old version (blue) to your new version (green) according to your specifications.
        - Deployment groups are used to match configurations to specific environments, such as a staging or production environments. An application can be deployed to multiple deployment groups.
        - You can integrate AWS CodeDeploy with your continuous integration and deployment systems by calling the public APIs using the AWS CLI or AWS SDKs.
    - Application Specification Files
        - The AppSpec file is a YAML-formatted or JSON-formatted file that is used to manage each deployment as a series of lifecycle event hooks.
        - For ECS Compute platform, the file specifies
            - The name of the ECS service and the container name and port used to direct traffic to the new task set.
            - The functions to be used as validation tests.
        - For Lambda compute platform, the file specifies
            - The AWS Lambda function version to deploy.
            - The functions to be used as validation tests.
        - For EC2/On-Premises compute platform, the file is always written in YAML and is used to
            - Map the source files in your application revision to their destinations on the instance.
            - Specify custom permissions for deployed files.
            - Specify scripts to be run on each instance at various stages of the deployment process.
    - Deployments

- You can use the CodeDeploy console or the create-deployment command to deploy the function revision specified in the AppSpec file to the deployment group.
- You can use the CodeDeploy console or the stop-deployment command to stop a deployment. When you attempt to stop the deployment, one of three things happens:
  - The deployment stops, and the operation returns a status of SUCCEEDED.
  - The deployment does not immediately stop, and the operation returns a status of pending. After the pending operation is complete, subsequent calls to stop the deployment return a status of SUCCEEDED.
  - The deployment cannot stop, and the operation returns an error.
- With Lambda functions and EC2 instances, CodeDeploy implements rollbacks by redeploying, as a new deployment, a previously deployed revision.
- With ECS services, CodeDeploy implements rollbacks by rerouting traffic from the replacement task set to the original task set.
- The CodeDeploy agent is a software package that, when installed and configured on an EC2/on-premises instance, makes it possible for that instance to be used in CodeDeploy deployments. The agent is not required for deployments that use the Amazon ECS or AWS Lambda.
- CodeDeploy monitors the health status of the instances in a deployment group. For the overall deployment to succeed, CodeDeploy must be able to deploy to each instance in the deployment and deployment to at least one instance must succeed.
- You can specify a minimum number of healthy instances as a number of instances or as a percentage of the total number of instances required for the deployment to be successful.
- CodeDeploy assigns two health status values to each instance:
  - Revision health - based on the application revision currently installed on the instance. Values include Current, Old and Unknown.
  - Instance health - based on whether deployments to an instance have been successful. Values include Healthy and Unhealthy.

- Blue/Green Deployments
  - EC2/On-Premises compute platform
    - You must have one or more Amazon EC2 instances with identifying Amazon EC2 tags or an Amazon EC2 Auto Scaling group.
    - Each Amazon EC2 instance must have the correct IAM instance profile attached.
    - The CodeDeploy agent must be installed and running on each instance.
    - During replacement, you can either
      - use the Amazon EC2 Auto Scaling group you specify as a template for the replacement environment; or
      - specify the instances to be counted as your replacement using EC2 instance tags, EC2 Auto Scaling group names, or both.
  - AWS Lambda platform

        - You must choose one of the following deployment configuration types to specify how traffic is shifted from the original Lambda function version to the new version:
            - Canary: Traffic is shifted in two increments. You can choose from predefined canary options that specify the percentage of traffic shifted to your updated Lambda function version in the first increment and the interval, in minutes, before the remaining traffic is shifted in the second increment.
            - Linear: Traffic is shifted in equal increments with an equal number of minutes between each increment. You can choose from predefined linear options that specify the percentage of traffic shifted in each increment and the number of minutes between each increment.
            - All-at-once: All traffic is shifted from the original Lambda function to the updated Lambda function version all at once.

        - With Amazon ECS, production traffic shifts from your ECS service's original task set to a replacement task set all at once.
    - Advantages of using Blue/Green Deployments vs In-Place Deployments
        - An application can be installed and tested in the new replacement environment and deployed to production simply by rerouting traffic.
        - If you're using the EC2/On-Premises compute platform, switching back to the most recent version of an application is faster and more reliable. Traffic can just be routed back to the original instances as long as they have not been terminated. With an in-place deployment, versions must be rolled back by redeploying the previous version of the application.
        - If you're using the EC2/On-Premises compute platform, new instances are provisioned and contain the most up-to-date server configurations.
        - If you're using the AWS Lambda compute platform, you control how traffic is shifted from your original AWS Lambda function version to your new AWS Lambda function version.
- With AWS CodeDeploy, you can also deploy your applications to your on-premises data centers. Your on-premises instances will have a prefix of “mi-xxxxxxxxx”.


