DOCKER COMPOSE CHEAT SHEET

Create A docker-compose.yaml

To manage multiple containers with Docker Compose, you need a YAML file. This file is named docker-compose.yaml. A docker-compose.yaml file consists of a version, services, volumes, networks, configs, and secrets.

```
# string that represents the
version: '3'

# an object where each key represents a new service
# e.g., your client application, web server, database, ...
services:
  client:
    # define your client
    # e.g., image, ports, env vars, networks, volumes, ...
  server:
    # define your server
    # e.g., image, ports, env vars, networks, volumes, ...
  database:
    # define your database
    # e.g., image, ports, env vars, networks, volumes, ...

# an object where each key represents a new volume
# e.g., to persist the database, store images, documents, ..
# volumes need to be explicitly bound to a service
volumes:
  database_volume:
    # define the settings of your volume
    # if you leave this empty, defaults will be applied

# an object where each key represents a network
# e.g., to communicate with containers in the same network
# networks need to be explicitly bound to a service
networks:
  # docker creates a default network for all services
  # in a docker-compose file, every service joins the
  # default network and can contact every other container
  # by its name e.g., docker sets up a DNS entry in server
  # for the client and database
  # so a call from the server container to
  # <protocol>://database:<port>
  # is equivalent to
  # <protocol>://<IP-address-of-database>:<port>
  # we can also define explicit networks
  # and let only some containers join
  # e.g., database and server
  server_database_network:
    # define the settings of your network
    # if you leave this empty, defaults will be applied

# an object where each key represents a config
# e.g., to adapt behavior without rebuilding an image
# configs need to be explicitly bound to a service
configs:
  some_config:

# an object where each key represents a secret
# e.g., to adapt behavior without rebuilding an image
# secrets act like configs but with a specific focus
# on sensitive information
# secrets need to be explicitly bound to a service
secrets:
  some_secret:
```

Services From Dockerfiles

Services are the core of Docker Compose. They define the containers and how to manage them. We can use Dockerfiles in Docker Compose.

```
services:
  server: # we are defining a service called server
    build: # we use this command to create the image
    # the dot (.) represents the current working directory
      context: .
      dockerfile: Dockerfile # the path to the file
      # ... other configuration ...
```

Services From Images

We can also use images from Docker Hub or a private registry. We can publish ports, set environment variables, join networks, create health checks, and persist data in volumes.

```
services:
  database: # we are defining a service called database
    # if the image is available locally, it will be used
    # otherwise, docker will pull the image from Docker Hub
    # use private images by adding the registry URL
    # https://myprivateregistry.com/<image>:<version>
    image: postgres:16.1
    ports:
      # publish ports to access the database from outside
      # <host-port>:<container-port>
      - 5432:5432
    environment:
      # set environment variables
      VARIABLE_NAME: variable_value
    # restart the container if it fails
    restart: always
    healthcheck:
      # the command to check if the container is healthy
      test: ["CMD", "pg_ready", "-U", "postgres"]
      interval: 30s
      timeout: 10s
      retries: 3
      start_period: 30s
```

Volumes And Networks

We can persist data throughout container starts in volumes (e.g., the data from our database). Networks are used to easily communicate between containers in a docker-compose.yaml.

```
services:
  database:
    # ... other configuration ...
    # volumes need to be explicitly bound to a service
    # mounted volumes: <host-path>:<container-path>
    # named volumes: <volume-name>:<container-path>
    volumes:
      - postgres_data_volume:/var/lib/postgresql/data
    # networks need to be explicitly bound to a service
    networks:
      - server_database_network # <network-name>
volumes:
  postgres_data_volume: # use explicitly in service
networks:
  server_database_network: # use explicitly in service
```

Start, Stop, Remove, And Access

```
# start all services in a docker-compose.yaml at once
$ docker compose up # add --detach to run in background
# stop one service
$ docker compose stop <service-name> # e.g., database
# restart stopped service (use start for removed services)
$ docker compose restart <service-name>
# remove a stopped service
$ docker compose rm <service-name>
# stop and remove all services
$ docker compose down
# rebuild all services
$ docker compose build
# create an ssh-like connection into a container
$ docker compose exec -it <service-name> <command>
# get the logs from all services
$ docker compose logs # extendable with <service-name>
```

Blog: https://devopscycle.com/blog/the-ultimate-docker-compose-cheat-sheet/
GitHub: https://github.com/aichbauer/the-ultimate-docker-compose-cheat-sheet/
Consulting: https://devopscycle.com/