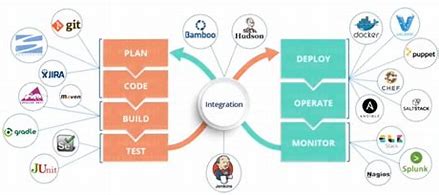
DEVOPS



Git

GIT :- 2.33.0 GIT :- version control system

mven :- 3.8.6 mven :- build the project

jenkins :- 2.333 (2.346.1) jenkins :-ci&cd(build,test,depoly)

ansible :- 2.9 ansible :- configrtion management

terrfrom :- 0.13 & 0.14 terrfrom :- infrastructure as code

docker :- 20.10.16 docker :-platfroOSlvirtualizton

k8's :- 1.22.12 k8's :- continerizedloadservic

elastic s :- 7.14 elastic:-distributsserc&anlytics en

Git

1$ cd c/

$ cd project/

$ c/ project/ "MASTER"

$ ll

$ ll -al

## .git

$ touch <filename>

$ ll

$ vim <filename>

< entry into any infermation edit>

$ cat <filename>

$ git status

$ git config --local user.name < name>

$ git config --local user.email < email id>

$ ll

$ git add .

$ git commit -m "commit edd"

$ git log

$ git remote add < github name> < url>

$ git push < name>

GIT:-

git is free and open source software for distributed version control

tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development.

version control system (VCS)?

< A VCS keeps track of the contributions of the developers working as a team on the projects>

They maintain the history of code changes done and with project evolution, it gives an upper hand to the developers to introduce new code, fixes bugs, and run tests with confidence that their previously working copy could be restored at any moment in case things go wrong.

GIT ADD:-

‘git add’ adds file changes in your existing directory to your index.

- You can add all changes at once using git add . command.

- You can add files one by one specifically using git add <file\_name> command.

GIT STAYUS:-

the working directory and the index which is helpful for understanding working directory.

keep track of the tracked and non-tracked changes

Git commit:-

Adding commits keep track of our progress and changes as we work. Git considers each commit change point or "save point".

It is a point in the project you can go back to if you find a bug, or want to make a change

Git Log:-

To view the history of commits for a repository, you can use the log command:

Git index:-

the list of all fils in the current directley

Git diff:-

This shows the changes between commits, working trees, etc.

GIT PULL:-

This command pulls new changes from the currently working branch located in the remote central repository.

Git push:-

From all the above information, it shows that git push is a command used to upload all changes done from the local branches to the targeted remote repository.

GIT REMOTE:-

git remote command creates an entry in git config that specifies a name for a particular URL.

GIT CLON:-

Whereas git clone creates a new git repository by copying an existing one located at the URL.

Git branch delete:-

simply use :-"git branch –d [head]"

local\_branch :- git branch -d <local\_branch\_name>

remote\_branch :- git push origin --delete <remote\_branch\_name>

Git repository creat:-

Then in order to create a git repository, create a folder for the project and then run git init.

Git checkout:-

A ‘git checkout’ command is used to update directories or specific files in your working tree with those from another branch without merging it in the whole branch.

Git architecture:-

distributed architecture

Git conflict:-

. When two separate branches have changes to the same line in a file

. A file is deleted in one branch but has been modified in the other.

Git config:-

The git config command is a convenient way to set configuration options for defining the behavior of the repository,

user information and preferences, git installation-based configurations, and many such things.

Git stash:-

The current branch running the "git stash" command basically pushes the current working directory state and index to the stack for future use and thereby providing a clean working directory for other tasks

Git pop:-

git stash pop command throws away the specified stash (topmost stash by default) after applying it.

git stash pop = git stash apply + git stash drop

Git Staging Area or Index:-

Before completing the commits, it can be formatted and reviewed in an intermediate area known as ‘Staging Area’ or ‘Index’.

Git branches:-

Git branch is simply a reference to the commit, where the following commits will be attached

Git master:-

Master is the name of a default branch in git terminology. Whenever a new repository is created in git, git gives the default name to a branch as ‘Master’

Git feach:-

Pulls all commits and changes from desired branch and stores them in a new branch of the local repository.

current

For changes to be reflected in the current / target branch, git fetch should be followed by git merge command.

