**EC2 Elastic Compute Cloud Introduction**

**Pre-Introduction**

* AWS Identity and Access Management (IAM) provides fine-grained access control across all of Amazon Web Services. With IAM, you can specify who can access which services and resources, and under which conditions. With IAM policies, you manage permissions to your workforce and systems to ensure least-privilege permissions.
* First and foremost, to be part of the IAM global service. The root user must add you to the IAM service

**Budget Setups – Setup IAM users to access billing data & creating a budget**

* 1. While logged in as the Root Use
* 2. Click on “my account”. Then click edit on “IAM User and Role Access to Billing Information”
* 3. Check the mark of “Activate IAM Access” Then click update
* This will allow, IAM users with administrative privileges to access billing data
* 4. Now while logged in as an IAM user with administrative privileges. You can click “Billing Dashboard” to access the AWS Billing Dashboard. Click on   
  Bills” to see the charges . If you would like to create a budget, you can click on “Create budget” in “Budgets” under “Cost Management”

**Introduction to Amazon EC2**

* EC2 stands for -> Elastic Compute Cloud and it is an Infrastructure as a Service (IaaS) cloud service
* Infrastructure as a Service (IaaS) is a cloud computing service where enterprises rent or lease servers for compute and storage in the cloud. Users can run any operating system or applications on the rented servers without the maintenance and operating costs of those servers.
* The user manages: The applications, data, runtime, middleware, operating system.
* AWS manages: virtualization, networking, storage, servers.
* Main EC2’s Capabilities:
* Renting virtual machines. When renting them using the EC2 service those machines are called EC2 instances
* Store the data on virtual drives known as Elastic Block Store (EBS). EBS is Amazon's block-level storage solution used with the EC2 cloud service to store persistent data
* Distribute the load across multiple machines using Elastic Load Balance (ELB) of EC2
* Scale services using Auto Scaling Group (ASG) of EC2
* The user’s responsibilities in the configuration of EC2’s virtual servers aka instance(s):
* 1. You must select the operating system (OS) of this virtual machine. You have three options : Linux, Windows, or Mac OS. Linux is the most popular
* 2. You must select the CPU of the instance(s). This includes power & cores
* 3. You must select the Random Access Memory (RAM)
* RAM and Hard Desk Drive (HDD), are both types of computer memory. RAM is used to store computer programs and data that CPU needs in real time. RAM data is volatile and is erased once computer is switched off. HDD, hard disk has permanent storage and it is used to store user specific data and operating system files
* 4. You need to select the amount of storage space. Storage space can be attached through:
* The network via (EBS & Elastic File System (EFS))
* The hardware via (EC2 instance Store)
* 5. You need to select the type of network that needs to be attached to your EC2 server (instance): You can choose the speed of the network card and the type of public IP address
* 7. You need to setup a “security group” to handle the firewall rules
* 8. Finally, you need to configure the Bootstrap script. This is only configured at first launch. This is known as EC2 User Data
* EC2 User Data
* It is possible to bootstrap our instances using an EC2 User data Script. This script is bootstrapped so it runs once only when the machine starts
* Bootstrapping: The ability to launch commands when the machine starts
* The more commands added to the User Data Script. The more computing power and time the server will require every time you run it. Because the server will boot strap so it will run all these commands
* The EC2 Data Script only works with the Root User. Therefore, when we bootstrap EC2 User Data Scripts. Those commands will have the sudo privileges
* The EC2 User Data’s purpose is to automate boot tasks. These tasks include:
* Install updates
* Installing software
* Download common files from the internet
* Can be anything but the first three are common
* EC2 Instance Types (Examples)
* The below image displays different type instances. T2.nano is the most basic instance or a server

Graphical user interface, table

Description automatically generated

**Launching An EC2 Instance using Linux**

* The steps to launch an EC2 instance using AWS:
  + 1. In the AWS Management Console. Search then go to EC2 to go to the EC2 console
  + 2. Make sure you have the correct region in your region’s selections
  + 3. Click on “Instances”. Then Click on “Launch instance”
  + 4. Step 1. You must choose an AMI (Amazon Machine Image). An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.
    - We selected “Amazon Linux 2 AMI (HVM) - Kernel 5.10, SSD Volume Type - ami-041d49677629acc40 (64-bit x86) / ami-0026aec60d20d226d (64-bit Arm)”
  + 5. Step 2. Select the instance type. We selected “t2 micro”. Then click “Next: Configure Instance Details”
  + 6. Step 3. We can modify the details in this step. In the course we did not modify anything, expect the User data. The User Data will be launched during the first very launch of the EC2 instance and it will only be launched once. Note vpc stands for Virtual Private Cloud/.
    - In the User data. We want to launch a webserver into the EC2 instance. Then, write a file into that web server. Thus we added the below in the User data box as a text. Then we clicked “Next Add Storage”:

#!/bin/bash

# Use this for your user data (script from top to bottom)

# install httpd (Linux 2 version)

yum update -y

yum install -y httpd

systemctl start httpd

systemctl enable httpd

echo "<h1>Hello World from $(hostname -f)</h1>" > /var/www/html/index.html

* + 7. Step 4. In the “Add Storge” page we can select the storage we can select the memory on the EC2’s instance. We left it as default
  + 8. Step 5. We can add tags. A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver. Tags are case sensitive. Therefore, if you want to add a name to the instance’s name. That tag’s key must be written as Name.
  + 9. Step 6. We must Configure the Security Group. We made sure that the “Assign a security group” was on “Create a new security group”. Then we clicked on “Add Rule” to add a new rule. In the new rule. The Type is HTTP, The Protocol is TCP, The Port Range is 80, The Source is Custom, The Description we wrote “
    - When the Source is -> Custom and 0.0.0.0/0, ::/0
    - This means the source is from anywhere. which, allows all IP addresses to access our EC2 instance
    - HTTP: Hypertext Transfer Protocol. WordlWideWeb is about communication between web clients and servers. Communication between client computers and web servers is done by sending HTTP Requests and receiving HTTP Responses. HTTP is the protocol used by the www systems. HTTP is the way data is transferred between websites and it is not most secure way. The steps to use HTTP are:
      * 1. A client (a browser) sends an HTTP request to the web
      * 2. A web server receives the request
      * 3. The server runs an application to process the request
      * 4. The server returns an HTTP response (output) to the browser
      * 5. The client (the browser) receives the response
    - The World Wide Web (WWW), commonly known as the Web, is an information system where documents and other web resources are identified by Uniform Resources.
    - HTTPS: is a secure version of the HTTP protocol that uses the SSL/TLS protocol for encryption and authentication. HTTPS is specified by RFC 2818 (May 2000) and uses port 443 by default instead of HTTP's port 80
    - SSH: SSH means “Secure Shell”. It has a built-in username/password authentication system to establish a connection. It uses Port 22 to perform the negotiation or authentication process for connection. Authentication of the remote system is done by the use of public-key cryptography and if necessary, it allows the remote computer to authenticate users. It is a secure way to transfer data between servers.
    - A port in networking is a software-defined number associated to a network protocol that receives or transmits communication for a specific service. In a specific cloud server, a specific port tells the server that we are requesting a specific service. Port numbers identify specific applications or services on a computer system, just like an IP address identifies a machine in an IP network and ascertains the destination of a data packet. A port is always associated with a protocol.
    - TCP: Transmission Control Protocol (TCP) is a standard that defines how to establish and maintain a network conversation by which applications can exchange data. TCP works with the Internet Protocol (IP), which defines how computers send packets of data to each other.
    - Note: ssh HTTP, HTTPS are protocols to communicate between client and server
  + 10. Step 7. Review the inputs and Click on “Launch” when done.
  + 11. Step 9. Once you click Launch. You must select a “Key Pair”. We clicked on create a Key Pair. Chose RSA and Download the Key Pair and save it. Then, when done click “Launch Instance”
    - Key Pairs are what the SSH used to access the EC2 instance. A key pair consists of a public key that AWS stores, and a private key file that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance. Amazon EC2 supports ED25519 and RSA key pair types.
    - SHH is: SSH: SSH means “Secure Shell”. It has a built-in username/password authentication system to establish a connection. It uses Port 22 to perform the negotiation or authentication process for connection. Authentication of the remote system is done by the use of public-key cryptography and if necessary, it allows the remote computer to authenticate users. It is a secure way to transfer data between servers.
    - RSA is: RSA (Rivest–Shamir–Adleman) is a public-key cryptosystem that is widely used for secure data transmission. It is a signature algorithm for key pairs that SSH supports

