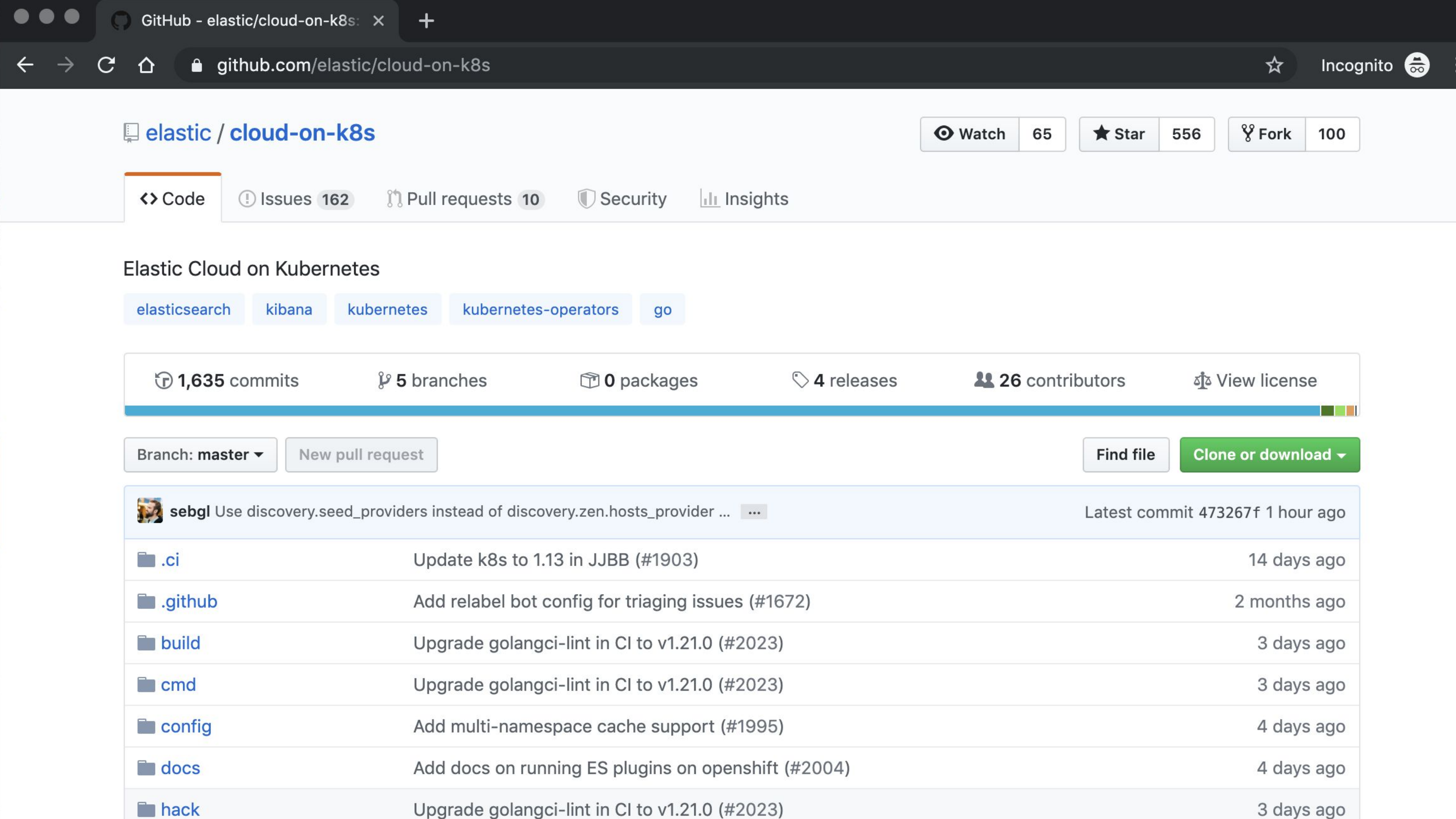
elastic

Writing a Kubernetes Operator: the Hard Parts

Sébastien Guilloux *(@_sebgl)*
KubeCon North America 2019 (San Diego)



