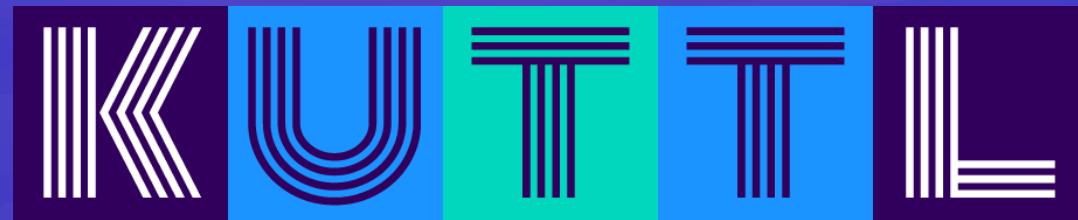


Declarative Cluster Testing with

Ken Sipe





Schnook

Actor

Henhouse Henery (1949)

Who Framed Roger Rabbit (1988)
Space Jam (1996)

KFC, Oscar Mayer and GEICO

 @FoghornRoost



Foghorn Leghorn

(he / him / rooster / schnook)

Rooster

Henhouse Henery (1949)

Who Framed Roger Rabbit (1988)

Space Jam (1996)

KFC, Oscar Mayer and GEICO

 @FoghornRoost



Ken Sipe

(he / him / boy / dog / rooster)

Distributed Application Engineer
And Orchestration Conductor

Apache Mesos, Kubernetes, KUDO, KUTTL

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

 @KenSipe
ken@d2iq.com

What is KUTTL



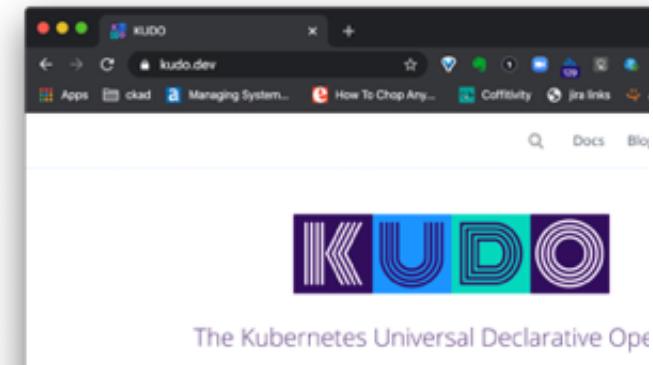
KUbernetes Test TooL (kuttl)

KUTTL Origins

D2
IQ

Kubernetes Universal Declarative Operator
(KUDO)

Declarative Testing



What is KUTTL



Unit

Integration

e2e



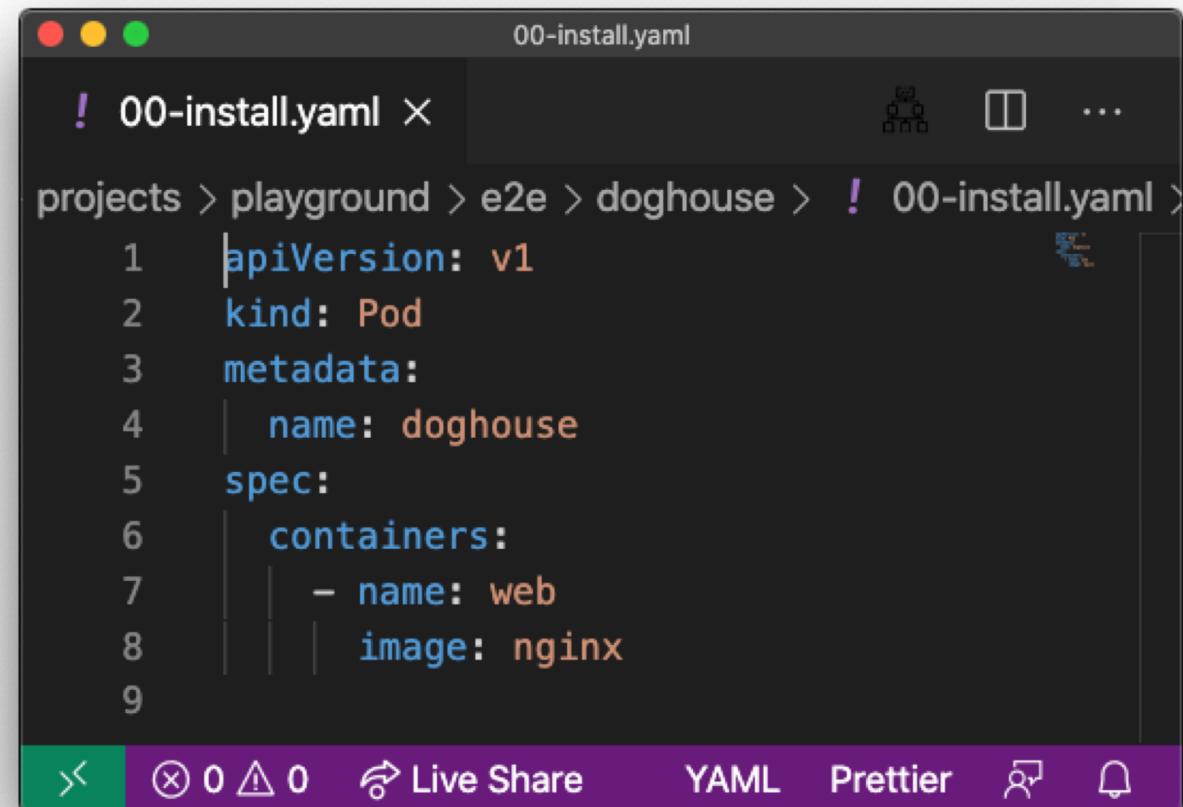
- Tests the full stack (front to back to front round trip)
- Not necessarily developers
- Variety of Clusters
- Various Versions of Clusters
- Test in The Client ENV

- Provides a Working Example of Expectations (from user perspective)

Declarative Testing

What does that mean?

