



Testing Kubernetes Operator

Artem Nikitin
Software Engineer, @artemnikitin

Agenda

1 Introduction

2 Testing

3 CI

Acronyms

- K8s, k8s - Kubernetes
- CRD - CustomResourceDefinition, extension of Kubernetes API
- GKE - Google Kubernetes Engine
- OpenShift - Kubernetes-based platform from Red Hat
- EKS - Amazon Elastic Kubernetes Service
- AKS - Azure Kubernetes Service
- kind - tool for running local Kubernetes clusters using Docker containers

Introduction

What is Kubernetes?



SwiftOnSecurity

@SwiftOnSecurity

One time I tried to explain Kubernetes to someone.
Then we both didn't understand it.

What is Kubernetes?



Kelsey Hightower 

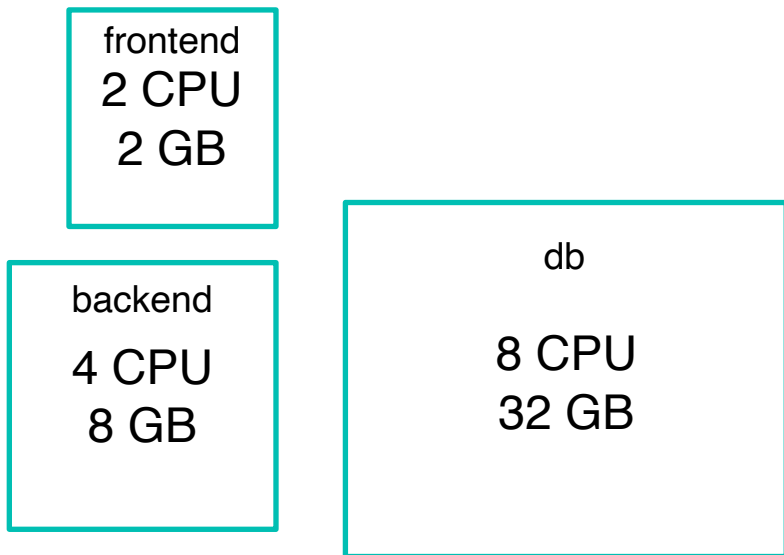
@kelseyhightower



Kubernetes is an infrastructure framework. It's YAML based configuration files and the kubectl command line tool make it approachable to developers, but far from the developer productivity you find in a PaaS or FaaS platform.

What is Kubernetes?

My explanation



Servers

Hardware or virtual servers with certain resources, like CPU, RAM, disk size, etc.

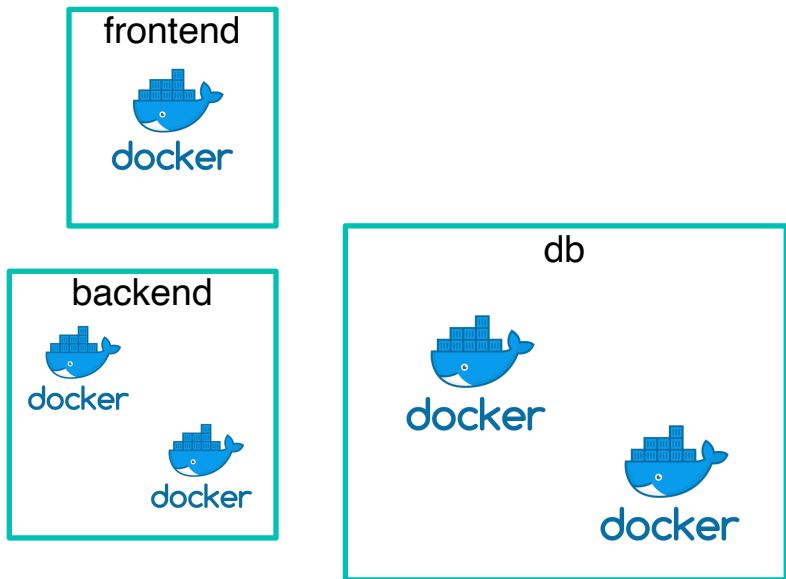


Management

Either via SSH or configuration management tools (Chef, Puppet, Ansible, etc.)

What is Kubernetes?

My explanation



Servers

We still have servers



Management

But we now running **Docker** images there!

What is Kubernetes?

My explanation

14 CPU, 42 GB RAM

frontend



docker

backend



docker



docker

db



docker



docker



Servers

We **don't care** about servers anymore



Management

We **don't care** about managing servers



Kubernetes

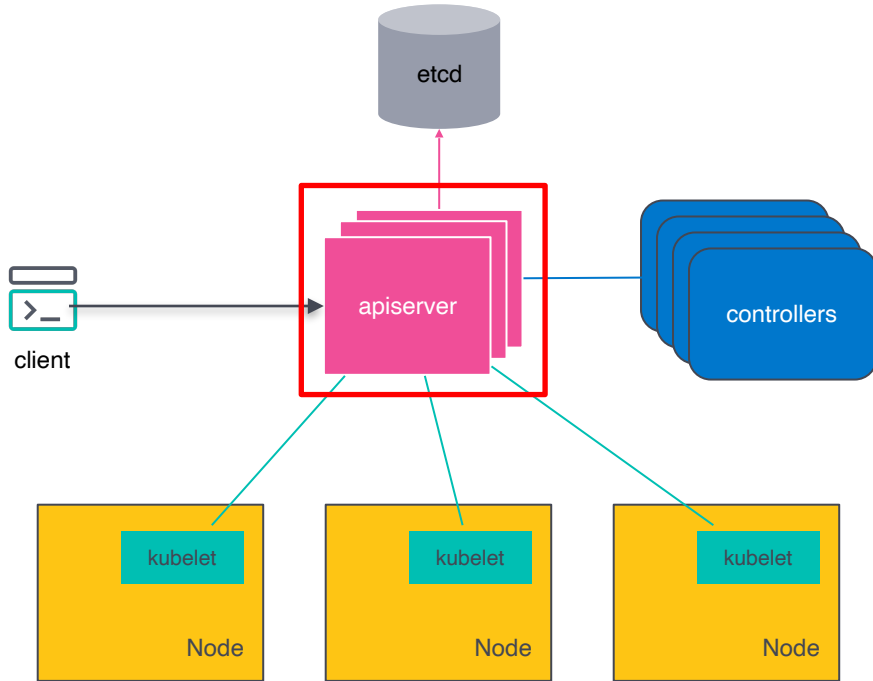
Now we are running **Docker** images on a **pool of resources!**

How to use Kubernetes

kubectl apply -f elasticsearch.yaml

```
apiVersion: elasticsearch.k8s.elastic.co/v1beta1
kind: Elasticsearch
metadata:
  name: quickstart
spec:
  version: 7.4.2
  nodeSets:
  - name: default
    count: 1
    config:
      node.master: true
      node.data: true
      node.ingest: true
      node.store.allow_mmap: false
```

