**Sure! Here's a list of various types of code and scripts that are commonly used in AWS cloud environments:**

1. **Infrastructure as Code (IaC):**
   * **AWS CloudFormation Templates: JSON or YAML scripts to provision and manage AWS resources.**
   * **AWS CDK (Cloud Development Kit): Allows you to define cloud infrastructure using familiar programming languages like TypeScript, Python, or Java.**
2. **Automation Scripts:**
   * **AWS CLI (Command Line Interface): Command-line tool to interact with AWS services.**
   * **AWS SDKs (Software Development Kits): Libraries available in various programming languages (e.g., Python, Java, Node.js) to interact with AWS services programmatically.**
3. **Serverless Computing:**
   * **AWS Lambda Functions: Code snippets that run in response to triggers and events without provisioning or managing servers.**
   * **AWS SAM (Serverless Application Model): An open-source framework for building serverless applications.**
4. **Containerization:**
   * **Dockerfiles: Text files that define a Docker image, which can be deployed and run on AWS ECS (Elastic Container Service) or EKS (Elastic Kubernetes Service).**
5. **Continuous Integration/Continuous Deployment (CI/CD):**
   * **AWS CodePipeline: Automates the build, test, and deployment phases of your release process.**
   * **AWS CodeBuild: Fully managed build service that compiles source code, runs tests, and produces software packages.**
   * **AWS CodeDeploy: Automates code deployments to any instance, including Amazon EC2 instances and AWS Lambda functions.**
6. **Monitoring and Logging:**
   * **AWS CloudWatch: Monitors AWS resources and applications, providing metrics, logs, and events.**
   * **AWS CloudTrail: Records API activity and events for auditing, compliance, and troubleshooting.**
7. **Security:**
   * **AWS IAM Policies: JSON documents that define permissions for AWS users, groups, and roles.**
   * **AWS WAF (Web Application Firewall) Rules: Rules that help protect web applications from common web exploits.**
8. **Data Processing:**
   * **AWS Glue ETL (Extract, Transform, Load): Prepares and transforms data for analytics.**
   * **AWS EMR (Elastic MapReduce): Hosts Hadoop clusters for processing large-scale data sets.**
9. **DevOps Tools:**
   * **AWS CodeCommit: Fully managed source control service.**
   * **AWS CodeArtifact: Fully managed artifact repository service.**
10. **Integration:**
    * **Amazon SQS (Simple Queue Service): Messaging queue service for decoupling and scaling microservices.**
    * **Amazon SNS (Simple Notification Service): Publish/subscribe messaging service for sending notifications.**

**These are just a few examples, as AWS provides a vast array of services and features that can be utilized with various types of code and scripts.**

**3.5**

**write codes for all the above Please write in English language.**

**ChatGPT**

**Certainly! Below are some example snippets of code for the various types of tasks mentioned earlier:**

1. **Infrastructure as Code (IaC):**
   * **AWS CloudFormation Template (YAML):**

**yaml**

**Copy code**

**Resources:**

**MyEC2Instance:**

**Type: AWS::EC2::Instance**

**Properties:**

**ImageId: ami-12345678**

**InstanceType: t2.micro**

* + **AWS CDK (TypeScript):**

**typescript**

**Copy code**

**import \* as ec2 from '@aws-cdk/aws-ec2';**

**const instance = new ec2.Instance(stack, 'MyInstance', {**

**instanceType: ec2.InstanceType.of(ec2.InstanceClass.T2, ec2.InstanceSize.MICRO),**

**machineImage: ec2.MachineImage.latestAmazonLinux(),**

**});**

1. **Automation Scripts:**
   * **AWS CLI (Bash):**

**bash**

**Copy code**

**aws ec2 describe-instances --region us-east-1**

* + **AWS SDK (Python):**

**python**

**Copy code**

**import boto3**

**ec2 = boto3.client('ec2', region\_name='us-east-1')**

**response = ec2.describe\_instances()**

**print(response)**

1. **Serverless Computing:**
   * **AWS Lambda Function (Python):**

**python**

**Copy code**

**import json**

**def lambda\_handler(event, context):**

**return {**

**'statusCode': 200,**

**'body': json.dumps('Hello from Lambda!')**

**}**

* + **AWS SAM Template (YAML):**

**yaml**

**Copy code**

**Resources:**

**MyFunction:**

**Type: AWS::Serverless::Function**

**Properties:**

**Handler: index.handler**

**Runtime: python3.8**

**CodeUri: .**

1. **Containerization:**
   * **Dockerfile:**

**Dockerfile**

**Copy code**

**FROM python:3.8**

**COPY . /app**

**WORKDIR /app**

**CMD ["python", "app.py"]**

1. **Continuous Integration/Continuous Deployment (CI/CD):**
   * **AWS CodePipeline Configuration:**

**yaml**

**Copy code**

**Stages:**

**- Name: Source**

**Actions:**

**- Name: SourceAction**

**ActionTypeId:**

**Category: Source**

**Owner: AWS**

**Version: 1**

**Configuration:**

**Repo: MyRepo**

**Branch: master**

1. **Monitoring and Logging:**
   * **AWS CloudWatch Logs (Python):**

**python**

**Copy code**

**import logging**

**logger = logging.getLogger()**

**logger.setLevel(logging.INFO)**

**logger.info("Hello from Python!")**

1. **Security:**
   * **AWS IAM Policy (JSON):**

**json**

**Copy code**

**{**

**"Version": "2012-10-17",**

**"Statement": [**

**{**

**"Effect": "Allow",**

**"Action": "s3:GetObject",**

**"Resource": "arn:aws:s3:::my-bucket/\*"**

**}**

**]**

**}**

1. **Data Processing:**
   * **AWS Glue ETL Job (Python):**

**python**

**Copy code**

**import sys**

**from awsglue.transforms import \***

**from awsglue.utils import getResolvedOptions**

**from pyspark.context import SparkContext**

**from awsglue.context import GlueContext**

**from awsglue.job import Job**

**sc = SparkContext()**

**glueContext = GlueContext(sc)**

**spark = glueContext.spark\_session**

**job = Job(glueContext)**

1. **DevOps Tools:**
   * **AWS CodeCommit (Bash):**

**bash**

**Copy code**

**git clone https://git-codecommit.us-east-1.amazonaws.com/v1/repos/MyRepo**

* + **AWS CodeArtifact (Python):**

**python**

**Copy code**

**import boto3**

**codeartifact = boto3.client('codeartifact', region\_name='us-east-1')**

1. **Integration:**
   * **Amazon SQS (Python):**

**python**

**Copy code**

**import boto3**

**sqs = boto3.client('sqs', region\_name='us-east-1')**

**queue\_url = 'https://sqs.us-east-1.amazonaws.com/123456789012/myqueue'**

**response = sqs.receive\_message(QueueUrl=queue\_url)**

**print(response)**

* + **Amazon SNS (Python):**

**python**

**Copy code**

**import boto3**

**sns = boto3.client('sns', region\_name='us-east-1')**

**response = sns.publish(**

**TopicArn='arn:aws:sns:us-east-1:123456789012:MyTopic',**

**Message='Hello from SNS!',**

**)**

**print(response)**

**These are simplified examples to illustrate the use of various AWS services with code. Actual implementations may require additional configuration and error handling.**

**3.5**

**more Please write in English language.**

**ChatGPT**

**Sure, let's continue with more examples:**

1. **Networking:**
   * **AWS VPC (Python):**

**python**

**Copy code**

**import boto3**

**ec2 = boto3.client('ec2', region\_name='us-east-1')**

**response = ec2.create\_vpc(**

**CidrBlock='10.0.0.0/16',**

**TagSpecifications=[**

**{**

**'ResourceType': 'vpc',**

**'Tags': [**

**{**

**'Key': 'Name',**

**'Value': 'MyVPC'**

**},**

**]**

**},**

**]**

**)**

**print(response)**

1. **Database:**
   * **Amazon RDS (Python):**

**python**

**Copy code**

**import boto3**

**rds = boto3.client('rds', region\_name='us-east-1')**

**response = rds.create\_db\_instance(**

**DBInstanceIdentifier='mydatabase',**

**DBInstanceClass='db.t2.micro',**

**Engine='mysql',**

**MasterUsername='admin',**

**MasterUserPassword='password123',**

**AllocatedStorage=20**

**)**

**print(response)**

1. **File Storage:**
   * **Amazon S3 (Python):**

**python**

**Copy code**

**import boto3**

**s3 = boto3.client('s3', region\_name='us-east-1')**

**response = s3.create\_bucket(**

**Bucket='my-bucket',**

**CreateBucketConfiguration={**

**'LocationConstraint': 'us-east-1'**

**}**

**)**

**print(response)**

1. **Machine Learning:**
   * **Amazon SageMaker (Python):**

**python**

**Copy code**

**import boto3**

**sagemaker = boto3.client('sagemaker', region\_name='us-east-1')**

**response = sagemaker.create\_training\_job(**

**TrainingJobName='my-training-job',**

**AlgorithmSpecification={**

**'TrainingImage': 'your-training-image',**

**'TrainingInputMode': 'File'**

**},**

**RoleArn='your-role-arn',**

**InputDataConfig=[**

**{**

**'ChannelName': 'train',**

**'DataSource': {**

**'S3DataSource': {**

**'S3DataType': 'S3Prefix',**

**'S3Uri': 's3://your-training-data',**

**'S3DataDistributionType': 'FullyReplicated'**

**}**

**}**

**},**

**],**

**OutputDataConfig={**

**'S3OutputPath': 's3://your-output-data'**

**},**

**ResourceConfig={**

**'InstanceType': 'ml.m4.xlarge',**

**'InstanceCount': 1,**

**'VolumeSizeInGB': 30**

**},**

**StoppingCondition={**

**'MaxRuntimeInSeconds': 86400**

**}**

**)**

**print(response)**

1. **Application Services:**
   * **Amazon SES (Python):**

**python**

**Copy code**

**import boto3**

**ses = boto3.client('ses', region\_name='us-east-1')**

**response = ses.send\_email(**

**Source='sender@example.com',**

**Destination={**

**'ToAddresses': [**

**'recipient@example.com',**

**]**

**},**

**Message={**

**'Subject': {**

**'Data': 'Test email'**

**},**

**'Body': {**

**'Text': {**

**'Data': 'This is a test email.'**

**}**

**}**

**}**

**)**

**print(response)**

**These examples cover a broader range of AWS services, showcasing how code can be used to interact with various components of the AWS ecosystem.**

**3.5**

**more Please write in English language.**

**ChatGPT**

**Certainly! Let's continue with more examples:**

1. **Content Delivery:**
   * **Amazon CloudFront (Python):**

**python**

**Copy code**

**import boto3**

**cloudfront = boto3.client('cloudfront', region\_name='us-east-1')**

**response = cloudfront.create\_distribution(**

**DistributionConfig={**

**'CallerReference': 'my-distribution',**

**'DefaultCacheBehavior': {**

**'TargetOriginId': 'my-origin',**

**'ForwardedValues': {**

**'QueryString': False,**

**'Cookies': {**

**'Forward': 'none'**

**}**

**},**

**'ViewerProtocolPolicy': 'allow-all'**

**},**

**'Origins': {**

**'Quantity': 1,**

**'Items': [**

**{**

**'Id': 'my-origin',**

**'DomainName': 'my-bucket.s3.amazonaws.com',**

**'S3OriginConfig': {**

**'OriginAccessIdentity': ''**

**}**

**}**

**]**

**},**

**'Enabled': True**

**}**

**)**

**print(response)**

1. **Identity and Access Management (IAM):**
   * **AWS IAM Role (Python):**

**python**

**Copy code**

**import boto3**

**iam = boto3.client('iam', region\_name='us-east-1')**

**response = iam.create\_role(**

**RoleName='MyRole',**

**AssumeRolePolicyDocument='{"Version":"2012-10-17","Statement":[{"Effect":"Allow","Principal":{"Service":"lambda.amazonaws.com"},"Action":"sts:AssumeRole"}]}'**

