Day

100 GiB = 100 × 1.073741824 GB

= 107.37 GB

...s Terminate

...instance ...ice permanently

...dow deleted along with

...memory EBS.

remains intact

can be restart Cannot be restarted

"AMAZON - SNS"

"Amazon Simple Notification Service (SNS) is a fully managed messaging service used to send notifications or msgs from one system to multiple other systems (or users) in a fan-out or publsub model.

- Publisher-Subscriber (Publ/Sub) Model
  - ↳ Publisher: A system or application that sends a message
  - ↳ Topic: A logical access point & communication channel (like a "mailing list")
  - ↳ Subscriber: A sys, service, or user that recvs messages.

Date _____ Day _____

Flow:

① A publisher sends a message to an SNS topic

② The SNS topic immediately delivers that message to all its subscribers (which can be different endpoints)

• SNS Component

① Topic

↳ Act as a "channel" for messages

↳ Publishers sends messages to a topic

↳ Subscriber Subscribe to that topic

② Subscription

↳ Defines where & how messages will be delivered

↳ Example Protocols

→ HTTP / HTTPS

→ Email / Email JSON

→ SMS (Text msgs)

→ AWS Lambda func

→ AWS SQS queue

③ Message

↳ The actual content being published

Date _____ Day _____

→ Makes it easy to send push notifications to iOS Android

→ With SNS, it's easy to send updates, promos, or news to individual users, a subset of users or all of users, using a single message.

→ SNS works with SQS to provide a powerful messaging solution for building cloud applications that are fault tolerant and easy to scale.

→ Benefits

↳ Delivers messages in real-time to multiple subscribers

↳ No need to manage messaging servers or infrastructure

↳ Can handle millions of msgs / second & deliver them to thousands of subscribers.

↳ SNS topic owners can keep sensitive data secure by setting topic policies that restrict who can publish & subscribe to a topic.

↳ Reliably deliver msg with durability

↳ Inexpensive — pay-as-you-go pricing with no upfront fees or commitments

Date _____ Day _____

"AMAZON - ELB"

'AWS Elastic Load Balancer distributes incomming applicatio or network traffic accross multiple targets, such as Amazon EC2 Instances, containers, IP Adresses, in multiple AZs.

→ ELB scales your loadbalancer as traffic to application changes overtime

▷ Features

→ High Availability — as LB automatically distributes traffic accross multiple targets, in a single/multiple AZs.

→ Health Checks — Stops sending traffic to unhealthy targets

→ Provides integrated SSL/TLS certification

→ Can divide traffic on all network layers.

Application Layer → HTTP, HTTPS

Transport Layer → TCP, UDP

→ Also provide AWS cloud watch for monitoring

Date _____ Day _____

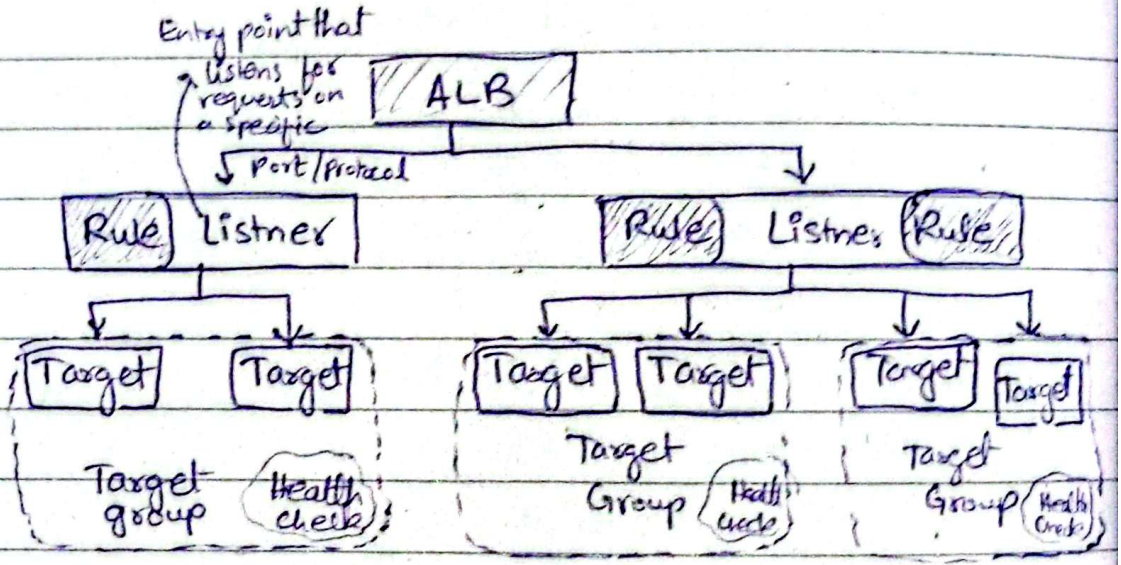
▷ Types of ELB

① Classic Load Balancers (Old Generation)
- ↳ Makes routing decisions at transport or application layer
- ↳ Requires a fixed relationship btw the LB port & container instance port
- ↳ Provide SSL
- ↳ Supports the ability to stick user sessions to a specific instance using cookies.
- ↳ Same no. of instances are required in each AZs for evenly traffic distribution
  - ↳ enable cross-zone load balancing

