Kubernetes with Containers and DevOps Workshop

Hands-on lab step-by-step

Aralık 2018

1. Create Docker images for apps and push to Azure Container Registry

**Build Container Images**

For the containers, the Dockerfiles are provided.

**Web Container**

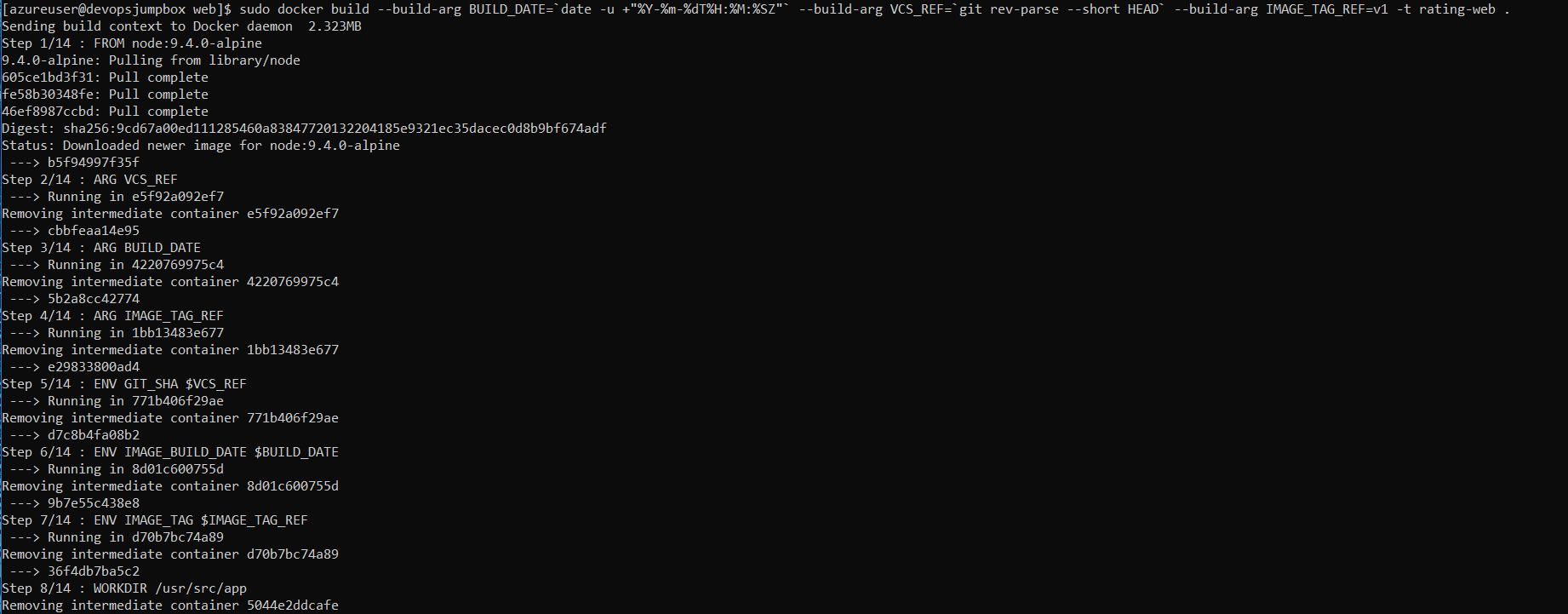
1. Change Directory to web app directory.
   * In the cd ~/cpx-oss-workshop/app/web/ directory,

cd ~/cpx-oss-workshop/app/web

1. Create a container image for the node.js Web app. Ignore npm warnings.

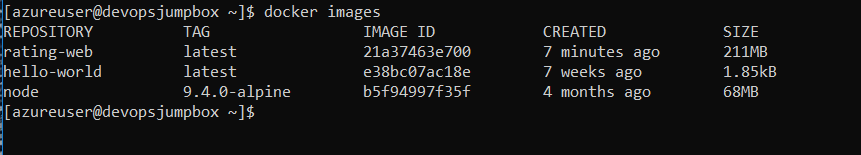
From bash shell:

sudo docker build --build-arg BUILD\_DATE=`date -u +"%Y-%m-%dT%H:%M:%SZ"` --build-arg VCS\_REF=`git rev-parse --short HEAD` --build-arg IMAGE\_TAG\_REF=v1 -t rating-web .



