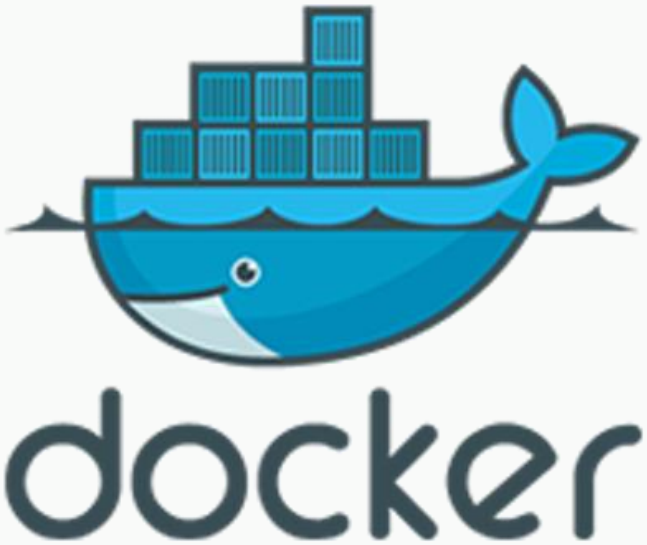


DOCKER I KUBERNETES – OD ZERA DO BOHATERA

Maciej Krajewski

Online – 18-22.11


KONFERENCJE ON-LINE SĄ JAK SEANSE SPIRYTYSTYCZNE
JOE MONSTER
GRAŻYNA JESTEŚ TAM?
POWIEDZ COŚ, JEŚLI NAS SŁYSZYSZ
JEST TAM KTOŚ Z TOBĄ?

RULES

Write your name in zoom

If you are done with task – use zoom reactions

Stand up during breaks

Slack

HI

BREAKS

DAY 1-5
9:00 – 10:30
10:45 – 12:15
13:15 – 14:45
15:00 – 16:00

DOCUMENTATION

UX Knowledge Base Sketch #72

NOTE: OF COURSE ANY ORGANIZATION CAN USE THE LIGHTWEIGHT FORM (E.G. BY CREATING SELF-SUFFICIENT TEAMS)

+ ANOTHER EXAMPLES: AGENCIES & FREELANCERS WHO'RE GETTING PAID FOR CREATING DOCUMENTATIONS. IF THEY WANT TO SWITCH TO LIGHTWEIGHT, FREQUENT CLIENT INVOLVEMENT CAN HELP.

LIGHTWEIGHT ← E.G.: AGILE + LEAN PROCESSES (BUT IN CASE OF THIRD PARTY DEVELOPMENT, YOU CAN'T DO EVERYTHING THE LEAN WAY) — IT CAN ALSO BE A MIX, A TRANSITION: AT THE BEGINNING OF THE PROJECT: LEAN, LATER: STANDARDIZED DOCUMENTATION — E.G.: ENTERPRISE-INTERNAL & REGULATORY COMPLIANCE → 1 TON HEAVY WEIGHT

CLOSE COLLABORATION OF A CROSS-FUNCTIONAL TEAM
EVERYONE IS ALREADY ON THE SAME PAGE.

"CUSTOM" CONTENT & STRUCTURE:
- WHAT IS THE MOST EFFECTIVE WAY?
- MINIMUM AMOUNT OF INFORMATION

IDEAL STAKEHOLDER MINDSET: FOCUSING ON THE OUTCOMES, IMPACT ACHIEVED BY THE PROJECT

SOME FORMS: SKETCHES (PAPER OR WHITEBOARD), PROTOTYPES (EVEN PAPER PROTOTYPES - YOU CAN EVOLVE IT BASED ON TEST RESULTS), DOCUMENTING COLLABORATIVE SESSIONS (E.G. CELLPHONE PHOTOS)

- THE DOCUMENTATION CONSISTS OF LIVING DOCUMENTS - EDITED & UPDATED CONTINUALLY
- CO-CREATED DOCUMENTATION - FEEDBACK & QUESTIONS

IDEAL FOR ITERATIVE & INCREMENTAL PRODUCT DEVELOPMENT PROCESSES "WORKING SOFTWARE OVER COMPREHENSIVE DOCUMENTATION" (AGILE MANIFESTO)

MAIN GOALS:
- ELIMINATING WASTE
- MOVING FORWARD FASTER
- CAPTURE CONVERSATIONS (→RECALL)

THE RELEASED PRODUCT/FEATURE ITSELF IS ALSO A DOCUMENTATION
→ CONTINUOUS DELIVERY & IMPROVEMENT → HEAVY DOCUMENTATION WOULD QUICKLY BECOME OBSOLETE

IN MANY CASES: SILOS, ISOLATED TEAMS

STRICT STANDARDS & RULES
BINDING CONTRACTS

USUAL STAKEHOLDER MINDSET: FOCUSING ON SHINY DESIGN ARTIFACTS & DELIVERABLES

SOME FORMS:
REQUIREMENTS DOCUMENTS
HUGE PILE OF DETAILED SPECIFICATION
BIG DESIGN UP FRONT

THE DOCUMENTATION IS "CARVED IN STONE", IT'S HARDER TO MODIFY IT LATER IN THE PROCESS

APPROPRIATE FOR WATERFALL PRODUCT DEVELOPMENT PROCESSES WITH BIG HAND-OFFS

MAIN GOALS, REASONS:
- THE PROCESS HEAVILY RELIES ON DOCUMENTATION
- IT NEEDS TO BE UNDERSTANDABLE FOR OTHER TEAMS

DRAWBACKS:
TAKES HUGE AMOUNT OF TIME & CREATES A LOT OF WASTE (BUT SOMETIMES IT IS NECESSARY)

ADVICE

WHO IS THE TARGET AUDIENCE OF YOUR DOCUMENTATION? → DESIGN IT!

E.G. FOR DEVELOPERS:
- IMPLEMENTATION
  - DATA MODELING
  - DATA TYPES ETC.

- FOR YOURSELF
- FOR YOUR TEAM
- FOR THE CLIENT
- FOR A THIRD PARTY TEAM

} DIFFERENT GOALS, MOTIVATION, TERMINOLOGY } UX OF THE DOCUMENTATION, E.G. FINDABILITY

YOUR DESIGN PROCESS SHOULD DETERMINE WHAT THINGS YOU DOCUMENT, DOCUMENTATION SHOULD COMPLEMENT & ENHANCE THE PROCESS (E.G.: RESEARCH, SYNTHESIZING, IDEATION, DESIGN, TEST)

ORGANIZE LIVE DEMO SESSIONS, INVITE YOUR TEAM TO OBSERVE USABILITY TESTS → FIRSTHAND INFORMATION IS THE BEST! } RESULT: NO DEPENDENCY ON THE DOCUMENTATION

→ SINGLE SOURCE OF TRUTH
DOCUMENTATION SHOULD BE EASILY ACCESSIBLE FOR THE WHOLE TEAM
- COLLABORATIVE: CONTRIBUTIONS, VERSIONING
- KEEP IT UP TO DATE

COMBINE WORDS & ILLUSTRATIONS (VISUALS)
E.G. ANIMATED GIFS

CREATE A GRAVEYARD FOR DISCARDED VERSIONS, IDEAS

INCLUDE WHAT PROBLEMS YOU ARE ADDRESSING, ADD CONTEXT, DISCUSSIONS, INSPIRATIONS...

COMMON GOALS

- DOCUMENTING YOUR WAY OF THINKING
- INFORMATION FOR YOUR FUTURE SELF/ A TEAM/THIRD PARTIES
- CREATING A DECISION-MAKING TOOL (MORE INFORMED DECISIONS)
- TEAM-ALIGNMENT, SHARED VISION
- EXPLAINING THE "HOW" AND THE "WHY"

(AND SOMETIMES THE GOAL IS TO BE IN ACCORDANCE WITH A CONTRACT)


TRY
FAIL
SUCCESS

AGENDA

Microservices

Rest API

DOCKER

KUBERNETES

Monitoring Aplikacji

Centralne Logowanie

12 Factor App

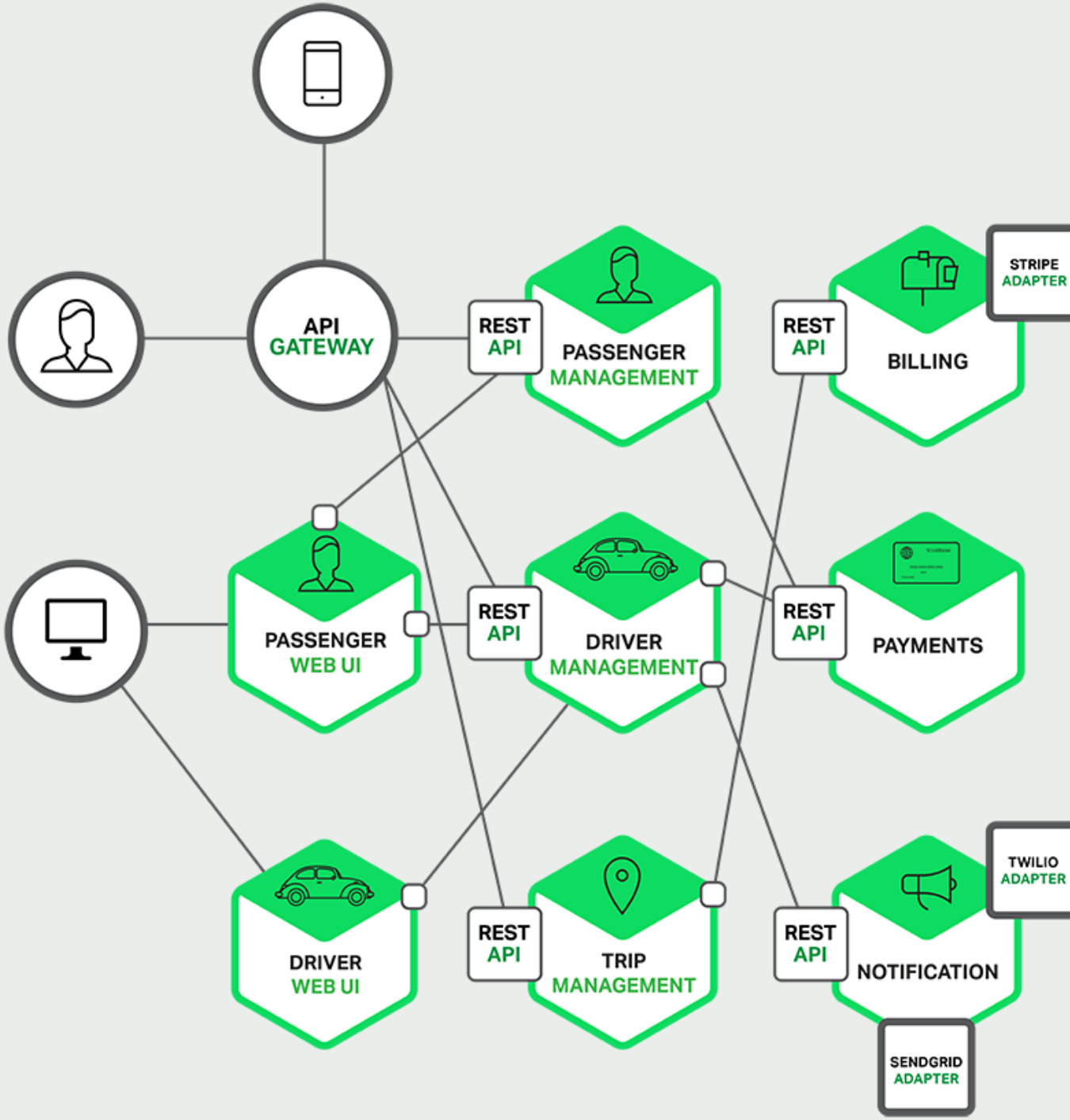
CI/CD

IDE preparation

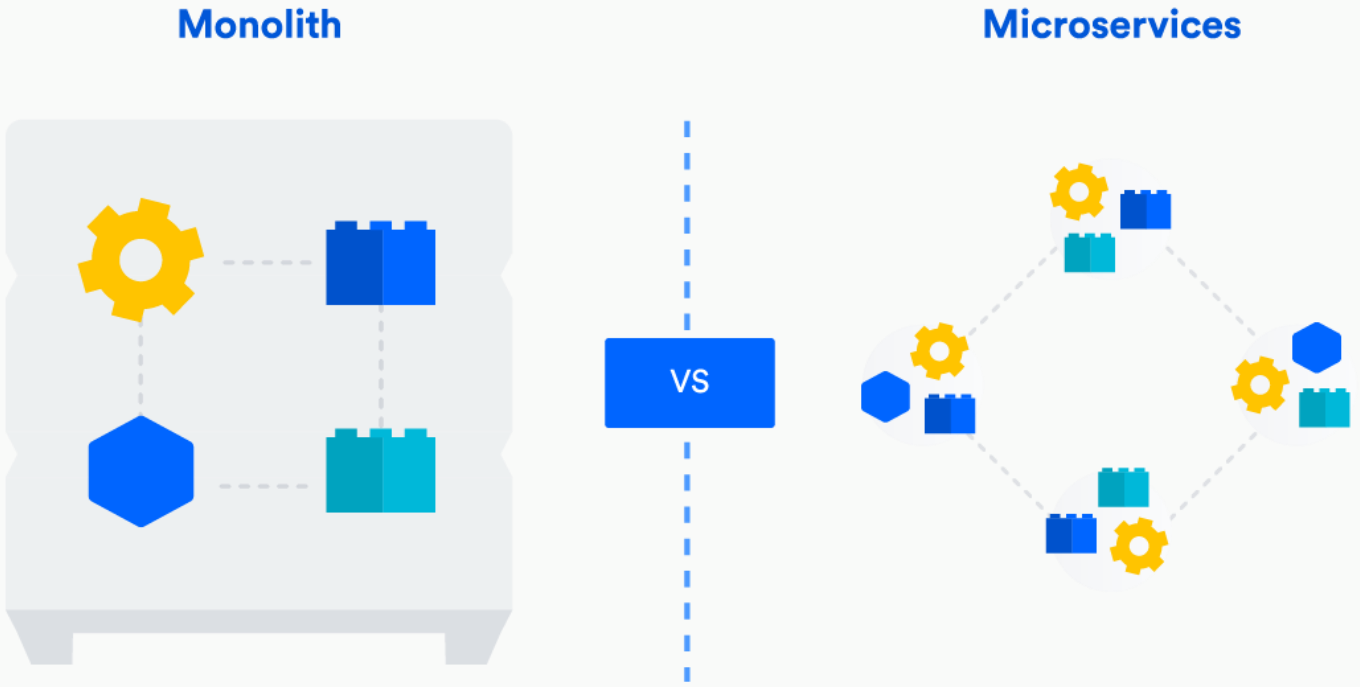
Kilka przydatnych pluginów do VSC:

- Docker
- Kubernetes
- Markdown All in One
- YAML

Microservices



Microservice vs Monolith

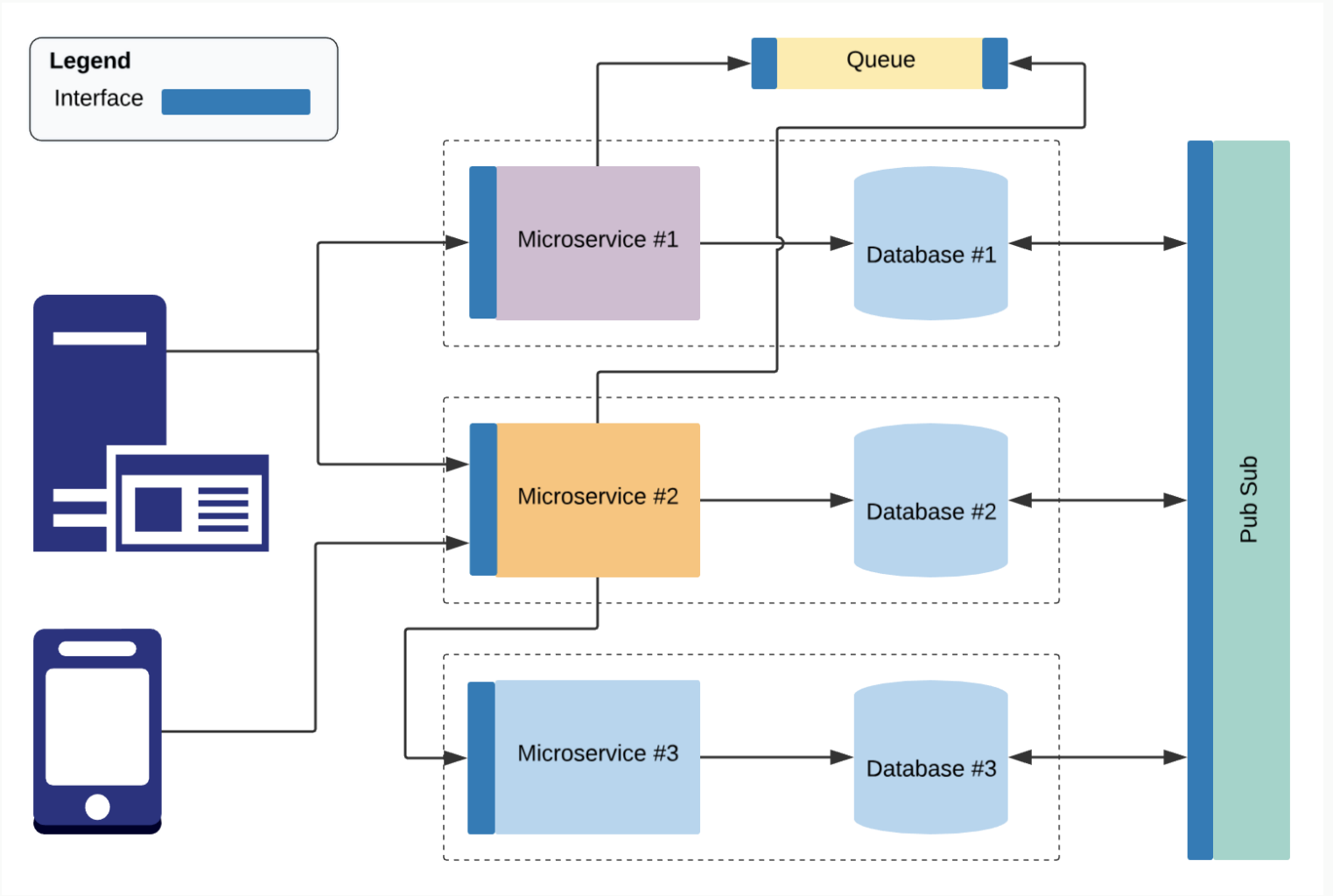


Microservice as an app

- Ls | grep | sed
- Microservice and domain

Microservice as an app – how to communicate?

Microservice as an app



Microservice as an app

Using containers (e.g; Docker) doesn't mean you are adopting Microservices

If different services, share code directly within the project, then you are doing it wrong.

Depending on too many Microservices to perform a single task, could potentially lead to dependency and performance issues.

There is a common misconception using middleware (API Gateway's, Messaging Queues & etc.) Microservices is a bad practice.