**Working with a Launched EC2 Instance using Linux**

* Steps To work with an EC2 instance
  + 1. In the AWS Management Console. Search then go to EC2 to go to the EC2 console
  + 2. Click on “Instances”. Then Click on select your EC2 instance
  + 3. Double click the number in the Public IPv4 address. Then paste it in the address input for a chrome tab. Do not click on “open address”. Then as a result you should get back something that looks like this: Hello World from ip-172-31-15-49.ca-central-1.compute.internal
    - This means we used the public IP to access the instance and the instance replied with the private UP address and the region. ip-172-31-15-49 is the private IP address in this instance example
    - “open address” will not work, because a secure HTTP has not been configured yet
* If we are not using the instance. We can select “Stop Instance” under the instance state. Once we stop an instance. The public IP address of the instance will change. While the private IP address does not change

**EC2 Instance(s) Types - Overview**

* AWS offers a wide selection instance(s) types. Which are used to fit different use cases. Instance(s) types vary in terms of: CPU, memory, storage, and networking capacity
* Network bandwidth is a measurement indicating the maximum capacity of a wired or wireless communications link to transmit data over a network connection in a given amount of time. Typically, bandwidth is represented in the number of bits, kilobits, megabits or gigabits that can be transmitted in 1 second.
* Each instance type includes one or more instance sizes, allowing the user to scale up or down their resources based on the requirements of the target workload
* The instance(s) are displayed in the following webpage: <https://aws.amazon.com/ec2/instance-types/?trk=8c0f4d22-7932-45ae-9a50-7ec3d0775c47&sc_channel=ps&sc_campaign=acquisition&sc_medium=ACQ-P|PS-GO|Brand|Desktop|SU|Compute|EC2|CA|EN|Text&s_kwcid=AL!4422!3!536323134234!e!!g!!ec2%20instance%20types&ef_id=EAIaIQobChMImoeK8tO79gIVL3FvBB1UiwhaEAAYASABEgK_kfD_BwE:G:s&s_kwcid=AL!4422!3!536323134234!e!!g!!ec2%20instance%20types>
* To compare the instance(s) go to the following webpage: https://instances.vantage.sh/
* EC2 instance(s) naming convention:
  + m5.2xlarge
    - m: is the instance class
    - 5: Is the generation of the instance (AWS upgrades the instances over time)
    - 2xlarge: The size within the instance class
* Instance Types:
  + General Purpose Instance(s):
    - General purpose instances provide a balance of compute, memory and networking resources, and can be used for a variety of diverse workloads. These instances are ideal for applications that use these resources in equal proportions such as web servers and code repositories.
    - Great for diversity of workload and works well for: web servers or code repos
    - Have a great balance between computing, memory, and networking
  + Compute Optimized Instance(s):
    - Compute Optimized instances are ideal for compute bound applications that benefit from high performance processors. Instances belonging to this family are well suited for batch processing workloads, media transcoding, high performance web servers, high performance computing (HPC), scientific modeling, dedicated gaming servers and ad server engines, machine learning inference and other compute intensive applications.
    - Great for computing. They provide high performance CPUs. Therefore, they can process high number of calculations.
    - They are good for: batch processing workloads, media transcoding, high performance web servers, high performance computing (HPC), scientific modeling/machine learning, and high-performance gaming servers
  + Memory Optimized Instance(s)
    - Memory optimized instances are designed to deliver fast performance for workloads that process large data sets in memory.
    - Great for storing memory. They provide high performance instance(s) with large RAM. (RAM along with registers and cache make up volatile memory). Virtual memory connects the volatile memory with the secondary memory . We say volatile, because everything is lost once the power is cut off
    - They are good for: high performance relational/non-relational databases, distributed web scale cache stores, in-memory databases optimized for BI (business intelligence), and applications performing real-time processing of big unstructured data
  + Accelerated Computing
    - Accelerated computing instances use hardware accelerators, or co-processors, to perform functions, such as floating point number calculations, graphics processing, or data pattern matching, more efficiently than is possible in software running on CPUs.
  + Storage Optimized Instance(s)
    - Storage optimized instances are designed for workloads that require high, sequential read and write access to very large data sets on local storage. They are optimized to deliver tens of thousands of low-latency, random I/O operations per second (IOPS) to applications.
    - Great fir storage intensive tasks that require high, sequential (read/write) access to large sets of data
    - They are good for: High frequency online transaction processing (OLTP) systems, Relational & NoSQL databases, Cache for in memory data bases (e.g., Redis), Data warehousing applications, Distributed file systems