② Application Load Balancer (ALB)
- ↳ Functions at Application layer (7th layer)
- ↳ It can route traffic intelligently based on
  - URL Path
    - ↳ /api/* → API Server
    - ↳ /images/* → Image server
  - Hostname (Host based routing)
    - ↳ api.example.com
    - ↳ app.example.com
  - HTTP headers, query strings or request methods

Date Day

- You register EC2 instances, containers into target groups
  - ↳ Each target group defines
    - ↳ The port (eg: 80, 8080)
    - ↳ Health Check configurations
- ALB Forwards traffic to targets based on listner rule.
- Provides SSL
- Rule determines which Target Group to route it to



③ Network Load Balancer

↳ Functions at the 4th Layer of OSI model.

↳ Design to handle traffic as it grows and can load balance millions of req/s/sec

↳ Provides extremely low latencies for latency sensitive app.

↳ Each NLB gets a fixed IP per AZ

Date _____ Day _____

↳ Designed for fault tolerance – failures in one zone don't affect others

↳ Usecases

↳ Load Balancing database traffic

↳ " " gamming servers, IoT, VoIP app

↳ Non-http protocols (SMTP, FTP etc)

↳ Routing is based on IP-Add & Port

④ Gateway Loadbalancer

↳ Functions at the 3rd Layer of OSI model

↳ Helps you deploy, scale, and manage 3rd party network security appliances such as

↳ Firewalls

↳ Intrusion Detection System

↳ " Prevention "

↳ Traffic Monitoring tools.

↳ It act as a single entry & exit point for all traffic in & out of the network

↳ GWLB uses GWLB endpoints to securely exchange traffic accross VPC boundaries

Date _____ Day _____

"AMAZON - ASG"

"Auto Scaling Group (ASG) in AWS automatically manages a fleet of EC2 instances"

→ It ensures the right no. of instances
→ Launches new instances when demand incr.
→ Terminates unnecessary ones & drops
→ Replaces unhealthy instances automatically

→ Maintains optimal app performance & availability, even when traffic is periodic, unpredictable or continuously changing.

→ Improve Fault Tolerance — Detects an unhealthy instance, terminates it and replaces it

→ Increase Availability — ensures that the application always has the right amount of compute capacity

→ Scales dynamically based on AWS Cloud Watch metrics
→ Periodically performs health checks
→ Balances capacity across AZs

Date _____ Day _____

→ Uses a template image to launch a new instance

→ ASG provides lifecycle hooks that allows us to perform so custom actions before instance launch & termination.
↳ like backing up of data.

→ Scale-out ⇒ Increase instances

→ Scale-in ⇒ Decrease instances

▸ Types Of Scaling Policies

• Target Tracking Scaling → AWS automatically decides the no. of instances to add/remove
↳ Increase or decrease the current capacity of the group based on a target value for a specific metric
Ex: "Keep avg CPU utilization at 50%"
↳ If CPU > 50% → scale out
↳ If CPU < 50% → scale in

• Step Scaling Policy
↳ This is a rule-based policy that scales in steps depending on how far a metric deviates from the threshold.
Ex: If CPU > 70% → Add 1 instance
If CPU > 85% → Add 2 instance

Date _____ Day _____

↳ Best when required fine-grained control and different scalling responses for diff load levels.

- Simple Scaling
  - ↳ Incr. or decr. the current capacity of the group based on single scaling adjustment
  - ↳ It uses a single Cloud watch alarm & performs on scaling action at a time
  - ↳ Ex: If CPU > 70% ⇒ Add 1 inst.
    After scaling, it waits for a cooldown period before scaling again
  - ↳ Best for very simple, low-traffic app.

- Scheduled Scaling
  - ↳ This is time-based — define when scaling should happen
  - ↳ Ex: Every ~~wednesday~~ week at 9PM → incr. to 5 instances
  - ↳ Best for predictable workloads

- Predictive Scaling
  - ↳ Use ML to forecast future demand based on historical patterns & scales ahead of time
  - ↳ Best for apps with predictable traffic patterns