zalando

Developing Operators with Kubernetes Operator Pythonic Framework

KOPF

KubeCon San Diego

2019-11-20

SERGEY VASILYEV

@nolar

About me

- My name is **Sergey Vasilyev**
  - https://twitter.com/nolar
- A Python developer from Siberia. Now in Berlin, Germany.
- ~20 years of experience in Software Engineering.
- In love with Python since 2009.
- Sr. Backend Engineer, Zalando SE.
- Running ML apps & infra on Kubernetes
  for Zalando Pricing & Forecasting.


ABOUT US

ZALANDO AT A GLANCE

~ 5.4 billion EUR
revenue 2018

> 300 million
visits per month

~ 14,000
employees in Europe

> 80%
of visits via mobile devices

> 29 million
active customers

> 450,000
product choices

> 2,000
brands

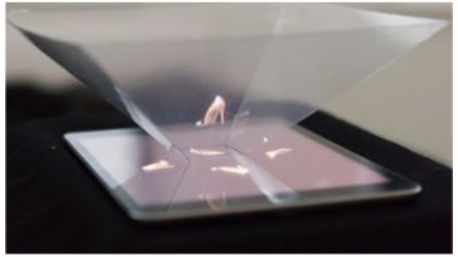
17
countries

as of September 2019

zalando

WE ARE CONSTANTLY INNOVATING TECHNOLOGY

HOME-BREWED, CUTTING-EDGE & SCALABLE
technology solutions



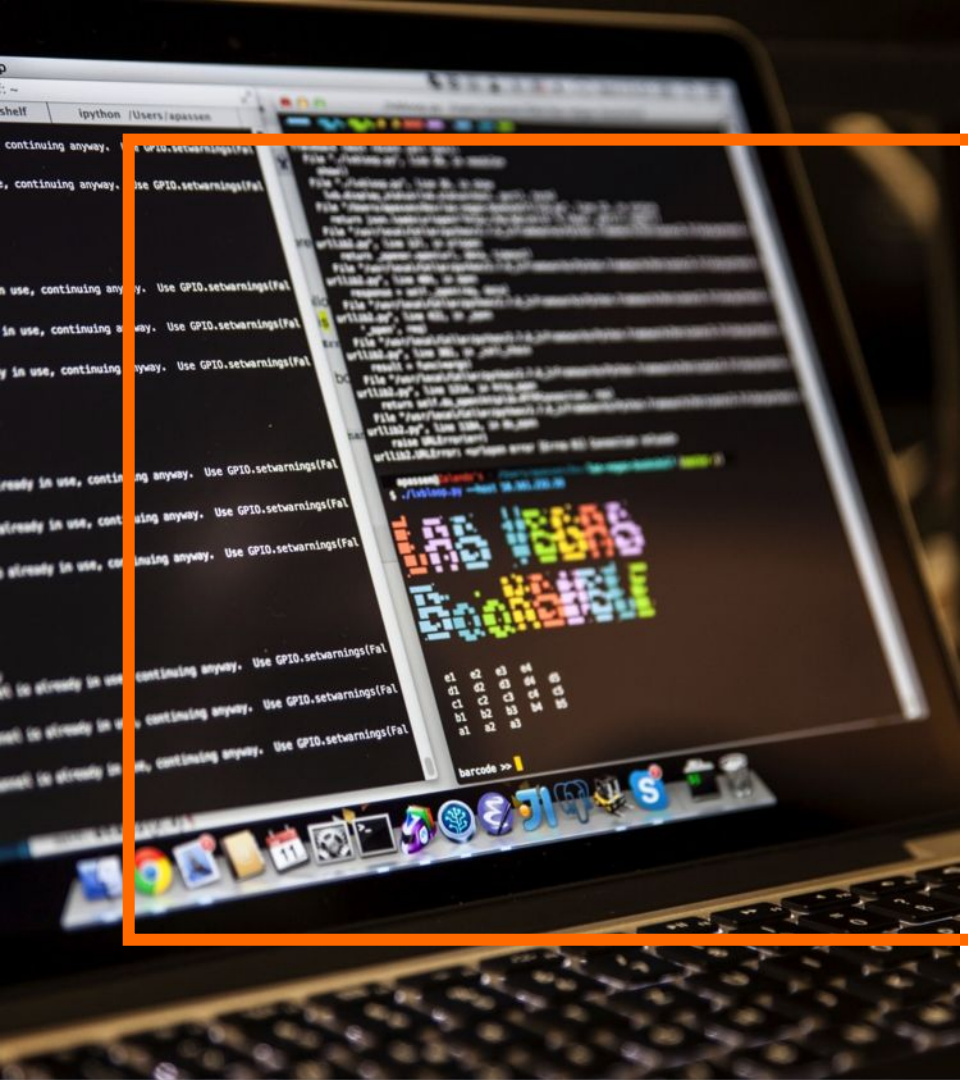
help our brand to
WIN ONLINE

7 international tech locations

HQs
in Berlin

> 2,000
employees at

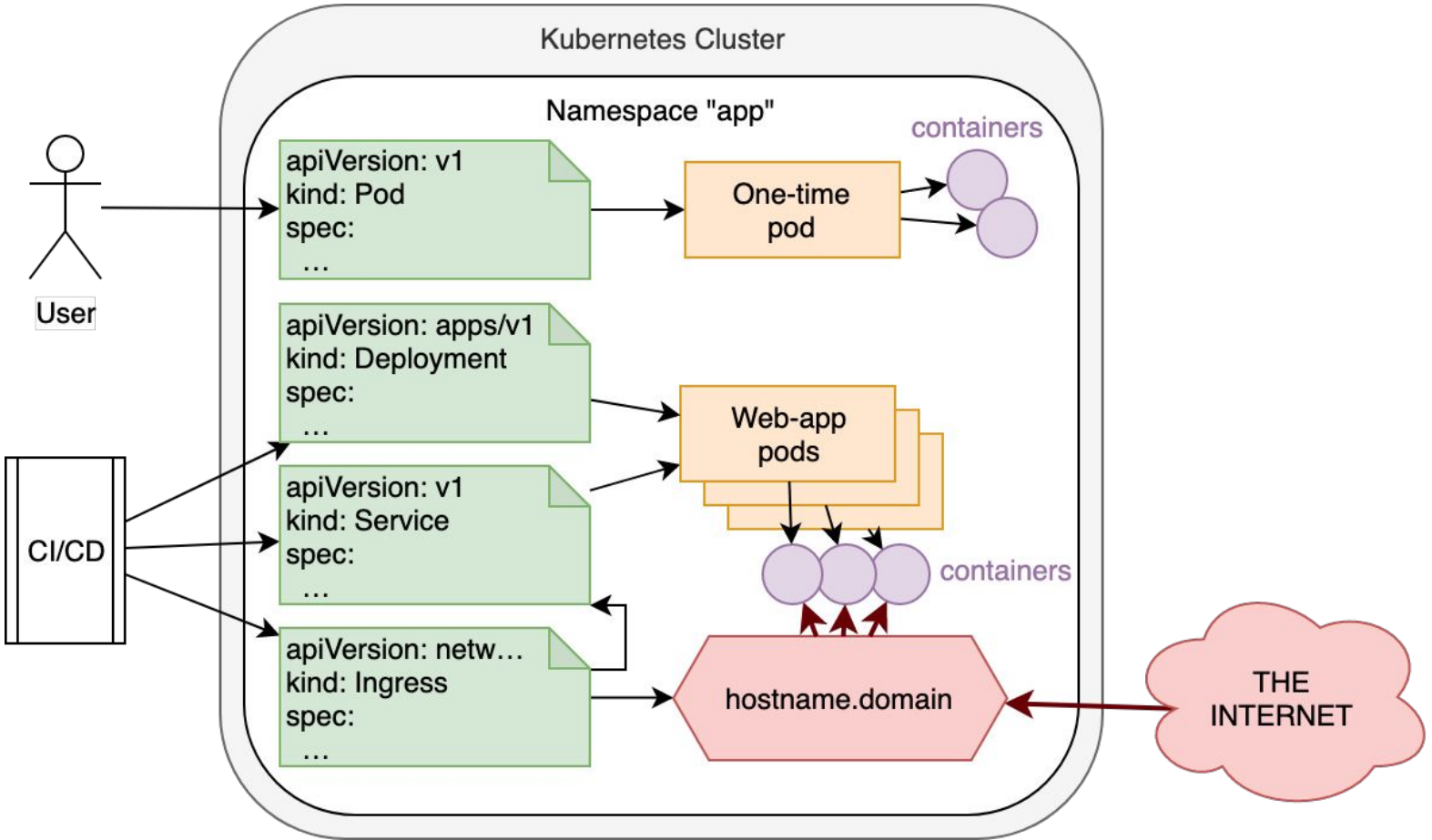
zalando



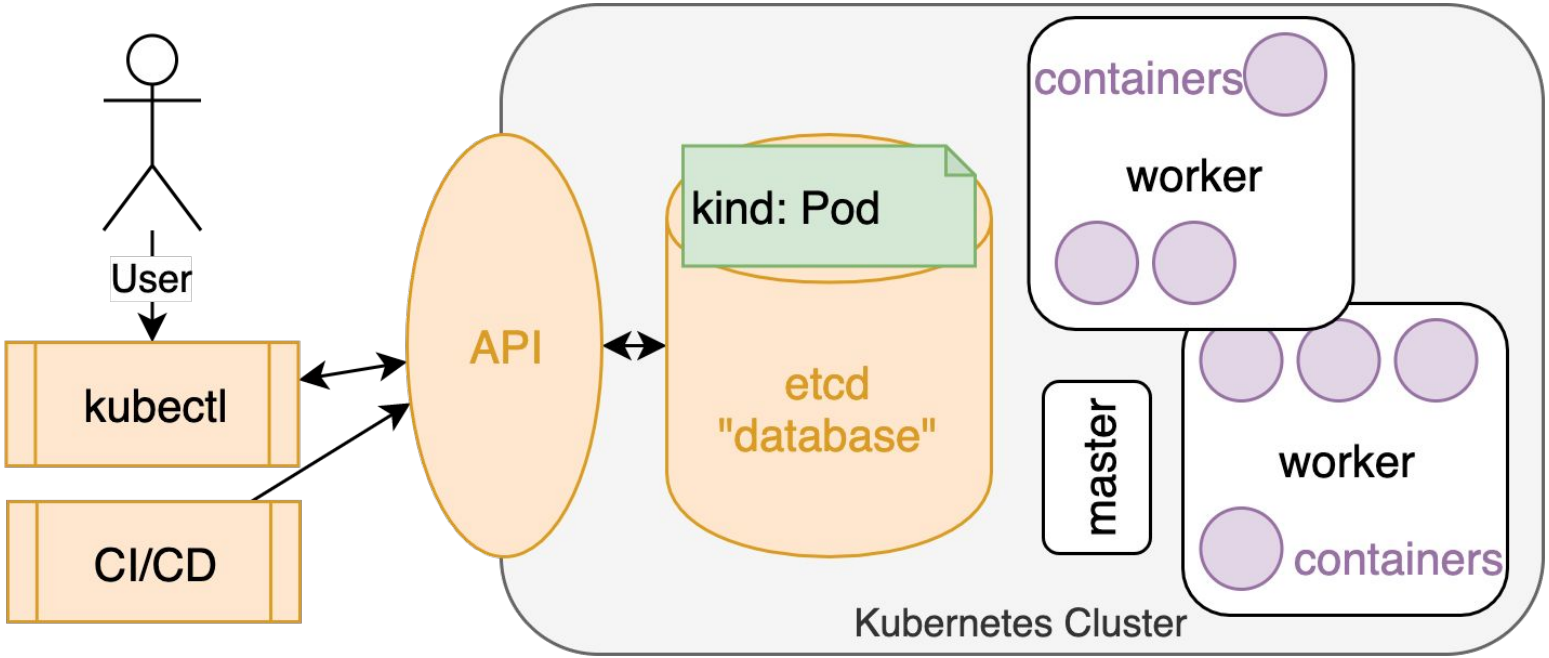
WE DRESS CODE

zalando

“Kubernetes is a container orchestrator”