**)**

**print(response)**

1. **Monitoring and Logging:**
   * **AWS CloudWatch Events (Python):**

**python**

**Copy code**

**import boto3**

**cloudwatch = boto3.client('events', region\_name='us-east-1')**

**response = cloudwatch.put\_rule(**

**Name='MyRule',**

**ScheduleExpression='cron(0 20 \* \* ? \*)',**

**State='ENABLED'**

**)**

**print(response)**

1. **Security and Compliance:**
   * **AWS Config Rules (Python):**

**python**

**Copy code**

**import boto3**

**config = boto3.client('config', region\_name='us-east-1')**

**response = config.put\_config\_rule(**

**ConfigRule={**

**'ConfigRuleName': 'MyConfigRule',**

**'Source': {**

**'Owner': 'AWS',**

**'SourceIdentifier': 'S3\_BUCKET\_VERSIONING\_ENABLED'**

**},**

**'Scope': {**

**'ComplianceResourceTypes': [**

**'AWS::S3::Bucket'**

**]**

**},**

**'Description': 'Checks if versioning is enabled for S3 buckets'**

**}**

**)**

**print(response)**

1. **Developer Tools:**
   * **AWS CodeBuild (Python):**

**python**

**Copy code**

**import boto3**

**codebuild = boto3.client('codebuild', region\_name='us-east-1')**

**response = codebuild.start\_build(**

**projectName='MyProject'**

**)**

**print(response)**

**These additional examples cover more AWS services and functionalities, demonstrating how code can be used to interact with a wider range of AWS resources and features.**

**3.5**

**more Please write in English language.**

**ChatGPT**

**Of course! Here are some more examples:**

1. **Serverless Application Repository:**
   * **AWS SAM (Serverless Application Model) Template:**

**yaml**

**Copy code**

**AWSTemplateFormatVersion: '2010-09-09'**

**Transform: 'AWS::Serverless-2016-10-31'**

**Description: 'My Serverless Application'**

**Resources:**

**MyFunction:**

**Type: 'AWS::Serverless::Function'**

**Properties:**

**Handler: index.handler**

**Runtime: nodejs14.x**

**CodeUri: .**

1. **Database Migration Service:**
   * **AWS DMS (Database Migration Service) Task (Python):**

**python**

**Copy code**

**import boto3**

**dms = boto3.client('dms', region\_name='us-east-1')**

**response = dms.create\_replication\_task(**

**MigrationType='full-load',**

**SourceEndpointArn='source-endpoint-arn',**

**TargetEndpointArn='target-endpoint-arn',**

**ReplicationInstanceArn='replication-instance-arn',**

**MigrationTaskName='my-migration-task',**

**TableMappings='table-mappings-json'**

**)**

**print(response)**

1. **Content Delivery:**
   * **Amazon CloudFront Distribution (Python):**

**python**

**Copy code**

**import boto3**

**cloudfront = boto3.client('cloudfront', region\_name='us-east-1')**

**response = cloudfront.create\_distribution(**

**DistributionConfig={**

**'CallerReference': 'my-distribution',**

**'DefaultCacheBehavior': {**

**'TargetOriginId': 'my-origin',**

**'ForwardedValues': {**

**'QueryString': False,**

**'Cookies': {**

**'Forward': 'none'**

**}**

**},**

**'ViewerProtocolPolicy': 'allow-all'**

**},**

**'Origins': {**

**'Quantity': 1,**

**'Items': [**

**{**

**'Id': 'my-origin',**

**'DomainName': 'my-bucket.s3.amazonaws.com',**

**'S3OriginConfig': {**

**'OriginAccessIdentity': ''**

**}**

**}**

**]**

**},**

**'Enabled': True**

**}**

**)**

**print(response)**

1. **Machine Learning:**
   * **Amazon Comprehend (Python):**

**python**

**Copy code**

**import boto3**

**comprehend = boto3.client('comprehend', region\_name='us-east-1')**

**response = comprehend.detect\_sentiment(**

**Text='I love AWS Comprehend!',**

**LanguageCode='en'**

**)**

**print(response)**

1. **Application Services:**
   * **Amazon SQS (Python):**

**python**

**Copy code**

**import boto3**

**sqs = boto3.client('sqs', region\_name='us-east-1')**

**response = sqs.send\_message(**

**QueueUrl='queue-url',**

**MessageBody='Hello from SQS!'**

**)**

**print(response)**

**These examples further showcase the versatility of AWS services and how code can be used to interact with them for various tasks and applications.**

**3.5**

**more Please write in English language.**

**ChatGPT**

**Certainly! Here are a few more examples:**

1. **Streaming Data:**
   * **Amazon Kinesis Data Streams (Python):**

**python**

**Copy code**

**import boto3**

**kinesis = boto3.client('kinesis', region\_name='us-east-1')**

**response = kinesis.put\_record(**

**StreamName='my-stream',**

**Data='{"example": "data"}',**

**PartitionKey='partition-key'**

**)**

**print(response)**

1. **AI Services:**
   * **Amazon Rekognition (Python):**

**python**

**Copy code**

**import boto3**

**rekognition = boto3.client('rekognition', region\_name='us-east-1')**

**response = rekognition.detect\_labels(**

**Image={**

**'S3Object': {**

**'Bucket': 'my-bucket',**

**'Name': 'my-image.jpg',**

**}**

**},**

**MaxLabels=10,**

**MinConfidence=90**

**)**

**print(response)**

1. **IoT:**
   * **AWS IoT Core (Python):**

**python**

**Copy code**

**import boto3**

**iot = boto3.client('iot', region\_name='us-east-1')**

**response = iot.publish(**

**topic='my-topic',**

**payload='{"message": "Hello from IoT!"}'**

**)**

**print(response)**

1. **Quantum Computing:**
   * **Amazon Braket (Python):**

**python**

**Copy code**

**import boto3**

**braket = boto3.client('braket', region\_name='us-east-1')**

**response = braket.create\_quantum\_task(**

**action='my-action',**

**quantumTaskArn='my-quantum-task-arn'**

**)**

**print(response)**

1. **Blockchain:**
   * **Amazon Managed Blockchain (Python):**

**python**

**Copy code**

**import boto3**

**managedblockchain = boto3.client('managedblockchain', region\_name='us-east-1')**

**response = managedblockchain.create\_network(**

**Name='my-network',**

**Framework='HYPERLEDGER\_FABRIC',**

**FrameworkVersion='1.2'**

**)**

**print(response)**

**These examples cover advanced AWS services for various specialized tasks, showcasing how code can interact with cutting-edge technologies within the AWS ecosystem.**

**3.5**

**more Please write in English language.**

**ChatGPT**

**Certainly! Here are a few more examples:**

1. **Augmented Reality and Virtual Reality:**
   * **Amazon Sumerian (Python):**

**python**

**Copy code**

**import boto3**

**sumerian = boto3.client('sumerian', region\_name='us-east-1')**

**response = sumerian.create\_scene(**

**name='my-scene',**

**description='My first Sumerian scene'**

**)**

**print(response)**

1. **Media Services:**
   * **Amazon Elastic Transcoder (Python):**

**python**

**Copy code**

**import boto3**

**transcoder = boto3.client('elastictranscoder', region\_name='us-east-1')**

**response = transcoder.create\_job(**

**PipelineId='my-pipeline',**

**Input={**

**'Key': 'my-input-file.mp4'**

**},**

**Output={**

**'Key': 'my-output-file.mp4',**

**'PresetId': 'my-preset'**

**}**

**)**

**print(response)**

1. **Game Development:**
   * **Amazon GameLift (Python):**

**python**

**Copy code**

**import boto3**

**gamelift = boto3.client('gamelift', region\_name='us-east-1')**

**response = gamelift.create\_game\_session(**

**FleetId='my-fleet',**

**MaximumPlayerSessionCount=10**

**)**

**print(response)**

1. **AR and 3D:**
   * **Amazon Sumerian (Python):**

**python**

**Copy code**

**import boto3**

**sumerian = boto3.client('sumerian', region\_name='us-east-1')**

**response = sumerian.create\_scene(**

**name='my-scene',**

**description='My first Sumerian scene'**

**)**

**print(response)**

1. **Customer Engagement:**
   * **Amazon Pinpoint (Python):**

**python**

**Copy code**

**import boto3**

**pinpoint = boto3.client('pinpoint', region\_name='us-east-1')**

**response = pinpoint.create\_campaign(**

**ApplicationId='my-application',**

**WriteCampaignRequest={**

**'Name': 'my-campaign',**

**'MessageConfiguration': {**

**'DefaultPushNotificationMessage': {**

**'Body': 'Hello from Amazon Pinpoint!'**

**}**

**}**

**}**

**)**

**print(response)**

**These examples showcase AWS services catering to specific industries or specialized use cases, demonstrating how code can interact with various AWS offerings to build comprehensive solutions.**

**3.5**

**how to find all this code in aws console with examples and steps Please write in English language.**

**ChatGPT**

**In the AWS Management Console, you can access various services and resources to deploy and manage your infrastructure, applications, and data. Here's how you can find code examples and resources for different AWS services:**

1. **AWS Documentation:**
   * **Visit the** [**AWS Documentation**](https://docs.aws.amazon.com/index.html) **website.**
   * **Navigate to the service you're interested in using the search bar or the services menu.**
   * **AWS Documentation provides detailed guides, tutorials, and code examples for each service.**
2. **AWS Developer Guides:**
   * **In the AWS Documentation, explore the "Developer Guides" section for each service.**
   * **These guides offer comprehensive explanations of service features, best practices, and code examples in various programming languages.**
3. **AWS Samples GitHub Repository:**
   * **Visit the** [**AWS Samples GitHub Repository**](https://github.com/aws-samples) **for curated samples and code examples across different services.**
   * **You can browse repositories by service or search for specific topics.**
4. **AWS Solutions Constructs:**
   * **AWS Solutions Constructs is an open-source extension of the AWS Cloud Development Kit (CDK) that provides pre-built architecture patterns for common use cases.**
   * **Visit the** [**AWS Solutions Constructs GitHub Repository**](https://github.com/awslabs/aws-solutions-constructs) **for reusable, well-architected patterns and examples.**
5. **AWS Blogs and Forums:**
   * **Explore the** [**AWS Blog**](https://aws.amazon.com/blogs/) **for articles, case studies, and announcements about AWS services.**
   * **Participate in the** [**AWS Developer Forums**](https://forums.aws.amazon.com/) **to ask questions, share knowledge, and learn from the community.**
6. **AWS Management Console:**
   * **Some AWS services offer built-in code examples and templates directly within the AWS Management Console.**
   * **For example, AWS Lambda, AWS Step Functions, AWS CloudFormation, and AWS Glue provide templates and sample code snippets to help you get started.**
7. **AWS Marketplace:**
   * **Explore the AWS Marketplace for third-party solutions, tools, and services that integrate with AWS.**
   * **Some offerings include sample code, tutorials, and documentation to help you leverage their products.**
8. **AWS Training and Certification:**
   * **Take advantage of the free and paid training courses offered by AWS to learn about different services and their use cases.**
   * **Visit the** [**AWS Training and Certification**](https://aws.amazon.com/training/) **page for on-demand courses, workshops, and certification programs.**

