Road Map for Deveops

Linux OS Ubuntu or CentOS🡪 Knowledge about OS,

Git HUB 🡪 Source code management or Version Control System

Mavbe 🡪 Build Automation Tool,

Shell/ Python 🡪 Scripting knowledge,

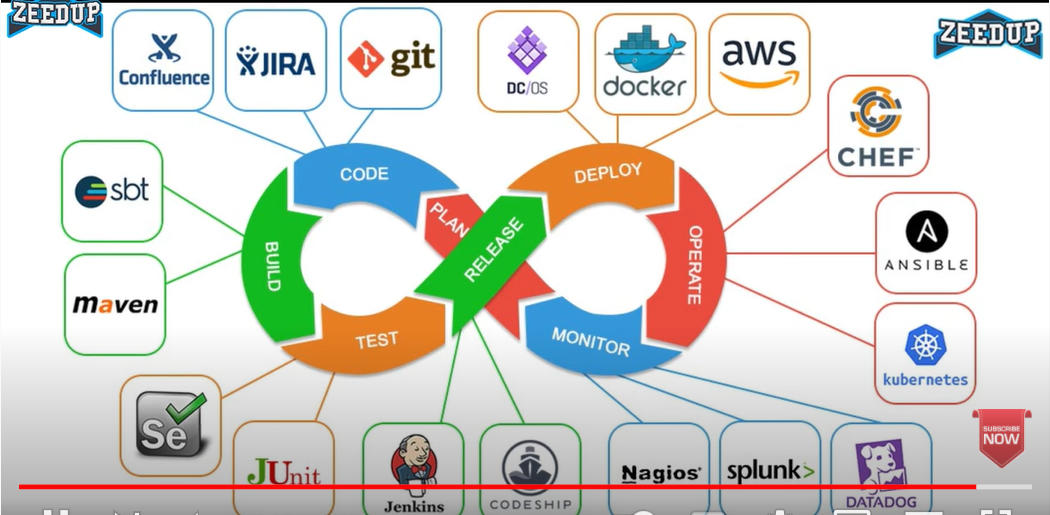
Jenkins / Bamboo 🡪 CI/CD tools,

Ansible/ chef 🡪 Configuration management tool,

AWS/Azure 🡪 Cloud storage tool,

Terraform/ Cloud Formation 🡪 Infra as a code,

Kubenetes 🡪 Container Orchestration.



What is devops and Purpose of devops ?

Devops is a culture, the combination of development & operations. It includes lot tools to deliver an application in a faster way without any pause or any glitches.

It provides communication, integration, automation, and close cooperation among all the people needed to plan, develop, test, deploy, release, and maintain a Solution. DevOps is part of the Agile Product Delivery competency of the Lean Enterprise

Deveops – GIT HUB

Storing a code in git hub and deploying in a server, so that we can access the website in internet – Manual procedure of doing it.

CICD – Continues integration and continues deployment – Automating the code or updating the code online is called CICD. Instead of deploying manually, code can be automatically deployed using CICD functionality.

Jenkins -

Who arranged for interviews call,

https://drive.google.com/drive/mobile/folders/1NutVlzffKdfaNxa07evP86xq16z3cJEp

LINUX:

Linux is the command line operating system.

All the operations are done through command.

Directories – folders in linux

Normal files

Special files

Commands: CAL – Calendar, whoami, who, halt – brings the system down immdtly, shutdown,

reboot.

Pwd is command will tell you on which path or which location we are in.

Ls will show the folder names and file names. Folder name will be in blue color and file name will be in white color.

Ls –l will display the detailed informations of folder and files.

Ls –a will disply the hidden files in the folder.

Vi file name will create a new file

After creating a file we are allowed to enter the txt using the letter “i” and “esc button” will stop or will not allow editing anything in that file.

:wq will help us to save the file and return to the command page.

Cat filename will display the text we have saved in that particular file.

