

Kubernetes i Docker [LIVE DEMO]

Marcin Makowski

About me

Marcin Makowski

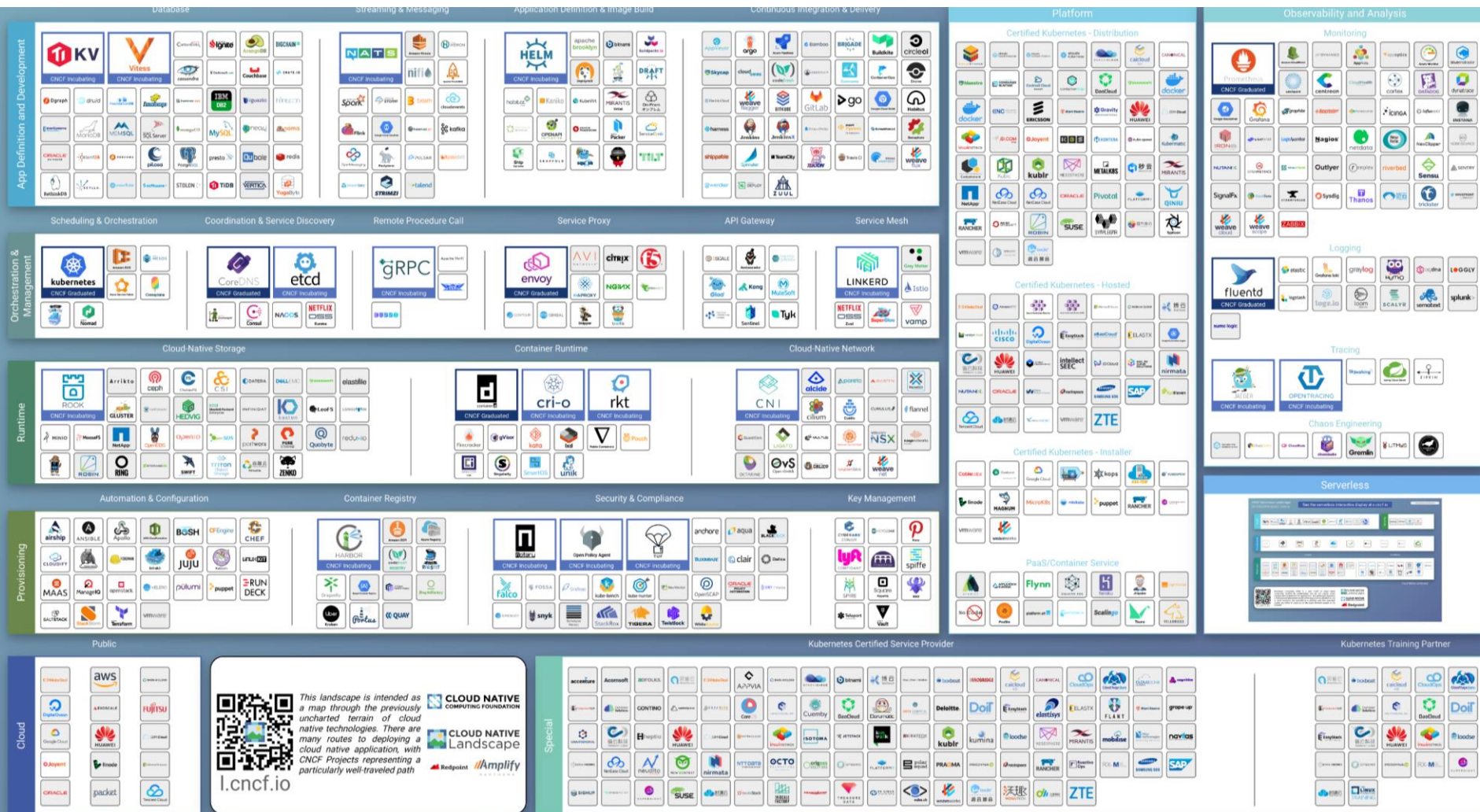
Cloud Architect & Senior
Software Developer w Powel



Why docker?