Developer Tools
CodeDeploy
Source • CodeCommit
Build • CodeBuild
Deploy • CodeDeploy
Getting started
Deployments
Applications
Deployment configurations
On-premises instances
Pipeline • CodePipeline
Go to resource
Feedback
Developer Tools > CodeDeploy > On-premises instances
On-premises instances
TutorialsDojo-Manila-On-Premises
1
Instance name
IAM ARN
Status
Not found
No results found for the following search: TutorialsDojo-Manila-On-Premises
Tutorials Dojo

AWS CodePipeline

- A fully managed **continuous delivery service** that helps you automate your release pipelines for application and infrastructure updates.
- You can easily integrate AWS CodePipeline with third-party services such as GitHub or with your own custom plugin.
- **Concepts**
    - A **pipeline** defines your release process workflow, and describes how a new code change progresses through your release process.
    - A pipeline comprises a series of **stages** (e.g., build, test, and deploy), which act as logical divisions in your workflow. Each stage is made up of a sequence of actions, which are tasks such as building code or deploying to test environments.
        - Pipelines must have **at least two stages**. The first stage of a pipeline is required to be a source stage, and the pipeline is required to additionally have at least one other stage that is a build or deployment stage.
    - Define your pipeline structure through a **declarative JSON** document that specifies your release workflow and its stages and actions. These documents enable you to update existing pipelines as well as provide starting templates for creating new pipelines.
    - A **revision** is a change made to the source location defined for your pipeline. It can include source code, build output, configuration, or data. A pipeline can have multiple revisions flowing through it at the same time.
    - A **stage** is a group of one or more actions. A pipeline can have two or more stages.
    - An **action** is a task performed on a revision. Pipeline actions occur in a specified order, in serial or in parallel, as determined in the configuration of the stage.
        - You can add actions to your pipeline that are in an AWS Region different from your pipeline.
        - There are six types of actions
            - Source
            - Build
            - Test
            - Deploy
            - Approval
            - Invoke
    - When an action runs, it acts upon a file or set of files called **artifacts**. These artifacts can be worked upon by later actions in the pipeline. You have an artifact store which is an S3 bucket in the same AWS Region as the pipeline to store items for all pipelines in that Region associated with your account.
    - The stages in a pipeline are connected by **transitions.** Transitions can be disabled or enabled between stages. If all transitions are enabled, the pipeline runs continuously.

    - An **approval action** prevents a pipeline from transitioning to the next action until permission is granted. This is useful when you are performing code reviews before code is deployed to the next stage.
- **Features**
    - AWS CodePipeline provides you with a graphical user interface to create, configure, and manage your pipeline and its various stages and actions.
    - A pipeline starts automatically (default) when a change is made in the source location, or when you manually start the pipeline. You can also set up a rule in CloudWatch to automatically start a pipeline when events you specify occur.
    - You can model your build, test, and deployment actions to run **in parallel** in order to increase your workflow speeds.
    - AWS CodePipeline can pull source code for your pipeline directly from AWS CodeCommit, GitHub, Amazon ECR, or Amazon S3.
    - It can run builds and unit tests in AWS CodeBuild.
    - It can deploy your changes using AWS CodeDeploy, AWS Elastic Beanstalk, Amazon ECS, AWS Fargate, Amazon S3, AWS Service Catalog, AWS CloudFormation, and/or AWS OpsWorks Stacks.
    - You can use the CodePipeline Jenkins plugin to easily register your existing build servers as a custom action.
    - When you use the console to create or edit a pipeline that has a GitHub source, CodePipeline creates a **webhook**. A webhook is an HTTP notification that detects events in another tool, such as a GitHub repository, and connects those external events to a pipeline. CodePipeline deletes your webhook when you delete your pipeline.
- As a best practice, when you use a Jenkins build provider for your pipeline's build or test action, install Jenkins on an Amazon EC2 instance and configure a separate EC2 instance profile. Make sure the instance profile grants Jenkins only the AWS permissions required to perform tasks for your project, such as retrieving files from Amazon S3.

