**JENKINS**

**INTRODUCTION:**

Jenkins is an open source automation server. It helps automate the parts of software development related to building, testing, and deploying, facilitating continuous integration and continuous delivery. It is server based system that runs in servlet containers such as Apache Tomcat. It supports version control tools, including AccuRev, CVS, Subversion, Git, Mercurial, Perforce, ClearCase, and RTC and can execute Apache Ant, Apache Maven and sbt based projects as well as arbitrary shell scripts and windows batch commands.

**Continuous Integration:**

* Reduced Risk
* Increased Confidence
* Better Quality Code
* Ready to Ship Code
* Systematic Versioning
* Code Quality Trend Analysis
* Time to market
* Reduced Cost

**Jenkins Feature:**

* Cost Savings
* Plugin Ecosystem
* Increased Productivity

**CD/CI Pipeline:**

This Pipeline is responsible for building codes, running tests, and deploying new software versions. The Pipeline executes the job in a defined manner by first coding it and the structuring it inside several blocks that may include several steps or tasks.

**Jenkins Pipeline:**

It is a suit of plugins which support implementing and integrating continuous delivery pipelines into Jenkin.

**Jenkin Commands:**

Jenkins has a built-in command line interface that allows users and administrators to access Jenkins from a script or shell environment.

**Jenkins Security:**

Jenkins have the ability to setup users and their relevant permissions on the Jenkins instance. By default, you will not want everyone to be able to define jobs or other administrative tasks in Jenkins. So Jenkins has the ability to have a security configuration in place.

**DOCKERS**

**INTRODUCTION:**

Dockers is a Linux-based, open-source containerization platform that developers use to build, run, and package applications for deployment using containers. Dockers enables you to separate your applications from your infrastructure so you can deliver software quickly.

**Important Docker Concepts:**

**Images:**

* Images are read only templates used to create containers.
* Images are created with the docker build command, either by us are buy other docker users.
* Images are composed of layers of other images.
* Images are stored in a Docker Registry.

**Containers:**

* If an image is a class, then a container is an instance of a class-a runtime object.
* Containers are light weight and portable encapsulations of an environment in which to run application.
* Containers are created from images. Inside a container, it has all binaries and dependencies needed to run the application.

**Registries and Repositories:**

* A Registry is where we store our images.
* You can host your own registry, or you can use docker’s public which is called Docker Hub .
* Inside a registry, images are stored in repositories.
* Docker repository is a collection of different docker images with same name, that have different tags, each tag usually represents a different version of the image.

**Docker Hub:**

It is a Public Repository.

WHY USING OFFICIAL IMAGES?

* Clear Documentation
* Dedicated Team for Reviewing Image Content
* Security update in a timely manner.

**Deep Dive into Docker Containers:**

* Running containers in a detached mode.
* Docker ps command
* Docker container name
* Docker inspect command

**Docker port Mapping and Docker logs:**

The format for port mapping is

**docker -p host\_port : container\_port**

**Docker commit:**

Docker commit command would save the changes we made to Docker container’s file system to a new image.

**Syntax:** docker commit container\_id repository\_name: tag

**Docker Commands:**

* **docker search**: It is used to search for public images on the docker hub.
* **docker pull:** If we know the name of the image, we can pull that form the Docker hub using the command docker pull.
* **docker images:** we should have some images in our local machine, and to confirm, we should run this command.
* **docker ps:** we can list all the running containers by using this command.
* **docker stop:** To stop a container this command is used with either the container id or container name.
* **docker logs:** This command is helpful to debug our Docker containers. It will fetch logs from a specified container.
* **docker rm:** If we want to remove a container, we can use this command.
* **docker rim:** If we want to free some disk space, we use this command.

**Docker Compose Commands:**

* **docker compose up:** Starts up all containers.
* **docker compose ps:** checks the status of the containers managed by the docker compose.
* **docker compose logs:** Outputs colored and aggregated logs the compose managed containers.
* **docker compose stop:** Stops all the running containers without removing them.
* **docker compose rm:** Removes all containers.
* **docker compose build:** Rebuilds all the images.