**AWS Cloud Practitioner**

Silver lining in the Cloud

* How about **provisioning (renting) resources** when you want the and releasing them back when you do not need them?
* **On-demand resource provisioning**
* Also called **Elasticity**

Cloud – Advantages

* Trade **“capital expense”** for **“variable expense”**
* Benefit from massive **economies of scale**
* Stop **guessing** capacity
* Increase speed and **agility**
* Stop spending money running and maintaining data centers
* **“Go global”** in minutes

Amazon Web services

* Leading cloud service provider
* Provides most (200+) services
* Reliable, secure and cost-effective

Best Path to learn AWS

1. Amazon S3
2. EC2
3. Amazon EBS
4. ELB
5. ECS

* Cloud applications make use of multiple AWS services.
* There is **no single path** to learn these services independently.

Users

1. Root User
2. IAM User

Create IAM

1. Login aws console
2. Go to services and type IAM (manage user access and encryption keys)

Will create a group

1. Click on the groups
2. Give a group name
3. Select policy name which u want
4. Click next step
5. Click on create group

Create user

1. Click on user
2. Add user
3. Give username (you can give multiple user at same time)
4. Select AWS access type 🡪 AWS Management Console access
5. Select AWS access type 🡪 Programmatic access ( I am using both access )
6. Console password 🡪 custom password
7. Click next
8. Select group which you were created previously, or you can create now as well.
9. Click next
10. Click next to review
11. Click on create user
12. If you need to login your user, you will get a URL after creating a user u can find there.

Region’s data centers

For example:

1. Image that your application is deployed in a data center in London
2. What would be the challenges?

* Challenge 1: slow access for users from other parts of the world (high latency)
* Challenge 2: what if the data center crashes?

1. Tour application goes down (low availability)

Multiple data centers

* Let’s add in one more data center in London
* What would be the challenges?

Challenge 1: slow access for users from other parts of the world

Challenge 2 (solved): what if one data center crashes?

Your application is still available from the other data center

Challenge 3: what if entire region of London is unavailable?

Your application goes down

Multiple Regions

1. Let’s add a new region: Mumbai
2. What would be the challenges?

* Challenge 1(partly solved): slow access for users from other parts of the world

You can solve this by add deployments for your applications in other regions

* Challenge 2 (solved): what if one data center crashes?

Your application is still live from the other data center

* Challenge 3 (solved): what if entire region if London is unavailable?

Your application is still served from Mumbai

Regions

* Image setting up your own data centers in different regions around the world.

Would that be easy?

* (solution) AWS provides 20+ regions around the world (expanding every year)

Step 1: Login AWS Management Console

Step 2: Click on the region name in the right-side corner you can find list regions

Regions – Advantages

* Low Latency
* Global Footprint
* Adhere to government **regulations**
* High Availability

Availability Zones

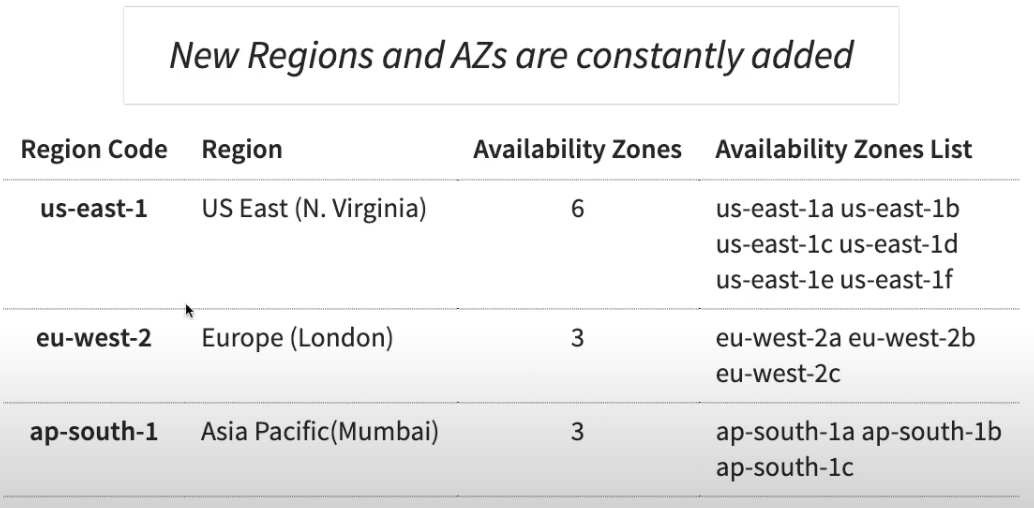
* Each AWS region consists of multiple, isolated, and physically separate AZ’s.
* Availability Zones in a Region are connected through **Low-latency** links.
* Each Availability Zone:

