Automate App Operation

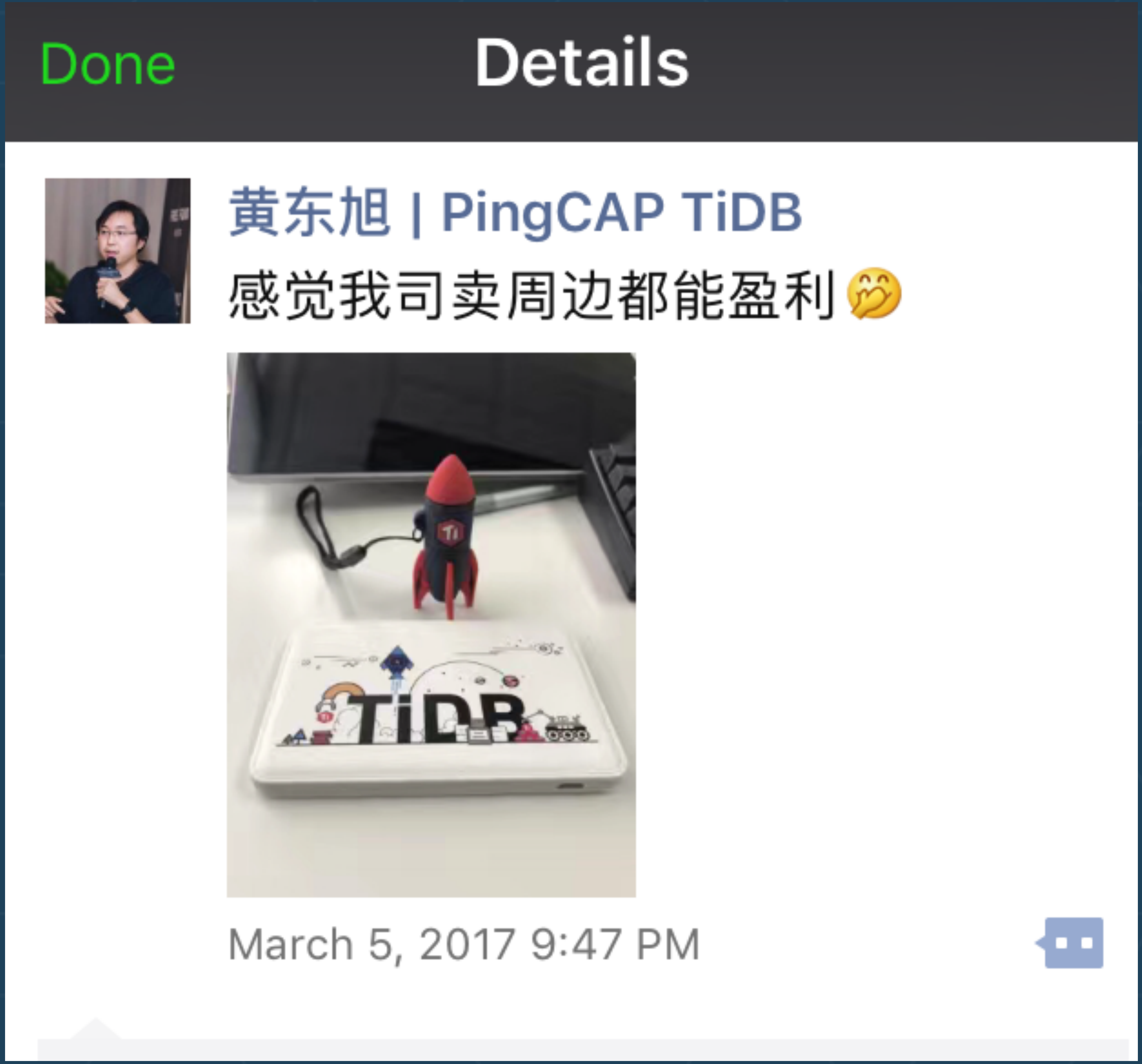
Hongchao Deng

hongchao.deng@coreos.com

App = ?

App = Code + Config

故事开始于...

