**CLOUD: -** A network of remote servers hosted on the internet to store, manage, and process data, rather than using a local server or a personal computer

* **PUBLIC CLOUD: -** Services are delivered over the public internet and shared across organizations. Examples: -AWS, Google Cloud Platform, and Microsoft Azure.
* **PRIVATE CLOUD:** -Services are maintained on a private network, typically for a single organization, offering greater control and security. Examples: -VMware and OpenStack.

**WHY PUBLIC CLOUD: -**

1. **Scalability on demand**: Instantly scale resources up and down based on traffic and workload
2. **Cost effective**: Pay-as-you-go model
3. **Global accessibility**: Access the infrastructure and services anywhere in the world
4. **Speed & agility**: Launch servers, databases, or entire environments in minutes
5. **Security & compliance**: Major providers (AWS, Azure, GCP) offer enterprise-grade security, encryption, and compliance certifications

**AWS SERVICES:** -

* **IAM (Identity and Access Management): -** It provides centralized control over AWS accounts, allowing us to manage users, groups, roles, and permissions. we can access what resources the user is using within our AWS environment
* **USERS: -** Individual identities used for authentication when accessing AWS services and resources.
* **GROUPS: -** Collections of users that share the same set of permissions, making it easier to manage permissions for multiple users simultaneously.
* **POLICIES: -** permissions are attached to users, groups, or roles to specify what actions they can perform on which AWS resources.
* **ROLES: -** IAM identities that can be assumed by AWS resources, such as EC2 instances or Lambda functions, to access other AWS services securely.
* **EC2(Elastic Cloud Compute): -** A virtual machine which has the resources that a physical server has. Like memory, CPU, storage etc.
* We can install a hypervisor on top of the physical server to create a multiple virtual machine

1. **GENERAL PURPOSE EC2 INSTANCE**
2. **COMPUTE OPTIMIZED EC2 INSTANCE**
3. **MEMORY OPTIMIZED EC2 INSTANCE**
4. **STORAGE OPTIMIZED EC2 INSTANCE**
5. **ACCELERATED COMPUTE EC2 INSTANCE**

* Inorder to create an EC2-instance and connect, the following fields are mandatory

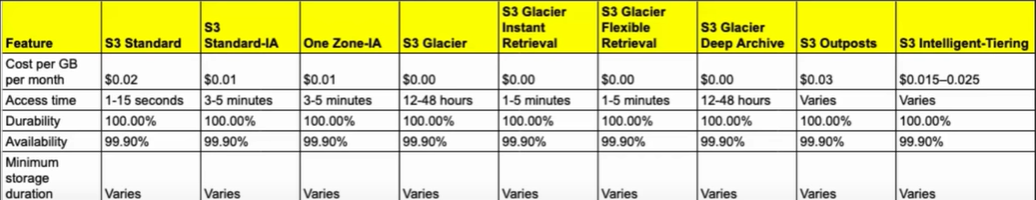
1. Click on **LAUNCH INSTANCE**
2. Give the **NAME**
3. Select **OPERATING SYSTEM**
4. Select **INSTANCE TYPE**
5. Give the **KEY-PAIR VALUE (Combination of public and the private key)**
6. Select **RSA as KEY PAIR TYPE and give the KEY PAIR NAME**
7. Then select **.PPK{Only for PUTTY} OR .PEM{For OpenSSH(**ex: - MobaXterm**)}**
8. Then click on **LAUNCH INSTANCE**
9. Use “**ssh -i example.pem ubuntu@public\_ip\_of\_ec2instance**” to connect to an ec2 instance through **OpenSSH.** For this we need to be in the location where this particular .pem file exists
10. If it shows “permissions are too open” then change the permissions of the .pem file using **chmod 777 filename**

