Lambda

Lambda is the ultimate abstraction layer;

- Data centres
- Hardware
- Assembly Code/Protocols
- High Level Languages
- Operating Systems
- Application Layer/AWS APIs
- AWS LAmbda

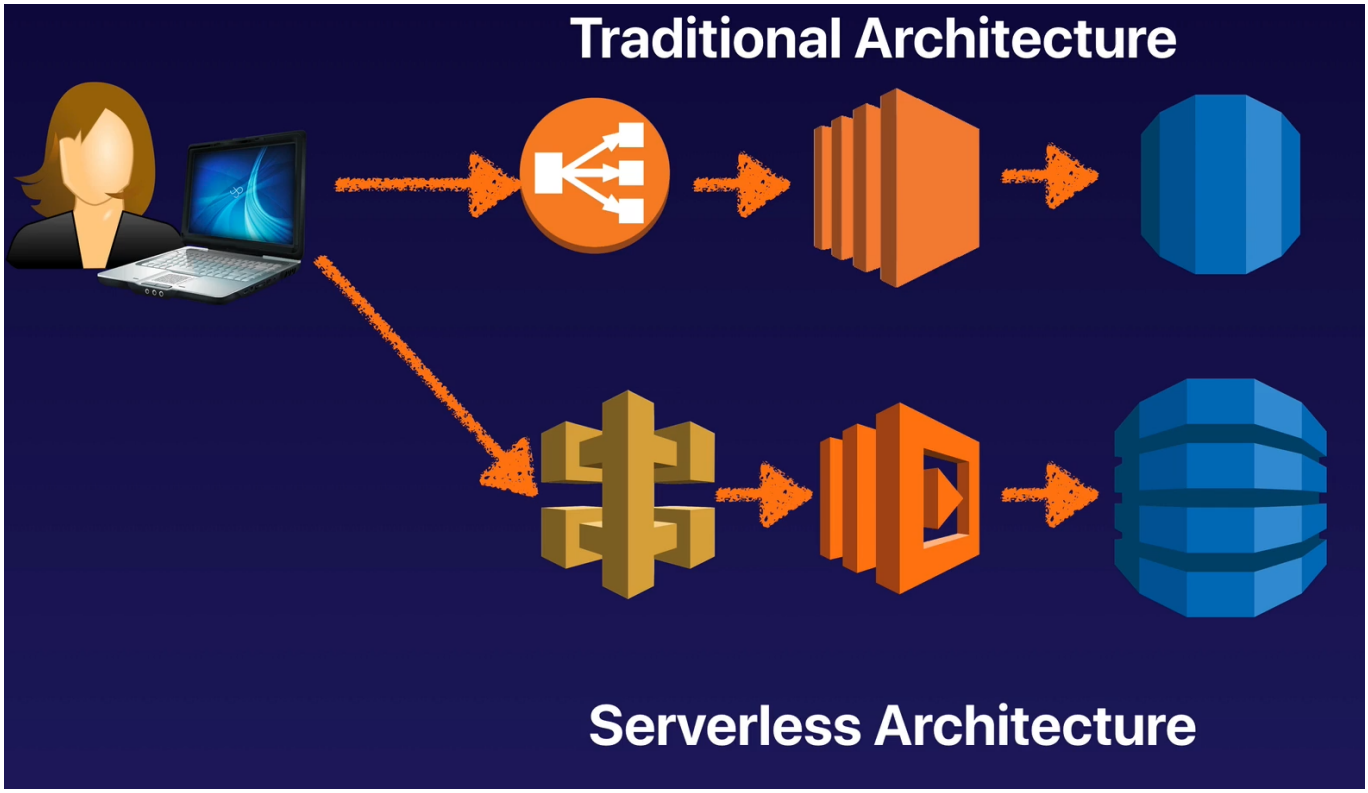
AWS Lambda is a compute service where you can upload your code and create a Lambda Function. AWS Lambda takes care of provisioning and managing the servers that you use to run the code. You don't have to worry about operating systems, pathing, scaling, etc.

