What are docker image, container, and registry?
- A docker image is an executable package which has the required software needed to run an app (libraries, code, runtimes, etc). The docker image can be considered the 'template'
- A docker container is the instantiated version of the image. It is basically the image, or template, in use.
- A docker registry is content delivery system to distribute named docker images. Users can push/pull docker images to the registry similar to github.

List the commands used in the video and give a brief description
1. "docker build -t imagename:tag" this command was used to create a docker image and assign it a tag.
2. "docker images" this command lets the user see the existing docker images.
3. "docker ps" this command shows the user the currently running docker containers.
4. "docker run imagename:tag" runs the docker image specified as a container.
5. "docker logs containerID" shows the current logs of the docker container. Useful if the container is being run in the background.
6. "docker run -d imagename:tag" runs the docker container in the background, so as to not flood the terminal with the docker logs.

At the end of the video there are two running containers, what command can be used to stop and delete them?
The commands "docker stop containerID" can be used to stop the container, and the command "docker rm -f containerID" can be used to delete the container.

What's a multi-container Docker application?
An application which uses more than one docker container. These are useful when a user wants to scale apis/databases differently than the app. If all versions of the app can use one database, you only need to replicate the database into one container. Thus a multi-container docker app is one where multiple containers can interact to function.

How these containers are communicated together?
These containers communicate through exposing their ports for communication. For example, the app container may expose its "port 8080" or the database may expose a port, and the containers can then send information through these ports.

What command can be used to stop the Docker application and delete its images?
To stop a docker application, and delete its images, use the commands:
docker rm -f containerID //this command will stop the running container and remove it.
docker rmi -f imageName //this command will forcefully remove and delete the image

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

docker rm -f app //This command will kill the currently running docker container (named app, in this case) forcefully and remove it
docker run --name Name -d -p 8080:8080 imageName:version //This command runs the docker image and version mentioned, in the background (due to the -d tag) and gives it the name "Name". Also, it exposes the hosts port 8080 to the containers port 8080. I.e, hostPort:ContainerPort.
docker network create app-network //This will create a network with the name "app-network". This will act as a bridge for the different containers.
docker network connect app-network app-db //This command will connect the app-db container to the app-network network
docker-compose up -d //This command runs the docker compose file and runs the containers in the background.

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

docker run -p 8080:80 nginx:latest //runs the docker image for the latest nginx, binding host port 8080 to 80
docker cp index.html [container-id]:/usr/share/nginx/html/ //Copies over the index.html into the container
docker commit [container-id] cad/web:version1 //commits the updated container as a new image
docker tag cad/web:version1 us.gcr.io/project-id/cad-site:version1 //tags the new image
docker push us.gcr.io/project-id/cad-site:version1 //pushes the image onto the container registry at us.gcr.io

gcloud config set project project-id //sets project ID so you are in the correct project (with priveleges, etc)
gcloud config set compute/zone us-central1-a //sets the zone the clusters are going to be running in

gcloud container clusters create gk-cluster --num-nodes=1 //creates the gk cluster with 1 node
gcloud container clusters get-credentials gk-cluster //gets the credentials for the cluster

kubectl create deployment web-server --image=us.gcr.io/project-id/cad-site:version1 //creates the deployment web-server based on the image in the registry

kubectl expose deployment web-server --type LoadBalancer --port 80 --target-port 80 //exposes that web servers ports using type LoadBalancer, this balances the ports loads
kubectl get pods //This shows the running pods

kubectl get service //This gets the services that are running so you can see the external IP and other information.

What is Kubernetes' pod, service, node, and deployment?
Pod - hosts the application instance. Its a kubernetes abstraction which is used to represent multiple containers and some of their shared resources such as networking, volumes, storage, image version, etc.

Nodes - This is where a pod runs. A node is just a machine (virutal or physical depending on the cluster). A node can have multiple pods, and handles the scheduling and running of these pods.

Service - A way to expose the application on a pod as a network service. As pods can have their own IP address, they can be used as web services.

Deployment - This is used to instruct kubernetes on how to create/modify pods. Deployments are used to scale the number of pods/rollout updates/generally control how the pods are being deployed/their version, etc.

What's meant by replicas?
These are replicas meant to serve as failsafes in the case that a pod may fail/crash or become inaccessible. They are basically instances of particular pod. A replica set is multiple replicas of the same pod to ensure greater availability through backup pods.

What are the types of Kubernetes' services? what is the purpose of each?
ClusterIP - This will expose a service to be accessed from within the cluster only (the IP is not for public use)
NodePort - This exposes a service pased on the nodes port, which means it can be accessed from outside the cluster.
LoadBalancer - This exposes the service via a load balancer which is basically the cloud service providers own "clusterIP". This will give the service an external IP address which is given to it by the cloud provider.
ExternalName - This will map a service to an externalName field. It returns a CNAME record which can be used to access the service.
Headless Service - This is when none is specified as the clusterIP. This is used when load balancing is not needed and only a single IP needs to be exposed.