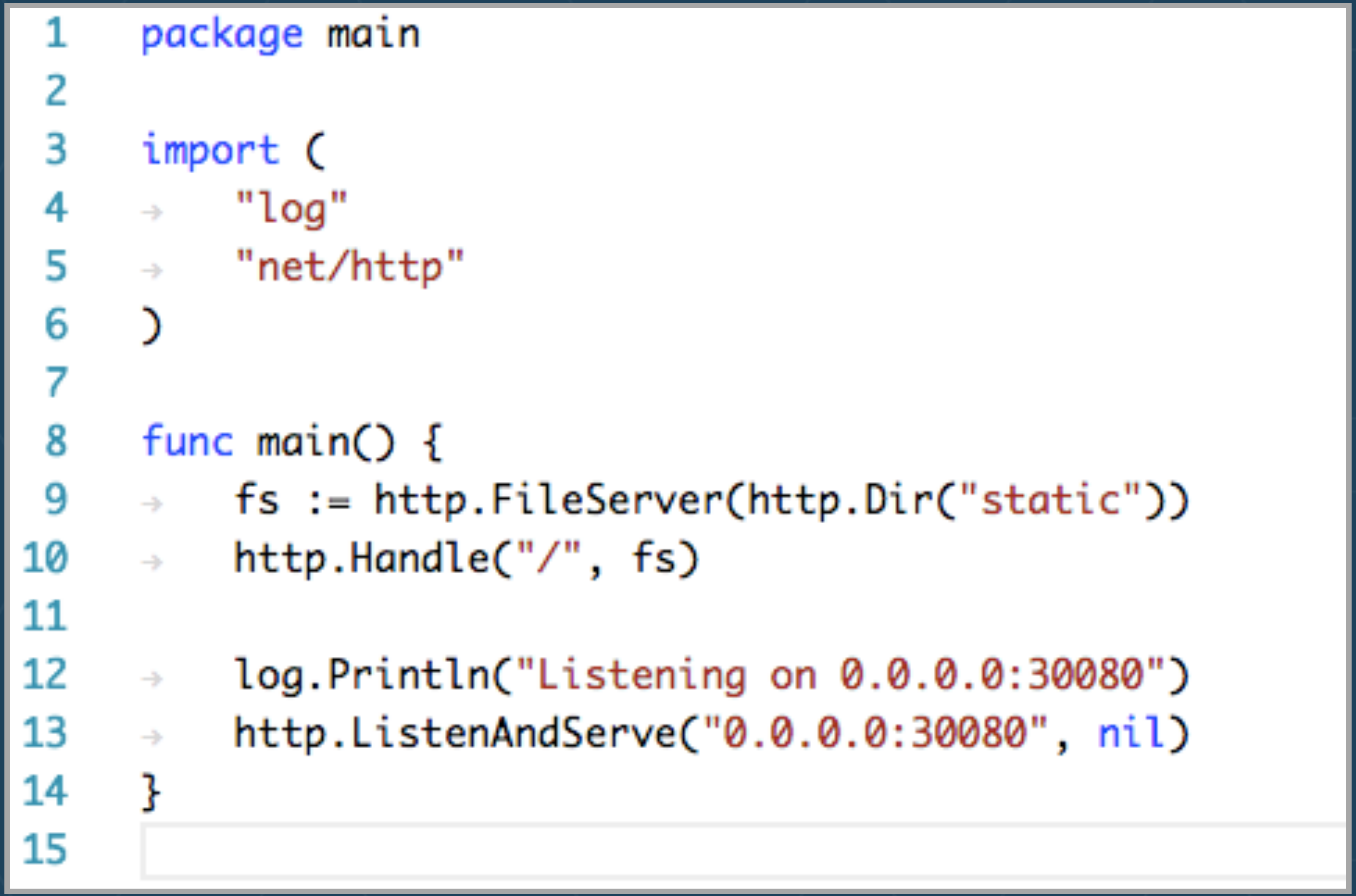


Details

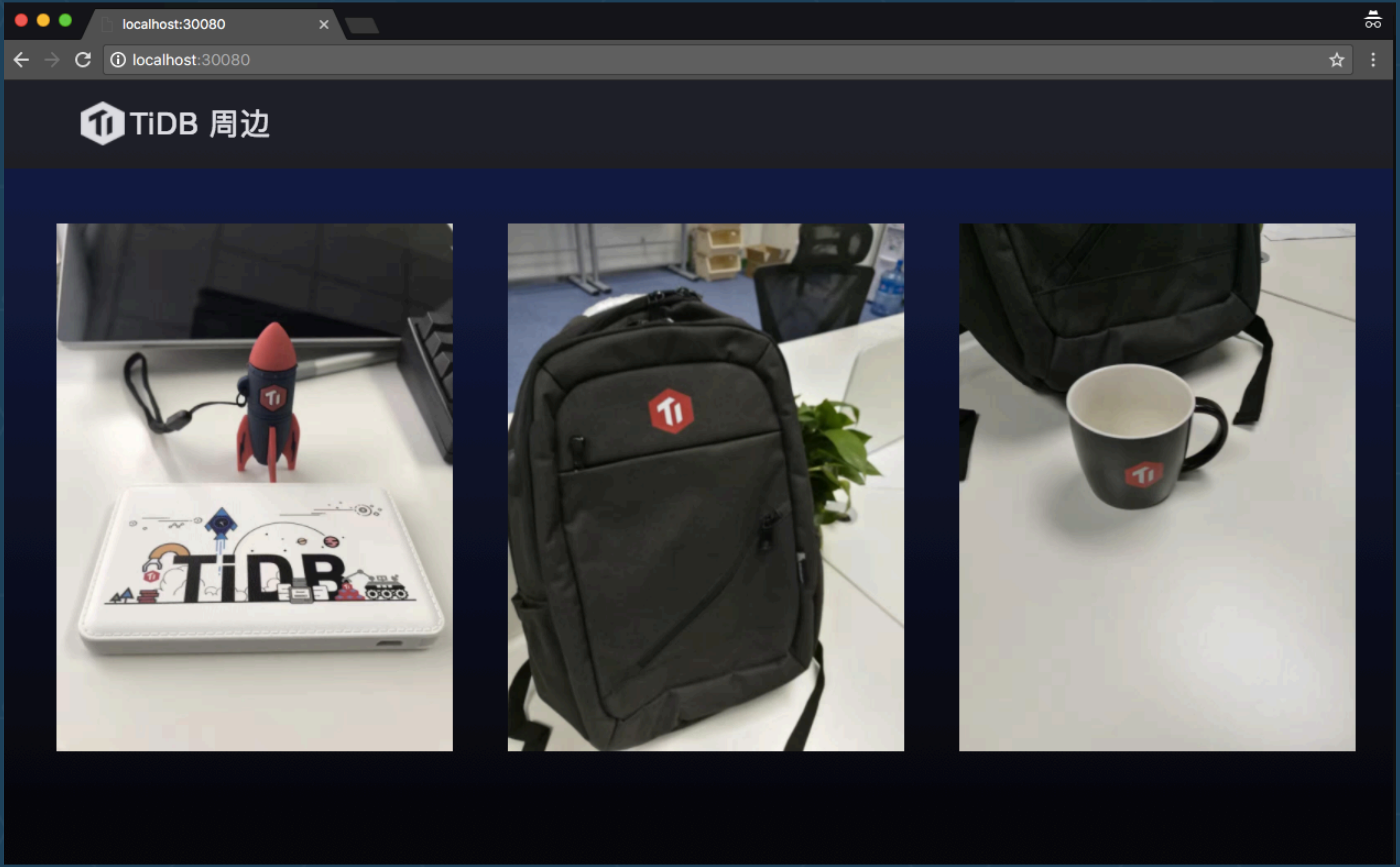
黄东旭 | PingCAP TiDB

感觉我司卖周边都能盈利🤭

March 5, 2017 9:47 PM

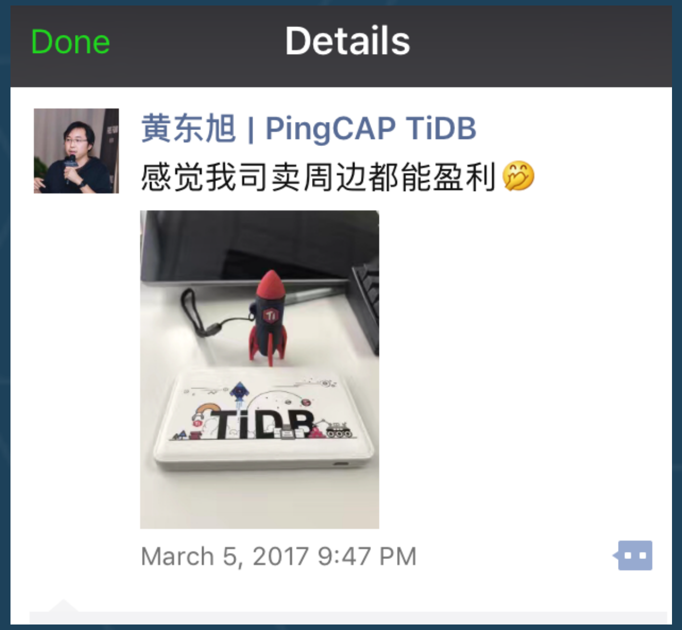


```go
package main

import (
	"log"
	"net/http"
)

func main() {
	fs := http.FileServer(http.Dir("static"))
	http.Handle("/", fs)

	log.Println("Listening on 0.0.0.0:30080")
	http.ListenAndServe("0.0.0.0:30080", nil)
}