You can use Lambda in the following ways;

- As an event-driven compute service where AWS Lambda runs your code in response to events. These events could be changes to data in an Amazon S3 bucket or an Amazon DynamoDB table
- As a compute service to run your code in response to HTTP requests using Amazon API Gateway or API calls made using AWS SDKs.

Traditional vs Serverless Architecture

- Traditional
    - user sends request → hits Route 53 → hits elastic load balancer → load balancer sends it to web servers → which then connects to back end database and send back a response to user
- Serverless
    - Send a response to API Gateway → which sends a response to lambda → which can write to dynamoDB or Aurora for example; and this can scale instantly with no worries about auto scaling, or EC2 instances becoming overwhelmed



Lambda supports the following languages:

- Node.js
- Java
- Python
- C#
- Go
- PowerShell

Lambda Priced

1. Number of Requests
    - First 1 million requests are free. $0.2 per 1 million requests thereafter
2. Duration
    - Duration is calculated from the time your code begins executing until it returns or otherwise terminates, rounded up to the nearest 100ms. The price depends on the amount of memory you allocate to your function. You are charged $0.00001667 for every GB-second used.

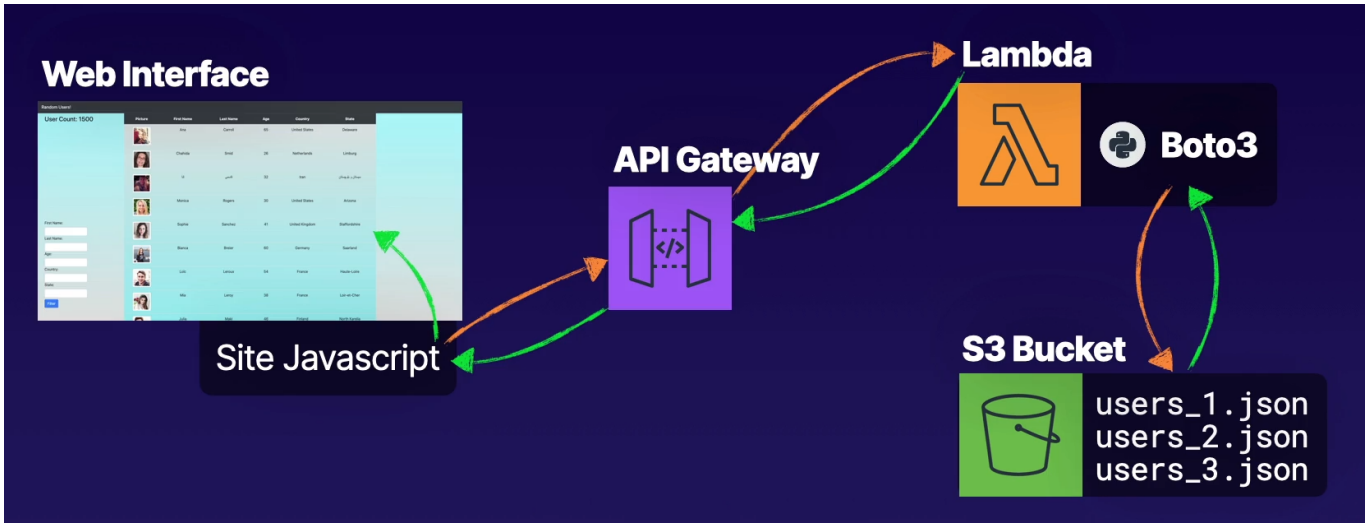
Benefits of Lambda

- No Servers
    - no system admins, no patching OS, no anti virus, no EC2 worries, etc.
- Continuous Scaling
- Super cheap!

Exam Tips

- Lambda Scales out (not up) automatically
- Lambda functions are independent, 1 event = 1 function
- Lambda is serverless
- Know what services are serverless
    - RDS isn't serverless, except for Aurora Serverless
    - DynamoDB, S3, Lambda are all serverless
    - EC2 is not serverless
- Lambda functions can trigger other lambda functions, 1 event can = x functions if functions trigger other functions
- Architectures can get extremely complicated, AWS X-ray allows you to debug what is happening
- Lambda can do things globally, you can use it to back up S3 buckets to other S3 buckets etc
- Know your triggers
    - RDS can't trigger lambda

Analyzing Data from S3 - Lab



Introduction

In this lab, we're tasked with completing a Lambda function that collects data from S3, performs some basic formatting, and returns it to API Gateway to be loaded into a simple web interface.