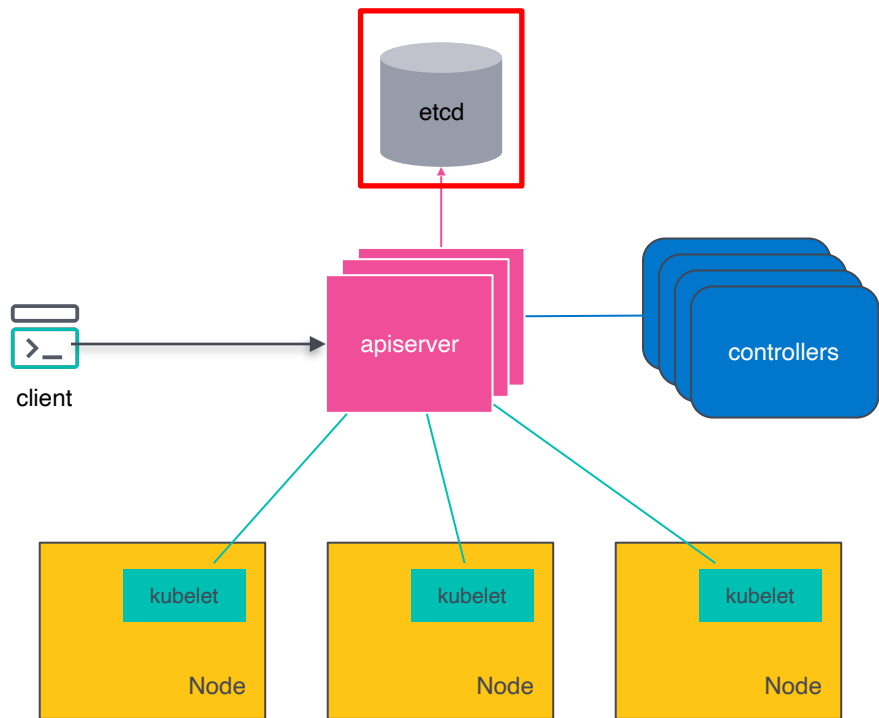
Kubernetes architecture



apiserver

API to create/update/delete k8s resources
Handles **authentication & authorization**
Horizontally **scalable**
With a **watch mechanism**

Kubernetes architecture



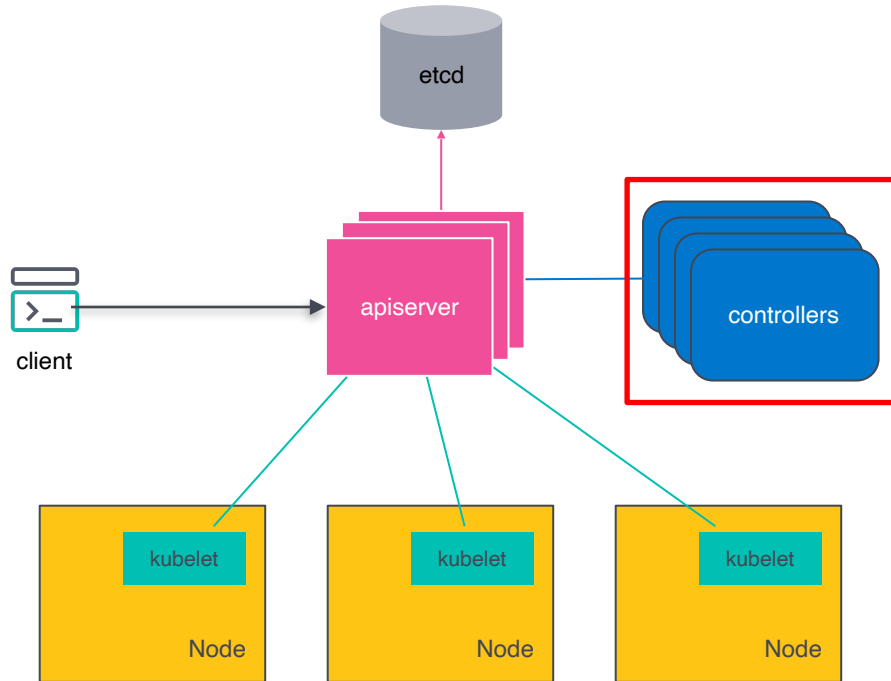
etcd

Persistent **distributed key-value store**, organized as a filesystem

Stores **all** k8s resources

With a **watch mechanism**

Kubernetes architecture



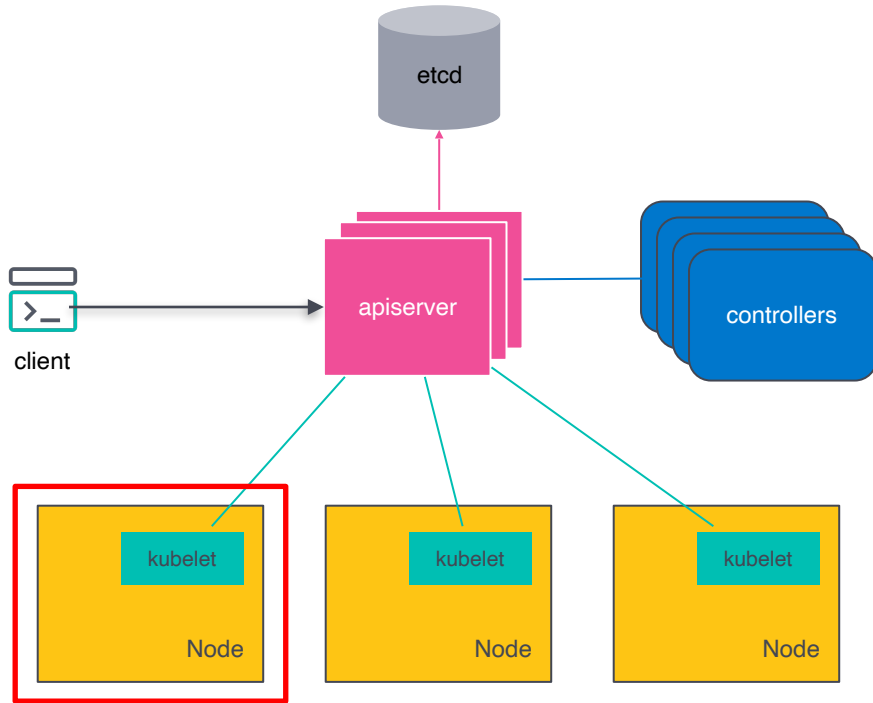
controllers

Watch resources in the apiserver

Reacts on resource changes

May interact with **external systems**

Kubernetes architecture



kubelet

Agent running on each Node

Watches Pods in the apiserver

Manages corresponding containers on the **host**

Operators in a nutshell



Kelsey Hightower 

@kelseyhightower



Kubernetes has made huge improvements in the ability to run stateful workloads including databases and message queues, but I still prefer not to run them on Kubernetes.

Operators in a nutshell

- Since Kubernetes 1.7
- Technically, it's yet another controller
- Using mostly for stateful apps

Operators in a nutshell

Wait... It sounds like a Helm Chart

- Add the elastic helm charts repo

```
helm repo add elastic https://helm.elastic.co
```

- Install it

```
helm install --name elasticsearch elastic/elasticsearch
```

<https://github.com/elastic/helm-charts>

Operators in a nutshell

Operator or Helm Chart?

