Devops-Question [Shubham Kumbhar]

1. what is DevOps?

```
Ans: DevOps is a set of software development practices that combines software development and information technology operations to shorten the systems development life cycle while delivering features, fixes, and updates frequently in
```

More About Devops

2. why do we need DevOps?

```
Ans: Five reasons why the industry has been so quick to adopt DevOps principles
1. Shorter Development Cycles, Faster Innovation
2. Reduced Deployment Failures, Rollbacks, and Time to Recover
3. Improved Communication and Collaboration
4. Increased Efficiencies
5. Reduced Costs and IT Headcount
```

3. Mention the key aspects or principle behind DevOps?

```
Ans:
    The key aspects or principle behind DevOps is

1. Infrastructure as code
2. Continuous deployment
3. Automation
4. Monitoring
5. Security
```

4. List out some of the popular tools for DevOps?

```
Ans:
Configuration management tools
    - Ansible
    - CHEF
    - CFEngine
    - Puppet
    - SALTSTACK
    - JUJU
    - RUDDER
CI/CD Tools
    Jenkins
    GoCD
    Drone.io
    TeamCity
    Wercker
    Codeship
    Travis CI
    CircleCI
    Bamboo
    Gradle

Container orchestration tools
    - Kubernetes
    - Docker Swarm
    - Google Container Engine
    - Mesosphere Marathon

version control tools
    Git
    Git hub
    Beanstalk
    Bitbucket
    PerForce
    Apache Subversion
    AWS CodeCommit
    Git-lab
```

5. what is a version control system?

```
Ans:
A version control system allows users to keep track of the changes in software development projects, and enable them to collaborate on those projects. Using it, the developers can work together on code and separate their tasks through

There can be several branches in a version control system, according to the number of collaborators. The branches maintain individuality as the code changes remain in a specified branch(s).

Developers can combine the code changes when required. Further, they can view the history of changes, go back to the previous version(s) and use/manage code in the desired fashion.
```

6. What is Git and explain the difference between Git and SVN?

```
Ans:
Git is a free, open source distributed version control system tool designed to handle everything from small to very large projects with speed and efficiency. It was created by Linus Torvalds in 2005 to develop Linux Kernel. Git has the fun
```

Difference between Git and SVN

```
Git and SVN are both software. Git is SCM, source code management , and a distributed revision control system. SVN is a revision control and software versioning system.

One of the most notable differences when switching to Git is its speed. Since the whole repository is stored locally on the developer’s machine, he or she can work for days with a very poor internet connection. Creating branches is lightn

Since Git encourages the use of branches, we can’t forget to give a shout-out to its merge capabilities. SVN before version 1.5 only did two-way merges that involved a change set applied to the current codebase, because it didn’t store m
```

7. what language is used in Git?

```
Ans:
    Git’s Source code is hosted on Github here: git/git

So, From that these are the languages used in that repository:

C - 45%
Shell - 35%
Perl - 8%
Tcl - 5%
Python - 2%
C ++ - 2%
```

GIT-Source Code

8. what are the advantages and disadvantages of using Git?

Ans:
GIT is a distributed version control system and source code management system with an emphasis to handle small and large projects with speed and efficiency.

let's start with advantages :

Data redundancy and replication
- High availability
- Only one.git directory per repository
- Superior disk utilization and network performance
- Collaboration friendly
- Any sort of projects can use GIT

There are very few disadvantages of using GIT:

- Git is less preferred for handling extremely large files or frequently changing binary files .

- GIT does not support 'commits' across multiple branches or tags.

9. Explain the difference between git pull and git fetch?

Ans:
Before we talk about the differences between these two commands, let's stress their similarities: both are used to download new data from a remote repository.

-> Downloading data is an essential step in your daily work - because the remote data you are looking at in your local repository is just a "snapshot". It's only as up-to-date as the last time you explicitly downloaded fresh data from the re

Let's now look at the fine but important differences between "fetch" and "pull".

Fetch
```
$ git fetch origin
```

-> git fetch really only downloads new data from a remote repository - but it doesn't integrate any of this new data into your working files. Fetch is great for getting a fresh view on all the things that happened in a remote repository.

-> Due to it's "harmless" nature, you can rest assured: fetch will never manipulate, destroy, or screw up anything. This means you can never fetch often enough.