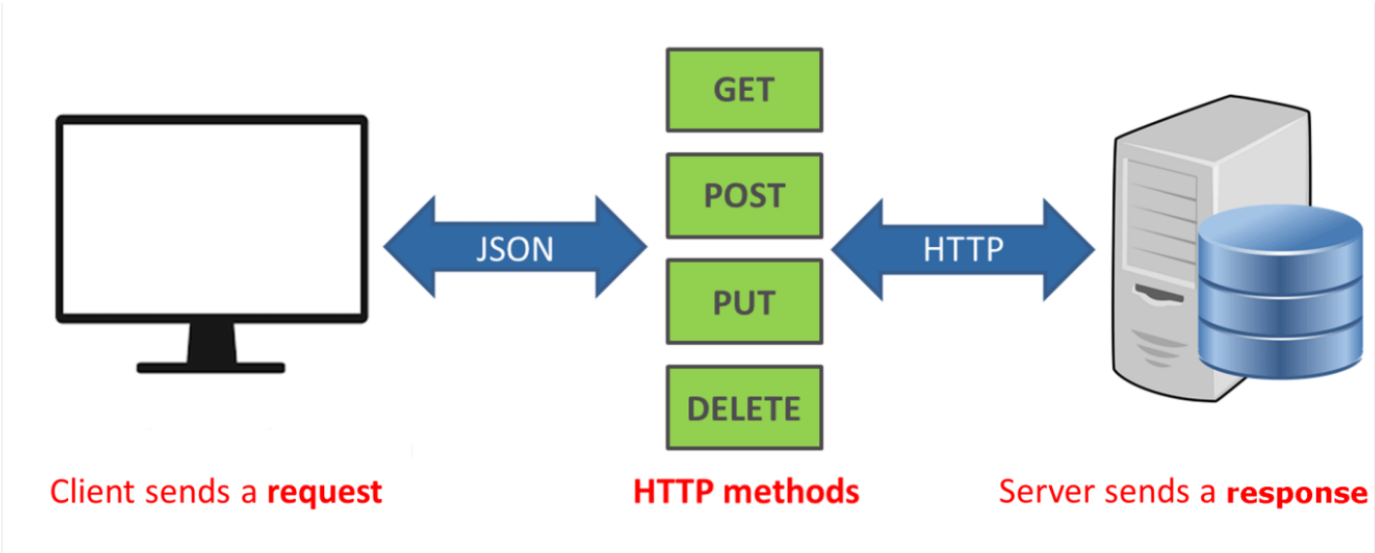
For certain applications, it is not mandatory for Microservices to communicate via REST APIs.

It is also important to understand that when designing Microservices, some of the best practices we used to follow in developing Monoliths could be challenger (eg. Common database).

Dividing Microservices based on technical layers also considered as a bad practice.

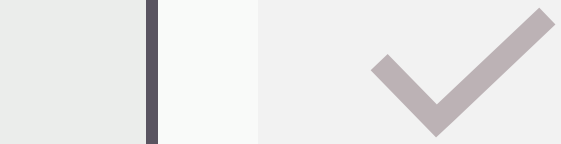
Microservice – REST API

REST = Represetation State Transfer



REST API Basis

Uniform Interface



Endpoints

Stateless (all infromation must be send)

REST API Basis

Use nouns not verbs (Products vs GetProducts)

Use query for filter (Products?name='ABC')

Proper HTTP codes (200, 201, 202,400, 204/404)

Use versions (v1/products)

https://docs.microsoft.com/pl-pl/azure/architecture/best-practices/api-design

REST API

Basis

GET – access data

POST – insert new data

PUT – modify data

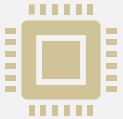
DELETE – remove data

PATCH – partial update

REST API Maturity

Level 0: Define one URI, and all operations are POST requests to this URI.

Level 1: Create separate URIs for individual resources.

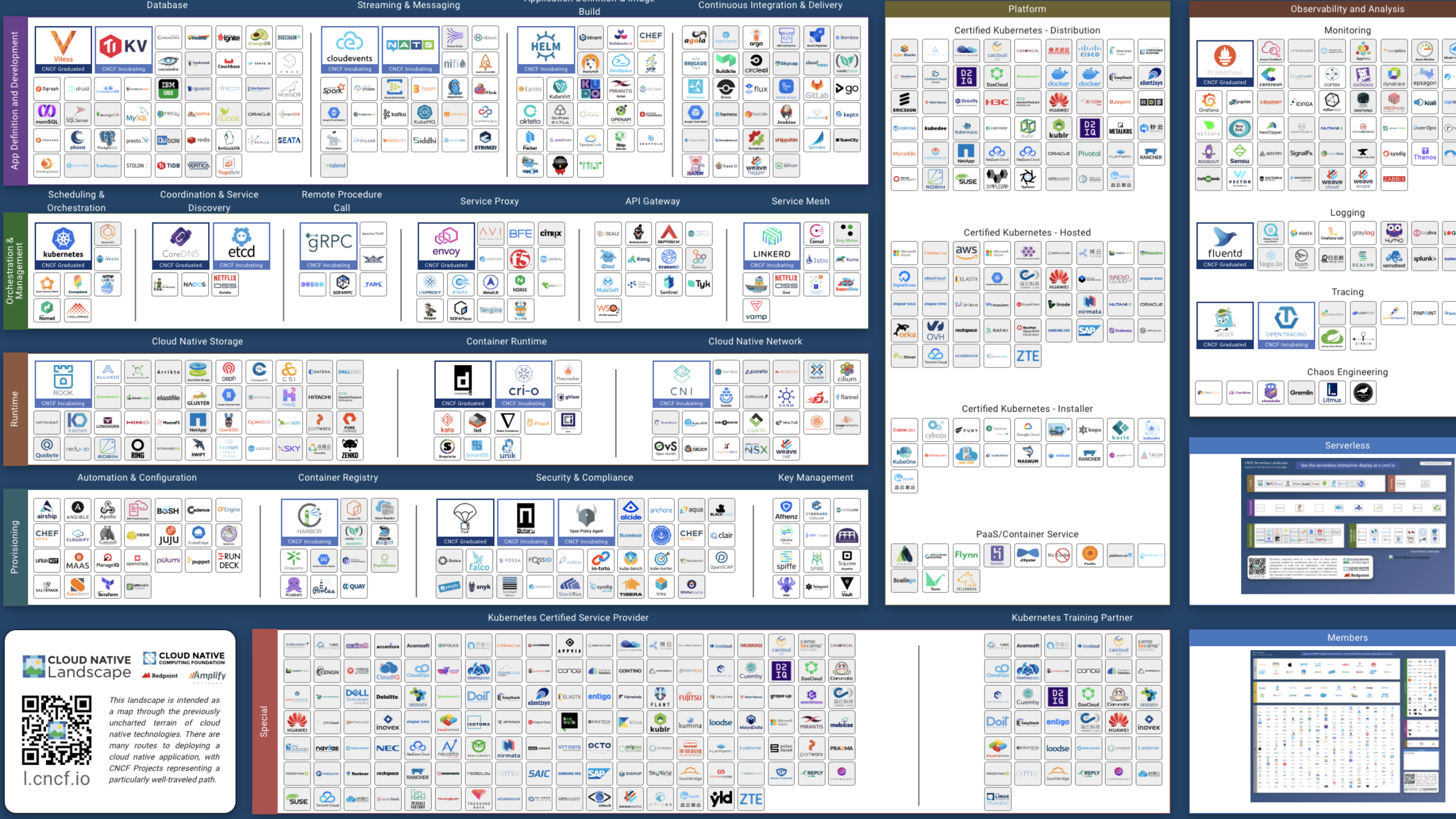


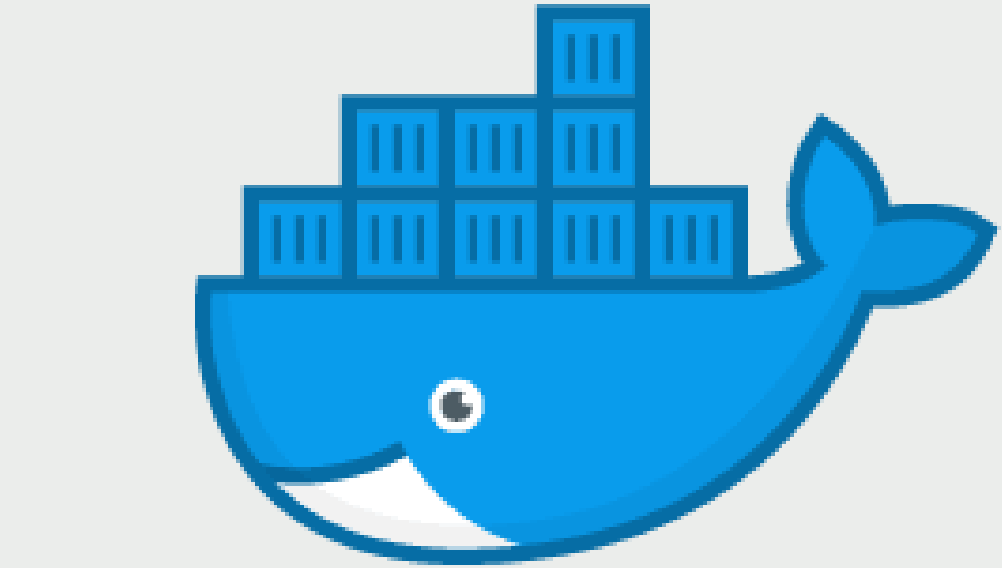
Level 2: Use HTTP methods to define operations on resources.

Level 3: Use hypermedia (HATEOAS).

SWAGGER

https://petstore.swagger.io

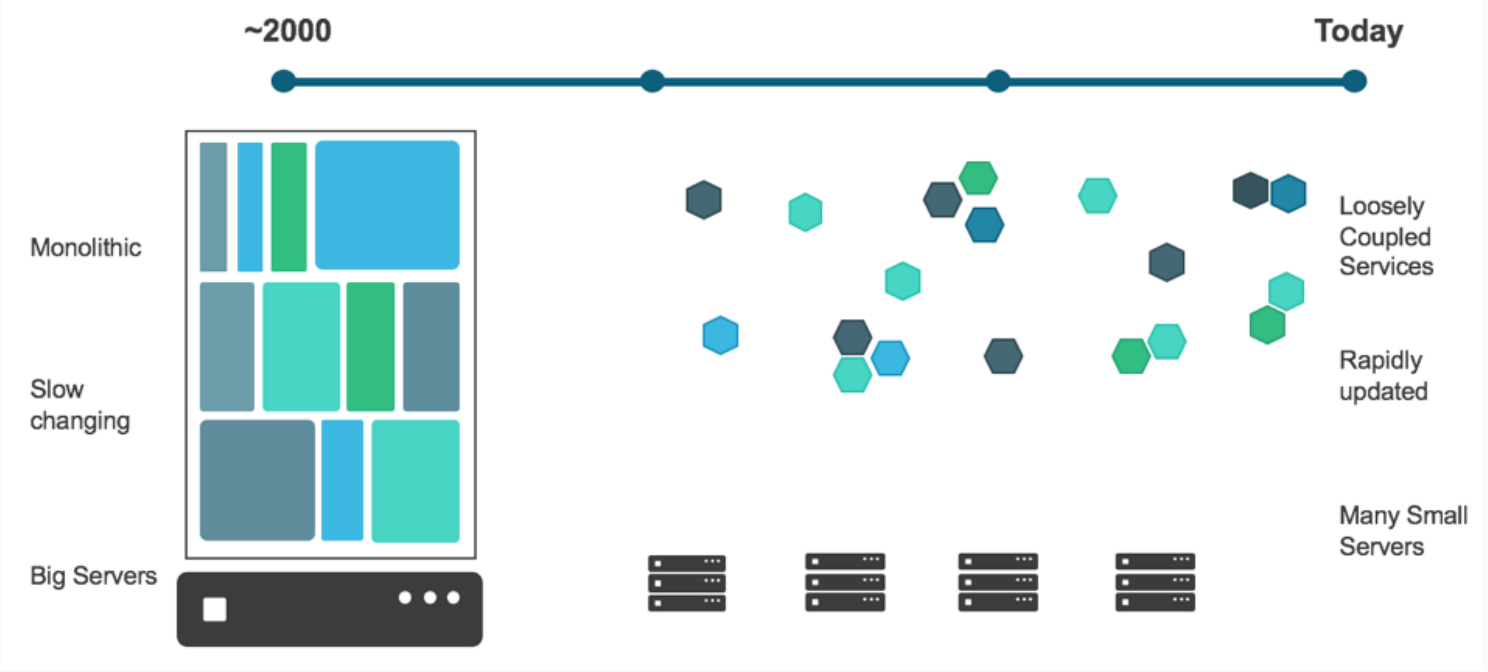

Database
Streaming & Messaging
Application Definition & Image Build
Continuous Integration & Delivery
App Definition and Development
Vitess
CNCF Graduated
KV
CNCF Incubating
cloudevents
CNCF Incubating
NATS
CNCF Incubating
HELM
CNCF Incubating
Scheduling & Orchestration
Coordination & Service Discovery
Remote Procedure Call
Service Proxy
API Gateway
Service Mesh
Orchestration & Management
kubernetes
CNCF Graduated
CoreDNS
CNCF Graduated
etcd
CNCF Incubating
gRPC
CNCF Incubating
envoy
CNCF Graduated
LINKERD
CNCF Incubating
Cloud Native Storage
Container Runtime
Cloud Native Network
Runtime
ROOK
CNCF Incubating
cri-o
CNCF Incubating
CNI
CNCF Incubating
Automation & Configuration
Container Registry
Security & Compliance
Key Management
Provisioning
HARBOR
CNCF Incubating
TUF
CNCF Graduated
Notary
CNCF Incubating
Open Policy Agent
CNCF Incubating
Platform
Certified Kubernetes - Distribution
Certified Kubernetes - Hosted
Certified Kubernetes - Installer
PaaS/Container Service
Observability and Analysis
Monitoring
Prometheus
CNCF Graduated
Logging
fluentd
CNCF Graduated
Tracing
JAEGER
CNCF Graduated
OPENTRACING
CNCF Incubating
Chaos Engineering
Serverless
Kubernetes Certified Service Provider
Kubernetes Training Partner
Special
Members
CLOUD NATIVE Landscape
CLOUD NATIVE COMPUTING FOUNDATION
Redpoint
Amplify
l.cncf.io
This landscape is intended as a map through the previously uncharted terrain of cloud native technologies. There are many routes to deploying a cloud native application, with CNCF Projects representing a particularly well-traveled path.



docker

Why Docker?

Why Docker?


~2000
Today
Monolithic
Slow changing
Big Servers
Loosely Coupled Services
Rapidly updated
Many Small Servers

Solution to problems

- Different libraries version
- Different OS versions
- Reproduction of behavior – DEV != STAGE != PROD
- Coexistence of software & applications (backward/forwad 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 (unless you can’t)

Docker – technologies under the hood

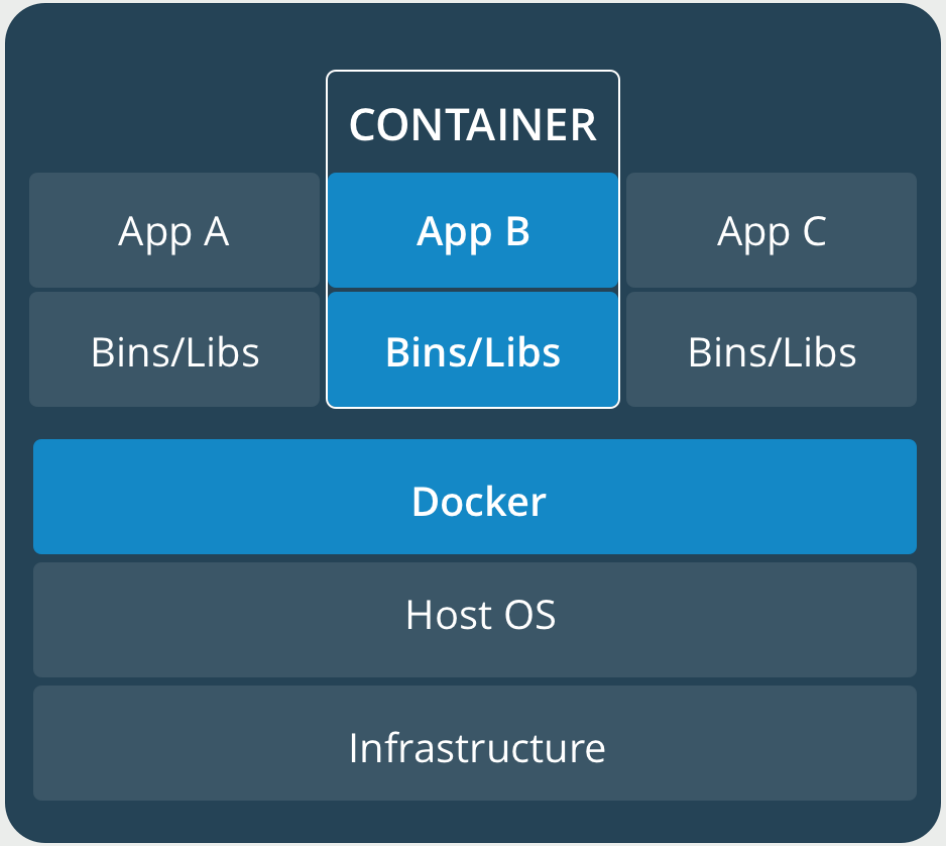
namespaces:

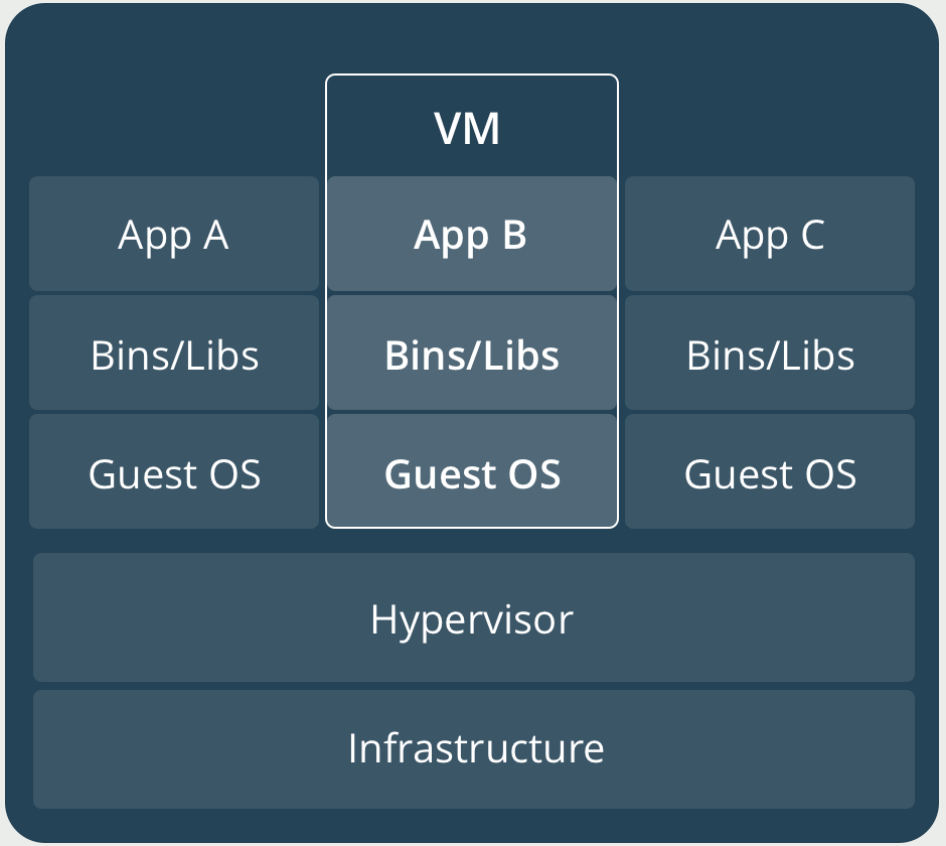
- **The pid namespace:** Process isolation (PID: Process ID).
- **The net namespace:** Managing network interfaces (NET: Networking).
- **The ipc namespace:** Managing access to IPC resources (IPC: InterProcess Communication).
- **The mnt namespace:** Managing filesystem mount points (MNT: Mount).
- **The uts namespace:** Isolating kernel and version identifiers. (UTS: Unix Timesharing System).

control groups (cgroups)

union file system (AUFS, brtfs, vfs, DeviceMapper, overlay)

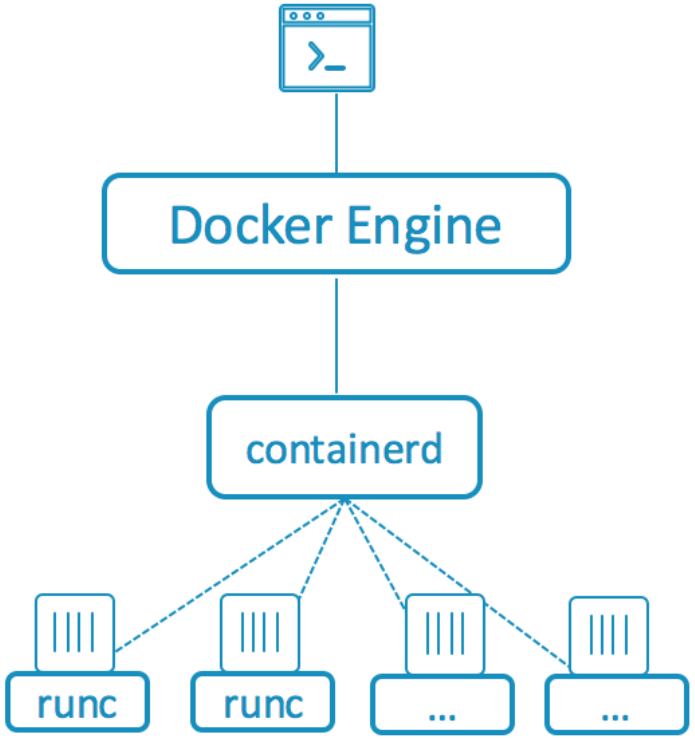
container format (libcontainer)





Is docker a VM?

