**AWS Developer: The Big Picture**

Auto scaling : Use EC2 for scaling AMIs, EB for scaling application

Security group : ip based communication rules for a single or group of EC2 instances.

SG scenarios : control who can SSH into EC2 instance, allow access b/w EC2 instances, allow access to DB, allow HTTP request

EC2 Instance pricing types:

* On demand inst - cheap
* Reserved inst - cheaper
* Spot inst – cheapest

**S3 Buckets can** : trigger events when objects are added/modified/deleted, preserver older versions off objects, replicate objects across regions. S3 can be access via URL



How to solve latency in S3 : use cloudfront to cache the content

S3 Pricing: Amount of data stores, number of req, amount of data transferred

**Route 53 :** The solution for your DNS needs

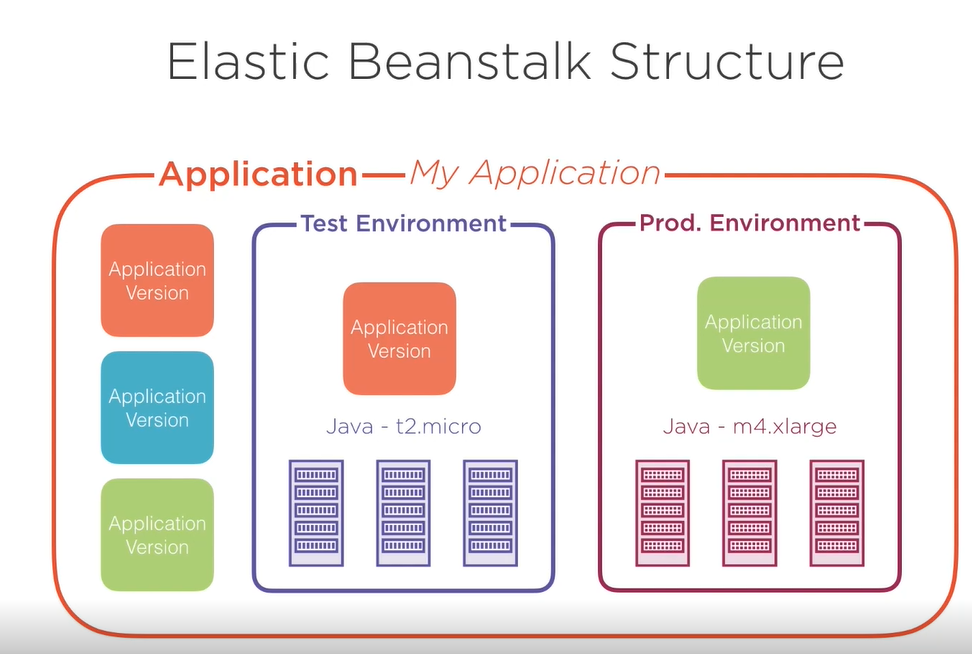
DNS (Domain Name system) : system that translates human-readable URLs to ip address.

EC2 inst can be config with public ip add but certain amazon resources like S3 or LB are a little more complicated and ddont have static visible ip addresses.

Route 53 allows you to set up url resolution to aws resources directly by bypassing any need for you to see an ip. Route 53 Lets users interact with services in AWS

**Elastic Beanstalk**

Deploying your app to EC2 : Manual config, manual code deploy, restricted command line interface, scale with AMIs, manual monitoring

Deploying your app to EB : Easy deploy with various tools, set it and forget it configuration, aggregated monitoring and logging  


* App Versions are stores in S3.
* There's also a limit to the number of application versions that one app can have and the default is 1000.

**Lambda**

The solution for your code execution needs.

Lambda provides function code execution as a service sometimes this is called function as a service or serverless.

Features : executes code, no server management required, only pay when the code is running

Lambda bills you by the number of req and per 100ms that your code was executing. Means you no longer pay for idling servers.

**Lambda structure:**

* fundamental structure in lambda is function. A function is a bundle of code with a specific execution entry point.
* Configured with a platform such as node, python, go or other languages and invocation targets such as API gateway, cloudwatch or cloudfront.
* Additional configuration : execution timeout, mem requirements and IAM role that lambda should execute

