Zurich University of Applied Sciences

zh aw School of Engineering
InIT Institute of Applied Information Technology

Zurich University of Applied Sciences (CH)

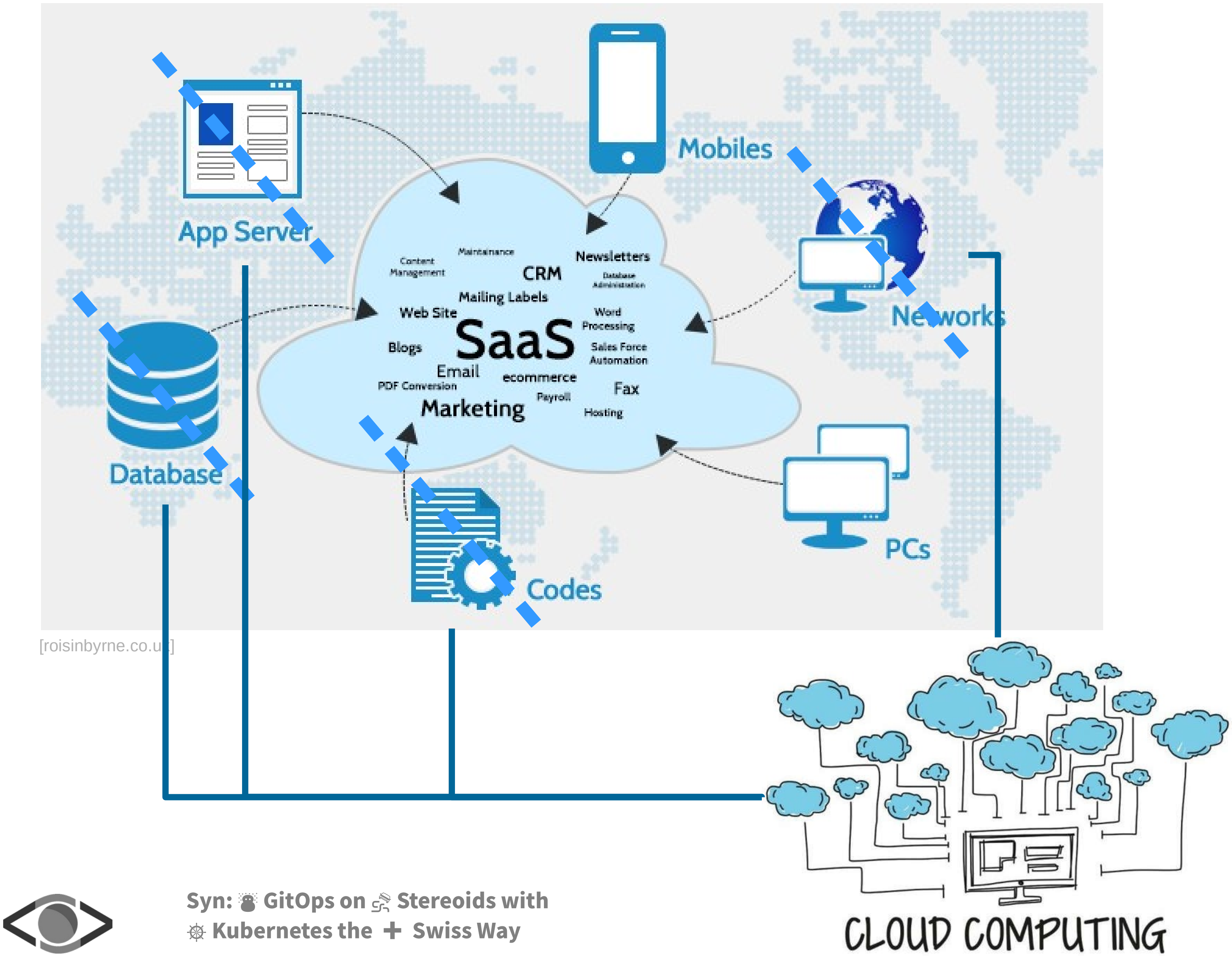
VSHN - The DevOps Company (CH)

