Kubernetes Storage-SIG Test Strawman

Overview

Kubernetes volume storage testing is unique in that it requires specific backing storage and cloud providers to fully test. Each plugin must be tested within the proper environment for full validation which may include running on a cloud provider or on nodes that have client software and physical access to specific backing storage types. In some cases the backing storage may be dynamically provisioned or hosted from a container (hyper-converged storage).

All testing discussed here is fully comprehensive and will target the more difficult case of E2E. Integration tests may also run in these environments.

Kubernetes Volume Types

Cloud Types

Cloud based volume types don't typically require storage drivers on the node. The node runs in the cloud as a virtual machine and storage is attached to the node which shows up as a block device. Kubernetes storage layer orchestrates the attach/detach of the block device to the VM.

CI Tested

- GCE_PD
- Azure
    - http://blog.kubernetes.io/2016/06/bringing-end-to-end-testing-to-azure.html

Gapped

- AWS EBS
- Cinder
- VSPHERE

Non Cloud Storage Types

The non-cloud (on prem) storage types are generally dedicated hardware that runs as a peer to the kubernetes cluster. Each node may need drivers and network considerations to access the storage. Most of these types qualify as “software defined storage”, meaning that there’s a software layer which handles the backing block device access. Many of these types can be simulated without actual hardware, except Fibre Channel.

CI Tested

- NFS

Gapped:

- CephFS (in volume test but missing in conformance test)
- FibreChannel (need HW env)
- ISCSI (in volume test but missing in conformance test)
- Glusterfs (in volume test but missing in conformance test)
- CEPH RBD (in volume test but missing in conformance test)

Misc Types

The kubernetes volume layer hosts multiple utility volume types. These types provide various function which aren't exactly storage, but useful at the volume layer.

CI Tested

- ConfigMap
- DownwardAPI ?
- emptyDir
- Host_path
- Secret

Gapped

- Flexvolume
- Flocker
- Git_repo

Current Kubernetes CI

CI Failure Types

Build failures can be done proactive in PRs or reactive during master commit & merge. The proactive build failures are resource intensive as it requires new build & test for every commit on every PR. Reactive build failures will block the build if tests fail post-merge. A commit that fails these tests must be manually reverted from master and tests re-run.

There are strategies which require less resources for the proactive storage test failures such as only testing PRs with a specific label.

Federated Volume Testing

Kubernetes CI supports external testing failures:
https://github.com/kubernetes/test-infra/blob/master/docs/federated_testing.md

This will allow external CI to run tests and block the main build.

Ideally each volume type that's not tested as part of the Kubernetes CI is tested externally in the correct environment. The tests would run once or more daily and fail the main CI based on the results. Each external testing environment should test only the plugin's it's responsible for and not the entire stack.

E2E Environment

Google runs Kubernetes E2E (and other tests) in GCE and GKE. Any non-cloud volume testing is done by simulated storage clusters in a container (hyper-converged). E2E cloud specific tests are only run if a supporting cloud provider is detected.

Existing Technical Limitations

Current volume design requires storage client software installed on host. This is not possible in the kubernetes CI environment. For this reason a limited set of volume types are tested as part of the kubernetes main CI.

Additionally, tests run in parallel so a single shared storage cluster of each volume type will not work. A dedicated instance of each storage type must be stood up and torn down for every test or dynamic provision & delete used. Hyper-converged storage endpoints are used as dedicated clusters for NFS tests. Other volume types aren't hyper-converged or require privileged container mode which is not allowed.

Proposed Kubernetes-Storage CI Environment

Google to continue full test of GCE and the misc storage types for every PR. As technical limitations are removed through evolution of the storage framework, the kubernetes CI will run additional volume tests.

Red Hat is investigating running E2E storage tests against EBS on AWS and hooking into kubernetes CI. Test failures will be against master build and not per-commit/PR. This may be expanded to every commit/PR or some other strategy in the future.

In an effort to eventually CI test the on-prem storage types Red Hat will investigate containerizing the remaining non-cloud storage types as endpoints for E2E tests and removing any privileged container requirements. Red Hat will report back to the storage-sig when complete.

Types of Test Coverage

Function Coverage

The volume plugins are tested to verify the following functionalities:

- SELinux
- FS Group
- Persistent Volume Claim
- Dynamic Provisioning (if supported)
- Attach/detach (if supported)
- Basic POSIX I/O test (test suites: fsx or TBD)
- Volume exclusion (including NoDiskConflicts scheduling policy, GCE PD RWO, RBD fencing, etc)
- Other features when they are implemented
    - Quota
    - Snapshot

Stress Test

This is to be coordinated with SIG-Scaling. The goal is to ensure storage stack functions as expected under stress and provide evidence to identify system bottleneck.

Some test cases are

- Create and delete many volumes, PVs, and PVCs.
- Create and delete many Pods with multiple volume types
- Create and delete Pods with volumes, disconnect backing storage during test and verify Pods and mounts are cleaned up properly.

Regression Test

This is to be coordinated with SIG-Test. The goal is to ensure baseline test cases are consistently reproduced throughout releases.

There is also open questions such as how to detect performance abnormality.

Operation Issues

Open questions on adding test cases, establishing baseline, coordinating trusted CI, and issue triage are still under investigation.