Verify Your Kubernetes Clusters with Upstream e2e Tests

Kenichi Omichi - NEC

Open Source Summit NA 2018
8/29/2018
Agenda

About me
Upstream e2e tests
Motivation to use e2e tests for production
How to use e2e tests
Test failures, solutions and workarounds
- PV Protection tests
- StatefulSet tests
- HPA tests
- MetricsGrabber tests
- Resource usage tracking tests
Summary
About me

Kenichi Omichi from NEC
Software engineer
Open source contributor

OpenStack
- OpenStack/QA core developer, ex-PTL (Project Team Lead)
- OpenStack/Nova core developer

Kubernetes
- Established community member
- CKA (Certified Kubernetes Administrator)
Upstream e2e tests

<table>
<thead>
<tr>
<th>Test end-to-end (e2e) behavior of system level</th>
</tr>
</thead>
<tbody>
<tr>
<td>● Unit tests cannot catch unforeseen changes at system level sometimes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CI for pull requests</th>
</tr>
</thead>
<tbody>
<tr>
<td>● Pull requests need to pass subset of e2e tests before merging</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>e2e tests continue increasing</th>
</tr>
</thead>
<tbody>
<tr>
<td>● The latest k8s(for v1.12) contains 1033 tests</td>
</tr>
</tbody>
</table>

![Graph showing e2e specs]
Motivation to use e2e tests for production

- k8s contains a lot of features and a lot of corresponding e2e tests
  - E2e test number is still increasing

- e2e tests are useful for verifying k8s clusters
  - We can find lacks of configurations or misconfigurations before productions
  - Rarely you face test issues

- Investigations of test failures are good chances to learn k8s deeply
  - We can understand overview/detail of k8s features and test expectations
How to use e2e tests – Build e2e test binary

Check version of target k8s cluster

$ kubectl version
...
Server Version: version.Info{Major:"1", Minor:"11", GitVersion:"v1.11.1"}

Download k8s source code and checkout same version

$ go get k8s.io/kubernetes
$ cd $GOPATH/src/k8s.io/kubernetes
$ git checkout refs/tags/v1.11.1

Build e2e test binary

$ sudo /usr/local/go/bin/go run hack/e2e.go -- --build
How to use e2e tests – Run all e2e tests

Choose provider type for your cloud platform:
- skeleton, local, gce, gke, aws

(skeleton) Export KUBE_MASTER_IP and KUBE_MASTER_MASTER

```
$ export KUBE_MASTER_IP="192.168.1.108:6443"
$ export KUBE_MASTER=k8s-master
```

Run all e2e tests

```
$ export KUBECONFIG=$HOME/admin.conf
$ go run hack/e2e.go -- --provider=skeleton --test
```
How to use e2e tests – Run Conformance tests

What are conformance tests

- A subset of e2e tests (167/999 in v1.11)
- Corresponding tests for interoperable features sig-architecture has approved
- k8s community expects to pass on any k8s clusters
- To get certification of Certified Kubernetes program, every vendor needs to pass conformance tests

Run conformance tests

$ export KUBECONFIG=$HOME/admin.conf
$ export KUBERNETES_CONFORMANCE_TEST=true
$ go run hack/e2e.go -- --provider=skeleton --test --test_args="--ginkgo.focus=¥[Conformance¥]"
We would face many failures

Conformance tests can be operated easily
- Just 167 tests of 999 all e2e tests
- These tests have been selected by expecting to be pass on any clusters
- It takes 1.5 hours only

But a little difficult to run ALL e2e tests
- It could take 12 hours if many failures
- Much timeout without any configuration, retry-number and timeout are hardcorded in test code

There is a lot of points we need to take care to use e2e tests for productions
Test failures, Solutions and Workarounds
Test environment

