Making an ‘aaS’ out of TiDB

Building a DBaaS on Kubernetes

PingCAP TiDB

TiDB Community Slack Channel
https://pingcap.com/tidbslack/

About Me

- Greg Weber
- Cloud Engineer @ PingCAP, focusing on our cloud initiative, leading our cloud DBaaS development
- Experienced
  - With cloud infrastructure
  - Using DBaaS
  - Building user facing products and backend infrastructure

Agenda

Goal: understand operating a distributed database on Kubernetes

- DBaaS concepts (5 minutes, high-level overview, move quickly)
- TiDB Kubernetes Operator (10 minutes, more depth)
- Making a DBaaS (5 minutes some tech depth here)
- Demo (5 minutes)

Assumptions

You are interested in Kubernetes and high availability databases

PingCAP

Part 1- DBaaS from principles

DBaaS?

DBaaS: DataBase as a Service

Goal: spend less time managing databases by leveraging automation and making expert knowledge operational.

Cloud? Often assumed, but a DBaaS can also be ran on-prem

Market Forces: Cloud Providers

DBaaS is increasingly popular. By taking more responsibility for the operations, it can capture more revenue than a traditional support model.

Cloud hosting providers offer open and proprietary databases.

- open source (MySQL, PostgreSQL, etc)
  - AWS Aurora forks MySQL for more scaleable storage
  - AWS offers a Mongo-compatible document DB
- Proprietary (DynamoDB, Azure Cosmos, GCP CloudStore, FireStore, Spanner)
  - Sometimes these use standard interfaces (AWS Neptune Graph DB)

Market Forces: Open Source Database vendors

Cloud providers will eat your workload, take all the profit, and give nothing back

- Problem: latency advantage: the application is already running on AWS
  - Solution: deploy to the user's cloud
- Problem: Its a new bill
  - Solution: bill through the cloud provider (possible at least on GCP)
- Problem: need an advantage over the cloud provider
  - Solution: Provide a much better service
  - Solution: Trademark protection (RedHat vs. CentOS)
  - Solution: Copyright
    - Open Core: provide features the cloud vendor does not (Redis)
    - Special restrictions against cloud providers (MongoDB)

Operating a database

Perhaps your database should already be operating itself for you?

TiDB was built to remove many of the problems of DBAs managing data at scale.

- Instead of traditional sharding, data is transparently split into regions
  - Regions are automatically replicated 3 ways
  - you can still do transactions across them
  - Multiple hot spot regions are automatically re-distributed
- Scaling is simple: just add another node

Infrastructure and Kubernetes

Our database must interact with infrastructure that varies greatly. Solution:

- Kubernetes == infrastructure compatibility layer
- Operator == automated operations on abstract infrastructure

Theory vs. Practice

- Install and operate Kubernetes
- Provision capacity to your K8s cluster for the database
- Kubernetes is a runtime, but some data is archival (backups, auditing)
- Local SSD is difficult to get working properly

DBaaS makes the practice look like the theory

Not just one, but many Databases

- Testing: quickly create and destroy databases
  - Easy in the cloud, harder for on prem
- Isolation: create multiple clusters
- Location: deploy to different datacenters

DBaaS should make this simple and help you know what is going on at all times

UI

- Provide a GUI
- Provide an API and a CLI for custom user automation

Serverless Future

What is Serverless?

- A cost revolution: pay only for what you use instead of idle machines
- Automatically Scale-out/in to match demand

BigQuery (OLAP)

- charges for queries by the number of bytes processed
- charge for data stored and data inserted
- But they still offer a flat-rate pricing for customers that prefer that!

PingCAP

Part 2 - Building an Operator on Kubernetes

Kubernetes and DB operators

MySQL on Kubernetes: StatefulSet + cloud disk?