GitHub - elastic/cloud-on-k8s

github.com/elastic/cloud-on-k8s

Incognito

elastic / cloud-on-k8s

Watch 65 | Star 556 | Fork 100

Code | Issues 162 | Pull requests 10 | Security | Insights

Elastic Cloud on Kubernetes

elasticsearch | kibana | kubernetes | kubernetes-operators | go

1,635 commits | 5 branches | 0 packages | 4 releases | 26 contributors | View license

Branch: master | New pull request | Find file | Clone or download

sebgl Use discovery.seed_providers instead of discovery.zen.hosts_provider ... Latest commit 473267f 1 hour ago

.ci	Update k8s to 1.13 in JJBB (#1903)	14 days ago
.github	Add relabel bot config for triaging issues (#1672)	2 months ago
build	Upgrade golangci-lint in CI to v1.21.0 (#2023)	3 days ago
cmd	Upgrade golangci-lint in CI to v1.21.0 (#2023)	3 days ago
config	Add multi-namespace cache support (#1995)	4 days ago
docs	Add docs on running ES plugins on openshift (#2004)	4 days ago
hack	Upgrade golangci-lint in CI to v1.21.0 (#2023)	3 days ago

Kubernetes Operators in a nutshell

elastic

Operators in a nutshell

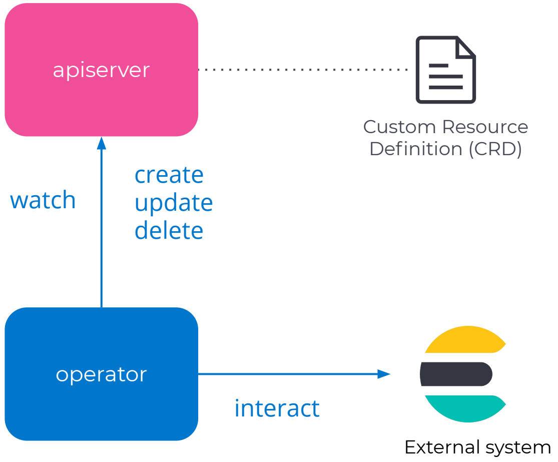
CRDs

apiserver
Custom Resource Definition (CRD)

```yaml
apiVersion:
elasticsearch.k8s.elastic.co/v1beta1
kind: Elasticsearch
metadata:
  name: elasticsearch-sample
spec:
  version: 7.4.0
  nodeSets:
  - name: master-nodes
    count: 3
    config:
       node.master: true
  - name: data-nodes
    count: 2
    config:
       node.data: true
```

Operators in a nutshell

Reconciliation loop



New event

A **watched resource** was created/updated/deleted

Reconcile!

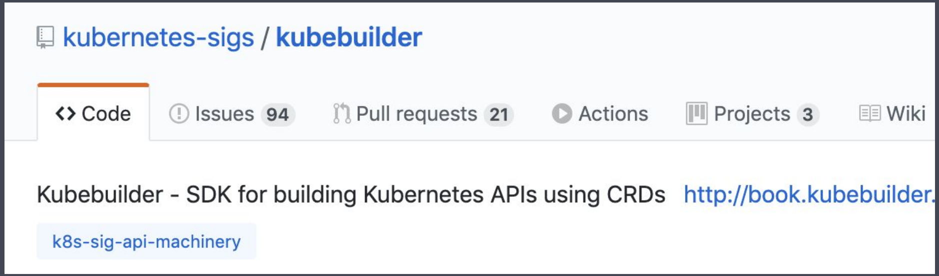
Get resource spec

Reconcile **Services, Secrets, Pods**, etc.

(maybe) Interact with an **external system**

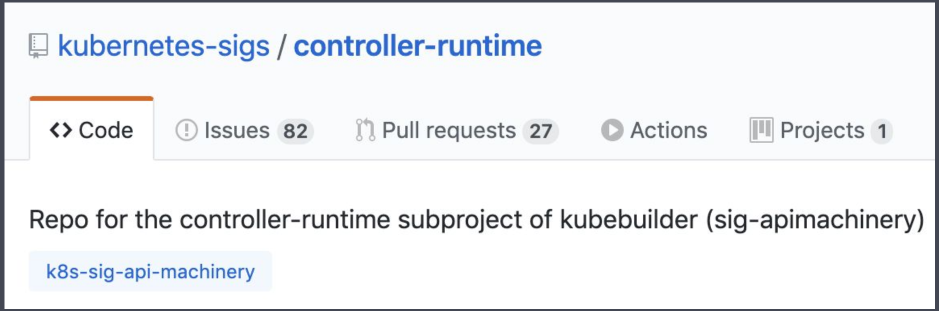
Sequential steps
Return early
Over and over again