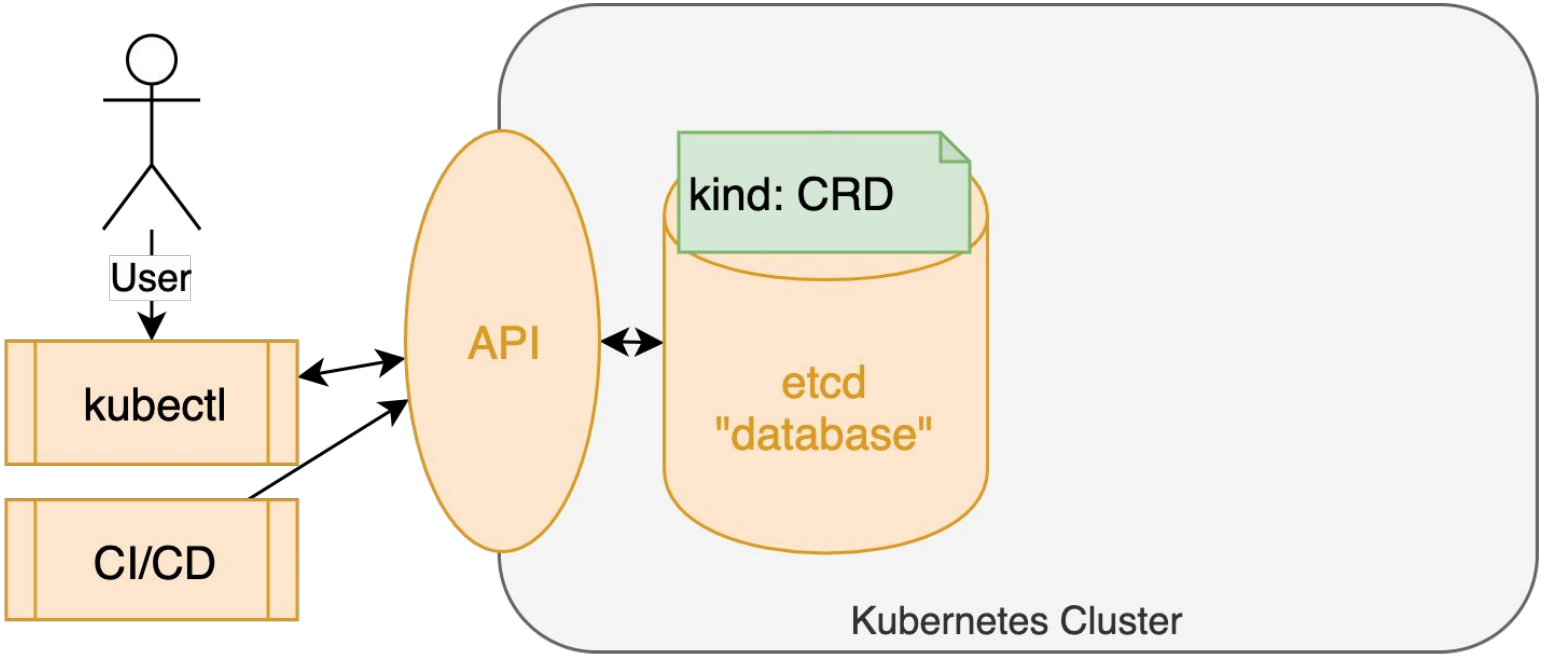
Kubernetes under the hood



kubectl get pods
kubectl create pod …
kubectl patch pod …
kubectl delete pod …

GET /api/v1/pods
POST …
PATCH …
DELETE …

Extending Kubernetes: Custom Resource Definitions



kubectl get kopfexamples
kubectl create kex …
kubectl patch kex …
kubectl delete kex …

GET /apis/zalando.org/v1/namespaces/default/kopfexamples
POST …
PATCH …
DELETE …

Defining a custom resource

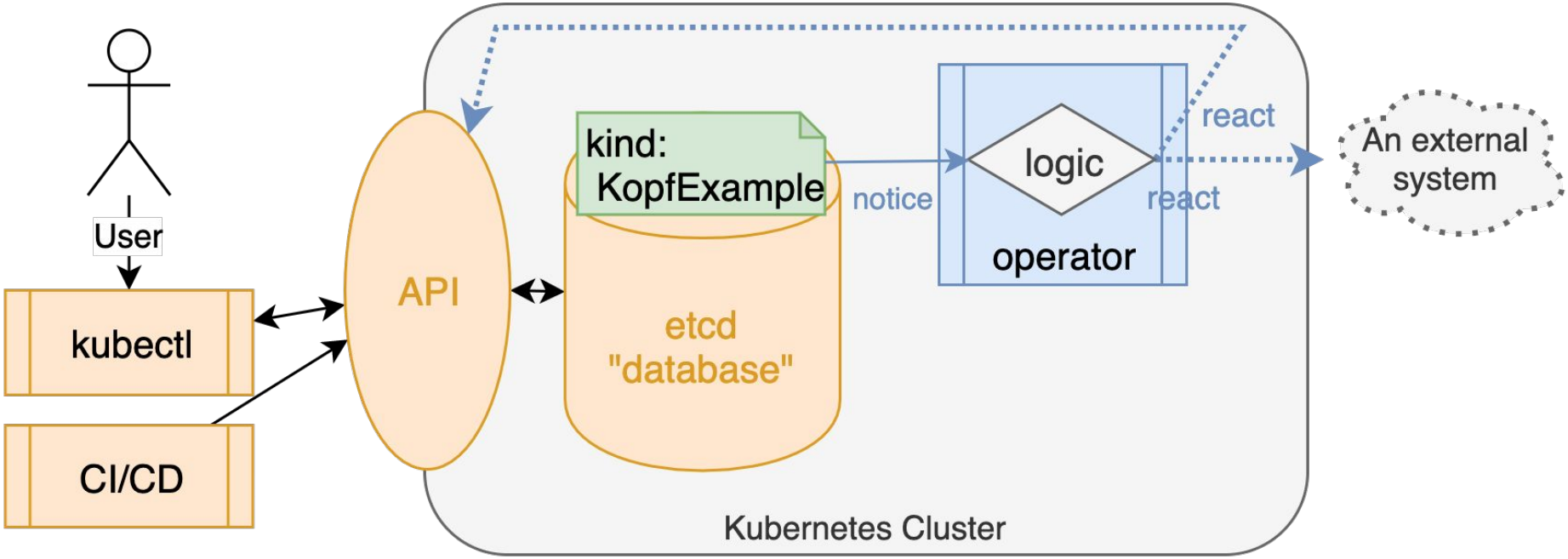
- **Required: “group/version”.**
- **Required: kind/plural/singular names.**
- **Required: scope (“Namespaced”).**
- Optional: short names (aliases).
- Optional: list formatting and columns.
- Optional: categories.

```yaml
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  name: kopfexamples.zalando.org
spec:
  scope: Namespaced
  group: zalando.org
  versions:
    - name: v1
      served: true
      storage: true
  names:
    kind: KopfExample
    plural: kopfexamples
    singular: kopfexample
    shortNames:
      - kopfexes
      - kopfex
      - kex
```

```yaml
apiVersion: zalando.org/v1
kind: KopfExample
metadata:
  name: kopf-example-1
  labels:
    somelabel: somevalue
  annotations:
    someannotation: somevalue
spec:
  duration: 1m
  field: value
  items:
  - item1
  - item2
```

```
$ kubectl apply -f crd.yaml
$ kubectl apply -f obj.yaml
```

