Capstone Synopsis <PGPCCNOV18> <Team 4>

Capstone Interim Report is a technical document, explaining "Solution Architecture" of the project. This document should communicate end-to-end IT solution for the project. Provide all views of the solution required for design, build, testing & implementation. The write up must adhere to the guidelines and should include the following.

Version History:

Version	***Date***	***Author***	***Key Changes***
1.0	*12 apr 2019*	*Rummy Maini*	*Solution arch diagram / 12 factor coverage*
1.1	*19-Apr-2019*	*Sridhar Sampath*	*Added the Arch diagram & 12 Factor apps*

Document References:

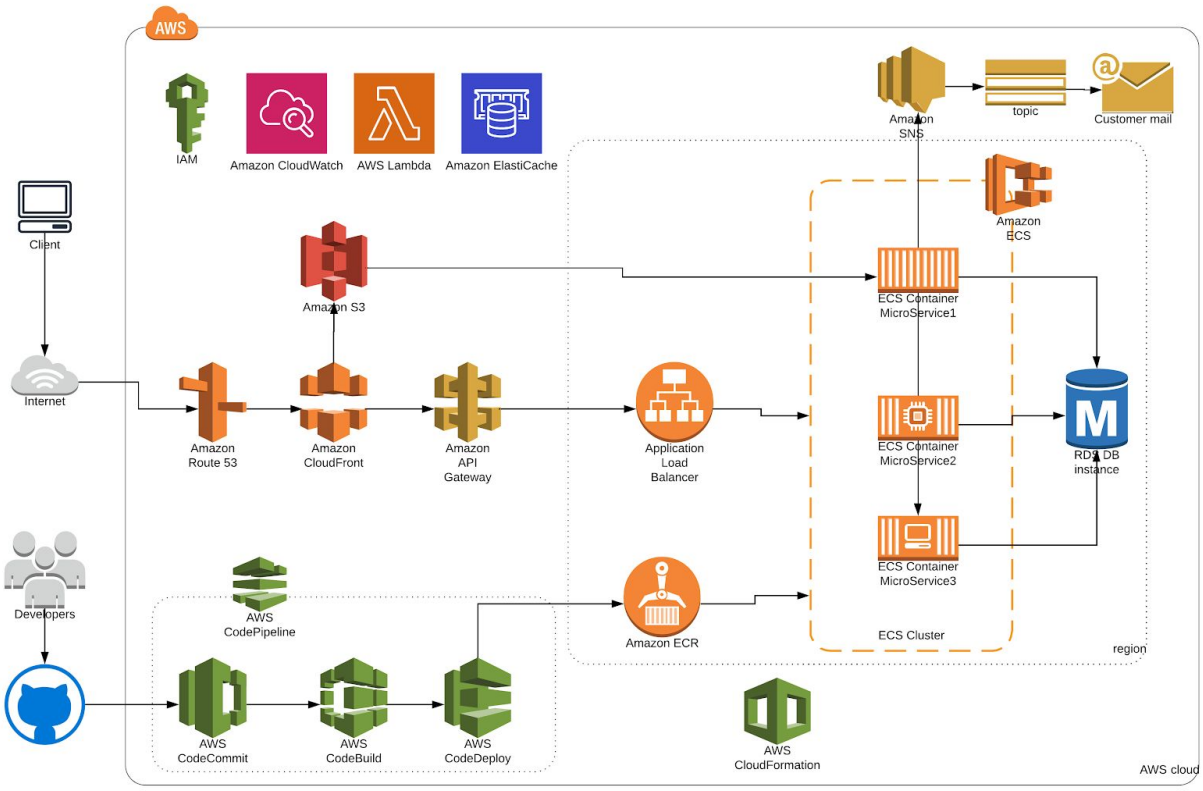
#	***Title***	***Link***
1.	*Break a Monolith Application into Microservices*	*https://aws.amazon.com/getting-started/projects/break-monolith-app-microservices-ecs-docker-ec2/*
2.	*The Twelve factor App*	*https://12factor.net*

Synopsis Reference:

Attach final approved Synopsis document here for reference to Problem Statement & Proposed Solution. You can also extend the Synopsis document itself with additional following sections.

Solution Architecture:

- *A detailed Architecture diagram.*



- *Choice of a particular cloud platform and its services/technology for each architectural component.*
- *Reasons behind architectural choices/trade offs, future scaling options/implications etc.*

Seq No	*Cloud Service*	*Reason for selection*	*Business Mapping*
1	*Amazon EC2*	*For deploying the application*	*Scaling*
2	*Elastic Container Service*	*Bundling the application*	*Scaling*
3	*Elastic Container Registry*	*Tagging the bundled application*	*Scaling*
4	*VPC*	*Having own virtual private cloud network*	*Security*
5	*CloudFormation*	*Faster Infrastructure onboarding*	*Management*

6	*Elastic Load Balancing*	*Balancing the application load*	*Networking*
7	*Route 53*	*DNS Web service*	*Availability*
8	*IAM*	*Authentication of the users*	*Security*
9	*RDS*	*Storing the data in RDBMS*	*Storage*
10	*Elastic Cache*	*Faster retrieval of data*	*Business turnaround*
11	*WAF*	*Restricting the malfunction access*	*Security*
12	*Cloud Watch*	*Logging*	*Business process flow*
13	*Simple Storage Service*	*Storage for website static content*	*Availability*
14	*API Gateway*	*Triggering Interface to application*	*Security/Reliability*
15	*Lambda*	*Serverless architecture*	*Scaling*
16	*Kinesis*	*Analytics*	*Business expansion*
17	*Content Front*	*Quicker availability of the content*	*Availability/response time*
18	*Simple Queue Service(SQS)*	*Messaging service as part of integration*	*Integration*

Prototype / Progress till date:

Share the progress made till date on the project implementation with screenshots. If you have done any prototype for the proposed architecture, share the same as well.

We have done a POC on the Microservices implementation using a node.js code base reference through Github.

https://aws.amazon.com/getting-started/projects/break-monolith-app-microservices-ecs-docker-ec2/

We were able to understand the concepts and the decoupling of the services. The stages are defined very well on how to go from a monolith to Microservices without any downtime as well. It enables to understand the 12 factor and it's implementation in a Microservices architecture.

Out of 12 factors, below are the -:

- *Codebase*
- *Dependencies*
- *Build/ release / run*
- *Stateless*
- *Portbinding*
- *Dev prod parity*
- *Logs*
- *Concurrency*

Note: For our Monolithic application we have planned to take an E-Commerce application which is already available in GitHub.

The GitHub link for the monolithic application is
https://github.com/Ekluv/ecommerce-django

Assumptions & Risks:

No change to the details as mentioned in synopsis.

Effort & Cost Analysis:

No change to the details as mentioned in synopsis.

Team Roles and Responsibilities:

Clearly mention team's role & responsibilities in the below format.

Team Member Name	*Contribution Till Date*	*Contribution Planned Ahead*
Ashutosh Tyagi *Rummy Maini* *Sridhar Sampath*	*We all be working in Agile mode thus keeping no individual dependencies and thus will help in meeting the deliverables on time. The team is capable enough of sharing the tasks between them and coordinating well and thus it doesn't get bundled on a single team member.*	