**HW**
- Dell desktop PCs * 2: Core i5-6500 CPU @3.2GHz, 16GB MEM
- Intel NUC PCs *2: Core i3-7100 @2.4GHz, 32GB MEM

**IaaS: OpenStack Queens**
- OS: Ubuntu 16.04 LTS
- Ubuntu package

**Kubernetes v1.11.1**
- 2 VMs(master, node) on top of IaaS
- OS: Ubuntu 16.04 LTS
- Install tool: kubeadm
- Network: Flannel
Failure case 1: PV Protection tests

2 PV Protection tests were failed

Verify "immediate" deletion of a PV that is not bound to a PVC
Verify that PV bound to a PVC is not removed immediately

Root Cause

- Tests were failed to check PV Protection flag in PV due to nonexistent flag
- The flag is set after the PV phase becomes “Available” from “Pending”, but tests didn’t check the phase
- e2e test machine was faster than k8s cluster, so the flag was nonexistent

Solution

- Add waiting for “Available” phase before checking the flag in tests
- PR: https://github.com/kubernetes/kubernetes/pull/67520 (Merged 8/20/2018)
You might want to backport the PR to e2e

But e2e doesn’t start if changing e2e code due to version difference

```bash
$ go run hack/e2e.go -- --provider=skeleton --test
exit status 1
```

Specify `--check-version-skew=false` to skip version check

```bash
$ go run hack/e2e.go -- --provider=skeleton --test --check-version-skew=false
Client Version: version.Info{Major:"1", Minor:"11+", ...}
Server Version: version.Info{Major:"1", Minor:"11", ...}
... Running Suite: Kubernetes e2e suite
```
Failure case 2: StatefulSet tests

3 StatefulSet tests were failed

should adopt matching orphans and release non-matching pods
should not deadlock when a pod's predecessor fails
should provide basic identity

Root Cause

● Since v1.11 k/k in-tree openstack-cloud-provider is deprecated, and the external openstack-cloud-provider is recommended

● Document issue: Tried using the external one for PersistentVolume, but misconfigured StorageClass because StorageClass example was written for the internal one

Solution

● Specify “openstack.org/standalone-cinder” instead of “kubernetes.io/cinder” as the provisioner in StorageClass

● PR: https://github.com/kubernetes/cloud-provider-openstack/pull/261 (Merged)
Failure case 3: HPA tests - 1

8 HPA (Horizontal Pod Autoscaling) tests were failed

[1] ReplicationController light [It] Should scale from 1 pod to 2 pods
[2] ReplicationController light [It] Should scale from 2 pods to 1 pod
[3] Should scale from 1 pod to 3 pods and from 3 to 5
[4] Should scale from 5 pods to 3 pods and from 3 to 1
[5] Deployment [It] Should scale from 1 pod to 3 pods and from 3 to 5
[6] ReplicaSet [It] Should scale from 5 pods to 3 pods and from 3 to 1
[7] ReplicaSet [It] Should scale from 1 pod to 3 pods and from 3 to 5 and verify decision stability
[8] ReplicationController [It] Should scale from 5 pods to 3 pods and from 3 to 1 and verify decision stability

How does HPA work

RC/RS/Deployment

scale

metrics.k8s.io API

update

fetch

HPA
Failure case 3: HPA tests - 2

Root Cause
- HPA feature requires metrics-server to fetch CPU usage on k8s v1.9+, but I didn’t deploy it

Solution
- Deploy metrics-server
  ```
  $ git clone https://github.com/kubernetes-incubator/metrics-server
  $ cd metrics-server/
  $ kubectl create -f deploy/1.8+/ readOnlyPort: 10255/tcp
  ```
- Configure all kubelets to allow 10255/tcp as read-only-port
  ```
  readOnlyPort: 10255
  ```
Failure case 3: HPA tests - 3