- Helm is a package manager. Think of it like apt for Kubernetes.
- Operators enable you to manage the operation of applications within Kubernetes.
- From <https://news.ycombinator.com/item?id=16969495>

Operators in a nutshell

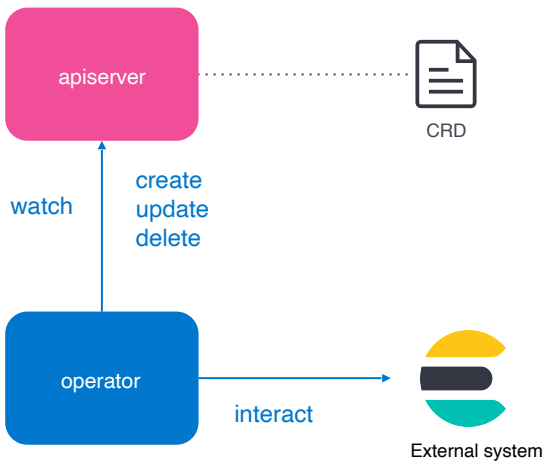
CRDs



```
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

Testing

Unit and integration tests

How do you test that monster you ended up with?

- Unit test as much as possible
 - Fake client helps with k8s interactions
- Integration tests
 - Local **apiserver** + **etcd** process
 - Might be flaky, example: <https://github.com/kubernetes-sigs/controller-runtime/pull/510>

Unit tests: example

```
func TestGarbageCollectPVCs(t *testing.T) {
    // Test_pvcstoRemove covers most of the testing logic,
    // let's just check everything is correctly plugged to the k8s api here.
    es := v1beta1.Elasticsearch{ObjectMeta: metav1.ObjectMeta{Namespace: "ns", Name: "es"}}
    existingPVCS := []runtime.Object{
        buildPVCPtr( name: "claim1-sset1-0", // should not be removed
        buildPVCPtr( name: "claim1-oldsset-0", // should be removed
    }
    actualSsets := sset.StatefulSetList{buildSsetWithClaims( name: "sset1", replicas: 1, claims...: "claim1")}
    expectedSsets := sset.StatefulSetList{buildSsetWithClaims( name: "sset2", replicas: 1, claims...: "claim1")}
    k8sClient := k8s.WrappedFakeClient(existingPVCS...)
    err := GarbageCollectPVCs(k8sClient, es, actualSsets, expectedSsets)
    require.NoError(t, err)

    var retrievedPVCs corev1.PersistentVolumeClaimList
    require.NoError(t, k8sClient.List(&retrievedPVCs))
    require.Equal(t, expected: 1, len(retrievedPVCs.Items))
}
```

Integration tests: example

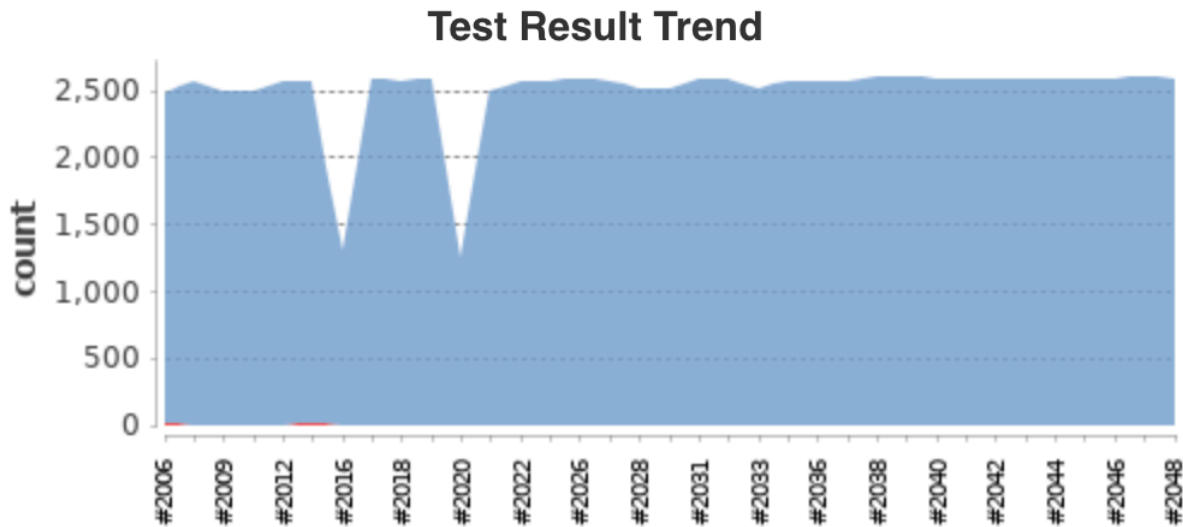
```
// RunWithK8s starts a local Kubernetes server and runs tests in m.
func RunWithK8s(m *testing.M, crdPath string) {

// TestDynamicEnqueueRequest tests the integration between a DynamicEnqueueRequest watch and
// a manager + controller, with a test k8s environment.
// The test just checks that everything fits together and reconciliations are correctly triggered
// from the EventHandler. More detailed behaviour is tested in `handler_test.go`.
func TestDynamicEnqueueRequest(t *testing.T) {
```


Unit and integration tests

Some stats

- ~2500 unit/integration tests
- 3-4 min to run them all



E2E Tests

How do you test that monster you ended up with?

E2E tests in a nutshell:

- Spawn a k8s cluster
- Deploy the operator
- 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

E2E Tests: Test Runner

<https://github.com/elastic/cloud-on-k8s/blob/master/test/e2e/cmd/run/command.go#L66>

```
cmd.Flags().BoolVar(&flags.autoPortForwarding, name: "auto-port-forwarding", value: false, usage: "Enable port forwarding to pods")
cmd.Flags().DurationVar(&flags.commandTimeout, name: "command-timeout", 90*time.Second, usage: "Timeout for commands executed")
cmd.Flags().StringVar(&flags.e2eImage, name: "e2e-image", value: "", usage: "E2E test image")
cmd.Flags().StringVar(&flags.elasticStackVersion, name: "elastic-stack-version", value: "7.1.1", usage: "Elastic stack version")
cmd.Flags().StringVar(&flags.kubeConfig, name: "kubeconfig", value: "", usage: "Path to kubeconfig")
cmd.Flags().BoolVar(&flags.local, name: "local", value: false, usage: "Create the environment for running tests locally")
cmd.Flags().StringSliceVar(&flags.managedNamespaces, name: "managed-namespaces", []string{"mercury", "venus"}, usage: "List of managed namespaces")
cmd.Flags().StringVar(&flags.operatorImage, name: "operator-image", value: "", usage: "Operator image")
cmd.Flags().BoolVar(&flags.skipCleanup, name: "skip-cleanup", value: false, usage: "Do not run cleanup actions after test run")
cmd.Flags().StringVar(&flags.testContextOutPath, name: "test-context-out", value: "", usage: "Write the test context to the given path")
cmd.Flags().StringVar(&flags.testLicense, name: "test-license", value: "", usage: "Test license to apply")
cmd.Flags().StringVar(&flags.scratchDirRoot, name: "scratch-dir", value: "/tmp/eck-e2e", usage: "Path under which temporary files should be created")
cmd.Flags().StringVar(&flags.testRegex, name: "test-regex", value: "", usage: "Regex to pass to the test runner")
cmd.Flags().StringVar(&flags.testRunName, name: "test-run-name", randomTestRunName(), usage: "Name of this test run")
cmd.Flags().StringVar(&flags.crdFlavor, name: "crd-flavor", value: "default", usage: "CRD flavor to install")
cmd.Flags().DurationVar(&flags.testTimeout, name: "test-timeout", 5*time.Minute, usage: "Timeout before failing a test")
cmd.Flags().BoolVar(&flags.logToFile, name: "log-to-file", value: false, usage: "Specifies if should log test output to file. Disabled by default.")
```

E2E Tests: Test Runner

<https://github.com/elastic/cloud-on-k8s/blob/master/test/e2e/cmd/run/run.go#L60>