Git cheerry pic:-

git cherry pic commit from one branch to anthoer branchin the in the repository

Git merge & GIT rebase:-

both commands can be used to commit .the difference is that rebase erites a liner commit history

Git hooks:-

script that are run after runing the respective git commands

Git rm:-

To remove the file from the staging area and also off your disk ‘git rm’ is used.

1)The GitHub flow works like this:

-Create a new Branch

-Make changes and add Commits

-Open a Pull Request

-Review

-Deploy

-Merge

5) What are the advantages of using GIT?

a) Data redundancy and replication

b) High availability

c) Only one.git directory per repository

d) Superior disk utilization and network performance

e) Collaboration friendly

f) Any sort of projects can use GIT

6) What language is used in GIT?

GIT is fast, and ‘C’ language makes this possible by reducing the overhead of runtimes associated with higher languages.

38) What is the function of ‘git reset’?

The function of ‘Git Reset’ is to reset your index as well as the working directory to the state of your last commit.

24) What is another option for merging in git?

“Rebasing” is an alternative to merging in git.

25) What is the syntax for “Rebasing” in Git?

The syntax used for rebase is “git rebase [new-commit]

29) What is Subgit? Why to use Subgit?

‘Subgit’ is a tool for a smooth, stress-free SVN to Git migration. Subgit is a solution for a company -wide migration from SVN to Git that is:

a) It is much better than git-svn

b) No requirement to change the infrastructure that is already placed

c) Allows to use all git and all sub-version features

d) Provides genuine stress –free migration experience.

1.What is git?

Open source ,code management system with version control system

VCS:-that can track angle changes files

2.What are git advantages?

Esay to use ; good performance, data replication, tracks files history

3.what is git language?

“c” programming language (simple, fast and no overheads)

4. what is git architechture?

Distributed architecture

5. what is diff b/w git and git hub?

This are diff tools. Git is a source code management system with version control tool.

Git hub has project management and collaboration features

6. what is git repository?

Directory with metadata – git stores all files with version information. Local repository (or) remote repository

7. how can you create agit repository?

Create a directory for the project.

Run gitn init command : creats a. Git directory with the project directory.

8. what is git configuration?

Git config –gobal usre.name “john” set up auser name

9. how to do git commit?

Git add <filename.file extension>: new file

Git commit –a : commit updated files

10. what is git diff?

Diff b/w files , commites branches

Git diff : files changed or added in a commit

e.g : git diff – tree –r : list all these files

11. what is git status?

Gives the diff b/w index and working directory.

Index :- the list of all files in the current directey

12. what is git log?

Gives version history recoreds about the project commites

Like :- developer & data ect......

13. what is git check out?

Git check out [branch name]:switch to the given branch

14. what is git revet?

Git revert [commit name] : under the changes made in the particular commit. Git revert can even undo multiple commites

15. what is head in git ?

Pointer to the currently cheked out branch for git commit.

A git repository can have multiple head

Master : the defacult head in git.

16. what is git staging area?

Staging area : intermediate area b/w working directory and the git repository. Git add : adds a file to the staging area for review the file to the repo.

17. what is git rm?

Git rm : delete a file from your working directory as well as the stating area

18. what is git stash?

Use full to move to another task. Git stash : putes the working directory on a stack and gives aclean working direcyory. Later, git stash apply

19. what are git master and branch ?

Master :- main copy of the code;

Branch :- independent copy of the code

e.g develop branch or hot fix branch git merge, git branch –merged

20. what is diff b/w git merge & git rebase?

Both commaneds can be used to commit. The diff is that rebase writes a liner commit history

21. how to check git branch meage?

Git branch –merge : shows the merged branch

Git branch –- no merged :shows the branch not yet merged

22. can be git branch be detected?

Yes. Git branch –d branchname

23. what is cherry –pick?

Git cherry –pick : commit from one branch to another branch in the repository

24. what is git conflict?

Two commit have changed in the same file and same location . a conflie has to be resolved for correct code to be committed

25. how can you resolve a git conf;it?

Edit the filemanually to remove the cofilct. Then git add the file followed by git commit.

26. what is git clone?

Git clone : copies the git repository

27. what is diff b/w git clon & git remote?

Git clone copies an existing repository in antoher directory and creates a new repository git remote the centeral repository

Remote :- use by the team nuber to store code update

28. what is area hooks in git?