Docker, containerd?



Same Docker UI and commands

User interacts with the Docker Engine

Engine communicates with containerd

containerd spins up runc or other OCI compliant runtime to run containers

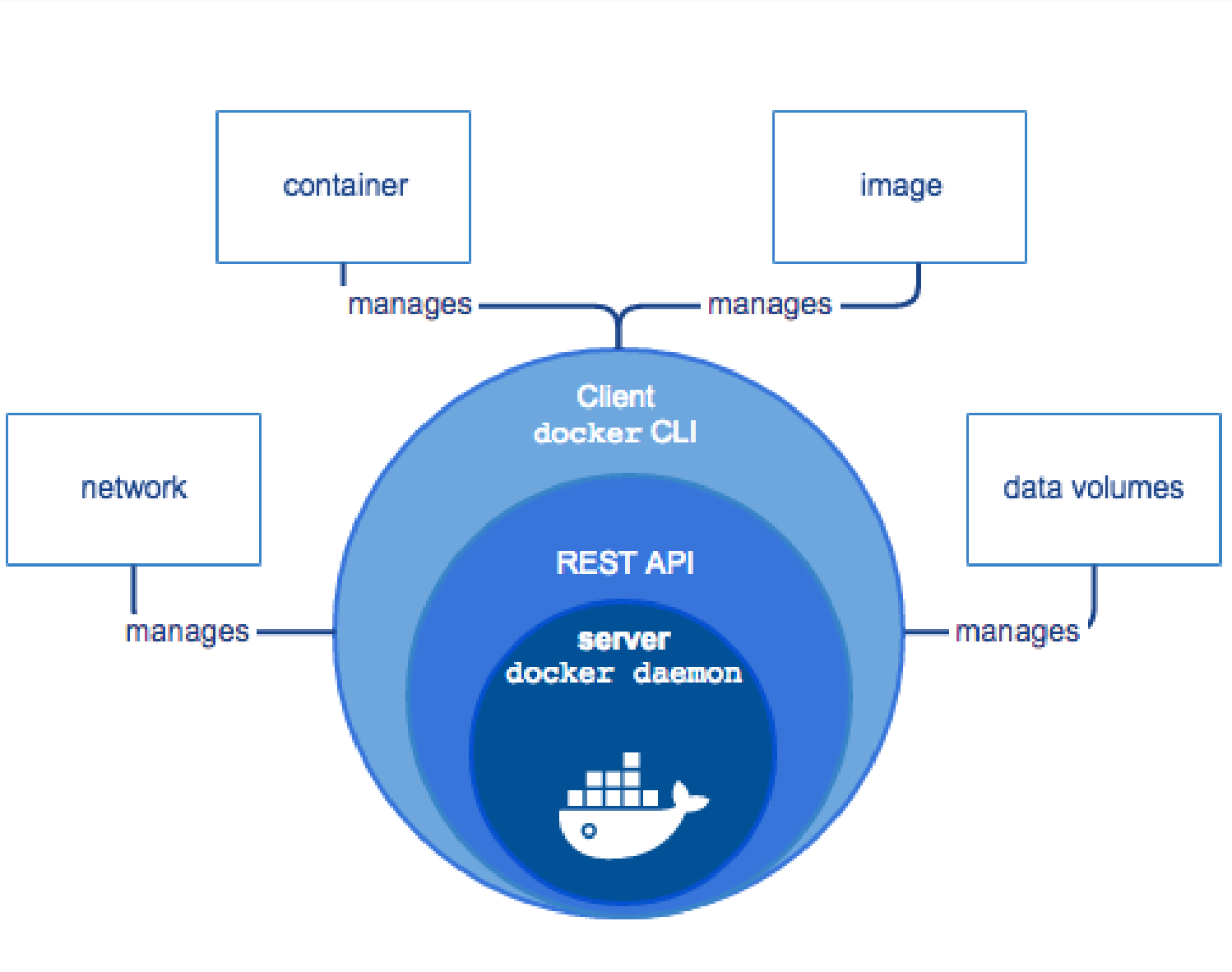
Docker – the only solution?

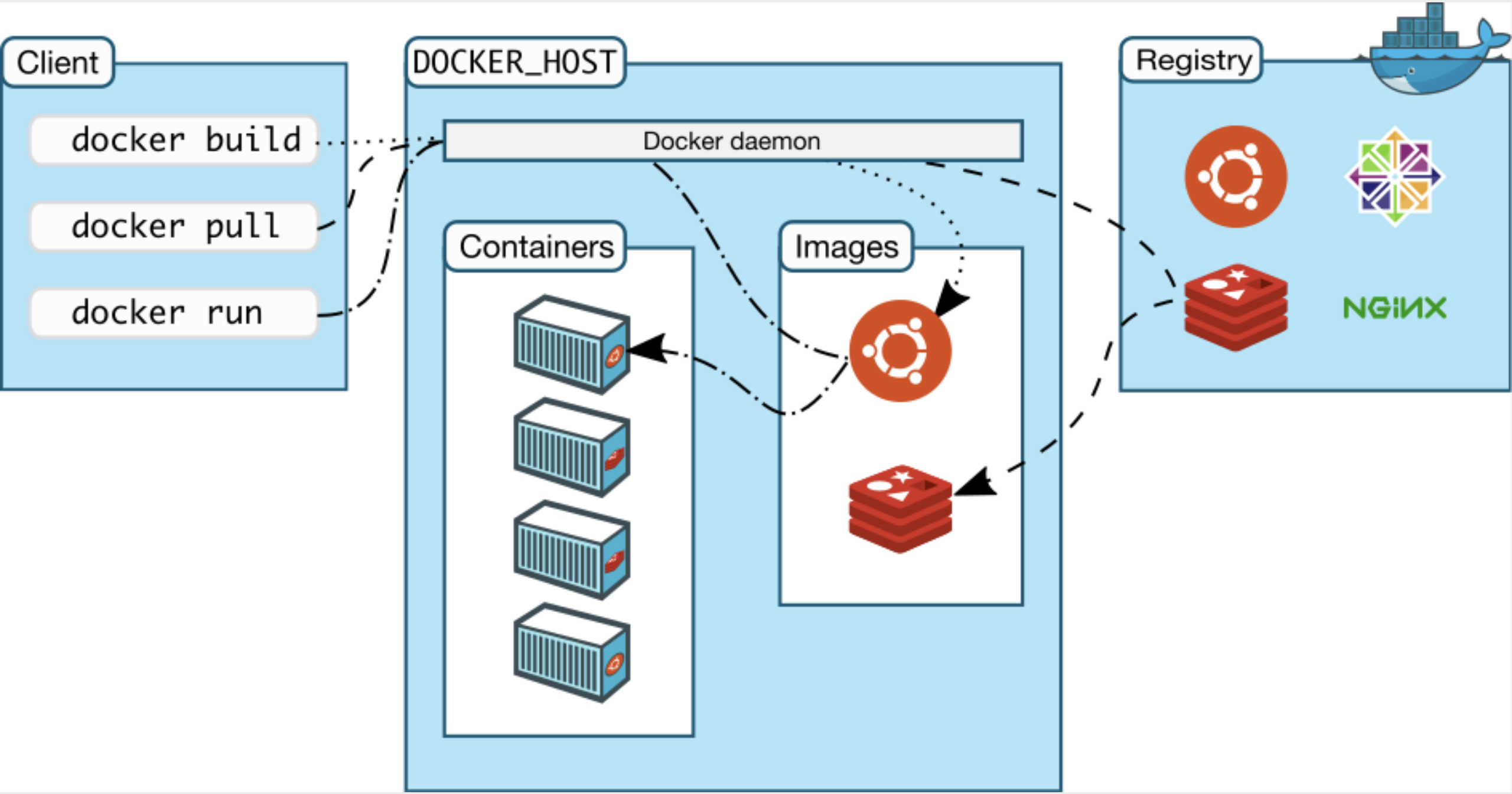
OCI Standard:

- runc
  - runtime-spec
  - image-spec
  - distribution-spec

- Containerd
- Podman
- Kata Containers
- gVisor
- CRI-O

What Docker is?




Client
docker build
docker pull
docker run
DOCKER_HOST
Docker daemon
Containers
Images
Registry
NGINX

Documentation

http://docs.docker.com

Docker - inastallation

Wspiera wszystkie OS'y:

- **Linux**
- **OS X* (Hyperkit)**
- **Windows** (Hyper-V/WSL)**

Windows Server >= 2016

Windows 10 64-bit: Pro, Enterprise, or Education (Build 17134 or later) or Home***.

Docker - inastallation

Dla Linux’a

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

Docker CLI

```
docker help
docker <COMMAND> --help
```

Docker – First Task

```
docker container run hello-world

~~docker run hello-world~~
```

Dockerfile

- FROM
- LABEL
- RUN
- COPY and ADD
- EXPOSE
- ENV
- ARG
- ENTRYPOINT and CMD
- VOLUME

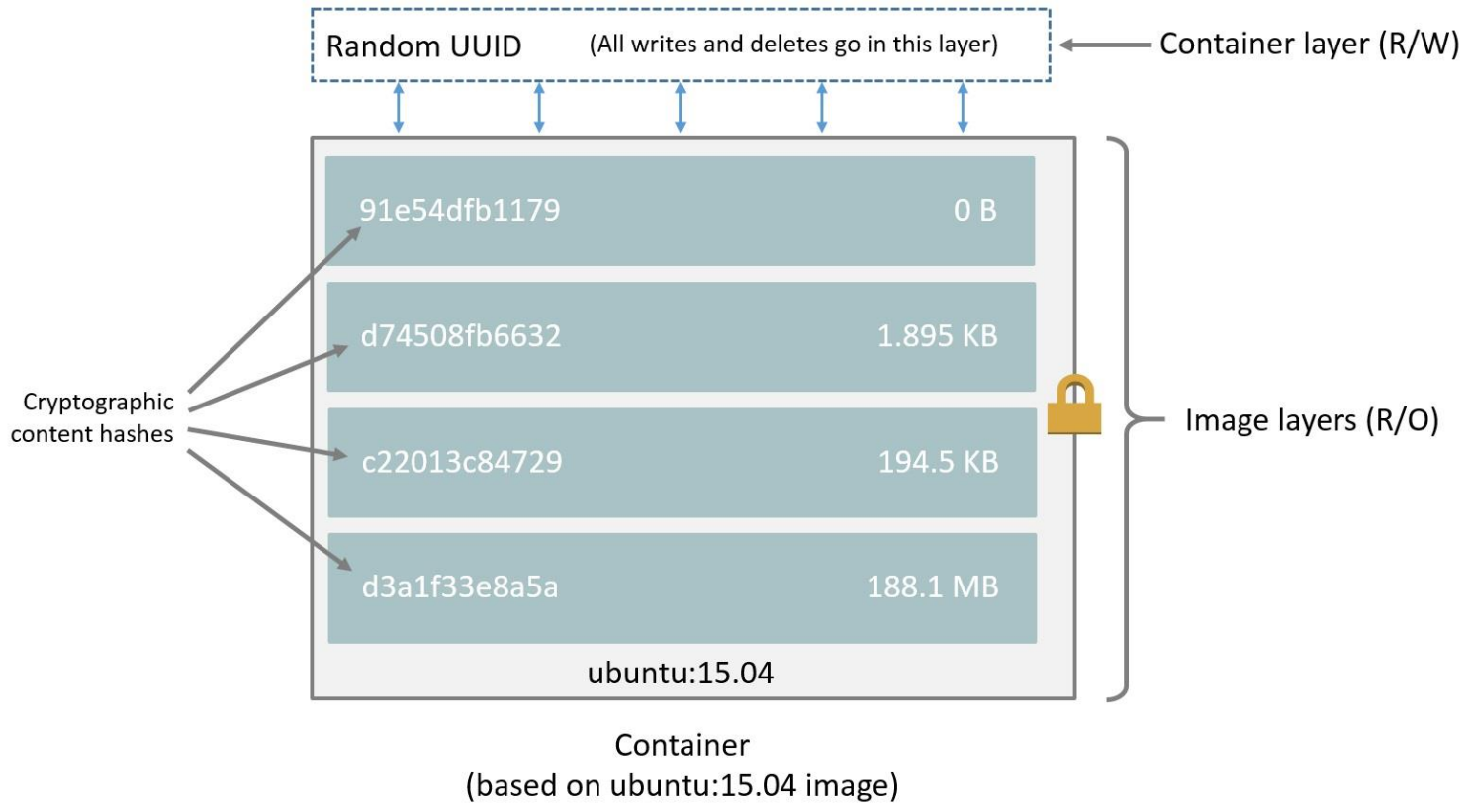
	No ENTRYPOINT	**ENTRYPOINT exec_entry p1_entry**	**ENTRYPOINT [“exec_entry”, “p1_entry”]**
No CMD	error, not allowed	/bin/sh -c exec_entry p1_entry	exec_entry p1_entry
CMD [“exec_cmd”, “p1_cmd”]	exec_cmd p1_cmd	/bin/sh -c exec_entry p1_entry	exec_entry p1_entry exec_cmd p1_cmd
CMD [“p1_cmd”, “p2_cmd”]	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

Entrypoint vs CMD

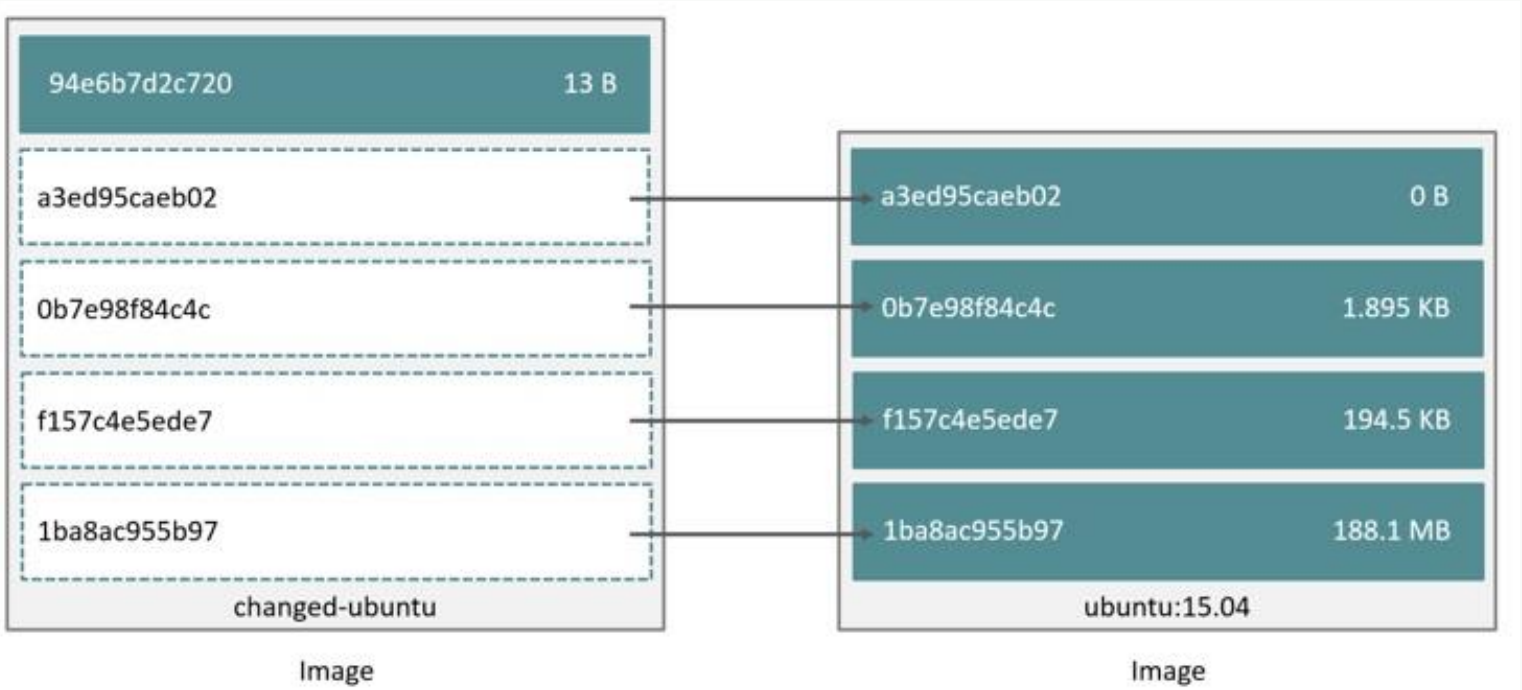
Docker – .dockerignore

Rule	Behavior
# comment	Ignored.
/temp	Exclude files and directories whose names start with temp in any immediate subdirectory of the root. For example, the plain file /somedir/temporary.txt is excluded, as is the directory /somedir/temp.
//temp*	Exclude files and directories starting with temp from any subdirectory that is two levels below the root. For example, /somedir/subdir/temporary.txt is excluded.
temp?	Exclude files and directories in the root directory whose names are a one-character extension of temp. For example, /tempa and /tempb are excluded.

Docker - Layers



Docker - Layers



Docker – Layers veryfication

```
docker run --rm -it  -v \
/var/run/docker.sock:/var/run/docker.sock \
wagoodman/dive obraz
```

Docker – Dockerfile optymalization

https://github.com/goodwithtech/dockle

```
docker run --rm  -v \
/var/run/docker.sock:/var/run/docker.sock \
goodwithtech/dockle wordpress
```

Docker – Security scan

https://github.com/aquasecurity/trivy

```
docker run --rm -v \
/var/run/docker.sock:/var/run/docker.sock \
aquasec/trivy image IMAGE
```

https://docs.docker.com/engine/scan/

Docker – Healthcheck

```
HEALTHCHECK [OPTIONS] CMD command
HEALTHCHECK --interval=5m --timeout=3s CMD curl -f http://localhost/ || exit 1
```

--interval=DURATION (default: 30s)
--timeout=DURATION (default: 30s)
--start-period=DURATION (default: 0s)
--retries=N (default: 3)

Exit CODES
•0: success - the container is healthy and ready for use
•1: unhealthy - the container is not working correctly
•2: reserved - do not use this exit code

Docker – Container

run

attach

exec –ti nginx [/bin/bash | /bin/sh]

stop

start

restart

rm

Docker – Container

PAUSE == SIGSTOP

STOP == SIGTERM + SIGKILL

KILL == SIGKILL

1

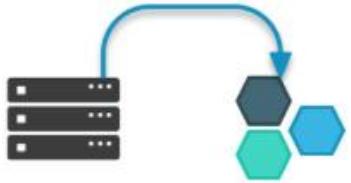
Containerize Monoliths

Build-Test for CI; Migrate to the Cloud;
Achieve better CapEx/OpEx than with VMs

2

Containerize Monolith; Transform to Microservices

Look for Shared Services to Transform



3

Enable New Microservices and Apps

Greenfield CaaS

Docker how to migrate

Docker – Docerfikle for DB

- easy way:
- Np. Tak jak w obrazie mariadb poprzez skopiowanie plików do katalogu *„/docker-entrypoint-initdb.d”*

Docker – Docerfikle for DB

hard way example #1 (mysql):

```
RUN /bin/bash -c "/usr/sbin/mysqld --bootstrap  \
 --verbose=0 < file.sql" && rm -f file.sql
```

Docker – Docerfikle for DB

hard way example #1 (postgres):

```
RUN setuser postgres postgres --single -jE ${POSTGRES_DB} \
 < file.sql; rm -f file.sql
```

Docker – registry

https://distribution.github.io/distribution/

TL;DR

```
docker run -d -p 5000:5000 --name registry registry:2
```

Docker – registry

```
docker tag ubuntu localhost:5000/myfirstimage
docker push localhost:5000/myfirstimage
```

https://distribution.github.io/distribution/about/configuration/

https://www.docker.com/pricing

https://distribution.github.io/distribution/recipes/mirror/

Docker lokalne repozytorium czy w chmurze?