Tools & libs


kubernetes-sigs / kubebuilder

Kubebuilder - SDK for building Kubernetes APIs using CRDs http://book.kubebuilder.

k8s-sig-api-machinery


kubernetes-sigs / controller-runtime

Repo for the controller-runtime subproject of kubebuilder (sig-apimachinery)

k8s-sig-api-machinery

The Hard Parts

(Well, some of them)

elastic

The operator lives in the past

Assume you're one step behind reality

```
// retrieve Pods
client.List(&pods)        // [podA, podB]
// we miss podC, create it
client.Create(&podC)
// retrieve Pods again
client.List(&pods)        // [podA, podB]
```

The apiserver client uses a **cached reader** (by default)

elastic

The operator lives in the past

So what? Examples from real life

The Infinite Pod Creation Loop
Pod missing? Create one.
Pod missing? Create one.

The Split Brain Situation
3 nodes? Quorum=2.
Add a 4th node. Quorum=3.
3 nodes? Quorum=2.

The Double Rolling Upgrade Reaction
Need to upgrade? Delete + Recreate Pods.
Need to upgrade? Delete + Recreate already upgraded Pods.

The operator lives in the past

What can we do?

Optimistic concurrency is good enough for most cases

```
> kubectl get pod my-pod -o json | jq .metadata
{
  “namespace”: “default”,
  “name”: “my-pod”,
  “uuid”: “6210565b-f985-11e9-8ca3-42010a8400bb”,
  “resourceVersion”: “3721702”
}
```

conflict on creation

conflict on update

The operator lives in the past

What can we do?

Optimistic concurrency is good enough for most cases

```
err := client.Delete(&pod, clientpkg.Preconditions{
    UID: &pod.UID,
    ResourceVersion: &pod.ResourceVersion,
})
if err != nil {
    return err
}
```

conflict on deletion

elastic

The operator lives in the past

What if we need some guarantees?

Sometimes this is **not enough**
Especially when dealing with stateful workloads

The operator lives in the past

What if we need some guarantees?

in-memory **expectations**

```
if !expectations.Satisfied() {
    // cache is not up-to-date yet
    return
}
err := client.Create(&pod)
// expect the Pod to be created
expectations.ExpectCreation(pod)
```

In the ReplicaSet controller *github.com/kubernetes/kubernetes/blob/master/pkg/controller/controller_utils.go*

In ECK *github.com/elastic/cloud-on-k8s/tree/master/pkg/controller/common/expectations*

The operator lives in the past

Best practices

Use **deterministic naming**

Always assume a **stale cache**

The entire reconciliation should be **idempotent**

Idempotent reconciliation

An example

```
if !exists(statefulSet) {
    err := c.Create(statefulSet)
    err := c.Create(headlessSvc)
}
```

The operator may crash here

Or return an error here

headlessSvc will never be created

Idempotent reconciliation

An example

```
if !exists(statefulSet) {
    err := c.Create(statefulSet)
    err := c.Create(headlessSvc)
}
```

```
if !exists(statefulSet) {
    err := c.Create(headlessSvc)
    err := c.Create(statefulSet)
}
```

Reorder operations

Idempotent reconciliation

An example

```
if !exists(statefulSet) {
    err := c.Create(statefulSet)
    err := c.Create(headlessSvc)
}
```

```
if !exists(statefulSet) {
    err := c.Create(headlessSvc)
    err := c.Create(statefulSet)
}
```

```
if !exists(statefulSet) {
    err := c.Create(statefulSet)
}
if !exists(headlessSvc) {
    err := c.Create(headlessSvc)
}
```