AWS X-Ray

- AWS X-Ray analyzes and debugs production, distributed applications, such as those built using a microservices architecture. With X-Ray, you can identify performance bottlenecks, edge case errors, and other hard to detect issues.
- **Features**
    - AWS X-Ray can be used with applications running on Amazon EC2, Amazon ECS, AWS Lambda, AWS Elastic Beanstalk. You just integrate the X-Ray SDK with your application and install the X-Ray agent.
    - AWS X-Ray provides an end-to-end, cross-service, application-centric view of requests flowing through your application by aggregating the data gathered from individual services in your application into a single unit called a *trace*.
    - You can set the **trace sampling rate** that is best suited for your production applications or applications in development. X-Ray continually traces requests made to your application and stores a sampling of the requests for your analysis.
    - AWS X-Ray creates a map of services used by your application with trace data. This provides a view of connections between services in your application and aggregated data for each service, including average latency and failure rates. You can create dependency trees, perform cross-availability zone or region call detections, and more.
    - AWS X-Ray lets you add annotations to data emitted from specific components or services in your application.

AWS Application Services

Amazon SNS

- A web service that makes it easy to set up, operate, and send notifications from the cloud. SNS follows the **"publish-subscribe" (pub-sub) messaging** paradigm, with notifications being delivered to clients using a **"push"** mechanism rather than to periodically check or "poll" for new information and updates.

Features

- SNS is an **event-driven** computing hub that has native integration with a wide variety of AWS event sources (including EC2, S3, and RDS) and AWS event destinations (including SQS, and Lambda).
    - **Event-driven computing** is a model in which subscriber services automatically perform work in response to events triggered by publisher services. It can automate workflows while decoupling the services that collectively and independently work to fulfil these workflows.
- **Message filtering** allows a subscriber to create a filter policy, so that it only gets the notifications it is interested in.
- **Message fanout** occurs when a message is sent to a topic and then replicated and pushed to multiple endpoints. Fanout provides asynchronous event notifications, which in turn allows for parallel processing.
- **SNS mobile notifications** allows you to fanout mobile push notifications to iOS, Android, Fire OS, Windows and Baidu-based devices. You can also use SNS to fanout text messages (SMS) to 200+ countries and fanout email messages (SMTP).
- **Application and system alerts** are notifications, triggered by predefined thresholds, sent to specified users by SMS and/or email.
- **Push email** and **text messaging** are two ways to transmit messages to individuals or groups via email and/or SMS.

SNS provides simple APIs and easy integration with applications.

Publishers and Subscribers

- Publishers communicate asynchronously with subscribers by producing and sending a message to a topic, which is a logical access point and communication channel.
- Subscribers consume or receive the message or notification over one of the supported protocols when they are subscribed to the topic.
- Publishers create topics to send messages, while subscribers subscribe to topics to receive messages.
- SNS FIFO topics support the forwarding of messages to SQS FIFO queues. You can also use SNS to forward messages to standard queues.

SNS Topics

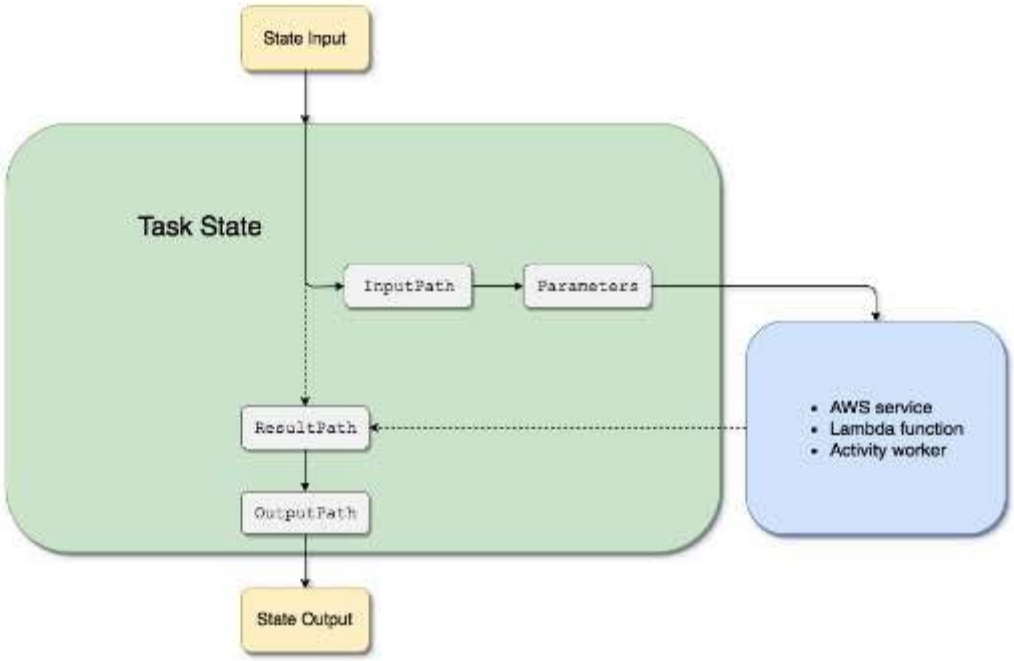
- Instead of including a specific destination address in each message, a publisher sends a message to a **topic**. SNS matches the topic to a list of subscribers who have subscribed to that topic, and delivers the message to each of those subscribers.
- Each topic has a unique name that identifies the SNS endpoint for publishers to post messages and subscribers to register for notifications.
- A topic can support subscriptions and notification deliveries over multiple transports.