**EC2 Instance(s) Security Groups**

* Security groups act as the fire wall for the EC2 instance(s).
* Security groups only contain “allow” rules. Which allow or not allow specific actions. Actions namely are inbound traffic into the EC2 instance from a www webpage & outbound traffic from the EC2 instance
* Security Groups operate at instance level and can control traffic.
* A good example is that the inbound of a security group must be through a specific port and by a specific IP address.
  + Therefore, if a server attempts to access the EC2 server using an allowed IP address but a different port they will be denied.
  + The same if a server attempts to access an EC2 instance with the correct port but they have an unallowed IP address they also will be denied.
  + The only servers that are allowed to access EC2 instances are the ones that have an allowed IP address and accessing through the correct port
* If you attempt to connect to an instance(s) but it takes a very long time, then the browser times you out. Then this is a security group issue
* The rules of security groups can be referenced via security group or via IP address
* Security groups regulate:
  + Access to ports
  + Authorized IP ranges. Either IPv4 and IPv6
    - IPv4 stands for Internet Protocol version 4. It is the underlying technology that makes it possible for us to connect our devices to the web. Whenever a device accesses the Internet, it is assigned a unique, numerical IP address such as 99.48. 227.227.
    - IPv6 is the next generation Internet Protocol (IP) address standard intended to supplement and eventually replace IPv4, the protocol many Internet services still use today. Every computer, mobile phone, home automation component, IoT sensor and any other device connected to the Internet needs a numerical IP address to communicate between other devices. The original IP address scheme, called IPv4, is running out of addresses due to its widespread usage from the proliferation of so many connected devices.
  + Control inbound network from others to the EC2 instance
  + Control outbound network from EC2 instance to others
* Security group types
  + HTTP, SSH, PostgreSQL, DNS, All traffic, and many more
  + You can also enter a custom type of connection under “Custom TCP”
* Security group protocols
  + TCP
* Security groups port range
  + The ranges are the ports that the Security groups allow the traffic to move within and can be:
    - 80
    - 22
    - 4567
* Security groups source
  + The source represents the IP address range
  + 0.0.0.0/0 means any IP address
* Other Security groups rules features
  + Security group rules are not an instance limited. They can be applied over multiple instances. Also, a single instance can have multiple security group rules
  + security groups are specific per region and per VPC. Therefore, if you create another region or VPC you must create a new security group
    - Amazon Virtual Private Cloud (Amazon VPC) enables you to launch AWS resources into a virtual network that you've defined
  + The security groups operate outside the EC2 instance. Thus, if an inbound traffic is blocked the EC2 instance(s) will not see it
  + A good practice is to maintain one separate security group for the SSH access. This is because SSH accesses are complex and very important
  + If you attempt to access a security but then you timeout then there is an error in the security group
  + If you attempt to access the security group but you get “connection refused” this means you the security group worked but you were denied access”
  + By default, all inbound traffic to an EC2 instance is blocked by security groups
  + By default, all outbound traffic from an EC2 instance is allowed by security groups
  + Authorizing two security groups of two different instances allows the instances to connect. This is known as Referencing other security groups
* Security groups ports to know for the AWS exam
  + Port 22 can be used for SSH (SecureShell) access. Which is used to log into a Linux instance(s)
  + Port 22 can also be used for SFTP(Secure File Transfer Protocol), which is the transfer of files via SSH. To improve the security of the file transfer
  + Port 21 can be used For FTP (File Transfer Protocol). Which is the transfer of files insecurely .
  + Port 80 used for HTTP. This port is used to access websites insecurely
  + Port 443 used for HTTPS. This port is to access websites securely
  + Port 3389 used for RDP(Remote Desktop Protocol). This port is used to log into Windows instance(s)

**Access Security Groups of EC2 Instance(s) via AWS console**

* Security groups allow users to control networking access to their EC2 instance(s). To access security groups and …… follow the following steps:
  + 1, While logged in in EC2 page. Click on “Security Groups” under “Network & Security”. This will show you the settings you need for security groups. You can:
  + Create new rules
  + Modify inbound or outbound rules
  + Also, you can reference security groups to each other. While selecting the source of the security group
* Security Group ID: It is used to identify the security group within your AWS account

**SSH Summary Table**

* SSH protocol is used to connect to the EC2 instance(s)
* SSH is a command line-based utility that can be used on Linux, Mac, and Windows version 10 and above Oss
* Other than SSH. There are:
  + Putty, which can be used to access windows before and after windows 10. Putty allows users to connect to windows machines using the SSH protocol
  + EC2 Instance Connect. This is a new method of connecting to EC2 instance(s) and it uses the web browser to connect to the EC2 instance. Also, it works with any OS

**Connect to the EC2 instance 🡪 SSH Into EC2 Instance (Linux/Mac)**

* SSH: allows you to login into your AWS EC2 machine and control everything via the command line
* The steps:
* 1. Copy the public IP address. Not the public DNS IP. They both are the same thing DNS just include the region as well. You can copy the DNS but you must delete the region when you logging on into the terminal
* 2. Check the inbound rules. Port 22 must be in the port range. The protocol must be TCP & the source must be 0.0.0.0/0 (which means anyone can login to this machine via port 22)
* 3. Open the command prompt or the terminal
* 4.Type the following to login into the virtual machine -> ssh -i <pwd the .pem key> ec2-user@<public IP address>
  + The .pem file is the key pair file you downloaded when you launched the EC2 instance
  + You should get the following once you logged on:
* Text

  Description automatically generated5. If you get a warning that the permission is denied & “Warning unprotected private key file”.
  + This occurs, because when you first download a file, it has a permission key of 0466, which means the file is open and accessible by others. If the file is open, then the key will leak. Therefore, Run this command, if necessary, to ensure your key is not publicly viewable -> chmod 0400 <pwd the .pem key
  + This is a common AWS question
* 6. Then run the following command again 🡪
  + ssh -i <pwd the .pem key> ec2-user@<public IP address>
  + You can also, connect to your instance using its Public DNS by typing ssh -i <pwd the .pem key> ec2-user@<public IP address DNS>
* 7. Type the following to know you logged in correctly 🡪 whoami
  + You should get back 🡪 ec2-user
* 8. To exit either type 🡪 exit
  + Or you can type Ctrl +d

**Connect to the EC2 instance 🡪 EC2 Instance Connect (Alternative Connection Method)**

* 1. Select “instances” in “EC2” in the “EC2 service”. Then click on the instance you want to connect too
* 2. Under “Actions” select “Connect”
* 3. You should be taken to “Connect to instance” page
* 4. There are multiple options to select. Make sure you select “EC2 Instance Connect”. Note, when we connect to the terminal it is the same as selecting “SSH client”. But you do not need to select anything. This is just for your knowledge if you connect via terminal or command prompt
* 5. Enter the following as a user name -> ec2-user
* 6. Click on Connect
* Note: EC2 Instance still uses SSH and in the security a rule for connecting with SSH must be enabled and the connection must happen at port 22

**Using IAM Roles In EC2 instance(s)**

* To use an IAM role in an EC2 instance.
* An IAM role is an IAM entity that defines a set of permissions for making AWS service requests. IAM roles are not associated with a specific user or group. Instead, trusted entities assume roles, such as IAM users, applications, or AWS services such as EC2.
* Therefore we will connect EC2 instance(2) to IAM users via an IAM
* 1. You first must log in into the EC2 instance(2) using the command line or EC2 instance connect
* 2. Once you log into the EC2 instance type 🡪 aws iam list-users
  + This will view the list of your aws iam users that have access to the EC2 machine. You might get a message that says “Unable to locate credentials. You can configure…..”
    - Now you can run the following command 🡪 aws configure
    - However, do not run the following command 🡪 aws configure
      * Running that command should not happen because:
        + When you configure your user ID using your credentials this will allow any other user who logs in into the EC2 instance to view your credentials and you can easily get hacked
        + A Rule of thumb. Never enter your IAM IP id (access key id & Secret access key) into an EC2 machine. Instead you must use IAM roles
* 3. Now to use an IAM role to access an EC2 machine you must do the following (The role will be attached to your iam user “admin user”)
  + 3.1 Go to IAM in the AWS services via the browser
  + 3.2 go into roles
  + 3.3 make a new role with a trusted entity of “AWS Service: ec2”. Also, this role has a policy name of “IAMReadOnlyAccess”
  + 3.4 Go to the “instances” and click on your EC2 instance in the EC2 service on AWS services via the browser
  + 3.5 Go to “security” in that instance and check what is written under the IAM role. If your newly made IAM user is there then you are fine. However, if the user is not there then go to 3.6
  + 3.6 From the list of “Actions” select “Security”. Then, select “Modify IAM role”
  + 3.7 Choose an IAM role from the list of IAM roles. The IAM user that you made in step 3.3. When done click “save”
  + 3.8 Go back to “security” in that instance and check what is written under the IAM role. The user you chose in step 3.7 should be there
* 4. In the command line you can type the following without the need to configure 🡪 aws iam list-users
  + You did not need to configure and create a user in the EC2 instance. Instead, you used a role. To configure in the EC2 instance you need to use your secure credentials
  + Typing 🡪 aws iam list-users
    - Will show you the list of users that have access to the EC2 instance
    - An IAM role is an IAM entity that defines a set of permissions for making AWS service requests. IAM roles are not associated with a specific user or group. Instead, trusted entities assume roles, such as IAM users, applications, or AWS services such as EC2.

