**Self-Intro**

* Good Afternoon / 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, so we need to assign some proper skillset person to fixing the problem.
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 Git & Github**

Git is the free and open source distributed version control system that's responsible for everything GitHub related that happens locally on your computer Ex: We have an app then later we added new features in app / After update new features app is break or corrupted / So here it will capture all the history details (Like when it was updated, who updated, etc..) whenever we changed or updated so we can easily go back to previous version and make it app run smoothly.

Github is the most popular one and lot of new features / Github is the platform online website that allows to host Git Repositories. Repository is just a folder where all the changes are saved / Git repository will collect all the history of projects and the folder name is .git

|  |  |
| --- | --- |
| **GIT** | **GITHUB** |
| **GIT** is one of the best version control available now | **GitHub**: Developers need a web/cloud-based code hosting platform |
| It is so flexible. You can create your own workflow strategies using git | Useful for version control |
| Git is the tool | Enables effective collaboration |
| A tool to manage our source code history | Download projects and file in one go |
| Installed & Maintained in your local system | Easy evaluation of each other's wor**k** |

**Git Commands**

* Push - pushing sends the recent commit history from your local repository up to GitHub.
* Pull - a pull grabs any changes from the GitHub repository and merges them into your local repository.

**Command is🡪 git reset commit-ID or git reset --soft commit-ID** (It will delete after this commit what & all happened) / **git revert commit-ID or git revert --soft commit-ID** (it will delete only one commit)

Most useful Command: **sudo apt-get install git –y /** **git init / git –version /git status / git log /**

**git config --global user.name "user name" / git config --global user.email** [**youremail@domain.com**](mailto:youremail@domain.com) **/**

**git config –list**

**/ git config user.name / git config user.email /**

**Clone from server (Github) to local**

**git clone repository-url address**

**Clone from local to server**

**git init** (In any location, Same repo name create in github) **/ git add . or particular file name /**

**git commit –m “any message” / git branch –M master** (Check branch name either master,main or other) **/**

**git remote add origin repo-url address / git push -u origin master or git push –f origin**

**git merge branchname** (It will create a new merge commit and existing branches are not changed in any way)

**git rebase branch1(from) branch2(to**) (If we want to moving or combining a sequence of commit one branch to another / alternate for merging) (merge is like a copy) (it will re-writes the project history)

**git cherry-pick commitID** (It will pick only one specific commit from another branch) (1st we need to go base branch)

**(squash) git rebase -i commit id (**Multiple commits into single commit) (We are changing the GIT history) (Once squash you can’t delete any particular commit) (b4 squash think very well)

**or git pull --rebase origin main / git push or git push origin main**

**git stash** (We will send to backstage or we can say hide in the background) **/** before complete the task if any other emergency task need to commit. Then we can use stash

**git stash pop** (come to stage again) **/ or git stash pop 0(stashnumber)**

**git stash list**

**git stash clear** (Delete the stash)

**git remote –v** (URLs that Git has stored)

**tree** /

**git branch branch\_name** (Branch Creation)

**git branch** (Current Branch)

**git branch –av** (To list all branch)

**git branch -D branch\_name** (Branch Deletion Locally)

**git push origin --delete branch\_name** (Branch Deletion to server)

**git checkout branch\_name** (Branch Switch)

**git push origin branchname –f** (Force Push)

**git branch New\_branch\_name1 branch\_name2** (create branch without checkout branch) /

**git checkout -b new\_branch\_name** (while checkout create new branch)

**git push origin branchname** (to merge)

**Difference b/w git fetch and git pull?**

|  |  |
| --- | --- |
| **Git Fetch** | **Git Pull** |
| **Git fetch** brings the changes from the remote repository to local repo then later we will merge manually | **Git pull** brings the changes from the remote repository to local repo and it will merge automatically (git pull = git fetch + git merge) |
| Git fetch no chance of merge conflicts | Here merge conflicts may occur |
| Git fetch will update in remote not in local so we need to merge to update in local workspace | It will update both remote and local branch / local branch need to be committed before running git pull |

**What is Git log? What information in git log help you?**

* Git Log is helpful command that shows us the history of over repository, information like the commits, the branch information, The merge, the fix, etc..
* This is critical for us in many cases that we want to revert our changes or we want to cherry pick some changes and test those changes in our branch

**003 CI pipeline job between github and jenkin pipeline**

(**Build** (Java, .Net, etc..)🡪**Test**(Unit test, integration test, etc..,Once test complete)🡪**Deploy**)

**Using Declarative Pipeline**

* Create Project in Jenkins: Click **New Item🡪** Type a project name🡪Select **pipeline** then click ok🡪Select **pipeline option**🡪**Definition**(Pipeline Script & **Pipeline Script from SCM**) We need select SCM and select **GIT**🡪Paste Repository URL**🡪**Choose if any **Credentials🡪Branches to build 🡪Branch Specifier** (Type branch name ex:\*/master)**🡪Script path** (Same file name of Git)🡪**Save🡪Build Now🡪**If any error fix it until output successful
* Other options are **General, Build Triggers** (Poll SCM set the time like \* \* \* \* \* is every minutes)**, Advanced Project Options**

**CI/CD & Automated**

* CICD is the main task and responsibilities of Devops. With Devops not just how we do this in any possible way but how we do this continuously and in an efficient fast and automated way.
* If any new features and bug fixes get added all the time that need to be deployed.
* CI: Continuous integration is the automated process, which generates software’s and its features quickly and efficiently

CI is a development practice that requires developer to integrate code into a shared repository several times a day

* CD: Continuous delivery is the automated process of delivering code changes to servers quickly and efficiently

**004 GitOps**

* GitOps is a Infrastructure as Code is hosted on a Git repository where its version controlled and allows team collaboration
* When we pull request process as for your application code so anyone in the team including junior engineers can create a pull request make changes to the code and collaborate with other team members on that pull request. For these changes we have a CI pipeline that will validate the config files and test them just like we test application code changes, after testing these commits other team members can approve the final changes. This could be developers or security professionals or other senior operations engineers who will review and approve the pull request.
* This way we tested well-reviewed config changes before they get app clied in any environment. So only after that changes will be merged back into the main branch and through a CD pipeline get deployed to the environment.
* Whether it’s changing something in Kubernetes cluster or updating underlying AWS infrastructure.
* So we have an automated process, which is more transparent and produces high quality infrastructure or configuration code where multiple people collaborate on the change and things get tested rather than one engineer doing all the stuff from their laptop manually than others don’t see or can’t review.

**005 CD Pipeline: Push Vs Pull Model:**

* Once merged into the main branch the changes will automatically applied to the infrastructure through a CD pipeline.
* In GITHUBS we have two ways of applying these changes. These are PUSH and PULL based deployments.
* Push-based deployments is what we traditionally know from the application pipeline on Jenkins or gitlab CICD, etc..
* PULL-Model: Here you have an agent installed in the environment like in Kubernetes cluster that actively pulls the changes from the GIT repository itself.
* The agent will check regularly what is the state of the infrastructure code in the repository and compare it to the actual state in the environment where it’s running.
* Example of GITOPS tools that work with the pull based model are flux CD and argo CD which run inside the Kubernetes cluster and sync the changes from the GIT repository to the cluster
* If any problem easily roll back to any previous state or working state.
* GIT-Single Source Of Truth:
* Increasing security

**006 JENKINS**

* Jenkins is an open source continuous integration/continuous delivery and deployment (CI/CD) automation software
* DevOps tool written in the Java programming language. It is used to implement CI/CD workflows, called pipelines
* Jenkins is used to build and test your software projects continuously making it easier for developers to integrate changes to the project, and making it easier for users to obtain a fresh build
* With Jenkins, organizations can accelerate the software development process through automation.
* Jenkins integrates development life-cycle processes of all kinds, including build, document, test, package, stage, deploy, static analysis, and much more.
* Jenkins achieves Continuous Integration with the help of plugins

**Comparison b/w Pain Points Vs Continuous Integration**

|  |  |
| --- | --- |
| Pain Points Before CI | Continuous Integration |
| Integration is Painful & Effort Consuming | Integration is automated and quick |
| Fixing issues at the end of iterations | issues show up early , because we integrate frequently |
| Intermediate merge issues can hold up teams | broken builds are given immediate priority by the developers |
| Long feedback cycle for functional defects | shorter feedback cycle - Developer is notified immediately |
| Long Iterations | Shorter Iterations. Faster time-to market |

**Pipeline (Declarative / Scripted)**

* The Jenkins pipelines are divided into two types. They are the declarative pipeline and scripted pipelines
* We used declarative pipeline.
* **Declarative pipelines** break down stages into individual stages that can contain multiple step (Stages, steps)
* **Scripted pipelines** use Groovy code and references to the Jenkins pipeline DSL.
* Devops Pipeline is: Plan-->Code-->Build-->Test-->Release-->Deploy-->Operate-->Monitor

**Jenkins (After Login Options)**

In Jenkins 🡪 **Manage Jenkins** (options are🡪

**In system Configuration** 🡪 Configure system / Global Tool Configuration / Manage Plugins / Manage Node and clouds

**Security**🡪 / Configure Global Security / Manage Credentials / Configure Credential Provider / Manage and Assign Roles / Manage Users / In-Process Script Approval

**Status Information🡪** / System Information / System Log / Load Statistics / About Jenkins

**Troubleshooting🡪** /Manage Old Data

**Tools and Actions🡪** / Reload Configuration from Disk / Jenkins Cli / Script Console / Prepare For Shutdown

**Jenkins user creation:** Manage Jenkins🡪 Manage users (Create, Delete, Modify)

**User Config:** click the drop down button from the username Ex: admin -->Click configure-->Here we can change password / Set SSH Public Key / Change name / Create API token / Time zone etc.

