**Self-Intro**

* Good Morning / Thank you for giving me this opportunity to introduce myself / my name is Faizul Ahamed /I'm from erode tamilnadu.
* My schooling and diploma were completed in my hometown / I studied engineering at crescent engineering college in Chennai, Tamil Nadu.
* I like playing cricket, long drives in the car and bike, and spending time in hill stations and peaceful areas.

**Miracle Systems**

* From 2005 to 2011, I worked for Miracle Systems as a system engineer.
* I will be responsible for installing operating systems, installing and troubleshooting hardware,
* Maintaining inventory, coordinating with the vendor and the customer, and ordering product from the vendor based on the requirements of the customer.

**HCL**

* Later, I got an opportunity with HCL info systems in Bangalore as an associate engineer.
* My roles and responsibilities are maintaining Java-based applications. Maintain up to date code documentation, participating in code reviews and Contributing in all phases of the development lifecycle

**IMSI**

* Then I got a job at IMSI India Pvt Ltd as a Devops support engineer.
* There my roles and responsibilities are I will co-ordinate with developer and understand the concepts of applications, which GIT workflow they are using (5 different Work flow is there, 1. Basic workflow, 2. Feature branch workflow, 3. Git Flow, 4. Gitlab flow and 5. forking workflow)
* Also I co-ordinate with operation team and I will prepare the infrastructure to run the application.
* We will package the application in .jar or zip by using Jenkins tools.
* We used GIT code repository, here I can tracks all changes made to files in my project, building a history over time.
* My main another responsibilities to 1. (Continuous Integration / Continuous Deployment or Continuous Delivery) / 2. Automated / I worked on ANSIBLE automations tools
* I was promoted to team leader and the team size was 5.
* The work style will be same, apart from that I need to co-ordinate with team make sure that work flow is going smooth and need to complete the project on given timeframe.

**Precision**

* My last company is precision tech serve private limited
* There my roles and responsibilities are same like I handled in the previous project.
* I will coordinate with all other teams to obtain the latest information regarding current processes or any important organization's work.

**My day to day activities are:**

1. I will create a pipeline in Jenkins by using declarative pipeline.

1st need to install the required plugins (git/Maven, etc..) (For .net msbuild plugin)

And set the environment variable path.(Global Tool Configuration)

Pipeline { agent any

stages stage('Compile') steps{ echo "Compiling the script" } } stage ('Build'){ steps{

echo "Building the script } } stage ('Test'){ steps{ echo "Testing the script" }}stage ('Deploy') steps echo "Deploying the script"

}} stage ('Release') steps { echo "Releasing the script" }}}}

Src (it’s not mandatory, but we can use for the source for vars) / vars (It’s mandatory and we will use all the logical function here only) / resources (it’s not mandatory , we can save our application here like json file )

1. **Configuration Of Maven With Selenium**:

In Jenkins🡪 New item🡪Select **Maven Project** 🡪**General** (Give What details you want to feed) 🡪 **Maven Info Plugin Configuration** (Options are GITHUB Project, Etc..) 🡪 **Source Code Mgmt** (None Or GIT or Subversion) 🡪 **Build Triggers** (Github Hook, Poll SCM, Etc..) 🡪 **Build Environment** (With Ant, etc..) 🡪 **Pre Steps** (Here also some options) 🡪 **Build (Choose Maven version So here we need to click tool config and install (Type Maven name and choose install automatically apply and save)** / Then need choose **Root POM (Copy the POM location and paste it here / Goals & Options (Install Clean etc..) / 🡪POST Steps** (Some rules) **🡪 Build Settings** (E-mail Notification) 🡪 **Post-Build Action** (Some Options is there)

We need to download **TESTNG Results Plugin** and Mention like this (**\*\*/name(Workplace).xml)**

1. **Maven Installtion:**

In Jenkins 🡪 Manage Jenkins 🡪 Manage Plugins 🡪 Search maven and install all related maven plugins.

