Who needs a KUTTL?

@devgerred
@kensipe
D2iQ

Gerred Dillon

Senior Staff Engineer

Kubernetes, KUDO, KUTTL

Developer: Clojure, Go

@devgerred
gdillon@d2iq.com

Ken Sipe

Distributed Application Engineer
And Orchestration Conductor

Apache Mesos, Kubernetes, KUDO, KUTTL

Developer: Java, Go, Scala, Groovy, C, C++, C#

@KenSipe
ken@d2iq.com

What is KUTTL

KUbernetes Test TooL (kuttl)

KUTTL Origins

Kubernetes Universal **Declarative** Operator (KUDO)

Declarative Testing

What is KUTTL

~~Unit~~

Integration

e2e

Who Needs a KUTTL?

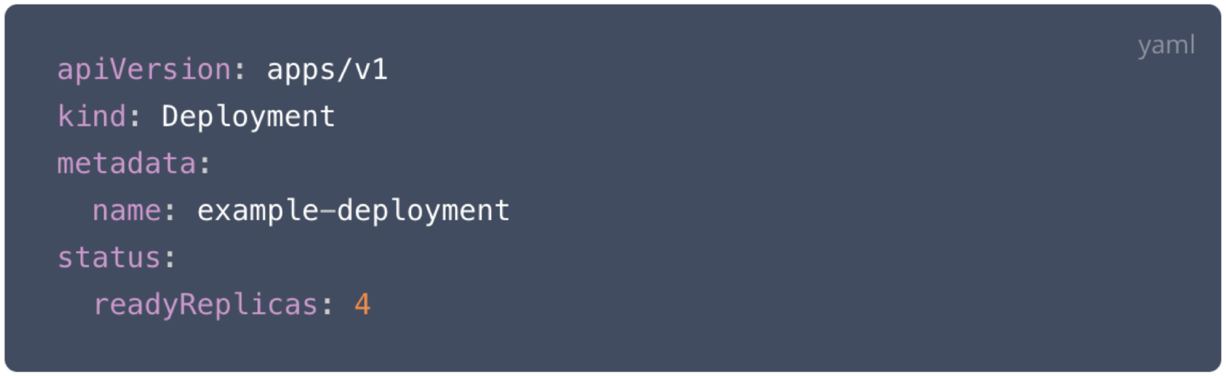
Testing harness to **declarative** test:

- operators
- KUDO
- helm charts
- any other Kubernetes applications or controllers.

Who Needs a KUTTL?

write portable end-to-end, integration, and conformance tests for Kubernetes without needing to write any code

Assert:



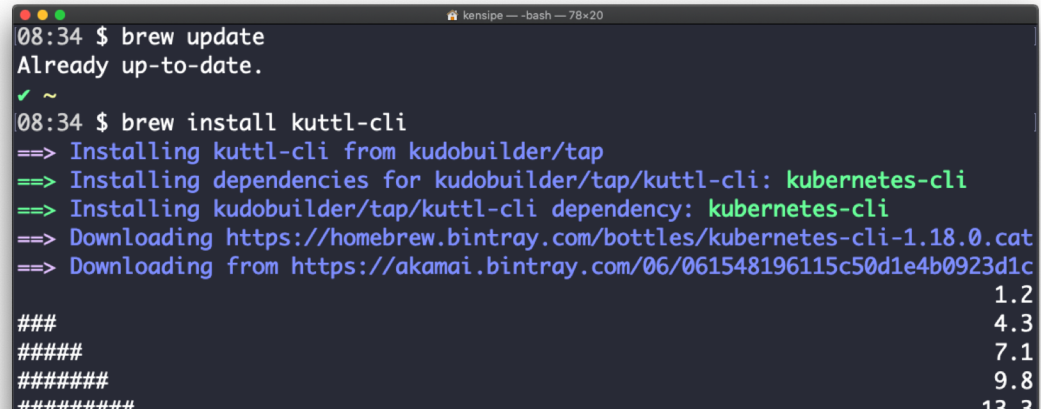
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: example-deployment
status:
  readyReplicas: 4
```

How Do I Start KUTTLing?

```
brew install kuttl-cli

kubectl krew install kuttl-cli*

API Integration
go get github.com/kudobuilder/kuttl
```



```
08:34 $ brew update
Already up-to-date.
✔ ~
08:34 $ brew install kuttl-cli
==> Installing kuttl-cli from kudobuilder/tap
==> Installing dependencies for kudobuilder/tap/kuttl-cli: kubernetes-cli
==> Installing kudobuilder/tap/kuttl-cli dependency: kubernetes-cli
==> Downloading https://homebrew.bintray.com/bottles/kubernetes-cli-1.18.0.cat
==> Downloading from https://akamai.bintray.com/06/061548196115c50d1e4b0923d1c
                                                                           1.2
###                                                                        4.3
#####                                                                      7.1
#######                                                                    9.8
```

KUTTL Abstract

“Kuttling releases a cocktail of hormones in our brains including dopamine, serotonin and oxytocin. It can lower your blood pressure and heart rate.”

-- Wikipedia

The First KUTTLers

Justin Taylor-Barrick	https://twitter.com/justinmbarrick
Gerred Dillon	https://twitter.com/devgerred
Tom Runyon	https://twitter.com/tommyrunyon
Ken Sipe	https://twitter.com/kensipe
Zain Malik	https://twitter.com/zMalikShxil

Special thanks to the rest of the KUDO team

Alena Varkockova	https://twitter.com/alenkacz
Aleksey Dukhovniy	https://kudo.dev/community/team/alex.html
Jan Schlicht	https://kudo.dev/community/team/jan.html
Marcin Owsiany	https://twitter.com/porridge80
Andreas Neumann	

https://kudo.dev/community/team/

Agenda

- Why Would you KUTTL?
- Your first KUTTL
- Ways to KUTTL
- Where do you want to KUTTL?
- Autonomy of a KUTTL
- KUTTLing an Operator
- KUTTL in Action
- Future KUTTLing

Why Would you **KUTTL?**

KUTTL Origins

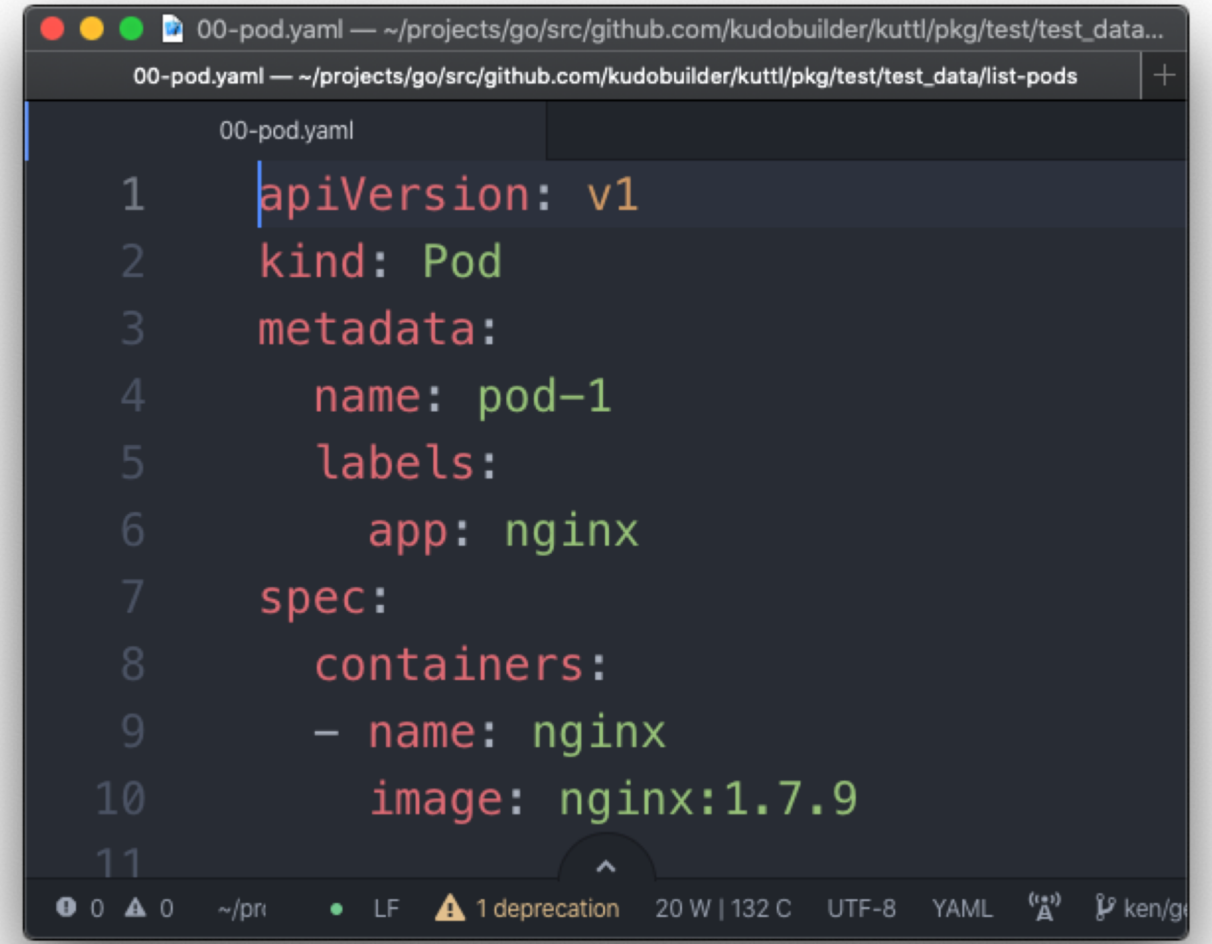
Kubernetes Universal **Declarative** Operator (KUDO)

Declarative Testing

Declarative Testing

What does that mean?

Test Setup

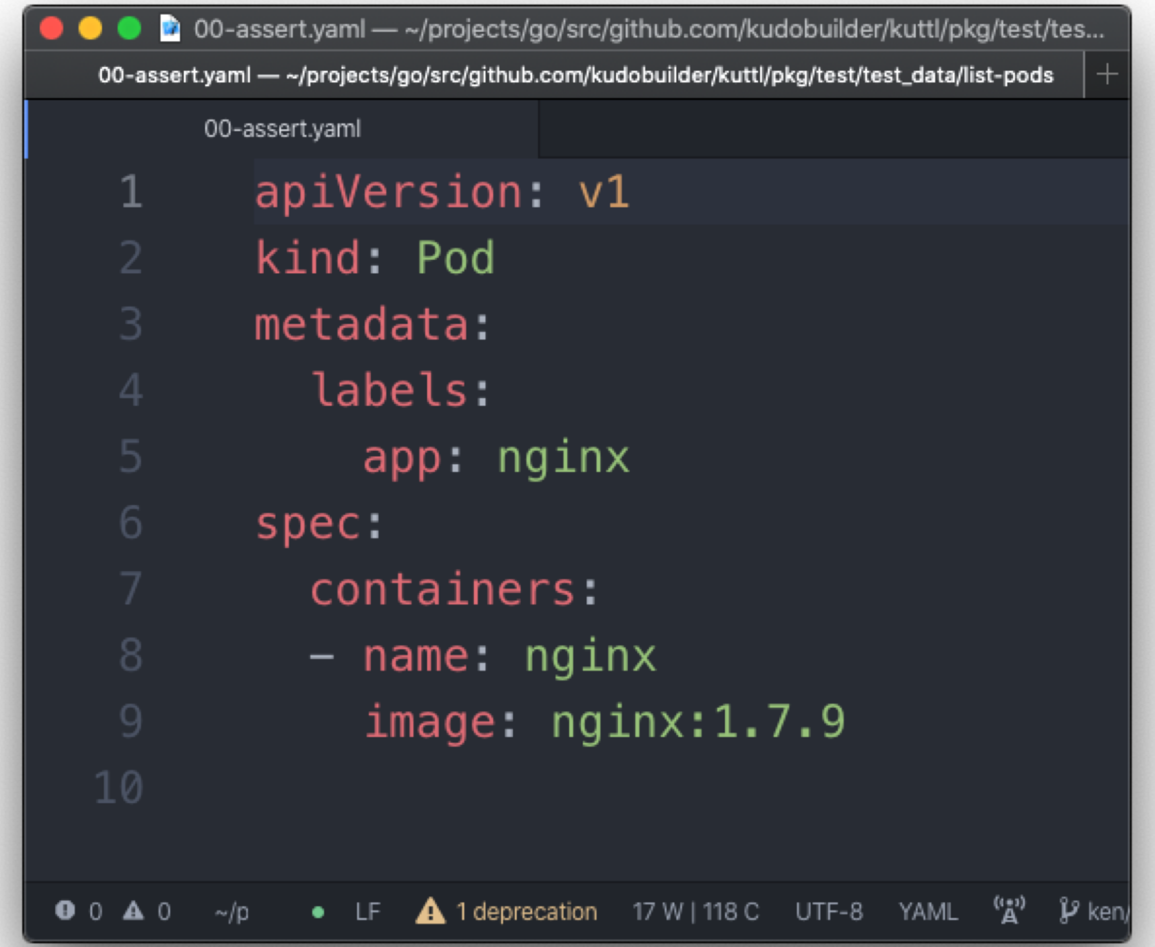

00-pod.yaml — ~/projects/go/src/github.com/kudobuilder/kuttl/pkg/test/test_data...
00-pod.yaml — ~/projects/go/src/github.com/kudobuilder/kuttl/pkg/test/test_data/list-pods
00-pod.yaml
0 0 ~/pr LF 1 deprecation 20 W | 132 C UTF-8 YAML ken/g

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-1
  labels:
    app: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.7.9
```

