Applications Deployment

Damian Bodnar <damian.bodnar@rtbhouse.com>
RTB House

mimuw.rtbhouse.com

Applications Deployment


```
run ...
```

```
apply ...
```

Plan

Motivation

Containers internals

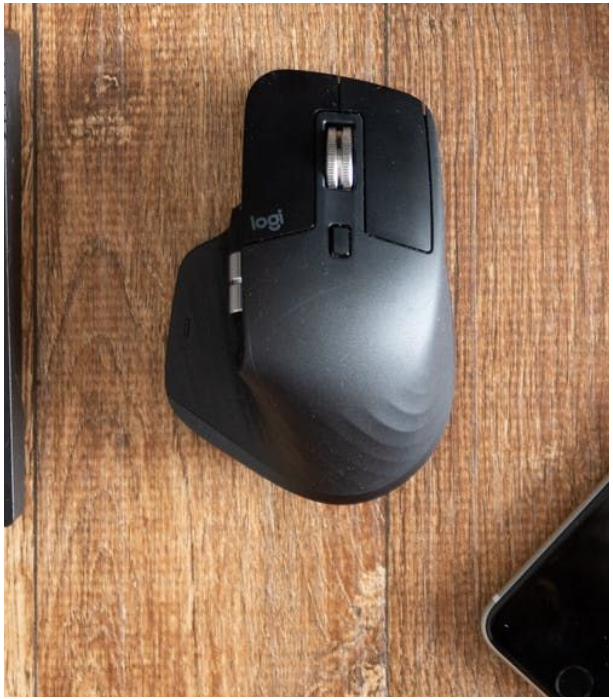
Deployment strategies

Versioning

Automation & orchestration

Toolings

Expectations



Downtime

mBank Polska
18 hrs ·

⚠️W najbliższy weekend - z 19 na 20 marca będziemy modernizować nasze systemy.
W związku z tym przez kilka godzin nie będziecie mieć dostępu do banku:

- między 2:00 a 8:30 nie zapłacicie kartą💳 w sklepach i nie skorzystacie z bankomatów,
- od 2:00 do 10:00 nie zrobicie zakupów online🌐 i nie zalogujecie się na swoje konto w serwisie transakcyjnym💻 ani w aplikacji mobilnej📱,
- w godzinach 1:30-11:00 nie będziecie mogli składać wniosków.

W czasie przerwy nie będzie możliwości logowania się na konto, a mLinia 📞 będzie działać w trybie informacyjnym.

Żeby przerwa nie pokrzyżowała Wam weekendowych planów, warto na ten czas zaopatrzyć się w gotówkę 💰 i zlecić z wyprzedzeniem ważne przelewy.