```


localhost:30080
localhost:30080
TiDB 周边

Development

Deployment

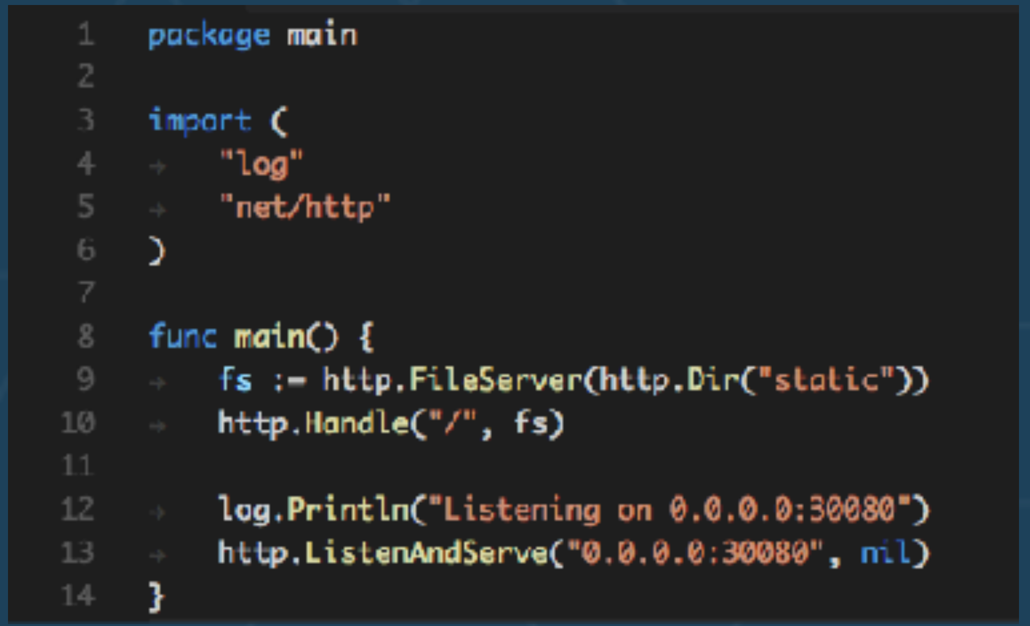


黄东旭 | PingCAP TiDB
感觉我司卖周边都能盈利🤔

March 5, 2017 9:47 PM

想法

实现



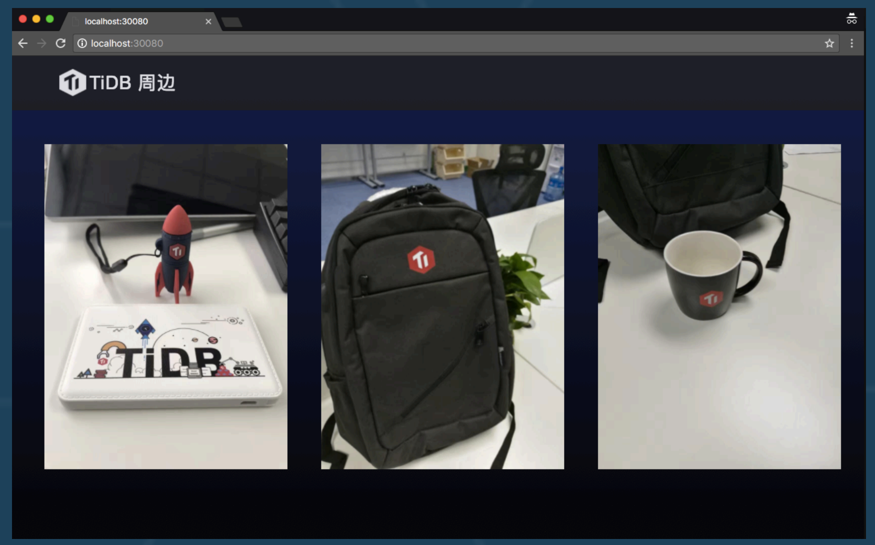
```
package main

import (
	"log"
	"net/http"
)

func main() {
	fs := http.FileServer(http.Dir("static"))
	http.Handle("/", fs)

	log.Println("Listening on 0.0.0.0:30080")
	http.ListenAndServe("0.0.0.0:30080", nil)
}
```