- Different libraries version
- Different OS versions
- Reproduction of behavior – DEV != STAGE != PROD
- Coexistence of software & applications (backward/forward compatibility)
- No need for new VM each time new software is installed
- Great for SAAS, PAAS solutions
- Run Tests same way against DEV, STAGE, PROD



Docker

OCI – Standard

UnionFS

CGroups

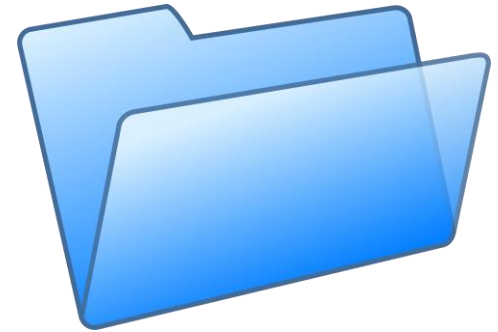
PID namespace for process isolation.

NET namespace for managing network interfaces.

IPC namespace for managing access to IPC resources.

MNT namespace for managing filesystem mount points.

UTS namespace for isolating kernel and version identifiers.



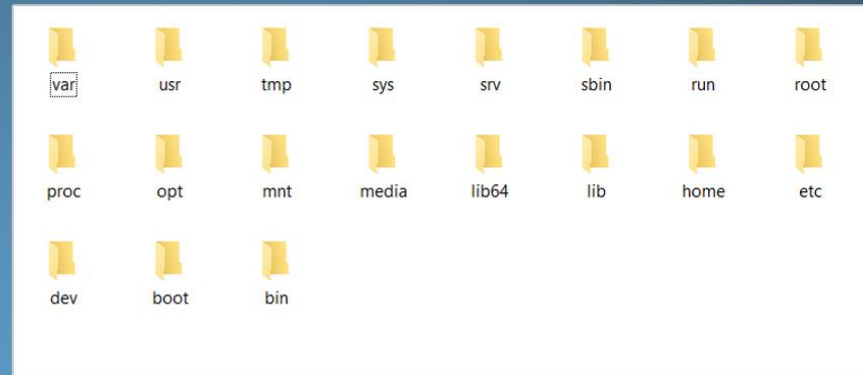
Container, Image, Dockerfile

```
FROM gcc:9.3.0
```

```
COPY main.cpp main.cpp
```

```
RUN c++ -static main.cpp -o main
```

```
CMD ["./main"]
```



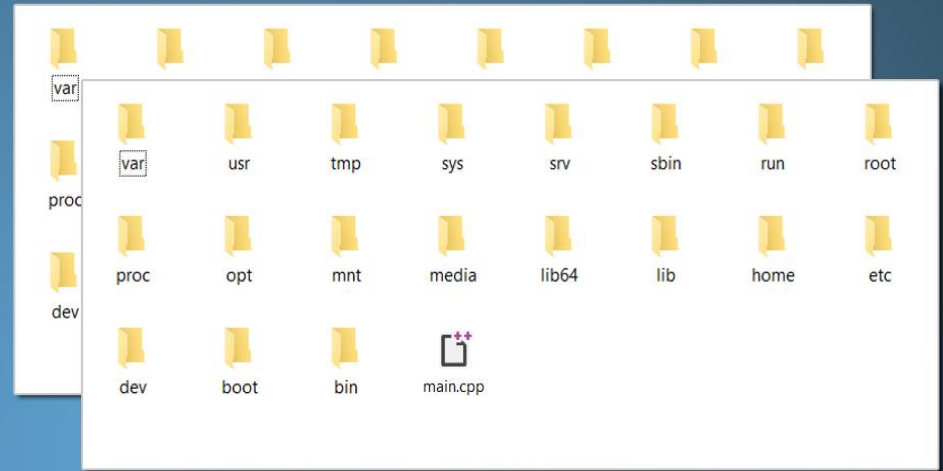
Container, Image, Dockerfile

```
FROM gcc:9.3.0
```

```
COPY main.cpp main.cpp
```

```
RUN c++ -static main.cpp -o main
```

```
CMD ["/main"]
```



