**Install Docker for Linux:**

Red Hat:

- `sudo yum install -y docker`

- `sudo service docker start`

- `sudo usermod -G docker $(whoami)`

- Restart operating system.

- Try: `docker` and `docker container ls`

Ubuntu:

- Download the docker install script: `wget -qO- https://get.docker.com/ | sh`

- `sudo usermod -aG docker $(whoami)`

- Restart operating system.

- Try: `docker` and `docker contaienr ls`

Pulling multiple Docker images from the docker using docker-compose.yml file

# docker-compose.yml

version: '3.1'

services:

a:

image: a-image

b:

image: b-image

c:

image: c-image

# .....

Then run below command

docker-compose pull --parallel

# Example docker-compose.yml file

version: '3.1'

services:

centos:

image: centos

ubuntu:

image: ubuntu

php:

image: php

node:

image: node

tomcat:

image: tomcat

python:

image: python

apache:

image: httpd

nginx:

image: nginx

redis:

image: redis

mysql:

image: mysql

golang:

image: golang

mongo:

image: mongo

consul:

image: consul

elasticsearch:

image: elasticsearch:8.1.2

mongo-express:

image: mongo-express

jenkins:

image: jenkins/jenkins:lts-jdk11

kibana:

image: kibana:8.1.2

drupal:

image: drupal

rocker.chat:

image: rocket.chat

bakdrop:

image: backdrop

postfixadmin:

image: postfixadmin

**Docker Containers and Images | Summary and Commands**

Nice work on completing the first section of this course on Docker containers and images. In this section, we learned plenty of important concepts. We got a grasp of how Docker works, from the running docker engine, the creation of docker images, and the execution of running containers themselves. We learned about the great engineering benefit of isolated environments through containers. Plus we also got a taste of some of the great engineering ideas of Docker, such as caching previous images to optimize performance, and sharing common files across containers.

Moving on, we’ll explore Docker images more deeply. We’ll create our own customized container images, and dive into more advanced features.

In the meantime, here’s a summary of the commands we’ve used thus far:

Docker Containers and Images

Docker Containers

Create an interactive terminal container with a name, an image, and a default command:

Usage: docker create -it --name=<name> <image> <command>

Example: docker create -it --name=foo ubuntu bash

List all running containers:

docker container ls

(list all containers, running or not): docker container ls -a

Start a docker container:

Usage: docker start <container name or id>

Example: docker start foo

Attach to a docker container:

Usage: docker attach <container name or id>

Example: docker attach foo

Remove a container:

Usage: docker rm <container name or id>

Example: docker rm foo

Force remove: docker rm foo -f

Run a new container:

Usage: docker run <image> <command>

Example with options: docker run --name=bar -it ubuntu bash

Remove all containers:

docker container ls -aq | xargs docker container rm

Docker Images

Remove all images:

docker image ls -aq | xargs docker rmi -f

Search for a docker image:

Usage: docker search <image>

Example: docker search ubuntu

**Docker Images in Depth | Summary and Commands**

Excellent job on completing this section on taking a deeper dive into docker images. In this section, we learned plenty of important lessons about images. For one, images are highly important since they provide the blueprints for docker containers. They are created by editing a customized document called the Dockerfile. Examples of dockerfiles, as well as hundreds of useful images are all stored on dockerhub - ready for us to explore. Next, we’ll explore more advanced features of containers. Soon, we’ll even allow multiple containers to interact in order to have really complex Docker application setups. In the meantime, here’s a collection of the Docker commands we have used thus far:

Docker Containers

Create an interactive terminal container with a name, an image, and a default command:

Usage: docker create -it --name=<name> <image> <command>

Example: docker create -it --name=foo ubuntu bash

List all running containers:

docker container ls

(list all containers, running or not): docker container ls -a

Start a docker container:

Usage: docker start <container name/id>

Example: docker start foo

Attach to a docker container:

Usage: docker attach <container name/id>

Example: docker attach foo

Remove a container:

Usage: docker rm <container name/id>

Example: docker rm foo

Force remove: docker rm foo -f

Run a new container:

Usage: docker run <image> <command>

Example with options: docker run --name=bar -it ubuntu bash

Remove all containers:

docker container ls -aq | xargs docker container rm

Execute a command in a running container:

Usage: docker exec <container name/id> <command>

Example (interactive, with tty): docker exec -it express bash

Docker Images

Remove a docker image:

Usage: docker image rmi <image id>

Example (only uses first 3 characters of image id): docker rmi 70b

Remove all images:

docker image ls -aq | xargs docker rmi -f

Search for a docker image on dockerhub:

Usage: docker search <image>

Example: docker search ubuntu

List docker images:

docker image ls

Build a Docker image:

Usage: docker build <path>

Example (also tags and names the build): docker build . -t org/serve:0.0.0

Dockerfiles

Specify a base image:

Usage: FROM <base image>

Example: FROM node:latest

Set a working directory for the container:

Usage: WORKDIR <dir>

Example: WORKDIR /usr/src/app

Run a command for the container image:

Usage: RUN command

Command: RUN npm install -g serve

Copy files into the container:

Usage: COPY <local files/directories> <container files/directories>

Example: COPY ./display ./display

Inform that a port should be exposed

Usage: EXPOSE <port>

Example: EXPOSE 80

Specify a default command for the container:

Usage (shell format): CMD <default command>

Example: CMD serve ./display

Usage (exec format, recommended): CMD [“default command”, “arguments”]

Example: CMD [“node”, “server.js”]

**Docker Container Storage - Mounts | Summary and Commands**

Excellent job on completing this section on storage and mounts with Docker containers. Knowing how to persist data across multiple containers and back to the host machine is crucial so that we can access the important information generated within containers.In this section, we covered three approaches to docker container storage. The volume mount persists data across containers by storing data in a docker-managed directory on the host machine. This is valuable when you need to hold on to container-created data, even if the container is removed.

The bind mount takes a directory from the host machine and maps them to container directories. This helps when you need to constantly update a directory on the Docker host machine that contains valuable files for containers. The tmpfs mount is unique since data in this kind of directory only lasts during the lifetime of the running container using it. The tmpfs mount is beneficial when you need to use secret data like passwords.

Next, we’ll learn how to use multiple containers simultaneously. We’ll discover how to multiple containers can and should interact. By having more than one container to work with, we’ll dive into more advanced features and concepts. Not to mention, we’ll build more complex and capable applications with Docker!

In the meantime, here’s a summary of the Docker commands we have mainly used so far:

Docker Containers

Create an interactive terminal container with a name, an image, and a default command:

Usage: docker create -it --name=<name> <image> <command>

Example: docker create -it --name=foo ubuntu bash

List all running containers:

docker container ls

(list all containers, running or not): docker container ls -a

Start a docker container:

Usage: docker start <container name/id>

Example: docker start foo

Attach to a docker container:

Usage: docker attach <container name/id>

Example: docker attach foo

Remove a container:

Usage: docker rm <container name/id>

Example: docker rm foo

Force remove: docker rm foo -f

Run a new container:

Usage: docker run <image> <command>

Example with options: docker run --name=bar -it ubuntu bash

Remove all containers:

docker container ls -aq | xargs docker container rm

Execute a command in a running container:

Usage: docker exec <container name/id> <command>

Example (interactive, with tty): docker exec -it express bash

Docker Images

Remove a docker image:

Usage: docker image rmi <image id>

Example (only uses first 3 characters of image id): docker rmi 70b

Remove all images:

docker image ls -aq | xargs docker rmi -f

Search for a docker image on dockerhub:

Usage: docker search <image>

Example: docker search ubuntu

List docker images:

docker image ls

Build a Docker image:

Usage: docker build <path>

Example (also tags and names the build): docker build . -t org/serve:0.0.0

Dockerfiles

Specify a base image:

Usage: FROM <base image>

Example: FROM node:latest

Set a working directory for the container:

Usage: WORKDIR <dir>

Example: WORKDIR /usr/src/app

Run a command for the container image:

Usage: RUN command

Command: RUN npm install -g serve

Copy files into the container:

Usage: COPY <local files/directories> <container files/directories>

Example: COPY ./display ./display

Inform that a port should be exposed

Usage: EXPOSE <port>

Example: EXPOSE 80

Specify a default command for the container:

Usage (shell format): CMD <default command>

Example: CMD serve ./display

Usage (exec format, recommended): CMD [“default command”, “arguments”]