:q! will not save the newly edited txt or file, it will restore the old txt in the file

Using key strokes we can do the following actions in linux OS

Keystroke is nothing but the letter “K”

Keystroke plus yy will copy the current line

Keystroke plus d will delete the entire lines

Keystroke plus d3d will delete from 3rd line

U will undo the previous command

/”Word” will find the particular word in the file

If we r in last line and need to find a word means need to give esc click in “?” and give the word

“:s/good/bad/g” this command will replace the word from good to bad

Cp n.txt to m.txt, this command will copy the entire file from n file to m file

Wc file name will display the word counts in that file

Rm file name will remove the particular file from the folder

Rm file file file name will delete the multiple files from the folder

Mkdir will create a new folder

To create an empty file need to use “touch” plus filename.txt

“rmdir foldername” will delete the folder if it is empty

“rm –rf foldername” will delete the folder even if it has a file in it

“mv foldername1 to foldername2” it will rename the old name to new name

“cd ..” will take u to the previous folder and “cd ../.. will take u back to the two folders like two directories back.

We can give permissions to the owner, group and others

Read has a value of 4, r – read (4)

Write has a value of 2, r – write (2)

Execute has a value of 1, x – execute (1)

Chmod 777 “file name”

777 means give full access to the file name like, read, write and execute.

Grep will search a file or string in a file keyword “grep string filename”

“Sort filename” will sort the file name in ascending order

“sort –n filename” will sort the file in numeric order like 1,2,3

“sort –r filename” will sort the file in the reverse order

“sort-f filename” will sort the file both the upper and lower case together

System related and process related commands

Top command is used to find the number of systems or process in the particular system

If we need to delete a system need to use the command “kill -9 and enter the PID number”

“ps –ef” command will display the process running in the systems

“ps –ef | grep process name” process name in the sense if we need to know python process or java process, we can get from the above command

\*\*\*\*\*\*\*\*\*\*\*\*LINUX BASICS OVER\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Mr devops

Rajesh devops

YB Mathu suthana reddy

Linux, AWS Cloud VPC EC2 S3 Cloud watch lamda , Git & GitHUB, Maven, Jenkins, Anisable, Terraform, Qubernets, Dockers

**GIT and GIT HUB:**

GIT and GIT HUB is code repository which is used to store the codes. From here we can directly upload the code to the server.

GIT or GITHUB usually work under VERSION CONTROL SYSTEM. It is a repository which is used for multiple people to work on the same file at the same time. Here we can find who made the changes and also can switch back to the older version.

Each machine act as a repository, if there is any glitch in server. Each machine has an original backup which will push the code back to the server. So that there is no lag in the project or process.

Once installed we need to config the git with following commands: **GIT Commands**

* Git config - -global user.name “manu”
* Git config - -global user.gmail [manu@gmail.com](mailto:manu@gmail.com)
* Git config –list
* Git –help
* Git init – will create an empty repository
* Ls –la – will display complete details of the folder
* After config very first command in git is “**gitinit**”
* Next one is “**mkdir gitclass**” and “**git status**”
* “**Touch filename”** will create a new file with dummy
* Once the file is added, when we are cking the status that particular folder will be untracked. So to add the file to tracked on use the command as “**git add filename**”
* “**git add .”**  this command will add the edited file to the repository
* “**git commit –m “add some words here”**” this command will commit the added files to the repository
* “**git log”** it will display what are the insertion happened in this commit
* “**git log –author=”username”**” this command will display the commits done by particular user like author in the sense the person who logged in
* “**ls –lrt”** will display only the files in the folder
* **“ls-la”** will display both the files and hidden files also.
* Whenever we are entering new text to the existing file, without giving the command “git add .” if we give “git status” it will show error. We can get the diff using the command “**git diff**”.