System to System Messaging

- When a message is published to an SNS topic that has a **Lambda function** subscribed to it, the Lambda function is invoked with the payload of the published message. The Lambda function receives the message payload as an input parameter and can manipulate the information in the message, publish the message to other SNS topics, or send the message to other AWS services.
- When you subscribe a **SQS queue** to a SNS topic, you can publish a message to the topic and SNS sends a SQS message to the subscribed queue. The SQS message contains the subject and message that were published to the topic along with metadata about the message in a JSON document.
- When you subscribe to an **HTTP/s endpoint** to a topic, you can publish a notification to the topic and SNS sends an HTTP POST request delivering the contents of the notification to the subscribed endpoint. When you subscribe to the endpoint, you select whether SNS uses HTTP or HTTPS to send the POST request to the endpoint.

AWS Step Functions

- AWS Step Functions is a web service that provides **serverless orchestration** for modern applications. It enables you to coordinate the components of distributed applications and microservices using visual workflows.
- Concepts
    - Step Functions is based on the concepts of **tasks** and **state machines**.
        - A task performs work by using an activity or an AWS Lambda function, or by passing parameters to the API actions of other services.
        - A finite state machine can express an algorithm as a number of states, their relationships, and their input and output.
    - You define state machines using the **JSON-based Amazon States Language**.
    - A state is referred to by its *name*, which can be any string, but which must be unique within the scope of the entire state machine. An instance of a state exists until the end of its execution.
        - There are 6 types of states:
            - Task state - Do some work in your state machine. AWS Step Functions can invoke Lambda functions directly from a task state.
            - Choice state - Make a choice between branches of execution
            - Fail or Succeed state - Stop an execution with a failure or success
            - Pass state - Simply pass its input to its output or inject some fixed data
            - Wait state - Provide a delay for a certain amount of time or until a specified time/date
            - Parallel state - Begin parallel branches of execution
        - Common features between states
            - Each state must have a *Type* field indicating what type of state it is.
            - Each state can have an optional *Comment* field to hold a human-readable comment about, or description of, the state.
            - Each state (except a Succeed or Fail state) requires a *Next* field or, alternatively, can become a terminal state by specifying an ***End*** field.
    - **Activities** enable you to place a task in your state machine where the work is performed by an ***activity worker*** that can be hosted on Amazon EC2, Amazon ECS, or mobile devices.
    - Activity tasks let you assign a specific step in your workflow to code running in an activity worker. Service tasks let you connect a step in your workflow to a supported AWS service.
    - With **Transitions**, after executing a state, AWS Step Functions uses the value of the *Next* field to determine the next state to advance to. States can have multiple incoming transitions from other states.
    - Individual states receive JSON as input and usually pass JSON as output to the next state.



- A **state machine execution** occurs when a state machine runs and performs its tasks. Each Step Functions state machine can have multiple simultaneous executions.
- **State machine updates** in AWS Step Functions are **eventually consistent**.
- By default, when a state reports an error, AWS Step Functions causes the execution to **fail entirely**.
  - Task and Parallel states can have a field named *Retry* and *Catch* to retry an execution or to have a fallback state.
- The Step Functions console displays a graphical view of your state machine's structure, which provides a way to visually check a state machine's logic and monitor executions.

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.

