Take Home Exam

Question 1:

Difference between Serverless Framework and Zappa

Serverless framework and Zappa both are widely used frameworks for deploying your application using AWS Lambda. There is a difference between serverless and zappa. Serverless provides develop, test and deploy in a single environment to any cloud provider such as AWS, Google Cloud Functions, Azure Functions, and IBM OpenWhisk, Kubeless, spotnst and webtasks. It supports Node.js, Python, java, C# etc. Zappa is a framework for running serverless python web applications and uses the AWS Lambda and API Gateway services for deployment. Zappa is not a completely serverless there is a machine returning HTTP response to the client. But server's whole lifespan exists within the lifecycle of a given HTTP request. It is based on frameworks like Django and Flask.

Zappa is simply turning a WSGI (Web Server Gateway Interface) applications into a serverless service. It supports various features like custom domain name, keeping lambda functions warm automatically, reading variables from S3, managing API key to secure endpoints. Serverless Frameworks has features like nanoservices in which it is supporting different lambda functions with a different code-base for each endpoint which is relevant for large applications.

Question 2:

Loose Coupling and High Cohesion

- **Loose Coupling:**

In general, coupling is joining two things together for example, links in chain. But in application development, coupling refers to the degree to which application components or modules or services depend upon each other. This defines whether these services or components operate in a tightly or loosely coupled relationships. The difference is that loosely coupled components can operate independently from each other but tightly coupled cannot. So, in loosely coupled system, each of its components has or make use of, little or nothing from other components. The major benefit of loose coupling is that it reduces the risk that when a change made within one service will not create any unwanted changes within other services. It also helps to separate problems that makes testing simple, maintenance and debugging.

- **High Cohesion:**
  Cohesion refers to the degree to which the elements of a service belong to each other. There are two types of cohesion low and high cohesion. In low cohesion one service does lot of jobs that don’t have much in common. In high cohesion, one service does a well-defined task. In high cohesion, tasks that are related to each other should be contained in one service so that we can change one service without updating other services. The advantages of using high cohesion are reduced module complexity, increased system maintainability and reusability. Because changes made in one service will require few or no changes in other services.