Devops

Syllabus

1. Software Development Life Cycle ,Design and Architectural Engineering,Object Oriented Analysis and Design,Introduction to Agile development model,Introduction to Atlassian Jira,Introduction to DevOps,Microservices,Fragmentation of business requirement,Containerisation, docker,Container life cycle,YAML,Docker Swarm and Docker Stack , Kubenetes,Istio Service Mesh,delivery pipeline,Jenkins,Selenium integration with Jenkins,Developing an application in a team, code versioning system,
2. Introduction Git,

- Introduction Git repository and git structure,
- Adding code to git,
- Introduction to GitHub,
- Creating pull requests,

3. Introduction to software testing, Verification and validation,Principles of software testing,Introduction to STLC and V Model, Tools used for automation testing,Introduction to testing methods,Introduction to functional testing,Introduction to non-functional testing,Introduction to Selenium,Introduction to TestNG, TestNG annotations,HTML test result reporting,
4. Introduction to Cloud,
5. Introduction to :

- Virtualization,
- Containerisation ,
- Cloud Computing,
- Cloud SPI Model,
- Cloud Computing Types,
- Cloud Security,
- Virtualization,
- Hardware Virtualization,
- Para-Virtualization,
- Cloning,
- Snapshot and Template,Containerization,
- Operating System Virtualization,
- Cloud architecture,
- Deployment models,Services provided by Cloud ,
- Cloud development best practices,
- Introduction to AWS

Day1

syllabus

0. Introduction Git,

- Introduction Git repository and git structure,
- Adding code to git,
- Introduction to GitHub,
- Creating pull requests,

1. VCS - Version Control System
2. Git - Distributed VCS .

- developed by Linus Torvalds
- it uses version method
- makes copies of files at every commit
- other VCS on commit saves the changes done in the file (difference in files versions called delta)

3. cryptography authentication history (SHA)
4. GIT
   - here , master branch managed by project manager

demo

1. > git config --global --list

- > git config --global user.name

- git config help

```
git-config - Get and set repository or global options

SYNOPSIS
       git config [<file-option>] [type] [--show-origin] [-z|--null] name
[value [value_regex]]
       git config [<file-option>] [type] --add name value
       git config [<file-option>] [type] --replace-all name value
[value_regex]
       git config [<file-option>] [type] [--show-origin] [-z|--null] --get
name [value_regex]
       git config [<file-option>] [type] [--show-origin] [-z|--null] --get-
all name [value_regex]
       git config [<file-option>] [type] [--show-origin] [-z|--null] [--
name-only] --get-regexp name_regex [value_regex]
       git config [<file-option>] [type] [-z|--null] --get-urlmatch name
URL
```

- > git status -s gives

- 1. ?? (untracked)
- 2. A _ (Add to staging area)
- 3. M _ (file modified, added to staging area)[color:green]
- ◦ i.e add is done

- > git add ___.file

- 4. _ M (file modified, but not staged)[color:red]
- 5. U U - files have conflict

- > git log

  - check your activity

- > git log --oneline --graph --decorate

  - to get graphical representation of files

- > git diff

  - givers difference between two version of a file
  - modified content

- > git add . or git add -A

Day2

2. demo on how to revert back /previous version of source folder and files in it / repo folder on PC

- works only when **changes not moved to staging area**

> git checkout . (for all) git checkout file.extension (for particular file)

- eg.

- > git checkout index.html

3. if changes moved to staging area then use command

- to reset to version before staging area

> git reset --hard

4. git stash area

> git stash

- move your data from your file to a temporary stack area, called stash
- it moves all modified content , after previous commit

> git stash list

- list the changes as array element

> git stash pop git stash apply

- to bring back the modified content from stash to your file
- > git stash pop 0
- gives [0] entry in stash
- > git stash show -p
- it shows your content in stash
- > git stash clear
- to clear stash from all elements(containing files data)
- git stash commands

```
usage: git stash list [<options>]
   or: git stash show [<stash>]
   or: git stash drop [-q|--quiet] [<stash>]
   or: git stash ( pop | apply ) [--index] [-q|--quiet] [<stash>]
   or: git stash branch <branchname> [<stash>]
   or: git stash save [--patch] [-k|--[no-]keep-index] [-q|--quiet]
                      [-u|--include-untracked] [-a|--all] [<message>]
   or: git stash [push [--patch] [-k|--[no-]keep-index] [-q|--quiet]
                       [-u|--include-untracked] [-a|--all] [-m <message>]
                       [-- <pathspec>...]]
   or: git stash clear
```

- to get documentation on a command

> git stash --help

5. Branching

- 1. it is another line of development
    - master branch contains
    - 1. lastest changes /to code
    - 2. stable code/functionality
    - 3. non bugged code/compilable code
    - 4. non -crashing / no exception
    - other than master branch ,all branches are called **features branch**
    - only one master branch in one repo
    - separate commit for separate branches
    - directory specific features common for all branches, i.e stash,virtual directory,staging area
- 2. demos
    - 1. to show which branches are there in git repo

- > git branch
- 2. to create a branch
- > git branch branchName
- 3. to switch to a branch with name branchName, use command
- > git checkout branchName
- 4. to delete a branch use
- > git branch -d branchName
- 5. git branch doucmentation
- > git branch --help

```
git-branch - List, create, or delete branches

SYNOPSIS
       git branch [--color[=<when>] | --no-color] [-r | -a]
               [--list] [-v [--abbrev=<length> | --no-abbrev]]
               [--column[=<options>] | --no-column] [--sort=<key>]
               [(--merged | --no-merged) [<commit>]]
               [--contains [<commit]] [--no-contains [<commit>]]
               [--points-at <object>] [--format=<format>] [<pattern>...]
       git branch [--track | --no-track] [-l] [-f] <branchname> [<start-
point>]
       git branch (--set-upstream-to=<upstream> | -u <upstream>)
[<branchname>]
       git branch --unset-upstream [<branchname>]
       git branch (-m | -M) [<oldbranch>] <newbranch>
       git branch (-c | -C) [<oldbranch>] <newbranch>
       git branch (-d | -D) [-r] <branchname>...
       git branch --edit-description [<branchname>]
```

6. Merging branches

- 1. merging two branches
- 2. demos to merge feature branch to master
- 1. go to master branch

> git checkout master;

- 2. use command

> git merge branchName; git merge courses;

- 3. deleteing branches after merge(optional)

> git branch -d branchName git branch -d courses

- 3. git merge doc

> git merge --help

```
SYNOPSIS
       git merge [-n] [--stat] [--no-commit] [--squash] [--[no-]edit]
               [-s <strategy>] [-X <strategy-option>] [-S[<keyid>]]
               [--[no-]allow-unrelated-histories]
               [--[no-]rerere-autoupdate] [-m <msg>] [<commit>...]
       git merge --abort
       git merge --continue
```

7. Branching advanced command

- 1. create a branch and switch to that branch use command

> git checkout -b branchName

- e.g

> git checkout -b about-us

8. Merge conflict

- two branches having change on same line
- when we try to merge these brnaches , confict part is shown for 2nd branch
- to resolve the conflict, developer has to make the decision,either
  - 1. either let it be
  - 2. or delete changes
  - then , we have to manually commit ,the master branch 3. > git add . 4. > git commit -m "brnach added"

9. A distributed version control system (DVCS)

- different DVCS, server for shared repo/remote repo
  - 1. GitHub
  - 2. GitLab
  - 3. BitBucket
- each git repository contain
- 1. code
- 2. meta data

10. GITHUB

- 1. create a repo on github
- 2. add it to local PC

- > git remote add origin https://github.com/porjesuraj/Sunbeam-website.git
- 3. to get info about remote repository on local PC
- > git remote -v
- 4. now if you create a branch , and push the changes to github remote repo from the feature branch ,
- > git add .
- > git commit -m ""
- > git push -u origin feature_branch
- e.g
  - > git push -u origin attendance-report
- after push, on github website,now we have created a merge request on github
  - now the person managing master can use this request
  - and merge it in origin reposity i.e remote repository
- 5. if you want to pull a feature this branch, i.e present in remote repository(github.lab) -we need to be using command

> git pull origin feature_branch;

- 6. to check files content :
- > vim filename.extension
- > cat filename.extension

the remote repository like on github and gitlab is absolute clone of local repository

Questions

1. What is GIT?

- Git is an open source distributed version control system and source code management SCM.
- system with an insistence to control small and large projects with speed and efficiency.

2. Which language is used in Git?

- Git uses 'C' language. Git is quick, and 'C' language makes this possible by decreasing the overhead of run times contained with high-level languages.

3. What is a repository in Git?

- A repository consists of a list named .git, where git holds all of its metadata for the catalog. The content of the .git file is private to Git.

4. What is 'bare repository' in Git?

- A "bare" repository in Git includes the version control information and no working files (no tree., and it doesnt include the special git sub-directory.
- Instead, it consists of all the contents of the .git sub-directory directly in the main directory itself, whereas working list comprises of:
  - A .git subdirectory with all the Git associated revision history of your repo.
  - A working tree, or find out copies of your project files.