Example: CMD [“node”, “server.js”]

Cross-Container Storage

Volumes

Create a volume

Usage: docker volume create <volume name>

Example: docker volume create shared-vol

Inspect a volume

Usage: docker volume inspect <volume name>

Example: docker volume inspect shared-vol

Mount a container with a volume using docker run

Usage: --mount source=<volume name>, target=<container dir>

Example:

docker run -it --name=foo --mount source=shared-vol,target=/src/shared ubuntu bash

Bind Mounts

Mount a container with a bind mount using docker run

Usage: --mount type=bind source=<host dir>, target=<container dir>

Example:

docker run -it --name=foo --mount type=bind source=/Users/foo/bindmountdir, \

target=/src/mountdir ubuntu bash

Tmpfs mounts

Mount a container with a tmpfs mount using docker run

Usage: --mount type=tmpfs, destination=<container dir>

Example:

docker run -it --name=baz --mount type=tmpfs, destination=/tmpdir ubuntu bash

**Multicontainer Docker: Networking and Compose | Summary and Commands**

Excellent work on completing this section on multicontainer docker, with networking and Compose. During this section, we covered a lot of very important ground with Docker.

With networking, we saw how more than one container could communicate in Docker. We have default networks to use in container, such as the bridge network which allows us to immediately connect running containers. We can also set up private networks for a collection of containers. This private network even has its own embedded DNS server so that containers can connect by container name.

We also worked with Docker Compose for the first time. Compose allows us to write the configuration for more than one container all at once. The group of containers created by compose makes up an application. Compose has a lot of features and benefits: it sets up a private network, and allows us to see the configuration for more than one of an application’s containers all at once.

Moving on, we’ll dive in more deeply into compose, and see more advanced ways we can configure multicontainer applications in Docker.

In the meantime, here’s a summary of the critical Docker commands we’ve used thus far:

Docker Containers

Create an interactive terminal container with a name, an image, and a default command:

Usage: docker create -it --name=<name> <image> <command>

Example: docker create -it --name=foo ubuntu bash

List all running containers:

docker container ls

(list all containers, running or not): docker container ls -a

Start a docker container:

Usage: docker start <container name/id>

Example: docker start foo

Attach to a docker container:

Usage: docker attach <container name/id>

Example: docker attach foo

Remove a container:

Usage: docker rm <container name/id>

Example: docker rm foo

Force remove: docker rm foo -f

Run a new container:

Usage: docker run <image> <command>

Example with options: docker run --name=bar -it ubuntu bash

Remove all containers:

docker container ls -aq | xargs docker container rm

Execute a command in a running container:

Usage: docker exec <container name/id> <command>

Example (interactive, with tty): docker exec -it express bash

Docker Images

Remove a docker image:

Usage: docker image rmi <image id>

Example (only uses first 3 characters of image id): docker rmi 70b

Remove all images:

docker image ls -aq | xargs docker rmi -f

Search for a docker image on dockerhub:

Usage: docker search <image>

Example: docker search ubuntu

List docker images:

docker image ls

Build a Docker image:

Usage: docker build <path>

Example (also tags and names the build): docker build . -t org/serve:0.0.0

Dockerfiles

Specify a base image:

Usage: FROM <base image>

Example: FROM node:latest

Set a working directory for the container:

Usage: WORKDIR <dir>

Example: WORKDIR /usr/src/app

Run a command for the container image:

Usage: RUN command

Command: RUN npm install -g serve

Copy files into the container:

Usage: COPY <local files/directories> <container files/directories>

Example: COPY ./display ./display

Inform that a port should be exposed

Usage: EXPOSE <port>

Example: EXPOSE 80

Specify a default command for the container:

Usage (shell format): CMD <default command>

Example: CMD serve ./display

Usage (exec format, recommended): CMD [“default command”, “arguments”]

Example: CMD [“node”, “server.js”]

Cross-Container Storage

Volumes

Create a volume

Usage: docker volume create <volume name>

Example: docker volume create shared-vol

Inspect a volume

Usage: docker volume inspect <volume name>

Example: docker volume inspect shared-vol

Mount a container with a volume using docker run

Usage: --mount source=<volume name>, target=<container dir>

Example:

docker run -it --name=foo --mount source=shared-vol,target=/src/shared ubuntu bash