Problems

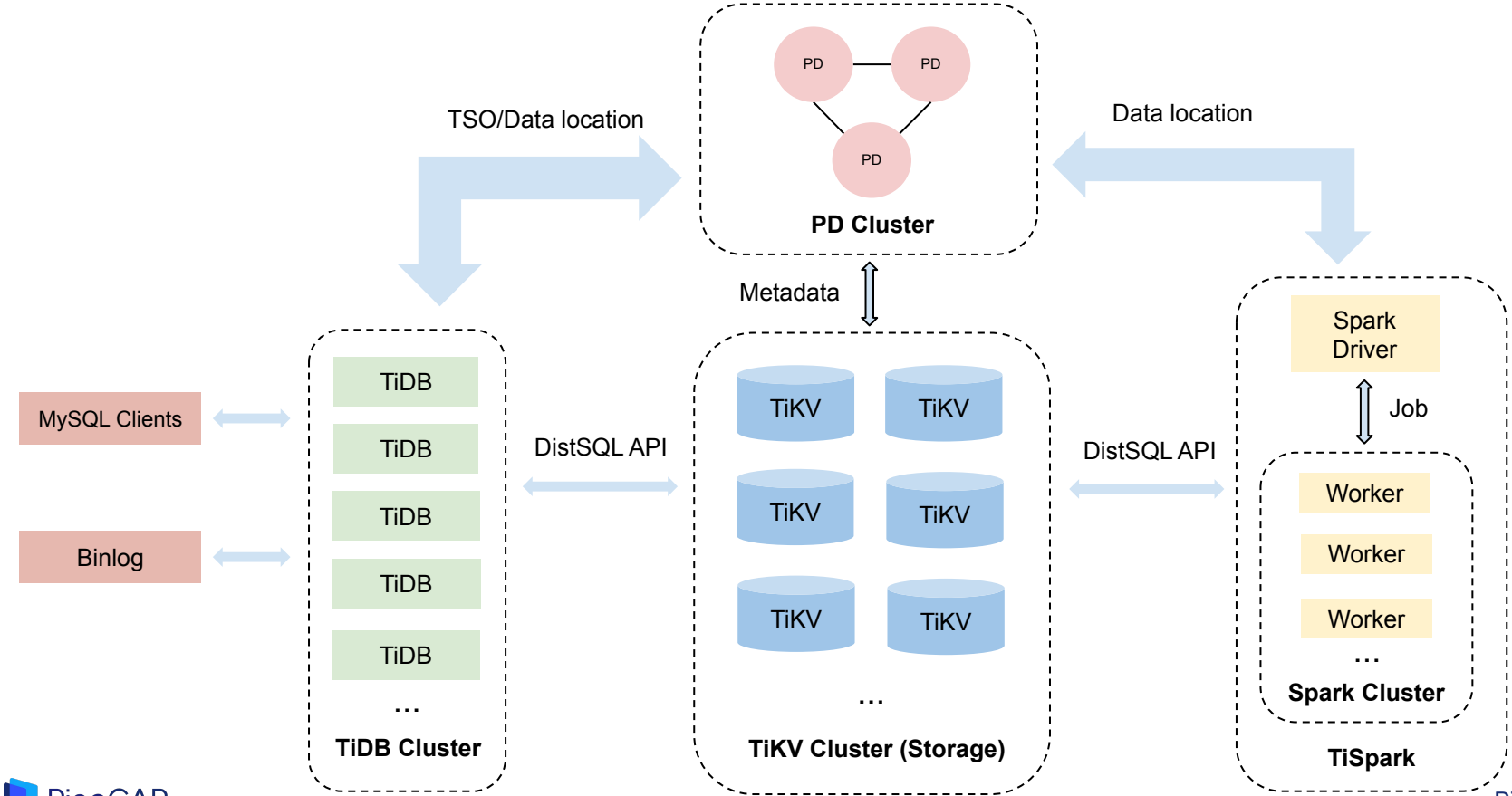
- cloud disk is slow/expensive compared to local SSD
- How much downtime during failover scenarios?

TiDB is a distributed system

- No downtime during many failure scenarios
- Deployment is more complex

We need an operator: code that monitors our database deployment.