Declarative Testing

What does that mean?

Assert!



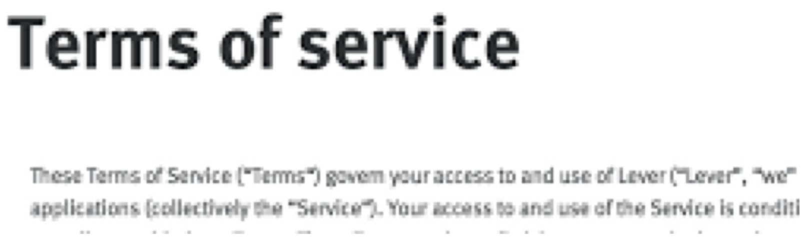
00-assert.yaml — ~/projects/go/src/github.com/kudobuilder/kuttl/pkg/test/tes...

00-assert.yaml — ~/projects/go/src/github.com/kudobuilder/kuttl/pkg/test/test_data/list-pods

00-assert.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    app: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.7.9

```


Terms of service

These Terms of Service ("Terms") govern your access to and use of Lever ("Lever", "we"
applications (collectively the "Service"). Your access to and use of the Service is conditi

TestSuite
- A collection of Tests

Test
- A collection of TestSteps

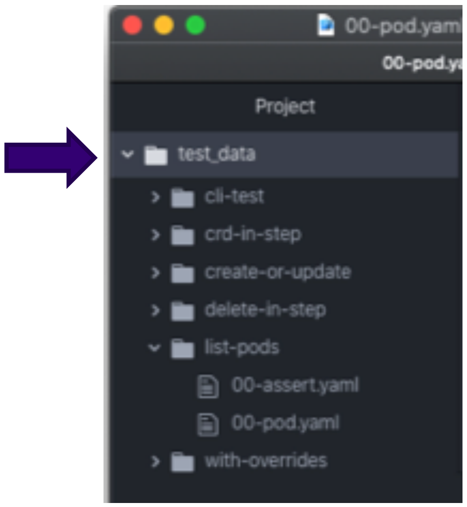
TestStep
- A "Step" in a Test
- A Collection of declarative CRUD
- Usually has an assert or error defined

TestAssert
- Assert conditions

TestSuite

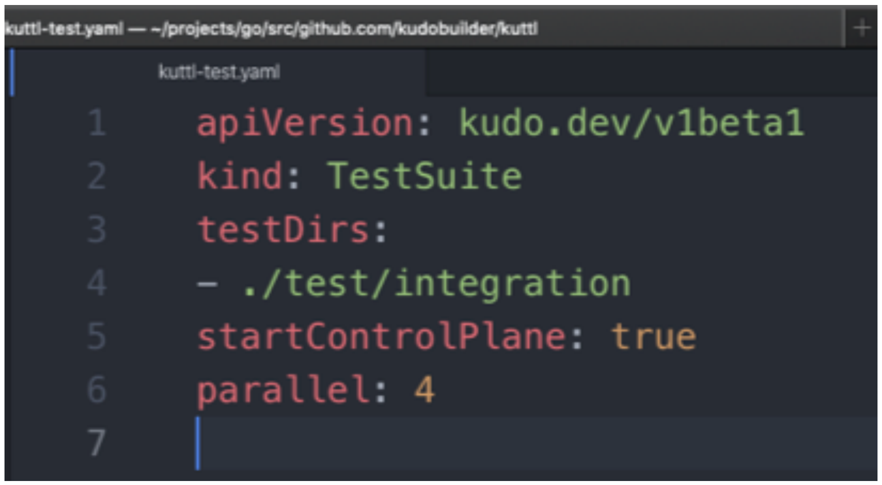
2 Concepts define a **TestSuite**

Folder of Tests



```
Project
test_data
  cli-test
  crd-in-step
  create-or-update
  delete-in-step
  list-pods
    00-assert.yaml
    00-pod.yaml
  with-overrides
```

Configuration File

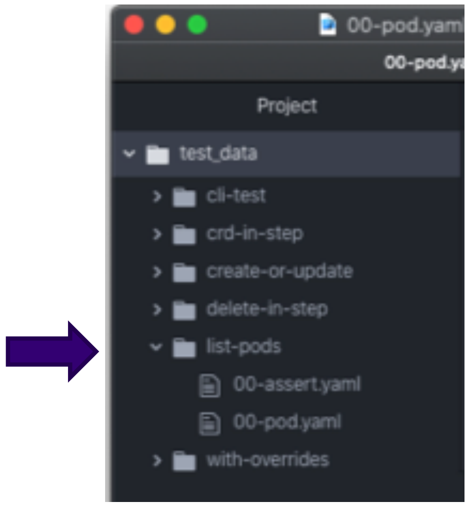


kuttl-test.yaml — ~/projects/go/src/github.com/kudobuilder/kuttl

```yaml
apiVersion: kudo.dev/v1beta1
kind: TestSuite
testDirs:
- ./test/integration
startControlPlane: true
parallel: 4
```

Test

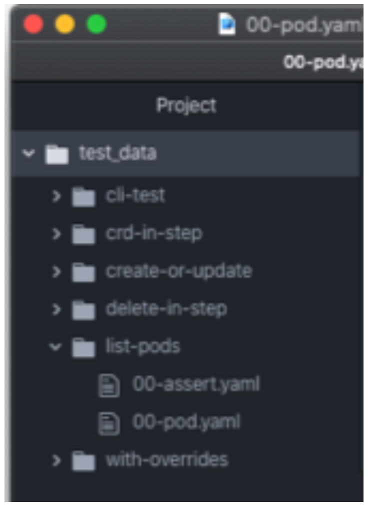
- A Collection of Test Steps
- Test Name == Folder Name
- “list-pods” is the name of this test



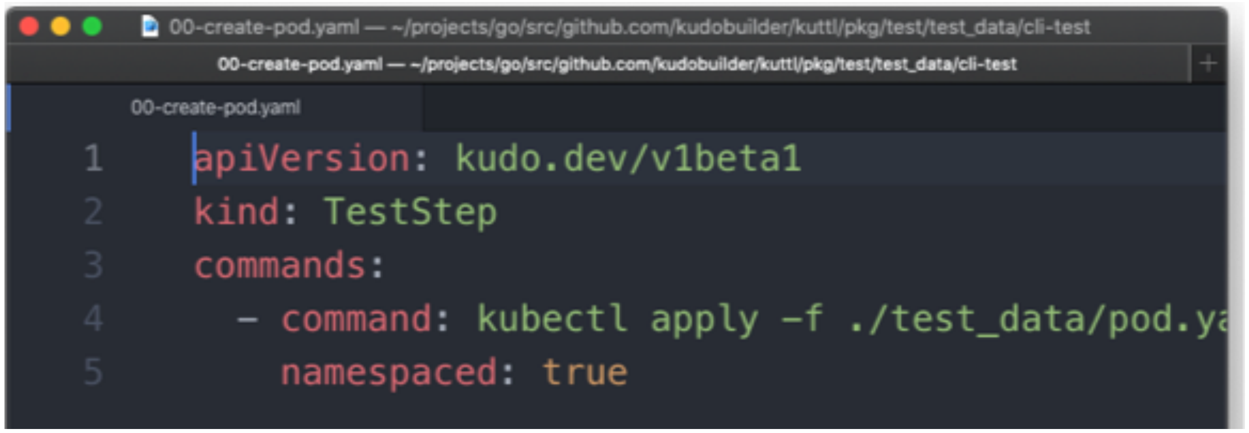
TestStep

2 Concepts define a **TestStep**

Indexed Files
Same Index, Same Step



Defined Kind

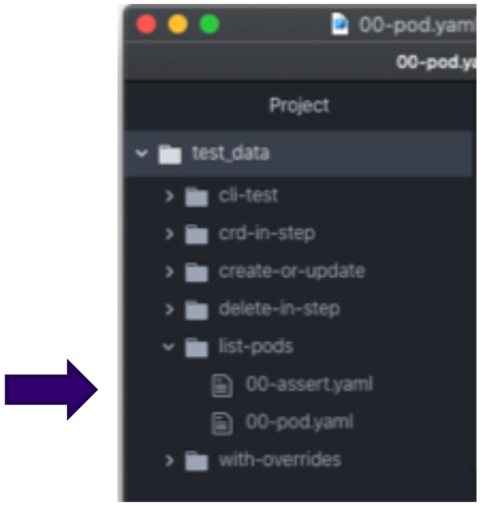

00-create-pod.yaml — ~/projects/go/src/github.com/kudobuilder/kuttl/pkg/test/test_data/cli-test
00-create-pod.yaml