Scripsthat are run after running the respective git commands

MAVEN

### 1) What is Maven?

Maven is a project management tool. It is based on POM (Project Object Model). **2) What aspects are managed by Maven?**

* Builds
* Documentation
* Reporting
* SCMs
* Releases
* Distribution

3) What are the advantages of Maven?

* No need to add jar file in each project
* Creates right directory structure
* Builds and deploys the proje

4) What is the command to check the maven version?

Type the following command on console to know the maven version.

-mvn -version  **5) What does the build tool?**

* Generates source code (if the auto-generated code is used)
* Generates documentation from source code
* Compiles source code
* Packages compiled code into a JAR or ZIP file
* Installs the packaged code in the local repository, server repository, or central repository

### ****Q5: How do you know what version of Maven you’re currently using?****

You can know the current Maven version by typing the command: maven-version.

### ****Q6: What is POM in Maven?****

POM in Maven stands for Project Object Model. POM is an XML file in Maven and the fundamental unit that contains crucial information regarding the project and other related configuration details needed to run the project.

### ****Q7: What is a Maven artifact?****

An artifact is a Jar file that is usually deployed to a repository. The build function creates artifacts such as a source Jar or a compiled Jar. Every artifact that is deployed comprises an artifact ID, a group ID, and a version string. Artifacts in Maven are identified by these three parameters.

### ****Q8: What is the Maven command to build your site?****

The command to build your site in Maven is: mvnsite

### ****Q9: What is the function of the mvnclean command?****

The mvnclean command deletes the target directory with all its data before commencing the build process for the project.

### ****Q10. What do you understand about the Maven Build Life Cycle?****

A Build Life Cycle in Maven specifies the order in which tasks need to be carried out and accomplished. Every build phase in Maven consists of a set of tasks or goals. This essentially implies that if one build life cycle is executed, all the tasks in that particular phase are executed. If a build phase is executed, the phases before it are executed.

### ****Q11. What are the phases that a clean life cycle in Maven consists of?****

A clean life cycle in Maven consists of the following three phases:

* The pre-clean phase
* The clean phase
* The post-clean phase

### ****Q12. What is the function of the command -- mvn clean dependency:copy-dependencies package?****

This command cleans the entire project and copies the dependencies in the project before packaging it.

### ****Q13. What do you understand about the Maven repository?****

A Maven repository is a place where all the artifacts related to the project, such as library jars, project jars, and plugins, are stored in order to be used for different project tasks.

### ****Q17. What are the different phases that a given site life cycle consists of?****

The different phases in a site life cycle are:

1. Pre-site
2. Site
3. Post-site
4. Site Deploy

### ****Q18. What is the default location of the local Maven repository?****

The default location of the local Maven repository is ~/m2./repository

### ****Q19. What are the different types of plugins used in Maven?****

There are basically two types of plugins that Maven provides for:

1. **The build plugins:** These plugins are used during the Build process and can be configured in the <build/> element of the POM.xml file.
2. **Reporting plugins:**These plugins come into effect during the site generation process. They can be configured in the <reporting/> element of the POM.xml file.

### ****Q20. What is an Archetype?****

Archetypes are Maven plugins that are used to create the structure of the project in accordance with the specifications of a predefined template.

### ****Q21. What Maven command would you use to create a new project that’s based on an Archetype?****

The command used for this function is -- maven archetype:generate.

### ****Q22. When does the “External Dependency” activity come into the picture while executing a Maven project?****

The External Dependency feature comes to the fore when dependencies aren’t located in the local repository, and certain libraries need to be fetched from the central repository.

### ****Q23. What do you understand about transitive dependency in Maven?****

Transitive dependency is when Maven can automatically locate libraries and other dependencies required for the project without you needing to manually locate these dependencies.

### ****Q24. What are the different dependency scopes in Maven?****

Dependency scopes include project dependencies that are in line with the current, active stage of the build process. The different Dependency scopes in Maven are:

1. **Compile:** This scope specifies that the current dependency is available in the classpath of the current project.
2. **Provided:** This scope specifies that the web server will provide the dependency during runtime.
3. **Runtime:** This particular scope indicates that the dependency isn’t needed during compile time and is only needed during execution.
4. **Test:** This scope indicates that the dependency is available limitedly during the test compilation phase.
5. **System:** This scope specifies that the system path for the given build phase needs to be provided.
6. **Import:** This scope indicates that the given POM should be replaced with congruent dependencies in the POM’s <DependencyManagement> section.