Can you **one or more discrete data centers**.

Has **redundant** power, networking, and connectivity.

* (Advantage) **Increase availability and fault tolerance** of applications in the same region.

Regions and availability of Zones



EC2 (Elastic Compute Cloud)



* In corporate data centers, applications are deployed to physical servers.
* Where do you deploy applications in the cloud?

Rent Virtual servers

**EC2 Instances** – virtual servers in AWS

**EC2 service** – provision EC2 instances or virtual servers.

EC2 Features

A logo of an elb

AI-generated content may be incorrect.

* Create and image lifecycle of EC2 instances
* **Attach storage** (&network storage) to your EC2 instances
* Manage **network connectivity** for an EC2 instance
* **Load balancing** and **auto scaling** for multiple EC2 instances

EC2 Hand-on

* Let’s create a few instances and play with them
* Let’s use EC2 instance connect to SSH into EC2 instances

How to find EC2?

Step 1: go to services

Step 2: search EC2 (virtual servers in the Cloud)

Step 3: select Running instances

Step 4: then click launch instances

Step 5: select amazon Linux 2 AMI (HVM), SSD Volume Type

Step 6: now you need select instance type now let’s go for default one

Step 7: click review and launch

Step 8: click launch

Step 9: once, you click launch you need to select key pair (if you have already select it otherwise create a new key pair)

Step 10: click new key pair and given name and you can download key pair.

Step 11: click launch instances.

Step 12: then, go to instance which you are create then you can see instance summary.

Step 13: click on connect

Step 14: click on connect

It will open in new tab as command prompt

* Whoami (if you search this command, it will show your user details)
* Python –version (it will show your python version)

You need install yum

Enter command 🡪 yum install httpd

Then it will ask yes/no to install if you enter yes, it will install

How to start

Enter 🡪 systemct1 start httpd

Enter 🡪 systemct1 enable httpd

EC2 concepts – AMI – Amazon Machine image



* What operating system and what software do you want on the instance?

1. Open Amazon machine image (AMI)
2. Search Windows
3. Choose anyone windows and create EC2 instance

* Three AMI sources:

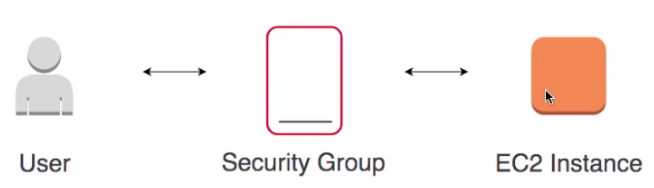
1. Provided by AWS
2. **AWS Market place:** online store for customized AMIs. Per hour billing
3. **Customized AMIs:** created by you.

EC2 concepts – Instance Families

* Optimized combination of **compute (CPU, GPU), memory, disk (storage) and networking** for specific workloads
* 270+ instances across 40+types for different workloads

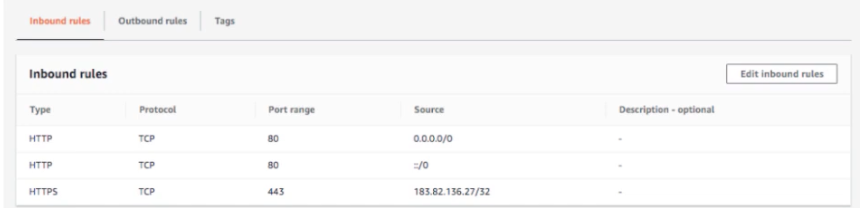
1. M (m4, m5, m6) – general purpose
2. C (c4, c5,c5n) – compute optimized
3. R (r4, r5, r5a, r5n) – memory (RAM) optimized
4. I (i3) – storage (I/O) optimized
5. G (g3, g4) – GPU optimize – graphics processing

