SIG Cloud Provider: Intro

KubeCon San Diego 2019

Fabio Rapposelli, VMware (@frapposelli)
Nick Turner, AWS (@nckturner)

kubernetes

Agenda

- SIG Intro
  - charter
  - areas of focus
  - subprojects
- Discussions!
  - Cloud provider extraction/migration
  - SIG governance
  - User groups
  - Code organization
  - Enhancements

Discussion

- Vote on discussion topics
- Choose based on interest
- Discussion about some gnarly questions

Engagement:

- Be respectful
- Introduce yourself the first time you speak
- Goal is to get thoughts/participation from everyone
- There is a talking stick (???) - literally - please raise hand and we'll get it to you

SIG Cloud Provider Charter

This charter adheres to the conventions described in the Kubernetes Charter README and uses the Roles and Organization Management outlined in sig-governance.

Scope

SIG Cloud Provider's mission is to simplify, develop and maintain cloud provider integrations as extensions, or add-ons, to Kubernetes clusters.

In Scope

Areas of Focus:

- Extension points between Kubernetes and any cloud provider
- APIs/interfaces for efficiently provisioning/de-provisioning cloud resources (nodes, routes, load balancers, etc.)
- Configuration of cluster components to enable cloud provider integrations
- Testing and testing frameworks to ensure vendor neutrality across all cloud providers

Code, Binaries and Services

- Common interfaces consumed by cloud providers
- Cloud-controller-manager and core controllers
- All cloud provider repositories in the Kubernetes organization
- E2e tests for cloud provider specific functionality
- All the subprojects formerly owned by cloud provider SIGs
- Any new subproject that is cloud provider specific, unless sponsored by another SIG

Cross-cutting and Externally Facing Processes

- We ensure that cloud providers are testing and reporting to test grid.
- We provide documentation on configuring Kubernetes clusters with cloud providers enabled.
- We work with new cloud providers who want to host their code in the Kubernetes organization.
- We engage with SIGs owning other external components of Kubernetes (CNI, CSI) to ensure to ensure a consistent integration story for users.

Subprojects

- **Cloud-provider-extraction-migration**
  - Legacy-cloud-providers, apiserver-network-proxy...
- **Provider-alibaba-cloud**
  - Cloud provider, CSI
- **Provider-aws**
  - Cloud provider, ALB, EBS-CSI, EFS-CSI, FSX-CSI, IAM, Encryption
- **Provider-azure**
  - Cloud Provider, Disk-CSI, File-CSI
- **Provider-gcp**
  - Cloud Provider, Disk-CSI, File-CSI
- **Provider-ibmcloud**
  - Cloud Provider
- **Provider-openstack**
  - Cloud Provider, Ingress, Cinder-CSI, Manila-CSI, KMS, Keystone-Auth{n,z}
- **Provider-vsphere**
  - Cloud Provider, CSI

Cloud Provider Extraction/Migration

Context:

Removal of in-tree provider code is a major ongoing task being led by this SIG. We recently migrated most of the in-tree providers to staging, next steps are to drive adoption of out-of-tree providers and solidify a migration mechanism for existing clusters.