Validate image was created with

docker images

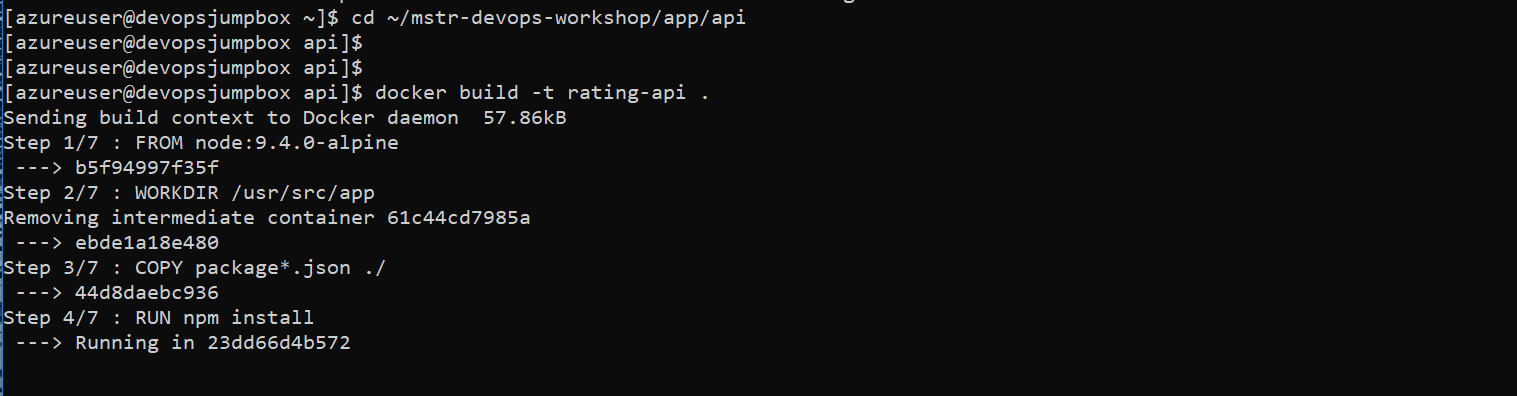


**API Container**

1. Create a container image for the node.js API app. And ignore npm warnings.

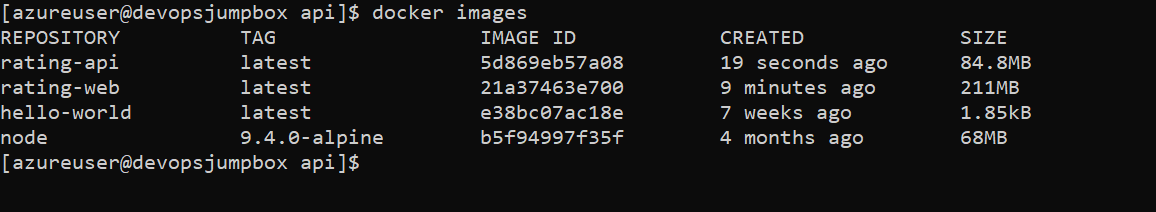
cd ~/cpx-oss-workshop/app/api

docker build -t rating-api .



Validate image was created with

docker images

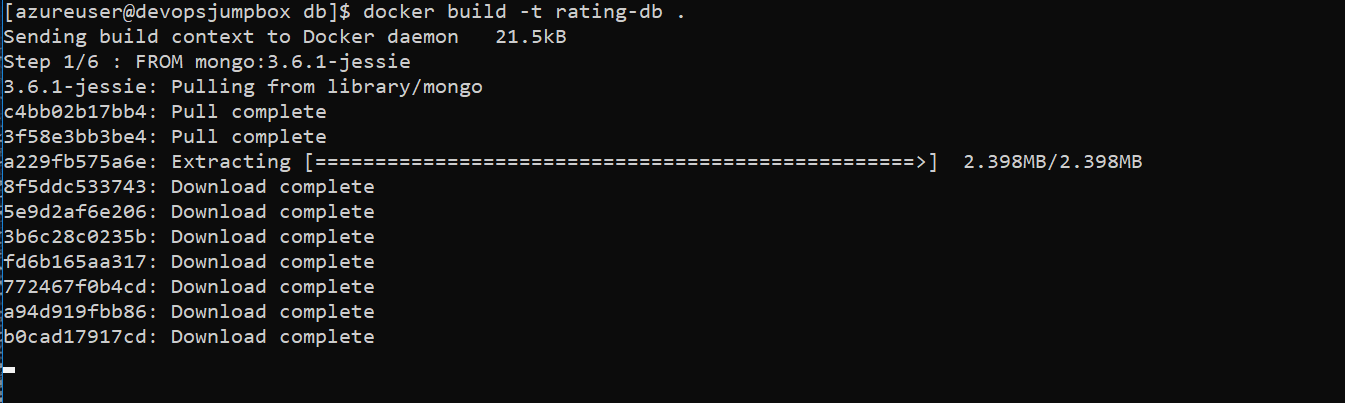


**MongoDB Container**

1. Create a MongoDB image with data files

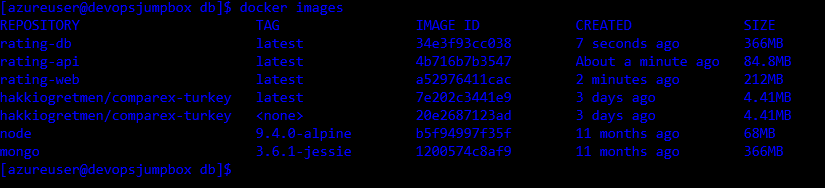
cd ~/cpx-oss-workshop/app/db

docker build -t rating-db .