Scenario

You've been brought into a team to complete a proof of concept for a low cost employee directory for your company. The team has created a simple web interface, organized some placeholder employee information, and placed it in S3. They're having trouble getting the data from S3 into Lambda and need your assistance. You will need to collect all 1500 placeholder records from JSON files stored in an S3 bucket and return them in a single return from the Lambda function provided in the lab environment.

Additional Resources

Boto3 S3 List Objects V2

Boto3 Get Object

Lambda Function

Solution

Log in to the AWS Management Console using the credentials provided for the lab. In another browser tab, open the **random-users** website provided for the lab. The site won't load yet because you haven't assigned the Lambda function an action.

Investigate the Lab Environment

1. From the AWS Management Console, navigate to *S3* using the *Services* menu or the unified search bar. You should see two buckets in your account:
   - `random-users-<ACCOUNT_NUMBER>`, which is your static website.
   - `random-users-data-<ACCOUNT_NUMBER>`, which is the data that will populate the website.
2. Select the `random-users-data-<ACCOUNT_NUMBER>` bucket, then select **users_1.json**.
3. Use the *Object actions* dropdown in the top right corner to select **Download**.
4. Open the file and review the user data. You will collect the data from all 3 objects and organize it into a single object that can be returned in the web interface.
5. Close the file and navigate back to the AWS Management Console.

Create the Employee Directory Using Objects Keys and Data from S3

1. Navigate to *Lambda* using the *Services* menu or the unified search bar. You should see 2 Lambda functions in your account.
2. Select the **Users_primary** function.
3. In the *Code Source* section, select **function.py** and review the code.

4. Replace the existing code with the *function_solved.py* code provided in the lab resources.
5. In a new browser tab, navigate to *S3* and copy the `random-users-data-<ACCOUNT_NUMBER` bucket name.
6. Navigate back to Lambda and paste the bucket name on the `s3_bucket =` line.
7. Click **Deploy**. The Lambda function executes a number of tasks:
    - Combines data from multiple S3 buckets into a single JSON object.
    - Initializes a list to hold this data before returning it.
    - Initializes a boto3 S3 client to interact with S3.
    - Lists the objects in the bucket and collects the object keys that begin with `users-`.
    - Combines the returned data with the existing data list. What makes this programmatic is that you can add or remove users files from the S3 bucket, and that changes how many users are loaded in the web interface.
    - Retrieves the data from the objects in the bucket and concatenates it so you see just one list of data in the web interface.

Observe the Results on the Web Interface

1. After the changes are successfully deployed, navigate to the `random-users` website. You may need to refresh the page and wait a few moments for the data to load. All 1500 users should load correctly.
2. Navigate back to the AWS Management Console, then navigate to *S3*.
3. Check the checkbox to the left of the `random-users-data-<ACCOUNT_NUMBER>` bucket, then select the **users_1.json** object.
4. Click **Delete**, then type `permanently delete` into the text field and click **Delete objects**.
5. After the file is successfully deleted, click **Close** to return to your S3 bucket.
6. Navigate back to the `random-users` website and refresh the page. You should now have 1000 employee records instead of 1500.

