**Deployment Plan and model for Mediawiki**

**Tools used:**

1. Cloud Formation – For provisioning the infrastructure
2. Ansible – Deploying and configuring web Service
3. Python – For performing DB tasks

**Plan:**

As per the given assignment, I have decided to have 1 web server instance and 1 DB instance initially.

Web server will be configured with “apache HTTPD” and will be placed under a ASG. The Scaling group will be associated with a load balancer listening on port “80” which distributes the traffic among the instances.

DB server will be configured with MariaDB and will be placed under a ASG.

**How Web server is spun:**

A Launch Configuration for the ASG is defined with all the necessary details. It also contains the Metadata for the necessary package to be installed during the launch, services that must be enabled and running once the instance is ready and commands that has to be executed once the instance is up.

In this case I have placed my ansible configuration and playbook in S3 and have assigned S3 access to this node. Once the system is up the files get downloaded from S3 and is executed locally.

When the playbook (setup\_mediawiki.yml) is executed it gets the mediawiki.tar file from the repository URL and unarchive the same in the machine. Once unarchived then necessary permissions are given to the files and then a softlink is created to the web server root path and the service is restarted.

**How DB server is spun:**

A Launch Configuraion for the ASG is defined with all the necessary details. It also contains the Metadata for the necessary package to be installed during the launch, services that must be enabled and running once the instance is ready and commands that has to be executed once the instance is up.

In this case I have placed a python script (db\_setup.py) in the S3 location. This scripts connects to the database using “mysql.connector” and then performs the necessary DB tasks that is defined.

**Things that can be improved:**

* In this case I have exposed the Web server to the internet. This can be avoided by making the LB available to public and configure webserver security group to accept traffic only from the LB.
* A self-signed certificate can be generated and the same can be associated with the web server LB .
* DB port can be restricted to the internal network in a way that only web server can talk to the database.
* HTTP health check can be configured in the LB, so that it monitors the web service in the nodes.
* In case the DB is auto scaled, we need to have shared volumes across the instances that shares the data.

**Pre-requisites:**

A custom ami has been made that has pip installed in it. Using pip I have installed few tools required for the deployment.