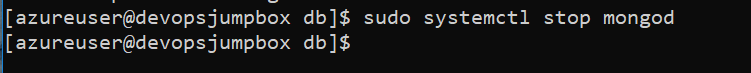


1. Validate image was created with

docker image ls



sudo systemctl stop mongod

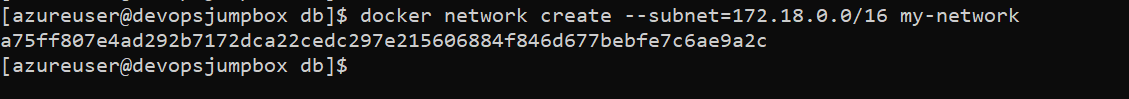


**Run Containers**

**Setup Docker Network**

Create a docker bridge network to allow the containers to communicate internally.

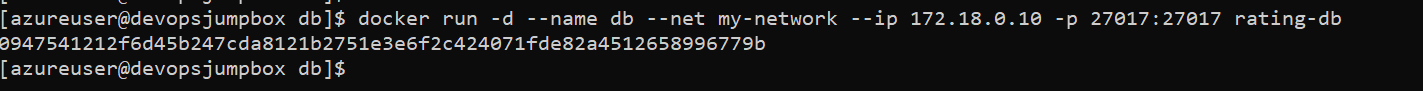
docker network create --subnet=172.18.0.0/16 my-network



**MongoDB Container**

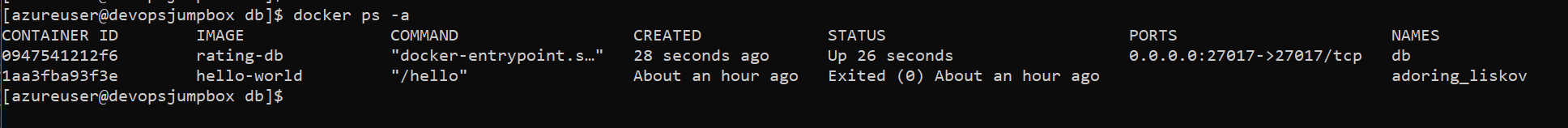
1. Run mongo container

docker run -d --name db --net my-network --ip 172.18.0.10 -p 27017:27017 rating-db



Validate by running

docker ps -a



1. Import data into database. Attach to the container by running bash process on it.

docker exec -it db bash

You will have a prompt inside the mongo container. From that prompt, run the import script which will import the documents to collection

./import.sh

root@61f9894538d0:/# **./import.sh**

2018-01-10T19:26:07.746+0000 connected to: localhost

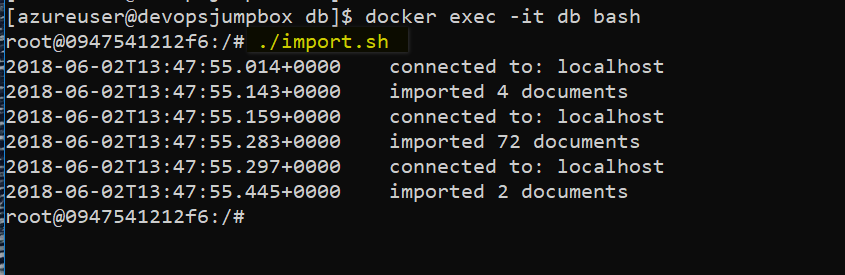
2018-01-10T19:26:07.761+0000 imported 4 documents

2018-01-10T19:26:07.776+0000 connected to: localhost

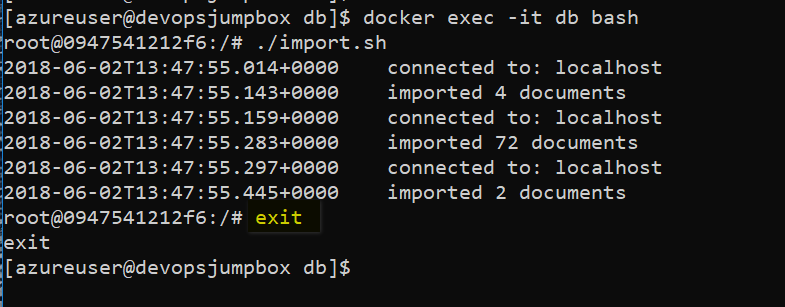
2018-01-10T19:26:07.787+0000 imported 72 documents

