Storage version migrator

Never worry about stale API objects in etcd again
Chao Xu (caesarxuchao@github)

Senior Software Engineer at Google Kubernetes team. Active contributor to Kubernetes.
Agenda

How Kubernetes stores objects in etcd

Risks of stale objects in etcd

The storage migrator
Agenda

How Kubernetes stores objects in etcd

Risks of stale objects in etcd

The storage migrator
$ curl 127.0.0.1:8080
{
  "paths": [
    "/apis/admissionregistration.k8s.io/v1beta1",
    "/apis/apiextensions.k8s.io/v1beta1",
    "/apis/apiregistration.k8s.io/v1",
    "/apis/apiregistration.k8s.io/v1beta1",
    "/apis/apps/v1",
    "/apis/apps/v1beta1",
    "/apis/apps/v1beta2",
    "/apis/authentication.k8s.io/v1",
    "/apis/authentication.k8s.io/v1beta1",
    "/apis/authorization.k8s.io/v1",
    "/apis/authorization.k8s.io/v1beta1",
    "/apis/autoscaling/v1",
    "/apis/autoscaling/v2beta1",
    "/apis/autoscaling/v2beta2",
    "/apis/batch/v1",
    "/apis/batch/v1beta1",
    "/apis/batch/v2alpha1",
    ...
  ]
}
$ curl 127.0.0.1:8080
{
  "paths": [
    "/apis/admissionregistration.k8s.io/v1beta1",
    "/apis/apiextensions.k8s.io/v1beta1",
    "/apis/apiregistration.k8s.io/v1",
    "/apis/apiregistration.k8s.io/v1beta1",
    "/apis/apps/v1",
    "/apis/apps/v1beta1",
    "/apis/apps/v1beta2",
    "/apis/authentication.k8s.io/v1",
    "/apis/authentication.k8s.io/v1beta1",
    "/apis/authorization.k8s.io/v1",
    "/apis/authorization.k8s.io/v1beta1",
    "/apis/autoscaling/v1",
    "/apis/autoscaling/v2beta1",
    "/apis/autoscaling/v2beta2",
    "/apis/batch/v1",
    "/apis/batch/v1beta1",
    "/apis/batch/v2alpha1",
    ...
  ]
}
Multi-version RESTful API

RESTful API

/apis/batch/v1beta1/jobs

/apis/batch/v1/jobs

/apis/batch/v2alpha1/jobs
Multi-version RESTful API

RESTful API

GET /apis/batch/v1beta1/jobs/namespace/
default/pi

GET /apis/batch/v1/jobs/namespace/
default/pi

GET /apis/batch/v2alpha1/jobs/namespace/
default/pi
Multi-version RESTful API

RESTful API

**GET** /apis/batch/v1beta1/jobs/namespace/default/pi

**GET** /apis/batch/v1/jobs/namespace/default/pi

**GET** /apis/batch/v2alpha1/jobs/namespace/default/pi

apiVersion: batch/v1
kind: Job
metadata:
  name: pi
spec:
  template: ...
Creating an Object

RESTful API

/apis/batch/v1beta1/jobs
/apis/batch/v1/jobs
/apis/batch/v2alpha1/jobs
Creating an Object

RESTful API

/apis/batch/v1beta1/jobs
/apis/batch/v1/jobs
/apis/batch/v2alpha1/jobs
Creating an Object

API Version of the request must match the URL

POST https://localhost:8080/apis/batch/v1beta1/namespaces/default/jobs

apiVersion: batch/v1beta1
kind: Job
metadata:
  name: pi
spec:
  template:
    spec:
      containers:
        - name: pi
          image: perl
...
Creating an Object

// batch/v1 is the storage version, tied to the apiserver version

apiVersion: batch/v1
kind: Job
metadata:
  name: pi
spec:
  template:
  ...

POST
/apis/batch/v1beta1/jobs
/apis/batch/v2alpha1/jobs
/apis/batch/v1/jobs

Convert & Write
Storage Version

- Built-in resources: tied to API server version
- CRD: defined in CRD.Spec
RESTful API

/apis/batch/v1/jobs

/apis/batch/v1beta1/jobs

/apis/batch/v2alpha1/jobs

GET

User

Master

Etcd
RESTful API

/apis/batch/v1/jobs
/apis/batch/v1beta1/jobs
/apis/batch/v2alpha1/jobs

// apiserver converts object
// to the requested version

```yaml
apiVersion: batch/v2alpha1
kind: Job
metadata:
  name: pi
spec:
  template:
    ...
```
“Why”s
● Why does the API server support multiple versions of an API?
"Why"s

- Why does the API server support multiple versions of an API?