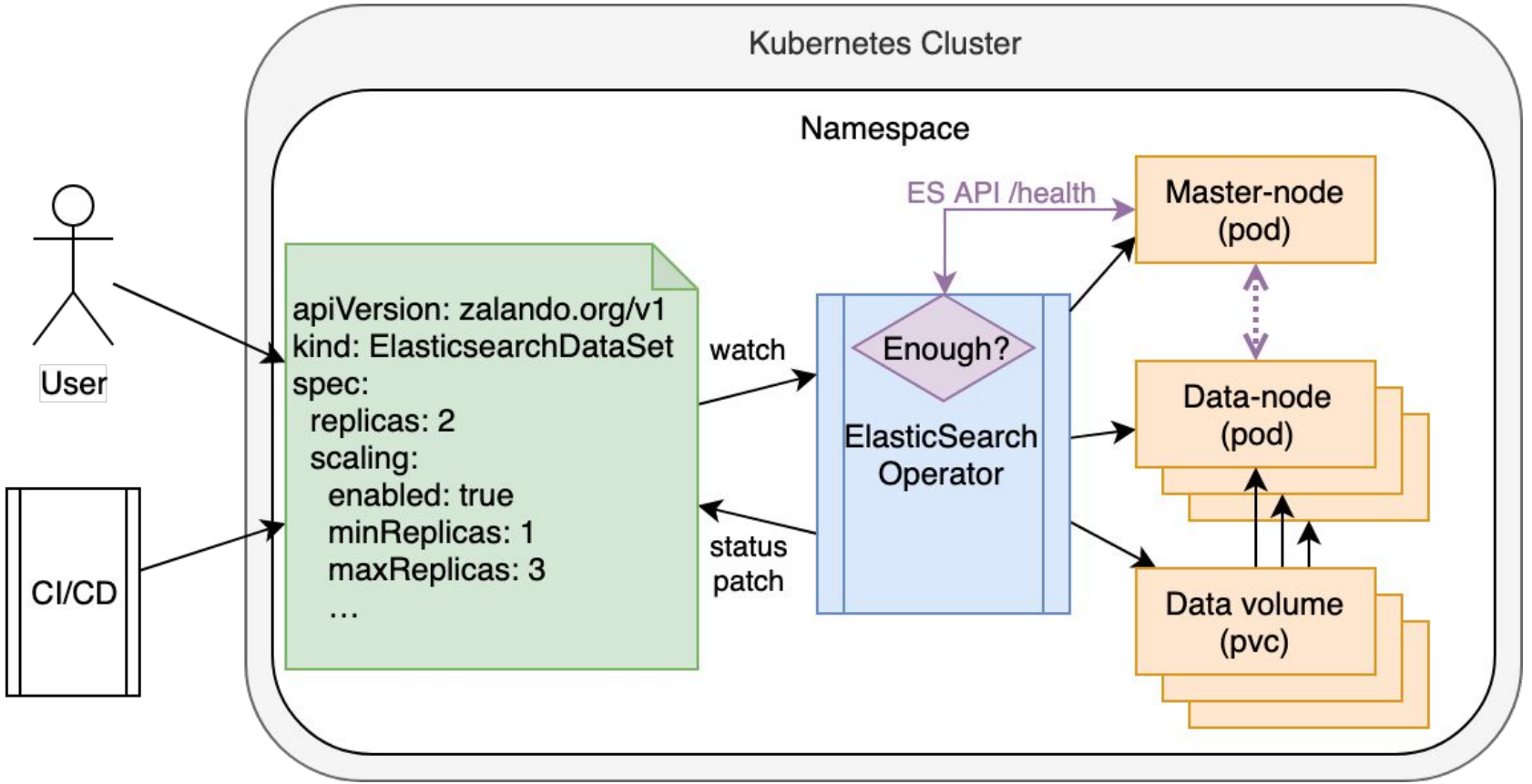
Extending Kubernetes: Controllers/Operators



kubectl get kopfexamples
kubectl create kex …
kubectl patch kex …
kubectl delete kex …