```yaml
apiVersion: kudo.dev/v1beta1
kind: TestStep
commands:
  - command: kubectl apply -f ./test_data/pod.ya
    namespaced: true
```

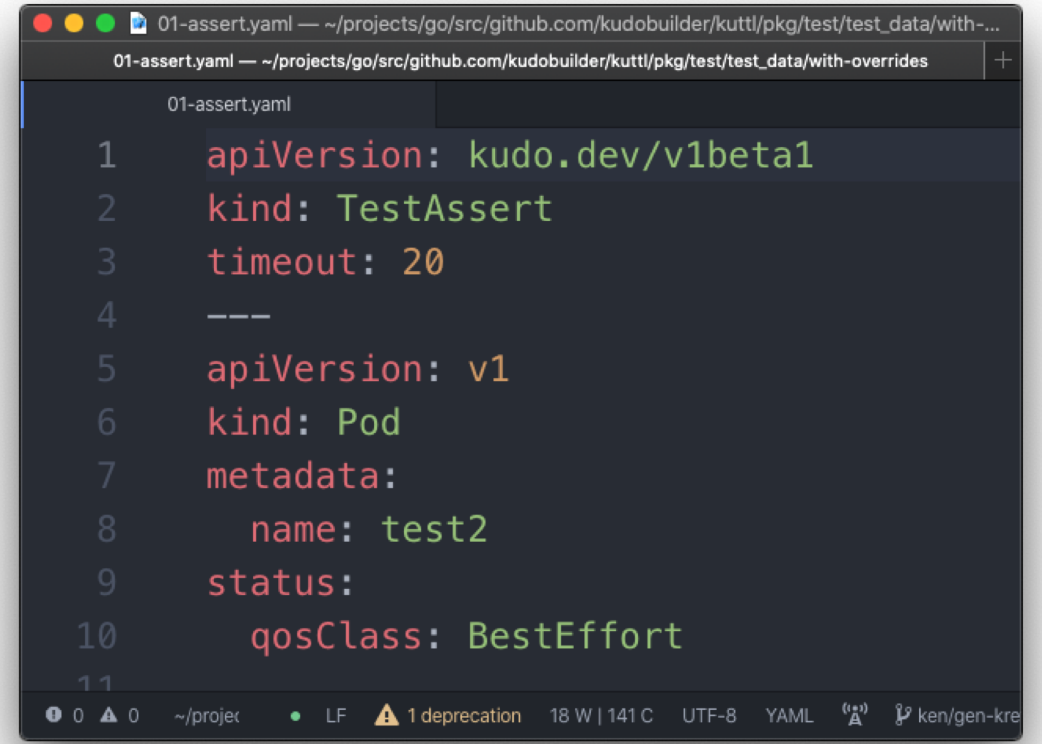
TestAssert

2 Concepts define a **TestAssert**

Step file named with "assert" or "errors"



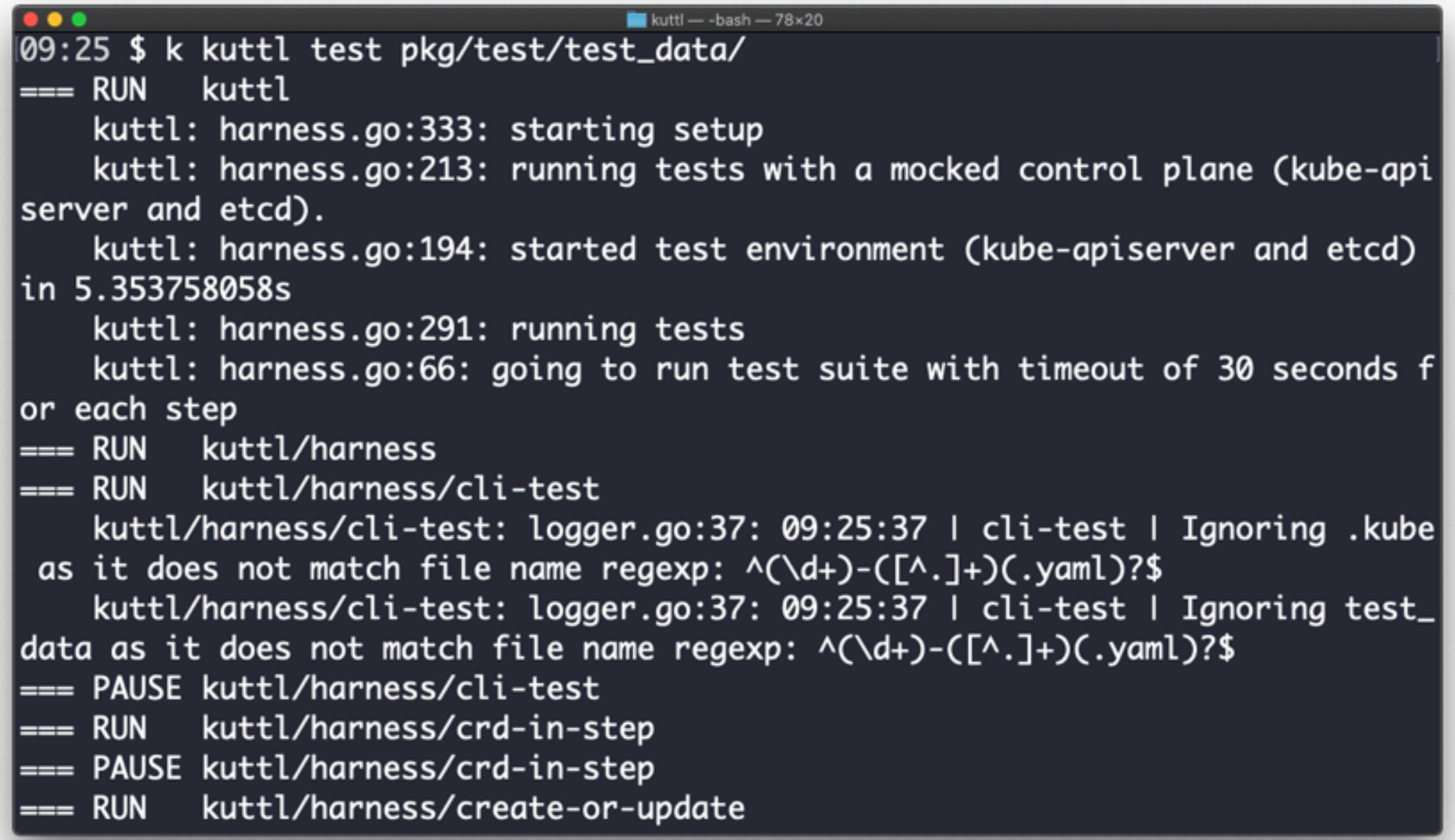
Defined Kind used within an assert step


01-assert.yaml — ~/projects/go/src/github.com/kudobuilder/kuttl/pkg/test/test_data/with-overrides
01-assert.yaml

```yaml
apiVersion: kudo.dev/v1beta1
kind: TestAssert
timeout: 20
---
apiVersion: v1
kind: Pod
metadata:
  name: test2
status:
  qosClass: BestEffort
```

0 0 ~/projec LF 1 deprecation 18 W | 141 C UTF-8 YAML ken/gen-kre

KUTTL a TestSuite



```
09:25 $ k kuttl test pkg/test/test_data/
=== RUN   kuttl
    kuttl: harness.go:333: starting setup
    kuttl: harness.go:213: running tests with a mocked control plane (kube-api
server and etcd).
    kuttl: harness.go:194: started test environment (kube-apiserver and etcd)
in 5.353758058s
    kuttl: harness.go:291: running tests
    kuttl: harness.go:66: going to run test suite with timeout of 30 seconds f
or each step
=== RUN   kuttl/harness
=== RUN   kuttl/harness/cli-test
    kuttl/harness/cli-test: logger.go:37: 09:25:37 | cli-test | Ignoring .kube
 as it does not match file name regexp: ^(\d+)-([^.]+)(.yaml)?$
    kuttl/harness/cli-test: logger.go:37: 09:25:37 | cli-test | Ignoring test_
data as it does not match file name regexp: ^(\d+)-([^.]+)(.yaml)?$
=== PAUSE kuttl/harness/cli-test
=== RUN   kuttl/harness/crd-in-step
=== PAUSE kuttl/harness/crd-in-step
=== RUN   kuttl/harness/create-or-update
```

Your first **KUTTL**

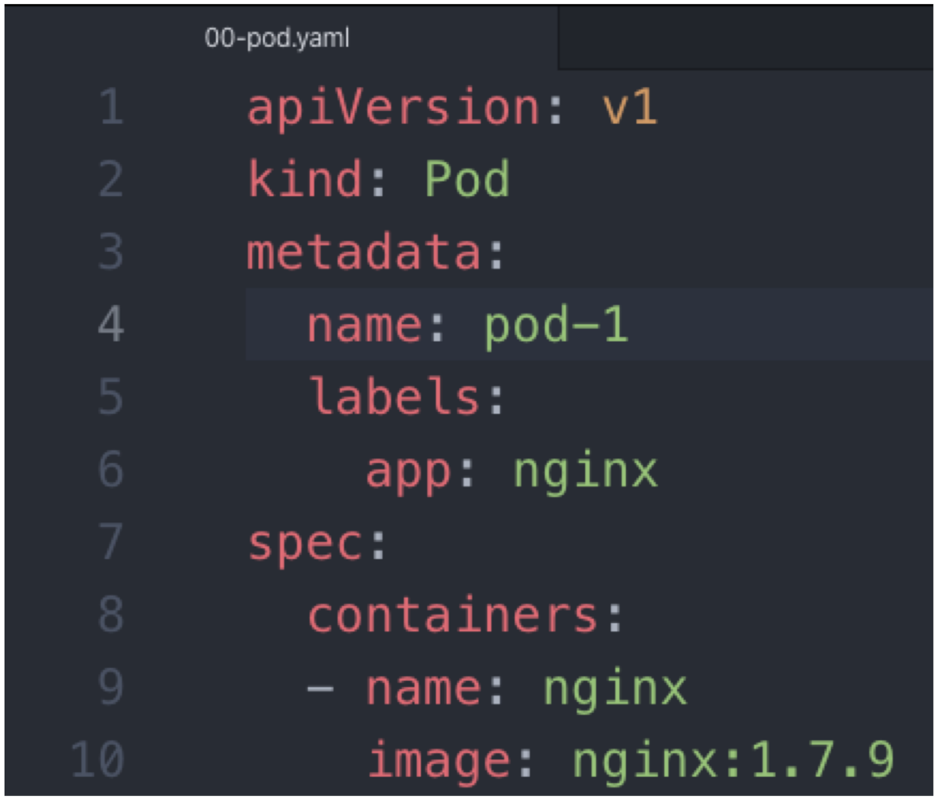
Test Case Setup