CloudWatch Agent vs SSM Agent vs Custom Daemon Scripts

CloudWatch Agent	SSM Agent (AWS Systems Manager)	Custom Daemon Scripts
CloudWatch agent allows you to collect more system-level metrics from your EC2 and on-premises servers than just the standard CloudWatch metrics. It also enables you to retrieve custom metrics from your applications or services using the *StatsD* and *collectd* protocols. *StatsD* is supported on both Linux servers and servers running Windows Server. *collectd* is supported only on Linux servers. You can use CloudWatch agent to collect logs from your servers and send them to CloudWatch Logs. Metrics collected by the CloudWatch agent are billed as custom metrics. You can install CloudWatch Agent using three ways: • via Command Line • via SSM Agent • via AWS CloudFormation	SSM Agent is Amazon software that runs on your EC2 instances and your hybrid instances that are configured for Systems Manager. SSM Agent processes requests from the Systems Manager service in the cloud and configures your machine as specified in the request. You can manage servers without having to log in to them using automation. SSM Agent sends status and execution information back to the Systems Manager service by using the *EC2 Messaging* service. SSM Agent runs on Amazon EC2 instances using root permissions (Linux) or SYSTEM permissions (Windows). CloudWatch agent replaces SSM agent in sending metric logs to CloudWatch Logs.	You use custom scripts (such as cron or bash scripts) if the two previously mentioned agents do not fit your needs. CloudWatch agent is useful for collecting system-level metrics and logs. You can create custom scripts that perform some modifications before the metrics are sent out. SSM Agent is also useful for automation purposes, though Systems Manager does not have a document for every case scenario. You may also have some compliance requirements that would require SSM Agent to be disabled (recall that SSM agent runs at root level permissions).

EC2 Container Services ECS vs Lambda

Amazon EC2 Container Service (ECS)

- Amazon ECS is a highly scalable, high performance container management service that supports Docker containers and allows you to easily run applications on a managed cluster of Amazon EC2 instances. ECS eliminates the need for you to install, operate, and scale your own cluster management infrastructure.
- With ECS, deploying containerized applications is easily accomplished. This service fits well in running batch jobs or in a microservice architecture.
  You have a central repository where you can upload your Docker Images from ECS container for safekeeping called Amazon ECR.
- Applications in ECS can be written in a stateful or stateless matter.
- The Amazon ECS CLI supports Docker Compose, which allows you to simplify your local development experience as well as easily set up and run your containers on Amazon ECS.
- Since your applications still run on EC2 instances, server management is your responsibility. This gives you more granular control over your system.
- It is up to you to manage scaling and load balancing of your EC2 instances as well, unlike in AWS Lambda where functions scale automatically.
- You are charged for the costs incurred by your EC2 instances in your clusters. Most of the time, Amazon ECS costs more than using AWS Lambda since your active EC2 instances will be charged by the hour.
- One version of Amazon ECS, know as AWS Fargate, will fully manage your infrastructure so you can just focus on deploying containers. AWS Fargate has a different pricing model from the standard EC2 cluster.
- ECS will automatically recover unhealth containers to ensure that you have the desired number of containers supporting your application.

Tutorials Dojo

AWS Lambda

- AWS Lambda is a function-as-a-service offering that runs your code in response to events and automatically manages the compute resources for you, since Lambda is a serverless compute service. With Lambda, you do not have to worry about managing servers, and directly focus on your application code.
- Lambda automatically scales your function to meet demands. It is noteworthy, however, that Lambda has a maximum execution duration per request of 900 seconds or 15 minutes.
- To allow your Lambda function to access other services such as Cloudwatch Logs, you would need to create an execution role that has the necessary permissions to do so.
- You can easily integrate your function with different services such as API Gateway, DynamoDB, CloudFront, etc. using the Lambda console.
- You can test your function code locally in the Lambda console before launching it into production. Currently, Lambda supports only a number of programming languages such as Java, Go, PowerShell, Node.js, C#, Python, and Ruby. ECS is not limited by programming languages since it mainly caters to Docker.
- Lambda functions must be stateless since you do not have volumes for data storage.
- You are charged based on the number of requests for your functions and the duration, the time it takes for your code to execute. To minimize costs, you can throttle the number of concurrent executions running at a time, and the execution time limit of the function.
- With Lambda@Edge, AWS Lambda can run your code across AWS locations globally in response to Amazon CloudFront events, such as requests for content to or from origin servers and viewers.
  This makes it easier to deliver content to end users with lower latency.

EC2 Instance Health Check vs ELB Health Check vs Auto Scaling and Custom Health Check

EC2 instance health check

- Amazon EC2 performs automated checks on every running EC2 instance to identify hardware and software issues.
- Status checks are performed every minute and each returns a pass or a fail status.
  - If all checks pass, the overall status of the instance is OK.
  - If one or more checks fail, the overall status is impaired.
- Status checks are built into EC2, so they cannot be disabled or deleted.
- You can create or delete alarms that are triggered based on the result of the status checks.
- There are two types of status checks
  - **System Status Checks**
    - These checks detect underlying problems with your instance that require AWS involvement to repair. When a system status check fails, you can choose to wait for AWS to fix the issue, or you can resolve it yourself.
  - **Instance Status Checks**
    - Monitor the software and network configuration of your individual instance. Amazon EC2 checks the health of an instance by sending an address resolution protocol (ARP) request to the ENI. These checks detect problems that require your involvement to repair.