Test Setup



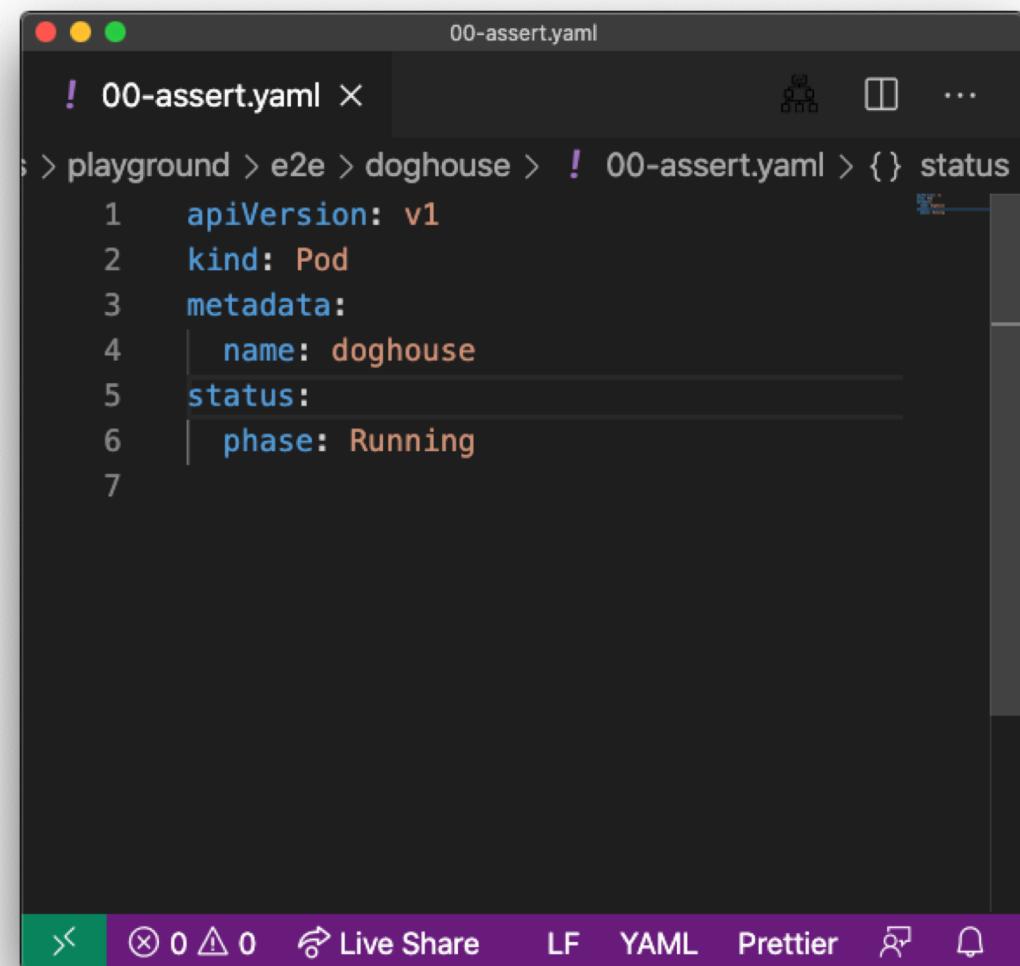
```
00-install.yaml
projects > playground > e2e > doghouse > ! 00-install.yaml >
1   apiVersion: v1
2   kind: Pod
3   metadata:
4     name: doghouse
5   spec:
6     containers:
7       - name: web
8         image: nginx
9
```

The screenshot shows a code editor window with a dark theme. The title bar says "00-install.yaml". The code itself is a YAML configuration for a Kubernetes Pod. It starts with "apiVersion: v1" and "kind: Pod". The "metadata" section contains a "name" field set to "doghouse". The "spec" section includes a "containers" list with one item, "web", which has an "image" field set to "nginx". The editor interface includes standard file operations (New, Open, Save) and a status bar at the bottom showing file statistics (0 errors, 0 warnings), a "Live Share" icon, and tabs for "YAML" and "Prettier".

Declarative Testing

What does that mean?

Assert!



```
00-assert.yaml
! 00-assert.yaml ×

: > playground > e2e > doghouse > ! 00-assert.yaml > {} status

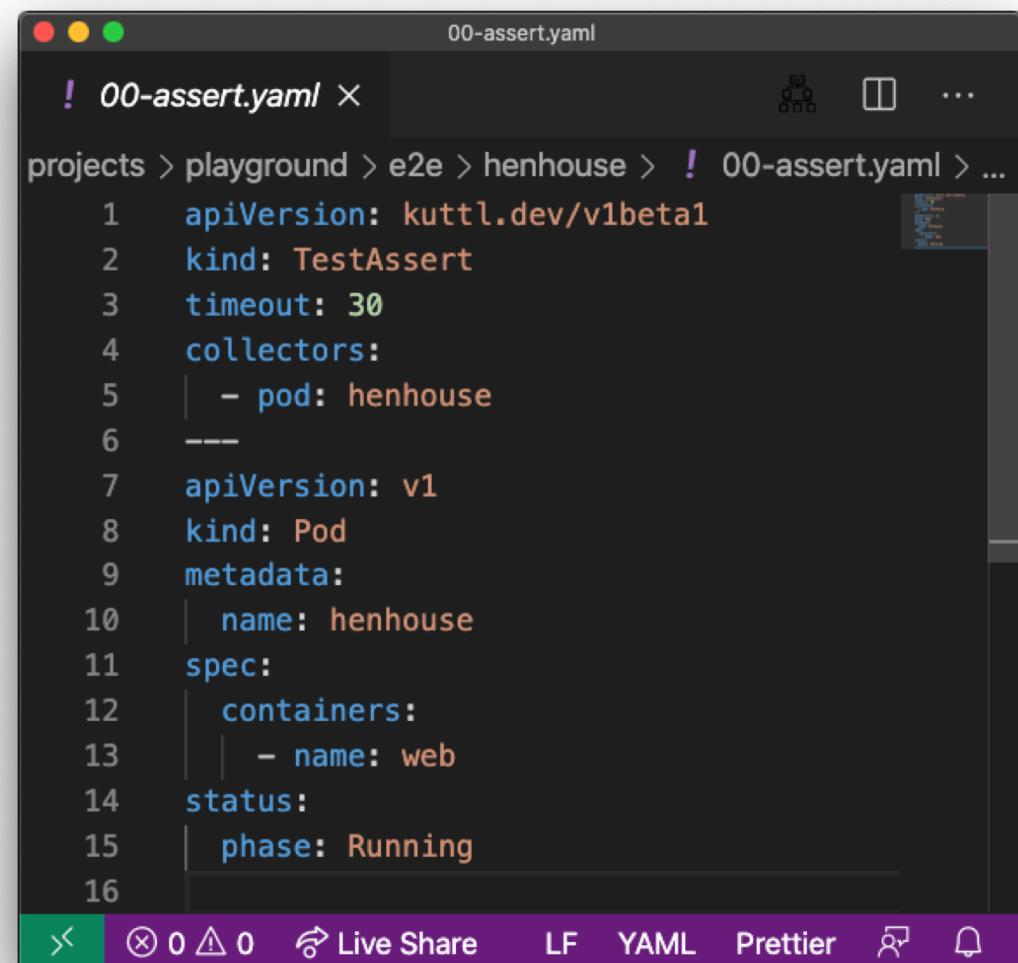
1   apiVersion: v1
2   kind: Pod
3   metadata:
4     name: doghouse
5   status:
6     phase: Running
7
```

The screenshot shows a dark-themed code editor window titled "00-assert.yaml". The file content is a single Pod definition with the name "doghouse" and phase "Running". At the bottom of the editor, there are several tabs: "Live Share", "LF", "YAML", and "Prettier".

