

What is Docker?

What is Docker?



- Docker is a software development platform to deploy apps
- Apps are packaged in containers that can be run on any OS
- Apps run the same, regardless of where they're run
 - Any machine
 - No compatibility issues
 - Predictable behavior
 - Less work
 - Easier to maintain and deploy
 - Works with any language, any OS, any technology
- Scale containers up and down very quickly (seconds)

1.

Docker is an open-source software platform that enables developers to package applications and their dependencies into standardized units called containers. Containers are isolated from one another and from the underlying operating system, making them lightweight, portable, and efficient.

Here are some practical examples of Docker:

- A web developer might use Docker to package their web application into a container. This would allow them to easily deploy the application to any environment that has Docker installed, without having to worry about the underlying operating system or hardware.
- A data scientist might use Docker to package their machine learning model into a container. This would allow them to easily share the model with others and to deploy it to production without having to worry about the underlying infrastructure.
- A DevOps engineer might use Docker to package their infrastructure as code into a container. This would allow them to easily automate the deployment and management of their infrastructure.

Docker is a powerful tool that can be used to simplify the development, deployment, and management of applications. It is used by a wide range of organizations, from small startups to large enterprises.

2.

Here are some of the benefits of using Docker:

- **Portability:** Containers are portable, meaning that they can be run on any system that has Docker installed. This makes it easy to deploy applications to different environments, such as development, staging, and production.
- **Efficiency:** Containers are lightweight, meaning that they use fewer resources than traditional virtual machines. This can save you money on hardware costs and improve the performance of your applications.
- **Isolation:** Containers are isolated from one another, meaning that they cannot affect each other. This makes it easier to develop and deploy applications without worrying about conflicts.
- **Security:** Containers can be used to isolate and secure applications. This can help to protect your applications from malware and other security threats.
- **Scalability:** Containers can be scaled easily, meaning that you can easily add or remove containers as needed. This makes it easy to handle sudden changes in demand.

If you are looking for a way to simplify the development, deployment, and management of your applications, then Docker is a good choice.

- 3.
4. For example lets imagine we have An ec2 instance ..inside that we can create containers
5. Like one container runs java project,another container runs sql..like that
6. We can create many containers in the same ec2 instance
7. So the idea is that if we managed to package our application in a Docker container, then it will become very easy for us to run it on an EC2 Instance.

Where Docker images are stored?

- Docker images are stored in Docker Repositories
- Public: Docker Hub <https://hub.docker.com/>
 - Find base images for many technologies or OS:
 - Ubuntu
 - MySQL
 - NodeJS, Java...
- Private: Amazon ECR (Elastic Container Registry)

Docker images are a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.

Here are some of the key features of Docker images:

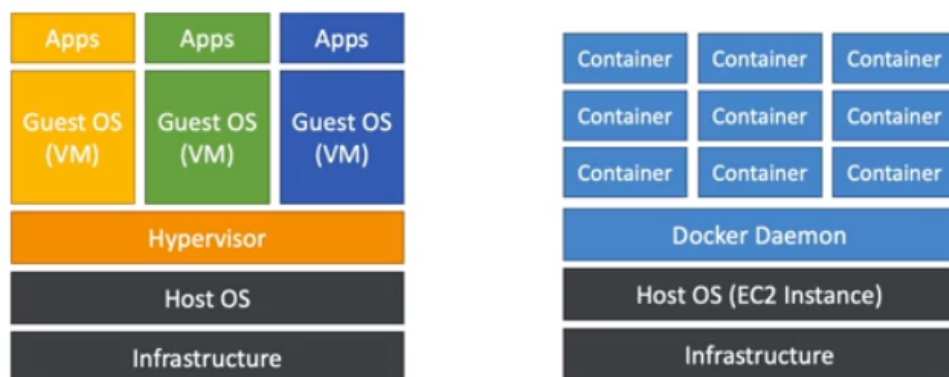
- **They are lightweight:** Docker images are typically much smaller than virtual machines, making them easier to store and transport.
- **They are portable:** Docker images can be run on any system that has Docker installed, regardless of the underlying operating system or hardware.
- **They are isolated:** Docker images are isolated from one another, meaning that they cannot affect each other. This makes it easier to develop and deploy applications without worrying about conflicts.
- **They are reproducible:** Docker images can be reproduced exactly, which makes it easy to share and deploy applications.

Docker images are created using a Dockerfile, which is a text file that contains the instructions for building the image. The Dockerfile specifies the base image to use, the packages to install, and the configuration settings.

9.

Docker versus Virtual Machines

- Docker is "sort of" a virtualization technology, but not exactly
- Resources are shared with the host => many containers on one server



10.

11. Hypervisor are used to manage ec2 instance ...

Sure. Here's a comparison of Docker and virtual machines (VMs) in AWS:

Feature	Docker	Virtual Machine
Definition	A software platform that creates and manages containers.	A hypervisor-based software program that creates a simulated computer environment.
Isolation	Containers are isolated from each other, but they share the host operating system.	Virtual machines are isolated from each other and from the host operating system.
Portability	Containers can be easily moved from one host to another.	Virtual machines are not as portable as containers.
Resource usage	Containers use less resources than virtual machines.	Virtual machines use more resources than containers.