Bind Mounts

Mount a container with a bind mount using docker run

Usage: --mount type=bind source=<host dir>, target=<container dir>

Example:

docker run -it --name=foo --mount type=bind source=/Users/foo/bindmountdir, \

target=/src/mountdir ubuntu bash

Tmpfs mounts

Mount a container with a tmpfs mount using docker run

Usage: --mount type=tmpfs, destination=<container dir>

Example:

docker run -it --name=baz --mount type=tmpfs, destination=/tmpdir ubuntu bash

Docker Networking

List docker networks

docker network ls

Inspect a docker network

Usage: docker network inspect <network name>

Example: docker network inspect bridge

Create a docker network

Usage: docker network create <network name>

Example: docker network create privatenw

Run a container with a custom docker network:

Usage: --network=<network name>

Example: docker run --network=privatenw -it --name=goo busybox

Docker Compose

Start a compose application

At the root (where docker-compose.yml is located): docker-compose up

Start a compose application and rebuild images:

Docker-compose up --build

docker-compose.yml

Version

Current version is 3. So at the top of the file, specify: version: ‘3’

Services with builds

Have a services key in the file. List out services one indent at a time.

Dependencies

Use the depends\_on key and specify dependencies with a list. Each container dependency is marked by a dash, such as: -backend

**Docker Compose in Depth - Volumes and Networks | Summary and Commands**

Excellent work on completing this section on even deeper dive into Docker compose.

In this section, we covered Docker-compose volumes. With volumes, we have the ability to connect a directory on the Docker host to one in the docker container. This makes sure that as we update files in the Docker host, they are simultaneously updated in our container. We also explored networking in our most ambitious compose application yet - using postgresql, node+express, flask, and php to create four services. The custom networks allow us to specifically connect services to each other, to create more privileged and secure access between containers in the applications.

Next, we’ll explore Docker swarm to explore Docker on more than one host. With more than one docker host, Docker will become even more powerful and exciting!

In the meantime, here’s the updated summary of commands that we’ve used so far:

Docker Containers

Create an interactive terminal container with a name, an image, and a default command:

Usage: docker create -it --name=<name> <image> <command>

Example: docker create -it --name=foo ubuntu bash

List all running containers:

docker container ls

(list all containers, running or not): docker container ls -a

Start a docker container:

Usage: docker start <container name/id>

Example: docker start foo

Attach to a docker container:

Usage: docker attach <container name/id>

Example: docker attach foo

Remove a container:

Usage: docker rm <container name/id>

Example: docker rm foo

Force remove: docker rm foo -f

Run a new container:

Usage: docker run <image> <command>

Example with options: docker run --name=bar -it ubuntu bash

Remove all containers:

docker container ls -aq | xargs docker container rm

Execute a command in a running container:

Usage: docker exec <container name/id> <command>

Example (interactive, with tty): docker exec -it express bash

Docker Images

Remove a docker image:

Usage: docker image rmi <image id>

Example (only uses first 3 characters of image id): docker rmi 70b

Remove all images:

docker image ls -aq | xargs docker rmi -f

Search for a docker image on dockerhub:

Usage: docker search <image>

Example: docker search ubuntu

List docker images:

docker image ls

Build a Docker image:

Usage: docker build <path>

Example (also tags and names the build): docker build . -t org/serve:0.0.0

Dockerfiles

Specify a base image:

Usage: FROM <base image>

Example: FROM node:latest

Set a working directory for the container:

Usage: WORKDIR <dir>

Example: WORKDIR /usr/src/app

Run a command for the container image:

Usage: RUN command

Command: RUN npm install -g serve

Copy files into the container:

Usage: COPY <local files/directories> <container files/directories>

Example: COPY ./display ./display

Inform that a port should be exposed

Usage: EXPOSE <port>

Example: EXPOSE 80

Specify a default command for the container:

Usage (shell format): CMD <default command>

Example: CMD serve ./display

Usage (exec format, recommended): CMD [“default command”, “arguments”]

Example: CMD [“node”, “server.js”]

Cross-Container Storage

Volumes

Create a volume

Usage: docker volume create <volume name>

Example: docker volume create shared-vol

Inspect a volume

Usage: docker volume inspect <volume name>

Example: docker volume inspect shared-vol

Mount a container with a volume using docker run

Usage: --mount source=<volume name>, target=<container dir>

Example:

docker run -it --name=foo --mount source=shared-vol,target=/src/shared ubuntu bash