Declarative Testing

What does that mean?

Assert!



```
! 00-assert.yaml × 00-assert.yaml

projects > playground > e2e > henhouse > ! 00-assert.yaml > ...
1   apiVersion: kuttl.dev/v1beta1
2   kind: TestAssert
3   timeout: 30
4   collectors:
5     - pod: henhouse
6   ---
7   apiVersion: v1
8   kind: Pod
9   metadata:
10    name: henhouse
11   spec:
12     containers:
13       - name: web
14   status:
15     phase: Running
16
```

The screenshot shows a code editor window titled "00-assert.yaml". The file content is a YAML document defining a "TestAssert" object. It specifies an API version of "kuttl.dev/v1beta1", a kind of "TestAssert", a timeout of 30 seconds, and a collector for a pod named "henhouse". The document then defines a "Pod" object with the same name, "henhouse", and a single "container" named "web". The "status" of the pod is set to "Running". The code editor interface includes a navigation bar with icons for file operations, a status bar at the bottom, and a dark-themed background.



WELL, I DO DECLARE

Who Needs a KUTTL?



Testing harness to **declarative** test:

- operators
- KUDO
- helm charts

- any other Kubernetes applications or controllers.

Testing Operators

A screenshot of a web browser window. The title bar says "Writing Kuttl Scorecard Tests". The address bar shows the URL "sdk.operatorframework.io/docs/advanced-topics/scorecard/kuttl-tests/". The page header includes the Operator SDK logo and navigation links for Home, Build, Documentation, and v1.1. The main content area contains the "Writing Kuttl Scorecard Tests" section.

Documentation
Overview
Installation
Building Operators
Upgrade SDK Version
Advanced Topics
Scorecard
Scorecard
Writing Custom
Scorecard Tests
**Writing Kuttl Scorecard
Tests**

Writing Kuttl Scorecard Tests

This guide outlines the steps which can be followed to implement scorecard tests using the [kuttl](#) project and specifically the scorecard kuttl test image.

Defining kuttl Tests in Scorecard

A screenshot of a web browser window. The title bar says "Getting Started | KUDO". The address bar shows the URL "kudo.dev/docs/". The page header includes the KUDO logo. The main content area contains the "What is KUDO" and "Getting Started" sections.

What is KUDO

Developing Operators ▶

Getting Started

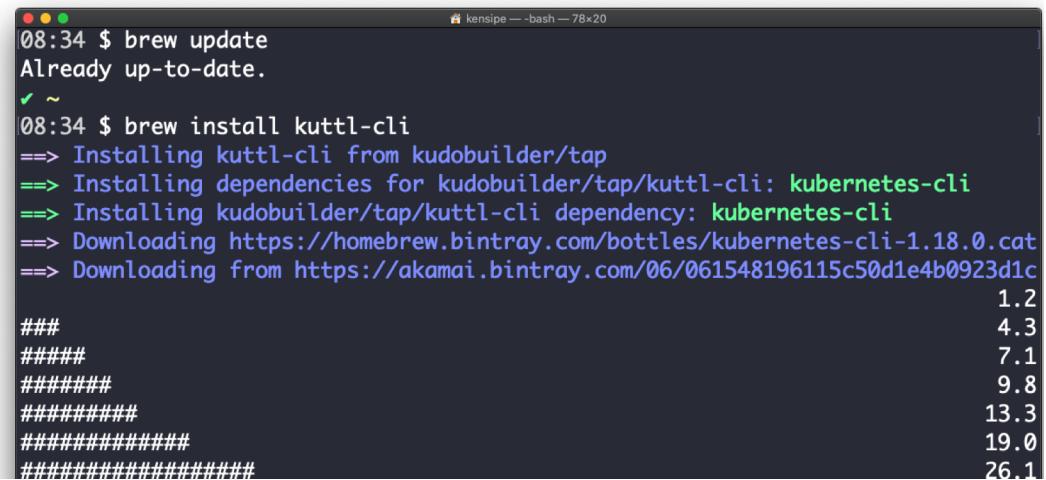
Pre-requisites

How Do I Start KUTTLing?

D2
IQ

```
brew install kuttl-cli
```

```
kubectl krew install 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
##### 13.3
##### 19.0
##### 26.1
```

KUTTL Basics

```
k kuttl --help
```

Available Commands:

assert	Asserts the declared state to be true.
errors	Asserts the declared errors state to NOT be true.
help	Help about any command
test	Test KUTTL and Operators.
version	Print the current KUTTL package version.

KUTTL CLI

D2
IQ

```
kubectl kuttl test ./test/
```

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/
```

```
Run tests against an existing Kubernetes cluster with a JUnit XML file output:  
kubectl kuttl test ./test/integration/ --report xml
```

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-assert.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

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

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

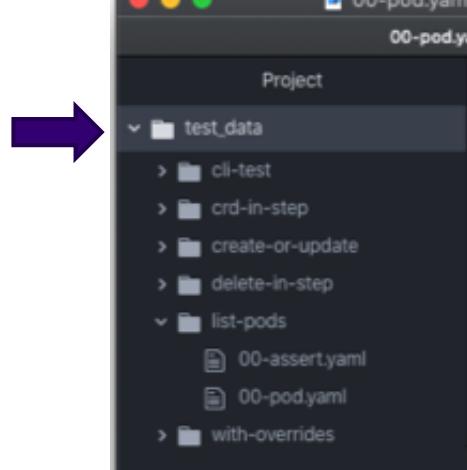
TestAssert

Assert conditions

TestSuite

2 Concepts define a **TestSuite**

Folder of Tests



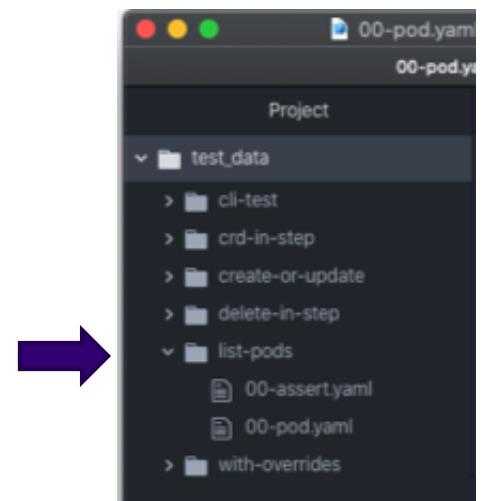
Configuration File