### ****Q25. How would you run the “clean” plugin during the build process?****

You can run the clean plugin by placing it inside the execution tag of the POM.xml file.

ANSIBLE

What is Ansible:-

Ansible is an open-source IT engine that automates application deployment, cloud provisioning, intra service orchestration, and other IT tools.

Configuration Manager:-

Configuration Manager is a job designation that can change your future.its change mainly packages.

Configuration management is the discipline that ensures every company-owned software or hardware is in the record

* CM Server:
  + This has necessary tools to automate the deployment
  + This server understands the desired state.

# Ansible Tutorial:-

# Ansible tutorial provides basic and advanced concepts of Ansible. Our Ansible tutorial is designed for beginners and professionals.

Why Use Ansible:-

Here are some important reasons for using Ansible, such as:

* Ansible is free to use by everyone.
* Ansible is very consistent and lightweight, and no constraints regarding the operating system or underlying hardware are present.
* It is very secure due to its agentless capabilities and open **SSH** security features.
* Ansible does not need any special system administrator skills to install and use it.
* Ansible has a smooth learning curve determined by the comprehensive documentation and easy to learn structure and configuration.
* Its modularity regarding **plugins, inventories, modules,** and **playbooks** make Ansible perfect companion orchestrate large environments.

# Ansible Workflow:-

# Ansible works by connecting to your nodes and pushing out a small program called **Ansible modules** to them. Then Ansible executed these modules and removed them after finished. The library of modules can reside on any machine, and there are no daemons, **servers,** or **databases** required.

# Ansible Architecture:-

# The Ansible orchestration engine interacts with a user who is writing the Ansible playbook to execute the Ansible orchestration and interact along with the services of private or public cloud and configuration management database.

|  |  |
| --- | --- |
| **Terms** | **Explanation** |
| Ansible Server | It is a machine where Ansible is installed and from which all tasks and playbooks will be executed. |
| Modules | The module is a command or set of similar commands which is executed on the client-side. |
| Task | A task is a section which consists of a single procedure to be completed. |
| Role | It is a way of organizing tasks and related files to be later called in a playbook. |
| Fact | The information fetched from the client system from the global variables with the gather facts operation. |
| Inventory | A file containing the data regarding the Ansible client-server. |
| Play | It is the execution of the playbook. |
| Handler | The task is called only if a notifier is present. |
| Notifier | The section attributed to a task which calls a handler if the output is changed. |
| Tag | It is a name set to a task that can be used later on to issue just that specific task or group of jobs. |

# Ansible ad-hoc Commands:-

# Ad-hoc commands are one of the simplest ways of using Ansible. These are used when you want to issue some commands on a server or bunch of servers. The ad-hoc commands are not stored for future use, but it represents a fast way to interact with the desired servers.

# Ansible Playbooks:-

# Playbooks are the files where the Ansible code is written. Playbooks are written in YAML format. **YAML** means "Yet Another Markup Language," so there is not much syntax needed.

# Ansible Tower:-

# Ansible Tower is like Ansible at a more enterprise level. It is a web-based solution for managing your organization with an easy user interface that provides a dashboard with all of the state summaries of all the hosts. And allows quick deployments, and monitors all configurations.

# Ansible Roles:-

# Roles provide a framework for fully independent or interdependent collections of files, tasks, templates, variables, and modules.

# Ansible Variables:-

# In playbooks, the variable is very similar to using the variables in a programming language. It helps you to assign a value to a variable and use it anywhere in the playbook. You can put the conditions around the value of the variables and use them in the playbook accordingly.

# Ansible Tags:-

# If you have a large playbook, it becomes useful to be able to run only a specific part of it rather than running everything in the playbook. Ansible supports a tag attribute for this reason

# Ansible Galaxy:-

# Ansible Galaxy is a galaxy website where users can share roles and to a command-line tool for **installing, creating,** and **managing** roles.

# Ansible Modules:-

# Ansible modules are discrete units of code which can be used from the command line or in a playbook task.The modules also referred to as task plugins or library plugins in the Ansible.

# Ansible Shell:-

# Ansible shell module is designed to execute the shell commands against the target UNIX based hosts. Ansible can run except any high complexes commands with pipes, redirection. And you can also perform the shell scripts using the Ansible shell module.