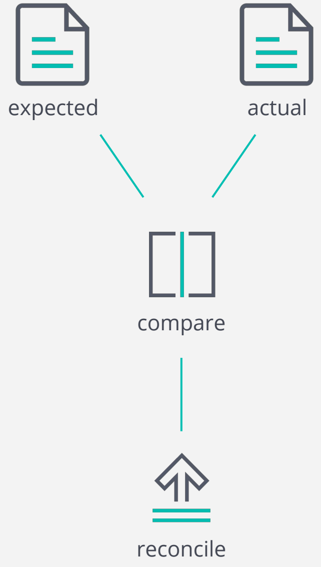
Decouple reconciliations

elastic

Reconciling resources

elastic

Reconciling resources



The deep way

```
if !reflect.DeepEqual(expected, actual) {
    // need to update
    // ...
}
```

Not a great fit for:

- metadata
- defaulted values

Defaulted values

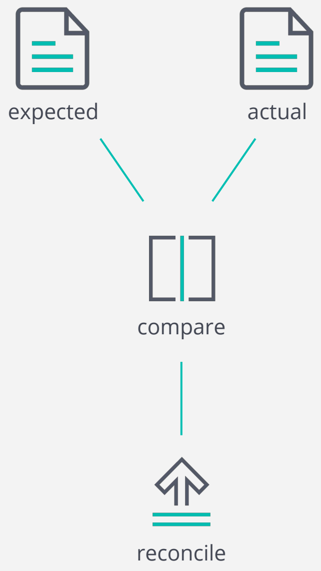
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: mypod
spec:
  containers:
  - name: busybox
    image: busybox
```

1. Create Pod

```yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: 2019-11-13T10:04:46Z
  namespace: default
  name: mypod
  uid: 052fa624-05fd-11ea-9ab1-42010a84001d
spec:
  containers:
  - name: busybox
    image: busybox
    imagePullPolicy: Always
    env:
    - name: KUBERNETES_PORT_443_TCP_ADDR
      value: c-111-dns-5e14.hcp.westus2.azmk8s.io
    resources:
      requests:
        cpu: 100m
  dnsPolicy: ClusterFirst
  securityContext: {}
```

2. Get Pod

Reconciling resources



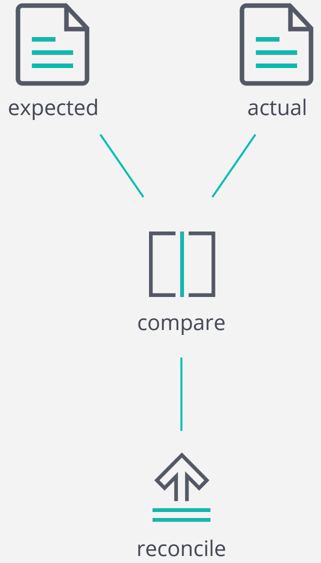
The hard way

```
if sameLabels(expected, actual) &&
   sameAnnotations(expected, actual) &&
   sameReplicas(expected, actual) &&
   sameResourcesLimits(expected, actual) &&
   sameEnvVars(expected, actual) && ...
```

Not a great fit for:

- unit tests
- PR reviewer brain
- real life
- defaulted values

Reconciling resources



The smart way

```
// annotate object with its hash
hash := HashObject(expected)
expected.Annotations[ResourceHash] = hash
// compare expected vs. actual hash
actualHash := actual.Annotations[ResourceHash]
if actualHash != hash {
   // need to update
}
```

Actual hash was **built at creation time**,
hence does not include defaulted fields

github.com/kubernetes/kubernetes/tree/master/pkg/util/hash
github.com/elastic/cloud-on-k8s/tree/master/pkg/controller/common/hash

elastic