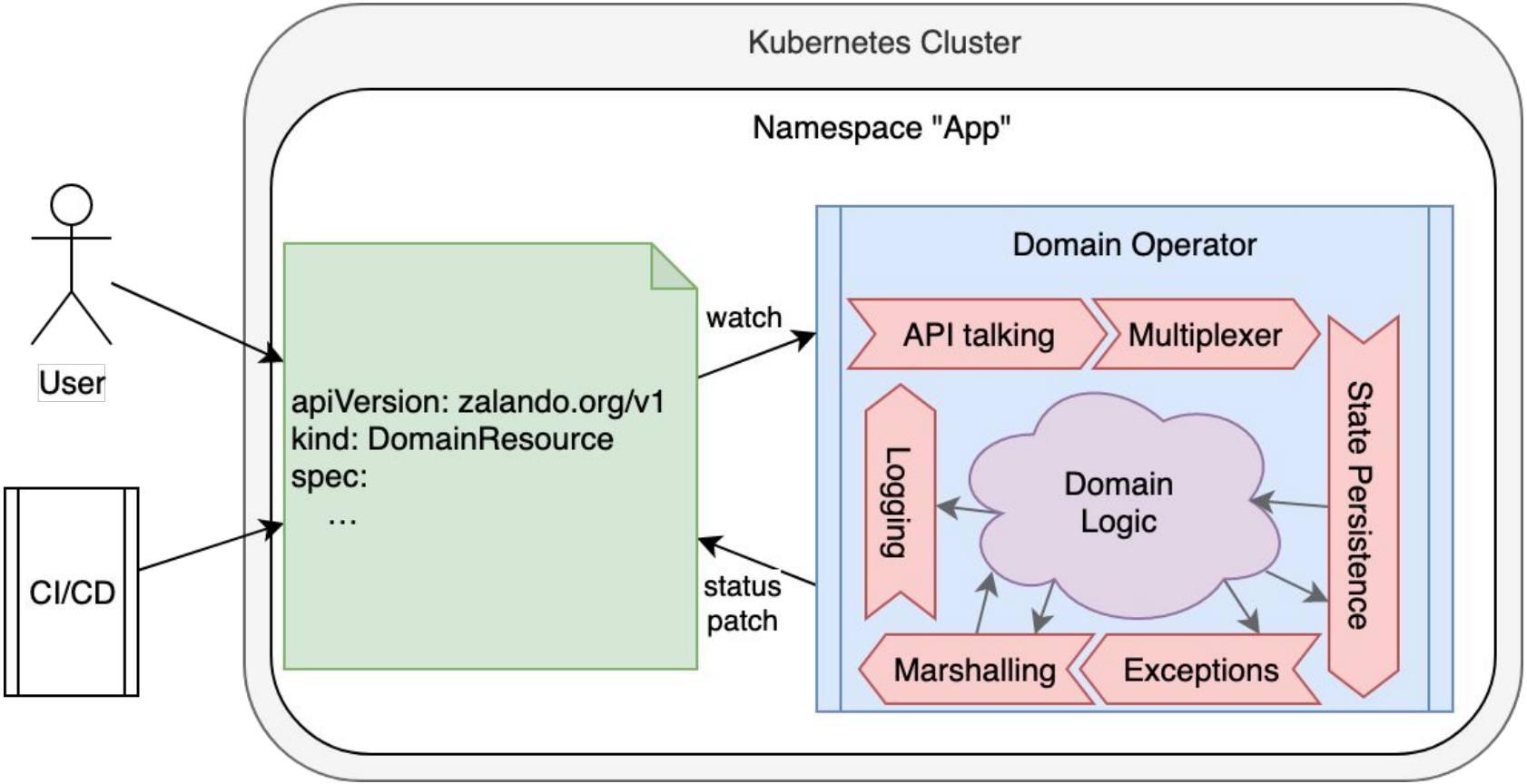
GET /apis/zalando.org/v1/namespaces/default/kopfexamples
POST …
PATCH …
DELETE …

Common use: an application-specific operator



https://github.com/zalando-incubator/es-operator

Common problem: infrastructure code hassle



MAKE A FRAMEWORK!

WILLKOMMEN, **KOPF**!

The simplest Kubernetes operator

```python
import kopf

@kopf.on.create('zalando.org', 'v1', 'kopfexamples')
def create_fn(spec, logger, **kwargs):
    logger.info("%s", f"And here we go: {spec}")
```

The simplest Kubernetes operator

```python
import kopf


@kopf.on.create('zalando.org', 'v1', 'kopfexamples')
def create_fn(spec, **kwargs):
    print(f"And here we are! Creating: {spec}")
    return {'message': 'hello world'}  # will be the new status
```

```yaml
# A demo custom resource for
apiVersion: zalando.org/v1
kind: KopfExample
metadata:
  name: kopf-example-1
  labels:
    somelabel: somevalue
spec:
  duration: 1m
  field: value
  items:
  - item1
  - item2
```

```
$ kopf run scripts.py [--verbose]
```

```
And here we are! Creating: {'duration': '1m', 'field': 'value',
'items': ['item1', 'item2']}

[2019-02-25 14:06:54,742] kopf.reactor.handlin [INFO    ]
[default/kopf-example-1] Handler create_fn succeeded.

[2019-02-25 14:06:54,856] kopf.reactor.handlin [INFO    ]
[default/kopf-example-1] All handlers succeeded for creation.
```

```
$ kubectl apply -f obj.yaml
$ kubectl describe -f obj.yaml
```

```
Name:         kopf-example-1
...
Status:
  create_fn:
    Message:  hello world
Events:
  Type    Reason   Age   From  Message
  ----    ------   ----  ----  -------
  Normal  Success  81s   kopf  Handler create_fn succeeded.
```

Resource-watching handlers

- As often, as the events arrive from K8s API.
- Raw data, no interpretation.
- Fire-and-forget, errors are ignored.
- Similar to `kubectl get --watch`

```python
import kopf


@kopf.on.event('zalando.org', 'v1', 'kopfexamples')
def event_fn_with_error(**kwargs):
    raise Exception("Oops!")


@kopf.on.event('zalando.org', 'v1', 'kopfexamples')
def normal_event_fn(event, **kwargs):
    print(f"Event received: {event!r}")