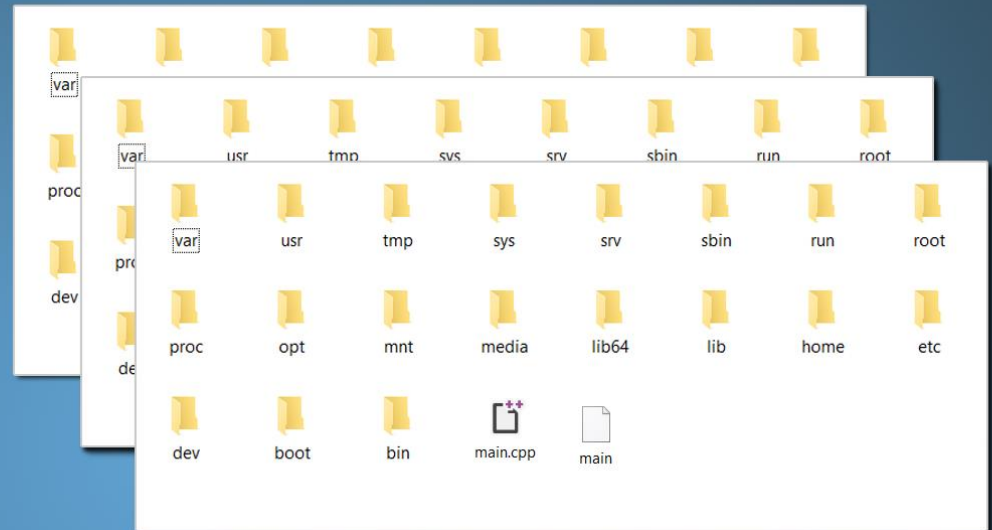
Container, Image, Dockerfile

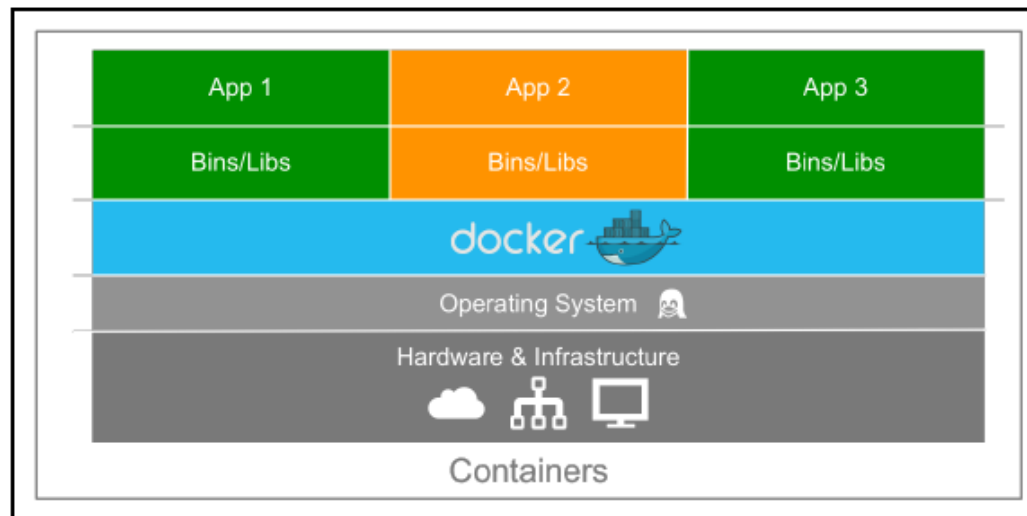
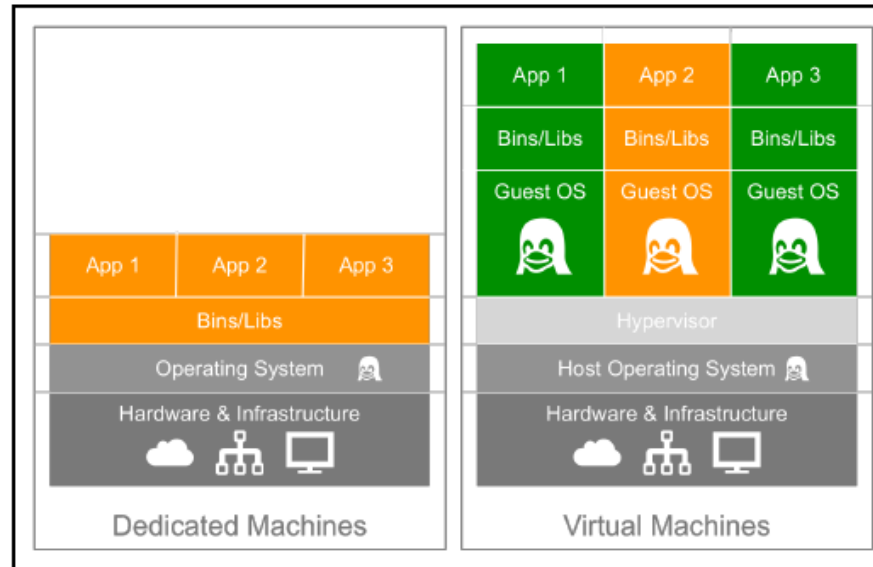
```
FROM gcc:9.3.0

COPY main.cpp main.cpp

RUN c++ -static main.cpp -o main

CMD ["./main"]
```





12th factor app for SaaS applications

1 application 1 codebase many Deploys

Many codebases means many applications

Common codebase means lack of dependency manager

Deploy from one codebase to each env (dev/stage/prod)

12th factor app for SaaS applications

Explicitly declare and isolate dependencies

Each application declares all dependencies

Application is not dependent on „not listed” dependencies

Application is not dependent on system tools (like bash/ps/curl)

12th factor app for SaaS applications

Store config in the environment

Can I share my app as open source right away without compromising information?

Can you run my app on your environment using just code?

Use ENV variables to set configuration

12th factor app for SaaS applications

Treat backing services as attached resources

All services are attached, there is no 'local' and 'remote' resources

Service can be easily attached and detached

12th factor app for SaaS applications

Strictly separate build and run stages

Build code (merge all dependencies)

Prepare release with version (use build and config)

Run release – should be possible without supervision

12th factor app for SaaS applications

Execute the app as one or more stateless processes

Any data that needs to persist should be attached through resource

Application can't be sure that information on disk or in memory would be available

Move „sticky sessions” to REDIS or similar tool

12th factor app for SaaS applications

Export services via port binding

Attach PORT to each possible application

If application doesn't use PORT then it is not service (job/run proces)

12th factor app for SaaS applications

Scale out via the process model

Consider that application will be scaled horizontally by increasing number of processes

Don't start process in background

12th factor app for SaaS applications

Maximize robustness with fast startup and graceful shutdown

processes are disposable – they can be stopped at any time

Handle SIGTERM gracefully

Consider „power off” situations

12th factor app for SaaS applications

**Keep development, staging, and production
as similar as possible**

developers who wrote code are closely involved in
deploying it and watching its behavior in production.

**The twelve-factor developer resists the urge to use different
backing services between development and production**

12th factor app for SaS applications

Treat logs as event streams

A twelve-factor app never concerns itself with routing or storage of its output stream. It should not attempt to write to or manage logfiles.

12th factor app for SaaS applications

Run admin/management tasks as one-off processes

One-off admin processes should be run in an identical environment as the regular long-running processes of the app. They run against a release, using the same codebase and config as any process run against that release.