EC2 Important Concepts – Security groups



* Virtual firewall to control incoming and outgoing traffic to/from AWS resources (EC2 instances, databases etc.)
* Provides additional layer of security – Defense in Depth

Security groups Rules



* **Default deny** – if there are no rules configured, no outbound/inbound traffic is allowed
* Allows **allow rules only**
* **Separate rules** for inbound and outbound traffic

EC2 security – key pairs

* EC2 use public key cryptography for protecting login credentials
* Key pair – public key and a private key

1. Public key is stored in EC2 instance
2. Private key is stored by customer

EC2 IP Addresses

* Public IP addresses are internet addressable.
* Private IP addresses are internal to corporate network
* You CANNOT have two resources with same public IP address.
* HOWEVER, two different corporate networks CAN have resources with same private IP addresses.
* All EC2 instances are assigned private IP addresses.
* (Remember) when you stop an EC2 instance, public IP address is lost.

Elastic IP Addresses

* Scenario: How do you get a Constant Public IP addresses for a EC2 instance?

1. Quick and dirty way is to use an Elastic IP!

* Elastic IP can be switched to another EC2 instance within the same region
* Elastic IP remains attached even if you strop the instance. You have to manually detach it.
* Remember: You are charged for an Elastic IP when you are NOT using it! Make sure that you explicitly release an Elastic IP when you are not using it.

EC2 Concepts – AMI – Amazon Machine Image

A close-up of a logo

AI-generated content may be incorrect.

* What operating system and what software do you want on the instance?
* Three AMI sources:

1. Provided by AWS
2. AWS market place: online store for customized AMIs. Per hour billing
3. Customized AMIs: created by you.

EC2 concepts – instance Families

* Optimized combination of compute (CPU, GPU), memory, disk (storage) and networking for specific workloads
* 270+ instances across 40+ types for different workloads

1. M (m4, m5, m6) – general purpose
2. C (c4, c5, c5n) – compute optimized
3. R ( r4, r5, r5a, r5n) – memory (RAM) optimized
4. I (i3) – storage (I/O) optimized
5. G (g3, g4) – GPU optimize – graphics processing

EC2 important concepts – security groups

A red rectangle with black text

AI-generated content may be incorrect.

* Virtual firewall to control incoming and outgoing traffic to/form AWS resources (EC2 instances, databases etc.)
* Provides additional layer of security – Defense in depth

Security groups Rules

* Default deny – if there are no rules configured, no outbound/inbound traffic is allowed
* Allows allow rules ONLY
* Separate rules for inbound and outbound traffic

IAAS (infrastructure as a service)

* Use **only infrastructure** from cloud provider

1. Computers (virtual or on dedicated hardware), data storage space and networking features

* Also called **“Lift and Shift”**
* **Example:** using EC2 to deploy your applications
* **Example:** using EC2 to create your database
* **Cloud provider** is responsible for:

1. Physical infrastructure (hardware, networking)
2. Virtualization layer (Hypervisor, Host OS)

* Customer is responsible for:

1. Guest OS upgrades and patches
2. Application code and runtime
3. Availability, fault tolerance, scalability etc.

PAAS (platform as a service)

* Use platform provided by cloud
* **Cloud provider** is responsible for:

1. OS (incl. upgrades and patches)
2. Application Runtime
3. Auto scaling, Availability & load balancing etc.

* Customer is responsible for:

1. Application code and /or
2. configuration

AWS Managed service offerings

A logo of a company

AI-generated content may be incorrect.

* Elastic Load Balancing – Disribute incoming traffic across multiple targets
* AWS Elastic Beanstalk – Run and Manage Web Apps
* Amazon RDS – Relational databases – MySQL, Oracle, SQL Server etc.
* And a lot more…

Elastic Load Balancer

Diagram of a diagram of a block diagram

AI-generated content may be incorrect.

