Cheatography

Docker Compose Cheat Sheet

by Gaurav Pandey (gauravpandey44) via cheatography.com/69622/cs/20583/

Basics

Docker-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.

Need to good to yml file directory to successfully run the docker-compose commands.
docker-compose start wordpress_db :it will only start 1 service
but docker-compose start : will start all the services
similar is the case with other commands.

docker-compose start

Starts an existing service container.

docker-compose stop

-t, --timeout specify a shutdown timeout in seconds.(default: 10)

Stops running containers without removing them. They can be started again with `docker-compose start`.

docker-compose pause

Pauses running containers of a service. They can be unpaused with docker-compose unpause

docker-compose unpause

Unpauses paused containers of a service.

docker-compose restart

Restarts all stopped and running services.

docker-compose ps

-q, --quiet Only display IDs

Shows list of containers for a service.

docker-compose logs

-f, --follow Follow log output.

Displays log output from services.

docker-compose top

View the processes running within each service container.

docker-compose pull

Pulls an image associated with a service defined in a docker-compose.yml file, but does not start containers based on those images.

docker-compose rm

Removes stopped service containers.By default, anonymous volumes attached to containers are not removed. You can override this with -v. To list all volumes, use docker volume ls.
-f, --force – Don't ask to confirm the removal
-s, --stop – Stop the containers, if required, before removing
-v – Remove any anonymous volumes attached to containers

docker-compose.yml

```
version: "3.7"
services:
  wordpress_db:
    container_name: "wordpress_db"
    image: "mysql:5.7"
    volumes:
      - ~/dockers/wordpress/.data/wordpress_-
db:/var/lib/mysql
    environment:
      MYSQL_USER: gaurav
      MYSQL_PASSWORD: victory
      MYSQL_DATABASE: db
      MYSQL_RANDOM_ROOT_PASSWORD: '1'
    networks:
      - wordpress_network
    ports:
      - 3307:3306
  wordpress_web:
    container_name: "wordpress_web"
    image: "wordpress"
    volumes:
      - ~/dockers/wordpress/.data/wordpress_-
web:/var/www/html
    environment:
      WORDPRESS_DB_HOST: wordpress_db
      WORDPRESS_DB_USER: gaurav
      WORDPRESS_DB_PASSWORD: victory
      WORDPRESS_DB_NAME: db
```


By **Gaurav Pandey**
(gauravpandey44)

cheatography.com/gauravpandey44/

Published 21st September, 2019.
Last updated 23rd September, 2019.

Cheatography

Docker Compose Cheat Sheet

by Gaurav Pandey (gauravpandey44) via cheatography.com/69622/cs/20583/

docker-compose.yml (cont)

```
    networks:
      - wordpress_network
    ports:
      - 8080:80
    depends_on:
      - wordpress_db
networks:
  wordpress_network:
```

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

Network

```
# creates a custom network called  frontend
networks:
  frontend:
```

docker compose up

docker-compose up	use docker-compose.yml
docker-compose -f <filename.yml> -f <filenamelocal.yml> up	use custom yml files
-d, --detach	background detached mode
--build	forcefully Build images before starting containers.
--no-build	skips the image build process
--force-recreate	Recreate containers even if their configuration and image haven't changed.
--no-color	Produce monochrome output.

docker compose up (cont)

--scale SERVICE=NUM	Scale SERVICE to NUM instances. Overrides the `scale` setting in the Compose file if present.

docker-compose up is used to start a project. It tries to automate a series of operations including building a mirror, (re)creating a service, starting a service, and associating a service-related container.
It also builds the images if the images do not exist and starts the containers:

docker-compose down

Stops containers and removes containers, networks, volumes, and images
By default, the only things removed are:
- Containers for services defined in the Compose file
- Networks defined in the `networks` section of the Compose file
- The default network, if one is used
created by `up`.Networks and volumes defined as `external` are never removed.
use -v to remove volumes also along with other things

docker-compose build

only builds the images, does not start the containers:

docker-compose run

Runs a one-time command against a service. For example, the following command starts the web service and runs bash as its command :
docker-compose run wordpress_db bash

docker-compose version

Prints the version of docker-compose.

docker-compose push

Pushes images for services to their respective registry/repository

docker-compose config

Validate and view the Compose file.

By **Gaurav Pandey**
(gauravpandey44)

cheatography.com/gauravpandey44/

Published 21st September, 2019.
Last updated 23rd September, 2019.

Cheatography

Docker Compose Cheat Sheet

by Gaurav Pandey (gauravpandey44) via cheatography.com/69622/cs/20583/

docker-compose kill

Forces running containers to stop by sending a SIGKILL signal.

docker-compose bundle

A Dockerfile can be built into an image, and containers can be created from that image. Similarly, a docker-compose.yml can be built into a distributed application bundle

Building

```
web:
  # build from Dockerfile
  build: .
  # build from custom Dockerfile
  build:
    context: ./dir
    dockerfile: Dockerfile.dev
  # build from image
  image: ubuntu
  image: ubuntu:14.04
  image: tutum/influxdb
  image: example-registry:4000/postgresql
  image: a4bc65fd
```

Ports

```
ports:
    - "3000"
    - "8000:80" # guest:host
  # expose ports to linked services (not to host)
  expose: ["3000"]
```

Commands

```
# command to execute
  command: bundle exec thin -p 3000
  command: [bundle, exec, thin, -p, 3000]
  # override the entrypoint
  entrypoint: /app/start.sh
  entrypoint: [php, -d, vendor/bin/phpunit]
```

Environment variables

```
# environment vars
  environment:
    RACK_ENV: development
  environment:
    - RACK_ENV=development
  # environment vars from file
  env_file: .env
  env_file: [.env, .development.env]
```


By **Gaurav Pandey**
(gauravpandey44)

cheatography.com/gauravpandey44/

Published 21st September, 2019.
Last updated 23rd September, 2019.