Sprawdźcie szczegóły na ➡️ www.mbank.pl/przerwa

Preparations

Environment

Application part

- dependencies
- configurations
- containers

Strategy of deployment

Automation


Machine
a piece of hardware or a VM

Server

software that provides a service

nginx

postgres

redis

envoy

memcached

kafka

Devs

Containerization

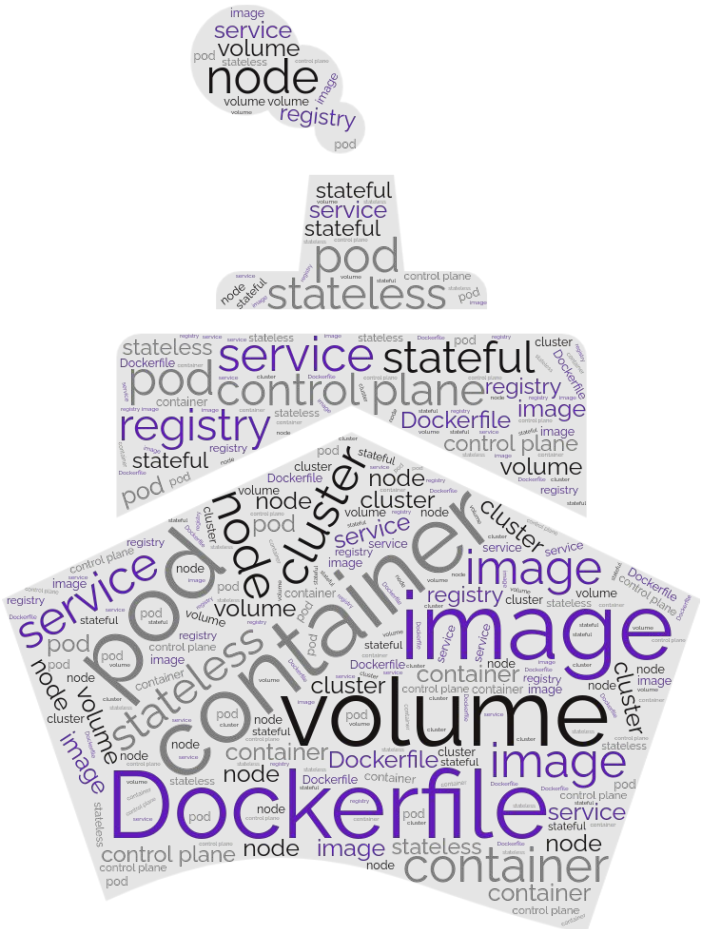
SysAdm

Image

Dockerfile

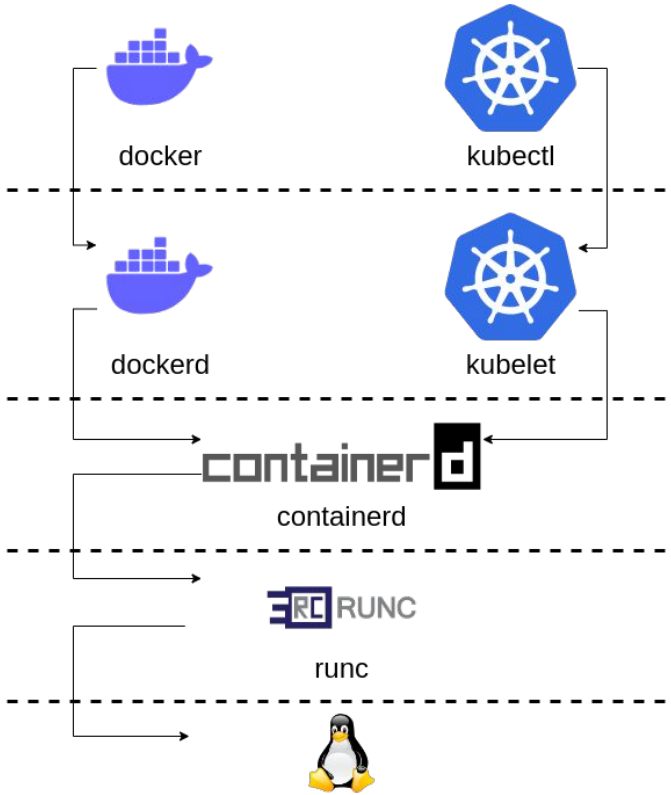

Image
ENSALADA DE "ATÚN"
EN CHIPOTLE
Chipotle "Tuna" Salad)
YIELD: 4 SERVINGS
For Serving
CHEF'S NOTES:
40 LA VIDA VERDE


Container



… more containers vocabulary

Single container spawn



Linux stuff behind containers

```
1    systemd  (prev: init)
523  vim

823  VBoxSVC
833  \ VirtualBoxVM

715  containerd-shim-runc-v2
719  \_ postgres
720       \_ postgres
721       \_ postgres
723       \_ postgres

351  containerd-shim-runc-v2
352  \_ redis-server
```

```
user@host:~$
user@host:~$
user@host:~$
user@host:~$
user@host:~$ docker pull mcr.microsoft.com/\
> windows/server:ltsc2022-amd64
```

```
user@host:~$
user@host:~$
user@host:~$
user@host:~$
user@host:~$ docker pull mcr.microsoft.com/\
> windows/server:ltsc2022-amd64
ltsc2022-amd64: Pulling from windows/server
6d889b139513: Pulling fs layer
60fff5ce9fed: Downloading  538.4kB/1.995GB
image operating system "windows" cannot be
used on this platform
user@host:~$
```