Tutorials Dojo

Elastic Load Balancer (ELB) health check

- To discover the availability of your registered EC2 instances, a load balancer periodically sends pings, attempts connections, or sends requests to test the EC2 instances.
- The status of the instances that are healthy at the time of the health check is InService. The status of any instances that are unhealthy at the time of the health check is OutOfService.
- When configuring a health check, you would need to provide the following:
  - a specific port
  - protocol to use
    - HTTP/HTTPS health check succeeds if the instance returns a 200 response code within the health check interval.
    - A TCP health check succeeds if the TCP connection succeeds.
    - An SSL health check succeeds if the SSL handshake succeeds.
  - ping path
- ELB health checks do not support WebSockets.
- The load balancer routes requests only to the healthy instances. When an instance becomes impaired, the load balancer resumes routing requests to the instance only when it has been restored to a healthy state.
- The load balancer checks the health of the registered instances using either
  - the default health check configuration provided by Elastic Load Balancing or
  - a health check configuration that you configure (auto scaling or custom health checks for example).
- Network Load Balancers use active and passive health checks to determine whether a target is available to handle requests.
  - With **active health checks**, the load balancer periodically sends a request to each registered target to check its status. After each health check is completed, the load balancer node closes the connection that was established.
  - With **passive health checks**, the load balancer observes how targets respond to connections, which enables it to detect an unhealthy target before it is reported as unhealthy by active health checks. You cannot disable, configure, or monitor passive health checks.

Auto Scaling and Custom health checks

- All instances in your Auto Scaling group start in the healthy state. Instances are assumed to be healthy unless EC2 Auto Scaling receives notification that they are unhealthy. This notification can come from one or more of the following sources:
  - Amazon EC2 (default)
  - Elastic Load Balancing
  - A custom health check.
- After Amazon EC2 Auto Scaling marks an instance as unhealthy, it is scheduled for replacement. If you do not want instances to be replaced, you can suspend the health check process for any individual Auto Scaling group.
- If an instance is in any state other than running or if the system status is impaired, Amazon EC2 Auto Scaling considers the instance to be unhealthy and launches a replacement instance.
- If you attached a load balancer or target group to your Auto Scaling group, Amazon EC2 Auto Scaling determines the health status of the instances by checking both the EC2 status checks and the Elastic Load Balancing health checks.
- Amazon EC2 Auto Scaling waits until the health check grace period ends before checking the health status of the instance. Ensure that the health check grace period covers the expected startup time for your application.
- Health check grace period does not start until lifecycle hook actions are completed and the instance enters the InService state.
- With custom health checks, you can send an instance's health information directly from your system to Amazon EC2 Auto Scaling.

Elastic Beanstalk vs CloudFormation vs OpsWorks vs CodeDeploy

AWS Elastic Beanstalk

- AWS Elastic Beanstalk makes it even easier for developers to **quickly deploy and manage applications** in the AWS Cloud. Developers simply upload their application, and Elastic Beanstalk **automatically handles the deployment details** of capacity provisioning, load balancing, auto-scaling, and application health monitoring.
- This **platform-as-a-service solution** is typically for those who want to deploy and manage their applications within minutes in the AWS Cloud without worrying about the underlying infrastructure.
- AWS Elastic Beanstalk supports the following languages and development stacks:
  - Apache Tomcat for Java applications
  - Apache HTTP Server for PHP applications
  - Apache HTTP Server for Python applications
  - Nginx or Apache HTTP Server for Node.js applications
  - Passenger or Puma for Ruby applications
  - Microsoft IIS for .NET applications
  - Java SE
  - Docker
  - Go
- Elastic Beanstalk also supports deployment versioning. It maintains a copy of older deployments so that it is easy for the developer to rollback any changes made on the application.

AWS CloudFormation

- AWS CloudFormation is a service that gives developers and businesses an easy way to create a **collection of related AWS resources** and provision them in an orderly and predictable fashion. This is typically known as **"infrastructure as code".**
- The main difference between CloudFormation and Elastic Beanstalk is that CloudFormation deals more with the AWS infrastructure rather than applications. AWS CloudFormation introduces two concepts:
  - The **template**, a JSON or YAML-format, text-based file that describes all the AWS resources and configurations you need to deploy to run your application.
  - The **stack**, which is the set of AWS resources that are created and managed as a single unit when AWS CloudFormation instantiates a template.
- CloudFormation also supports a rollback feature through template version controls. When you try to update your stack but the deployment failed midway,
- CloudFormation will automatically revert the changes back to their previous working states.
- CloudFormation supports Elastic Beanstalk application environments. This allows you, for example, to create and manage an AWS Elastic Beanstalk–hosted application along with an RDS database to store the application data.
- AWS CloudFormation can be used to bootstrap both Chef (Server and Client) and Puppet (Master and Client) softwares on your EC2 instances.
- CloudFormation also supports OpsWorks. You can now model OpsWorks components (stacks, layers, instances, and applications) inside CloudFormation templates, and provision them as CloudFormation stacks. This enables you to document, version control, and share your OpsWorks configuration.
- AWS CodeDeploy is a recommended adjunct to CloudFormation for managing the application deployments and updates.