Syn: GitOps on Stereoids with ☸ Kubernetes the 🇨🇭 Swiss Way

Josef Spillner, Daiana Boruta, Tobias Brunner, Simon Gerber, Adrian Kosmaczewski

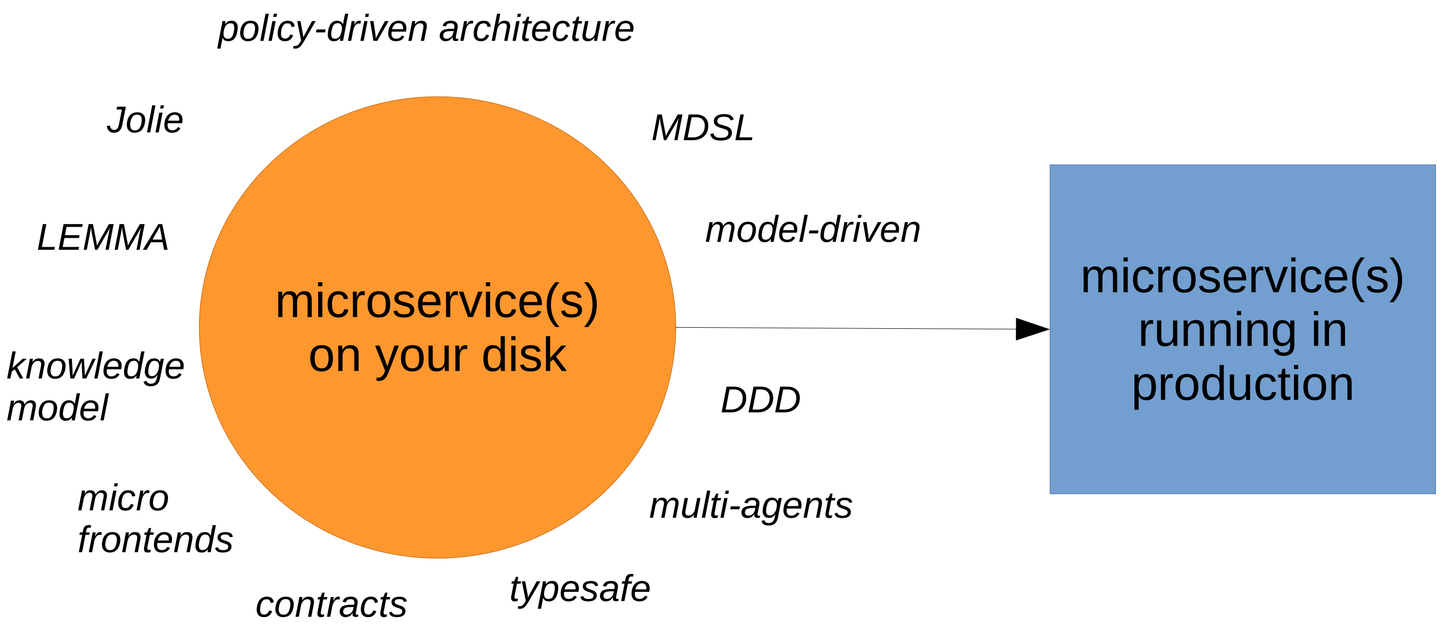
Sep 8-10, 2020 | Microservices | ~~Bologna~~ Online

Managed Cloud Applications



[roisinbyrne.co.uk]

Lifting Microservices → Cloud Apps



policy-driven architecture

Jolie

MDSL

model-driven

LEMMA

microservice(s) on your disk

microservice(s) running in production

knowledge model

DDD

micro frontends

multi-agents

contracts

typesafe

Bringing apps into the cloud? Easy.

1. Download app
2. Oops, prerequisite: Install Kubernetes
    - Find right flavour. (K8s, K3s, µK8s, Minikube, EdgeKube, Kind, FLEDGE, ...)
    - Set cluster + extensions up the right way. (K3d, K9s, Fission, ...)
    - Set up namespaces, document them, adhere to docs...
    - Ensure networking is up. Depending on flavour. (traefik, ...)
    - Do something with secrets. (vault, ...)
    - Deploy app.
    - Oops, wrong parameters. Find where to change. (helm, ...)
    - Test. (What is the IP address again?)
    - Oops, wrong version. Undeploy, deploy again.
    - Could we not automate that? (argocd, ...)
    - Fiddle with Git commit hooks.
    - ...

3a. Run out of time

3b. Pay somebody to run out of time.

3c. Come up with a smarter solution.