```yaml
apiVersion: elasticsearch.k8s.elastic.co/v1beta1
kind: Elasticsearch
metadata:
  name: elasticsearch-sample
spec:
  version: 7.4.0
  nodeSets:
  - name: default
    count: 1
    podTemplate:
      metadata:
        labels: {"foo": "bar"}
      spec:
        affinity:
          nodeAffinity:
            requiredDuringSchedulingIgnoredDuringExecution:
              nodeSelectorTerms:
              - matchExpressions:
                - key: environment
                  operator: In
                  values: ["e2e", "production"]
        containers:
        - name: elasticsearch
          env:
          - name: ES_JAVA_OPTS
            value: "-Xms2g -Xmx4g"
```

Empower users

But provide good defaults

optional podTemplate

elastic

StatefulSets

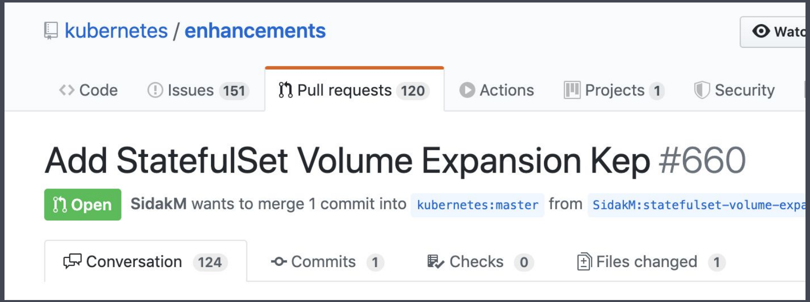
A few things to know about

StatefulSets

A few things to know about

No volume resize

KEP in progress



kubernetes / enhancements

Watc

Code | Issues 151 | Pull requests 120 | Actions | Projects 1 | Security

Add StatefulSet Volume Expansion Kep #660

Open SidakM wants to merge 1 commit into `kubernetes:master` from `SidakM:statefulset-volume-expa`

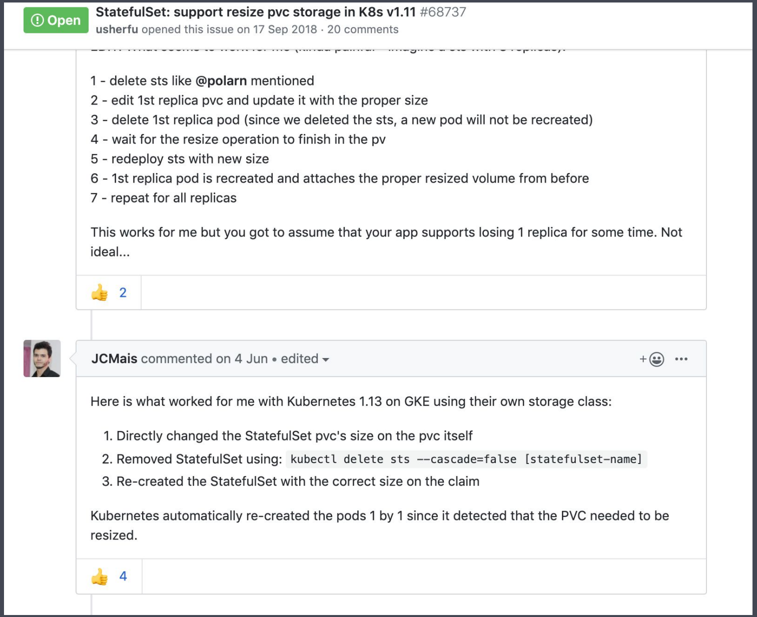
Conversation 124 | Commits 1 | Checks 0 | Files changed 1

StatefulSets

A few things to know about

No volume resize

There are workarounds



Open StatefulSet: support resize pvc storage in K8s v1.11 #68737

usherfu opened this issue on 17 Sep 2018 · 20 comments

1 - delete sts like @polarn mentioned
2 - edit 1st replica pvc and update it with the proper size
3 - delete 1st replica pod (since we deleted the sts, a new pod will not be recreated)
4 - wait for the resize operation to finish in the pv
5 - redeploy sts with new size
6 - 1st replica pod is recreated and attaches the proper resized volume from before
7 - repeat for all replicas

This works for me but you got to assume that your app supports losing 1 replica for some time. Not ideal...

👍 2

JCMais commented on 4 Jun • edited

Here is what worked for me with Kubernetes 1.13 on GKE using their own storage class:

1. Directly changed the StatefulSet pvc's size on the pvc itself
2. Removed StatefulSet using: `kubectl delete sts --cascade=false [statefulset-name]`
3. Re-created the StatefulSet with the correct size on the claim

Kubernetes automatically re-created the pods 1 by 1 since it detected that the PVC needed to be resized.

👍 4

StatefulSets

A few things to know about

Scheduling conflicts

Pod vs. PV

