**Docker:**

1. Managing the environment for your applications across the different system
2. Docker Hub -> repo of all the official and public docker images
3. Terminologies
   1. Dockerfile -> is the place where you provide the list of softwares and execution commands needed to run the application
   2. Image -> snapshot of your applications
   3. Containers -> is a place of execution of your application.
   4. Docker Hub -> images public : cloud portals
   5. Client – server
   6. Daemon
4. Commands
   1. docker images
   2. docker ps
   3. docker ps -a
   4. docker pull <imagename>
   5. docker run <imagename>
   6. docker run -name <name to container> <imagename>
   7. docker run -d -it -name <name to container> <imagename>
   8. docker rm <container id or name>
   9. docker rmi <image id or name> partial ids
   10. docker start <container name>
   11. docker restart <containername>
   12. docker stop <containername
   13. docker attach <containername>
   14. docker rename <oldcontainer> <newcontainer>
   15. docker run -v <your foler>:<docker folder>:ro -p <yourport>:<dockerport> <imagename>
   16. docker exec -ti <containername>
   17. docker stats
   18. docker logs
   19. docker inspect
5. Docker Compose :
   1. To contain references of all the services that needs to start , deployed and communicate with each other within the docker engine
   2. docker-compose.yml file
   3. 2 containers running within the docker engine cannot communicate as they run isolation.
   4. The 2 containers can communicate either by using –link, or create network or if you configure in docker-compose, docker creates a network bridge between 2 services
   5. 2 microservices

**Kubernetes:**

1. Container orchestration tool
2. Manage, scale, monitor the containers at runtime across the distributed environment
3. Cluster of containers -> scale across the network
4. Terminologies:
   1. Master node
      1. Api server -> rest endpoints
      2. Controller
      3. Scheduler
      4. ETCD -> key – values
   2. Worker node
      1. Pods -> smallest unit where containers are actually deployed
      2. Kubectl => CLI access to the internal pods
      3. Kube-proxy
5. C ommands
   1. kubectl version
   2. kubectl get pods
   3. kubectl get deployment
   4. kubectl get services
   5. kubectl create deployment <deployment name> --image=<imagename>
   6. kubectl create deployment <deployment name> --yml deploy.yml
   7. kubectl create service <type> <servicename> --port=80
   8. kubectl expose deployment <deployment name> --type = --port= --name =<servicename>
   9. kubectl proxy