Testing Kubernetes Clusters

Building Confidence in Your Changes

@gjtempleton @ruio @SkyscannerEng

Agenda

1. **A little about us**
2. **The problem(s) we faced**
3. **Potential solutions**
4. **Our solution**
5. **Gotchas, Tips & Other Options**

Who We Are

Guy Templeton

Principal Software Engineer

- Twitter: @**gjtempleton**
- GitHub: **@gjtempleton**

Matteo Ruina

Senior Software Engineer

- Twitter: **@ruio**
- GitHub: **@maruina**

Skyscanner

We’re the travel company who puts you first

- Skyscanner helps millions of people in 52 countries and over 30 languages find the best travel options for flights, hotels and car hire every month.
- Skyscanner is available on desktop, mobile web and its highly rated app has 100 million downloads.
- Working with 1200 travel partners, Skyscanner’s mission is to lead the global transformation to modern and sustainable travel.

K8s @ Skyscanner

- Running K8s since 1.6
- 35+ production clusters across 4 AWS regions
- 475+ services
- 40k+ CPU cores
- 150+ TB RAM

The Problem(s) We Faced

The Kubernetes platform is a complex set of distributed components running together

- Kubernetes has a good set of unit, integration and end-to-end (e2e) tests
- Every addons has its own set of unit tests
- The problem is how to test everything in **your** specific environment

Why testing

- Reduce disruption on clusters ☺
- Increase squad velocity
- Build confidence, for the squad and the users
- Give us a baseline where we can compare the cluster

Exploring Potential Solutions

Manual Acceptance Testing

- Doesn't scale as the number of clusters increases
- Easy to make mistakes
- Inconsistent, even if made into runbooks

Custom Service

- Constantly running tests - easier to catch a degradation
- Harder to block update pipelines on
- No (easy) ability to run ad-hoc tests

Building on top of existing frameworks

- Existing infrastructure and Kubernetes testing frameworks
- Would minimise the custom work we had to do
- Number of different options

kubetest

https://github.com/kubernetes/test-infra/tree/master/kubetest

- Allows for creation and teardown of test clusters
- Supports the addition of custom ginkgo tests
- Supported by the kubernetes project
- Comes with extra overhead
- Not necessarily representative of your production clusters

The Conformance Suite

"*The standard set of conformance tests is currently those defined by the [Conformance] tag in the [kubernetes e2e]() suite.*"

- Easy for us to use the community's existing tests
- Tests a huge range of functionality of clusters
- Very time consuming to run the entire suite
- Some disruptive tests if running the full suite

Sonobuoy

https://sonobuoy.io/

- Open source tool
- Active community and constantly improving project
- Ability to run custom tests via plug-ins

Our Solution

Requirements

- Easy to run from a local laptop and from a CI/CD pipeline
- Can run tests in parallel
- Should expose and store the test results
- Allow alerting if the tests are failing
- Write as little boilerplate as possible

Solution

- Custom Sonobuoy plugin
- Custom smoke tests leveraging Kubernetes/Kubernetes test framework
- Not open source yet ☹

Our setup

- A custom Sonobuoy image with a script to pass a return code and with meaningful output on the failed tests.