**User roles:** Manage Jenkins🡪Manage and assign roles

**Jenkins Shared Library**

* Jenkins shared library is the concept of having a common pipeline code in the version control system that can be used by any number of pipelines just by referencing it.
* We can create our own custom step without having to create a plug-in, So it’s much easier and simpler
* In fact, multiple teams can use the same library for their pipelines
* Code reusability (We can save in cenral repo) (We can call whenever required that function)
* Avoid code duplication
* Changes has been maintained in version control

**To configure :** go to **Manage Jenkins🡪Configure system🡪Global Pipeline Libraries**🡪We need provide **name**, **Default version** (Main or master or any branch name), Source code mgmt is **GIT**, Projecr rep **URL** this is one time configuration

Declarative pipeline Script Example:

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 )

**007 What is the controller and agent in JENKINS? Jenkins Master & Salve**

* Controllers in Jenkins is responsible for scheduling the jobs or the pipelines
* Agents are the ones that run these schedule jobs as a part of the CICD pipeline
* Agents take care of the running of the job, It can be a building of your job file, tuning of your repo, testing your code, uploading your jar file to a certain repository, anything.
* This set of controller and agent will help us to achieve what we call as a scaled out or highly available Jenkin setup in an organization
* **Jenkins Master**: Jenkins master will distribute its workload to the slaves / Schedules Build Job / It will Dispatch build jobs to slave for execution / It will monitor slaves / Present build result / Execute jobs directly
* **Jenkins Slave**: Jenkins slaves are generally required to provide the desired environment. It works on the basis of requests received from Jenkins master
* **Setting up Slave Nodes**: Using basic credentials / Using SSH Keys
* **Prepare Nodes As Slaves:** Create Jenkins User / Login as a created user / Create a Jenkins Salve directory / Install Java (JDK or JRE)

**008 How do you handle security in Jenkins?**

* 1st ask counter question to interviewer / could you please tell me which security around Jenkins are be a focusing about?
* When it comes to login authentications of Jenkins, By default, we do have admin username and password that is present using which we can log in, but we understand that this is not feasible when we deploy this application in an organization where many people will use it, Hence, I would suggest to integrate Jenkins with different held up service present in our organization. For example, we might be using an official Gmail account, We might be using an official single sign on service like Octa, etc..
* I would integrate Jenkins with this. This is very easy to do bcos there are multiple plugins that can do this and we have to make certain changes in this security part of Jenkins.

**Jenkins Installation**

* 1. Download Jenkins 2. Save to any location 3. goto command prompt (Windows) or Terminal (mac)
* 4. Go to folder where jenkins.war is located 5. java -jar jenkins.war
* (After install we will get jenkins administrator password (save it some where)
* 6. goto browser http://localhost:8080 (Type administrator password to continue)
* 7. Install required plugins like suggested or manually install
* 8. Create user or skip / Get started with Jenkins

**Jenkins Restart**

* 1. Go to command prompt --> press ctrl+c key to come out and again type java -jar jenkins.war
* 2. Another method from browser http://localhost:8080/restart

**009 How is your Jenkins setup done / managed?**

* 1. Scalable setup 2. At least 30 to 40 builds setup in a given day
* JENKINS is doing builds for both staging and live, We don't have a Jenkins separate for staging and separate for live
* So we have one master and then we have multiple agents
* Under master-->auto scaling group will be there --> Here we have ec2 instance --> under the ec2 we have agents
* The master will assign the tasks of build to the agent and bcos agent is in an auto scaling group so ASG will scale
* If we getting 20 builds all of a sudden then the ASG can really scale and the agents are able to build
* Second Live staging , We can create another separate auto scaling group / So one for live and another for ASG

**010 Poll SCM Vs Webhook**

**Poll SCM will check every particular times ones to check any changes is there are not / We will set Poll SCM like 5 mins or 10 mins /**

**Webhook is very efficiency and it will connect jenkins when ever any changes happened // for this webhook we need to install GitHub Hook trigger for GITSCM Polling plugins**

In Github -->> go to particular repo 🡪 Settings 🡪 Webhook 🡪Add a webhook 🡪 Paste the jenkin url followed by /github-webhook/ 🡪Select the content type (Application Json) 🡪

In Jenkin Create a project 🡪 in source code management select git and paste git repo url 🡪 In build trigger choose github hook trigger for GITSCM polling

**011 SELENIUM**

* The Selenium testing software tool is used to automate tests across browsers for web applications.
* It's used to ensure high-quality web applications — whether they are responsive, progressive, or regular.
* Selenium is an open-source tool. And, Selenium releases regularly.
* **Selenium only to test web applications**, we cannot test desktop applications or any other software
* Selenium doesn’t have any native reporting facility but we can overcome this issue by integrating with frameworks like test ng or j-unit.
* A Web application (Web app) is an application program that is stored on a remote server and delivered over the Internet through a browser interface.

Components Of Selenium: 1. Selenium IDE 2. Selenium RC (Remote Contr017ol) 3. WebDriver (Most Of them Using) 4. Grid

**Selenium Installation 🡪 Need to install JDK 🡪Need to install Eclipse 🡪Then Install JRE🡪Last install selenium**

**Selenium Test On Jenkins**

* Open Eclips🡪 There need to copy the location of projects
* Open Jenkins🡪 Create Project🡪 In **Build** **Environment**🡪 Choose **Execute Windows batch command** (If Windows (Feed the command like \* C: // cd (Location of the eclipse project) // mvn clean test // echo success) 🡪 **Save** 🡪 **Build Now**

**012 ANSIBLE**

* It’s tool to help automate different IT tasks.
* Ansible is a powerful automation tool, it can be automate any repeatable task.
* It’s more efficient & less time consuming in 4 different ways
  1. To Execute task from our own machine
  2. To Configuration / Installation / Deployment steps in a single YAML file
  3. Re-use same file multiple times and for different environments
  4. More reliable and less likely for errors
* It’s supporting all infrastructure from operating system to cloud provider
* **Ansible is agentless** , So we can execute Ansible files on the target servers on our own control machine on simple SSH access to the target servers, This is one of the unique advantages of Ansible / No deployment effort in beginning, No upgrade effort even moving to new Ansible version / We can Manages servers remotely

**How Ansible Work: MODULES**

* Ansible works with **Modules** / Modules are basically small programs that do the actual work
* They can push from the control machine to the target servers.
* It will do their work on target servers like installing application, stop a process, apply firewall rules, etc.. and when their job done, they get removed.
* Modules are very granular (very small) , one module does one small **specific task**, So you have a module to create or copy a file, Modules to install Ngnix server, Modules will start NGnix server, Start a Docker container will some arguments, Create a cloud instance so all of these are specific task that represent the module.
* Ansible has a hundreds of modules that each execute one small specific task
* Modules for many different clouds tools like AWS, Google Cloud, It has modules to work with docker, or many different databases, CICD tools , different networking security.
* If you want to execute IT task there is a high chance that there is a Ansible module to do that.
* Ansible is using simple YAML language, Which means no need to for learning a specific language for Ansible

Example Module Usages in YAML / This is the example of Jenkins Module We can do many different things in Jenkins like creating a job, Deleting a job using different arguments / Or in Docker Module🡪we can create container, Start container, apply some configuration or delete a container / or in Postgres Module like DB task we can create table, Set user privilege, etc..