# Ansible Templates:-

# Ansible is used to manage configurations of multiple servers and environments. But these configuration files can vary for each cluster or remote server. But apart from a few parameters, all other settings will be the same.

# Ansible YAML:-

# YAML is used to describe configuration that has been increasing in the past few years with the help of **Ansible** and **SaltStack**.

# Ansible Inventory:-

# Ansible works against multiple managed hosts in your infrastructure at the same time, using a list or group of lists is known as the inventory.

# Ansible Debug:-

# Ansible provides a debug module option that makes the tasks more manageable. It is a handy tool to figure out any problem areas.

# Ansible Apt:-

# APT stands for "Advanced Packaging Tool" is the preferred package management toolset in Ubuntu. It allows us to install new packages, update them, and remove the packages from Ubuntu or Debian systems. Here are 3 APT related command-line tools, such as:

**Apt-get:** All the basic package management operations can be done by using this tool. Ansible apt-get module provides this functionality.

**Apt-add-repository:** It is used for adding a new repository to the repository list. The default repository may not have the latest version of all the packages. So you need to add additional repositories for some software maintainers. Ansible apt\_repository module provides the functionality for adding a new repository.

**Apt-key:** It is used to manage the list of keys for authenticating apt packages. Ansible apt\_key module is used to manage the keys

# Ansible Unarchive:-

# Ansible unarchive module is used to unpack or uncompressed the files from an archive file such as zip, tar, tar.gz. It can optionally copy the files to the remote server before uncompressing them.

## Why Use Unarchive Module:-

## The Ansible Unarchive module is useful for moving large files and folders across host machines.

# Ansible Pip:-

# Ansible pip module is used when you need to manage python libraries on the remote servers

# Ansible Lineinfile:-

# The lineinfile is one of the most powerful modules in the Ansible toolbox. Ansible lineinfile module is used to insert a line, modify, remove, and replace an existing line.

# Ansible Copy:-

# Ansible provides the functionality of copying the files and directories with the help of copy and fetch modules. The copy module is versatile.

# Ansible Command:-

# Ansible command module is used to run any commands or run any scripts in the remote target machine. Or used to execute commands on a remote node.

# Ansible File:-

# Ansible file module is used to creating and deleting the file or multiple files in the remote server. You can also create and delete the directories and change the permissions of the data

# Ansible Vault:-

# Ansible Vault is a feature which allows user to encrypt values and data structures within Ansible projects. This provides the ability to secure any secrets or sensitive data that is necessary to run Ansible plays successfully but should not be publicly visible, such as private keys or passwords. Ansible automatically decrypts the vault-encrypted content at runtime when the key is provided.

# Ansible Windows:-

# Using Ansible to manage Windows, many of the syntax and rules that apply for Unix/Linux hosts also apply to Windows. Still, there are some differences when it comes to components like path separators and OS-specific tasks. This document covers details specific to using Ansible for Windows

# Ansible Yum:-

# Ansible has a specific module for managing the Yum packages. You can install, remove, upgrade or downgrade versions and many more by using this module

# Ansible AWX:-

# Ansible AWX is the open-source freely available project for testing and you can use Ansible AWX in a lab, development, or other POC environment. It was the foundation on which Ansible Tower was created.

* PUSH BASED CM:-
  + CM Server will initiate the connection, so it requires the ipaddress/name of the node
  + It will login into the Node and Execute the configuration. For this we need credentials of the node.
  + In this model the Configuration Management Server has to maintain ip address/names and credentials of the nodes.
  + Ansible is a Push Based CM implementations
* PULL BASED CM:-
  + Node will initiate the communication and ask for the configuration to be executed and when the CM Server Responds, it executes the configuration and shares the execution report (success/failure & some logs about what happended)
  + On the nodes a CM agent will be installed which is responsible for communication, execution and reporting