```

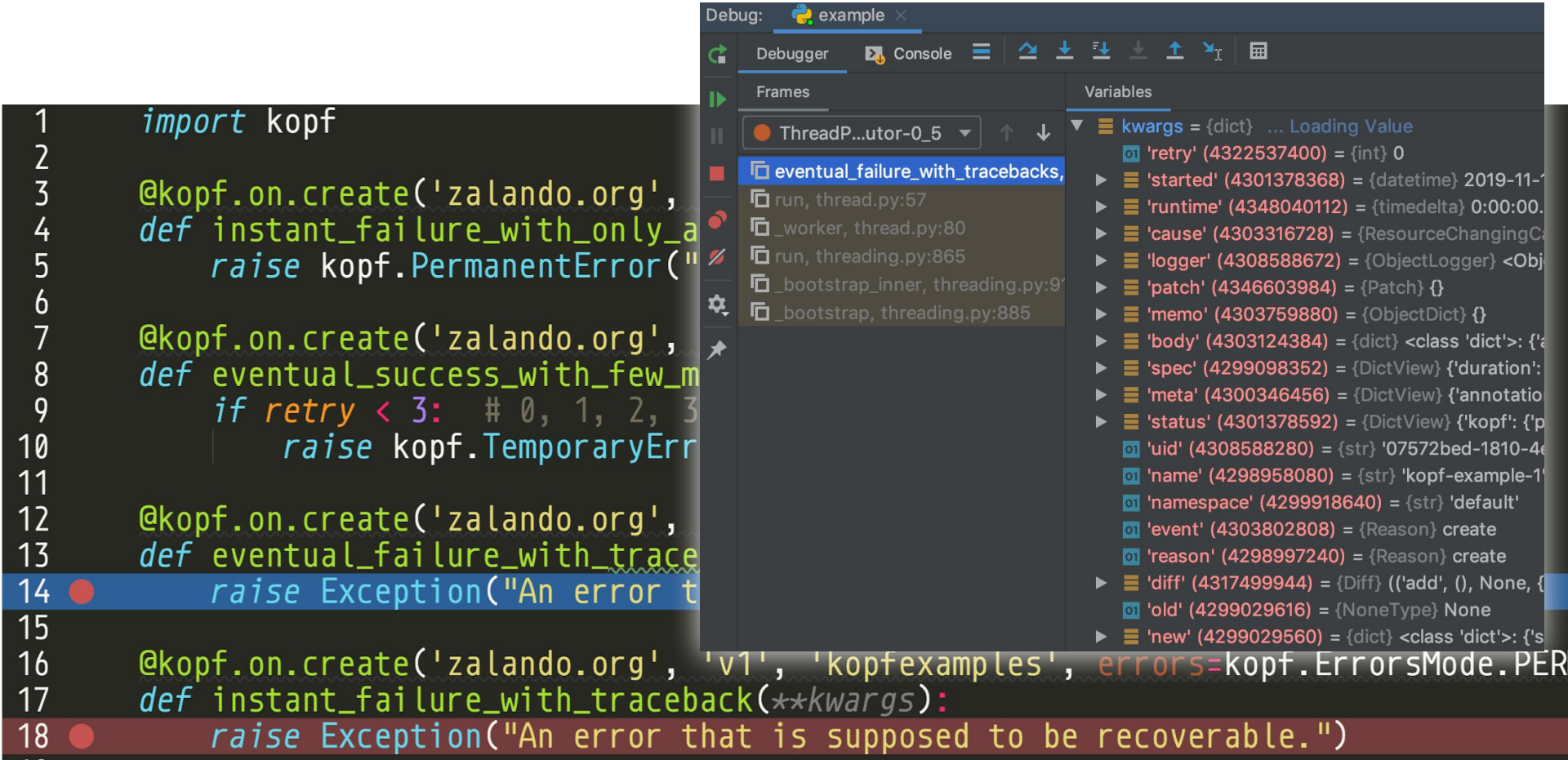
Change-detection handlers

```python
import kopf

@kopf.on.create('zalando.org', 'v1', 'kopfexamples')
def create_fn_1(spec, **kwargs):
    print(f'CREATED 1st: field={spec.field}')

@kopf.on.create('zalando.org', 'v1', 'kopfexamples')
def create_fn_2(meta, **kwargs):
    print(f'CREATED 2nd: name={meta["name"]}')

@kopf.on.update('zalando.org', 'v1', 'kopfexamples')
def update_fn(old, new, diff, **kwargs):
    print(f'UPDATED: diff={diff}')

@kopf.on.delete('zalando.org', 'v1', 'kopfexamples')
def delete_fn_1(**kwargs):
    print('DELETED')

@kopf.on.field('zalando.org', 'v1', 'kopfexamples', field='spec.field')
def field_fn(old, new, **kwargs):
    print(f'FIELD CHANGED: {old} -> {new}')
```

Error handling

```python
import kopf

@kopf.on.create('zalando.org', 'v1', 'kopfexamples')
def instant_failure_with_only_a_message(**kwargs):
    raise kopf.PermanentError("Fail once and for all.")

@kopf.on.create('zalando.org', 'v1', 'kopfexamples')
def eventual_success_with_few_messages(retry, **kwargs):
    if retry < 3:  # 0, 1, 2, 3
        raise kopf.TemporaryError("Expected recoverable error.", delay=1.0)

@kopf.on.create('zalando.org', 'v1', 'kopfexamples', retries=3)
def eventual_failure_with_tracebacks(**kwargs):
    raise Exception("An error that is supposed to be recoverable.")

@kopf.on.create('zalando.org', 'v1', 'kopfexamples', errors=kopf.ErrorsMode.PERMANENT)
def instant_failure_with_traceback(**kwargs):
    raise Exception("An error that is supposed to be recoverable.")
```