ONE CONTAINER IS NOT ENOUGH
WE NEED TO GO DEEPER
emegen.fr

```
docker run --privileged -it -d docker:stable-dind

docker run -v /var/run/docker.sock:/var/run/docker.sock -it docker
```

DiD

Docker - ReCap

Image build | rm | prune | --help

Container | create | run | stop… | prune | --help Volume | create | rm | prune | --help

Tag image/repository:version dst_address/repository:version | --help

Build it -> Run it -> Tag it -> Push it

Docker – bigger numer of containers

Docker natively has 2 tools that support the management of bigger number of containers:

- compose
- swarm

Docker – bigger numer of containers

Docker compose – configuration

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

Docker – bigger numer of containers

Setting up variables:

$ {VARIABLE:-default} will set a variable if VARIABLE is unset or empty

$ {VARIABLE-default} will set the variable if VARIABLE is unset

Required variables:

$ {VARIABLE:?Err} fails if VARIABLE is unset or empty

$ {VARIABLE?Err} fails if VARIABLE is unset

Using YAML Anchors

https://docs.docker.com/compose/compose-file/#extension-fields

Docker – Logging

Using docker we have two options of managing logs:

- Standard way – each application logs to a local file, logs are rotated, etc.
- Logs are redirected to STDOUT

https://docs.docker.com/engine/admin/logging/overview/


docker
SWARM


portainer.io

Documentation

http://docs.docker.com

Docker – Best practices

- Consider one Dockerfile per folder
- Use GIT!
- Keep image small
- Don’t put secrets! (or any other ‘personal’ stuff)
- Execute one app per container
- Use .dockerignore
- Layers are your friend – use them as middle step before build

Docker – Production Ready

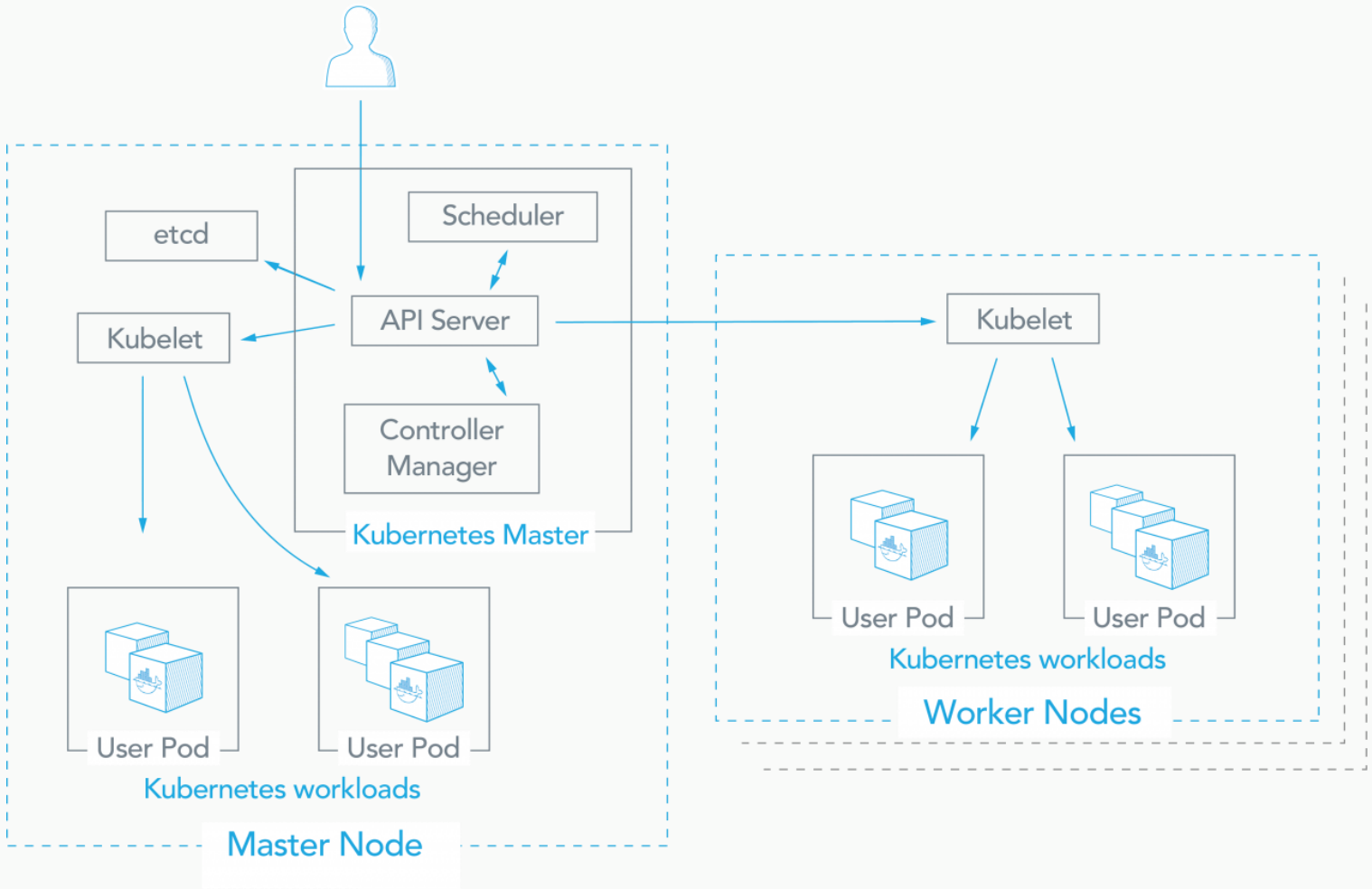
- Docker Host only
- Avoid Docker0 „cloud”
- Consider Haproxy, Nginx
- Healthz / Healtchecks
- Use ElK / GELF / LOKI stacks for logs
- Use clustered services
- Consider host discovery / dns automation (Consul)
- Consider Network Driver
- BUT USE K8S!

kubernetes

Kubernetes

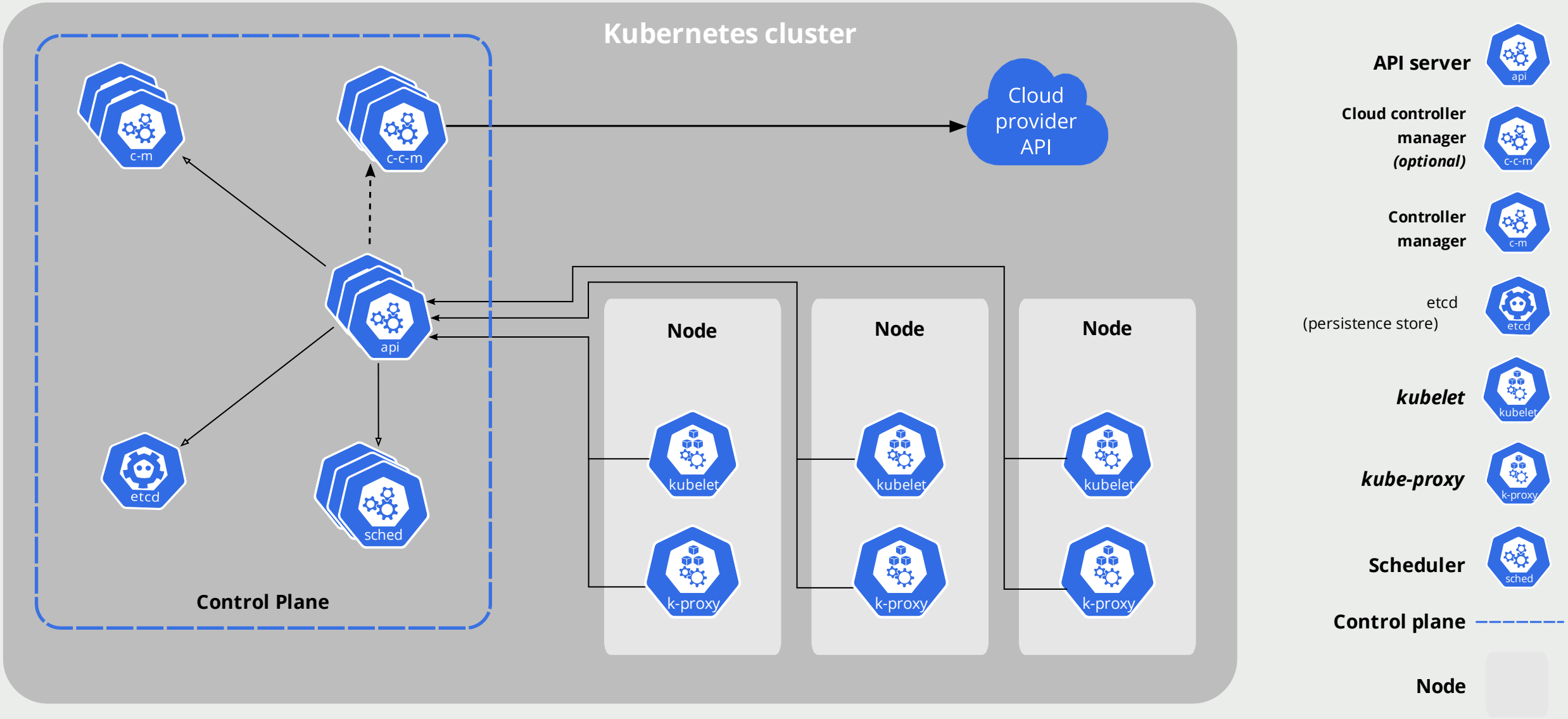
-

Architecture



API Server: management hub for Kubernetes
Scheduler: places a workload on the appropriate Node
Controller Manager: scales workloads up/down
etcd: stores configuration data which can be accessed by API Server

Kubelet: Receives pod specifications from API Server, updates Nodes
Master Node: places workloads on Nodes
Worker Nodes: receives requests from Master Nodes and dispatches them
User Pod: a group of containers with shared resources


Kubernetes cluster
c-m
c-c-m
Cloud provider API
api
etcd
sched
Control Plane
Node
kubelet
k-proxy
Node
kubelet
k-proxy
Node
kubelet
k-proxy
API server
api
Cloud controller manager (optional)
c-c-m
Controller manager
c-m
etcd (persistence store)
etcd
kubelet
kubelet
kube-proxy
k-proxy
Scheduler
sched
Control plane
Node

Kubernetes is just a tool!

Building blocks can be binaries or docker containers or ...

There is no „one setup for all”

Kubernetes – Orchestrator concepts

Replicas

Ingress

Healtcheck

Pod/Services

Configs

Secrets

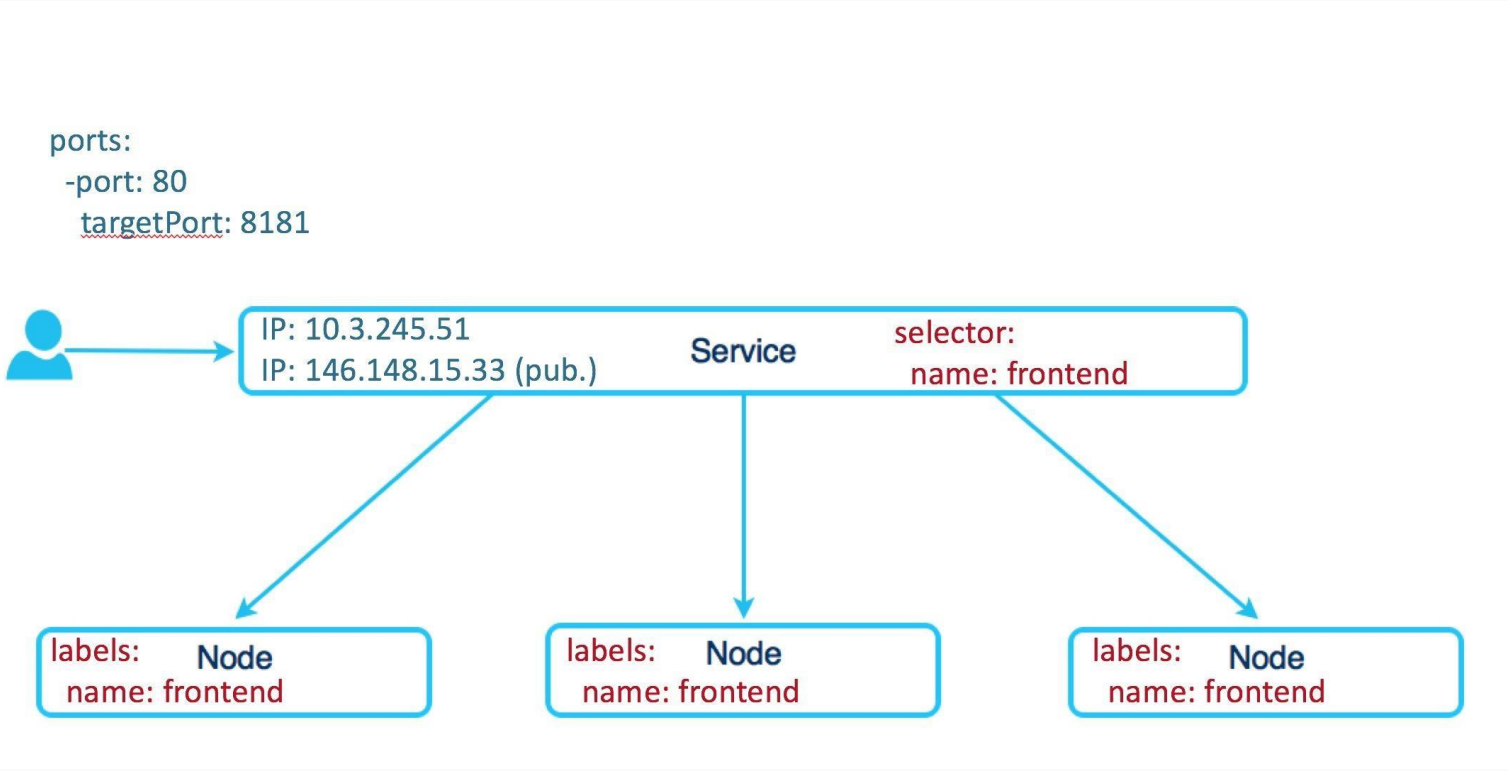
Volumes

Kubernetes
- kubectl

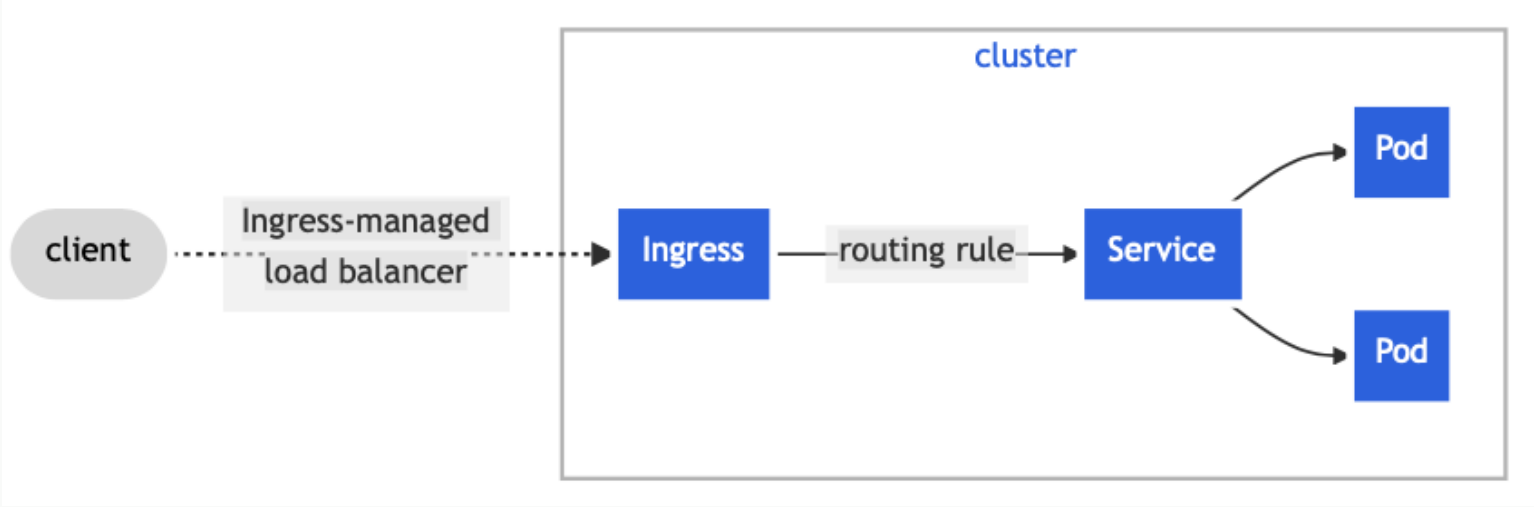
https://kubernetes.io/docs/tasks/access-application-cluster/configure-access-multiple-clusters/

```
$ 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
```

Kubernetes
- Service



Kubernetes
- Ingress



Kubernetes – Logic objects

Deployments

StatefulSet (for stateful services like redis)

DaemonSet

Job

CronJob

Kubernetes – Deployment

Rolling Update

Canary release

Kubernetes – Deployment

LivenessProbe

ReadinessProbe

Detach from service when fail!