2018-01-10T19:26:07.746+0000 connected to: localhost

2018-01-10T19:26:07.761+0000 imported 2 documents



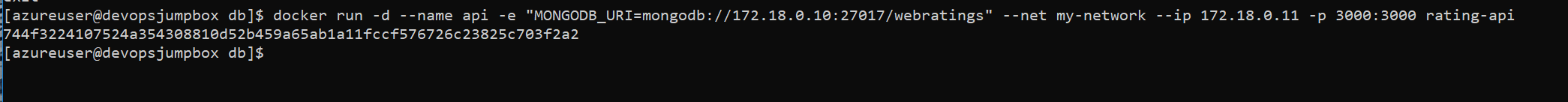
1. Type exit to exit out of container. Typing exit will kill the bash process we have created. Container will be still alive after exiting.



**API Container**

1. Run api app container

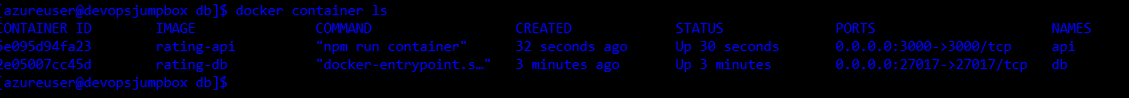
docker run -d --name api -e "MONGODB\_URI=mongodb://172.18.0.10:27017/webratings" --net my-network --ip 172.18.0.11 -p 3000:3000 rating-api



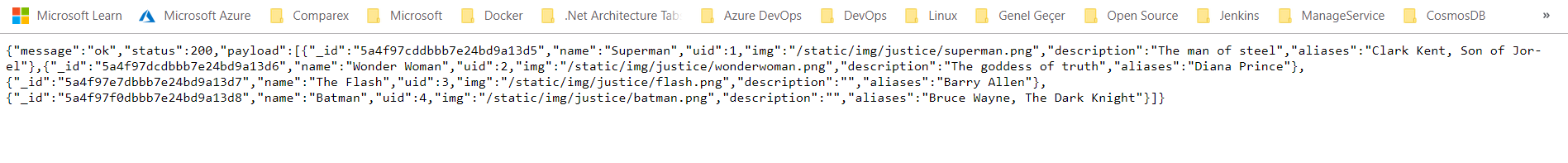
Note that environment variables are used here to direct the api app to mongo.

Validate by running (same command with docker ps)

docker container ls



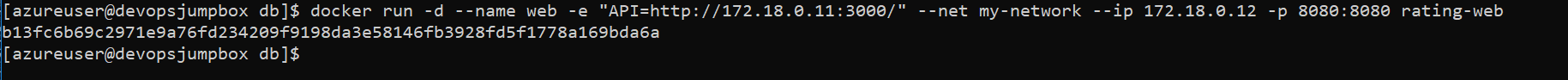
1. Test api app by browsing to http://{PUBLIC\_IPADDRESS\_VM}:3000/api/heroes



**Web Container**

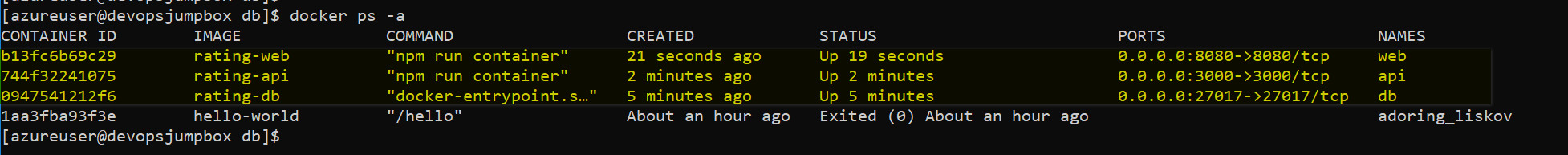
Run web app container

docker run -d --name web -e "API=http://172.18.0.11:3000/" --net my-network --ip 172.18.0.12 -p 8080:8080 rating-web

