SIG-Testing Deep Dive

Katharine Berry @katharine
Cole Wagner @cjwagner

Prow

- CI/CD system built on Kubernetes for Kubernetes
- Executes jobs for building, testing, publishing, and deploying.
- Jobs can be triggered by different types of events and report their status to many different services.
- Also provides GitHub automation:
  - Policy enforcement.
  - Chat-ops via `/foo` style commands.
  - Automatic pull request merging.
- Used by:





ProwJobs

cjwagner (

/test all

BenTheE

/retest

PR #71831 pull-kubernetes-integration #37853

Recent runs || View in Spyglass

PR	uablrek: Only sync addresses of the own ip family
Result	FAILURE
Tests	1 failed / 578 succeeded
Started	2018-12-07 02:36 PST
Elapsed	25m32s
Version	v1.14.0-alpha.0.906+82694a03816e83
Builder	gke-prow-default-pool-3c8994a8-tgkk
Refs	master:2ea50003 71831:5e9fb9e0
pod	dad010b3-fa0b-11e8-bc26-0a580a6c030d
infra-commit	d6f7bb8bf
pod	dad010b3-fa0b-11e8-bc26-0a580a6c030d
repo	k8s.io/kubernetes
repo-commit	82694a03816e83b9059a4a0ad5c2b16f57063877
repos	{u'k8s.io/kubernetes': u'master:2ea50003797952e599ae24cd58e6e9f98caa4d5a,71831:5e9fb9e0f664fd9fb6a29b444f351ab3e809b1d6'}

artifacts | build log

References

PR #71831 Only sync addresses of the own ip family

Test Failures

k8s.io/kubernetes/test/integration/scheduler TestPDBInPreemption 19s

```
Command: go test -v k8s.io/kubernetes/test/integration/scheduler -run TestPDBInPreemption$

I1207 10:54:42.944027  120808 services.go:33] Network range for service cluster IPs is unspecified. Defaulting to {10.0.0.0 ffffff00}.
I1207 10:54:42.944062  120808 master.go:272] Node port range unspecified. Defaulting to 30000-32767.
I1207 10:54:42.944073  120808 master.go:228] Using reconciler:
I1207 10:54:42.945752  120808 clientconn.go:551] parsed scheme: ""
I1207 10:54:42.945783  120808 clientconn.go:557] scheme "" not registered, fallback to default scheme
I1207 10:54:42.945825  120808 resolver_conn_wrapper.go:116] ccResolverWrapper: sending new addresses to cc: [{127.0.0.1:2379 0  <nil>}]
I1207 10:54:42.945881  120808 balancer_v1_wrapper.go:125] balancerWrapper: got update addr from Notify: [{127.0.0.1:2379 <nil>}]
I1207 10:54:42.946320  120808 balancer_v1_wrapper.go:125] balancerWrapper: got update addr from Notify: [{127.0.0.1:2379 <nil>}]
I1207 10:54:42.947014  120808 clientconn.go:551] parsed scheme: ""
I1207 10:54:42.947035  120808 clientconn.go:557] scheme "" not registered, fallback to default scheme
I1207 10:54:42.947068  120808 resolver_conn_wrapper.go:116] ccResolverWrapper: sending new addresses to cc: [{127.0.0.1:2379 0  <nil>}]
I1207 10:54:42.947133  120808 balancer_v1_wrapper.go:125] balancerWrapper: got update addr from Notify: [{127.0.0.1:2379 <nil>}]
I1207 10:54:42.947488  120808 balancer_v1_wrapper.go:125] balancerWrapper: got update addr from Notify: [{127.0.0.1:2379 <nil>}]
I1207 10:54:42.947588  120808 clientconn.go:551] parsed scheme: ""
I1207 10:54:42.947619  120808 clientconn.go:557] scheme "" not registered, fallback to default scheme
```

Show all reviewers

e. Learn more.