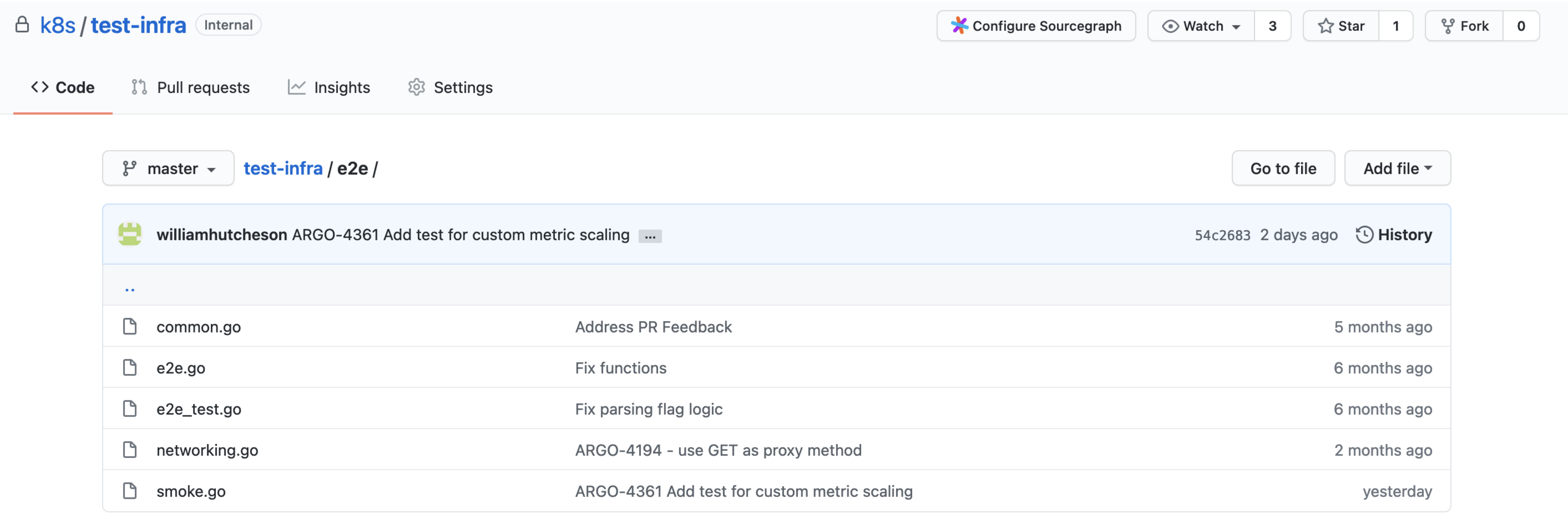
```
/sonobuoy run "$@"

RESULTS_FILE="$(/sonobuoy retrieve)"
# Output the report in a human readable way for debugging purposes
/sonobuoy results "$RESULTS_FILE"

# Use detailed mode for more detailed analysis
FAILED_TESTS=$(/sonobuoy results "$RESULTS_FILE" --mode=detailed | jq 'select (.status=="failed").name')

if [ -z "$FAILED_TESTS" ]; then
    echo "All tests were successful!"
    exit 0
else
    printf "Some tests failed:\n"
    echo "$FAILED_TESTS"
    exit 1
fi
```

Our setup

- The e2e docker image based off the upstream Kubernetes conformance image.

```
# Build stage
FROM golang:1.16 AS build-stage
WORKDIR /go/src/github.skyscannertools.net/k8s/test-infra
COPY . .
RUN make build-e2e

# Create the tests image
FROM k8s.gcr.io/conformance:v1.19.8
ENV UNREADY_NODES="0"
COPY --from=build-stage /go/src/github.skyscannertools.net/k8s/test-infra/scripts/run.sh /run.sh
RUN chmod +x /run.sh
COPY --from=build-stage /go/src/github.skyscannertools.net/k8s/test-infra/bin/e2e.test /usr/local/bin/
COPY --from=build-stage /go/src/github.skyscannertools.net/k8s/test-infra/e2e-repo-spec.yaml /usr/local/
```