* Modules are granular and specific / so when he have complex configuration on deploying application we will need multiple modules in a certain sequence and need to grouped together to represent that one whole configuration

**ANSIBLE PLAYBOOKS (WHICH TASK / WHICH HOSTS / WHICH USER)**

* The sequential modules are grouped in tasks, Where each task make sure the module gets executed with certain arguments also describe the tasks using the name
* We can execute multiple modules in a sequence.
* In **HOSTS, REMOTE\_USER** will execute these tasks
* Use variables for repeating values.
* Multiple Plays in a single YAML file / In the file that contains one or multiple plays is called a PLAYBOOK
* So PLAYBOOK describes **how** and in **which order** / at **what time** and **where** (on which machine) / **what (the modules)** should be executed or we can say **orchestrates the module execution**
* We can get HOSTS or IP Address, USER List from **Ansible inventory**
* Inventory is all the machine there are involved in Ansible task executions.

**ANSIBLE USAGE FOR DOCKER**

* DockerFile will basically prepare the application environment, So you have application artifact which could be a .jar file for example (In jar file may contains Config file, log dir, env variables, start script, etc..) So all these basically prepare the environments for the application startup.
* Docker File then produce the docker container.
* With Ansible PLAYBOOK we can create authority to docker file which is more powerful we can do the same kind of configuration with the application artifact and those surroundings. / Instant of just creating docker container, we can create docker container, vagrant container, cloud instance, bare metal, etc..
* Ansible allows you to reproduce the application across many environments
* You can manage not only docker container but also the HOST where the container is running. For example if the containers is dependency on storage, HOST, Network so by using ANSIBLE we can manage all these different parts. Which can make it possible to create one PLAY in a PLAYBOOK then basically manage both container and its HOST.

**ANSIBLE TOWER**

* Ansible tower is a UI dashboard from Red Hat which basically gives you centrally stored all your automation tasks across the teams.
* Configure permissions, manage inventory which jobs are running etc..

**Comparable Tools**

|  |  |
| --- | --- |
| **Ansible** | **Puppet & Chef** |
| Simple YAML | Ruby more difficult to learn |
| Agentless | Installation needed |
|  | So need for managing updated on target servers |

**ANSIBLE JENKINS INTEGRATION**

* We can run ANSIBLE Playbook from Jenkins pipeline using a plug-in.
* From Apache tomcat we will execute the Ansible Playbook
* I need to install Ansible on my Jenkins (Sudo yum install Ansible)
* If Ansible not install, Then enable epel and try (sudo yum-config-manager –enable epel)

**In Jenkins Side:**

* Install Ansible Plugins in Jenkins (**Ansible** & Ansible Tower)
* We need to configure Jenkins installation path under **global tool configuration**
* Click **Add Ansible** (Provide **name** & **Path to Ansible executables directoy**) / To get Ansible path from linux type **which Ansible** (it will display the path, until **\bin** copy the path)
* Save
* Create a New item (provide **Name** / Select **Pipeline** Project / **Ok**
* Go to **pipeline** tab / Write the **Own SCRIPT** (Declarative pipeline) / Click **Pipeline Syntax** / **Sample step** select **git** / paste Ansible file git **repository url** / Select **Branch** (Master or main or any)
* Again go to **pipeline syntax** / **Sample step** select **Ansible playbook: Invoke an Ansible playbook** / Provide **Ansible tool** name / **playbook file path in workspace** (Ex: **apache.yml** in git, same file name need to provide here) / **inventory filepath in workspace (**Ex: **dev.inv** in git, same name need to provide here) / Then we add **SSH credential details** / Check box tick in **Disable the host SSH key check** / Click **Generate pipeline script** (Copy that script and paste it in **OWN Script**) / Save it / **Build now**

**013 ANSIBLE AUTOMATION - LAUNCHING DOCKER CONTAINERS USING ANSIBLE**

* Linux environment: Install python / Ubuntu installation (sudo apt update) / install dependencies / add an Ansible repository / Install Ansible / SSH into the server / You can use SSH in using just a password or SSH in using key based authentication now **Ansible recommended ssh key based authentication** (We need install sshpass **sudo apt install sshpass)**
* By using two servers (One is **Ansible control node** and second is **docker host)**
* Docker **py python module** should installed on **Docker host** so that Ansible can work with docker
* Commands (**rpm –qa | grep -i docker-py**)
* Scenario (**\* Pull a centos image \* Create a custom image \* create a container \* validate deployment)** So we need **to write 4 playbooks** and later on we’ll use **Jenkins to automate** our deployment (<https://www.youtube.com/watch?v=lqEnooL0BkI>) **write 4 playbooks** or all in 1 playbook
* Login to **Ansible Host** / Commands **vi pull\_image.yml** / In vi editor

**-name: Pull Centos Image**

**Host: dockerhost**

**Tasks:**

**-name: pull**

**Docker\_image:**

**Name: centos**

Save it and run the playbook

* **Ansible-playbook pull\_image.yml** (Now playbook logged into the docker host and it’s trying to download the centos image)
* To check in docket host / command is **docker images or watch docker images** /

**014 Container (Containerization)**

**https://www.youtube.com/watch?v=CqRvGx3Syv0&t=806s**

* A container is a standard unit of software that Packages up code and all its dependencies so the applications runs quickly and reliably from one computing environment to another.
* Containerization is a lightweight virtualization technology alternative to hypervisor virtualization
* Any application can be bundled in a container can run without any worries about dependencies, libraries and binaries. Because containers create the isolated environment with all other required dependencies.
* Libraries and Binaries to run your application without any issue. Hence you can build the packages, ship the application to any environment and run it. **(BUILD🡪SHIP🡪RUN)**
* Containers don’t need OS / Containers uses HOST OS for compute Resource
* Containers have a huge impact on application delivery and are a real game changer for DevOps teams
* Flow: **Infrastructure**(Physical Server)🡪**HOST Operating System**🡪**DOCKER(Docker DAEMON)**🡪**CONTAINER** (Notes: Here if OS down all the container will effect but we have docker swarm, kubernetes with cluster feature service so if this OS down another will support to run the container) (In Virtual Machine also we can install OS and then install docker daemon to create containers)

**Advantages of Containers**

* Containers are isolated, doesn’t require operating system and it’s share a host kernel.
* So containers run on the same server and use the same resources, they do not interact with each other because it’s isolated.
* If one application crashes, other containers with the same application will keep running without any issues.
* It’s a portable and light weight operating system and it’s contain only required binaries, dependencies and libraries to run the application. So it can be move anywhere easily and run without worrying about compatibility, dependencies kind of issues.
* Faster and resource efficiency – Its very fast to boot, because containers are lightweight and start in less than a second since they do not require an operating system boot.
* Improves scalability and lower costs – By allowing more container in the environment without the need for more servers, Containerization increases scalability anywhere from 10 to 100 times that of traditional VM environments.
* Scenario: Developers codes is run in local machine but not in server (Due to any dependencies , different version, binaries, libraries, etc..) So avoid this we are using containerization

**Container Orchestration**

* Container orchestration tool, that allows the user to manage multiple containers deployed across multiple host machines.
* Container Orchestration tools are: Docker Swarm, Kubernetes, Amazon EKS, Google Container Engine, Azure Kubernetes Service, Openshift, Rancher, etc..
* Main useful is container management

**015 Virtualization or Hypervisor (Virtual Machines)**

* A hypervisor is a special type of operating system that enables a single physical computer or server to run multiple virtual machines with different operating systems.
* Virtualization technology that allow us to have multiple operating systems to share a single hardware processor. i.e, Allow one computer to run multiple OS
* Hypervisor is the software to create multiple VM
* HOST OS (Physical Computer / Here we have one OS Ex: Laptop)
* Guest OS (Virtual Machine)

**Two types of hypervisor: 1. Bare Metal Hypervisor 2. Hosted Hypervisor**

1. Bare metal flow: Hardware🡪hypervisor🡪Virtual Machines (Using in big data center, organization)
2. Hosted Hypervisor flow: Hardware🡪OS🡪hypervisor🡪Virtual Machines (Using in small companies, This is nothing but in our system itself we can create, Under our OS we can install virtualbox,vmserver then we can create vm’s)

* Two of the most popular hypervisors today are Windows HyperV and VMware’s ESXi.