```
kuttl-test.yaml -- ~/projects/go/src/github.com/kudobuilder/kuttl
kuttl-test.yaml
1  apiVersion: kudo.dev/v1beta1
2  kind: TestSuite
3  testDirs:
4    - ./test/integration
5  startControlPlane: true
6  parallel: 4
7
```

Test

D2
IQ

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



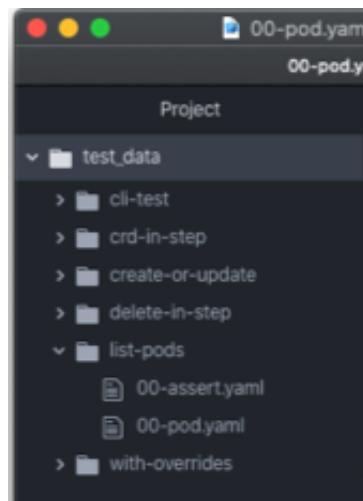
TestStep

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

00-create-pod.yaml
00-create-pod.yaml

1  apiVersion: kudo.dev/v1beta1
2  kind: TestStep
3  commands:
4    - command: kubectl apply -f ./test_data/pod.yaml
5      namespaced: true
```

TestAssert

2 Concepts define a TestAssert

Step file named with
“assert” or “errors”



A screenshot of a terminal window showing a file tree. The current directory is 'test_data'. Inside 'test_data' are several sub-directories: 'cli-test', 'crd-in-step', 'create-or-update', 'delete-in-step', and 'list-pods'. Under 'list-pods', there are two files: '00-assert.yaml' and '00-pod.yaml'. A purple arrow points from the left towards this terminal window.

Defined Kind used within an
assert step

A screenshot of a terminal window showing a YAML configuration file named '01-assert.yaml'. The file defines a 'TestAssert' kind object with the following properties:

```
1 apiVersion: kudo.dev/v1beta1
2 kind: TestAssert
3 timeout: 20
4 ---
5 apiVersion: v1
6 kind: Pod
7 metadata:
8   name: test2
9 status:
10    qosClass: BestEffort
```

The terminal also shows other tabs open, including another '01-assert.yaml' file and a '00-pod.yaml' file. The bottom of the terminal shows standard command-line metrics like file size and encoding.

KUTTL a TestSuite

D2
IQ

```
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
```

KUTTL Namespace



By Default... KUTTL creates a namespace for each test

Providing test isolation



Your first KUTTL

Test Case Setup



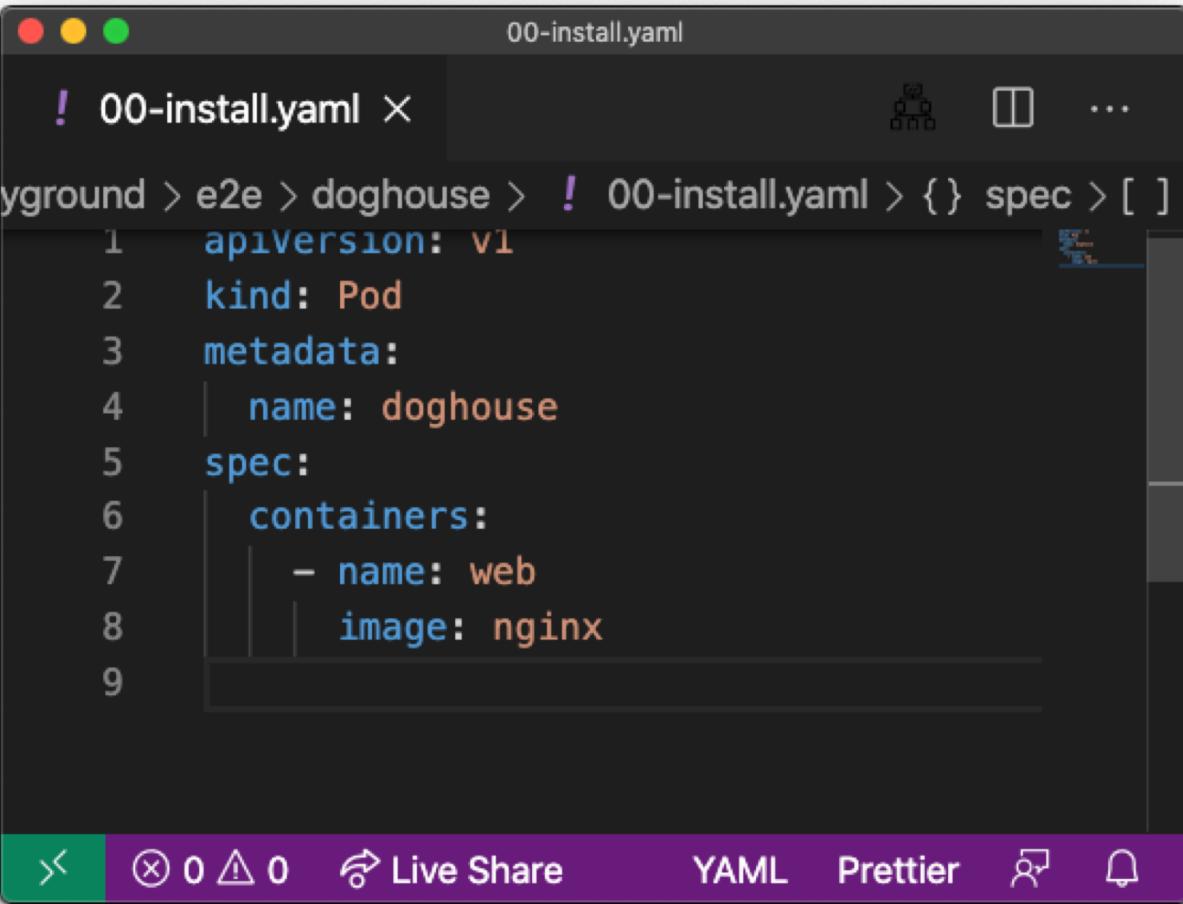
```
mkdir -p tests/e2e
```

```
mkdir tests/e2e/doghouse
```

sh

Test Step 00

Setup



```
00-install.yaml
! 00-install.yaml × ⚙️ ⌂ ...  
yground > e2e > doghouse > ! 00-install.yaml > {} spec > [ ]  
1   apiVersion: v1  
2   kind: Pod  
3   metadata:  
4     name: doghouse  
5   spec:  
6     containers:  
7       - name: web  
8         image: nginx  
9
```