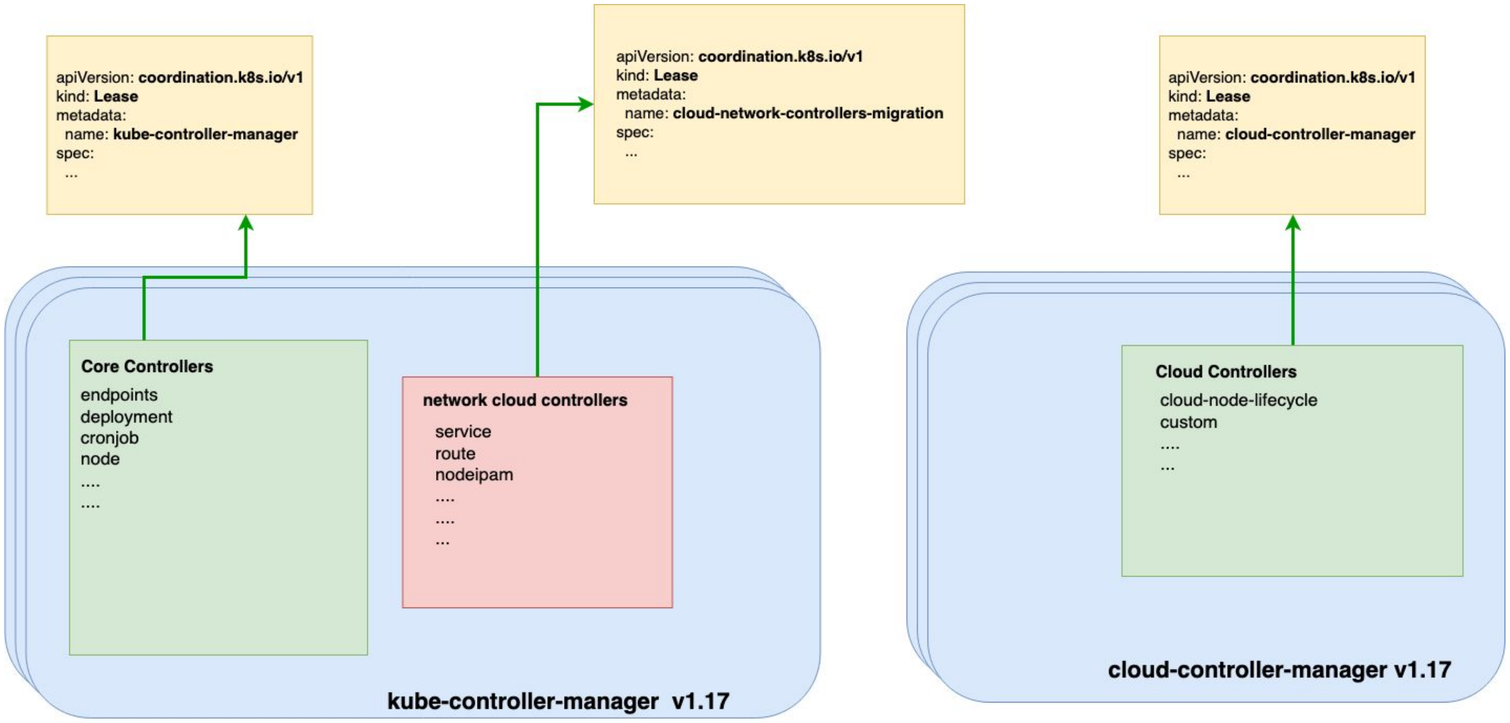
See the Removing in-tree providers KEP for information on the legacy-cloud-providers location in kubernetes/staging.

Cloud Provider Extraction/Migration

The Cloud Controller Manager Migration KEP details the migration process for cloud provider controller manager loops into a separate, cloud provider owned binary called the CCM.

- Primary and Migration locks
- Each lock defines an immutable set of controllers who can acquire it
- In each Kubernetes version, each controller will use exactly one lock
- Some set of migrating controllers will use a migration lock for at least two versions, the version before the migration and the version after the migration

Cloud Provider Extraction/Migration



```
apiVersion: coordination.k8s.io/v1
kind: Lease
metadata:
  name: kube-controller-manager
spec:
  ...
```

```
apiVersion: coordination.k8s.io/v1
kind: Lease
metadata:
  name: cloud-network-controllers-migration
spec:
  ...
```

```
apiVersion: coordination.k8s.io/v1
kind: Lease
metadata:
  name: cloud-controller-manager
spec:
  ...
```

Core Controllers

endpoints
deployment
cronjob
node
....
....

network cloud controllers

service
route
nodeipam
....
....
...