Trade-offs

Knowledge

DevOps

CI / CD

Debug

Development overhead

HA

Scalability

0 downtime deploy

Infrastructure as a code

OS independence

Docker CLI

> docker help

> docker <COMMAND> --help

> docker container run hello-world

Dockerfile

FROM

LABEL

RUN

COPY and ADD

EXPOSE

ENV

ARG

ENTRYPOINT and CMD

VOLUME

	No ENTRYPOINT	ENTRYPOINT exec_entry p1_entry	ENTRYPOINT ["exec_entry", "p1_entry"]
No CMD	<i>error, not allowed</i>	/bin/sh -c exec_entry p1_entry	exec_entry p1_entry
CMD ["exec_cmd", "p1_cmd"]	exec_cmd p1_cmd	/bin/sh -c exec_entry p1_entry	exec_entry p1_entry exec_cmd p1_cmd
CMD ["p1_cmd", "p2_cmd"]	p1_cmd p2_cmd	/bin/sh -c exec_entry p1_entry	exec_entry p1_entry p1_cmd p2_cmd
CMD exec_cmd p1_cmd	/bin/sh -c exec_cmd p1_cmd	/bin/sh -c exec_entry p1_entry	exec_entry p1_entry /bin/sh -c exec_cmd p1_cmd

Docker container

run

attach

exec -ti nginx [/bin/bash | /bin/sh]

stop

start

restart

rm

Docker container

PAUSE == SIGSTOP

STOP == SIGTERM + SIGKILL

KILL == SIGKILL

Orchestrator concepts

Replicas

Ingress

Healthchecks

Pods / Services

Configs

Secrets

Volumes

Rolling updates

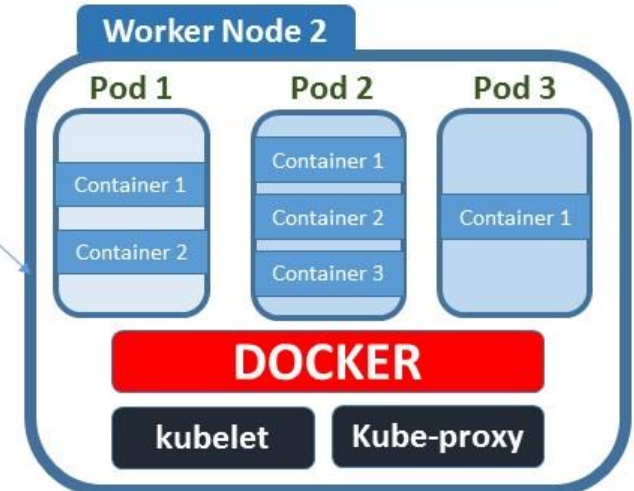
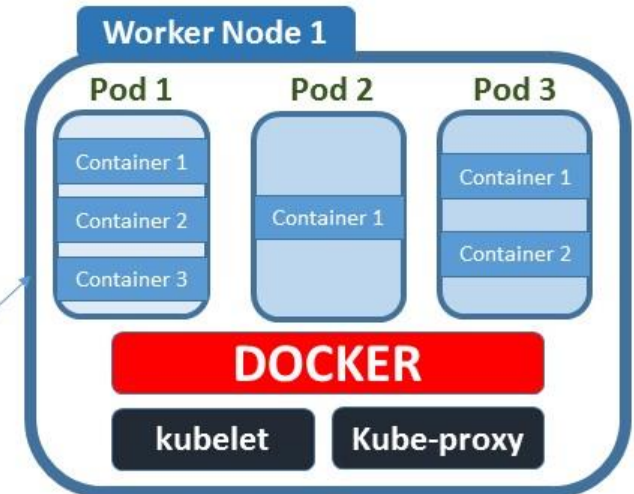
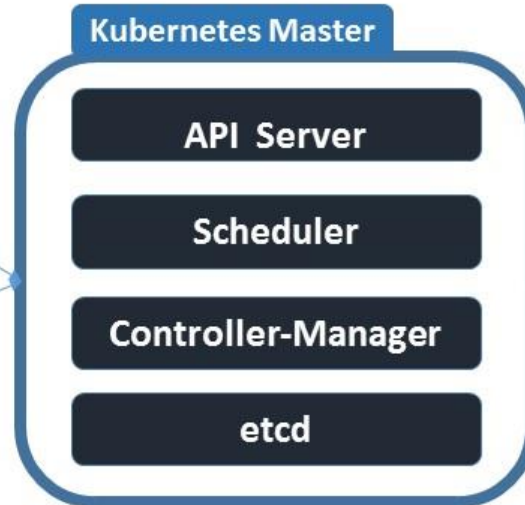
www.learnitguide.net