```
// CI test run steps
steps = []stepFunc{
    helper.createScratchDir,
    helper.initTestContext,
    helper.initTestSecrets,
    helper.createE2ENamespaceAndRoleBindings,
    helper.installCRDs,
    helper.createOperatorNamespaces,
    helper.createManagedNamespaces,
    helper.deployGlobalOperator,
    helper.deployNamespaceOperator,
    helper.deployTestJob,
    helper.runTestJob,
}
}
```

```
defer helper.runCleanup()
```

E2E Tests: Example test

https://github.com/elastic/cloud-on-k8s/blob/master/test/e2e/es/failure_test.go#L19

```
func TestKillOneDataNode(t *testing.T) {
    // 1 master + 2 data nodes
    b := elasticsearch.NewBuilder( name: "test-failure-kill-a-data-node").
        WithESMasterNodes( count: 1, elasticsearch.DefaultResources).
        WithESDataNodes( count: 2, elasticsearch.DefaultResources)

    matchDataNode := func(p corev1.Pod) bool {
        return label.IsDataNode(p) && !label.IsMasterNode(p)
    }

    test.RunRecoverableFailureScenario(t,
        test.KillNodeSteps(matchDataNode, test.ESPodListOptions(b.Elasticsearch.Namespace, b.Elasticsearch.Name)...),
        b)
}
```

E2E Tests: KillNodeSteps

https://github.com/elastic/cloud-on-k8s/blob/master/test/e2e/test/run_failure.go#L59

```
return StepList{
{
  Name: "Kill a node",
  Test: func(t *testing.T) {
    pods, err := k.GetPods(opts...)
    require.NoError(t, err)
    var found bool
    killedPod, found = GetFirstPodMatching(pods, podMatch)
    require.True(t, found)
    err = k.DeletePod(killedPod)
    require.NoError(t, err)
  },
},
{
  Name: "Wait for pod to be deleted",
  Test: Eventually(func() error {
    pod, err := k.GetPod(killedPod.Namespace, killedPod.Name)
    if err != nil && !apierrors.IsNotFound(err) : err ↗
    if apierrors.IsNotFound(err) || killedPod.UID != pod.UID : nil ↗
    return fmt.Errorf("pod #{killedPod.Name} not deleted yet")
  }),
},
}
```

E2E Tests: TestKillOneDataNode in reality

```
---- PASS: TestKillOneDataNode (147.64s)
---- PASS: TestKillOneDataNode/K8S_should_be_accessible (0.01s)
---- PASS: TestKillOneDataNode/Elasticsearch_CRDs_should_exist (0.01s)
---- PASS: TestKillOneDataNode/Remove_Elasticsearch_if_it_already_exists (0.01s)
---- PASS: TestKillOneDataNode/Creating_an_Elasticsearch_cluster_should_succeed (0.05s)
---- PASS: TestKillOneDataNode/Elasticsearch_cluster_should_be_created (0.02s)
---- PASS: TestKillOneDataNode/ES_certificate_authority_should_be_set_and_deployed (3.01s)
---- PASS: TestKillOneDataNode/All_expected_Pods_should_eventually_be_ready (66.51s)
---- PASS: TestKillOneDataNode/ES_version_should_be_the_expected_one (0.01s)
---- PASS: TestKillOneDataNode/ES_services_should_be_created (0.01s)
---- PASS: TestKillOneDataNode/ES_pods_should_eventually_have_a_certificate (0.02s)
---- PASS: TestKillOneDataNode/ES_services_should_have_endpoints (0.01s)
---- PASS: TestKillOneDataNode/ES_cluster_health_should_eventually_be_green (0.00s)
---- PASS: TestKillOneDataNode/Elastic_password_should_be_available (0.00s)
---- PASS: TestKillOneDataNode/Elasticsearch_data_volumes_should_be_of_the_specified_type (0.01s)
---- PASS: TestKillOneDataNode/ES_nodes_topology_should_eventually_be_the_expected_one (0.14s)
---- PASS: TestKillOneDataNode/ES_version_should_be_the_expected_one#01 (0.03s)
---- PASS: TestKillOneDataNode/ES_endpoint_should_eventually_be_reachable (0.03s)
```

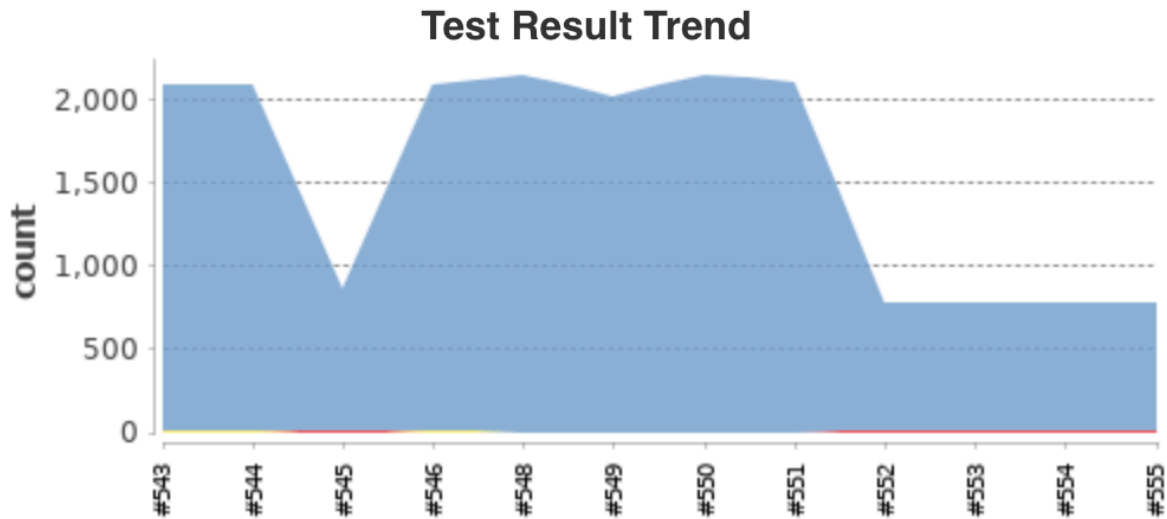
E2E Tests: TestKillOneDataNode in reality

