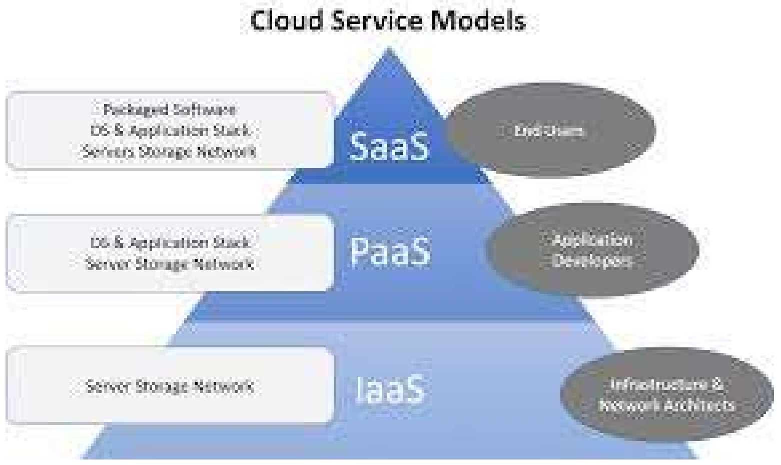


UE16CS352: Cloud Computing
Jan-May 2019

Selfie Less Acts

Fully Orchestrated Containerized Online Photo Sharing App

UE16CS352 | 27-04-2019

SNo	Name	USN	Class/Section
1	Akash Nagaraj	01FB16ECS041	A
2	Prajwal S	01FB16ECS262	E
3	RV Prithvi	01FB16ECS282	E
4	Rakesh Reddy	01FB16ECS292	E

Introduction

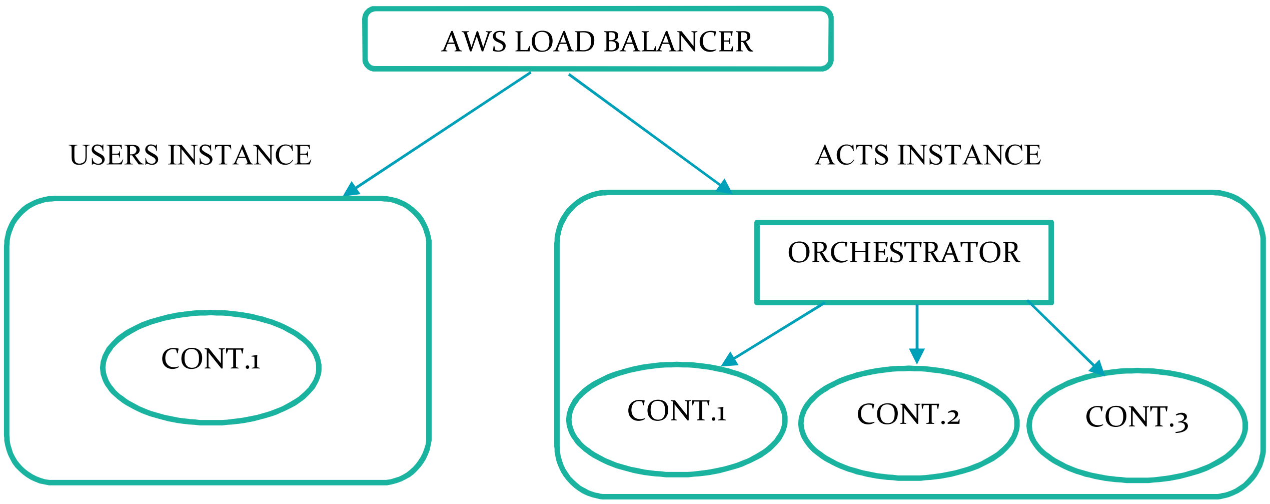
- It is an Online photo sharing app, that is containerized and orchestrated automatically. There are two instances of AWS hosting two different parts of the app, users and acts. An orchestrator running in acts instance automatically does load balancing, fault tolerance, auto-scaling of the containers running in the same instance. The backend of the application is built using Django a web framework for python.

Related work

Dockers were used to build, run, manage, and orchestrate the containerization of the containers in two components of the app's instances, users and acts. Django a web framework was used to build the backend of the application, and to handle with all the database related updates. Docker SDK a python library was used to build the orchestrator residing in the acts instance.

EXPERIMENTS/DESIGN

Our design is represented by the figure below. It has a AWS load balancer which forwards requests by users based on the paths to users and acts instance. There is a container orchestrator in acts instane which redirects requests to contaienrs runnning in the instance in a round robin order to balance the load. It also keeps checking the health of the containers and replaces the unhealthy containers with new ones. It does the part of auto scaling where it incerases or decreases the number of containers running based on rules defined automatically. The users and acts instances are in sync with each other about the databases.



The container creation and management is done using the docker container manager. Each of the containers run their own versions of Django web application updating the single database keeping the data in synchronization. One of the most important tasks was to implement database synchronization. The acts and users instances were communicating with each other to synchronize their data over REST API, but the containers in acts instance got connected to one single database on instance and kept updating their changes and migrating their modifications.

TESTING/RESULTS

All the testing of the REST API implementations was performed using Postman application. All the container orchestration was performed using a python script checking for the balancing, fault tolerance, and auto scaling. Although the containers were checked for manually, we tried to automate the process as much as possible.

REFERENCES

- Documentation on Docker - https://docs.docker.com/
- Documentation on Django - https://docs.djangoproject.com/en/2.2/
- Documentation on docker SDK for python - https://docker-py.readthedocs.io/en/stable/
  The above materials were referenced while working on the project.

EVALUATIONS (Leave this for the faculty)

Date	Evaluator	Comments	Score

CHECKLIST

SNo	Item	Status
1.	Source code documented	
2	Source code uploaded to CCBD server	
3	Source code in GitLab. Please do not upload your source code to github where it can be seen by everyone.	