* Distribute traffic across EC2 instances in one or more AZs in a single region
* **Managed service** – AWS ensures that it is highly available
* Auto scales to handle huge loads
* Load Balancers can be **Public or Private**
* **Health checks** – route traffic to healthy instances.

Three types of Elastic Load Balancers

* **Classic** Load balancer (layer 4 and Layer 7)

1. Old generation supporting layer 4(TCP/TLS) and Layer 7(HTTP/HHTPS) protocols
2. Not Recommended by AWS

* **Application** Load Balancer (Layer 7)

1. **Most popular** and frequently used ELB in AWS
2. New generation supporting HTTP/HTTPS
3. Supports advanced routing approaches (headers, Query params, path and Host Based)

* Network Load Balancer (Layer 4)

1. New generation supporting TCP/TLS and UDP
2. Very high performance usecases

Availability

* Are the Applications available when the users need them?
* Percentage of time an application provides the operations expected of it
* Example: 99.99% availability. Also called four 9’s availability

Availability Basics – EC2 and ELB

A diagram of a diagram

AI-generated content may be incorrect.

* Deploy to multiple AZs
* Deploy to multiple regions

Scalability

* A System is handling 1000 transactions per second. Load is expected to increase 10 times in the next month.

1. Can we handle a **growth in users, traffic, or data size** without any drop in performance?
2. Does ability to serve more growth increase proportionally with resources?

* Ability to **adapt** to changes in demand (users, data)
* What are the options that can be considered?

1. Deploy to a bigger instance with bigger CPU and more memory
2. Increase the number of application instances and setup a load balancer
3. And a lot more

Vertical scaling



* Deploying application/database to bigger instance

1. A larger hard drive
2. A faster CPU
3. More RAM, CPU, I/O, or networking capabilities

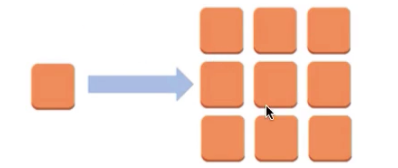
* There are limits to vertical scaling

Vertical scaling for EC2

* Increasing EC2 instance size:

1. T2. Micro to t2. small or
2. T2. small to t2.2xlarge

Horizontal scaling



* Deploying multiple instances of application/database
* (typically, but not always) Horizontal scaling is preferred to vertical scaling:

1. Vertical scaling has limits
2. Vertical scaling can be expensive
3. Horizontal scaling increases availability

* (BUT) Horizontal scaling needs additional infrastructure:

1. Load Balancers etc.

Horizontal scaling for EC2

* Distribute EC2 instances

1. In a single AZ
2. In multiple AZs in single person
3. In multiple AZs in multiple regions

* **Auto scale:** auto scaling group
* **Distribute load:** Elastic load Balancer

EC2 Tenancy – shared vs Dedicated

* Shared Tenancy (default)

1. Single host machine can have instances from multiple customers

* EC2 dedicated instances

1. Physical servers dedicated to one customer
2. You have visibility into the hardware of underlying host (sockets and physical core)
3. (use cases) Regulatory needs or server0bound software licenses like windows server, SQL server

EC2 pricing Models overview

|  |  |  |
| --- | --- | --- |
| Pricing Model | Description | Details |
| On Demand | Request when you want it | Flexible and most expensive |
| Spot | Quote the maximum price | Cheapest (upto 90% off) BUT NO guarantees |
| Reserved | Reserve ahead of time | Upto 75% off. 1- or 3-years reservation. |
| Savings plans | Commit spending $X per hour on (EC2 or AWS fargate or Lambda) | Upto 66% off. No restrictions. 1 or 3 years reservation. |

EC2 om-Demand

* On demand resource provisioning – use and throw!
* Highest cost and highest flexibility
* This is what we have been using until now in this course
* Ideal for:

1. A web application which receives spiky traffic
2. A batch program which has unpredictable runtime and cannot be interrupted
3. A batch program being moved from on-premises to cloud for the first time

EC2 spot instances

* **(0ld Model)** Bid price. Highest bidder wins
* **(new Model)** quote your maximum price. Prices decided by long trends
* Upto 90% off (compared to On-demand)
* Can be terminated with a **2-minute** notice
* Ideal for **non-time critical workloads** that can **tolerate interruptions** (fault-tolerant)

