# Some basic Docker Commands:

sudo apt update

sudo apt install docker.io

sudo service docker status

docker --version

docker info

sudo docker run --name 1stcont -it ubuntu /bin/bash

sudo docker diff 1stcont

sudo docker commit 1stcont 1stcontimg (to create an image from an existing container)

sudo docker images

sudo docker run -it --name 2ndcont 1stcontimg

sudo docker volume ls (to get the **list** of available **volumes**)

sudo docker volume create ‘volumename’ (to **create** a **volume**)

sudo docker volume remove ‘volumename’ (to **remove** a **volume**)

sudo docker volume prune (to **remove unused volumes**)

sudo docker volume inspect ‘volumename’ (to get the **details** about **volume**)

sudo docker container inspect ‘containername’ (to **remove** a **volume**)

sudo docker stop $(docker ps -a -q) - to **stop** all the **containers**

sudo docker rm $(docker ps -a -q) - to **delete** all the **containers**

sudo docker rmi -f $(docker images -q) - to **delete** all the **images**

apt list – To get a list of installed packages

apt list | grep docker

Reach out to below link to learn about docker compose.

<https://docs.docker.com/> - link for docker documentation

<https://www.freecodecamp.org/news/what-is-docker-compose-how-to-use-it/>

# Steps to create an image from a Dockerfile: #6 (T G)

Create a file with filename Dockerfile

Add instructions in Dockerfile

Build Dockerfile to create image

Run image to create container

# Docker Volume:

1. Volume is simply a directory inside our container.
2. Firstly, we declare this directory as a volume and then share it.
3. Even if we stop/delete container, we still can access volume.
4. Volume will be created in one container.
5. A directory can be declared as volume only while creating container and can’t be created in existing container.
6. A volume can be shared across multiple containers.
7. If we create an image from an existing image having volume, then this volume will not be shared with the newly created image.
8. Volume can be shared between container to container & container to host.

## Uses of Docker Volume:

1. Decoupling container from storage.
2. Single volume can be shared among multiple containers.

## Creating Volume from Dockerfile:

FROM ubuntu

VOLUME [“/myvolume”]

Docker build -t ‘imagename’ (.) – command to create an image from docker file

Now we will **create a container** using the command below from the image we have created using above dockerfile/ commands.

Docker run -it –name ‘containername i.e cont1’ ‘imagename’

You can now ls to check your container.

## Sharing volume between containers:

Use below command to create another container to share volume with it.

docker run -it --name ‘containername i.e cont2’ --privileged=true(this is for giving full access to the container with which we are sharing the volume) --volumes-from ‘containername i.e cont 1’

# Lab:

|  |
| --- |
| sudo apt update  sudo apt install docker.io -y  sudo docker info  service docker status  touch file1 file2  vi Dockerfile  sudo docker build -t img1 (.)  sudo docker images  sudo docker run -it --name cont1 img1  ls  sudo docker run -it --name cont2 --privileged=true --volumes-from cont1 ubuntu  sudo docker ps  sudo docker ps -a  sudo docker start cont1 |

Now creating **volume** using **docker command** instead of dockerfile.

docker run -it --name cont3 **-v** ‘/volumename i.e /vol2’ ubuntu

creating another container to share volume with (volume we created just above this line using command)

docker run -it --name cont4 --privileged=true --volumes-from cont3 ubuntu

# Lab:

|  |
| --- |
| sudo docker run -it --name cont3 -v /vol2 ubuntu  sudo docker run -it --name cont4 --privileged=true --volumes-from cont3 ubuntu  sudo docker start cont3  sudo docker attach cont3 |

## Sharing (aka mapping) volume from host to container:

sudo docker run -it --name cont5 **-v /home/SyedZaka:/volume** --privileged=true ubuntu

Host path Volume name

# Docker Port Expose: # 7(29)

A container doesn’t have its own Ip. Logical ports 0-65535. vs expose. Expose only used for communication between container & container(inter container communication) while -p is used to communicate over the internet.

docker run -t**d** –name ‘containername i.e cont6’ **-p 80:80** ubuntu

Daemon port/ publish Host Port container port

docker port ‘containername’ – to get the **port details** for a container

**exec** and **attach** both commands are used to get into a container but by using **attach** we can go to a process(main) running at that time but **exec** will create a new process with a new process id(pid) and take us to that created task.

docker **exec** -it ‘containername’