**Disadvantages of Hypervisor:**

* [Heavy work load application and database might face some performance issues](https://www.trustradius.com/reviews/hyper-v-2022-04-02-07-17-28)
* It becomes challenging to find reliable hypervisor services for your particular enterprise needs within a budget

**016 Difference of Containerization and Virtualization**

|  |  |
| --- | --- |
| **Virtualization** | **Containerization** |
| Virtualization is the technology which can simulate your physical hardware (such as CPU cores, memory, disk)  and represent it as seperate machine | Containerization is os-level virtualization. It doesn't simulate the entire physical machine |
| It used  Hypervisor to detach the physical machine | It used docker engine in case Docker |
| It has hardware level isolation so fit is fully secured | It has process level isolation |
| It is heavyweight | It is very lightweight |
| It is not portable | It is very portable. We can build, ship and run anywhere |

**017 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

**8 Kubernetes**

* Kubernetes is an open source orchestration tool
* Developed by google
* Helps you manage containerized applications in different deployment environment (Physical, Virtual, cloud, hybrid environments)

**Need for a container orchestration tool**

* Trend from monolith to micro services
* Increased usage of containers
* Demand for a proper way of managing those hundreds of containers

**Features of orchestration Tools Offer:**

* High Availability or no downtime
* Scalability or high performance
* Disaster recovery – backup and restore

**Kubernetes Basic Architecture:**

* Kubernetes cluster is made up with at least one master node and then connected it to you have a couple of worker nodes.
* Where each node has a kubelet process running on it.
* Kubelet is actually a Kubernetes process that makes it possible for the cluster to talk to each other to communicate to each other and execute some task on those node like running application processes.
* Each worker node has docker containers of different applications deployed on it.
* Worker nodes are where the actual work is happening so here is where your applications are running.
* Master node runs several Kubernetes process that are absolutely necessary to run and manage the cluster properly.
* One of such processes is an API server which also is a container.
* An API server is actually the entry point to the Kubernetes cluster, so this is the process which the different Kubernetes clients will talk to like UI if you are using Kubernetes dashboard / an API if you’re using some scripts and automating technologies a CLI(Command-line tool) so all these will talk to the API server.
* Another process that is running on master node is controller manager which basically keeps an overview what’s happening in the cluster whether something need to be repaired or maybe if a container died and it needs to be restarted etc..
* Another one is scheduler which is basically responsible for scheduling containers on different node based on the workload and available server resources on each node. So it’s an intelligent process that decides on which worker node the next container should be scheduled on based on the available resources on those worker node and the load that the containers needs.
* Another very important components of the whole cluster is an etcd key value storage which basically holds at any time the current state of the Kubernetes cluster. So it has all the configuration data inside and all the status data of each node and each container inside of that node. The backup and restore made from these etcd snapshots because you can recover the whole cluster state using that etcd snapshot
* Last components is virtual network. Which enables master node, worker nodes talk to each other.
* In production environments usually you would have at least 2 masters inside of your kubernetes cluster. If one master node is down the cluster continues to function smoothly because you have other masters available

**Kubernetes Basic Concepts:**

* Kubernetes PODS: is the smaller unit that you as a Kubernetes user will configure and interact with in POD is basically a wrapper of a container and on each worker node you have a multiple PODS and inside of a POD you can actually have multiple containers
* Usually per application you would have one POD. So the only time you would need more than one containers inside of a POD is when you have a main application that needs some helper containers.
* Example: Database would be one POD, A message broker will be another POD, A server will be another POD
* Virtual network dispense the Kubernetes cluster so its assign each POD its own IP address, So each POD is its own self containing server with its own IP address. The way that they can communicate with each other is using the internal IP addresses
* We don’t actually configure or create containers inside of Kubernetes cluster but we only work with the PODS which is an abstraction layer over containers.
* POD is a components of Kubernetes that manages the containers running inside itself without intervention. Example: If a container stops or dies inside of a POD it will be automatically restarted inside of the POD
* PODs are ephemeral components which means that PODS are die very frequently and when a POD dies a new one gets created and here is where the notion of service comes into play.
* Whenever a POD gets restarted or weak a new POD is created and its get a new IP address. Example: If you have your application talking to a database POD using the IP address the PODs have. The PODs restarts it gets a new IP address obviously it would be very inconvenient but just that IP address all the time so because of that another component of Kubernetes called service.
* Service is used which basically is an alternative or a substitute to those IP addresses so instead of having this dynamic IP addresses their services setting in front of each POD that talk to each other so now if a POD behind the service dies and gets recreated the service stays in place because their life cycles are not tied to each other and the service has two main functionalities one is an IP address so it’s a permanent IP address which you can use to communicate with between the PODS and at the same time it is a load balancer.

PODS

* A pod **is the smallest execution unit in Kubernetes**. A pod encapsulates one or more applications. Pods are ephemeral by nature, if a pod (or the node it executes on) fails, Kubernetes can automatically create a new replica of that pod to continue operations.

**Kubernetes Configuration**

* All the configurations in Kubernetes cluster actually goes through a master node with the process called API server
* Kubernetes clients which could be a UI a Kubernetes dashboard for example or an API which could be a script or curl command or a command line tool like cube CTL they all talk to the API sever and they send their configuration requests to the API server which is the main talk to the API sever entry point or the only entry point into the cluster.
* In this request have to be either in YAML format or JSON format. In this way we are sending a request to Kubernetes to configure a component called deployment
* Deployment which is basically a template or a blueprint for creating PODS and in this specific
* Configuration example: we tell Kubernetes to create to replica PODS for us called my app. with each POD replica having a container based on my image running inside. Addition we configure what the environment variables and the port configuration of this container inside of the POD should be. The configuration requests in Kubernetes our declarative format so we declare what is our desired outcome from Kubernetes and Kubernetes tries to meet those requirements
* Ex: We want 2 replica PODS of my app deployment to be running in the cluster and one of those PODS dies the controller manager will see that the ease and shoot states now. Actual state is one POD our desired state is two so it goes to work to make sure that this desired state is recovered automatically
* Restarting the second replica of that POD.

**9 Difference between Docker and Kubernetes:**

|  |  |
| --- | --- |
| Docker | Kubernetes |
| Docker is a container technology, It’s create an isolated environment for applications | Kubernetes is an infrastructure for managing those containers |
| Automated Building and Deploying process of applications. It’s used in the CI CD process. | Kubernetes comes into action after the application container has been deployed. It’s take care of automating, scheduling and management of that deployed application containers. |
| Docker is a container platform to configure build and distribute those build containers | Kubernetes is an ecosystem for managing a cluster of multiple docker containers |
| Docker is mainly used in the local development process. When you developing a software applications you would use docker containers for different services. Like database, message, broker etc.. |  |
| It’s also used in the CI process to build your application and package it into an isolated container environment |  |
| Once build the container get stored or pushed into a private repository. | Here kubernetes comes-in. If you have a development server that is made up of multiple virtual or physical servers you would basically Kubernetes on top of those servers. Once Kubernetes is running you will create a cluster that would actually run your docker containers. |
|  | Cluster Work: We have a Kubernetes engine that spends multiple virtual or physical servers to create one cluster. i.e, where docker containers are actually running. |
|  | We can distribute the number of docker containers across those physical or virtual servers as our wish. Where each containers will be its own application and the Kubernetes service that actually enables docker to run in the cluster is kubelet |
|  | So each node in the community’s cluster will actually have kubelet and the technology that is actually comparable with Kubernetes is Docker swarm |

**10 Difference between Docker Swarm and Kubernetes**

|  |  |
| --- | --- |
| Kubernetes | Docker SWARM |
| Kubernetes is much more complex to install | Easier Installation |
| More complex with a high learning curve, but more powerful | More lightweight and easier to use, but limited functionality |
| Some of the powerful functionality that Kubernetes offers in comparison is supports auto-scaling configuration | Manual Scaling configuration |
| Built in monitoring | Needs 3rd party tools for monitoring (DOC Station, Portainer,etc..) |
| Manual setup of load balancer | Auto load balancing |
| Need for a separate CLI tool (kubectl) | Integrated Docker CLI |

**11 EKS (AWS / Elastic Kubernetes Service)**

**AWS:**

* AWS is one of the most popular cloud platforms which has a tons of services for different use cases. Here EKS is just one of those in many services

**What is EKS?**

* It’s a managed Kubernetes cluster service
* AWS will manage the master nodes.
* It will create the master nodes and install all the necessary applications on them like container, runtime, Kubernetes master process
* It will take care of scaling it, When needed doing backups on that etc.
* If it’s small team of people then usually it’s good idea to let the platform do this maintenance for you. So you can focus on deploying your applications in Kubernetes without worrying about whether master nodes are properly backed up, etc..
* It’s mean you only create and take care about the Worker nodes

**How to use EKS?**

* If we have small projects which you want to run in Kubernetes and you want to do that with EKS.
* In order to create the cluster with EKS they’re following steps involved.
* Before you even create a cluster you need to do some setup

1. Create AWS account (Free tier for 1 year for new user)
2. Create a VPC (Virtual Private Cloud) it’s our private stuff, it won’t interfere with other AWS users.

NAT

1. Create an IAM role with Security Group (Create an AWS user with a list of permissions, Permissions to create and do stuff with EKS service)
2. Create Cluster Control Panel: (Master Nodes)

You create the cluster with the AWS role or this IAM role with a permission to create and configure the EKS cluster. Ex: Name of the cluster, Kubernetes version, which region, set some security for clusters

We can do these are through AWS UI or Management Console or AWS Command line.