```
mkdir -p tests/e2e
```

```sh
mkdir tests/e2e/example-test
```

Test Step 00

Setup

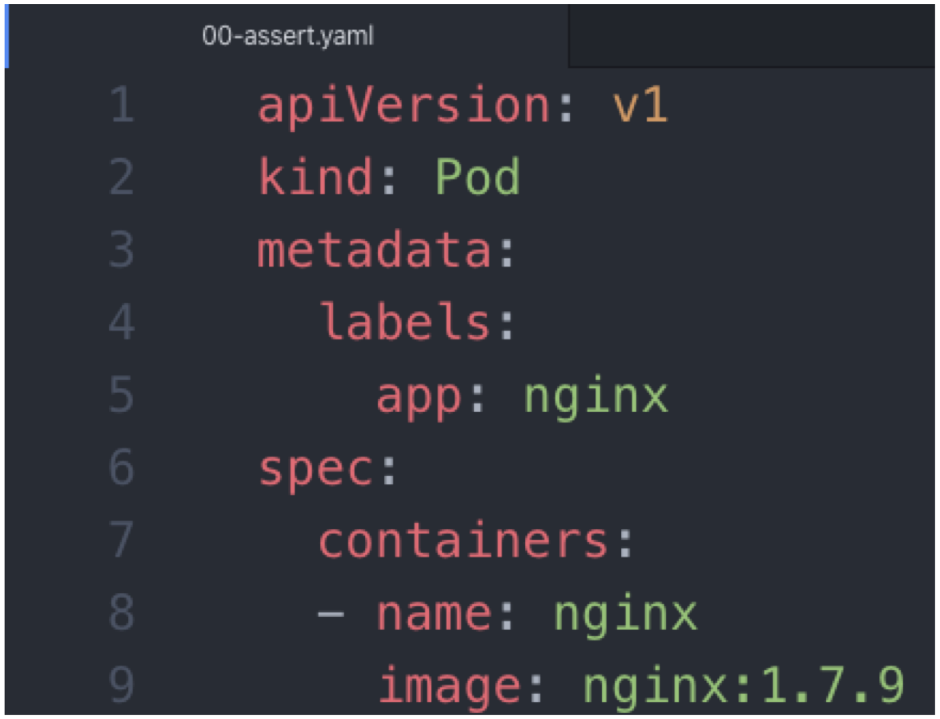


00-pod.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-1
  labels:
    app: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.7.9
```

Test Step 00

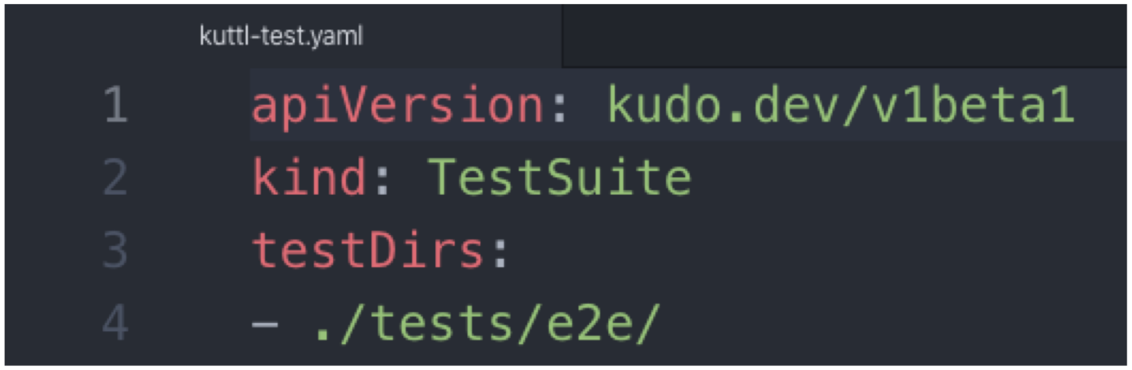
Assert


00-assert.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    app: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.7.9
```

Test Suite Configuration

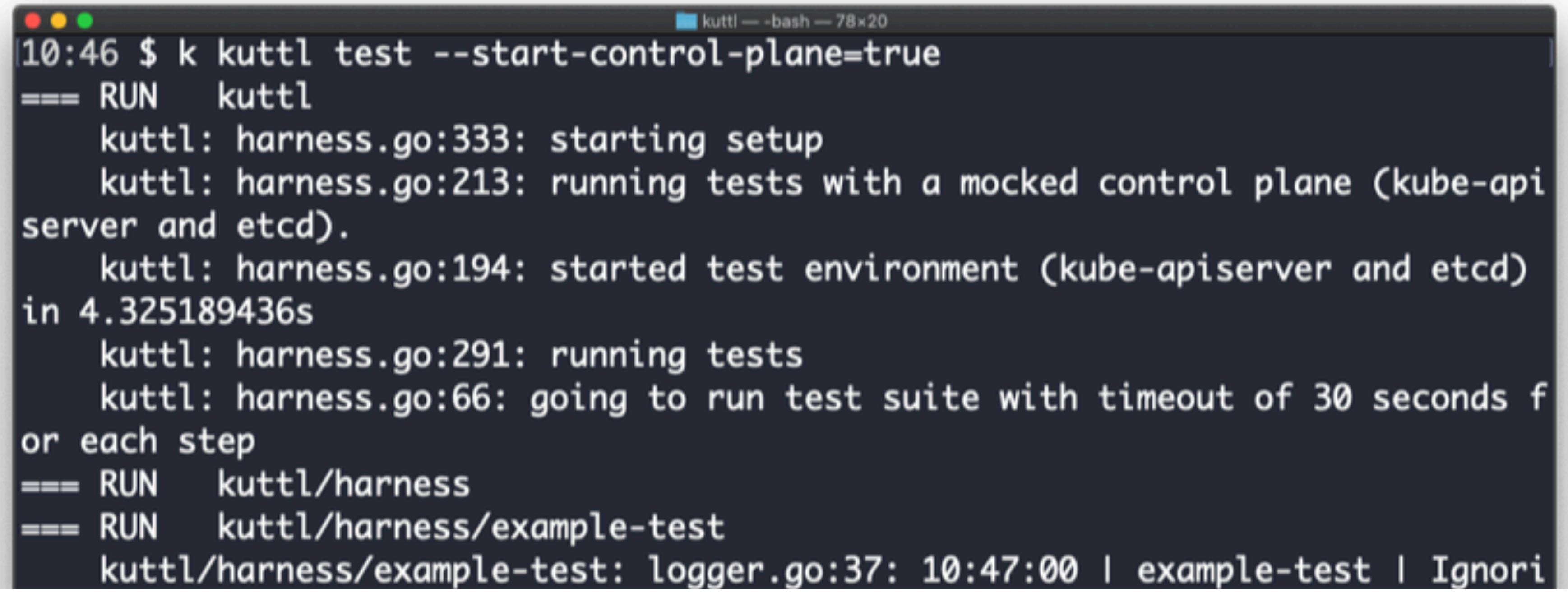
kuttl-test.yaml


kuttl-test.yaml

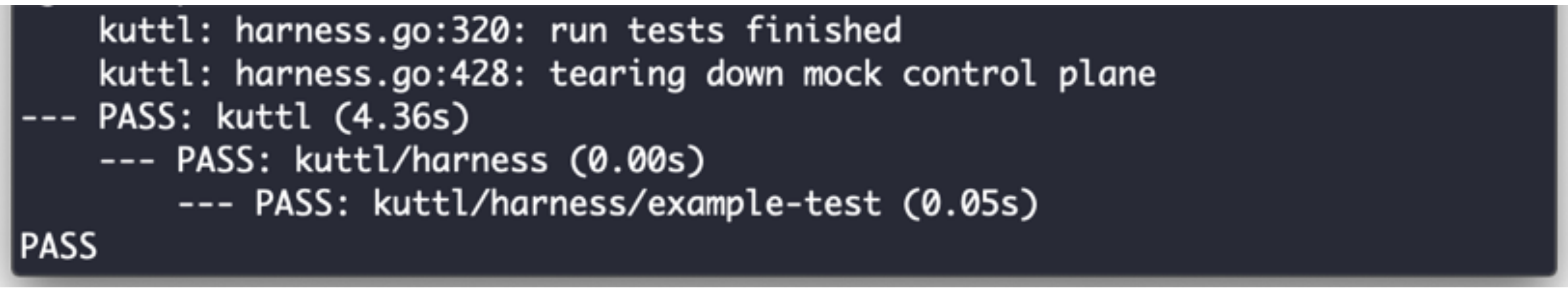
```
apiVersion: kudo.dev/v1beta1
kind: TestSuite
testDirs:
- ./tests/e2e/
```

Located in the working directory of kuttl

Run Test Suite



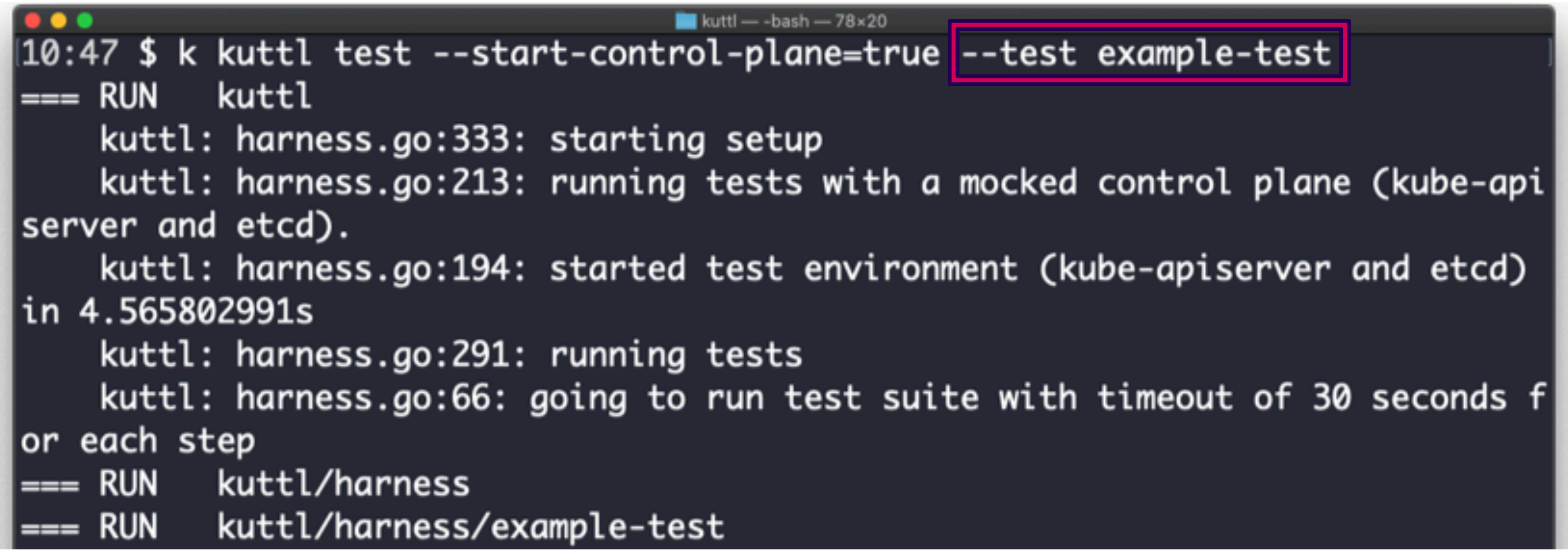
```
10:46 $ k kuttl test --start-control-plane=true
=== RUN   kuttl
    kuttl: harness.go:333: starting setup
    kuttl: harness.go:213: running tests with a mocked control plane (kube-api
server and etcd).
    kuttl: harness.go:194: started test environment (kube-apiserver and etcd)
in 4.325189436s
    kuttl: harness.go:291: running tests
    kuttl: harness.go:66: going to run test suite with timeout of 30 seconds f
or each step
=== RUN   kuttl/harness
=== RUN   kuttl/harness/example-test
    kuttl/harness/example-test: logger.go:37: 10:47:00 | example-test | Ignori
```