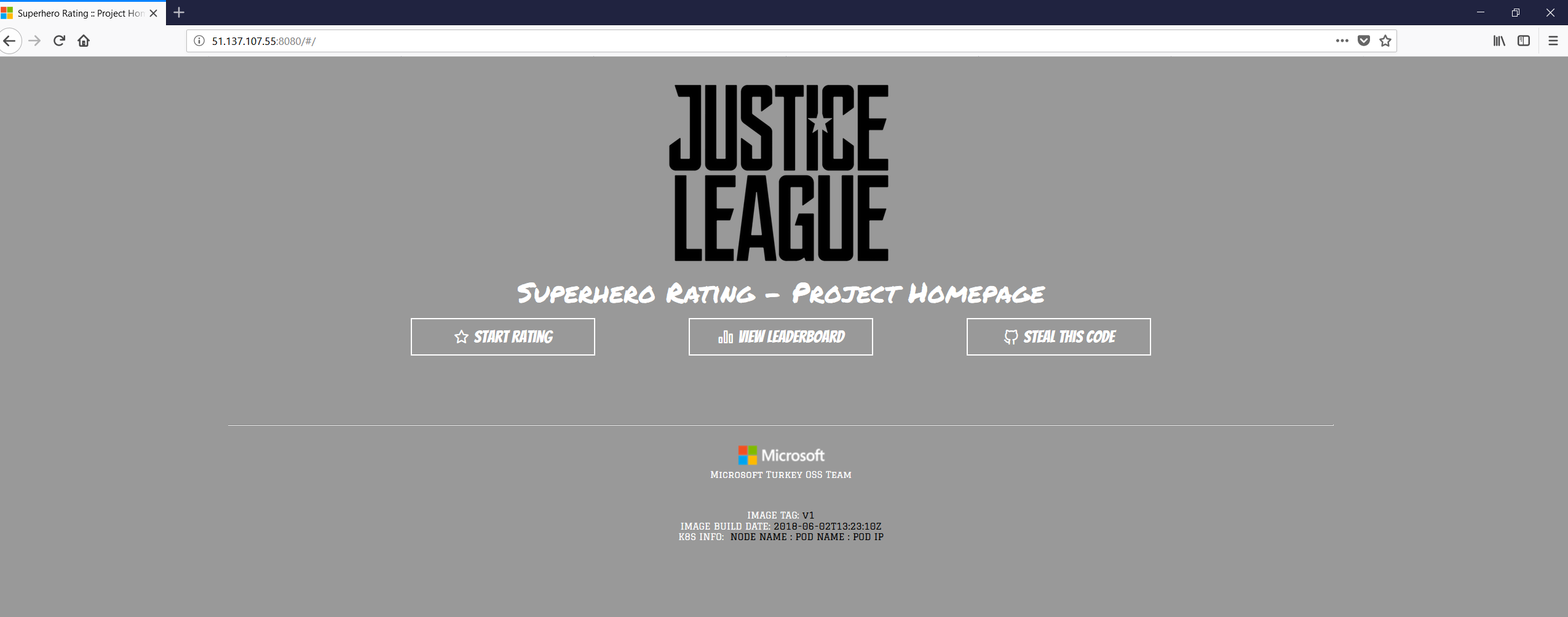


Validate by running

docker ps -a



Test web app by browsing to http://{PUBLIC\_IPADDRESS\_VM}:8080

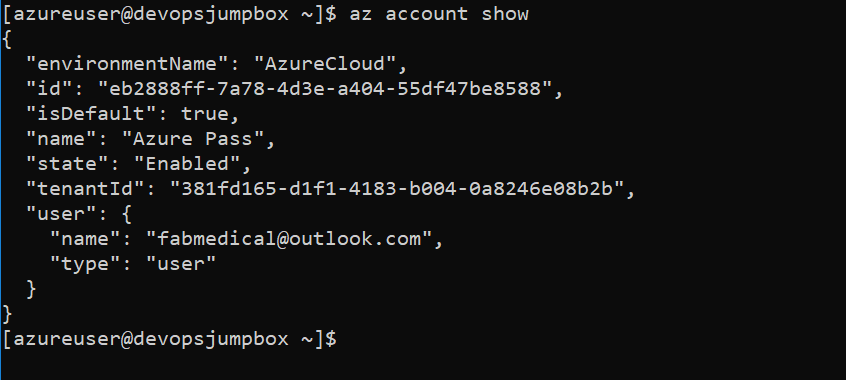


**Azure Container Registry (ACR)**

Now that we have container images for our application components, we need to store them in a secure, central location. In this lab we will use Azure Container Registry for this.

