Contents

[Amazon Instance Compute Cloud 2](#_Toc79343501)

[Amazon EC2 Instance Lifecycle 6](#_Toc79343502)

[Demonstration: Launch the Employee Directory Application on Amazon EC2 8](#_Toc79343503)

[Container Services 15](#_Toc79343504)

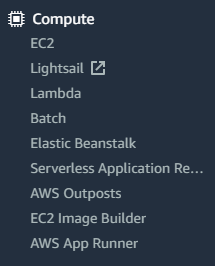
[Serverless 18](#_Toc79343505)

[AWS Lambda 19](#_Toc79343506)

**MODULE 2 – AWS Compute** 

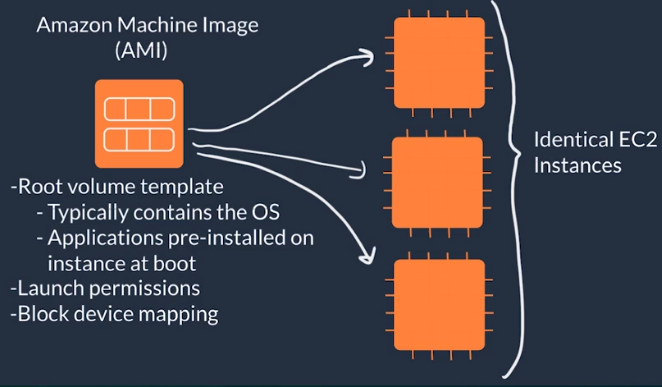
Amazon EC2 **is a web service** that provides secure, resizable compute capacity in the cloud. It allows you to **provision virtual servers called EC2 instances**.

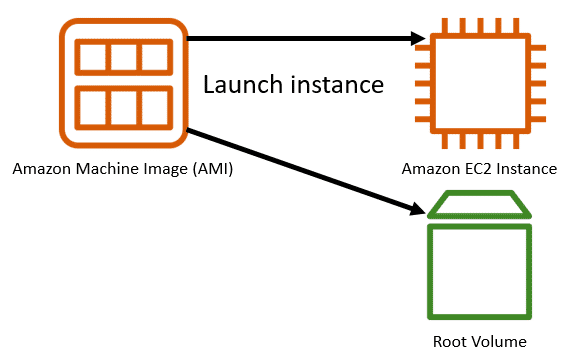
At a fundamental level, three types of compute options are available – virtual machines (VMs), container services, and server-less.

 **AMI = Cake Recipe or Class** and **EC2 Instance = Cake** from the recipe or **Object**.

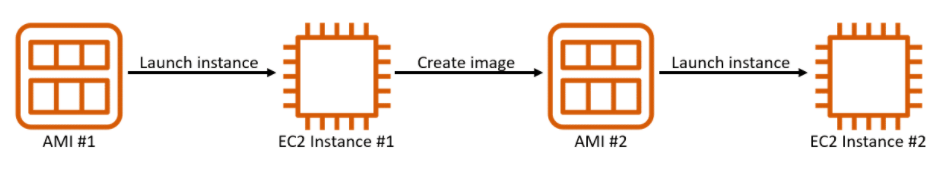
# Amazon Instance Compute Cloud

* In AWS, virtual machines are called Amazon Elastic Compute Cloud, or Amazon EC2.
* Behind the scenes, AWS operates and manages the host machines and the hypervisor layer.
* AWS also installs the virtual machine operating system, called the guest operating system.
* Some AWS compute services use Amazon EC2 or use virtualization concepts under the hood.
* AWS supports a range of operating systems including - Linux, MacOS, Ubuntu, Windows and more.
* To select an OS for your server, you need to select an **Amazon Machine Image (AMI)**. AMI will have below configuration’s in place -
  + Root Volume Template -> typically contains the OS, applications pre-installed on instance at boot.
  + Launch permissions
  + Block device mapping
* You can launch one or many instances from a single AMI which will create multiple EC2 instances all having the same configurations.



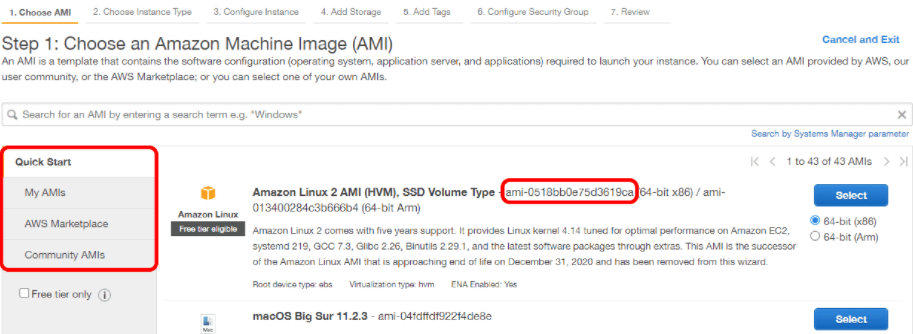


* You could create an AMI from your running instance and use the AMI to start a new instance. That way, your new instance would have the same configurations as your current instance, because the configurations set in the AMIs are the same.

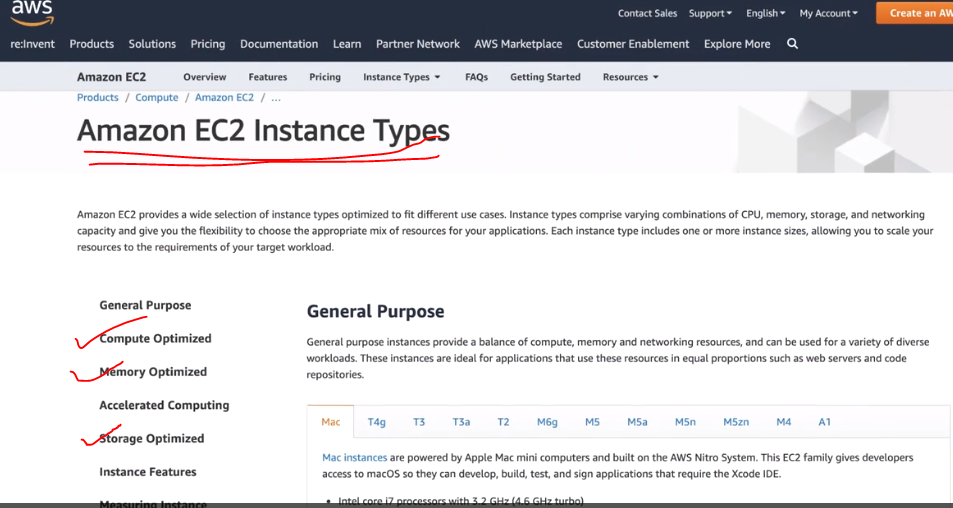


* Some AMI’s are provided by AWS and others are provided by the community which can be found using the “**AWS Marketplace**”.

For example, AMI-Linux-2 we selected for our employee application is provided by AWS which is pre-loaded and optimized Linux image having long term support provided by AWS.