Debugging & breakpoints

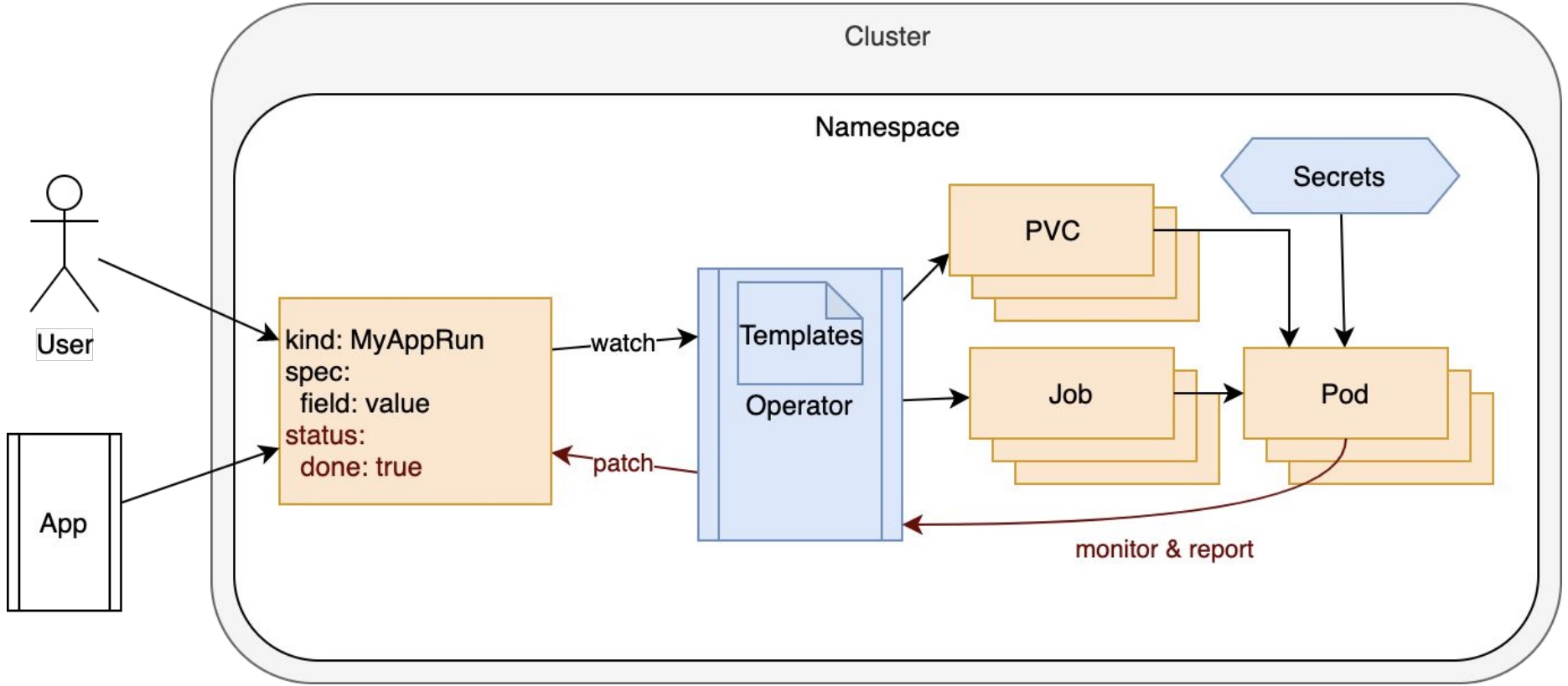


Other things

- Python logging → Kubernetes Events.
- Custom authentication → @on.login
- @on.startup / @on.cleanup
- Embeddable.
- Testing toolkit.
- Resource hierarchies.
- *And more…*

- *Other patterns? Create a feature request!*

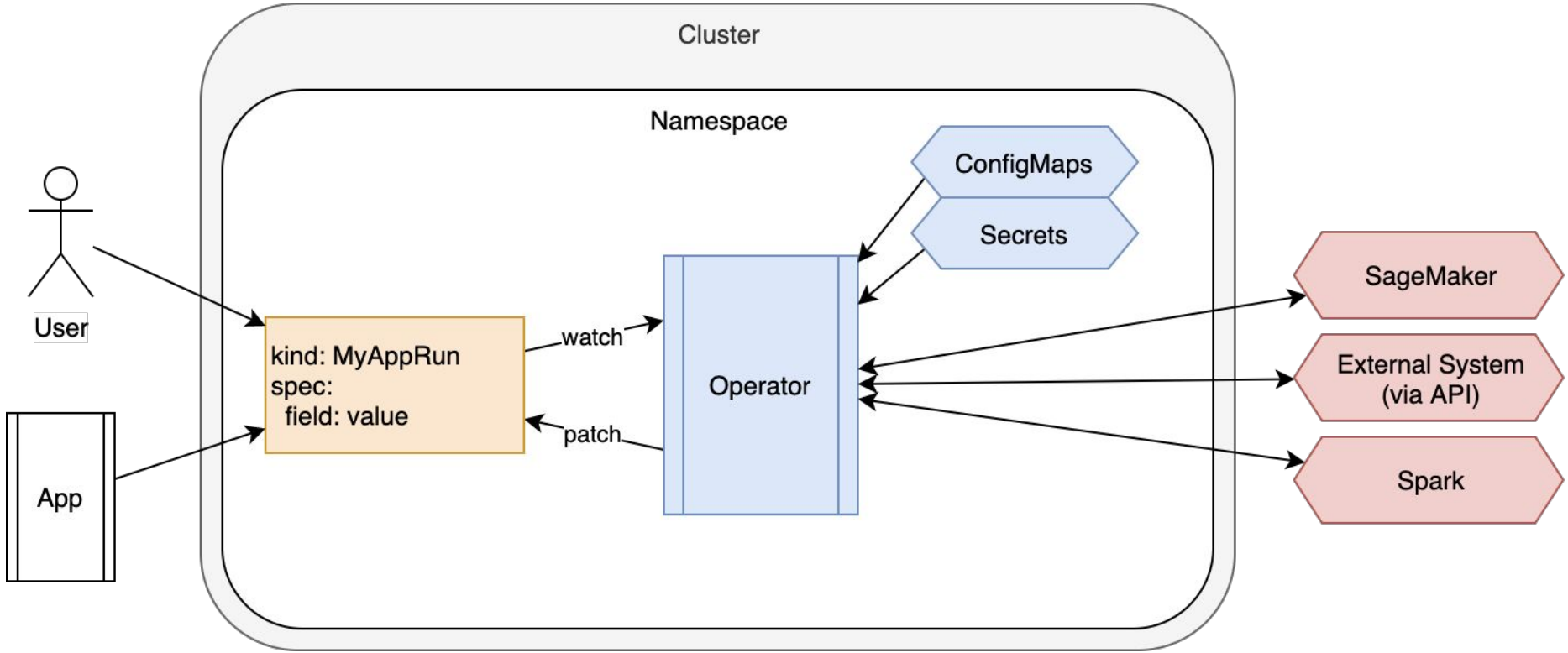
Cross-resource relations



Cross-resource relations

```python
@kopf.on.create('zalando.org', 'v1', 'kopfexamples')
def kex_created(spec, name, **kwargs):
    doc = yaml.safe_load("""...""")
    kopf.adopt(doc)
    kopf.label(doc, {'parent-name': name}, nested=['spec.template'])

    child = pykube.Job(api, doc)
    child.create()


@kopf.on.event('', 'v1', 'pods', labels={'parent-name': None})
def event_in_a_pod(meta, status, namespace, **kwargs):
    phase = status.get('phase')  # make a decision!
    query = KopfExample.objects(api, namespace=namespace)
    parent = query.get_by_name(meta['labels']['parent-name'])
    parent.patch({'status': {'children': phase}})
```

