zalando

KOPF

Kubernetes

Operator

Pythonic

Framework

SERGEY VASILYEV

Twitter: @nolar

(Zalando SE)

Python Pizza Berlin, 23.08.2019

Kopf,

a Kubernetes Operator Pythonic Framework

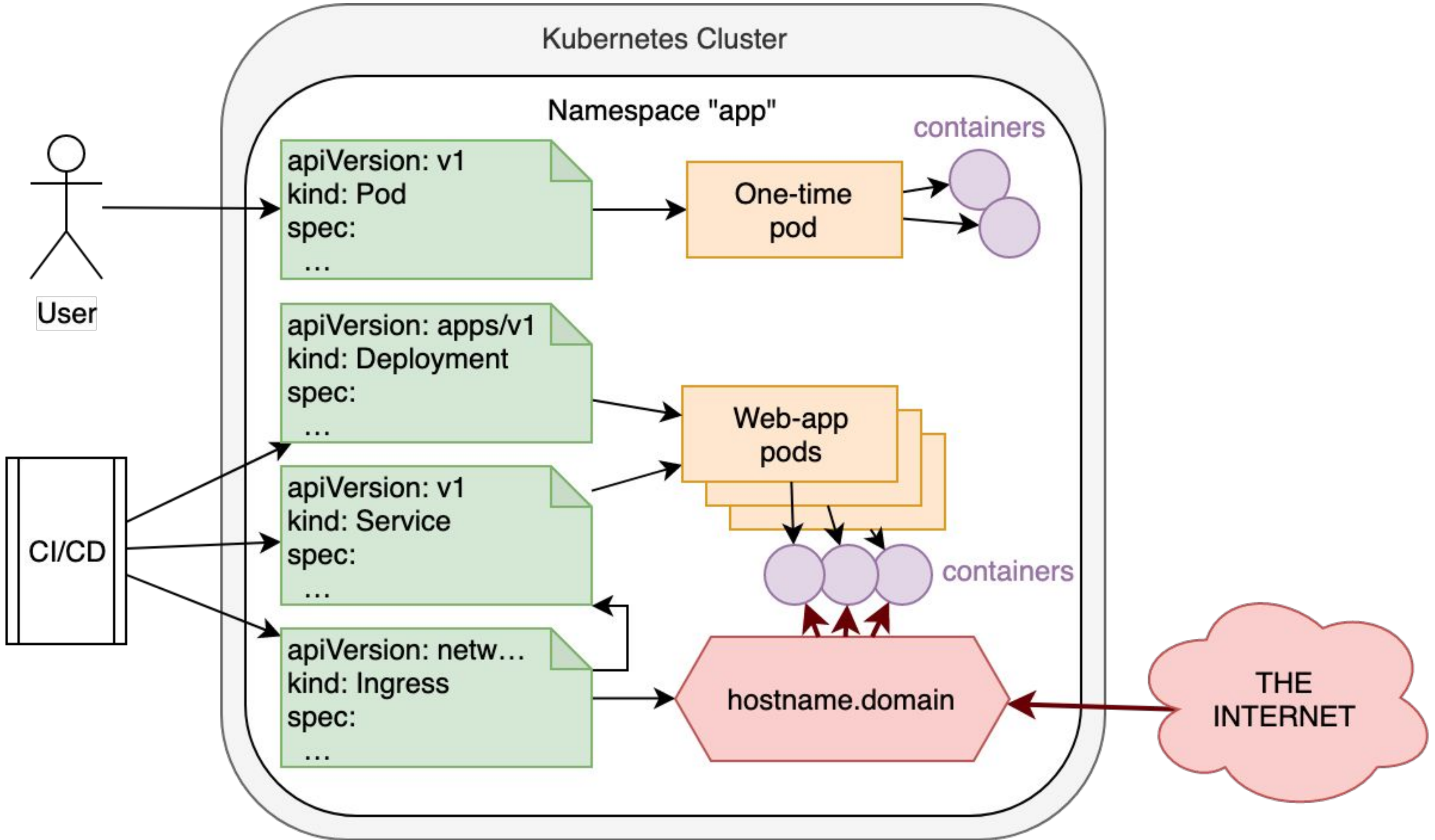
Sergey Vasilyev
Twitter: @nolar
Zalando SE