**Git – It is a version control management tool**

**Github – It is a Centralized repository**

**GITHUB Repository connection of GIT and GITHUB:**

* There is a green color “code” button will be available, copy the URL from there
* And use the command **git remote add origin “url”**
* **Git pull** command will link from main repository to the local
* **Git pull origin master** will fetch the branch master
* If we get error like “**fatal: refusing to merge unrelated histories**” in the above command we need to use this command to resolve the error **“git pull origin master --allow-unrelated-histories**”
* First we need to connect local repository and server repository using the command “**git remote add origin + url of the server repository**”
* Then we need to git commit command i.e., “**git commit –m “command of commiting”**” It will commit in the local repository
* “**git add .**” will help u to add the new deletion or addition of the folder or file
* “**git status**”this will show file which is recently added or deleted
* “**git pull origin master**” will load the previous folder from server repository to local one or if we directly create a file in server repository will also send to local one by using pull command
* “**git push origin master**” will push the edited file or folder to the server repository
* So far we have worked in the “**master**” branch, now we are going to add a new branch
* When we are pushing our code to master existing code or entire project will collapse
* To avoid it each developer need to create their own branch with their name, once the push was done to their branch or repository. After verifying the code it will be merged or pushed to the master branch
* By creating the new branch it will show the new files and existing files also
* But when we are creating a new file in new branch will not reflect in the older or previous branch. Branch is also known as pointers.
* “**git branch branchname**” will create a new branch
* After creating a new file in the branch have to be committed as usual
* “**git branch**” will tell u which branch we are actually in
* “**git checkout branchname**” will switch from one branch to another branch
* “**git merge branch1 master”** in this command we are merging the files from “**branch1**” to the “**master**” i.e., folders from branch1 will be copied to the master branch
* **“git revert branch1 master”** this straight opposite to the merge option it will remove or delete the merged file and return back to the same status of the folder or branch or pointer
* After doing “**merging**” & “**reverting**” actions need to give push command i.e., “**git push origin master**” after giving push command we can see the changes in the centralized server or repository
* “**git reset head <filename>**” this command will reset the command “**git add .**” after editing any file we will give git add . Command to add that particular file to the local machine or local repository. So to remove from local repository we need to give the “**git** **reset head <file name>**” command
* “**git branch –m oldfilename newfilename**” it will rename the files. Eg name of the branch1 will be rename to branch2
* “**git branch –d branchname**” this will delete file from the repository
* “**git stash”** this command will commit the existing code, which will not save the newly added or edited txt to the particular file. **FYI, newly added text will not exist in that file** without giving git add. Command, we can commit using “**git stash** is ntg but **q!**”
* “**git tag tagname**” it is something called creating a clone of the repository
* “**git tag –d tagname**” will delete the particular tag from the git
* “**clear**” will clear all the command we entered and take u to top of the page
* “**git clone url**” will help us to clone from the repository which is created in the github directly without git bash interaction
* In github we may get some problem in coding, in that case we need to revert the previous code with the command “**git revert –commit <commit id#>**” this command will take us to the previous stage of the program, incase if we get any problem in the code
* “**git revert –commit <commit id#>**”, this command will revert the commit based the commit id which we are giving, from the id to previous everything will be reverted
* Very first thing we need to do in github is need to create an organization

**JENKINS:**

* Jenkins will be connected to the GITHUB, which will convert the code to JAR or WAR file by writing some codes
* It will also deploy in the tomcat server and do the testing, like unit testing and other things, if all the test cases are passed, the code is ready to deploy in the real server
* In Jenkins, continuous integration and continuous deployment will take place after every commit or build (development of the code or application development)
* **Continuous integration:** It helps to catch the issues faster and resolve in easier way and it will automate the build CI will leads to CD in a rapid manner, it is a framework that achieves the work through plugins and it supports around 1k plugins
* In CI the codes are continuously added as a pipeline and in delivery the codes are ready for the production and finally it will be deployed as a CD, this is CI/CD pipeline.
* Particularly in CI Jenkins there will be two testing takes places they are **Unit testing and Integration testing**
* Continuous delivery is the testing environment, here two testing take places they are acceptance testing and load testing, where the codes are ready to deliver
* **In** Ubuntu linux OS need to enter the command as “**sudo apt install default-jre**”, to check whether java is installed or not use command “**java -version**” in cmd or Ubuntu work space and “**javac –version**” command will specify the version i.e., 1.0.8.8 like that
* After that need to g it as install Jenkins in Ubuntu click on “**Debian/ubuntu**” in that page, then select the code from “**Long term support release**” and paste it in the Ubuntu, it will respond as “**ok**”
* Under “**Long term support release**”, there will be three codes left, need to enter one after another and it includes Jenkins install need to give “Y” denotes **YES**
* In browser need to give localhost:8080 it will ask for password, need to get the password by selecting the link and paste in Ubuntu after “**sudo cat link**” it will give the password, copy and paste it
* Same as windows need to select the option “**install the selected plugins**”
* “**localhost:8080**” login to Jenkins and create a build using following options **click on create job or new item in the dashboard -> click on freestyle project for practice -> scroll down deep under the build click dropdown & select Execute Windows batch command**
* Inside the box type as **echo “build is created”** type any text which need to be executed
* It will take you to the dashboard by clicking on “**Build Now**” option it will create a build or job, can see under “**Build History**” #1 will be created & this is the build
* Created **build** using Jenkins and also **pipeline** was created in the pipelines check the administrator or login admin localhost:8080 for more info
* In Jenkins there are two types of pipelines they are “**Declarative pipeline**” and “**Scripted pipeline**”
* **Declarative pipeline ->** It is a new feature that supports the pipeline as code concept, it made pipeline code easier to read and write

