Zero to Hero Beginner Udemy Course

- AWS Introduction
- Lab: Intro to Storage Services
- Lab: Intro to Database Services
- Lab: intro to Compute and Networking Services
- Lab: Intro to Management Services
- Lab: Intro to Application Services
- Lab: Intro to Analytics and Machine Learning
- Lab: Intro to Security, Identity, and Compliance
- Lab: Intro to Developer, Media, Mobile, Migration, Business, IoT
- Elastic Beanstalk
- AWS CLI

AWS Introduction

AWS Region and Availability zone

An AWS region is a geographical location with a collection of availability zones mapped to a physical data center in that region. Each region operate independent of one of another, they have each of their own power, water supply, etc. This means that if one region fails it won't impact the other.

Regions are available world wide with more regions coming soon. There are regions in US of course, Asia, Europe, and Australia.

Within each region it contains at least two availability zones which is the actual physical data center, each zone doesn't necessarily mean it is backed by exactly one physical data center, each zone might be backed by one or more physical data center. Each zone is a region also have redundant and separate power, networking to prevent the likelihood of two zones failing simultaneously. **Hence, data for each region are replicated within each availability zone.**

Basically, AWS divide up the service places into regions, which are collection of availability zones. Each region contain multiple availability zones to have redundancy. Regions exist so that they operate independent of each different regions.

AWS resources

aws.amazon.com/documentation: Summary information about each of the product in AWS

aws.amazon.com/whitepapers: Discuss technical issues and use cases for AWS

aws.amazon.com/products: Main product of AWS

aws.amazon.com/new: Latest things occurring in AWS

Lab: Intro to Storage Services

Cloud computing models

Infrastructure as a service

Offer services that are present in on-premise data center. Servers, storage, and networking hardware, so like VPN are all available as a service in AWS.

Ex: EC2, VPC, EBS.

Platform as a service

AWS helps you manage the underlying infrastructure, hardware and operating systems for you and provide you the application up and running without you having to do anything.

So you can set up a RDS (relational database) without having to worry about setting up a Linux server, and downloading MySQL database. You can just leave the setting up of the database to cloud and then use it after it is up and running

Ex: RDS, EMR, ElasticSearch

Software as a service

Completed product that is run and managed by the service provider. So the cloud provide you the complete the application for you to use, you don't need to worry about how it is being deployed what kind of server it is set up with. You just leave the application managing to the cloud and use it.

Serverless computing

Refer to as function as a service. It allows you to build and run applications and services without thinking about managing the servers. You just have to worry about the actual coding of the business logic that you are implementing without having to worry about provisioning the servers underneath and how you are going to host it.

AWS storage services

1. Simple storage service (S3): Designed to store object files, files that you don't need to worry about a traditional tree hierarchy of path to place the file. It creates a unique identifier for each object file that you store into them for retrieval.

   It is serverless service, you don't need to worry about how it is implemented underneath, what operating system is used
2. Glacier: Cheapest storage for long-term storage. Only used for content that is to be archived.
3. Elastic block store (EBS): Highly available, low latency block storage, designed to be attached to EC2 instances. **Similar to attaching a hard drive to your computer.** Cannot be shared between multiple EC2 instances.
4. Elastic file system (EFS): Network attached storage, allow multiple EC2 instances to mount onto this filesystem.
5. Storage gateway: Enable hybrid storage between on-premise environment and AWS cloud. Cache frequently used data on-premises and storing less frequently used data to AWS.
6. Snowball: Used to migrate large amount of data from on-premise data center to AWS.

AWS VPC

Virtual private cloud is the basis of how to set up your AWS instances. Similar to a traditional network, all of the instances that are launched in AWS will be under a VPC. Public internet will not be allowed to access to those resources without you explicitly permitting it via some kind of routing.

Think of VPC just as a traditional data center network, except it is on the cloud so you don't have to worry about setting it up yourself, it is done for you already.

VPC endpoint is used to allow traffics in and out of the specified VPC, otherwise, say if you launched a S3 bucket in AWS, and you want your EC2 instances to deposit some object files into it. Your EC2 instance is under the VPC while S3 buckets are not, then the EC2 instance have no way of communicating with S3. To allow it, you create a VPC endpoint to allow traffics from VPC out to the AWS S3 instance.