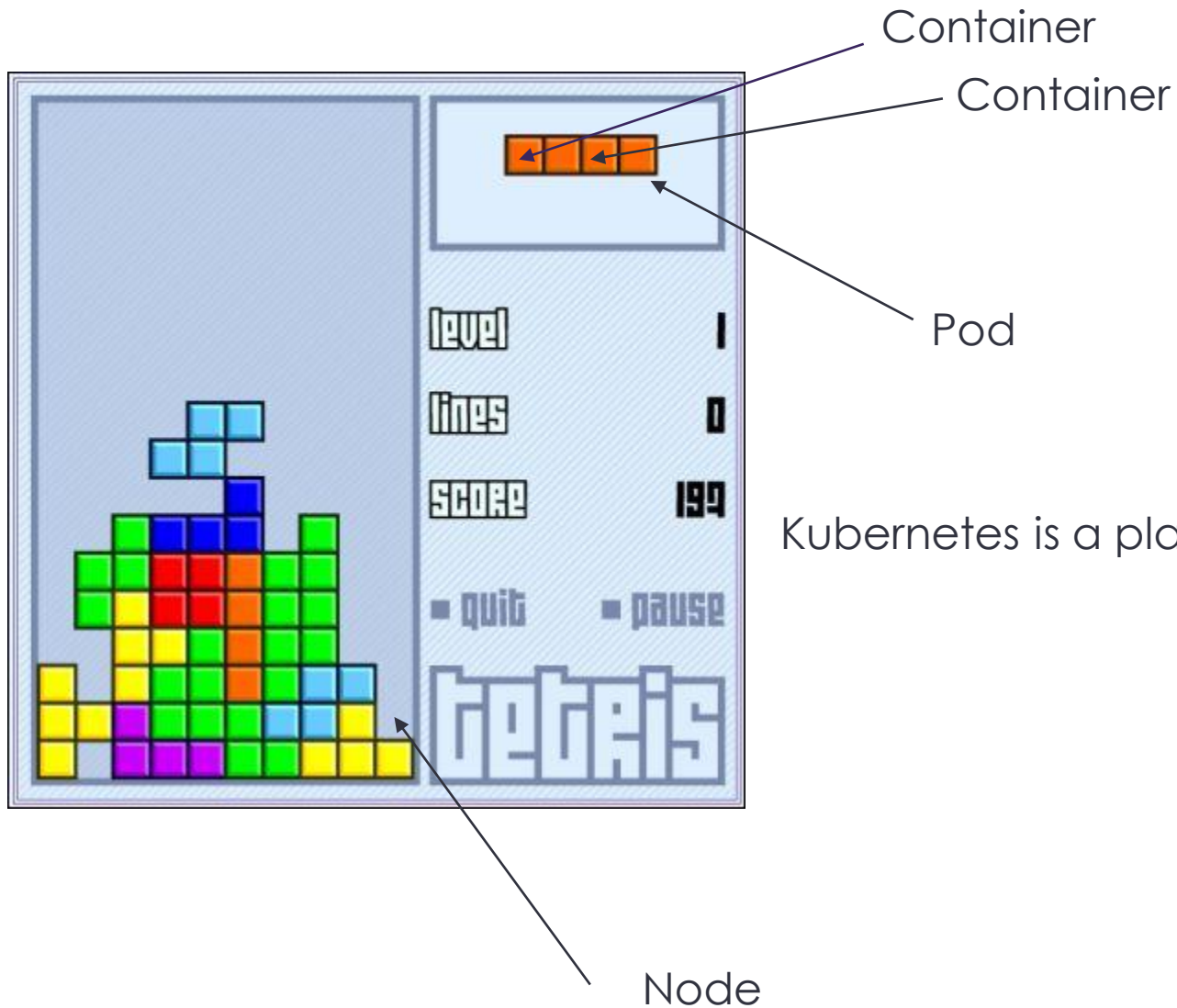
KUBERNETES ARCHITECTURE

User Interface



kubectl





Kubernetes is a player trying to win!

Kubectl

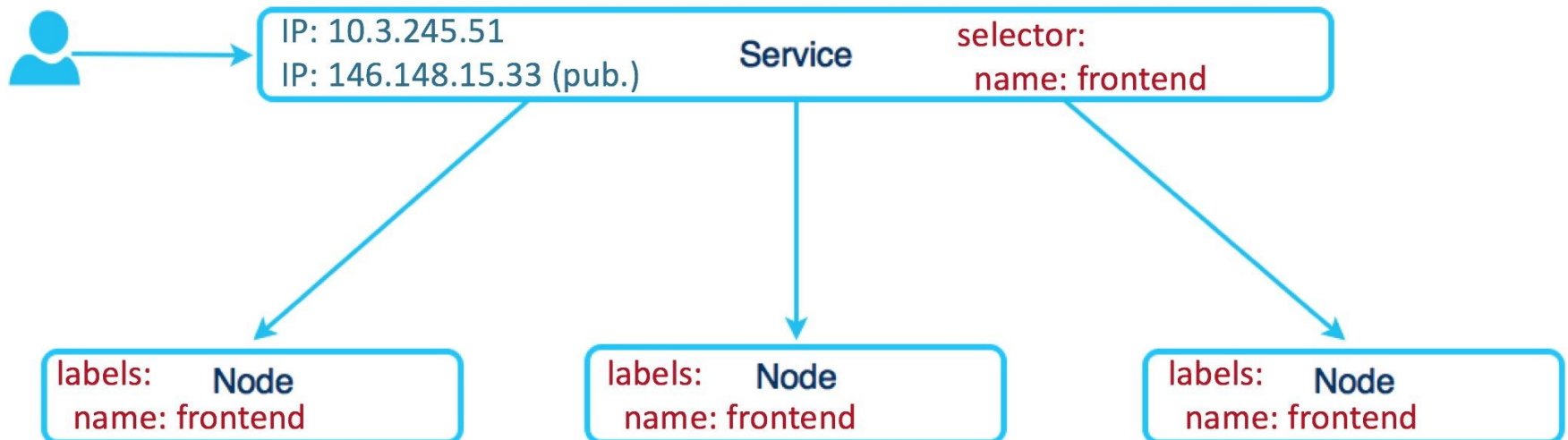
```
$ kubectl config view
$ kubectl config use-context
$ kubectl create -f ./<DIR>
$ kubectl create -f <URL>
$ kubectl create -R -f ./<DIR>
$ kubectl port-forward POD PORT:PORT
$ kubectl exec -ti .. /bin/bash
$ kubectl logs
$ kubectl --kubeconfig=./config -n monitoring get pods
$ kubectl get events
$ kubectl top nodes (requires metric server)
$ kubectl describe <>
$ kubectl get -o json
```

Kubernetes - essentials

ports:

-port: 80

targetPort: 8181



Dziękujemy za uwagę.

Podziel się opinią ze szkolenia [@sagespl](#) na