1. I will monitor the notification by using the Nagios monitoring tool and if any critical alerts then we need to fix the issue on high priority basis.
2. We will help and support developers and ask them everything running smoothly
3. Support existing infrastructure and maintaining the infrastructure / Need to check overall infrastructure of Devops to figure out whether it is sufficient or not, If not sufficient need to coordinate with concern team to fix the issues.
4. Make sure that pipeline is running smoothly
5. I will keep Documentation like server info, daily week charted

**Complicated task:**

* Build failure was happened and development team is saying I configured wrongly in JENKINS
* I checked everything from my end, and everything is fine
* So when we check their side, they configured maven in local repository so because of that build failure happened
* So I fix those issues and we build successfully.
* JENKINS was down and I was supported for US team
* I was worked in Indian time and I’m the only engineer available to support
* So I did google search because I have the access then I make that service up and Issue got fixed

**001 DevOps Pipeline**

* Plan-->Code-->Build-->Test-->Release-->Deploy-->Operate-->Monitor

**DevOps Lifecycle**

* Continuous Development 🡪Continuous Testing 🡪Continuous Integration 🡪Continuous Deployment 🡪Continuous Monitoring

**DEVOPS TOOLS**

* Version Control System tools. Eg.: github, gitlab, Bitbucket etc..
* Continuous Integration tools. Eg.: Jenkins, Teamcity, TravisCI, Buddy, etc..
* Continuous Testing tools. Eg.: Selenium, Appium, Eggplant, Testsigma etc..
* SCM (Software Configuration Management Tools) : Eg.: Ansible, Puppet, Chef, etc..
* Continuous Monitoring tool. Eg.: Nagios, Solarwinds, Tenable
* Containerization tools. Eg.: Docker Hub, Kubernetes, Marathon, Fleet, etc..

Local🡪Github🡪Jenkins🡪Through Jenkins Plugin Maven🡪Stored in artifactory server As well as it will go to sonarqube(Code test) 🡪Testing server(Selenium will be there it will test everything / If everything ok then it will send to master)🡪Github master🡪From github to Ansible🡪

**002 AWS:**

* AWS is one of the most popular cloud platforms which has a tons of services for different use cases.

**AWS Managed Services (AMS):**

**Login to AWS🡪Search Managed Services 🡪 Select RFCs option 🡪Click Create RFC 🡪 Choose Change Type (Browse change type / Select by category) 🡪In execution mode (Automated or Manual) 🡪 Example for creating snapshot for amazon elastic block store volume (Search EBS) 🡪 Choose Create EBS snapshot🡪Then fill the required configuration details like (Subject, Region, Parameters Volume ID 🡪Choose create RFC**

**Creating Tags 🡪 Select RFCs option 🡪Click Create RFC 🡪 Choose Change Type (Browse change type / Select by category) 🡪In execution mode (Automated or Manual) 🡪 Example for creating snapshot for amazon elastic block store volume (Search tag)🡪Select create resource tags (Auto) 🡪Then fill the required configuration details like (Subject, Region, Parameters Resources ARN, Add the tag 🡪Choose create**

* It’s AWS enabled service that helps enterprise adopt AWS at scale and operate more efficiently
* There are 4 use cases
* **Monitoring & Observability**:

24\*7\*365 proactive monitoring

AMS Uses:

Amazon GuardDuty

Amazon Cloudwatch

Amazon Macie

Incident detection

* **Configuration, Compliance, and Auditing:**

Aligned with National Institute for Standards and Technology (NIST)

Over 150 managed guardrails and security checks

Compliance certifications and attestations

FedRamp Moderate

Library of AWS configuration rules and remediations

* **Centralized Operations Management:**

Continuous learning mechanism

Tuning alarms

Creating automations

Adding new monitors and Service Level Indicators

* **Enterprise Governance and Controls**:

Comprehensive reporting on key performance metrics

Cost-saving insights

Incidents

Performance

Compliance certifications and attestations

**003 Team City, Octopus:**

TeamCity is used **to build and test software products in an automated manner**. It provides rapid feedback on every code change, reduces code integration problems, and leads to more effective teamwork. Many popular games, websites, banking systems, and all of JetBrains products are built with TeamCity.

**TeamCity from JetBrains is a popular continuous integration server that supports a variety of different version control systems and build runners**. Octopus Deploy and TeamCity can work together to make automated, continuous delivery easy.

TeamCity builds the code and runs tests, while Octopus takes care of: Distributing applications to all the remote machines, securely. Environment-specific configuration, like connection strings. Configuring IIS sites and installing Windows Services.

**TeamCity is a CI server written in Java**. It allows developers to integrate, code, and is easier to configure with simple steps

Octopus Deploy is an automated deployment and release management server. It is designed **to simplify deployment of ASP.NET applications Windows Services and databases**.

Jenkins is an open source continuous integration tool, while TeamCity is a proprietary offering from JetBrains. **TeamCity is easier to configure and more straightforward to use, while Jenkins has a rich plugin ecosystem and integrations**.