**EC2 instance(s) Launch Types**

* The exam will ask:
  + 1. what is the best EC2 instance purchasing option. Based on cost or based on roles etc
* There are many types of instance(s) purchasing options
* 1. On Demand Instance: Those instances are made for short/predictable workloads. Their pricing is also predictable
* 2. Reversed Instance(s). Those instance(s) must be leased for a year minimum. They are categorized as:
* 2.1 Reserved Instances: The same instance(s) are used for long workloads
* 2.2 Convertible Reserved Instances: The instances(s) are flexible and their types can be changed over time and are also used for long workloads
* 2.3. The Scheduled Reserved Instance: Instance(s) used during specific time based on an agreed-on schedule. This type of reversed instance is “deprecated” but still might come in the exam
* 3. Spot Instance(s): These instances are cheap and made for short workloads. However, their biggest downside is that they can be lost. Therefore, they are not very reliable
* 4. Dedicated Hosts: This is the rental of entire server (supercomputer). Then when you rent a dedicated host, you control the EC2 instance fully.
* A server is a computer but much stronger. A distributed system is multiple computers

**EC2 instance(s) Launch Types – EC2 On Demand (What we used in the course)**

* Properties:
  + You pay for what you use:
    - As a Linux/Windows user: You are billed for every second after the first minute
    - As any other OS you are billed per hour
  + Has the highest cost in comparison to the other EC2 instance(s) types
  + Has no upfront cost
  + Does not require long-term commitment
* Cases:
  + Recommended to be used for “Short Term” and “Un-interrupted workloads”. Cases where you cannot predict how your application will behave

**EC2 instance(s) Launch Types – EC2 Reserved Instance(s) (General)**

* Properties:
  + Up to 75% discounted cost in comparison to EC2 On Demand launch type
  + Reservation
    - Either 1 or 3 years. You must reserve the instance for either 1 year or 3 years, not any time in between.
    - The longer the reservation the better the discount
  + Purchasing options are:
    - no upfront cost (monthly payments)
    - partial upfront cost (monthly payment)
    - full payment during purchase
    - The larger the payment up front is the better discount
  + Users must reverse specific types of instances and stick to them throughout the reservation period. Instances can not change
* Case
  + This type of instance is recommended for steady-state applications that will be needed for at least a year (databases)
  + This EC2 Purchasing Option should is ideally used for an application you plan on running on a server continuously for 1 year or 3 years. Because continuously mean we do not need change the instances

**EC2 instance(s) Launch Types – EC2 Reserved Instance(s) (Convertible)**

* Properties:
  + Up to 54% discounted cost in comparison to EC2 On Demand launch type
  + Reservation
    - Either 1 or 3 years. You must reserve the instance for either 1 year or 3 years, not any time in between.
    - The longer the reservation the better the discount
  + Purchasing options are:
    - no upfront cost (monthly payments)
    - partial upfront cost (monthly payment)
    - full payment during purchase
    - The larger the payment up front is the better discount
  + Users do not need to reverse specific types of instances and stick to them throughout the reservation period. Instances can be changed
* Case
  + This type of instance is recommended for steady-state applications that will be needed for at least a year (databases)

**EC2 instance(s) Launch Types – EC2 Reserved Instance(s) (Scheduled) “Deprecated”**

* Properties:
  + Up to 54% discounted cost in comparison to EC2 On Demand launch type
  + Reservation
    - Either 1 or 3 years. You must reserve the instance for either 1 year or 3 years, not any time in between.
    - The longer the reservation the better the discount
  + Purchasing options are:
    - no upfront cost (monthly payments)
    - partial upfront cost (monthly payment)
    - full payment during purchase
    - The larger the payment up front is the better discount
  + Users can specify a schedule on when they require the instance(s). For example: every Friday 8am -12 pm
  + Unsure if users can switch instance(s)
* Case
  + This type of instance is recommended for steady-state applications that will be needed for at least a year (databases)

**EC2 instance(s) Launch Types – EC2 Spot Instance(s)**

* Properties:
  + Up to 90% discounted cost in comparison to EC2 On Demand launch type
  + The most efficient launch type in comparison to all other launch types of EC2
  + Users may lose access to the spot instances at any point if the maximum price users are willing to pay is less than the current spot price. Spot Instance prices are set by EC2 and are adjusted based on trends in supply and demand.
  + Cases:
    - Jobs that you are not fully afraid to lose. In which you have already backed its data
    - Jobs with flexible start & finish times
    - Batch jobs
    - Data analysis
    - Image processing
    - Distributed workloads
    - Distributed systems, when you have multiple computers connected and if one fails (the spot instance) the other connected computers know how to react with that change in process
  + DO NOT use for critical jobs or databases

**EC2 instance(s) Launch Types – Dedicated Hosts**

* An Amazon EC2 dedicated host is a physical server with EC2 instance capacity that is fully dedicated to the user. Dedicated hosts **address compliance** **requirements**. Also, they reduce costs by allowing you to use your existing **server-bound software licenses.** They leverage pre-existing server-bound software licenses
  + Compliance requirements: Some organizations need to run their instances on dedicated servers instead of multi-tenant servers. With Dedicated Hosts, you get a physical server that is dedicated for your use. Dedicated Hosts provide visibility and the option to control how you place your instances on a specific, physical server. This enables you to deploy instances using configurations that help address corporate compliance and regulatory requirements. (e.g., following certain laws and requirements enacted by federal, state and local governments.)
  + server-bound software licenses: existing server-bound licenses, including Windows Server, SQL Server, and SUSE Linux Enterprise Server. Therefore, you can use your specific already existing software
* Properties:
  + Must be reserved for a 3-year period
  + It has the highest cost in comparison to the other launch types, because users are acquiring a full server for their service
  + Great to use if you are using software that have a complicated licensing model (BYOL) Bring your own license.
    - Dedicated Hosts also give you the visibility into the physical host that is required for a Bring Your Own License (BYOL) model — i.e., if you want to use your own Windows Server, SQL Server, SUSE, or RHEL licenses that are provided on a CPU core basis.
  + Great for companies with a strong/specific regulatory and compliance needs.
    - AWS allow users to share instances. There are security measures in which users of an instance can not see what others that share their instance are using. However, certain companies do not fully want to rely on AWS security measures, or they must follow laws that prevent that type of sharing . Thus dedicated hosts are ideal for that
  + Users are charged for the full host
  + Users can view sockets, cores, host id, affinity between a host and instance
  + Users can control the instance placement but also when purchased dedicated hosts allow for automatic instance placement
  + Users are able add capacity using an allocation request ( a request that indicates what should be allocated)
* **EC2 instance(s) Launch Types – Dedicated Instances**
* You can use Dedicated Hosts and Dedicated instances to launch Amazon EC2 instances on physical servers that are dedicated for your use
* An important difference between a Dedicated Host and a Dedicated instance is that a Dedicated Host gives you additional visibility and control over how instances are placed on a physical server, and you can consistently deploy your instances to the same physical server over time
  + As a result, Dedicated Hosts enable you to use your existing server-bound software licenses and address corporate compliance and regulatory requirements and dedicated instance do not
* Properties
* You cannot use existing server bound software on dedicated instances
* Users are charged per instance
* Users can not view sockets, cores, host id, affinity between a host and instance
* Users can not control the instance placement but when purchased dedicated instances allow for automatic instance placement