Hide all checks

Details

Required Details

Required Details

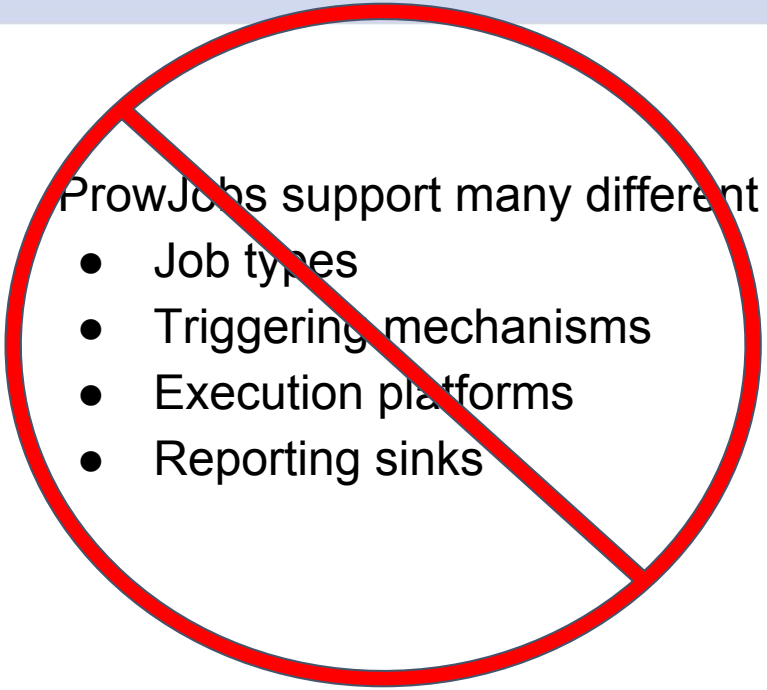
Required Details

Required Details

protected branches.

Life of a ProwJob

Life of a ProwJob



ProwJobs support many different
- Job types
- Triggering mechanisms
- Execution platforms
- Reporting sinks

Only going to talk about
=> Presubmit
=> `/test all`
=> Kubernetes Pod
=> GitHub status context

Webhook Event | Hook | Plank | Sinker | Crier

Life of a ProwJob

GitHub webhook payload:

```
{
  "comment": {
    "body": "/test all",
      ...
    "user": {
      "login": "cjwagner",
    },
  },
  "repository": {
    "full_name": "kubernetes/test-infra",
      ...
  },
  "action": "created",
  "issue": {
    ...
  }
}
```

Webhook Event | Hook | Plank | Sinker | Crier

Life of a ProwJob

Ingress Rule -> Service -> Deployment

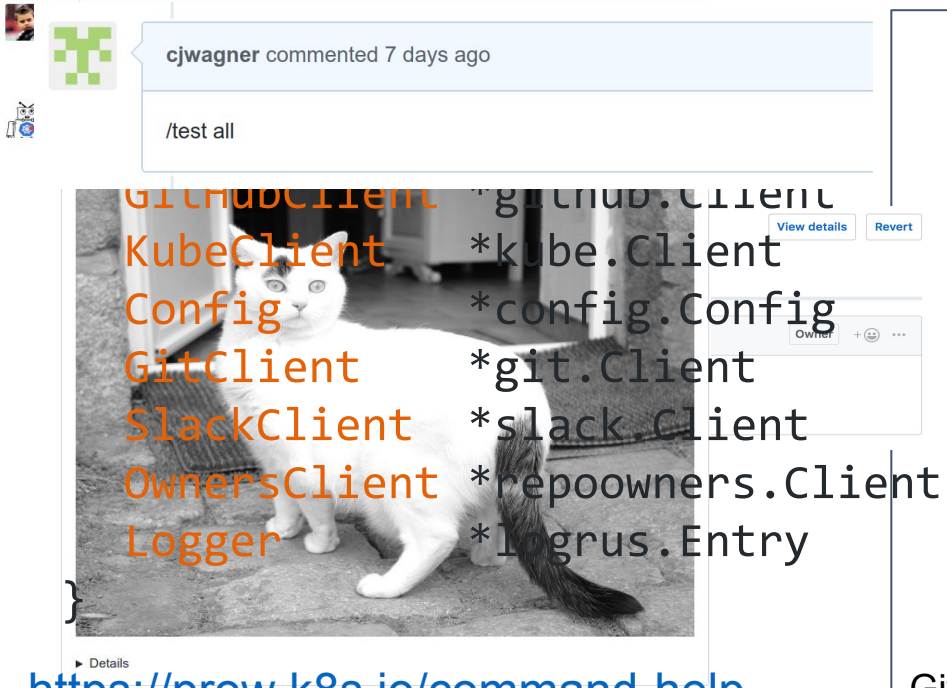
```yaml
rules:
- host: prow.k8s.io
  http:
    paths:
    - path: /*
      backend:
        serviceName: deck
        servicePort: 80
    - path: /hook
      backend:
        serviceName: hook
        servicePort: 8888
```

```yaml
apiVersion: v1
kind: Service
metadata:
 name: hook
spec:
 selector:
   app: hook
 ports:
 - port: 8888
 type: NodePort
```