Syn - Concepts & Innovation

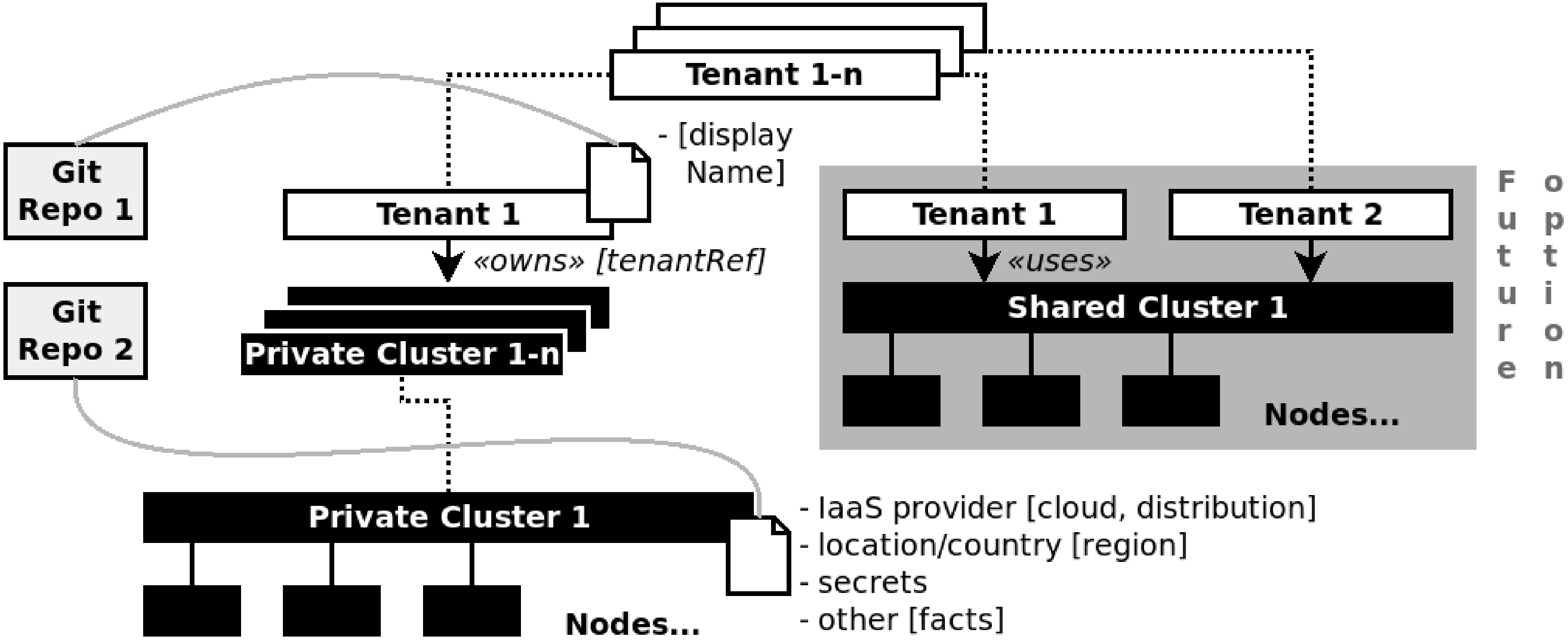
Bring your own cloud (BYOC).
Customisation grows with the application project needs.
Redirect platform endpoints flexibly.

Explicit local mode.
Check and validate configuration before breaking something.