1. Create Worker Nodes and Connect to Cluster:

In AWS these worker nodes will be some ec2 instances with certain CPU, RAM and Storage Resources

How to create worker node:

We create them as node group or group of nodes and not a separate ec2 instances.

When you create a node group you choose the cluster it will attach, you define also security group for that you select the instance type of ec2 instances which resources ec2 instances should have and node group you also have auto-scaling. Here we need mention minimum and maximum number of nodes

Once create the node group as well as your worker nodes connected to the cluster.

1. Finally you will connect to the cluster from your local machine because we need to deploy our applications from our laptop or local computer using cubectl which is Kubernetes command line tool.

Configure cubectl to talk to the remote cluster.

* It’s an complex compare to other managed Kubernetes services like linux Kubernetes engine or digital Kubernetes service but AWS is of course very powerful and that increases the complexity.
* But good news is there is a way to simply this process of creating a cluster on AWS much faster and more efficiently without having to do all of these one by one manually. We can do that with a command line tool called EKS control, It is not a AWS tool, it is from weave works and has a lot of contributors from community.

**Install EKSctl:**

From command line tool:

* brew tap weave works/tap
* brew install weave works/tap/eksctl

eksctl needs to authenticate with AWS in order to create a cluster.

Before execute any command you have to have your AWS user credentials locally in the path like this

* .aws/credentials in command prompt then type aws access key & aws secret access key

**Now create a cluster in the eksctl:**

* **In amazon EKS**🡪Click create cluster🡪

**In Command Prompt**:

eksctl create cluster

--name faizulcluster

--vsersion 1.17

--region (any) ex: --region eu-centra-1

**Worker node creation:**

--nodegroup -name (any)

--node-type t2.micro(This we will choose from aws)

--nodes 2(no of worker)

**13 DevOps KPIs that you monitor in your team today?**

* KPIs: stands for Key Performance Indicators
* There are Three Key KPI / the three KPIs that our team is currently monitoring are 1. Deployment Frequency 2. Percentage of failed deployments 3. Meantime to failure recovery
* Deployment frequency is nothing but how often are we deploy successfully to production
* Percentage of failed deployments is out of these deployment that is happening to production, How many them are failing
* Meantime to failure recovery the services that we own, If they go down how fast our recover
* Based on this three key metrics, we try to improve our system and the process to achieve better DevOps lifecycle in our company

**14 Local and remote artifact? Which one is commonly used in organizations and why?**

* Local and remote artifact are common way of storing your final code in a zip, jar, wheel format or compressed format
* Any given project will have hundreds of lines of code, many files, you will not be able to send all of them to the machine
* you would like to compress it and then unzip it during the time of deployment
* Jar file is famous for any java application, Wheel is for python, etc / this kind of building that is compressing this whole file will be a part of your Jenkins job or let's say a pipeline job.
* If you do your compression in local machine during your development, the jar, zip or the wheel file will be present on your local machine
* If somebody wants to do a deployment, they will have to come and ask you, can i have your jar file and you have to share the jar file from your laptop with them so that they can do the deployment
* **Having jar file locally or artifacts locally will not help**. Hence, we upload these artifacts into a **central artifact repository most famous one example is MAVEN**, where you upload it with the version title.
* After anyone in your organization anywhere in the world can download and deploy on their infrastructure.
* Local artifacts are only good for your testing purpose and during the development.

**15 Secret engine used in your organization and how does it help?**

* The secret engine that we tend to use in our organization is Vault.
* The reason is, it is easy for us to integrate with many other services that we have. Example: Kubernetes, Terraform etc..
* This play a very big role in the world of DevOps bcos we should never commit any kind of secrets through our repository or even through our commands

**16 SDLC (Software Development Life Cycle) :**

**Requirement🡪Planning🡪Design🡪Development🡪Software Testing🡪Deployment🡪Maintenance**

* Software Development Life Cycle (SDLC): 1. Requirement (Gathering & Analysis: \* Product Features \* Users \* Usage \* User Requirement \* Market State) 2. Planning (What do we want? Cost, Resource & Risk) 3. Design Architects (Based on detailed requirement system design documents are created) 4. Development (Developers / Software development based on inputs of design documents) 5. Software Testing (Quality Assurance will identify the defects to ensure the quality product is good) 6. Deployment (System Admin / Operations team will deploy to the production environment so user will access the products) 7. Maintenance (Changes & Uptime)

**17 SDLC Different Model(Agile, Kanban, Waterfall Scrum)**

* Different Models in SDLC: 1. Waterfall 2. Agile 3. Kanban 4. Spiral 5. Big Bang etc.. (Ex: The destination is same but we need to choose which path is best based on usage, cost, maintenance, time taken etc..)
  1. Waterfall SDLC: Each phase should be completed before the next phase can begin / Very long Iterations (Month or More) / The flow of waterfall is (Requirement -- Design -- Implementation -- Testing -- Maintenance)
  2. Agile SDLC: We can make it Smaller lists Iterations of each modules, after every iteration we will get feedback from customer and If any changes is required we can modify in next iterations (2 to 4 Weeks each module) But here several times we need to change the Code, Deploy, Testing/QA for single iteration / Like that We have so many iterations / Operation team need to keep the server uptime but here regular deploy request

**Agile**

* Agile is a popular methodology that offers guidance to teams on how to approach their projects
* Agile is based on the values and principles outlined in the agile manifesto which was published on 2001.

**Agile have 4 factors:**

1. Individuals and Interactions
2. Deliverables and documentation
3. Customer collaboration
4. Quick response to change

* Agile teams break their projects down into short phases called iterations. They make continuous improvements and provide deliverables within each iteration and provide high quality deliverables on a regular basis

**Kanban:**

* Kanban is a project management framework that boosts workflow visibility. Teams use a Kanban board to categorize their tasks by status.
* In a Kanban project teams use card to symbolize their tasks on the Kanban board.
* Cards are moved to a different column when the task status changes. For example: Team member might move a task from a column marked to do to a column marked doing or new to doing

**Kanban Principles:**

Kanban teams followed 4 main principles

1. Start with the most urgent task
2. Make small changes regularly
3. Maintain existing roles
4. Encourage Leadership (Encourage every team member to act as leader)

**Agile and Kanban Similarity:**

* Kanban falls under the agile umbrella meaning the two are linked in project management. They represent an alternative to traditional project management which includes framework such as waterfall agile and Kanban have a few features in common.

1. BITE SIZE TASKS

* The 1st similarity is that both Kanban and agile break projects into smaller chunks to accelerate work process and deliver result on time. So it’s easier to manage.

1. TRANSPARENCY

* 2nd similarity is Kanban and Agile teams have a clear understanding of their tasks and provide regular updates.
* Kanban boards clear view of what tasks are planned in progress and completed / Agile team communicate openly and regularly so every team is kept in the loop

**Difference between Agile and Kanban**

|  |  |
| --- | --- |
| Agile | Kanaban |
| Agile is a overarching methodology that offers general guidelines to varying teams. | Kanban is a visual framework that provides specific details on how to manage projects |
| TIME: Agile teams work in short, time boxed periods | Kanban projects have no set time structure |
| TASKS: Agile teams use metrics known as story points to eliminate the problem for more accurate planning. | Kanban tasks vary in length and effort, making it difficult to plan resources. |
| VISUALIZATION: Agile teams may use boards or other visual aids but they are not mandatory | Kanban teams use a Kanban board to visualize workflow |
| MEETINGS: While daily stand-up meeting are a key requirement in Agile projects (Where team discuss past, future plans) | Kanban daily meetings are optional and team member might instead choose to communicate status update via the Kanban board |
| TEAMS: Agile projects have cross-functional teams, This is the key element of the Agile | The Kanban structure is looser meaning the framework is also suitable for functional team that operates separately |

**Scrum Vs Waterfall:**

**Waterfall:**

Typically goes through a lengthy planning process, it will take several months followed by building the products which again take many months and then testing the products. Reviewing and deploying the products.

Several problems with this method first of all the planning must be completed before any work begins and in most cases the planning is done without entirely understanding the project

**SCRUM:**

Scrum an implementation of Agile the process is broken up into smaller pieces.

1st we do just enough planning to get started with building the minimal features set.

We build what was planned.

We test and review that small features set and get it ready to ship

Then we will review and ready the products.

It will repeatedly for every time.