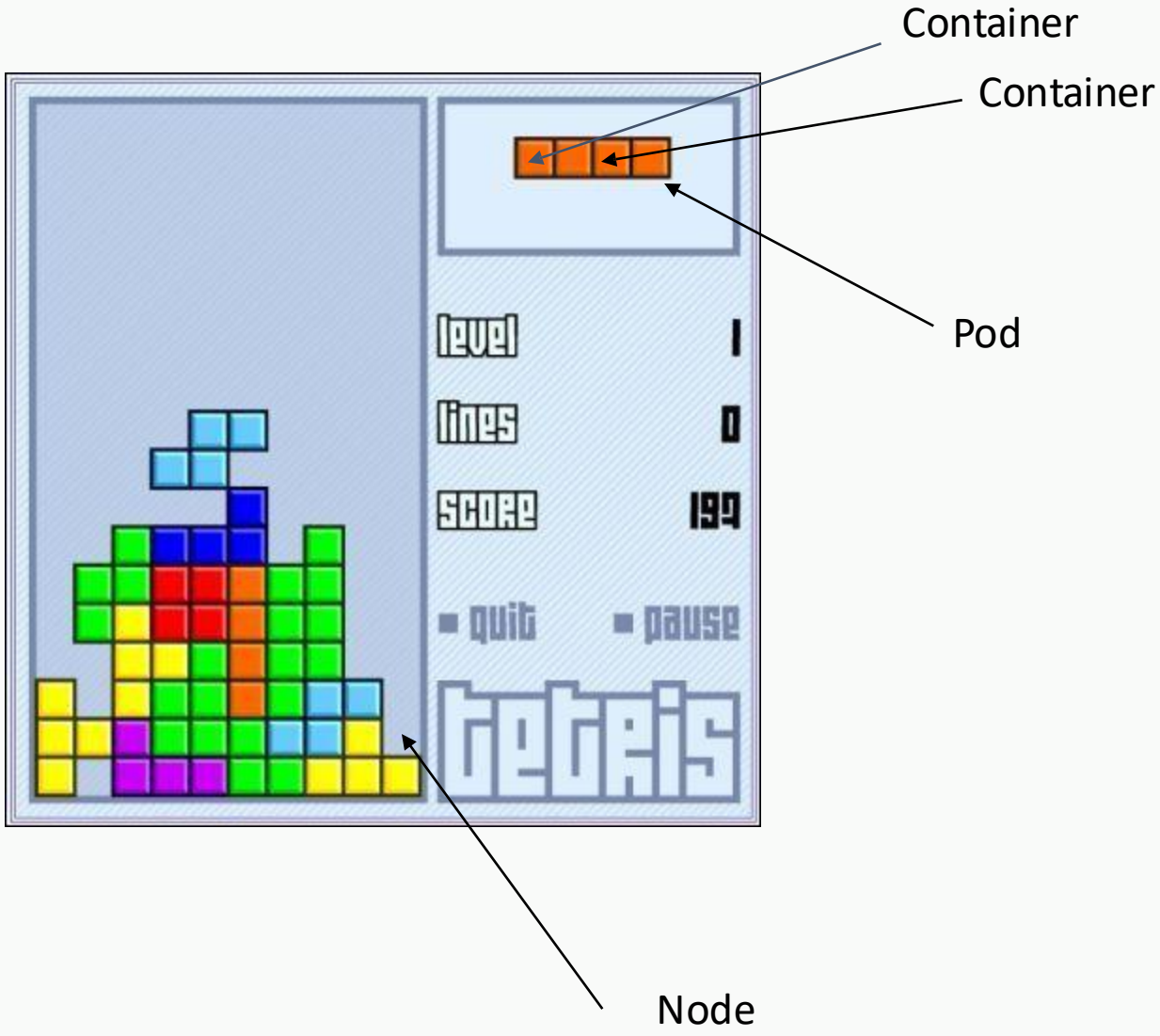
Kubernetes – QoS

Burstable

Guaranted

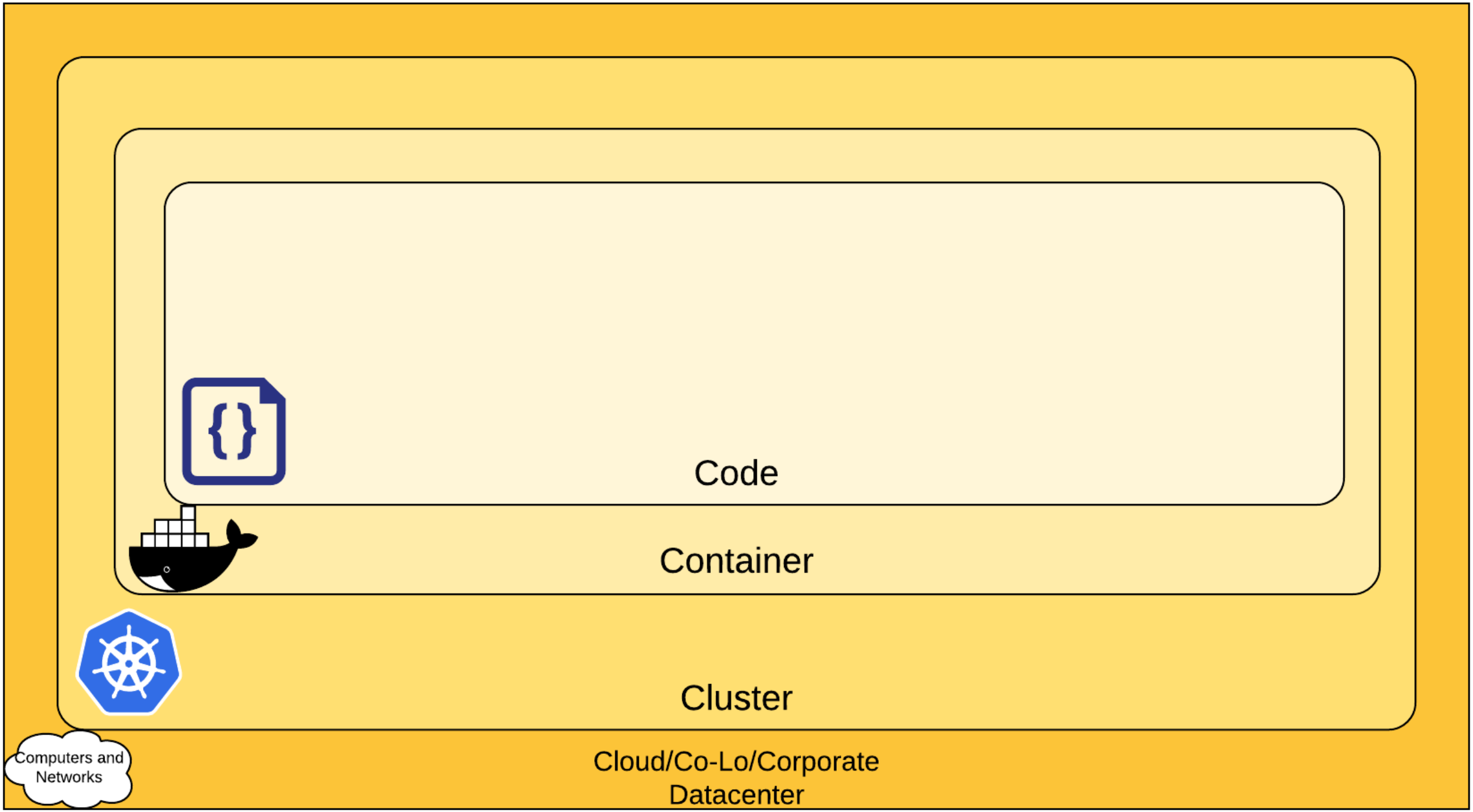
Best effort

Kubernetes is a player trying to win!



Kubernetes – Dashboard

https://kubernetes.io/docs/tasks/access-application-cluster/web-ui-dashboard/


Code
Container
Cluster
Computers and Networks
Cloud/Co-Lo/Corporate Datacenter

Kubernetes – Cloud Security concerns

Network access to API Server (Control plane)

Network access to Nodes (nodes)

Kubernetes access to Cloud Provider API

Access to etcd

etcd Encryption

Kubernetes – Cluster Security concerns

- RBAC Authorization (Access to the Kubernetes API)
- Authentication
- Application secrets management (and encrypting them in etcd at rest)
- Pod Security Policies
- Quality of Service (and Cluster resource management)
- Network Policies
- TLS For Kubernetes Ingress

Kubernetes – Container Security concerns

Container Vulnerability Scanning and OS Dependency Security

Image Signing and Enforcement

Disallow privileged users

Kubernetes – Code Security concerns

Access over TLS only

Limiting port ranges of communication

3rd Party Dependency Security

Static Code Analysis

Dynamic probing attacks

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Impact
Using Cloud credentials	Exec into container	Backdoor container	Privileged container	Clear container logs	List K8S secrets	Access the K8S API server	Access cloud resources	Images from a private registry	Data Destruction
Compromised images in registry	bash/cmd inside container	Writable hostPath mount	Cluster-admin binding	Delete K8S events	Mount service principal	Access Kubelet API	Container service account		Resource Hijacking
Kubeconfig file	New container	Kubernetes CronJob	hostPath mount	Pod / container name similarity	Access container service account	Network mapping	Cluster internal networking		Denial of service
Application vulnerability	Application exploit (RCE)	Malicious admission controller	Access cloud resources	Connect from Proxy server	Applications credentials in configuration files	Access Kubernetes dashboard	Applications credentials in configuration files		
Exposed Dashboard	SSH server running inside container				Access managed identity credential	Instance Metadata API	Writable volume mounts on the host		
Exposed sensitive interfaces	Sidecar injection				Malicious admission controller		Access Kubernetes dashboard		
							Access tiller endpoint		
							CoreDNS poisoning		
							ARP poisoning and IP spoofing		

= **New technique**

= **Deprecated technique**

https://www.microsoft.com/security/blog/2021/03/23/secure-containerized-environments-with-updated-threat-matrix-for-kubernetes/

Kubernetes – Security concerns

CIS Benchmarks

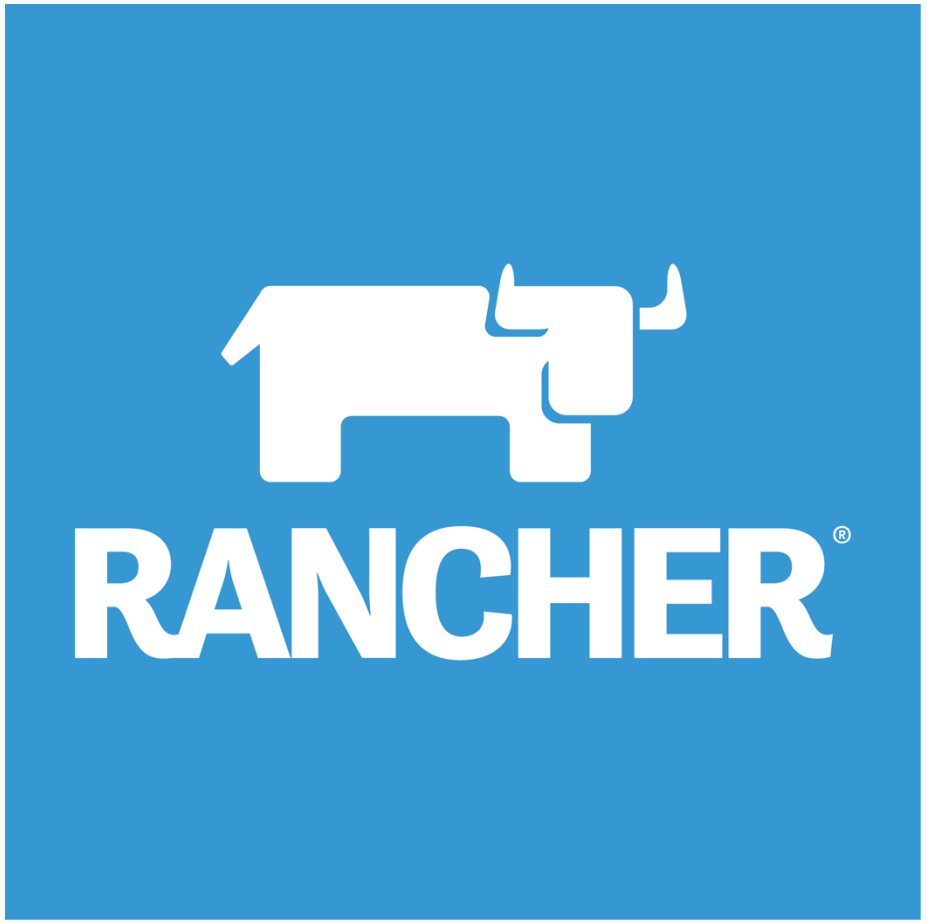
https://www.cisecurity.org/benchmark/kubernetes/

https://github.com/aquasecurity/kube-bench

Kubernetes – instalacja

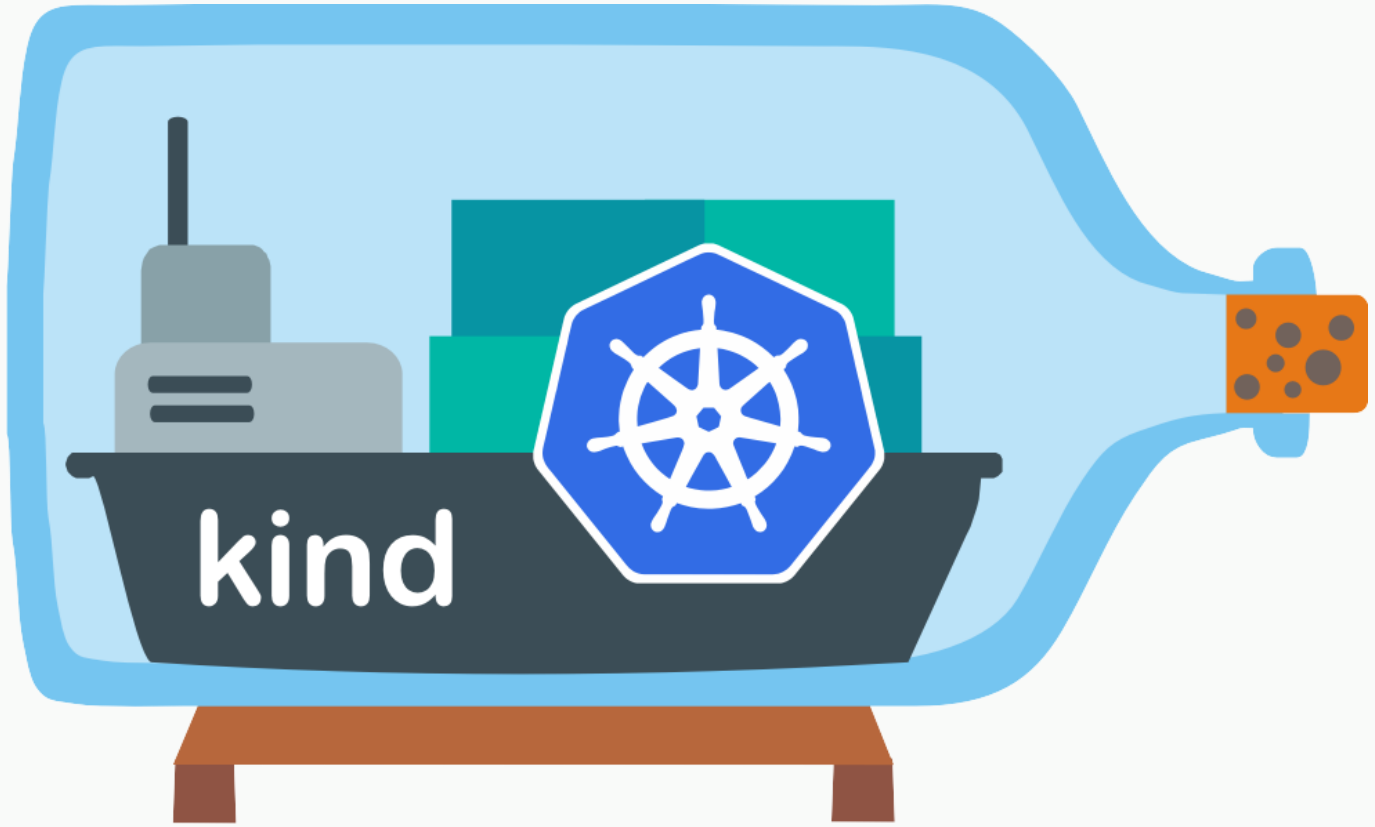
https://kubernetes.io/docs/setup/production-environment/tools/

Kubernetes – instalacja



https://rancher.com/

Kubernetes – kind



https://kind.sigs.k8s.io

Kubernetes – Narzędzia ułatwiające pracę z K8s

https://k8slens.dev

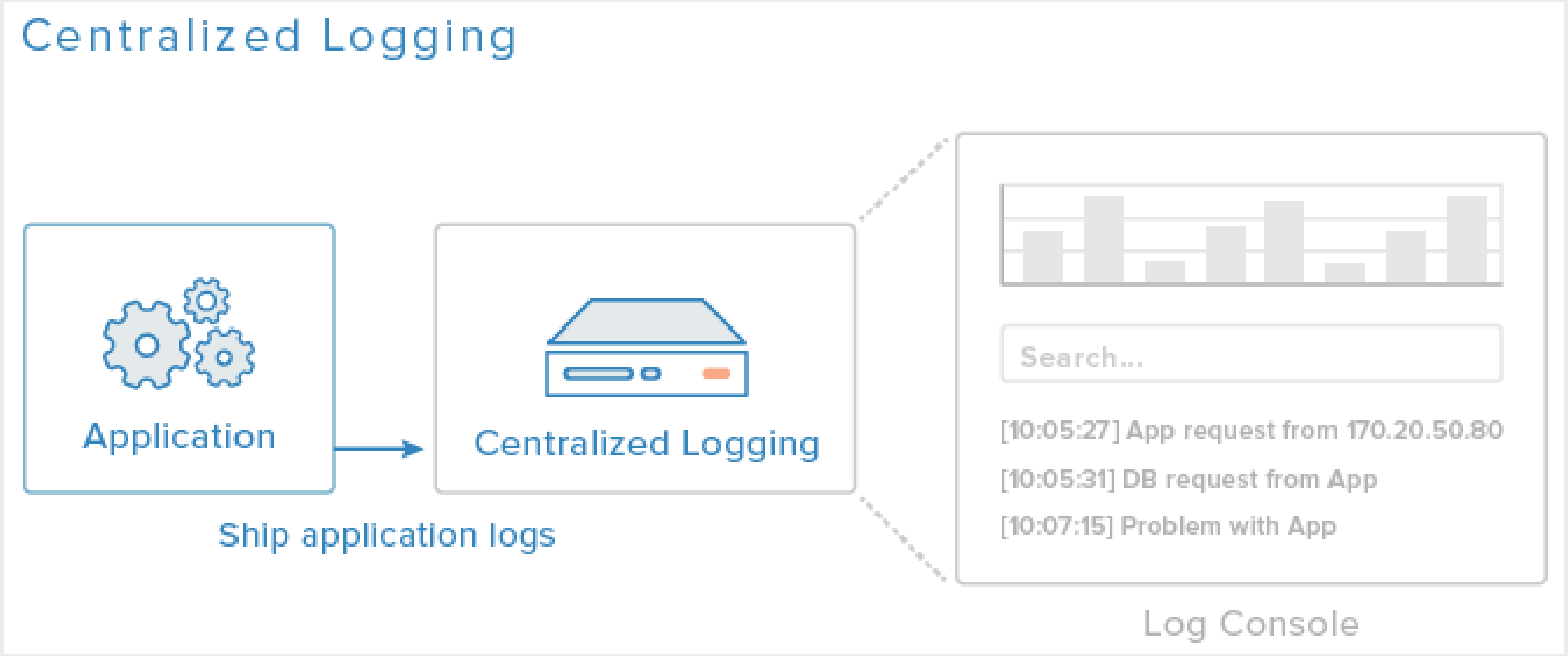
https://www.telepresence.io

Documentation

https://kubernetes.io/docs/home/

Documentation

https://learnk8s.io/production-best-practices


Centralized Logging
Application
Centralized Logging
Ship application logs
Search...
[10:05:27] App request from 170.20.50.80
[10:05:31] DB request from App
[10:07:15] Problem with App
Log Console

Jak przesyłać logi?

- Syslog
- API:
  - Elasticsearch
  - Graylog
  - Loki
  - Rozwiązania chmurowe

Logowanie – podstawowe pojęcia

- Rotacja
- Retencja

Rodzaje logów

- Systemowe (OS)
- Aplikacyjne (nginx, postgres itd.)
- Naszych aplikacji (to co napisliśmy)
- Audytowe

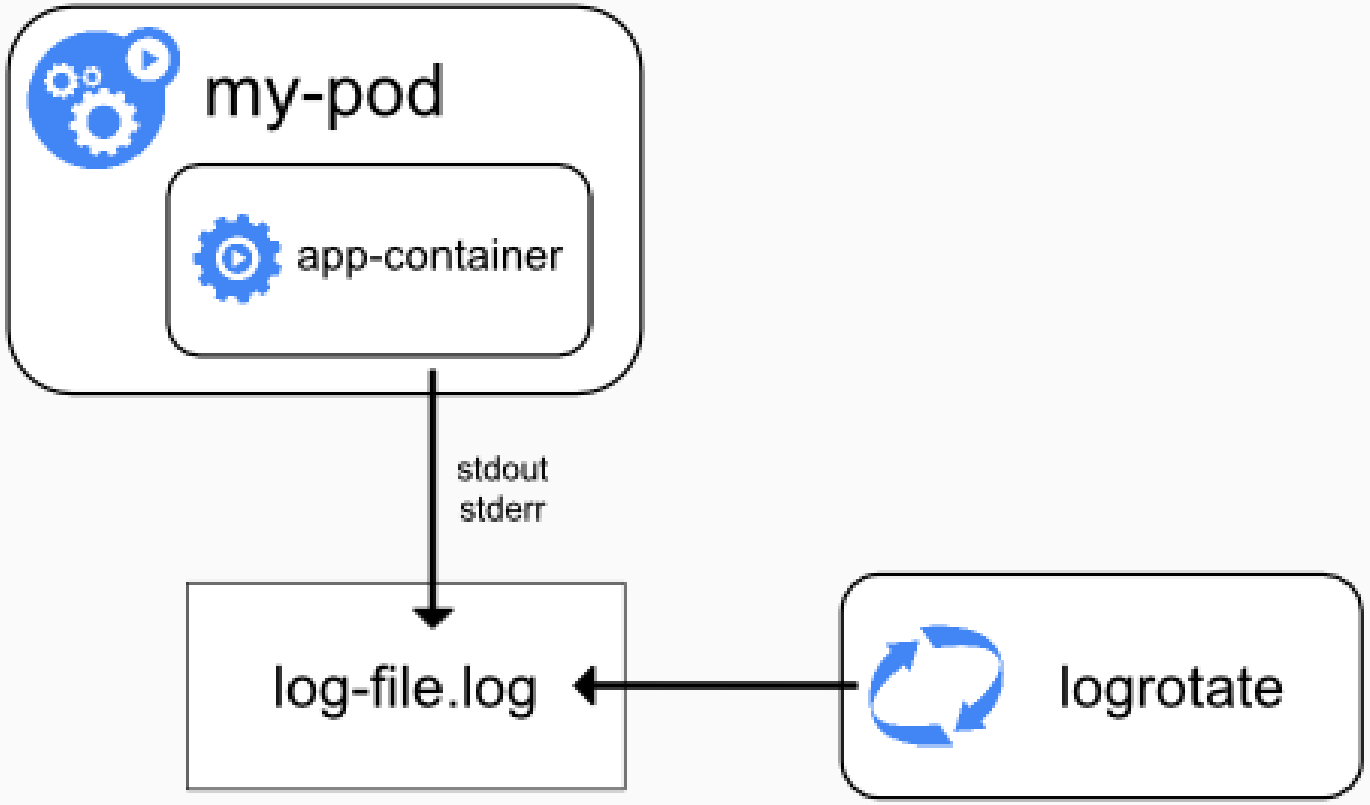
Local or centralized?