**By leveraging these resources, you can find comprehensive documentation, code examples, and best practices for utilizing various AWS services to build and manage your cloud infrastructure and applications.**

what are the different tools used for aws cloud pen testing ans security

ChatGPT

There are several tools and services available for conducting penetration testing (pen testing) and enhancing security in AWS cloud environments. Here are some of the commonly used ones:

1. **AWS Inspector**:
   * AWS Inspector helps to analyze the behavior of your AWS resources and applications to identify potential security vulnerabilities and deviations from security best practices.
2. **Amazon GuardDuty**:
   * Amazon GuardDuty is a managed threat detection service that continuously monitors for malicious activity and unauthorized behavior in your AWS accounts, workloads, and data stored in AWS.
3. **AWS Security Hub**:
   * AWS Security Hub provides a comprehensive view of your security posture across multiple AWS accounts, integrates with other security tools, and helps you automate security checks and compliance checks.
4. **AWS Config**:
   * AWS Config provides detailed information about the configuration of AWS resources in your account. It helps you assess, audit, and evaluate the configurations for compliance with security policies and industry standards.
5. **AWS Identity and Access Management (IAM)**:
   * IAM enables you to manage access to AWS services and resources securely. It allows you to create and manage users, groups, roles, and permissions to control who can access what resources.
6. **AWS CloudTrail**:
   * AWS CloudTrail records API activity and events for your AWS account, providing visibility into actions taken by users, roles, and services. It helps with security analysis, compliance auditing, and troubleshooting.
7. **Third-Party Security Tools**:
   * There are many third-party security tools and solutions available on the AWS Marketplace and beyond that offer advanced capabilities for vulnerability scanning, threat detection, intrusion detection and prevention, and security analytics.
8. **Network Security Tools**:
   * Tools like Nmap, Wireshark, and Burp Suite can be used for network scanning, packet analysis, and web application testing in AWS environments.
9. **Container Security Tools**:
   * For securing containerized workloads running on AWS ECS (Elastic Container Service) or EKS (Elastic Kubernetes Service), tools like Docker Bench for Security, Clair, and Twistlock provide vulnerability scanning, runtime protection, and compliance checks.
10. **Serverless Security Tools**:
    * Tools such as AWS Lambda Powertools, AWS SAM (Serverless Application Model) Local, and Snyk can help ensure the security of serverless applications by detecting vulnerabilities, enforcing best practices, and monitoring function invocations.

When conducting penetration testing in AWS environments, it's essential to adhere to AWS guidelines and policies, obtain proper authorization, and ensure that testing activities do not disrupt production services or violate terms of service. Additionally, it's crucial to stay updated on the latest security threats, vulnerabilities, and mitigation techniques specific to cloud environments.

3.5

Show with examples and steps

ChatGPT

Certainly! Here are examples and steps for using some of the tools mentioned for AWS cloud penetration testing and security:

1. **AWS Inspector**:
   * **Steps**:
     1. Sign in to the AWS Management Console.
     2. Open the AWS Inspector console.
     3. Create an assessment target by specifying the EC2 instances or applications you want to assess.
     4. Configure assessment templates to define the rules and checks for security vulnerabilities.
     5. Start an assessment run to analyze the selected resources for security issues.
     6. Review the findings generated by AWS Inspector and take necessary actions to remediate vulnerabilities.
2. **Amazon GuardDuty**:
   * **Steps**:
     1. Sign in to the AWS Management Console.
     2. Open the Amazon GuardDuty console.
     3. Enable GuardDuty in your AWS account and specify the regions to monitor.
     4. Configure threat detection settings and set up cloud trail event logging.
     5. Monitor GuardDuty findings in the console, set up alerts, or integrate with AWS CloudWatch for automated response actions.
3. **AWS Security Hub**:
   * **Steps**:
     1. Sign in to the AWS Management Console.
     2. Open the AWS Security Hub console.
     3. Enable Security Hub for your AWS account.
     4. Set up automated compliance checks, security controls, and best practice standards.
     5. Review security findings and prioritize remediation actions based on severity and impact.
4. **AWS Identity and Access Management (IAM)**:
   * **Steps**:
     1. Sign in to the AWS Management Console.
     2. Open the IAM console.
     3. Create IAM users, groups, and roles with least privilege access.
     4. Define IAM policies to grant or deny permissions for specific AWS resources and actions.
     5. Implement multi-factor authentication (MFA) and password policies for IAM users.
     6. Regularly review IAM policies, users, and roles to ensure compliance with security requirements.
5. **AWS CloudTrail**:
   * **Steps**:
     1. Sign in to the AWS Management Console.
     2. Open the CloudTrail console.
     3. Create a new CloudTrail trail to record API activity and events in your AWS account.
     4. Specify the S3 bucket to store CloudTrail logs and optionally enable log file validation.
     5. Configure CloudTrail event selectors to capture specific API actions and resources.
     6. Analyze CloudTrail logs using Amazon Athena, Amazon CloudWatch Logs Insights, or third-party log analysis tools.

These steps provide a high-level overview of how to use these AWS security tools. Each tool may require further configuration and customization based on your specific security requirements and compliance standards. Always refer to the official AWS documentation and best practices for detailed instructions and guidance.

3.5

Amazon Inspector Continual vulnerability management at scale Console Home View resource insights, service shortcuts, and feature updates Show with examples and steps AWS Private Certificate Authority Managed private certificate authority service AWS Private CA Connector for Active Directory AWS Signer Ensuring trust and integrity of your code

ChatGPT

Show with examples and steps amazon Inspector Continual vulnerability management at scale View resource insights, service shortcuts, and feature updates Show with examples and steps AWS Private Certificate Authority Managed private certificate authority service AWS Private CA Connector for Active Directory AWS Signer Ensuring trust and integrity of your code

ChatGPT

Certainly! Let's start with Amazon Inspector and then move on to AWS Private Certificate Authority and AWS Signer.

**Amazon Inspector:** Amazon Inspector is a service that helps you to improve the security and compliance of your applications deployed on AWS. It provides automated security assessments of your AWS resources, identifying security vulnerabilities and deviations from best practices. Here's how you can use it:

1. **Sign in to the AWS Management Console**:
   * Go to the AWS Management Console at <https://console.aws.amazon.com/>.
2. **Navigate to Amazon Inspector**:
   * In the AWS Management Console, use the search bar at the top to find "Amazon Inspector" or navigate to it under the "Security, Identity, & Compliance" section.
3. **Create an Assessment Target**:
   * Click on "Assessment targets" in the left navigation pane.
   * Click on the "Create assessment target" button.
   * Provide a name and description for your assessment target.
   * Select the EC2 instances or applications you want to assess.
4. **Create an Assessment Template**:
   * Click on "Assessment templates" in the left navigation pane.
   * Click on the "Create assessment template" button.
   * Choose the rules packages you want to include in the assessment template (e.g., Common Vulnerabilities and Exposures, CIS Benchmarks).
   * Configure assessment duration and other settings as needed.
5. **Start an Assessment Run**:
   * Click on "Assessment runs" in the left navigation pane.
   * Click on the "Start assessment run" button.
   * Select the assessment template you created earlier.
   * Review the settings and start the assessment run.
6. **Review Assessment Findings**:
   * Once the assessment run is complete, click on the assessment run ID to view the findings.
   * Amazon Inspector will provide detailed findings categorized by severity levels (e.g., high, medium, low).
   * Review the findings and take necessary actions to remediate security vulnerabilities and improve your security posture.

**AWS Private Certificate Authority:** AWS Private Certificate Authority (CA) is a managed service that enables you to easily and securely manage the lifecycle of private certificates. It helps you to create, manage, and distribute private certificates within your organization. Here's how to use it:

1. **Sign in to the AWS Management Console**:
   * Go to the AWS Management Console at <https://console.aws.amazon.com/>.
2. **Navigate to AWS Certificate Manager (ACM)**:
   * In the AWS Management Console, use the search bar at the top to find "Certificate Manager" or navigate to it under the "Security, Identity, & Compliance" section.
3. **Create a Private CA**:
   * Click on "Private CAs" in the left navigation pane.
   * Click on the "Create private CA" button.
   * Provide a name and description for your private CA.
   * Configure certificate subject settings, key algorithm, and key size.
4. **Set Up Certificate Revocation List (CRL) Distribution**:
   * Configure CRL distribution settings to specify where the CRL will be published and how it will be accessed.
5. **Request and Issue Certificates**:
   * Click on "Certificate signing requests (CSRs)" in the left navigation pane.
   * Upload or generate a CSR for the certificate you want to request.
   * Submit the CSR to your private CA for signing.
   * Once the certificate is issued, download it and deploy it to your applications or devices.

**AWS Signer:** AWS Signer is a service that helps you to ensure the trust and integrity of your code by digitally signing it. It provides a way to sign code artifacts such as AWS Lambda functions, Docker images, and IoT device firmware. Here's how to use it:

1. **Sign in to the AWS Management Console**:
   * Go to the AWS Management Console at <https://console.aws.amazon.com/>.
2. **Navigate to AWS Signer**:
   * In the AWS Management Console, use the search bar at the top to find "Signer" or navigate to it under the "Security, Identity, & Compliance" section.
3. **Create a Signing Profile**:
   * Click on "Signing profiles" in the left navigation pane.
   * Click on the "Create signing profile" button.
   * Provide a name and description for your signing profile.
   * Choose the signing algorithm and key to be used for signing.
4. **Upload Code Artifacts**:
   * Upload the code artifacts (e.g., Lambda function package, Docker image) that you want to sign to an Amazon S3 bucket.
5. **Create a Signing Job**:
   * Click on "Signing jobs" in the left navigation pane.
   * Click on the "Create signing job" button.
   * Select the signing profile you created earlier.
   * Specify the S3 bucket and object key of the code artifact to be signed.
   * Review the settings and create the signing job.
6. **Download Signed Code Artifacts**:
   * Once the signing job is complete, download the signed code artifact from the specified location in Amazon S3.
   * Deploy the signed code artifact to your applications or devices, ensuring the trust and integrity of your code.

These steps provide a guide on how to use Amazon Inspector, AWS Private Certificate Authority, and AWS Signer to enhance security and manage certificates in your AWS environment.