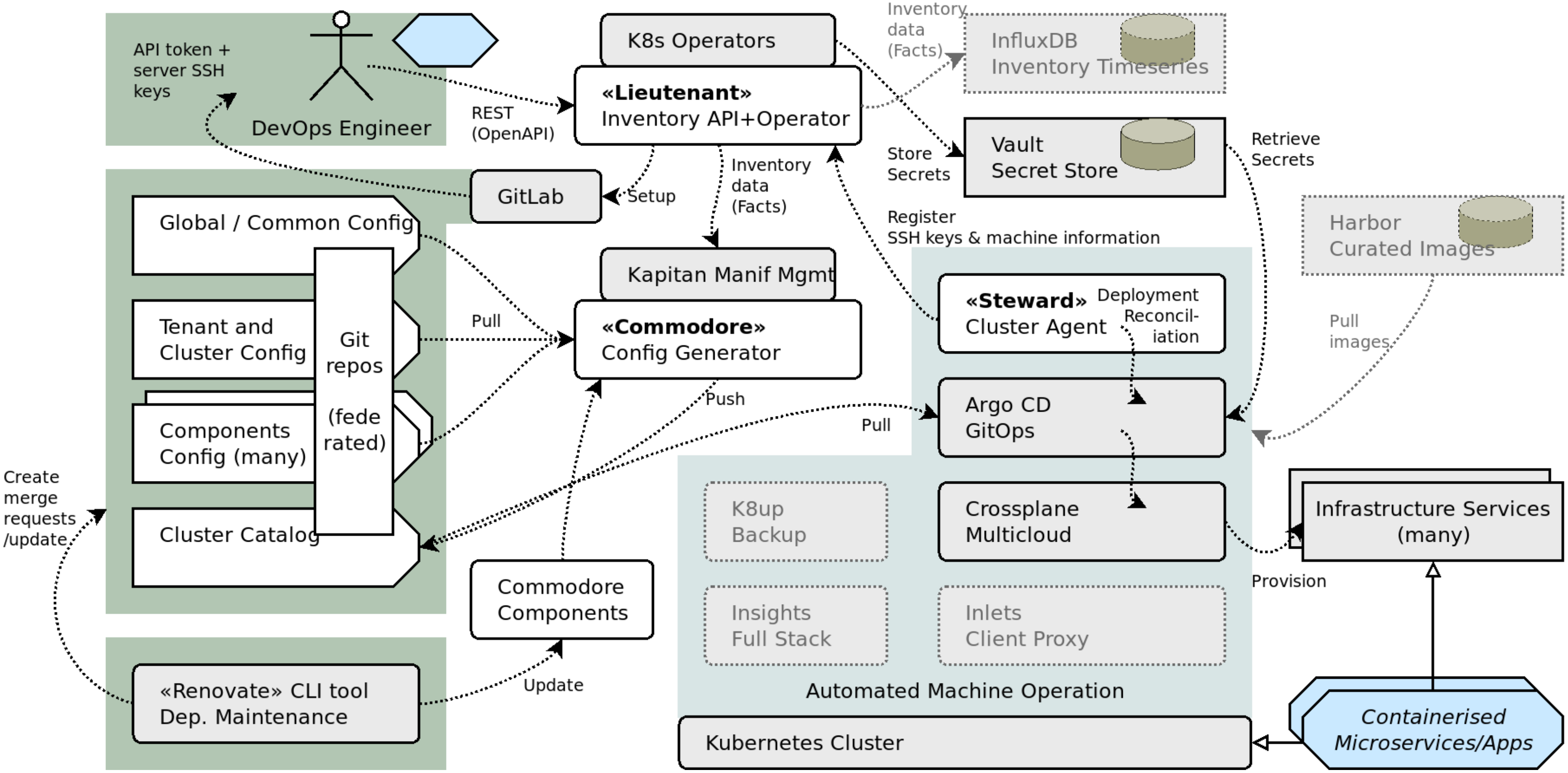
Strong multi-cloud support.
Deploy to arbitrary combinations of private/public clusters.
Care less about inhomogeneous infrastructure.

Hierarchical code-driven development.
Attach Git repos as needed. Hook into hierarchy config trees.

Syn - System Model



Syn - Architecture



API token + server SSH keys

DevOps Engineer

REST (OpenAPI)

K8s Operators

«Lieutenant» Inventory API+Operator

Inventory data (Facts)

InfluxDB Inventory Timeseries

Store Secrets

Vault Secret Store

Retrieve Secrets

GitLab

Setup

Inventory data (Facts)

Register SSH keys & machine information

Harbor Curated Images

Global / Common Config

Tenant and Cluster Config

Components Config (many)

Cluster Catalog

Git repos (federated)

Pull

Kapitan Manif Mgmt

«Commodore» Config Generator

Push

Pull

«Steward» Cluster Agent

Deployment Reconciliation

Argo CD GitOps

Pull images

Create merge requests /update

K8up Backup

Crossplane Multicloud

Infrastructure Services (many)

