In azure I have basic knowledge like VM's, storage accounts, vnet, NSG. I have not worked extensively on it. Most of our applications are hosted now in azure and we have reader access to the portal.

Storage account act has a container to store blobs, file, tables

NSG is firewall were we can control the traffic inbound and outbound

Vnet is virtual network which is main network for an subscription and from that vnet multiple subnets will be created.

Kubernetes:

Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications.

API server act as the frontend which receives the input from kubcetl command and interacts between Kubernetes master and worker nodes

Controller is the brain which monitors the Kubernetes cluster and pods and acts accordingly using replication controller, node controller etc.

Etcd is a database which maintains all data of entire cluster like which pod is running on which node and what is the number replicas set for that pod etc.

Scheduler schedules the pods on respective nodes depending on the utilization in nodes

Kubelet is agent which runs in worker nodes and sends and recevies information from api server

Kubeproxy handles the networking stuff within the pods

Docker:

We have containerized the internal applications which we were running as a VM.

How ? We collaborated with the developer to get requirement (like dependencies, port configured etc ) and we created the docker file to create the image and did a testing by running in a container.

Kubernetes:

We deployed Kubernetes cluster to test our ansible script using ansible molecule. When ever we run a molecule it creates a pod and execute the ansible script inside the pod.

Apart of this few applications we are performing POC to migrate from docker to kubernetes, we are using docker swarm most the applications require to be highly available and also leavarge the monitoring application

Livelyness probe - montioring probe

Rediness probe - montioring kube cluster

Difference b/w docker swarm and kubernetes

Jenkins:

we are using jenkins for continuous integration. We have configured jenkins pipeline for our ansible automation role whenever we push our changes to sourcecode(bitbucket) build will be triggered in jenkins after the succesfull build we will push to master.

Also we have configured the pipeline jobs for other teams based on there requirement.

Ansible :

Ansible we are using for automating installation/upgradtion and even for some the application config changes(plugin installation) . For this we create a molecule scenario to perform the test before deploying to the actual environment.

We also started migration existing ansible playbooks and roles to ansible collection.

Python script: also we have written to run the customized ansible module to run ansible playbook based on the requirement.

Jira & confluence: we also own atlassian tool set where are administrating all the atlassian tools. We are the primary support for customizing the project based on users requirement. We also write groovy script for some of the autoamstion/customization in jira.

**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:

* 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
* Docker Swarm can’t do auto-scaling (as can Kubernetes); however, Docker scaling is five times faster than Kubernetes
* Docker Swarm doesn’t have a GUI; Kubernetes has a GUI in the form of a dashboard
* Docker Swarm does automatic load balancing of traffic between containers in a cluster, while Kubernetes requires manual intervention for load balancing such traffic
* Docker requires third-party tools like ELK stack for logging and monitoring, while Kubernetes has integrated tools for the same
* Docker Swarm can share storage volumes with any container easily, while Kubernetes can only share storage volumes with containers in the same pod
* Docker can deploy rolling updates but can’t deploy automatic rollbacks; Kubernetes can deploy rolling updates as well as automatic rollbacks

**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

**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.

**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.

**What is Kubectl?**

Kubectl is a CLI (command-line interface) that is used to run commands against Kubernetes clusters.

**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.

**What are the main drawbacks of Docker?**

Some notable drawbacks of Docker are:

* Doesn't provide a storage option
* Offer a poor monitoring option.
* No automatic rescheduling of inactive Nodes
* Complicated automatic horizontal scaling set up

**What is Docker image?**

The Docker image help to create Docker containers. You can create the Docker image with the build command. Due to this, it creates a container that starts when it begins to run. Every docker images are stored in the Docker registry.

**What are the common instruction in Dockerfile?**

The common instruction in Dockerfile are: FROM, LABEL, RUN, and CMD.

**What is Hypervisor?**

The hypervisor allows you to create a virtual environment in which the guest virtual machines operate. It controls the guest systems and checks if the resources are allocated to the guests as necessary.

**What is Virtualization?**

Virtualization is a method of logically dividing mainframes to allow multiple applications to run simultaneously.

However, this scenario changed when companies and open source communities were able to offer a method of handling privileged instructions. It allows multiple OS to run simultaneously on a single x86 based system.

**Write a Docker file to create and copy a directory and built it using python modules?**

FROM pyhton:2.7-slim

WORKDIR /app

COPY . /app

docker build –tag

**Where the docker volumes are stored?**

You need to navigate:

/var/lib/docker/volumes

**Can you tell something about docker container?**

* In simplest terms, docker containers consist of applications and all their dependencies.
* They share the kernel and system resources with other containers and run as isolated systems in the host operating system.
* The main aim of docker containers is to get rid of the infrastructure dependency while deploying and running applications. This means that any containerized application can run on any platform irrespective of the infrastructure being used beneath.
* Technically, they are just the runtime instances of docker images

**What are docker images?**

They are executable packages(bundled with application code & dependencies, software packages, etc.) for the purpose of creating containers. Docker images can be deployed to any docker environment and the containers can be spun up there to run the application.