Hybrid storage example

Say you have on-premise site storage and you want to migrate the petabyte of data from your on-premise data center up to AWS cloud. How would you do that? Uploading those files directly using Internet is not going to be viable because you have petabyte of data! That's where snowball can come into play. You would get a device from AWS and you would upload all of your on-premise data into the device and ship it back to AWS. They will then upload all of that data into the AWS cloud say into a S3 bucket directly.

Then in order to keep the keep between on-premise data center and the data in S3 in sync, AWS storage gateway can be used to orchestrate a synchronization mechanism. Data are replicated by storage gateway into S3 buckets, you can have that as a disaster backup.

Lab: Intro to Database Services

AWS relational database services (RDS)

A fully managed database services, makes it easy for you to launch a database servers.

You get to choose the different type of database engine from MySQL, MariaDB, PostgresSQL, Oracle, Microsoft SQL Server.

AWS Aurora

AWS's implementation of MySQL database.

DynamoDB

Amazon's NoSQL database. Provde high speed, extremely low latency performance

Redshift

Fast fully managed petabyte scale data storage that is based on postgre SQL. It can store up to petabyte of data.

ElasticCache

In-memory data store, retrieve information from fast RAM cache instead of relying on slower disk based database. Like Redis.

Database Migration services

Orchestrate migration of data from one database engine type to another database type.

For example: migrate Oracle to AWS Aurora.

Neptune

Graph database. High performance graph database engine optimized for storing billions of relationships.

Database Example

ElasticCache node can be used together with a database that is being queried frequently. For example, you spin up an Aurora database instance but it is being overwhelmed by more request than it can handle. You can either choose to scale up (which will cost more money) or you can put an ElasticCache node in front of it. Any data that is being handled will be put into the ElasticCache so that further of the same request will be answered by the ElasticCache from a fast in-memory query, rather than having to go to the database.

And any data that is not in ElasticCache that request will simplify be forwarded to the Aurora database. This will offload frequently accessed data in Aurora to ElasticCache that can be answered easily since it exists in memory, and those infrequently accessed ones will be forwarded to Aurora and then stored into cache.

Lab: intro to Compute and Networking Services

AWS compute services

EC2

Allow you to launch on-demand virtual machines.

EC2 autoscaling

Allow you to dynamically scale EC2 capacity up or down according to conditions you defined. Scale up by launching more EC2 instances, and scale down by terminating EC2 instances that you don't need.

The condition you can pre-define such as CPU usage which is typically a good metric to use for autoscaling.

Amazon Lightsail

Easy way to launch virtual server to run an application. It will configure DNS management and storage for you

Elastic container service

Container management service for docker containers. They will be run in a EC2 instances.

AWS Lambda

Serverless service, allow you to run code without worrying about provisioning and managing the servers at all. You can just focus completely on writing the business logic and the code, just have to upload it and you're good. It runs on pay as you call rather than by the hourly. If the function isn't invoked at all then you won't be billed for it.

Webserver example

Vertical scaling: Say for example your current EC2 web server instance is being overwhelmed by requests. You will need to scale somehow. The old way 10/20 years ago to scale is by taking down

your EC2 instance and then deploy a new one that is bigger and more powerful.

But this takes time, and your application is not running. What if the demand is only a peak and goes back down after. So vertical scaling doesn't handle that really well.

Horizontal scaling: To solve the problem that vertical scaling cannot handle we use horizontal scaling. We add more instance of the same EC2 instance to handle those demand, and as those demand decreases we terminate those instances.

Our application will still be up. However, the problem with horizontal scaling is that customers aren't going to know which EC2 instance is up and down when demand goes up and down. We will need a load balancer, and in AWS, Elastic Load Balancer will keep track of all the EC2 instances and distribute the incoming request to the available EC2 instances.

Auto scaling services: It will launch and terminate EC2 instances for you automatically according to the demand. It can also do health check, replacing unhealthy EC2 instances for you automatically.

Networking & content delivery

CloudFront

Securely delivery frequently accessed content, similar to a cache at a high transfer speed for your end-users. It also provides protection against DDos attack. It can reach many edge locations that are far from data centers.

Virtual private cloud

Rather than a traditional network where the network are done via hardware switchers and routers, there is another way of creating a computer network, which is via software network, or virtual network.