程序
打包
发布

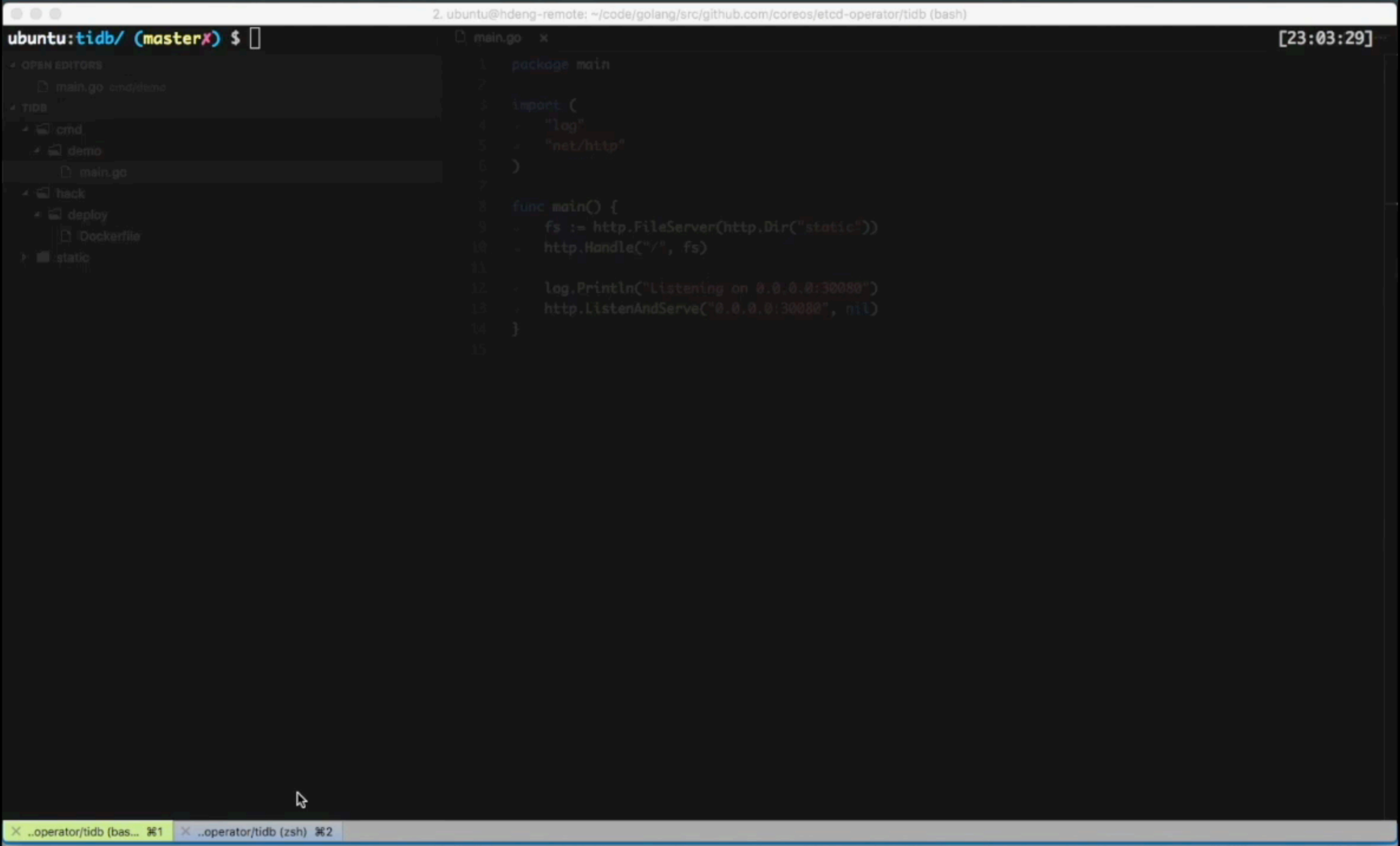
docker build
docker push

DNS

LoadBalancer



Demo



```
2. ubuntu@hdeng-remote: ~/code/golang/src/github.com/coreos/etcd-operator/tidb (bash)
ubuntu:tidb/ (master✗) $                                                [23:03:29]
```

```go
package main

import (
	"log"
	"net/http"
)

func main() {
	fs := http.FileServer(http.Dir("static"))
	http.Handle("/", fs)

	log.Println("Listening on 0.0.0.0:30080")
	http.ListenAndServe("0.0.0.0:30080", nil)
}
```

Deploy App Container

- Docker/OCI
  - Standard app packaging format
- Kubernetes/Swarm
  - Resource scheduling, cluster management

It is easy to deploy stateless apps. But how to deploy stateful apps?

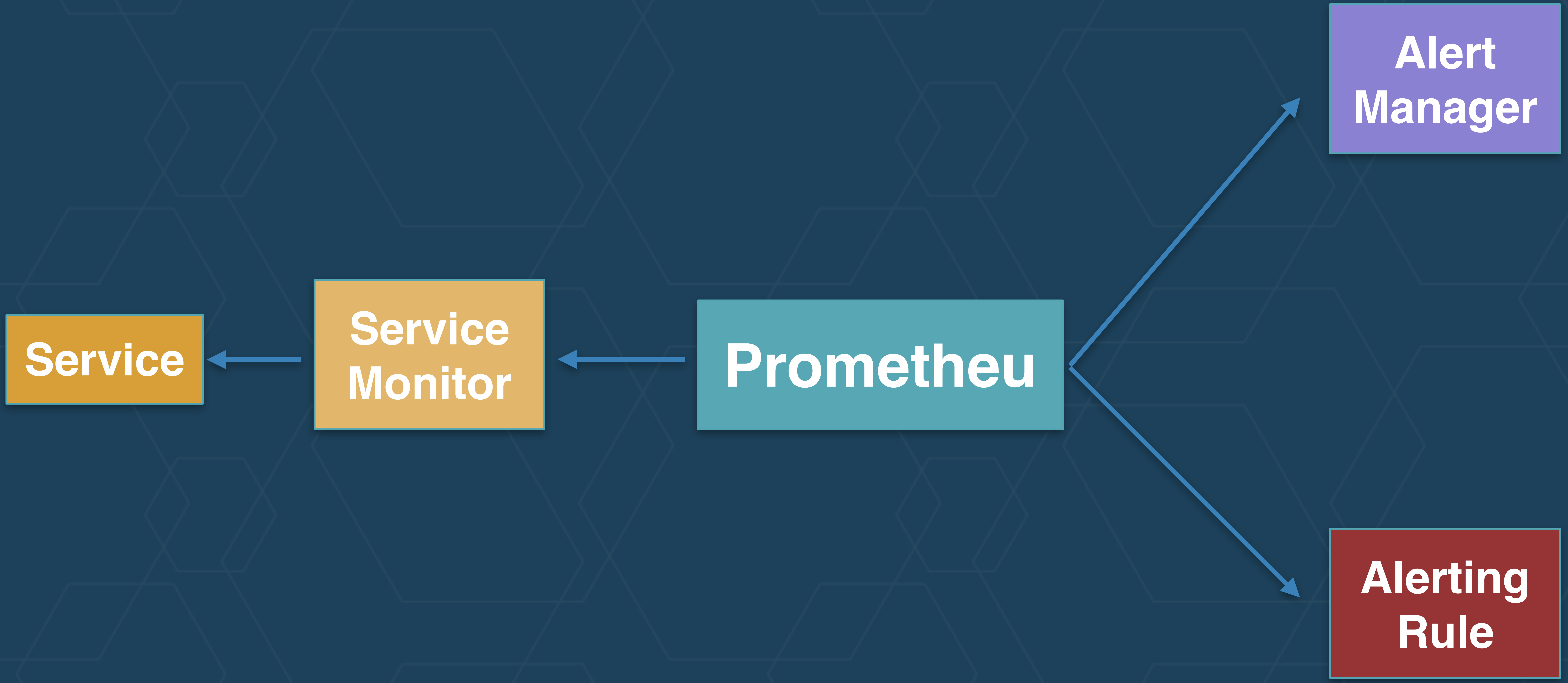
How to Deploy

- Database: PostgreSQL, MySQL, TiDB
- Coordination service: etcd, ZooKeeper
- Streaming: Kafka, Heron
- Big data: Spark, Hadoop
- Storage: Ceph, GlusterFS
- Logging: ElasticSearch
- Monitoring: Prometheus

Deploying those are much harder than stateless web apps

Prometheus

Complex dependencies



etcd

Membership Configuration

```
etcd --name=example-etcd-cluster-0002 ...
--initial-cluster=
example-etcd-cluster-0001=http://example-etcd-cluster-0001.example-etcd-
cluster.default.svc.cluster.local:2380,
example-etcd-cluster-0000=http://example-etcd-cluster-0000.example-etcd-
cluster.default.svc.cluster.local:2380,
example-etcd-cluster-0002=http://example-etcd-cluster-0002.example-etcd-
cluster.default.svc.cluster.local:2380
```

