Electronic Shopping (E-Shopping)

Cloud Computing Course Project, *E-Shop Phase 2*

Submission Deadline: *Bahman 23rd, 1398*

Introduction

In the previous section, you were familiarized with the architecture of a simple elastic electronic shop, called ***MSA-Shop***, which consisted of four modules; *User Interface, Authentication*, *Account Management* and *Trade Management*. To dive a little deeper into the world of microservices, you were assigned to write the code for a microservice of this system, called *"Account Management"*. This module is responsible for handling all interactions regarding *profile* and *wallet* entities. It interacts closely with the *"Authentication module"* for the purpose of *Identity Management*. After the development was through, you were supposed to package your service as a Docker Image and provide a *docker-compose.yml* file that could run your microservice alongside the *Authentication* microservice and their respective databases. This task gave you an idea of how microservices work and how Docker connects each microservice to another. Also, you learned that the *docker-compose* tool provides an easy way to run multiple microservices side by side on a ***single machine*** which could come in handy at the time of development.

Be that as it may, one of the most important challenges that we were hoping to tackle was ***elasticity*** and ***scalability***. In order to do that, we need to replicate our microservices on demand. Also, to avoid ***single point of failure***, we need our replicated microservices to spread over several machines. Obviously, this can be achieved by deploying your microservices to multiple machines and setting a load balancer to distribute user requests between them. The happy news is that you may sort this out by utilizing a *Cluster Orchestration* tool like ***Docker Swarm*** or ***Kubernetes***. These tools will take care of the hard part, while you only write the code for your microservices and provide a configuration file for the orchestrator to connect them all together with the settings that suits best for you (just like what you did with the *docker-compose* tool). Without further ado, let's move forward and deploy MSA-Shop in a cluster!

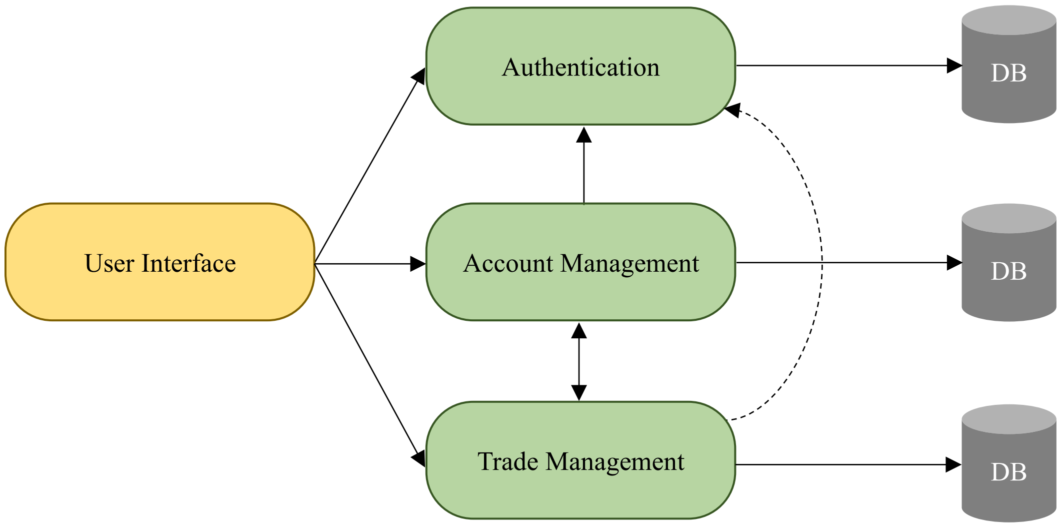

Figure 1 – **MSA-Shop Microservices Architecture**

Outline

What we want you to do is pretty simple. You are required to deploy the *MSA-Shop* microservices into a cluster using ***Docker Swarm*** with the configuration explained later on. To make your *"Account Management"* module integrate into the cluster, you are going to make minor modifications.

How to Go About It

Initially, you should provision *three* machines and install *docker* and *docker-compose* on them. Enable ***Swarm Mode*** by considering one of them as manager and the other two as workers. You may have an operating system of your choice but I recommend a *general-purpose* Linux OS like *Ubuntu Server*. Your main task is to write a *docker-compose.yml* file that deploys the following microservices:

- **Visualizer:** Responsible for visualizing cluster and container placement state
  - *URL:* https://hub.docker.com/r/dockersamples/visualizer
  - *Port:* 8080
  - *Replicas:* 1
  - *Placement:* manager
- **Front-end:** The User Interface
  - *URL:* https://hub.docker.com/r/msashop/front-end
  - *Port:* 443
  - *Replicas:* 3
  - *Placement:* spread
  - *Health Check:* required
  - *Environments:* NODE_ENV (*production*), AUTHENTIQ_URL, ACCOUNTICO_URL
- **Authentiq:** Responsible for identity management
  - *URL:* https://hub.docker.com/r/sayid/authentiq
  - *Port:* 2052
  - *Replicas:* 3
  - *Placement:* spread
  - *Health Check:* required
  - *Environments:* refer to git repository
- **Authentiq-db:** Authentiq database
  - *URL:* https://hub.docker.com/r/msashop/mongo
  - *Port:* not published
  - *Replicas:* 1
  - *Placement:* worker
  - *Health Check:* required
  - *Environments:* refer to git repository