VPC is the AWS implementation of the virtual network, allows you to allocate a private section on the AWS cloud where you can launch your AWS instances into and interact with each other or when allowed interact with the public internet. Without any security group, **no traffics are allowed in to a VPC, but all traffics are allowed out from a VPC.**

Think of VPC as your own personal space that no one can enter without you explicitly allowing it.

Direct connect

A dedicated high speed network connection to AWS from on-premise data centers.

Elastic load balancing

Automatically distribute incoming traffics for your application across multiple EC2 instances and also in multiple availability zones. So if one availability zone goes down, it will automatically route to other availability zone, and because data are replicated and redundant in availability zone, it is okay if the traffic is routed to other availability zone.

Let you achieve high availability and fault tolerance by distributing traffic evenly among those instances.

Route 53

Highly scalable and available DNS server. Direct your domain name to say a backend web server. A DNS server basically.

API gateway

Easy for developer to create and deploy secure API at any scale. Handle tasks of accepting and processing up to hundreds and thousands of concurrent API calls.

Serverless service, no need to worry about provisioning the servers underneath.

CDN Example

You can leave static files such as large videos and images to CloudFront CDN to handle rather than passing it to a EC2 instance to handle. Let EC2 instances handle those dynamic content request that aren't static and require a server to compute and handle that request for the user.

You would set CloudFront as the entry point, let it serve static content and forward request for dynamic content to the load balancer which will distribute the request evenly among EC2 instances.

CloudFront CDN will generate a complicated domain name that is associated to that CloudFront instances, you are not going to give that to the end user right? They are not going to be remember it, and to solve that we can use Route 53 to map the domain name of your application to the CDN.

Lab: Intro to Management Services

Management tools

CloudFormation

Use a text file to define and deploy your infrastructure on AWS.

AWS Service Catalog

Allow enterprises to look at what type of resources can be deployed to be governance compliant. What is allowed to be deployed on AWS cloud

CloudWatch

Monitoring service for cloud resources and applicaiton that are deployed on AWS Cloud. Used for triggering scaling operations, or provide insights into your deployed resources.

For free tier: You're fine as long as you don't define too much alarms, you won't have to pay anything at all which is very nice. Always free, no expiration!

AWS systems managers

Let you view operational data from multiple AWS services and automate task across AWS services.

Help detect and resolve operational problems.

Trusted advisor

Analysis your AWS resources that you have deployed and advise you on how to achieve higher security, best performance, and optimized cost for those resources.

CloudTrail

Monitors and logs AWS account activity. Management console, SDK, CLI are all tracked.

AWS Config

Allow you to assess, audit, and evaluate the configuration of your AWS resources. Simplifies security analysis and change management control, and operational troubleshooting

OpsWorks

Automate deployment of AWS resources

Lab: Intro to Application Services

Application integration

Step functions

Make it easy to orchestrate bunch of microservice in a particular order. You define your application visually as series of step and then you can deploy it. Say you only want microservice B to run after microservice A is run, you can do that.

Simple WorkFlow Service

Similar to step function, coordinate multiple component for an application. However, use step function is better and easier rather than SWF

Simple notification service

Hub-sub service. Create topic and user can subscribe to that topic, when you publish a message to the topic, users who are subscribed to the topic will be notified with the message you have published.

Also be used for push notification for mobile devices.

Simple Queue Service

Fully managed message queueing service, make it easy to decouple application from demand. You can store demand (in form of requests) inside Simple Queue Service, and then forward those request to the servers.

There are two types of queues. Standard queues and FIFO queues. Standard queues is best effort delivery, meaning messages inside queue might be lost, but it will be at-least once delivery. The ordering aren't guaranteed because of message lost.

FIFO queue on the other hand guarantees that each message will only be processed once, and the ordering preserves. First in, then you're first out.

Process decoupling example

So let's say we got bunch of servers that is processing say requests. And it is backed by a auto scaling service. Without a queue, the request have chances of overwhelming the instances, because auto scaling might not be able to keep up with the sudden spike in demand since it will take 5/10 minutes to spin up an instance. While the surge in demand could period of less than a minute.

So request will be dropped if we don't do something. How do we solve this? We can store those requests into a queue, so that if the servers are overwhelmed the requests will not be dropped. Instead, a CloudWatch metric is set up to watch the queue, if the request are increasing then it will trigger an auto scaling group to spin up more instances, and then the queue can finally forward those request to the free instances. Now when the demand decreases, the instances that are created can be terminated.

