UNIT 6 - WEB APPLICATION DEVELOPMENT

CLOUD

- Cloud Computing = the delivery of computer services over the internet. Eg. Google drive, Microsoft OneDrive, AWS, Netflix (runs on AWS).
- Benefits of CC = cost effective, security, low maintainance, pay for what you use, automatic updates, scalability, accessibility.

AWS Cloud

- Amazon Web services is comprehensive cloud computing platform that provides on-demand IT resources over the internet with pay-as-you-go pricing model.
- No need to invest on physical hardware.
- Features = Pay-as-you-go, Scalability & Flexibility, Global Reach, Managed Infrastructure Speed and Agility (provisioning of resources), High Availability, Disaster Recovery.
- AWS services = Compute (Amazon EC2, AWS Lambda)
  Storage (Amazon S3, Amazon EBS, Amazon EFS)
  Databases (Amazon RDS, Amazon Dynamo DB)
  Networking (Amazon VPC, Elastic Load Balancer)
  Security (AWS Identity and Access Management)

There are 200 fully featured services

EC2 (Elastic Compute Cloud)

cc unit 3 notes for reference

- Steps to deploy ~~Website~~ website on EC2:-

Login to AWS Console. 1
Launch EC2 instance. 2
Configure instance & security groups. 3
Connect via PUTTY/SSH. 4
5 Install web server.
6 Upload website files (HTML/CSS).
7 Access via public IP in browser.

- PUTTY is a free SSH and telnet client for Windows used to securely connect to remote servers (like EC2)

Steps to connect EC2 via PUTTY:-

1. Download & install PUTTY & PUTTYgen.
2. Convert .pem file to .ppk using PUTTYgen.
3. Open PUTTY → Enter EC2 public IP
4. Load .ppk under SSH > Auth
5. Click Open → Login as e2c-user/ubuntu.

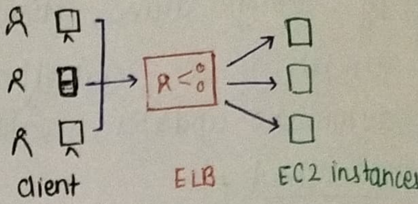
- Features = Scalable Computing, Reliable, Fully Controlled, Easy to start, Designed for AWS, Secure, flexible Tools, Inexpensive.

AWS Elastic Load Balancer (ELB)

- ELB automatically distributes incoming application traffic across multiple targets, such as EC2 instances, containers and IP addresses, within one or more availability zones.
- ELB work : ELB accepts all the traffic from the client and then routes this traffic to the target that the user wants.
If the load balancer finds an unhealthy target, then it will stop redirecting

it users there and it will move with other healthy targets until target is declared healthy.

- Features = Automatic Traffic distribution, Healthy Monitoring, Scalability, Security, High Availability, Sticky Sessions, Monitoring & Logging.



- Types of ELB :

1) Application Load Balancer (ALB) -

Layer: Works at Layer 7 (Appln Layer) of OSI model.

Protocols: HTTP, HTTPs, gRPC

Features: supports advanced request routing based on content/URL path/hostname. Ideal for modern web applications needing flexible routing and high-level features.

2) Network Load Balancer (NLB) -

Layer: Works at Layer 4 (Transport Layer) of OSI model.

Features: Designed for TCP, TLS and UDP traffic.
Handles millions of requests per second with ultra-low latency.
High performance and scalable.
Suitable for real-time appln, gaming, IoT or finance appln.

Eg. Use NLB if you're hosting TCP-based service like a custom protocol/VPN server.

3) Gateway Load Balancer (GLB) -

Layer: Works at layer 3/4 (Network + Transport layer)

features: Used to deploy, scale and manage third party virtual appliances.
Combines load balancing with traffic ~~mon~~ mirroring.
Routes all VPC traffic through appliances transparently.

Eg. Ideal for use cases where you want to inspect & monitor traffic with security appliances.

4) Classic Load Balancer (CLB) -

Layer: Works at layer 4 and 7 but with limited features.

Features: Used for EC2 - Classic (older network model).
Basic load balancing for HTTP, HTTPS, TCP traffic.
Lacks advanced routing, monitoring and flexibility of NLB/ALB.