The screenshot shows a code editor window with a dark theme. The title bar says "00-install.yaml". The editor displays a YAML configuration for a Kubernetes Pod. The Pod is named "doghouse" and contains a single container named "web" which uses the "nginx" image. The code is numbered from 1 to 9. At the bottom of the editor, there are several status icons: a green square with an "X", a purple square with a circle and a triangle, a purple square with a double arrow, a purple square with a "Live Share" icon, a purple square with "YAML", a purple square with "Prettier", a purple square with a magnifying glass, and a purple square with a bell.

Test Step 00

Assert

00-assert.yaml

! 00-assert.yaml X

Projects > playground > e2e > doghouse > ! 00-assert.yaml > ...

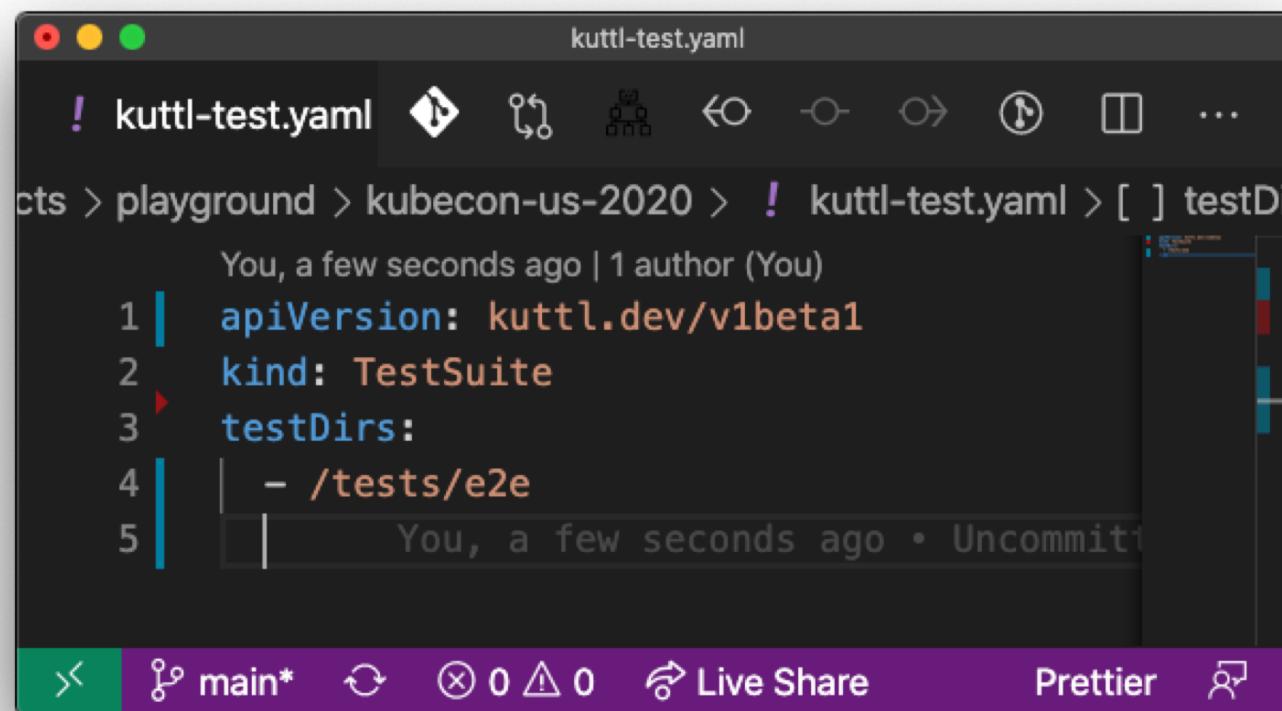
```
1 apiVersion: v1
2 kind: Pod
3 metadata:
4   name: doghouse
5 status:
6   phase: Running
7
```

< × ⑧ 0 △ 0 ⚡ Live Share YAML Prettier ⚡ 🔔

Test Suite Configuration

kuttl-test.yaml

Located in the working directory of kuttl



```
! kuttl-test.yaml
  apiVersion: kuttl.dev/v1beta1
  kind: TestSuite
  testDirs:
    - /tests/e2e
```

Run Test Suite

```
kensipe@kens-mbp-2:~/projects/playground/kubecon-us-2020
▶ k kuttl test
==== RUN kuttl
    harness.go:441: starting setup
    harness.go:247: running tests using configured kubeconfig.
    harness.go:341: running tests
    harness.go:73: going to run test suite with timeout of 30 seconds for each step
==== RUN kuttl/harness
==== RUN kuttl/harness/doghouse
==== PAUSE kuttl/harness/doghouse
==== RUN kuttl/harness/henhouse
    logger.go:42: 13:33:37 | henhouse | Ignoring pod.yaml as it does not match file name regexp: ^(\d+)-([^.]+)(.yaml)?$
==== PAUSE kuttl/harness/henhouse
==== CONT kuttl/harness/doghouse
==== CONT kuttl/harness/henhouse
==== CONT kuttl/harness/doghouse
    logger.go:42: 13:33:37 | doghouse | Creating namespace: kudo-test-verified-shepherd
==== CONT kuttl/harness/henhouse
    logger.go:42: 13:33:37 | henhouse | Creating namespace: kudo-test-fond-walrus
==== CONT kuttl/harness/doghouse
    logger.go:42: 13:33:37 | doghouse/0-install | starting test step 0-install
==== CONT kuttl/harness/henhouse
    logger.go:42: 13:33:37 | henhouse/0-install | starting test step 0-install
--- PASS: kuttl (3.85s)
    --- PASS: kuttl/harness (0.00s)
        --- PASS: kuttl/harness/doghouse (2.77s)
        --- PASS: kuttl/harness/henhouse (2.77s)
PASS
```