Buiild an Alexa Skill - Lab

AWS Console → S3 → Create → Name it → Create

Edit public access settings → make it public

Select bucket → Permissions → bucket policy

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "PublicReadGetObject",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": "Paste bucket ARN here"
        }
    ]
}
```

→ Save

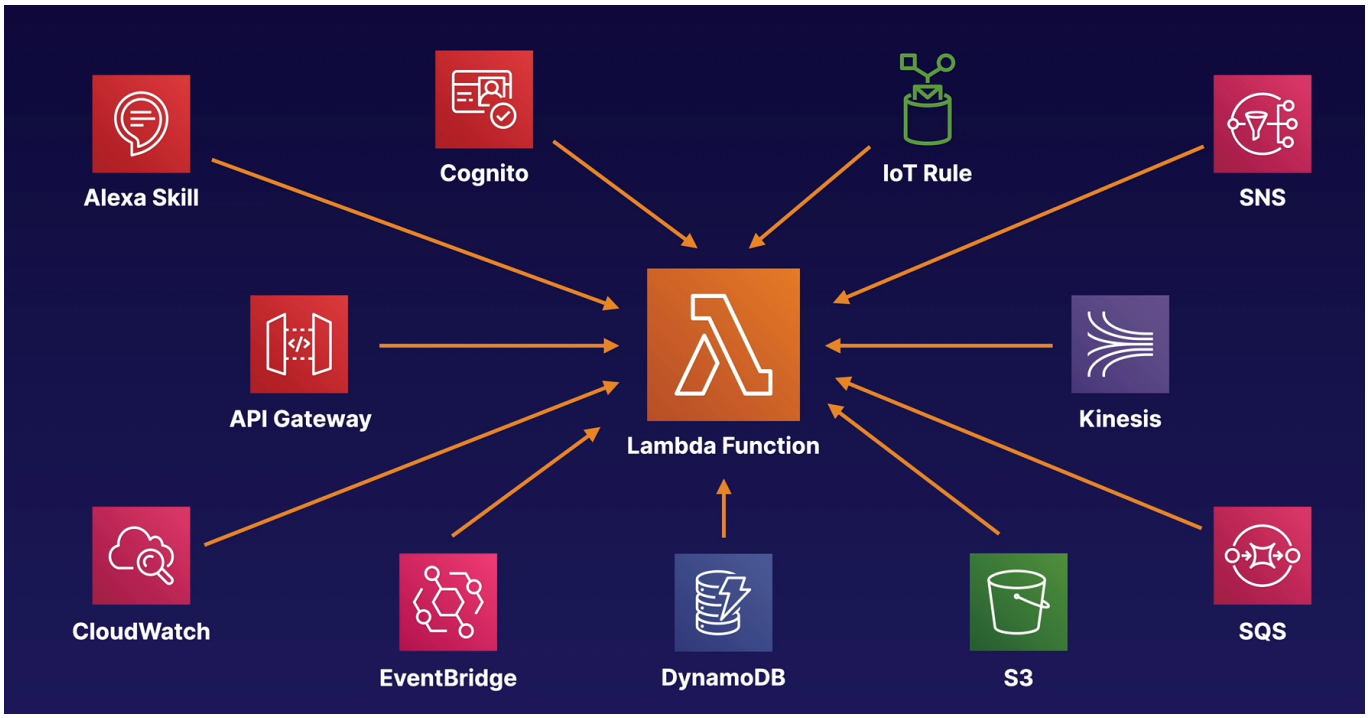
AWS → Amazon Polly → Enter some random text → Click S3 Sythensize, and enter the S3 bucket name → This will create a MP3 version of the file, and save it to the bucket.

