Cloud computing

Project milestone: Iaas:

Virtualization and Containerization

Ibrahim Abdo

Feb 3, 2022

100618561

1. **Watch the video on the difference between containerization and virtualization**

* Create a NodeJS application and push it into production.
* Container: first you have a manifest (docker file),
* Image 🡪 container that contains all the runtimes and libraries and binaries needed
* Runs on a very similar setup to the VMS but the difference is that instead of a hypervisor, we have something like a runtime engine (docker-engine)

The containers will be more lightweight where you just have the libraries and the application itself. The applications will take fewer resources.

1. Installing Docker:
2. Watch the video to create containers from images
3. Answer the following questions:
   1. **What is a docker image**: A docker image is what is used to create a running container. It is a template that is used to create and run a container. This can be done by using a Docker file that will include all the instructions on how to build a docker image.
   2. **What is a docker container**: A docker container packages an application and all its dependencies and it allows you to share it easily with the world. Containers are portable, isolated, and lightweight.
   3. **What is a docker registry**: A Docker Registry is a scalable server-side application that lets you store and deliver Docker images.
4. **List the Docker commands used in the video with a brief description for each command and option:**
   1. **Docker version**: outputs version of docker installed on the machine
   2. **These are lines of codes used in the docker file:**
      1. FROM openjdk:11 🡪 This will use the specified version of the OpenJDK image
      2. RUN mkdir /app 🡪 this will make a directory called app
      3. COPY out/production/HelloWorldDocker/ /app 🡪 this will copy the app files from the host machine to the image filesystem
      4. WORKDIR /app 🡪 sets /app as the directory for executing commands
      5. CMD java Main 🡪 run the main class
   3. **docker build -t hello-world:1.0 .** 🡪 docker build to create the image, then -t hello-world:1.0 to specify the image name and tag. The “.” At the end is to specify where the docker file is.
   4. **docker run -d hello-world:1.0** 🡪 This command will create a new container and run the program using the “hello-world:1.0” image created in the previous step. “-d” option is used when you want the program to run in the program without it taking up the console.
   5. **docker ps -a**🡪 This command will output the list of currently running containers on the machine. “-a” option is used to output all containers including running and stopped containers.
   6. **docker images** 🡪 lists all created images
   7. **docker logs fdfb47977ac0** 🡪 displays the out of applications that are running in the background using the -d option with docker run command.
5. **At the end of the video, there are two running containers, what commands can be used to stop and delete those two containers?**
   1. **To stop containers, use** 🡪 “docker container stop $id” $id is the id of containers you would like to stop. If there are multiple you can use docker container stop $(docker container ls -aq), which will list all docker containers id’s and include them in the command
   2. **After stopping the containers, you would like to remove, use the following command** 🡪

docker container rm $(docker container ls -aq)

1. **Prepare a video showing the container(s) created on your machine, displaying their logs, stopping them, and then deleting them. (Note: the JDK version must match that installed in your machine and used to compile the java code. If you have a problem compiled it can download it from the repository from the path: “/v1/out/production/HelloWorldDocker/Main.class” and use OpenJDK:14 in your Dockerfile).**

Video link: <https://www.youtube.com/watch?v=kLpAsQ1YXdo>

1. Watch the following video:

7) answer the following questions:

8) **what’s a multi-container docker application?**

* A multi-container docker application is when your application contains more than one component. If their containers are on the same network, they can talk to each other with networking. This is done using docker-compose. The components are allowed to talk to each other to allow and handle multiple container applications.

9) **How are these containers communicated together?**

* Containers can communicate together using networking and file sharing. These containers can send and receive requests from other applications using networking. This means that applications will create a network connection to a port on another container. Other containers communicate by writing and reading files to each other on the same shared disk.

10) **What command can be used to stop the Docker application and delete its images?**

* Docker rm -f app will delete the old application and delete its images.

