Ireni Ruthirakuhan (100657302) - Lab 1

Difference between containerization and virtualization (VMs)

- Containers (OS virtualizations)
    - Has a Runtime engine instead of a Hypervisor
    - Enables software to run predictability and good when moving from one server environment to another
    - Provides a way to run isolated systems on a single server/host OS
    - Containers - sit on top of Host OS
        - Each container shares the host OS kernel/binaries/libraries
    - Containers represent packages of SW
        - The SW would contain the necessary components in order to run in any environment
        - Containers virtualize an OS and run anywhere
    - In the case that container processes are not using the shared memory, another container can use it if needed within the hardware
    - Are portable and scalable
- VM
    - Virtual machines run SW on top of the physical servers
        - This emulates the particular hardware system
    - Has a hypervisor
        - A SW/firmware/HW that creates and runs VMs

Docker Image

- A file that is used to execute code within the docker container
- Its functionality can be seen as a set of instructions that are utilized
    - Helps build/construct the docker container

Docker Container

- An executable package of SW that incorporates all necessary components to successfully run the application
    - This includes: code, system tools, libs, runtime
- The container has the ability to allow developers to package up an application
    - Package it with all necessary elements/components

Docker Registry

- Represents the storage and content delivery system within docker
- Represents the system for versioning, storing, and the distribution of docker images

Dockerfile Commands

- FROM
    - Initializes new build stage
    - Prepares the base image for the instructions
    - Has the ability to create multiple images/use one build stage as a dependency for another

- RUN
    - Execute commands on a new layer
        - This layer is on top of the current image
        - Commits results
- COPY
    - Has the ability to copy new file(s) from one directory and then add/include those files into a filesystem
        - The path set depicts where the filesystem of the container resides
- WORKDIR
    - Used to define the working directory of the docker container at any given time
- CMD
    - Main purpose is to provide defaults for an executing container
    - Can only be one command

Docker terminal commands

- Docker ps - process status, can see list of containers
- Docker stop - stops running containers
- Docker images - shows all images that match the argument
- Docker build -t hello-world:1.0 - builds docker container with name and tag
- Docker run hello-world:1.0 - runs docker container

    Pull and open jdk image
    Copy my application files into the image
    And run the container when the application starts

- Docker rm container_id
    - deletes container
    - Docker rm -f → removes all containers
- Docker stop container_id
    - Stops running the container

Video Link 1
https://drive.google.com/file/d/1Z1AO8fhjhKL-URceabJEHkHJUOdHRO5J/view?usp=sharing

Multi-container in a docker application

- Allows multiple containers to run at the same time on separate host ports since ports can be already allocated to a previous running container
- Minimizes setup to run on machines
- Runs multiple instances of the container's ports (simultaneously)
    - Need to make the others available outside of docker → change host port value to bind to

Containers communicated together via bridge networks

- Docker network create app-network

        - App-network = network name

Video Link 2
https://drive.google.com/file/d/1a3gkX4CCLEXGm5dHopIwP1agbAy59JAQ/view?usp=sharing

New commands

- Docker pull mysql
    - Downloads mysql docker image/ local repository onto the registry
    - Creates mysql container
    - Pulls latest version
    - Can create the container from it
- Docker run --name app-db -d -e MYSQL_ROOT_PASSWORD=password -e MYSQL_DATABASE=myDB mysql
    - Creating a container with the name app-db
    - -d is in detached mode so the terminal isn't taken over by database logs
    - -e environment variable is used to set up db name and password
        - Outputs container id
    - mysql = image that was recently pulled
- Docker logs app-db/container_id
    - Shows the log of the running container
    - Can see if the server started successfully
- Docker build -t my-web-app:1.0 .
    - Container name is my-web-app with a 1.0 tag
    - . is for the current directory
    - This command builds an image
- Docker run --name app -d -p 8080:8080 my-web-app:1.0
    - Run application's container
    - Need to specify that there is a process in the application container that listens to a specific port at runtime → need to expose the port
    - Need to make port available outside of docker. This is done by telling docker the port and bind it to a port on the host machine
    - -p means to publish
    - Binds host machine port to containers port
        - Tell docker to make port 8080 available outside of docker and bind that port to the host machine port
- Getting rid of a container
    - Docker rm -f container_id/ app name
        - Stops the container and removes it
        - Docker stop container_id also works
- Docker network create app-network
    - Create new network
- Docker network ls
    - Lists all active networks
    - Has 3 default networks it creates host, none, bridge