* **VPC (Virtual Private Cloud): - A VPC is like a own data center inside a public cloud**. A Virtual network allowing us to launch AWS resources, such as Amazon EC2 instances, into a virtual network that we've defined.
* **INTERNET GATEWAY: -** A component of VPC that allows communication between instances in VPC and the internet. Public subnet is connected to the internet gateway
* **PUBLIC SUBNET: -** Component of VPC that divide the ip address range. Public subnet is accessible from the internet.
* **ELASTIC LOAD BALANCER: -** Elastic Load Balancing is a service by AWS that distributes incoming traffic across different resources
* **ROUTE TABLE: -**The route table contain rules that determined how traffic is diverted in VPC
* **SECURITY GROUPS: -**Act as virtual firewalls to control inbound and outbound traffic at the instance level and subnet level
* **PRIVATE SUBNETS: -** Component of VPC that divide the ip address range. Private subnet is not accessible from the internet.
* **NAT GATEWAY: -** Allow instances in a private subnet to connect to the internet or other AWS services, but prevent the internet from initiating connections with the instances. Masked ip will be used to connect to internet
* **SECURITY GROUPS: -** A virtual firewalls that control the inbound and outbound traffic to and from our AWS resources, such as EC2 instances. Allowing us to define the rules that permits the traffic based on our requirements
* **INBOUND RULES: -** The rules which are defined by us based on our requirements for which traffic needs to allow.
* **OUTBOUND RULES: -**The rules which are defined by the AWS itself. By default, it will allow all the ports in outbound rule **except port 25, because it is a mailing port.**
* **NACL (Network Access Control List): -**It is as same as the security groups but NACL has the special quality of blocking the traffic as well as allow the traffic as per our requirements
* **NACL acts as the first level of defense and security groups acts as the second level of defense.**
* **NACL is with the VPC and security groups are with ec2 instances**
* **ROUTE 53: -** It is a domain name system **(DNS).** With which we can use the domain names like **www.example.com** instead of **numeric IP address**
* Allows us to register new domain names or transfer existing domain names into Route 53.
* Continuously monitors the health of our application endpoints.
* **S3 BUCKET (Simple Storage Service): -** It allows us to store and retrieve any amount of data from anywhere on the web. It is Scalable, highly available, Secure, Cost effective and High performance

**Advantages of S3**

1. Availability & Durability
2. Scalability
3. Security
4. Cost Effective
5. Performance

* **Availability & Durability: - It is 99.99999999999 % of reliability. Which means there is a chance of deleting one file among one billion files in 100 years of time period. We can consider it as 100% reliable**
* **Scalability: - It will store almost unlimited data in one single s3 bucket. But one single object should not exceed more than 5TB size. If we have more than 5TB object then we can upload as multipart uploads**
* **Security: - It will provide bucket policies, access control, and encryption settings. It will encrypt the data at rest as well as in transit Integrates with AWS Key Management Service (KMS) for encryption key management.**
* **Cost Effective: - We will use this as pay-as-you-go service**

****

* **S3 bucket** is globally accessible using http protocol
* The bucket name should be unique across all the aws accounts in the world
* If we want to upload the file which is already exists in s3 bucket will override it and uploads the new file
* **We can host a website using s3 as well, which is by disabling “Block public access” and by giving the bucket policy**
* **AWS CLI (COMMAND LINE INTERFACE): -** It allows us to manage AWS services and resources from the command line**.** It is used to automate the resources in AWS.
* **For automating the AWS resources, we have CLI, CFT, TERRAFORM AND CDK.**
* **We use CLI to get the quicker results. EX: - (“aws s3 ls”) it will list all the s3 buckets in the aws**
* **It acts a middleman between user and the aws**

**INSTALATION OF AWS CLI AND USAGE - API CALLS**

1. Search **aws cli** in internet, click on **documentation** and then click on **user guide**
2. Then click on **get started,** then select **install/update,** then select the operating system and **copy the code** and paste it in the terminal to install aws cli
3. **We must have python installed inorder to use aws cli**
4. Hit **aws --version** to know that aws cli is installed or not
5. We can run the aws cli commands in the terminal by using aws as the prefix word
6. We need to run **aws configure** command to connect to the particular aws account
7. It will ask us for the access key and the secret access key to connect
8. For these keys, click on the **account** in aws and click on **security credentials.** Then we’ll have **access keys** 🡪 **create access key 🡪 done.** Note the keys and store it somewhere for further use
9. [aws — AWS CLI 1.33.13 Command Reference (amazon.com)](https://docs.aws.amazon.com/cli/latest/reference/) 🡪 Reference for aws cli commands

