**My Pipeline Architecture**

Introduction :

Pipeline is designed to easily use Docker images as the execution environment for a single Stage or the entire Pipeline. Meaning that a user can define the tools required for their Pipeline, without having to manually configure each service.

Practically any tool which can be packaged in a Docker container can be used with ease by making edits in a docker-compose file.

Build Pipeline :

Prerequisites

Basically to be able to run my solution, you have to prepare a Docker environment.

Install Docker and docker-compose packages with all dependencies needed.

Solution:

My application composed with 2 parts:

1-Continous Integration

2-Continous Deployment

I have used those technologies:

**Gitlab :**

Where developers can push, commit and pull their codes.

It provides a variety of management by which we can streamline our collaborative workflow for completing the software development lifecycle. It also allows us to import the repository from Bitbucket github etc .And here are some features why I chose gitlab.

* Open-source community edition repository management platform
* Easy Maintaining of repository on a server.
* User Permission and Branch protection are enhanced.

**Jenkins :**

Jenkins is an open-source Continuous Integration server, able to run on various platforms ( Unix /windows)

Jenkins Continuous Integration Pipeline is a powerful instrument that consists of a set of tools designed to host, monitor, compile and test code, or code changes.

It has 1000+ plugins to ease your work.

**Sonarqube :**

Is the most widespread Source Code Quality Management Tool. As part of the CI build, Jenkins triggers a static source code analysis and the results are stored in SonarQube. It uses typical code analysis frameworks like FindBugs, Checkstyle, PMD and others.

**Nexus Repository Manager :**

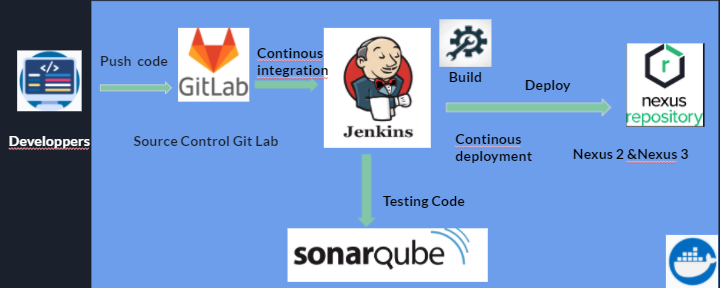
Is a typical Maven Artifact Repository. The Maven build uses Nexus as a Proxy Repository for third party libs. After the build the packaged artifacts are deployed to the Nexus Release Repository.

* Nexus2 for Maven Artefacts
* Nexus 3 to store docker images.

Architecture :

Here the architecture of my solution and how services will interacts together .

My platform hosted on a docker environment and each services is isolated from others by a separate Docker container.



I have deployed a docker-compose.yml file that will run those services.

* I have used the 3.3 version of compose file.
* I have used the last version of each image( Jenkins / nexus /sonarqube …)

In this section, I will answer questions mentioned in the technical task.

2- How to automate infrastructure :

Using AWS Cloud formation Service, we can easily model , provision and set up resources .

So that you can spend less time managing those resources and more time focusing on your applications.

You create a template that describes all the AWS resources that you want (like Amazon EC2 instances or Amazon RDS DB instances), and CloudFormation takes care of provisioning and configuring those resources for you.

You don't need to individually create and configure AWS resources and figure out what's dependent on what; CloudFormation handles that.

3- How to handle Scalability :

In my solution we can scale docker with kubernates .Where Kubernates is an open source project to manage cluster of containers as a single system .Kubernates handle managing and running Docker containers across multiple hosts.

Offering co-location of containers, service discovery and replication control.

4-How to handle Logging :

We can deploy the ELK stack ( Elastic search ,logstash and Kibana ).

It can search and index document files in diverse formats. Elasticsearch is compatible to run on every platform. Elasticsearch is an open-source. So, there is no need to pay any license cost for downloading it.

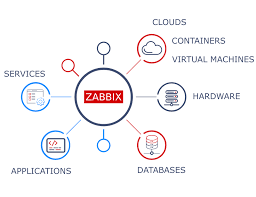
5-How to handle Security :

The best practice is to adopt a DevSecOps Culture to secure DevOps pipeline. Establish Credential Controls, Shifting security and prioritizing security as a part of the application’s design instead of leaving it to the end of the development pipeline.

6- How to handle Monitoring :

I have deployed zabbix platform services in my architecture .To supervise containers and workflow.

Zabbix is a free monitoring system for tracking statuses of various services of a computer network, servers, and network equipment. It is an open source enterprise solution that can perform a complex infrastructure monitoring (servers, network devices and virtual machines), visualize received information as graphs, monitor load, and performance of the equipment using own agents that are supported by almost all operating systems).



When deployed zabbix platform I’ve chose to run it separately using the docker run command .

The zabbix platform composed with 3 containers ( mysql db,zabbix server and front Apache)

|  |
| --- |
| docker run --name zabbix-mysql-db -t \ -e MYSQL\_DATABASE="zabbix" \ -e MYSQL\_USER="zabbix" \ -e MYSQL\_PASSWORD="zabbix123" \ -e MYSQL\_ROOT\_PASSWORD="zabbix123" \ -d mysql:5.7 |

|  |
| --- |
| docker run --name zabbix-server-mysql5 -t \ -e DB\_SERVER\_HOST="zabbix-mysql-db" \ -e MYSQL\_DATABASE="zabbix" \ -e MYSQL\_USER="zabbix" \ -e MYSQL\_PASSWORD="zabbix123" \ -e MYSQL\_ROOT\_PASSWORD="zabbix123" \ --link zabbix-mysql-db:mysql \ -p 10051:10051 \ -d zabbix/zabbix-server-mysql:latest |

|  |
| --- |
| docker run --name zabbix-web-apache-mysql8 -t \ -e DB\_SERVER\_HOST="zabbix-mysql-db" \ -e MYSQL\_DATABASE="zabbix" \ -e MYSQL\_USER="zabbix" \ -e MYSQL\_PASSWORD="zabbix123" \ -e MYSQL\_ROOT\_PASSWORD="zabbix123" \ --link zabbix-mysql-db:mysql \ --link zabbix-server-mysql5:zabbix-server \ -p 9009:80 \ -d zabbix/zabbix-web-apache-mysql:latest |