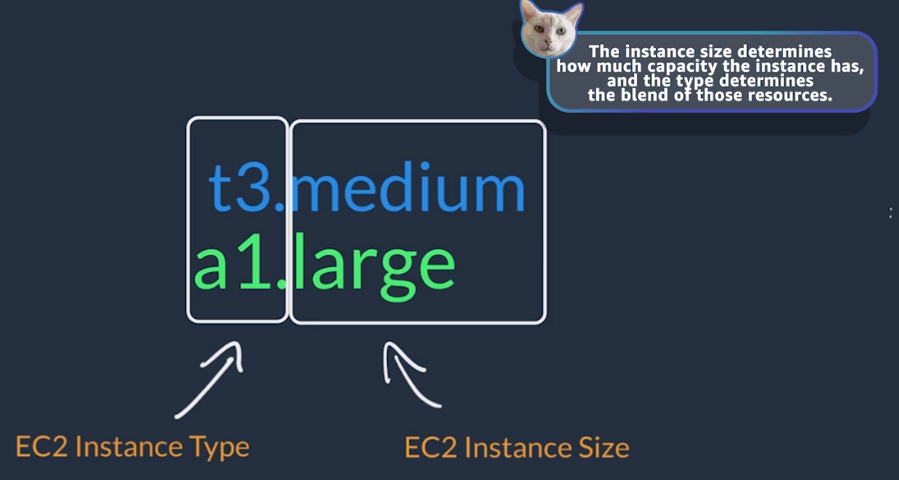


* Each AMI in the AWS Management Console has an AMI ID, which is prefixed by “ami-”, followed by a random hash of numbers and letters. The IDs are unique to each AWS Region.
* Apart of selecting the AMI, you can configure the “Instance Type” and “Instance Size” which will determine the compute and network capabilities.

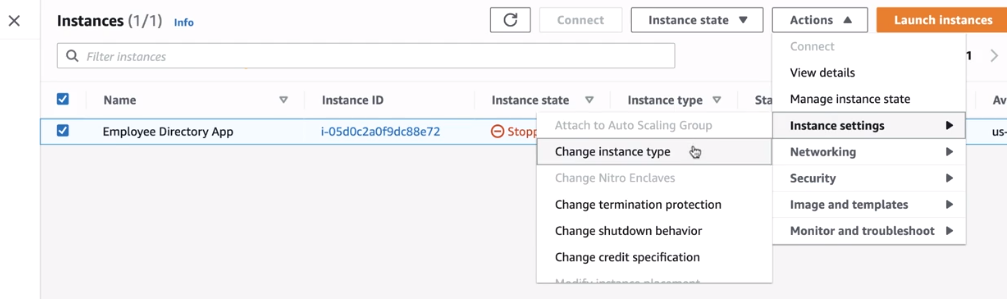


For example, the **g-instance type family** are optimized for graphic intensive applications such as 3D Visualizations and Video Encoding.

The **m5 ec2 general instance family**, provides a balance of resources and it’s great for applications which use these resources in equal proportions like web servers, our employee directory app or code repositories. Refer snippet below when an instance is triggered.



EC2 can be configurable with few clicks as shown below via API calls.

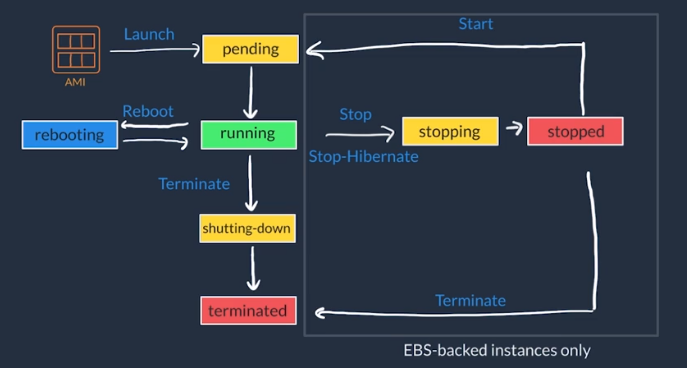


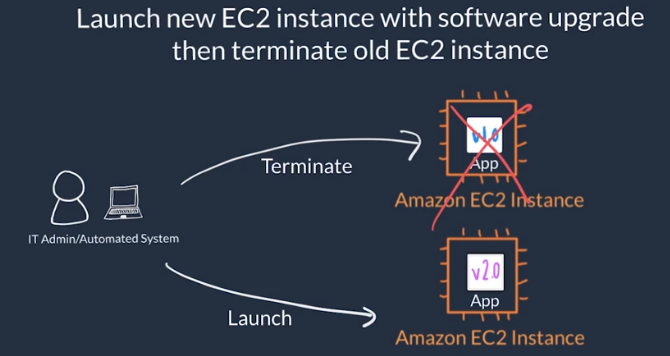


# Amazon EC2 Instance Lifecycle

Building an elastic app means that more or less resources to power your app are provisioned dynamically.

1. Launch an EC2 instance from an AMI.
2. Once, an EC2 instance is launched, it enters a “**pending**” state which means our VM is booting up and billing has not started yet.
3. Once the EC2 instance is ready to be used, then it enters the “**running**” state. Here, you will be charged for the EC2 instance at this stage and also, have below options;
   1. Option to “**reboot**” the EC2 instance similar to rebooting your laptop.
   2. Option to “**stop**” the EC2 instance which will enter the stopping phase and finally reaching to “stopped” phase. This is similar to powering down your laptop.
   3. Option to “**stop-hibernate**” which is similar to our laptop hibernate option.
   4. Option to “**terminate**” the EC2 instance which will enter the “shutting-down” phase and finally to the “terminated” phase. This is similar to throwing away your laptop in the ocean and not found forever.





Amazon EC2 Instance Types:

Amazon EC2 instances are a combination of virtual processors (vCPUs), memory, network, and, in some cases, instance storage and graphics processing units (GPUs).



For example, the instance type **c5.large** can be broken down as follows:

* the first letter, **c**, stands for compute-optimized instance family.
* **c5** determines the instance family and generation number. Here, the instance belongs to the fifth generation of instances in an instance family that’s optimized for generic computation.
* **large** determines the amount of instance capacity.

Amazon EC2 Instance Families:

|  |  |  |
| --- | --- | --- |
| **INSTANCE FAMILY** | **DESCRIPTION** | **USE CASES** |
| **General Purpose** | Provides a balance of compute, memory, and networking resources. | web servers, containerized microservices, caching fleets, distributed data stores, and development environments. |
| **Compute Optimized** | Ideal for compute-bound applications that benefit from high-performance processors. | High-perf web servers, scientific modeling, batch processing, distributed analytics, high-perf computing (HPC), machine/deep learning, ad serving, highly scalable multiplayer gaming. |
| **Memory Optimized** | Designed to deliver fast performance for workloads that process large datasets in memory. | high-perf DBs, distributed web-scale in-memory caches, mid-size in-memory databases, real-time big-data analytics, |
| **Accelerated Computing** | Use hardware accelerators or co-processors to perform functions such as floating-point number calculations, graphics processing, or data pattern matching. | 3D visualizations, graphics-intensive remote workstations, 3D rendering, application streaming, video encoding. |
| **Storage Optimized** | workloads that require high, sequential read and write access to large datasets on local storage. | NoSQL DBs, such as Cassandra, MongoDB, and Redis, in-memory databases, scale-out transactional DBs, DWH, Elasticsearch, and analytics |

Pricing:

AWS offers three main purchasing options for EC2 instances – On-Demand, Reserved, and Spot Instances.