Several incremental releases called Sprints. Sprints usually takes from one to three weeks and you just keep repeating these sprints until your product feature complete. May be you end up shipping your products in Sprint 2, Sprint 3, or even further. But you eventually end up with a shipping product

**In SCRUM there are three key roles that are needed for the framework to work well.**

* 1. Product Owner: Product owner has the bright ideas turn into products
  2. SCRUM Master: Is a servant leader to the team responsible for protecting the team and the process running the meetings and keeping things going.
  3. TEAM: The team can be made up of developer’s testers, writers and anyone else that helps in building a product. Team members often play multiple roles, somedays developers may end up doing tests or testers may end up writing either way the team works to get the product done

**There are 3 artifacts or documents that are used in SCRUM**

1. Product backlog: The product owners create a prioritized list of features known as user stories that could go into the products. This list evolves and changes priority with every sprint
2. Sprint Backlog: Highest priority user stories go into the sprint backlog and are committed to the next sprint
3. Burndown Chart: Show the progress during a sprint on the completion of tasks in the sprint backlog. This chart should approach zero points as the work is being completed

**Three Ceremonies: 1. Sprint Planning 2. Daily SCRUM 3. Sprint review**

**SCRUM WORKFLOW:**

Product Backlog🡪Sprint Planning🡪Sprint Backlog🡪Sprint (1 to 3 weeks and daily scrum)🡪Potentially Shippable Products🡪Sprint Review🡪Retrospective (Repeat this workflow for each sprint)

**Software to manage the scrum workflow:**

**Utility** **software** has been built around the scrum process to help people filling the three scrum roles manage the 3 artifacts and better run the three ceremonies

**18 Cluster (OS Redundant)**

* A cluster is a group of servers and other resources that act like a single system and enable high availability and, in some cases, load balancing and parallel processing.
* Anything happening our hardware we will use redundant to fix this issue but for OS?
* We don’t have redundant option for OS. Now high availability cluster comes in to the picture.
* A high availability cluster is a group of computer hardware pieces that provide solution for redundant operation in the event of component failures.
* Clusters mostly for your OS bcos we don’t have a redundant for the OS till now without having the options of clusters.
* Clusters can be two node or three node clusters, but the maximum number of nodes in a cluster varies with the software; some clusters can have 20 nodes, while others can have 60 nodes
* Assume server A and server B is there, so we will merge server A and server B by using the cluster solution and we will configure the IP address on this cluster node.
* If anything happen to server A, The configured IP will be go to server B so still services are available and no downtime.

**There are two ways of clusters: 1. Manual Cluster 2. Automatic Cluster**

* Manual Cluster: We logged into other node, there all the down node data’s are available in this node and configured the IP address manually bring your services up. Here we have some downtime.
* Automatic Cluster: Here we use some sort of cluster software’s such as VERITAS cluster, HP Service-guard cluster, Red Hat Linux Native Cluster, Sun Cluster, IBM AIX based cluster.
* These software’s will do the necessary switchover automatically which is already configured by us in the respective configurations.

**There are four major types of clusters for our needs:**

* 1. High availability 2. Load Balancing 3. High Performance 4. Storage

1. High Availability: (Active-Passive Cluster)

A high availability cluster is a group of hosts that act like a single system and provide continuous uptime.

High availability clusters are often used for load balancing, backup and failover purposes.

Here only one active at a time

Disadvantages: Only one node is active so we need to spend money for passive node too.

1. Load Balancing: (Active-Active)

Load balancing scales the performance of server based programs, such as web server, by distributing client requests across multiple servers.

Here both server will be active, so user will get 200% utilization of server when both are active, if any one goes down so user will get 100% utilization.

1. High Performance:

To have high performance we use this cluster.

1. Storage:

Storage clustering is the use of two or more storage servers working together to increase performance, capacity, or reliability.

**19 KERNEL**

The kernel is a [computer program](https://en.wikipedia.org/wiki/Computer_program) at the core of a computer's [operating system](https://en.wikipedia.org/wiki/Operating_system) and generally has complete control over everything in the system.

It is the portion of the operating system code that is always resident in memory and facilitates interactions between hardware and software components

The kernel is one of the first programs loaded on [startup](https://en.wikipedia.org/wiki/Booting)

**20 API (Application Programing Interface)**

* API is a set of routines, protocols, and tools for building software applications. An API specifies how software components should interact. Additionally, APIs are used when programming graphical user interface (GUI) components.
* Different types of API: Facebook is a messaging API, for website some other API, YouTube some other API.
* Disadvantage: It will consume more data’s
* Protocols means set of rules, By using protocols we can create software’s, game, application

**21 ELK (Elasticsearch, Logstash Kibana):**

**Elasticsearch:**

* Elasticsearch is the distributed no sequel database which uses JSON like messages where you can store messages in the form of JSON across distributed system where you can scale across indexes.
* As the heart of the elastic stack, it centrally stores your data so you can discover the expected and uncover the unexpected.

**Logstash:**

* Logstash is an open source, server-side data processing pipeline that ingests data from a multitude of sources simultaneously, transforms it, and then sends it to your favorite “stash.” (Ours is elasticsearch, naturally)

**Kibana:**

* Kibana is nothing but UI component which displays the data in the way we wanted
* Elasticsearch is a separate component in a separate instance you will be having log stash which is streaming data from different machines or infrastructure and we have a kibana UI deployed over the elasticsearch instance where we can see different reports, different analytics

**22 Grafana:**

* Grafana server is the front end that actually visualizes the data.
* In order to get the data, Grafana queries the data source and the data source return the requested data based on the query and that data is then displayed on a Grafana dashboard

**There are 2 main categories in Grafana**

1. Metrics Ex: CPU Load or Current memory usage or outside temperature or cpu temp
2. Logs Ex: Timestamp, Error Info, Warning, Debug, Logs entry, History

**23 WEBHOOK**

* A webhook is a web callback or HTTP push API.
* It is a way for an application to provide other applications with real-time notification.
* It will allow us to automate the execution of some action based on the GitHub events
* Need to add webhook to Jenkins task to responsible for packing or deploying the service

**Build a WebHook in Github:**

* Go to your project repository
* Click “Settings”🡪“WebHooks”
* Paste jenkin URL here. It is the URL where our Jenkins is running (jenkin url last we need to add **/github-webhook/ Ex: hostname:8080/github-webhook/**
* Select content type (Either JSON or what?)
* Secret tab is used to secure your WebHook.
* Select which events (Just the push event, Send me Everything, Individual events) **🡪** “Add WebHooks”
* Now go to Jenkins create project under **item name** and under **source code management** tab select **GIT** and paste the **GitHub repository URL🡪** In **Build Triggers** select **GitHub hook trigger for GITScm Polling**

**024 BUILD TOOL & MAVEN**

**Build Tool:**

* Build tool is essential for the process of building. It is needed for the following processes.
* Its generating source code
* It will generating documentation from the source code
* Compiling of source code
* Packaging of compiled codes into JAR or ZIP files
* Installing the packaged code in local repository, server or central repository.

**MAVEN:**

* Maven is a build tool that helps in project management. The tool helps in building and documenting the project.
* Maven is written in JAVA and C# and is based on the project object mode (POM)
* POM is an XML file that has all the information regarding projects and configuration details.
* When we tend to execute a task, Maven searches for the POM in the current directory.
* The Maven tool is used to build and manage any JAVA-based project. It simplifies the day to day work of JAVA developers and helps them in their projects.
* It helps in downloading dependencies, which refer to the libraries or JAR files used in the project.

**The Problem That Maven Solved:**

* Getting right JAR files for each projects to avoid conflict
* Helps to create the **right project structure** which is essential for execution
* Building and deploying the project to make it work.

**What MAVEN Does?**

* It simplifies the process of building the project by automatically
* APACHE MAVEN helps manage all the processes, such as building, documentation, releasing and distribution in project management.