Self-updating Kubernetes

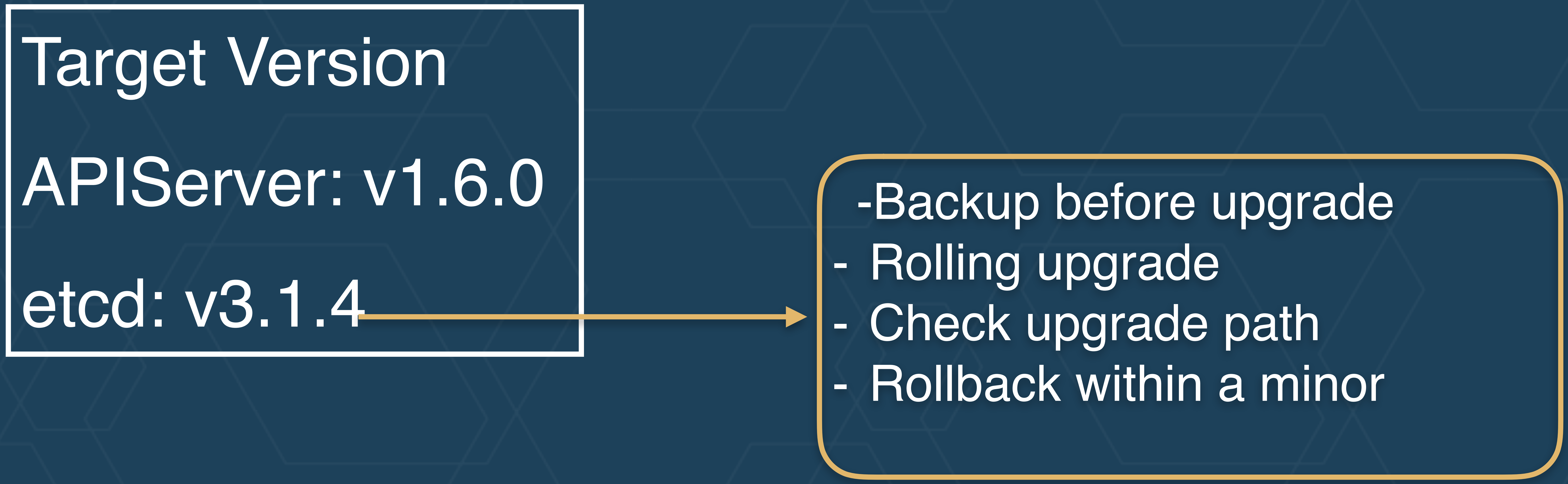
Update Strategy

Target Version

APIServer: v1.6.0

etcd: v3.1.4

Update Strategy



Target Version

APIServer: v1.6.0

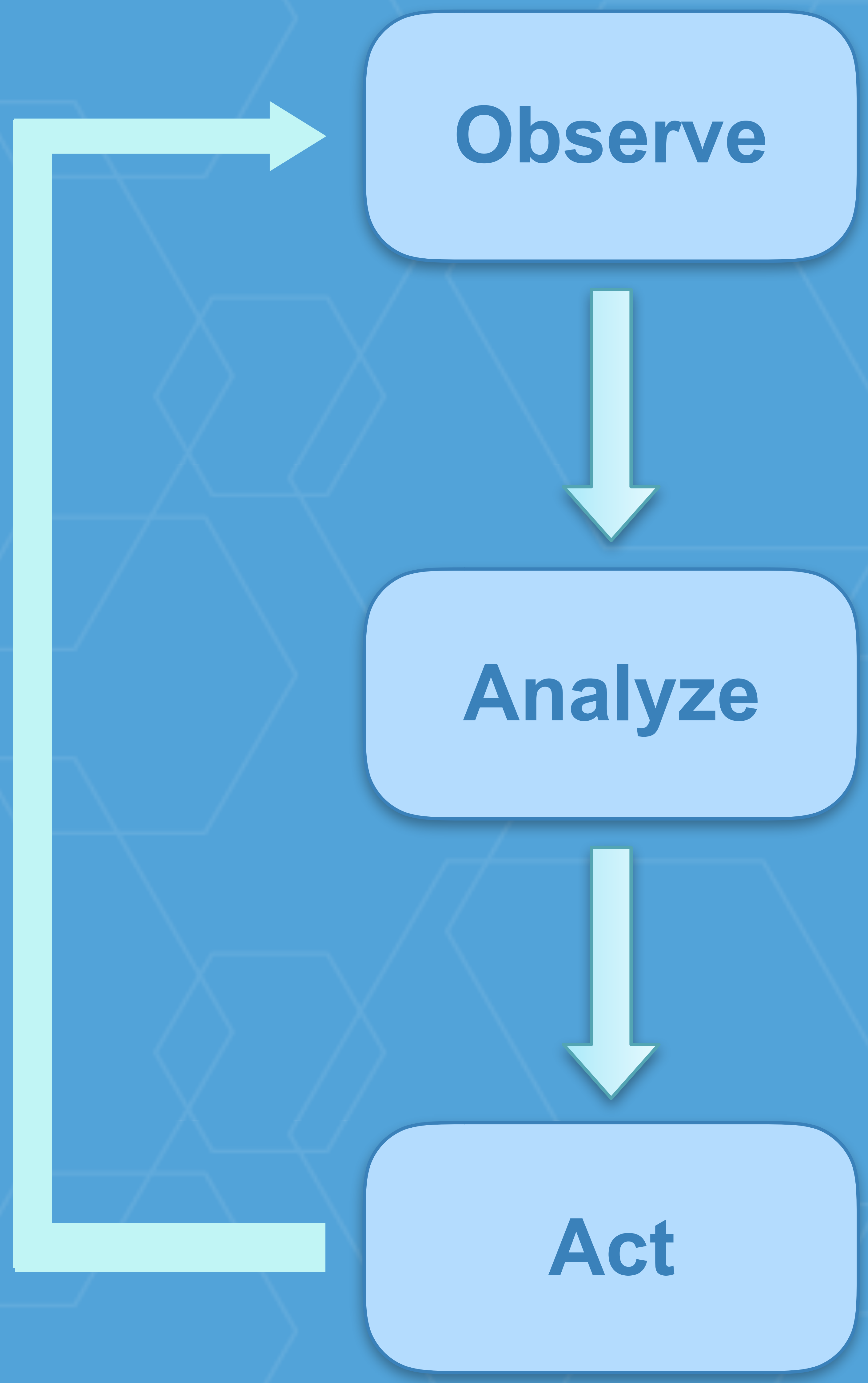
etcd: v3.1.4

-Backup before upgrade
- Rolling upgrade
- Check upgrade path
- Rollback within a minor

What makes them difficult to deploy and manage?