**SampleDeclarativeJob: Program**

Pipeline

{ agent any

stages {

stage(‘build’){

steps{

echo “build the code”

}}

stage(‘test’){

steps{

echo “test the code”

}}

stage(‘deploy’){

steps{

echo “deploy the code”

}}

}}

* **Scripted pipline ->** It is a traditional way of writing the pipeline in Jenkins, it is easy to write a pipeline compare to declarative pipeline and it helps for developers to write a complex pipeline as a code

**SampleScriptedJob: Program**

node (‘master’) {

Stage (‘build’){

echo “build the code”

}

Stage (‘test’){

echo “test the code”

}

Stage (‘deploy’){

echo “deploy the code”

}}

* **Master & Slave Nodes in Jenkins:** It help us to distribute the jobs from master to slave, which means master will allocate the work or information from master will be send to all the slaves
* Slave will act as a executors, jobs assigned from master will be executed and send the results to the master
* It will launch in JNLP or SSH method, to execute task or jobs, Can schedule the time to start the execution.
* LABEL, group of jobs added in a single label so that multiple task will execute at the same time
* JNLP is nothing but “**Java network launch protocol”** which will run the programs in the windows server or windows OS
* SSH is nothing but “**Secure Shell**” which will run the programs in the linux server or OS
* Always we need to use “**Master & Slave”** Architecture, always need to execute the Jobs from “**SLAVE”**

Service-oriented architecture (SOA)- This is the old version, like in java all the page for eg login page after that home page all the pages are, so if there is a big billion day. There will be a multiple logins like lakhs and crs, at that time there will be glitches or chaos will happen so to avoid this MICRO SERVIES introduced.

MICRO SERVICES: In this every page will be act or perform as a diff application, it will deploy in the docker, which will have containers (coz it is lit weight), these containers will next moved to kubernetes orchestration. In this kubernetes orchestration we have an auto-scaling concept that will increase the size based on the volume of the logins. It will increase the size if there is a huge login and reduce the size when there is less logins.

CLUSTER: The size increases and decreases based on the concept called cluster of node or cluster of server.