Python Pizza Berlin, 23.08.2019

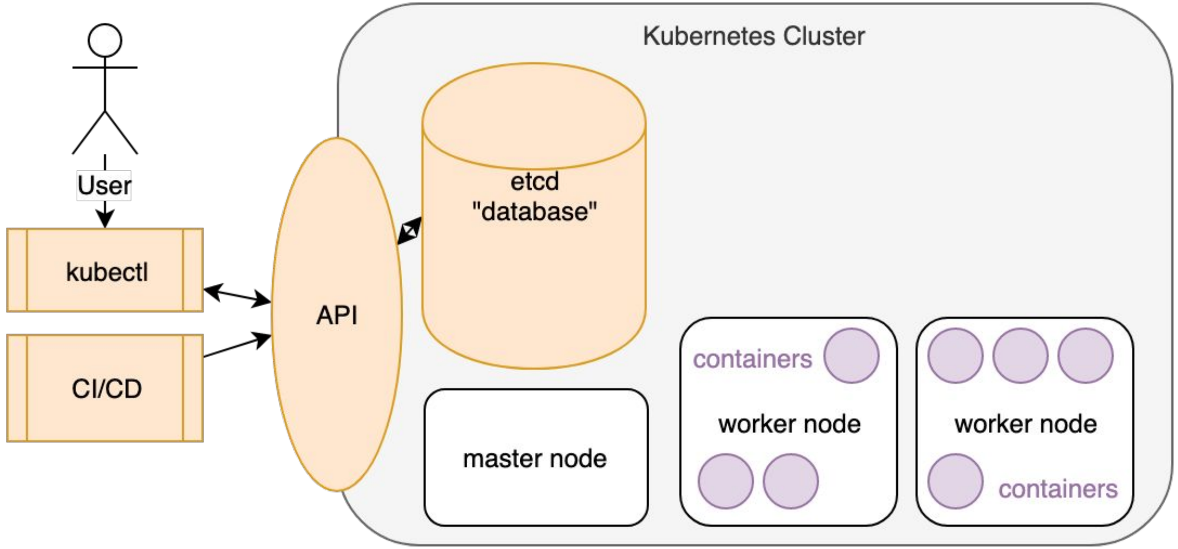
About me

- My name is **Sergey Vasilyev**
    - https://twitter.com/nolar
- Sr. Backend Engineer in Zalando SE.
- Running ML apps & infra on Kubernetes for Zalando Pricing & Forecasting.

“Kubernetes is a container orchestrator”



Kubernetes under the hood



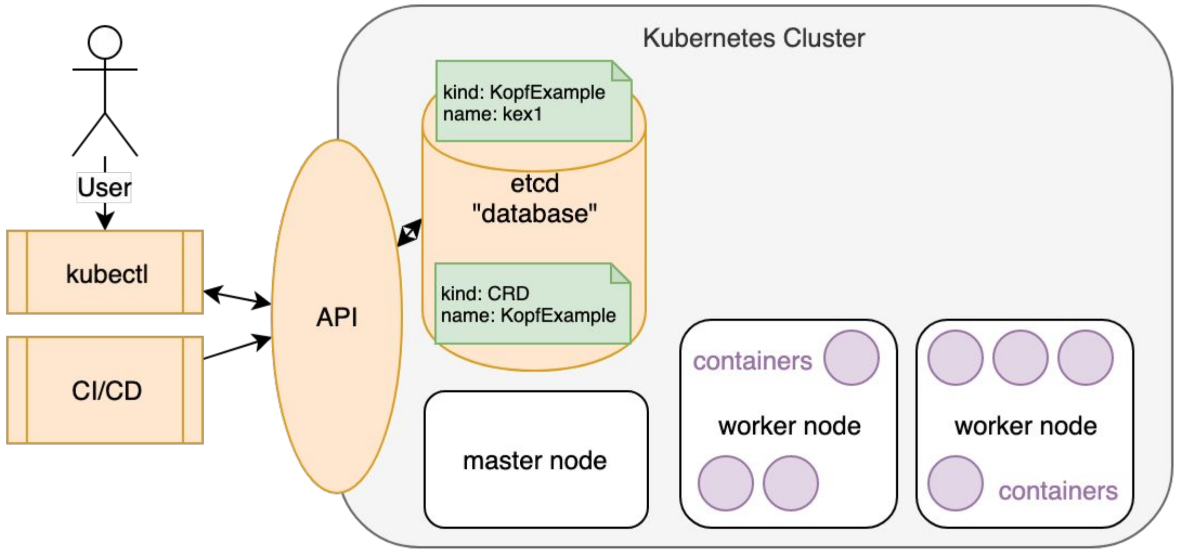
CLI commands

kubectl get pods [-w]
kubectl describe pod pod1
kubectl create pod …
kubectl delete pod …
kubectl patch pod …
and so on…