5. What is the purpose of GIT stash?

- GIT stash takes the present state of the working file and index and puts in on the stack for next and gives you back a clean working file.
- So in case if you are in the middle of object and require to jump over to the other task, and at the same time you don't want to lose your current edits, you can use GIT stash.

6. What is GIT stash drop?

- When you are done with the stashed element or want to delete it from the directory,

> run the git 'stash drop' command.

- It will delete the last added stash item by default, and it can also remove a specific topic if you include as an argument.

7. What are the advantages of using GIT?

- Here are some of the essential advantages of Git:
  - Data repetition and data replication is possible
  - It is a much applicable service
  - For one depository you can have only one directory of Git
  - The network performance and disk application are excellent
  - It is effortless to collaborate on any project
  - You can work on any plan within the Git

8. What is the function of 'GIT PUSH' in GIT?

- 'GIT PUSH' updates remote refs along with related objects

9. Why do we require branching in GIT?

- With the help of branching, you can keep your branch, and you can also jump between the different branches.
- You can go to your past work while at the same time keeping your recent work intact.

10. What is the purpose of 'git config'?

- The 'Git config' is a great method to configure your choice for the Git installation.
- Using this command, you can describe the repository behavior, preferences, and user information.

11. What is the definition of "Index" or "Staging Area" in GIT?

- When you are making the commits, you can make innovation to it, format it and review it in the common area known as 'Staging Area' or 'Index'.

12. What is a 'conflict' in git?

- A 'conflict' appears when the commit that has to be combined has some change in one place, and the current act also has a change at the same place.
- Git will not be easy to predict which change should take precedence.

13. What is the difference between git pull and git fetch?

- Git pull command pulls innovation or commits from a specific branch from your central repository and updates your object branch in your local repository.
- Git fetch is also used for the same objective, but it works in a slightly different method.
  - When you behave a git fetch, it pulls all new commits from the desired branch and saves it in a new branch in your local repository.
  - If you need to reflect these changes in your target branch,
  - git fetch should be followed with a git merge.
  - Your target branch will only be restored after combining the target branch and fetched branch. To make it simple for you, remember the equation below:

> Git pull = git fetch + git merge

14. How to resolve a conflict in Git?

- If you need to resolve a conflict in Git,
  - edit the list for fixing the different changes, and
  - then you can run "git add" to add the resolved directory,
  - and after that, you can run the 'git commit' for committing the repaired merge.

15. What is the purpose of the git clone?

- The git clone command generates a copy of a current Git repository.
- To get the copy of a central repository, 'cloning' is the simplest way used by programmers.

16. What is git pull origin?

- pull is a get and a consolidation.
- 'git pull origin master' brings submits from the master branch of the source remote (into the local origin/master branch), and then
- it combines origin/master into the branch you currently have looked out.

17. What does git commit a?

- Git commits "records changes to the storehouse"
- while git push " updates remote refs along with contained objects"
- So the first one is used in a network with your local repository,
- while the latter one is used to communicate with a remote repository.

18. Why GIT better than Subversion?

- GIT is an open source version control framework;
- it will enable you to run 'adaptations' of a task, which demonstrate the changes that were made to the code over time also it allows you keep the backtrack if vital and fix those changes.

- Multiple developers can check out, and transfer changes, and each change can then be attributed to a particular developer.

19. Explain what is commit message?

- Commit message is a component of git which shows up when you submit a change.
- Git gives you a content tool where you can enter the adjustments made to a commit.

20. Why is it desirable to create an additional commit rather than amending an existing commit?

- There are couples of reason
  - The correct activity will devastate the express that was recently saved in a commit.
  - If only the commit message gets changed, that's not a problem.
  - But if the contents are being modified, chances of excluding something important remains more.
  - Abusing "git commit- amends" can cause a small commit to increase and acquire inappropriate changes.

21. What does 'hooks' comprise of in Git?

- This index comprises of Shell contents which are enacted after running the relating git commands.
- For instance, Git will attempt to execute the post-commit content after you run a commit.

22. What is the distinction between Git and Github?

- 1. Git is a correction control framework, a tool to deal with your source code history.
- 2. GitHub is a hosting function for Git storehouses.
  - GitHub is a website where you can transfer a duplicate of your Git archive.
  - It is a Git repository hosting service, which offers the majority of the distributed update control and source code management (SCM) usefulness of Git just as including its features.

22. In Git, how would you return a commit that has just been pushed and made open?

- There can be two answers to this question and ensure that you incorporate both because any of the below choices can be utilized relying upon the circumstance:
  - 1. Remove or fix the bad document in another commit and push it to the remote repository.
    - This is a unique approach to correct a mistake.
    - Once you have necessary changes to the record, commit it to the remote repository for that I will utilize
    - > git submit - m "commit message."
  - 2. Make another commit that fixes all changes that were made in the terrible commit.
  - to do this, I will utilize a command
  - > git revert

24. What does the committed item contain?

- Commit item contains the following parts; you should specify all the three present below:
- 1. A set of records, representing to the condition of a task at a given purpose of time

- 2. References to parent commit objects
- 3. An SHAI name, a 40 character string that uniquely distinguishes the commit object.

25. Describing branching systems you have utilized?

- This question is a challenge to test your branching knowledge with Git along these lines,
- inform them regarding how you have utilized branching in your past activity and what reason does it serves, you can refer the below mention points:
- 1. Feature Branching:
    - A component branch model keeps the majority of the changes for a specific element within a branch.
    - At the point when the item is throughout tested and approved by automated tests, the branch is then converged into master.
- 2. Task Branching
    - In this model, each assignment is actualized on its branch with the undertaking key included in the branch name.
    - It is anything but difficult to see which code actualizes which task, search for the task key in the branch name.

26. By what method will you know in Git if a branch has just been combined into master?

- The appropriate response is immediate.
- To know whether a branch has been merged into master or not you can utilize the below commands:

> git branch - merged

- It records the branches that have been merged into the present branch.

> git branch - no merged

- It records the branches that have not been merged.

27. How might you fix a messed up submit?

- To fix any messed up commit, you will utilize the order
- "git commit-- amend?"
- By running this direction, you can set the wrecked commit message in the editor.

28. Mention the various Git repository hosting functions.

- The following are the Git repository hosting functions: + Pikacode + Visual Studio Online + GitHub + GitEnterprise + SourceForge.net

29. Mention some of the best graphical GIT customers for LINUX?

- Some of the best GIT customer for LINUX is + Git Cola + Smart git + Git-g + Git GUI + Giggle + qGit

30. What is Subgit? Why use it?

- 'Subgit' is a tool that migrates SVN to Git.
- It is a stable and stress-free migration.

- Subgit is one of the solutions for a company-wide migration from SVN to Git that is:
    - It is much superior to git-svn
    - No need to change the infrastructure that is already placed.
    - It allows using all git and all sub-version features.
    - It provides stress free migration experience.

CLoud COmputing

Syllabus

1. Introduction to Cloud,
2. Introduction to

- Virtualization,
- Containerisation ,
- Cloud Computing,
- Cloud SPI Model,
- Cloud Computing Types,
- Cloud Security,
- Virtualization,
- Hardware Virtualization,
- Para-Virtualization,
- Cloning,
- Snapshot and Template,Containerization,
- Operating System Virtualization,
- Cloud architecture,
- Deployment models,Services provided by Cloud ,
- Cloud development best practices,
- Introduction to AWS

Day1

notes

-

1. **Desktop computing**

- The first computer was invented by Charles Babbage (1822) but was not built until 1991!
  - Alan Turing invented computer science.
  - The ENIAC (1945) was the first electronic general-purpose digital computer, it filled a room.
  - The Micral N was the world's first “personal computer”(1973).
- first type of computing(mostly 70's era super computer to desktop)

1. **Client- server computing**

- 1. second type of computing (in 21 st century came client server ctype computing)
- 2. the internet came into being in 1983
    - Vinton Cerf and Bob Kahn are credited with inventing the Internet communication protocols we use today and the system referred to as the Internet.