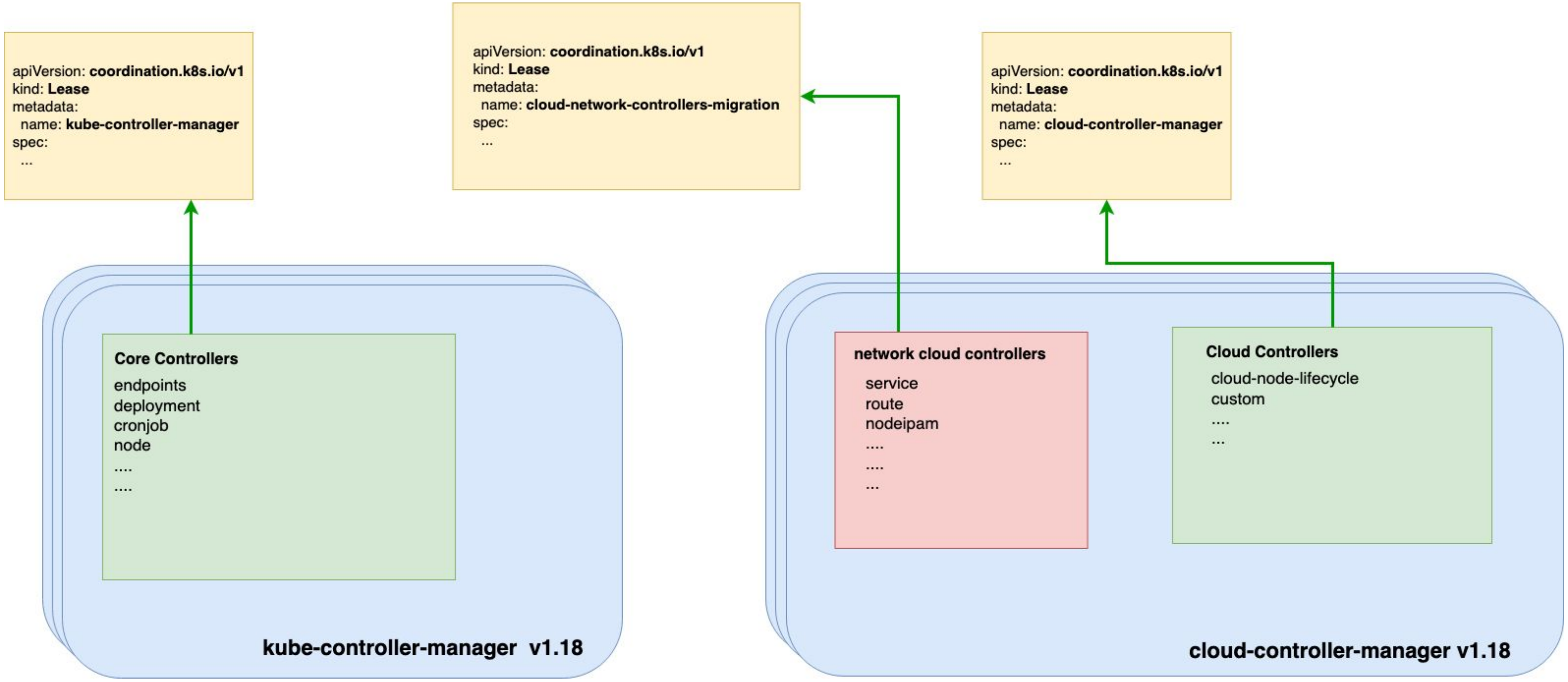
kube-controller-manager v1.17

Cloud Controllers

cloud-node-lifecycle
custom
....
...

cloud-controller-manager v1.17

Cloud Provider Extraction/Migration



```
apiVersion: coordination.k8s.io/v1
kind: Lease
metadata:
  name: kube-controller-manager
spec:
  ...
```

```
apiVersion: coordination.k8s.io/v1
kind: Lease
metadata:
  name: cloud-network-controllers-migration
spec:
  ...
```

```
apiVersion: coordination.k8s.io/v1
kind: Lease
metadata:
  name: cloud-controller-manager
spec:
  ...
```

Core Controllers

endpoints
deployment
cronjob
node
....
....

kube-controller-manager v1.18

network cloud controllers

service
route
nodeipam
....
....
...

Cloud Controllers

cloud-node-lifecycle
custom
....
...