**Maven Installtion:**

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

**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)**

**25 Sonarqube**

**Static code analysis**

* SonarQube is a Code Quality Assurance tool that collects and analyzes source code, and provides reports for the code quality of your project. It combines static and dynamic analysis tools and enables quality to be measured continually over time.
* SonarQube offers reports on [duplicated code](https://en.wikipedia.org/wiki/Duplicate_code), [coding standards](https://en.wikipedia.org/wiki/Programming_style), [unit tests](https://en.wikipedia.org/wiki/Unit_testing), [code coverage](https://en.wikipedia.org/wiki/Code_coverage), [code complexity](https://en.wikipedia.org/wiki/Cyclomatic_complexity), [comments](https://en.wikipedia.org/wiki/Comment_(computer_programming)), [bugs](https://en.wikipedia.org/wiki/Defensive_programming), and security recommendations.
* SonarQube can record metrics history and provides evolution graphs. SonarQube provides fully automated analysis and integration with [Maven](https://en.wikipedia.org/wiki/Apache_Maven), [Ant](https://en.wikipedia.org/wiki/Apache_Ant), [Gradle](https://en.wikipedia.org/wiki/Gradle), [MSBuild](https://en.wikipedia.org/wiki/MSBuild" \o "MSBuild) and [continuous integration](https://en.wikipedia.org/wiki/Continuous_integration) tools ([Atlassian Bamboo](https://en.wikipedia.org/wiki/Bamboo_(software)), [Jenkins](https://en.wikipedia.org/wiki/Jenkins_(software)), [Hudson](https://en.wikipedia.org/wiki/Hudson_(software)), etc.)
* Once the SonarQube platform has been installed, we’re ready to install a scanner and begin creating projects. To do that, you must install and configure the scanner that is most appropriate for your needs. Do you build with Maven – use the [SonarScanner for Maven](https://docs.sonarqube.org/latest/analysis/scan/sonarscanner-for-maven/)
* Go to that download sonarqube folder / go to bin / then select appropriate OS / then run startsonar / login with default username & password then need to change in 1st login
* SonarQube reduces the risk of software development within a very short amount of time. It detects bugs in the code automatically and alerts developers to fix them before rolling it out for production.
* SonarQube developed by Sonar Source
* Sonarqube default port is 9000 / localhost:9000

#Quality mgmt tool # Consolidated report # Community / Enterprise version # Java based application # Extension using plugins

# Quality of the software # Management of report # Forecast Quality improvements # Dashboard based report

# Route cause identification

# 3 Main components in sonarqube is Analyzer or Scanner , Web UI, Database

# Working Architecture: Device(Laptop)(Installed Sonarqube)🡪Scanner(It will analys scan and check the code)🡪Configuration Property File 🡨 🡪Set of rules🡪Web Ui🡪Database

**Sonarqube Integration With Jenkins:**

**In Sonarqube server**: We need token to authenticate from Jenkins

We will copy code from sonarqube server itself under sonarscanner for Jenkins, sonarscanner for maven, etc..

**Click Add a project🡪We can create a project manually or integrate with version control system**

**Enter project key & Display name (Nothing but project name)🡪Provide a Token Name & click generate**

**In Jenkins server**: 1. Install sonarqube plugins 2. Configure Sonarqube Credentials 3. Install SonarScanner 4. Run pipeline job

1. Install sonarqube plugins

**Manage Jenkins--> Manage Plugins🡪Available🡪Search Sonarqube Scanner🡪Install**

2. Configure Sonarqube Credentials

**Manage Jenkins🡪Manage Credentials🡪Jenkins(Here we need to add credential)🡪Global Credentials (Unrestricted)🡪Add Credential🡪Kind (Choose secret test)🡪In secret paste that sonarqube tokencode🡪ID (Give some name)🡪Ok.**

**Manage Jenkins🡪Configure System🡪Sonarqube Server (Enable Environment Variable)🡪Add Sonarqube (Name, server URL)🡪Server Authentication code(Select Already Added)🡪Apply & Save**

**Install**

**Code coverage calculate / Code quality /ans: unit test //**

**NAT (Network Address Translation)**

**NAT (Network Address Translation)** is a process of changing the source and destination IP addresses and ports. Address translation reduces the need for IPv4 public addresses and hides private network address ranges. This process is usually done by routers or firewalls.

There are three types of address translation:

1. **Static NAT** – translates one private IP address to a public one. The public IP address is always the same.
2. **Dynamic NAT** – private IP addresses are mapped to the pool of public IP addresses.
3. **Port Address Translation (PAT)** – one public IP address is used for all internal devices, but a different port is assigned to each private IP address. Also known as **NAT Overload**.

**Cloud Computing**

* Cloud computing is on-demand access, via the internet, to computing resources—applications, servers (physical servers and virtual servers), data storage, development tools, networking capabilities, and more
* hosted at a remote [data center](https://www.ibm.com/in-en/cloud/learn/data-centers) managed by a cloud services provider (or CSP). The CSP makes these resources available for a monthly subscription fee or bills them according to usage.

**Cloud Service Models**

* Service models are the reference models on which the Cloud Computing is based. These can be categorized into three basic service models.

**Infrastructure as a Service (IaaS):**

* IaaS provides on-demand access to fundamental computing resources–physical and virtual servers, networking, and storage—over the internet on a pay-as-you-go basis. IaaS enables end users to scale and shrink resources on an as-needed basis, reducing the need for high, up-front capital expenditures or unnecessary on-premises or ‘owned’ infrastructure and for overbuying resources to accommodate periodic spikes in usage.
* Ex: Amazon Web Service, Microsoft Azure, Google Compute Enginer

**Platform as a Service (PaaS):**

* With PaaS, the cloud provider hosts everything—servers, networks, storage, operating system software, middleware, databases—at their data center. Developers simply pick from a menu to ‘spin up’ servers and environments they need to run, build, test, deploy, maintain, update, and scale applications.
* Ex: Google App Engine, AWS Elastic Beanstalk, Adobe Commerce

**Software as a Service (SaaS):**

* SaaS—also known as cloud-based software or cloud applications—is application software that’s hosted in the cloud and that you access and use via a web browser, a dedicated desktop client, or an API that integrates with your desktop or mobile operating system. In most cases, SaaS users pay a monthly or annual subscription fee; some may offer ‘pay-as-you-go’ pricing based on your actual usage.
* Ex: Gmail, Ms office 365

![Graphical user interface, application

Description automatically generated]()

**Cloud Deployment Models**

Public Cloud:

* Allows systems and services to be easily accessible to the general public. Public cloud may be less secure because of its openness ex: email

Private Cloud:

* Allows systems and services to be accessible within an organization. It offers increased security because of its private nature.

Community Cloud:

* Community cloud allows systems and services to be accessible by group of organizations.

Hybrid Cloud:

* The Hybrid cloud is mixture of public and private cloud. However, the critical activities are performed using private cloud while the non-critical activities are performed using public cloud.

**My Explanation & Promise**

I’m ready to work with anykind of DevOps technologies like Git, Jenkins, Maven, Kubernetes, Dockers, Docker Swarm, or EKS

Even if I don’t have an hands on experience I will manage and I will be expert very soon bcos i like devops

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.

Hopefully, I answered the majority of your questions and I hope you are satisfied to take on this project as well

I look forward to your feedback and am eager to expose my talent as well as learn some new tools and technologies

**On-Premises or Cloud:**

I worked on-premises server i-e, Ansible, but I have knowledge on AWS, so if I get chance to work on AWS it’s perfectly ok for me.

**SWOT (Strengths, Weaknesses, Opportunities and Threats)**

**Strengths:**

**Team Strengths:**

**Teams have people having different areas of expertise allowing them to take care of problems much more universally. Not only does it improve efficiency, it improves the quality of the work being done.**

**Teams are able to work much faster than individuals to meet deadlines earlier because many people are ticking points of the to-do list.**

**Teams offer emotional support, friendship and a sense of belonging to all the people who are a part of them thereby increasing their motivation and dedication towards organizational success.**

**Personal Strengths:**

**Can learn quickly / Have an analytical mind / Responsible and organized / Try to do my best / I like to be busy, I prefer having a lot of things to do than have free time / I am never late, I rather be waiting than be waited**

**Team Weaknesses:**

**People within teams can get competitive which defeats the entire purpose of**[**team building**](https://pestleanalysis.com/5-challenges-sales-teams-2021-how-to-overcome/)**. Their own goals are more important to them than the teams as a whole.**

**Not every team has the right people; not enough people, not the right fit with each other, not the right group of talents etc. This kind of takes away from all the strengths we talked about earlier**

**Personal Weaknesses:**

**I am little shy / I am not interested in politics / I prefer doing things to my own /**

**Opportunities:**

**Get involved in others work and cooperate with them, so we all learn from each other and don’t fight to be the best / Make an effort and watch the news**