**MAVEN: Valaxy 01:12 starts**

* It is used to manage the build, testing, reporting, software configuration management, releases and distributions
* It provides many plug-ins and add-ons which is more powerful
* Before maven, we had a previous tool called “**ANT**”, this requires lot of configuration
* Maven uses “**Convention**” over “**Configuration**”, which means developers are not required to create a build process themselves
* “**pom.xml**”, stands for “**Project Object Model**”, It is a fundamental unit of maven
* **POM** contains goal and plugins, while executing a task or goal, maven looks for POM in the current directory
* Group ID will define the no of projects under one group ID, for eg calculator is a group id or name, whereas SUBPROGRAM like Add, sub, mul, div all are projects under one group id calculator
* “**artifactid**” is something to identify the projects or sub-projects easily or in a unique way
* “**Version**” is our wish we can give lijke 1.0, 2.0, 2.1, 3.3 its depends on our projects
* If we need to pull any project we need to use notation like “**groupid:artifactid:version**” which is nothing but “**Packaging**”
* There are three build-in build life cycles, “**Default, Clean & Site”**
* Default -> Handles the Project Deployment, Clean -> Handles the Project Cleaning and Site-> Handles the Creation of your Projects Documentation
* Steps involved in maven projects are, **Compiling the code or application**, **run the test cases**, once the test cases done. Need to create **JAR or WAR** file
* After creating a project, we need to run the file as “**Maven Build**”, there we need to give goals
* **Goals** are based on maven build lifecycle, three lifecycles are, “**default, clean & site”**
* Most commonly used goals are, **“clean, validate, compile, test-compile, test, package, verify, install, deploy”**
* Before creating “**Package**” Need to run a “**Clean**” goals to run the recently updated code
* It will clean all the previous executed goals and run all the codes from first if we use the goals “**clean package”**
* Steps for developer connecting GIT HUB to Maven. Create a maven project in eclipse and create a repository in GIT HUB
* In the “**Maven**” project folder open the git bash and create local repository, now do all the basics things like creating user.name, user.email. Later connect local repo to the server repo in the maven project folder
* Now as DevOps Engineer we need to connect maven project to the EC2 instance. Create an EC2 instance, using putty. Copy the public IP and browse the key or can create new key
* Login as “**ec2-user**” & use “**sudo su -**” to become a root user it’s mandatory
* Now need to install java in putty using “ **yum install java-1.8\* ”**
* **“whereis java”** will give you the path where it is installed
* **“ find /usr/lib/jvm/java-1.8\*|head -n 3”** to check the java home path
* Go to “**vi** **~/.bash\_profile**” and set the path as **JAVA\_HOME=”**PATH address which is copied from the address **whereis java**”, then in path mention **:$JAVA\_HOME**
* Then check the path using command “**echo $PATH**” it will not be updated then give the command as “**source ~/.bash\_profile**” and now echo $PATH command will work
* After this need to copy the maven link from maven.org in this page copy the “**tar.gz**” link
* Go to the folder “**cd /opt**” in linux page copy the link after **wget** [**https://dlcdn.apache.org/maven/maven-3/3.8.5/binaries/apache-maven-3.8.5-bin.tar.gz**](https://dlcdn.apache.org/maven/maven-3/3.8.5/binaries/apache-maven-3.8.5-bin.tar.gz)
* After this it will be installed in linux EC2 instance, then need to extract using the command “**tar –xvzf apache-maven-3.8.5-bin.tar.gz**” need to enter in putty
* Need to enter “**ls**” copy the **apache-maven-3.8.5** and change the folder using **cd apache-maven-3.8.5**
* Give pwd and copy the link later go to “**vi ~/.bash\_profile** ” and enter this command in that page as “**M2\_HOME=COPIED LINK**” & “**M2=COPIED LINK/bin**”
* Later in path add **:$M2:$M2\_HOME**
* Give mvn –version to check which version of maven has been installed
* Need to install the git like “**yum intall git**”
* Then connect git hub with ec2 server using “**git clone** [**https://github.com/manukohi/demo\_project.git**](https://github.com/manukohi/demo_project.git)”
* When entering the command ls we will have pom.xml and src file
* When we give the goals as “**mvn validate**” we will get a output as build success, finally got output
* There won’t be any target file, so need to give “**mvn compile**” it will generate a JAR file in local repo and now when we enter ls it will have “**Target**” & cd target
* Ls we will find classes, now enter “**cd classes**” there will be a “**com**” now get into com by “**cd com**” & ls now we will have demo, the name we given initially in git hub, “**cd demo**” and ls we will have our repository “**cd demo\_project**” and now enter
* “**mvn test-compile” & “mvn package”** and later “**find / -name .m2**”