Pull
```
$ git pull origin master
```

-> git pull, in contrast, is used with a different goal in mind: to update your current HEAD branch with the latest changes from the remote server. This means that pull not only downloads new data; it also directly integrates it into your cu

-> Since "git pull" tries to merge remote changes with your local ones, a so-called "merge conflict" can occur. Check out our in-depth tutorial on How to deal with merge conflicts for more information.

-> Like for many other actions, it's highly recommended to start a "git pull" only with a clean working copy. This means that you should not have any uncommitted local changes before you pull. Use Git's Stash feature to save your local c

10. what is Docker?

Ans:
Docker is a tool designed to make it easier to create, deploy, and run applications by using containers. Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, an

11. what is Docker image?

Ans:

A Docker image is made up of multiple layers. A user composes each Docker image to include system libraries, tools, and other files and dependencies for the executable code. Image developers can reuse static image layers for different

Most Docker images start with a base image, although a user can build one entirely from scratch, if desired. Each image has one readable/writable top layer over static layers. Layers are added to the base image to tailor the code to run

When a new container is created from an image, a writable layer is also created. This layer is called the container layer, and it hosts all changes made to the running container. This layer can store newly written files, modifications to exis

12. what is Docker Container?

Ans:

Package Software into Standardized Units for Development, Shipment and Deployment
A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another. A Docker container image is a lightweight, standalone, ex

Container images become containers at runtime and in the case of Docker containers - images become containers when they run on Docker Engine. Available for both Linux and Windows-based applications, containerized software will al

Docker containers that run on Docker Engine:

Standard: Docker created the industry standard for containers, so they could be portable anywhere
Lightweight: Containers share the machine's OS system kernel and therefore do not require an OS per application, driving higher server efficiencies and reducing server and licensing costs
Secure: Applications are safer in containers and Docker provides the strongest default isolation capabilities in the industry

13. What are the components of Docker Architecture and explain?

Ans:
Docker Client
The Docker client enables users to interact with Docker. The Docker client can reside on the same host as the daemon or connect to a daemon on a remote host. A docker client can communicate with more than one daemon. The Docker

The main purpose of the Docker Client is to provide a means to direct the pull of images from a registry and to have it run on a Docker host. Common commands issued by a client are:

docker build
docker pull
docker run
DockerHost
The Docker host provides a complete environment to execute and run applications. It comprises of the Docker daemon, Images, Containers, Networks, and Storage. As previously mentioned, the daemon is responsible for all container-rel

Docker Objects
Various objects are used in the assembling of your application. The main requisite Docker objects are:

Images
Images are a read-only binary template used to build containers. Images also contain metadata that describe the container's capabilities and needs. Images are used to store and ship applications. An image can be used on its own to buil

Containers
Containers are encapsulated environments in which you run applications. The container is defined by the image and any additional configuration options provided on starting the container, including and not limited to the network connec

Networking
Docker implements networking in an application-driven manner and provides various options while maintaining enough abstraction for application developers. There are basically two types of networks available - the default Docker netw

The other type of networks is user-defined networks. Administrators can configure multiple user-defined networks. There are three types:

Bridge network: Similar to the default bridge network, a user-defined Bridge network differs in that there is no need for port forwarding for containers within the network to communicate with each other. The other difference is that it ha
Overlay network: An Overlay network is used when you need containers on separate hosts to be able to communicate with each other, as in the case of a distributed network. However, a caveat is that swarm mode must be enabled for a
Macvlan network: When using Bridge and Overlay networks a bridge resides between the container and the host. A Macvlan network removes this bridge, providing the benefit of exposing container resources to external networks withou

Storage
You can store data within the writable layer of a container but it requires a storage driver. Being non-persistent, it perishes whenever the container is not running. Moreover, it is not easy to transfer this data. In terms of persistent stora

Data Volumes: Data Volumes provide the ability to create persistent storage, with the ability to rename volumes, list volumes, and also list the container that is associated with the volume. Data Volumes sit on the host file system, outside
Data Volume Container: A Data Volume Container is an alternative approach wherein a dedicated container hosts a volume and to mount that volume to other containers. In this case, the volume container is independent of the applicatio
Directory Mounts: Another option is to mount a host's local directory into a container. In the previously mentioned cases, the volumes would have to be within the Docker volumes folder, whereas when it comes to Directory Mounts any d
Storage Plugins: Storage Plugins provide the ability to connect to external storage platforms. These plugins map storage from the host to an external source like a storage array or an appliance. A list of storage plugins can be found on D