ansible:-

sudo apt update

sudo apt install software-properties-common

sudo add-apt-repository --yes --update ppa:ansible/ansible

sudo apt install ansible

sudo apt date

sudo apt install software-properties-common

sudo add-apt-repository --yes --update ppa:ansible/ansible

sudo apt install ansible

ansible --verstion

sudo vi /etc/ssh/sshd\_config

sudo service sshd restart

sudo adduser devops

sudo visudo

ALL=(ALL:ALL) NOPASSWD:ALL

SU devops

sudo ssh-keygen

ssh --copy-devops ip

ssh devops@ip addrues2

vi nopcommcers

ansible -i playbook -i host| nopcommcers playbook

openjdk-11-jdk install:-

sudo apt update

sudo apt install openjdk-11-jdk

java -version

sudo apt install openjdk-8-jdk

java -version

sudo update-alternatives --config java

sudo nano /etc/environment

source /etc/environment

echo $JAVA\_HOME

sudo apt remove openjdk-11-jdk

DOCKER

$ docker --version

$ docker info -----------> all of the infermation

$ docker images

$ docker search < image name>

$ docker pull < image name>

$ docker run < image name> /bin/bash

$ docker images

$ docker ps

$ docker ps -a

$ docker run -it ubunt /bin/bash

$ ll

exit

$ docker run -it --name "name" image name /bin/bash

$ exit

$ docker start "braham"

$ docker attach "braham"

$ exit

$ cat /etc/os-version

$ cat /etc/os-release

$ exit

$ docker start "braham"

$ docker stop "braham"

$ docker ps

$ docker stop "braham"

$ docker rm "braham"

$ touch file1 1 2 3 4 a b c

$ rm -rf \*

$ vim < filename>

FROM ubunt

RUN echo "hello world"

$ docker build -t ubunt .

error

$ systemctl restart docker

$ systemctl status docker

$ docker build -t ubunt

FROM ubunt

$ docker image

$ vim "braham"

FROM ubunt

RUN echo "hello world" /temp/testfile

$ docker build -t ubunt

successfull

successfull

$ ls

$ docker images

$ docker run -it ubunt

$ ls < container add>

$ cd tmp/

$ ls

testfile

$ cat testfile

hello world

$ exit "docker entry"

$ cat dockerfile

$ echo

< error>

FROM:- basic image name this command must be on top of the file

RUN:- execute command,file lyear crat

COPY:- copy file form in loca system (docker vm)

ADD:- downld files from internet also (DOCK img)

EXPOSE:- ports such as 8080

WORKDIR- to set warning directory for the container

CMD:- executes commands but during container creation

ENTRYPOINT:- high priority than CMD

EVN:- environment variables

ARG:- variable that can be use to build container img

$ docker build -t ubunt

DOCKER index :-

-------------

build docker image

install docker

image build proces

manual approach

automated approach

common directives

exercise

-dockerfor spring petclinic

-docker for gameofile

docker file directives

container work flow

docker image layers & container

docker volume

docker noteworking

cluster (pods)

docker swrm setup & swarm work

manage nodes & work nodes

software work flow:-

design--->development---->deployment--->testing/relese

Docker stage:-

deployment

Docker process supported:-

frontend components, backend worker, db, env labs, dependences using of docker

Docker:-

Docker is a set of platform as a service products that use OS-level virtualization to deliver software in packages called containers.

- Docker is a very popular and powerful open-source containerization platform that is used for building, deploying, and running application

Docker file:-

It is a text file that has all commands which need to be run for building a given image.

Docker image:-

Docker Image is an executable package of software that includes everything needed to run an application. This image informs a container.

Docker Container:-

A Docker container may be created using a Docker image. It is an executable package of the software, which holds everything that is required to run an application,

The software that hosts the containers is called Docker Engine.

Docker architecture are given below:

Host:-

This component holds the Docker Daemon, Images, and Containers. While the Docker Daemon establishes a link with the Registry, the Docker Images act as metadata for the applications which are held in the Docker Containers.

Client:-

The Docker Client component runs operations to set up communication with the Docker Host.

Registry:-

This Docker Component is used to store the Docker Images. Docker Hub and Docker Cloud are public registries, which can be utilized by anyone

Docker Container life cycle:

Creation of the container-->Running the container-->Pausing the container-->Unpausing the container-->Starting the container-->Stopping the container-->Restarting the container-->Killing the container-->Destroying the container

Docker Swarm:-

Docker Swarm is a native tool used for clustering and scheduling Docker containers. Using Docker Swarm

Docker Objects:-

docker image,docker Containers,docker Services,docker vloum

docker cluster:-

A cluster in docker refers to multiple nodes joined using swarm mode.

docker pods :-

A Pod represents a single instance of a running process in your cluster. Pods contain one or more containers, such as Docker containers. When a Pod runs multiple containers, the containers are managed as a single entity and share the Pod's resources.

docker replica:-

A replica

docker entery point:-

used to set executables that will always run when the container is initiated

Docker volumes:-

Image result for docker volumes

Docker volumes are file systems mounted on Docker containers to preserve data generated by the running container. The volumes are stored on the host, independent of the container life cycle. This allows users to back up data and share file systems between containers easily.