**How many Docker components are there?**

There are three docker components, they are - Docker Client, Docker Host, and Docker Registry.

* **Docker Client:** This component performs “build” and “run” operations for the purpose of opening communication with the docker host.
* **Docker Host:** This component has the main docker daemon and hosts containers and their associated images. The daemon establishes a connection with the docker registry.
* **Docker Registry:** This component stores the docker images. There can be a public registry or a private one. The most famous public registries are Docker Hub and Docker Cloud.

**Differentiate between virtualization and containerization.**

The question indirectly translates to explaining the difference between virtual machines and Docker containers.

|  |  |
| --- | --- |
| **Virtualization** | **Containerization** |
| This helps developers to run and host multiple **OS** on the hardware of a single physical server. | This helps developers to deploy multiple **applications** using the same operating system on a single virtual machine or server. |
| **Hypervisors** provide overall virtual machines to the guest operating systems. | **Containers** ensure isolated environment/ user spaces are provided for running the applications. Any changes done within the container do not reflect on the host or other containers of the same host. |
| These virtual machines form an **abstraction of the system hardware** **layer**this means that each virtual machine on the host acts like a physical machine. | Containers form **abstraction of the application** **layer** which means that each container constitutes a different application. |

**Differentiate between COPY and ADD commands that are used in a Dockerfile?**

Both the commands have similar functionality, but COPY is more preferred because of its higher transparency level than that of ADD.

COPY provides just the basic support of copying local files into the container whereas ADD provides additional features like remote URL and tar extraction support.

Git Rebase vs. Merge: Which to Use

Some developers believe you should always rebase. And others think that you should always merge. Each side has benefits.

Benefits

Here are the top three benefits for Git rebase and for Git merge.

Git Rebase

* Streamlines a potentially complex history.
* Avoids merge commit “noise” in busy repos with busy branches.
* Cleans intermediate commits by making them a single commit, which can be helpful for DevOps teams.

Git Merge

* Simple and familiar.
* Preserves complete history and chronological order.
* Maintains the context of the branch.

**Explain the git pull command.**

Git pull is used to fetch and merge changes from the remote repository to the local repository. Git pull is a combination of two commands: git fetch; followed by git merge

**Difference between git fetch and git pull.**

|  |  |
| --- | --- |
| **Git fetch** | **Git pull** |
| * Git fetches only downloads new data from a remote repository. * It does not integrate any of these new data into your working files. * Can be done any time to update the remote-tracking branches   **Command - git fetch origin**  **git fetch –-all** | * Git pull updates the current HEAD branch with the latest changes from the remote server. * Downloads new data and integrate it with the current working files. * Tries to merge remote changes with your local ones.   **Command - git pull origin master** |

**What is a merge conflict in Git?**

A [merge conflict](https://www.simplilearn.com/tutorials/devops-tutorial/devops-interview-questions) is an event that takes place when Git is unable to resolve differences in code between the two commits automatically.

Git is able to automatically merge the changes only if the commits are on different lines or branches.

**What is the process to revert a commit that has already been pushed and made public?**

There are two processes through which you can revert a commit:

1. Remove or fix the bad file in a new commit and push it to the remote repository. Then commit it to the remote repository using:

**git commit –m “commit message”**

2. Create a new commit to undo all the changes that were made in the bad commit. Use the following command:

**git revert <commit id>**

**What does the git reset --mixed and git merge --abort commands do?**

**git reset --mixed** is used to undo changes made in the working directory and staging area.

**git merge --abort** helps stop the merge process and return back to the state before the merging began.

**What is the difference between git merge and git rebase?**

To incorporate new commits into your feature branch, you use ***merge***

Chart

Description automatically generated

* Creates an extra merge commit every time you need to incorporate changes
* Pollutes your feature branch history

As an alternative to merging, you can rebase the feature branch into master.

Chart

Description automatically generated

* Incorporates all the new commits in the master branch
* Rewrites the project history by creating brand new commits for each commit in the original branch

**What is “git cherry-pick”?**

The command **git cherry-pick**enables you to pick up commits from a branch within a repository and apply it to another branch. This command is useful to undo changes when any commit is accidentally made to the wrong branch. Then, you can switch to the correct branch and use this command to cherry-pick the commit.

**Describe the branching strategies you have used.**

* **Feature branching** – A feature branch model keeps all of the changes for a particular feature inside of a branch. When the feature is fully tested and validated by automated tests, the branch is then merged into master.
* **Task branching** – In this model, each task is implemented on its own branch with the task key included in the branch name. It is easy to see which code implements which task, just look for the task key in the branch name.
* **Release branching** – Once the develop branch has acquired enough features for a release, you can clone that branch to form a Release branch. Creating this branch starts the next release cycle, so no new features can be added after this point, only bug fixes, documentation generation, and other release-oriented tasks should go in this branch. Once it is ready to ship, the release gets merged into master and tagged with a version number. In addition, it should be merged back into the develop branch, which may have progressed since the release was initiated.