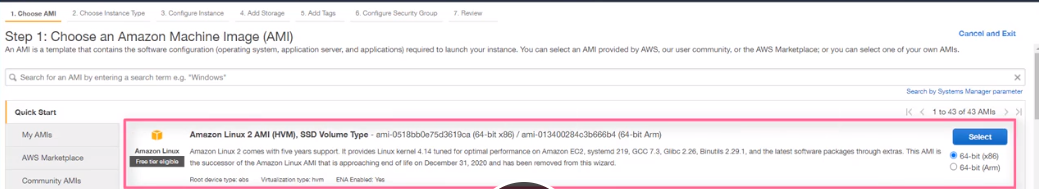


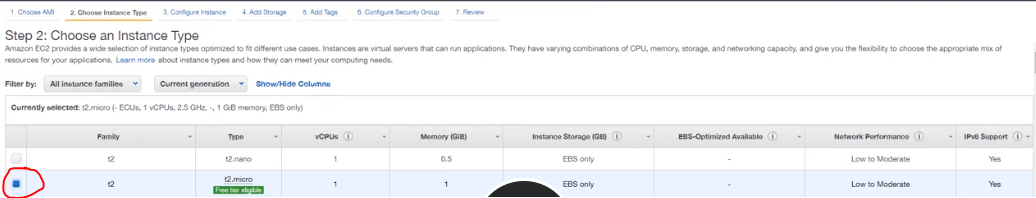
# Demonstration: Launch the Employee Directory Application on Amazon EC2

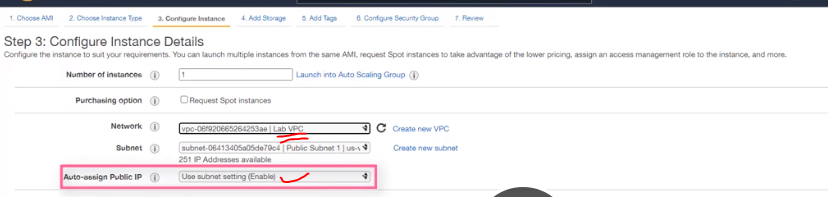
1. **Launch an Amazon EC2 instance.**

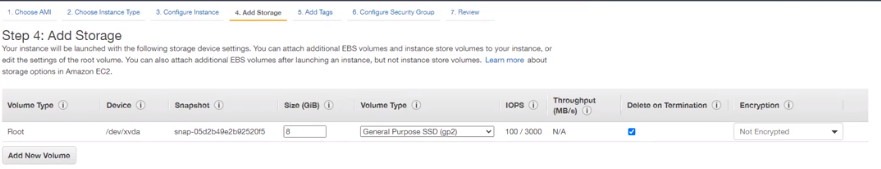
~ Login using IAM user id, go to EC2 dashboard and then launch instance.