Users can not add capacity

**Shared responsibility model when using EC2 instance(s)**

* AWS is responsible for:
  + The infrastructure
  + The isolation of hosts (if dedicated hosts are selected)
  + Maintenance of hardware (replacement of faulty hardware)
  + Compliance validation (comply with all the laws that was agreed on)
* Users are responsible for:
  + The security to access the instance
  + The security of the data in the instance and whatever leaves the instance
  + Instance(s) OS/software updates
  + The status of all software available on the instance
  + Assigning IAM roles to EC2 instances to allow IAM users to access EC2

**EC2 Elastic Compute Cloud Storage**

1. **EC2 Instance(s) Storage Options -EBS (Network Virtual Storage)**

* Storage options for EC2 are:
* 1. EBS Volume 🡪 (Elastic Block Store) Volume
  + Definition: It is a network drive (not a physical drive) that can be attached into instance(s) while the instance(s) are running. (i.e., network USB stick).
  + EBS Volume are Network Drives
    - They are not physical storage devices
    - They might cause slight delay because the communication is occurring over the internet
    - They can be detached from one EC2 instance and be attached to another
  + Purpose: EBS Volume allows instance(s) to persist data (keep data), even after the instance(s) have been terminated
  + EBS Volume are availability zone (AZ) specific
    - An EBS Volume can not change zones (e.g., An EBS Volume in us-east-1 a cannot be attached to a us-east- 1 b)
    - To over come the AZ specific constraint. We can snapshot an EBS Volume from one AZ to another. In a way use the copied EBS Volume
  + EBS Volume have specific capacities, speeds and must be provisioned (set prior to the job)
    - The EBS Volume size in GB must be set prior to the operations
    - The EBS Volume operations speed must be set prior to the operations. The speed is known as IOPS (input/output operations per second)
    - SSG GB and IOPS can be increased over time
    - AWS bills users based on the provisioned capacity/speed
  + In the free tier, users get 30 GB of free EBS storage of type general purpose   
    (SSD) or magnetic per month
  + EBS Volume: can be left unattached to any EC2 instance(s) and be available to be used on demand
  + EBS Volume Termination
    - In the AWS console / AWS Command Lin, users can control the behavior of EBS volume when the EC2 instance(s) is deleted
    - By default:
      * The root EBS volume is deleted when the instance is deleted
      * Attached EBS volumes are not deleted when the instance is deleted
    - EBS Volume Termination type of control enable users to have the option preserve the root EBS volume storage when the EC2 instance is deleted
  + At the AWS exam level of CCP (Certified Cloud Practitioner) level all we need to know is that 🡪 1 EBS volume can be mounted on 1 EC2 instance. A single EBS Volume can not be attached to 2 or more EC2 instance(s). However, a single EC2 instance can have multiple EBS Volume Storage Units attached to it. This is to keep things simple. But in reality, we have EBS multi-attach feature we allows to attach multiple instances to a single EBS instance of the same availability zone, but this is out of the scoop of this course
* Volumes can be created. Then attached to EC2 instances

**EC2 Instance(s) EBS Volume Creation**

* Steps to create an EBS Volume storage:
  + 1. Go to EC2 services
  + 2. Select the region you have your instance(s) made in
  + 3. Click on instances. Then select the instance you want to create an EBS Volume for
  + 4. In the selected instance. Go to Storage. You can see the following there: The root device name, The root device type (EBS), the block devices.
    - In the block devices you can see the volume ID, The device name, The volume size, status, time, encryption, KMS key ID, and the option to “Delete on Termination. In the block devices only EBS Volumes that are attached to the instance are visible
  + 5. To create a new EBS Volume. Click on any of the attached to instance, EBS Volume under the “Volume ID” column.
  + 6. You will be taken into the “Volumes” page and in the “Volumes” page, click on create volume
  + 7. You will be taken to the “Create Volume page”
  + 8. When done click on “Create volume”
  + 9. The newly created volume will be available in the “Volume” page and under “Storage” of the specific EC2 instance. If you are in the “storage” tab you will not not be able to see the newly created volume until you attach it to the specific EC2 instance
  + 10. To attach a volume to an EC2 instance . Go to the “Volume” page. Right click on the EBS Volume you wish to attach and click “attach volume”. In the “Attach volume” window, select your “EC2 Instance”. Click on “Attach Volume” when done
* Note: The course instructor showed us how to attach the volume but not how to use it in our instance. Also, when we create an EC2 instance we will also select a root storage. If we do not tick the “delete on termination” option for the root storage then it wont delete when the instance terminates

**EC2 Instance(s) EBS Volume Snapshots (i.e., backup)**

* Backups to EBS volumes are known as snapshots.
* Snapshots can be taken at any point in time
* To create a snapshot of an EBS Volume it is recommended (But not necessarily) that the EBS Volume is detached from the EC2 instance(s)
* Commonly, backups from one EBS Volume are taken to transfer the data stored in the EBS Volume into different Availability Zones (AZ) or Regions

**EC2 Instance(s) EBS Volume Snapshot Creation**

* Steps to create an EBS Volume snapshot:
* 1. Go to EC2 services
* 2. Select the region you have yourEC2 instance is at
* 3. Select the instance you want to create a snapshot for its EBS Volume
* 4. Click on “storage”
* 5. Select the EBS Volume you wish to create a snapshot from
* 6. After selecting the EBS Volume you will be taken into a “Volumes” page. In the “Volumes” page, click on “Actions” and select “create snapshot”. Then you will be directed in to “Create snapshot” page
* 7. In the “Create snapshot” page. Add a description, encryption, and tags into your snapshot and click on “Create snapshot”
* 8. To verify the created snapshot. Go to “EC2 Dashboard” of EC2. Under the Resources, click “Snapshots”. The snapshot you created must be available there

**EC2 Instance(s) EBS Volume Snapshot Attachment to EC2 instance(s) in the same region but different availability zones (AZ)**

* EBS Volumes can be attached to EC2 Instance(s) only when both share the same AZ and Region
* Steps to attachment to another EC2 instance in another AZ in the same region
* 1. A backup of an EBS Volume must be created and it is known as snapshot. Prior to anything a snapshot must be created look at “**EC2 Instance(s) EBS Volume Snapshot Creation**”
* 2. The created snapshot is available in the “Snapshot” page under “Resources” in the “EC2 Dashboard” an EBS Volume snapshot
* 3. Click on “Actions” and select “Create Volume”. Select newly created volume’s settings including the all the availability zones (AZ) of the region
  + This Volume will only be available for attachment to instances in the “Availability Zone” you selected
* 4. Click “Create Volume” when done
* 5. The created volumes will be available at the “Volumes” page of the “EC2 Dashboard” with their availability zones mentioned
* The newly created EBS volume shares the same region as the previous EBS volume. However, the new volume can be attached to any eC2 instance that shares the same AZ as it.