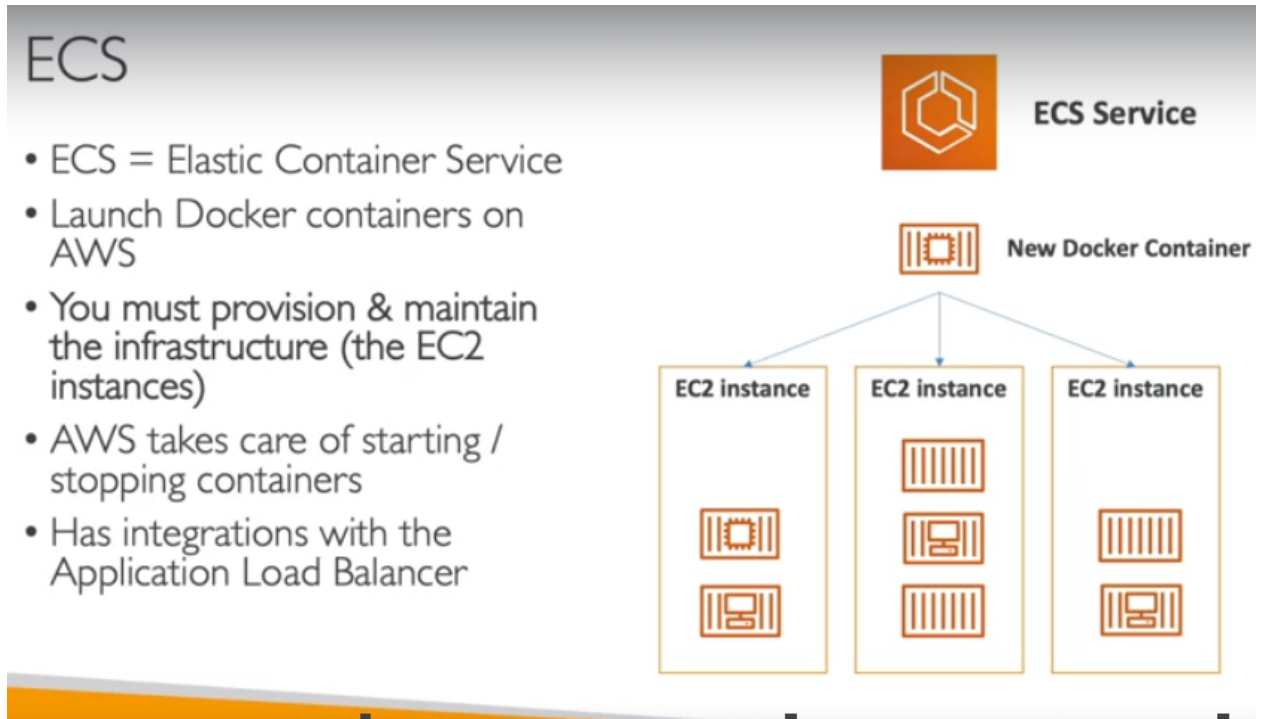
12.

Setup	Docker is easier to set up than virtual machines.	Virtual machines can be more complex to set up than Docker.
Application development	Docker is a good choice for developing and deploying applications.	Virtual machines can be used for a variety of purposes, including application development and deployment.
Cost	Docker is a more cost-effective solution than virtual machines.	Virtual machines can be more expensive than Docker.
AWS service	Amazon Elastic Container Service (ECS)	Amazon Elastic Compute Cloud (EC2)

13.

ECS, Fargate and ECR Overview

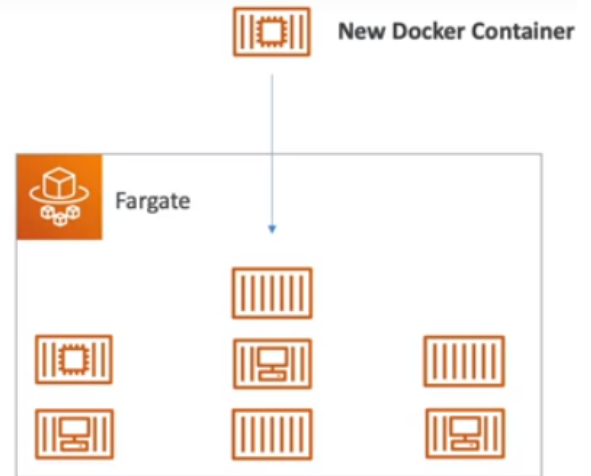
1. ECS is used to launch docker services on aws..
2. We have maintain and provision our own ec2 instance(infrastructre)
- 3.



4. As we can see in the diagram..we have multiple instances and we need to create them in advance..in the picture we can see there are diff containers which are already in our ec2 instance
5. Now the ECS service, if in any times it has a new docker container, it will be smart enough to find on which EC2 instance to place that docker container.
6. So anytime in the exam you see, I want to run docker containers on AWS, think of ECS.
7. Fargate

Fargate

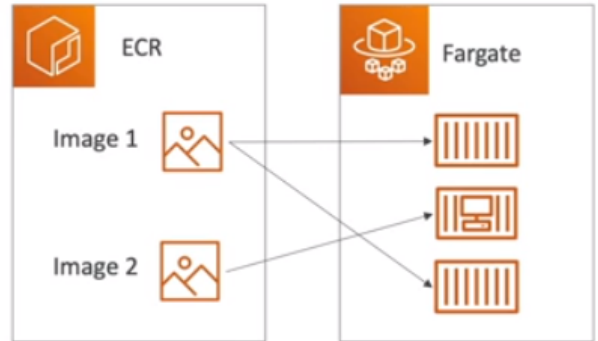
- Launch Docker containers on AWS
- You do not provision the infrastructure (no EC2 instances to manage) – simpler!
- Serverless offering
- AWS just runs containers for you based on the CPU / RAM you need