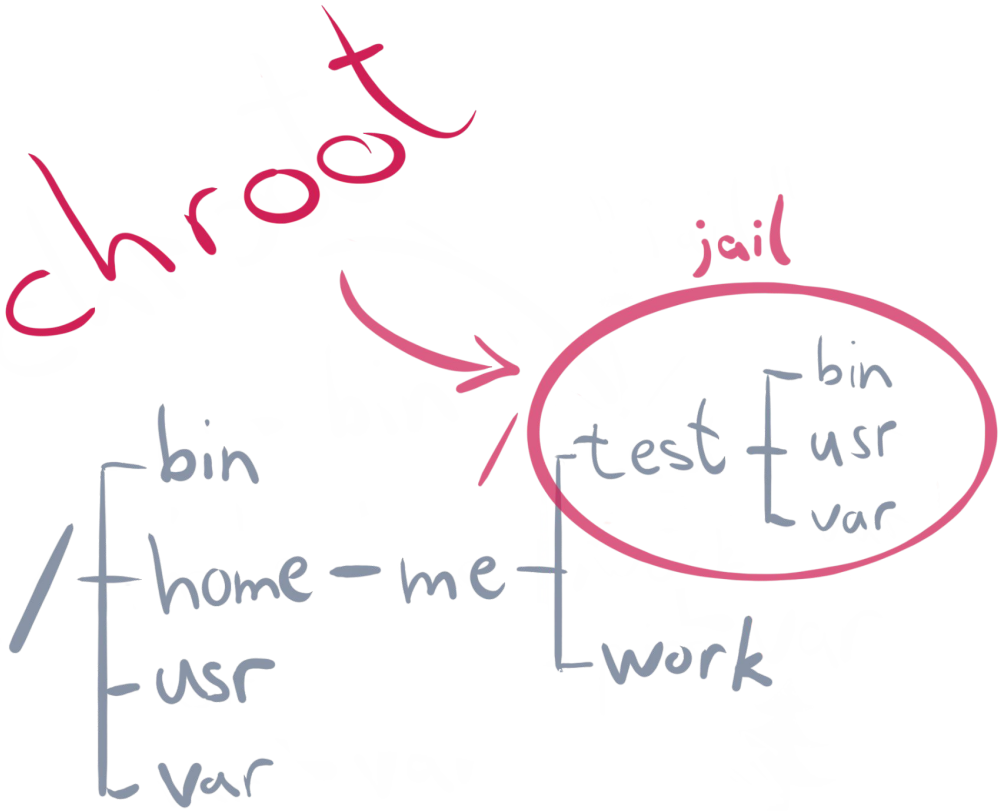
Containers

namespaces

cgroups

(capabilities)

chroot



src: https://www.cncf.io/blog/2019/06/24/demystifying-containers-part-i-kernel-space/

Namespaces

```
1    systemd  (prev: init)
523  vim

823  VBoxSVC
833  \ VirtualBoxVM

715  containerd-shim-runc-v2
719  \_ postgres
720       \_ postgres
721       \_ postgres
723       \_ postgres

351  containerd-shim-runc-v2
352  \_ redis-server
```

```
1    postgres
49   \_ postgres
50   \_ postgres
51   \_ postgres
```

```
1    redis-server
```

Namespaces

- mount [the first one introduced in 2002]
- cgroup
- ipc
- network
- pid
- uts (hostname)
- user [2013, allowed to make kernelspace-containers]
- time [most recent one, delivered in 2020]

unshare --help

```
-m, --mount[=<file>]      unshare mounts namespace
-u, --uts[=<file>]        unshare UTS namespace (hostname etc)
-i, --ipc[=<file>]        unshare System V IPC namespace
-n, --net[=<file>]        unshare network namespace
-p, --pid[=<file>]        unshare pid namespace
-U, --user[=<file>]       unshare user namespace
-C, --cgroup[=<file>]     unshare cgroup namespace
-T, --time[=<file>]       unshare time namespace

-f, --fork                fork before launching <program>
--map-user=<uid>|<name>   map current user to uid (implies --user)
--map-group=<gid>|<name>  map current group to gid (implies --user)
-r, --map-root-user       map current user to root (implies --user)
-c, --map-current-user    map current user to itself (implies --user)
```