**Threats:**

**Important resources are lost when people leave teams including all the training hours and the sensitive information sharing.**

|  |  |
| --- | --- |
| **Basic Linux Command** | |
| **Location** | **Description** |
| /root | Home Directories |
| /root/home/username | User Home Directories |
| /bin or /usr/bin or /usr/local/bin | User Executable files like **ls, pwd, cat, etc (This is normal user)** |
| /sbin or /usr/sbin or usr/local/sbin | System Executable files like **adding user, install software, rpm command (This is system admin command)** |
| /media or /mnt | If we connect any additional USB drive or CD, It will automatically mount here |
| /etc | All the configuration will execute here Ex: User configuration, Network Config and Server Config etc.. |
| /tmp | If we want to keep temporary file we will keep here, But after reboot it won’t be there so be caution |
| /boot | Kernels and Bootloader |
| /Var or /srv | Server Data Ex: Web server file we will be here |
| /proc or /sys | System information will stored here |
| /lib or /usr/lib or usr/local/lib | Shared Libraries |
| Read the command prompt username@hostmane$ **($ mean normal user**) Ex: faiz@desizors$ / If **# it’s represent root user //** If you see this **~ it’s represent we are in home directory** | |
| **Commands** | **Description** |
| ifconfig | To display IP Address and broadcast ID detail To display IP Address and broadcast ID detail |
| date | To display the date and time |
| Uptime | It will show the uptime |
| Cal | To display the calendar |
| whoami | To know the logged in user name |
| who | How many users are logged in |
| top | To know the what are the processes are running / CPU, Memory utilization / How many users are available / PID (Process ID) |
| Kill -9 PID | To stop the process Ex: kill -9 101 (this will kill the process ID 101) / We will use this when more slowness issue or more utilization time |
| Ps or ps –ef or ps –ef | grep any | It will show all the process with all details / by using grep we can search only particular process or any other details. |
| q or ctrl+c | To exit from the prompt |
| Clear shortcut is ctrl+l | It will clear the page and it will go to 1st line. / shortcut for that is ctrl+l key |
| free –m | It will show total RAM size, used and free |
| Pwd | It will show the present working directory |
| history | It will show all the commands we used early |
| Man | Man command is help to give the full details (Ex: Man pwd) It will explain the full details of pwd command |
| Uname | It will show which OS we are working on |
| Sudo passwd root | This will create a password for root |
| Ls | List command |
| Ls -l | Ls – l (List the command with full details) |
| Ls-a | it will show the hidden file |
| Useradd –m name | It will create a user (Ex: useradd –m faiz) –m menas home directory creation |
| -m | Home directory creation for users Ex: useradd -m |
| Sudo -i | It will go to root or admin |
| Su | Switchuser (Ex su faiz) |
| Reboot | Reboot the system |
| Halt | Brings the system down immediately |
| Shutdown | Shutdown the system |
| Cd ~ (tilt symbol) | Cd(Change Directory) It will useful to go to user directory folder Ex: cd ~ of user faiz,  It will directly goes to faiz, to check pwd command (Doubt if multiple users) |
| Cd .. / cd../.. | cd.. It will go to previous directory / cd../.. it will go to previous of previous folder |
| cd / | It will go to starting path |
| Mv | Rename file name or move the file Ex for rename: mv filename new filename / mv faiz.txt renamed.txt  (Now filename was changed from Faiz.txt to renamed.txt) Ex for move: mv filename faizul/(folder name is  faizul) / mv faiz.txt Faizul/ (It will move to Faizul folder) / Same methods for moving or  renaming for folders too |
| Wc | Word count of the file Ex: wc filename / wc faiz.txt / 1st column is no of lines / 2nd column is no of words / 3rd column is no of letter+line (Ex: I am Faizul / (Letters is 11 + 1 line total 12) /  4th is filename |
|  |  |
| **File Management Types** | |
| **1.Normal Files (Image, Text, Mp3), 2. Directories (Folders) 3 Special Files (Hidden, System related, link) / Hidden files b4 dot will be there** | |
| Mkdir | It will Make directories Ex; mkdir ffshz |
| CD (Directories name) | Ex: CD ffshz (It will go to ffshz directory) |
| Touch (File name) | To create a empty file Ex: Touch expense (It will create a empty file and file name is expense) |
| Cat | Cat is used to see the content of the file (Ex: cat expense) |
| Rm / rm \* | Remove (it will delete the file) Ex: rm expense / Ex2: rm faiz.txt work.txt (It will delete  multiple files) / Ex3: rm expe\* (It will delete all the files name starting with expe) |
| Rm | Remove file from one directory location to another directory location  Ex: rm faiz/(destination folder name)expense.txt(filename) / rm faiz/expense.txt |
| Rmdir | It will delete directory Ex rmdir utest2(directory name) |
| Rm –rf | Rm –rf ffshz (If directory have some files then rmdir ffshz won’t delete the directory bcos  inside expense file is there) So if you want delete then use rm -rf ffshz (now it will delete)  f represents force / Forcefully deleting |
| Cp | Copy command / Ex: cp expense expense backup (Cp sourcefilename destinationfilename) In case if you want to change anything in expense b4 we can take the backup and make it  any change / This will copy to the same folder / Destination file name should be differ |
| Cp | This example is for to copy a file to different folder cp expense.txt(filename) faiz/  (folder name) (It will copy that file here) or cp expense.txt(filename) faiz/(foldername)  expense2.txt / It will copy and paste that file expense2 name |
|  |  |
| **VI Editor is useful for editing in linux / In windows we will edit or create any excel files** | |
| Vi | 2 Modes (edit and escape) Text Editor (Ex: Vi expense) It will create new file and  It will go to edit mode, There we can type what we want: Ex: Welcome to DESIZORS |
| I | Insert mode |
| :wq | In VI editor use this command to save and exit |
| :q! | Without save and exit |
| dd or d8d or 8dd | To Delete the single line press 2times dd, If you want to delete more lines then press d8d or any required number. |
| Yy | To copy the line |
| P | Paste |
| U | Undo |
| :s | Single String replace Ex: :s/car/truck or :s/car/truck/g (This will replace only one string) /  case sensitive is there, so if you want to replace (Letter F) Need to type F,  If u type f it won’t replace |
| /string or ?string | Ex /faiz (it will search faiz from top) / ?faiz (It will search from bottom) |
| :%s | String replace Ex : :%s/car/truck/ or :%s/car/truck/g (This will replace entire string) |
| Grep string filename | To search a file or a string in a file / Ex: grep welcome faiz.txt (It will search inside the faiz.txt file and if found it will display else not found) |
| Ls -l | grep foldername | To search the folder name or file name starting from fold (ex folder1, folder2, folder3  folders are available) So it will show all that three Ex: ls -l | grep fold  (Need to use pipeline symbol not exclamatory) |
|  |  |
| **Sort Command is used for list in alphabetic or numeric or Ascending or descending order (n represent numeric, r is reverse, f is \*\*\*)** | |
| Sort filename | It will display special characters 1st, numbers 2nd (1,11,2,25,3) 3rd Capital Letter in  ascending order and 4th small letters in ascending order |
| Sort –n filename | **It will show numbers in small to big**, It will display special characters 1st,  2nd Capital Letter in ascending order,  3rd small letters in ascending order and 4th numbers in small to big (1,2,3,11,25) |
| Sort –r filename | It will display in reverse (1st small letters in descending, 2nd capital letter in descending, 3rd numbers(3,25,2,11,1)  and 4th special characters |
| Sort –f filename | Upper case and lower case together (Ex: A,a,am Faizul, B, C,c,cat…) / 1st special character,  2nd number and 3rd upper case and lower case together |
|  |  |
| **Folder Permission** | |
| There are three permissions are there 1. Read 2. Write and 3. Execute (RWX) (R value is 4, W value is 2, X value is 1) | |
| When you use (ls -l filename) It will show some details like (–rw-rw-r—1 myfaizul sdteam 0 apr 21 10:20 faiz.txt)  myfaizul is owner permission (rw-) / sdteam group permission (rw-) / 0 others permission (r--))  (r - read value is 4 / w - write value is 2 / e – execute value is 1) Total 7 In the list first letter is – (Its is file)  If 1st letter is D (Its is directory) | |
| Chmod | Chmod is used to change the folder permission details: Ex1: Owner only need access ,  groups and others don’t want access Then chmod 700 faiz.txt / Ex2: need full permission  for owner, groups read and write, others only read so chmod 764 |
| **Sudo privilege (To install anything user need the sudo privileges)** | |
| Su - | It’s indicate switch user to root / type the root password |
| Usermod – aG sudo  username | Ex: usermod –aG sudo myfaizul (aG is append group) |
| **Now myfaizul user got the sudo privilege / To check the group of user command is groups username /**  **If you see root / then successful** | |
| Groups username | Ex: groups myfaizul |
| **To install python in Ubuntu** | |
| Sudo apt update -y | Need to update by using this command |
| Apt list | It will show all the packages of updated list |
| Sudo apt install software-  Properties-common | After this |
| Sudo add-apt-repository ppa:  Deadsnakes/ppa | Again enter to continue |
| Sudo apt install python3.9 | It will install |
| Python3.9 –version | It shows the full version of the python Ex:python3.9.12 |
| Ls –l /usr/bin/python\* | It shows the all the versions of python |
| Sudo apt update | It will update after entering the password (faizfaiz@58) |
| Sudo apt install gedit | It will install gedit |
| Sudo apt install xclip | Use to copy & paste with the terminal |
| Cat filename | xclip –sel clip | Use to copy & paste with the terminal |
|  |  |

|  |  |
| --- | --- |
| how do you freestyle | customized pipeline code / groovy |