Our setup



k8s / test-infra Internal

Configure Sourcegraph | Watch 3 | Star 1 | Fork 0

Code | Pull requests | Insights | Settings

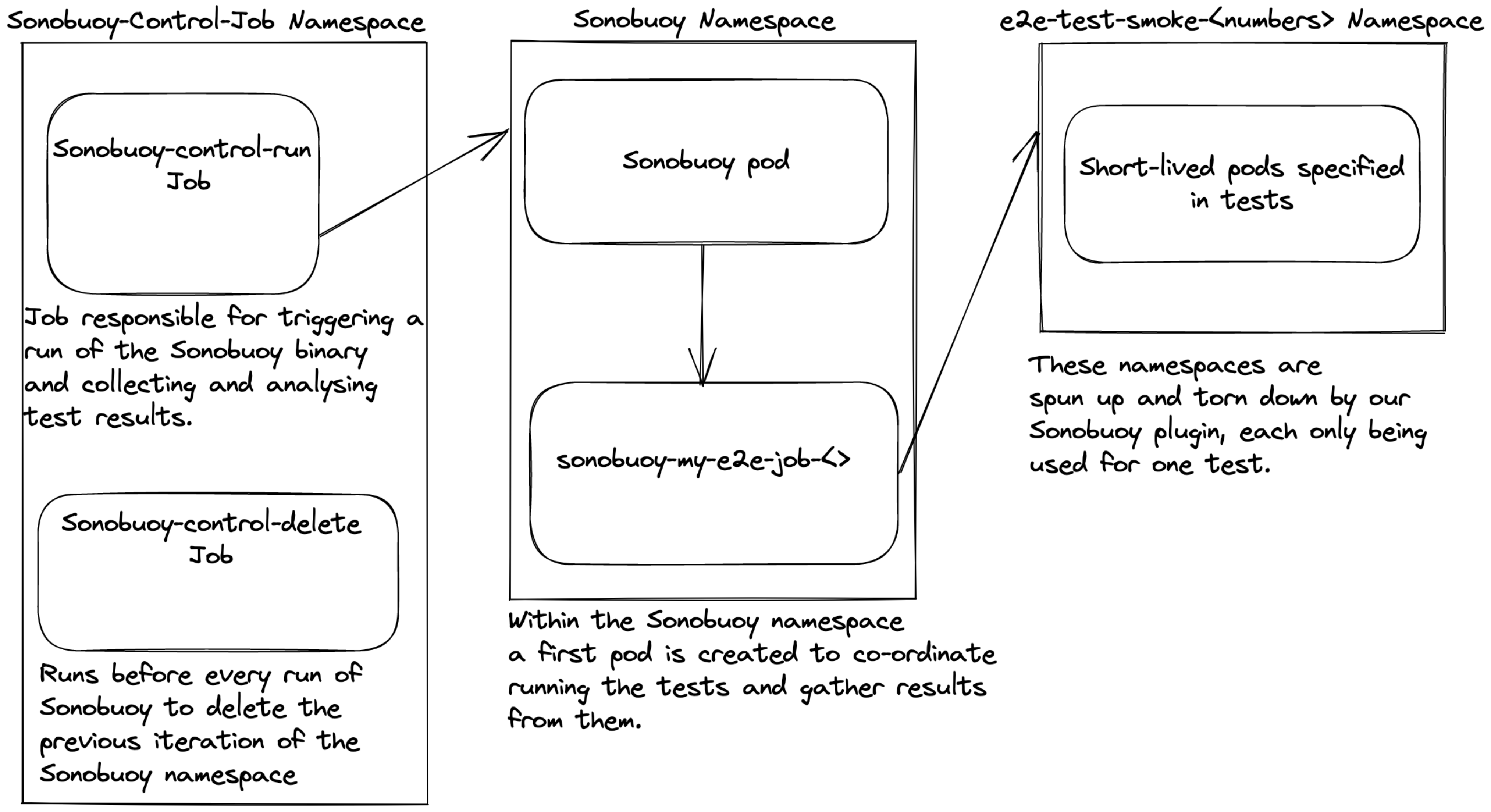
master | test-infra / e2e / | Go to file | Add file

williamhutcheson ARGO-4361 Add test for custom metric scaling … 54c2683 2 days ago History

..

common.go	Address PR Feedback	5 months ago
e2e.go	Fix functions	6 months ago
e2e_test.go	Fix parsing flag logic	6 months ago
networking.go	ARGO-4194 - use GET as proxy method	2 months ago
smoke.go	ARGO-4361 Add test for custom metric scaling	yesterday

Our CD setup


Sonobuoy-Control-Job Namespace
Sonobuoy-control-run Job
Sonobuoy-control-delete Job
Sonobuoy Namespace
Sonobuoy pod
sonobuoy-my-e2e-job-<>
e2e-test-smoke-<numbers> Namespace
Short-lived pods specified in tests

Job responsible for triggering a run of the Sonobuoy binary and collecting and analysing test results.

