**Welcome to day3 of Compute and Network Services!!**

**Lex Link:**

[**https://lex.infosysapps.com/en/app/toc/lex\_19685426003365850000/overview**](https://lex.infosysapps.com/en/app/toc/lex_19685426003365850000/overview)

**Attendance Link:**

[**http://iscls4apps/ITMS/Learning/Index**](http://iscls4apps/ITMS/Learning/Index)

**Auto scaling groups:**

* high availability of your application
* add/remove instances --> the policies defined by user
* if CPU utilization > 90% --> add more instances
* when an instance an unhealthy --> replaced
* optimize costs

**Launch template**

* AMI, Instance type, key pair, security groups
* Contain multiple versions

**Capacity:**

* Desired capacity - 4 - initial capacity
* Min capacity - 1
* Max capacity - 10

**Options:**

* Create ASG --> Attach to load balancer --> Target group --> target-787923
* ASG --> launched 2 instances --> added to target group of ELB

**VPC:**

* Virtual private cloud
* on prem --> firewall configurations --> private network
* logically isolated segment -- resources
* more control
* can define set of firewall rules -- more secure
* regional service
* spread across the region

**Default VPC:**

* every region --> one default VPC -- created by AWS
* Default VPC -- public subnets

**CIDR range:**

create VPC --> range of IP addresses

IPv4 - 32 bits -- 8 bits each \* 4

IP address/net masking value

10.0.0.0/16 -- first 16 bits remain same

10.0.0.1 -- 10.0.255.254

65,536 addresses

10.0.0.0/24 – 256 IP addresses

**components of VPC:**

Subnets:

* divide your VPC into smaller sections
* belongs to single AZ
* specify the CIDR range -- can i specify any random range???

1.it must be subset of VPC CIDR range

2.must not be overlapping with other subnets

Public subnet -- resources can be accessed from the internet

Private subnet -- resources cannot be accessed from internet

create a new subnet --> by default private subnet

/16 to /28 – allowed CIDR block

More on VPC: <https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Subnets.html>

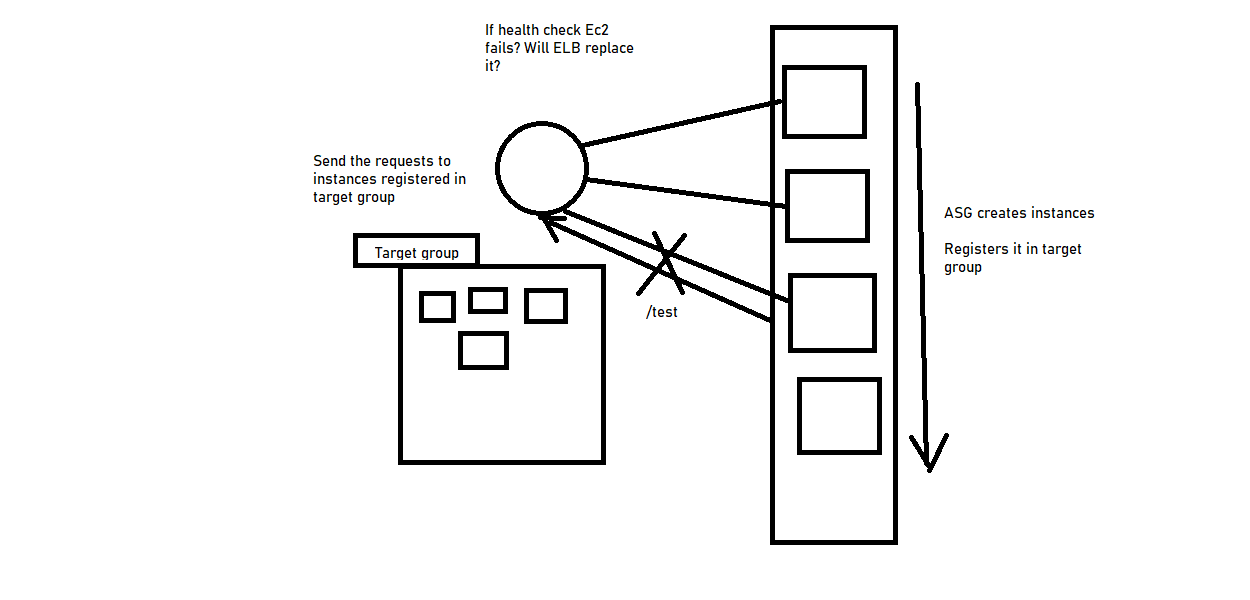
**Route tables:**

* set of rules to guide your traffic
* VPC creation --> main route table
* any new subnet which is created --> will be implicitly associated with main route table
* one subnet --> can be associated with only 1 route table
* 1 route table --> multiple subnets