**EC2 Instance(s) EBS Volume Snapshot Copy creation (For data backup purposes) to EC2 instance(s) in different region and different availability zones (AZ)**

* EBS Volumes can be attached to EC2 Instance(s) only when both share the same AZ and Region
* Steps to attachment to another EC2 instance in another AZ in different regions
* 1. A backup of an EBS Volume must be created and it is known as snapshot. Prior to anything a snapshot must be created look at “**EC2 Instance(s) EBS Volume Snapshot Creation**”
* 2. The created snapshot is available in the “Snapshot” page under “Resources” in the “EC2 Dashboard” an EBS Volume snapshot
* 3. Click on “Actions” and select “Copy”. Select the “Destination region” and add a “Description”
  + This copy of the snapshot will only be available for attachment to instances in the “Destination region” you selected
* 4. Click “Copy” when done
* 5. Change your “Region”. Then, in the “Snapshot” page select the snapshot you created and click on “Actions” and select “Create Volume From snapshot”.
  + This Volume will only be available for attachment to instances in the “Availability Zone” you selected
* 6. Click “Create Volume” when done
* 7. The created volumes will be available at the “Volumes” page of the “EC2 Dashboard” with their availability zones mentioned
* 8. Then you can attach that volume to any EC2 instance of the same region and AZ

**EC2 Instance(s) EBS Volume Snapshot Data Backup to Different Regions (In case of a disaster)**

* This is needed if you want to back up your data in different regions in case a disaster occurs in the region you are at
* 1. A backup of an EBS Volume must be created and it is known as snapshot. Prior to anything a snapshot must be created look at “**EC2 Instance(s) EBS Volume Snapshot Creation**”
* 2. The created snapshot is available in the “Snapshot” page under “Resources” in the “EC2 Dashboard” an EBS Volume snapshot
* 3. Click on “Actions” and select “Copy”. Select the “Destination region” and add a “Description”
  + This copy of the snapshot will only be available for attachment to instances in the “Destination region” you selected
* 4. Click “Copy” when done

**AMI (Amazon Machine Image)**

* AMI powers EC2 instance(s) are launched by AMI. The purpose an AMI is power the EC2 instance
* Customers can use the AMI provided by AWS or customize their own AMI
* Customization of AMI include:
  + Software, configuration, OS, monitoring, and many more
* Benefits of a customized AMI
  + The EC2 will have a faster boot/configuration because the packages are already installed in the AMI. Therefore, no need to install them on the EC2
  + A custom AMI can improve provisioning times when instances are launched in your environment if you need to install a lot of software that isn’t included in the standard AMIs. Using configuration files is great for configuring and customizing your environment quickly and consistently.
  + **Main Benefits Note**: In the case of the course our only benefit was the decrease of the bootstrapping time. However, in real life situation when you have many software packages. Those packages can travel with your AMI. Therefore, when you make a new EC2 instance based on an existing AMI then you will save the time it takes to install all the software packages
* AMI settings
  + AMI are built for specific regions and can be copied across regions
* The launch of an EC2 Instance using an AMI
  + Public AMI (Provided/maintained by AWS). In the course we are using Amazon Linux 2 AMI
  + Private AMI, will be made and maintained by the user
  + Semi-public AMI. This refers to the usage of AWS Marketplace AMI, which is an AMI that is made by a third party and sold by that party
* The process of creating an AMI from an EC2 Instance
  + In this case the AMI is the brain of the instance
    - 1. We create and customize an EC2 instance
    - 2. We stop the instance to maintain its data integrity
    - 3, Then, we build an AMI based on that EC2 instance. This step will also create EBS snapshots
    - 4. Now we can launch a new EC2 instance at any region. If we use the developed AMI from the initial EC2 instance. Then, the new EC2 instance will have all the settings/packages of the initial EC2 instance

**Steps to Creating an AMI (Amazon Machine Image) via EC2 Instance**

* 1. Log in to the EC2 main page
* 2. Launch an instance
  + In the bootstrap code section. Do not include:

echo “<h1>Hello World from $(hostname -f)</h1>” > /var/www/html/index.html

* + This is because we want to make an AMI based on this EC2 server where we install the “Apache HTTP Server” and we do not want to customize the html file yet. The html will be customized from the AMI at the run time of the AMI
* 3. If you try to copy the public ID of the instance and paste it into a browser . You will be taken to a “Test Page”.
  + Side Note: When you make an instance with bootstrapping you get a bootstrapping message
* 4. Now create an AMI from the newly made instance. In the “Instance” page, right click the recently made instance. Click on “Images and Temples” & select “Create image”
* 5. In the “Create image” page. Fill the name section. Then click on “Create image”
* 6. The newly created AMI should be found in the “AMI” section in the “EC2 Dashboard”
* 7. Now that an AMI has been created from the Instance. You can create instance(s) from that AMI by doing the following:
  + 1. Click on “Launch instance”
  + 2. In “Step 1: Choose an Amazon Machine Image (AMI)”. Go and click on ‘My AMIs” and your recently created image should be there
  + 3. Select the newly created AMI
  + 4. Choose an Instance Type & click “Next”
  + 5. In the “Advanced Details-User data” section of the “Configure Instance Details”. You can add only the following lines:

#!/bin/bash

echo “<h1>Hello World from $(hostname -f)</h1>” > /var/www/html/index.html

* + Note you do not need to customize the httpd anymore, because it has already been customized by AMI
    - This is the main benefits of an AMI. It will decrease the boot strapping time
    - A custom AMI can improve provisioning times when instances are launched in your environment if you need to install a lot of software that isn’t included in the standard AMIs. Using configuration files is great for configuring and customizing your environment quickly and consistently.
  + 6. Add storage
  + 7. Add tags
  + 8. Configure security group. Then, launch
* 8. Open the newly made EC2 instance from AMI & you should get a “Hello world” message

**EC2 Image Builder Overview**

* Generally, an image builder is an application that is used to automate the creation of “Virtual Machines” or “Container Images”.
* This section will be able to create instance(s) automatically and AMIs. We can run the pipelines to do both. Or we can create an instance manually and utilize the created AMI as a reference point. This is not the same as Creating an AMI (Amazon Machine Image) via EC2 Instance
* For AWS, the EC2 Image Builder automates the creation, maintenance, validation, and testing of EC2 instance(s) AMI
* The EC2 Image builder may run on a schedule (Time(Weekly), Action( every time applications are updated.))
* The EC2 Image Builder is a free service. Customers pay only for the underlying resource. The underlaying resources are: The actual EC2 instances, the storage, etc.
* AWS EC2 Image Builder Operational Steps
  + 1. An “Image Builder” is created
  + 2. When the EC2 “Image Builder” runs, it will automatically create a new instance that is called “Builder EC2 instance”
    - The Builder EC2 instance can build components & customize software, which may include:
      * Installing Java, Python, firewalls etc.
      * Update Command Line Interface (CLI)
  + 3. Once step 2 of creating “Builder EC2 instance” is complete a new AMI is created, and this AMI is based on the “Builder EC2 Instance”
  + 4. Then, the AMI is validated. “The Image Builder” will create a “Text EC2 Instance” to validate the AMI. The validation will consist of various tests that the user defines in advance. Tests may include:
    - Checking if the AMI is working
    - Checking if the AMI is secure & many more possible tests
  + 5. Once the testing/validation is complete and passed. The created “AMI” is distributed. The distribution can occur to multiple regions