Eg. used for legacy EC2

How ELB exactly works :-

1) Traffic Analysis - ELB receives incoming client traffic via listners configured with specific protocols & ports.
2) Health Checks - ELB monitors the health of registration/registered targets and only routes traffic to healthy ones.
3) Traffic Distribution - ELB distributes requests across available, healthy targets in enabled availability zones, ensuring high availability & fault tolerance.
4) Scaling - ELB automatically adjusts its capacity in response to changes in incoming traffic
5) Session Management - ELB can maintain session stickiness to ensure that requests from the same client are constantly & consistently routed to same target

AWS VPC and Components of VPC

- AWS Virtual Private cloud (VPC) is logically isolated virtual network within the AWS Cloud. It allows you to launch AWS resources, such as EC2 instances or database in a private, customizable network environment.
(It's like having your own private data center in the cloud.
- Purpose : Customize network configuration.
Control IP addressing subnets, routing tables and gateways.
Secure communication between resources.
Supports public, private and hybrid cloud models.

- Components of VPC
1) Subnets : These are smaller networks within your VPC.
They divide you VPC's IP address range into segments.
Each subnet lives in a specific Availability Zone.
Public Subnet : Connected to the internet (via an Internet Gateway)
Private Subnet : Not directly connected to the internet (no direct route to internet Gateway)
2) Route Table : These contain rules that decide where network traffic goes.
Each subnet is linked to a route table.
Routes can send traffic to the internet, other subnets or private networks.
3) Security Groups : Acts as virtual firewalls for EC2 instances
Control inbound & outbound traffic at the instance level using rules for ports, protocols, and IP addresses.
4) Network Access Control Lists (ACLs) : Another layer of security, controlling traffic at subnet level.
You can allow/deny traffic based on rules for IP add^r, ports & protocols.
NACLs are stateless, so rules must be set for both inbound & outbound traffic
5) CIDR block : Its a way to define a group of IP addresses.
Written like : 10.0.0.0/16. '/16' tells how big the group is.
Bigger no. = fewer IP add^r , smaller no. = more IP addresses.
When you create VPC, you give CIDR block to decide how many IP addresses it can use.
6) Gateway : Its like a door betn VPC and other networks.
Internet Gateway - This lets VPC talk to the internet & lets internet talk to public servers
NAT Gateway - Lets private servers connect to the internet to get updates but keeps the internet from connecting back to them.

AWS Storage

AWS offers many storage services. Some of them are:

1) S3 (Simple Storage service)
2) Amazon EBS (Elastic Block Store)
3) Amazon EFS (Elastic file system)

} Refer CC unit 3 notes

4) Amazon FSx : Fully managed file storage built for specific needs (like Windows or high performance workloads). Supports popular file systems.
5) AWS Storage Gateway: Contacts your on-premises data center to AWS cloud storage. Lets you use AWS storage as if its part of your local network. Good for backup, archieving and hybrid cloud setups.
6) Amazon Glacier (now part of S3 Glacier) : Very low cost storage for long term archiving. Used for data you rarely need to access, like backups/records.

Create a bucket in S3 :

1) Login to AWS management console.
2) Navigate to S3 and click "Create bucket".
3) Enter a unique bucket name & select a region.
4) Configure options (versioning, encryption, permissions) as needed.
5) Create the bucket and start uploading data.

Deploy website or Web Application on AWS.

Create a bucket → Enable website hosting → Upload files → Make public → Use the website URL to access your site.

Launch an Application with AWS Elastic Beanstalk.

- AWS Elastic Beanstalk is ~~is~~ a Platform-as-a-Service (PaaS) that simplifies deploying & managing applications in AWS Cloud.
- It is a managed service that is easy to deploy and run web applications & services in the AWS cloud.
- We need to first upload our application code, and Elastic Beanstalk automatically takes care of all the details. (setting up servers, load balancing, auto scaling, etc)

- Advantages :
1) Easy Deployment
2) Automatic management
3) Support multiple languages (JAVA, .NET, Node.js, PHP, Py, etc)
4) Customizable
5) Pay as you go
6) Scalability
7) ~~Custom~~ Monitoring & Updates.
8) multiple deployment options.

- Steps to deploy using Elastic Beanstalk :
1) Prepare your app in zip file
2) Go to Elastic Beanstalk console & create application
3) Choose platform (eg Node.js, Python or Tomcat)
4) Upload you code (.zip file)
5) Configure environment (EC2 type instance)
6) Launch Environment
7) Access Application.