- 8.
9. Here in fargate we will not be having any instances..it is serverless
10. AWS will just run the containers that we need based on the specification of CPU and RAM for each container.
11. Fargate is here.
12. Say we have to have a new docker container to be run on Fargate, then Fargate will automatically run that container for us. We don't exactly know where, but it will be run.
13. ECR
14. For storing these docker images ..we need to main a contain registry..
15. Docker images are nothing but the project requirements ..like it has everything to run our application in the container

ECR

- Elastic Container Registry
- Private Docker Registry on AWS
- This is where you store your Docker images so they can be run by ECS or Fargate



- 16.
17. we have ECR and Fargate.
18. We're going to store our images of our application onto Amazon ECR, and then Fargate will be able to look at these images and create a container from them, and run them directly on the Fargate service.
19. So it could be one container here, one container there, but this is ECR, so we could have multiple images as well, creating different containers on Fargate.

Serverless introduction

Serverless in AWS is a computing model where you don't manage servers. AWS takes care of provisioning and managing the servers, so you can focus on developing and running your applications.

Here are some of the benefits of using serverless computing in AWS:

- **Cost-effectiveness:** You only pay for the resources that you use, so you can save money on infrastructure costs.
- **Scalability:** AWS can scale your applications automatically to meet demand, so you don't have to worry about provisioning or managing servers.
- **Reliable:** AWS has a proven track record of reliability, so you can be confident that your applications will be available when you need them.
- **Secure:** AWS offers a variety of security features to protect your applications, so you can be confident that your data is safe.

If you're looking for a cost-effective, scalable, and reliable way to run your applications, then serverless computing in AWS is a good option.

1.

What's serverless?

- Serverless is a new paradigm in which the developers don't have to manage servers anymore...
- They just deploy code
- They just deploy... functions !
- Initially... Serverless == FaaS (Function as a Service)
- Serverless was pioneered by AWS Lambda but now also includes anything that's managed: "databases, messaging, storage, etc."
- Serverless does not mean there are no servers...
it means you just don't manage / provision / see them

2.

3. So far in this course..we have seen amazon s3 which is serverless,next we have learn dynamodb and fargates for docker container..next we will learn lambda

So far in this course...



Amazon S3



DynamoDB



Fargate



Lambda

4.

Lambda Overview:

Why AWS Lambda



Amazon EC2

- Virtual Servers in the Cloud
- Limited by RAM and CPU
- Continuously running
- Scaling means intervention to add / remove servers



Amazon Lambda

- Virtual functions – no servers to manage!
- Limited by time - short executions
- Run on-demand
- Scaling is automated!

- 1.
2. In an aws ec2..the instance will run continuously even though some times we don't use it
3. And if we want to scale we can use an autoscaling group, but that means that we need to add or remove servers over time. That may be a little slow or there may be sometimes very complicated to implement.
4. In lambda we don't have servers to manage

Benefits of AWS Lambda

- Easy Pricing:
 - Pay per request and compute time
 - Free tier of 1,000,000 AWS Lambda requests and 400,000 GBs of compute time
- Integrated with the whole AWS suite of services
- Event-Driven: functions get invoked by AWS when needed
- Integrated with many programming languages
- Easy monitoring through AWS CloudWatch
- Easy to get more resources per functions (up to 10GB of RAM!)
- Increasing RAM will also improve CPU and network!

5.

AWS Lambda language support

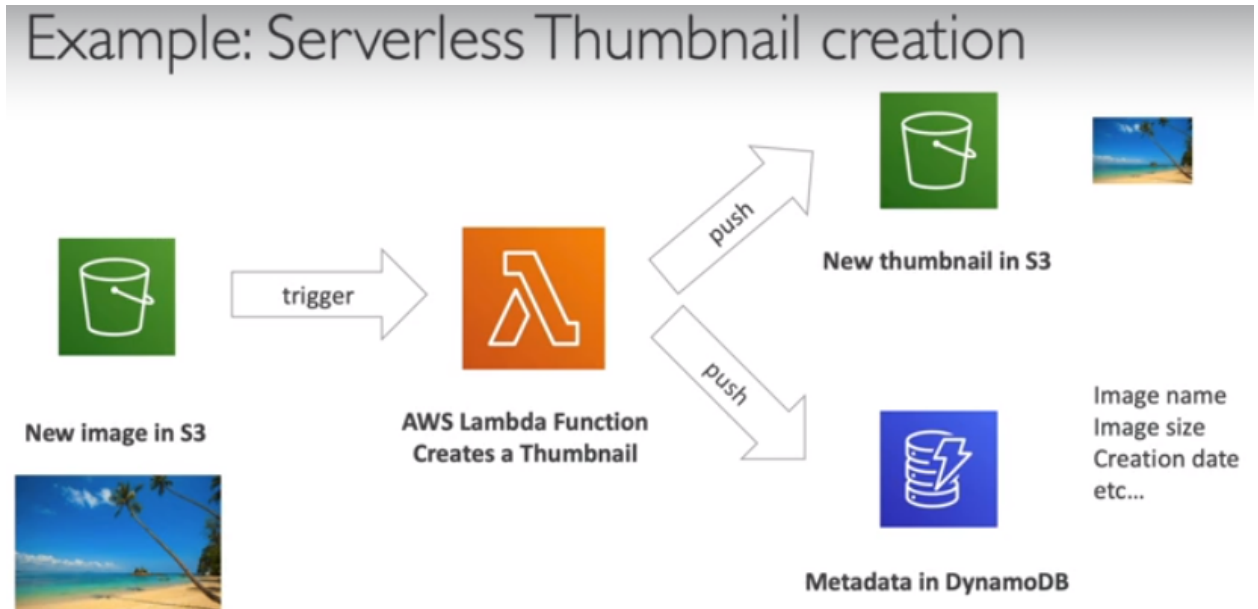
- Node.js (JavaScript)
- Python
- Java (Java 8 compatible)
- C# (.NET Core)
- Golang
- C# / Powershell
- Ruby
- Custom Runtime API (community supported, example Rust)
- Lambda Container Image
 - The container image must implement the Lambda Runtime API
 - ECS / Fargate is preferred for running arbitrary Docker images

6.