Provision

Commodore Components

Insights Full Stack

Inlets Client Proxy

«Renovate» CLI tool Dep. Maintenance

Update

Automated Machine Operation

Kubernetes Cluster

Containerised Microservices/Apps

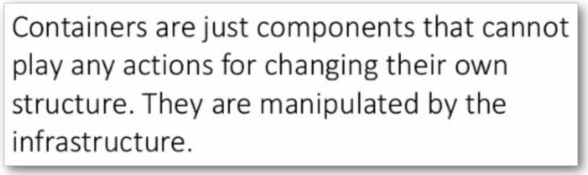
Lieutenant API

API objects & methods

- **bootstrapping:**
  - **installSteward (per machine; incl. ArgoCD)**
- **cluster:**
  - **create, delete, get, list, update**
- **inventory:**
  - **update, query**
- **tenant:**
  - **create, delete, get, list, update (incl. Git repo property)**
- **system:**
  - **docs, healthz, openapi**

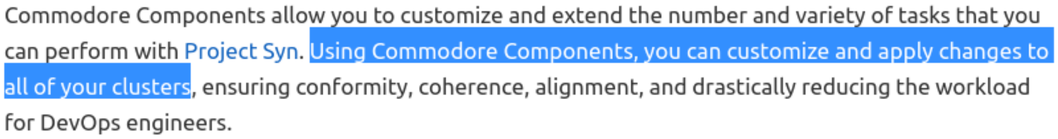
Commodore Components

From Claudio`s talk on autonomic microservices:



Containers are just components that cannot play any actions for changing their own structure. They are manipulated by the infrastructure.

True for most platforms, but no longer (necessarily) with Syn:



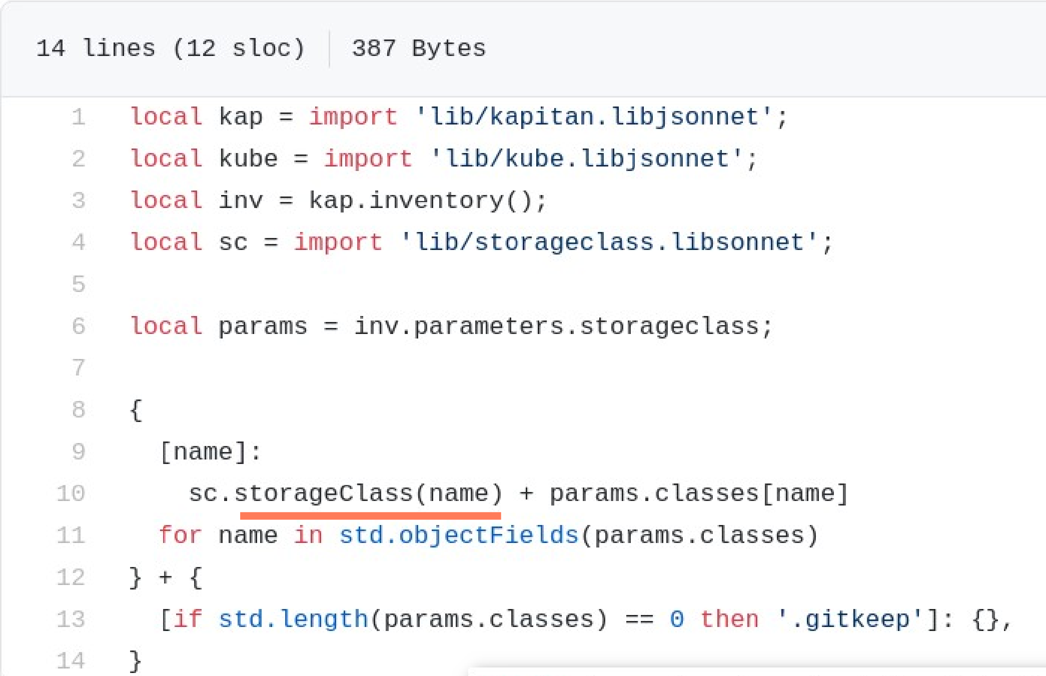
Commodore Components allow you to customize and extend the number and variety of tasks that you can perform with Project Syn. Using Commodore Components, you can customize and apply changes to all of your clusters, ensuring conformity, coherence, alignment, and drastically reducing the workload for DevOps engineers.

How?