- Accountico: Responsible for Account Management (Profile and Wallet)
  - *URL:* Your Microservice
  - *Port:* 2082
  - *Replicas:* 2
  - *Placement:* spread
  - *Health Check:* required
  - *Environments:* accordingly
- Accountico-db: Accountico database
  - *URL:* Your Microservice db
  - *Port:* not published
  - *Replicas:* 1
  - *Placement:* worker
  - *Health Check:* required
  - *Environments:* accordingly

Notes:

- Trade Management module is not ready yet and therefore you won't deploy it.
- Since there will be some updates in the following days, try to pull the images on a daily basis.
- There will be a demonstration of the project and you'll receive the link when its ready.
- You are required to parameterize your *docker-compose.yml* file just like the Authentiq microservice and provide a *.env* file to modify its behavior. Use *docker-compose* tool as a preprocessor for *docker stack deploy* command.
- For the *"Verify User"* and *"Forgot Password"* capabilities of the Authentiq module, you need to provide SMTP mail server host, port and credentials as environment variables. You are free to use any SMTP server of your choice like *Gmail* (for *Gmail*, do not forget about enabling ***allow less secure apps*** option).

Changes in Account Management

As mentioned earlier, it is necessary for you to make minor modifications to your module to fit in. Here is the list of changes:

- ✓ Utilize the health check capability in your *docker-compose.yml* file.
- ✓ Modify your **Profile** model to consider only the user ID to be unique instead of user's email.
- ✓ The **Create Profile** API should no longer auto login users, since there is a verification process available.
- ✓ Your service needs to accept an *environment variable* for payment callbacks; since users need to be redirected to the *User Interface* module after payments. (the *callback* API is not modified)

Extra Credit

You may perform the following task to earn extra credit:

- ✓ *Docker Swarm* does not support *autoscaling* out of the box. Using an innovative idea, try to provide autoscaling for microservices in *MSA-Shop*. You need to scale up or down (increase or decrease number of replicas) based on monitoring a parameter like ***resource usage*** (CPU, RAM or both) or ***number of requests*** per a specific amount of time. Furthermore, you should think of some way to test it out. This is called ***"Vertical Autoscaling"***.
- ✓ Innovative approaches for supporting **"Horizontal Autoscaling"** will be rewarded, as well. Autoscaling can occur the same way with *Vertical Autoscaling*; the only difference would be to provision / unprovision another machine to your *Swarm Cluster*, instead of increasing / decreasing number of replicas.
- ✓ Using any CI/CD mechanism, automate the process of deployment and updating your *Swarm Cluster*. Commits should trigger the process and in the case of a healthy build, lead to the auto deployment of the system.
- ✓ Deploy your cluster on the internet and use a CDN to serve the *User Interface* and the *Visualizer* to the world. Note that the User Interface should be available on ***HTTPS*** instead of *HTTP*.

Submission

Students need to pair for this phase of the project, as well. Groups of only ***two*** people are allowed; although you may try to do this without a teammate. Each group is required to create a ***private*** repository on GitLab and invite my account *"SayidHosseini"* as a *Collaborator*. Your repository should contain the docker-compose.yml file and its .env file, alongside it. Your deployment needs to be ready on ***Bahman 23rd, 1398 at 6 P.M***, when we will have an in-person delivery. You may make necessary changes up until your in-person delivery; however each team needs to notify me by email "sayidhosseini@hotmail.com", a day sooner (by *Bahman 22nd, 1398 at 6 P.M.*) . It is necessary for *each* student to commit changes via his/her own GitLab account while preparing the deployment. *Commits will be reviewed!*

Please note that the project will not be accepted after the deadline and that the deadline will not be extended!

Good Luck!

SayidHosseini

SayidHosseini

References

- https://docs.docker.com/engine/swarm/key-concepts/
- https://docs.docker.com/engine/swarm/
- https://docs.docker.com/v17.09/machine/overview/
- https://docs.docker.com/v17.12/machine/get-started/
- https://docs.docker.com/engine/swarm/swarm-tutorial/
- https://docs.docker.com/get-started/part4/
- https://docs.docker.com/engine/swarm/services/
- https://docs.docker.com/compose/environment-variables/
- https://github.com/makbn/docker_basics_tutorial