7. Example: thumbnail creation

8. If a user uploads an image into s3 bucket..then s3 buckets will trigger the function once the image is uploaded

9. and that Lambda function will take that image and will change it to create a thumbnail. It will push the thumbnail back into Amazon S3.
10. or it will also push some metadata about the thumbnail into DynamoDB. That includes the image size, the image name, the creation dates, etc, etc. And all of this is fully event-driven and fully serverless. With S3 we don't provision servers.



- 11.
12. Example: serverless CRON job
- 13.

Sure. A serverless cron job in AWS Lambda is a Lambda function that is invoked on a recurring schedule. This means that the Lambda function is executed at a specific time or interval, regardless of whether there is any traffic to the function.

To create a serverless cron job in AWS Lambda, you need to use the CloudWatch Events cron pattern. The cron pattern is a string that specifies the schedule at which the Lambda function should be invoked. The cron pattern can be used to specify a specific time, a specific interval, or a combination of both.

For example, the following cron pattern specifies that the Lambda function should be invoked every minute:

Example: Serverless CRON Job



14.

AWS Lambda Pricing: example

- You can find overall pricing information here: <https://aws.amazon.com/lambda/pricing/>
- Pay per calls:
 - First 1,000,000 requests are free
 - \$0.20 per 1 million requests thereafter (\$0.0000002 per request)
- Pay per duration: (in increment of 1 ms)
 - 400,000 GB-seconds of compute time per month if FREE
 - == 400,000 seconds if function is 1 GB RAM
 - == 3,200,000 seconds if function is 128 MB RAM
 - After that \$1.00 for 600,000 GB-seconds
- It is usually very cheap to run AWS Lambda so it's very popular

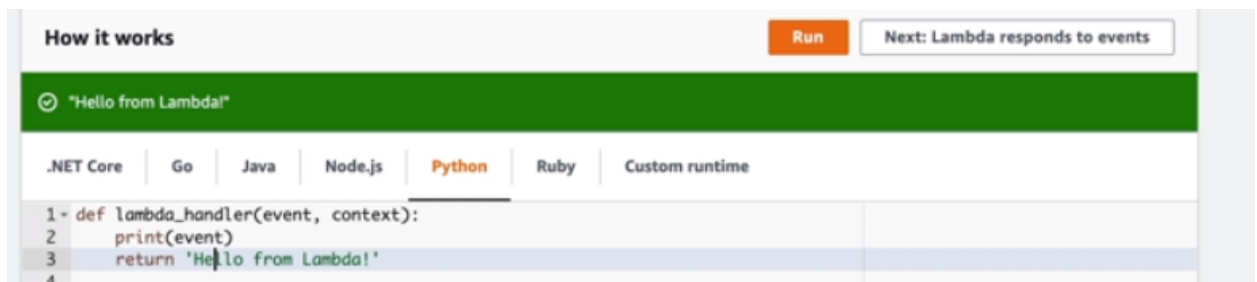
15. _____

16. So all in all the bottom line is that it's going to be very cheap to run Lambda on AWS. And so it's a very popular service to run your serverless applications and websites.

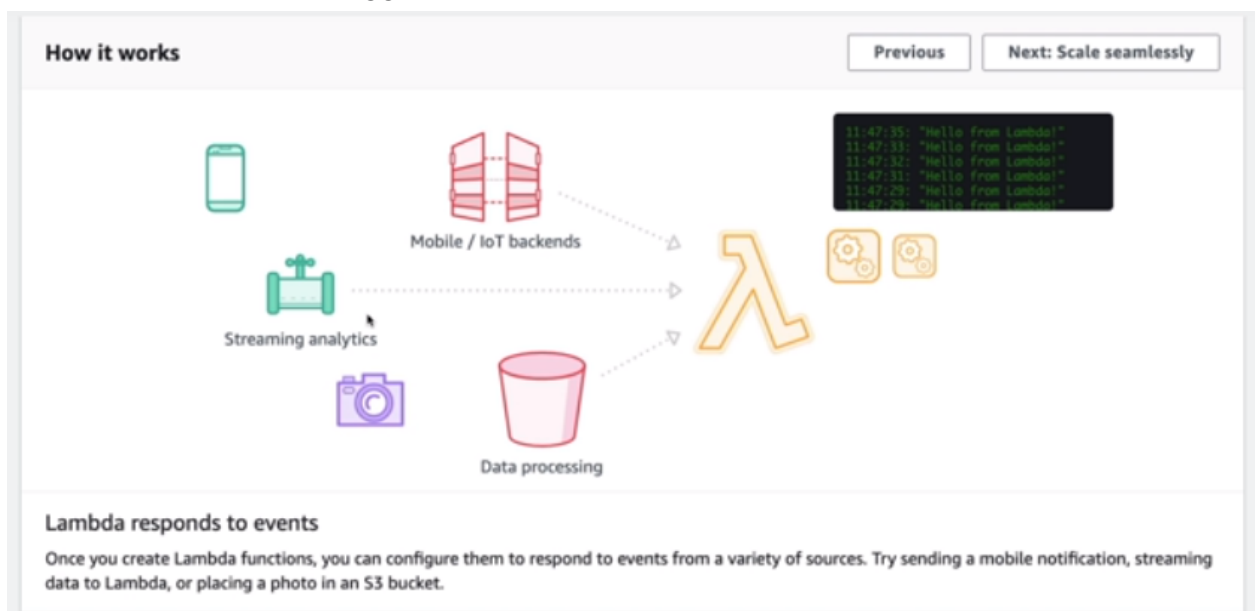
17. And going into the CCP exam, you need to remember that Lambda pricing is based on calls and service