Cgroups

monitor

limit

Resources

```
$ docker stats --no-stream

CONTAINER ID   CPU %     MEM USAGE / LIMIT     MEM %     NET I/O           BLOCK I/O         PIDS
8146f077d595   0.08%     1.502GiB / 31.09GiB   4.83%     145kB / 8.43MB    39.3MB / 132MB    96
7e9ba2b3294f   0.04%     986.7MiB / 31.09GiB   3.10%     1.5MB / 1.71MB    40.5MB / 277MB    10
fbcc1b97a31e   4.28%     377.7MiB / 31.09GiB   1.19%     188kB / 1.46MB    25MB / 169MB      5
c96a75487a8d   10.05%    2.183GiB / 31.09GiB   7.02%     106kB / 10.7MB    47.8MB / 623kB    34
2e21a1888ae0   0.00%     2.012MiB / 31.09GiB   0.01%     29.8kB / 0B       0B / 0B           6
5e7719e3faf6   0.13%     2.684MiB / 31.09GiB   0.01%     1.65MB / 1.12MB   3.86MB / 16.4kB   4
0c0efa52440e   0.00%     77.78MiB / 31.09GiB   0.24%     6.65MB / 6.82MB   528kB / 350MB     20
649ba7510e1b   0.05%     11.34MiB / 31.09GiB   0.04%     175kB / 126B      0B / 0B           13
```

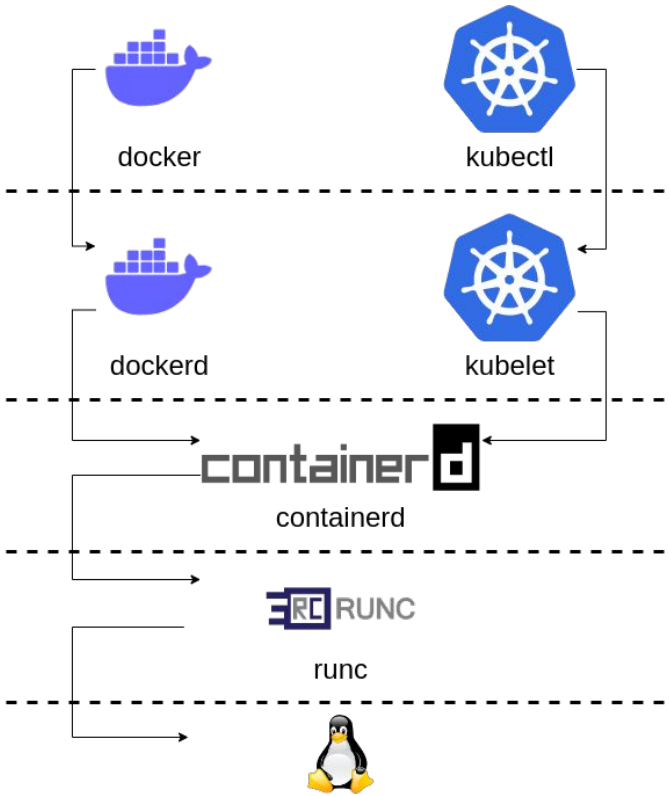
Single container spawn

Top-level API (docker, kubectl)

Rich on-host daemon (dockerd, kubelet)

Containers runtime daemon (containerd)

Low-level containers runtime (runc)