Docker - Logging

Driver	Description
none	No logs are available for the container and docker logs does not return any output.
local	Logs are stored in a custom format designed for minimal overhead.
json-file	The logs are formatted as JSON. The default logging driver for Docker.
syslog	Writes logging messages to the syslog facility. The syslog daemon must be running on the host machine.
journald	Writes log messages to journald. The journald daemon must be running on the host machine.
gelf	Writes log messages to a Graylog Extended Log Format (GELF) endpoint such as Graylog or Logstash.
fluentd	Writes log messages to fluentd (forward input). The fluentd daemon must be running on the host machine.
awslogs	Writes log messages to Amazon CloudWatch Logs.
splunk	Writes log messages to splunk using the HTTP Event Collector.
etwlogs	Writes log messages as Event Tracing for Windows (ETW) events. Only available on Windows platforms.
gcplogs	Writes log messages to Google Cloud Platform (GCP) Logging.
logentries	Writes log messages to Rapid7 Logentries.

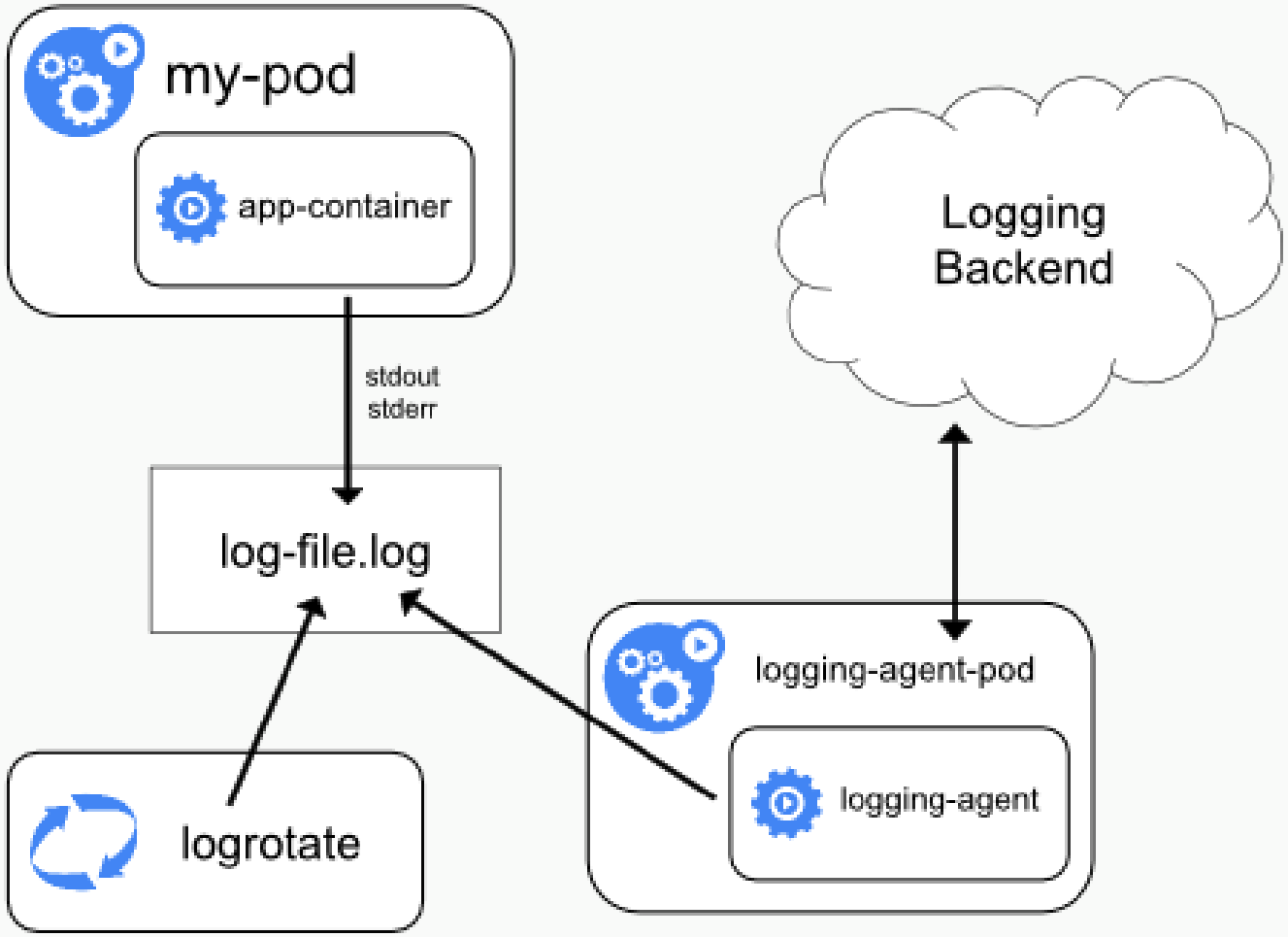
https://docs.docker.com/config/containers/logging/configure/

Kubernetes
- Logging

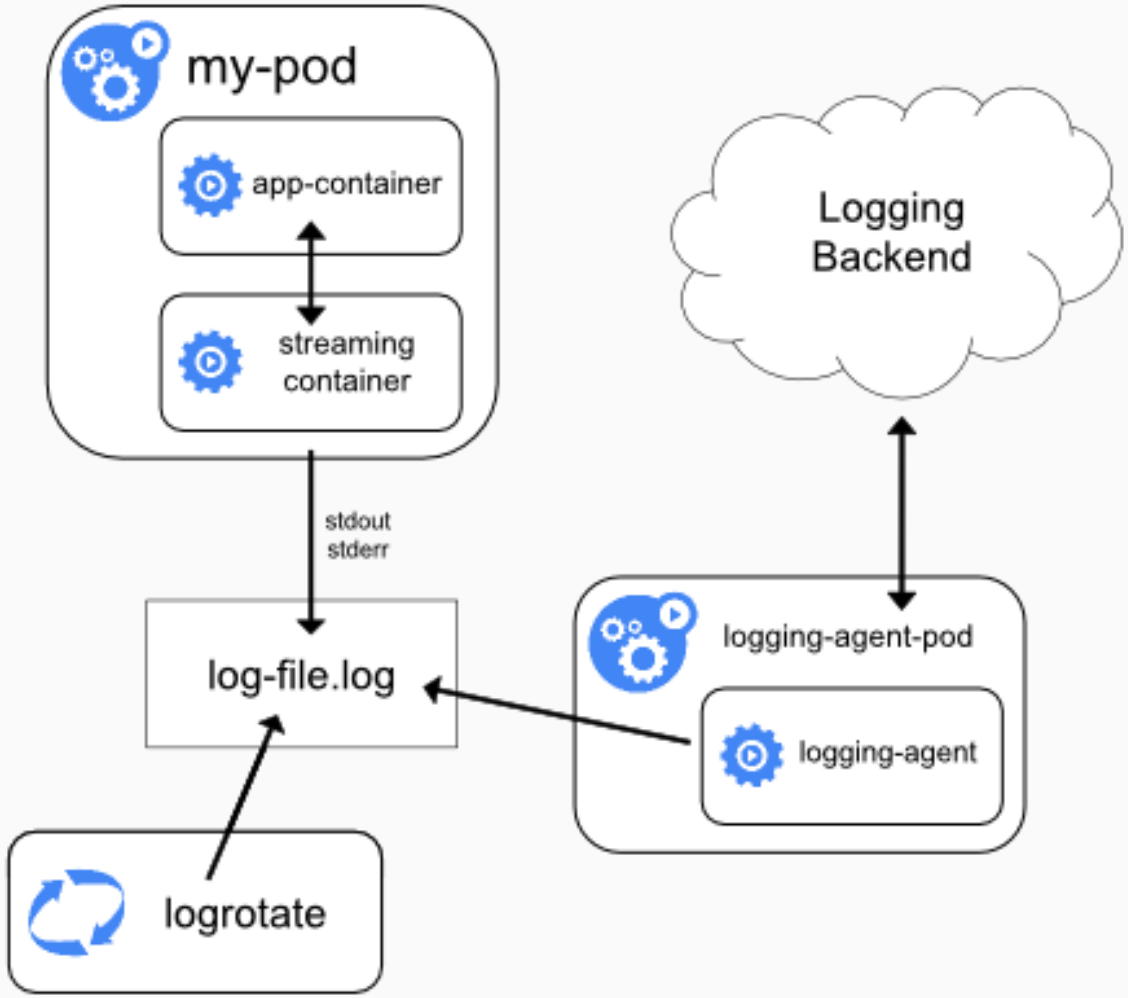


Kubernetes

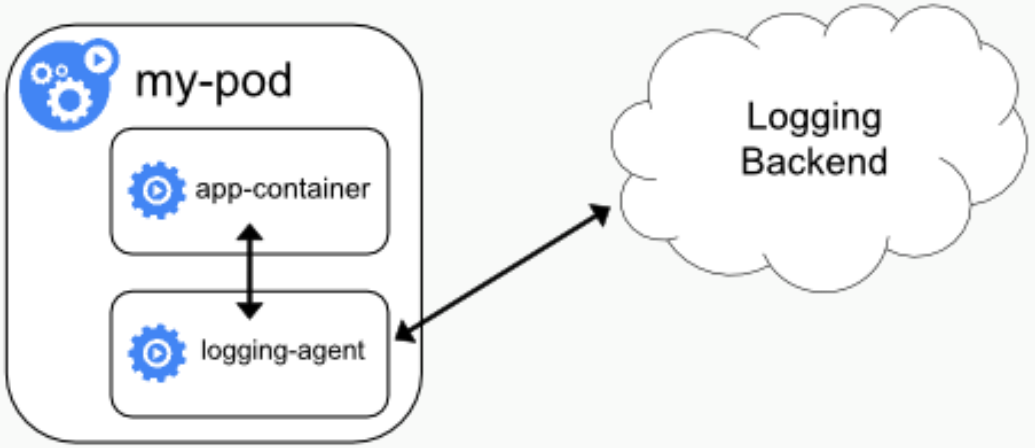
- Logging

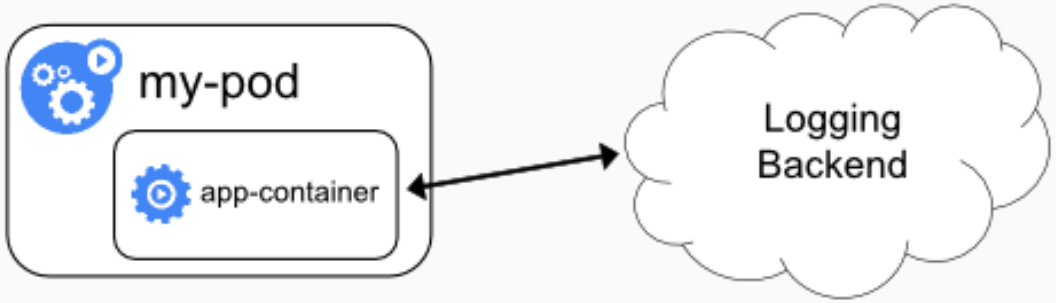


Kubernetes
- Logging



Kubernetes
- Logging





Elasticsearch

>> logging server

- Logs
- Metrics
- APM
- Uptime
- Site Search
- App Search
- Workspace Search
- Maps
- SIEM
- Endpoint Security

https://www.elastic.co/elasticsearch/

Elasticsearch

So what is wrong?

Elasticsearch - Architecture

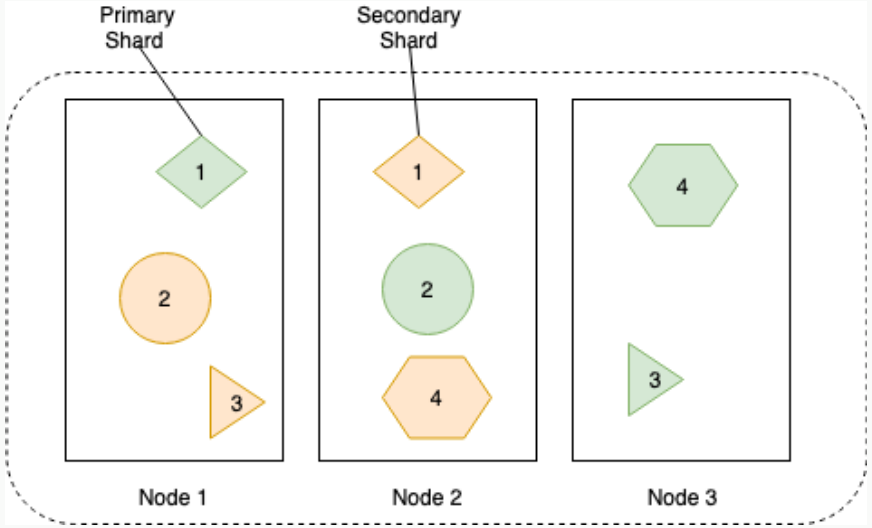
There are following type of nodes in the cluster:

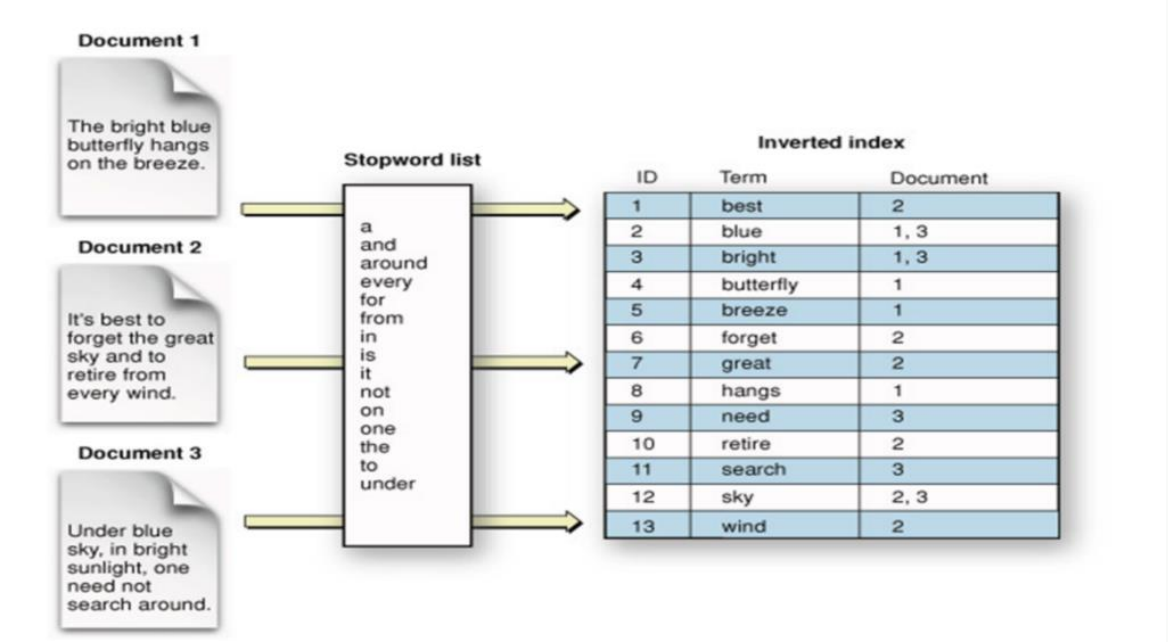
- Master Nodes – controls the cluster, requires a minimum of 3, one is active at all times
- Data Nodes – to hold index data and perform data-related tasks
- Ingest Nodes – used for ingest pipelines to transform and enrich the data before indexing
- Coordinating Nodes – to route requests, handle search reduce phase, coordinates bulk indexing
- Alerting Nodes – to run alerting jobs
- Machine Learning Nodes – to run machine learning jobs

Elasticsearch

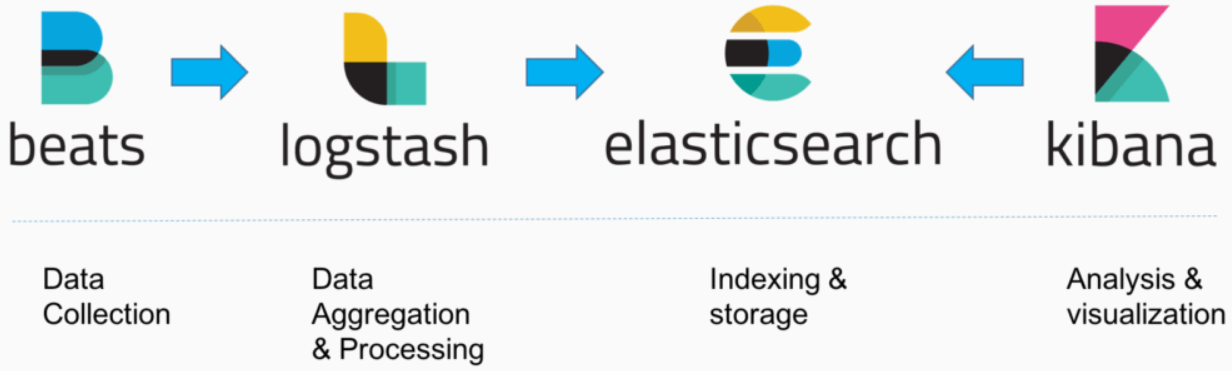
-

Architecture





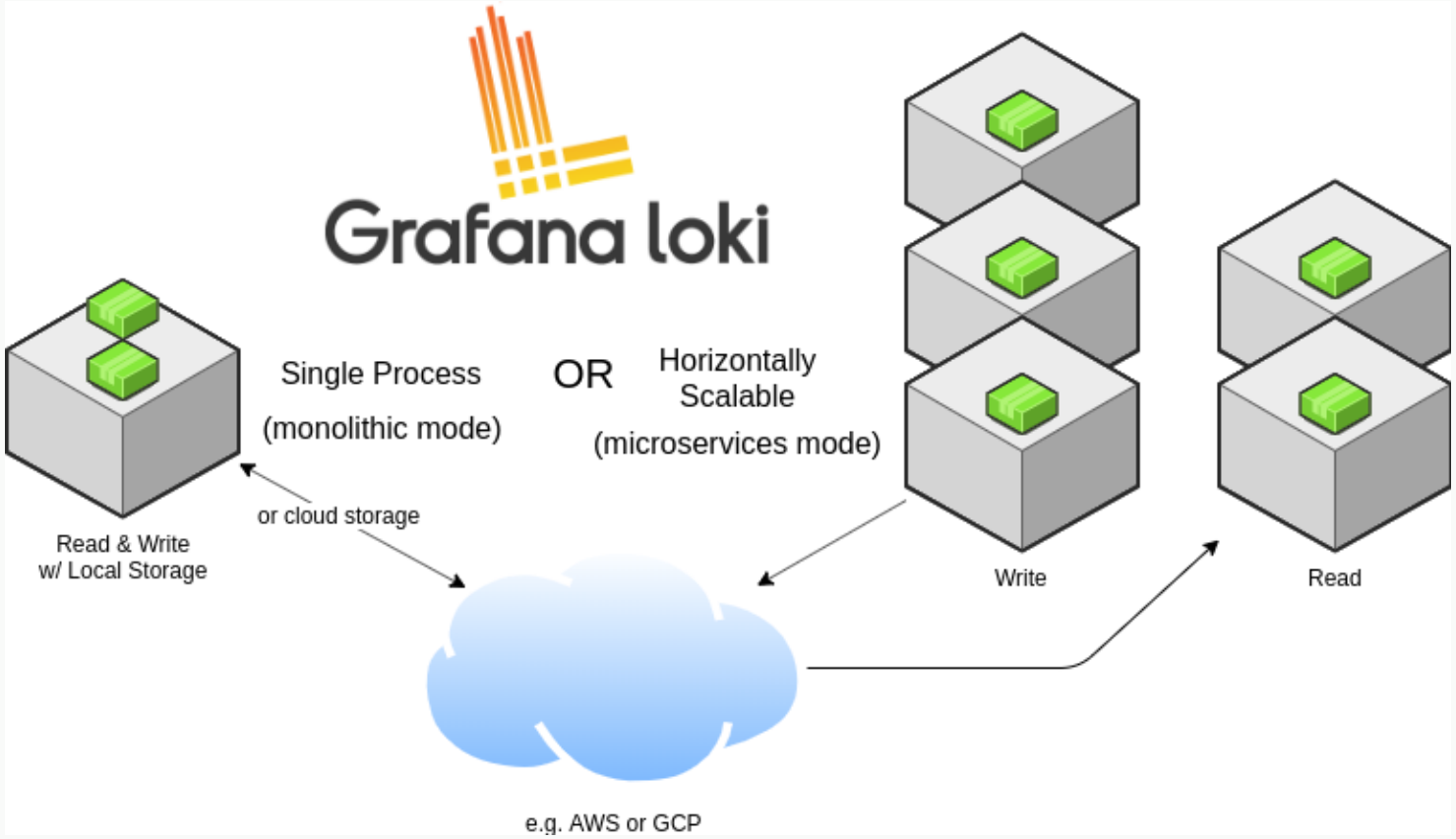
ELK, EFK?