API endpoints

GET /api/v1/pods[?watch=true]
GET /api/v1/namespaces/ns1/pods[?watch=true]
POST /api/v1/namespaces/ns1/pods
GET /api/v1/namespaces/ns1/pods/pod1
PATCH /api/v1/namespaces/ns1/pods/pod1
DELETE /api/v1/namespaces/ns1/pods/pod1
and so on…

Extending Kubernetes: Custom Resource Definitions



CLI commands

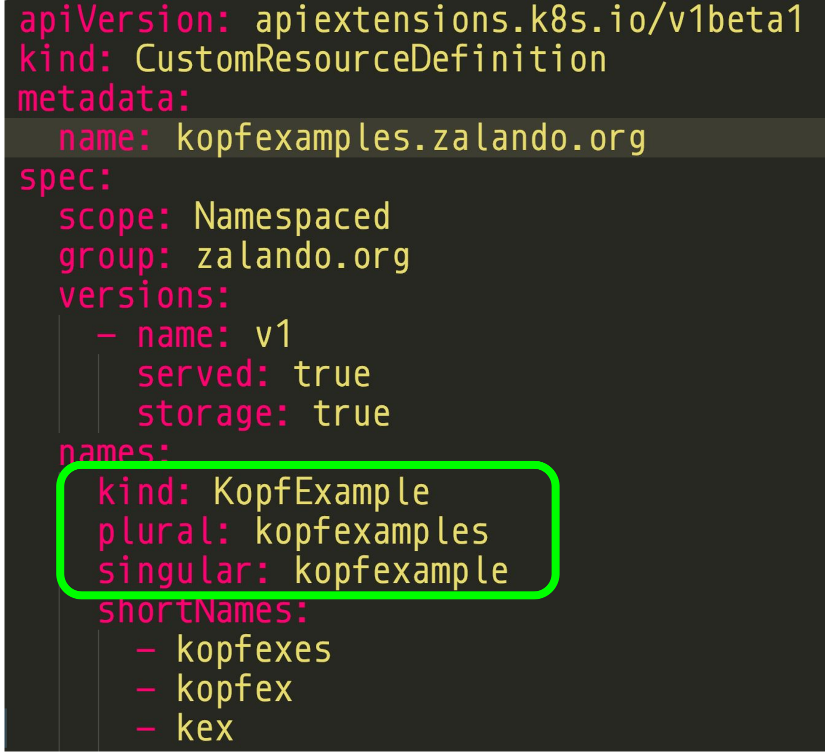
```
kubectl get kopfexamples [-w]
kubectl describe kopfexample kex1
kubectl create kopfexamples …
kubectl delete kopfexamples …
kubectl patch kex …
and so on…
```

API endpoints

```
GET /apis/zalando.org/v1/kopfexamples[?watch=true]
GET /apis/zalando.org/v1/namespaces/ns1/kopfexamples[?watch=true]
POST /apis/zalando.org/v1/namespaces/ns1/kopfexamples
GET /apis/zalando.org/v1/namespaces/ns1/kopfexamples/kex1
PATCH /apis/zalando.org/v1/namespaces/ns1/kopfexamples/kex1
DELETE /apis/zalando.org/v1/namespaces/ns1/kopfexamples/kex1
and so on…
```