AWS → S3 → The mp3 file should be here now.

AWS → Laambda → Create → AWS Serverless Application Repository → Alexa-skills-kit-nodejs-factskill → deploy

Serverless Application Model

- CloudFormation extension optimized for serverless applications
- New types: functions, APIs, tables
- Supports anything CloudFormation supports
- Run serverless applications locally
- Package and deploy using CodeDeploy



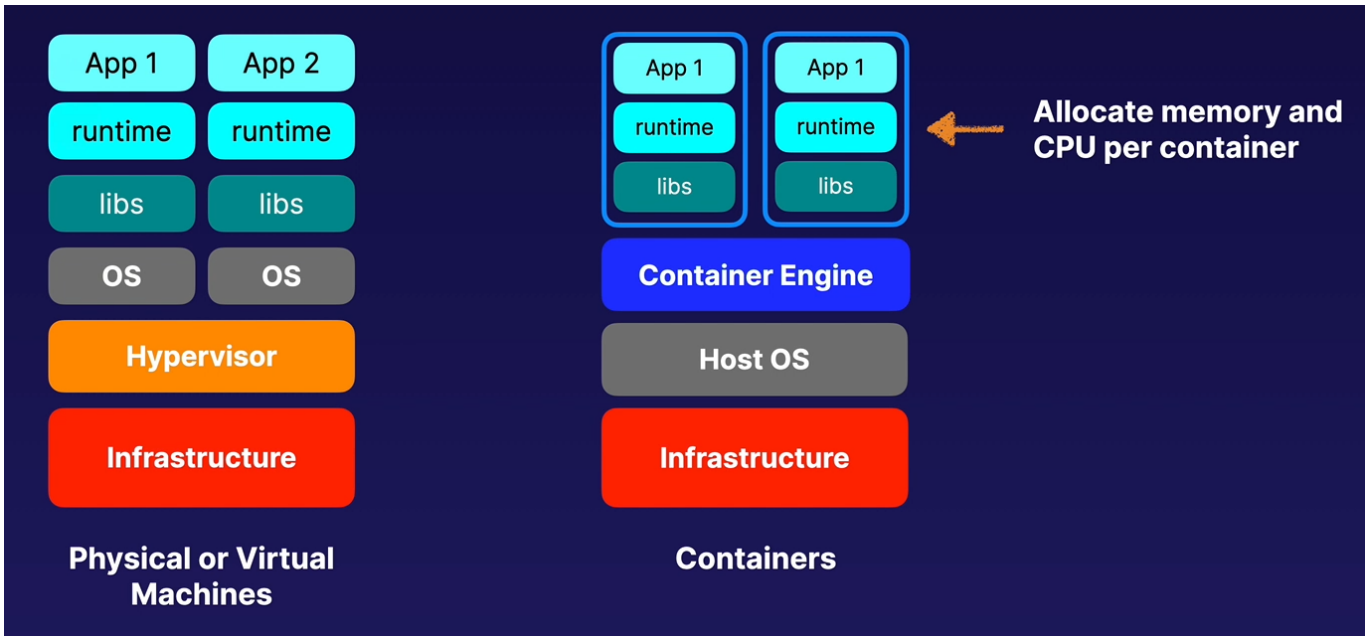
SSH into EC2 and install SAM CLI