11) **List the new docker commands used in the video with a brief description for each command and option.**

* **docker pull mysql** 🡪 this will pull the official docker mysql image for the application to use
* **docker run --name app-db -d -e MYSQL\_ROOT\_PASSWORD=password -e MYSQL\_DATABASE=myDB mysql** 🡪 This command will create a container and run it while passing in variables using the -e command and –name for the application name.
* **docker ps** 🡪 This command is used to view images.
* **Docker logs app-db** 🡪 this is used to show logs from the database app app-db
* **Mvn clean install** 🡪 rebuilds the WAR file and makes sure it’s ready for the image.
* **docker build -t my-web-app:1.0 .** 🡪 creates an image using the docker build command. “-t” option for the image name and 1.0 for the tag.
* **docker run --name app -d -p 8080:8080 my-web-app:1.0** 🡪 This will run the web app on port 8080 with the name app
* **docker network create app-network** 🡪 This will create a network for the docker application
* **docker network connect app-network app-db** 🡪 this will connect the application with the application database
* **docker run --name app -d -p 8080:8080 --network=app-network my-web-app:1.0** 🡪 this will create a container for the app on host 8080 from host machine and application with the –network being the app-network.

**12) Prepare a video showing the created application, run the webapp, stop the application and delete the application containers. (Note: if you have a problem generating the war file, you can download it from the repository from the path: “/v2/target/MyWebApp.war”).**

Video link: <https://www.youtube.com/watch?v=Dwkyv1MhP-A>

**V3:**

**11) Follow the following video to deploy dockers containers (valid until the shell session is expired) on GCP or by using Kubernetes (until you change it)**

**12) Prepare a video showing how the container is deployed using Docker and Kubernetes in GCP.**

Video link: <https://www.youtube.com/watch?v=Q3_6Q0AguR4>

13) **List all used GCP shell commands and their description in your report.**

* gcloud config set project youtube-demo-255723 🡪 initialize project name
* docker run -d -p 8080:80 nginx:latest 🡪 runs the container on port 8080 using the latest image from nginx. “-d” is to run the container in the background
* docker ps 🡪 to view all running containers
* docker commit 58a79a1bce2a cad/web:version1 🡪 commits the image
* docker cp index.html 40b22ebd3ebe:/usr/share/nginx/html/ 🡪 this will use docker to copy the index.html file to the appropriate file location
* docker tag cad/web:version1 us.gcr.io/youtube-demo-255723/cad-site:verison1 🡪 docker tag will
* docker push us.gcr.io/youtube-demo-255723/cad-site:version1 🡪 pushes the docker

**Kubernetes:**

* gcloud container clusters create gk-cluster --num-nodes-1 🡪 creates the gk cluster container
* gcloud container clusters get-credentials gk-cluster --zone us-central1-c 🡪 authorizes user to use the application
* kubectl create deployment web-server --image=us.gcr.io/youtube-demo-340213/ca

d-site:version1 🡪 deploys the web-server application with the image created in the previous steps

* kubectl get pods
* kubectl get service web-server

16) Answer the following questions:

17) **What is Kubernetes’ pod, service, node, and deployment?**

a**. Kubernetes pod:** A pod is a small deployable object in Kubernetes. It is an instance of a running process in the cluster. These pods hold multiple containers which are managed as a single entity and use shared resources

b. **service**: A Kubernetes service is used to expose an app that is deployed on different pods by using a single endpoint.

c. **node**: nodes can be either virtual or physical machines. Nodes are worker machines that are managed by the control plane.

d. **deployment**: Deployment is when you tell Kubernetes how you want pods to be created or modified.

18) **What’s meant by replicas?**

* Replicas mean that pods will be running more than one copy of a container, this way it will manage to keep the application up and running even if one container fails since it already has an exact replica.

19) **What are the types of Kubernetes’ services? what is the purpose of each?**

* ClusterIP: can expose services that is only accessible from the cluster
* NodePort: uses a static port on each node’s IP to expose a service
* LoadBalancer: uses the cloud provider’s load balancer to expose the service
* ExternalName: returns a value for the CNAMe record when a service is being mapped to a predefined externalName field.