**EC2 Image Builder In AWS. Pipelines Creation**

* This section will be able to create instance(s) automatically and AMIs. We can run the pipelines to do both. Or we can create an instance manually and utilize the created AMI as a reference point. This is **not** the same as Creating an AMI (Amazon Machine Image) via EC2 Instance
* The section deals with working within the “EC2 Image Builder” & creating “Image Pipelines”
  + An Image Pipeline: The pipeline automatically distributes your image to the Region where it runs the built. The pipeline is an automation configuration for building Amazon Machine Images (AMI). It can build AMI automatically
    - An Image pipeline is an automation configuration for building Amazon Machine Images (AMIs) or Docker images on AWS. The Pipeline’s configuration is broken down into three steps:
      * 1. The “Create Recipe” 🡪 This is the creation of the recipe document, which defines the components to be applied to the base image to produce the desired configuration for the output image. In other words the recipe defines how the source image will be customized
      * 2. The optional “Infrastructure Configuration” 🡪 This includes the settings of instances, VPC (Amazon Virtual Private Cloud), IAM roles, and other settings
      * 3. The optional “Distribution Settings” 🡪 The regions that are covered for the output AMI and the Docker image distribution
* 1. In the AWS Management Console search and go to the “EC2 Image Builder” service
* 2. Start creating a pipeline to automate the creation of (AMI). Click on “Create image pipeline”
* 3. In the “Specific Pipeline details” page fill/select the following”
  + 1. Add a name for the pipeline by filling the “Pipeline name”
  + 2. In the “Build schedule” section, select “Manual”
    - This section creates the automation schedule. We selected “Manual” just for the sake of the example
  + 3. Click “Next”
* 4. In the “Choose recipe” page fill/select the following”:
  + In the “Recipe” section click on “Create a new recipe”
    - You two options AMI and Docker Image
      * AMI: provides the information required to launch an instance.
      * Docker Image: a read-only template that contains a set of instructions for creating a container that can run on the Docker platform. It provides a convenient way to package up applications and preconfigured server environments, which you can use for your own private use or share publicly with other Docker users
  + In the “Image type” section, select “AMI”
  + In the “General” section. Fill the “Name” and the “Version”
  + In the “Base image – Select image” section, choose “Select managed images”
  + In the “Base image – OS” section, choose “Amazon Linux 2”
  + In the “Base image – Image origin” section, choose “Quick start (Amazon-managed)”
  + In the “Base image – Image name” section, choose “Quick start (Amazon Linux 2 x86)”
  + In the “Base image – auto-versioning options” section, choose “Use latest available OS version)”
  + In the “Components – Step 1: Choose build components to produce the desired output AMI-Build components-Amazon Linux” section. Build components are software scripts that define a sequence of steps for downloading, installing and configuring software packages. Therefore, select the following “build components – Amazon Linux”:
    - amazon-corretto-11-headless
    - aws-cli-version-2-linux
  + In the “Components – Step 1: Choose build components to produce the desired output AMI-Selected components” section. You can choose the sequence of which components would run first. Put “aws-cli-version-2-linux”. Then, put “amazon-corretto-11-headless”
  + In the “Components Step 2: Optional Select tests to verify the output AMI (post-build)- Test components” section. This section has multiple tests that are used to see if the AMI is working or not. In the lecture we skipped this part
  + In the “Storage (volumes) – optional” section. This section configures the pipelines storage or memory. In the course all the settings were kept at default and the size was kept at “8” Gib
  + Click “Next”
  + **Note**: an image recipe is a document that defines the base image to use as your starting point to create a new image, along with the set of components that you add to customize your image and verify that everything is working as expected.
* 5. In the “Define infrastructure configuration” page fill/select the following”:
  + In the “Infrastructure configuration” click on “Create a new infrastructure configuration”
  + In the “general section” we created an IAM role. By filling up the “Name” and the ‘IAM role”
    - A new “IAM role” was created by following these steps:
      * 1. Click on “create new role”
      * 2. You will be taken to the (IAM) dashboard and in there select “Roles”
      * 3. Click on “create role”
      * 4. In the “Trusted entity type” select “AWS service”. Then, in the “Use case” select “EC2”. Then, click “Next”
      * 5. In the “Add permissions” section add the following policies. Then, click “Next”:
        + EC2InstanceProfileForImageBuilder
        + EC2InstanceProfileForImageBuilderECRContainerBuilds
        + AmazonSSMManagedInstanceCore
      * 6. In the “Name, review, and create” page. Add the “Role name” I put “EC2IMAGEBuilder\_Lecture”. This is not default role name. The default name is “EC2InstanceProfileForImageBuilder”
      * 7. Click “Create role”
  + After creating the AMI role. In the “General” section of the “Define infrastructure configuration – optional” page. Add a “Name” and select the same role in the “IAM role” section
  + In the “AWS infrastructure section”
    - Select the “instance type” as “t2.micro”. The reason why we selected “t2.micro” because it is free
    - **Note**: The AWS infrastructure section allows users to customize the infrastructure (e.g., select the instance type)
  + Click on “Next” when done
  + **Note**: In this section we selected the type of EC2 instance that the image was created on.
* 6. In the “Define distribution settings – optional-Distribution settings” page
  + in this section, you want to see where your AMI will be distributed at. Start by selecting” Create distribution settings using service default” and if the “region settings” is in your regions. Then, you can click “Next”
  + Note: You can select “Create new distribution settings” and distribute your AMI into different regions
* 7. The is the “Review” section. Therefore, review the work you inputted and click “Create pipeline” when done
* Now when you want to create a new instance you can. Either run the pipeline which will create the instance automatically. Or you can create instance manually. But select the AMI you created as a reference point

**EC2 Image Builder In AWS. Pipelines Execution (running) To Create EC2 instances (Automatically)**

* The section deals with working within the “EC2 Image Builder” & running “Image Pipelines”
  + 1. In the “EC2 Image Builder” click on “Image pipelines”
  + 2. Select a pipeline (click the check mark only) then on the top right click on “Actions”. In “Actions” click on “Run pipeline”
  + 3. Now search and go to “EC2 Service”
  + 4. In the “instances” a new instance should have been automatically created. In terms of the course instance’s name is “Build instance for CourseRecip”
* Now when you want to create a new instance you can. Either run the pipeline which will create the instance automatically. Or you can create instance manually. But select the AMI you created as a reference point
* 3.In the “Specify pipeline details” page. Enter a “Pipeline name”, Build a schedule. In the lecture we select “Manual” Then clicked on “Next”
* 4. In the “Recipe” Page, click on “Create recipe”. Then, Select “Amazon Machine Image (AMI).
* 5. In the “Recipe” Page, in the “General” section enter a name and a version. Then in the “Base image” section. Click “Select Managed images”. In the “Image Operating System (OS), select “AMAZON Linux” . Then, in the “Image origin” select “Quick start (Amazon managed)”. Then, in the “image name” select “Amazon Linux 2 x86”. For the “Auto-versioning OS version” select “Use the latest available OS version”. Leave everything else up to components as the default settings
* 6. In the “Components” section of the “”Recipe” page. You can customize your Image. By:
  + Applying the build components that are prebuilt by AWS
    - The course selected the following components:
      * Amazon-crretto-11 headless
      * Aws0cli0version-2-linux
  + Create your own built components
* 7. In the “Components” section of the “”Recipe” page you can change the sequence of the components, because what ever is in the top will be installed first