# Lab:

|  |
| --- |
| sudo docker run -td --name cont6 -p 80:80 ubuntu – creating a container with name cont 6  sudo docker port cont6 – checking port status for cont6  sudo docker exec -it cont6 /bin/bash – to get into the cont6  sudo docker run -td --name myjenkins -p 8080:8080 jenkins/Jenkins – to create a container with name myjenkins, installing jenins in it and exposing its 8080 port |

# Dockerhub: # 8(30)

Pushing image to dockerhub

## Lab:

|  |
| --- |
| docker login  sudo chmod 666 /var/run/docker.sock – Need to run this command because of permission denial issue while trying to login to docker  docker tag image syedzakaurrahman/newproject – tagging images  docker push syedzakaurrahman/newproject – pushing images to dockerhub |

|  |  |  |
| --- | --- | --- |
| **Dockerfile and docker-compose comparison chart** | | |
| Comparison | Dockerfile | docker-compose |
| Purpose | Image creation | Run a container |
| Type | Extensionless file | Command and a YAML file |
| Associated command | docker build | docker-compose up |
| Docker SDK | Yes | Yes |
| Reciprocity | Can’t call docker-compose | Can invoke the Dockerfile |
| Kubernetes SDK | No | No |
| Deprecated | No | No |

<https://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/Dockerfile-vs-docker-compose-Whats-the-difference>

BDT

Docker - Understanding Virtual Machines and Containers: #1

D:\4.5\OneDrive - Visionet Systems Inc\Zaka\Learnings\DevOps\Docker Administration\

https\_\_\_bestdotnettraining.azureedge.net\_documents\_Azure\_Devops\_2021\_21\_Working\_with\_Containerization\_using\_Docker\_21\_1\_Introduction\_to\_Docker.pdf

Virtualization tools (VMWare and VirtualBox are famous one)

mkdir DockerDemos

cd DockerDemos

dir

doskey /history A container = A VM – OS

A container is a running instance of an image.

What is Docker, its Benefits, Docker Architecture and its Taxonomy: #2

Visit the link above for the document.

Creating a Docker Image: #3

D:\4.5\OneDrive - Visionet Systems Inc\Zaka\Learnings\DevOps\Docker Administration\

https\_\_\_bestdotnettraining.azureedge.net\_documents\_Azure\_Devops\_2021\_21\_Working\_with\_Containerization\_using\_Docker\_21\_Working\_with\_Docker\_Images.pdf

Visit the links above for the document.

Lab: using windows cmd

|  |
| --- |
| Simple way to create an image from Dockerfile using below code by Rashid  FROM mcr.microsoft.com/dotnet/nightly/sdk  WORKDIR C:\app  COPY . .  ENTRYPOINT ["YourApplication.exe"] |

Requirements:

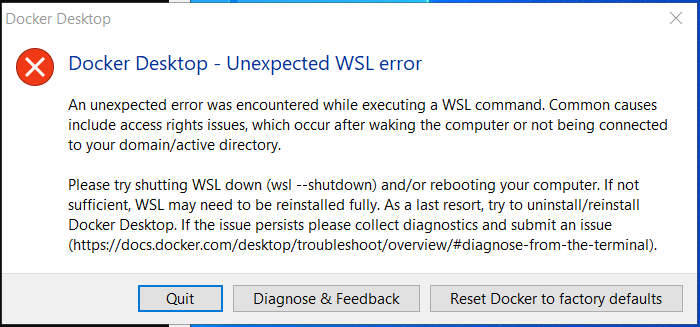
* .Net SDK
* VS Code
* Docker Desktop for Windows

|  |
| --- |
| C:\Users\Windows>d:  docker version  dotnet version  mkdir DockerDemos  cd DockerDemos  dotnet new mvc -n HelloWorldApp  cd HelloWorldApp  cls  dotnet run  code .  docker images  docker build -t dckrhb/HelloWorldApp:V1 .  dir  doskey /history |

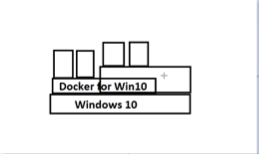
We need to copy the code from the pdf file to create a dockerfile using the above links and need to place that dockerfile in the same file where we have .csproj file.

We will receive below error if we don’t choose a machine with security type standard and select with trusted ……. Security type and want to run docker desktop on it, so the simple solution for this is to select a machine with standard security type.





When we make use of docker build command and using tags than make sure that repository name must be lowercase. 