Bind Mounts

Mount a container with a bind mount using docker run

Usage: --mount type=bind source=<host dir>, target=<container dir>

Example:

docker run -it --name=foo --mount type=bind source=/Users/foo/bindmountdir, \

target=/src/mountdir ubuntu bash

Tmpfs mounts

Mount a container with a tmpfs mount using docker run

Usage: --mount type=tmpfs, destination=<container dir>

Example:

docker run -it --name=baz --mount type=tmpfs, destination=/tmpdir ubuntu bash

Docker Networking

List docker networks

docker network ls

Inspect a docker network

Usage: docker network inspect <network name>

Example: docker network inspect bridge

Create a docker network

Usage: docker network create <network name>

Example: docker network create privatenw

Run a container with a custom docker network:

Usage: --network=<network name>

Example: docker run --network=privatenw -it --name=goo busybox

Docker Compose

Start a compose application

At the root (where docker-compose.yml is located): docker-compose up

Start a compose application and rebuild images:

Docker-compose up --build

docker-compose.yml

Version

Current version is 3. So at the top of the file, specify: version: ‘3’

Services with builds

Have a services key in the file. List out services one indent at a time.

Dependencies

Use the depends\_on key and specify dependencies with a list. Each container dependency is marked by a dash, such as: -backend

Volumes

Have a volume key per service.

Connect a Docker host directory to a container directory, by joining them with a colon.

Example: ./dockerhostdir:/containerdir

Networks

Declare networks at the bottom of the file.

Specify each service’s network(s) with the networks option for each service.

**Docker Swarm | Summary and Commands**

Great work on completing this section on Docker Swarm. In this section, we went over quite a few important concepts.

For one, we explored docker swarm and learned how having more than one machine running docker can really help engineering efforts. With docker swarm, we have a system for extreme scalability and reliability. Not to mention, we great benefits such as the swarm routing mesh, and load balancing between nodes (docker hosts connected to the swarm).

NOTE: Be sure to terminate your amazon ec2 instances once you’re done with them. Even if it’s less than a dollar, you probably don’t want charges going to your credit card for unused instances. You can terminate the instances by updating their instance state with the “Actions” dropdown in the management console.

Finally, here’s the summary of commands that we’ve used so far. Head to the bottom of the summary to see new notes on Docker swarm.

Docker Containers

Create an interactive terminal container with a name, an image, and a default command:

Usage: docker create -it --name=<name> <image> <command>

Example: docker create -it --name=foo ubuntu bash

List all running containers:

docker container ls

(list all containers, running or not): docker container ls -a

Start a docker container:

Usage: docker start <container name/id>

Example: docker start foo

Attach to a docker container:

Usage: docker attach <container name/id>

Example: docker attach foo

Remove a container:

Usage: docker rm <container name/id>

Example: docker rm foo

Force remove: docker rm foo -f

Run a new container:

Usage: docker run <image> <command>

Example with options: docker run --name=bar -it ubuntu bash

Remove all containers:

docker container ls -aq | xargs docker container rm

Execute a command in a running container:

Usage: docker exec <container name/id> <command>

Example (interactive, with tty): docker exec -it express bash

Docker Images

Remove a docker image:

Usage: docker image rmi <image id>

Example (only uses first 3 characters of image id): docker rmi 70b

Remove all images:

docker image ls -aq | xargs docker rmi -f

Search for a docker image on dockerhub:

Usage: docker search <image>

Example: docker search ubuntu

List docker images:

docker image ls

Build a Docker image:

Usage: docker build <path>

Example (also tags and names the build): docker build . -t org/serve:0.0.0

Dockerfiles

Specify a base image:

Usage: FROM <base image>

Example: FROM node:latest

Set a working directory for the container:

Usage: WORKDIR <dir>

Example: WORKDIR /usr/src/app

Run a command for the container image:

Usage: RUN command

Command: RUN npm install -g serve

Copy files into the container:

Usage: COPY <local files/directories> <container files/directories>

Example: COPY ./display ./display

Inform that a port should be exposed

Usage: EXPOSE <port>

Example: EXPOSE 80

Specify a default command for the container:

Usage (shell format): CMD <default command>

Example: CMD serve ./display

Usage (exec format, recommended): CMD [“default command”, “arguments”]

Example: CMD [“node”, “server.js”]

Cross-Container Storage

Volumes

Create a volume

Usage: docker volume create <volume name>

Example: docker volume create shared-vol

Inspect a volume

Usage: docker volume inspect <volume name>

Example: docker volume inspect shared-vol

Mount a container with a volume using docker run

Usage: --mount source=<volume name>, target=<container dir>

Example:

docker run -it --name=foo --mount source=shared-vol,target=/src/shared ubuntu bash

Bind Mounts

Mount a container with a bind mount using docker run

Usage: --mount type=bind source=<host dir>, target=<container dir>

Example:

docker run -it --name=foo --mount type=bind source=/Users/foo/bindmountdir, \

target=/src/mountdir ubuntu bash

Tmpfs mounts

Mount a container with a tmpfs mount using docker run

Usage: --mount type=tmpfs, destination=<container dir>

Example:

docker run -it --name=baz --mount type=tmpfs, destination=/tmpdir ubuntu bash

Docker Networking

List docker networks

docker network ls

Inspect a docker network

Usage: docker network inspect <network name>

Example: docker network inspect bridge

Create a docker network

Usage: docker network create <network name>

Example: docker network create privatenw

Run a container with a custom docker network:

Usage: --network=<network name>

Example: docker run --network=privatenw -it --name=goo busybox

Docker Compose

Start a compose application

At the root (where docker-compose.yml is located): docker-compose up

Start a compose application and rebuild images:

Docker-compose up --build

docker-compose.yml

Version

Current version is 3. So at the top of the file, specify: version: ‘3’

Services with builds

Have a services key in the file. List out services one indent at a time.

Dependencies

Use the depends\_on key and specify dependencies with a list. Each container dependency is marked by a dash, such as: -backend

Volumes

Have a volume key per service.

Connect a Docker host directory to a container directory, by joining them with a colon.

Example: ./dockerhostdir:/containerdir

Networks

Declare networks at the bottom of the file.

Specify each service’s network(s) with the networks option for each service.

Docker Swarm

Initialize a swarm in a node

Usage: docker swarm init --advertise-addr=<node ip>

Example: docker swarm init --advertise-addr=172.31.17.31

After initializing the swarm, you will find a join command for worker/other manager nodes

Example: docker swarm join --token SWMTKN-1-592fo0c31guwi9cw58jpaw89fafzyw5fzbk9dwiw8bm4xxpad-94vn587o9o3r73h3e5esujxm9 172.31.17.31:2377

List docker nodes from a manger:

docker node ls

Create a service for the swarm:

Usage: docker service create --name=<service name> --publish=<host port:service port> <service image>

Example: docker service create --name=site --publish=80:80 nginx

List services:

docker service ls

List the running tasks for a service:

Usage: docker service ps <service name>

Example: docker service ps site

Update a service

Usage: docker service update [options] <service name>

Example: docker service update --replicas=6 site

**Docker Cloud and Continuous Integration | Summary and Commands**

We just explored docker cloud and some development operation workflows with continuous deployment and continuous integration. We saw how to configure automatic image builds. We also set up automatic tests for requests to add new code to the master branch of our codebase.

The extent of Docker Cloud far exceeds the scope of this course. Docker Cloud becomes even more powerful when you consider its ability to host services with compose files. Not to mention, you can host layouts of entire applications with stackfiles that outline a group of services that create an application.

We’ve reached the end of the course. In the meantime, here’s the overall summary of major commands we’ve used with Docker so far:

Docker Containers

Create an interactive terminal container with a name, an image, and a default command:

Usage: docker create -it --name=<name> <image> <command>

Example: docker create -it --name=foo ubuntu bash

List all running containers:

docker container ls

(list all containers, running or not): docker container ls -a

Start a docker container:

Usage: docker start <container name/id>

Example: docker start foo

Attach to a docker container:

Usage: docker attach <container name/id>

Example: docker attach foo

Remove a container:

Usage: docker rm <container name/id>

Example: docker rm foo

Force remove: docker rm foo -f

Run a new container:

Usage: docker run <image> <command>

Example with options: docker run --name=bar -it ubuntu bash

Remove all containers:

docker container ls -aq | xargs docker container rm

Execute a command in a running container:

Usage: docker exec <container name/id> <command>

Example (interactive, with tty): docker exec -it express bash

Docker Images

Remove a docker image:

Usage: docker image rmi <image id>

Example (only uses first 3 characters of image id): docker rmi 70b

Remove all images:

docker image ls -aq | xargs docker rmi -f

Search for a docker image on dockerhub:

Usage: docker search <image>

Example: docker search ubuntu

List docker images:

docker image ls

Build a Docker image:

Usage: docker build <path>

Example (also tags and names the build): docker build . -t org/serve:0.0.0

Dockerfiles

Specify a base image:

Usage: FROM <base image>

Example: FROM node:latest

Set a working directory for the container:

Usage: WORKDIR <dir>

Example: WORKDIR /usr/src/app

Run a command for the container image:

Usage: RUN command

Command: RUN npm install -g serve

Copy files into the container:

Usage: COPY <local files/directories> <container files/directories>

Example: COPY ./display ./display

Inform that a port should be exposed

Usage: EXPOSE <port>

Example: EXPOSE 80

Specify a default command for the container:

Usage (shell format): CMD <default command>

Example: CMD serve ./display

Usage (exec format, recommended): CMD [“default command”, “arguments”]

Example: CMD [“node”, “server.js”]

Cross-Container Storage

Volumes

Create a volume

Usage: docker volume create <volume name>

Example: docker volume create shared-vol

Inspect a volume

Usage: docker volume inspect <volume name>

Example: docker volume inspect shared-vol

Mount a container with a volume using docker run

Usage: --mount source=<volume name>, target=<container dir>

Example:

docker run -it --name=foo --mount source=shared-vol,target=/src/shared ubuntu bash

Bind Mounts

Mount a container with a bind mount using docker run

Usage: --mount type=bind source=<host dir>, target=<container dir>

Example:

docker run -it --name=foo --mount type=bind source=/Users/foo/bindmountdir, \

target=/src/mountdir ubuntu bash

Tmpfs mounts

Mount a container with a tmpfs mount using docker run

Usage: --mount type=tmpfs, destination=<container dir>

Example:

docker run -it --name=baz --mount type=tmpfs, destination=/tmpdir ubuntu bash

Docker Networking

List docker networks

docker network ls

Inspect a docker network

Usage: docker network inspect <network name>

Example: docker network inspect bridge

Create a docker network

Usage: docker network create <network name>

Example: docker network create privatenw

Run a container with a custom docker network:

Usage: --network=<network name>

Example: docker run --network=privatenw -it --name=goo busybox

Docker Compose

Start a compose application

At the root (where docker-compose.yml is located): docker-compose up

Start a compose application and rebuild images:

Docker-compose up --build

docker-compose.yml

Version

Current version is 3. So at the top of the file, specify: version: ‘3’

Services with builds

Have a services key in the file. List out services one indent at a time.

Dependencies

Use the depends\_on key and specify dependencies with a list. Each container dependency is marked by a dash, such as: -backend

Volumes

Have a volume key per service.

Connect a Docker host directory to a container directory, by joining them with a colon.

Example: ./dockerhostdir:/containerdir

Networks

Declare networks at the bottom of the file.

Specify each service’s network(s) with the networks option for each service.

Docker Swarm

Initialize a swarm in a node

Usage: docker swarm init --advertise-addr=<node ip>

Example: docker swarm init --advertise-addr=172.31.17.31

After initializing the swarm, you will find a join command for worker/other manager nodes

Example: docker swarm join --token SWMTKN-1-592fo0c31guwi9cw58jpaw89fafzyw5fzbk9dwiw8bm4xxpad-94vn587o9o3r73h3e5esujxm9 172.31.17.31:2377

List docker nodes from a manger:

docker node ls

Create a service for the swarm:

Usage: docker service create --name=<service name> --publish=<host port:service port> <service image>

Example: docker service create --name=site --publish=80:80 nginx

List services:

docker service ls

List the running tasks for a service:

Usage: docker service ps <service name>

Example: docker service ps site

Update a service

Usage: docker service update [options] <service name>

Example: docker service update --replicas=6 site