Runs before every run of Sonobuoy to delete the previous iteration of the Sonobuoy namespace

Within the Sonobuoy namespace a first pod is created to co-ordinate running the tests and gather results from them.

These namespaces are spun up and torn down by our Sonobuoy plugin, each only being used for one test.

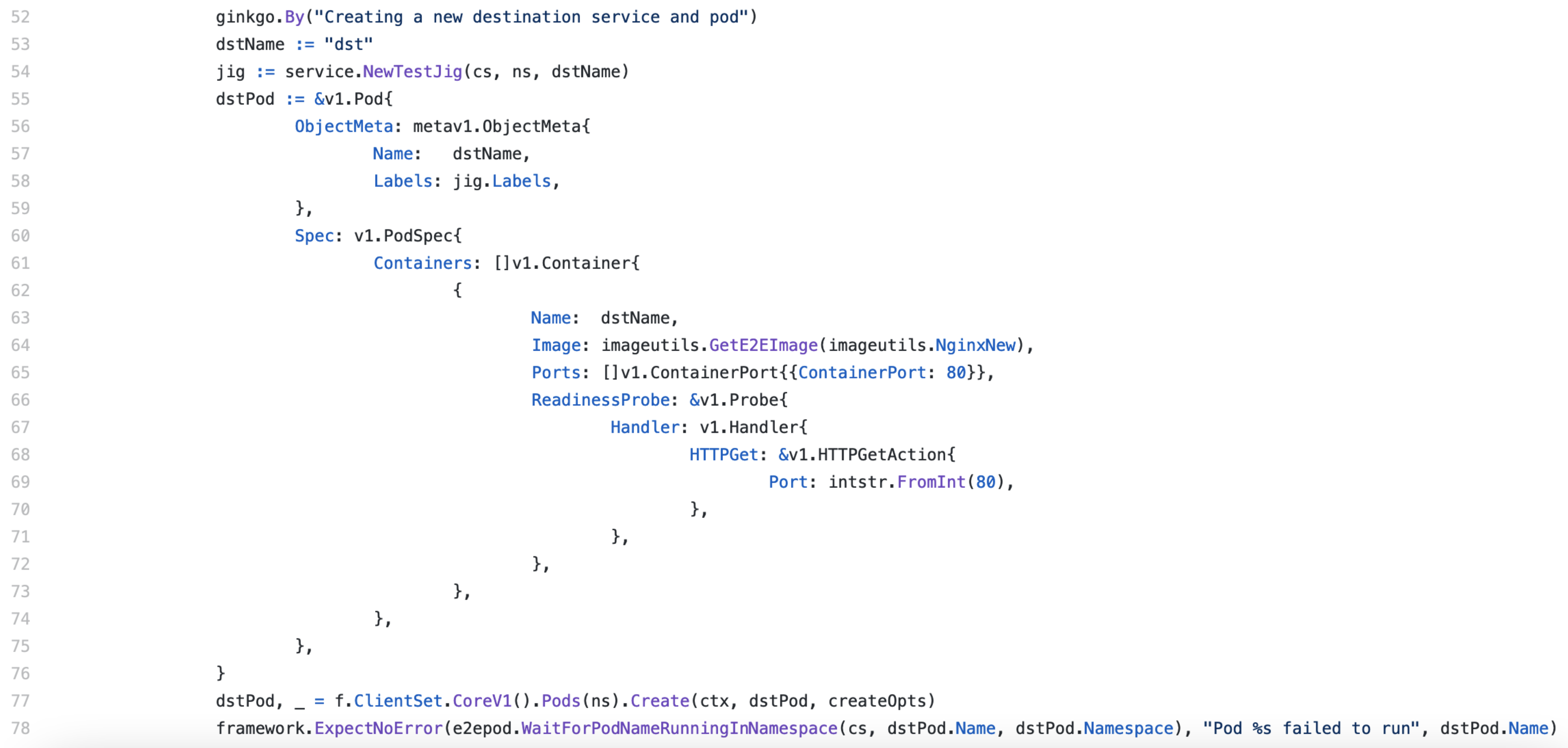
Our e2e tests

- Networking: pod-to-pod communication
- Istio: configuration related to our multi-cluster setup
- DNS: internal and external resolution
- HPA and scaling on custom metrics
- Ability of pods to assume their IAM role
- All nodes are in a known state

Example

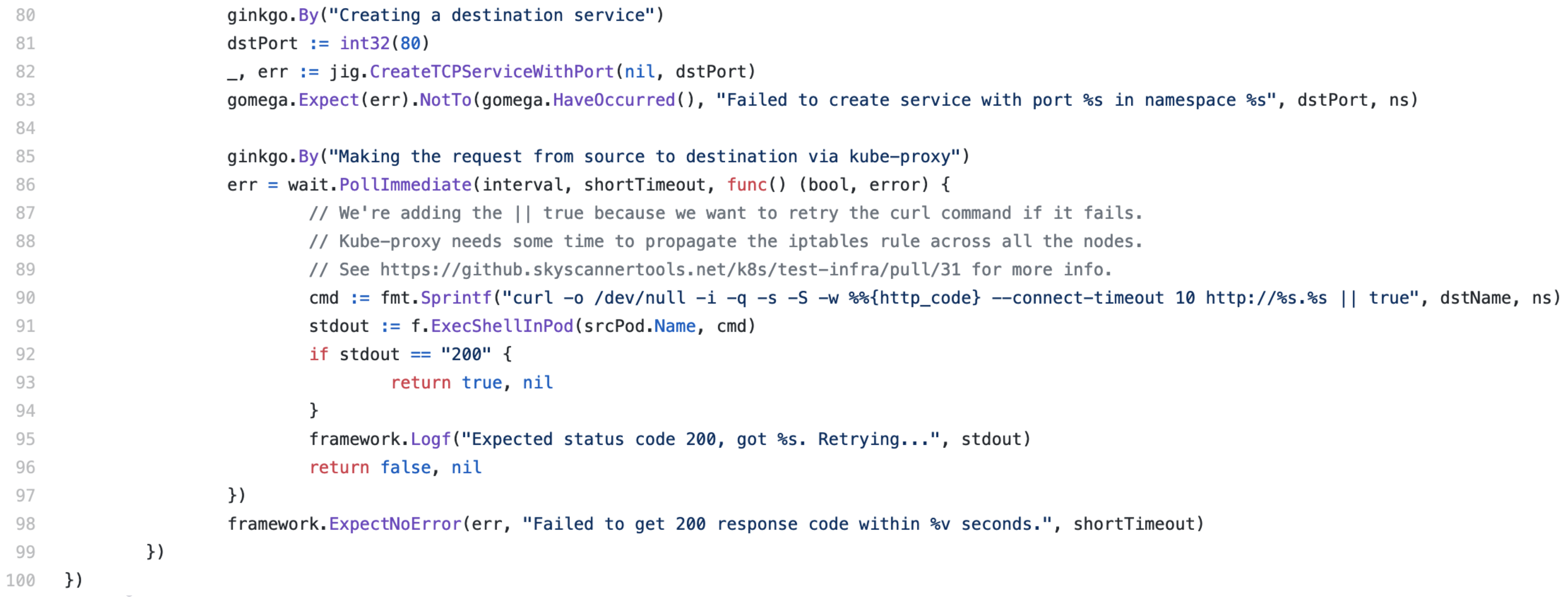
```go
var _ = SmokeDescribe("kube-proxy", func() {
	f := framework.NewDefaultFramework("e2e-test-smoke")

	ginkgo.It("should allow pod to pod traffic", func() {
		ns := f.Namespace.Name
		cs := f.ClientSet
		ctx := context.Background()
		createOpts := metav1.CreateOptions{}

		ginkgo.By("Creating a source pod")
		srcName := "src"
		srcPod := &v1.Pod{
			ObjectMeta: metav1.ObjectMeta{
				Name: srcName,
				Labels: map[string]string{
					"name": srcName,
				},
			},
			Spec: v1.PodSpec{
				Containers: []v1.Container{
					{
						Name:  srcName,
						Image: imageutils.GetE2EImage(imageutils.Agnhost),
					},
				},
			},
		}
		srcPod, _ = f.ClientSet.CoreV1().Pods(ns).Create(ctx, srcPod, createOpts)
		framework.ExpectNoError(e2epod.WaitForPodNameRunningInNamespace(cs, srcPod.Name, srcPod.Namespace), "Pod %s failed to run", srcPod.Name)
```