Server-client compatibility
If only one version is supported...

POST /apis/batch/v1beta1/jobs

v1.x
If only one version is supported...

POST to /apis/batch/v1beta1/jobs

/apis/batch/v1/jobs

v1.x+1
If only one version is supported...

POST to /apis/batch/v1beta1/jobs

Error code 404

/apis/batch/v1/jobs

v1.x+1
Why does the API server support multiple versions of an API?

Server-client compatibility

Why does the API server convert objects to storage version before writing to etcd?
“Why”s

- Why does the API server support multiple versions of an API?
  
  Server-client compatibility

- Why does the API server convert objects to storage version before writing to etcd?
  
  Old server - new server compatibility
What’s the storage version for?

RESTful API

POST
/apis/batch/v1beta1/jobs
/apis/batch/v1/jobs
/apis/batch/v2/jobs

v1.15
What's the storage version for?

USER

POST
/apis/batch/v1beta1/jobs
/apis/batch/v1/jobs
/apis/batch/v2/jobs

RESTful API

master

Etcld key: registry/jobs/default/pi

apiVersion: batch/v1beta1
kind: Job
metadata:
  name: pi
spec:
  template:
What's the storage version for?

RESTful API

/user
/apis/batch/v1/jobs
/apis/batch/v2/jobs
/master
/etc

Etcd key: registry/jobs/default/pi

apiVersion: batch/v1beta1
kind: Job
metadata:
  name: pi
spec:
  template:
  ...

Etcd key: registry/jobs/default/pi
What’s the storage version for?

RESTful API

/user
/api/batch/v1/jobs
/api/batch/v2/jobs

/master

/etcd

Etcd key: registry/jobs/default/pi

```
apiVersion: batch/v1
kind: Job
metadata:
  name: pi
spec:
  template:
    ...
```
Storage version changes

RESTful API

/api/batch/v1/jobs
/api/batch/v2/jobs

master

Etcd

Etcld key: registry/jobs/default/pi

apiVersion: batch/v2
kind: Job
metadata:
  name: pi
spec:
  template:
  ...
Storage version changes

