DevOps

Thomas Hofmann

2020-12-04

What is DevOps?

- DevOps combines software development (Dev) and IT operations (Ops).
- It aims to shorten the development life cycle and provide continuous delivery.
- DevOps complements agile software development.

https://en.wikipedia.org/wiki/DevOps

Desired Properties

- Reproduceable
  - versioned
  - explicit configuration
    - software
    - dependencies
- Reliable
- Automatic

personal point of view

Toolchains

1. Coding – development, review SCM, merge
2. <u>Building</u> – CI, build status
3. Testing – continuous testing
4. Packaging – artifact repository, pre-deploy
5. Releasing/Deploying
   – change management,
   – release approval/automation
6. <u>Configuring</u> – infrastructure config. & mgmt.
7. Monitoring – performance monitoring,
   – end-user experience

1. Coding

- Editor/IDE
  - vs-code
  - linter
    - ESLint
    - automatic code formatting (Prettier)
- SCM – Source Code Management
  - git
- Merging
  - meld

https://en.wikipedia.org/wiki/Lint_(software)
https://en.wikipedia.org/wiki/List_of_tools_for_static_code_analysis#JavaScript

2. Building

- CI – Continuous Integration
  - Jenkins
  - drone
  - gitea
  - Gitlab
  - Bitbucket
  - ...

https://www.jenkins.io/
https://drone.io
https://gitea.io
https://gitlab.com
https://bitbucket.org

3. Testing

- Unit Tests
  intra component
- Integration Tests
  inter component, testing interfaces
  & interactions between components
- System Tests
  testing the fully integrated system, – e. g.
  login, [interactions]*, logout
- (Acceptance Tests)
  user, operational, contractual/regulatory,
  alpha/beta testing

https://en.wikipedia.org/wiki/Software_testing

4. Packaging

- create artifacts
  - binaries/modules
  - installers (deb, rpm, msi)
  - images (container)
- store artifacts in pre-release-repository e. g.
  - Sonatype‘s „Nexus“
  - docker registry
- Database Migrations (https://flywaydb.org)

https://www.sonatype.com/nexus/repository-pro
https://hub.docker.com

5. Releasing/Deploying

- Change Management
- Release Approvals
- Release/Deployment Automation
  - store artifacts in release-repository
  - provision downloads (homepage, shops, ...)

https://en.wikipedia.org/wiki/Application-release_automation

6. Configuring

- Infrastructure (Infrastructure as Code)
  - Configuration
  - Management

Tools:

- Ansible
- Puppet
- Terraform
- ...

https://en.wikipedia.org/wiki/Infrastructure_as_code
https://www.ansible.com/
https://puppet.com/
https://www.terraform.io/

7. Monitoring

- Applications Performance Monitoring
- End-User Experience
- Tools:
  - syslog e. g. with ELK-stack: ElasticSearch, LogStash, Kibana
  - icinga/nagios
  - Prometheus, Grafana

https://www.elastic.co/kibana
https://icinga.com/
https://prometheus.io/

1. & 2. & 4.
Coding & Building & Packaging

Containers/ Docker

https://www.docker.com/
https://en.wikipedia.org/wiki/Docker_(software)