- Based on all YAML/JSON manifest information
- Jsonnet templating language
  - extensions to JSON: variables, functions
  - OOP access to system components (K8s, Kapitan) & app
- activation → rebuild, redeployment
  - code push (GitOps, immutable infrastructure); or
  - CRD/app Commodore request (programmable infrastructure)

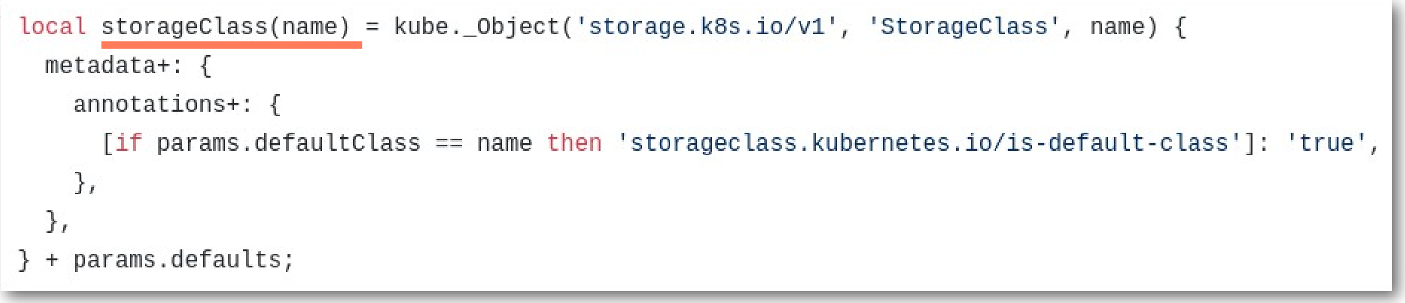
Commodore Components

Jsonnet example: variable vs. annotation pattern



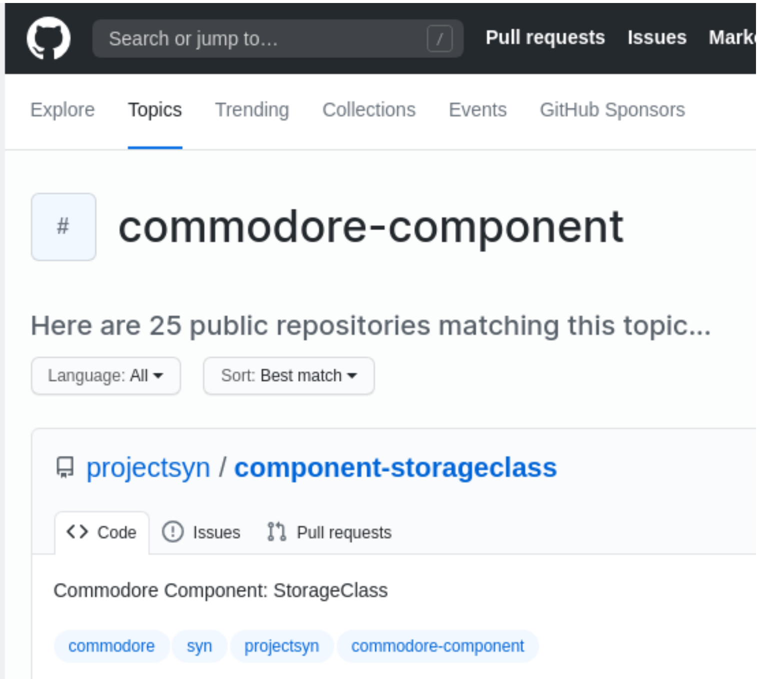
14 lines (12 sloc) | 387 Bytes

```
local kap = import 'lib/kapitan.libjsonnet';
local kube = import 'lib/kube.libjsonnet';
local inv = kap.inventory();
local sc = import 'lib/storageclass.libsonnet';

local params = inv.parameters.storageclass;

{
  [name]:
    sc.storageClass(name) + params.classes[name]
  for name in std.objectFields(params.classes)
} + {
  [if std.length(params.classes) == 0 then '.gitkeep']: {},
}
```



```
local storageClass(name) = kube._Object('storage.k8s.io/v1', 'StorageClass', name) {
  metadata+: {
    annotations+: {
      [if params.defaultClass == name then 'storageclass.kubernetes.io/is-default-class']: 'true',
    },
  },
} + params.defaults;
```

Commodore Components



Search or jump to… /

Pull requests Issues Mark

Explore Topics Trending Collections Events GitHub Sponsors

commodore-component

Here are 25 public repositories matching this topic...