 23:00

<https://kapeli.com/cheat_sheets/Dockerfile.docset/Contents/Resources/Documents/index>

<https://dockerlabs.collabnix.com/docker/cheatsheet/>

<https://docs.docker.com/engine/reference/builder/> Common commands used in a dockerfile:

| **Instruction** | **Description** |
| --- | --- |
| [ADD](https://docs.docker.com/engine/reference/builder/#add) | Add local or remote files and directories. |
| [ARG](https://docs.docker.com/engine/reference/builder/#arg) | Use build-time variables. |
| [CMD](https://docs.docker.com/engine/reference/builder/#cmd) | Specify default commands. |
| [COPY](https://docs.docker.com/engine/reference/builder/#copy) | Copy files and directories. |
| [ENTRYPOINT](https://docs.docker.com/engine/reference/builder/#entrypoint) | Specify default executable. |
| [ENV](https://docs.docker.com/engine/reference/builder/#env) | Set environment variables. |
| [EXPOSE](https://docs.docker.com/engine/reference/builder/#expose) | Describe which ports your application is listening on. |
| [FROM](https://docs.docker.com/engine/reference/builder/#from) | Create a new build stage from a base image. |
| [HEALTHCHECK](https://docs.docker.com/engine/reference/builder/#healthcheck) | Check a container's health on startup. |
| [LABEL](https://docs.docker.com/engine/reference/builder/#label) | Add metadata to an image. |
| [MAINTAINER](https://docs.docker.com/engine/reference/builder/#maintainer-deprecated) | Specify the author of an image. |
| [ONBUILD](https://docs.docker.com/engine/reference/builder/#onbuild) | Specify instructions for when the image is used in a build. |
| [RUN](https://docs.docker.com/engine/reference/builder/#run) | Execute build commands. |
| [SHELL](https://docs.docker.com/engine/reference/builder/#shell) | Set the default shell of an image. |
| [STOPSIGNAL](https://docs.docker.com/engine/reference/builder/#stopsignal) | Specify the system call signal for exiting a container. |
| [USER](https://docs.docker.com/engine/reference/builder/#user) | Set user and group ID. |
| [VOLUME](https://docs.docker.com/engine/reference/builder/#volume) | Create volume mounts. |
| [WORKDIR](https://docs.docker.com/engine/reference/builder/#workdir) | Change working directory. |

### Pushing Image to dockerhub and azure container registry: #4

Syntax to give a tag to an image

<registry>/username/repository:tag

Docker.io/dockrhub/project:v1, if we haven’t mentioned the tag and left that empty the default tag will be latest.

For pushing image to Container Registry, Create Container Registry Using Portal

1. Create a resource Þ Azure Container Registry.
2. In container registry properties go to **Access key** Under settings, select Enable. Take note of the following values:
3. Login to ACR using below commands:

* docker login <login server name>.azurecr.io
* docker images
* docker image tag testzero.azurecr.io/helloworldapp testingzero.azurecr.io/hellowebapp:v1
* docker push testingzero.azurecr.io/hellowebapp:v1

4. Tag Local Images using command: docker image tag sandeepsoni/hellowebapp:v1 dssdemo.azurecr.io/hellowebapp:v1

5. Push images to ACR docker push dssdemo.azurecr.io/hellowebapp:v1

Deploying Docker Images in app service and container instance: #5

* Create an **app** service using azure portal.
* Change the publish to **docker** container instead of **code** while creatin app service.
* Select the same **OS** as you have for your docker image.
* **ACR** or **Docker hub** can be chosen according to the requirement from the docker option while creating the web app.
* Select **ACR** if we have any repository in that specific ACR and if it is in the same **subscription,** it will get **populated** automatically in azure container registry options in docker tab. 03:25
* You can select **private registry** if your ACR repository is in different subscription, give url as <acr name>.azurecr.io then the username/password(ACR key) and **image** and **tag** at last.

**Docker Hub**

Select **docker hub** in image source in the docker tab and select access type **public/private/private** as same as your repository in docker hub and **image** and **tag** at last. 09:00

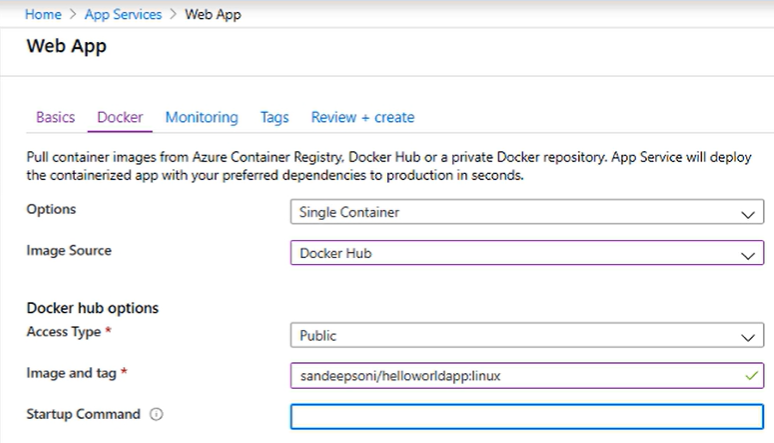


Image name can be changed in future by going into the container settings.