**004 Team City Project:**

A project in TeamCity is **a collection of build configurations**. A TeamCity project can correspond to a software project, a specific version/release of a project or any other logical group of the build configurations. The project has a name, an ID, and an optional description

**005 Create Pipeline:**

Import Projects🡪Configure Snapshot Dependency🡪Configure Artifact Dependency🡪Run Simple Chain🡪Configure Trigger and Checkout Rules🡪Add VCS Trigger🡪Restrict Checkout Scope🡪Complete Chain with Tests🡪Takeaway

Go to **Administration | Projects** and click **Create project**.

Open the **Dependencies** settings tab (you might need to click **Show more** to display this item) and click **Add new snapshot dependency**.

Open the **Dependencies** settings tab and click **Add new artifact dependency**.

TodoImage build with the **Run** button

Open the **Triggers** page and click **Add new trigger**.

Opposite our only VCS root, click **Edit checkout rules**.

A build chain is a sequence of builds connected with snapshot dependencies. A snapshot corresponds to a certain commit in the source code.

**006 Blue Green Deployment**

Blue green deployment is an application release model that gradually transfers user traffic from a previous version of an app or microservice to a nearly identical new release—both of which are running in production.

The old version can be called the blue environment while the new version can be known as the green environment. Once production traffic is fully transferred from blue to green, blue can standby in case of rollback or pulled from production and updated to become the template upon which the next update is made.

Because blue-green deployments utilize two parallel production environments, you can quickly flip back to the stable one should any issues arise in your live environment. This **reduces the risks inherent in experimenting in production**

What is blue green deployment in Kubernetes?

The blue/green step **copies your existing deployment and changes its version, creating a second one with the updated Docker image**. Note: At this point, both versions (old and new) of your application are deployed in the Kubernetes cluster. All live traffic is still routed to the old application.

**007 Docker**

* **Why Docker On Container**: Docker is a open source platform tool designed to manage the containers, which allows us to build the application in a container with required libraries, binaries and dependencies to run the application, ship the container and run anywhere
* **Why** **do we use Docker:** Portability, Light weight, Fast Delivery & Scalable,
* Docker is a tool to automate the deployment of an application as a lightweight containers so that the application can work efficiently in different environments
* Docker container is light weight packages of software that consists of all the dependencies (code, frameworks, libraries etc.) required to run an application
* Docker containers that run on Docker Engine, Docker engine is a docker daemon(Server)
* Standard: Docker created the industry standard for containers, so they could be portable anywhere
* Lightweight: Containers share the machine’s OS system kernel and therefore do not require an OS per application, driving higher server efficiencies and reducing server and licensing costs.
* Secure: Applications are safer in containers and Docker provides the strongest default isolation capabilities in the industry.
* Docker Image: The application container can be packed into the docker image (template) and the image can be shared with multiple platform.
* Docker Registry: Docker images are stored in Registry(Storage system)
* Docker Hub: It is a public repository and has millions of container images which can be reused.

**Docker Architecture and Components**

* **Docker Client, Docket Host, Registry**
* **Docker client** uses commands and REST APIs to communicate with the Docker Daemon (Server)
* When a client runs any docker command on the docker client terminal, the client terminal sends these docker commands to the Docker daemon. Docker daemon receives these commands from the docker client in the form of command and REST API's request.
* Docker Client uses Command Line Interface (CLI) to run the following commands -
* **docker build \* docker pull \* docker run**
* **Docker Host** is used to provide an environment to execute and run applications. It contains the **docker daemon, images, containers, networks, and storage.**
* **Docker Registry** manages and stores the Docker images.
* There are two types of registries in the Docker -
* **Pubic Registry -** Public Registry is also called as **Docker hub**
* **Private Registry -** It is used to share images within the enterprise.

**Docker Networking**

* Using Docker Networking, an isolated package can be communicated. Docker contains the following network drivers –
* Bridge - Bridge is a default network driver for the container. It is used when multiple docker communicates with the same docker host.
* Host - It is used when we don't need for network isolation between the container and the host.
* None - It disables all the networking.
* Overlay - Overlay offers Swarm services to communicate with each other. It enables containers to run on the different docker host.
* Macvlan - Macvlan is used when we want to assign MAC addresses to the containers.

**Docker Storage**

* Docker Storage is used to store data on the container. Docker offers the following options for the Storage -
* Data Volume - Data Volume provides the ability to create persistence storage. It also allows us to name volumes, list volumes, and containers associates with the volumes.
* Directory Mounts - It is one of the best options for docker storage. It mounts a host's directory into a container.
* Storage Plugins - It provides an ability to connect to external storage platforms.

