**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.

**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

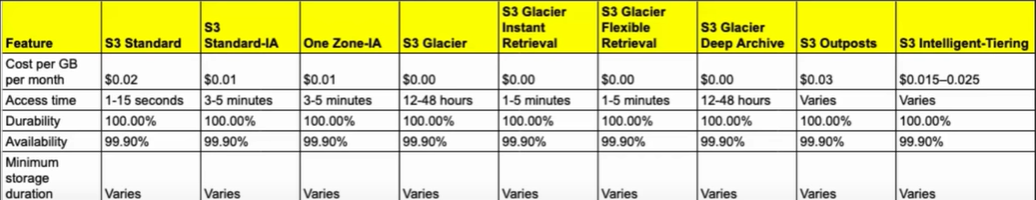
1. Click on **LAUNCH INSTANCE**
2. Give the **NAME**
3. Select **OPERATING SYSTEM**
4. Select **INSTANCE TYPE**
5. Give the **KEY-PAIR VALUE (Combination of public and the private key)**
6. Select **RSA as KEY PAIR TYPE and give the KEY PAIR NAME**
7. Then select **.PPK{Only for PUTTY} OR .PEM{For OpenSSH(**ex: - 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 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**

****

* **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

* 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**
5. **AWS CODE BUILD**
6. **AWS CODE DEPLOY**