[**https://us-east-1.console.aws.amazon.com/iam/home?region=us-east-1#/users/details/Dharam?section=security\_credentials**](https://us-east-1.console.aws.amazon.com/iam/home?region=us-east-1#/users/details/Dharam?section=security_credentials)

<https://www.youtube.com/watch?v=HaRki0xPPd4&list=PL6XT0grm_TfgtwtwUit305qS-HhDvb4du&index=13>

<https://explore.skillbuilder.aws/pages/16/learner-dashboard>

<https://github.com/aws-samples>

<https://repost.aws/>

[https://aws.amazon.com/blogs/?awsf.blog-master-category=\*all&awsf.blog-master-learning-levels=\*all&awsf.blog-master-industry=\*all&awsf.blog-master-analytics-products=\*all&awsf.blog-master-artificial-intelligence=\*all&awsf.blog-master-aws-cloud-financial-management=\*all&awsf.blog-master-blockchain=\*all&awsf.blog-master-business-applications=\*all&awsf.blog-master-compute=\*all&awsf.blog-master-customer-enablement=\*all&awsf.blog-master-customer-engagement=\*all&awsf.blog-master-database=\*all&awsf.blog-master-developer-tools=\*all&awsf.blog-master-devops=\*all&awsf.blog-master-end-user-computing=\*all&awsf.blog-master-mobile=\*all&awsf.blog-master-iot=\*all&awsf.blog-master-management-governance=\*all&awsf.blog-master-media-services=\*all&awsf.blog-master-migration-transfer=\*all&awsf.blog-master-migration-solutions=\*all&awsf.blog-master-networking-content-delivery=\*all&awsf.blog-master-programming-language=\*all&awsf.blog-master-sector=\*all&awsf.blog-master-security=\*all&awsf.blog-master-storage=\*all](https://aws.amazon.com/blogs/?awsf.blog-master-category=*all&awsf.blog-master-learning-levels=*all&awsf.blog-master-industry=*all&awsf.blog-master-analytics-products=*all&awsf.blog-master-artificial-intelligence=*all&awsf.blog-master-aws-cloud-financial-management=*all&awsf.blog-master-blockchain=*all&awsf.blog-master-business-applications=*all&awsf.blog-master-compute=*all&awsf.blog-master-customer-enablement=*all&awsf.blog-master-customer-engagement=*all&awsf.blog-master-database=*all&awsf.blog-master-developer-tools=*all&awsf.blog-master-devops=*all&awsf.blog-master-end-user-computing=*all&awsf.blog-master-mobile=*all&awsf.blog-master-iot=*all&awsf.blog-master-management-governance=*all&awsf.blog-master-media-services=*all&awsf.blog-master-migration-transfer=*all&awsf.blog-master-migration-solutions=*all&awsf.blog-master-networking-content-delivery=*all&awsf.blog-master-programming-language=*all&awsf.blog-master-sector=*all&awsf.blog-master-security=*all&awsf.blog-master-storage=*all)

# AWS Tutorials - 10

<https://www.youtube.com/watch?v=bpjtOFoOiVw&list=PLisYSIHvE9qNqE21etGZgzLdKMErFS1OJ>

<https://www.youtube.com/watch?v=B8i49C8fC3E>

<https://www.youtube.com/watch?v=B8i49C8fC3E>

<https://www.aws.training/SessionSearch?pageNumber=1&courseId=10012>

<https://explore.skillbuilder.aws/learn/mycourses>

<https://explore.skillbuilder.aws/learn/course/1851/play/85986/aws-technical-essentials>

<https://www.udemy.com/course/aws-certified-solutions-architect-associate-step-by-step/?couponCode=ST19MT60324>

<https://www.udemy.com/course/aws-solutions-architect-professional/?couponCode=ST19MT60324>

<https://www.youtube.com/live/0Rgt4DIoDjI>

<https://www.twitch.tv/videos/2160148494>

<https://www.youtube.com/watch?v=lae0mfHBSjk>

<https://www.youtube.com/playlist?list=PLz8UpOu_f4zpr2ojgGWS07VuyubPEjR9i>

2 jUNE 9.00 A.M

#### [Virtual Location3 June, 2024 @ 09:00 AM](https://www.koenig-solutions.com/architecting-aws-training-course" \l "schedulec" \t "_blank)

[Virtual Classroom Live3 daysEnglishTime zone: Asia / Kolkata (UTC+05:30)](https://www.koenig-solutions.com/architecting-aws-training-course" \l "schedulec" \t "_blank)

[Offered by: Koenig Solutions Pvt. Ltd.](https://www.koenig-solutions.com/architecting-aws-training-course" \l "schedulec" \t "_blank)

<https://aws.amazon.com/marketplace/professionalservices/procurement?productId=prod-orf3i7gtn7dbw>

<https://explore.skillbuilder.aws/learn>

<https://docs.aws.amazon.com/cli/v1/userguide/install-windows.html>

<https://explore.skillbuilder.aws/pages/56/aws-ramp-up-guides>

<https://explore.skillbuilder.aws/learn/course/199/play;state=%5Bobject%20Object%5D;autoplay=0>

<https://explore.skillbuilder.aws/pages/16/learner-dashboard>

<https://explore.skillbuilder.aws/pages/16/learner-dashboard>

<https://explore.skillbuilder.aws/learn/course/17763/play/93257/foundations-of-prompt-engineering>

[https://aws.amazon.com/architecture/?cards-all.sort-by=item.additionalFields.sortDate&cards-all.sort-order=desc&awsf.content-type=\*all&awsf.methodology=\*all&awsf.tech-category=\*all&awsf.industries=\*all](https://aws.amazon.com/architecture/?cards-all.sort-by=item.additionalFields.sortDate&cards-all.sort-order=desc&awsf.content-type=*all&awsf.methodology=*all&awsf.tech-category=*all&awsf.industries=*all)

<https://aws.amazon.com/architecture/icons/>

Sign-in URL for IAM users in this account

<https://944346522085.signin.aws.amazon.com/console>

<https://docs.aws.amazon.com/>

Account ID

944346522085

Account Alias

Create

n this course, the Morgan and Raf cover some of the AWS data services that can be used for a solution that ingests clickstream data. Clickstream is the term for small events that contain pieces of data that are generated continuously with high speed and volume. Remember, clickstream data is usually data that’s collected by systems (mostly frontends) regarding user interactions with that system. In this week’s scenario, clickstream data will be used to collect information about users’ behaviors when they interact with a restaurant menu. Regardless of whether you are architecting for clickstream data, AWS provides various analytics services that meet your needs for data analytics. Organizations of all sizes and industries can use these services to reinvent their business with data.

## AWS data services

From data movement, data storage, data lakes, big data analytics, log analytics, streaming analytics, business intelligence, and machine learning (ML) to many things in between, AWS offers purpose-built services that provide price performance, scalability, and low costs. [Data Lakes and Analytics on AWS](https://aws.amazon.com/big-data/datalakes-and-analytics/) lists all the AWS services that can be used for data analytics, and places the services into four distinct categories. This reading summarizes the most popular services in each category.

## Data lakes and data storage

#### Amazon S3

Amazon Simple Storage Service (Amazon S3) is an object storage service that offers scalability, data availability, security, and performance. Customers of all sizes and industries can store and protect virtually any amount of data for virtually any use case, such as data lakes, cloud-native applications, and mobile apps. With cost-effective storage classes and easy-to-use management features, you can optimize costs, organize data, and configure fine-tuned access controls to meet specific business, organizational, and compliance requirements.  
The following list details some use cases for Amazon S3:

* Archive data at the lowest cost: Move data archives to the Amazon Simple Storage Service Glacier (Amazon S3 Glacier storage classes to lower costs, reduce operational complexities, and gain new insights.
* Run cloud-native applications: Build fast, powerful, mobile and web-based cloud-native applications that scale automatically in a highly available configuration, such as static websites that use the client side for coding.
* Build a data lake: Run big data analytics, artificial intelligence (AI), machine learning (ML), and high performance computing (HPC) applications to unlock data insights.
* Back up and restore critical data: Meet Recovery Time Objectives (RTO), Recovery Point Objectives (RPO), and compliance requirements with the robust replication features of Amazon S3.

For more information, see [Amazon S3](https://aws.amazon.com/s3/).

#### Amazon S3 Glacier

**Note:** Amazon S3 storage classes will have their own dedicated reading later in this week.  
  
The Amazon S3 Glacier storage classes are purpose-built for data archiving. They are designed to provide you with high performance, retrieval flexibility, and low-cost archive storage in the cloud. All S3 Glacier storage classes provide virtually unlimited scalability and are designed for 99.999999999 percent (11 nines) of data durability. In addition to low-cost storage, the S3 Glacier storage classes also deliver options for fast access to your archival data.  
  
For more information, see [Amazon S3 Glacier storage classes](https://aws.amazon.com/s3/storage-classes/glacier/).

#### AWS Lake Formation

AWS Lake Formation is a service that you can use to set up a secure data lake in days. A data lake is a centralized, curated, and secured repository that stores all your data, both in its original form and prepared for analysis. You can use a data lake to break down data silos and combine different types of analytics to gain insights and guide better business decisions.  
  
For more information, see [AWS Lake Formation](https://aws.amazon.com/lake-formation/?whats-new-cards.sort-by=item.additionalFields.postDateTime&whats-new-cards.sort-order=desc).

## Data analytics

#### Amazon Athena

Amazon Athena is an interactive query service that you can use to analyze data in Amazon S3 by using standard Structured Query Language (SQL). Athena is serverless, so you don’t need to manage infrastructure, and you pay only for the queries that you run.  
  
Using Athena is straightforward. You point to your data in Amazon S3, define the schema, and start querying by using standard SQL. Most results are delivered within seconds. With Athena, you don’t need complex extract, transform, and load (ETL) jobs to prepare your data for analysis. Anyone with SQL skills can use Athena to quickly analyze large-scale datasets.  
  
For more information, see [Amazon Athena](https://aws.amazon.com/athena/).

#### Amazon EMR

Amazon EMR is a big data solution for petabyte-scale data processing, interactive analytics, and machine learning that use open-source frameworks, such as Apache Spark, Apache Hive, and Presto.  
  
In this course, we didn’t use or explore Amazon EMR for this week’s scenario because the customer was short-staffed, and Amazon EMR requires a learning curve to operate the open-source frameworks that it uses. In fact, Amazon EMR has so many features that we could make an entire 4-week course about it! The following list details some of the use cases for Amazon EMR. (Notice that the third item aligns well with the week’s scenario—Amazon EMR would a good candidate for the task if the customer already had big data knowledge.)

* Run large-scale data processing and what-if analysis by using statistical algorithms and predictive models to uncover hidden patterns, correlations, market trends, and customer preferences.
* Extract data from various sources, process it at scale, and make the data available for applications and users.
* Analyze events from streaming data sources in real time to create long-running, highly available, and fault-tolerant streaming data pipelines.
* Analyze data using open-source ML frameworks, such as Apache Spark MLlib, TensorFlow, and Apache MXNet.
* Connect to Amazon SageMaker Studio for large-scale model training, analysis, and reporting.

For more information, see [Amazon EMR](https://aws.amazon.com/emr/?c=a&sec=srv).

#### Amazon OpenSearch Service

You can use Amazon OpenSearch Service to perform interactive log analytics, real-time application monitoring, website search, and more. OpenSearch is an open source, distributed search and analytics suite that is derived from Elasticsearch. Amazon OpenSearch Service is the successor to Amazon Elasticsearch Service. It offers the latest versions of OpenSearch, support for 19 versions of Elasticsearch, and visualization capabilities that are powered by OpenSearch Dashboards and Kibana. Amazon OpenSearch Service currently has tens of thousands of active customers, with hundreds of thousands of clusters under management, processing hundreds of trillions of requests per month.  
  
For more information, see [Amazon OpenSearch Service](https://aws.amazon.com/opensearch-service/?c=a&sec=srv).

## Data movement

#### Amazon Kinesis

**Note:** Later this week, Amazon Kinesis will have its own dedicated reading that further explains each of the Kinesis services (such as Amazon Kinesis Data Streams, Amazon Kinesis Data Firehose, and Amazon Kinesis Data Analytics).  
  
With Amazon Kinesis, you can collect, process, and analyze real-time, streaming data so that you can get timely insights and react quickly to new information. Amazon Kinesis offers key capabilities to cost-effectively process streaming data at virtually any scale, along with the flexibility to choose the tools that best suit the requirements of your application. With Amazon Kinesis, you can ingest real-time data such as video, audio, application logs, website clickstreams, and Internet of Things (IoT) telemetry data for machine learning, analytics, and other applications. You can use Amazon Kinesis to process and analyze data as it arrives, which means that you can respond quickly—you don’t need to wait for all your data to be collected before processing can begin.  
  
For more information, see [Amazon Kinesis](https://aws.amazon.com/kinesis/).