**Docker Installation**

* Linux or windows (in Ubuntu: sudo apt-get update / sudo apt-get install docker docker.io) (In Centos, RHEL sudo yum –y install docker / sudo systemctl start docker / sudo systemctl enable docker) (docker –v)
* Windows Containers runs on only Windows OS
* Linux Containers runs on only linux OS
* Bcos its shares the kernel. So windows containers won’t support linux OS
* We can access linux container in windows machine but we need a linux virtual machine.

Docker Client : What are the commands that we are using that’s called docker client

Docker Server: It’s engine, i.e Docker DAEMON

**Docker Commands**

**Sudo Docker version / docker –v**

**Docker** (it will show what are the commands available in docker)

**Docker volume help** (It will display the command of volume related help) / Another ex: docker build help it will show the build command related help)

**Man docker volume** (It will show the manual of volume)

**Docker info** (It will display the client and server full details/ If we enable cluster then swarm will be active)

**Docker system df** (It will display docker disk usage / How much size consuming for Images, Containers, Local Volume, & Build Cache)

**Docker system events** (Get real time events from the server) (Anything failed or any problem in docker)

**Docker system prune** (Remove unused data / it’s risky carefully use)

**Docker stats** (It will show the memory usage) (Container ID / Name / CPU% / MEM USAGE /LIMIT / MEM% / NET I/O / BLOCK I/O / PIDS)

**Docker images** (It will list all the docker images)

**Docker search tomcat(anyname)** (It will search form docker registry (hub.docker.com (Here also we can search like tomcat or any name))

**Docker pull image\_name** (ex: **docker pull ubuntu** / to launch the container we need image so we pull some docker image) (If we want to pull any other older version then **docker pull ubuntu:67hhsd**(Tag No) (Tag is nothing but version no) (images are stored in local docker host so it will use our disk)

**Docker ps** (It will show already any docker is running)

**Docker ps –a** (it will show running, stopped all the containers)

**Docker run –name containername imagename** (Ex: docker run –name faiz1 httpd , it will create the container. Container name is faiz1, imagename is httpd) If we close or quit the command container will shut down.

**Docker run –d –name containername imagename** (So for continuous running use this command)

**Docker create** (docker will create but it won’t start, If we use run command it will create and start)

**Docker exec –it containername /bin/sh or bash** (We will login to the container, --it stands for interactive terminal)

**Docker exec containername command** (We can perform the task without login to container) Ex: docker exec faiz1 mkdir /tmp/foldername

**Docker cp source-file destination-containerID or containername:/tmp/** (Copy from dockerhosh to container ex: docker cp faiz.sh contaierID:/tmp/)

**Docker cp containerID-path dockerhost-destination-path** (container to dockerhost Ex: docker cp faiz1:/tmp/faiz.sh d:… )

**Docker exec container-name ls /tmp/** (It will list all)

**Docker stop containername** (It will stop the container)

**Docker start containername** (it will start the container)

**Docker rm containername** (It will remove the stopped container, this command won’t remove running container) (To remove smoothly 1st stop the container and remove)

**Docker rm –f containername** (it will remove the running container without any warning)

**Docker rmi imagename** (It will remove the image but before remove image we need to stop the container & remove the container)

**Docker rmi $(docker images -a –q)** (It will delete all the images)

**docker logs containername** (It will show the container logs) (If you want to see the application logs then need to login to container and check)

**docker run –d –it –name containername imagename /bin/sh** (To run OS image)

**docker top conatinername** (We will get what processes are running) (UID / PID / PPID / C / STIME / TTY /TIME)

**docker system prune** (It will delete exited, stopped container or unused containers)

**docker inspect imagename\_with\_tag or Image ID** (It will display all the information of images like PortID / config details / Layers details / Environment path / Working dir / Drivers, etc…) (We need to find out the image PortID for network access) (for detail about docker image use **inspect** command)

**docker run -d -it --name containername -p 8001:8001 imagename-with-tag** (to run through network port)

**docker save imagename or image ID > newfilename.tar** (save & backup docker image / Now we can copy this and move to any any server)

(moving that above backed-up tar file to another docker host / we need to know the destination docker host Ip address)

**Ip a** (It will display the IP address)

**Scp newfilename.tar faiz@remote-host-IP address:/username(ex: home/faiz)** (Now we backed up docker image file in tar , Now we want to restore)

**Docker load -i newfilename.tar** (Now it will extract from local machine and load as image)

**To create docker images from containers** (1st Log into container / Create one directory / Create one file)

(Now exit from he container ) Run this command **docker commit container-ID any-image-name:any-TAG**