Running 1 Test From the Suite

--test <test-name>

```
● ● ● kensipe@kens-mbp-2:~/projects/playground/kubecon-us-2020 ⌂⌘2
❯ k kuttle test --test doghouse
== RUN kuttle
harness.go:441: starting setup
harness.go:247: running tests using configured kubeconfig.
harness.go:341: running tests
harness.go:73: going to run test suite with timeout of 30 seconds for each step
== RUN kuttle/harness
== RUN kuttle/harness/doghouse
== PAUSE kuttle/harness/doghouse
== CONT kuttle/harness/doghouse
logger.go:42: 13:36:12 | doghouse | Creating namespace: kudo-test-wired-bulldog
logger.go:42: 13:36:12 | doghouse/0-install | starting test step 0-install
logger.go:42: 13:36:12 | doghouse/0-install | Pod:kudo-test-wired-bulldog/doghouse created
logger.go:42: 13:36:14 | doghouse/0-install | test step completed 0-install
logger.go:42: 13:36:14 | doghouse | doghouse events from ns kudo-test-wired-bulldog:
logger.go:42: 13:36:14 | doghouse | 2020-10-20 13:36:12 -0500 CDT Normal Scheduled Successfully assigned kudo-test-wired-bulldog/doghouse to kind-control-plane
logger.go:42: 13:36:14 | doghouse | 2020-10-20 13:36:13 -0500 CDT Normal Pulling Pulling image "nginx"
logger.go:42: 13:36:14 | doghouse | 2020-10-20 13:36:13 -0500 CDT Normal Pulled Successfully pulled image "nginx" in 538.741131ms
logger.go:42: 13:36:14 | doghouse | 2020-10-20 13:36:13 -0500 CDT Normal Created Created container web
logger.go:42: 13:36:14 | doghouse | 2020-10-20 13:36:13 -0500 CDT Normal Started Started container web
logger.go:42: 13:36:14 | doghouse | Deleting namespace: kudo-test-wired-bulldog
== CONT kuttle
harness.go:382: run tests finished
harness.go:486: cleaning up
harness.go:541: removing temp folder: "
```

KUTTL Features

Test Steps

Delete

Delete is possible through a TestStep Object:

```
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
```

yaml

Test Steps

commands

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

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

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

Control the Namespace

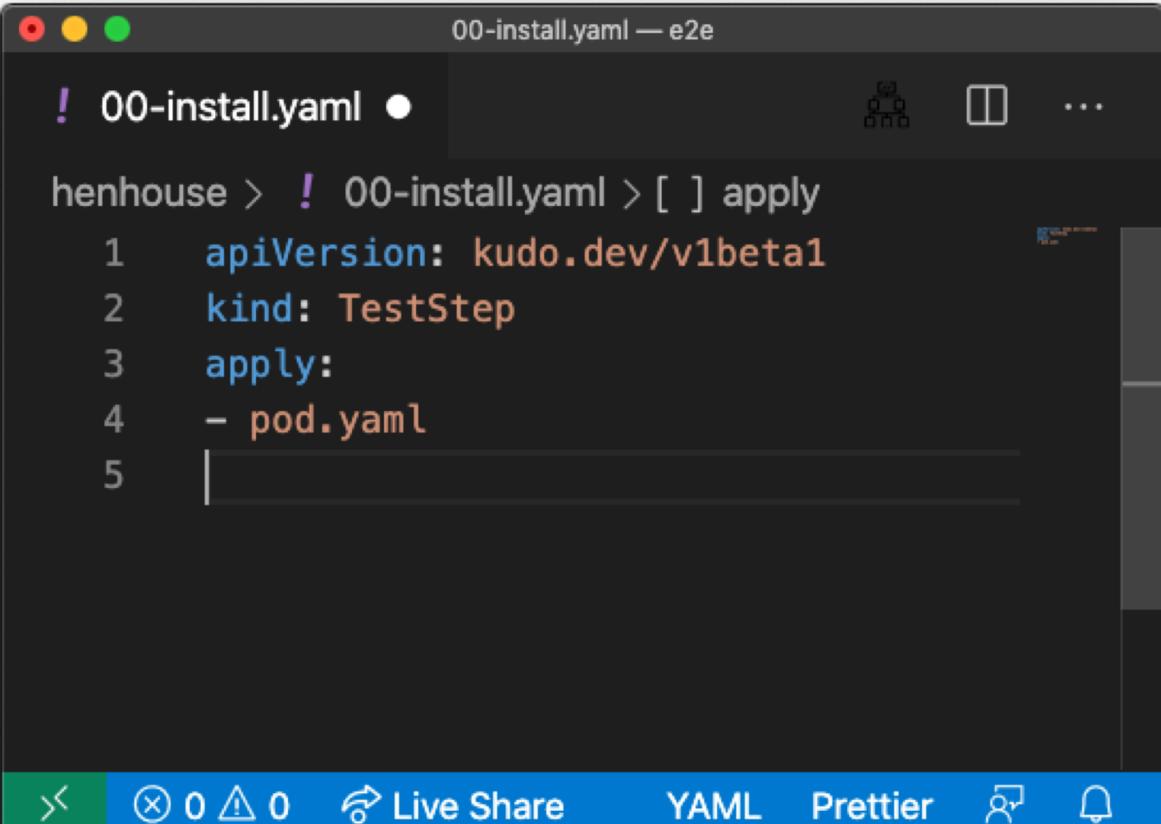
```
k k kuttle test -n default
==== RUN kuttl
    harness.go:441: starting setup
    harness.go:247: running tests using configured kubeconfig.
    harness.go:341: running tests
    harness.go:73: going to run test suite with timeout of 30 seconds for each step
==== RUN kuttl/harness
==== RUN kuttl/harness/doghouse
==== PAUSE kuttl/harness/doghouse
==== RUN kuttl/harness/henhouse
    logger.go:42: 13:48:55 | henhouse | Ignoring pod.yaml as it does not match file name regexp: ^(\d+
==== PAUSE kuttl/harness/henhouse
==== CONT kuttl/harness/doghouse
==== CONT kuttl/harness/henhouse
==== CONT kuttl/harness/doghouse
    logger.go:42: 13:48:55 | doghouse | Skipping creation of user-supplied namespace: default
==== CONT kuttl/harness/henhouse
    logger.go:42: 13:48:55 | henhouse | Skipping creation of user-supplied namespace: default
==== CONT kuttl/harness/doghouse
    logger.go:42: 13:48:55 | doghouse/0-install | starting test step 0-install
```

Reusing Apply and Assert

Example

TestStep can have array

- Apply
- Asserts



```
! 00-install.yaml •
henhouse > ! 00-install.yaml > [ ] apply
1   apiVersion: kudo.dev/v1beta1
2   kind: TestStep
3   apply:
4     - pod.yaml
5   |
```

The screenshot shows a code editor window titled "00-install.yaml — e2e". The file content is a YAML document with the following structure:

```
! 00-install.yaml •
henhouse > ! 00-install.yaml > [ ] apply
1   apiVersion: kudo.dev/v1beta1
2   kind: TestStep
3   apply:
4     - pod.yaml
5   |
```

The code editor interface includes standard window controls (red, yellow, green circles), a toolbar with icons for file operations, and a status bar at the bottom with icons for file navigation, live share, YAML, Prettier, and other tools.