```
kind: StorageClass
metadata:
  name: my-storage-class
volumeBindingMode: WaitForFirstConsumer
```

elastic

StatefulSets

A few things to know about

Scheduling conflicts

Local Volumes vs. resources

StatefulSet **upgrade**:

1. **Delete** Pod
2. **Recreate** Pod
3. No resources available

1.5 **Another Pod** scheduled

elastic

StatefulSets

A few things to know about

Stateful workloads UpdateStrategy

Pick one

RollingUpdate

RollingUpdate.Partition

OnDelete

elastic

StatefulSets

A few things to know about

You **don't have to** use StatefulSets.

You **can** manage Pods and PVCs directly.

elastic

StatefulSets

A few things to know about

You **don’t have to** use StatefulSets.

You **can** manage Pods and PVCs directly.

We tried.

elastic

elastic

Writing a Kubernetes Operator: the Hard Parts

Sébastien Guilloux
KubeCon North America 2019 (San Diego)

Backup slides

Testing

How do you test that monster you ended up with?

- Unit test as much as possible
    - Fake client helps with k8s interactions
- Unit tests for the entire reconciliation are hard
    - Too many code paths to visit & things to mock
- Kubebuilder integration tests
    - Local apiserver + etcd process
    - Our CI had a hard time running ITs in parallel

elastic

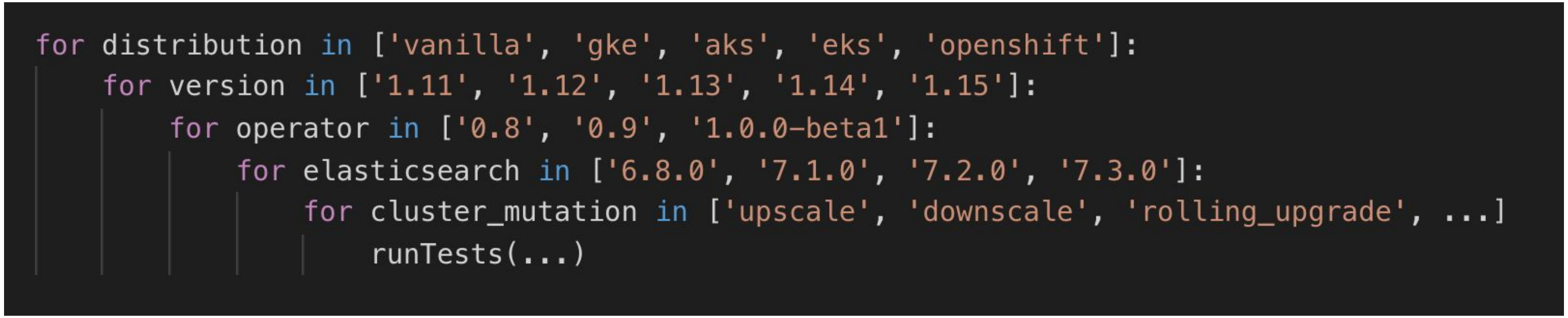
Testing

How do you test that monster you ended up with?

- End-to-end tests
  - 1. Spawn a k8s cluster
  - 2. Deploy the operator
  - 3. Run tests
    - Create an Elasticsearch cluster
    - Verify it’s available, with the expected spec
    - Mutate the cluster
    - Verify it eventually has the expected spec
    - Continuously ensure no downtime nor data loss during the mutation

Testing

Multidimensional E2E test matrix



```
for distribution in ['vanilla', 'gke', 'aks', 'eks', 'openshift']:
    for version in ['1.11', '1.12', '1.13', '1.14', '1.15']:
        for operator in ['0.8', '0.9', '1.0.0-beta1']:
            for elasticsearch in ['6.8.0', '7.1.0', '7.2.0', '7.3.0']:
                for cluster_mutation in ['upscale', 'downscale', 'rolling_upgrade', ...]
                    runTests(...)
```

Namespace management

Full flexibility

- One operator for the entire cluster
- One operator per namespace
- One operator for [*namespaceA, namespaceB*]
    - Thanks controller-runtime 0.2!
    - Need tooling for RBAC generation

elastic