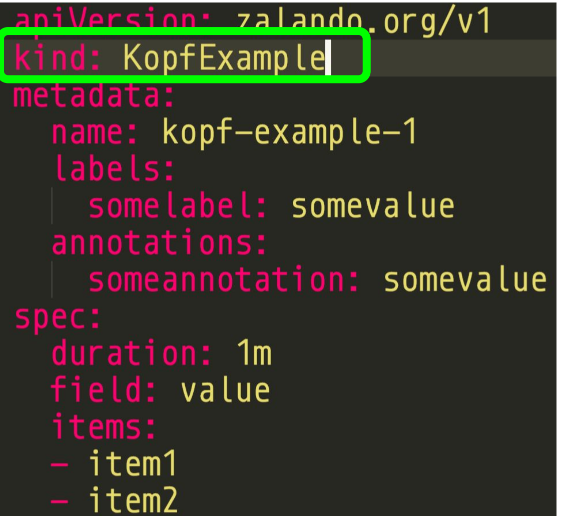
Declaring a CRD

- **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
```

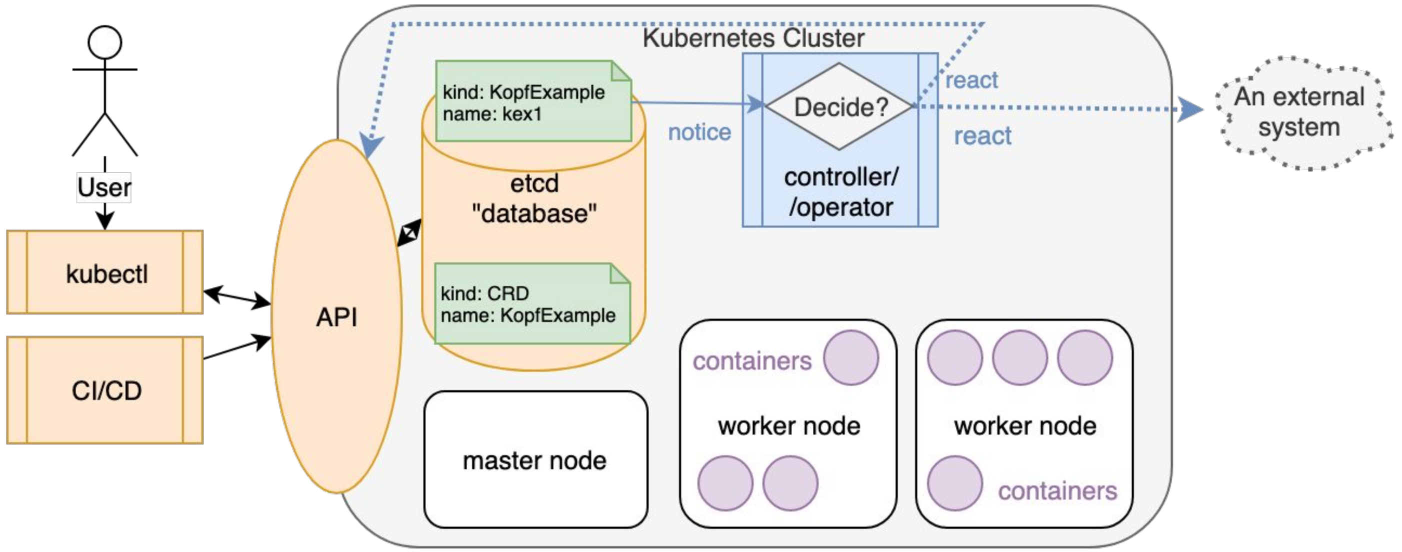
```
$ kubectl apply -f examples/crd.yaml
```