AWS Lambda Hands On:

1. First go to lambda services ..in the compute section
2. We can go thru its homepage and check how it runs
3. Here we ran sample python program



- 4.
5. If we click run button it triggers a lambda function



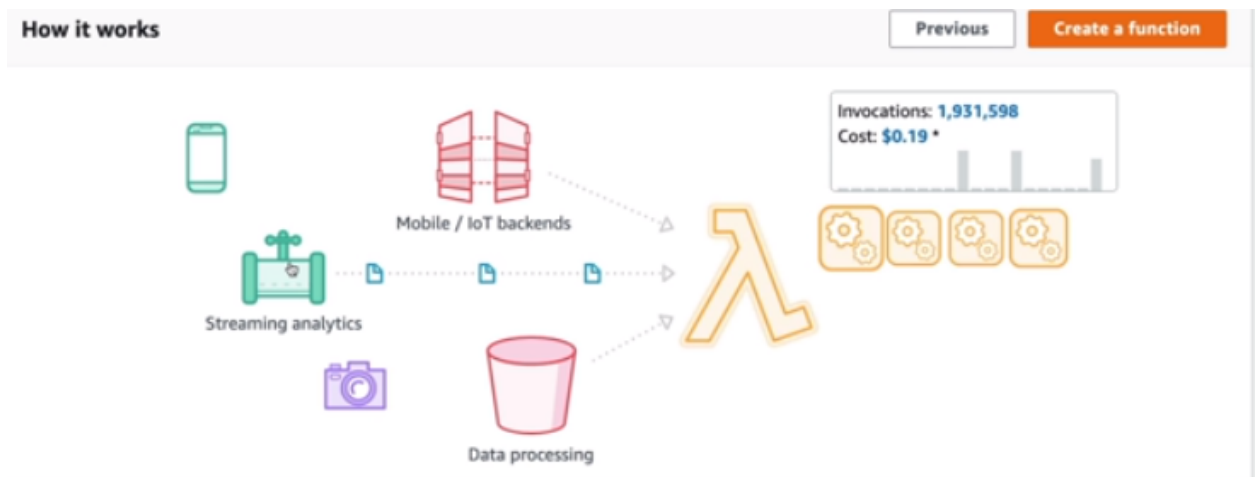
- 6.
7. Here if we place a image in s3 bucket for data processing ..the lambda function gets invoked ..similary for streaming analytics

Sure. I'll try my best to explain how Lambda functions work in AWS Lambda.

A Lambda function is a piece of code that runs in response to an event. For example, a Lambda function can be invoked when a file is uploaded to an S3 bucket, when a database record is updated, or when a user clicks a button on a web page.

When a Lambda function is invoked, it is executed in a stateless environment. This means that the function does not have any persistent state. When the function finishes executing, it is destroyed and any data that it was using is lost.

8.



9.

10. Here if we click on images continuously ..then s3.. invokes the lambda functions...and we are charged when invocation cross 1m

11. Next if we click on create function..we get 3 options to choose from

12. For this hands on we use a blue print

Create function [Info](#)

AWS Serverless Application Repository applications have moved to [Create application](#).

☐ Author from scratch
Start with a simple Hello World example.

☒ Use a blueprint
Build a Lambda application from sample code and configuration presets for common use cases.

☐ Container image
Select a container image to deploy for your function.

13.

14. Next we choose blueprint name..we choose hello world with python

15. Then we have to give a function name..we gave "demo-lambda"

```
1 import json
2
3 print('Loading function')
4
5
6 def lambda_handler(event, context):
7     #print("Received event: " + json.dumps(event, indent=2))
8     print("value1 = " + event['key1'])
9     print("value2 = " + event['key2'])
10    print("value3 = " + event['key3'])
11    return event['key1'] # Echo back the first key value
12    #raise Exception('Something went wrong')
13
```

16.

17. This code will be imported into our function..it will print 3 values and return one value..see code

18. Next click on create function..It will load our code into code editor

19. Next we have to give demo event and after saving the demo event click on test it runs our code

20. So basically from a programmer's perspective, as you can see you had just some code and it was uploaded into Lambda and then it was run by Lambda very quickly.

21. Here we can see the duration etc

REPORT RequestId: c42a9d8b-82d4-4039-92ac-6160306b17c9 Duration: 2.32 ms Billed Duration: 3 ms Memory Size: 128 MB

Memory Size: 128 MB Max Memory Used: 47 MB Init Duration: 135.61 ms

22. We can also configure our lambda by going into configuration

23. If we click on edit..we can edit basic settings

Description - *optional*

A starter AWS Lambda function.