Example



```go
ginkgo.By("Creating a new destination service and pod")
dstName := "dst"
jig := service.NewTestJig(cs, ns, dstName)
dstPod := &v1.Pod{
	ObjectMeta: metav1.ObjectMeta{
		Name:   dstName,
		Labels: jig.Labels,
	},
	Spec: v1.PodSpec{
		Containers: []v1.Container{
			{
				Name:  dstName,
				Image: imageutils.GetE2EImage(imageutils.NginxNew),
				Ports: []v1.ContainerPort{{ContainerPort: 80}},
				ReadinessProbe: &v1.Probe{
					Handler: v1.Handler{
						HTTPGet: &v1.HTTPGetAction{
							Port: intstr.FromInt(80),
						},
					},
				},
			},
		},
	},
}
dstPod, _ = f.ClientSet.CoreV1().Pods(ns).Create(ctx, dstPod, createOpts)
framework.ExpectNoError(e2epod.WaitForPodNameRunningInNamespace(cs, dstPod.Name, dstPod.Namespace), "Pod %s failed to run", dstPod.Name)
```

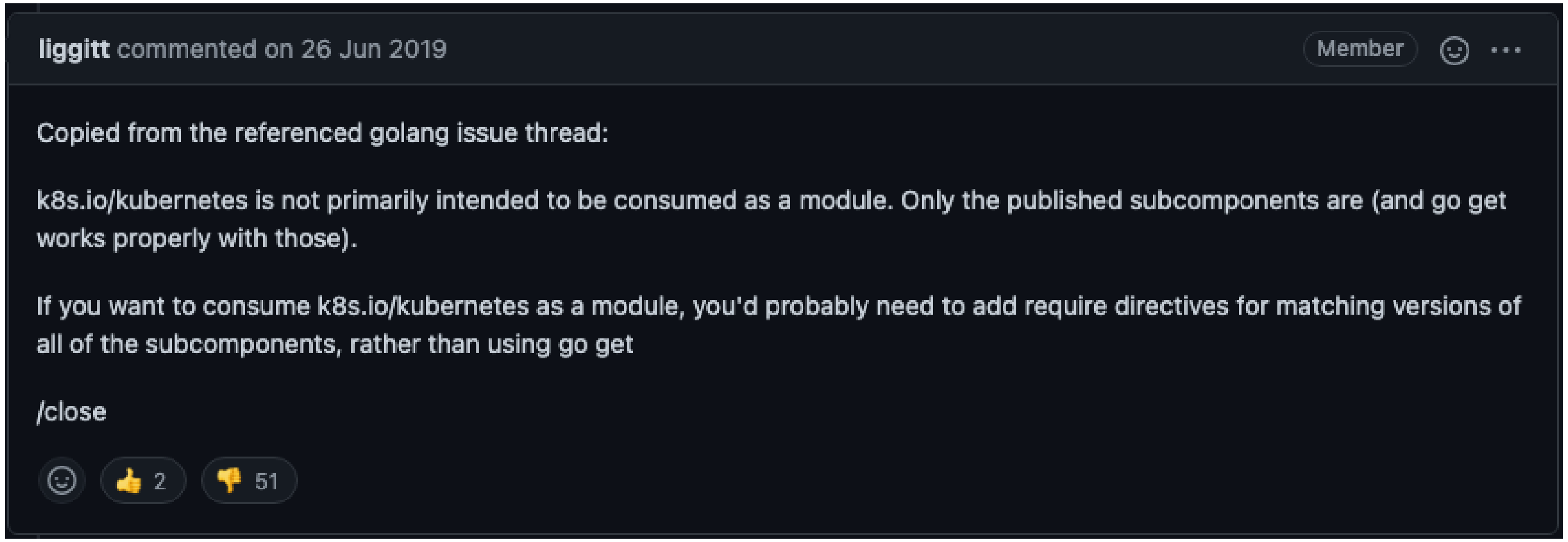
Example



```go
		ginkgo.By("Creating a destination service")
		dstPort := int32(80)
		_, err := jig.CreateTCPServiceWithPort(nil, dstPort)
		gomega.Expect(err).NotTo(gomega.HaveOccurred(), "Failed to create service with port %s in namespace %s", dstPort, ns)

		ginkgo.By("Making the request from source to destination via kube-proxy")
		err = wait.PollImmediate(interval, shortTimeout, func() (bool, error) {
			// We're adding the || true because we want to retry the curl command if it fails.
			// Kube-proxy needs some time to propagate the iptables rule across all the nodes.
			// See https://github.skyscannertools.net/k8s/test-infra/pull/31 for more info.
			cmd := fmt.Sprintf("curl -o /dev/null -i -q -s -S -w %%{http_code} --connect-timeout 10 http://%s.%s || true", dstName, ns)
			stdout := f.ExecShellInPod(srcPod.Name, cmd)
			if stdout == "200" {
				return true, nil
			}
			framework.Logf("Expected status code 200, got %s. Retrying...", stdout)
			return false, nil
		})
		framework.ExpectNoError(err, "Failed to get 200 response code within %v seconds.", shortTimeout)
	})
})
```