Containerisation
Isolating/Restricting a Running Program

- files
- devices
- networks
- memory
- CPU power
- [quantifiable resource]*

https://en.wikipedia.org/wiki/OS-level_virtualization

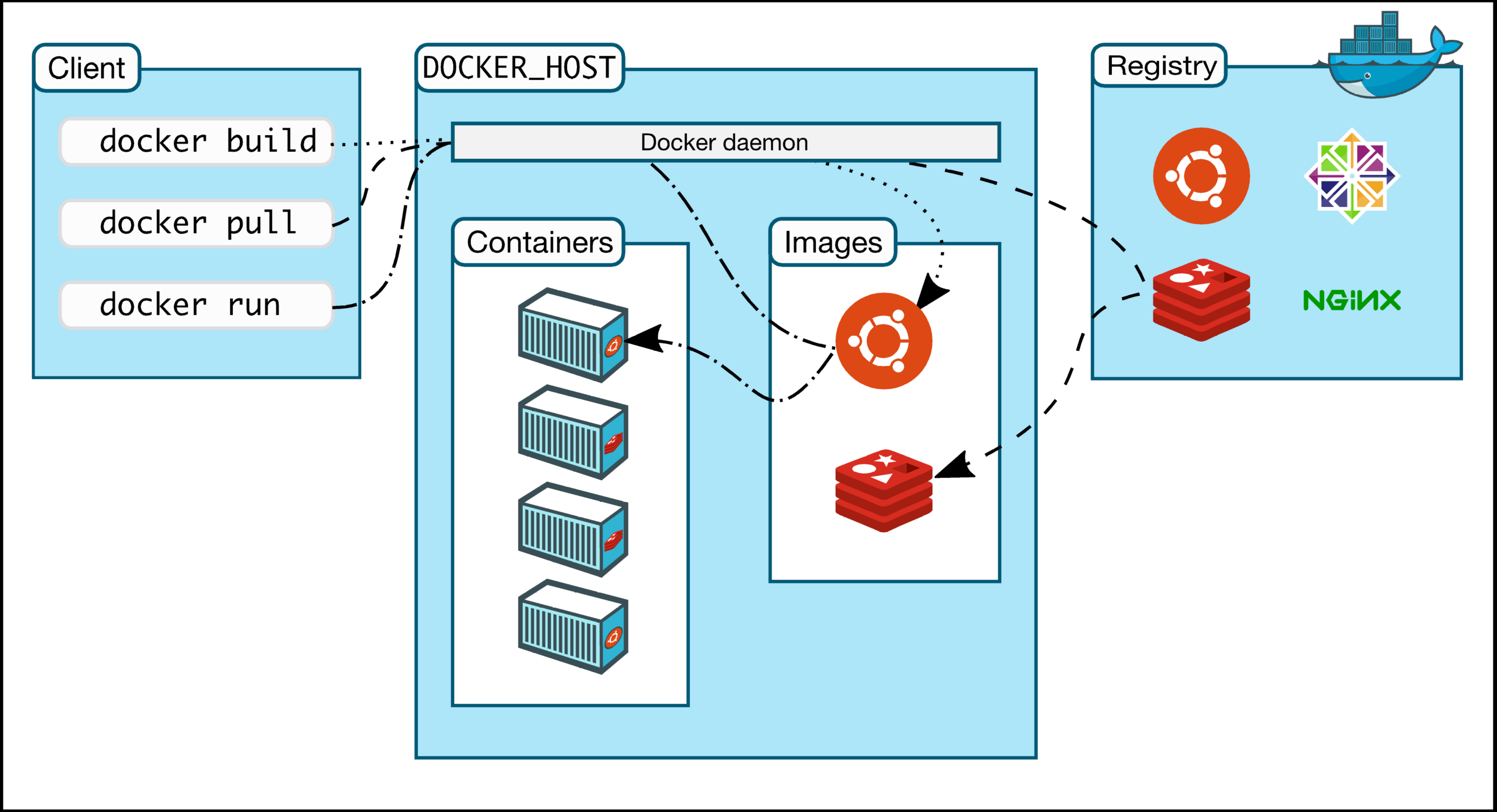
Containerisation

Explicit Definition of Dependencies

- internal
  - packages
  - files
- external
  - services
  - networks
  - ports

In addition reproducable states are gained – ease to freshly recreate a container

Docker – Architecture



from: https://docs.docker.com/get-started/overview/

Building an Image – Definition

Dockerfile

```
FROM node

ENV DEBIAN_FRONTEND=noninteractive

RUN apt-get update \
    && apt-get install -y \
        PACKAGE_A \
    && apt-get autoremove -y \
    && rm -rf /var/lib/apt/lists/*
```