**2.0 EC2 Instance(s) Storage Options -EC2 Instance Store (Hard drive Storage)**

* Unlike **EBS**, EC2 Instance store are hard drive storge. They are attached to the EC2 instances physically not over a network
* Benefits of EC2 Instance Store over EBS:
  + They provide higher storage
  + They provide better Input/Output I/O performance
* Pros of Instance Store
  + They are ideal to store: buffer, cache, scratch data, and temporary content
* Cons of Instant Store
  + They are known to be (ephemeral), meaning if the EC2 instance they attached goes down they also go down and all of the stored data on them is deleted
  + While using Instant Store you rick data loss if the hardware fails
  + Backups and replications are the duties of the user not AWS
* In the AWS exam
  + If an EC2 instance is presented with a “100% Random Read IOPs” and “Write IOPS” above 100,00 and 35,000, respectively. Then you can assume that the eC2 instance stores its data locally (i.e., Instant Store). An i3 instance is usually of local drive, which shows in their performance
    - IOPS (input/output operations per second). IOPS tells us how quickly each drive can process IO requests
    - ‘Sequential Read/Write Speed’ simply means that how fast that drive can write or read the data from a series of blocks.
    - Random means the files are scattered all over the drive, not in neat rows or groups, so take more work to find. Random IO is the most difficult and time-consuming type a storage device must deal with.
      * when we turn on our computer, it starts to process numbers of files from multiple locations, which simply means that the drive needs to access the data from random blocks, again and again.

**3.0 EC2 Instance(s) Storage Options -EFS (Elastic File System)**

* EFS is an NFS (Network File System) and it is a managed file system. Which means a single EFS can be attached to multiple EC2 instances at once using the “EFS Mount Target” which allows for this connection
  + This is not the case with EBS. However, if you want to attach an EBS to another EC2 instance you must make a snapshot of it and attach the snapshot to the new EBS. Also, you are not attaching the same EBS to multiple instances at the same time
* EFS works only with “Linux EC2 Instance(s)”
* EFS works across multiple availability zones AZ (e.g., (us-east1a, us-east1b, us-east1c) )
* EFS are:
  + widely available,
  + scalable,
  + expensive (3x the cost of gp2 of EBS). When using the EFS Standard Class
  + customers pay per use
    - No need to plan for capacity. You pay for what you use
* The EFS Standard Class
  + This is the default class used for EFS and it makes up the expensive of the EFS (3x the cost of gp2 of EBS
* The EFS Infrequent Access (EFS-IA)
  + EFS-IA is a storage class, and it is used for cost optimizations, because in it customers cab store files that they do not access every day
  + EFS-IA costs 92% less than EFS Standard
  + EFS-IA is activated by enabling EFS-IA with a “Lifecycle Policy”
    - For example. The “Lifecycle Policy” can be set to move files from EFS Standard to EFS-IA after 60 days if those files have not been open/edited. However, if you access those files that have been moved to EFS-IA then, those files will move back to EFS Standard and so on.
    - This is all done behind the scenes by the Elastic File System EFS (The idea of moving files between EFS Standard and EFS-IA). Therefore, the applications that accessing those files will not be impacted

**Shared responsibility between AWS and Customers in EC2 Storage**

* AWS us responsible for:
  + Infrastructure
  + Data replication for EBS Volumes and EBS drives
  + Maintenance of hardware and the replacement of faulty hardware
  + Ensuring that the customer’s data is secure and can not be accessed by anyone even AWS employees
* The Customer is responsible for
  + Setting up data’s backup and snapshots
  + Setting up data encryption (as a second layer of security)
  + Customers are responsible for the content of any data on the storage units or the drives
  + Customers are responsible to understanding how to use the second type of storage system in EC2. Namely EC2 Instance Store, because for this type of storage if the hardware fails the data will be lost and its on the customers to backup their data and customers’ can not accuse AWS on the hardware failure or lost of their data when they use EC2 Instance Store

**AMAZON FSx (Third Party File System)**

* FSx (File System x) is like Amazon’s EC2 EFS NFS (Network File System). However, it is a file system that is fully managed by a 3rd party :
* FSx are known to be high performance
* There are three types of Amazon FSx: Amazon FSx for Lustre, Amazon FSx for Windows File Server, and Amazon FSx for NetApp ONTAP. The main types are the first two types.
  + Amazon FSx for Windows File Server
    - Amazon FSx for Windows File Server provides fully managed Microsoft Windows file servers, backed by a fully native Windows file system
    - It works for windows powered machines and can also work for Instance(s) of windows OS
  + Amazon FSx For Lustre
    - FSx for Lustre is a fully managed high performance, scalable file storage for High Performance Computing (HPC)
    - The name Lustre comes from “Linux” and “Cluster”
    - It is used for Machine Learning, Analytics, Video processing, Financials, and Modeling
    - It can scale up to 100s GB/s and millions of IOPs

**Summary of Section EC2 Instance(s) Storage**

* EC2 Storage - EBS Volumes “Elastic Block Storage”
  + Network drivers (Not a physical driver)
  + Can be attached to one EC2 Instance at once. One EBS Volume for each EC2 instance
  + An EBS Volume cannot change zones (e.g., An EBS Volume in us-east-1 a cannot be attached to a us-east- 1 b)
  + To overcome the AZ specific constraint. We can snapshot an EBS Volume from one AZ to another. In a way use the copied EBS Volume
  + There are ways to copy an EBS Volume to EC2 instance in different regions and AZ
  + EBS Volumes allows instances' data to persist even after their termination.
  + EBS Snapshots are used to backup data on your EBS Volumes at a point in time.
* AMI: “Amazon Machine Image”.
  + It has the property of being able to create a ready to use EC2 instances. Therefore, it decreases the setup time
  + The are customized to our taste
  + You cannot use AMIs to add your IP addresses. IP addresses are added to an instance as you create it.
  + You can use AWS Marketplace AMIs to use someone else's AMI.
* EC2 Image Builder:
  + It is an Amazon builder than automates the process of building AMI, tests, and distributes AMIs
  + EC2 Image Builder is an automated pipeline for the creation, maintenance, validation, sharing, and deployment of Linux or Windows images for use on AWS and on-premises.
* EC2 Storage – EC2 Instance Store
  + It is a physical storage device not a network storage device
  + It has high performance
  + The main downside if the EC2 instance is stopped or terminated. Then all the data on the EC2 Instance Store are to be lost
  + EC2 Instance Store has a better I/O performance, but data is lost if: the EC2 instance is stopped or terminated, or when the underlying disk drive fails.
* EC2 Storage – EFS “Elastic File System”
  + A network file system that can support multiple instances in the same region. Therefore, over multiple AZ of the same region
  + By default, users will get EFS-Standard
* EC2 Storage - EFS Infrequent Access (EFS-IA)
  + EFS-IA is a storage class, and it is used for cost optimizations, because in it customers cab store files that they do not access every day
* EC2 Storage FSx For Windows
  + A Network File System for Windows Servers
* EC2 Storage FSx For Lustre
  + A Network File System for Linux Servers and used for high performance computing