#### AWS Glue

AWS Glue is a serverless data integration service that you can use to discover, prepare, and combine data for analytics, machine learning, and application development. AWS Glue provides capabilities that are needed for data integration so that you can start analyzing your data and using your data in minutes instead of months. Data integration is the process of preparing and combining data for analytics, machine learning, and application development. It involves multiple tasks, such as discovering and extracting data from various sources; enriching, cleaning, normalizing, and combining data; and loading and organizing data in databases, data warehouses, and data lakes. These tasks are often handled by different types of users who each use different products.  
  
For more information, see [AWS Glue](https://aws.amazon.com/glue/?whats-new-cards.sort-by=item.additionalFields.postDateTime&whats-new-cards.sort-order=desc).

#### AWS DMS

AWS Database Migration Service (AWS DMS) helps you migrate databases to AWS quickly and securely. The source database remains fully operational during the migration, which minimizes downtime to applications that rely on the database. AWS DMS can migrate your data to and from the most widely used commercial and open-source databases.  
  
For more information, see [AWS Database Migration Service](https://aws.amazon.com/dms/).

## Predictive analytics and machine learning

You won’t explore this area much in this course, but—as was mentioned earlier in this reading—regardless of the kind of architecture you want to create, AWS offers services that can help. (That is, use the right tool for the job!) Though you might not see the following AWS services typically listed with AWS data services, you can use them to handle or process data for predictive analytics and machine learning (ML).

#### Amazon SageMaker

SageMaker can be used for any generic ML solution. You can use it to build, train, and deploy ML models for virtually any use case with fully managed infrastructure, tools, and workflows. SageMaker requires a learning curve to use, but it’s a managed serverless service that many people can use to innovate with ML through a choice of tools—such as integrated development environments (IDEs) for data scientists and no-code interfaces for business analysts.  
  
For more information, see [Amazon SageMaker](https://aws.amazon.com/sagemaker/).

#### Amazon Rekognition

Amazon Rekognition is one of Raf’s favorite ML services from the entire list of AWS services! It is easy to use, serverless, and abstracted, in the sense that you interact with it by doing API calls. With Amazon Rekognition, you can automate image and video analysis by adding pretrained or customizable computer vision API operations to your applications without building ML models and infrastructure from scratch.  
  
On its own, Amazon Rekognition is not a data analytics service. However, it’s listed here because you can use it as part of a data analytics solution. It is designed to analyze millions of images, streaming, and stored videos within 3 seconds. Check out some examples:  
In this image, Amazon Rekognition is detecting objects (such as packages, pets, or people) in real time from live video streams.  
In this image, Amazon Rekognition is detecting faces that appear in images and videos, and recognizing attributes (such as open eyes, glasses, and facial hair) for each face.  
For more information, see [Amazon Rekognition](https://aws.amazon.com/rekognition/).

#### Amazon Comprehend

Amazon Comprehend is a natural-language processing (NLP) service that uses ML to uncover valuable insights and connections in text, which is instrumental for a data analytics solution. For example, you could mine business and call center analytics or process financial documents. For medical use cases, you can use Amazon Comprehend Medical, which focuses on extracting information accurately and quickly from unstructured medical text.  
  
For more information, see [Amazon Comprehend](https://aws.amazon.com/comprehend/) and [Amazon Comprehend Medical](https://aws.amazon.com/comprehend/medical/).

**Here are 20 free AWS courses and tutorials that you can use to get started:**

1. **AWS Certified Solutions Architect Associate Introduction** - An introduction to the AWS Certified Solutions Architect Associate exam and designing, deploying, and managing applications on AWS. Link: <https://lnkd.in/d4eR5gsW>
2. **Amazon Web Services** - Learning and Implementing AWS Solution - An overview of AWS and basics of using AWS services such as EC2 and S3. Link: <https://lnkd.in/dATppQUa>
3. **Amazon Web Services (AWS) - Zero to Hero** - Covers fundamentals of AWS, including EC2, S3, and VPC, as well as advanced topics such as high availability and disaster recovery. Link: <https://lnkd.in/dMM9CgmU>
4. **Starting your Career with Amazon AWS** - Designed for beginners, provides an introduction to AWS and cloud computing. Link: <https://lnkd.in/dU7RYfJb>
5. **Learn Amazon Web Services (AWS) easily to become Architect** - Overview of AWS and covers topics such as EC2, S3, and VPC. Link: <https://lnkd.in/d4wx6yYv>
6. **Amazon Web Services (AWS) EC2**: An Introduction - Covers the basics of using EC2, including launching instances and connecting to them. Link: <https://lnkd.in/drUvNuFk>
7. **Amazon Web Services (AWS): CloudFormation** - Covers CloudFormation, a service that enables you to create and manage AWS resources using templates. Link: <https://lnkd.in/dAc65c-H>
8. **AWS VPC Transit Gateway - Hands On Learning!** - Covers VPC Transit Gateway, a service that enables you to connect VPCs and on-premises networks. Link: <https://lnkd.in/dTzjiTVv>
9. **Introduction to Cloud Computing for Beginners in 30 mins** - Provides an introduction to cloud computing, including the benefits and challenges of using cloud services. Link: <https://lnkd.in/dfzCaPEN>
10. **Cloud Computing With Amazon Web Services** - Covers the basics of cloud computing and provides an overview of AWS services. Link: <https://lnkd.in/dGD3FnAW>
11. **Amazon AWS Cloud IAM Hands-On** - Covers AWS Identity and Access Management (IAM), a service that enables you to manage access to AWS resources. Link: <https://lnkd.in/ddSBhiST>
12. **Multitier architecture with AWS** - Covers how to use AWS services to implement a multitier architecture for your applications. Link: <https://lnkd.in/diKgCeSW>
13. **Create and manage VPC on AWS Cloud** - Covers how to create and manage Virtual Private Cloud (VPC) on AWS. Link: <https://lnkd.in/dumuhvnG>
14. **Mastering AWS**: Featuring IAM - Covers advanced topics related to IAM, including policies, roles, and federation. Link: <https://lnkd.in/dUs4NZeV>
15. **AWS + Serverless** - Covers serverless computing on AWS, including AWS Lambda and API Gateway. Link: <https://lnkd.in/dtGAZmUK>
16. **Amazon AWS Cloud EC2 Hands-On** - Provides a hands-on introduction to EC2, including launching and managing instances. Link: <https://lnkd.in/dTc93nkq>
17. **A Practical Introduction to Cloud Computing** - Provides a practical introduction to cloud computing, including the benefits and challenges of using cloud services. Link: <https://lnkd.in/dSMcrC_U>
18. **Hosting your static website on Amazon AWS S3 service** - Covers how to host a static website on Amazon S3, which is a highly available and scalable storage service. Link: <https://lnkd.in/dBw4RKs2>
19. **AWS Tutorials - DynamoDB and Database Migration Service** - Covers DynamoDB, a fully managed NoSQL database service, and Database Migration Service, which enables you to migrate databases to AWS. Link: <https://lnkd.in/dhSY8j7T>
20. **Getting Started with Amazon Web Services** - Provides an overview of AWS services and covers how to create and manage instances using the AWS Management Console. Link: <https://lnkd.in/dvabaewb>

In this reading, you will find further information about the topics that Morgan and Raf talked about in the video where they both played Solutions Architects.

## Caching for Amazon DynamoDB by using Amazon DynamoDB Accelerator

Raf mentioned that if you need to improve Amazon DynamoDB performance to microsecond latency, to look into using Amazon DynamoDB Accelerator (DAX).   
  
DAX is a fully managed, highly available, in-memory [cache](https://aws.amazon.com/caching/) for DynamoDB that’s designed to deliver up to a 100-times performance improvement—from milliseconds to microseconds—even at millions of requests per second.  
  
DAX does the heavy lifting that’s required to add in-memory acceleration to your DynamoDB tables—and developers don’t need to manage cache invalidation, data population, or cluster management.  
  
A benefit of using DAX is that you don’t need to modify the application logic because DAX is compatible with existing DynamoDB API calls.  
  
DAX is designed to run within an Amazon Virtual Private Cloud (Amazon VPC) environment. Amazon VPC defines a virtual network that closely resembles a traditional data center. With a VPC, you have control over its IP address range, subnets, routing tables, network gateways, and security settings. You can launch a DAX cluster in your virtual network, and control access to the cluster by using Amazon VPC security groups.  
  
For more information about DAX, see [Amazon DynamoDB Accelerator.](https://aws.amazon.com/dynamodb/dax/)

## Optimizing AWS Lambda

### AWS Lambda Power Tuning

Morgan mentioned that you can fine-tune the memory or power configuration for your AWS Lambda functions to potentially increase performance and lower costs.  
  
[AWS Lambda Power Tuning](https://github.com/alexcasalboni/aws-lambda-power-tuning) is an open-source tool that helps you visualize and fine-tune the memory or power configuration of Lambda functions. It runs in your own AWS account, and it supports three optimization strategies: cost, speed, and balanced.  
  
AWS Lambda Power Tuning is a state machine that’s powered by AWS Step Functions. It helps you optimize your Lambda functions for cost or performance in a data-driven way.  
  
The state machine is designed to be easy to deploy and fast to execute. Also, it's language agnostic, so you can optimize any Lambda functions in your account.  
  
To work with AWS Lambda Power Turning, you provide a Lambda function Amazon Resource Name (ARN) as input. The state machine then invokes that function with multiple power configurations (from 128 MB to 10 GB—you decide which values). Then, it analyzes all the execution logs and suggests the best power configuration to minimize cost or maximize performance.  
  
Note that the input function will run in your AWS account, which means that it will perform HTTP requests, SDK calls, cold starts, and so on. The state machine also supports cross-Region invocations, and you can enable parallel execution to generate results in a few seconds.  
  
The state machine generates a visualization of average cost and speed for each power configuration.  
  
For example, the following diagram shows results for two CPU-intensive functions, which become both cheaper and faster with more power:  
For more information, see [AWS Lambda Power Tuning](https://github.com/alexcasalboni/aws-lambda-power-tuning).

### AWS Lambda Powertools

Morgan also mentioned that you can use another suite of tools called AWS Lambda Powertools to optimize your Lambda functions and use best practices. AWS Lambda Powertools is a suite of utilities for AWS Lambda functions that is designed to make it easier to adopt best practices such as tracing, structured logging, custom metrics, idempotency, batching, and more.   
  
For more information, see [AWS Lambda Powertools](https://awslabs.github.io/aws-lambda-powertools-python/latest/).

### AWS Lambda execution environment reuse

Another optimization technique that Morgan mentioned for Lambda is to move certain initialization tasks in your code so they are outside the handler. These tasks can then be reused across invocations (which is also known as execution environment reuse).  
  
You can take advantage of execution environment reuse to improve the performance of your function. To do this, initialize SDK clients and database connections outside of the function handler, and cache static assets locally in the /tmp directory. Subsequent invocations that are processed by the same instance of your function can reuse these resources. This reuse saves cost by reducing function run time.   
  
To avoid potential data leaks across invocations, don’t use the execution environment to store user data, events, or other information with security implications. If your function relies on a mutable state that can’t be stored in memory within the handler, consider creating a separate function or separate versions of a function for each user.  
  
To learn more, see [Best practices for working with AWS Lambda.](https://docs.aws.amazon.com/lambda/latest/dg/best-practices.html)

### Resources

* For more information about how to optimize serverless applications for cost, see [Building well-architected serverless applications: Optimizing application costs](https://aws.amazon.com/blogs/compute/building-well-architected-serverless-applications-optimizing-application-costs/) in the *AWS Compute Blog*.
* For more information about well-architected serverless applications, visit [Serverless Applications Lens - AWS Well-Architected Framework](https://docs.aws.amazon.com/wellarchitected/latest/serverless-applications-lens/welcome.html?did=wp_card&trk=wp_card).