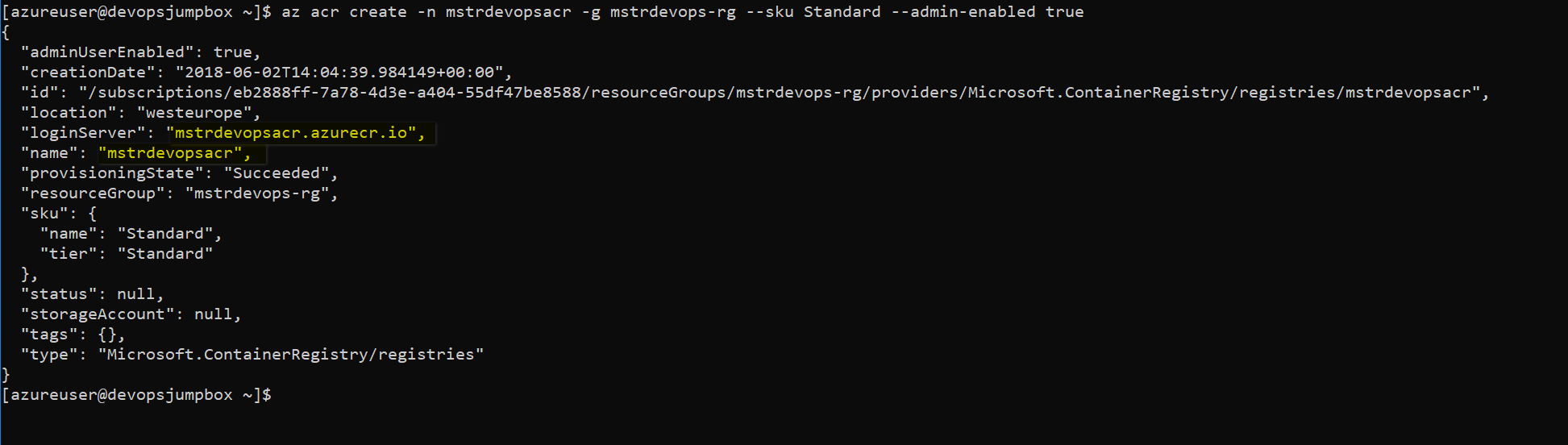
**Create Azure Container Registry instance**

az account show



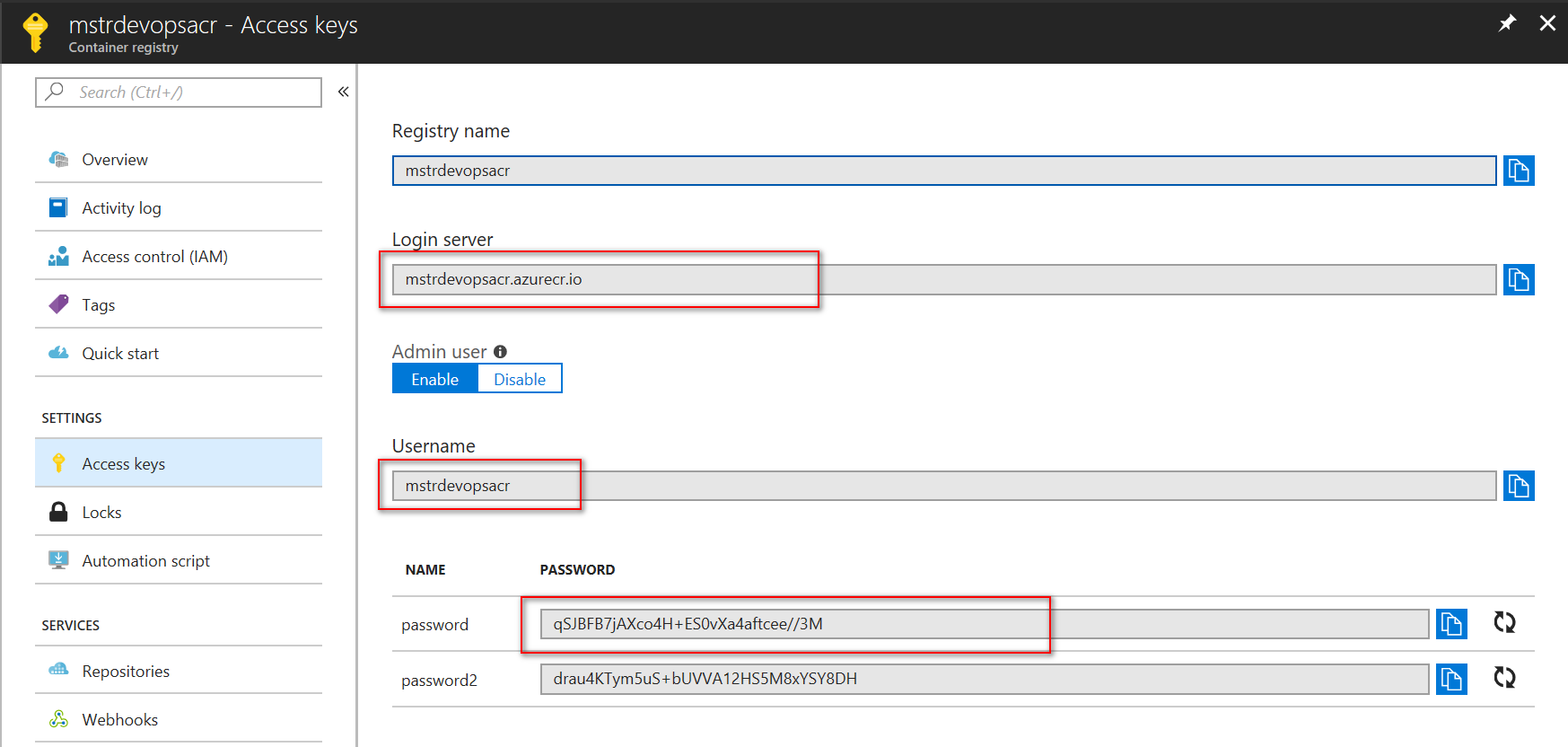
To create Azure Container Registry run the command. Do not forget to change acr name (You may use mstrdevopsacr{yourname} naming format).

