**CSCI-GA- 3033-010 - Cloud Computing**

**Course Project 1:**

**Archemy Application Cloudification/Migration**

|  |  |
| --- | --- |
| **Contents:** | |
|  | **Local Installation of the Application** |
|  | **Cloudification/Migration of the Application** |
|  | **Problems/ Challenges faced** |
|  | **Reasons and Alternatives** |

**About the Application:**

The Application has 4 components,

1. LDAP Server- Apache DS (Authentication)
2. Tomcat Server- Fortress Core (Fortress WEB and Fortress REST) ( Security)
3. MySQL Database (Database)
4. Archemy webapp ( Application)
5. **Local Installation of the Application:**

* The first step was to create the schema used by the archnav application on a locally running MYSQL instance along with importing stored procedures and populating the tables with data.
* Second step was setting up glassfish server on which the application is going to be hosted. To connect the glassfish server to the database, a JDBC connection pool had to be configured.
* Third step was configuring fortress application- 3 components fortress core, web, rest. All built with maven. Fortress core was the first to be installed as it generates fortress.properties, which is used bu the web and the rest. Foretress web and fortress rest required a running LDAP server. After getting LDAP up and running, then built fortress web and rest and deploy it on tomcat.
* Last step was to build the Archnav using jdeveloper and deploy it on glassfish server

1. **Cloudification/Migration of the Application**

Approach: Each component was made work in local and then was containerized.

Why Containerize?

* Light weight, they share resources without a full OS to underpin each app
* Platform Independence
* Ability to move from one cloud to another

1. ***Containerization:***

The Archemy Application is deployed in 3 containers

1. One for Tomcat Server housing fortress web and fortress rest
2. One for LDAP Server
3. Glassfish Server housing the Archnav Application
4. For MySQL Database - AWS RDS( Relational Database Service )
5. *LDAP Server- Apache DS*

* Base image with Apache DS pulled from Dockerhub

1. *Tomcat Server:*

* Base docker image pull from Dockerhub
* Docker file created with tomcat startup script ( Catalina.sh), adding of fortress web and fortress rest application
* Run the docker file

1. *MySQL- AWS RDS:*

* Created a Database in AWS RDS
* Established connection to MySQL Workbench
* Import Archnav Schema to MySQL and exported it using forward engineering
* Import Stored procedures
* Create Archemy Database user
* Import ArchNav Data to populate the tables in the schema.

1. *Glassfish Server:*

* Centos Base Image
* Install Oracle Java 8.251
* Download and install glassfish 5.0.1 from Oracle
* Set up Environment Variable and Environment PATH
* Configured connectivity to MySql (domain.xml + other configurations to support Oracle ADF)
* Add ADF Essentials libraries
* Add Archemy fortress security dependencies libraries
* Add JDBC Dependencies libraries
* Add archemy.ear file from the locally installed application
* Run the docker

1. ***Cloudification:***

Approach: Amazon Elastic Container Service, Amazon Elastic Registry

* 3 Repositories were created to host the docker container images for each of the Server, namely Tomcat, Glassfish and Archnav
* Pull the docker images and run the containers on Amazon Elastic Container Service
  + Create a cluster in ECS
  + Set up task definitions ( Blue print of application ) so that ECS knows which docker image to use for containers , number of containers and resource allocation
  + Configure ECS service with required service option such as service name, number of task, container port etc
  + Configure the cluster and underlying EC2 instance type.

A screenshot of a cell phone

Description automatically generated

1. **Problems/ Challenges faced:**

* Glassfish –
* Old Server, so a lot of version incompatibilities, do not work well with JDK > 1.8. Not supported by Oracle.
* And, also work with only specific version of the JDK 1.8. In my case, it is Oracle JDK 1.8.252.
* Jdeveloper version and Oracle ADF version should match for the application to compile successfully and work.
* Oracle ADF libraries have to be unzipped and all the jars in it should be in the top directory without nested directories.
* Cannot open the admin console in Remote login, so configuring the server and containerize is hard with the correct setting. **Solution**: Configure locally and use the domain.xml while building the container.
* Tomcat – Fortress
* Understand how the Fortress thing works, and the components (core, realm, rest and web).
* Update Fortress.properties with the correct LDAP server host/port, and load the fortress ldif file into the LDAP server, before configuring the Fortress WEB and REST, using maven.
* Update the catlina.sh to add the JAVA\_OPTs for fortress.
* ApacheDS
* Anytime the ApacheDS container is restarted, the fortress ldif file has to be loaded, and the maven setup of fortress web and rest has to be completed again to load the web demo users required to finish the setup.
* MySQL
* The MySQL server provided by RDS does not allow a DEFINER syntax for another user, so had to remove the definer syntax part.

1. Alternatives – Considered:

* **All in one EC2** – no scalability (if more tasks, create more machines ), no flexibility(fixed size machine), and with all the version incompatibilities it would be very hard in EC2 and implement installation steps to install and configure all the components and the interdependencies.
* **MySQL in a EC2** - RDS – managed Rdbms service – SaaS, Amazon provides the backup, security, admin.
* **Multicloud** – Docker containers, OS/environment independence.