Building an Image

.dockerignore

```
.dockerignore
.git
.gitignore
```

```
$> docker build -t IMAGE_NAME:latest .
```

Running – Container

docker run

```
docker run -it \
   --name plack.container \
   -v /home/tom/projects/plack/app:/app \
   -p 80:5000 \
   plack plackup /app/HelloWorld.psgi/app.psgi
```

Commands (selection)

```
#  --- Running Containers ---
$> docker start       # start container
$> docker stop        # stop container

#  --- Information ---
$> docker ps          # list containers
$> docker images      # list images
$> docker inspect     # details on docker objects
```

Commands (selection) ...continued

```
#  --- Remove Containers/Images ---
$> docker rm CONTAINER_ID/-_NAME   # remove container
$> docker rmi IMAGE_ID/-_NAME      # remove image

#  --- Create Images (specialties) ---
$> docker commit      #  make image from container
$> docker tag         # make tagged image
```

docker-compose

- allows to separate build- from runtime-configuration
- definition of multiple services and their dependencies in `docker-compose.yml`:

```
version: “3.8“
services:
   SERVICE_NAME_A:
      image: IMAGE_NAME
      ports: “OUTSIDE_PORT:CONTAINER_PORT”
      …
   SERVICE_NAME_B
```

docker-compose

In addition to using prebuilt images it is possible to define the built of custom images. For this a section ”build” can be added for the service in the `docker-compose.yml`:

```
services:
  SERVICE_NAME:
    image: IMAGE_NAME # if not existing it is built as follows
    build:
      context: ./http/build  # the context for the built
      dockerfile: Dockerfile.apache-php # optional; default: Dockerfile
```

There exists an associated command:

```
$> docker-compose build
```

docker-compose – basic commands

docker-compose COMMAND

COMMANDs:

- up – creates and starts containers for each service
- down – stops services and removes resources
- start – starts services (resources must exist)
- stop – stops services (without removing resources)
- ps – provides an overview of the services

5. & 6.
Deploying & Configuring

Ansible

Requirements

Ansible Server

- python-minimal
- ansible
- ssh

Target Hosts

- python-minimal
- ssh-server

Ansible – Terminology

- Inventory
  - Host-/Group-Vars
- Playbook
  - Tasks
  - Roles – https://docs.ansible.com/ansible/latest/user_guide/playbooks_reuse_roles.html
- Modules
- Vars
  - group_vars
  - host_vars
  - vault

Ansible – Commandline

```
#  Test Ansible Connection to Hosts
$> ansible HOST_DEFINITION -m ping

#  Secrets
$> ansible-vault create FILE
$> ansible-vault edit FILE
$> ansible-vault view FILE
```

Ansible – Commandline ...continued

```
#  Playbooks
$> ansible-playbook \
     -i inventory/ \
     --ask-pass \
     --ask-become-pass \
     --ask-vault-pass \
     --limit HOST_FILTER \
     PLAYBOOK
```

Ansible – Roles

In the following directories Ansible will automatically use files named „main.yml“, „main.yaml“, or „main“.

- playbooks/roles/
  - common/

– tasks/	– the main list of tasks that the role executes
– handlers/	– handlers, which may be used within or outside this role
– library/	– modules, which may be used within this role – see embedding modules and plugins in roles
– files/	– files that the role deploys
– templates/	– templates that the role deploys
– vars/	– other variables for the role (see Using Variables)
– defaults/	– default variables (lowest priority; see Using Variables)
– meta/	– metadata for the role, including role dependencies

  - ANOTHER_ROLE/

https://docs.ansible.com/ansible/latest/user_guide/playbooks_reuse_roles.html

Ansible – Resources

- Modules
  https://docs.ansible.com/ansible/2.3/modules_by_category.html
- Roles/Collections
  https://galaxy.ansible.com
- Testing Ansible
  Testing Ansible with Molecule and Podman