Complex operation logic

Introduce Operator



Current state of a cluster

- cluster size is 3
- cluster version is v1.2.3

Differences from desired config

- cluster size should be 5
- cluster version should be v1.3.0

How to get there

- configure and add members to 5
- upgrade member to v1.3.0 safely one-by-one

etcd Operator

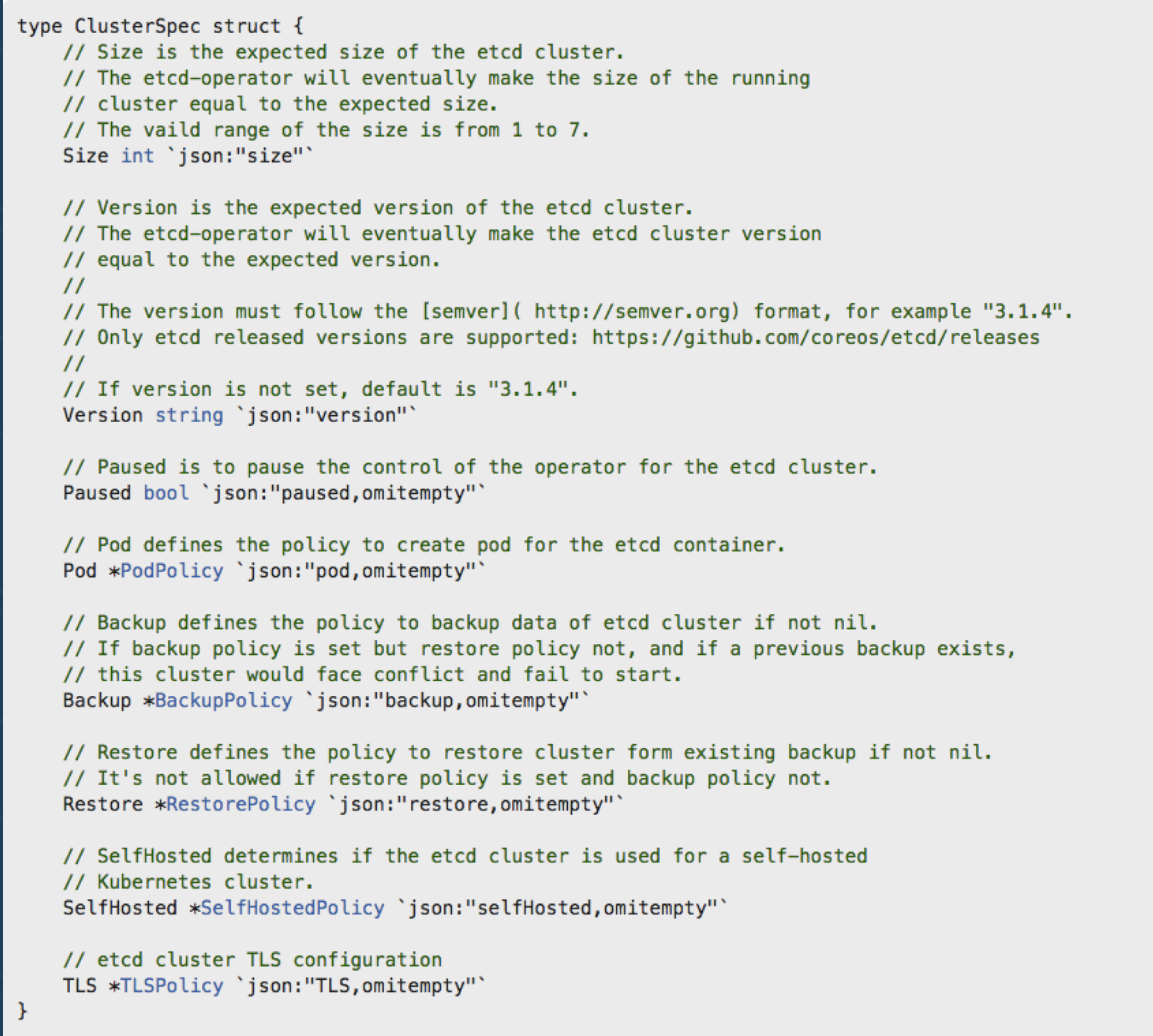
Common Tasks

- Resize
- Upgrade
- Backup
- Failover

Advanced

- Restore
- TLS
- Monitoring/Alerting

Declarative API

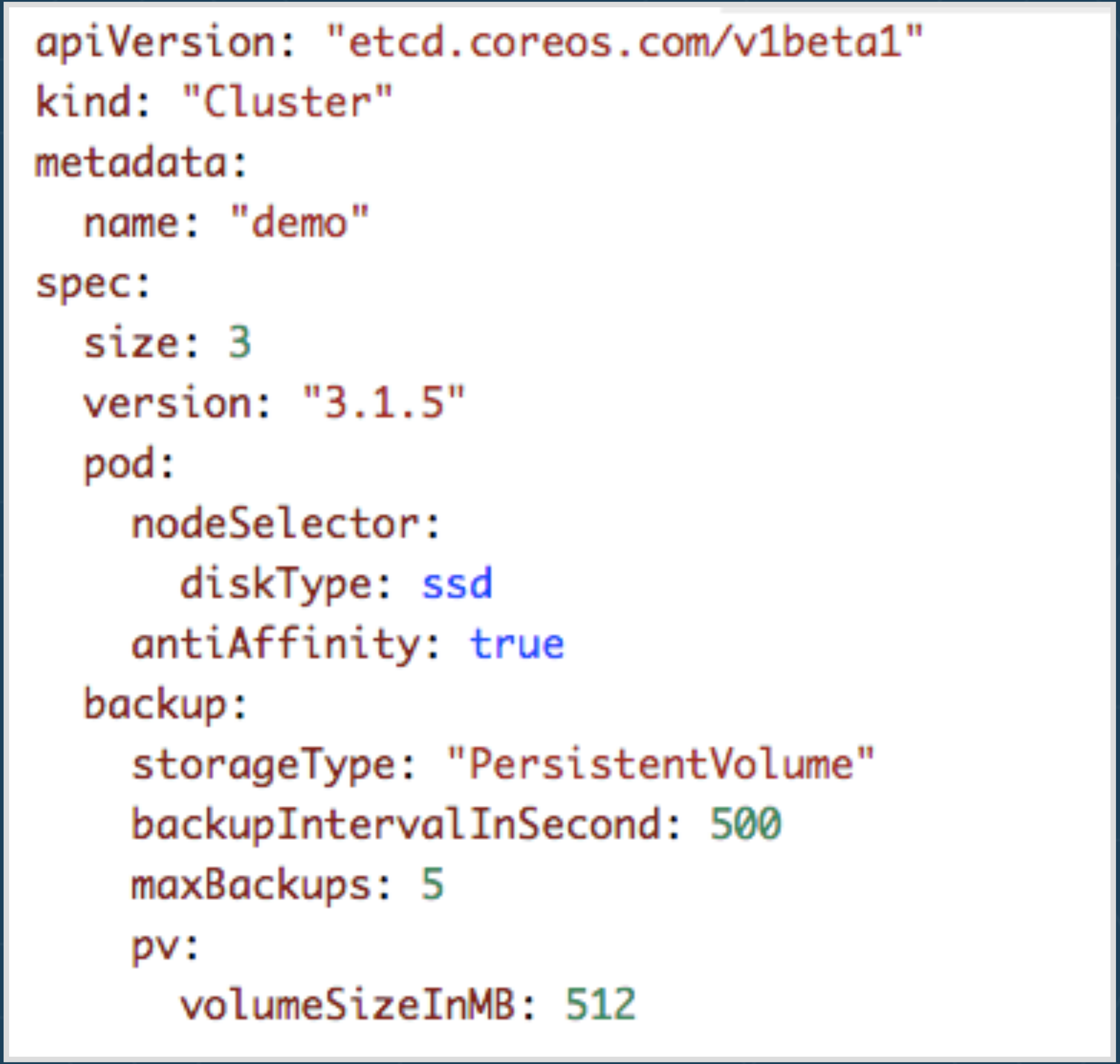


```go
type ClusterSpec struct {
    // Size is the expected size of the etcd cluster.
    // The etcd-operator will eventually make the size of the running
    // cluster equal to the expected size.
    // The vaild range of the size is from 1 to 7.
    Size int `json:"size"`

    // Version is the expected version of the etcd cluster.
    // The etcd-operator will eventually make the etcd cluster version
    // equal to the expected version.
    //
    // The version must follow the [semver]( http://semver.org) format, for example "3.1.4".
    // Only etcd released versions are supported: https://github.com/coreos/etcd/releases
    //
    // If version is not set, default is "3.1.4".
    Version string `json:"version"`

    // Paused is to pause the control of the operator for the etcd cluster.
    Paused bool `json:"paused,omitempty"`

    // Pod defines the policy to create pod for the etcd container.
    Pod *PodPolicy `json:"pod,omitempty"`