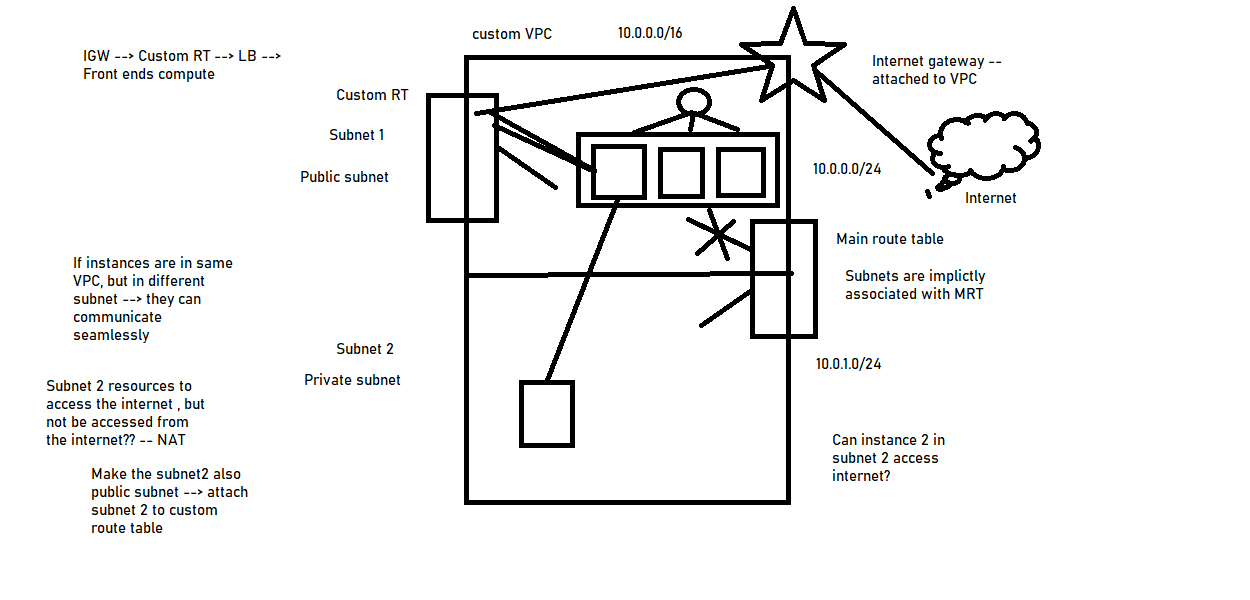
**Making subnet 1 as public subnet:**

* Step1: enabling auto assign public IP option
* Internet gateway: to give an internet access, create and attach to VPC
* Step2: create a custom route table, attach it to subnet1, modify route in route table to send traffic to IGW

ELB + ASG:



VPC:



**Additional notes regarding beanstalk and lambda:**

Beanstalk:

* Want to focus more on development
* Increases developer productivity as it frees developers from managing and configuring servers, databases, load balancers, networks etc.
* Do not want to worry about the underlying infrastructure
* Beanstalk is a solution for deploying your applications easily
* Platform as a service
* In AWS beanstalk developers can just upload their code and beanstalk will take care of load balancing, auto scaling and monitoring

Environment --- collection of AWS resources running an application version

No additional charge for beanstalk, will be charged only for underlying resources

Lambda:

**Serverless computing:**

<https://aws.amazon.com/serverless/>

<https://aws.amazon.com/lambda/pricing/>

* No need to manage any servers, we don’t provision them and can’t manage them as well

Examples of Serverless in AWS:

* AWS lambda
* DynamoDB
* Cognito
* API Gateway
* S3, SNS, SQS
* Kinesis Data firehose
* Aurora serverless
* Step Functions

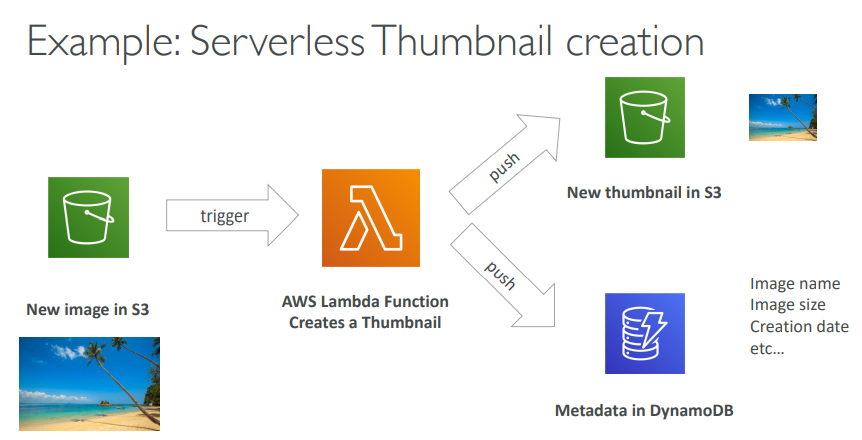
EC2:

* Virtual servers in the cloud, Infrastructure as a service
* We have to provision them
* Continuously running
* For scaling we need to use ASG

AWS lambda:

* No servers to manage
* We just deploy functions as a code ---- and serverless --- so called as Function as a service
* Cannot manage the configurations of underlying servers
* Usually, short executions
* Run on demand --- only when invoked
* Scaling is automated

Use case of lambda:



Additional links:

<https://docs.aws.amazon.com/autoscaling/ec2/userguide/AutoScalingGroup.html>

<https://docs.aws.amazon.com/autoscaling/ec2/userguide/create-launch-template.html>

<https://aws.amazon.com/blogs/aws/new-aws-elastic-load-balancing-inside-of-a-virtual-private-cloud/>