```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 examples/obj.yaml
```

Extending Kubernetes: Controllers/Operators



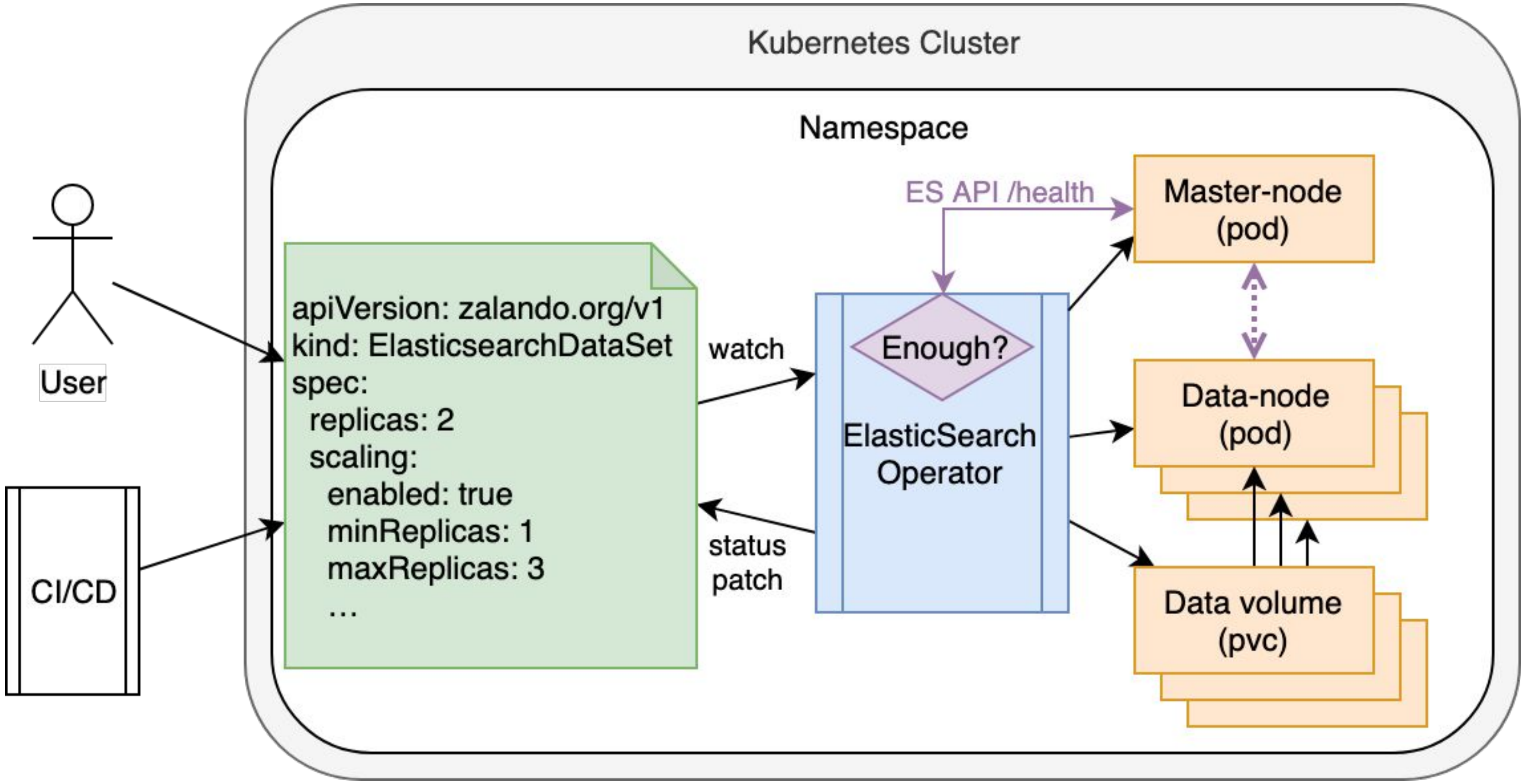
CLI commands

```
kubectl get kopfexamples [-w]
kubectl describe kopfexample kex1
kubectl create kopfexamples …
kubectl delete kopfexamples …
kubectl patch kex …
and so on…
```

API endpoints