TiDB architecture

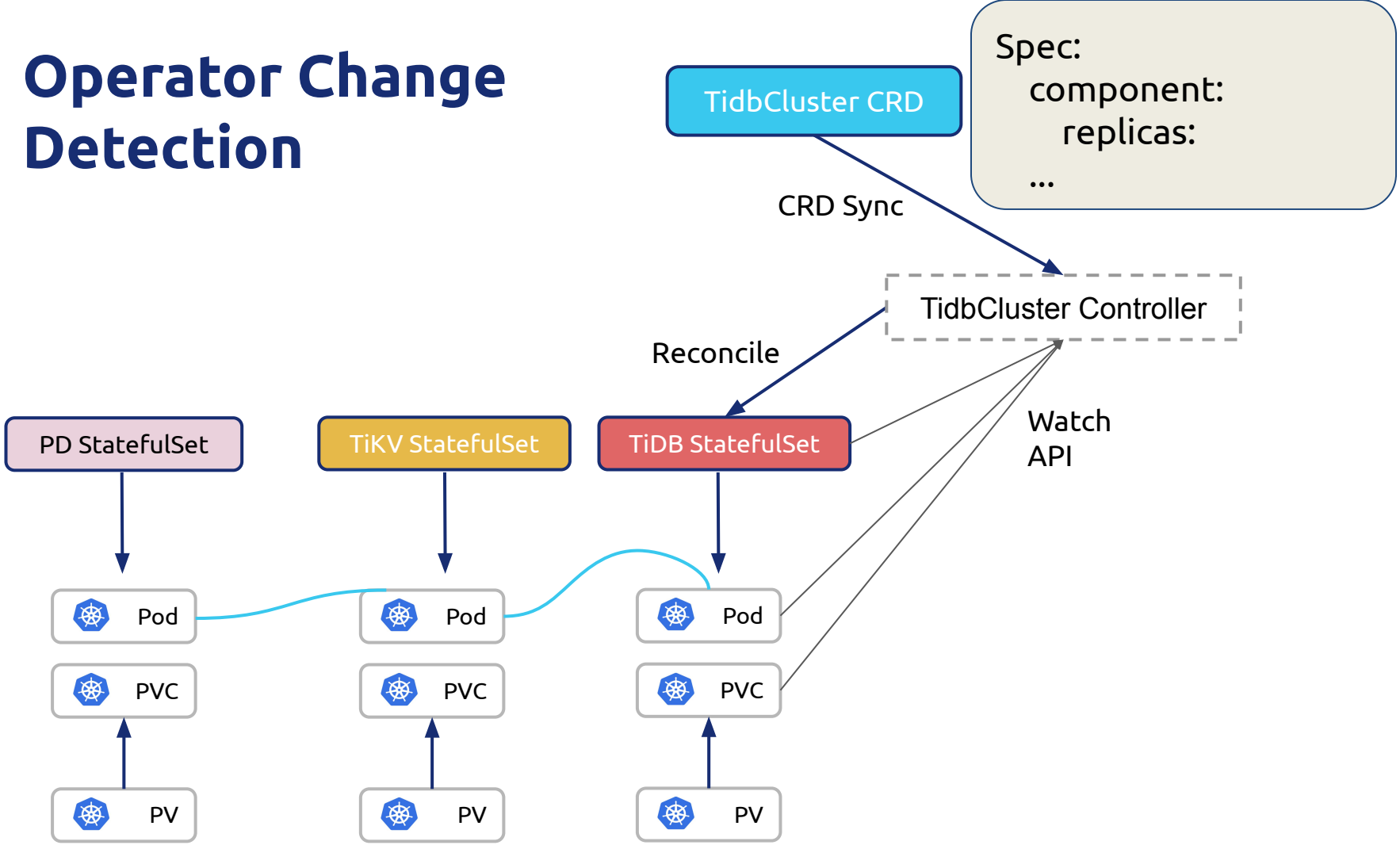


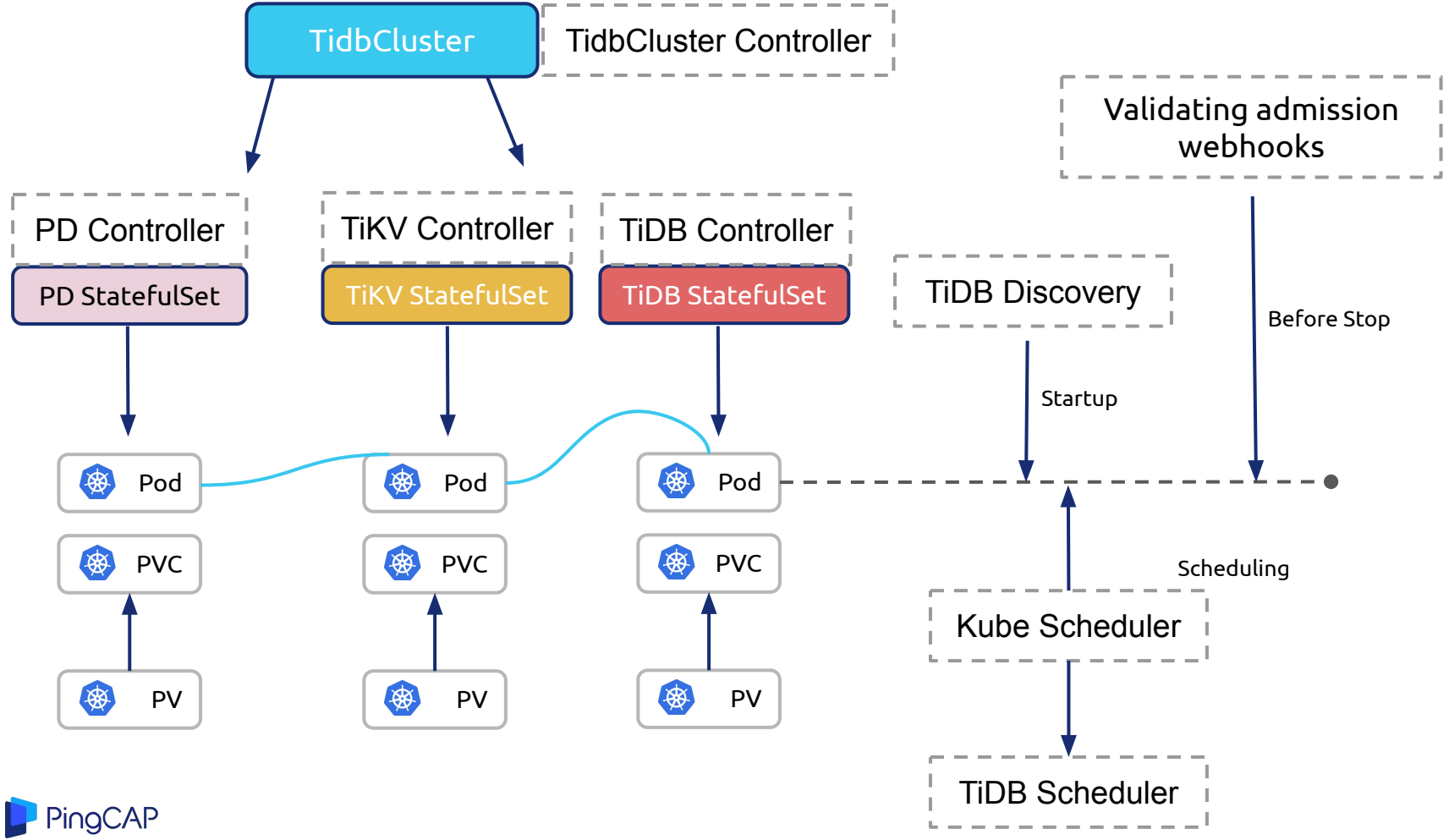
TiDB Operator challenges

PingCAP

- Manage multiple different clusters
- Bootstrap
- High availability scheduling
- Local SSD, not cloud disk. Network/Local PV support
    - Data is already replicated by TiDB
- Automatic monitoring of the TiDB cluster
    - Failover: detect failure and create new pods to maintain availability
- User operations
    - Scale-out
    - rolling upgrade
- Integration with TiDB related tools (backup, binlog, etc)
- Easy of use: configure and install with Helm charts