```
    kuttl: harness.go:320: run tests finished
    kuttl: harness.go:428: tearing down mock control plane
--- PASS: kuttl (4.36s)
    --- PASS: kuttl/harness (0.00s)
        --- PASS: kuttl/harness/example-test (0.05s)
PASS
```

Running 1 Test From the Suite

--test <test-name>

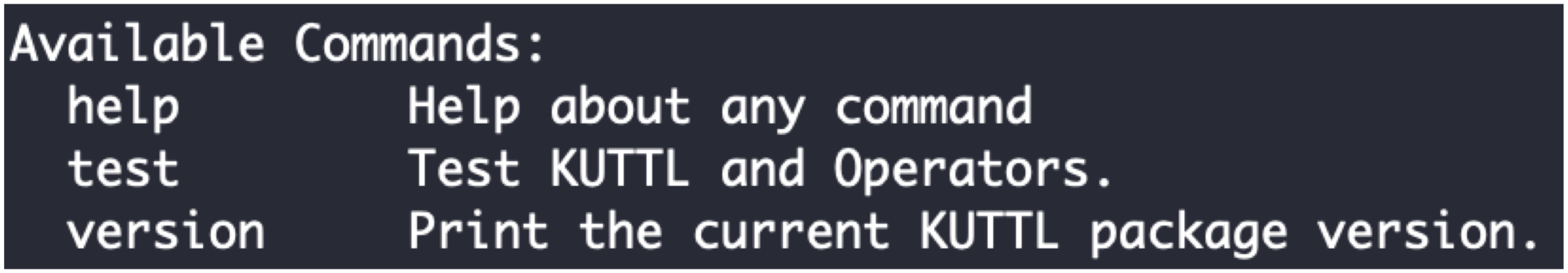


```
10:47 $ k kuttl test --start-control-plane=true --test example-test
=== RUN   kuttl
    kuttl: harness.go:333: starting setup
    kuttl: harness.go:213: running tests with a mocked control plane (kube-api
server and etcd).
    kuttl: harness.go:194: started test environment (kube-apiserver and etcd)
in 4.565802991s
    kuttl: harness.go:291: running tests
    kuttl: harness.go:66: going to run test suite with timeout of 30 seconds f
or each step
=== RUN   kuttl/harness
=== RUN   kuttl/harness/example-test
```

Ways to **KUTTL**

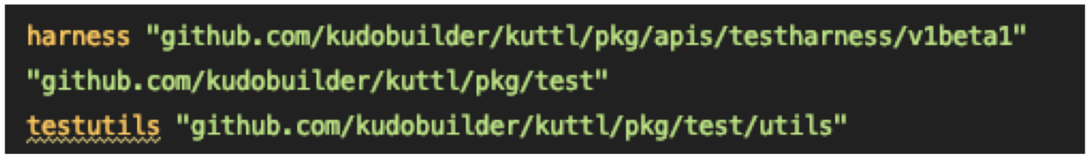
KUTTL CLI

```
k kuttl --help
```

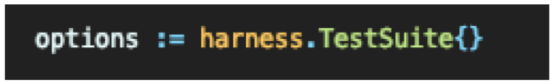


```
Available Commands:
  help        Help about any command
  test        Test KUTTL and Operators.
  version     Print the current KUTTL package version.
```

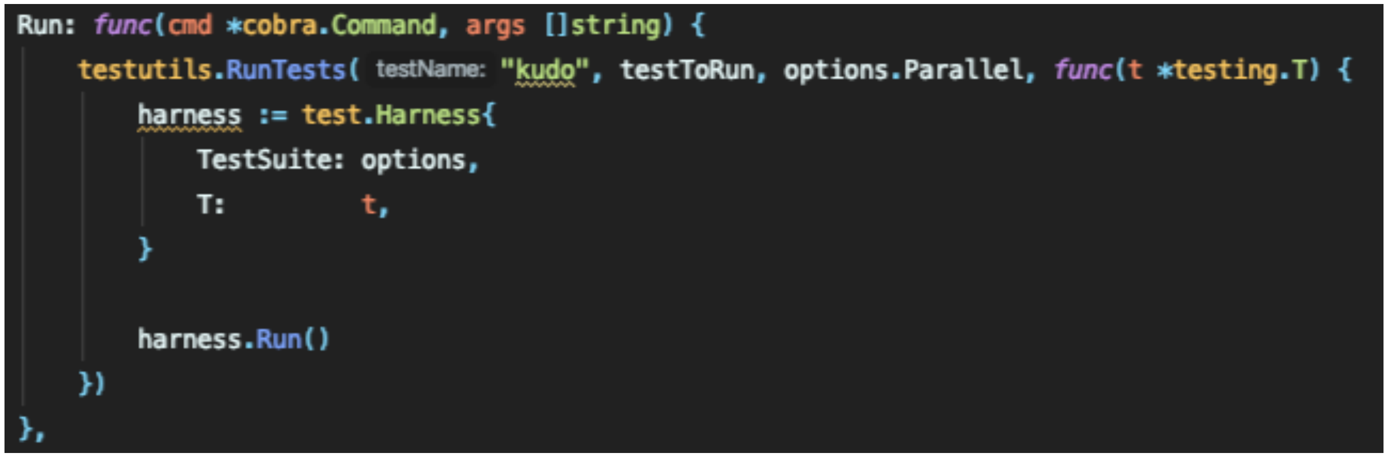
KUTTL Library Integration



