Docker Compose, Machine, Swarm & CI

Ondrej Sika
ondrej@ondrejsika.com
@ondrejsika

https://sika.link/docker

Agenda

- Docker Compose
- Docker Machine
- Docker Swarm
- Docker in CI

Docker Compose

What is Docker Compose?

Compose is a tool for defining and running multi-container **Docker** applications.

With **Compose**, you use a **Compose file** to configure your application's services.

Install Docker Compose

https://docs.docker.com/compose/install/

https://docs.docker.com/compose/completion/

Compose File

A **docker-compose.yml** file is a YAML file that defines how Docker containers should behave in production.

Example Compose File

```yaml
version: '3.6'
services:
  web:
    build: .
    ports:
     - 8000:80
  redis:
    image: redis
```

https://docs.docker.com/compose/compose-file/

Service

Service is a container running and managed by Docker Compose.

Build

```
services:
  app:
    build: .

services:
  app:
    build:
      context: ./app
      dockerfile: Dockerfile-prod
    image: myapp
```

Image

```
services:
  app:
    image: redis:alpine
```

Port forwarding

```
services:
  app:
    ports:
      - 8000:80
```

Volumes

```
services:
  app:
    volumes:
      - /data1
      - data:/data2
      - ./data:/data3
volumes:
  data:
```

Environment variables

```
services:
  app:
    environment:
      RACK_ENV: development
      SHOW: 'true'
      SESSION_SECRET:

services:
  app:
    environment:
      - RACK_ENV=development
```

Command

```
services:
  app:
    command: ["python", "app.py"]
```

Deploy

```
services:
  app:
    deploy:
      placement:
        constraints: [node.role == manager]

services:
  app:
    deploy:
      mode: replicated
      replicas: 4
```

Create a Composite

app.py

```
import os
from flask import Flask
from redis import Redis

app = Flask(__name__)
redis = Redis(os.environ.get('REDIS', 'redis'))
hostname = os.environ['HOSTNAME']

@app.route("/")
def index():
    counter = redis.incr('counter')
    return "%s %d" % (hostname, counter)

if __name__ == "__main__":
    app.run(host='0.0.0.0', port='80')
```

requirements.txt

```
flask
redis
```

Dockerfile

```
FROM python:3.7-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install -r requirements.txt
COPY . .
CMD [ "python", "app.py" ]
```

Create docker-compose.yml

```
version: '3.6'
services:
    app:
       build: .
       image: reg.istry.cz/ondrej/app
       ports:
           - 8000:80
    redis:
       image: redis
```

Compose Commands

Basic Compose Commands

```
docker-compose config
docker-compose help
docker-compose ps
docker-compose exec <service> <command>
docker-compose version
docker-compose logs [-f] [<service>]
```

Build Compose

```
docker-compose build

docker-compose build --no-cache
```

Run Compose

```
docker-compose up

docker-compose up -d

# Missing images will be downloaded or
builded
```

Compose Up Arguments

-d - run in detached mode
--force-recreate - always create new cont.
--build - build on every run
--no-build - don't build, even images not exist
--remove-orphans
--abort-on-container-exit

Manage Compose

```
docker-compose start [<service>]
docker-compose stop [<service>]
docker-compose restart [<service>]
docker-compose kill [<service>]
```

Remove Compose

```
docker-compose down

# stop and remove compose
```

Scaling Compose

```
docker-compose up --scale <service>=<n>
```

Docker Machine

What is Docker Machine?

Docker Machine is a tool that lets you install **Docker Engine** on virtual hosts, and manage the hosts with docker-machine commands.

You can use **Machine** to create **Docker** hosts on your local Mac or Windows box, on your company network, in your data center, or on cloud providers like AWS or Digital Ocean.