* **AWS CFT (CLOUD FORMATION TEMPLATE): -** It is as same as the aws cli, but CFT will implements the principle of IAC (INFRASTRUCTURE AS CODE). Whereas cli don’t
* It will only support aws
* CFT support only **Yaml or json**
* It Is also useful for the drift detection. Which means if there is any changes happened it will notify us with what the changes has done
* We can implement the templates using stack

**CFT YAML FORMAT**

1. **Version of cft 🡪** which is standard one
2. **Description 🡪** about the cft template
3. **Metadata information 🡪** about author etc: -
4. **Parameters 🡪** pass variables during runtime
5. **Rules 🡪** Validate the parameters
6. **Mapping 🡪** Assigning the parameters to variables
7. **Conditions**
8. **Resources 🡪** It is a mandatory field in the creation of CFT template
9. **Outputs**

**USAGE OF AWS CFT TEMPLATES**

1. Search **aws cft** in internet and click on **template reference** then **resource and property reference** and then we can search for the aws service we want.

* **Terraform is for multi cloud whereas aws cft is only for aws**
* [AWS resource and property types reference - AWS CloudFormation (amazon.com)](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-template-resource-type-ref.html) 🡪 Reference for aws cft.
* **AWS CICD: -** It refers to the set of tools and services provided by AWS that enable Continuous Integration and Continuous Delivery (CI/CD) processes.

1. **AWS CODE COMMIT: -** It is similar to github, for hosting the code

* It should not be used in root user. Create an IAM user instead and add **AWSCODECOMMITPOWERUSER** policy to it
* For this we need to have git installed in our system (**windows/mac**).
* Use git clone <http://url> to clone the repository into the terminal
* Use git config –user.name inorder to commit the code from the terminal

**ADV: -**

1. Managed GIT
2. Scalability
3. Reliability

**DISADV: -**

1. It has very less features
2. It is only restricted to AWS
3. Less integrations with services outside AWS
4. **AWS CODE PIPELINE: -** It is similar to Jenkins for building a pipeline
5. **AWS CODE BUILD: -** It is similar to maven for building a code
6. **AWS CODE DEPLOY: -** It is similar to argocd or shell script for deploying a code

**Interview Questions and answers**

1. **You have been assigned to design a VPC architecture for a 2-tier application. The application needs to be highly available and scalable. How would you design the VPC architecture?**

**ANS: -** I would create 2 subnets private and public. In which the public subnet would contain the load balancers and be accessible from the internet. The private subnet would host the application server

I would distribute the subnets across multiple availability zones for high availability. And configure auto scaling groups for the application servers

1. **Your organization has a VPC with multiple subnets. You want to restrict outbound internet access for resources in one subnet, but allow outbound internet access for resources in another subnet. How would you achieve this?**

**ANS: -** To restrict outbound internet access for resources in one subnet, we can modify the route table associated with that subnet. In the route table, we can remove the default route (0.0.0.0/0) that points to an internet gateway.This would prevent resources in that subnet from accessing the internet. For the subnet where outbound internet access is required, we can keep the default route pointing to the internet gateway.

1. **You have a VPC with a public subnet and a private subnet. Instances in the private subnet need to access the internet for software updates. How would you allow internet access for instances in the private subnet?**

**ANS: -** To allow internet access for instances in the private subnet, we can use a NAT Gateway or a NAT instance. We would place the NAT Gateway/instance in the public subnet and configure the private subnet route table to send outbound traffic to the NAT Gateway/instance. This way, instances in the private subnet can access the internet through the NAT Gateway/instance.

1. **You have launched EC2 instances in your VPC, and you want them to communicate with each other using private IP addresses. What steps would you take to enable this communication?**

**ANS: -** By default, instances within the same VPC can communicate with each other using private IP addresses. To ensure this communication, we need to make sure that the instances are launched in the same VPC and are placed in the same subnet or subnets that are connected through a peering connection or a VPC peering link. Additionally, we should check the security groups associated with the instances to ensure that the necessary inbound and outbound rules are configured to allow communication between them.

1. **You want to implement strict network access control for your VPC resources. How would you achieve this?**

**ANS: -** To implement granular network access control for VPC resources, we can use Network Access Control Lists (ACLs). NACLs are stateless and operate at the subnet level. We can define inbound and outbound rules in the NACLs to allow or deny traffic based on source and destination IP addresses, ports, and protocols. By carefully configuring NACL rules, we can enforce fine-grained access control for traffic entering and leaving the subnets.