KUTTLing Tips



Kubernetes Events are Objects

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

yaml

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

```
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

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

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

KUTTLing Tips



Helm

```
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 --serviceaccount=kube-system:tiller
  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/nginx-stable
testDirs:
- ./test/integration
startKIND: true
kindNodeCache: true
```

yaml

Also possible in a TestStep

KUTTL a URL

Pulls TestSuite and Runs!

```
kensipe@kens-mbp-2:~
$ k kuttl test https://github.com/kensipe/kubecon-us-2020/raw/main/e2e.tgz
== RUN kuttl
harness.go:441: starting setup
harness.go:247: running tests using configured kubeconfig.
harness.go:341: running tests
harness.go:173: temp folder created /var/folders/c1/j5vflbys41v_pzh5881sktqr0000gn/T/kuttl353942940
harness.go:396: downloading https://github.com/kensipe/kubecon-us-2020/raw/main/e2e.tgz
Downloading (e2e.tgz) 1.3 kB complete
    harness.go:73: going to run test suite with timeout of 30 seconds for each step
== RUN kuttl/harness
== RUN kuttl/harness/doghouse
    logger.go:42: 13:09:52 | doghouse | Ignoring ._00-assert.yaml as it does not match file name regexp: ^(\d+)-{[^.]+}(.yaml)$
    logger.go:42: 13:09:52 | doghouse | Ignoring ._00-install.yaml as it does not match file name regexp: ^(\d+)-{[^.]+}(.yaml)$
== PAUSE kuttl/harness/doghouse
== RUN kuttl/harness/henhouse
    logger.go:42: 13:09:52 | henhouse | Ignoring ._00-assert.yaml as it does not match file name regexp: ^(\d+)-{[^.]+}(.yaml)$
    logger.go:42: 13:09:52 | henhouse | Ignoring ._00-install.yaml as it does not match file name regexp: ^(\d+)-{[^.]+}(.yaml)$
    logger.go:42: 13:09:52 | henhouse | Ignoring ._pod.yaml as it does not match file name regexp: ^(\d+)-{[^.]+}(.yaml)$
    logger.go:42: 13:09:52 | henhouse | Ignoring pod.yaml as it does not match file name regexp: ^(\d+)-{[^.]+}(.yaml)$
== PAUSE kuttl/harness/henhouse
== CONT kuttl/harness/doghouse
```

KUTTL URL in TestSuite

D2
IQ

A screenshot of a code editor displaying a YAML configuration file named `kuttl-test.yaml`. The file is located in a directory structure: `Users > kensipe > projects > playground > kubecon-us-2020 > kuttl-test.yaml`. The content of the file is as follows:

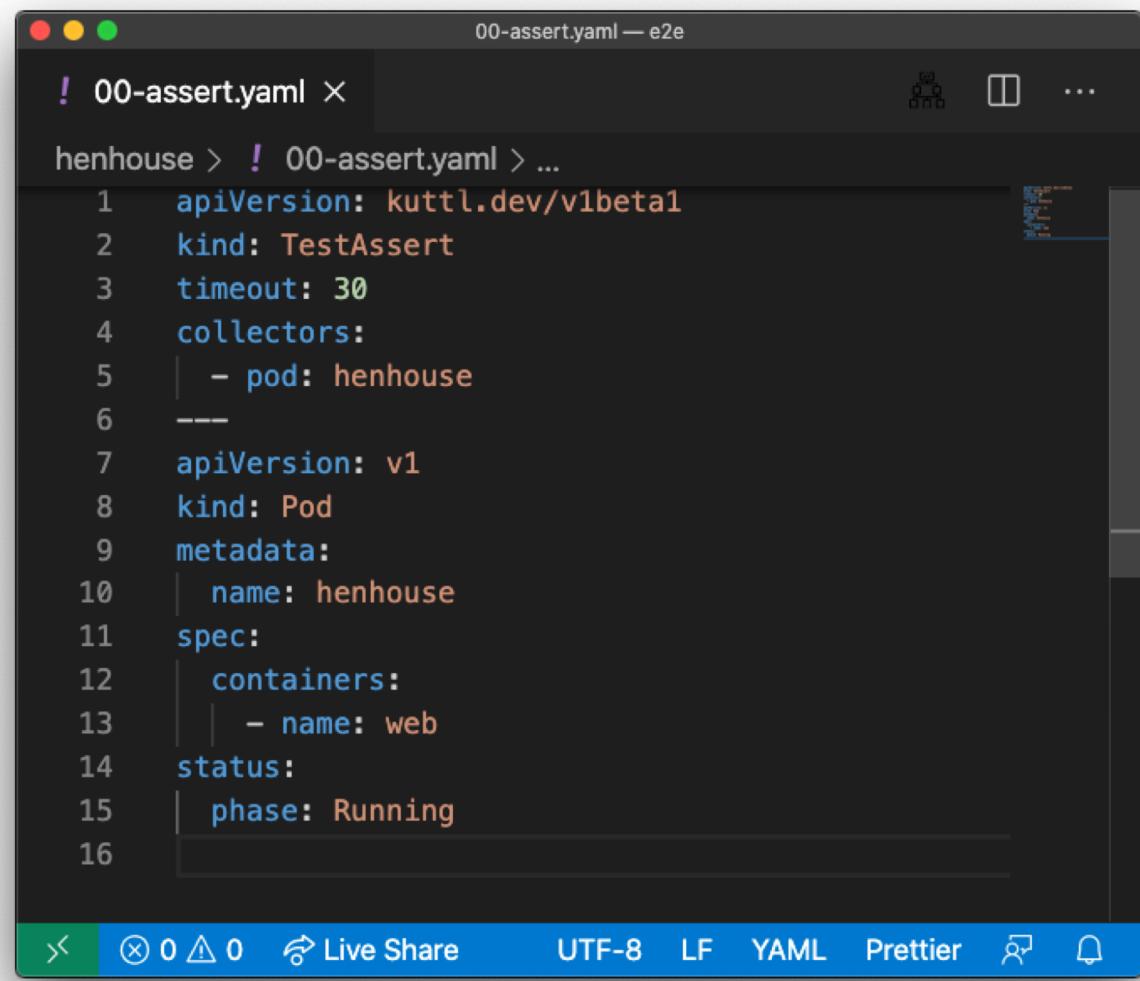
```
apiVersion: kuttl.dev/v1beta1
kind: TestSuite
parallel: 4
timeout: 120
testDirs:
- https://github.com/kensipe/kubecon-us-2020/raw/main/e2e.tgz
# example of using a url for a test dir
```

The code editor interface includes a toolbar with various icons for file operations, a status bar at the bottom showing tabs like `main*`, `Live Share`, and `Prettier`, and a status bar at the top indicating the file is a `test_data`.

KUTTL Assert Collectors

D2
IQ

- Pod
- Command