1. **vertical scaling**

- scale up : by increasing server cpu configuration , like cpu processor, ram , storage

- scale down : decreasing your cpu configuration due to decreased website traffic people visiting

- issues with vertical scaling

- 1. dependency on one machine
- 2. limit on single server cpu configuration

3. **Cluster computing**

- here we use horizontal scaling,so it is also called **distributed computing**
- as requested are distributed between multiple machine

1. **Horizontal scaling**

- 1. due to issues with vertical scaling , we turn to horizontal scaling 1.we are cloning server machine containing web server having the website ,this collection is called fleat of machines

- this is known as CLuster computing

- 2. here we need to connect fleet of machines to the client,

- so we add Load Balancer to the cluster for handing request manqagement

- it check the free machine , and send the request to the clone machine which can handle the request

- example of load balancer :

    - **High Availability Proxy(HA Proxy)**
    - HAProxy is free, open source software that provides a high availability load balancer
    - and proxy server for TCP and HTTP-based applications that spreads requests across multiple servers.
    - It is written in C and has a reputation for being fast and efficient
    - using load balancer, we dont need public IP addresses for machines, it's a private address on the same network as the server
- 3. advantage of horizontal scaling compared to vertical scaling :
        - no dependency on single machine, comes with increased cost
        - HIGH- AVAILABILITY
            - even some clone machine fails, still website is working
- 4. scaling out :adding more machines, increase in web request
- 5. scaling in : removing th machine , as decrease in web request
- 6. challenges with horizontal /distributed scaling

- need to physically adding machine. needed time for it , - so problem not solved in real time

- so to solve this , cloud computing was introduced

- 3. **scaling and problem with scaling in client server computing**

> https://www.geeksforgeeks.org/overview-of-scaling-vertical-and-horizontal-scaling/

4. **Cloud Computing**

-

1. resources

- ppt on cloud

> https://www.journaldev.com/25061/cloud-computing

2. notes

-

1. here the cluster/ fleat of machine are virtual machine server, and not physical server machine, based in a data center
2. in data center,

- 1. there are server racks,
    - where multiple servers are located, with computing power, its located in one rack of server rack,so that rack is called rack server
    - with hard drive for storage ,where each HD has 128TB of SSD
    - GPU + CPU + SSD + physical CPU : these are componenet of Server rack
- 2. multiple server racks makeas a dat center

5. **Virtualization**

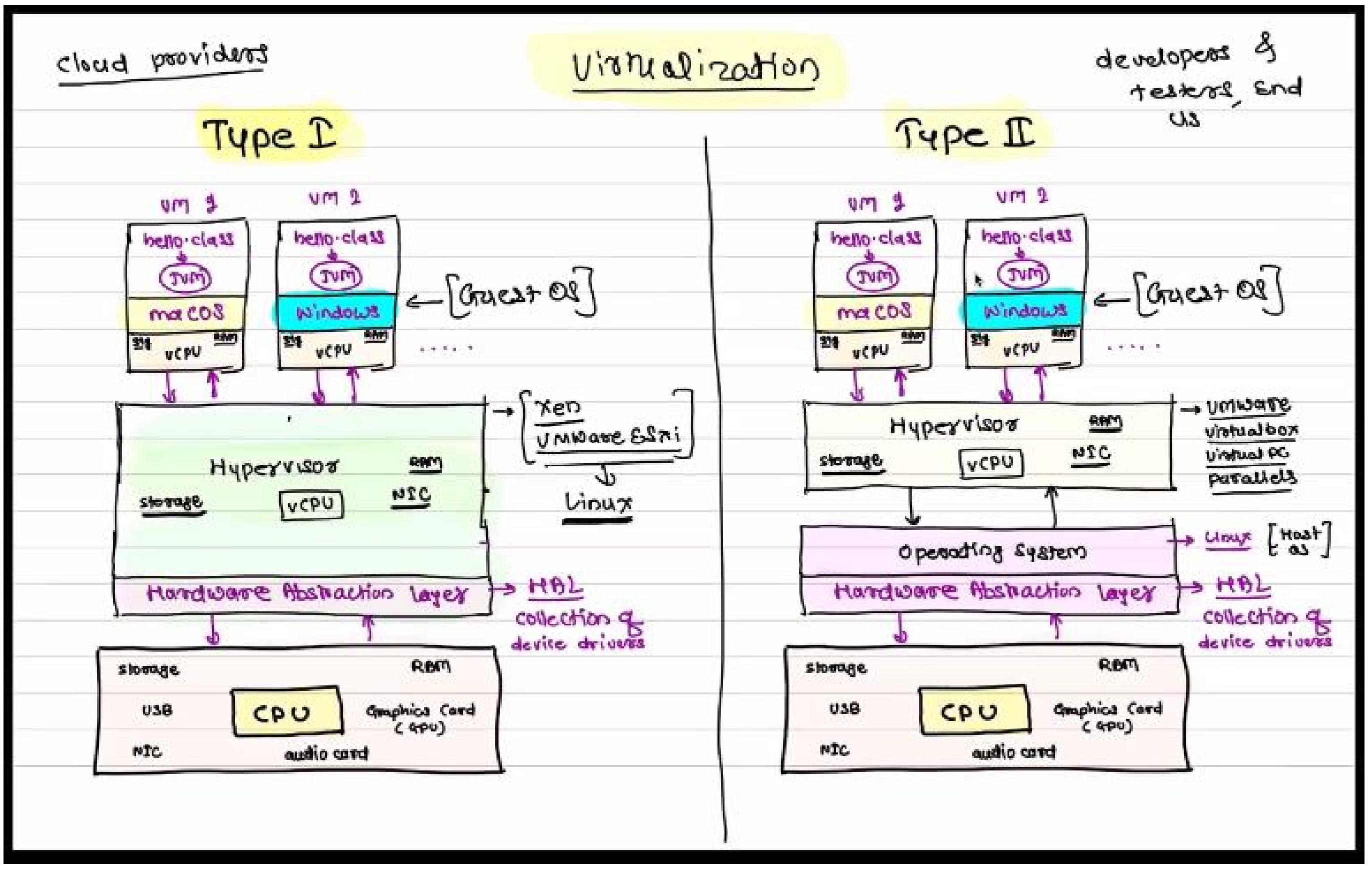
-

1. act of creating a vitual version of something like virutal hardware platform,storage devices and computer network resources

- mostly virtual OS like linux,mac os,windows
    - two type of hardware virtualization:
    - 1. type I
    - 2. TYPE II
    - Containerization

2. as we needed to test application like java, on multiple os, like windows ,mac os, and linux

- it was time consuming switching between OS, so virtualisation came into picture



1. TYPE II

-

1. Hardware
    - have CPU,NIC,GPU,Storage
2. HAL (Hardware abstraction layer)

- collection of device driver

3. OS called Host OS
4. Hypervisor

- convert physical version to virtual one
    - like virtual cpu (vcpu), ram, storage, NIC,

1. now on this hypervisor,

    - we have virtual machine , here we can install an OS ,called guest OS

2. example of Hypervisor :

    - virtual box
    - VMWare
    - vitual pc
    - parallels

3. for high PC hardware configuration, go for virtual OS,

    - for lesser configuration ,use multi booting

4. Type I ,practical use

-

1. easy to install and use , for personal computer,

- but we cant use maximum configuration , due to physical OS on machine for UI , etc occupying resources

2. best suited for developers,tester,and user

---

5. TYPE I

-

1. almost fullest configuration can be used here , as no physical OS i.e host OS om PC machine

- here, they are customized OS, for virtualizing certian hardware ,and not include UI

2. example of type I :

- 1. Xen
    - Xen is an open-source type-1 or baremetal hypervisor,
    - which makes it possible to run many instances of an operating system or indeed different operating systems in parallel on a single machine (or host).
- 2. VMWare esxi
    - VMware ESXi are hypervisors that use software to abstract
    - processor, memory, storage and networking resources into multiple virtual machines (VMs).
    - Each virtual machine runs its own operating system and applications.

3. best suited for cloud providers

Day2

1. to connect to ec2 instance

```
 > chmod 400 webserver.pem

> ssh -i webserver.pem ubantu@34.239.106.128
```

2. after ec2 instance get connected, for single html page

```
sudo apt-get update
sudo apt-get install apache2
cd /var/www/html/
sudo rm index.html
sudo vim index.html
```

```
     history
    exit
```

3. to deploy a angular application

-

1. to connect to ec2 instance

```
 > chmod 400 webserver.pem

> ssh -i webserver.pem ubantu@34.239.106.128
```

2. after ec2 instance get connected,

```
       sudo apt-get update
       sudo apt-get install mysql-server mysql-client

       sudo mysql

        alter user 'root'@'localhost' identified with mysql_native_password
by 'manager';
Query OK, 0 rows affected (0.00 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)

mysql> exit


sudo mysql -u root -p

create database sunbeam;


mysql> create table movie(movieId integer auto_increment primary key,name
varchar(25),rating integer,description varchar(50),image varchar(150));
Query OK, 0 rows affected (0.01 sec)

mysql> exit

sudo apt-get install apache2

  service apache2 status

sudo apt-get install nodejs
sudo apt-get install unzip


packet_write_wait: Connection to 18.208.144.123 port 22: Broken pipe


-- make server folder a zip file

 scp -i /home/sunbeam/dac/DevOps/Devops-module/Classwork/aws/suraj-aws.pem
AMAZON-SERVER.zip ubuntu@18.208.144.123:~/
AMAZON-SERVER.zip
```

Questions

1. What is cloud computing?

- Cloud computing is an internet based new age computer technology.
- It is the next stage technology that uses the clouds to provide the services whenever and wherever the user need it.
- It provides a method to access several servers world wide.

2. What are the benefits of cloud computing?

- The main benefits of cloud computing are:
  - Data backup and storage of data.
  - Powerful server capabilities.
  - Incremented productivity.
  - Very cost effective and time saving.
  - Software as Service known as SaaS.

3. What is a cloud?

- A cloud is a combination of networks ,hardware, services, storage, and interfaces that helps in delivering computing as a service. It has three users :
  - End users
  - Business management users
  - cloud service provider

4. What are the different data types used in cloud computing?

- There are different data types in cloud computing like emails, contracts, images , blogs etc. As we know that data is increasing day by day so it is needed to new data types to store these new data. For an example, if you want to store video then you need a new data type.

