sig-cluster-lifecycle v1.6 objectives

Lucas Käldström 15.11.2016

During the sig-cluster-lifecycle in-person meeting, KubeCon and the Dev summit, we came up with what we should as a group focus on in time for v1.6.

TODO: Describe what kubeadm is and isn't

On a high level, we'd like to:

- Move kubeadm out of the monorepo into kubernetes/kubeadm
- Maybe make a new stable version of kubeadm that defaults to deploying v1.5 clusters w/ etcd3, Done!
- Implement Joe's new token bootstrap framework in the core so we can use it, in progress!
- Refactor kubeadm so all individual steps are invokable separately
- Refactor the kubeadm API, in progress!
- Provide self-hosting as one control-plane boot-up alternative, in progress!
- Provide basic node and cluster conformance tests, in progress!
- Document more, especially upgrades the hard way

Additional information:

- Nice-to-haves
- Further features we want on our road to production-grade
- What we depend on from others

Move kubeadm out of the monorepo into kubernetes/kubeadm

Initially Mike Danese, Brian Grant or someone else with permission will create kubernetes/kubeadm and we will start to centralize all our issues and work to that repo

luxas has some WIP work of how we can do this code-wise here: https://github.com/luxas/kubeadm-source

We have to wait until master opens for v1.6 until we can split kubeadm out.

TODO:

- Merge all pending PRs [assignee: luxas]
  - https://github.com/kubernetes/kubernetes/pulls?utf8=%E2%9C%93&q=is%3Apr%20is%3Aopen%20kubeadm%20label%3Algtm
  - Done!
- Incorporate the system verification check into the preflight checks [assignee: luxas]
  - https://github.com/kubernetes/kubernetes/pull/36334
  - Done!
- Switch to client-go instead of using the unversioned client [assignee: caesarxuchao]

    - https://github.com/kubernetes/kubernetes/pull/36673

- Create the kubeadm repository in https://github.com/kubernetes/kubeadm
    - Vendor minimal amount of code (ideally only client-go and some other deps)
    - Deal with the sig-api-machinery stuff that doesn't work great out-of-monorepo
    - Establish a "release process" w/ dockerized builds => makefile
    - Rename k8s.io/kubernetes/cmd/kubeadm to k8s.io/kubeadm/cmd/kubeadm
    - In progress: https://github.com/kubernetes/kubeadm/pull/72

Release a new stable version of kubeadm that defaults to deploying v1.5 clusters w/ etcd3

Right before we split kubeadm out of the repo, we should consider to release a new stable version with v1.5 kubernetes and etcd3. This will just be a small follow-up to the earlier released version: https://github.com/kubernetes/kubernetes/issues/34884

TODO:

- Merge the right PRs [assignee: luxas]
    - https://github.com/kubernetes/kubernetes/pulls?utf8=%E2%9C%93&q=is%3Apr%20is%3Aopen%20kubeadm%20label%3Algtm
    - Done!
- Use etcd3 by default: https://github.com/kubernetes/kubernetes/pull/35921
    - Done!
- Use k8s v1.5 by default when released: https://github.com/kubernetes/kubernetes/pull/37222
    - Done!
- Make a docker-based installer that works with CoreOS as well
    - PoC is here: https://github.com/luxas/kubeadm-installer
    - This will be a relatively small thing to implement but it will be significant improvement for our users
    - Will be deferred to v1.6!

Implement Joe's new token bootstrap framework in the core so we can use it

- Implement the BootstrapSigner and TokenCleaner controllers [assignee: mikedanese, jbeda primarily]
    - https://github.com/kubernetes/kubernetes/pull/36101
- Implement noop commands in kubeadm:
    - https://github.com/kubernetes/kubernetes/pull/35805
- Hook up `kubeadm token` commands to the new token management stuff
    - ?
    - This will require a v1.6.0-alpha.x build of the control plane

Refactor kubeadm so all individual steps are invokable separately

This will probably take some time to get right, but we should aim to have a working PoC (still alpha) in v1.6 of this

TODO:

- Refactor the code structure to support this
- Decide what the command should be called
    - kubeadm phase
    - kubeadm manual
    - ?
- Reflect this in kubeadm init so it logs output in the form
    - [init] Running the phase certificates.
    - [init] You could invoke this separately as well: kubeadm phase certs --foo
    - [certificates] ca.crt created, blablabla
    - [certificates] service account created
    - [init] Running the phase controlplane
    - [init] You could invoke this separately as well: kubeadm phase controlplane
    - [controlplane] Using selfhosting as the boot-up method

Refactor the kubeadm API

This is a hard one as well.