Memory (MB) [Info](#)
Your function is allocated CPU proportional to the memory configured.

128 MB

Set memory to between 128 MB and 10240 MB

Timeout

0 min 3 sec

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

☒ Use an existing role
☐ Create a new role from AWS policy templates

Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

service-role/demo-lambda-role-eaek950w

[View the demo-lambda-role-eaek950w role](#) on the IAM console.

Cancel Save

24.

The timeout in AWS Lambda is the maximum amount of time that a Lambda function can run before it is terminated. The default timeout is 3 seconds, but you can increase it up to 15 minutes.

If a Lambda function exceeds its timeout, it will be terminated and will not return a result. This can happen if the function is taking too long to execute, or if it is making too many requests to other AWS services.

25.

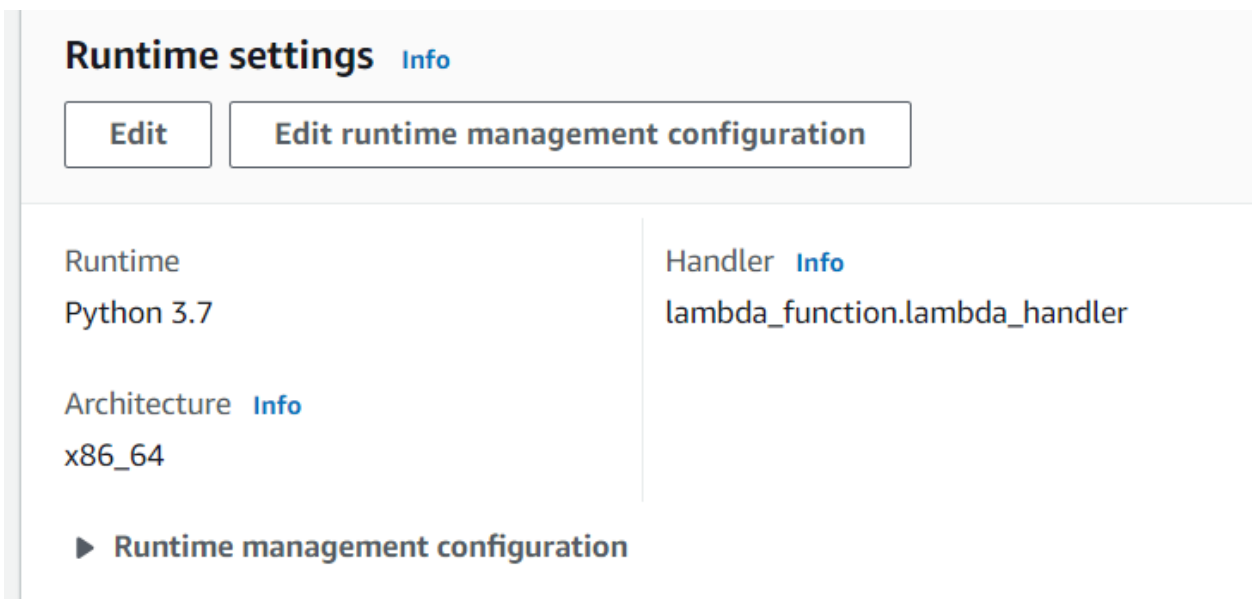
26. We can also monitor our lambda..we can check invocations,duration etc

Cloud Watch in AWS Lambda is a monitoring service that allows you to track the performance of your Lambda functions. You can use Cloud Watch to monitor the following metrics for your Lambda functions:

- **Invocations:** The number of times that your function has been invoked.
- **Errors:** The number of times that your function has failed to execute.
- **Duration:** The average amount of time that your function takes to execute.
- **Memory:** The amount of memory that your function uses.
- **CPU:** The amount of CPU that your function uses.

You can also use Cloud Watch to create alarms that will notify you when your Lambda functions are not performing as expected. For example, you could create an alarm that would notify you if your function's invocations or errors exceed a certain threshold.

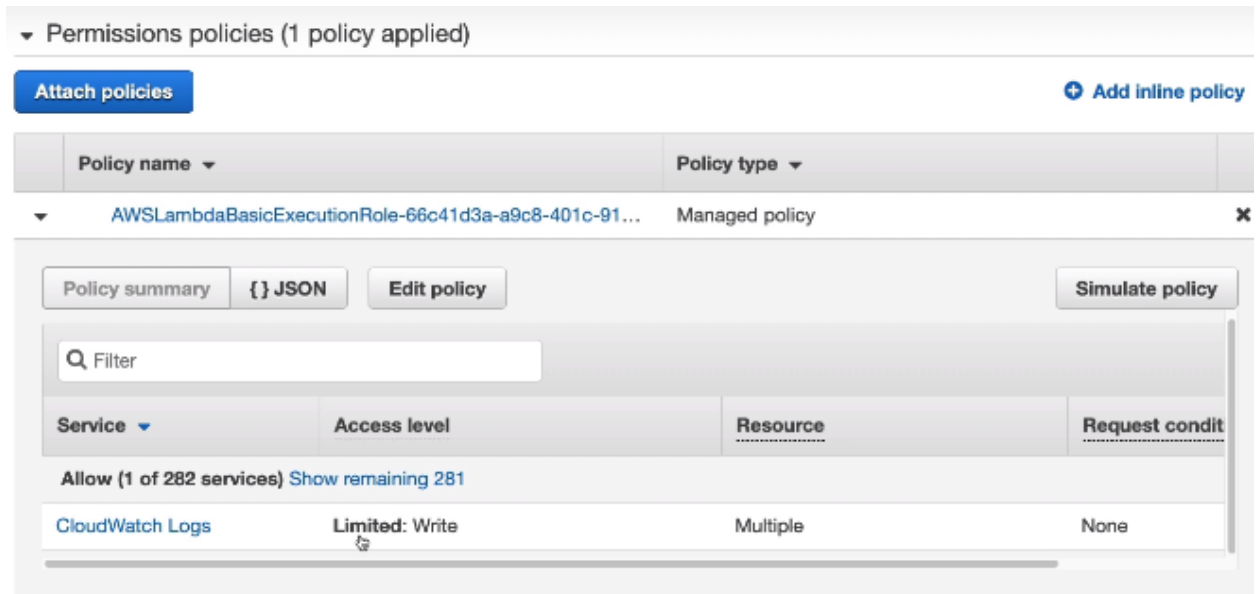
- 27.
28. Now if we want to modify the code..first we have to change the code and click on deploy..and after that we can test our code
29. If we scroll down we get