AWS OpsWorks

- AWS OpsWorks is a configuration management service that provides managed instances of Chef and Puppet. OpsWorks lets you use **Chef** and **Puppet** to automate how servers are configured, deployed, and managed across your EC2 instances or on-premises compute environments.
- OpsWorks offers three services:
  - Chef Automate
  - Puppet Enterprise
  - OpsWorks Stacks
- OpsWorks for Puppet Enterprise lets you use Puppet to automate how nodes are configured, deployed, and managed, whether they are EC2 instances or on-premises devices.
- OpsWorks for Chef Automate lets you create AWS-managed Chef servers, and use the Chef DK and other Chef tooling to manage them.
- OpsWorks Stacks lets you create stacks that help you manage cloud resources in specialized groups called layers. A layer represents a set of EC2 instances that serve a particular purpose. Layers depend on **Chef recipes** to handle tasks such as installing packages on instances, deploying apps, and running scripts.
- Compared to CloudFormation, OpsWorks focuses more on orchestration and software configuration, and less on what and how AWS resources are procured.

AWS CodeDeploy

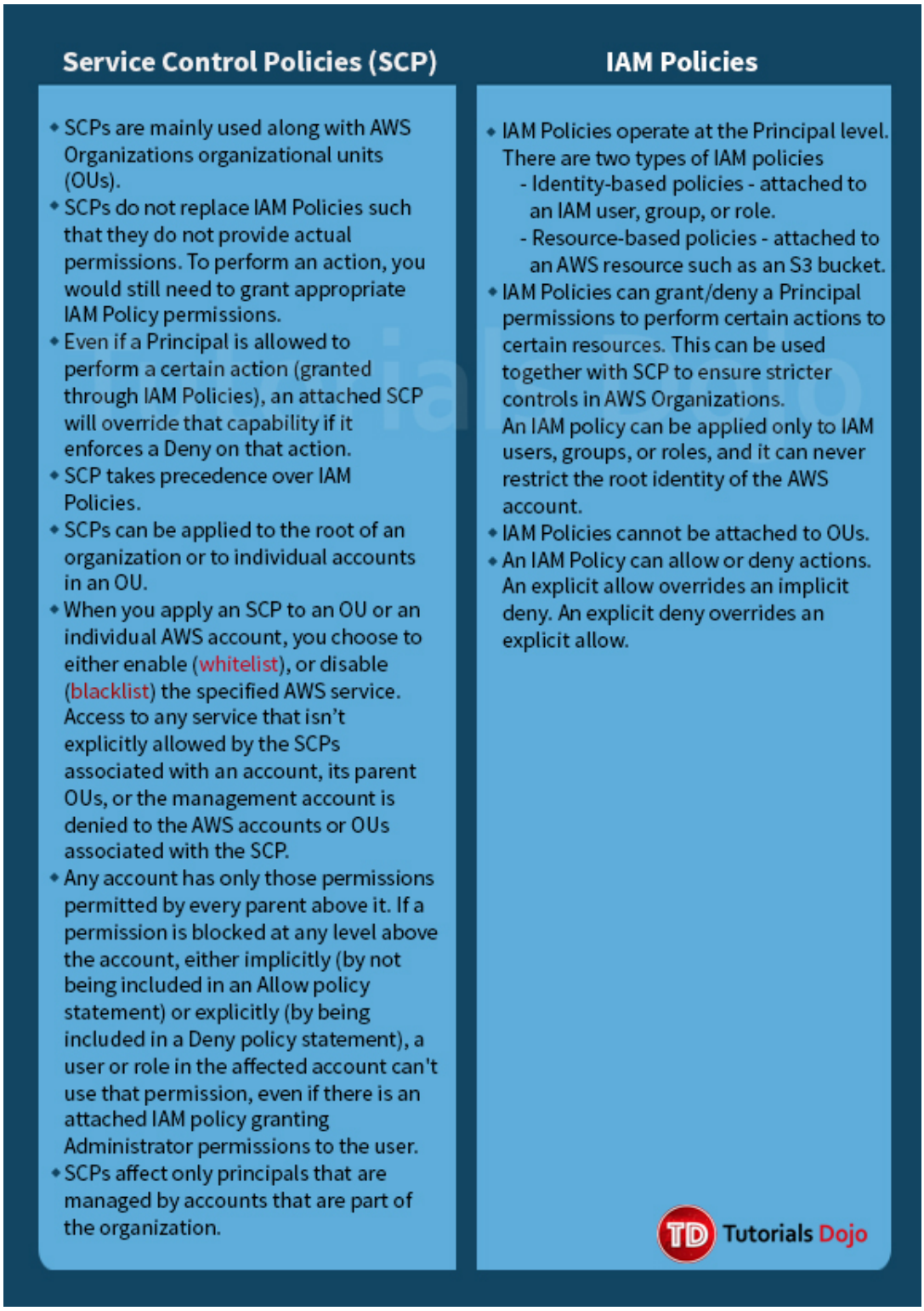
- AWS CodeDeploy is a service that coordinates application deployments across EC2 instances and instances running on-premises. It makes it easier for you to rapidly release new features, helps you avoid downtime during deployment, and handles the complexity of updating your applications.
- Unlike Elastic Beanstalk, CodeDeploy does not automatically handle capacity provisioning, scaling, and monitoring.
- Unlike CloudFormation and OpsWorks, CodeDeploy does not deal with infrastructure configuration and orchestration.
- AWS CodeDeploy is a building block service focused on helping developers deploy and update software on any instance, including EC2 instances and instances running on-premises. AWS Elastic Beanstalk and AWS OpsWorks are end-to-end application management solutions.
- You create a **deployment configuration file** to specify how deployments proceed/
- CodeDeploy complements CloudFormation well when deploying code to infrastructure that is provisioned and managed with CloudFormation.