        - Host removes the network isolation between the container and the host machine
        - None network enables all networking
        - Bridge network where the containers are attached to by default
- Connect both containers to the network
    - Docker network connect app-network app-db
    - Docker run --name app -d -p 8080:8080 --network=app-network my-web-app:1.0
        - Can also call network right away instead of calling it separately
- Docker-compose up -d
    - Brings the application up starting the 2 containers
    - Automatically creates a bridge network for the application services and attaches the containers to it

Video Link 3
https://drive.google.com/file/d/1_XM2eZW8XDGmNnHJ1DFZmj0Nymn-gUrp/view?usp=sharing

GCP shell commands

- Setting the config project
    - gcloud config set project project_name
- Can run docker commands
    - Docker run -d -p 8080:80 nginx:latest
        - -p exposes the port
        - Nginx = image name
        - -d runs container in background
    - Docker ps -a
        - Shows previous running containers
    - Docker ps
        - Shows currently running containers
    - Docker cp index.html container_id:/usr/share/nginx/html/
    - Docker commit container_id cad/web:version1
        - cad/web:version1 name of image you are going to commit
        - Committed latest changes of the container into a local repository
    - Before you push images to cloud registry you need to tag it
        - Docker tag cad/web:version1 us.gcr.io/projectname/cad-site:version1
            - Us.gcr.io = host name
            - Cad-site:version1 = repository name
        - Docker push us.gcr.io/projectname/cad-site:version1
            - Pushed to gcr
- Deploy in gke >> can be deployed from UI/command line
    - Also need to set zone
        - Gcloud config set compute/zone us-central1-a
- Create the cluster
    - Gcloud container clusters create gk-cluster --num-nodes=1
        - Container = service
        - Gk-clusters = cluster name

        - Creating 1 node
- Deploy container
    - First get credentials
        - Gcloud container clusters get-credentials gk-cluster
        - Configures kubectl to use the cluster created
    - Deploying app to cluster
        - Kubectl create deployment web-server --image=us.gcr.io/projectname/cad-site:version1
        - Web-server = name of app
- Expose to the internet
    - Kubectl expose deployment web-server --type LoadBalancer --port 80 --target-port 8080
        - web-server = deployment name
        - Port initializes the public port 80 to the internet, the target port routes the traffic to port 8080 of the app
        - LoadBalancer type creates a Compute engineer load balancer
- See status of pods
    - Kubectl get pods
- Kubectl get service web-server
    - cp external ip then paste in browser

Video Link 4
https://drive.google.com/file/d/1RuxKmjnXfNccnfvbHnPdYEFIxEOf8w2K/view?usp=sharing

What is kubernetes pod

- Small and most deployable objects that has shared storage and network resources
- Are designed to manage/support multiple containers
- Run on nodes

Kubernetes service

- A deployed group of pods within a cluster
- Used to connect the pods to the service name and IP address
- Provide discovery and routing between pods

Kubernetes node

- Runs services necessary for containers that make up the cluster’s workload
- Has a kublet
    - A process that communicates between the control plane and the node
- Manages pods and the containers on the machine
- Has a container run time
    - Pulls container image from registry, unpacks container and runs the application

Kubernetes deployment

- Used to tell kubernetes how to modify/create instances of pods that hold the containerized application
- Scales the replica pods, enable rollout of new code/rollback to earlier deployment version
    - Provides updates for pods and replica sets
    - Can replace a failed pod/bypass down nodes
        - Replaces pods to make sure that the applications continue to work as expected
    - Ensures that they are running, as expected, across all nodes within the cluster
- Deployments are used to create new replicas/remove existing deployments
    - It has the ability to adopt their resources with new deployments

What replicas mean

- ReplicaSet within kubernetes is a controller that ensures that there is a specific number of pods running
- regional clusters/replicas are more suitable for high availability since they have multiple control planes across multiple zones in a region
- Changes takes longer to propagate

Types of kubernetes services and their purpose

- ClusterIP
    - A (default) service only used within the cluster
    - Internal clients are able to send requests to a stable IP
        - This lasts for the life of the service
- NodePort
    - A service that has a static port on each of the node's IP
    - Where clients send requests to IP of node on 1/+ nodePort values
- LoadBalancer
    - A service that uses the cloud provider's load balancer
    - Clients send requests to IP address of network load balancer
- ExternalName
    - A service that is directed to an external named field by a returning value