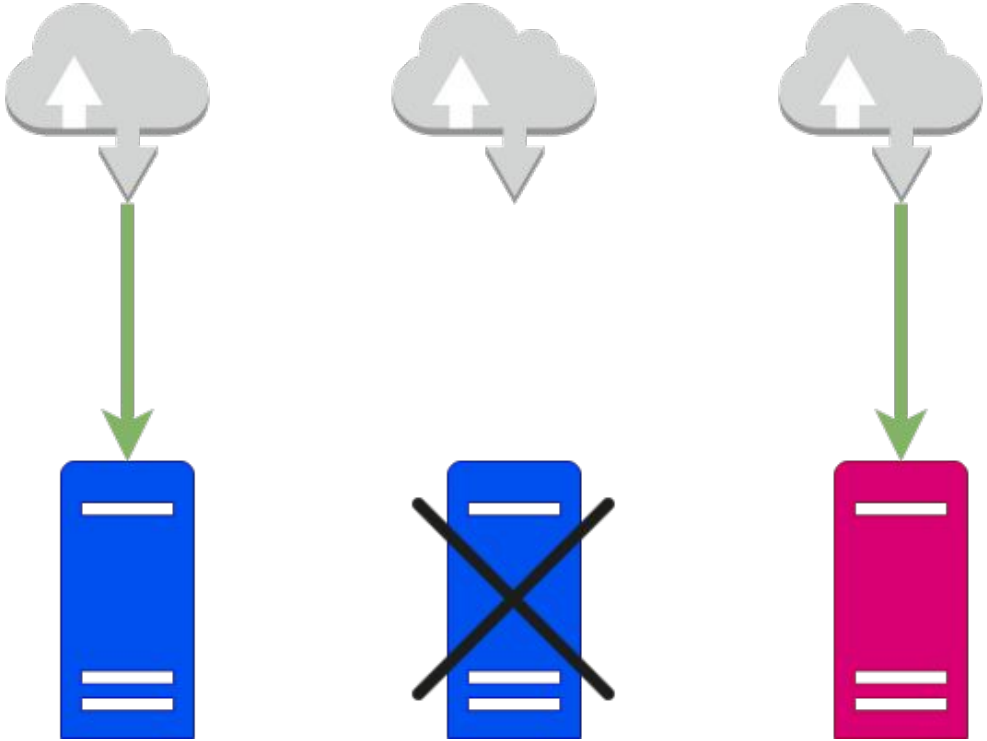


It works for me

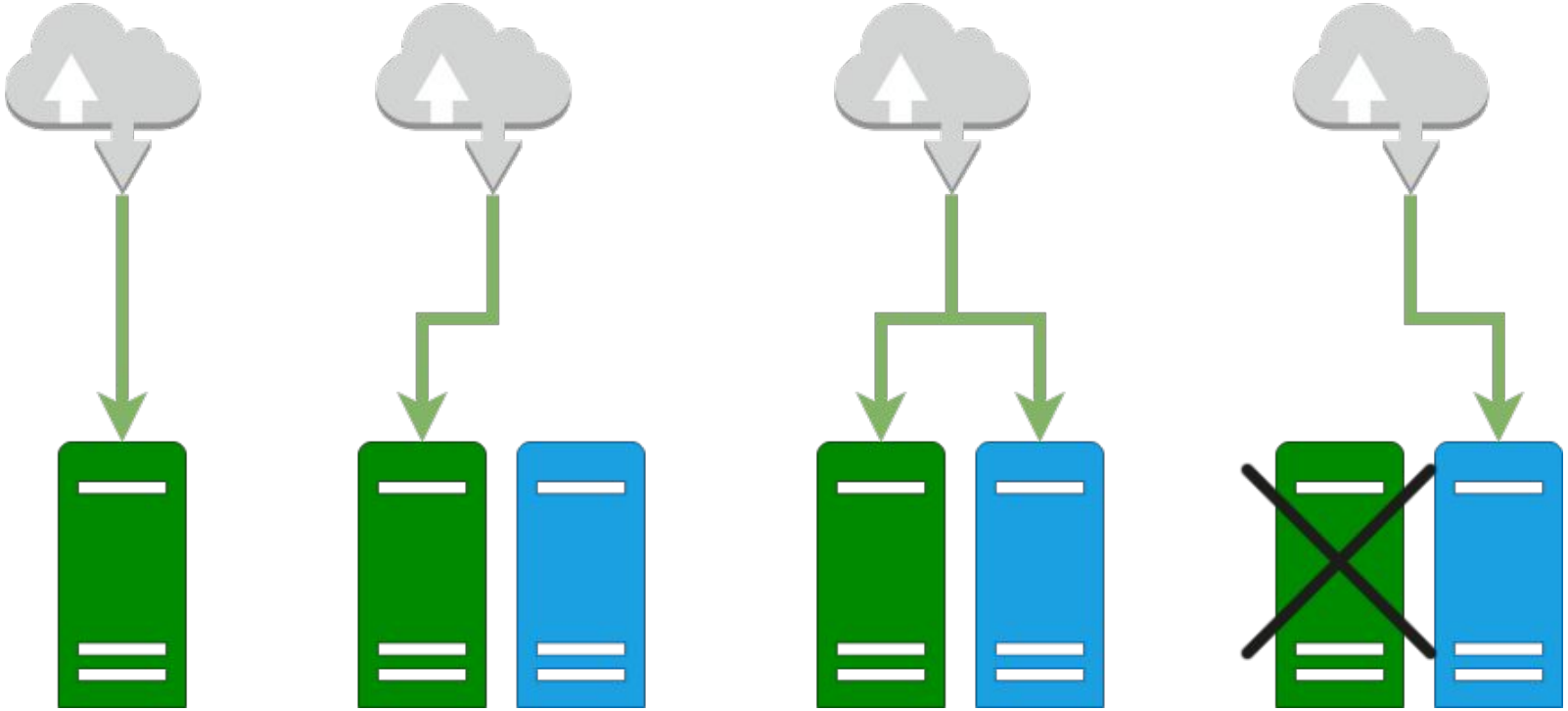
Containers to build, to test and to deploy