```
00-assert.yaml — e2e
! 00-assert.yaml ×

henhouse > ! 00-assert.yaml > ...
1   apiVersion: kuttl.dev/v1beta1
2   kind: TestAssert
3   timeout: 30
4   collectors:
5     - pod: henhouse
6   ---
7   apiVersion: v1
8   kind: Pod
9   metadata:
10    name: henhouse
11   spec:
12     containers:
13       - name: web
14   status:
15     phase: Running
16
```

Live Share UTF-8 LF YAML Prettier

KUTTL Reports

D2
IQ

k kuttl test --report xml

- Junit XML
- JSON

The image shows two separate terminal windows side-by-side. Both windows have a dark theme and are titled "kuttl-test.xml". The top window displays JUnit XML test results, and the bottom window displays JSON test results. Each window contains a message stating: "This XML file does not appear to have any style information associated with it. The document tree is shown below." Below this message, the XML document tree is displayed in purple and orange syntax highlighting.

Top Window (JUnit XML):

```
<testsuites name="" tests="2" failures="0" time="4.907">
  <testsuite tests="2" failures="0" time="3.823" name="/Users/kensipe/projects/playground/e2e">
    < testcase classname="e2e" name="henhouse" time="2.795" assertions="1" />
    < testcase classname="e2e" name="doghouse" time="3.821" assertions="1" />
  </testsuite>
</testsuites>
```

Bottom Window (JSON):

```
<testsuites name="" tests="2" failures="1" time="32.135">
  <testsuite tests="2" failures="1" time="31.060" name="/Users/kensipe/projects/playground/e2e">
    < testcase classname="e2e" name="henhouse" time="2.922" assertions="1" />
    < testcase classname="e2e" name="doghouse" time="31.059" assertions="1" >
      < failure message="failed in step 0-install" type="" >resource Pod:kuttl-test-more-bee/doghouse:
        .status.phase: value mismatch, expected: Runing != actual: Running</failure>
      </ testcase >
    </ testsuite >
  </ testsuites >
```

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
```

```
1  apiVersion: kudo.dev/v1alpha1
2  kind: TestSuite
3  manifestDirs:
4  - ./test/manifests/
5  commands:
6    - 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.

```
1  apiVersion: kudo.dev/v1alpha1
2  kind: TestSuite
3  manifestDirs:
4    - ./test/manifests/
5  commands:
6    - command: ./bin/kubectl-kudo init --crd-only
7    - command: ./bin/manager
8      background: true
```

Project KUTTL

More KUTTLing

https://github.com/kudobuilder/kuttl/tree/main/pkg/test/test_data

The screenshot shows a GitHub repository page for 'kudobuilder / kuttl'. The URL in the address bar is https://github.com/kudobuilder/kuttl/tree/main/pkg/test/test_data. The page displays the contents of the 'test_data' directory under the 'main' branch. The directory structure includes subfolders like 'cli-test', 'crd-in-step', 'create-or-update', 'delete-in-step', 'deprecate-test', 'list-pods', 'teststep-apply', 'teststep-assert', and 'teststep-apply'. Each folder has a corresponding commit message and a green checkmark icon indicating it is up-to-date. The commit messages mention switching to 'kuttl.dev/v1beta' and deprecated support for 'kudo.dev'.

Folder	Commit Message	Status
cli-test	Switch to kuttl.dev/v1beta. Deprecated support for kudo.dev (#119)	Up-to-date (green checkmark)
crd-in-step	Bump Dependencies (#25)	Up-to-date (green checkmark)
create-or-update	Bump Dependencies (#25)	Up-to-date (green checkmark)
delete-in-step	Switch to kuttl.dev/v1beta. Deprecated support for kudo.dev (#119)	Up-to-date (green checkmark)
deprecate-test	Switch to kuttl.dev/v1beta. Deprecated support for kudo.dev (#119)	Up-to-date (green checkmark)
list-pods	initial project move from kudo into its own project	Up-to-date (green checkmark)
teststep-apply	Switch to kuttl.dev/v1beta. Deprecated support for kudo.dev (#119)	Up-to-date (green checkmark)
teststep-assert	Switch to kuttl.dev/v1beta. Deprecated support for kudo.dev (#119)	Up-to-date (green checkmark)
teststep-apply	Switch to kuttl.dev/v1beta. Deprecated support for kudo.dev (#119)	Up-to-date (green checkmark)

KUTTL Released



- KUTTL v0.1.0
 - Released March 26, 2020
 - However it was based on 1 year of KUDO development
- KUTTL v0.7.0
 - Released Oct 20, 2020
- Roughly 1 Month Cadence

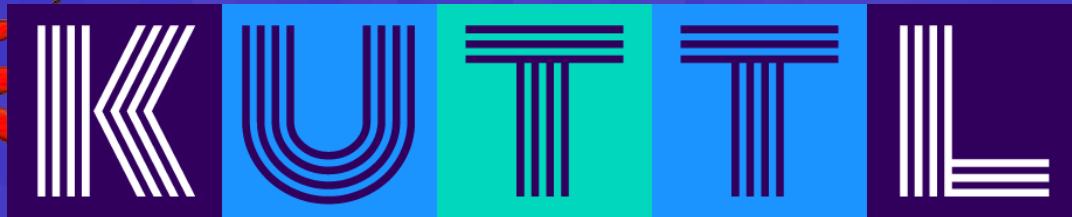
Call to Action

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!



<https://github.com/kudobuilder/kuttl>

@kensipe
slack: kensipe
kensipe@gmail.com