Operator Change Detection

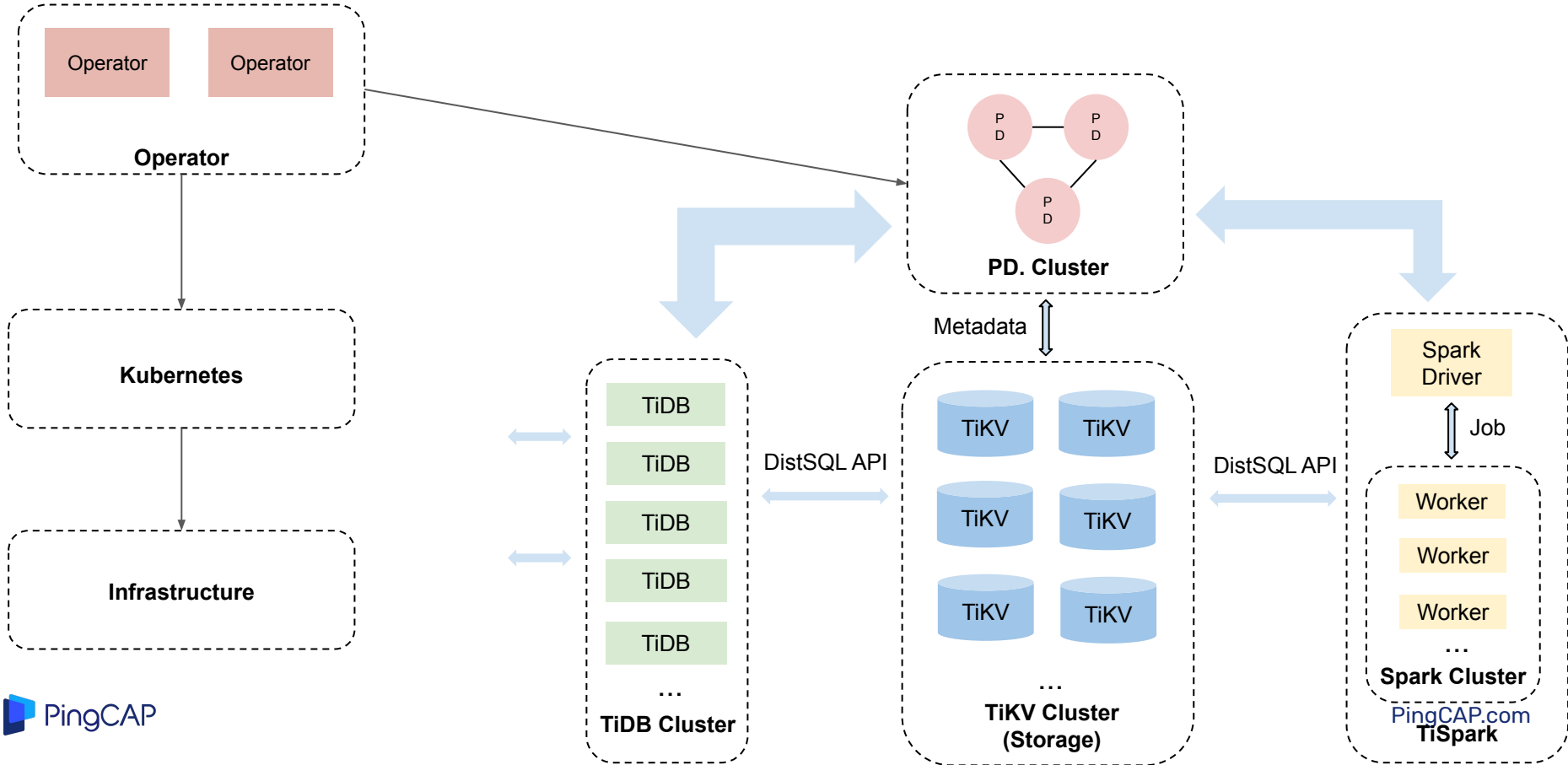



TidbCluster
TidbCluster Controller
Validating admission webhooks
PD Controller
TiKV Controller
TiDB Controller
PD StatefulSet
TiKV StatefulSet
TiDB StatefulSet
TiDB Discovery
Before Stop
Startup
Pod
Pod
Pod
PVC
PVC
PVC
PV
PV
PV
Scheduling
Kube Scheduler
TiDB Scheduler

TiDB Operation

- Cluster bootstrap: Initial PD -> New PD joins existing cluster
- Safely delete PD
  - remove member using PD API
  - stop pd-server
- Safely delete TiKV
  - offline store using PD API
  - stop tikv-server
- Graceful upgrade
  - PD: transfer Raft leader
  - TiKV: evict Raft leaders
  - TiDB: evict DDL owner