Customer engagemenet serives

Amazon connect

Drag-and-drop graphical user interface. Allow you to define customer interaction without any coding at all.

Amazon pinpoint

Sent notification and email for targeted marketing campaign.

Simple email service (SES)

Cloud based bulk email sending service. Mass email sending.

Lab: Intro to Analytics and Machine Learning

Analytics service

Amazon EMR

Hadoop framework as service. Data can be analyzed.

Athena

Analyzed data stored in Amazon S3 bucket using standard SQL statement

Elasticsearch Service

Allow high speed query

Kinesis

Collect and process and analyze real-time streaming data

QuickSight

Business intelligence reporting tool. Allow you to make graphs and analyze data.

Machine learning services

DeepLens

Create advanced vision applications

SageMaker

AWS flagship machine learning product, build and train your own machine learning model and deploy them, use it as backend of your application

Amazon Rekognition

Recognition of video and pictures

Amazon Lex

Build first-line support chatting model for your customers

Amazon polly

Natural sounding text-to-speech service

Comprehend

Deep learning to analyze text for insights and relationships.

Translate

Accurately translate text to another language

Transcribe

Analyze audio file stored in S3 and return transcribed text

Lab: Intro to Security, Identity, and Compliance

Security, identity, and compliance

AWS Artifact

Online portal that give access to AWS security and compliance documentation. You can read documentation about security and how to make your application government compliance.

AWS Certificate Manager

Issues SSL certificates for HTTPS, it is integrated into Route 53. It is completely free.

Amazon cloud directory

Cloud-based directory service, hierarchy of data in multiple dimensions.

Amazon directory service

Fully managed Microsoft active directory service in AWS cloud. Used for controlling users, admins, groups and manage their access to resources.

CloudHSM

Dedicated hardware security module in AWS. Achieve corporate and government compliance, rather than using your own HSM.

Cognito

Sign-in and sign up capability for your applications. Can integrate external OAuth like Google and Facebook provider as well.

IAM (Identity and access management)

Allow you to manage user access to your AWS services and resources in your account.

Users and groups have their own permission whether they are allowed or not to access the resources you specified.

AWS Organizations

Policy based management for multiple AWS accounts.

AWS Inspector

Automated seucrity assessment service. Help identify vulnerability or areas of improvement in your AWS account

Key managmenet service

Create and control encryption keys. Also use hardware security module for protecting your keys

Incorporated into S3, redshift and EBS

AWS Shield

Help protect against DDos.

Automatically into all AWS accounts

Web Application Firewall

Provide additional protection in front of your web applications, such as SQL inejction attacks.

Lab about IAM

Up until now the account we are using to play with the AWS services are our email and password, that is the root user. And most of the time you don't want to login to the root account since it has access to everything, deleting, creating, any instances. Finances, credit card informations, can lock other people out.

To protect your root user: Have a long and complicated password and use MFA

IAM user is better login as when your are interacting with the console because it will just have enough permission to do what it needs to do, those permission are granted by the root user.

The user you create for IAM can have both management console access and programmatic access (meaning they get their respective access key ID and secret access key in order to use them to use the CLI and SDK)

After you create the user you can then attach permission policy to specify what they can and cannot do with our resources.

Lab: Intro to Developer, Media, Mobile, Migration, Business, IoT

Developers tools

Cloud9

IDE in AWS, let you code and then develop services directly from the IDE

CodeStar

Can manage entire CI/CD pipeline for your application. Have project amangement dashboard, JIRA.

X-Ray

Make it easy to analyze and debug application, give you insight for your application performance.

CodeCommit

Git repository but run in AWS cloud

CodePipeline

CI/CD pipeline. Help you build, test, and deploy whenever you commit to your repository

CodeBuild

Compiles your source code, runs test, and produce software package that is ready for deploymnet

CodeDeploy

Automates software deployments to different AWS services, EC2, Lambda

Media services

Elemental Mediaconvert

File based video transcoding service, can help you convert video formats

MediaPackage

Prepare video content for delivery over the internet, also have piracy protection.