    // Backup defines the policy to backup data of etcd cluster if not nil.
    // If backup policy is set but restore policy not, and if a previous backup exists,
    // this cluster would face conflict and fail to start.
    Backup *BackupPolicy `json:"backup,omitempty"`

    // Restore defines the policy to restore cluster form existing backup if not nil.
    // It's not allowed if restore policy is set and backup policy not.
    Restore *RestorePolicy `json:"restore,omitempty"`

    // SelfHosted determines if the etcd cluster is used for a self-hosted
    // Kubernetes cluster.
    SelfHosted *SelfHostedPolicy `json:"selfHosted,omitempty"`

    // etcd cluster TLS configuration
    TLS *TLSPolicy `json:"TLS,omitempty"`
}
```

Create a cluster

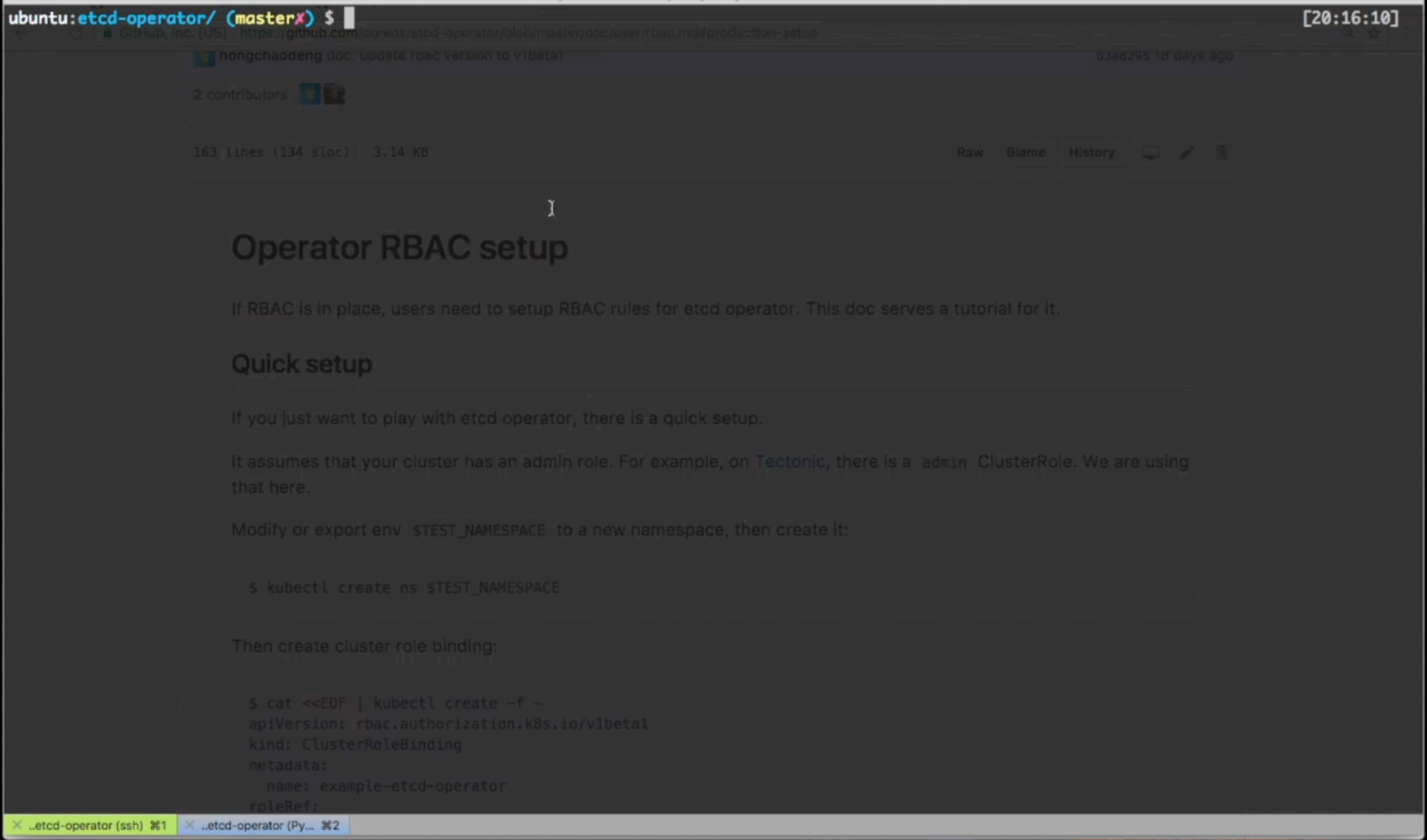
```
&spec.Cluster{
    Metadata: v1.ObjectMeta{
        Name: "demo",
    },
    Spec: spec.ClusterSpec{
        Size:    3,
        Version: "3.1.5",
        Pod: &spec.PodPolicy{
            NodeSelector: map[string]string{
                "diskType": "ssd",
            },
            AntiAffinity: true,
        },
        Backup: &spec.BackupPolicy{
            StorageType:              "PersistentVolume",
            BackupIntervalInSecond: 300,
            MaxBackups:              5,
            StorageSource: spec.StorageSource{
                PV: &spec.PVSource{
                    VolumeSizeInMB: 512,
                },
            },
        },
    },
}
```