**ANSIBLE:**

* It is an open source IT Configuration Management, Deployment & Orchestration tool
* It will configure 10k or 20 k servers at the same time
* It is used for application deployment, continuous testing for existing applications, provisioning the storage, orchestration and automate the tasks
* **Ansible is used to install software, up-grading and down-grading software, deployment and provisioning servers**
* No need any agent in the nodes for installation and management
* Python / YAML based application, High flexible for configure and fast
* N number of modules are available and can do roll-back
* Simple and Human readable application
* Ansible is depends on the playbook, code is very easy when compare to shell scripting
* Ansible is easy to deploy because it doesn’t requires any agent or custom security infrat-structure
* It uses playbooks to describe automation jobs, playbook uses very simple languages called YAML (Yet Another Markup Language)
* Ansible work over SSH & doesn’t requires any daemons, special servers or libraries, whereas a text editor or a command line tool is enough
* Advantages: Agentless, simple written in YAML playbooks, powerful & Flexible, efficient no additional software on your servers, open source, secure (SSH) which is encrypted, Ease of use
* Tools like Puppet and Chef are pull based architecture, whereas Ansible is a **push based mechanism**
* Chef uses clients and puppet use agent, there is no agent for Ansible
* Ansible orchestration engine consists of inventory, API, Modules & Plugins
* Three things are important while installing ansible in centos or in Ubuntu, they are Inventory, Playbook, Module which will be available in the controller
* INVENTORY normally contains IP Address of Database, Application and Production Server, whereas if we need to configure any package to database server or any server, first thing we need is its IP Address
* PLAYBOOK is used to store or transmit data we need YAML script, in case if we need to execute hundreds of database server at single shot we will write a script in YAML playbook, so that we execute all in one time
* MODULES, it consists of services like httpd, cp, mc, ping, simply it will perform the task or do actions like move, copy, ping also it will manage the agents for each server
* So, Inventory, Playbook & Module everything will be in one place called “**Controller**” which is ANSIBLE
* Playbook is also called as combination of ADHOC commands

**DOCKER:**

* Docker is an open source platform, designed to manage container, which allows us to build applications in a container with required libraries, binaries, dependencies to run the application, ship the container and run anywhere
* Docker is nothing but containerization technique which is lightweight virtualization technology, an application inside the container will have codes, dependencies and libraries are bundled together
* Wherever you run that container will run without any error or any external support
* Infrastructure (AWS) -> Host OS (Centos, Ubuntu) -> Docker (a package or Daemon) -> Container (Bundle of App PACK) <- this is the structure of Docker operations
* Containers are isolated, doesn’t require OS and its shares the host OS, its portable and lightweight, faster to run, improve scalability and lower costs
* Portability, Lightweight, Fast delivery and scalable, Continuous deployment and testing, Multiple isolated OS on single Host and Resource optimization
* Ubuntu -> “**sudo apt-cache search docker**” is the command used to check whether docker is available or not in the Ubuntu OS, if u found the s/w then enter the command “**sudo apt-get install docker.io”** then giving “Y” will install docker in that particular server
* “**sudo systemctl status docker”** will tell whether docker is enabled and active or not
* **“Docker –v”** will give the version info of the docker, “**sudo** **docker version**” this command will display the client and server info, our commands will be stored in client whereas server is nothing but docker daemon or engine
* **Note:** if we are in root user then “**sudo**“ is not mandatory
* “**docker**” this will display the list of management commands and commands used in the docker
* For ex cp, mv, info, volume, swarm, stop, commit etc will be available so “**docker volume ls**” will display some assigning options and “**docker volume help**” will display the commands can be performed under the “volume” option
* Docker containers are used to run applications in an isolated environment. By default, all the changes inside the container are lost when the container stops. **If we want to keep data between runs**, Docker volumes and bind mounts can help
* “**docker images help**” will display the list of images if we created previously, for actions need to use the command as “**docker image help**”
* Docker will have many commands, in each command there will a command called “**prune**” which is used to remove unused volumes or images and etc, the purpose “**prune**” is to delete the unused actions or spaces
* “**sudo docker info**” will display the overall info about the containers, like running, stopped and etc, also it will display what kind of drivers we are using “**swarm**” means cluster was active or inactive etc
* “**docker system help**” will display df, events, info & prune, info is explained in above point, prune is alrdy we know abt it. “**df**” will display the docker disk usage and “**sudo docker system df**” will display no of images, containers & etc. “**sudo docker system events**” will display the issues or process problems or docker corrupted it will show where it is
* Below listed some docker commands and its usages

**Management Commands:**

* builder - Manage builds
* config - Manage Docker configs
* container - Manage containers
* context - Manage contexts
* image - Manage images
* manifest - Manage Docker image manifests and manifest lists
* network - Manage networks
* node - Manage Swarm nodes
* plugin - Manage plugins
* secret - Manage Docker secrets
* service - Manage services
* stack - Manage Docker stacks
* swarm - Manage Swarm
* system - Manage Docker
* trust - Manage trust on Docker images
* volume - Manage volumes

