AWS compute and network services

Compute services:
EC2 instance
Autoscaling groups
Elastic load balancers
Elastic beanstalk
Cloud functions

Network services:
VPC
Route 53
Cloud front
APi gateway
Direct Connect
VPC peering
VPN

Shopping application:
On-prem:
Purchase more servers to handle raise in traffic - high latency
Resources will be idle after festive season - committed to resources

Cloud:
Spin up resources in few minutes
No commitment
Autoscaling group
Save cost and low latency

EC2 instance: IaaS
Virtual server
CPU, RAM, storage, OS, n/w

1. AMI -> amazon machine image -> region specific

Read only file system -> details of OS
Custom AMIs

2. Instance types:

General purpose -> balanced resources
- t2.micro, m5.large

Compute optimized -> high performance processors
- C6g.large, c5.large

Memory optimized -> memory intensive workloads -> bigdata
- R5.large, x1.16xlarge

Accelerated computing -> graphical workloads, GPU
- P3.2xlarge, p2.xlarge

Storage optimised -> high speed seq read and write access, data warehousing, NoSql
- I3.large, d2.xlarge

3. Instance details

   Purchase models:

   On demand instances: 100$/2 days/24x7
   - No commitment, initial investment -> zero
   - Users have complete control on instance
   - Short term workloads, irregular workloads -> workload should not be interrupted

   Spot instances: 25$/2 days/24x7
   - 75% discount when compared to on-demand instance
   - Short term workloads -> workloads can be interrupted, batch processing
   - CSP can ask you to return the instance at any time
   - No commitment, initial investment->zero

   Reserved instances: 10$/2 days/24x7
   - 90% discount when compared to on-demand instance
   - You will have commitment to instance -> 1 year or 3 years
   - Long term workloads

   User data -> application data

4. Added storage
5. Added tags
6. Security group

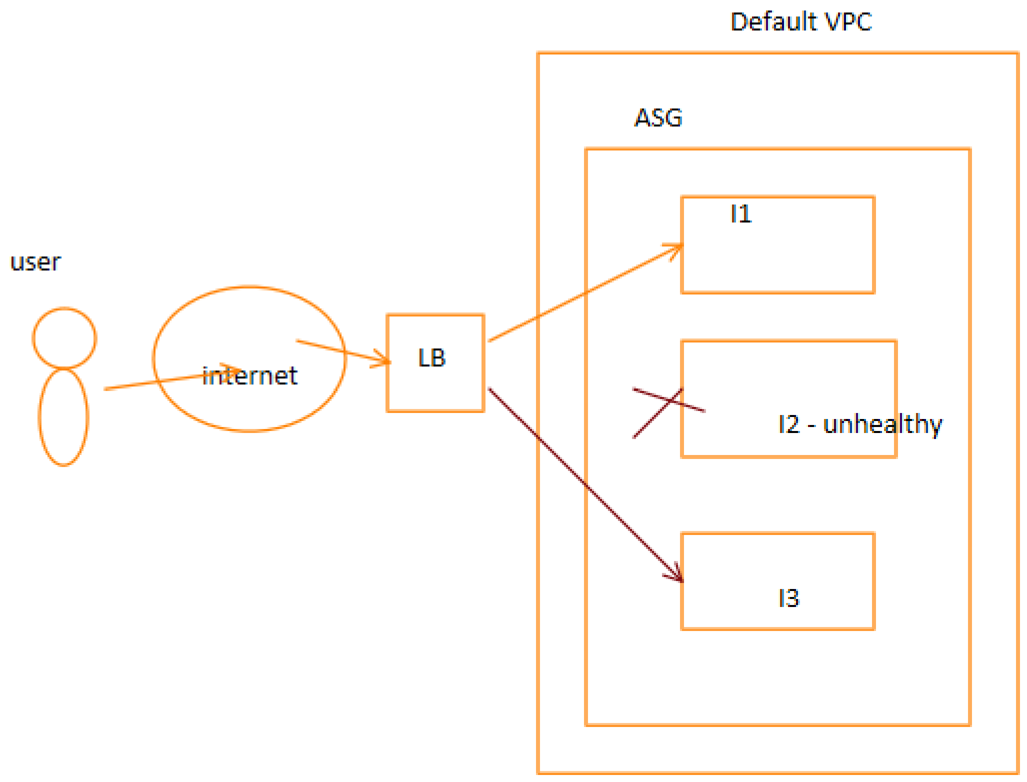
   Restriction on type of traffic that should be allowed to access your application

ASG:
Desired capacity - number of instances that should be created and running on launching the ASG(4)
Minimum capacity - minimum number of instance that should be running everytime(2)
Maximum capacity - maximum number of instance that should be running (8)

Lambda:
Photo gallery application.
Whenever an user uploads an image, you want the application to convert that image into a thumbnail and store it in a storage.

Event: converting the image to thumbnail
Trigger: user upload image

Elastic load balancer:



Health checks:

LB will check whether the endpoint is healthy or not
Redirect the req only to healthy ones.