Kubernetes is an orchestrator of everything



Kubernetes is an orchestrator of everything

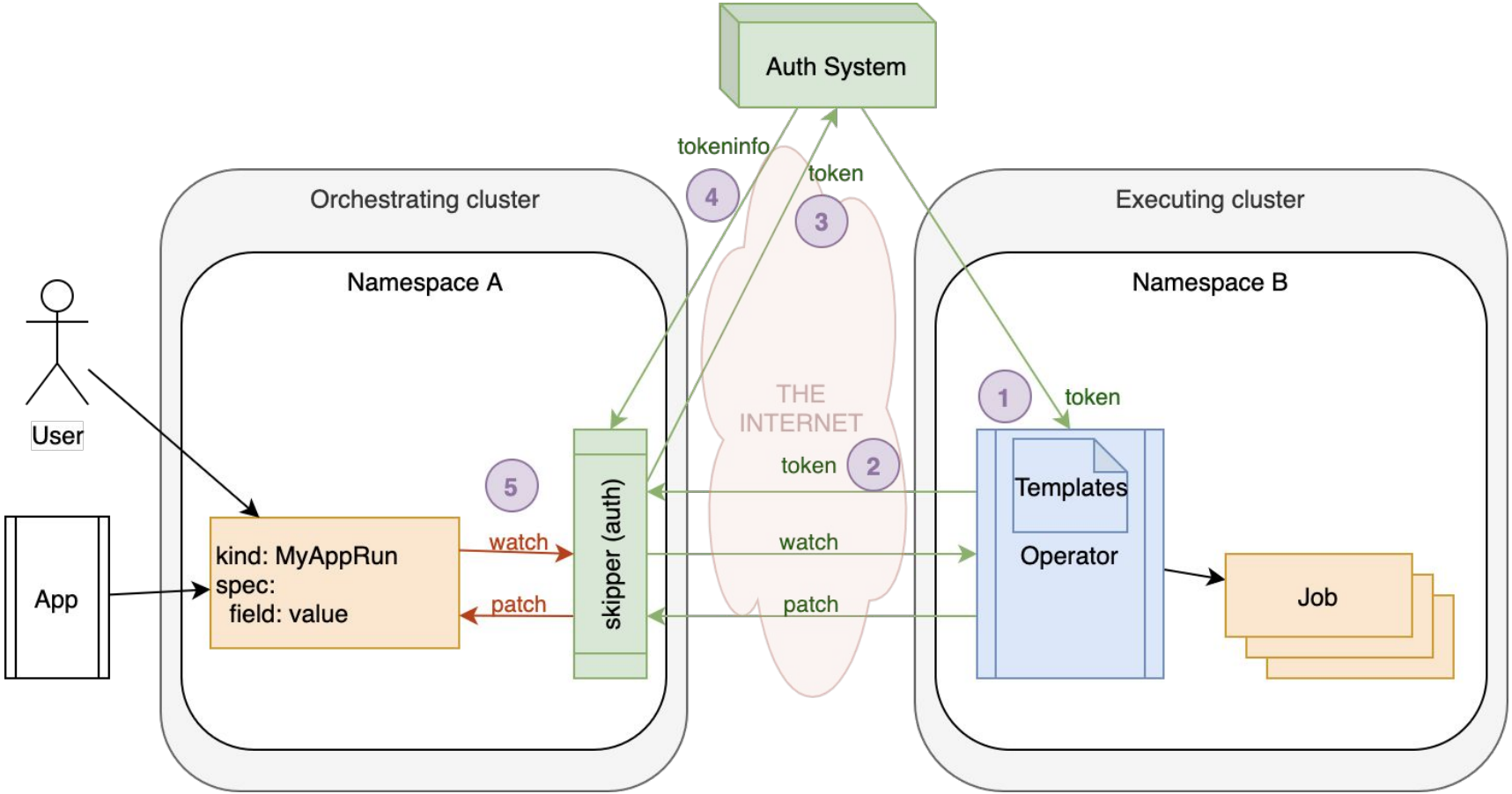
```python
@kopf.on.create('zalando.org', 'v1', 'kopfexamples')
@kopf.on.resume('zalando.org', 'v1', 'kopfexamples')
async def created(body, logger, memo, name, namespace, **kwargs):
    task = asyncio.create_task(bg(name, namespace, logger))
    memo.task = task

@kopf.on.delete('zalando.org', 'v1', 'kopfexamples')
async def deleted(memo, **kwargs):
    if hasattr(memo, 'task'):
        memo.task.cancel()
        try:
            await memo.task
        except Exception:
            pass

async def bg(name, namespace, logger):
    while True:
        try:
            logger.info("Still alive. Checking...")
            some_remote_progress = random.randint(0, 100)

            api = pykube.HTTPClient(pykube.KubeConfig.from_env())
            obj = KopfExample.objects(api, namespace=namespace).get_by_name(name)
            obj.patch({'status': {'message': f'{some_remote_progress}% done'}})

        except Exception as e:
            logger.error("Dead: %r", e)
        else:
            await asyncio.sleep(10)
```

Kubernetes is an orchestrator of everything

Coming soon!

```python
@kopf.on.timer('zalando.org', 'v1', 'kopfexamples', interval=10)
async def check_it(logger, patch, **kwargs):
    logger.info("Checking...")
    some_remote_progress = random.randint(0, 100)
    patch.setdefault('status', {})['message'] = f'{some_remote_progress}%'
```

Cross-cluster connectivity



Cross-cluster connectivity

```python
import kopf


@kopf.on.login()
def remote_cluster_a(**kwargs):
    return kopf.ConnectionInfo(
        server='https://remote-kubernetes/',
        token='abcdef123')


@kopf.on.create('zalando.org', 'v1', 'kopfexamples')
def kex_created(name, **kwargs):
    print(name)
```

FUTURE

- More patterns: cross-resource handlers.
- Reconciliation handlers: by time; threads/tasks.
- Admission handlers: validation/mutation.

- SDK: YAML from Python — CRDs, RBAC, Deployments.
- Operator Lifecycle Manager integration.
- More Kopf-based operators.

- Bring Python to Kubernetes, build a community.
- Conquer the world [of Kubernetes operators].

SUMMARY

- Kubernetes operators can be easy.
- Kubernetes operators can be ad-hoc.
- Kubernetes operators can be Pythonic.
- Kubernetes is an orchestrator of everything.
- Focus on the domain, not on the infrastructure.
- Simplicity & human-friendliness as the #1 priority.
- Python community as a huge leverage for Kubernetes.
- Use Kopf.

LINKS

- Source: https://github.com/zalando-incubator/kopf/
- Examples: https://github.com/zalando-incubator/kopf/tree/master/examples
- Documentation: https://kopf.readthedocs.io/

- Me (Sergey Vasilyev):
  - Twitter: @nolar
- Us (Zalando SE):
  - Twitter: @ZalandoTech & https://jobs.zalando.com/tech/blog/

- Other Zalando operators (not Kopf-based):
  - ElasticSearch operator: https://github.com/zalando-incubator/es-operator
  - Postgres operator: https://github.com/zalando/postgres-operator
- Worth reading:
  - Kubernetes Failure Stories: https://k8s.af/

zalando

QUESTIONS?

SERGEY VASILYEV

SENIOR BACKEND ENGINEER

sergey.vasilyev@zalando.de

@nolar

Why Python? — Scale x15!