az acr create -n mstrdevopsacr001 -g mstrdevops-rg --sku Standard --admin-enabled true

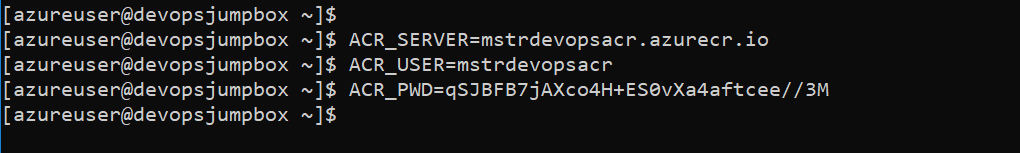


**Login to your ACR with Docker**

1. Browse to your Container Registry in the Azure Portal
2. Click on "Access keys"

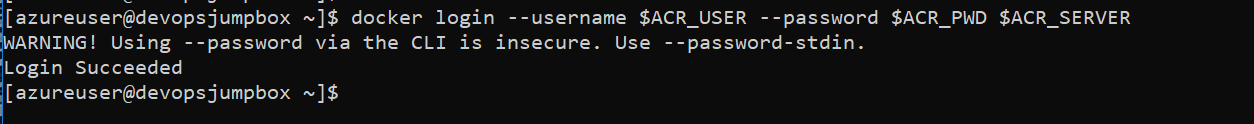


1. Make note of the "Login server", "Username" and "password"
2. Set each value to a variable as shown below.
3. Login in your Bash shell:
   * 1. # set these values to yours
     2. ACR\_SERVER=mstrdevopsacr001.azurecr.io
     3. ACR\_USER=mstrdevopsacr001
     4. ACR\_PWD=VBi8PrMLXE2Yr6psMfbZ1=806qsWiEJ4



docker login --username $ACR\_USER --password $ACR\_PWD $ACR\_SERVER

\*\* Ignore “using --password via the CLI is insecure” warning.



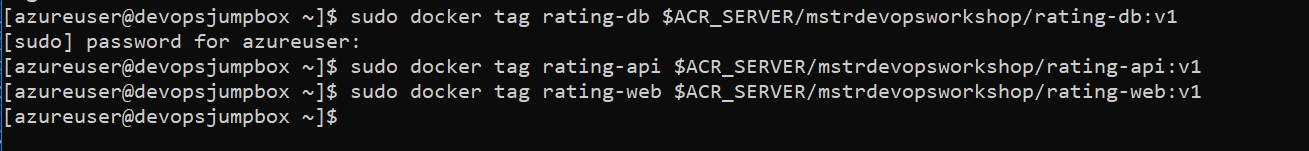
**Tag images with ACR server and repository**

# Be sure to replace the login server value

docker tag rating-db $ACR\_SERVER/mstrdevopsworkshop/rating-db:v1

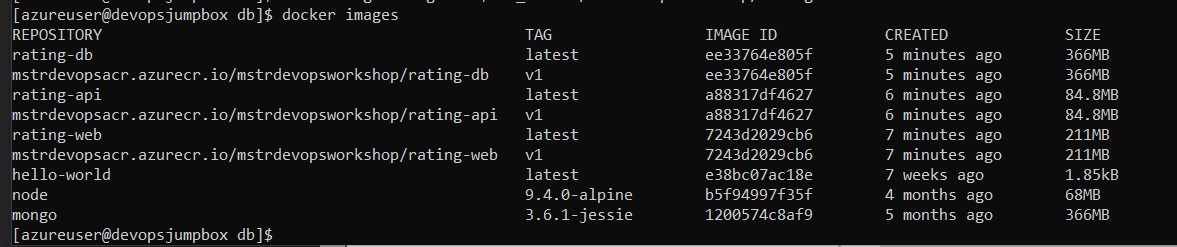
docker tag rating-api $ACR\_SERVER/mstrdevopsworkshop/rating-api:v1