beats
logstash
elasticsearch
kibana
Data Collection
Data Aggregation & Processing
Indexing & storage
Analysis & visualization

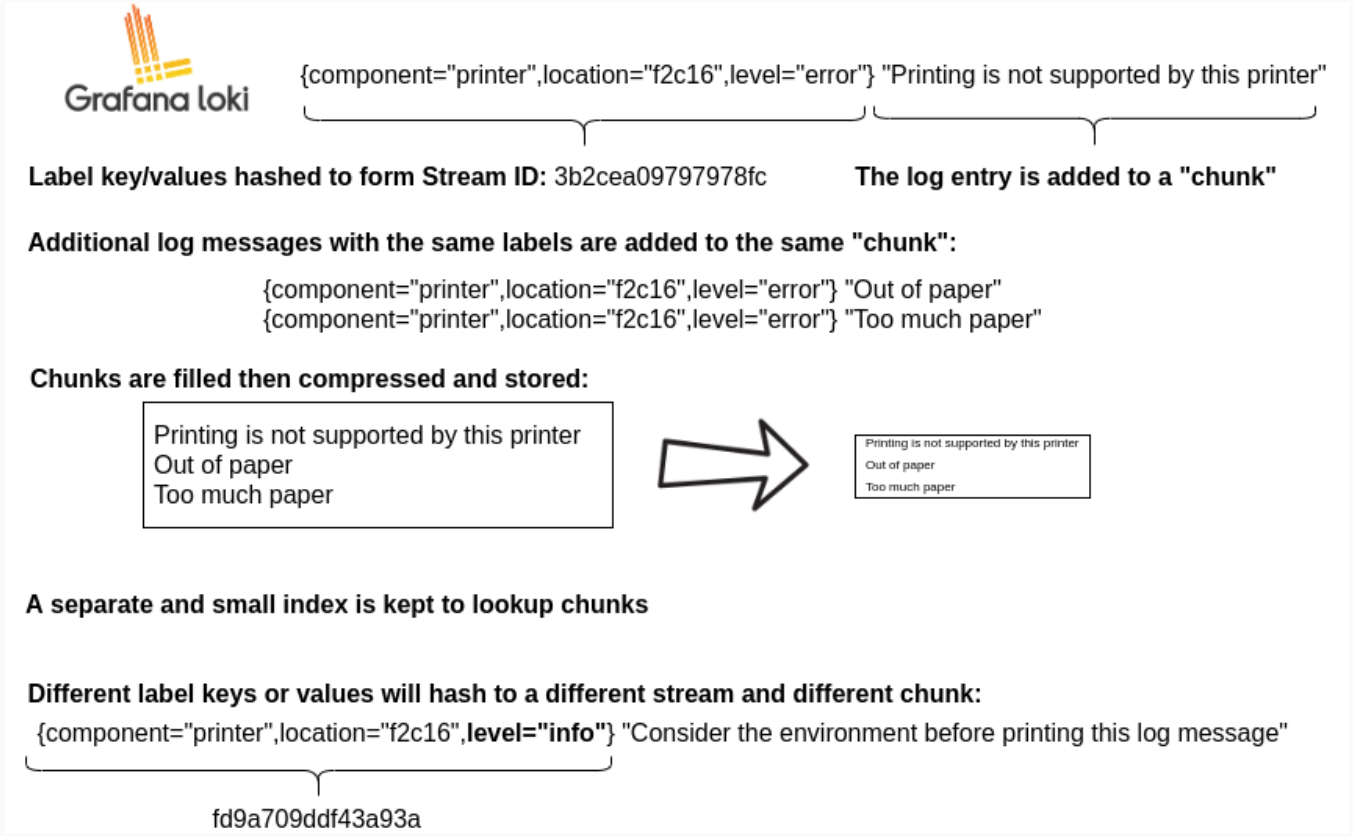

fluentd
Elastic Search
Kibana
Logs

LOKI

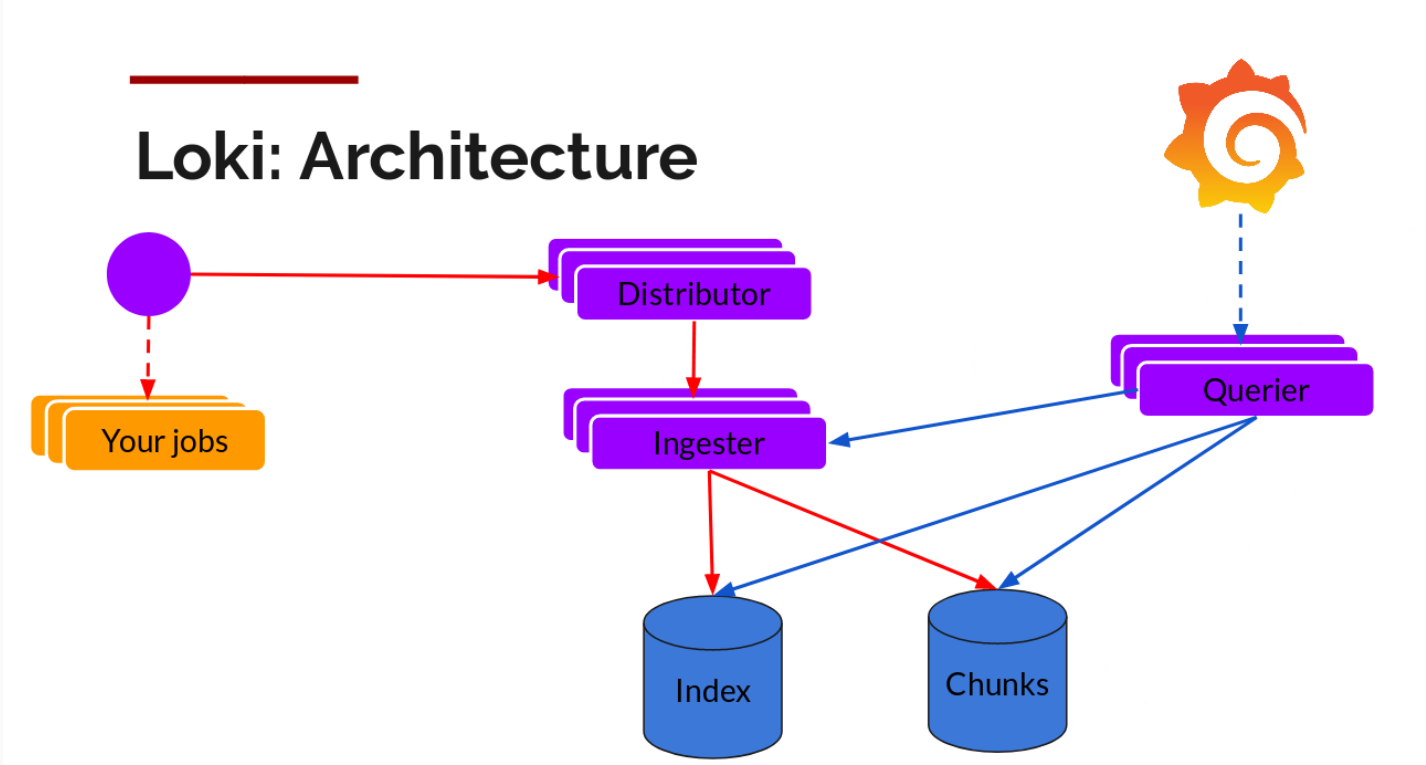
LOKI - Architecture



LOKI - Architecture



LOKI - Architecture



LOKI - LogQL

https://grafana.com/docs/loki/latest/logql/

Elasticsearch vs LOKI (only from logging perspective)

ITEM	Elasticsearch	Loki
Query Language	DSL and Lucene – very powerfull with extensive operator support	LogQL – inspired by Prometheus, basic operator support
Scalability	Support	Support – but better due to microservice architecture. You can easely seperate read from write. Can consume much bigger ammount of data.
Multi-tenancy	one index per tenant, tenant-based routing, using unique tenant fields, and use of search filters	X-Scope-OrgId in the HTTP header request
Cost	$$$$$$$$$$$$$$....	Extreamly cost effective

Monitoring

Monitoring

1. Po co monitorować?
2. Co monitorować?

Monitoring – co monitorować?

Wszystko

Monitoring – co monitorować?

- Fizyczne elementy:
  - Serwery
  - Urządzenia sieciowe
- Sieć:
  - Dostępność (z zewnątrz)
  - Opóźnienia/straty/jitter
- OS:
  - HDD
  - CPU
  - RAM, itd.
- Aplikacje:
  - Czy jest uruchomiona
  - Czy odpowiada
  - Czy działa
  - Metryki
  - Logi
  - Performance
- Trendy

Monitoring – co monitorować?

- Fizyczne elementy:
  - Serwery
  - Urządzenia sieciowe
- Sieć:
  - Dostępność (z zewnątrz)
  - Opóźnienia/straty/jitter
- OS:
  - HDD
  - CPU
  - RAM, itd.
- Aplikacje:
  - Czy jest uruchomiona
  - Czy odpowiada
  - **Czy działa**
  - **Metryki**
  - **Logi**
  - **Performance**
- **Trendy**

Monitoring – co monitorować?

A co jeżeli nie wiemy co monitorować w aplikacji?

RED

Monitoring - sposoby

Ze względu na sposób:
- Pull
- Push

Ze względu na miejsce:
- Centralnie
- Rozproszenie (np. poprzez agentów)

Monitoring - narzędzia

Open-source (legacy):
- Zabbix
- Nagios
- Icinga
- Sensu

Open-source (new):
- Prometheus
- Jaeger/Tempo
- Telepresence

SaaS:
- DataDog
- NewRelic

Monitoring - Zabbix

Monitoring - Zabbix

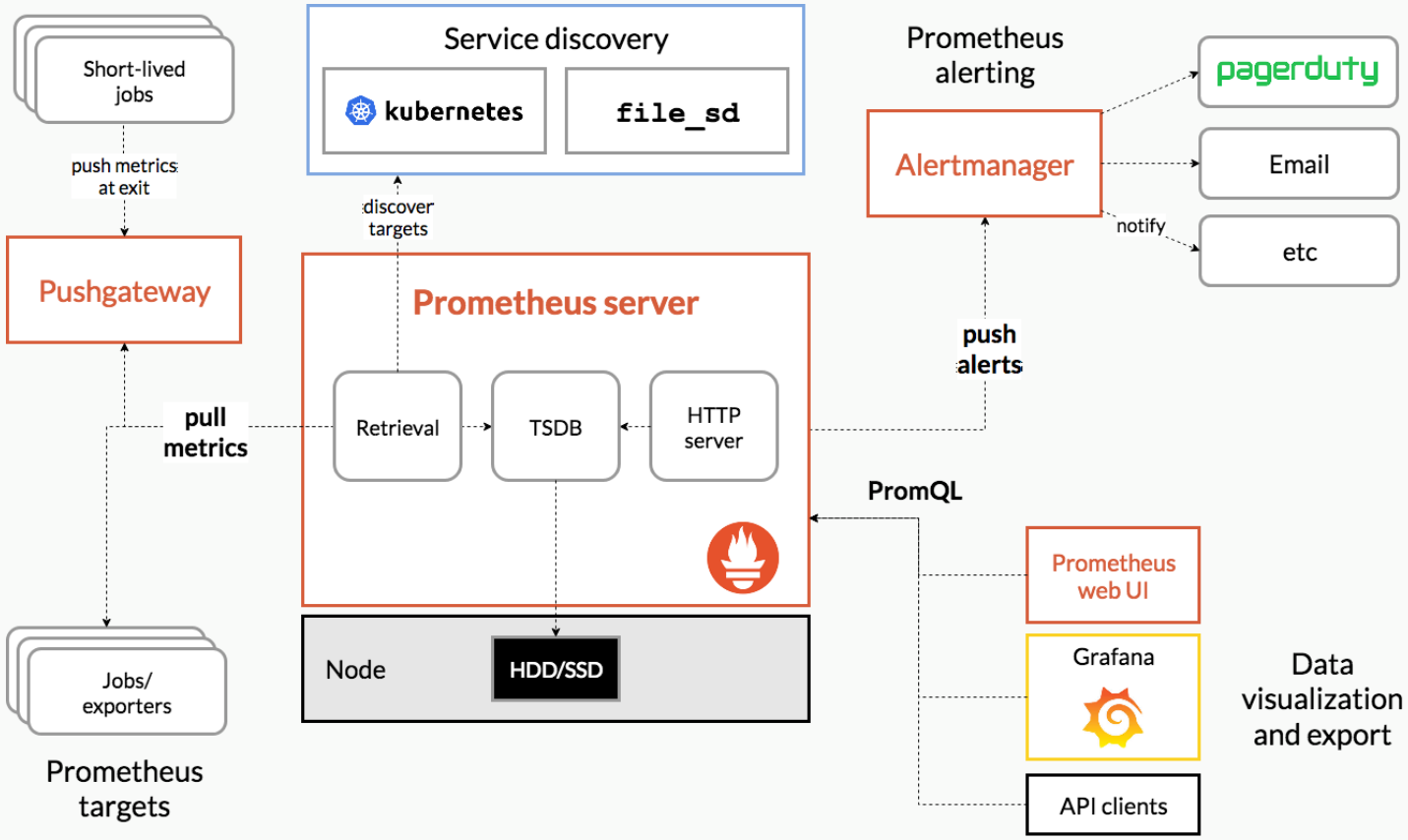
Why Zabbix?

- Data gathering
- Highly configurable alerting
- Web monitoring capabilities
- Extensive visualization options
- Historical data storage
- Easy configuration
- Use of templates
- Network discovery
- Fast web interface
- Zabbix API
- Permissions system
- Full featured and easily extensible agent
- Binary daemons
- Ready for complex environments

Monitoring – Zabbix – Installation

https://www.zabbix.com/documentation/current/manual/installation

Monitoring – Prometheus - Architecture



Monitoring – Prometheus - endpoints

```
<metric name>{<label name>=<label value>, ...}
```

```
api_http_requests_total{method="POST", handler="/messages"}
```

Monitoring – Prometheus - querying

https://prometheus.io/docs/prometheus/latest/querying/basics/

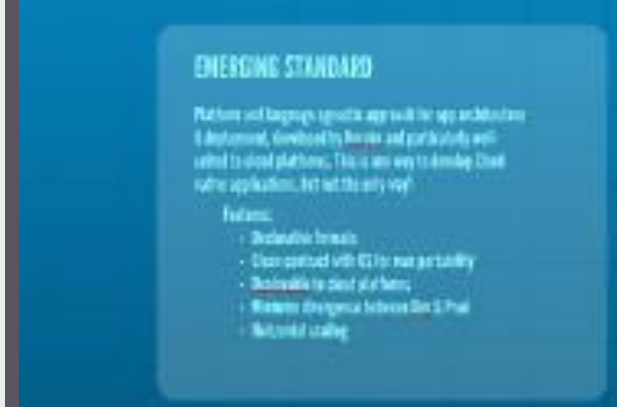
Monitoring – Prometheus – important topics to consider

- Security
- Retention period (default 15d)

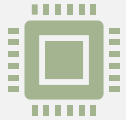
https://prometheus.io/docs/prometheus/latest/storage/

The 12 Factor App

Design Considerations in the World of Containers



12th factor app for SaaS applications



Agnostic? What is that?

What is Software as a Service

Declarative approach

Contract

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)

Explicitly declare and isolate dependencies

Each application declares all dependecies

Application is not dependent on „not listed” dependencies

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

Store config in the environment

Resource handles to the database

Credentials to external services such as Amazon S3 or Twitter

Per-deploy values such as the canonical hostname for the deploy

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

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

Use ENV variables to set configuration

Treat backing services as attached resources

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

Service can be easily attached and detached

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

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

Export services via port binding

Attach PORT to each possible application

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

Scale out via the process model

Consider that application will be scalled horizontaly by increasing number of processes

Don’t start process in background

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

Keep development, staging, and production as similar as possible

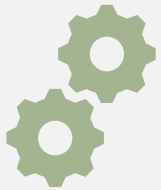
developers who wrote code are closely involved in deploying it and watching its behavior in production.

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

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.

Run admin/management tasks as one-off processes

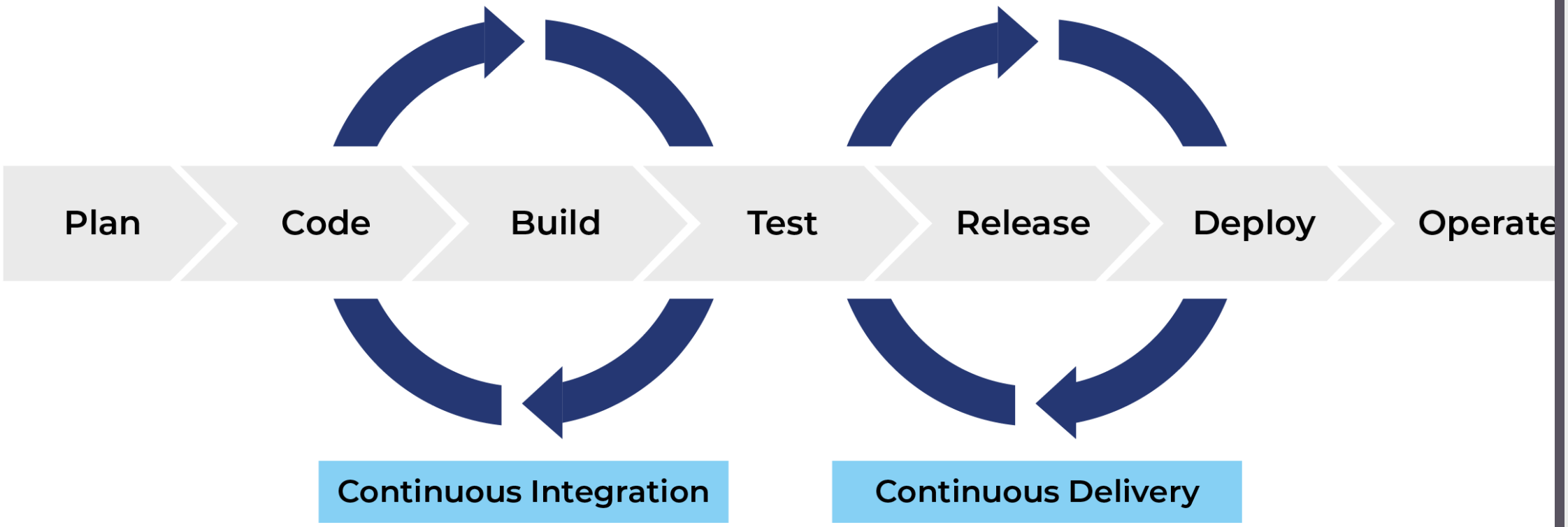


One-off admin processes should be run in an identical environment as the regular long-running processes of the app. They run against a release, using the same codebase and config as any process run against that release.