```
--- PASS: TestKillOneDataNode/Kill_a_node (0.02s)
--- PASS: TestKillOneDataNode/Wait_for_pod_to_be_deleted (9.05s)
--- PASS: TestKillOneDataNode/ES_certificate_authority_should_be_set_and_deployed#01 (0.01s)
--- PASS: TestKillOneDataNode/All_expected_Pods_should_eventually_be_ready#01 (31.68s)
--- PASS: TestKillOneDataNode/ES_version_should_be_the_expected_one#02 (0.01s)
--- PASS: TestKillOneDataNode/ES_services_should_be_created#01 (0.01s)
--- PASS: TestKillOneDataNode/ES_pods_should_eventually_have_a_certificate#01 (0.02s)
--- PASS: TestKillOneDataNode/ES_services_should_have_endpoints#01 (0.01s)
--- PASS: TestKillOneDataNode/ES_cluster_health_should_eventually_be_green#01 (0.00s)
--- PASS: TestKillOneDataNode/Elastic_password_should_be_available#01 (0.00s)
--- PASS: TestKillOneDataNode/Elasticsearch_data_volumes_should_be_of_the_specified_type#01 (0.01s)
--- PASS: TestKillOneDataNode/ES_nodes_topology_should_eventually_be_the_expected_one#01 (0.05s)
--- PASS: TestKillOneDataNode/ES_version_should_be_the_expected_one#03 (0.13s)
--- PASS: TestKillOneDataNode/ES_endpoint_should_eventually_be_reachable#01 (0.13s)
--- PASS: TestKillOneDataNode/Deleting_Elasticsearch_should_return_no_error (0.01s)
--- PASS: TestKillOneDataNode/Elasticsearch_should_not_be_there_anymore (0.00s)
--- PASS: TestKillOneDataNode/Elasticsearch_pods_should_be_eventually_be_removed (36.10s)
--- PASS: TestKillOneDataNode/PVCs_should_eventually_be_removed (0.01s)
```

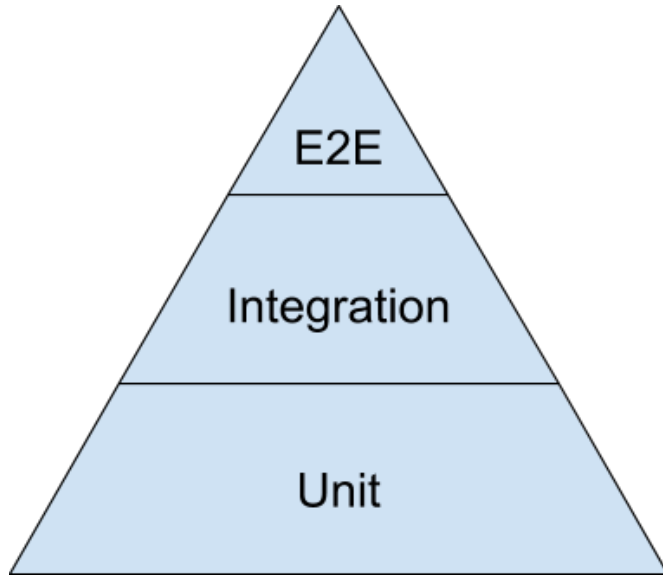

E2E Tests

Some stats

- ~2000 E2E tests
- 2 - 2.5 hours to run them all (sequentially, on GKE)

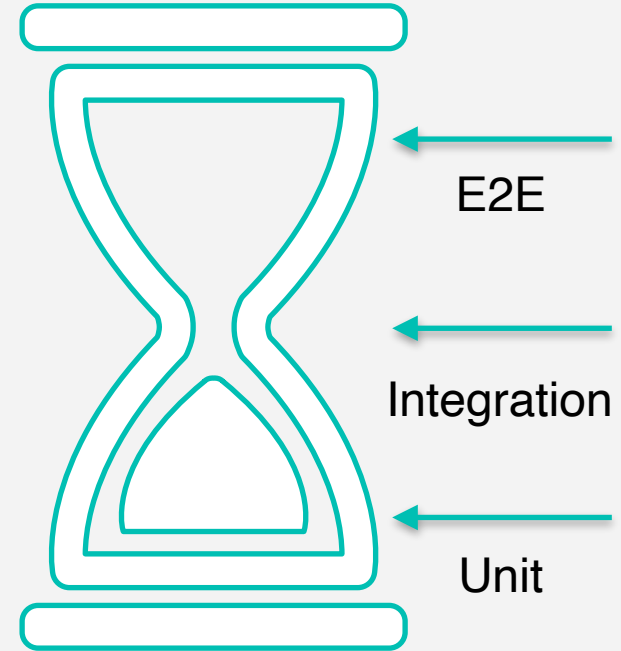


Your typical test pyramid



<https://blog.primehammer.com/test-pyramid/>

Our test hourglass



ACCORDING TO THIS



YOU'RE A HERETIC

Why?!

Burn the heretic!!!

- Unit/integration tests for the entire reconciliation are hard
 - Too many code paths to visit & things to mock
- No guarantees that it will work on a real k8s cluster

Why?!

The operator lives in the past



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.

Why?!

AKS inserts default values

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

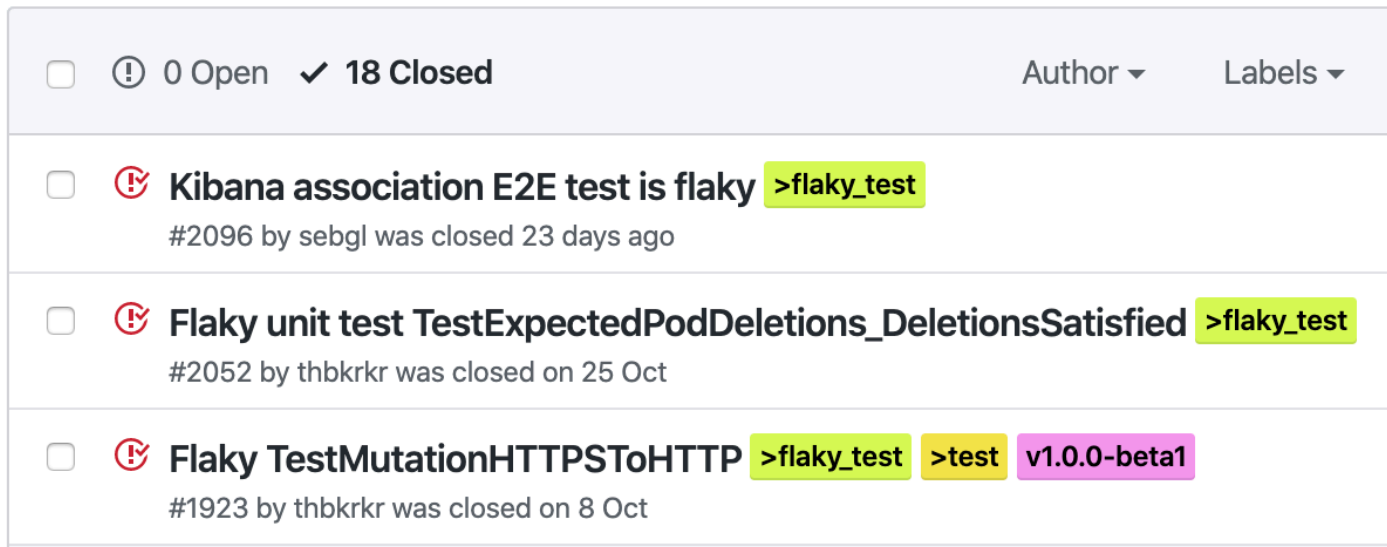
1. Create Pod