The screenshot shows the 'Runtime settings' page for an AWS Lambda function. At the top, there are two buttons: 'Edit' and 'Edit runtime management configuration'. Below these, the settings are organized into two columns. The left column contains 'Runtime' set to 'Python 3.7' and 'Architecture' set to 'x86_64', both with an 'Info' link. The right column contains 'Handler' set to 'lambda_function.lambda_handler', also with an 'Info' link. At the bottom, there is a section titled 'Runtime management configuration' with a right-pointing triangle icon.

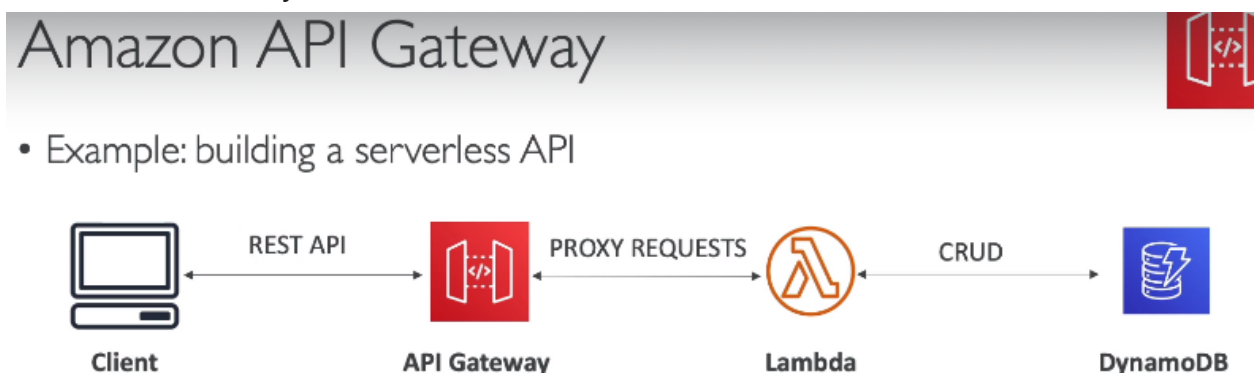
Runtime settings Info	
Runtime Python 3.7	Handler Info lambda_function.lambda_handler
Architecture Info x86_64	
▶ Runtime management configuration	

- 30.
31. Here handler is our lambda function and lambda_handle is our code
32. If we go to permission tab in configuration...we have a role name ..which we have created...if we go inside our role ..we can check its services and attached policies



API Gateway Overview

1. Here we are using lambda and we're Reading, Creating, Updating and Deleting data from DynamoDB, which are both server less technologies but we want external clients to be able to access our Lambda function.
2. But a Lambda function is not exposed as an API right away.
3. For this we need to expose it through an API Gateway which is going to provide the client with the rest HTTP API to connect directly to your website.
4. And so, as we can see the client will talk to the API Gateway. The API gateway will proxy the request to your lambda functions which will execute the transformations on your data.



- Fully managed service for developers to easily create, publish, maintain, monitor, and secure APIs
- 5.

- Serverless and scalable
 - Supports RESTful APIs and WebSocket APIs
 - Support for security, user authentication, API throttling, API keys, monitoring...
- 6.
7. So when you see going to exam which creates a server less API you need to think about API Gateway and Lambda.

AWS Batch Overview

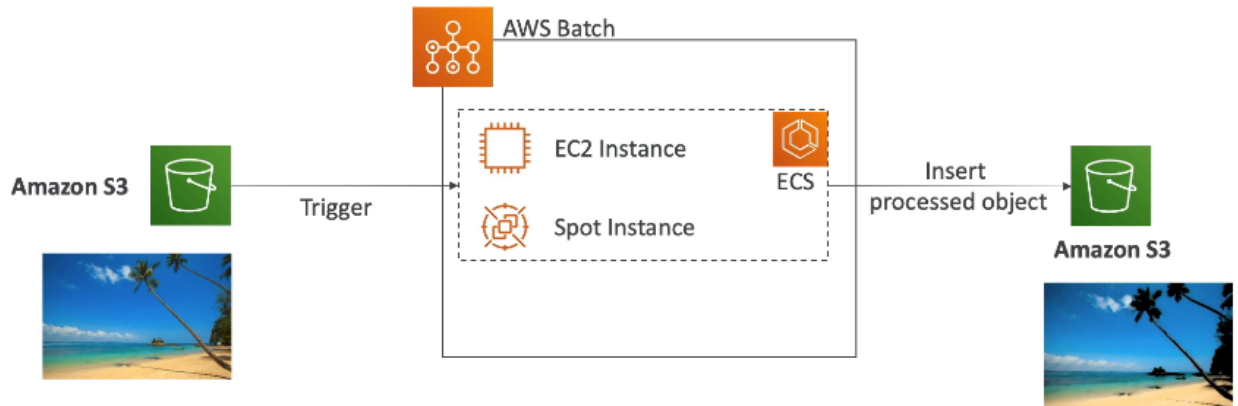
1. batch is a fully managed batch processing service that can allow you to do batch processing at any scale.
2. Well, a batch job is a job that has a start and an end.
3. for example, starts at 1 a.m. and finishes at 3 a.m.
4. So a batch job has a point of time when it happens and so the batch service will dynamically launch EC2 instances or Spot Instances to accommodate with the load that you have to run these batch jobs.