```
sam init
```

Elastic Container Service (ECS)

What are Containers and Docker?

- A container is a package that contains an application, libraries, runtime, and tools required to run it
- Run on a container engine like Docker
- Provides the isolation benefits of virtualiztion with less overhead and faster starts than VMs
- Containered applications are portable and offer a consistent environment



What is ECS

- managed container orchestration service
- Create clusters to manage fleets of container deployments
- ECS manages EC2 or Fargate instances
- Schedules containers for optimal placement
- Defines rules for CPU and memory requirements
- Monitors resource utilization
- Deploy, update, roll back
- Free...
    - the scheduling and orchestrating components are free, as well as the clusters
    - but any EC2 instances or Fargate tasks are not free
- VPC, security groups, EBS volume integration
- ELB
- CloudTrail and CloudWatch

Cluster

Logical collection of ECS resources -- either ECS EC2 instances or Fargate instances

Task Definition

Defines your application. Similar to a Dockerfile but for running containers in ECS. Can

contain multiple containers

Container Definition
Inside a task definition, it defines the individual containers a task uses. Controls CPU and memory allocation and port mappings.

Task
Single running copy of any containers defined by a task definition. One working copy of an application (e.g. DB and web containers)

Service
Allows task definitions to be scaled by adding tasks. Defines minimum and maximum values

Registry
Storage for contianer images (e.g. Elastic Container Registry (ECR) or Docker Hub). Used to download images to create containers.

Fargate

- Serverless container engine
- Eliminates need to provision and manage servers
- Specify and pay for ersources per application
- Works with both ECS and EKS
- Each workload runs in its own kernel
- Isolation and security
- Choose EC2 instead if
    - Compliance requirements
    - Require broader customization
    - Require GPUs

EKS

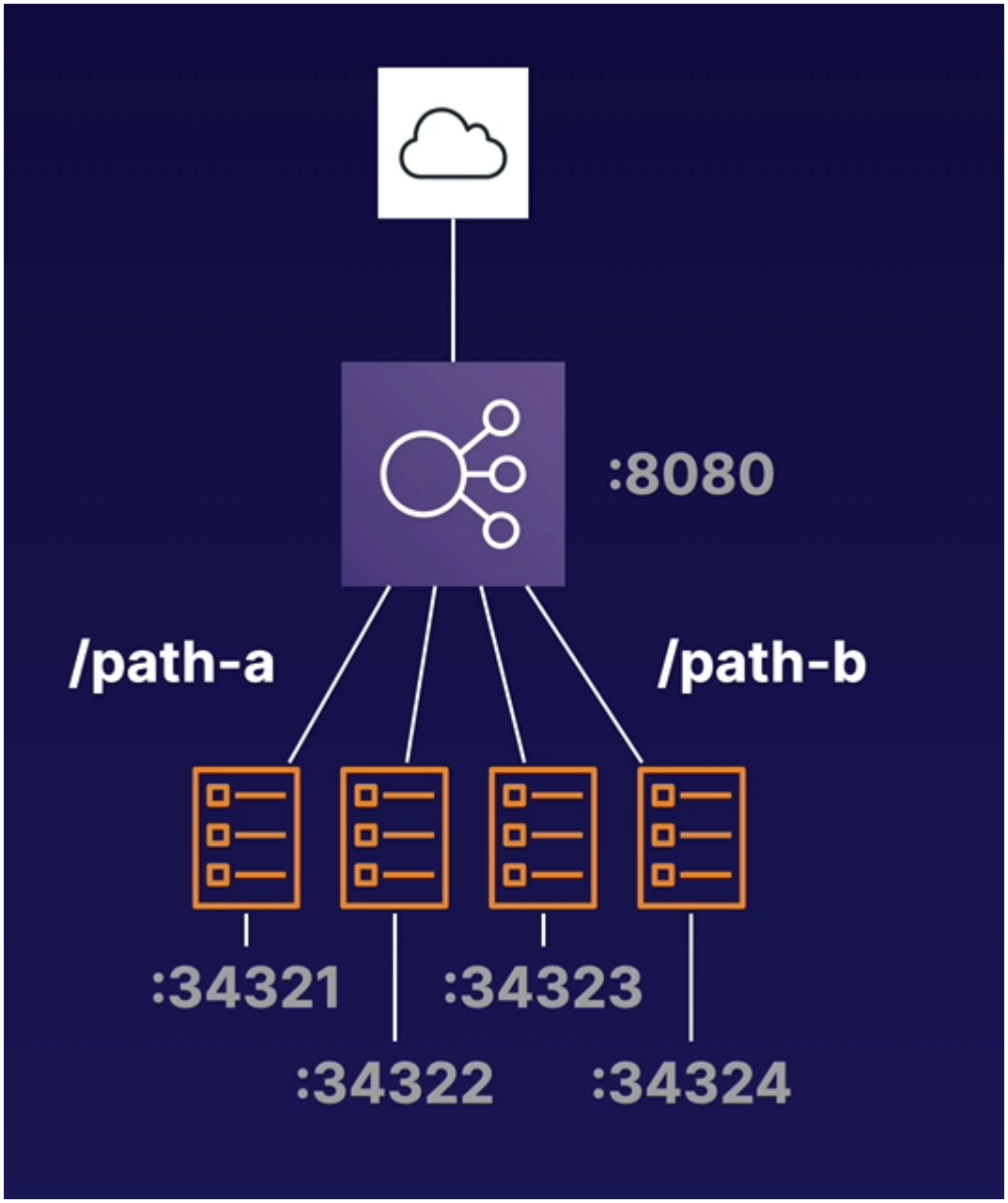
- Elastic Kubernetes Service
- K8s is open-source software that lets you deploy and manage containerized applications at scale
- Same toolset on-premises and in cloud
- Containers are grouped in pods
- Like ECS, supports both EC2 and Fargate
    - Why use EKS
        - Already using K8s
        - Want to migrate to AWS

ECR

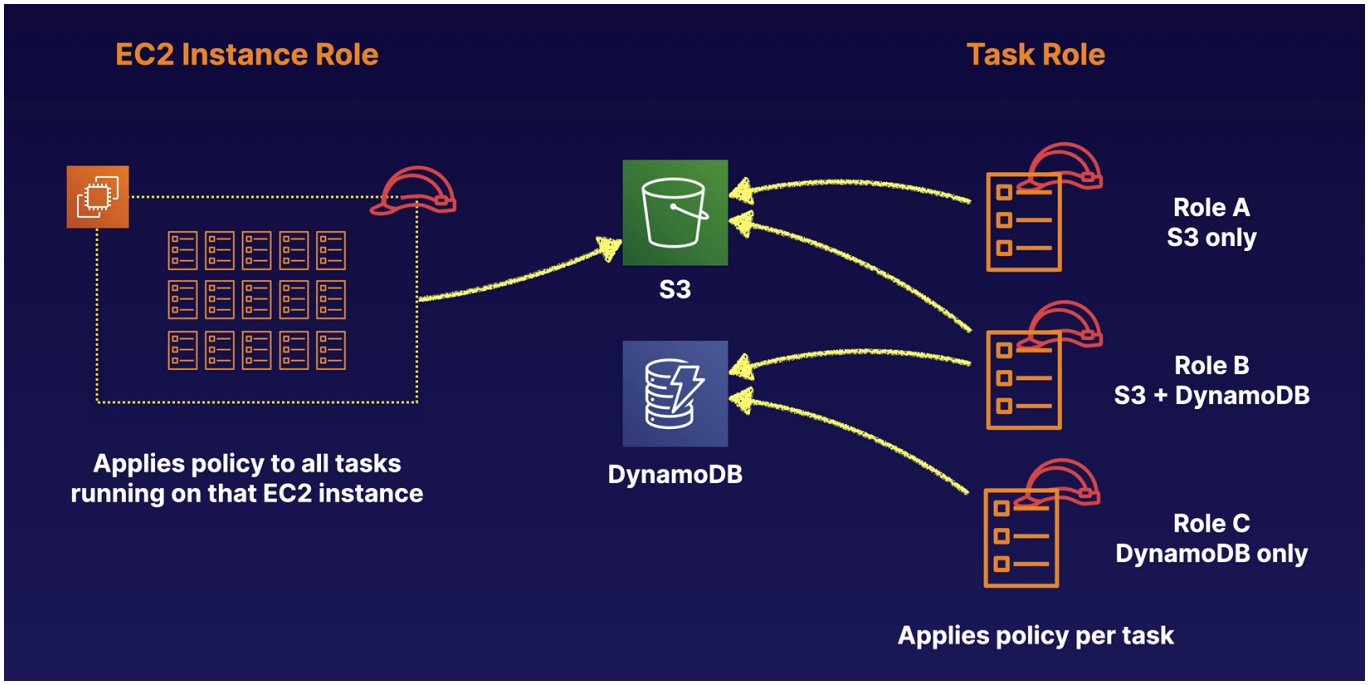
- Managed Docker container registry
- Store, manage, and deploy images
- Integrated with ECS and EKS
- Works with on-premises deployments
- Highly available
- Integrated with IAM
- Pay for storage and data transfer

ECS + ELB

- Ditribute traffic evenly across tasks in your service
- Supports ALB, NLB, CLB
- Use ALB to route HTTP/HTTPS (layer 7) traffic
- Use NLB or CLB to route TCP (layer 4) traffic
- Supported by both EC2 and Fargate launch types
- ALB allows:
    - Dynamic host port mapping
    - Path-based routing