5. Which are the different layers that define cloud architecture?

- Following are the different layers that are used by cloud architecture:
  - CLC or Cloud Controller

  - Walrus
  - Cluster Controller
  - SC or Storage Controller
  - NC or Node Controller

6. Which platforms are used for large scale cloud computing?

- The following platforms are used for large scale cloud computing:
  - Apache Hadoop
  - MapReduce

7. What are the different layers in cloud computing? Explain working of them.

- There are 3 layers in the hierarchy of cloud computing.
- 1. Infrastructure as a service (IaaS):
  - It provides cloud infrastructure in terms of hardware as like memory, processor, speed etc.
- 2. Platform as a service (PaaS):
  - It provides cloud application platform for the developer.
- 3. Software as a service (SaaS):
  - It provides the cloud applications to users directly without installing anything on the system. These applications remains on cloud.

8. What do you mean by software as a service?

- Software As a Service (SaaS) is an important layer of cloud computing.
- It provides cloud applications like Google is doing.
- It facilitate users to save their document on the cloud and create as well.

9. What is the platform as a service?

- It is also a layer in cloud architecture. This model is built on the infrastructure model and provide resources like computers, storage and network.
- It is responsible to provide complete virtualization of the infrastructure layer, make it look like a single server and invisible for outside world.

10. What is on-demand functionality? How is it provided in cloud computing?

- Cloud computing provides a on-demand access to the virtualized IT resources. It can be used by the subscriber. It uses shared pool to provide configurable resources. Shared pool contains networks, servers, storage, applications and services.

11. What are the platforms used for large scale cloud computing?

- Apache Hadoop and MapReduce are the platforms use for large scale cloud computing.

12. What are the different models for deployment in cloud computing?

- These are the different deployment model in cloud computing:
  - Private cloud
  - Public cloud
  - Hybrid cloud

    - Community cloud

13. What is private cloud? Private clouds are used to keep the strategic operations and other reasons secure.

14. What is public cloud? The public clouds are open to the people for use and deployment. For example: Google and Amazon etc. The public clouds focus on a few layers like cloud application, infrastructure providing and providing platform markets.

15. What is the difference between scalability and elasticity?

- 1. Scalability is a characteristic of cloud computing which is used to handle the increasing workload by increasing in proportion amount of resource capacity.
    - By the use of scalability, the architecture provides on demand resources if the requirement is being raised by the traffic.
- 2. Whereas, Elasticity is a characteristic which provides the concept of commissioning and decommissioning of large amount of resource capacity dynamically.
    - It is measured by the speed by which the resources are coming on demand and the usage of the resources.

18. What are the security benefits of cloud computing?

- Cloud computing authorizes the application service, so it is used in identity management.
- It provides permissions to the users so that they can control the access of another user who is entering into the cloud environment.

19. What is the usage of utility computing?

- Utility computing is a plug-in managed by an organization which decides what type of services has to be deployed from the cloud.
- It facilitates users to pay only for what they use.

20. What is "EUCALYPTUS" in cloud computing? Why is it used?

- It is an acronym stands for

> Elastic Utility Computing Architecture For Linking Your Program To Useful Systems. - It is an open source software infrastructure in cloud computing and used to implement clusters in cloud computing platform. It creates public, private and hybrid cloud. It facilitate a user to create his own data center into a private cloud and use its functionalities to many other organizations.

21. What are the open source cloud computing platform databases?

- MongoDB, CouchDB, LucidDB are the example of open source cloud computing platform database.

22. Give some example of large cloud provider and databases?

    - Google bigtable
    - Amazon simpleDB
    - Cloud based SQL

23. What is the difference between cloud and traditional datacenters?

- The cost of the traditional datacenter is higher than cloud because in traditional databases, there is overheating problems and some software and hardware issue.

23. Why API's is used in cloud services? API's (Application Programming Interfaces) is used in cloud platform because:

It provide an alternative way that you don't need to write the fully fledged program.

It makes communication between one or more applications.

It creates applications and link the cloud services with other systems.

24. What are the advantages of cloud services?

- Following are the main advantages of cloud services:

    - Cost saving: It helps in the utilization of investment in the corporate sector. So, it is cost saving.
    - Scalable and Robust: It helps in the developing scalable and robust applications. Previously, the scaling took months, but now, scaling takes less time.
    - Time saving: It helps in saving time in terms of deployment and maintenance.

28. What are the different datacenters in cloud computing?

- Containerized datacenter
- Low density datacenter

29. What do you mean by CaaS?

- CaaS is a terminology used in telecom industry as Communication As a Service. CaaS offers the enterprise user features such as desktop call control, unified messaging and desktop faxing.

30. What do you mean by VPN? What does it contain?

- VPN stands for Virtual Private Network. VPN is a private cloud that manage the security of the data during the communication in the cloud environment.
- With VPN, you can make a public network as private network.

31. What are the basic clouds in cloud computing?

- There are three basic clouds in cloud computing:
    - Professional cloud
    - Personal cloud
    - Performance cloud

32. What are the most essential things that must be followed before going for cloud computing platform?

- Compliance
- Loss of data
- Data storage
- Business continuity
- Uptime

- Data integrity in cloud computing

33. What is the usage of virtualization platform in implementing cloud?

- The main usage of virtualization platform in implementing cloud is:
  - It is used to manage the service level policies.
  - Cloud Operating System.
  - Virtualization platforms help to keep the backend level and user level concepts different from each other.

35. We source cloud computing platform databases?

- Following are the open source cloud computing platform databases:
  - MongoDB
  - CouchDB
  - LucidDB

36. What are some large cloud providers and databases?

- Following are the mostly used large cloud providers and databases:
  - Google bigtable
  - Amazon simpleDB
  - Cloud based SQL

37. How would you secure data for transport in cloud?

- This is the most obvious question accurued in mind that if the cloud data is secure; To ensure that, check that there is no data leak with the encryption key implemented with the data you sending while the data moves from point A to point B in cloud.

syllabus

1. Introduction to DevOps,
2. Microservices,
3. Fragmentation of business requirement,
4. Containerisation,
5. docker,
6. Container life cycle,
7. YAML,
8. Docker Swarm and Docker Stack ,
9. Kubenetes,
10. Istio Service Mesh,
11. delivery pipeline,
12. Jenkins,
13. Selenium integration with Jenkins,Developing an application in a team,
14. code versioning system,

notes

1. deisadvantage of monolithic services

- 3 dis
- to avoid this use

2. Microservices

3. What is REST API and why it is used?

- A RESTful API is an architectural style for an application program interface (API) that uses HTTP requests to access and use data. That data can be used to GET, PUT, POST and DELETE data types, which refers to the reading, updating, creating and deleting of operations concerning resources.

day 4 : 29/11/2020

Containerization using Docker

docker command

0. check docker version

> sudo docker version

1. command to start docker

   > sudo systemctl start docker

2. coomand to stop docker

> sudo systemctl stop docker

3. command to restart docker

> sudo systemctl restart docker

4. change to root user

   > su - password : Enter password

5. docker image

- 1. inspect docker image

> sudo docker image inspect

```
sudo docker image inspect jenkinsci/blueocean
```

- 2. check dpcker image list

> sudo docker image ls

- 3. check image history

> sudo docker history <repository/image>

```
sudo docker history hello-world
```

- 4. get OS image from docker library/hub

> sudo docker image pull <repo/image>

```
sudo docker image pull ubuntu
```

5. Container command

- 1. to create container out of docker image
  - it creates and runs the image ,
  - it get closed as no nothing to run on it

> sudo docker container run ubuntu

- 2. list of working and stopped container

> sudo docker container ls -a

- 3. list of working container

> sudo docker container ls

Note :

- docker creates Virtual machine / or sharing same OS on physical machine if same OS required for container
- if not , diff OS, a OS VM is created
- VM runs full OS
- for container it runs kernel :
  - its starts , if there are commands/process to run it does it , or container get exited.
- 4. to delete a container

> sudo docker container rm ContainerID

- 5. to start the docker , with its Terminal in Container OS image

> sudo docker container run -it ubuntu

- where -i : interactive, t : tele type(Terminal)

```
$ sudo docker container run -it  ubuntu
root@d5809cf98cc0:/# ls
bin   dev  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  etc  lib   lib64  media   opt  root  sbin  sys  usr
root@d5809cf98cc0:/# mkdir code
root@d5809cf98cc0:/# cd code/
root@d5809cf98cc0:/code# touch file1.txt
root@d5809cf98cc0:/code# touch file2.txt
root@d5809cf98cc0:/code# ls
file1.txt  file2.txt
root@d5809cf98cc0:/code# exit
exit
```

- 5. install docker image

> sudo docker image pull mysql

6. use httpd image of apache instead of below commmad to install apache2 on container

```
sudo docker container run -it ubuntu
root@b33888469f38:/#
1. apt-get update

2. apt-get install apache2

3. service apache2 start

4. apt-get install curl
```

```
5. curl http://localhost
```

- 1. instead use

> sudo docker image pull httpd

- 2. now to run httpd

> sudo docker container run -it httpd

- 3. inspect httpd

> sudo docker image inspect httpd

- 4. check network info

> ifconfig

- 172.17.0.1 : ip address
- 5.
- 6. to run server in detached mode , in background using command -d (demion)