AWS Batch



- Fully managed batch processing at any scale
 - Efficiently run 100,000s of computing batch jobs on AWS
 - A “batch” job is a job with a start and an end (opposed to continuous)
 - Batch will dynamically launch EC2 instances or Spot Instances
 - AWS Batch provisions the right amount of compute / memory
 - You submit or schedule batch jobs and AWS Batch does the rest!
 - Batch jobs are defined as Docker images and run on ECS
 - Helpful for cost optimizations and focusing less on the infrastructure
- 5.

AWS Batch – Simplified Example



- 6.
7. Here in the example..we want to process the images submitted by users into amazon s3
8. So image will be put into Amazon S3, and this will trigger a batch job.
9. Batch will have ECS cluster that has ec2 instance or spot instances and it will make sure we have all computation resources to accommodate the load of batch jobs
10. And then these instances will be running your Docker images that will be doing your job.
11. And then maybe that job will be to insert the processed object. Maybe it's a filter on top of the image into another Amazon S3 buckets.

Batch vs Lambda

- Lambda:

- Time limit
- Limited runtimes
- Limited temporary disk space
- Serverless



- Batch:

- No time limit
- Any runtime as long as it's packaged as a Docker image
- Rely on EBS / instance store for disk space
- Relies on EC2 (can be managed by AWS)



12.

AWS Batch

- **Use cases:** AWS Batch is a good choice for running long-running, large-scale, or computationally intensive tasks. For example, you could use AWS Batch to process large amounts of data, train machine learning models, or render videos.
- **Execution:** AWS Batch executes your batch jobs on a fleet of Amazon EC2 instances. You can use Amazon EC2 instances of various types, such as on-demand instances, spot instances, or reserved instances.
- **Cost:** AWS Batch charges you for the compute resources that you use to run your batch jobs. You are charged for the number of vCPUs that your jobs use and the amount of memory that your jobs use.

AWS Lambda

- **Use cases:** AWS Lambda is a good choice for running short-lived, event-driven tasks. For example, you could use AWS Lambda to process HTTP requests, respond to database events, or generate reports.
- **Execution:** AWS Lambda executes your Lambda functions in isolated containers. You do not need to provision or manage servers for your Lambda functions.
- **Cost:** AWS Lambda charges you for the amount of time that your Lambda functions run. You are charged for the number of milliseconds that your functions run.

13.

Amazon LightSail

1. Aws lightsail gives us

- Virtual servers, storage, databases, and networking

Amazon Lightsail

- Virtual servers, storage, databases, and networking
- Low & predictable pricing
- Simpler alternative to using EC2, RDS, ELB, EBS, Route 53...
- Great for people with little cloud experience!

- 2.
3. It is for people that have little cloud experience and don't want to learn how the services work intricately, for example, how their networking works, how the storage works, the server works etc.

Amazon Lightsail



- Virtual servers, storage, databases, and networking
- Low & predictable pricing
- Simpler alternative to using EC2, RDS, ELB, EBS, Route 53...
- Great for people with little cloud experience!
- Can setup notifications and monitoring of your Lightsail resources
- Use cases:
 - Simple web applications (has templates for LAMP, Nginx, MEAN, Node.js...)
 - Websites (templates for WordPress, Magento, Plesk, Joomla)
 - Dev / Test environment
- Has high availability but no auto-scaling, limited AWS integrations

4. —
5. but from an exam perspective
6. if you see someone that has no cloud experience and need to get started quickly with a low and predictable pricing without configuring things much that will be Lightsail, otherwise Lightsail will almost always be a wrong answer.

AWS Lightsail Hands On:

1. Chatgpt -
<https://chat.openai.com/share/aee75818-b7c2-4bf6-8dec-0f40123f3e87>

2. See other online resources as well

Other compute Summary:

- 1.

Other Compute - Summary

- **Docker:** container technology to run applications
- **ECS:** run Docker containers on EC2 instances
- **Fargate:**
 - Run Docker containers without provisioning the infrastructure
 - Serverless offering (no EC2 instances)
- **ECR:** Private Docker Images Repository
- **Batch:** run batch jobs on AWS across managed EC2 instances
- **Lightsail:** predictable & low pricing for simple application & DB stacks

2. Lambda Summary

Lambda Summary

- Lambda is Serverless, Function as a Service, seamless scaling, reactive
- **Lambda Billing:**
 - By the time run x by the RAM provisioned
 - By the number of invocations
- **Language Support:** many programming languages except (arbitrary) Docker
- **Invocation time:** up to 15 minutes
- **Use cases:**
 - Create Thumbnails for images uploaded onto S3
 - Run a Serverless cron job
- **API Gateway:** expose Lambda functions as HTTP API

3. _____