In this reading, you will find further information about the topics that Morgan and Raf talked in the video where they both played solutions architects.  
  
The following diagram is a more detailed view of the solution that was covered this week.

## Disaster recovery

Morgan mentioned that because this week’s customer is an enterprise company, they could consider designing a disaster recovery (DR) plan for this workload.  
  
The DR strategies that are available to you within AWS can be broadly categorized into four approaches, which range from the low cost and low complexity of making backups to more complex strategies that use multiple active Regions. Active/passive strategies use an active site (such as an AWS Region) to host the workload and serve traffic. The passive site (such as a different AWS Region) is used for recovery. The passive site doesn’t actively serve traffic until a failover event is triggered.  
  
The following diagram outlines the different approaches to disaster recovery on AWS.

## Backup and restore

Backup and restore is a suitable approach for mitigating against data loss or corruption. This approach can also be used to mitigate against a regional disaster by replicating data to other AWS Regions, or to mitigate a lack of redundancy for workloads that are deployed to a single Availability Zone. In addition to data, you must redeploy the infrastructure, configuration, and application code in the recovery Region. To enable infrastructure to be redeployed quickly without errors, you should always deploy by using infrastructure as code (IaC), with services such as AWS CloudFormation or the AWS Cloud Development Kit (AWS CDK). Without IaC, it might be complex to restore workloads in the recovery Region, which can lead to increased recovery times and possibly exceed your Recovery Time Objective (RTO). In addition to backing up your user data, back up your code and configurations—including [Amazon Machine Images (AMIs)](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AMIs.html) that you use to create Amazon Elastic Compute Cloud (Amazon EC2) instances. You can use [AWS CodePipeline](http://aws.amazon.com/codepipeline) to automate the redeployment of your application code and configurations.

## Pilot light

With the pilot light approach, you replicate your data from one Region to another, and you provision a copy of your core workload infrastructure. Resources that are needed to support data replication and backup—such as databases and object storage—are always on. Other elements (such as application servers) are loaded with application code and configurations, but are turned off. They are only used during testing or when DR failover is invoked. In the cloud, you have the flexibility to deprovision resources when you don’t need them, and provision them when you do. A best practice for a resource that’s turned off is to not deploy the resource, and then create the configuration and capabilities to deploy it (turn on) when needed. Unlike the backup and restore approach, your core infrastructure is always available. Thus, you always have the option to quickly provision a full scale production environment by turning on and scaling out your application servers.

## Warm standby

The warm standby approach involves provisioning a scaled down—but fully functional—copy of your production environment in another Region. This approach extends the pilot-light concept and decreases the time to recovery because your workload is always-on in another Region. With this approach, you can perform testing or implement continuous testing more easily, which can increase your confidence in your ability to recover from a disaster.

## Multi-site active/active

You can run your workload simultaneously in multiple Regions as part of either a multi-site active/active strategy or a hot standby active/passive strategy. A multi-site active/active approach serves traffic from all Regions where it’s deployed. In contrast, hot standby approach serves traffic from only a single Region, and the other Regions are used only for DR. With a multi-site active/active approach, users can access your workload in any Regions where it’s deployed. This approach to DR is the most complex and most costly. However, it can reduce your recovery time to near zero for most disasters, with the correct technology choices and implementation. (Note that data corruption might need to rely on backups, which usually results in a non-zero recovery point.) Hot standby uses an active/passive configuration, where users are directed to only a single region and DR Regions don’t take traffic. Most customers find that if they’re going to stand up a full environment in the second Region, it makes sense to use it in an active/active approach. Alternatively, if you don’t want to use both Regions to handle user traffic, then warm standby offers an approach that’s more economical and operationally less complex.  
  
For more information about DR, see the [Disaster Recovery of Workloads on AWS: Recovery in the Cloud](https://docs.aws.amazon.com/whitepapers/latest/disaster-recovery-workloads-on-aws/disaster-recovery-workloads-on-aws.html) whitepaper.

## AWS Direct Connect with AWS VPN for failover

Raf commented that to make the connection to AWS more resilient, the customer can consider using Amazon Site-to-Site VPN as a failover for AWS Direct Connect so that the connection is redundant. This setup would protect the customer if the AWS Direct Connect connection became unavailable. With this setup, the customer would be able to fail over to the virtual private network (VPN) connection and remain connected to their AWS resources.  
  
For more information about how to set up Site-to-Site VPN as a failover for Direct Connect, see [How do I configure Direct Connect and VPN failover with Transit Gateway?](https://aws.amazon.com/premiumsupport/knowledge-center/dx-configure-dx-and-vpn-failover-tgw/)

## Automatic scaling for containers

Morgan commented that this week’s customer needs to look into how they will scale their containers. When the customer uses Amazon Elastic Container Service (Amazon ECS), they will need to think about scaling both their underlying EC2 cluster and the containers themselves.

### Scaling the cluster

Amazon ECS can manage the scaling of EC2 instances that are registered to your cluster. This capability is referred to as *Amazon ECS cluster auto scaling*, and is performed by an Amazon ECS Auto Scaling group capacity provider that has managed scaling turned on.   
  
When you use an Auto Scaling group capacity provider with managed scaling, Amazon ECS creates two custom Amazon CloudWatch metrics, and a target tracking scaling policy that attaches to your Auto Scaling group. Amazon ECS then manages the scale-in and scale-out actions of the Auto Scaling group based on the load that your tasks put on your cluster.

### Scaling the containers

*Automatic scaling* is the ability to increase or decrease the desired count of tasks in your Amazon ECS service automatically. Amazon ECS uses the Application Auto Scaling service to provide this functionality.   
  
Your Amazon ECS service can be optionally configured automatically scale its desired count of tasks in your Amazon ECS service, either up or down, in response to CloudWatch alarms.  
  
Amazon ECS Service Auto Scaling supports the following types of scaling policies:

* Target tracking scaling policies (Recommended): Increase or decrease the number of tasks that your service runs, based on a target value for a specific metric. This scaling approach is similar to the way that your thermostat maintains the temperature of your home. You select the temperature, and the thermostat manages the temperature.
* Step scaling policies: Increase or decrease the number of tasks that your service runs, based on a set of scaling adjustments, which are also known as step adjustments. These adjustments vary based on the size of the alarm breach.

For more resources about automatic scaling and Amazon ECS, see the following:

* For more information about how to scale an ECS cluster, see [Amazon ECS cluster Auto Scaling](https://docs.aws.amazon.com/AmazonECS/latest/developerguide/cluster-auto-scaling.html).
* For a hands-on tutorial about how to scale an ECS cluster, see the [Amazon ECS Workshop: Deploy ECS Cluster Auto Scaling](https://ecsworkshop.com/capacity_providers/ec2/).
* For more information about service auto scaling, see [Service auto scaling](https://docs.aws.amazon.com/AmazonECS/latest/developerguide/service-auto-scaling.html) in the *Amazon ECS Developer Guide*.

## Automatic scaling for Amazon RDS

Morgan commented that this week’s customer should look into RDS storage autoscaling.  
  
When you enable storage autoscaling, Amazon RDS automatically scales up your storage when it detects that you are running out of free database space.  
  
  
If your workload is unpredictable, you can enable storage autoscaling for an RDS DB instance. To do so, you can use the Amazon RDS console, the Amazon RDS API, or the AWS Command Line Interface (AWS CLI).   
  
For more information about Amazon RDS storage autoscaling, see [Managing capacity automatically with Amazon RDS storage autoscaling.](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_PIOPS.StorageTypes.html#USER_PIOPS.Autoscaling)

## Amazon S3 Intelligent-Tiering

Raf commented that this customer could also optimize for cost by using S3 Intelligent-Tiering on the S3 bucket where their data is being stored.  
  
S3 Intelligent-Tiering is the only cloud storage class that delivers automatic storage cost savings when data access patterns change, thus reducing performance impact or operational overhead. The Amazon S3 Intelligent-Tiering storage class is designed to optimize storage costs by automatically moving data to the most cost-effective access tier when access patterns change. For a small monthly object monitoring and automation charge, S3 Intelligent-Tiering monitors access patterns and automatically moves objects that haven’t been accessed to lower-cost access tiers.  
  
S3 Intelligent-Tiering is a good storage class for data with unknown, changing, or unpredictable access patterns—independent of object size or retention period. You can use S3 Intelligent-Tiering as the default storage class for virtually any workload, especially data lakes, data analytics, new applications, and user-generated content.  
  
For more information about S3 Intelligent-Tiering, see [Amazon S3 Intelligent-Tiering storage class](https://aws.amazon.com/s3/storage-classes/intelligent-tiering/).

[version\_1.0]

### Note

The exercises in this course will have an associated charge in your AWS account. In this exercise, you create or use the following resources:

* AWS Identity and Access Management (IAM) policy and user (policies and users are AWS account features, offered at no additional charge)
* Amazon Simple Storage Service (Amazon S3) bucket
* AWS Lambda function
* Amazon Kinesis Data Firehose delivery stream
* Amazon API Gateway
* Amazon Athena table
* Amazon QuickSight dashboards

**The final task in this exercise includes instructions to delete all the resources that you create.**

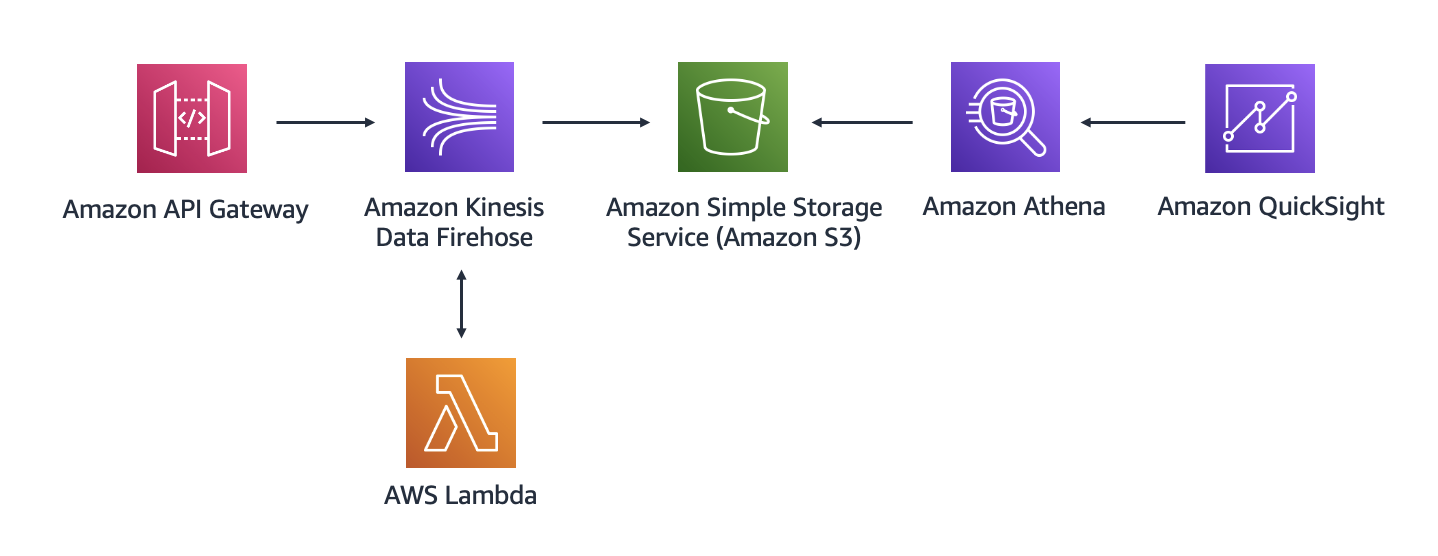
Familiarize yourself with [**IAM**](https://aws.amazon.com/iam/), [**Amazon S3 pricing**](https://aws.amazon.com/s3/pricing/), [**Lambda pricing**](https://aws.amazon.com/lambda/pricing/), [**Kinesis Data Firehose pricing**](https://aws.amazon.com/kinesis/data-firehose/pricing/), [**API Gateway pricing**](https://aws.amazon.com/api-gateway/pricing/), [**Amazon Athena pricing**](https://aws.amazon.com/athena/pricing/), [**Amazon QuickSight pricing**](https://aws.amazon.com/quicksight/pricing/), and the [**AWS Free Tier**](https://aws.amazon.com/free/).