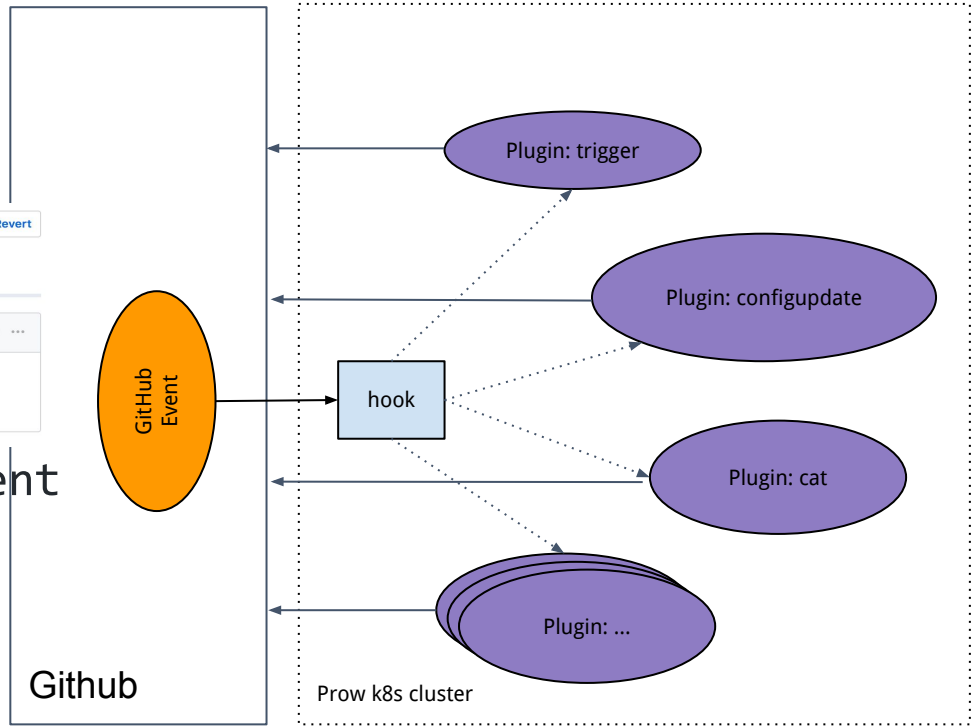
```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
 name: hook
spec:
    ...
 template:
   metadata:
     labels:
       app: hook
   spec:
       ...
```

Webhook Event | Hook | Plank | Sinker | Crier

Life of a ProwJob



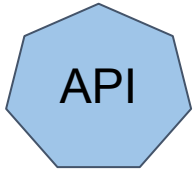
https://prow.k8s.io/command-help



Webhook Event | Hook | Plank | Sinker | Crier

Life of a ProwJob

- `trigger` plugin determines which jobs to run based on the config.
- Creates a new ProwJob custom resource for each job ->
- CRDs lets us store state in the Kubernetes API server.




```yaml
apiVersion: prow.k8s.io/v1
kind: ProwJob
metadata:
  name: 32456927-35d9-11e7-8d95-0a580a6c1504
spec:
  job: pull-test-infra-bazel
  decorate: true
  pod_spec:
    containers:
    - image: gcr.io/k8s-testimages/bazelbuild:0.11
  refs:
    base_ref: master
    base_sha: 064678510782db5b382df478bb374aaa32e577ea
    org: kubernetes
    pulls:
    - author: ixdy
      number: 2716
      sha: dc32ccc9ea3672ccc523b7cbaa8b00360b4183cd
    repo: test-infra
  type: presubmit
status:
  state: triggered
```