```
harness "github.com/kudobuilder/kuttl/pkg/apis/testharness/v1beta1"
"github.com/kudobuilder/kuttl/pkg/test"
testutils "github.com/kudobuilder/kuttl/pkg/test/utils"
```



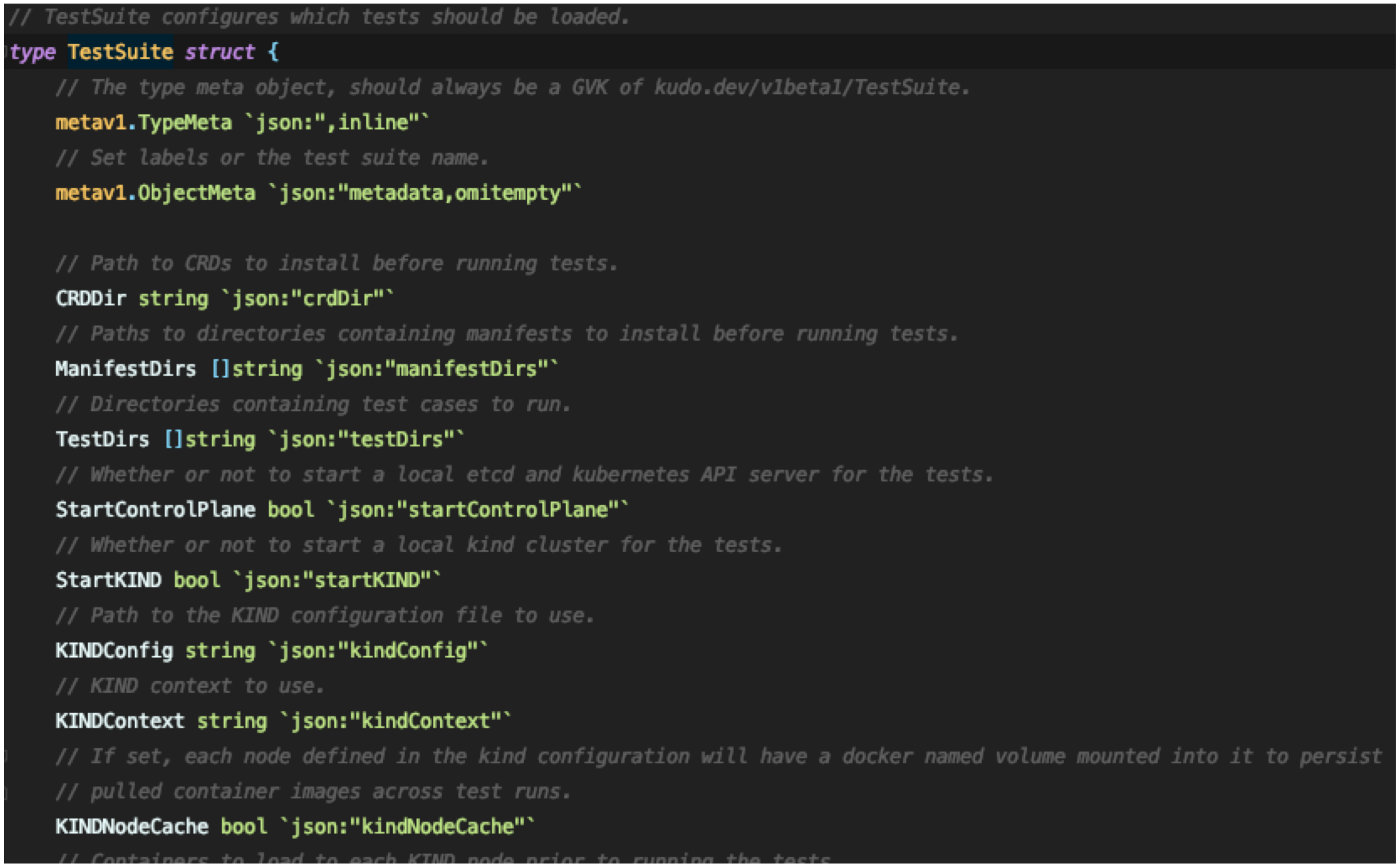
```
options := harness.TestSuite{}
```



```
Run: func(cmd *cobra.Command, args []string) {
    testutils.RunTests( testName: "kudo", testToRun, options.Parallel, func(t *testing.T) {
        harness := test.Harness{
            TestSuite: options,
            T:         t,
        }

        harness.Run()
    })
},
```

TestSuite Configuration

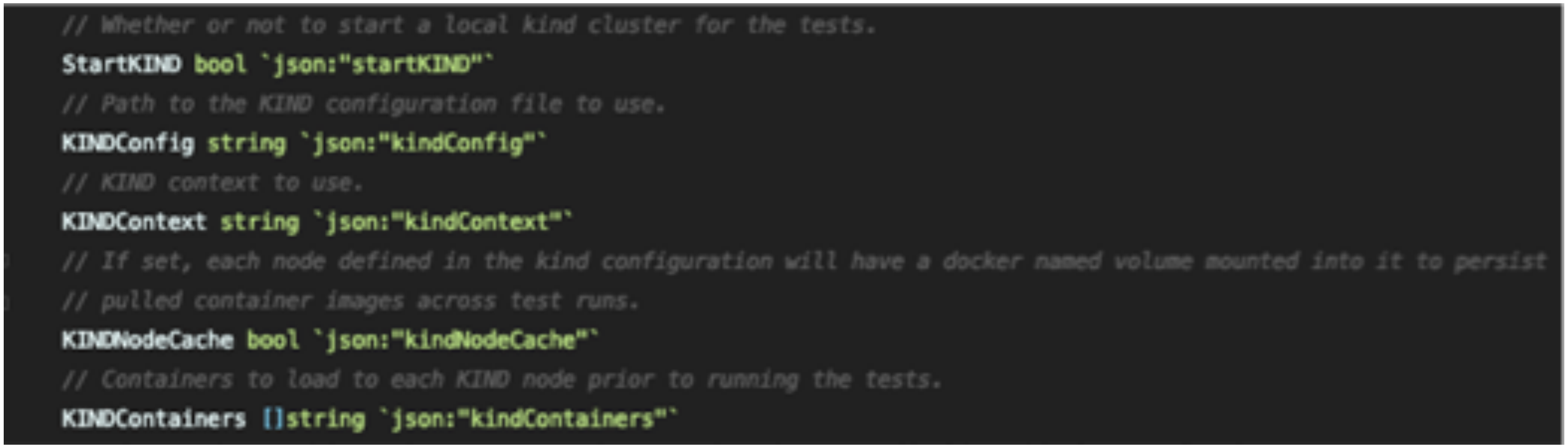


```go
// TestSuite configures which tests should be loaded.
type TestSuite struct {
    // The type meta object, should always be a GVK of kudo.dev/v1beta1/TestSuite.
    metav1.TypeMeta `json:",inline"`
    // Set labels or the test suite name.
    metav1.ObjectMeta `json:"metadata,omitempty"`

    // Path to CRDs to install before running tests.
    CRDDir string `json:"crdDir"`
    // Paths to directories containing manifests to install before running tests.
    ManifestDirs []string `json:"manifestDirs"`
    // Directories containing test cases to run.
    TestDirs []string `json:"testDirs"`
    // Whether or not to start a local etcd and kubernetes API server for the tests.
    StartControlPlane bool `json:"startControlPlane"`
    // Whether or not to start a local kind cluster for the tests.
    StartKIND bool `json:"startKIND"`
    // Path to the KIND configuration file to use.
    KINDConfig string `json:"kindConfig"`
    // KIND context to use.
    KINDContext string `json:"kindContext"`
    // If set, each node defined in the kind configuration will have a docker named volume mounted into it to persist
    // pulled container images across test runs.
    KINDNodeCache bool `json:"kindNodeCache"`
    // Containers to load to each KIND node prior to running the tests.
```

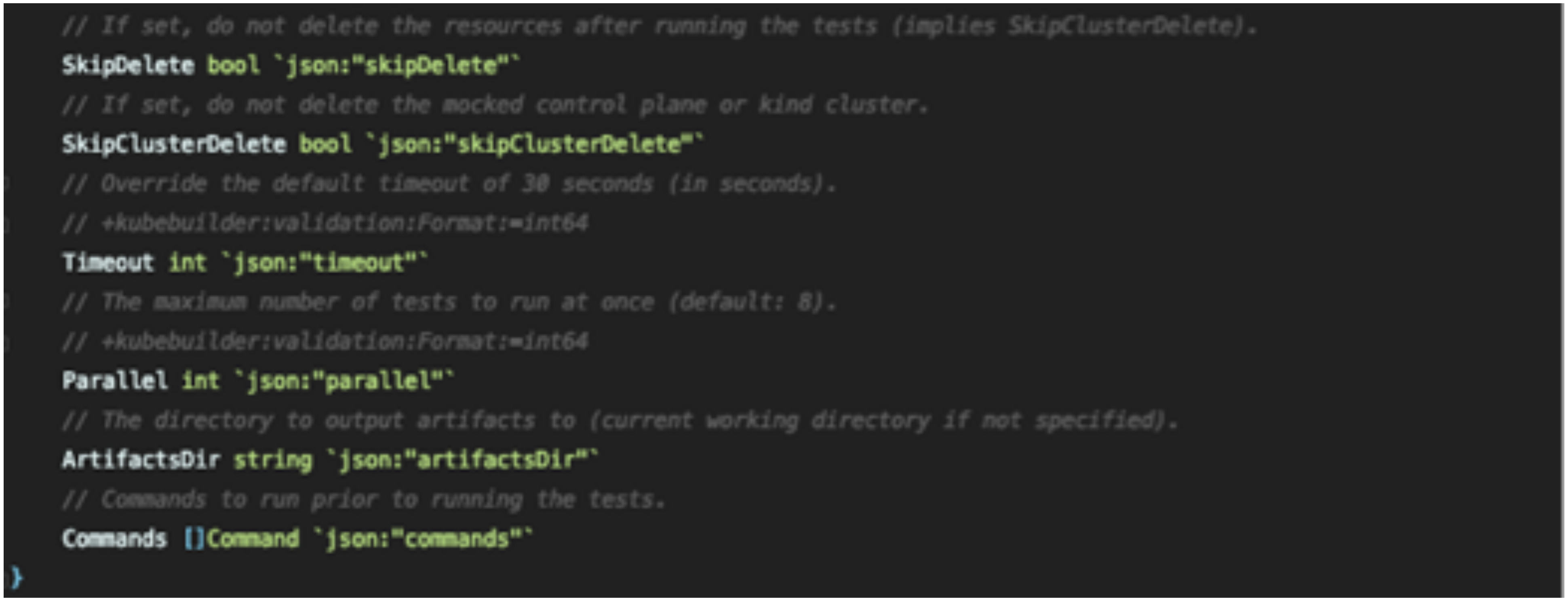
TestSuite Configuration

Kind Config



```
// Whether or not to start a local kind cluster for the tests.
StartKIND bool `json:"startKIND"`
// Path to the KIND configuration file to use.
KINDConfig string `json:"kindConfig"`
// KIND context to use.
KINDContext string `json:"kindContext"`
// If set, each node defined in the kind configuration will have a docker named volume mounted into it to persist
// pulled container images across test runs.
KINDNodeCache bool `json:"kindNodeCache"`
// Containers to load to each KIND node prior to running the tests.
KINDContainers []string `json:"kindContainers"`
```

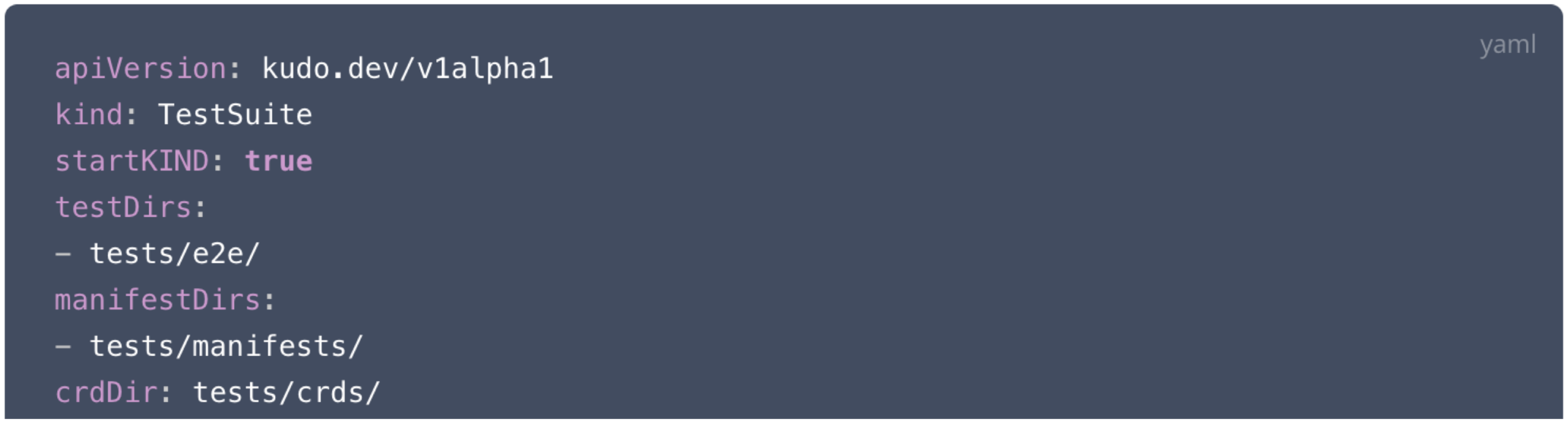
TestSuite Configuration