    - Priority rules
- ALB is recommended over NLB or CLB



ECS Security



AWS → ECS → httpd (Sample-app) → Next → default → next → name the cluster → next → create

View service → Tasks → select task → copy public ip → open in browser to view the app

Quiz

1. On Friday morning your marketing manager calls an urgent meeting to celebrate that they have secured a deal to run a coordinated national promotion on TV, radio, and social media over the next 10 days. They anticipate a 500x increase on site visits and trial registrations. After the meeting you throw some ideas around with your team about how to ensure that your current 1 server web site will survive. Which of these best embody the AWS design strategy for this situation.

- Create a duplicate sign up page that stores registration details in DynamoDB for asynchronous processing using SQS & Lambda.
- Work with your web design team to create some web pages with embedded JavaScript to emulate your 5 most popular information web pages and sign up web pages. Host those pages on S3.
    - Use NoSQL database to collect customer registration for asynchronous processing, and SQS backed by scalable compute to keep up with the requests. An AWS solution for this situation might include S3 static web pages with client side scripting (JavaScript) to meet high demand of information pages.

2. What AWS service can help you to understand how your Lambda functions are performing?

- AWS X-Ray
    - AWS X-Ray helps developers analyze and debug production, distributed applications, such as those built using a microservices & serverless architectures. With X-Ray, you can understand how your application and its underlying services are performing to identify and troubleshoot the root cause of performance issues and errors.

3. You have created a serverless application to add metadata to images that are uploaded to a specific S3 bucket. To do this, your lambda function is configured to trigger whenever a new image is created in the bucket. What will happen when multiple users upload multiple different images at the same time?

- Multiple instances of the Lambda function will be triggered, one for each image
    - Each time a Lambda function is triggered, an isolated instance of that function is invoked. Multiple triggers result in multiple concurrent invocations, one for each time it is triggered

4. What does the common term 'Serverless' mean according to AWS

- A native Cloud Architecture that allows customers to shift more operational responsibility to AWS.
- The ability to run applications and services without thinking about servers or capacity provisioning.
    - 'Serverless' computing is not about eliminating servers, but shifting most of the responsibility for infrastructure and operation of the infrastructure to a vendor so that you can focus more on the business services, not how to manage the infrastructure that they run on. Billing does tend to be based on simple units, but the choice of services, intended usage pattern (RIs), and amount of capacity needed also influences the pricing.