```
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

Flaky tests

- Usually they point to potential issues or misconfiguration!
- https://github.com/elastic/cloud-on-k8s/issues?q=is%3Aopen+is%3Aissue+label%3A%3Eflaky_test



A screenshot of a GitHub issues page showing a list of closed issues. The header indicates 0 Open and 18 Closed issues. The issues listed are:

- Kibana association E2E test is flaky** >flaky_test
#2096 by sebg1 was closed 23 days ago
- Flaky unit test TestExpectedPodDeletions_DeletionsSatisfied** >flaky_test
#2052 by thbkrkr was closed on 25 Oct
- Flaky TestMutationHTTPSToHTTP** >flaky_test >test v1.0.0-beta1
#1923 by thbkrkr was closed on 8 Oct

Flaky tests

How do we deal with it

- Fix it :)
- Use a tool to get debug info from K8s cluster

<https://github.com/elastic/cloud-on-k8s/blob/master/hack/eck-dump.sh>

```
if (env.SHELL_EXIT_CODE != 0) {  
    failedTests.addAll(lib.getListOfFailedTests())  
    googleStorageUpload bucket: "gs://devops-ci-artifacts/jobs/$JOB_NAME/$BUILD_NUMBER",  
        credentialsId: "devops-ci-gcs-plugin",  
        pattern: "*.tgz",  
        sharedPublicly: true,  
        showInline: true  
}
```


Flaky tests










How we are going to deal with it (in the future)

- Instrumentation for tests and Operator
- Send test results and k8s cluster data to Elasticsearch cluster for aggregation and analyze





















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(...)
```

Multidimensional E2E test matrix

	kind	OpenShift	GKE	EKS	AKS
1.11					
1.12					
1.13					
1.14					
1.15					
1.16					

Multidimensional E2E test matrix

	0.8.0	0.8.1	0.9.0	1.0.0-beta1	1.0.0
6.8					
7.1					
7.2					
7.3					
7.4					
7.5					

CI



Kelsey Hightower 

@kelseyhightower



If you don't have a CI system capable of building your application, then Kubernetes is the least of your problems. Focus on CI first.



Kelsey Hightower ✓

@kelseyhightower



There is no single continuous integration and delivery setup that will work for everyone. You are essentially trying to automate your company's culture using bash scripts.

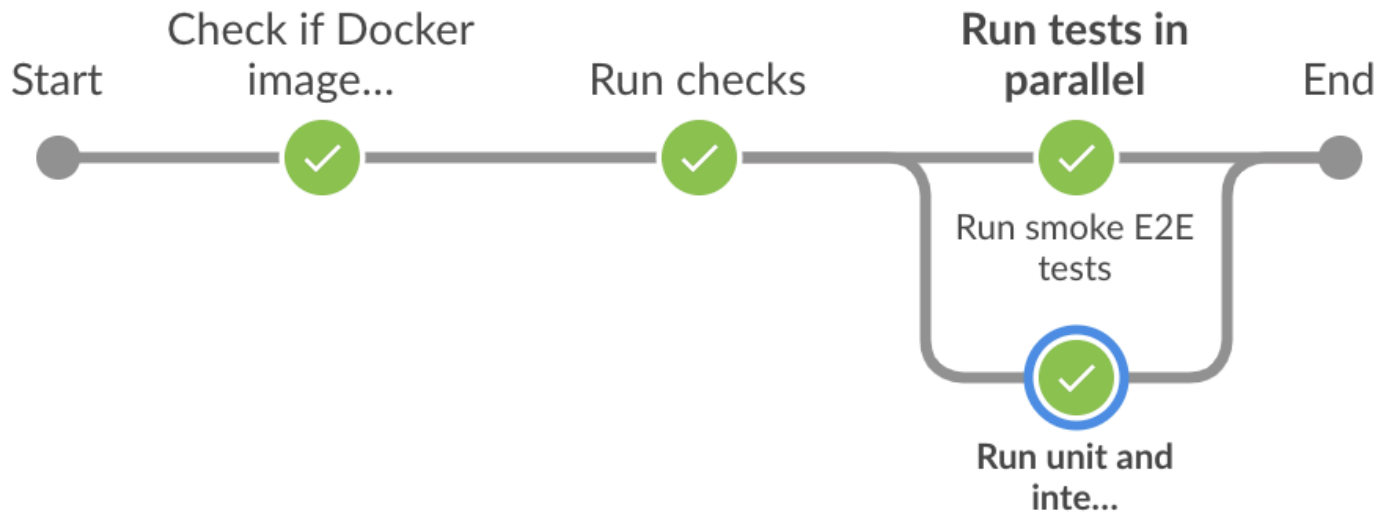
Pre-commit verification

<https://devops-ci.elastic.co/view/cloud-on-k8s/job/cloud-on-k8s-pr/>

- <https://github.com/elastic/cloud-on-k8s/blob/master/build/ci/pr/Jenkinsfile>
- Triggered by Github PR
- Run unit and integration tests, linters, smoke E2E test, verifying Jenkins pipelines

Pre-commit verification

<https://devops-ci.elastic.co/view/cloud-on-k8s/job/cloud-on-k8s-pr/>



Pre-commit verification

<https://devops-ci.elastic.co/view/cloud-on-k8s/job/cloud-on-k8s-pr/>

Declarative: Checkout SCM	Check if Docker image needs rebuilding	Run checks	Run tests in parallel	Run unit and integration tests	Run smoke E2E tests	Declarative: Post Actions
15s	4s	3min 24s	61ms	8min 18s	19min 30s	866ms
7s	3s	2min 50s	63ms	8min 11s	16min 59s	845ms
8s	3s	4min 5s	61ms	7min 12s	17min 10s	1s
6s	3s	3min 28s	60ms	7min 25s	16min 13s	885ms
18s	3s	3min 30s	61ms	9min 12s	16min 33s	821ms
6s	3s	3min 26s	62ms	10min 15s	17min 11s	843ms

Pre-commit verification

CI job evolution

- Only unit and integration tests
- Smoke E2E test
- Linters
- Docs
- Optimisation for Docker image
- xUnit compatible test output

Optimising build scripts

- Building and pushing the same Docker image for 4 times in a row 🤯

```
# Run e2e tests in a dedicated gke cluster,  
# that we delete if everything went fine  
ci-e2e:  
    $(MAKE) bootstrap-gke dep-vendor-only docker-build docker-push deploy e2e
```

Re-using Docker images

- <https://github.com/elastic/cloud-on-k8s/blob/master/build/ci/Makefile#L22>

```
CI_IMAGE ?= docker.elastic.co/eck/eck-ci:${shell \  
md5sum $(ROOT_DIR)/go.mod $(ROOT_DIR)/build/ci/Dockerfile | awk '{print $$1}' | md5sum | awk '{print $$1}'
```

Re-using Docker images

- <https://github.com/elastic/cloud-on-k8s/blob/master/build/ci/Makefile#L46>