TODO:

- The first breaking piece of it is here:
    - Provide Token, File and HTTPS as different node join auth paths
    - https://github.com/kubernetes/kubernetes/pull/36462
- We need to think about it with well-defined phases as well
- Certificates API will be beta in v1.6?
    - https://github.com/kubernetes/kubernetes/pull/34488?
- Provide both kubenet and cni as viable options for networking
    - Make it possible to specify cni manifest directly
- Remove the old env parameters

Provide self-hosting as one control-plane boot-up alternative

The conclusion at the dev summit and in-person meeting was that not everyone wants self-hosting, but many do. We will/should implement it as the default “happy path” with kubeadm, but swappable if the user runs the control plane with binaries for instance

TODO:

- Provide self-hosting as an alternative to static pods for easier upgrading
    - Still etcd is run in a static pod locally

        - Should reuse bootkube's userspace checkpointer initially

Provide basic node and cluster conformance tests

We identified three tiers of testing we'd like to have:

- Basic node validation
    - This is our preflight checks. We will integrate sig-node's system verification test that validates the kernel and the cgroup mounts
    - Implementation: https://github.com/kubernetes/kubernetes/pull/36334
    - This is a must-have in v1.6, and we will get it in, already lgtm'd
    - Done!
- Cluster-wide conformance validation
    - Basic question: I have my cluster up and running. Can I validate I've configured it right so I can be sure that it satisfies the functionality of the Kubernetes Standard Base™?
    - Probably the right way to do this is to package this test in a `kubeadm verify` command
    - We could both link in some crucial tests into the binary that always are required and/or provide a way to fetch the full `e2e_test` conformance test binary from the internet and execute it in the right way
    - We should keep these tests minimal if possible and aim that they do not take long to execute (5min or so would be preferable)
    - This would be really useful to get into v1.6, and we should aim to come up with a good solution to this. We should begin with defining the Kubernetes Standard Base™
- Cluster-wide full e2e validation
    - Full-blown testing with lots of functionality tests for all areas of the cluster
    - Maybe with disruptive tests also
    - This will probably not make v1.6

Document more, especially upgrades the hard way

Documentation is crucial, and we definitely need more of it.

TODO:

- "Upgrading Kubernetes the hard way"
- Upgrading a kubeadm-built cluster w/ self-hosting on the "happy path"
- Upgrade/rollback and failure disaster recovery

Nice-to-haves

TODO:

- CoreOS, Fedora 24 and Debian 8.2 support (might have to set cgroup => document!)
- Others?

Further features we want on our road to production-grade

Write down here what blocks us from recommending kubeadm as a production-grade solution

TODO:

- Make configuration flow down from the control plane to all kubelets
    - Requires Dynamic Kubelet Settings
- It uses CNI and it has some rough edges
    - For instance, and most critically, the CNI HostPort/HostIP issue
- Multi-master/HA
    - Depends on a lot of other general improvements, for example we need a HA client first
- We should support other runtimes
    - cri-o when ready, rkt and hyper eventually
- The cloud provider functionality should be configurable
    - The user should just specify if he wants a cloud provider or not at cluster init and install it when the control-plane is up with `kubectl apply -f http://my-favourite-cloudprovider.yaml`
- Secure mutual auth HA etcd cluster
- Further steps on the path to a usable production cluster (definitely post 1.6)
    - Essential add-ons
    - Onboarding of users
        - Access control
        - kubectl configuration management
        -
    - Dashboard UI
    - Logging
    - Monitoring
- TODO: explain exactly what kops for instance needs yet before they can use kubeadm
- TODO: Addon management proposal in detail
    - We might be able to reuse helm
    - It would be cool with a place for “official” place for addons that are required in all production clusters. For instance kubernetes/addons

What we depend on from others

- The sig-api-machinery stuff
- Releasing from multiple repos, we maybe could fix this ourselves
- A dynamic kubelet configuration that serves as a base for all kubelets
- Smarter clients that support finding api-servers
- Take advantage of the cloud provider refactoring work

- Shared volume mounts
- CNI HostPort/HostIP
- Different runtimes

From what we wrote down at the 2017 year planning:
https://docs.google.com/spreadsheets/d/154cAee2mOn3LoQDgpgG2ZzAIdlIQ_KnMIghGjnGwQ1w

Conformance Testing: Packaging both node, kubernetes standard base and full e2e testing into kubeadm
Split kubeadm up into logical phases and provide individual commands to run
Add a new token bootstrap mechanism with a configmap in the kube-public namespace
Upgrade/Rollback (degraded/lame duck mode, backwards compatibility)
HA: make it possible for kubelets to automatically discover new apiservers dynamically (maybe via the configmap), remove dependencies on local filesystem on the master (tokens, addon manager)
Documentation
Use kubelet dynamic settings to configure all kubelets
Cluster Config (document and standardize the known configmaps that configures a cluster)
Extend the support for kubeadm to more distributions
In place upgrades
Upgrade testing
Define the kubernetes standard base => what's the minimum features a k8s cluster should have in order to be counted as "supported"
Make kubeadm composable and stable enough so kops can use it
How to deal with add-on management that isn't in the "minimum standard uniform k8s cluster" like monitoring, logging, dashboard, etc. Should we use something like helm?