Still 6 HPA (Horizontal Pod Autoscaling) tests were failed

- [1] Solved
  - Controller light [It] Should scale from 1 pod to 2 pods
- [2]
  - Controller light [It] Should scale from 2 pods to 1 pod
- [3] Should scale from 1 pod to 3 pods and from 3 to 5
- [4] Should scale from 5 pods to 3 pods and from 3 to 1
- [5] Deployment [It] Should scale from 1 pod to 3 pods and from 3 to 5
- [6] ReplicaSet [It] Should scale from 5 pods to 3 pods and from 3 to 1
- [7] ReplicationController [It] Should scale from 1 pod to 3 pods and from 3 to 5 and verify decision stability
- [8] ReplicationController [It] Should scale from 5 pods to 3 pods and from 3 to 1 and verify decision stability

What does test [3] expect the result in autoscaling 1 to 3 pods

1. Set target cpu: 20%
2. Make 50% CPU workload
3. Scale to 3 pods:
   $$\frac{50\%}{20\%} = 2.5$$
Failure case 3: HPA tests - 4

Root Cause

- Actual CPU workload (100%) was bigger than expected (50%)
- Then HPA scaled pods to 5 instead of expected 3:
  \[ \frac{100\%}{20\%} = 5 \]
- CPU workload is generated by many resource consumer processes. Each process tried to make 2% CPU workload and 25 processes existed, but it is hard to make such small workload and actual workload was 4%:
  - Test expectation: 50% = 2% * 25 processes
  - Actual workload: 100% = 4% * 25 processes

Solution

- Change the CPU workload of each process to 10% for the test stability
  \[ 50\% = 10\% \times 2 \text{ processes} + 5\% \times \text{process} \]
- PR: https://github.com/kubernetes/kubernetes/pull/67053 (Merged 8/7/2018)
- All HPA tests have passed successfully
Failure case 4: MetricsGrabber test

1 MetricsGrabber test was failed

should grab all metrics from a Scheduler.

- k8s API call in the test was failed

```
GET /v1/namespaces/kube-system/pods/
kube-scheduler-k8s-master:10251/proxy/metrics
```

Root Cause

- kube-scheduler listens 10251/tcp locally(127.0.0.1) and it denies connections to its node ip-address.

Workaround

- Change listening ip-address to the host address in /etc/kubernetes/manifests/kube-scheduler.yaml
- Issue: https://github.com/kubernetes/kubernetes/issues/67685

We can get to know what APIs are called in tests with "-v=8":
```
$ go run hack/e2e.go -- --test --test_args="-v=8"
```
Failure case 5: Resource usage tracking tests - 1

2 Resource usage tracking tests were failed

- resource tracking for 0 pods per node
- resource tracking for 100 pods per node

Tests expect CPU usage of kubelet is under 10%, but it was over 12%

Root Cause

- k8s VMs were running on NUC PCs
- CPU speed is not enough
  - NUC PCs: i3-7100 @2.4GHz
  - Desktop PCs: i5-6500 CPU @3.2GHz
Solution

- Migrate VMs to another host which has better CPUs
- Question: Isn’t such test environment supported with e2e test?
- Issue: [https://github.com/kubernetes/kubernetes/issues/67621](https://github.com/kubernetes/kubernetes/issues/67621)
Summary

Comformance tests are useful for verifying basic functions

All e2e tests are useful for verifying extended functions (HPA, StatefulSets, etc.)
- We can find lacks of configurations or misconfigurations before productions
- Rarely you face test issues

Investigations of test failures are good chance to learn k8s deeply
- We can understand overview/detail of k8s features and test expectations

Great to bring test issues up to community
- Few tests can run on some environments only (GCP, AWS, etc as upstream CI)
- Discuss test results which are gotten from different environments, we can make tests more stable
- That is great contribution to community by making development smooth
References

- e2e tests
  - https://github.com/kubernetes/community/blob/master/contributors/devel/e2e-tests.md

- Conformance tests
  - https://github.com/kubernetes/community/blob/master/contributors/devel/conformance-tests.md

- Certified Kubernetes Program
  - https://www.cncf.io/certification/software-conformance/