```
# reads Docker password from Vault,  
# checks if Docker image exists by trying to pull it. If there is no image, then build and push it.  
ci-build-image:  
  @ docker pull $(CI_IMAGE) || (docker build -f $(ROOT_DIR)/build/ci/Dockerfile -t push.$(CI_IMAGE) \  
    --label "commit.hash=$(shell git rev-parse --short --verify HEAD)" $(ROOT_DIR) && docker login -u $(DOCKER_LOGIN) \  
    -p $(shell VAULT_TOKEN=$(VAULT_TOKEN) vault read -address=$(VAULT_ADDR) -field=value secret/devops-ci/cloud-on-k8s/eckadmin) \  
    push.docker.elastic.co && docker push push.$(CI_IMAGE))
```

Caching Docker images on CI

- https://github.com/elastic/cloud-on-k8s/blob/master/.ci/packer_cache.sh

```
# This script used to "bake" Docker images into base image for Jenkins nodes.
```

```
set -eou pipefail
```

```
DOCKER_CI_IMAGE=$(cd build/ci/ && make show-image)
```

```
declare -a docker_images=("$DOCKER_CI_IMAGE" "kindest/node:v1.11.10" "kindest/node:v1.15.3")
```

```
# Pull all the required docker images
```

```
for image in "${docker_images[@]}"
```

```
do
```

```
    docker pull "$image"
```

```
done
```

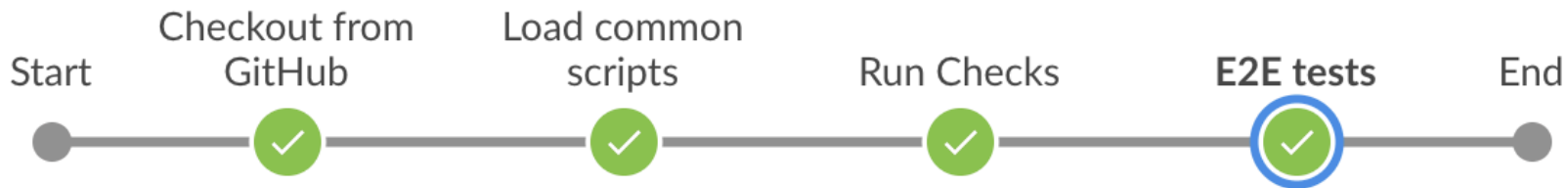
Post-commit verification

<https://devops-ci.elastic.co/view/cloud-on-k8s/job/cloud-on-k8s-e2e-tests/>

- <https://github.com/elastic/cloud-on-k8s/blob/master/build/ci/e2e/Jenkinsfile>
- Triggered by merge in master
- Run E2E tests on a real cluster in GKE
- Tests runs as Job in K8s cluster <https://kubernetes.io/docs/concepts/workloads/controllers/jobs-run-to-completion/>

Post-commit verification

<https://devops-ci.elastic.co/view/cloud-on-k8s/job/cloud-on-k8s-e2e-tests/>



Post-commit verification

<https://devops-ci.elastic.co/view/cloud-on-k8s/job/cloud-on-k8s-e2e-tests/>

Declarative: Checkout SCM	Checkout from GitHub	Load common scripts	Run Checks	E2E tests	Declarative: Post Actions
24s	2s	614ms	3min 34s	1h 57min	543ms
7s	4s	589ms	4min 3s	1h 8min failed	744ms
37s	1s	532ms	3min 24s	2h 13min	459ms
53s	2s	606ms	3min 45s	2h 11min	450ms
5s	1s	522ms	3min 6s	2h 12min	440ms
41s	2s	563ms	3min 41s	2h 12min	445ms

Issues with Cloud: Fail

Run out of instances in AZ in GCP

- GCP run out of instances in one of AZ in europe-west3 region
- Can't bootstrap GKE cluster anymore
- CI jobs started to fail massively

Issues with Cloud: Solution

Run out of instances in AZ in GCP

- Switch to different region
- Select region randomly before cluster creation (on roadmap)

Issues with Cloud: Fail

GKE fails to remove resources after deleting cluster

- Accidentally, we found 800+ existed but unused disks :)
- Later we found orphaned load balancers
- And some more resources

Issues with Cloud: Solution

GKE fails to remove resources after deleting cluster

- Add tool to check for existence of unused resources and remove them

Issues with cloud: Fail

Cleanup tool for GKE deleted disks in use

- Related to refactoring and switch to StatefulSet's
- During cluster upgrade disk might be considered orphaned
- And it will be removed by tool
- We unintentionally introduced some chaos testing into our tests :)

Issues with Cloud: Solution

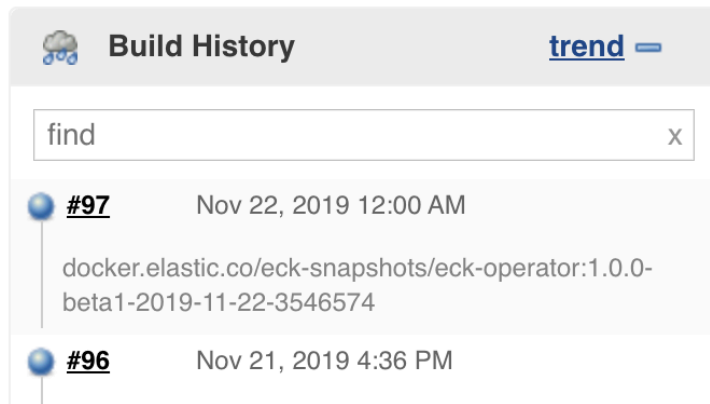
Cleanup tool for GKE deleted disks in use

- Link disk name to CI job name
- Clean disks for particular CI job name

Nightly builds

<https://devops-ci.elastic.co/view/cloud-on-k8s/job/cloud-on-k8s-nightly/>

- <https://github.com/elastic/cloud-on-k8s/blob/master/build/ci/nightly/Jenkinsfile>
- Triggered nightly during working days
- Builds snapshot version and pushes it to docker.elastic.co

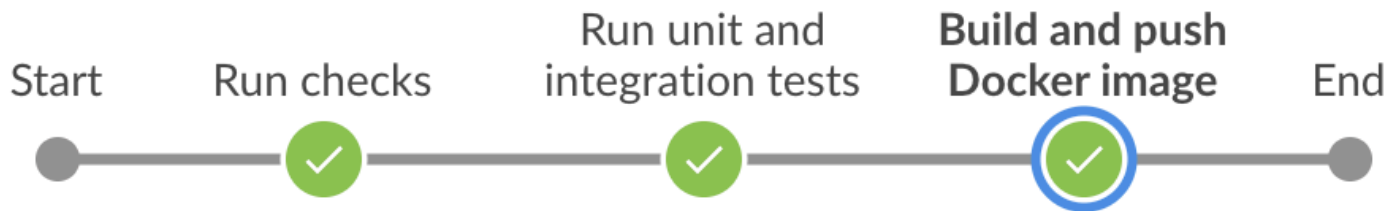


The screenshot shows the Jenkins 'Build History' page. At the top, there is a search bar containing the text 'find' and a close button 'x'. Below the search bar, there are two build entries. The first entry is for build #97, dated Nov 22, 2019 12:00 AM, with the image name 'docker.elastic.co/eck-snapshots/eck-operator:1.0.0-beta1-2019-11-22-3546574'. The second entry is for build #96, dated Nov 21, 2019 4:36 PM.

Build Number	Timestamp	Image Name
#97	Nov 22, 2019 12:00 AM	docker.elastic.co/eck-snapshots/eck-operator:1.0.0-beta1-2019-11-22-3546574
#96	Nov 21, 2019 4:36 PM	

Nightly build

<https://devops-ci.elastic.co/view/cloud-on-k8s/job/cloud-on-k8s-nightly/>



Nightly build

<https://devops-ci.elastic.co/view/cloud-on-k8s/job/cloud-on-k8s-nightly/>

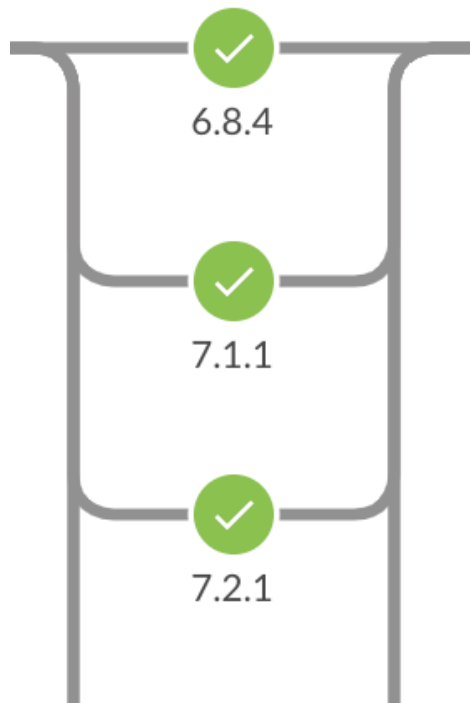
```
success {
  script {
    def image = readFile("$WORKSPACE/eck_image.txt").trim()
    currentBuild.description = image