cloud-controller-manager v1.18

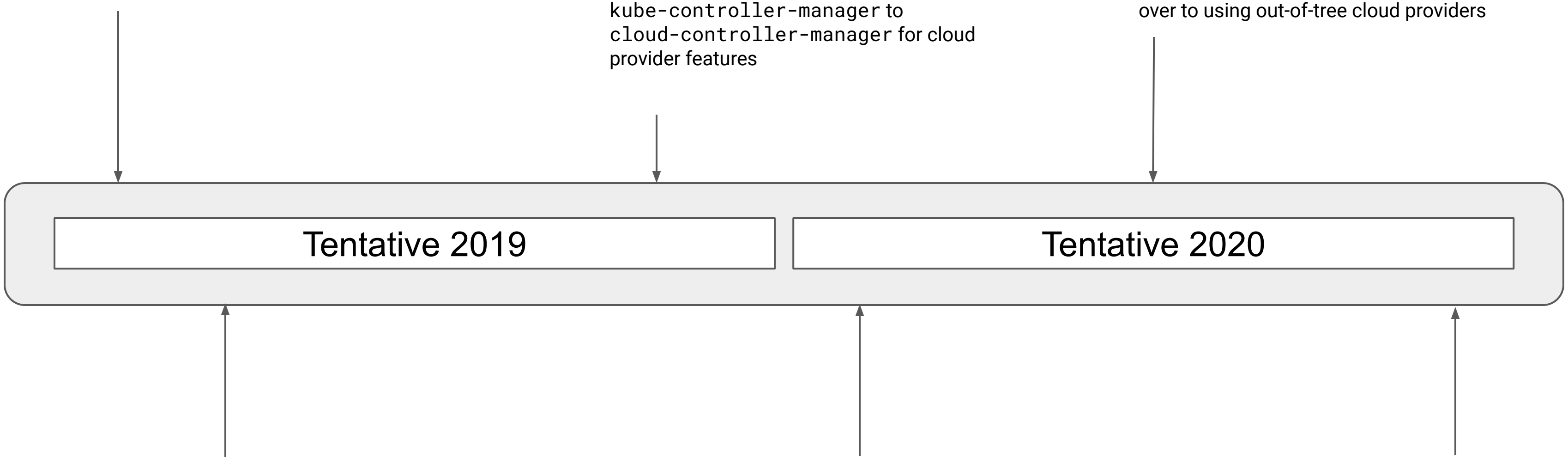
Start migration of in-tree cloud providers out-of-tree by removing dependencies to `k8s.io/kubernetes` in `pkg/cloudprovider/providers` into their respective out-of-tree repositories

Promote new clusters to use `cloud-controller-manager` as the default

Work on migration strategy for existing production clusters to go from using `kube-controller-manager` to `cloud-controller-manager` for cloud provider features

Migration strategy should be battle tested and ready for production clusters.

All Kubernetes clusters (using supported versions) should have a clear path to migrate over to using out-of-tree cloud providers



Tentative 2019

Tentative 2020

All dependencies to `k8s.io/kubernetes` should be removed and all in-tree cloud providers should be moved to their respective staging directories.

Announce removal of in-tree cloud providers

Cut over in-tree cloud provider code into their respective out-of-tree repos.

There should be no cloud provider specific code in `k8s.io/kubernetes` or any of its staging directories.

Cloud Provider Extraction/Migration

Questions for Discussion:

1. What is a reasonable timeline to fully remove in-tree providers?
2. What is the current state of existing cloud providers and their out-of-tree providers?

Code Organization for Cloud Provider Subprojects

Context:

With SIG Cloud Provider sponsoring new projects from various providers, it’s often difficult to discover what repos are out there today. Should we set standards/conventions for code organization/structure?

Questions for Discussion:

1. Consistent naming for new repositories
2. Well-defined ownership of provider-specific repositories
3. Improve discoverability of new code/subprojects

Goal: improve overall code organization for cloud providers in Kubernetes

Enhancements!

Context:

Over the past few months, we’ve had a number of enhancement requests to the cloud provider integrations. Some which are needed as part of the extraction/migration efforts.

Possible Enhancements:

1. Out-of-tree cloud-based image credential provider
2. Improved rate limiting libraries/utilities for cloud APIs
3. APIServer network proxy (replaces provider-specific ssh integrations)
4. Dynamic volume operations based on node state (terminated, stopped, shutdown, etc)
5. ???

Goal: agree on a set of enhancements to prioritize for the next few releases

SIG Governance

Context:

As previously agreed with the Steering Committee the folding of all the existing provider-specific SIGs into subprojects happened on Friday July 12th 2019.

Questions for Discussion:

1. Has this change affected you? Either positive or negative, we'd like to hear your opinions.
2. If you're a user, do you feel you have enough support with the new meeting/slack structure?
3. Are you getting value out of the newly-created user groups? If your platform doesn't have it, would you be interested in starting one?

Goal: learn from change, tighten the feedback loop

Questions & Discussions

KubeCon San Diego 2019

Fabio Rapposelli, VMware (@frapposelli)
Nick Turner, AWS (@nckturner)

kubernetes