**Commands:**

* attach - Attach local standard input, output, and error streams to a running container
* build - Build an image from a Dockerfile
* commit - Create a new image from a container's changes
* cp - Copy files/folders between a container and the local filesystem
* create - Create a new container
* diff - Inspect changes to files or directories on a container's filesystem
* events - Get real time events from the server
* exec - Run a command in a running container
* export - Export a container's filesystem as a tar archive
* history - Show the history of an image
* images - List images
* import - Import the contents from a tarball to create a filesystem image
* info - Display system-wide information
* inspect - Return low-level information on Docker objects
* kill - Kill one or more running containers
* load - Load an image from a tar archive or STDIN
* login - Log in to a Docker registry
* logout - Log out from a Docker registry
* logs - Fetch the logs of a container
* pause - Pause all processes within one or more containers
* port - List port mappings or a specific mapping for the container
* ps - List containers
* pull - Pull an image or a repository from a registry
* push - Push an image or a repository to a registry
* rename - Rename a container
* restart - Restart one or more containers
* rm - Remove one or more containers
* rmi - Remove one or more images
* run - Run a command in a new container
* save - Save one or more images to a tar archive (streamed to STDOUT by default)
* search - Search the Docker Hub for images
* start - Start one or more stopped containers
* stats - Display a live stream of container(s) resource usage statistics
* stop - Stop one or more running containers
* tag - Create a tag TARGET\_IMAGE that refers to SOURCE\_IMAGE
* top - Display the running processes of a container
* unpause - Unpause all processes within one or more containers
* update - Update configuration of one or more containers
* version - Show the Docker version information
* wait - Block until one or more containers stop, then print their exit codes

**How to launch and run docker container:**

* **“sudo docker search busapp”** it will display the no of images created and made it public by external users, these images are stored in “**docker registry**” or search it in google as “**hub.docker.com**”
* **“sudo docker search httpd”** this is the readily available images in the docker storage
* **“sudo docker images”** it will display if there is any images available previously
* Before building a container we need an image so downloading image by using command “**sudo docker pull httpd**” here httpd is a image name
* After downloading or pulling enter “sudo docker images” to ck whether it is downladed or not
* “**sudo docker pull alpine**” is another images pulled or downloaded
* Now enter “sudo docker system df” it has images two mentioned in the system
* After downloading images we need to run launch the containers for that we have commands
* “**sudo docker create & sudo docker run**” create -> will create dummy container where it won’t run the container, but run -> will do both create and run the container
* “**sudo docker ps**” will display container id, image name & etc
* “**sudo docker run --name web1 httpd** ” this is the syntax to run the container, if we exit from the command then the container will be shutting down so to avoid it we have to include as “**sudo docker run –d --name web2 http**” now it will run in the background.
* “**sudo docker ps -a**” will display the shutdown container as well as running container

**AWS:**

* Amazon web services it’s a cloud providing services, also called as infrastructure as a service
* Types: IAAS, PAAS, SAAS. Infrastructure as a service, Platform as a service, Software as a service
* There is also another service called “**ON PREMISES**”, which is a physical data center
* Important cloud services provided by AWS are Compute, Networking, Storage, Databases
* Compute services -> EC2 Instance, Networking -> VPC , Storage -> S3, Databases -> RDS

**Interview Question:**

 Work closely with product management team to ensure technical and business requirements are best reflected in end-to-end deployment strategies

 Requirement analysis and requirement gathering technical details regarding the environment

 Troubleshooting and resolving issues related to application deployment and operations

 Interacting with developers to sort out issues related to Build Automation

 Experience in creating and maintaining IAM roles

Resourceful DevOps Engineer with 4+ years of experience, seeking for an opportunity to drive my skills and efficieny in delivering the project. Experience with Source Code Management, Continuous Integration, Continuous Deployment and Configuration Management

Jenkins, Maven, Kubernetes, Ansible, GITGithub, TerraformShell, Scripting, Nexus, JIRA, Devops, Ansible, Docker, Puppet, Continuous Delivery, Devops, Continuous Integration, Configuration Management, Aws Lambda, Linux