Tutorials Dojo

Additional Notes:

- Elastic Beanstalk, CloudFormation, or OpsWorks are particularly useful for **blue-green deployment method** as they provide a simple way to clone your running application stack.
- CloudFormation and OpsWorks are best suited for the **prebaking AMIs**.
- CodeDeploy and OpsWorks are best suited for performing **in-place application upgrades**. For **disposable upgrades**, you can set up a cloned environment with Elastic Beanstalk, CloudFormation, and OpsWorks.

Service Control Policies vs IAM Policies



FINAL REMARKS AND TIPS

That's a wrap! Thank you once again for choosing our Study Guide and Cheat Sheets for the AWS Certified DevOps Engineer Professional (DOP-C01) exam. The Tutorials Dojo team spent considerable time and effort to produce this content to help you pass the AWS exam.

We also recommend that before taking the actual DOP-C01 exam, allocate some time to check your readiness by taking our **AWS practice test course** in the Tutorials Dojo Portal. You can also try the free sampler version of our full practice test course here. This will help you identify the topics that you need to improve on and help reinforce the concepts that you need to fully understand in order to pass this certification exam. It also has different training modes that you can choose from such as Timed mode, Review mode, Section-Based tests, and Final test plus bonus flashcards. In addition, you can read the technical discussions in our forums or post your queries if you have one. If you have any issues, concerns or constructive feedback on our eBook, feel free to contact us at support@tutorialsdojo.com.

On behalf of the Tutorials Dojo team, we wish you all the best on your upcoming AWS Certified DevOps Engineer Professional exam. May it help advance your career, as well as increase your earning potential.

With the right strategy, hard work, and unrelenting persistence, you can definitely make your dreams a reality! You can make it!

Sincerely,
Jon Bonso, Kenneth Samonte, and the Tutorials Dojo Team

ABOUT THE AUTHORS

Jon Bonso (8x AWS Certified)

Born and raised in the Philippines, Jon is the Co-Founder of **Tutorials Dojo**. Now based in Sydney, Australia, he has over a decade of diversified experience in Banking, Financial Services, and Telecommunications. He's 8x AWS Certified and has worked with various cloud services such as Google Cloud and Microsoft Azure. Jon is passionate about what he does and dedicates a lot of time creating educational courses. He has given IT seminars to different universities in the Philippines for free and has launched educational websites using his own money and without any external funding.

Kenneth Samonte (3x AWS Certified)

Kenneth is a registered Electronics Engineer and a Cloud Architect by profession. He's been tinkering with Linux servers and VMs since 2010 for his University degree. He's certified with AWS and Google Cloud platforms as well as Red Hat and VMware systems. When he's not busy typing away bash commands, you'll find him online playing League of Legends.