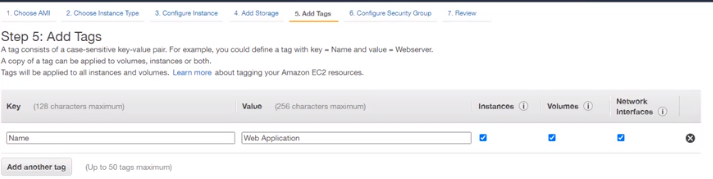


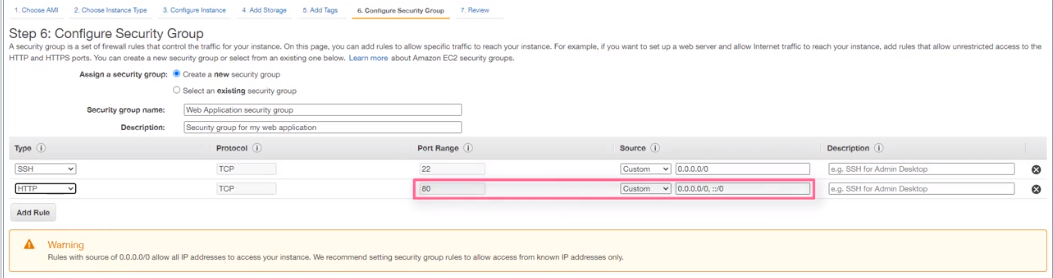


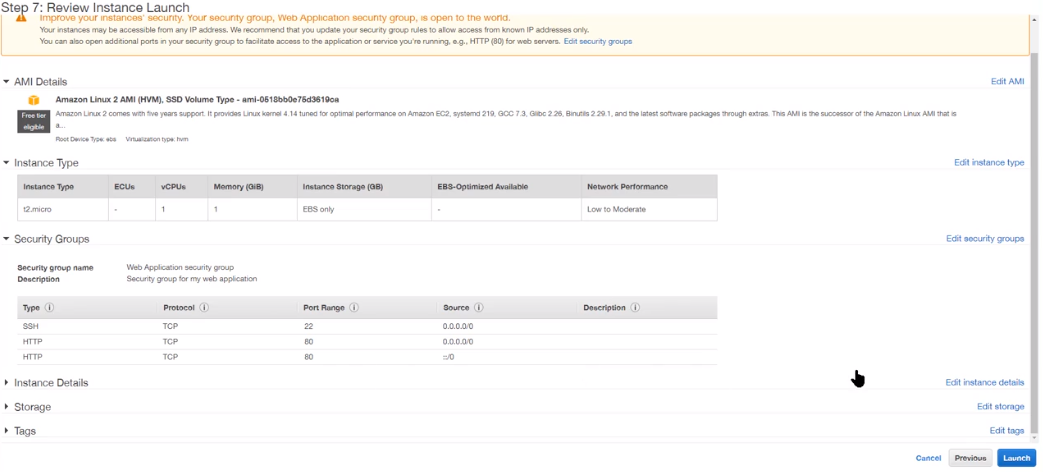


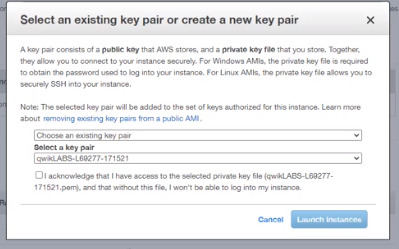


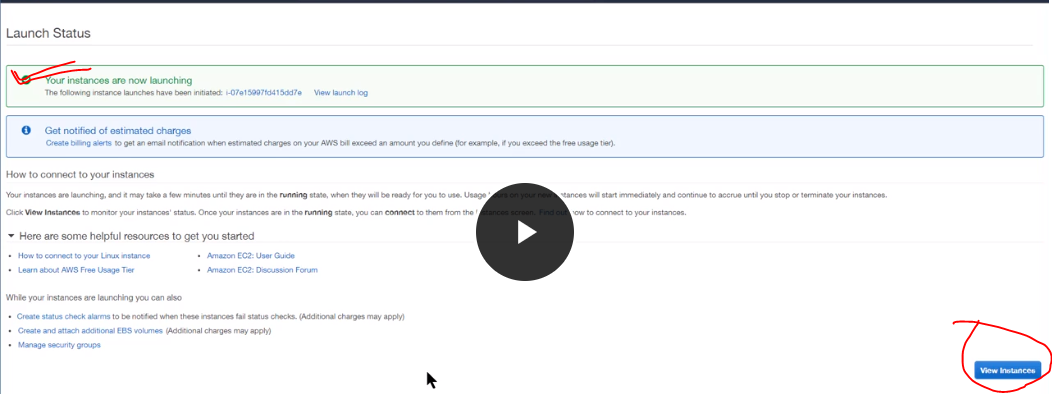


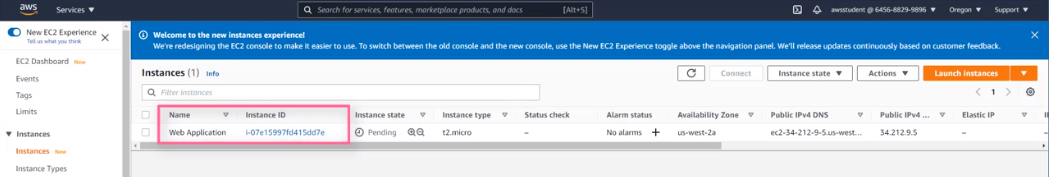


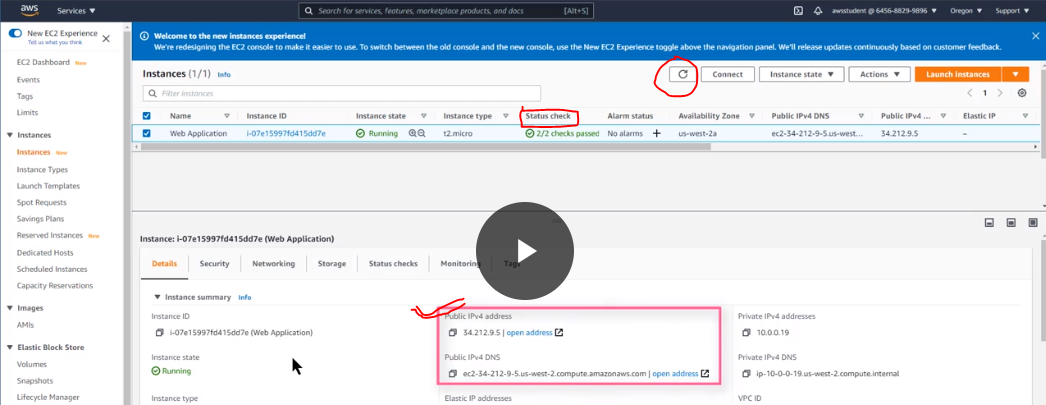




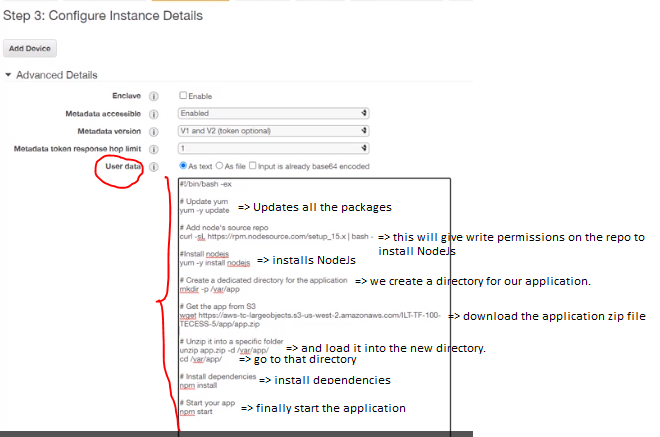






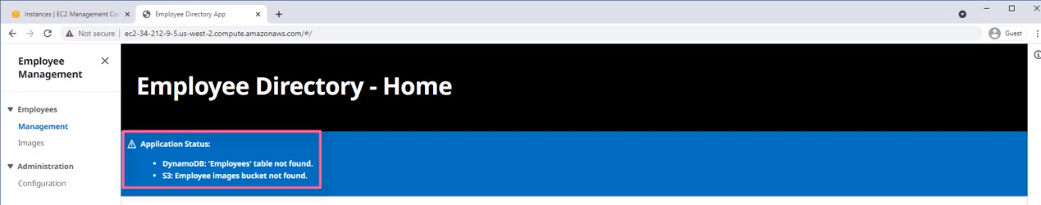


1. **Adding instructions to user data.**

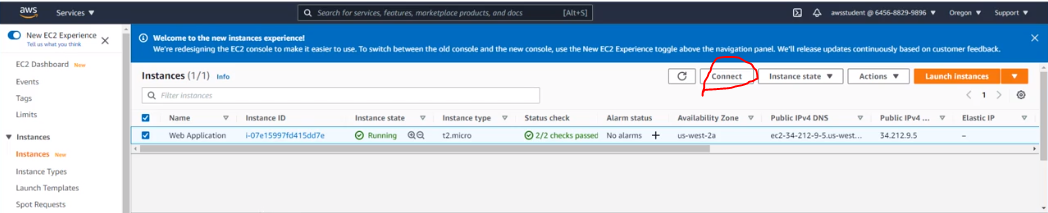


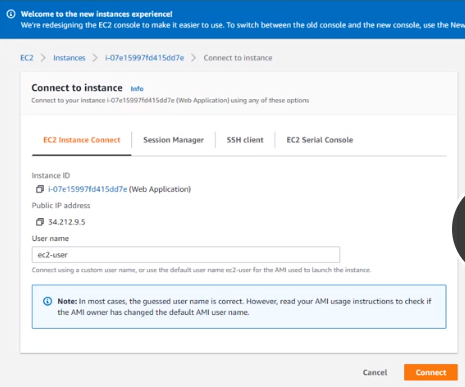
1. **Test web application.**

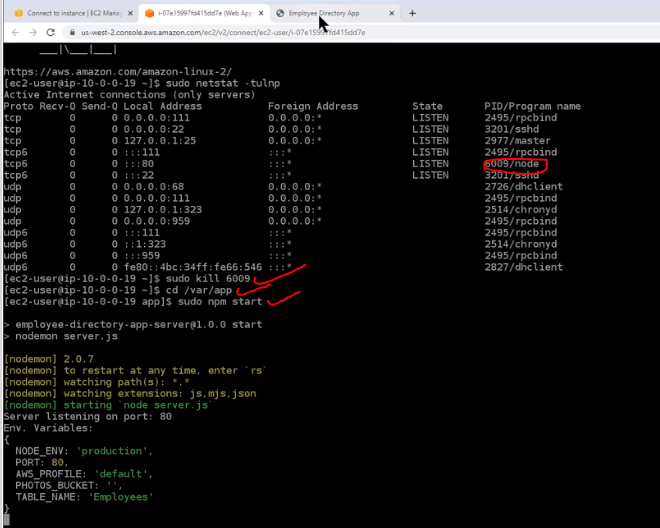
Copy the Public IPv4 DNS and use it with https:// as prefix in the browser.