5. Which of the following services can invoke a Lambda function synchronously (with functionality built-in with the invoking service)?

- Kinesis Data Firehose
- API Gateway
- Amazon Lex
    - ALB, Cognito, Lex, Alexa, API Gateway, CloudFront, and Kinesis Data Firehose are all valid direct (synchronous) triggers for Lambda functions. S3 is one of the valid asynchronous triggers.

6. Lambda pricing is based on which of these measurements after the free tier?

- The amount of memory assigned.
- The number of requests for each time the lambda executes in response to an event notification, or invoke call.
- Duration of each request (in ms).
    - Lambda billing is based on both The MB of RAM reserved and the execution duration in 100ms units. The number of requests for each time the lambda executes in response to an event notification, or invoke call is one of the factors involved in Lambda pricing. As of December 2020, the Lambda compute duration is billed in 1ms increments instead of being rounded up to the nearest 100 ms increment per invoke. For example, a function that runs in 30ms on average used to be billed for 100ms. Now, it will be billed for 30ms resulting in a 70% drop in its duration spend. Reference: AWS Lambda changes duration billing granularity from 100ms down to 1ms.

7. In which direction(s) does Lambda scale automatically?

- Out
    - Lambda scales out automatically - each time your function is triggered, a new, separate instance of that function is started. There are limits, but these can be adjusted on request.

8. As a DevOps engineer you are told to prepare a complete solution to run a piece of code that requires multi-threaded processing. The code has been running on an old custom server using a 4 core Intel Xeon processor. Which of these options best describes the AWS compute services that could be used for multi-threaded processing?

- EC2, ECS, & Lambda.
    - The exact ratio of cores to memory has varied over time for Lambda instances, however Lambda like EC2 and ECS supports hyper-threading on one or more virtual CPUs (if your code supports hyper-threading).

9. You have created a simple serverless website using S3, Lambda, API Gateway and DynamoDB. Your website will process the contact details of your customers, predict an expected delivery date of their order and store their order in DynamoDB. You test the website before deploying it into production and you notice that although the page executes, and the lambda function is triggered, it is unable to write to DynamoDB. What could be the cause of this issue?

- Your lambda function does not have sufficient Identity Access Management (IAM) permissions to write to DynamoDB.

- Like any services in AWS, Lambda needs to have a Role associated with it that provide credentials with rights to other services. This is exactly the same as needing a Role on an EC2 instance to access S3 or DDB.