Deployment strategies

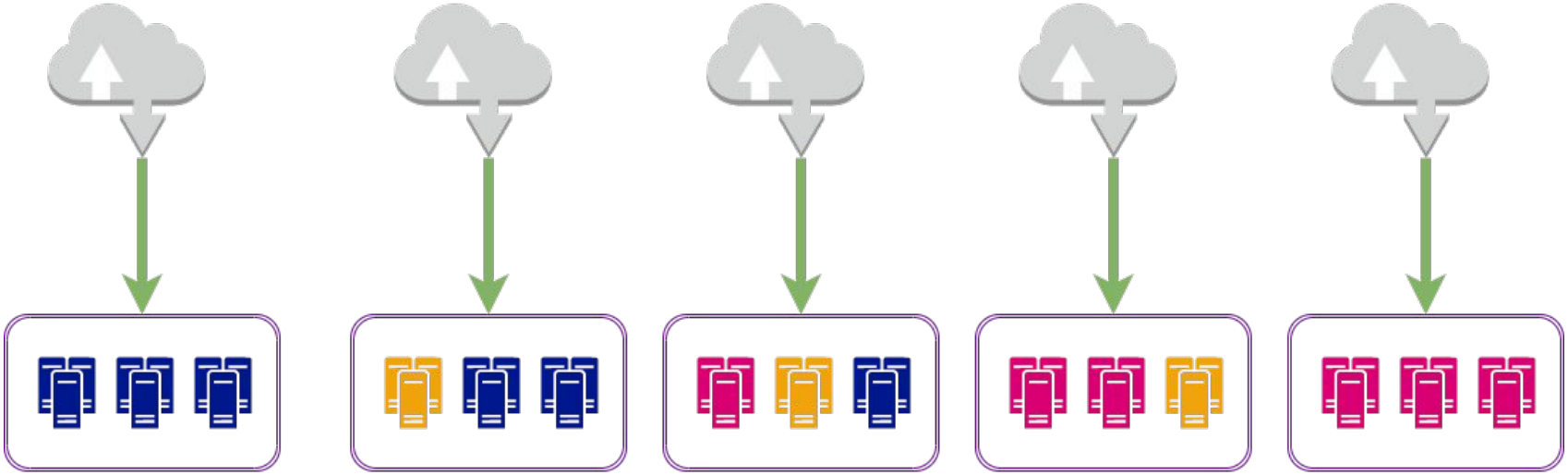
Take-down



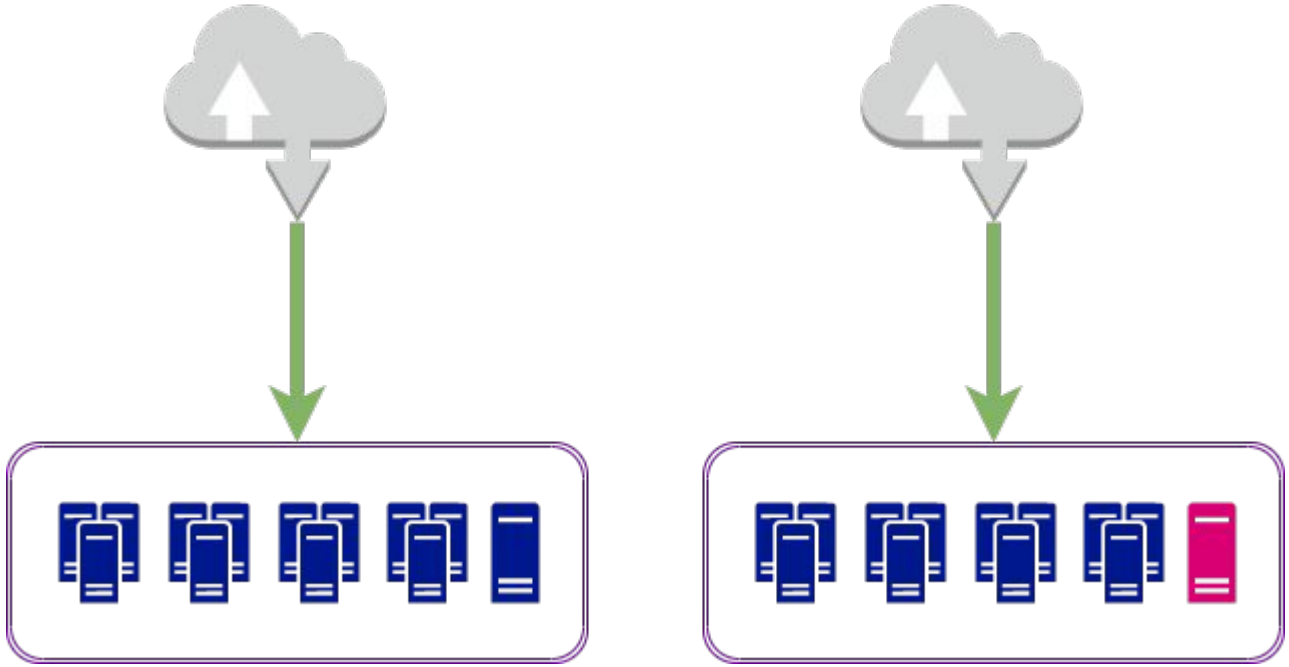
Blue/green



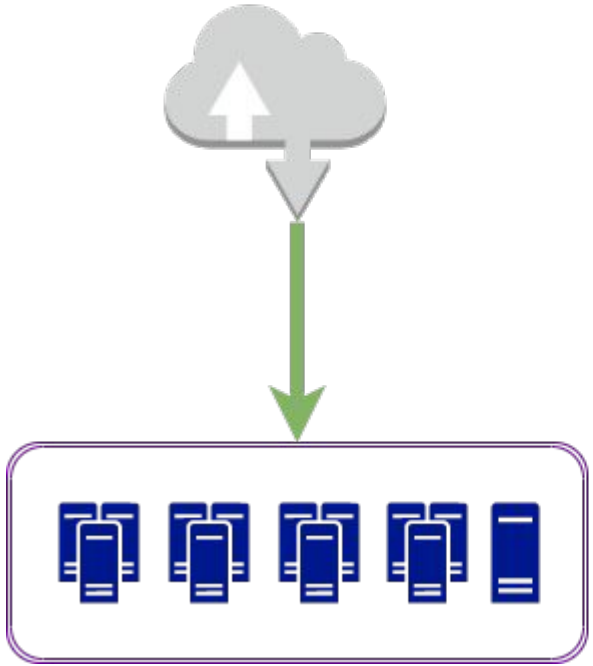
Rolling



Canary



Microservices deployment



Versioning

4.26.53

1.734.34-ef23c9

2021.11.351

Versioning

- (obviously) sources
- network configurations
- database schemas
- credentials
- OS configuration
- filesystem content
- … something else?

Automation & orchestration

Infrastructure as code

Approach

- declarative
- imperative

Method

- push
- pull

Infrastructure as code

Terraform

validate

plan

apply

*Ops

Continuous

Integration

Delivery

Deployment

Configuration Automation

Continuous Integration

CD /Continuous Delivery
/Continuous Deployment

Continuous Configuration Automation

Dockerfiles for build, test, deploy

```
FROM node

WORKDIR /usr/src/app

COPY . /usr/src/app

RUN npm install

RUN npm run bundle
```

```
FROM nginx

RUN rm /etc/nginx/conf.d/default.conf

COPY content /usr/share/nginx/html

COPY conf /etc/nginx
```

Multistage Dockerfile

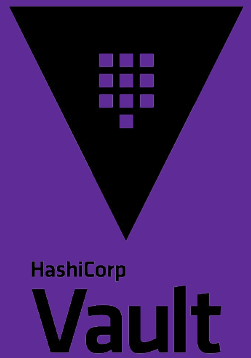
```
FROM node as build

WORKDIR /usr/src/app

COPY . /usr/src/app

RUN npm install

RUN npm run bundle
…
```

```
…
FROM nginx

RUN rm /etc/nginx/conf.d/default.conf

COPY --from=build content
/usr/share/nginx/html

COPY conf /etc/nginx
```