# Exercise 2: Architecting Solutions: Building a Proof of Concept for Data Analytics

This exercise provides you with instructions for how to build a data analytics solution.

This week, you will design an architecture for a customer who needs an analytics solution to ingest, store, and visualize clickstream data. The customer is a restaurant owner who wants to derive insights for all menu items that are ordered in their restaurant. Because the customer has limited staff for running and maintaining this solution, you will build a proof of concept by using managed services on AWS.

The following architectural diagram shows the flow that you will follow.

Architecture diagram for exercise 2

In this architecture, you use API Gateway to ingest clickstream data. Then, the Lambda function transforms the data and sends it to Kinesis Data Firehose. The Firehose delivery stream places all files in Amazon S3. Then, you use Amazon Athena to query the files. Finally, you use Amazon QuickSight to transform data into graphics.

In this exercise, you will learn how to do the following:

* Create IAM policies and roles to follow the best practices of working in the AWS Cloud.
* Create the object storage S3 bucket to store clickstream data.
* Create the Lambda function for Amazon Kinesis Data Firehose to transform data.
* Create a Kinesis Data Firehose delivery stream to deliver real-time streaming data to Amazon S3.
* Create a REST API to insert data.
* Create an Amazon Athena table to view the obtained data.
* Create Amazon QuickSight dashboards to visualize data.

**Note**:

To complete the instructions in this exercise, choose the US East (N. Virginia) us-east-1 Region in the menu bar the AWS Management Console.

The instructions might prompt you to enter your account ID. Your account ID is a 12-digit account number that appears under your account alias in the top-right corner of the AWS Management Console. When you enter your account number (ID), make sure that you remove hyphens (-).

## Task 1: Setup: Creating the IAM policy and role

In this task, you create a custom IAM policy and role to grant limited permissions to specific AWS services.

### Step 1.1: Creating custom IAM policies

1. Sign in to the AWS Management Console.
2. In the search box, enter IAM and from the list, choose **IAM**.
3. In the navigation pane, choose **Policies** and then choose **Create policy**.

The **Create policy** page appears.

1. In the **JSON** tab, replace the placeholder code by pasting the following policy:
2. {
3. "Version": "2012-10-17",
4. "Statement": [
5. {
6. "Sid": "VisualEditor0",
7. "Effect": "Allow",
8. "Action": "firehose:PutRecord",
9. "Resource": "\*"
10. }
11. ]

}

1. Choose **Next**.
2. For the policy name, enter API-Firehose.
3. Choose **Create policy**.

This IAM policy grants permissions to write records to Amazon Kinesis Data Firehose delivery stream.

### Step 1.2: Creating an IAM role and attaching a policy to it

In this step, you create an IAM role that enables API Gateway to send streaming data to Amazon Kinesis Data Firehose. You then add the API-Firehose policy that you created to the role.

1. In the navigation pane of the IAM dashboard, choose **Roles** and then choose **Create role**.
2. For **Trusted entity type**, select **AWS service**
3. In the **Use case** section choose **API Gateway**.
4. Choose **Next** and then choose **Next** again.
5. For **Role name**, enter APIGateway-Firehose.
6. Choose **Create role**.
7. From the roles list, choose the **APIGateway-Firehose** role.
8. In the **Permissions policies** section, on the **Add permissions** menu, choose **Attach policies**.
9. From the policies list, select **API-Firehose** and choose **Add permissions**.
10. In the **Summary** section, copy the **APIGateway-Firehose** ARN and save it for your records. Your APIGateway-Firehose ARN might look similar to the following: arn:aws:iam::<account ID>:role/APIGateway-Firehose.

## Task 2: Creating an S3 bucket

In this task, you create an object storage bucket in Amazon S3 to store streaming data in the AWS Cloud.

1. In the AWS Management Console search box, enter S3 and open the service by choosing **S3**.
2. Choose **Create bucket**.
3. For **Bucket name**, enter a unique name.

**Note**: An S3 bucket name must be globally unique. You may name your bucket similar to the following example: architecting-week2-<your initials>. Replace <your initals> with your own value. Make sure that you also delete the angle brackets (<>). For example: architecting-week2-mw.

1. Make sure that **AWS Region** is set to **US East (N. Virginia) us-east-1**.
2. At the bottom of the page, choose **Create bucket**.
3. Open the bucket details by choosing the name of the bucket that you just created.
4. Choose the **Properties** tab.
5. In the **Bucket overview** section, copy your bucket’s Amazon Resource Name (ARN) and save it for your records. The ARN of your bucket may look similar to the following: arn:aws:s3:::DOC-EXAMPLE-BUCKET.

## Task 3: Creating a Lambda function

In this task, you create a Lambda function that transforms data before Amazon Kinesis Data Firehose ingests it into the object storage bucket.

1. In the AWS Management Console, search for and open the **Lambda** service.
2. Choose **Create a function**.
3. Select the **Use a blueprint** card.
4. In the filter box of the **Blueprints** section, enter Kinesis.
5. In the list of results, you may see two blueprints called **Process records sent to a Kinesis Firehose stream** for Node.js and Python. Select the Python 3.8 blueprint called **Process records sent to a Kinesis Firehose stream** and choose **Configure**.
6. For **Function name**, enter transform-data.
7. Keep all the other default settings and choose **Create function**.
8. In the **Code** tab, replace the default code with the following:
9. import json
10. import boto3
11. import base64
12. output = []
13. **def** lambda\_handler(event, context):
14. **for** record **in** event['records']:
15. payload = base64.b64decode(record['data']).decode('utf-8')
16. row\_w\_newline = payload + "\n"
17. row\_w\_newline = base64.b64encode(row\_w\_newline.encode('utf-8'))
18. output\_record = {
19. 'recordId': record['recordId'],
20. 'result': 'Ok',
21. 'data': row\_w\_newline
22. }
23. output.append(output\_record)

**return** {'records': output}

1. Choose **Deploy**.
2. Choose the **Configuration** tab.
3. In the **General configuration** section, choose **Edit** and change the **Timeout** setting to 10 seconds.
4. Choose **Save**.
5. In the **Function overview** section, copy the function ARN and save it for your records.

Your ARN might look similar to the following example: arn:aws:lambda:us-east-1:<account ID>:function:transform-data.

## Task 4: Creating a Kinesis Data Firehose delivery stream

In this task, you complete two steps. First, you create a Kinesis Data Firehose delivery stream to ingest streaming data. Then, you copy the ARN of the delivery stream’s IAM role. Later in the exercise, you use this ARN to connect the delivery stream to your storage bucket.

### Step 4.1: Creating the Kinesis Data Firehose delivery stream

1. In AWS Management Console, search for and open the **Kinesis** service.
2. On the **Get started** card, select **Kinesis Data Firehose** and then choose **Create delivery stream**.
3. For **Source and destination**, configure these settings:
   * **Source**: Direct PUT
   * **Destination**: Amazon S3
4. For **Transform and convert records - optional**, configure these settings:
   * **Enable data transformation**: Enabled
   * **AWS Lambda function**: ARN of the function that you created in Lambda
     + **Note**: For example, the ARN might look like the following: arn:aws:lambda:us-east-1:<account ID>:function:transform-data
   * **Version and alias**: $LATEST
5. For **Destination settings**, choose **Browse**.
6. Select the S3 bucket that you created for this exercise and then select **Choose**.
7. At the bottom of the page, choose **Create delivery stream**.

It may take up to 5 minutes to create the Kinesis Data Firehose delivery stream.

### Step 4.2: Copying the ARN of the IAM role

1. If needed, open the details page for the delivery stream that you created.
2. Choose the **Configuration** tab.
3. In the **Service access** section, choose the IAM role.

The role opens in a new IAM window.

1. Copy the ARN of the IAM role and save it. Your ARN role might look similar to the following: arn:aws:iam::<account ID>:role/service-role/KinesisFirehoseServiceRole-PUT-S3-7HMmt-us-east-1-1664893091685.

You will need this ARN in the next task.

## Task 5: Adding the Firehose delivery stream ARN to the S3 bucket

In this task, you edit permissions for the S3 bucket so it can store data that’s ingested by Kinesis Data Firehose.

1. Open the Amazon S3 console and open the details page for the bucket you created in this exercise.
2. Choose the **Permissions** tab.
3. In the **Bucket policy** section, choose **Edit** and paste the following script:
4. {
5. "Version": "2012-10-17",
6. "Id": "PolicyID",
7. "Statement": [
8. {
9. "Sid": "StmtID",
10. "Effect": "Allow",
11. "Principal": {
12. "AWS": "<Enter the ARN for the Kinesis Firehose IAM role>"
13. },
14. "Action": [
15. "s3:AbortMultipartUpload",
16. "s3:GetBucketLocation",
17. "s3:GetObject",
18. "s3:ListBucket",
19. "s3:ListBucketMultipartUploads",
20. "s3:PutObject",
21. "s3:PutObjectAcl"
22. ],
23. "Resource": [
24. "<Enter the ARN of the S3 bucket>",
25. "<Enter the ARN of the S3 bucket>/\*"
26. ]
27. }
28. ]

}

1. In the script code, replace the following placeholders:
   * **AWS**: Paste the ARN of the Kinesis Data Firehose IAM role. You saved this ARN in the last step of the previous task. For example:
   * **"Principal":** {
   * "AWS": "arn:aws:iam::<account ID>:role/service-role/KinesisFirehoseServiceRole-PUT-S3-7HMmt-us-east-1-1664893091685"

}**,**

* + **Resource**: Paste the ARN of the S3 bucket for both placeholders. For example:
  + **"Resource":** [
  + "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
  + "arn:aws:s3:::DOC-EXAMPLE-BUCKET/\*"

]

1. Save your changes.

## Task 6: Creating an API in API Gateway

In this task, you create a REST API in API Gateway. The API serves as a communication gateway between your application and AWS services. In this exercise, you use API Gateway to insert mock data.

1. Open the API Gateway service console.
2. On the **REST API** card with an open authentication, choose **Build** and configure these settings:
   * **Choose the protocol**: REST
   * **Create new API**: New API
   * **API name**: clickstream-ingest-poc
   * **Endpoint Type**: Regional
3. Choose **Create API**.
4. In the **Resources** pane, on the **Actions** menu, choose **Create Resource**.
5. On the **New Child Resource** page, name the resource poc and choose **Create Resource**.
6. On the **Actions** menu, choose **Create Method**.
7. On the method menu (down arrow), choose **POST**. Choose the checkmark to save your changes.
8. On the **POST - Setup** page, configure the following settings:
   * **Integration type**: AWS Service
   * **AWS Region**: us-east-1
   * **AWS Service**: Firehose
   * **AWS Subdomain**: Keep empty
   * **HTTP method**: POST
   * **Action Type**: Use action name
   * **Action**: PutRecord
   * **Execution role**: Paste the ARN of the APIGateway-Firehose role that you created in task 1
     + **Note**: For example, the ARN might look like the following: arn:aws:iam::<account ID>:role/APIGateway-Firehose
   * **Content Handling**: Passthrough
   * **Use Default Timeout**: Keep selected
9. Save your changes.
10. Choose the **Integration Request** card.
11. Expand **Mapping Templates** and for **Request body passthrough**, choose **When there are no templates defined (recommended)**.
12. Choose **Add mapping template**.
13. For **Content-Type**, enter application/json and save your changes by choosing the check mark.
14. In the **Generate template** box, paste the following script:
15. {
16. "DeliveryStreamName": "<Enter the name of your delivery stream>",
17. "Record": {
18. "Data": "$util.base64Encode($util.escapeJavaScript($input.json('$')).replace('\', ''))"
19. }

}

1. In the script code, replace the **DeliveryStreamName** placeholder value with the name of the Kinesis Data Firehose delivery stream that you created.