docker tag rating-web $ACR\_SERVER/mstrdevopsworkshop/rating-web:v1



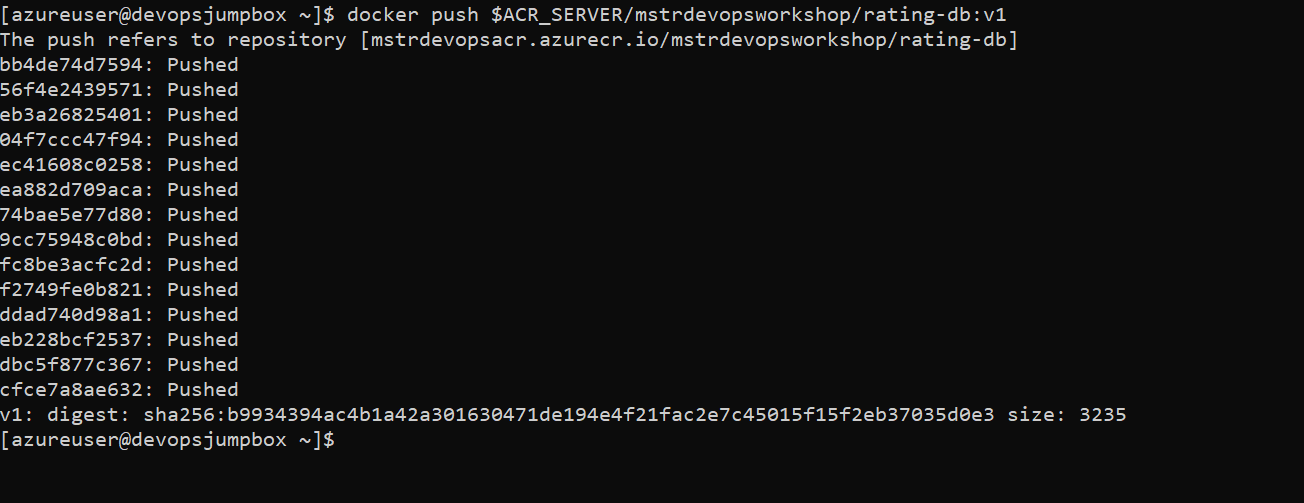
Validate images

docker images



**Push images to registry**

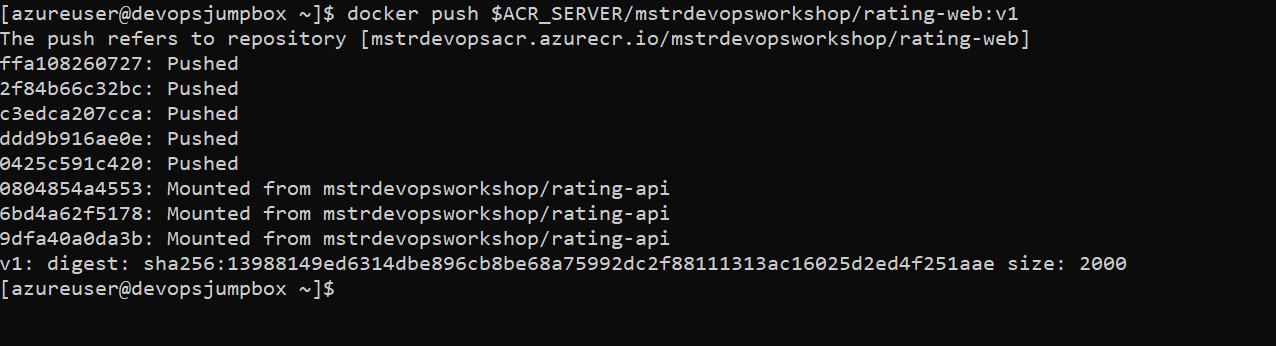
docker push $ACR\_SERVER/mstrdevopsworkshop/rating-db:v1



docker push $ACR\_SERVER/mstrdevopsworkshop/rating-api:v1

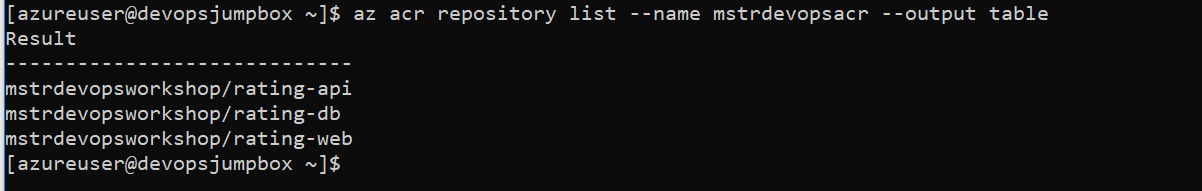


docker push $ACR\_SERVER/mstrdevopsworkshop/rating-web:v1



**Validate images in Azure**

az acr repository list --name mstrdevopsacr001 --output table



**You can see container images in Container Repositories**