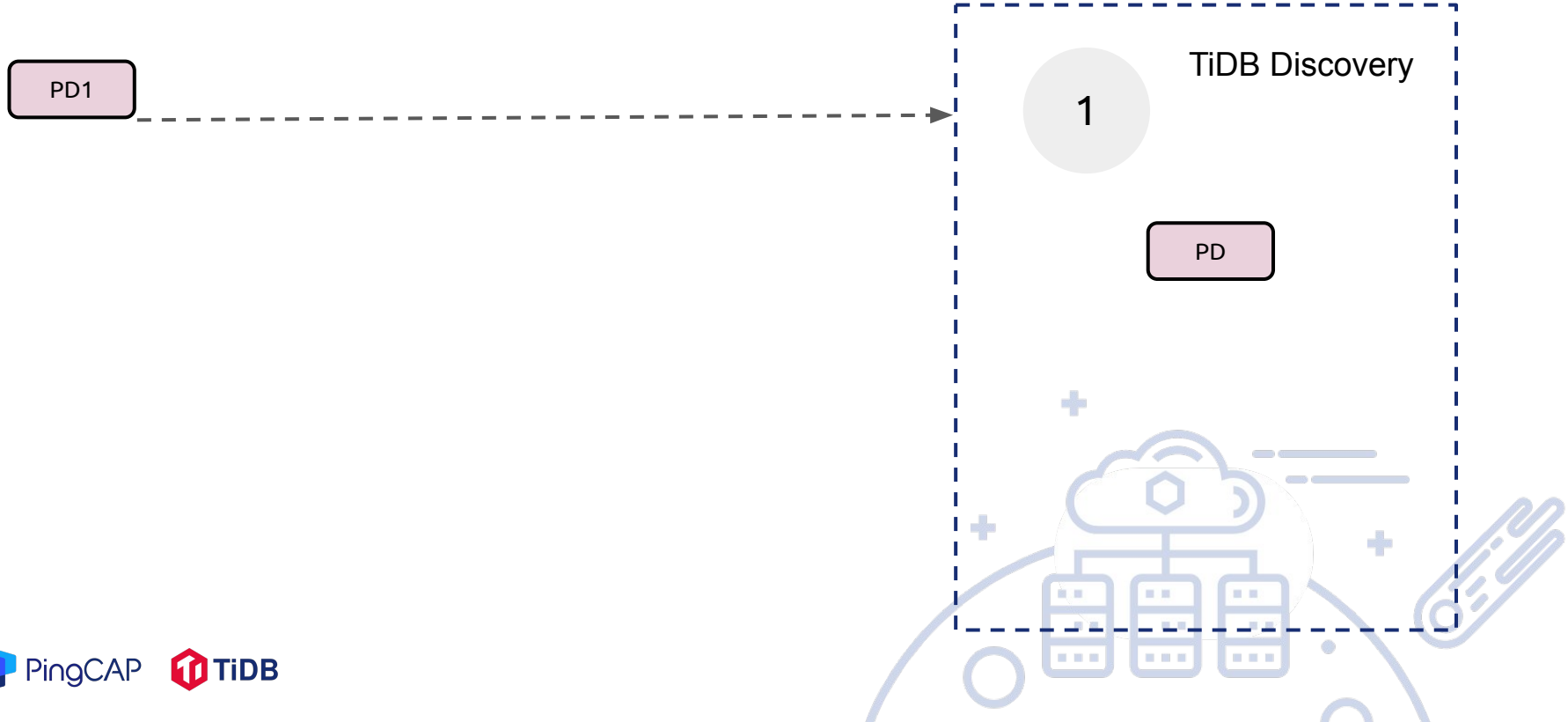
TiDB Operator