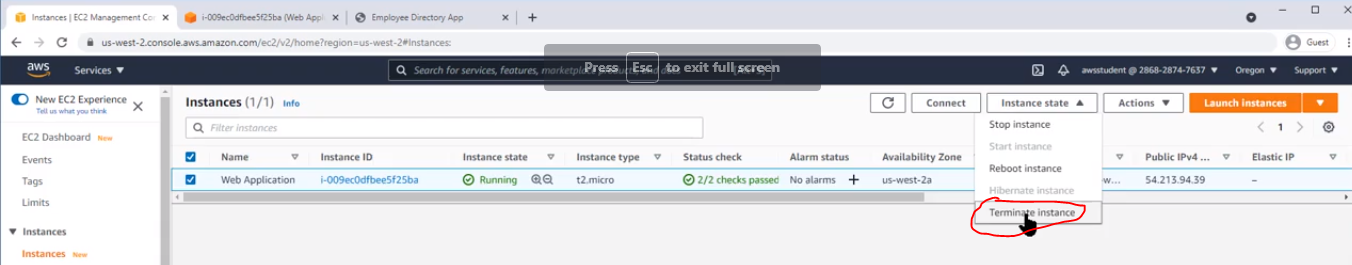


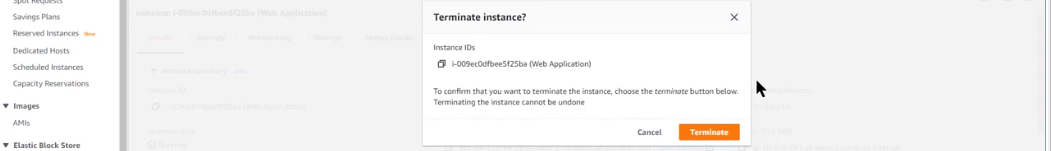
1. **Connect to EC2 instance console.**











# Container Services

The three main categories of compute are virtual machines (VMs), containers, and serverless.

Containers can host a variety of different workloads, including web applications, lift and shift migrations, distributed applications, and streamlining of development, test, and production environments.

**A container is a standardized unit that packages your code and its dependencies.**

**This package is designed to run reliably on any platform, because the container creates its own independent environment.**

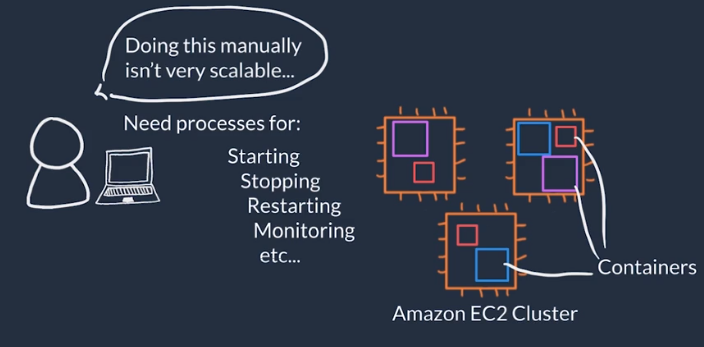
With containers, workloads can be carried from one place to another, such as from development to production or from on premises to the cloud.

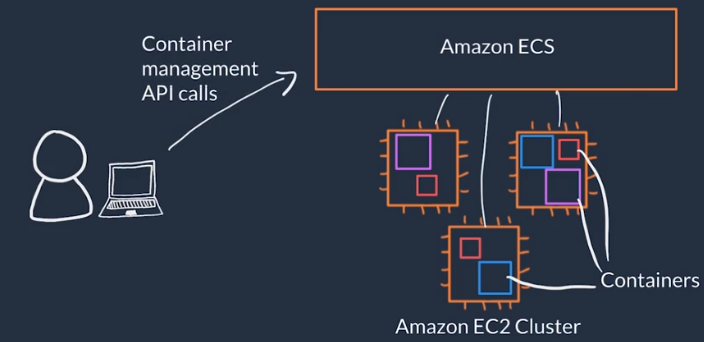
AWS offers two Container Orchestration services as:

* Amazon **Elastic Container Service** (ECS)
* Amazon **Elastic Kubernetes Service** (EKS)

Container orchestration is to monitor a cluster of Amazon EC2 instances (*multiple instances*). Here, the Amazon Elastic Container Service (ECS) does the job hassle-free.

* We can automate the scaling of the Amazon EC2 instances and Amazon ECS container services.
* ECS and EKS have service-specific scaling mechanisms for both container and cluster scaling.
* Containers have shorter boot-up time when compared to VMs.
* EKS or ECS can be hosted on top of our Amazon EC2 clusters as the hosting options. However, you can have an alternative to host and manage the OS inside your EC2 instances as compute platform known as “AWS Fargate”.
* **AWS Fargate** is a Serverless compute platform for ECS or EKS.

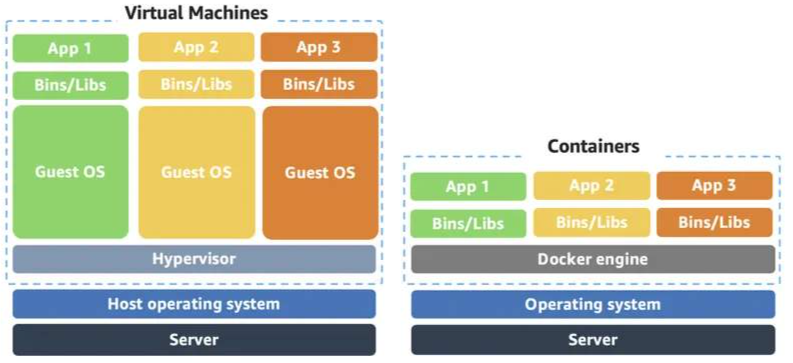




**Docker** is a popular container runtime that simplifies the management of the entire operating system stack needed for container isolation, including networking and storage.

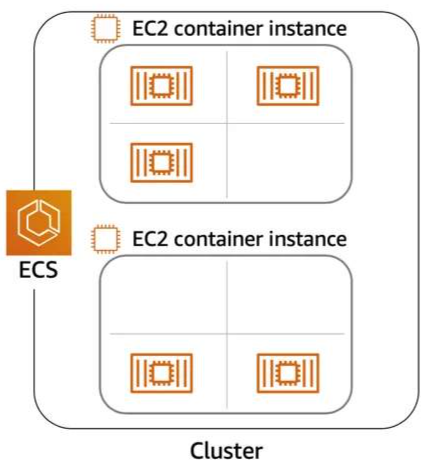
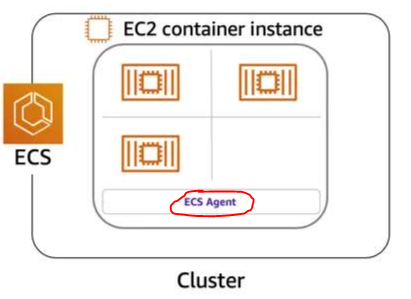
Docker helps customers create, package, deploy, and run containers.