> docker container run -d httpd

- 7. running with port forwarding

> sudo docker container run -d -p 5678:80 httpd

- 8. to stop -d contianer

```
- sudo docker container ls

CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES
2bd496dbce31        httpd               "httpd-foreground"   16 seconds ago
Up 13 seconds       80/tcp              upbeat_ardinghelli

- sunbeam@sunbeam-Inspiron-3583:~$ sudo docker
container rm 2bd

Error response from daemon: You cannot remove a running container
2bd496dbce31428c52bf5cc37f8fc32f3dbb435456cf421901e7c7d6525cb212. Stop the
container before attempting removal or force remove

- sunbeam@sunbeam-Inspiron-3583:~$ sudo docker container stop 2bd
2bd

- sunbeam@sunbeam-Inspiron-3583:~$ sudo docker container rm 2bd
2bd
```

7. connect/start apacheserver in container using httpd

- 1. check ip address/ network info

  > ifconfig

- check ip
- 2. for port forwarding from pc to port 60 of container http5

> $sudo docker container run -d -p 5678:80 httpd

- 3. to execute any command inside container

> sudo docker container exec -it (containerID) bash

```
sudo docker container exec -it 1d712ffb3df6 bash

root@1d712ffb3df6:/usr/local/apache2# ls
bin    cgi-bin  error   icons    logs
build  conf htdocs  include  modules

root@1d712ffb3df6:/usr/local/apache2# cd htdocs

root@1d712ffb3df6:/usr/local/apache2/htdocs# ls
index.html

root@1d712ffb3df6:/usr/local/apache2/htdocs# cat index.html
<html><body><h1>It works!</h1></body></html>

root@1d712ffb3df6:/usr/local/apache2/htdocs# echo "hello world"
hello world

root@1d712ffb3df6:/usr/local/apache2/htdocs# apt-get install vim

root@1d712ffb3df6:/usr/local/apache2/htdocs# vim index.html
root@1d712ffb3df6:/usr/local/apache2/htdocs# vim index.html
root@1d712ffb3df6:/usr/local/apache2/htdocs# exit
exit
```

- 4. to see content of html file

> cat ____.html

- 5. vim commands
-
- 1. open html file

  > vim ___.html

- 2. to delete a line

> dd

- 3. to insert in file

> Esc + i

- 4. to exit

> Esc + :wq

- 5. now open browser to open your webpage on port 5678 coming from container port 80

> http://localhost:5678/ http://172.17.0.1:5678

custom docker image creation 7/12/2020

docker hub log in

1. docker id: surajporje
2. password : 8668951369

-

how to create a Container/Instance of a standard image

1. to manually create DB

- 1. after downloading mysql image
  - create mysql container , with schema using
  - need environment variable
  - create port : mandetory

```
# code to create a container from standard image
sudo docker container run --name mydb -d -e MYSQL_ROOT_PASSWORD=root -e
MYSQL_DATABASE=mydb -p 9090:3306 mysql:5.7
```

for standard approach for image creation

0. mandetory to create Dockerfile

- docker file commands :
- 1. FROM : => indicates base image, from which we will create a custom image
- 2. ENV : => used to set environment variable , like mysql_root_password
- 3. COPY : => used to copy file from local machine to container specific folder
- 4. ADD : => used to copy file from local machine to container specific folder, and also download content from url and unzip file

- 5. EXPOSE => used to expose port 3306 from container for port forwarding , outside the container
- 6. to COPY everything from current directory on machine to home directory of container

> COPY . .

- 7. run a command when container starts

> CMD

- e.g:

> CMD ["node","server.js"] CMD node server.js

1. for our db image

- Dockerfile

```
# base image(mysql) used to create my image
FROM mysql:5.7

# ENV : used to set environment variable , like mysql_root_password
# set root password to root
ENV mysql_root_password "root"

# create a database named ecommercedb
ENV MYSQL_DATABASE "ecommercedb"

# copy db.sql file to container i.e folder /docker-entrypoint-initdb.d for
docker to run it
COPY ./db.sql /docker-entrypoint-initdb.d

# EXPOSE port 3306 from container for port forwarding
EXPOSE 3306
```

- sql file

```
create table user(id integer primary key auto_increment,firstname
varchar(40),lastname varchar(40),email varchar(50),password varchar(100));
create table product (id integer primary key auto_increment,title
varchar(100),price float);
```

- code on terminal

```
sudo docker image build . -t myappdb2
```

- where ,
- > . => indicates current directory, where we can find both Docker file and sql file
- > -t => t for tag : depicts the building image name
- so now to run our image i.e Container commands

```
$ : sudo docker container run -d -p 9090:3306 --name mydb myappdb
# here, myappdb : image name , mydb : container name , -d : detached , -p:
port mapping
# verify using ls command
$ : sudo docker container ls
CONTAINER ID        IMAGE               COMMAND                  CREATED
STATUS              PORTS                               NAMES
c121fb10f354        myappdb             "docker-entrypoint.s…"   49 seconds
ago      Up 46 seconds       33060/tcp, 0.0.0.0:9090->3306/tcp   mydb

# now open mysql
sudo mysql -u root --port 9090 --protocol tcp -p

# successfull, now can perform db operation here
```

2. create image for backend : node /express

- docker file

```
# create the backend image with node as base image
FROM node

# COPY everything from current directory on machine to home directory of
container
COPY . .

# expose port 3000
EXPOSE 3000

# run a command when container starts

CMD ["node","server.js"]
```

- code on terminal

```
# build iamge
sudo docker image build . -t mybackend
# check using ls
$ sudo docker image ls
# create a container
sudo docker container run -d -p 4000:4000 --name run mybackend
```

4. create index.html on container

- create Dockerfile

```
# use http as base image
FROM httpd

# copy index.html to the htdocs
COPY ./index.html /usr/local/apache2/htdocs/

# expose port 80
EXPOSE 80

# run apache in foreground(continuously )
CMD apachectl -D FOREGROUND
```

- create index.html
- commands on terminal

```
# build docker custom image
$ : sudo docker image build . -t myweb

# commands In Docker file

# Step 1/4 : FROM httpd

# Step 2/4 : COPY ./index.html /usr/local/apache2/htdocs/

# Step 3/4 : EXPOSE 80

#Step 4/4 : CMD apachectl -D FOREGROUND



$  sudo docker container run -d -p 3000:80 --name myweb myweb
```

5. to run angular application

- docker file commands

```
FROM httpd

COPY ./dist/mywebsite/ /usr/local/apache2/htdocs/

EXPOSE 80
```

```
CMD apachectl -D FOREGROUND
```

- terminal commands

```
$ : ng new mywebsite
# create application pages , then
$ :  ng serve --open

$ :~/dac/DevOps/docker-project/angular/mywebsite$ ng build --prod

# on ng build production level angualar copy is created , contains all
js,css files ,all ts files get compiled to js

$ :~/dac/DevOps/docker-project/angular/mywebsite/dist/mywebsite$ ls
3rdpartylicenses.txt  index.html
polyfills.35a5ca1855eb057f016a.js  styles.3ff695c00d717f2d2a11.css
favicon.ico           main.e225af7c98748907695e.js
runtime.acf0dec4155e77772545.js



# build custom image
$:~/dac/DevOps/docker-project/angular/mywebsite$ sudo docker image build .
-t angularwebsite

# check if image created
$: sudo docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
angularwebsite      latest              141617bee912        11 seconds
ago      138MB
# create container of custom image
dac/DevOps/docker-project/angular/mywebsite$ sudo docker container run -d -
p 8080:80 angularwebsite

# now open browser to check website on
> localhost:8080/
```

command to push/pull the images to/from docker hub

- 1. tag image name on machine as DockerId/image to push on docker hub

> sudo docker image tag dockerId/image-name

- 2. to push the image

> sudo docker image push dockerId/image-name

- 3. to pull the image stored on docker hub

- > sudo docker image pull dockerId/image-name

```
# login to docker
$  sudo Docker login
# tag image name on machine as DockerId/image to push on docker hub
 sudo docker image tag myappdb surajporje/myappdb

 # to push the image
 sudo docker image push surajporje/myappdb

 # now we can delete images on our image ,
 # now we can be download/pull from docker hub
 sudo docker image pull surajporje/myappdb
```

Question

1. What is Docker?

- Docker is a containerization platform which packages your application and all its dependencies together in the form of containers so as to ensure that your application works seamlessly in any environment be it development or test or production.
- Docker containers, wrap a piece of software in a complete filesystem that contains everything needed to run:
- code, runtime, system tools, system libraries etc. anything that can be installed on a server.
- This guarantees that the software will always run the same, regardless of its environment.

2. What is the need for DevOps?

- Nowadays instead of releasing big sets of features, companies are trying to see if small features can be transported to their customers through a series of release trains. This has many advantages like quick feedback from customers, better quality of software etc. which in turn leads to high customer satisfaction. To achieve this, companies are required to:
  - Increase deployment frequency
  - Lower failure rate of new releases
  - Shortened lead time between fixes
  - Faster mean time to recovery in the event of new release crashing
  - DevOps fulfills all these requirements and helps in achieving seamless software delivery.

