Working with Docker

- Cheatsheet II -

MVP: Build and attach a cluster

docker-compose up

--attach

--build

MVP: Bring down a cluster

docker-compose down

--remove-orphans

--rmi --volumes

Stop and remove all containers (hard)

```
$ docker container stop $(docker container ls -a -q) &&
   docker container rm $(docker container ls -a -q)
```

Execute a bash command in a container

```
$ docker exec <container-name> bash -c 'echo "Hello!"'
```

Execute bash script in container (3 steps)

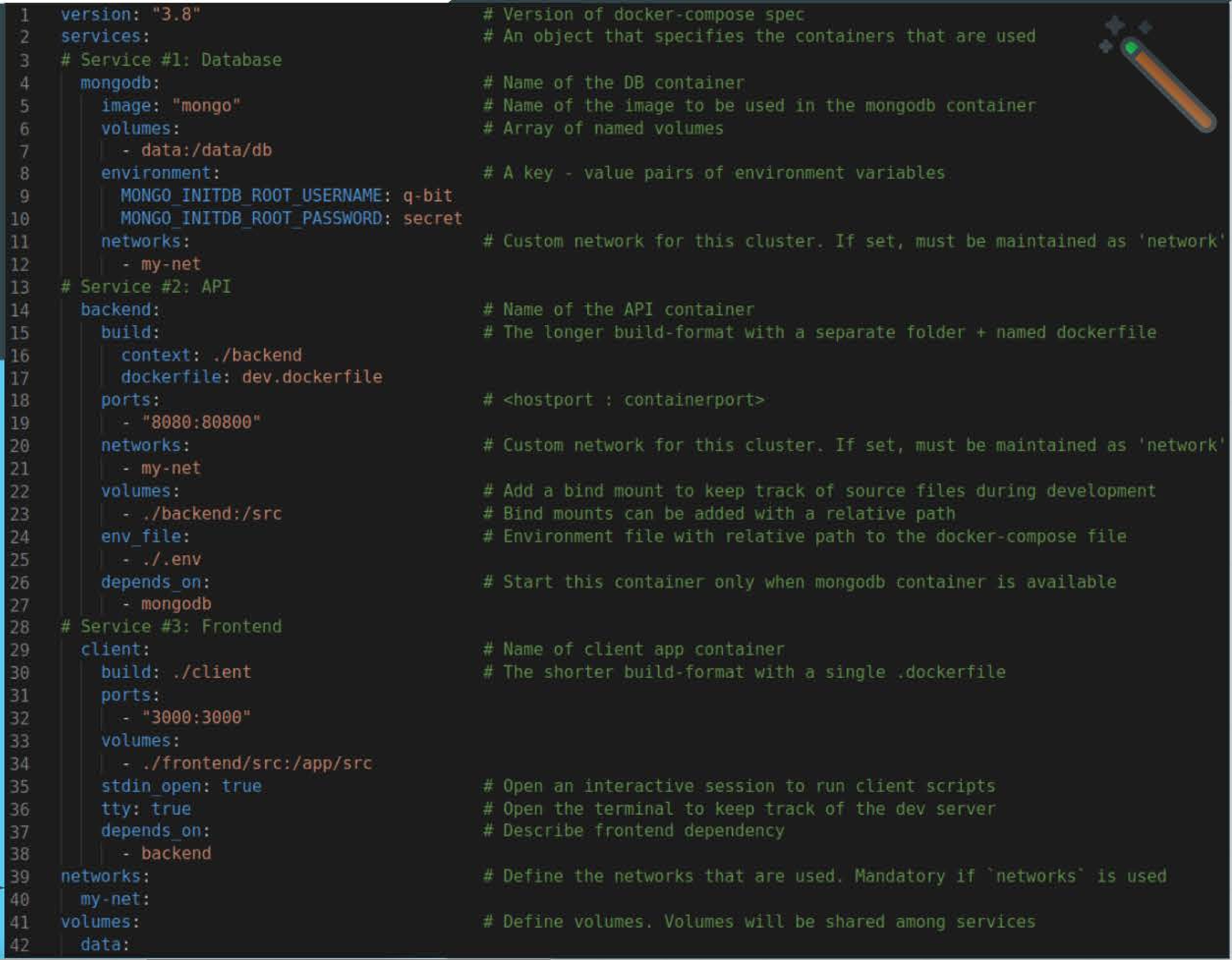
```
$ chmod +x /tmp/echo.sh
$ docker cp /tmp/echo.sh <container-name>:/tmp/echo.sh
$ docker exec -it <container-name> bash -c /tmp/echo.sh
> script.out
```

Get a list of names of all containers

```
$ docker container ls -a -q
| xargs -n 1 docker inspect --format '{{.Name}}'
| sed 's#^/##'
```

Launch a docker-compose powered fullstack app

docker-compose.yaml



```yaml
version: "3.8"                            # Version of docker-compose spec
services:                                 # An object that specifies the containers that are used
# Service #1: Database
  mongodb:                                # Name of the DB container
    image: "mongo"                        # Name of the image to be used in the mongodb container
    volumes:                              # Array of named volumes
      - data:/data/db
    environment:                          # A key - value pairs of environment variables
      MONGO_INITDB_ROOT_USERNAME: q-bit
      MONGO_INITDB_ROOT_PASSWORD: secret
    networks:                             # Custom network for this cluster. If set, must be maintained as 'network'
      - my-net
# Service #2: API
  backend:                                # Name of the API container
    build:                                # The longer build-format with a separate folder + named dockerfile
      context: ./backend
      dockerfile: dev.dockerfile
    ports:                                # <hostport : containerport>
      - "8080:80800"
    networks:                             # Custom network for this cluster. If set, must be maintained as 'network'
      - my-net
    volumes:                              # Add a bind mount to keep track of source files during development
      - ./backend:/src                    # Bind mounts can be added with a relative path
    env_file:                             # Environment file with relative path to the docker-compose file
      - ./.env
    depends_on:                           # Start this container only when mongodb container is available
      - mongodb
# Service #3: Frontend
  client:                                 # Name of client app container
    build: ./client                       # The shorter build-format with a single .dockerfile
    ports:
      - "3000:3000"
    volumes:
      - ./frontend/src:/app/src
    stdin_open: true                      # Open an interactive session to run client scripts
    tty: true                             # Open the terminal to keep track of the dev server
    depends_on:                           # Describe frontend dependency
      - backend
networks:                                 # Define the networks that are used. Mandatory if `networks` is used
  my-net:
volumes:                                  # Define volumes. Volumes will be shared among services
  data:
```