```
    // If set, do not delete the resources after running the tests (implies SkipClusterDelete).
    SkipDelete bool `json:"skipDelete"`
    // If set, do not delete the mocked control plane or kind cluster.
    SkipClusterDelete bool `json:"skipClusterDelete"`
    // Override the default timeout of 30 seconds (in seconds).
    // +kubebuilder:validation:Format:=int64
    Timeout int `json:"timeout"`
    // The maximum number of tests to run at once (default: 8).
    // +kubebuilder:validation:Format:=int64
    Parallel int `json:"parallel"`
    // The directory to output artifacts to (current working directory if not specified).
    ArtifactsDir string `json:"artifactsDir"`
    // Commands to run prior to running the tests.
    Commands []Command `json:"commands"`
}
```

TestSuite Configuration

kuttl-test.yaml

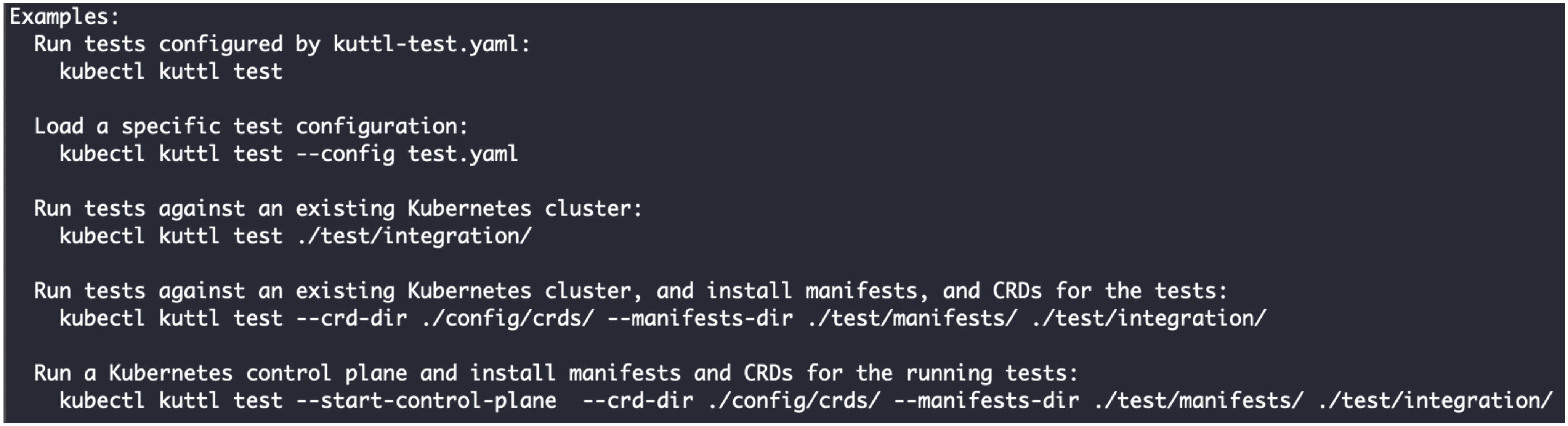


yaml

```yaml
apiVersion: kudo.dev/v1alpha1
kind: TestSuite
startKIND: true
testDirs:
- tests/e2e/
manifestDirs:
- tests/manifests/
crdDir: tests/crds/
```

KUTTL CLI

Configuration or Override



```
Examples:
  Run tests configured by kuttl-test.yaml:
    kubectl kuttl test

  Load a specific test configuration:
    kubectl kuttl test --config test.yaml

  Run tests against an existing Kubernetes cluster:
    kubectl kuttl test ./test/integration/

  Run tests against an existing Kubernetes cluster, and install manifests, and CRDs for the tests:
    kubectl kuttl test --crd-dir ./config/crds/ --manifests-dir ./test/manifests/ ./test/integration/

  Run a Kubernetes control plane and install manifests and CRDs for the running tests:
    kubectl kuttl test --start-control-plane  --crd-dir ./config/crds/ --manifests-dir ./test/manifests/ ./test/integration/
```

Where do you want to KUTTL?

Where to KUTTL

Test Environments

- Live Cluster
  - **$KUBECONFIG** or the **`--kubeconfig`** flag
- Kind
  - `startKIND: true` in kuttl-test.yaml or `--start-kind=true`
  - Lots of kind control
- Mocked Control Plane
  - `startControlPlane: true` in kuttl-test.yaml or `--start-control-plane`

Kind Cluster

Special Kind Configuration
- `kubectl kuttl test --kind-config=kind.yaml`

Setting Kind Context
- `kubectl kuttl test --kind-context=foo`

Preload Container Images
* In kuttl-test.yaml `kindContainers`:

Keep Cluster for analysis
- `kubectl kuttl test  --skip-cluster-delete`

Autonomy of a KUTTL

Test Steps

Files and Format

Test files: *.yaml or *.yml

Other files ignored
- useful for docs, license, etc.

<index>-<step-name>.yaml
- tests/e2e/example/00-pod.yaml
- tests/e2e/example/00-example.yaml
- tests/e2e/example/01-staging.yaml

Step is all indexed files, evaluated followed by asserts (more to come)

Multiple YAML docs is common in a file

Test Steps

Create or Update

Step files are:

- **Created** if they do not exist in cluster
- Patch **Updated** if they exist
  - Possible to express minimum updates
- Delete is possible through a TestStep Object

Test Steps

Delete

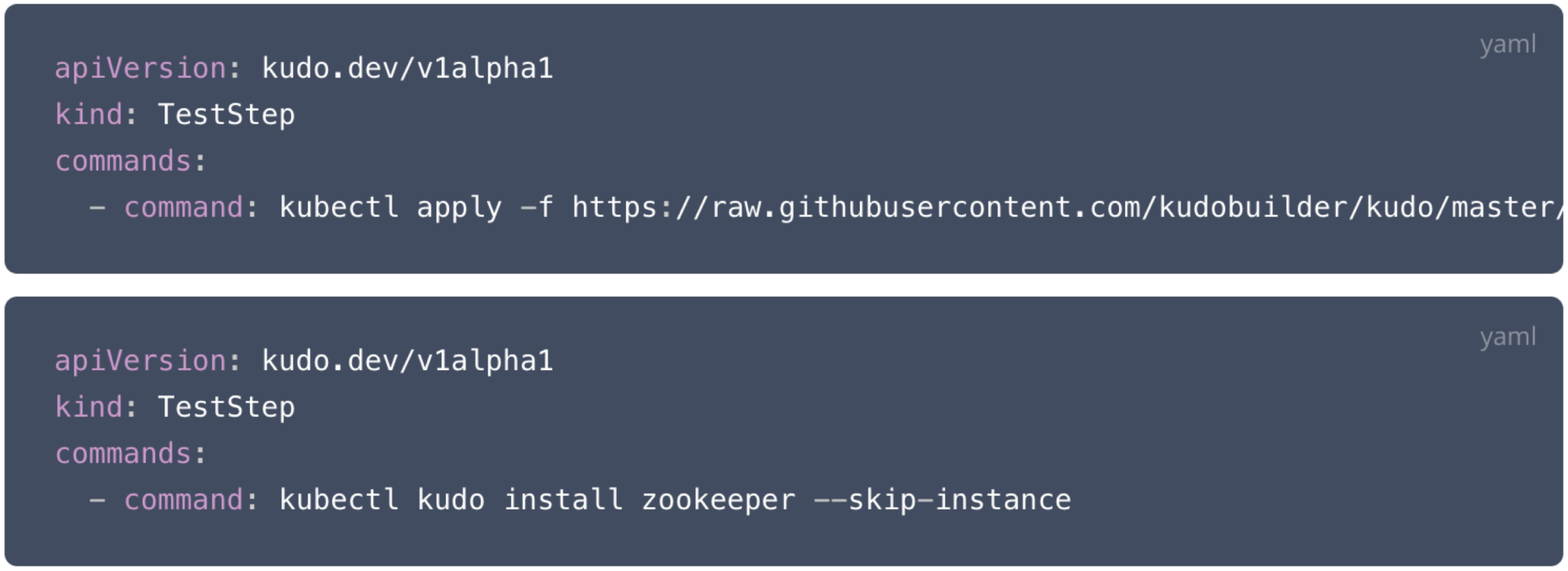
Delete is possible through a TestStep Object:

```yaml
apiVersion: kudo.dev/v1alpha1
kind: TestStep
delete:
# Delete a Pod
- apiVersion: v1
  kind: Pod
  name: my-pod
# Delete all Pods with app=nginx
- apiVersion: v1
  kind: Pod
  labels:
    app: nginx
# Delete all Pods in the test namespace
- apiVersion: v1
  kind: Pod
```

Test Steps

commands

Arbitrary commands are possible and are run at the beginning of the step and run until complete



```yaml
apiVersion: kudo.dev/v1alpha1
kind: TestStep
commands:
  - command: kubectl apply -f https://raw.githubusercontent.com/kudobuilder/kudo/master/
```

```yaml
apiVersion: kudo.dev/v1alpha1
kind: TestStep
commands:
  - command: kubectl kudo install zookeeper --skip-instance
```

Asserts and Errors

Format

<index>-assert.yaml

- Asserts the state was met within a time limit (default: 30 secs)

<index>-errors.yaml

- Asserts if a state exists that it is an error
- Asserts the absence of an object

Asserts and Errors

Example

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
status:
  phase: Successful
```