Gotchas, Tips & Other Options

Gotchas with our approach


liggitt commented on 26 Jun 2019

Member

Copied from the referenced golang issue thread:

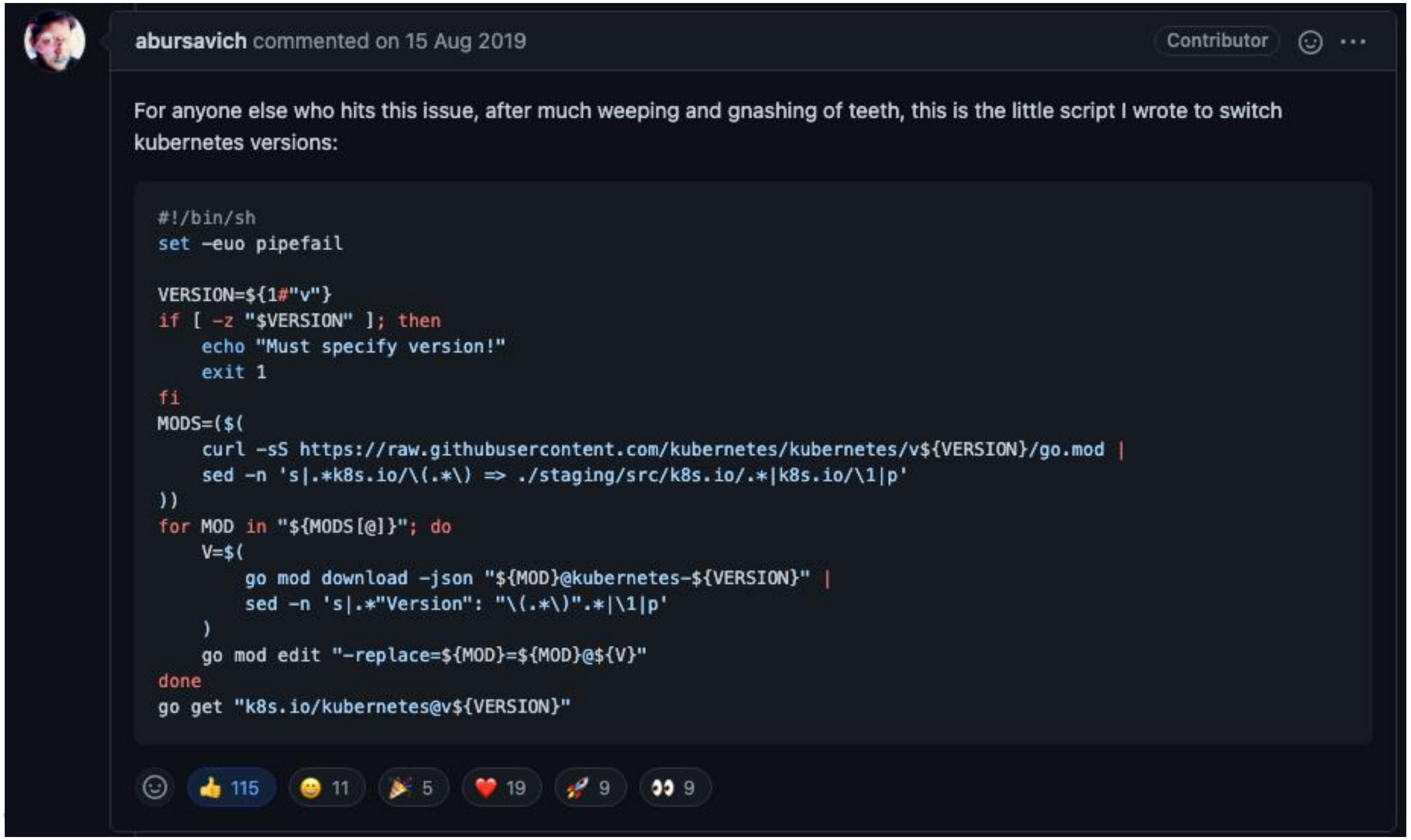
k8s.io/kubernetes is not primarily intended to be consumed as a module. Only the published subcomponents are (and go get works properly with those).

If you want to consume k8s.io/kubernetes as a module, you'd probably need to add require directives for matching versions of all of the subcomponents, rather than using go get

/close

👍 2 👎 51

Gotchas with our approach



abursavich commented on 15 Aug 2019

Contributor

For anyone else who hits this issue, after much weeping and gnashing of teeth, this is the little script I wrote to switch kubernetes versions:

```
#!/bin/sh
set -euo pipefail

VERSION=${1#"v"}
if [ -z "$VERSION" ]; then
    echo "Must specify version!"
    exit 1
fi
MODS=($(
    curl -sS https://raw.githubusercontent.com/kubernetes/kubernetes/v${VERSION}/go.mod |
    sed -n 's|.*k8s.io/\(.*\) => ./staging/src/k8s.io/.*|k8s.io/\1|p'
))
for MOD in "${MODS[@]}"; do
    V=$(
        go mod download -json "${MOD}@kubernetes-${VERSION}" |
        sed -n 's|.*"Version": "\(.*\)".*|\1|p'
    )
    go mod edit "-replace=${MOD}=${MOD}@${V}"
done
go get "k8s.io/kubernetes@v${VERSION}"
```

👍 115 😄 11 🎉 5 ❤️ 19 🚀 9 👀 9

Gotchas with our approach

- Versioning the e2e code
- Kubernetes version upgrades
- Testing images
- Flaky tests
- May need to modify the run.sh script

Tips

- Capture the behavior your users care about
- Leverage the community's efforts – not just the code, also the images
- Figure out which tests can safely be run in parallel

Other Projects

- Kubetest2 framework
  - https://github.com/aws/aws-k8s-tester
- https://testinfra.readthedocs.io/en/latest/
- https://github.com/vapor-ware/kubetest
- https://github.com/kuberhealthy/kuberhealthy

Thanks

@gjtempleton @ruio @SkyscannerEng