Difference between containers and virtual machines (VMs)



**Manage containers with Amazon Elastic Container Service (Amazon ECS)**

* Amazon ECS is an end-to-end container orchestration service that helps you spin up new containers and manage them across a cluster of EC2 instances.
* To run and manage your containers, you need to install the Amazon ECS container agent on your EC2 instances.
* This agent is open source and responsible for communicating to the Amazon ECS service about cluster management details.
* An instance with the container agent installed is often called a container instance.

* To prepare your application to run on Amazon ECS, you create a **task definition**.
* The task definition is a text file, in JSON format, that describes one or more containers.
* A task definition is similar to a blueprint that describes the resources you need to run a container, such as CPU, memory, ports, images, storage, and networking information.
* For example, below is a sample task definition for our employee directory application which runs on a Nginx web server.



**Use Kubernetes with Amazon Elastic Kubernetes Service (Amazon EKS)**

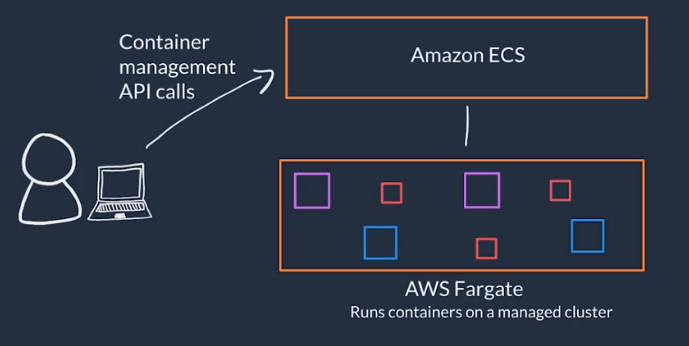
Kubernetes is a portable, extensible, open source platform for managing containerized workloads and services.

Amazon ECS vs Amazon EKS:

* An EC2 instance with the ECS agent installed and configured is called **a container instance**. In Amazon EKS, it is called a **worker node**.
* An ECS container is called a **task**. In Amazon EKS, it is called a **pod**.
* While **Amazon ECS runs on AWS native technology**, **Amazon EKS runs on top of Kubernetes**.

# Serverless

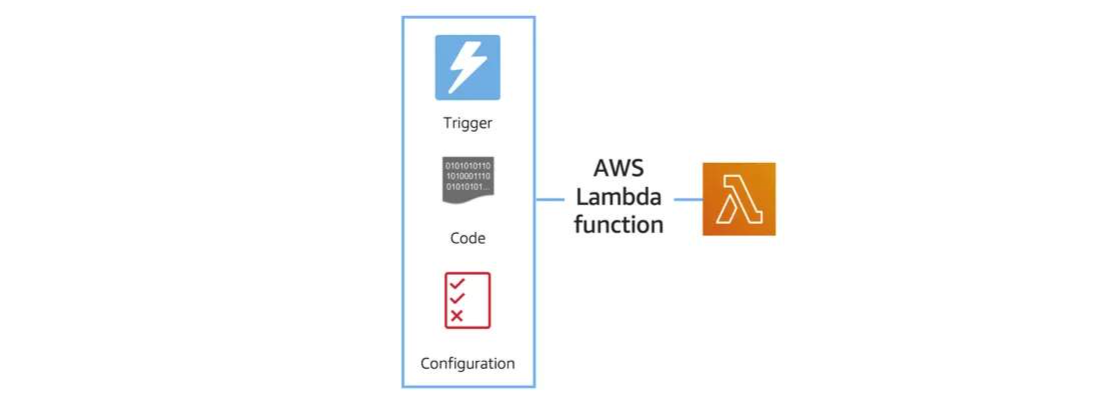
* Serverless means you cannot access or see the underlying infrastructure that is hosting the solution.
* Instead the management of the underlying environment from a provisioning, scaling, fault-tolerance and maintenance perspective is taken care for us.
* The “Shared Responsibility Model” is also followed in this case of serverless setup.
* With Amazon EC2, customer is expected to patch the OS and security system upon new release. Now, with serverless support system, the customer will not have access to OS & will not be responsible for patching but data encryption and access management is still owned by the customer.
* For example, **AWS Fargate is a serverless compute platform system that can be run on top of ECS or EKS**.
* Using AWS Fargate, the containers are run on a managed serverless compute platform; the scaling and fault-tolerance is built in; no need to worry about underlying OS or the environment.
* Instead, the customer needs to define their container and how the container needs to be ran & it scales on demand.
* AWS services are based on the below spectrum (Control----Convenience) as shown below.

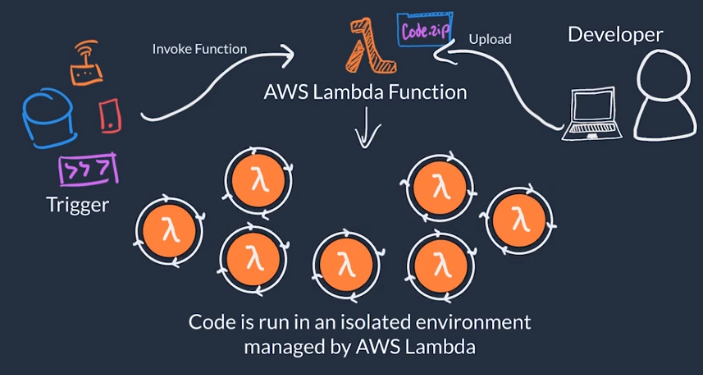


# AWS Lambda

* Serverless compute option available at AWS.
* Lambda allows you to package and upload your code to **Lambda Service** creating a **Lambda Function**.
* Lambda function does not run all of the time instead it runs in response to **Triggers**.
* Using trigger, we can configure on when to run the Lambda Function and from there the Lambda Service waits for the trigger.
* Trigger examples are: an http request, an upload of a file to AWS S3 (Storage Service), events originating from other AWS services and in-app activity from mobile devices