```
GET /apis/zalando.org/v1/kopfexamples[?watch=true]
GET /apis/zalando.org/v1/namespaces/ns1/kopfexamples[?watch=true]
POST /apis/zalando.org/v1/namespaces/ns1/kopfexamples
GET /apis/zalando.org/v1/namespaces/ns1/kopfexamples/kex1
PATCH /apis/zalando.org/v1/namespaces/ns1/kopfexamples/kex1
DELETE /apis/zalando.org/v1/namespaces/ns1/kopfexamples/kex1
and so on…
```

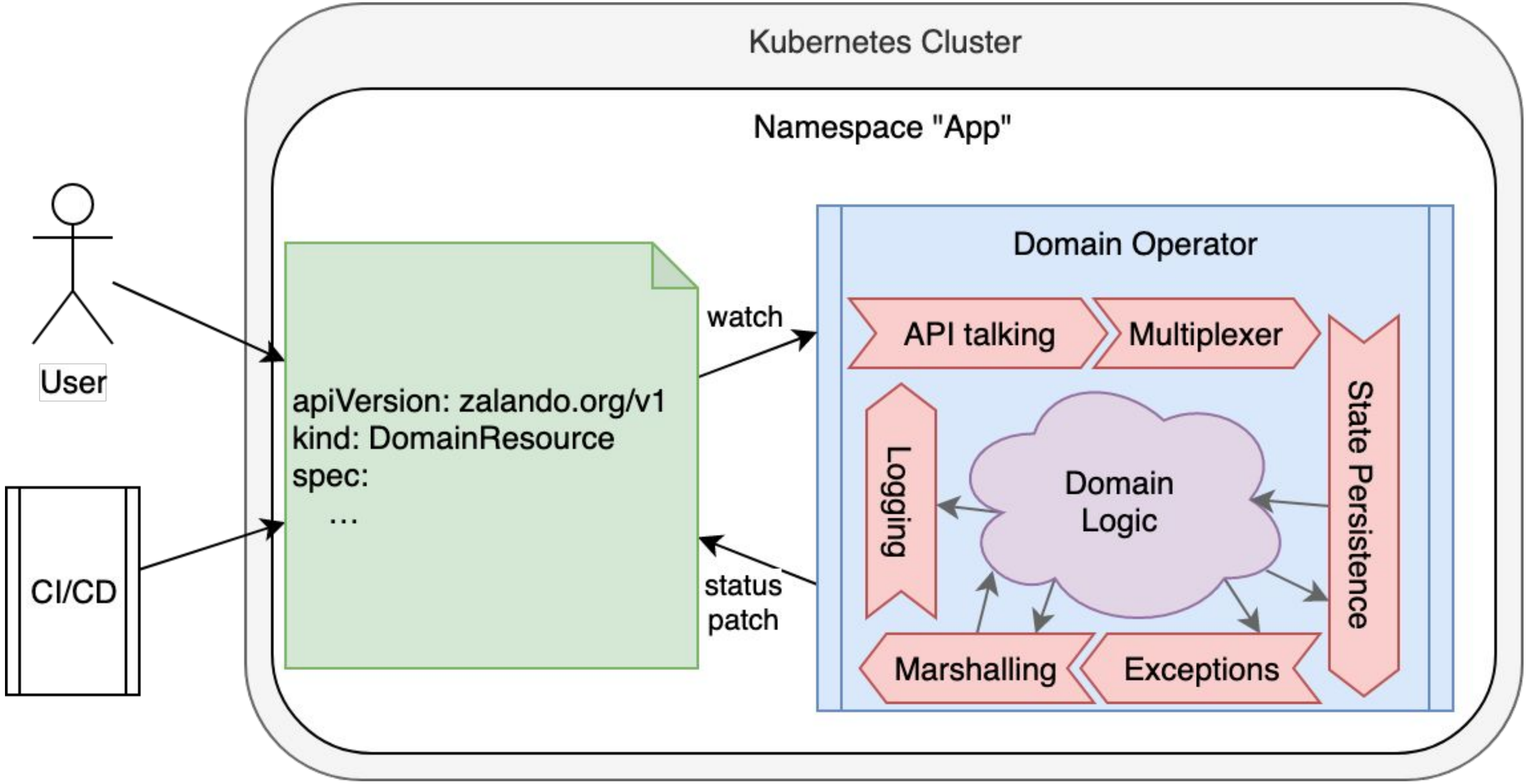
Common use: an application-specific operator



```
apiVersion: zalando.org/v1
kind: ElasticsearchDataSet
spec:
  replicas: 2
  scaling:
    enabled: true
    minReplicas: 1
    maxReplicas: 3
    ...
```

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

Problem: infrastructure code hassle

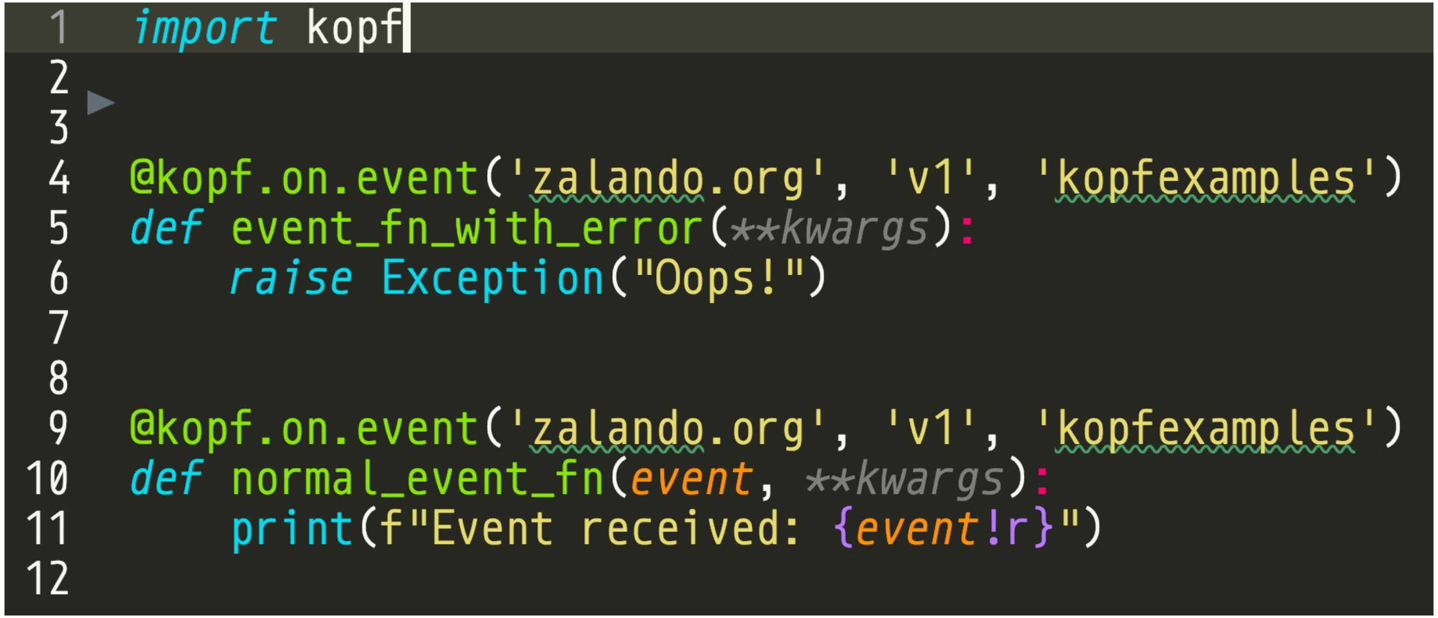