Install Docker Machine

https://docs.docker.com/machine/install-machine/

Basic Machine Command

```
docker-machine ls

docker-machine version
```

Create a Machine

```
docker-machine create [-d <driver>] <machine>

# Eg.:

docker-machine create default
docker-machine create --driver digitalocean ci



# List of drivers
https://docs.docker.com/machine/drivers/
```

Inspect a Machine

```
docker-machine inspect <machine>
docker-machine ip <machine>

# Eg.:

docker-machine inspect default
docker-machine inspect

docker-machine ip default
docker-machine ip
```

Connect Shell to the Machine

```
eval "$(docker-machine env <machine>)"

# Eg.:

eval "$(docker-machine env default)"
eval "$(docker-machine env)"
```

SSH to the Machine

```
docker-machine ssh <machine>

# Eg.:

docker-machine ssh default
docker-machine ssh
```

Manage a Machine

```
docker-machine start <machine>
docker-machine stop <machine>
docker-machine restart <machine>
docker-machine kill <machine>
```

Remove a Machine

```
docker-machine rm <machine>

# Eg.:

docker-machine rm default
docker-machine rm
```

Docker Swarm

What is Docker Swarm?

A native clustering system for **Docker**. It turns a pool of **Docker** hosts into a single, virtual host using an API proxy system. It is **Docker's** first container orchestration project that began in 2014. Combined with **Docker Compose**, it's a very convenient tool to manage containers.

Create a Swarm

Initialize Swarm

```
docker swarm init --advertise-addr <manager_ip>

# Eg.:

docker swarm init --advertise-addr 192.168.99.100
```

Add Worker to Swarm

```
docker swarm join --token <token> <manager_ip>:2377

# Eg.:

docker swarm join \
    --token SWMTKN-1-49nj1cmql0...acrr2e7c \
    192.168.99.100:2377
```

Manage Swarm

Manage Swarm Nodes

docker node ls - list nodes
docker node rm <node> - remove node from swarm
docker node inspect <node>
docker node ps [<node>]- list swarm task
docker node update ARGS <node>

Swarm - Single Container

Deploy a Service to the Swarm

```
docker service create [ARGS] <image> [<command>]

# Eg.:

docker service create --name ping debian ping oxs.cz
```

Manage Services

```
docker service ls
docker service inspect <service>
docker service ps <service>
docker service scale <service>=<n>
docker service rm <service>
```

Scale the Service

```
docker service scale <service>=<n>

# Eg.:

docker service scale ping=5
```

Docker Swarm - Composes

Build & Push

```
# Build

docker-compose build

# Push

docker-compose push
```

Deploy App to Swarm

```
# run

docker stack deploy \
    --compose-file docker-compose.yml \
    counter
```

Load Balancing

Test App

```
# on host run

curl `docker-machine ip manager`
curl `docker-machine ip manager`
curl `docker-machine ip worker1`
curl `docker-machine ip worker1`
curl `docker-machine ip worker2`
```

Manage Services

```
docker stack ls
docker stack services <stack>
docker stack ps <stack>
docker stack rm <stack>
```

Docker & CI

Install Gitlab Runner - Docker

```
docker-machine create ci-runner

docker-machine ssh ci-runner

docker run -d \
    --name gitlab-runner \
    --restart always \
    -v /var/run/docker.sock:/var/run/docker.sock \
    -v /builds:/builds \
    gitlab/gitlab-runner:latest
```

Register Gitlab Runner - Docker

```
docker exec -ti gitlab-runner gitlab-runner register \
   --non-interactive \
   --url $GITLAB_URL/ \
   --registration-token $GITLAB_CI_TOKEN \
   --name $(hostname) \
   --executor docker \
   --docker-image docker:git \
   --docker-volumes '/var/run/docker.sock:/var/run/docker.sock' \
   --docker-volumes '/builds:/builds'
```

Docker Environment

```
image: ondrejsika/ci

job1:
  script: make

job2:
  image: ondrejsika/ci-go
  script: make go
```

Docker

```
job:
  script:
    - 'docker login $CI_REGISTRY \
       -u $CI_REGISTRY_USER \
       -p $CI_REGISTRY_PASSWORD'
    - docker build -t $CI_REGISTRY_IMAGE .
    - docker push $CI_REGISTRY_IMAGE
```

Thank you & Questions

Ondrej Sika

email:	ondrej@ondrejsika.com
twitter:	@ondrejsika
linkedin:	/in/ondrejsika/

Slides:	https://sika.link/spel-docker
	https://github.com/ondrejsika/docker-training-examples

Did you enjoy the course?

Tweet about it!

@ondrejsika @rootcz