**A Lambda function has three primary components – trigger, code, and configuration.**

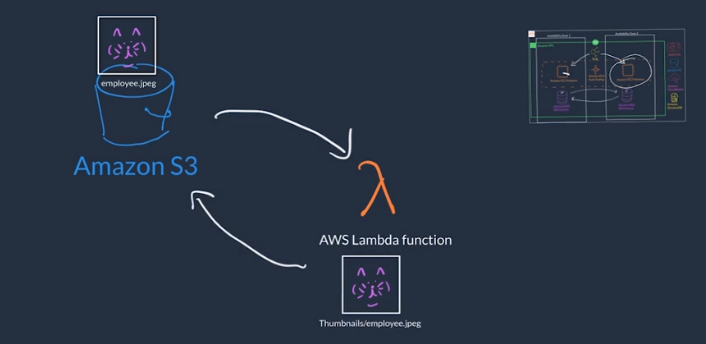


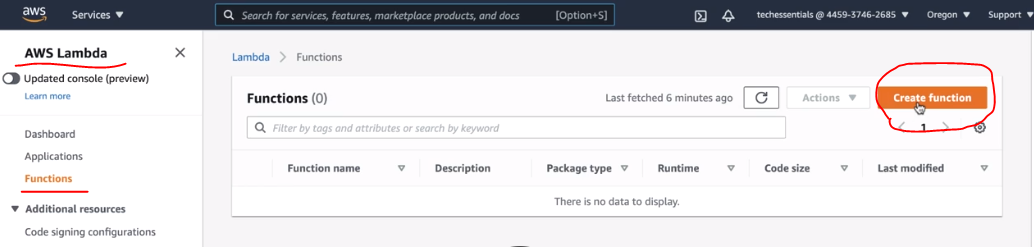


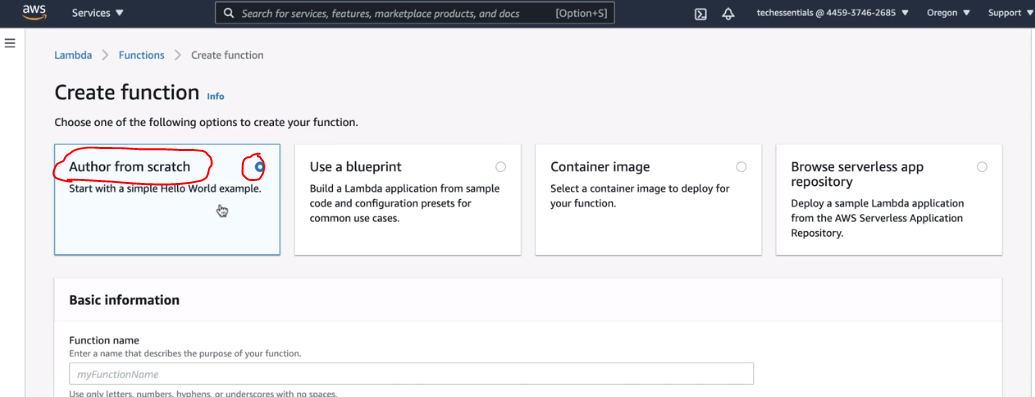
* When the trigger is detected then the code is executed in an isolated environment which AWS owns and manages but however, the customer can choose the code language (Python/C/R), CPU, Memory allocation, permissions, dependencies and other aspects related to function.
* If there are multiple incoming triggers, then AWS Lambda will scale your Lambda function to demand each in its own secure & isolated environment.
* **AWS Lambda is designed to run code which has a runtime of under 15 minutes only**. Hence, not suitable for long-running processes like deep learning or batch jobs.

**Task:**

Our objective is to build a Lambda Function to re-size a newly uploaded photo in Amazon S3 and store it in a separate location without impacting the original copy. This will help maintain the photo size uniformity across all the employee profile thumbnails.

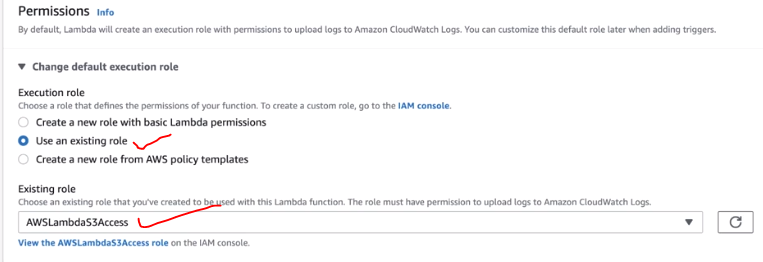




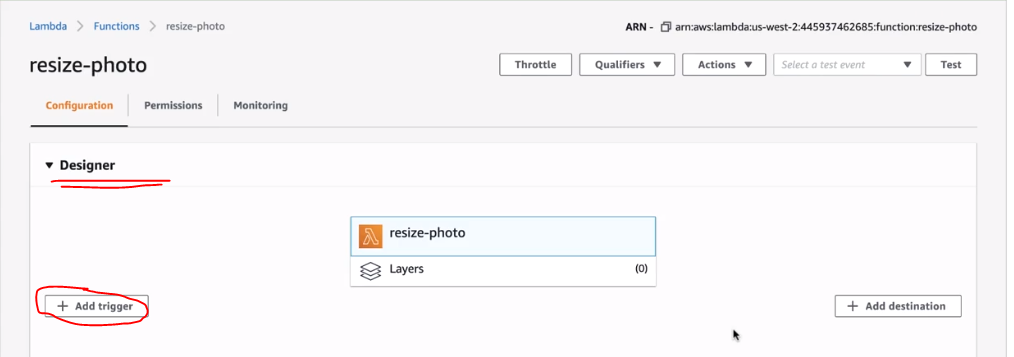


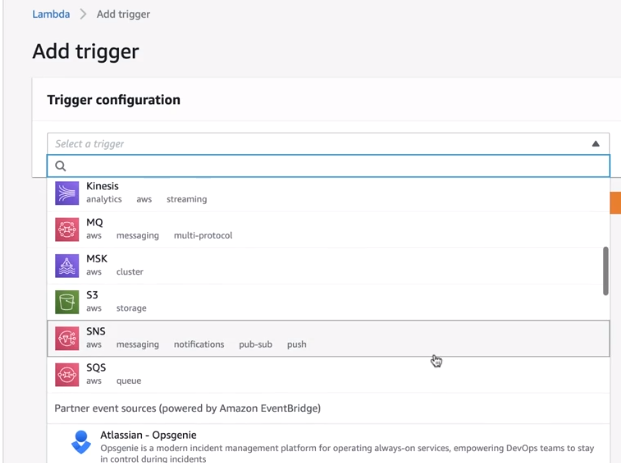


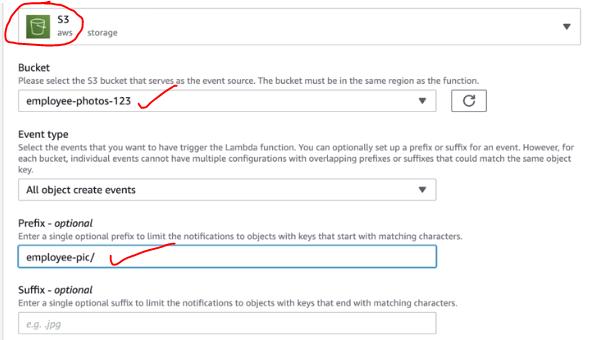
**Selecting an existing IAM user to give appropriate permissions to API calls when Lambda Function having python based code get executed.**



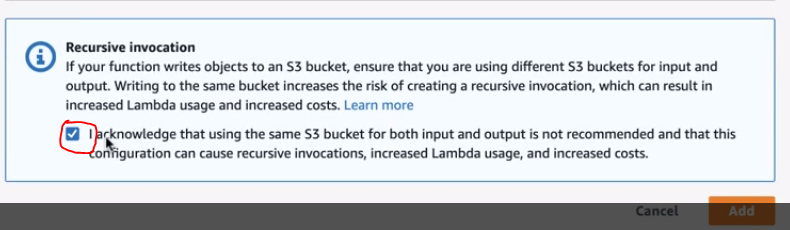
**Finally, the Lambda Function is created and this is the designer view with an option to add a Trigger.**