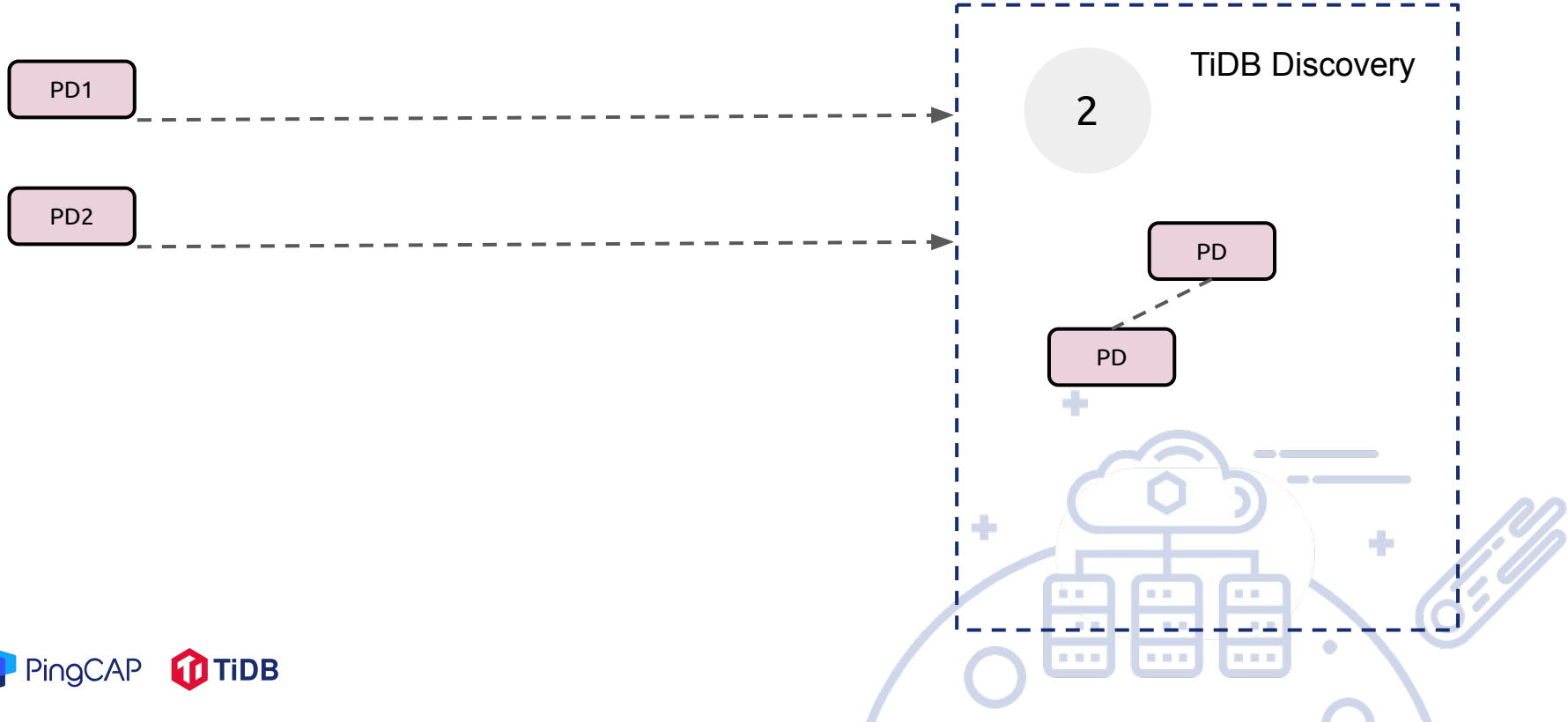
PingCAP

PingCAP.com

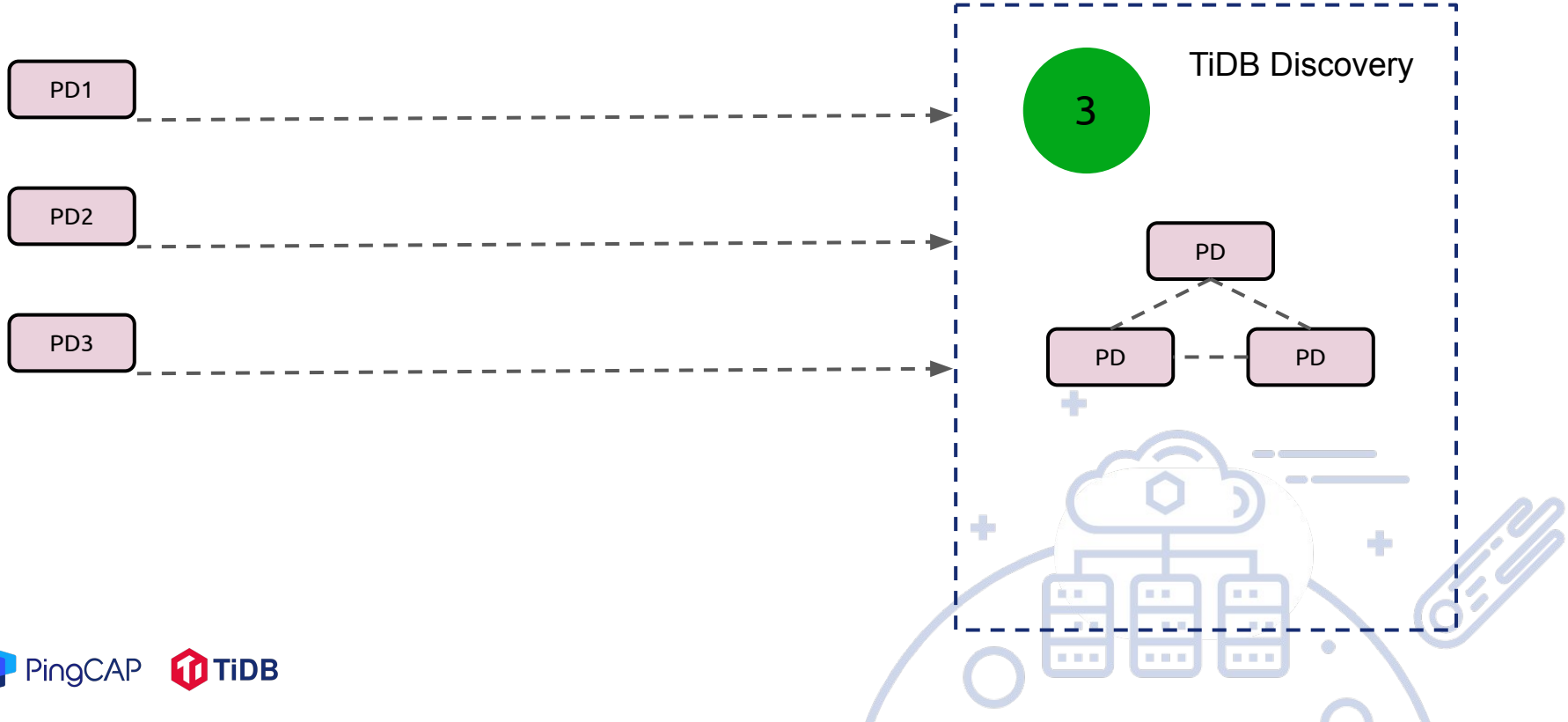