Storage Plugins
There are storage plugins from various companies to automate the storage provisioning process. For example,
HPE 3PAR
EMC (ScaleIO, XtremIO, VMAX, Isilon)
NetApp
There are also plugins that support public cloud providers like:

Azure File Storage
Google Compute Platform.
Docker Registries
Docker registries are services that provide locations from where you can store and download images. In other words, a Docker registry contains repositories that host one or more Docker Images. Public Registries include Docker Hub an

```
docker push
docker pull
docker run
```

Service Discovery
Service Discovery allows containers to find out about the environment they are in and find other services offered by other containers.

It is an important factor when trying to build scalable and flexible applications.

14. What is the lifecycle of Docker Container?

Ans:

** Create container

Create a container to run it later on with required image.

```
docker create --name <container-name> <image-name>
```

Run docker container
Run the docker container with the required image and specified command / process. '-d' flag is used for running the container in background.

```
docker run -it -d --name <container-name> <image-name> bash
```

** Pause container
Used to pause the processes running inside the container.

```
docker pause <container-id/name>
```

**Unpause container
Used to unpause the processes inside the container.

```
docker unpause <container-id/name>
```

**Start container
Start the container, if present in stopped state.

```
docker start <container-id/name>
```

**Stop container
To stop the container and processes running inside the container:

```
docker stop <container-id/name>
```

To stop all the running docker containers

```
docker stop $(docker ps -a -q)
```

**Restart container
It is used to restart the container as well as processes running inside the container.

```
docker restart <container-id/name>
```

**Kill container
We can kill the running container.

```
docker kill <container-id/name>
```

**Destroy container
Its preferred to destroy container, only if present in stopped state instead of forcefully destroying the running container.

```
docker rm <container-id/name>
```

To remove all the stopped docker containers

```
docker rm $(docker ps -q -f status=exited)
```

15. Explain what is continuous integration?

Ans:
Continuous Integration (CI) is a development practice where developers integrate code into a shared repository frequently, preferably several times a day. Each integration can then be verified by an automated build and automated tes

One of the key benefits of integrating regularly is that you can detect errors quickly and locate them more easily. As each change introduced is typically small, pinpointing the specific change that introduced a defect can be done quickly.

In recent years CI has become a best practice for software development and is guided by a set of key principles. Among them are revision control, build automation and automated testing.

Additionally, Continuous Deployment and Continuous Delivery have developed as best-practices for keeping your application deployable at any point or even pushing your main codebase automatically into production whenever new chan

Continuous Integration doesn't get rid of bugs, but it does make them dramatically easier to find and remove.

--> Continuous Integration brings multiple benefits to your organization:

1.Say goodbye to long and tense integrations
2.Increase visibility enabling greater communication
3.Catch issues early and nip them in the bud
4.Spend less time debugging and more time adding features
Build a solid foundation
Stop waiting to find out if your code's going to work
Reduce integration problems allowing you to deliver software more rapidly

16. what is a Jenkins Pipeline?

Ans:
In Jenkins, a pipeline is a group of events or jobs which are interlinked with one another in a sequence.

In simple words, Jenkins Pipeline is a combination of plugins that support the integration and implementation of continuous delivery pipelines using Jenkins. A pipeline has an extensible automation server for creating simple or complex c

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

Ans:
To Start with all four e.g. ANT, Maven, Jenkins and Hudson are tools to help Java developers on build, unit testing, continuous integration and project management.

Main difference between ANT and Maven is that in ANT you need to define every thing i.e. source directory, build directory, target directory etc while Maven adopts principle of Convention over configuration.

Maven also provides dependency management, standard project layout and project management. Maven has predefined project structure i.e. standard directory for source files, test files and resources.

On the other hand, Jenkins and Hudson are Continuous Integration tool, which gives you power to automate your build and deployment process. By using Jenkins or Hudson you can trigger build whenever developer commit code, to see i

====================================

18. What is Scrum?

Ans:
    Scrum is a framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value.

Scrum itself is a simple framework for effective team collaboration on complex products. Scrum co-creators Ken Schwaber and Jeff Sutherland have written The Scrum Guide to explain Scrum clearly and succinctly. This Guide contains

Scrum is:

Lightweight
Simple to understand
Difficult to master