```
apiVersion: zalando.org/v1
kind: DomainResource
spec:
    ...
```

MAKE A FRAMEWORK!

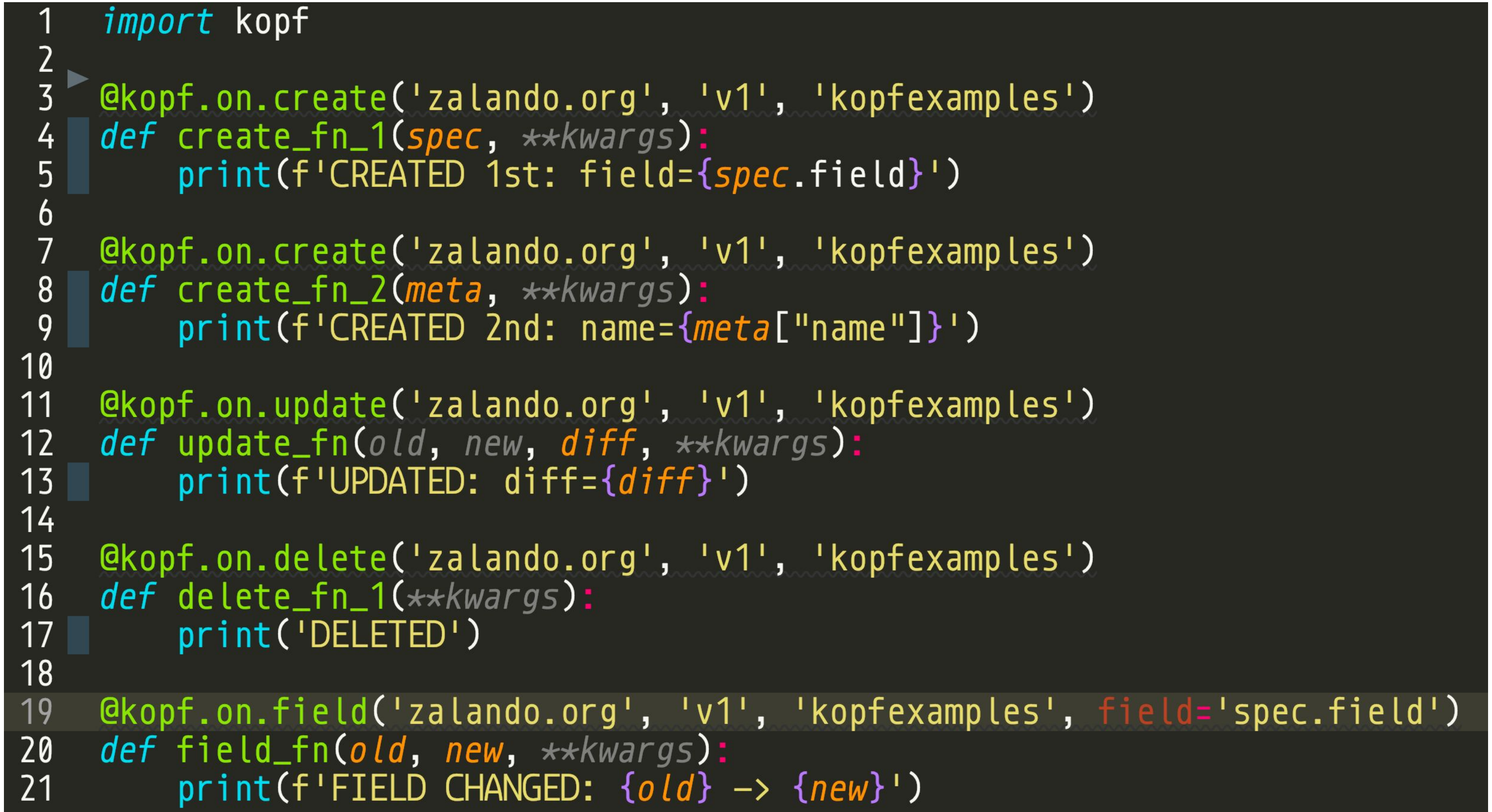
Kopf: simple spy-handlers

- As often, as the events arrive from K8s API.
- Raw payload, no interpretation.
- Fire-and-forget, ignore errors.



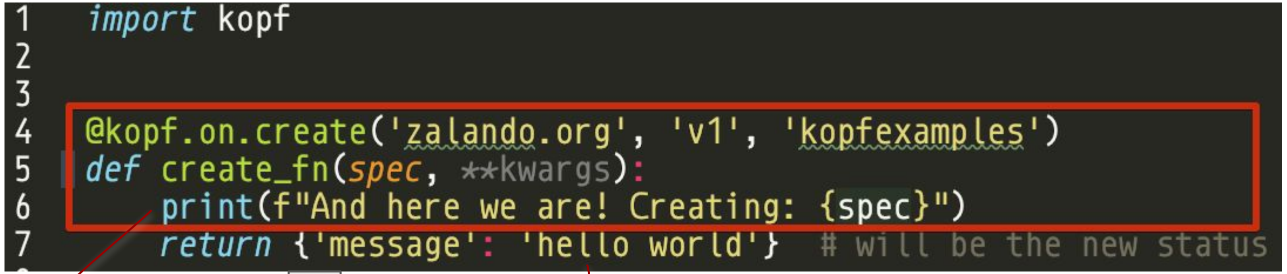
```python
import kopf