Webhook Event | Hook | Plank | Sinker | Crier

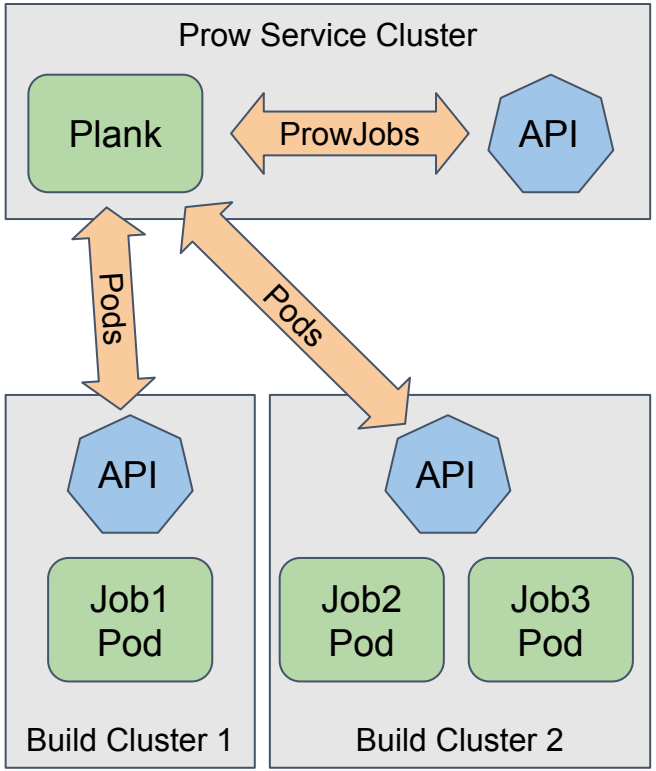
Life of a ProwJob

- Plank syncs ProwJob CRDs with Pods.
- Typical lifecycle:
  - New PJ without a pod

    => Create pod
  - Running PJ with completed pod

    => Complete the PJ (Pass/Fail)
  - Complete PJ

    => Ignore



Webhook Event | Hook | Plank | Sinker | Crier

Life of a ProwJob

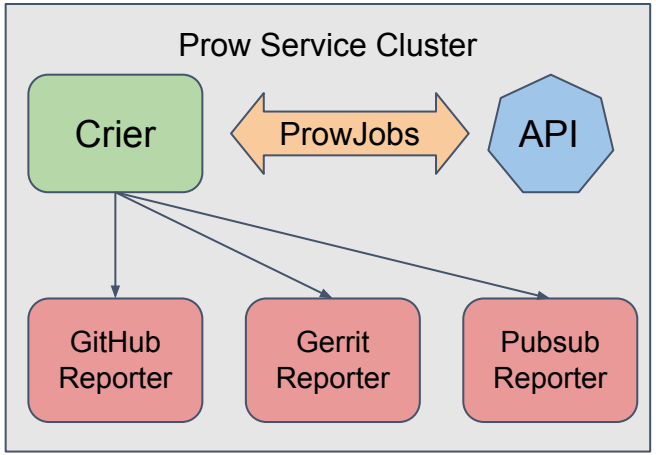
- Sinker is responsible for garbage collection.
  - Completed ProwJobs: after 2 days
  - Completed Pods: after 30 minutes

- Historic results are served from GCS

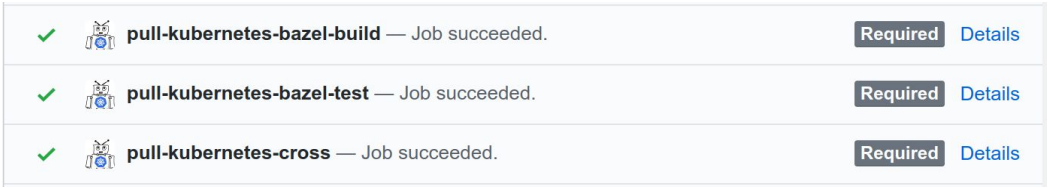
Webhook Event | Hook | Plank | **Sinker** | Crier

Life of a ProwJob

- Crier detects changes to ProwJob CRDs and notifies reporting clients.