1. **Your organization requires an isolated environment within the VPC for running sensitive workloads. How would you set up this isolated environment?**

**ANS: -** To set up an isolated environment within the VPC, we can create a subnet with no internet gateway attached. This subnet, known as an "isolated subnet," will not have direct internet connectivity. We can place the sensitive workloads in this subnet, ensuring that they are protected from inbound and outbound internet traffic. However, if these workloads require outbound internet access, we can set up a NAT Gateway or NAT instance in a different subnet and configure the isolated subnet's route table to send outbound traffic through the NAT Gateway/instance.

1. **Your application needs to access AWS services, such as S3 securely within your VPC. How would you achieve this?**

**ANS: -** To securely access AWS services within the VPC, we can use VPC endpoints. VPC endpoints allow instances in the VPC to communicate with AWS services privately, without requiring internet gateways or NAT gateways. We can create VPC endpoints for specific AWS services, such as S3 and DynamoDB, and associate them with the VPC. This enables secure and efficient communication between the instances in the VPC and the AWS services.

1. **What is the difference between NACL and Security groups? Explain with a use case?**

**ANS: -** For example, I want to design a security architecture, I would use a combination of NACLs and security groups. At the subnet level, I would configure NACLs to enforce inbound and outbound traffic restrictions based on source and destination IP addresses, ports, and protocols. NACLs are stateless and can provide an additional layer of defense by filtering traffic at the subnet boundary. At the instance level, I would leverage security groups to control inbound and outbound traffic. Security groups are stateful and operate at the instance level. By carefully defining security group rules, I can allow or deny specific traffic to and from the instances based on the application's security requirements. By combining NACLs and security groups, I can achieve granular security controls at both the network and instance level, providing defense-in-depth for the sensitive application.

1. **What is the difference between IAM users, groups, roles and policies?**

**ANS: - IAM User**: An IAM user is an identity within AWS that represents an individual or application needing access to AWS resources. IAM users have permanent long-term credentials, such as a username and password, or access keys (Access Key ID and Secret Access Key). IAM users can be assigned directly to IAM policies or added to IAM groups for easier management of permissions.

**IAM Role**: An IAM role is similar to an IAM user but is not associated with a specific individual. Instead, it is assumed by entities such as IAM users, applications, or services to obtain temporary security credentials. IAM roles are useful when you want to grant permissions to entities that are external to your AWS account or when you want to delegate access to AWS resources across accounts. IAM roles have policies attached to them that define the permissions granted when the role is assumed.

**IAM Group**: An IAM group is a collection of IAM users. By organizing IAM users into groups, you can manage permissions collectively. IAM groups make it easier to assign permissions to multiple users simultaneously. Users within an IAM group inherit the permissions assigned to that group. For example, you can create a "Developers" group and assign appropriate policies to grant permissions required for developers across your organization.

**IAM Policy**: An IAM policy is a document that defines permissions and access controls in AWS. IAM policies can be attached to IAM users, IAM roles, and IAM groups to define what actions can be performed on which AWS resources. IAM policies use JSON (JavaScript Object Notation) syntax to specify the permissions and can be created and managed independently of the users, roles, or groups. IAM policies consist of statements that include the actions allowed or denied, the resources on which the actions can be performed, and any additional conditions.

1. **You have a private subnet in your VPC that contains a number of instances that should not have direct internet access. However, you still need to be able to securely access these instances for administrative purposes. How would you set up a bastion host to facilitate this access?**

**ANS: -** To securely access the instances in the private subnet, you can set up a bastion host (also known as a jump host or jump box). The bastion host acts as a secure entry point to your private subnet. Here's how you can set up a bastion host: Create a new EC2 instance in a public subnet, which will serve as the bastion host. Ensure that this instance has a public IP address or is associated with an Elastic IP address for persistent access. Configure the security group for the bastion host to allow inbound SSH (or RDP for Windows) traffic from your IP address or a restricted range of trusted IP addresses. This limits access to the bastion host to authorized administrators only. Place the instances in the private subnet and configure their security groups to allow inbound SSH (or RDP) traffic from the bastion host security group. SSH (or RDP) into the bastion host using your private key or password. From the bastion host, you can then SSH (or RDP) into the instances in the private subnet using their private IP addresses.