3. How to build envrionment-agnostic systems with Docker?

- There are three main features helping to achieve that: + Volumes + Environment variable injection + Read-only file systems

4. Is there a way to identify the status of a Docker container?

- We can identify the status of a Docker container by running the command

> docker ps –a

- which will in turn list down all the available docker containers with its corresponding statuses on the host.
- From there we can easily identify the container of interest to check its status correspondingly.

5. What are the advantages of DevOps?

- Technical benefits:
  - Continuous software delivery
  - Less complex problems to fix
  - Faster resolution of problems
- Business benefits:
  - Faster delivery of features
  - More stable operating environments
  - More time available to add value (rather than fix/maintain)

6. What are the most common instructions in Dockerfile?

- Some of the common instructions in Dockerfile are as follows:
- 1. FROM: We use FROM to set the base image for subsequent instructions. In every valid Dockerfile, FROM is the first instruction.
- 2. LABEL: We use LABEL to organize our images as per project, module, licensing etc. We can also use LABEL to help in automation. In LABEL we specify a key value pair that can be later used for programmatically handling the Dockerfile.
- 3. RUN: We use RUN command to execute any instructions in a new layer on top of the current image. With each RUN command we add something on top of the image and use it in subsequent steps in Dockerfile.
- 4. CMD: We use CMD command to provide default values of an executing container. In a Dockerfile, if we include multiple CMD commands, then only the last instruction is used.

7. What are the various states that a Docker container can be in at any given point in time?

- There are four states that a Docker container can be in, at any given point in time. Those states are as given as follows: + Running + Paused + Restarting + Exited

8. What is Docker container?

- Docker containers include the application and all of its dependencies, but share the kernel with other containers, running as isolated processes in user space on the host operating system.
- Docker containers are not tied to any specific infrastructure: they run on any computer, on any infrastructure, and in any cloud.

9. What is Docker hub?

- Docker hub is a cloud-based registry service which allows you to link to code repositories, build your images and test them, stores manually pushed images, and links to Docker cloud so you can deploy images to your hosts. It provides a centralized resource for container image discovery, distribution and change management, user and team collaboration, and workflow automation throughout the development pipeline.

10. What is Docker image?

- Docker image is the source of Docker container. In other words, Docker images are used to create containers. Images are created with the build command, and they'll produce a container when started with run.
- Images are stored in a Docker registry such as registry.hub.docker.com because they can become quite large, images are designed to be composed of layers of other images, allowing a minimal amount of data to be sent when transferring images over the network.

11. What is the difference between the COPY and ADD commands in a Dockerfile?

- Although ADD and COPY are functionally similar, generally speaking, COPY is preferred.
- That's because it's more transparent than ADD. COPY only supports the basic copying of local files into the container, while ADD has some features (like local-only tar extraction and remote URL support) that are not immediately obvious. Consequently, the best use for ADD is local tar file auto-extraction into the image, as in ADD rootfs.tar.xz /.

12. What is the function of CI (Continuous Integration) server?

- CI server function is to continuously integrate all changes being made and committed to repository by different developers and check for compile errors. It needs to build code several times a day, preferably after every commit so it can detect which commit made the breakage if the breakage happens.

13. What type of applications - Stateless or Stateful are more suitable for Docker Container?

- It is preferable to create Stateless application for Docker Container. We can create a container out of our application and take out the configurable state parameters from application. Now we can run same container in Production as well as QA environments with different parameters. This helps in reusing the same Image in different scenarios. Also a stateless application is much easier to scale with Docker Containers than a stateful application.

14. Explain basic Docker usage workflow

- Everything starts with the Dockerfile. The Dockerfile is the source code of the Image. Once the Dockerfile is created, you build it to create the image of the container. The image is just the "compiled version" of the "source code" which is the Dockerfile. Once you have the image of the container, you should redistribute it using the registry. The registry is like a git repository -- you can push and pull images. Next, you can use the image to run containers. A running container is very similar, in many aspects, to a virtual machine (but without the hypervisor).

```
    +------------+  docker build   +--------------+  docker run -dt   +----
-------+  docker exec -it   +------+
    | Dockerfile | --------------> |    Image     | --------------->  |
Container | -----------------> | Bash |
    +------------+                 +--------------+                   +----
-------+                    +------+
                                      ^
                                      | docker pull
                                      |
                                   +--------------+
                                        |   Registry   |
                                        +--------------+
```

15. How will you monitor Docker in production?

- Docker provides tools like docker stats and docker events to monitor Docker in production. We can get reports on important statistics with these commands.

Docker stats: When we call docker stats with a container id, we get the CPU, memory usage etc of a container. It is similar to top command in Linux. Docker events: Docker events are a command to see the stream of activities that are going on in Docker daemon.

Some of the common Docker events are: attach, commit, die, detach, rename, destroy etc. We can also use various options to limit or filter the events that we are interested in.

16. What is Docker Swarm?

- Docker Swarm is native clustering for Docker. It turns a pool of Docker hosts into a single, virtual Docker host. Docker Swarm serves the standard Docker API, any tool that already communicates with a Docker daemon can use Swarm to transparently scale to multiple hosts.

17. What is Hypervisor?

- The hypervisor handles creating the virtual environment on which the guest virtual machines operate. It supervises the guest systems and makes sure that resources are allocated to the guests as necessary. The hypervisor sits in between the physical machine and virtual machines and provides virtualization services to the virtual machines. To realize it, it intercepts the guest operating system operations on the virtual machines and emulates the operation on the host machine's operating system.

- The rapid development of virtualization technologies, primarily in cloud, has driven the use of virtualization further by allowing multiple virtual servers to be created on a single physical server with the help of hypervisors, such as Xen, VMware Player, KVM, etc., and incorporation of hardware support in commodity processors, such as Intel VT and AMD-V.

18. What is the difference between Docker Image and Layer?

- Image: A Docker image is built up from a series of read-only layers Layer: Each layer represents an instruction in the image's Dockerfile. The below Dockerfile contains four commands, each of which creates a layer.

```
FROM ubuntu:15.04
COPY . /app
RUN make /app
CMD python /app/app.py
```

- Importantly, each layer is only a set of differences from the layer before it.

19. What is virtualisation?

- In its conceived form, virtualisation was considered a method of logically dividing mainframes to allow multiple applications to run simultaneously. However, the scenario drastically changed when companies and open source communities were able to provide a method of handling the privileged instructions in one way or another and allow for multiple operating systems to be run simultaneously on a single x86 based system.

- The net effect is that virtualization allows you to run two completely different OS on same hardware. Each guest OS goes through all the process of bootstrapping, loading kernel etc. You can have very tight security, for example, guest OS can't get full access to host OS or other guests and mess things up.

- The virtualization method can be categorized based on how it mimics hardware to a guest operating system and emulates guest operating environment. Primarily, there are three types of virtualization:

    - 1. Emulation
    - 2. Paravirtualization
    - 3. Container-based virtualization

kubernetes

1. what is kubernetes?

- Kubernetes is an open source system for
  - automating deployments,
  - scaling and management of containerized applications
- it is a container orchestration system
- using Kubernetes that means your application is following
  - 1. the microservices architecture and
  - 2. is already containerized

2. what does Orchestrator do ?

- 1. cloning container / replica creation
- 2. restarting container
- 3. destroying container
- 4. replacing existing container

3. which are the Container Orchestation tools

- 1. Docker Swarm
  - used when replica < 10,000
  - simpler than kubernetes
- 2. Kubernetes
  - used when replica(above) > 10,000
- 3. Apache Mesos
- 4. Marathon

4. work on K for developers

- 1. make K , application development ready ---> developer
  - here we have certification : (CKAD= Certified K Application Developer)
  - upload app to K
- 2. cluster administeration : setting,config cluster ---> administrator
  - here we have certification : (CKA : where Administrator) + (CKS : where S: security Specialist)

5. K history?

- developed by Google , in Borg project
- taken over by Cloud Native Computing Foundation (CNFC)
- open source/community

kubernetes ppt

yaml in docker ppt

Kubernetes Cluster demo

0. install virtual box

> https://www.virtualbox.org/wiki/Linux_Downloads

OR

> https://download.virtualbox.org/virtualbox/6.1.16/virtualbox-6.1_6.1.16-140961~Ubuntu~bionic_amd64.deb

1. install vagrant

> $ curl -O https://releases.hashicorp.com/vagrant/2.2.14/vagrant_2.2.14_x86_64.deb

> $ sudo apt install ./vagrant_2.2.14_x86_64.deb

1. Install kubernates from

> https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/

- OR use

```
    sudo apt-get update && sudo apt-get install -y apt-transport-https curl
    curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo
apt-key add -
    cat <<EOF | sudo tee /etc/apt/sources.list.d/kubernetes.list
    deb https://apt.kubernetes.io/ kubernetes-xenial main
    EOF
    sudo apt-get update
    sudo apt-get install -y kubelet kubeadm kubectl
    sudo apt-mark hold kubelet kubeadm kubectl
```

2. akms in pod.yaml file

- a : apiversion
- k : <kind of object>
- m : <metadata>
- spec : <specification of object>

4. create a directory for vagrant and use

```
# creates a Vagrantfile in the directory
vagrant init hashicorp/bionic64
```