**Push the image to docker registry, We need a docker hub account / docker login** (It will ask the username and password) **sudo docker push containername-with-tag**

**Docker Image**

* Docker images are just a template of a docker container and it is very similar to snapshot image with smaller in size.
* Single docker image can be used to create multiple containers for different environment like development, UAT and production.
* Docker images are very lightweight, small and fast to deploy the containers.
* Docker images are consists of many layers with unique image ID from base images. Each layer may have some changes committed on top of a existing layer.
* Docker images are read only layer of docker containers
* Docker containers are read write layer of docker images
* We can search and get all the images from hub.docker.com (Ex: search tomcat and we will get official tomcat images as well as other user modified images too)
* So we will get all the images from docker hub registry or image registry (hub.docker.com)
* We can create the docker image by using 2 method. 1. We can change the container and create the docker image or 2. Through docker file we can create the docker image

**Restart the EC2 instance will the Docker containers also be restarted?**

* **Docker Restart policies**: I believe I need to know a bit information, but let me assume something and give it the answer.
* When we ran the Docker run command, I remember that there is --restart subcommand that is present, this --restart subcommand is responsible to monitor the container status **By default it is no**.
* So if all the six containers are run with the default NO. Then obviously all of these containers shall not be restarted if the ec2 machine is restart
* But if the **--restart=always** , In this case if the ec2 instance is restarted then all the six containers shall also be restarted by Docker daemon

**(Always, on-failure, unless-stopped, no (default) )**

**016 Docker Swarm**

* Docker swarm is a service which allows users to create and manage a cluster of Docker nodes and schedule containers.
* Each node of a docker swarm is a Docker DAEMON and all docker daemons interact using the docker API
* Here, services can be deployed and accessed by nodes of same cluster
* Here one of our container fails we were able to do in use the SWARM to be able to correct that failure so the docker swarm manager is able to come in and reschedule containers.

Reverse proxy

* So SWARM node has full backups and full redundancy for any kind of failures that would happen and all of this work through command-line interface.
* Within a docker environment we have the **1. Docker Container 2. DAEMON 3. Docker Images 4. Docker Client 5. Docker Registry**

**DOCKER DAEMON**

* Docker Daemon interacts with the Host OS in order to create or manage Docker Container

**DOCKER SWARM Features**

* It is full decentralized, It very easy for teams to be able to access and manage the environment
* The communication that happens between the manager and client nodes would be the SWARM and it is highly secure.
* It is also auto load balancing within your environment and you can actually script that into how you write out and structure your SWARM environment.
* That load balancing then also allows you to then convert that SWARM environment into a highly scalable infrastructure.
* The rollback tasks allows you to be able to rollback environments to previous safe environments Ex: Something get pushed out or breaks you’re able to immediately roll back into a safe environment
* In swarm, containers are launched using services
* A service is a group of containers of the same image
* Services enables to scale your application
* Before you can deploy a service in Docker swarm, you must have at least one node deployed.

**There are two types of nodes in Docker swarm: Manager Node and Worker Node**

* Manager Nodes: Knows the status of all the worker nodes in a cluster.
* Worker Nodes: accept tasks sent from the manager node.
* Every worker node has as an agent, which reports on the state of the node’s tasks to the manager.
* The worker nodes communicate with the manager node using API over HTTP.
* In docker swarm, services can be deployed and accessed by any node of same cluster
* While creating a service, a user has to specify which container image to use.
* Here, a service is either global or replicated.
* A global service will run on every swarm node
* In a replicated service, the manager node distributes tasks to worker nodes
* A Service is a description of a task or the state, whereas a task does the work.
* Docker enables a user to create services, which can start tasks
* When a task is assigned to a node, it cannot be assigned to another node

**Primary Manager & Secondary Manager**

* It is possible to have multiple manager nodes on swarm, but there will be only one primary manager node, which gets elected by the other manager nodes.
* You have to elect one manager to be the primary manager and the other managers to be secondary managers.
* In many ways those secondary managers are really similar in concept to worker nodes in which they have the capability of a manager but they are dependent on that single primary manager to be able to provide the right instructions and for services and tasks to entire SWARM environment

I’m ready to work with any kind of DevOps technologies like Git, Jenkins, Maven, Dockers, Team City

And practically talking if you take me for your projects I believe I will get to work on some of technologies and not all. So I’m confident for even if I don’t have hands on experience on some devops tool I will easily understand the concept and I will give my 100% real and honesty effort. That’s my promise.