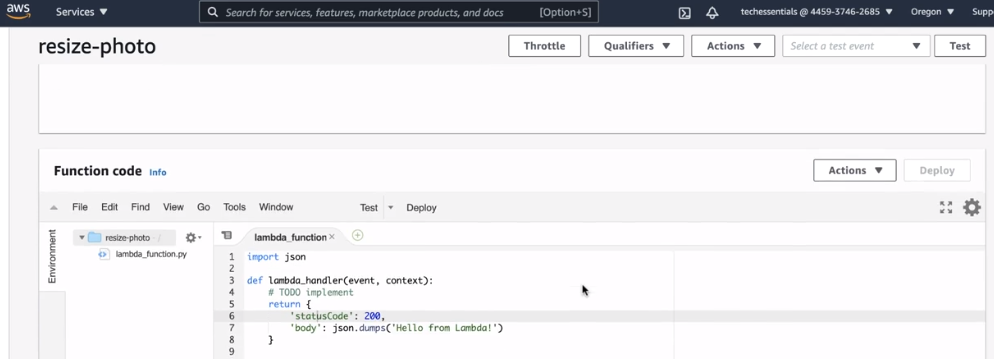


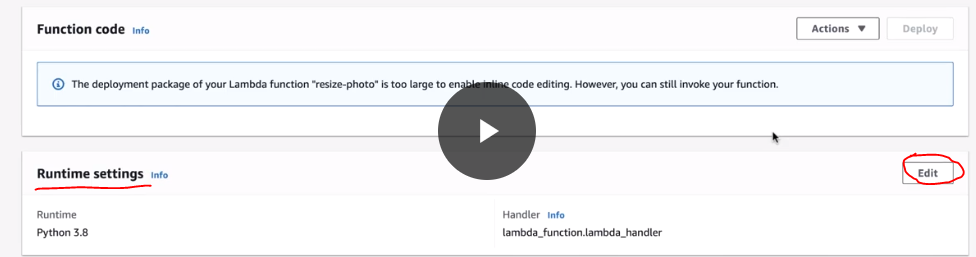


**This is necessary because in this case we are reading and writing in the same Amazon S3 bucket which triggers the Lambda Function in a cyclic process (infinite loop).**



**Add the source code**



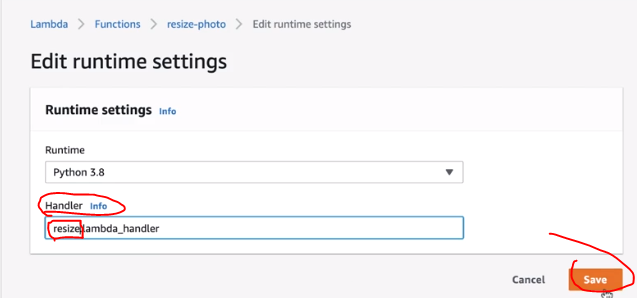


**Changing the handler from existing “lambda\_function.lambda\_handler” to “resize.lambda\_handler” so that it refers to our actual file.**

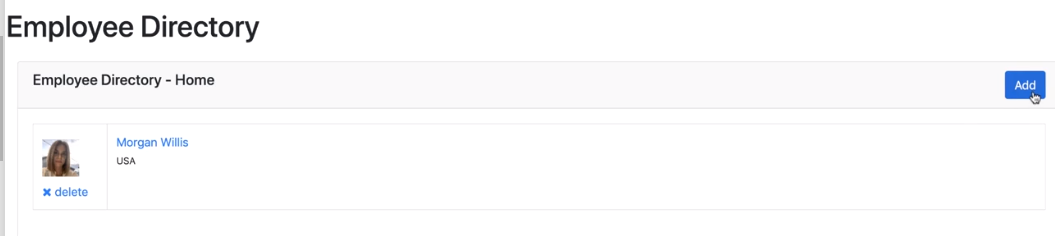
The AWS Lambda function handler is the method in your function code that processes events. When your function is invoked, Lambda runs the handler method.

You can use the following general syntax when creating a function handler in Python.

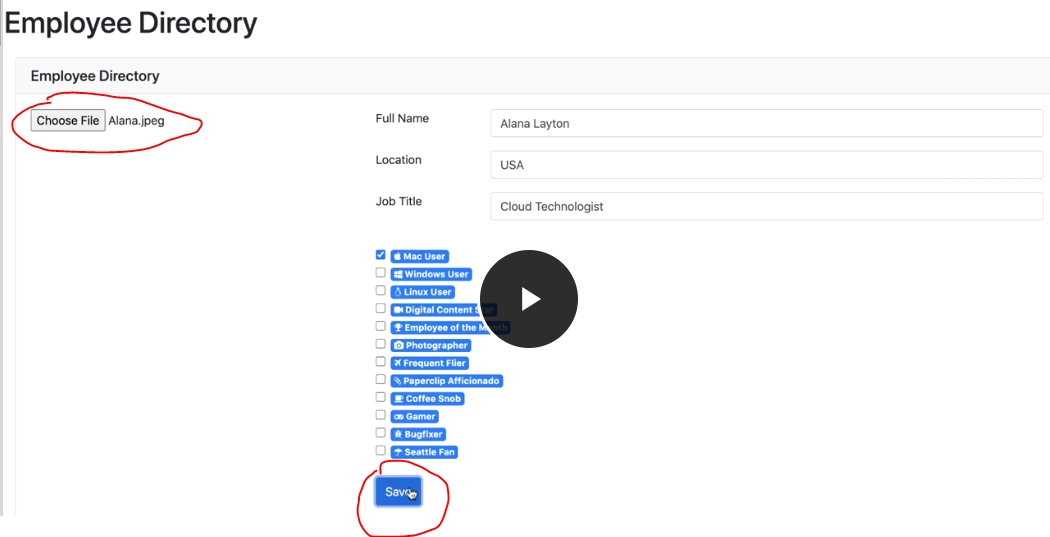
**def handler\_name(event, context):**  
**...**  
**return some\_value**

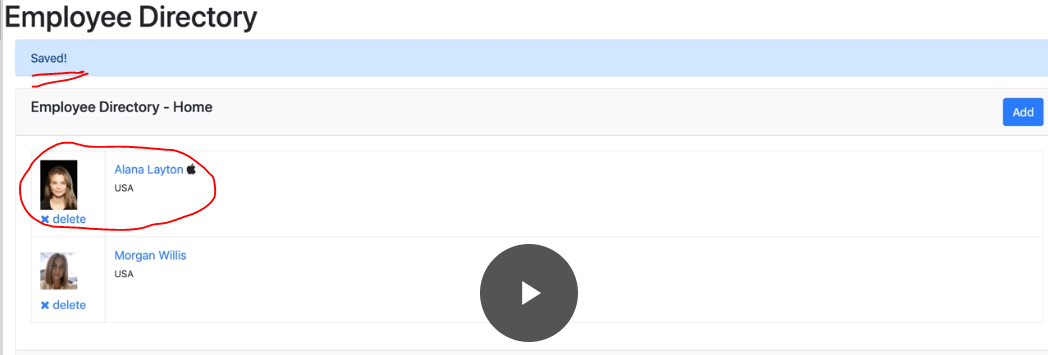


**Now, testing this through our employee directory application:**



**Creating a new user and uploading a photo as highlighted.**





**Now, we want to see whether it worked or not in CloudWatch logs.**



**Select the latest log stream which is the latest collection of log files:**