**Note**: You can find the name of the Firehose delivery stream in the Amazon Kinesis console, in the **Delivery streams** tab. For example: "DeliveryStreamName": "PUT-S3-AAaAA"

1. Choose **Save**.
2. Go back to the **/poc - POST - Method Execution** page.
3. Choose **Test**.

For the test, you will send several discrete JSON payloads to the API. These payloads simulate how an application frontend sends small bits of data each time a person looks through an item in the menu.

1. In the **Request Body** box, paste the following JSON:
2. {
3. "element\_clicked":"entree\_1",
4. "time\_spent":67,
5. "source\_menu":"restaurant\_name",
6. "created\_at":"2022–09–11 23:00:00"

}

1. Choose **Test**.
2. Review the request logs (on the right side of the window) and confirm that you see the following messages: “Successfully completed execution” and “Method completed with status: 200”.

These messages indicate that API Gateway processed the data successfully.

1. Replace the previous JSON by pasting the following JSON payload and then choose **Test**. Again, confirm that API Gateway processed the data successfully.
2. {
3. "element\_clicked":"entree\_1",
4. "time\_spent":12,
5. "source\_menu":"restaurant\_name",
6. "created\_at":"2022–09–11 23:00:00"

}

1. Repeat the steps for pasting the code, testing the API, and confirming success for each of the following JSON payloads:

**Entree 4**

{

"element\_clicked":"entree\_4",

"time\_spent":32,

"source\_menu":"restaurant\_name",

"createdAt":"2022–09–11 23:00:00"

}

**Drink 1**

{

"element\_clicked":"drink\_1",

"time\_spent":15,

"source\_menu":"restaurant\_name",

"created\_at":"2022–09–11 23:00:00"

}

**Drink 3**

{

"element\_clicked":"drink\_3",

"time\_spent":14,

"source\_menu":"restaurant\_name",

"created\_at":"2022–09–11 23:00:00"

}

In the next task, you will verify that the solution is working by creating an Amazon Athena table that displays ingested entries.

## Task 7: Creating an Athena table

In this task, you create a table in Athena. You also run a Structured Query Language (SQL) query to view the payloads that you inserted with the REST API.

1. Open the Athena service console.
2. At the left choose **Query editor**.
3. Choose the **Settings** tab and then choose **Manage**.
4. Choose **Browse S3** and choose the S3 bucket that you created in this exercise.
5. Select **Choose** and then **Save**.
6. Choose the **Editor** tab.
7. In the **Tables and views** section, on the **Create** menu, choose **CREATE TABLE AS SELECT**.
8. In the **Query** editor, replace the placeholder code with the following script:
9. **CREATE** **EXTERNAL** **TABLE** **my\_ingested\_data** **(**
10. **element\_clicked** **STRING,**
11. **time\_spent** **INT,**
12. **source\_menu** **STRING,**
13. **created\_at** **STRING**
14. **)**
15. **PARTITIONED** **BY** **(**
16. **datehour** **STRING**
17. **)**
18. **ROW** **FORMAT** **SERDE** **'org.openx.data.jsonserde.JsonSerDe'**
19. **with** **serdeproperties** **(** **'paths'='element\_clicked,** **time\_spent,** **source\_menu,** **created\_at'** **)**
20. **LOCATION** **"s3://<Enter** **your** **Amazon** **S3** **bucket** **name>/"**
21. **TBLPROPERTIES** **(**
22. **"projection.enabled"** **=** **"true",**
23. **"projection.datehour.type"** **=** **"date",**
24. **"projection.datehour.format"** **=** **"yyyy/MM/dd/HH",**
25. **"projection.datehour.range"** **=** **"2021/01/01/00,NOW",**
26. **"projection.datehour.interval"** **=** **"1",**
27. **"projection.datehour.interval.unit"** **=** **"HOURS",**
28. **"storage.location.template"** **=** **"s3://<Enter** **your** **Amazon** **S3** **bucket** **name>/$**{**datehour**}**/"**

**)**

1. In the script you pasted, replace the following placeholder values:
   * **LOCATION**: Replace <Enter your Amazon S3 bucket name> with the name of your bucket
     + **Note**: Make sure that you have s3:// before the bucket name and a slash (/) at the end (for example: "s3://DOC-EXAMPLE-BUCKET/")
   * **storage.location.template**: Replace <Enter your Amazon S3 bucket name> with the name of your bucket
     + **Note**: For example: "s3://DOC-EXAMPLE-BUCKET/${datehour}/"
2. Choose **Run**.

The query creates the **my\_ingested\_data** table.

1. Create a new query by choosing the plus sign (+) (at the top-right of the query editor).
2. In the query editor, paste SELECT \* FROM my\_ingested\_data; and choose **Run**.

The query should produce results with the entries that you ran in API Gateway.

## Task 8: Visualizing data with QuickSight

After the clickstream data is processed successfully, you can use QuickSight to visualize data. With QuickSight, you can gain better insights into your streaming data by analyzing it and publishing data dashboards.

The instructions for how to visualize data in QuickSight might differ, depending if you are a new user or an existing user.

**Note**: Amazon QuickSight is a subscription service. If you need to delete your QuickSight account after you complete this exercise, follow the instructions in the final task.

### For new Amazon QuickSight users

1. Open the QuickSight service console.
2. Choose **Sign up for QuickSight**. Choose **Enterprise** and **Continue**.
3. Set up an account and choose **Finish**.
4. In the upper-right corner, open the user menu by choosing the user icon and then choose **Manage QuickSight**.
5. In the navigation pane, choose **Security & permissions** and in **QuickSight access to AWS services,** choose **Manage**.
6. Under **Amazon S3**, choose **Select S3 buckets**.
7. Select the bucket that you created in this exercise, and also select **Write permission for Athena Workgroup**.
8. Choose **Finish** and save your changes.
9. Return to the QuickSight console.
10. In the **Analyses** tab, choose **New analysis**.
11. Choose **New dataset**.
12. Choose **Athena** and configure the following settings:
    * **Name datasource**: poc-clickstream
    * **Select workgroup**: [primary]
13. Choose **Create data source**.
14. In the **Choose your table** dialog box, select the **my\_ingested\_data** table, and choose **Select**.
15. In the **Finish dataset creation** dialog box, make sure that **Import to SPICE for quicker analytics** is selected, and choose **Visualize**.
16. View your visualization results by selecting field items and visual types for the diagram.

For more information about how to visualize data in Amazon QuickSight, see [Tutorial: Create an AmazonQuickSight analysis](https://docs.aws.amazon.com/quicksight/latest/user/example-create-an-analysis.html).

### For existing Amazon QuickSight users

1. Open the QuickSight service console.
2. In the upper-right corner, open the user menu by choosing the user icon and then choose **Manage QuickSight**.
3. In the navigation pane, choose **Security & permissions** and in **QuickSight access to AWS services,** choose **Manage**.
4. Under **Amazon S3**, choose **Select S3 buckets**.
5. Select the bucket that you created in this exercise, and also select **Write permission for Athena Workgroup**.
6. Choose **Finish** and save your changes.
7. Return to the QuickSight console by choosing the QuickSight icon (in the upper-left area of the webpage).
8. Choose **New analysis**.
9. Choose **New dataset**.
10. Choose **Athena** and configure these settings:
    * **Data source name**: poc-clickstream
    * **Athena workgroup**: [primary]
11. Choose **Create data source**.
12. In the **Choose your table** dialog box, select **my\_ingested\_data** and choose **Select**.
13. In the **Finish dataset creation** dialog box, keep **Import to SPICE for quicker analytics** selected and choose **Visualize**.
14. View your visualization results by selecting field items and visual types for the diagram.

For more information about how to visualize data in Amazon QuickSight, see [Tutorial: Create an AmazonQuickSight analysis](https://docs.aws.amazon.com/quicksight/latest/user/example-create-an-analysis.html).

## Task 9: Deleting all resources

To avoid incurring costs, we recommend that you delete the AWS resources that you created in this exercise.

1. Delete the QuickSight dashboards.
   1. Return to the QuickSight console by choosing the **QuickSight** icon (in the upper-left area of the webpage).
   2. If needed, in the navigation pane, choose **Analyses**.
   3. On the **my\_ingested\_data** card, open the actions menu by choosing the ellipsis icon.
   4. On the actions menu, choose **Delete** and confirm your action.
   5. Confirm your action.
   6. In the navigation pane, choose **Datasets**.
   7. For each dataset, on the actions menu (ellipsis icon), choose **Delete** and confirm your action.
2. Delete your QuickSight account.

If you will not be using QuickSight in the future, you can delete your account.

**Note**: Amazon QuickSight is a subscription service. For more information about QuickSight pricing, see the “Standard Edition” tab on the [Amazon QuickSight Pricing](https://aws.amazon.com/quicksight/pricing/) page.

* 1. In the QuickSight console, choose the user icon and then choose **Manage QuickSight**.
  2. In the navigation pane, choose **Account settings**.
  3. Choose **Delete account** and confirm your action.

1. Delete the S3 bucket.
   1. Return to the Amazon S3 console.
   2. Choose the bucket that you created for this exercise and delete all files in the bucket.
   3. Select the bucket name, choose **Delete**, and confirm your action.
2. Delete the Athena table and queries.
   1. Return to the Athena console.
   2. Make sure that you are on the **Editor** tab.
   3. In the navigation pane, on the actions menu (ellipsis icon) for **my\_ingested\_data**, choose **Delete table** and confirm your action.
   4. Close the queries that you created.
   5. Choose the **Settings** tab and then choose **Manage**.
   6. Remove the path to the S3 bucket and save your changes.
3. Delete the API Gateway configuration.
   1. Return to the API Gateway console.
   2. Select the API that you created.
   3. On the **Actions** menu, choose **Delete** and confirm your action.
4. Delete the Kinesis Data Firehose delivery stream.
   1. Return to the Kinesis console.
   2. In the navigation pane, choose **Delivery streams**.
   3. Select the delivery stream that you created, choose **Delete**, and confirm your action.
5. Delete the Lambda function.
   1. Return to the Lambda console.
   2. Select the **transform-data** Lambda function.
   3. On the **Actions** menu, choose **Delete**, and confirm your action.
6. Delete the IAM roles and policies.
   1. Return to the IAM dashboard.
   2. In the navigation pane, choose **Roles**.
   3. Select **APIGateway-Firehose**, choose **Delete**, and confirm your action.
   4. In the navigation pane, choose **Policies**.
   5. Select **API-Firehose**.
   6. On the **Actions** menu, choose **Delete** and confirm the deletion.

Congratulations! You successfully completed the exercise. The clickstream results that you ingested, processed, and visualized as part of this exercise will help the restaurant owner better understand their menu options.