TiDB on Kubernetes

Presented by Deng Shuan
Agenda

- TiDB introduction
  - TiDB architecture
  - TiDB ecosystem

- Why combine TiDB & Kubernetes
  - Cloud vendor agnostic
  - Automation

- How we make it possible
  - TiDB Operator architecture & features
  - How we manage state
  - How we schedule stateful app
Part I - Intro to TiDB
TiDB ecosystem

Monitor/Logging

Core

PD

TiKV

TiDB

Data migration

Lightning

Syncer

Loader

Binlog

...
Part II - Why TiDB on Kubernetes
Cloud-Native applications

- Microservice architecture
- Easy deployment on any cloud
- Elastic scaling
- Highly available
- Automatic operation
Kubernetes: standard platform

De-facto Container Orchestration System (Google Sponsored)

Distributed, cloud provider agnostic OS

- CPU, Memory, Storage and other Devices across all nodes
- Container <=> Process
- Docker image <=> Executable artifacts
- Deployment, StatefulSet <=> Systemd/Supervisor …
- Helm / Charts <=> apt yum / deb rpm
Kubernetes: powerful extensibility

- Standard interface: CNI, CRI, CSI
- Scheduler: scheduler extender
- Controller: CRD
- API Server: Aggregated API Server
- Kubelet: virtual kubelet
- Cloud Provider: LoadBalancer, PersistentVolume
- ...
Part III - How we make it possible
TiDB Operator

https://github.com/pingcap/tidb-operator
Features

- Manage multiple TiDB clusters
- Safely scale the TiDB cluster
- Easily installed with Helm charts
- Network/Local PV support
- Automatically monitoring the TiDB cluster
- Seamlessly perform rolling updates to the TiDB cluster
- Automatic failover
- TiDB related tools integration
Architecture
How we manage state

Kubernetes builtin controllers

Deployment:
- Start ✓
- Scale ✓
- Upgrade ✓
- Failover ✓

StatefulSet:
- Start ✓
- Scale ✓
- Upgrade ✓
- Failover ✗
Custom controller

Domain operation logic

● ThirdPartyResource (TPR), CustomResourceDefinition (CRD):
  ○ Simple & easy
  ○ Lack schema & versionning (added in newer version)

● Aggregated API Server (AA):
  ○ Powerful but complicated
  ○ Coupled with the built-in API Server, hard to deploy
Custom controller

Spec:
  component:
    image:
    replicas:
...

Status:
  image
  replicas
  state

Sync
Custom controller

type Manager interface {
    Sync(*TidbCluster) error
}

```yaml
apiVersion: pingcap.com/v1alpha1
kind: TidbCluster
metadata:
  name: demo
spec:
  pd:
    image: pingcap/pd:v2.1.0
    replicas: 3
    requests:
      cpu: "4"
      memory: "8Gi"
  tikv:
    image: pingcap/tikv:v2.1.0

status:
  tikv:
    stores:
      "5":
        podName: demo-tikv-2
        state: Up
```
Custom controller

StatefulSet with Local PV:
- Maintain node B
- Failover: node B down
How we schedule stateful app

- Schedule based on scheduling history
How we schedule stateful app

- Schedule consider *virtual* resource for local volume
Thank You!