Language: All ▾ Sort: Best match ▾

projectsyn / **component-storageclass**

<> Code ⓘ Issues Pull requests

Commodore Component: StorageClass

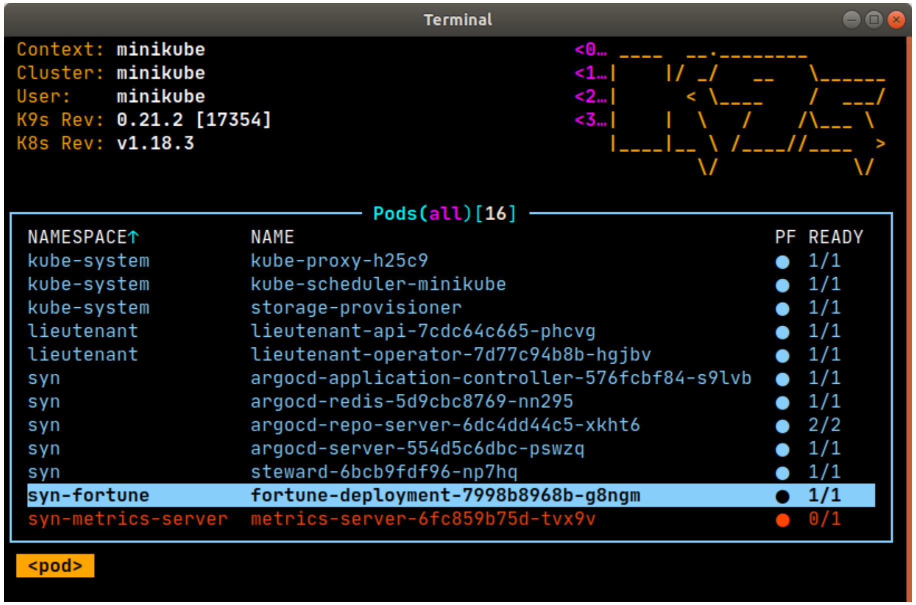
commodore syn projectsyn commodore-component

argocd
backup-k8up
espejo
fluentbit
keycloak
metallb
metrics-server
networkpolicy
openshift4-authentication
openshift4-cloudscale
openshift4-console
openshift4-registry
prometheus-adapter
resource-locker
steward
storageclass
…

Commodore Components

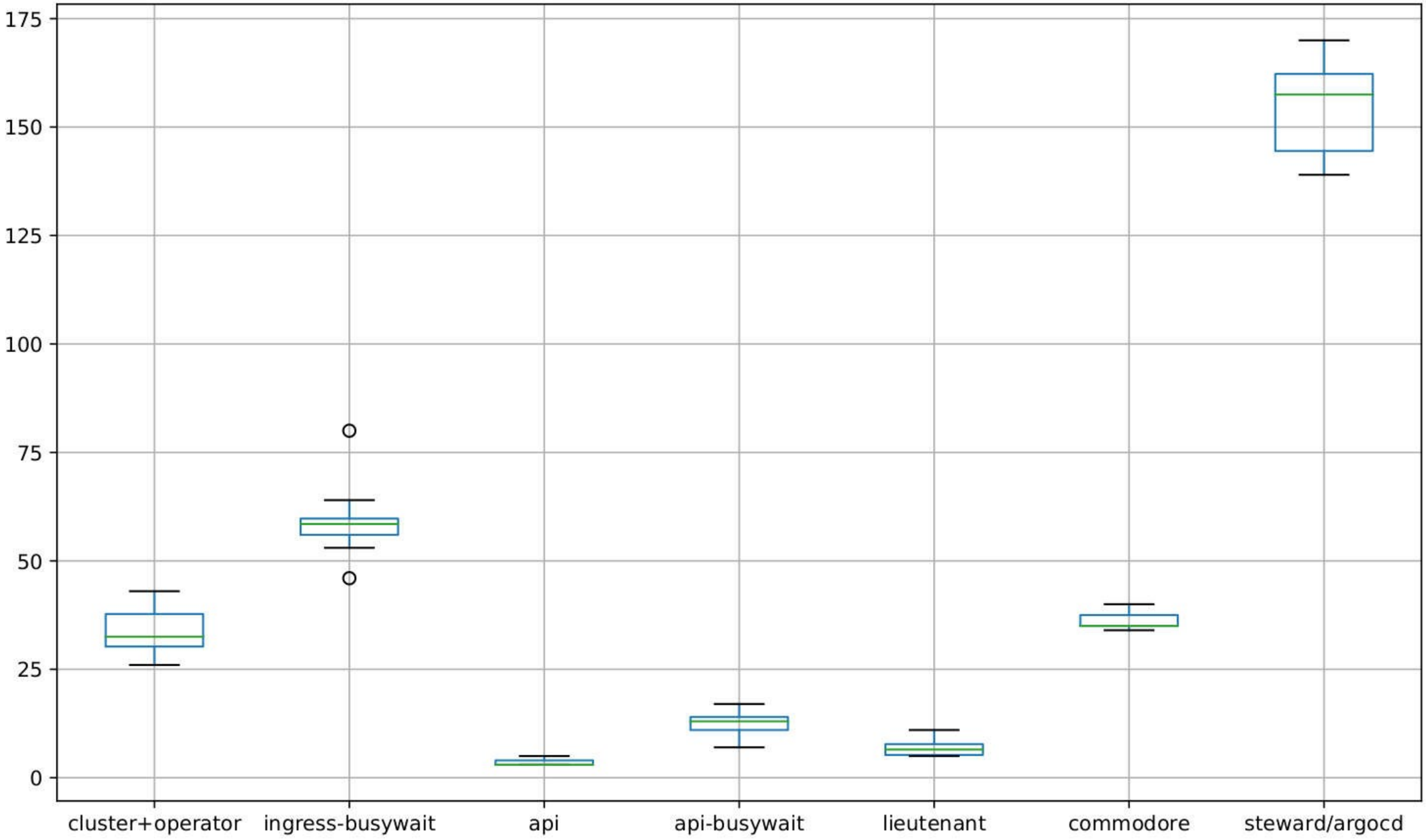
Advantages:

- re-use factor known from CBSE/SOSE adjusted to microservices
  - growing library
  - hierarchical configuration support
- easy cloud-nativeness («exploiting cloud possibilities»)
  - «I need backup» but do not want to care about e.g. S3 backup
- stronger self-service aspect
  - trusted components may require higher privileges, e.g. to create new system objects
- policies & conventions
  - e.g. namespaces



```
Context: minikube                                    <0… ____  __.________
Cluster: minikube                                    <1…|    |/ _/   __   \______
User:    minikube                                    <2…|      < \____    /  ___/
K9s Rev: 0.21.2 [17354]                              <3…|    |  \   /    /\___ \
K8s Rev: v1.18.3                                        |____|__ \ /____//____  >
                                                                \/            \/

                                    Pods(all)[16]
 NAMESPACE↑          NAME                                            PF READY
 kube-system         kube-proxy-h25c9                                ●  1/1
 kube-system         kube-scheduler-minikube                         ●  1/1
 kube-system         storage-provisioner                             ●  1/1
 lieutenant          lieutenant-api-7cdc64c665-phcvg                 ●  1/1
 lieutenant          lieutenant-operator-7d77c94b8b-hgjbv            ●  1/1
 syn                 argocd-application-controller-576fcbf84-s9lvb   ●  1/1
 syn                 argocd-redis-5d9cbc8769-nn295                   ●  1/1
 syn                 argocd-repo-server-6dc4dd44c5-xkht6             ●  2/2
 syn                 argocd-server-554d5c6dbc-pswzq                  ●  1/1
 syn                 steward-6bcb9fdf96-np7hq                        ●  1/1
 syn-fortune         fortune-deployment-7998b8968b-g8ngm             ●  1/1
 syn-metrics-server  metrics-server-6fc859b75d-tvx9v                 ●  0/1

<pod>
```

Syn - Bootstrapping



Demo (self-service)

```
# 0. Install k3d, kubectl, jq, curl etc.
#     (automation: commodore-component-tutorial/0_requirements.sh)
#     (git@github.com:projectsyn/commodore-component-tutorial.git)
# 1. Login to gitlab.com (create account if needed)
# 2. Go to settings
# 3. Generate Personal Access Token for ‘api‘ access
# 4. Download script and run with token [not secure but convenient]

curl 'https://drive.switch.ch/index.php/s/sNQz0yfHMmBMV5j/download' \
 > allsetup.sh && bash !$
```

Project Syn - Resources

Roadmap

- 0.1: March 2020
- 0.2: July 2020
- Next: dynamic config hierarchies, component testing, ...

Website

- https://syn.tools/syn/index.html

Tutorials

- https://syn.tools/syn/tutorials/getting-started.html
- https://docs.syn.tools/tuto/index.html