- edit the vagrantFile as

```
Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/bionic64"
  config.vm.provision "shell", path: 'init2.sh'
  config.vm.define "node1" do |app|
      app.vm.hostname = 'node1'
      app.vm.network "private_network", ip: '192.168.2.201'
      app.vm.provider "virtualbox" do |vb|
          vb.name = "node1"
      end
  end

  config.vm.define "node2" do |app|
      app.vm.hostname = 'node2'
      app.vm.network "private_network", ip: '192.168.2.202'
      app.vm.provider "virtualbox" do |vb|
          vb.name = "node2"
      end
  end
end
```

- create init.sh file

```
sudo apt-get update

sudo apt-get install -y \
  apt-transport-https \
  ca-certificates \
  curl \
  gnupg-agent \
  software-properties-common

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add
-

sudo add-apt-repository \
  "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) \
  stable"

sudo apt-get update

sudo apt-get install -y docker-ce=5:19.03.12~3-0~ubuntu-bionic

sudo apt-mark hold docker-ce

sudo swapoff -a

sudo sed -i '/ swap / s/^\(.*\)$/#\1/g' /etc/fstab

curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-
key add -

cat <<EOF | sudo tee /etc/apt/sources.list.d/kubernetes.list
```

```
deb https://apt.kubernetes.io/ kubernetes-xenial main
EOF

sudo apt-get update

sudo apt-get install -y kubelet=1.19.1-00 kubeadm=1.19.1-00 kubectl=1.19.1-
00

sudo apt-mark hold kubelet kubeadm kubectl
```

- run vagrant commands on terminals

```
$  vagrant up
$  vagrant ssh node1
# initiate kube on terminal 1
sudo kubeadm init --apiserver-advertise-address=<ip-address> --pod-network-
cidr=10.244.0.0/16
# get all nodes
$ sudo kubectl get nodes

$ sudo kubectl describe node1



# initiate kube on terminal 2
$ vagrant ssh node2

$ sudo kubeadm init --apiserver-advertise-address=<ip-address> --pod-
network-cidr=10.244.0.0/16
```

on terminal 1

- create a yaml file for pod creation

> $ vim mypod.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: mypod

spec:
  containers:
  - name: mypod-container
    image: nginx
    ports:
    - containerPort: 80
    name: http-port
```

```
      protocol: TCP
```

- create pod using

> $ kubectl create -f mypod.yml

- get info of pod

> $ kubectl describe pod mypod

day2

1. creation Replica of POD using Replica Set and Replica Controller

- 1. demo for need for label in yml file
  - create a 3rd node and create a pod3.yaml file
  - edit pod.yaml file
  - to add replica set info in pod.yml metadata
    - like : desired pod count
  - so label is used to tag/identify the pod by replica set and added by replica set
  - > $ vim pod3.yml

    ```
    apiVersion: v1
    kind: Pod
    metadata:
      name: pod3
      labels:
        app: myapp
        type: frontend
        version: v1
    spec:
      containers:
      - name: mypod-container
        image: nginx
        ports:
        - containerPort: 80
          protocol: TCP
          # label/tag for RS to identify it
          name: http-port
    ```

  > :e :qw

- 2. create a replica set , for that
    - create rs.yml (rs = ReplicaSet)

> vim rs.yml

- command in yml

```
# for replica app/? , as they are complex object
apiVersion: app/v1
# for rs
kind: ReplicaSet

metadata:
  name: myrs
spec:
#specify no of pod
  replicas: 5
  selector:
   # label/tag for RS to identify it
    matchLabels:
      type: frontend
  template:
    metadata:
      name: pod-rs
      label:
        type: frontend
    spec:
      containers:
      - name: pod-1-container
        image: nginx
        ports:
        - containerPort: 80
          protocol: TCP
          name: http-port
```

- when pod created add to replica set , based on matchLabels: type
- commands

```
# create pods based on rs1.yml
 $ kubectl create -f rs1.yml
# shows replica set
 $ kubectl get replicaset

$ kubectl describe pods
$ kubectl get pod
```

CREATE DEPLOYMENT

1. create a deployment , for that

- create deployment.yml

> vim deployment.yml :i

- command in yml

```yaml
# for replica app/? , as they are complex object
apiVersion: app/v1
# for deployment
kind: Deployment

metadata:
  name: myapp-deployment
spec:
#specify no of pod
  replicas: 5
  selector:
    matchLabels:
      app: myapp
  template:
    metadata:
      name: myapp-pod
      label:
        type: myapp
    spec:
      containers:
      - name: myapp-container
        image: nginx
        ports:
        - containerPort: 80
          protocol: TCP
          name: http-port
```

> :w :wq

- cames out of vim
- adv of deployment
  - can create different versions of deployment
  - it uses RS at backend
- commands

```
$ kubectl get rs
$ kubectl get deployments
# now create one deployment
$ kubectl create -f deployment.yml
# see the running pods
```

```
$ kubectl get pods
# it show pod get recreated as terminated,for count remains constant
$ kubectl delete pod <pod-NAME>
```

installing minikube

```
# installing
$ curl -LO
https://storage.googleapis.com/minikube/releases/latest/minikube-linux-
amd64
$ sudo install minikube-linux-amd64 /usr/local/bin/minikube

# initialize

minikube start
```

kuber for service

1. services

- create a pod

```
mkdir practice
$ vim pod.yml
```

-

```
apiVersion: v1
kind: pod
metadata:
  name: pod1
  labels:
    app: web
spec:
  containers:
  - name: pod1-container
    image: nginx
    port
    - containerPort: 80
    name: http-port
    protocol: TCP
```

-

```
# create a pod
$ kubectl create -f pod.yml
# check
$ kubectl describe pod pod1
```

2.

- service.yml

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  type: NodePort
  selector:
    app: web
  ports:
  - port: 80
    targetPort: 80
    Protocol: TCP
```

- create service

```
# create a service
$ kubectl create -f service.yml
# check the service
$ kubectl get service

$ kubectl describe service nginx-service
```

final app

1. creating a pod using a docker custom image

- pod.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-myapp
  label:
    app: myapp
spec:
  containers:
  - name: container-myapp
    image: surajporje/myapp
    ports:
     - port: 80
       targetPort: 80
       Protocol: TCP
```

2. application running in pod with docker image, and expose to outside

```
apiVersion: v1
kind: Service
metadata:
  name: service-myapp
spec:
  type: NodePort
  selector:
    app: myapp
  ports:
  - port: 80
    targetPort: 80
    Protocol: TCP
```

```
kubectl create -f service.yml

kubectl describe service
kubectl get pod

kubectl delete pod  <metadata:name>
```

3. to run a deployment with docker image

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myapp-d
spec:
  replicas : 5
  selector:
    matchLabels:
      app: myapp
  template:
    metadata:
```

```
    name: pod1
    labels:
      app: myapp
  spec:
    containers:
    - name: myapp-container
    image: surajporje/myapp
    ports:
    - containerPort: 80
      name: http-port
      protocol: TCP
```

-

```
# create a deployment
$ kubectl create -f deployment.yml

$ kubectl get pods
$ kubectl get service
```

Kubernetes Interview Questions and Answers

1. What is Kubernetes?

- Kubernetes is an open-source container orchestration tool or system that is used to automate tasks such as the management, monitoring, scaling, and deployment of containerized applications.
- It is used to easily manage several containers (since it can handle grouping of containers), which provides for logical units that can be discovered and managed.

2. What are K8s?

- K8s is another term for Kubernetes.

3. What is orchestration when it comes to software and DevOps?

- Orchestration refers to the integration of multiple services that allows them to automate processes or synchronize information in a timely fashion.
- Say, for example, you have six or seven microservices for an application to run.
- If you place them in separate containers, this would inevitably create obstacles for communication

- Orchestration would help in such a situation by enabling all services in individual containers to work seamlessly to accomplish a single goal.

4. How are Kubernetes and Docker related?

- 1. Docker is an open-source platform used to handle software development.
- Its main benefit is that it packages the settings and dependencies that the software/application needs to run into a container,

- which allows for portability and several other advantages.
- 2. Kubernetes allows for the manual linking and orchestration of several containers, running on multiple hosts that have been created using Docker.

5. What are the main differences between the Docker Swarm and Kubernetes?

- - Docker Swarm is Docker's native, open-source container orchestration platform that is used to cluster and schedule Docker containers.
- Swarm differs from Kubernetes in the following ways:
- 1. Docker Swarm is more convenient to set up but doesn't have a robust cluster,
- while Kubernetes is more complicated to set up but the benefit of having the assurance of a robust cluster
- 2. Docker Swarm can't do auto-scaling (as can Kubernetes); however, Docker scaling is five times faster than Kubernetes
- 3. Docker Swarm doesn't have a GUI;
- Kubernetes has a GUI in the form of a dashboard
- 4. Docker Swarm does automatic load balancing of traffic between containers in a cluster,
- while Kubernetes requires manual intervention for load balancing such traffic
- 5. Docker can deploy rolling updates but can't deploy automatic rollbacks;
- Kubernetes can deploy rolling updates as well as automatic rollbacks

6. What are the main components of Kubernetes architecture?

- There are two primary components:
- the master node and
- the worker node.
- Each of these components has individual components in them.