- Clients report job status to external services.
  - GitHub status context
  - Gerrit comment
  - Pubsub message



Webhook Event | Hook | Plank | Sinker | Crier

End to End Coverage for Kubernetes

What are we *really* testing?

k8s.io/kubernetes/pkg/

File	master	1.12	1.11
api/	▲49.8%	▲49.6%	48.1%
apis/	▲32.8%	▲32.6%	32.4%
auth/	9.8%	9.8%	9.8%
capabilities/	36.4%	36.4%	36.4%
client/	1.3%	1.3%	1.3%
controller/	30.2%	▲30.3%	28.1%
credentialprovider/	11.1%	11.1%	11.1%
features/	0.0%	0.0%	0.0%
fieldpath/	51.4%	51.4%	51.4%
kubeapiserver/	2.8%	2.8%	2.8%
kubectl/	0.0%	0.0%	0.0%
kubelet/	▲39.4%	▲39.0%	38.3%
master/	16.1%	16.1%	16.1%

Using e2e coverage

- Build time: `make KUBE_BUILD_WITH_COVERAGE=yes`
- Run time:
  - Destination: `KUBE_COVERAGE_FILE` (default: `/tmp/k8s-component.cov`)
  - Flush interval: `KUBE_COVERAGE_FLUSH_INTERVAL` (default: 5s)
  - Run any workload of your choosing

Using e2e coverage


```
kubetest --up --dump-before-and-after --test --down
```

(using the default GCE provider only)

Using e2e coverage

- `gopherage merge`: merge a batch of coverage files
- `gopherage diff`: to constrain what coverage counts
- `gopherage aggregate`: to aggregate across multiple runs

All use standard go coverage files, so any coverage-related tooling works!

Using e2e coverage

Existing jobs

Conformance

Coverage: 29%

All tests

Coverage: 35%

How does any of this work?

Kubernetes:
A distributed system of unit tests

How does any of this work?


```
21 + # arguments: target, item1, item2, item3, ...
22 + # returns 0 if target is in the given items, 1 otherwise.
23 + kube::util::array_contains() {
```

thockin on Aug 31 **Member**

My litmus test for "should I write this in shell" has always been "once you pass an array to a function, the answer becomes no". I don't see an immediately better answer here, but this sets off all sorts of alarms.

👍 2

How does any of this work?


```
create_coverage_dummy_test "${package}"

go test -c -o "$(kube::golang::outfile_for_binary "${package}"
"${platform}")" \
  -covermode count \
  -coverpkg k8s.io/...,k8s.io/kubernetes/vendor/k8s.io/... \
  "${build_args[@]}" \
  -tags coverage \
  "${package}"
```

How does any of this work?


```
func TestMain(m *testing.M) {
    // Get coverage running
    coverage.InitCoverage("${name}")
    // Go!
    main()
    // Make sure we actually write the profiling information to disk,
    // if we make it here.
    coverage.FlushCoverage()
}
```

How does any of this work?


```
func InitCoverage(name string) {
    destFile = "/tmp/k8s-" + name + ".cov"
    flushInterval := 5 * time.Second

    flag.CommandLine.Parse([]string{"-test.coverprofile", destFile})

    go wait.Forever(FlushCoverage, flushInterval)
}
```

How does any of this work?


```
func FlushCoverage() {
    tests := []testing.InternalTest{}
    benchmarks := []testing.InternalBenchmark{}
    examples := []testing.InternalExample{}
    var deps fakeTestDeps

    dummyRun := testing.MainStart(deps, tests, benchmarks, examples)
    dummyRun.Run()
}
```


Demo time!


KubeCon
CloudNativeCon
North America 2018
Questions?

Get Involved!

- Learn more about Prow and deploy your own!
- Contribute to Prow!
- Use and contribute to Kind!
- Join the SIG-Testing Slack channel!