Types:
Classic load balancer(previous gen)

Application load balancer:
Operates at layer 7 in OSI(OSI model) -> application layer
Support http and https traffic
Web application or containerized workloads

Network load balancer
Operates at layer 4(transport layer)
Support TCP, UDP, TLS traffic
Handle millions of req/sec -> ultra low latency
Workloads that require high network performance

Gateway load balancer
Operates at layer3(network layer)
Easy to deploy and manage firewalls and intrusion detection system

User upload the image -> stored in some storage service -> s1 -> internet facing -> s1 and internet
Trigger -> whenever an image gets stored in S1 the event gets triggered -> internal -> s1 and lambda func
Event -> convert that image into thumbnail -> store it in s2

Healthy threshold -> specify after how many successful health checks you should label this endpoint is heathy
Unhealthy threshold -> specify after how many unsuccessful health checks you should label this endpoint is unhealthy
Timeout -> maximum duration that can be taken by LB for performing health check
Interval ->between health checks

Elastic Beanstalk:

-- PAAS
-- High productivity
-- Develop /deploy -- applns

Features :

Security
Developer productivity
Monitor -- Health check

Working:

Create an appln --> Platform selection --> Version selection --> Upload ur code / Sample appln --> Create ur appln

Code updation : Yes

Platform Version updation : Yes
Platform updation : No

Beanstalk Concepts :

Appln :
Appln version
Env
Env Tier
Env config
Saved config
Platform
****

Lambda :

Serverless service

Focus only on business part

-- FAAS

-- Event driven approach

-- supports all pgmg lang

-- Built fault tolerance

-- container

https://www.qwiklabs.com/focuses/16506?catalog_rank=%7B%22rank%22%3A1%2C%22num_filters%22%3A0%2C%22has_search%22%3Atrue%7D&parent=catalog&search_id=10483415

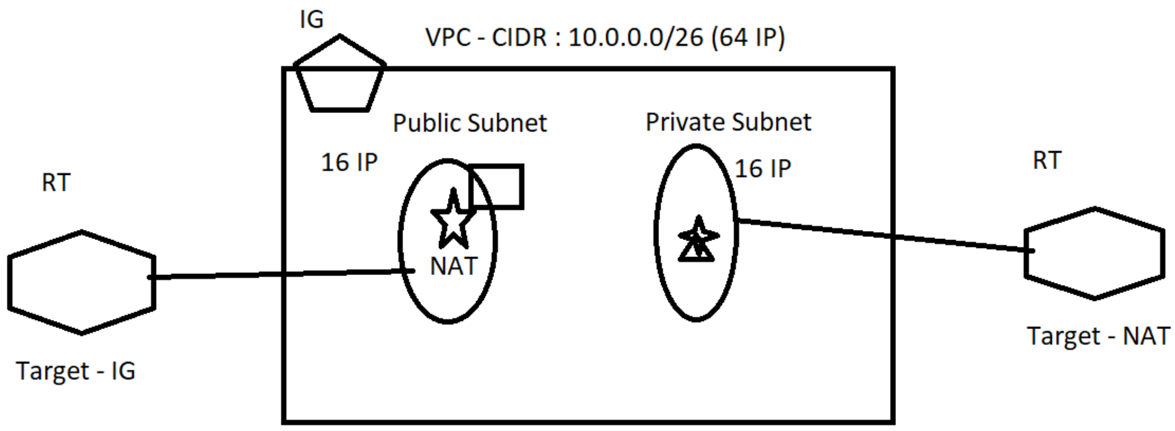
Working :

Check weather updates --> FAAS --> DB /Server --> Response to end user

****

VPC:

-- Secured
-- Private network
-- Customizable ntw



Components :

CIDR
Subnets
Internet gateway
Route table
SG
NACL
NAT gateway

CIDR :

-- List of IP ranges
-- Syntax: start_IP_address/Netmasking
-- Ntwg --0 to 32
-- AWS 16 to 28

CIDR : 10.0.0.0/28

32 - 28 = 4

2 power 4 = 16 IP address

10.0.0.0/28 = 10.0.0.0 till 10.0.0.15

CIDR : 10.0.0.0/26

32 - 26 = 6
2 power 6 = 64 IP address
10.0.0.0 till 10.0.0.63

Subnets:

-- Subsets of ur VPC

-- Two types

-- Public -- Access to internet via IG -- 10.0.0.0 till 10.0.0.15

-- Private -- No access -- 10.0.0.16 till 10.0.0.31

Route table :

-- Destination

-- Target

IG:

-- To enable access to internet

NAT gateway

-- Provide internet access to pvt subnet

-- create it in Public subnet

-- Specify NAT as target in pvt subnet

SG :

-- Security group

-- At resource level

-- Statefull --> independent on inbound rules

-- Allow rules

NACL :

-- Ntw access control list
-- At Subnet level
-- Stateless --> dependent on inbound rules
-- Allow and deny rules
Allow
IP a
IP b
Deny
IP c

Resource Cleanup:

Delete RT :

-- Removing the subnet associations

-- Remove the target

-- delete RT

Delete IG :

-- Detach from VPC

-- Delete IG

Delete NAT :

-- yes

Delete Subnets :

-- yes

Delete VPC:

-- Yes

****

Route 53 :

-- DNS service

-- Domain name into IP address

-- Route policies

Simple RP

Latency based RP

Failover RP

Weighted RP

Geolocation RP