KUBERNETES

k8's:-

Kubernetes is a portable, extensible, open source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation

To start with are world be using k8s as can approach to creat a k8s cluster

Docker Networking:-

Docker Networking Architecture is based on Container Network Model (CNM)

Cluster:-

A server cluster is group of two or more nodes (servers), that run in parallel to acheive a common goal

A cluster functions as if it were a single system. A user accessing the cluster should not need to know whether the system is a cluster or individual machine

* Benifits:-

High Availability

Load Balancing

Scaling

Performance

* What is Monolith:-

This is a unit of deployment, where all the functionality of the system has to be deployed together

* name space:-

k8s use name space to organize objects in the cluster

name space can be thought as a virtual cluster

* Kubernetes object:-

Kubernetes objects are represented in JSON or YAML files and describe the state of your cluster. The state of the cluster defines what workloads should be running in it

* 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. Pods include one or more containers

* k8s health check:-

Liveness detection and readiness detection are two kinds of health check mechanisms.

* k8s labels:-

Basically, labels are key/value pairs that we can give to k8s objects. By using labels we can identify attributes of objects and also can select those objects by the selector.

* replica:-

A replica ensures that a number of pods creates in cluster

* k8s annotations:-

Annotations Are used for non-identifying information. Stuff not used internally by k8s. You can’t specify selectors over them within Kubernetes, but they can be used by external tools and libraries.

* scallind replica:-

its 3 types

1)imperatively scale

2)declarative scale

change the yml field replica in replica manifest and apply

3)auto scalling:-

k8s cn handle scallind automatically based on pl & memory conuption using horizontal pods autoscalling H.P.A

K8s Deployment Features A Deployment owns and manages one or more ReplicaSets. And Replica Set manages the basic units in Kubernetes – Pods. Relationship among Pods, ReplicaSet and Deployment

* maxSurge:-

the maximum number of pods that can be created over the desired number of pods

* maxunavailable:-

the maximum number pods that can be maxunavailable during the update

* daemonset in K8s:-

A little K8s knowledge everyday! As it name suggests, the main function of DaemonSet is to let you run a Daemon Pod in a K8s cluster. A DaemonSet ensures that all (or some) work nodes run a copy of a Pod. As nodes are added to the cluster, Pods are added to them.

* storges :-

1)volume:-

2)persistent volumes:-

easy-to-scale systems with persistent volumes.

* k8s vloums:-

K8s Volume enables us to plug-in an external storage into the container workspace. Because of the plugged-in storage, the containers can save and re-use data across restarts and failures

types:-emptey,host,other cloud based volume

* persistentvolume in K8s:-

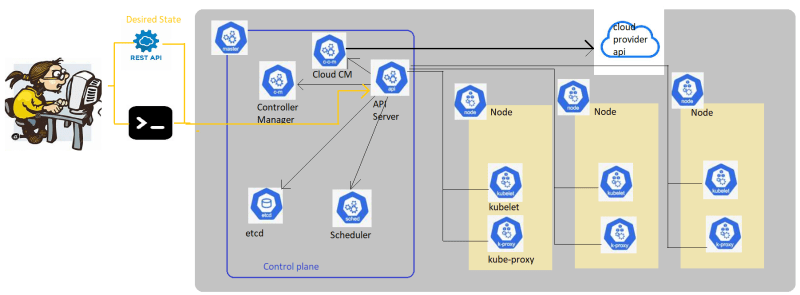
The Volume being part of a pod, it follows the life cycle of a pod and provides only a runtime reference to the storage. But, to manage the storage beyond the usage time of the pods, K8s needs a standalone handle which is addressed by the PersistentVolume.

* What is Kubernetes?