kubectl create -f



```yaml
apiVersion: "etcd.coreos.com/v1beta1"
kind: "Cluster"
metadata:
  name: "demo"
spec:
  size: 3
  version: "3.1.5"
  pod:
    nodeSelector:
      diskType: ssd
    antiAffinity: true
  backup:
    storageType: "PersistentVolume"
    backupIntervalInSecond: 500
    maxBackups: 5
    pv:
      volumeSizeInMB: 512
```

Demo



```
ubuntu:etcd-operator/ (master✗) $                                   [20:16:10]
```

GitHub, Inc. (US) https://github.com/coreos/etcd-operator/blob/master/doc/user/rbac.md#production-setup

hongchaodeng doc: update rbac version to v1beta1 b3ad295 18 days ago

2 contributors

163 lines (134 sloc) 3.14 KB Raw Blame History

Operator RBAC setup

If RBAC is in place, users need to setup RBAC rules for etcd operator. This doc serves a tutorial for it.

Quick setup

If you just want to play with etcd operator, there is a quick setup.

It assumes that your cluster has an admin role. For example, on Tectonic, there is a `admin` ClusterRole. We are using that here.

Modify or export env `$TEST_NAMESPACE` to a new namespace, then create it:

```
$ kubectl create ns $TEST_NAMESPACE
```

Then create cluster role binding:

```
$ cat <<EOF | kubectl create -f -
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
  name: example-etcd-operator
roleRef:
```

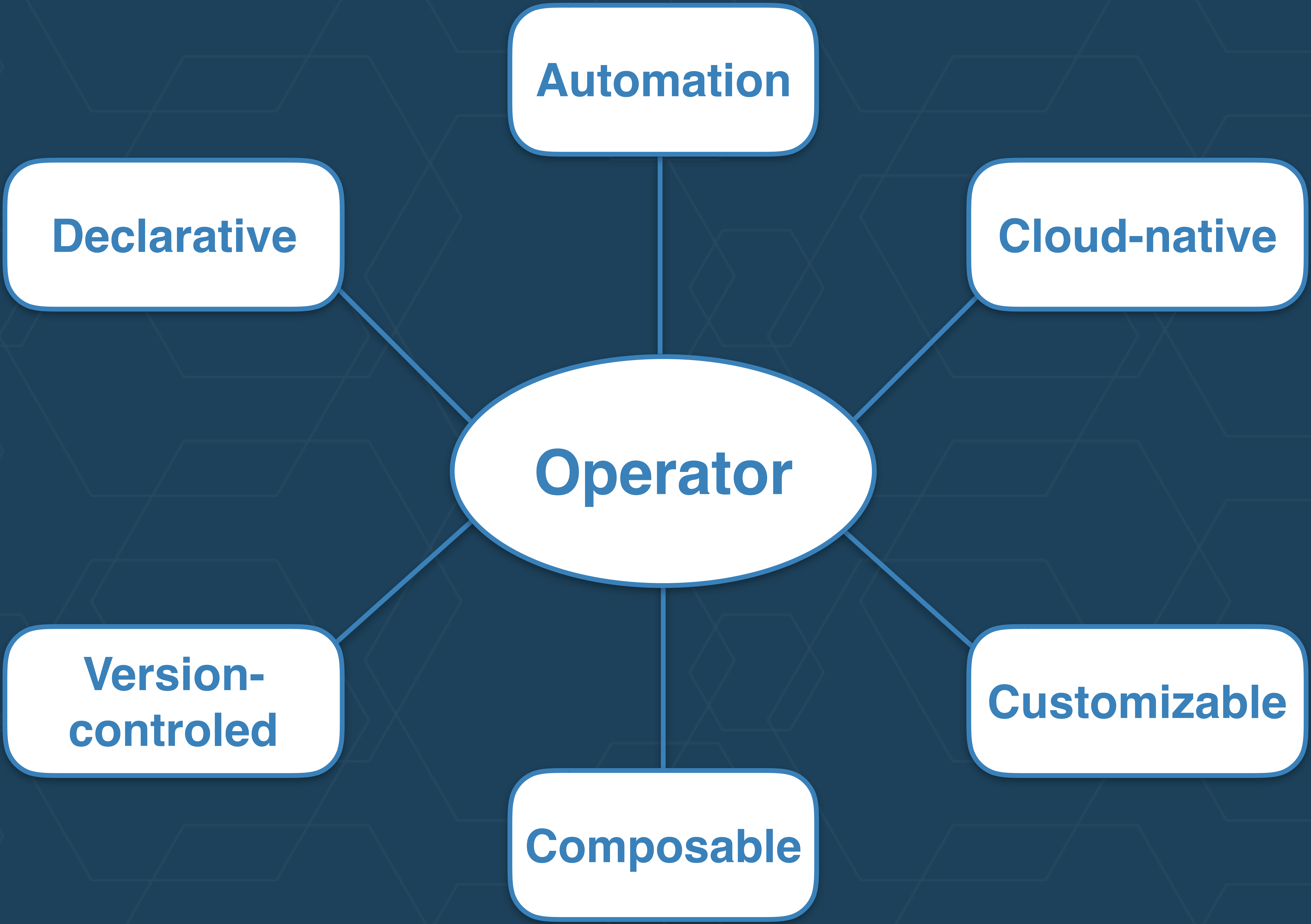
..etcd-operator (ssh) ⌘1 ..etcd-operator (Py... ⌘2

Deploy App Container

- Docker/OCI
  - Standard app packaging format
- Kubernetes/Swarm
  - Resource scheduling, cluster management

Deploy App Container

- Docker/OCI
  - Standard app packaging format
- Kubernetes/Swarm
  - Resource scheduling, cluster management
- Operator
  - App specific operation automation


Automation
Declarative
Cloud-native
Operator
Version-controled
Customizable
Composable

Thanks!

Special thanks to:
PingCAP 和黄东旭，对 slide 相关内容允许
胡宽敏，提供 demo 网站

Hongchao Deng
Github: @hongchaodeng
Email: hongchao.deng@coreos.com