    build job: 'cloud-on-k8s-versions-gke',
           parameters: [string(name: 'IMAGE', value: image)],
           wait: false

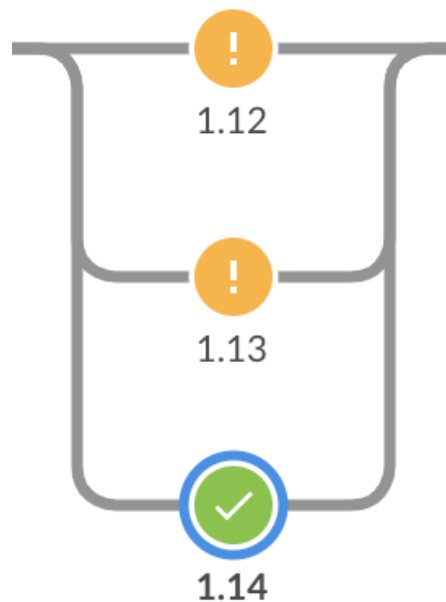
    build job: 'cloud-on-k8s-stack',
           parameters: [string(name: 'IMAGE', value: image)],
           wait: false
  }
}
```

Test matrix - different flavours of the same E2E tests

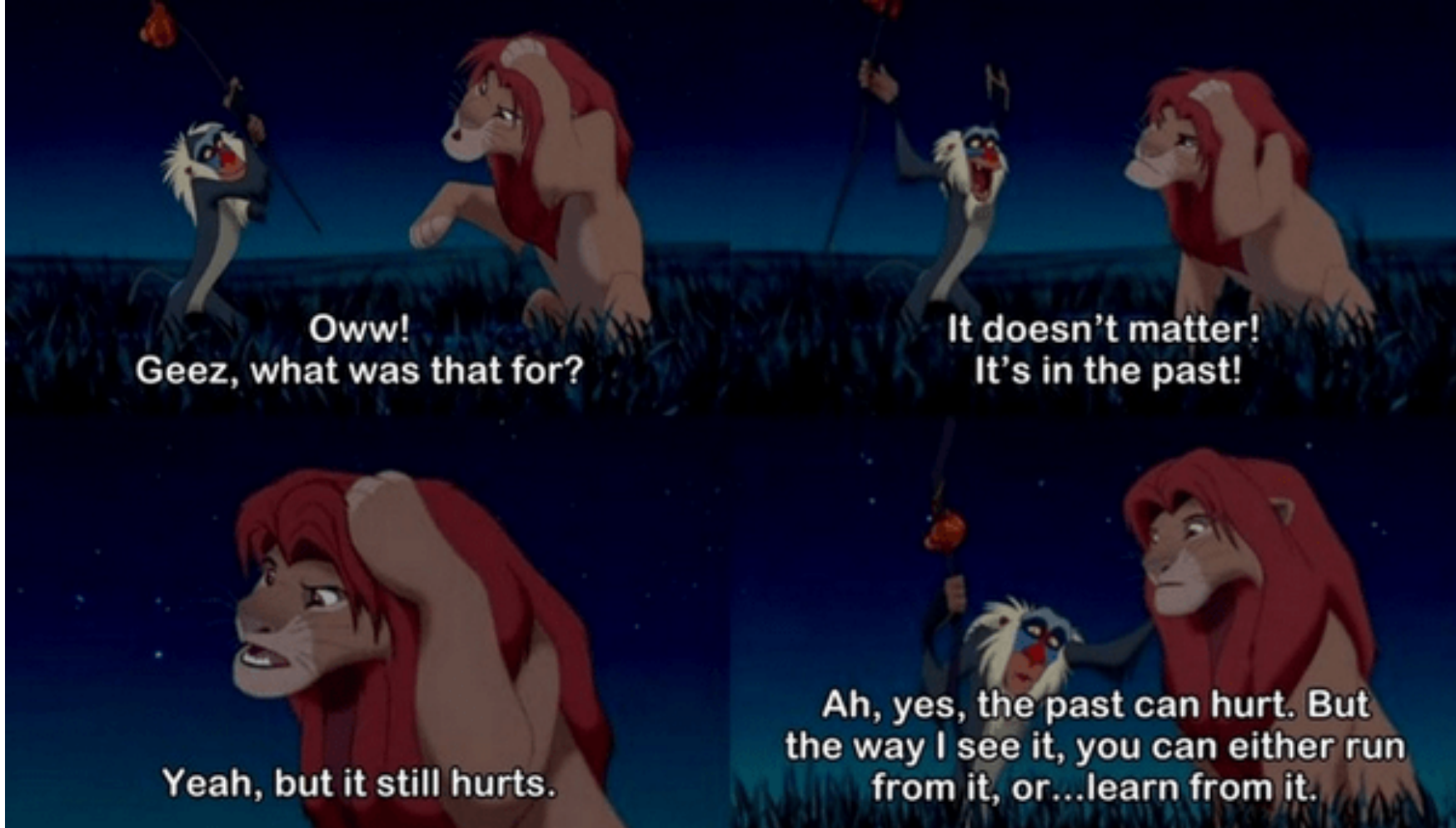
Run tests for
different ELK st...



Run tests for
different k8s ver...



And finally...



Thank you!

hi@artemnikitin.com



[artemnikitin](https://twitter.com/artemnikitin)



[artemnikitin](https://github.com/artemnikitin)