Layers

```
FROM python:3.10-alpine

COPY src /code

WORKDIR /code

RUN apk add bash

RUN pip install -r /code/requirements.txt

RUN python3 py_compile *.py

CMD python3 app.py
```

Secrets management



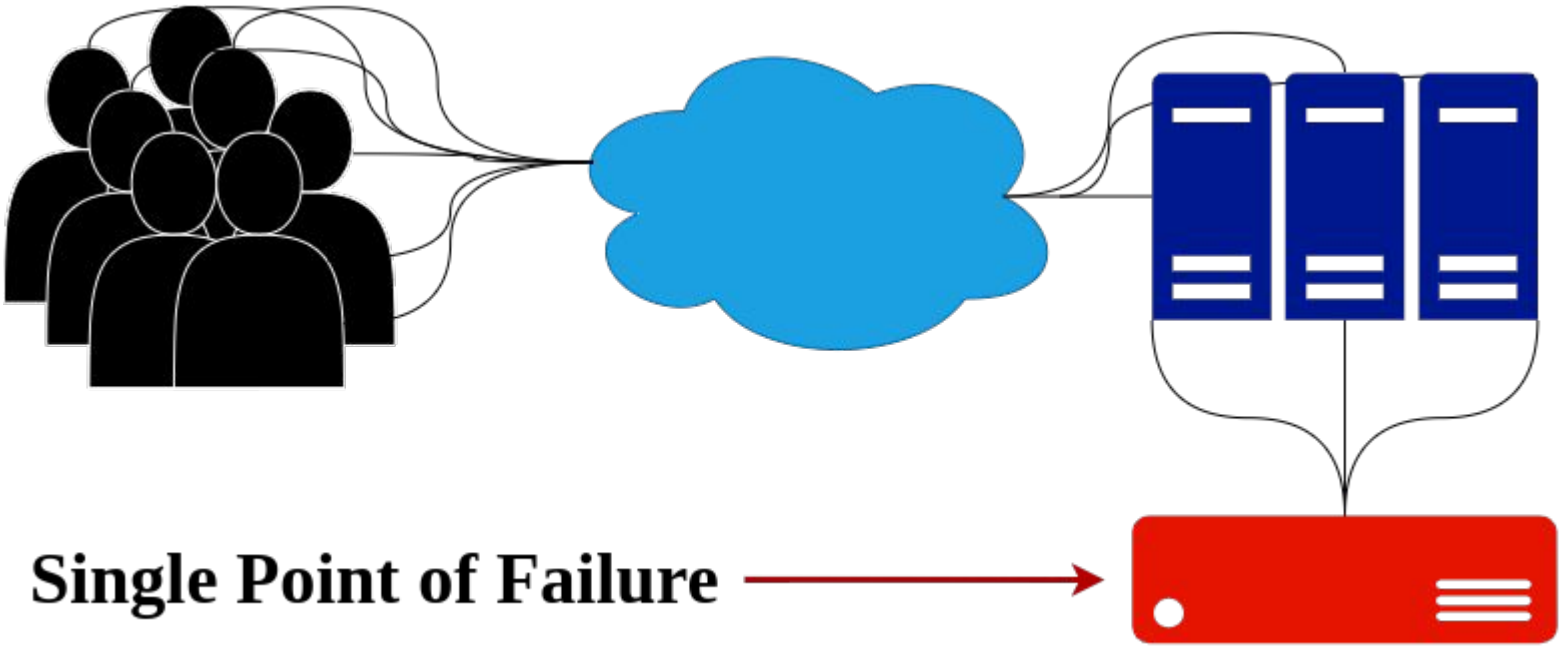
Jenkins

src: https://qautomation.blog/2019/05/28/jenkins-pipeline/

Docker Swarm

Kubernetes

SPOF



Intro to laboratory

```
@app.route('/run/<string:task_name>',
methods=['POST'])
def run_attendee_program(task_name: str):
    assert request.content_type == 'text/plain'
    code = request.data
    validate_attendee_code(task_name, code)
    return {
        'exit_code': 0,
        'results': [0] * 10
    }
```