P.s. container settings are only visible if we select **docker** container instead of **code** while creating app service. 10:50

If we want to deploy our images to other than web apps, we can deploy them in VMs or in azure container instances as well. 14:00

Azure Pipeline for Build, Publish Docker Image and push it onto Dockerhub: #6

Azure Pipelines can be used to build images for any repository containing a Dockerfile. Building of both Linux and Windows containers is possible based on the agent platform used for the build.

**Creating a New Service Connection:**

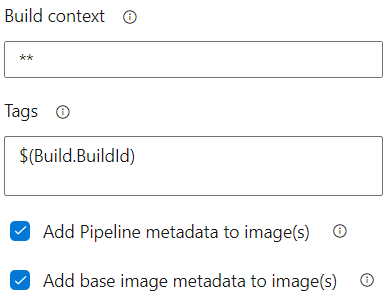
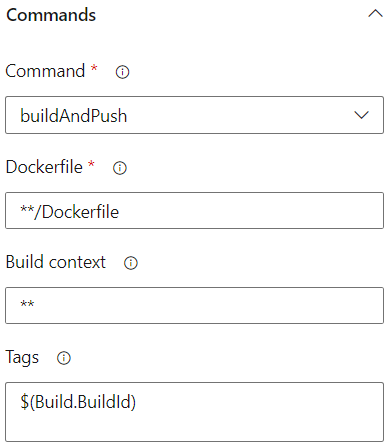
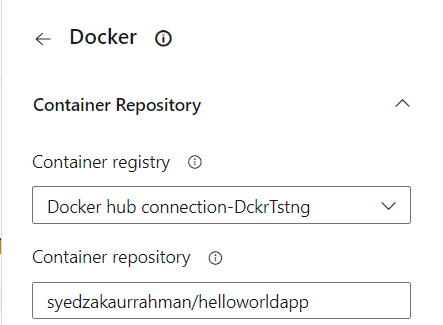
Go to the project setting of the selected project then under project settings go to service connection.

Organization Properties **-------->** Service Connection **---------->** New Service Connection **----------->** Docker Registry **--------->** Select Docker Hub under **registry type**

* Give docker Id, password and **verify**.
* Give **suitable** service connection name of your **choice**.
* Grant access permission to all pipelines.
* Verify and save.

Push the code to azure **repo** if haven’t already,including the docker **file** using which you want to **build**

* Create a new Yaml pipeline (Azure Repo Git).
* Select repository
* In configure tab you should see docker options because of the dockerfile you have pushed in the repository.
* In the configure tab select starter pipeline.
* Remove everything (all lines) below steps.
* Then using show assistant type and search Docker task and add it.
* Select the service connection you created above under **container registry**.
* Give the name of the docker repository under **container repository.**
* Select build and push under **command.**
* Leave all other options as they are by **default.**
* buildContext: 07:00



Azure Pipeline for Build, Publish Docker Image and push it onto ACR: #7

Use the same steps as [above](#Bookmark1) to create and login to an ACR and then follow the steps below.

Go to the project setting of the selected project then under project settings go to service connection.

Organization Properties **-------->** Service Connection **---------->** New Service Connection **----------->** Docker Registry **--------->** Select Azure container registry under **registry type** if you have the Azure AD access and if you don’t have Azure AD access, you need to select **others**

* Enter Docker **registry** as https://<ACR name>.azurecr.io.
* Give docker Id as <ACR name>, password (access key of ACR) and **verify**.
* Give **suitable** service connection name of your **choice**.
* Grant access permission to all pipelines and save.

Push the code to azure **repo** if haven’t already,including the docker **file** using which you want to **build**

* Create a new Yaml pipeline (Azure Repo Git).
* Select repository
* In configure tab you should see docker options because of the dockerfile you have pushed in the repository.
* In the configure tab select starter pipeline.
* Remove everything (all lines) below steps.
* Then using show assistant type and search Docker task and add it.
* Select the service connection you created above under **container registry**.
* Give the name of the ACR repository under **container repository.**
* Select build and push under **command.**
* Leave all other options as they are by **default.**