Logo

Description automatically generated

**Etech Team 5-Group6-DataAnalytics**

Saadatou, Nelson, Francis, Aubin

May 06, 2023

***DOCKER-PROJECT***

1. **What is docker?**

Docker is a containerization platform which packages your application and all its dependencies together in the form of containers to ensure that your application works seamlessly in any environment, be it development, test, or production.

1. **List 3 differences between a docker container and virtual machine**

|  |  |
| --- | --- |
| ***Docker container*** | ***Virtual machine*** |
| Docker engine/Docker daemon | Hypervisor |
| Lightweight | Heavyweight |
| Only application at a time | Multiple applications at the same time |
| Using the host OS and will have containers without OS | Using the host OS and will have VM with OS |
| Exit when there’s no application running | Can stay idle even without any application on it and continue consuming resource |
| Share the same kernel | Have separate kernels separated by hypervisor |

1. **Why do companies prefer microservice over monolith applications?**

Greater flexibility, easy deployment and updated independently must account for the advantages that companies would be interested in.

These include:

1. Rapid application deployment

2. Portability across machines

3. Version control and component reuse

4. Sharing

5. Lightweight footprint and minimal overhead

6. Simplified maintenance

1. **What is a docker image?**

A combination of the artifact, Dockerfile and all dependencies.

1. **What is a docker container?**

A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.

1. **How do you troubleshoot miscommunications of docker containers?**

$Ping -c4 command can be used to troubleshoot miscommunications of docker containers. If ping response is positive, then the containers are on the same network if negative response then they are not on the same network.

1. **How are you using docker in your current role at Etech Consulting?**

At Etech, we use Docker mainly to containerize applications. In this sense, micro services start with Docker, which helps to decuple applications into small modules, and separate services that are running in a cluster. Also, we will use AWS ECS to provide for instances used to host Docker containers. Brought together, this will prepare the ground for Kubernetes that will orchestrate everything we dockerize.

1. **What implementation will you deploy to ensure data is persisted from a docker container application?**

Persistent mount will be implemented to ensure data is persisted from a docker container application.

1. **What plugin can you use to integrate Jenkins to docker APIs?**

The plugin used to integrate Jenkins to docker APIs is Docker API.

Docker API

Version 3.2.13-68.va\_875df25a\_b\_45  
Library plugins (for use by other plugins) docker  
This plugin provides docker-java API for other plugins.

In order to configure Jenkins with Docker plugin:

Step 1: Head over to Jenkins Dashboard –> Manage Jenkins –> Manage Plugins. Step 2: Under the Available tab, search for “Docker” and install the docker cloud plugin and restart Jenkins.

1. **Write down the process to delete a docker image that is currently running** **an application**

We need first to identify the targeted docker image

1. List all the docker images available: use the command *docker images*

Once identified, then

1. Stop running container that contains the docker image: use the command *docker stop container\_id*

Only then we can proceed with removing the docker image

1. Remove the docker image: use the command *docker rmi image\_id*
2. **What will you do if your docker image was wrongly created?**

The docker image should be deleted if it was wrongly created. Docker images cannot be edited. As a result, a new docker image must be created to replace it.

In order to do that in the local system, we can directly remove image using: *docker rmi image\_id* or prune the images using: *docker image prune*.

**Part2: Write down your CICD pipeline workflow from developer environment till creating your docker image and backing up the docker image to a container registry of your choice**.

At Etech Consulting it is a standard that each developer must install the talisman tools in their environment which ensure that whenever they try to make a commit it is going to check and make sure that they are not committing any vulnerable information into the code so that we can have clean code for our deployment. Once developer push there code into GitHub, (where we maintained a minimum of 3 branches feature, staging and main branches at the end code at the feature branches will be deploy to the dev environment, code in the staging branch will be deploy to the staging environment and code in the main branch will be deploy to the production) when developer commit their code into GitHub, and they create a pull request Jenkins is going to pull the code on the branch where the pull request is created and is going to deploy an application into our dev Environment. After everything is done the code is going to be move from the staging branch to the main branch. From the main branch where we are creating a change in our application Jenkins is going to pull those code then integrate with maven using plugins. Maven then is going to use the project object model .XML file definition which count down with the clone of the repository. Jenkins is going to build our artifact then connect to SonarQube using sonar- sonar plugging. Once it is connected to SonarQube it run quality code analysis. At Etech consulting we define high level parameter like code coverage, code smell, vulnerability, duplicate line to make sure that the code we use to deploy our application are free of bug, free from vulnerabilities. Once the SonarQube code analysis is done, Jenkins is going to integrate with nexus using the nexus artifact uploader plugging it is going to back up those artifacts into a release repository in nexus. Once Jenkins backup the artifact into nexus, it is going to connect with Docker. With Docker file that came with the repository, Jenkins is going to build our Docker image. Because we are deploying microservices Jenkins is going to build our docker image. Once Jenkins build our Docker image, Jenkins is going to backup those images into our Elastic container private registry in AWS.

(Hints: diagrams are also accepted include the explanations)