Docker Compose

Docker compose

- Compose is a tool for **defining and running** multi-container Docker applications.
- With Compose, you use a YAML file to configure your application's services. Then, with a single command, you create and start all the services from your configuration.
- Using Compose is basically a three-step process:
  - Define your app's environment with a Dockerfile so it can be reproduced anywhere.
  - Define the services that make up your app in docker-compose.yml so they can be run together in an isolated environment.
  - Run docker-compose up and Compose starts and runs your entire app.

Docker compose

- Docker Compose has commands for managing the whole lifecycle of your application:
  - Start, stop, and rebuild services
  - View the status of running services
  - Stream the log output of running services
  - Run a one-off command on a service

Docker compose features

- Preserve volume data when containers are created
- Only recreate containers that have changed
- Support Variables

Preserve volume

- Docker Compose **preserves all volumes** used by your services.
- When docker-compose up runs, if it finds any containers from previous runs, it copies the volumes from the old container to the new container.
- This process ensures that any data you've created in volumes isn't lost.

Recreate containers

- Compose caches the configuration used to create a container.
- When you restart a service that has not changed, Compose re-uses the existing containers.
- Re-using containers means that you can make changes to your environment very quickly.

Variables

- Compose supports variables in the Compose file.
- You can use these variables to customize your composition for different environments, or different users

Docker compose

- Assuming that you already have Docker containers ready.
- In order to use `docker-compose` we will need to add the compose file
- Create a new yml file named: `docker-compose.yml`
- docker compose is updated very often and the current version (not docker but the compose version) is 3
  - The first line is the `version`

```
version : '3' #The compose file format
```

Docker compose

- The second part are the `services`
- This part tells docker what containers to build

```
version: '3'
services:
    our-service:        # Any desired name for our service
        build: ...      # The build folder relative to the compose file
        volumes:        # List of volumes
            - ...
        ports:
            - XX:XX     # Ports to expose
    service2:
        image: ...      # In this case we deploy image and not building the container form folder
        depends-on: ... # Add service dependency if required
            - our-service
```

Docker compose

- To execute docker compose we type:

```
# in the folder where the docker-compose is
docker-compose up

# Run in detached mode
docker-compose up -d
```

- `docker-compose up` will pull, install and restart the services
- We can split docker containers into separate folders and compose will build the container from the folders itself

Docker compose - Networking

- Docker compose create virtual network for our containers which allow them to communicate with each other
- By default all the containers within the same compose file can find each other
- The host name is the set to the service name we defined in the yml file

```
version: '3'
services:
    my-service:
        build: <image>
```

```
# In the code we can reference it like:
http://my-service
```

Docker compose - commands

Command	Description
build	Build the containers
start/stop	Start/Stop the containers
pause/unpause	Start the containers
ps	Display container states
start	Start the containers
up	Builds, (re)creates, starts, and attaches to containers for a service.
up --scale	scale more instances of a given container
down	Stops containers and removes containers, networks, volumes, and images created by up.
events	Display changes (history) of containers events

Docker compose - config

- Validate and view the Compose file.
- It will print out the "complied" file with full paths, volumes and more

```
docker-compose config
```

Docker compose - build

`docker-compose build` has several ways to build container:

```
# build from Dockerfile
build: .  # Default docker file in the current folder
args:     # Add build arguments
    arg1: ...
    arg2: ...

# build from Custom Dockerfile
build:
    context: <dir>
    dockerfile: <Dockerfile name>

# build from image
image: <image name>
```

Docker compose - CMD / ENTRYPOINT

- Command

```
# command to execute
command: bundle exec thin -p 3000
command: [bundle, exec, thin, -p, 3000]
```

- entrypoint

```
# override the entrypoint
entrypoint: <path to script>

# or - CLI commands (Array)
entrypoint: [nodemon, server.js]
```

Docker compose - Dependencies

- Makes the `db` service available as the hostname `database` (implies depends_on)

```
links:
    - db:database
    - redis
```

- Make sure `db` is alive before starting

```
depends_on:
    - db
```

HANDS ON

THE END.