7. What is a node in Kubernetes?

- A node is the smallest fundamental unit of computing hardware.
- It represents a single machine in a cluster, which could be a physical machine in a data center or a virtual machine from a cloud provider.
- Each machine can substitute any other machine in a Kubernetes cluster.
- The master in Kubernetes controls the nodes that have containers.

8. What does the node status contain?

- The main components of a node status are Address, Condition, Capacity, and Info.

9. What process runs on Kubernetes Master Node?

- The Kube-api server process runs on the master node and serves to scale the deployment of more instances.

10. What is a pod in Kubernetes?

- Pods are high-level structures that wrap one or more containers.
- This is because containers are not run directly in Kubernetes.
- Containers in the same pod share a local network and the same resources, allowing them to easily communicate with other containers in the same pod as if they were on the same machine while at

the same time maintaining a degree of isolation.

11. What is the job of the kube-scheduler?

- The kube-scheduler assigns nodes to newly created pods.

12. What is a cluster of containers in Kubernetes?

- A cluster of containers is a set of machine elements that are nodes.
- Clusters initiate specific routes so that the containers running on the nodes can communicate with each other.
- In Kubernetes, the container engine (not the server of the Kubernetes API) provides hosting for the API server.

13. What is the Google Container Engine?

- The Google Container Engine is an open-source management platform tailor-made for Docker containers and clusters to provide support for the clusters that run in Google public cloud services.

14. What are Daemon sets?

- A Daemon set is a set of pods that runs only once on a host.
- They are used for host layer attributes like a network or for monitoring a network, which you may not need to run on a host more than once.

15. What is 'Heapster' in Kubernetes?

- A Heapster is a performance monitoring and metrics collection system for data collected by the Kublet.
- This aggregator is natively supported and runs like any other pod within a Kubernetes cluster, which allows it to discover and query usage data from all nodes within the cluster.

16. What is a Namespace in Kubernetes?

- Namespaces are used for dividing cluster resources between multiple users.
- They are meant for environments where there are many users spread across projects or teams and provide a scope of resources.

17. Name the initial namespaces from which Kubernetes starts?

- Default
  - Kube – system
  - Kube – public

18. What is the Kubernetes controller manager?

- The controller manager is a daemon that is used for embedding core control loops, garbage collection, and Namespace creation.
- It enables the running of multiple processes on the master node even though they are compiled to run as a single process.

19. What are the types of controller managers?

- The primary controller managers that can run on the master node are the
  - endpoints controller,
  - service accounts controller,
  - namespace controller,
  - node controller,
  - token controller, and
  - replication controller.

20. What is etcd?

- Kubernetes uses etcd as a distributed key-value store for all of its data,
- including metadata and configuration data, and allows nodes in Kubernetes clusters to read and write data.
- Although etcd was purposely built for CoreOS, it also works on a variety of operating systems (
- e.g., Linux, BSB, and OS X, because it is open-source.
- Etcd represents the state of a cluster at a specific moment in time and is a canonical hub for state management and cluster coordination of a Kubernetes cluster.

21. What are the different services within Kubernetes?

- Different types of Kubernetes services include: + Cluster IP service + Node Port service + External Name Creation service and + Load Balancer service

22. What is ClusterIP?

- The ClusterIP is the default Kubernetes service
- that provides a service inside a cluster (with no external access) that other apps inside your cluster can access.

23. What is NodePort?

- The NodePort service is the most fundamental way to get external traffic directly to your service.
- It opens a specific port on all Nodes and forwards any traffic sent to this port to the service.

24. What is the LoadBalancer in Kubernetes?

- The LoadBalancer service is used to expose services to the internet.
- A Network load balancer, for example, creates a single IP address that forwards all traffic to your service.

25. What is a headless service?

- A headless service is used to interface with service discovery mechanisms without being tied to a ClusterIP, therefore allowing you to directly reach pods without having to access them through a proxy.
- It is useful when neither load balancing nor a single Service IP is required.

26. What is Kubelet?

- The kubelet is a service agent that controls and maintains a set of pods by watching for pod specs through the Kubernetes API server.

- It preserves the pod lifecycle by ensuring that a given set of containers are all running as they should.
- The kubelet runs on each node and enables the communication between the master and slave nodes.

27. What is Kubectl?

- Kubectl is a CLI (command-line interface)
- that is used to run commands against Kubernetes clusters.
- As such, it controls the Kubernetes cluster manager through different create and manage commands on the Kubernetes component

28. Give examples of recommended security measures for Kubernetes.

- Examples of standard Kubernetes security measures
- include defining resource quotas, support for auditing, restriction of etcd access, regular security updates to the environment, network segmentation, definition of strict resource policies, continuous scanning for security vulnerabilities, and using images from authorized repositories.

29. What is Kube-proxy?

- Kube-proxy is an implementation of a load balancer and network proxy used to support service abstraction with other networking operations.
- Kube-proxy is responsible for directing traffic to the right container based on IP and the port number of incoming requests.

30. How can you get a static IP for a Kubernetes load balancer?

- A static IP for the Kubernetes load balancer can be achieved by changing DNS records since the Kubernetes Master can assign a new static IP address.

Jenkins, need for it?

- ppt on devops
- ppt on jenkins
-

1. SCM : Source code management

day2

Q. Identify the type of pipelines in Jenkins?

1. Declarative pipeline
2. Scripted pipeline
3. CI CD pipeline (Continuous Integration Continuous Delivery)
4. All of the above

- Answer :- 4 All of the above are the types of pipelines in Jenkins.

jenkins questions

1. What is Jenkins?

- Jenkins is an open source continuous integration tool written in Java. It keeps a track on version control system and to initiate and monitor a build system if changes occur.

2. What is the difference between Maven, Ant and Jenkins?

- The most basic difference is:
- Maven and Ant are Build Technologies whereas Jenkins is a continuous integration tool.

3. Which SCM tools does Jenkins support?

- Jenkins supports the following SCM tools:

  - AccuRev
  - CVS
  - Subversion
  - Git
  - Mercurial
  - Perforce
  - Clearcase
  - RTC

4. What is continuous integration in Jenkins?

- In software development, multiple developers or teams work on different segments of same web application so you have to perform integration test by integrating all modules. In order to do that an automated process for each piece of code is performed on daily bases so that all your codes get tested. This process is known as continuous integration.

5. What is the relation between Hudson and Jenkins?

- Hudson was the earlier name and version of current Jenkins. After some issue, the project name was changed from Hudson to Jenkins.

6. What is the requirement for using Jenkins?

- For using Jenkins, you have to need a source code repository which is accessible. For example, a Git repository and a working build script, e.g., a Maven script, checked into the repository.

7. What are the advantages of Jenkins?

- Advantage of Jenkins includes:

    - Bugs tracking are easy at early stage in development environment.
    - Provides a large numbers of plugin support.
    - Iterative improvement to the code.
    - Build failures are cached at integration stage.
    - For each code commit changes an automatic build report notification generates.
    - To notify developers about build report success or failure, it is integrated with LDAP mail server.
    - Achieves continuous integration agile development and test driven development.
    - With simple steps, maven release project is automated.

8. How to make sure that your project builds doesn?t break in Jenkins?

- You must follow these steps to make sure that your project builds doesn?t break in Jenkins:
-     1. First, perform successful clean install on your local machine with all unit tests. Check all your code changes.
-     2. Synchronize with repository to make sure that all required config and POM changes and any difference is checked into the repository.

9. How can you move or copy Jenkins from one server to another?

- Follow these steps to move or copy Jenkins from one server to another:
-     1. First, copy the related job directory and slide a job from one installation of Jenkins to another.
-     2. Make a copy of an already existing job by making clone of a job directory by a different name.
-     3. Renaming an existing job by rename a directory.

10. Which commands can be used to start Jenkins manually?

- You can use any one of the following commands to start Jenkins manually:
-     1. (Jenkins_url)/restart: Forces a restart without waiting for builds to complete.
-     2. (Jenkin_url)/safeRestart: Allows all running builds to complete.

11. What are the most useful plugins in Jenkins?

- Some most useful plugins in Jenkins:
    - Maven 2 project

    - Amazon EC2
    - HTML publisher
    - Copy artifact
    - Join
    - Green Balls

12. How to create a backup and copy files in Jenkins?

- If you want to create a back-up of your Jenkins setup, just copy the directory that saves all the setting, build artifacts and logs of Jenkins in its home directory. You can also copy a job directory to clone or replicate a job or rename the directory.

13. How can you clone a Git repository via Jenkins?

- If you want to clone a Git repository via Jenkins, you have to enter the e-mail and user name for your Jenkins system. Switch into your job directory and execute the "git config" command for that.

14. How can you setup Jenkins jobs?

- Follow these steps:
    - Select new item from the menu.
    - After that enter a name for the job and select free-style job.
    - Then click OK to create new job in Jenkins.
    - The next page enables you to configure your job.

15. What are the two components Jenkins is mainly integrated with?

- Jenkins is integrated with these two components:
    - Version Control system like GIT,SVN
    - And build tools like Apache Maven.