RESTful API

/api/batch/v2/jobs

Etcd key: registry/jobs/default/pi

```yaml
apiVersion: batch/v2
kind: Job
metadata:
  name: pi
spec:
  template:
    ...```

v1.18
Objects encoded in storage version => Safe upgrade
Agenda

How Kubernetes stores objects in etcd

Risks of stale objects in etcd

The storage migrator
Etcd key: registry/jobs/default/pi

RESTful API

/user

/apis/batch/v1/jobs
/apis/batch/v1beta1/jobs
/apis/batch/v2alpha1/jobs

/master

/v1.15

/etc

apiVersion: batch/v1
kind: Job
metadata:
  name: pi
spec:
  template:
    ...

Etcd key: registry/jobs/default/pi
Stale Objects

RESTful API

Etcd key: registry/jobs/default/pi

apiVersion: batch/v1
kind: Job
metadata:
  name: pi
spec:
  template:
    ...
Agenda

- How Kubernetes stores objects in etcd
- Risks of stale objects in etcd
- The storage migrator
A control loop that makes sure persisted API objects are encoded in their respective storage versions.
Deployed via kubectl

Migrations API: Kubernetes-style

Resilient to failures

Vendor-agnostic
Deploying Storage Migrator

$ git clone git@github.com:kubernetes-sigs/kube-storage-version-migrator.git

$ cd kube-storage-version-migrator

$ make local-manifests

$ kubectl apply -f manifests.local
Migration API

apiVersion: migration.k8s.io/v1alpha1
kind: StorageVersionMigration
metadata:
  name: jobs.batch-cglnt
  namespace: kube-storage-migration
spec:
  resource:
    Group: batch
    Resource: jobs
    Version: v1
  continueToken: AL043vER
status:
  conditions:
  - lastUpdateTime: "2019-06-13T23:51:54Z"
    status: "True"
    type: Succeeded
apiVersion: migration.k8s.io/v1alpha1
kind: StorageVersionMigration
metadata:
  name: jobs.batch-cglnt
  namespace: kube-storage-migration
spec:
  resource:
    Group: batch
    Resource: jobs
    Version: v1
  continueToken: AL043vER
status:
  conditions:
  - lastUpdateTime: "2019-06-13T23:51:54Z"
    status: "True"
    type: Succeeded
apiVersion: migration.k8s.io/v1alpha1
kind: StorageVersionMigration
metadata:
  name: jobs.batch-cglnt
  namespace: kube-storage-migration
spec:
  resource:
    Group: batch
    Resource: jobs
    Version: v1
  continueToken: AL043vER
status:
  conditions:
  - lastUpdateTime: "2019-06-13T23:51:54Z"
    status: "True"
    type: Succeeded
Wait for migration to complete before upgrading/downgrading API server:

$ kubectl wait --all --for=condition=Succeeded \nStorageversionmigrations.migration.k8s.io \n--namespace=kube-storage-migration
Agenda

How Kubernetes stores objects in etcd

Risks of stale objects in etcd

The storage migrator

- User-facing highlights
- The internals
Controllers

Migration Trigger Controller
- Fetches discovery docs

API Server
- Checks storage version changes

Migrator Controller

Time
Controllers

Migration Trigger Controller:
- Fetches discovery docs
- Checks storage version changes

API Server:
- Server Upgrades

Migrator Controller:
- Fetches discovery docs
- Checks storage version changes

Time
Controllers

Migration Trigger Controller
- Fetches discovery docs
- Checks storage version changes

API Server
- Server Upgrades
  - Fetches discovery docs
  - Checks storage version changes

Migrator Controller
- POST Migrations

Time
Controllers

Migration Trigger Controller
- Fetches discovery docs
- Checks storage version changes

API Server
- Server Upgrades
- Fetches discovery docs
- WATCH Migrations

Migrator Controller
- POST Migrations
Controllers

Migration Trigger Controller
- Fetches discovery docs
- Checks storage version changes

API Server
- POST Migrations
- Watches Migrations

Migrator Controller
- Watches Migrations
- Migrates resources whose storage versions change
Controllers

Migration Trigger Controller
- Fetches discovery docs
- Checks storage version changes

POST Migrations

Time

API Server
- Fetches discovery docs
- Watches Migrations
- Server Upgrades

Migrator Controller
- Watches Migrations
- Migration from the Watch channel
- Migrates resources whose storage versions change
Migrator Controller

Chunking LIST 500 objects

API Server

etcd

Time
Migrator Controller

Chunking LIST 500 objects

GET 1st object

API Server

etcd

Time
Migrator Controller

- Chunking LIST 500 objects
- GET 1st object
- UPDATE 1st object, with no change

API Server

etcd
Migrator Controller

API Server

etcd

Chunks LIST 500 objects

GET 1st object

UPDATE 1st object, with no change

Convert to storage version, write to etcd
Migrator Controller

- Chunking LIST 500 objects
- GET 1st object
- UPDATE 1st object, with no change

Repeat 500 times

API Server

- Convert to storage version, write to etcd

etcd
Migrator Controller

Chunking LIST 500 objects

API Server

GET 1st object

etcd

UPDATE 1st object, with no change

Convert to storage version, write to etcd

Repeat 500 times

Record progress in migration.spec

Time
Migrator Controller

API Server

etcd

Time

Chunking LIST 500 objects

Record progress in migration.spec

Repeat until iterate through all instances of a resource
Migrator Controller

API Server

etcd

Time

Chunking LIST 500 objects

Repeat until iterate through all instances of a resource

Record progress in migration.spec

UPDATE migration.status = “Succeeded”
Timeline

Beta: 2019 Q2

GA: 2019 Q3

https://github.com/kubernetes-sigs/kube-storage-version-migrator
How Kubernetes stores objects in etcd

Risks of stale objects in etcd

The storage migrator
Takeaways

How Kubernetes stores objects in etcd

Risks of stale objects in etcd

The storage migrator
  ● User’s perspective
  ● The internals

Bonus: useful meta APIs used by the storage migrator
apiVersion: v1
kind: PodList
metadata:
  resourceVersion: 10245
  continue: ENCODED_CONTINUE_TOKEN
...
Continue token is returned with 410 error:

```yaml
apiVersion: meta.k8s.io
kind: Status
metadata:
  continue: ENCODED_CONTINUE_TOKEN
code: 410
status: Failure
reason: Expired
...
apiVersion: v1
kind: PodList
metadata:
  continue: ENCODED_CONTINUATE_TOKEN
remainingItemCount: 10245
...
Unfreeze API Removal. Removing O(10k) lines of code.