@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}")

```

Kopf: convenient cause- & diff-handlers

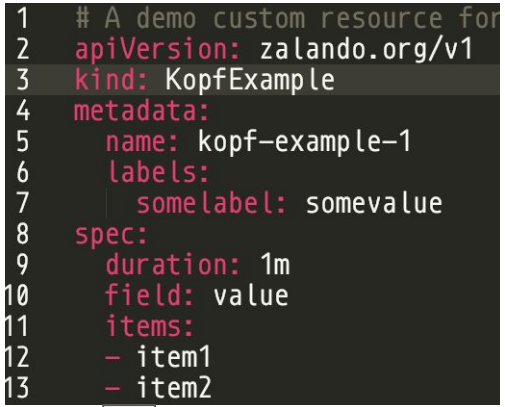


```python
import kopf

@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}')
```

Kopf: running from the development environment



```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    ]
[asf-preprocessing/kopf-example-1] Handler create_fn succeeded.

[2019-02-25 14:06:54,856] kopf.reactor.handlin [INFO    ]
[asf-preprocessing/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.
```

Features

- Custom & built-in resources supported (crds, pods, services, etc).
- Agnostic to API clients: kubernetes-client, pykube-ng, raw HTTP, etc.
- Immediate reaction to changes and events.
- Predefined behavioural patterns:
    - Simple spy-handlers for event watching.
    - Advanced cause & diff detection for actual change tracking.
    - Retry-until-success approach to handlers.
- Operator resilience:
    - Restores its state on restarts.
- Operator testing toolkit (minimally sufficient).

Roadmap

- Cross-handler communication and state management.
- Cross-object interactions & relations (e.g. children pods of a parent resource).
- Per-object thread/task orchestration handlers.

- Code generation (CRDs/RBAC/Dockerfile/etc), verification, packaging.
- Integration with Operator Lifecycle Manager.

- Your suggestions? ;-)

Links

- Kopf sources:
  - https://github.com/zalando-incubator/kopf
- Kopf documentation:
  - https://kopf.readthedocs.io/
- Kopf examples:
  - https://github.com/zalando-incubator/kopf/tree/master/examples
- Other Zalando operators:
  - https://github.com/zalando-incubator/es-operator
  - https://github.com/zalando/postgres-operator
- Me (Sergey Vasilyev):
  - https://twitter.com/nolar
  - https://www.linkedin.com/in/sergeyvasilyev/

THE END

(questions — later)