Kubernetes is a portable, extensible, opensource platform for managing containerized workloads and services

Architecture:-

Components of kubernetes (k8s) cluster



* Control Plane Components
* The control plane components make decissions about the cluster
* Control plane can be run on any machine in the cluster
* We can create a highly available cluster by using multiple machines for control plane components
* The components are:

kube-apiserver

etcd

kube-scheduler

kube-controller-manager

kube-cloud-controller-manager

* Node Components:
* Thy run on every node, maintianing running pods and providing k8s runtime environment.
* Our applications will be running on nodes
* The Node Components are
* kubelet
* kube-proxy

1. Container runtime

* kube-apiserver:-

The API server is a component of k8s control plane that exposes k8s API (Front-end of k8s)

All the communication between control plane and nodes is also handled by api server

To make k8s HA (highly Available), we can horizontal scale api-server

As a user of k8s cluster we can interact with kube-api server using API with json or a tool called a kubectl which is a command line tool

* etcd:

This is distribute key-value store.

k8s uses etc to store all the cluster data

* kube-scheduler:-

Control plane component that creates Pods on the nodes by selecting them

* kube-controller-manager:

1. Control plane component runs controller proceses. Each controller is a seperate process, but to reduce complexity they run in single process
2. Some major types of controller are

* Node Controller: Responsible for noticing and responding when node goes down
* Job Controller
* Endpoints controller
* Cloud-controller-manager:-
* This component embeds cloud-specific logic
* Kubelet:
* This is an agent that runs on each node in the cluster.
* Kubelet recieves requests/orders to create new Pods
* kube-proxy:
* This is a network proxy that runs on each node in k8s
* This maintains network rules on the nodes
* Container runtime:
* Kubernetes suppors container run times such as containerd, CRI-O and any implementation of Kubernetes CRI (Container Runtime instance)
* K8s Installation
* K8s has wide variety of installation options
* local machine:
* minikube
* kind
* k3s
* k3d
* microk8s
* Manual cluster installation:
* kubeadm
* Automatic cluster installation
* kubespray
* kops
* RKE
* KubeSphere
* Kubermatic
* Managed clusters:-
* Azure Kubernetes Service
* Elastic Kubernetes Service
* Google Kubernetes Engine
* K8s API
* The core of k8s control plane is API Server.
* This API Server exposes HTTP API that lets users, different parts o cluster and external components to interact with one another
* K8s Objects
* k8s objects are persistent entities in the k8s system
* Object Spec and Status: Almost every k8s object includes two nested object fields
* Levels of API Version
* Alpha:
* version contains alpha (v1alpha1)
* Not recommended for production usage
* Beta
* version contains beta (v1beta1)
* This is well tested & enabling is safe
* The support for this feature will not be dropped but details might change
* Not recommended for production usage
* Stable
* version name is vX where X is an integer (v1, v3)
* Recommended for production usage
* API Groups:
* k8s api’s are grouped to make it easier to extend k8s API
* There are several groups
* core group: apiVersion: <version> => apiVersion: v1
* other groups: apiVersion: <group>/<version> => apiVersion: batch/vi
* Pod Life Cycle
* Docker container has the states
* running
* exited
* paused
* terminated
* Pods are not self healing
* Pod phase (phase in the PodStatus object) gives the summary of Where the pod is in life cycle
* Pod Phases
* Resource Management
* k8s allows users to specify two different resource metrics
* requests: here we specify minimum amount of resources required to run the application
* limits: here we specify the maximum amount of resources required to run the application.
* Job
* The Job object is responsible for creating and managing Pods in a template in Job specification. These pods run until successful completion.
* If the Pod fails before successful termination, the job controller will create a new Pod based on the Pod template in the job spec.
* Jobs are designed to manage batch-like worloads, where the items are processed by one or more pods
* Service Discovery
* The service-discovery tools help solve the problem of finding which process are listening at which address.
* Real discovery in k8s starts with a Service Object.
* Service object is a way to create a named label selector
* k8s Service
* K8s service when created gets a cluster ip which is virtual in nature, when any other resource tries to access the service using cluster ip it forwards to request to one of the pod
* An easier way to create the service is by using kubectl expose
* K8s Storage Solutions
* Volumes: k8s Volume has a lifecycle equal to Pod. Once the Pod is deleted, the data will be lost
* Persistent Volumes: These volumes have lifecylce independent of Pod, So data will not be lost
* To create Persitent Volumes, We have two options

1. Manual Provisioning: In this case we need to manually create the storage to be used by the k8s cluster
2. Dynamic Provisioning: In this K8s will try to automatically create the storage based on the details provided. We prefer this approach on clouds