MediaTailor

Insert targeted ads into your video

MediaLive

Broadcast live streams for both TV and internet

MediaStore

Storage service for storing media

Kinesis Video Streams

Stream connected device video into AWS cloud for machine learning and other processing applications

Mobile services

AWS Mobile Hub

Configure AWS application for mobile services

AWS Device Farm

App testing service for Andriod, iOS, and web application. Test app against large collection of physical devices.

AWS AppSync

GraphQL backend for mobile applications.

Migration

AWS Appliation discovery Service

Help plan migration over to AWS by collecting data from on-premise data center.

AWS Database Migration Service

Help migrate database to another data type

AWS Server Migration Service

Automate migration work-load

AWS Snowball

Portal petabyte scale data storage device. Used for migrate on-premise database to AWS cloud

Other

WorkDocs

Basically googledocs, but can accept lots of different doc types

WorkMail

Business email and calendar service

Chime

Online meeting service, basically Zoom.

WorkSpaces

Fully managed secure desktop as a service. Give you a virtual desktop that you can use

AppStream 2.0

Allow you to stream desktop applications to html5 compatible web browser.

Internet of things

AWS IoT

Let embedded devices, microcontrollers and Raspberry pi to interact with AWS

FreeRTOS

OS for microcontrollers, allow small low-cost low-power to connect to AWS

Greengrass

Let you run local AWS lambda functions and messaging data, and machine learning applications

Gamelift

Deploy scale anad manage dedicated game servers

Lumberyard

Game development environment

Elastic Beanstalk

EB

Let you deploy your application without you having to worry about allocating the servers. You just need to worry about writing the code.

It also automatically handle capacity provisioning, load balancing, scaling, and application health monitoring.

To update your application you can upload the files via console or CLI.

Applications can be Docker container, NodeJS, Java, PHP, Ruby, Python & Go. You can choose server to deploy via Apache or Nginx

Deployment strategy

- All at once: If you plan 20 EC2 instances in total, all at once strategy will just deploy those 20 instances. The down side is that your applications won't be responding to request while you are deploying to those 20 instances.
- Rolling: Deploy application to a batch at a time. Say a batch consist of 10 instances, that means with 20 instances, you will do 2 batches of deployment. Your application will still be responding which is good
- Immutable: Variation of all at once, but when you deploy a new application to those 20 instances, you will deploy another 20 instances, so 40 in total temporarily, and then terminate those old 20 instances. Your application will still be up without any downtime
- Blue-green deployment: Blue environment, green environment. Blue you can say is for development environment, and green is production environment, or the other way. When your development environment is ready for production you just toggle it to become production environment. The old production environment will become the new development environment. No downtime is involved.

Lab: Deploying a Highly Available and Fault Tolerant NodeJS Server

Head to Beanstalk, give your application a name. Pick a platform, what kind of application you are deploying, is it NodeJS, Python, Docker container, ... etc.

You can use a sample application or upload your code. Then you can configure if it is highly available or fault tolerant, before you deploy it.

Configure have load balancer and auto scaling if you configure it.

Deleting the beanstalk instance will delete all of the resources that are allocate with it.

AWS CLI

AWS rest API

This is used by AWS Management Console, AWS CLI, AWS SDK, and other AWS services. So in the backend, HTTP restful request is used for actually carrying out those resource management actions.

Documentation is available for those rest API call for lots of services. You will resort to using rest API to interact with the AWS resources if a programming language doesn't have a library that provides the wrapper for those rest API calls.

AWS API Security

API calls must be authenticated before you can use it. Authentication can be made with valid

1. Account username and password (For console access)
2. IAM user access key ID and secret (For CLI)
3. IAM temporary credentials (This is for SDKs). OAuth for authentication.

All API calls can be logged using CloudTrail for security.

AWS CLI

Send those API calls ot the AWS cloud.

AWS Cloud9 IDE

Integrated development environment running on a EC2.

Has CLI pre-installed. It has increased security for credentials because it isn't saved locally on your computer.

AWS configure

By default the default profile (user) that it uses for its credential in the CLI is named "default". Use `aws configure list` to see is AWS currently using, if it is empty then it is "default".

To change the credential that you are using export the environmental variable `export AWS_PROFILE=<profile>` and set it to the profile that you want to use the credentials for. Then from onward the CLI will be using that user's credential for the AWS access.

By default, all of the AWS credentials are stored under `~/.aws/credentials`. Non-sensitive information like configuration of the regions are under `~/.aws/configure`