1. A batch program that does not have a strict deadline AND can be stopped at short notice and re-started

EC2 reserved instances

* Reserve EC2 instances ahead of time
* Get upto 75% OFF!
* Payment methods

1. No upfront - $0 upfront. Pay monthly installment.
2. Partial upfront - $XYZ upfront. Pay monthly installment.
3. All upfront – full amount upfront. $0 monthly installment.
4. Cost wise: earlier you pay, more the discount. All upfront<partial upfront<No upfront
5. A difference upto 50%

EC2 Savings plans

* EC2 compute Savings Plans

1. Commitment: I would spend X dollars per hour on AWS compute resources (Amazon EC2 instances, AWS fargate and/or AWS Lambda) for a 1- or 3-year period.
2. Upto to 66% off (compared to on demand instances)
3. Provides complete flexibility:
4. You can change instance family, size, OS, tenancy or AWS region of your Amazon EC2 instances
5. You can switch between amazon EC2, AWS fargate and/or AWS Lambda

* EC2 instance Savings Plans

1. Commitment: I would spend X dollars per hour on amazon EC2 instances of a specific instance family (general purpose, for example) within a specific region (us-east-1, for example)
2. Upto 72% off (compared to on demand instances)
3. Youn can switch operating systems (windows to Linux, for example)

AWS Elastic Beanstalk

* Next level of platform as a service!
* Simplest way to deploy and scale your web applications in AWS

1. Provides end-to-end web applications management

* Supports Java, .Net, Node.js, PHP, Ruby, Python, Go, and docker applications
* No usage charges – pay for AWS resources provisioned
* Features:

1. Automatic load balancing
2. Auto scaling
3. Managed platform updates
4. Application health monitoring

AWS Elastic Beanstalk Concepts

* Application – A container for environments, versions and configuration
* Application version – A specific version of deployable code (stored in S3)
* Environment – An application version different application versions for the same application.

Auto Scaling Components

A diagram of a person's auto scaling group

AI-generated content may be incorrect.

* Launch configuration/Template (what?)

1. EC2 instance size and AMI

* Auto scaling group (where?)

1. Min, max and desired size of ASG
2. Health checks

* Auto scaling policies (when?)

1. When and how to execute scaling?

Dynamic scaling policy types

A group of squares with text

AI-generated content may be incorrect.

A white paper with black text

AI-generated content may be incorrect.

Scaling Policies – Background

A logo with arrows and text

AI-generated content may be incorrect.

* Two parts:

1. CloudWatch alarm (Is CPU utilization >80%? Or <60%).
2. Scaling action (+5 EC2 instances or -3 EC2 instances)

Serverless

A logo with text on it

AI-generated content may be incorrect.

* What are the things we think about when we develop an application?

1. Where do we deploy the application?
2. What kind of server? What OS?
3. How do we take care of scaling the application?
4. How do we ensure that it is always available?

* **What if we do not need to worry about servers and focus on building our application?**
* Enter **serverless**
* Remember: **Serverless does NOT mean “No servers”**
* **Serverless for me:**

1. You don’t’ worry about infrastructure
2. Flexible scaling
3. Automated high availability
4. Pay for use:
5. You don’t have to provision servers or capacity!

* **You focus on code** and the cloud managed service takes care of all that vis needed to scale your code to serve millions of requests!

AWS Lambda

A close-up of logos

AI-generated content may be incorrect.

* World before lambda – ELB with EC2 servers!
* You don’t worry about servers or scaling or availability
* You only worry about your code
* You pay for what you use

1. Number of requests
2. Duration of requests
3. Memory consumed

AWS Lambda – supported Languages

* Java
* Go
* PowerShell
* Node.js
* C#
* Python,
* Ruby
* And a lot more…

AWS Lambda Event sources

* Amazon API gateway
* AWS Cognito
* Amazon DynamoDB (event)
* Amazon CloudFront (Lambda@Edge)
* AWS Step Functions
* Amazon Kinesis (event)
* Amazon simple storage service
* Amazon simple Queue service (event)
* Amazon Simple notification service