For Assert:
Passes if there is a pod

- named my-pod
- status.phase=Successful

No other fields are evaluated

KUTTLing Tips

Kuberenetes Events are Objects

```yaml
apiVersion: v1
kind: Event
reason: Started
source:
  component: kubelet
involvedObject:
  apiVersion: v1
  kind: Pod
  name: my-pod
```

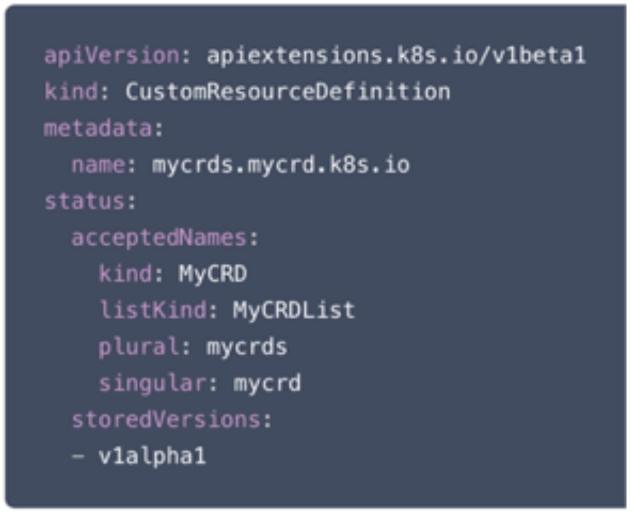
Asserts that an Event with reason “Started” happened for `my-pod`

KUTTLing Tips

CRDs or Waiting for K8S

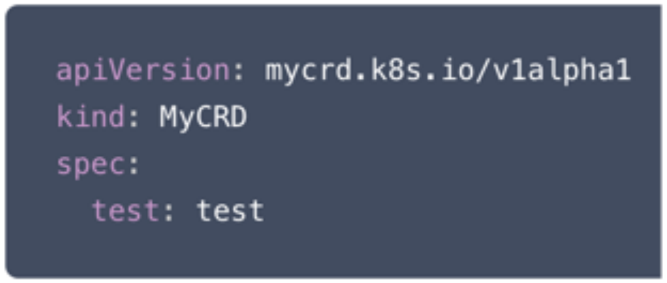
Certain objects (like CRDs) **take time** before they are available resources.
At the TestSuite level, defined CRDs are waited for prior to tests
IF you have a **CRD as part of a step**, it is necessary to assert for that CRD prior to using.
Assuming 00-crd.yaml

`00-assert.yaml`



```yaml
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  name: mycrds.mycrd.k8s.io
status:
  acceptedNames:
    kind: MyCRD
    listKind: MyCRDList
    plural: mycrds
    singular: mycrd
  storedVersions:
  - v1alpha1
```

`01-use.yaml`



```yaml
apiVersion: mycrd.k8s.io/v1alpha1
kind: MyCRD
spec:
  test: test
```

https://kudo.dev/docs/testing/tips.html#custom-resource-definitions

KUTTLing Tips

Helm

yaml

```yaml
apiVersion: kudo.dev/v1alpha1
kind: TestSuite
commands:
- command: kubectl create serviceaccount -n kube-system tiller
  ignoreFailure: true
- command: kubectl create clusterrolebinding tiller --clusterrole=cluster-admin --servic
  ignoreFailure: true
- command: helm init --wait --service-account tiller
- command: helm delete --purge memcached
  ignoreFailure: true
- command: helm install --replace --namespace memcached --name nginx stable/memcached
testDirs:
- ./test/integration
startKIND: true
kindNodeCache: true
```

Also possible in a TestStep

KUTTLing an Operator

Operators

CRD

Installing CRDs

crdDir in kuttl-test.yaml

Or

```
k kuttl test --crd-dir
```

Loads and Waits for CRD

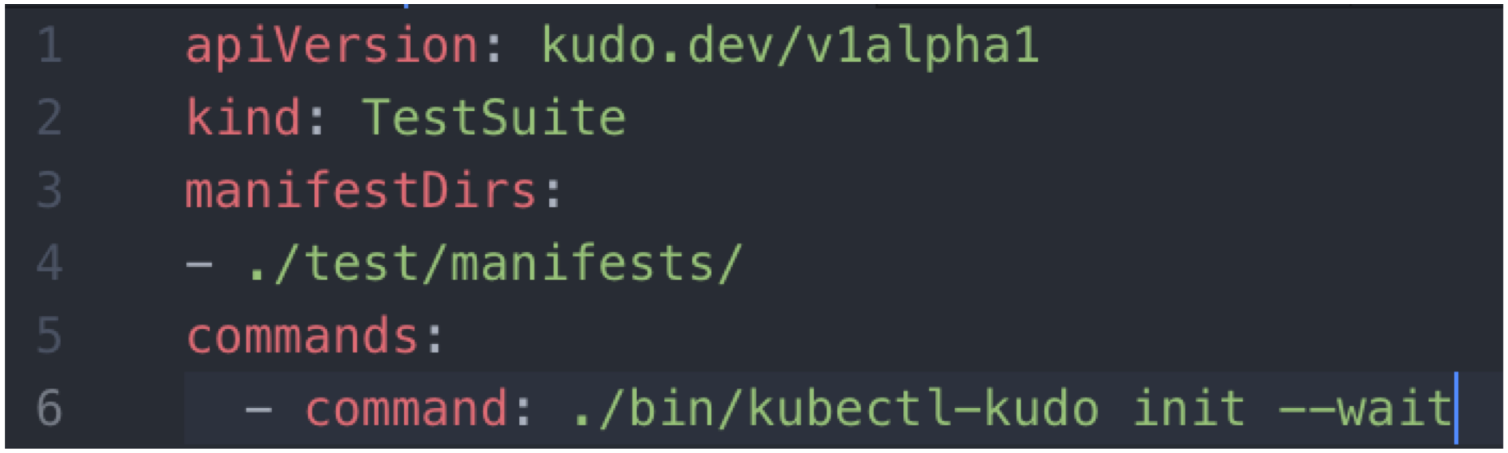
Operators

Controller

Examples for KUDO

KUDO controller (named manager) can be installed from the kudo cli

```
* k kudo init --wait
```



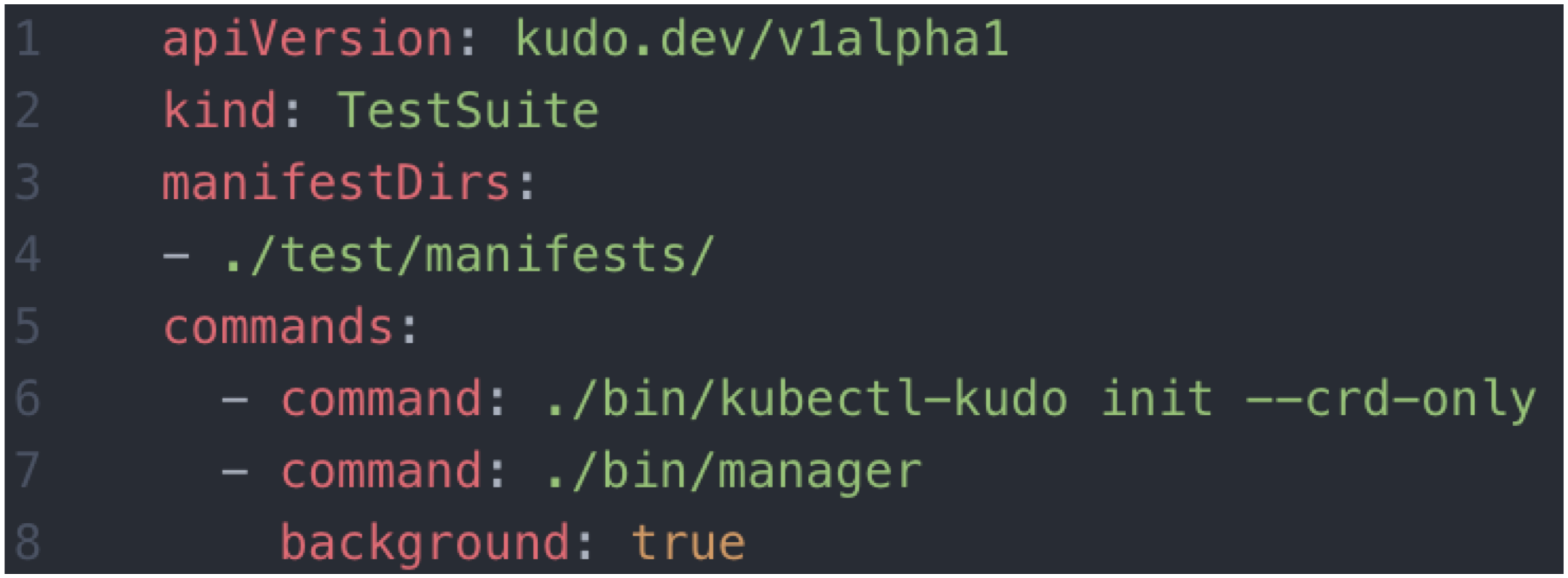
```yaml
apiVersion: kudo.dev/v1alpha1
kind: TestSuite
manifestDirs:
- ./test/manifests/
commands:
  - command: ./bin/kubectl-kudo init --wait
```

Operators

Controller in Dev

Examples for KUDO

After a `make manager` makefile task, run the `bin/manager` and set the `background` to true.



```yaml
apiVersion: kudo.dev/v1alpha1
kind: TestSuite
manifestDirs:
- ./test/manifests/
commands:
  - command: ./bin/kubectl-kudo init --crd-only
  - command: ./bin/manager
    background: true
```

KUTTL in Action

Future KUTTLing

KUTTL Released

- KUTTL v0.1.0
  - Released March 26, 2020
  - However it was based on 1 year of KUDO development

Call to Action

Get Involved

- KUTTL Project
- https://github.com/kudobuilder/kuttl

- k8s.io slack #kudo
- https://app.slack.com/client/T09NY5SBT/CG3HTFCMV

- Current docs:
  - http://kuttl.dev

- KEP Process
  - https://github.com/kudobuilder/kuttl/blob/master/keps/0001-kep-process.md

Thank you for KUTTLing with us!

KUTTL https://github.com/kudobuilder/kuttl

@devgerred
@kensipe