Documentation

https://12factor.net

CI/CD


Plan
Code
Build
Test
Release
Deploy
Operate
Continuous Integration
Continuous Delivery

CI/CD Pipleline

Split State from Code

Code repository

State repository (k8s yamls)

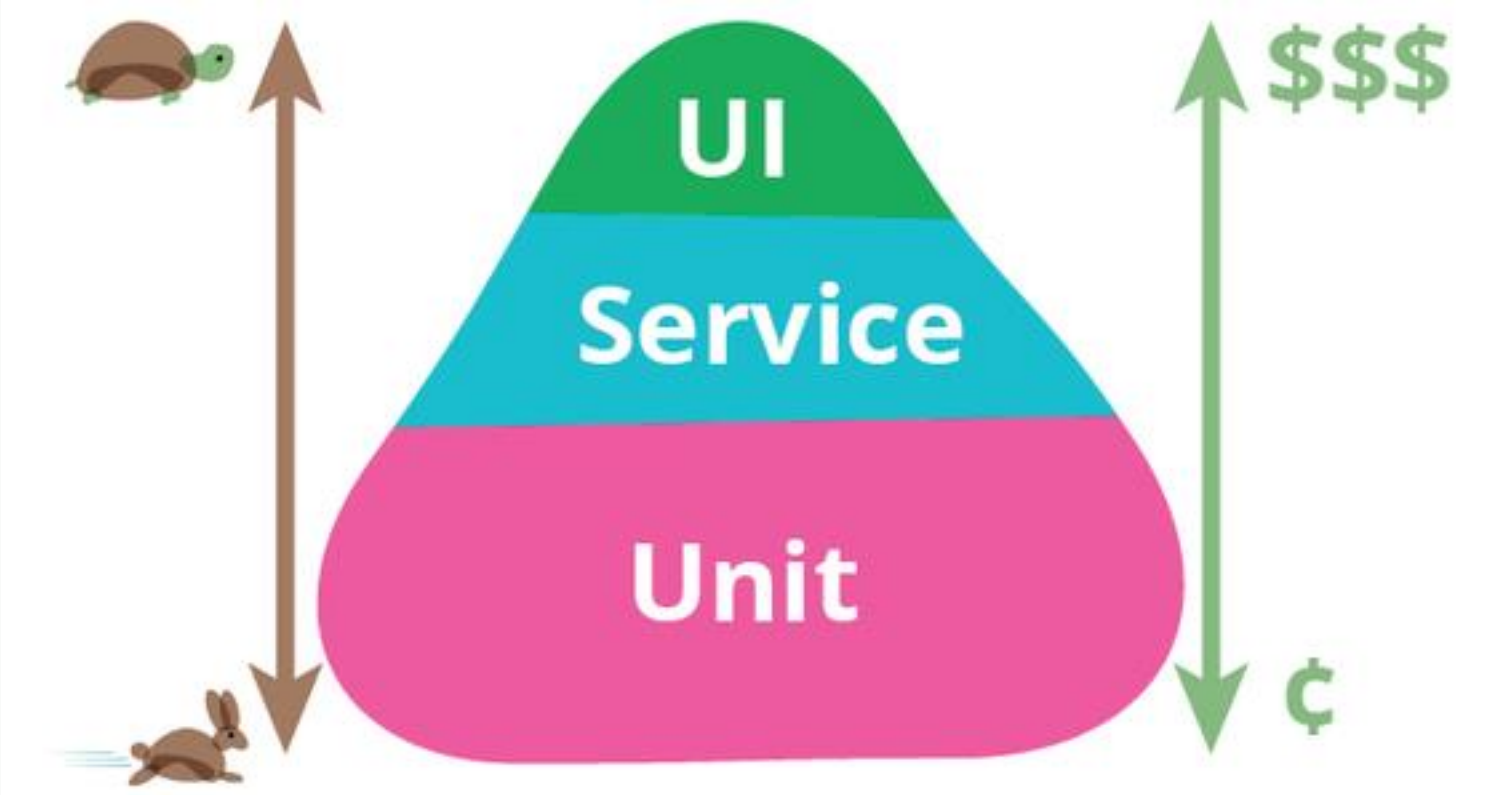
Master is the TRUTH

Disable push to master (only PR)

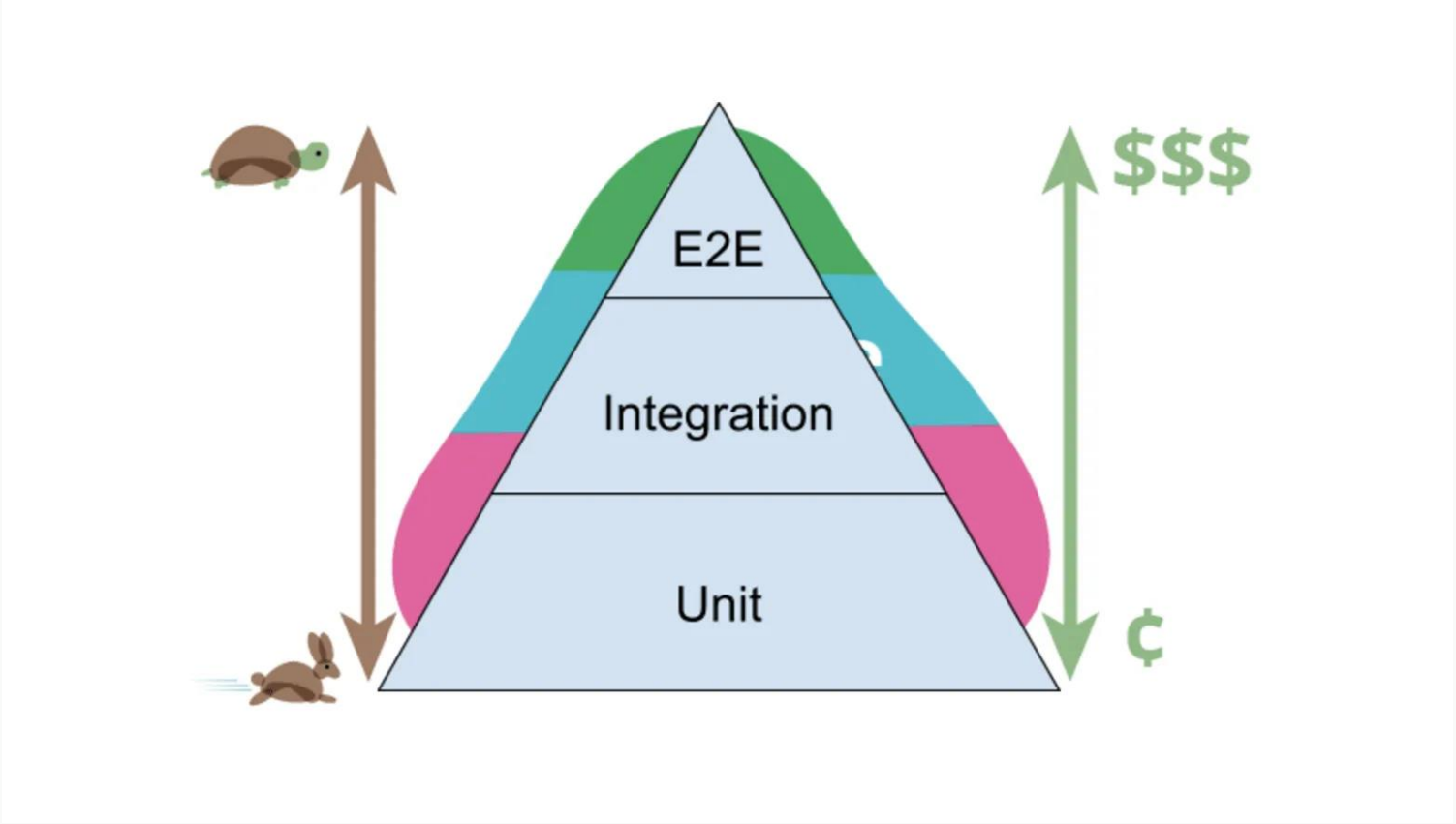
Consider yamllint

Place pipeline into repository

CI/CD Pipleline



CI/CD Pipleline

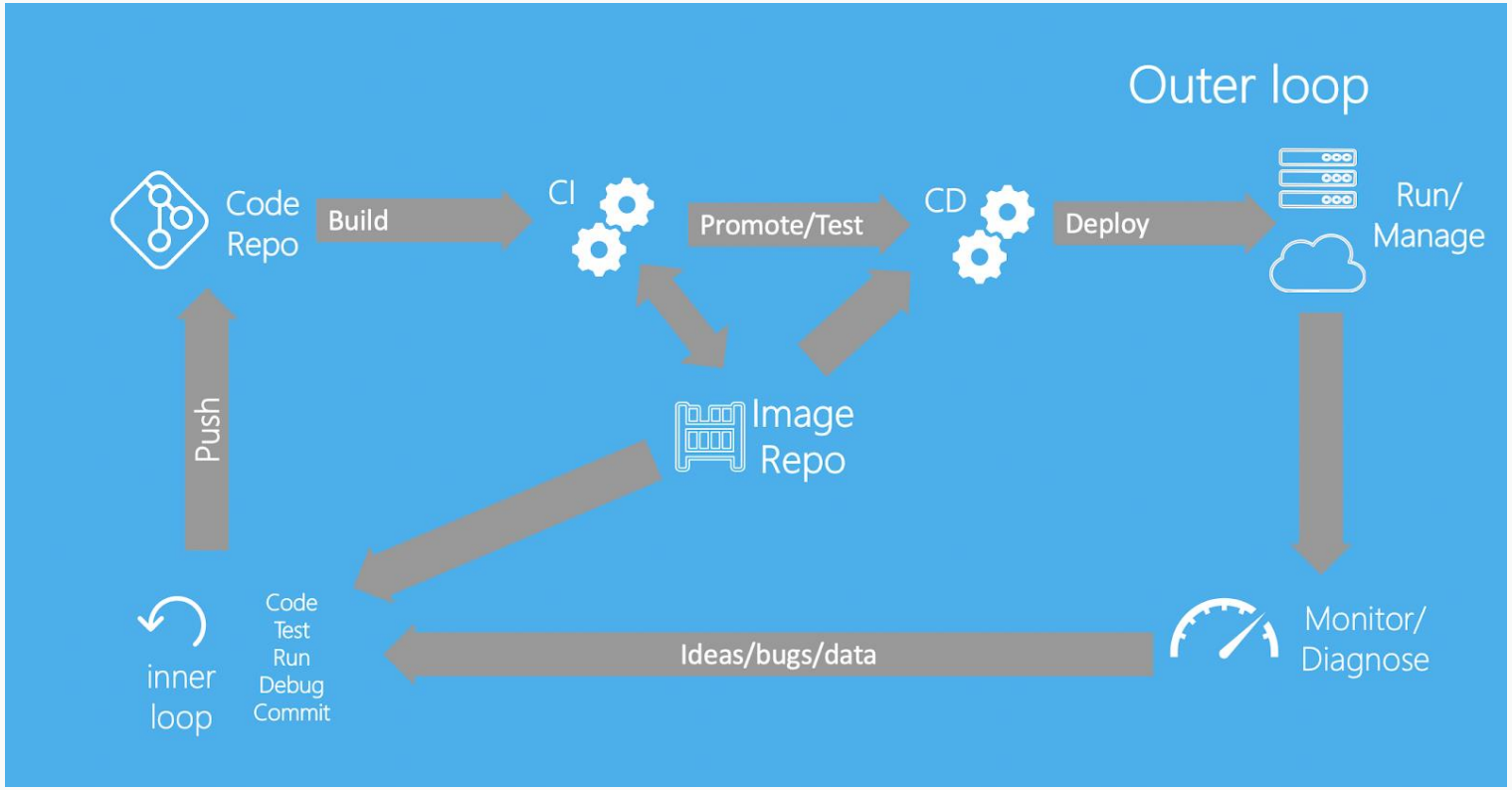


CI/CD Pipleline

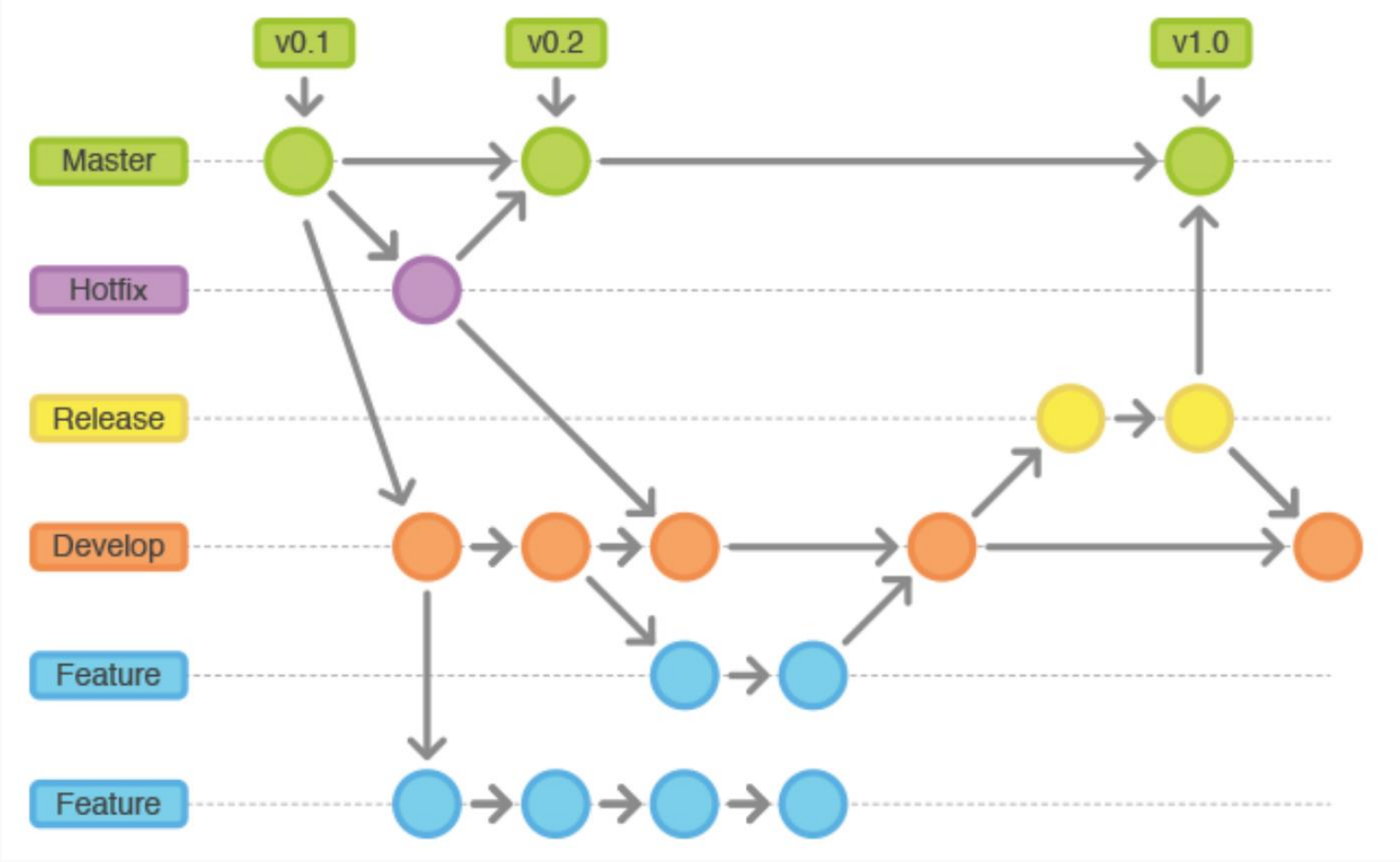
CI/CD Pipleline

CI vs CD?

CI/CD Pipleline



CI/CD Pipleline



CI/CD Pipleline

HELM / Kustomize

Consider only templating

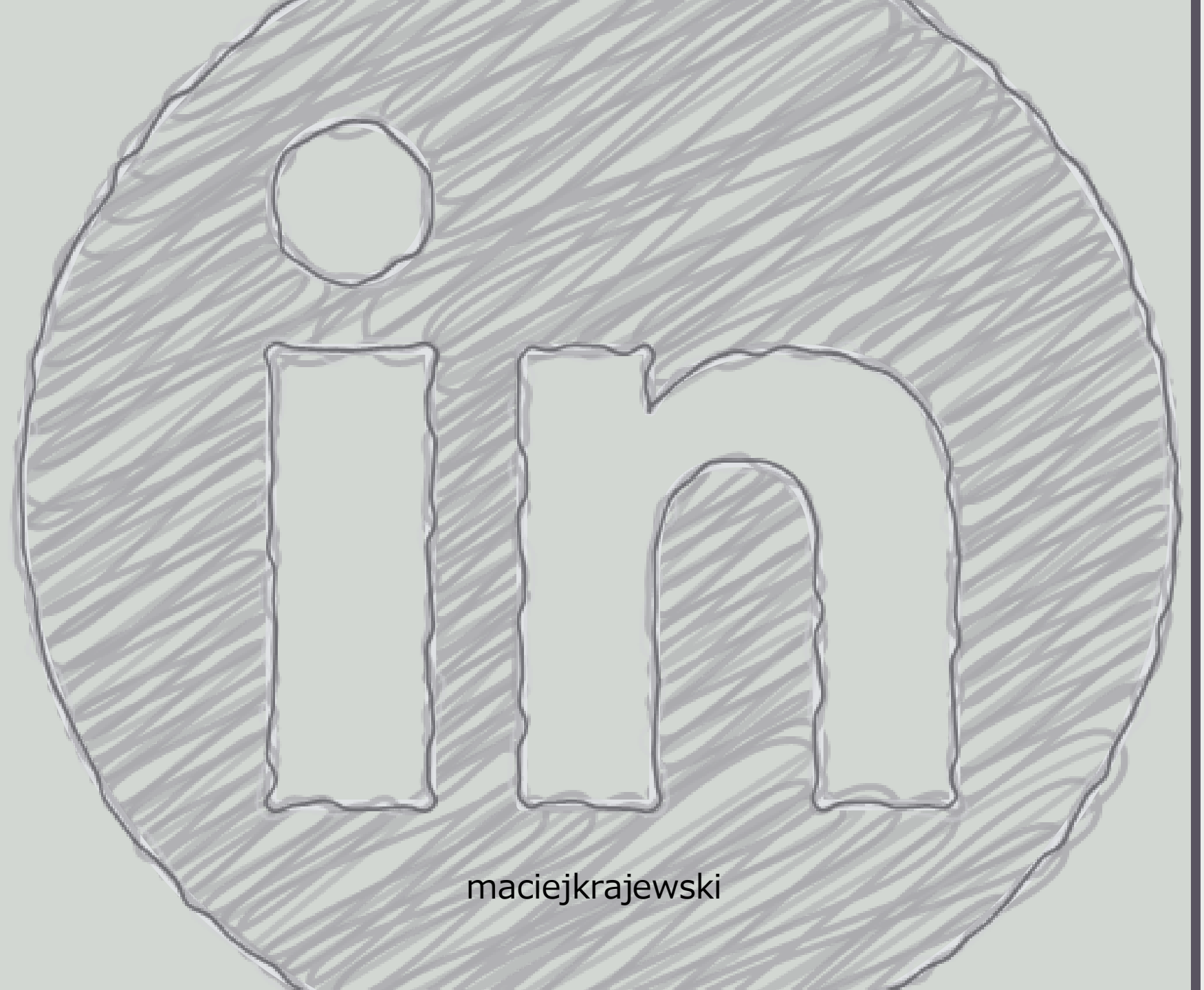
Kubectl apply –prune –namespace mobees –l app=mob

Own images

Own templates

Security

CI/CD
HELM
&
KUSTOMIZE



Dziękuję za uwagę