PD Startup



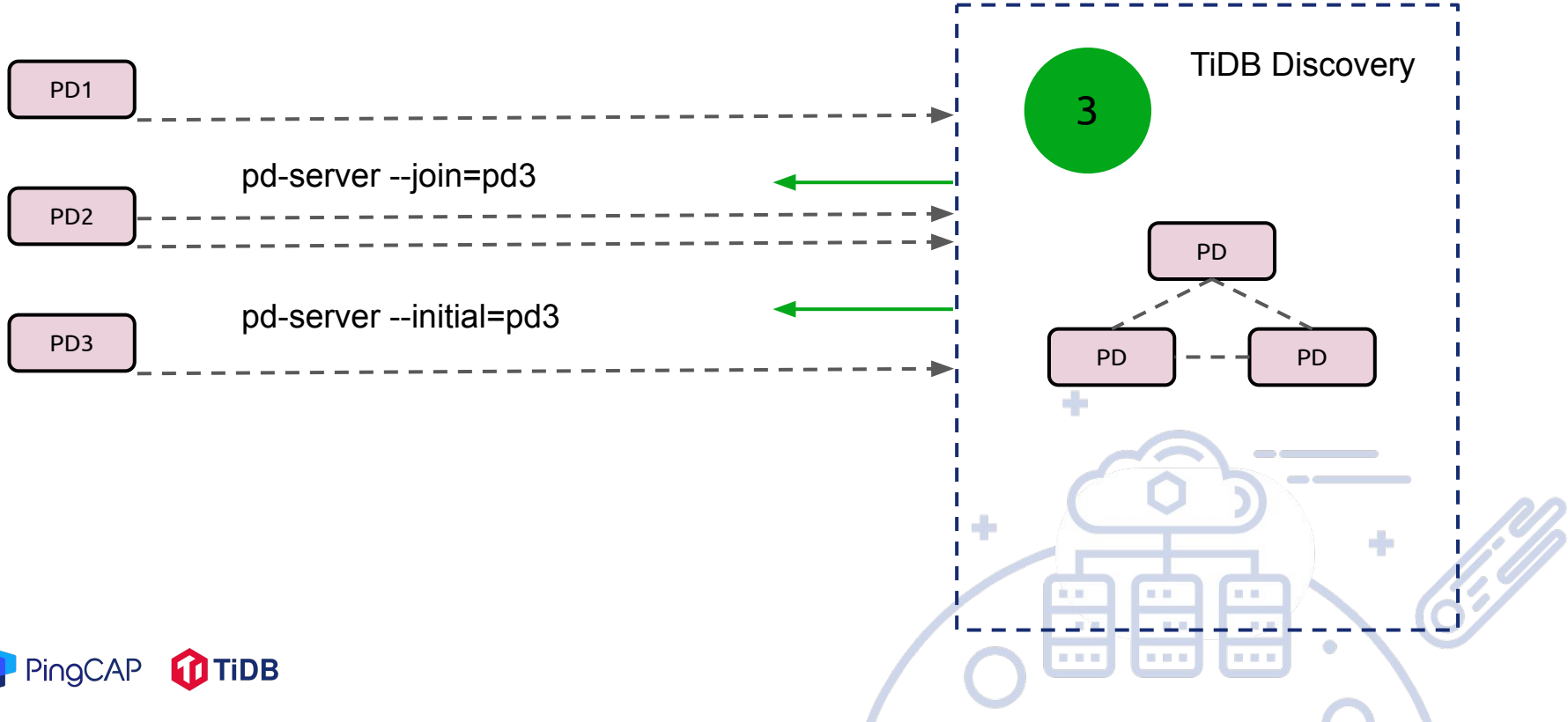
PD Startup



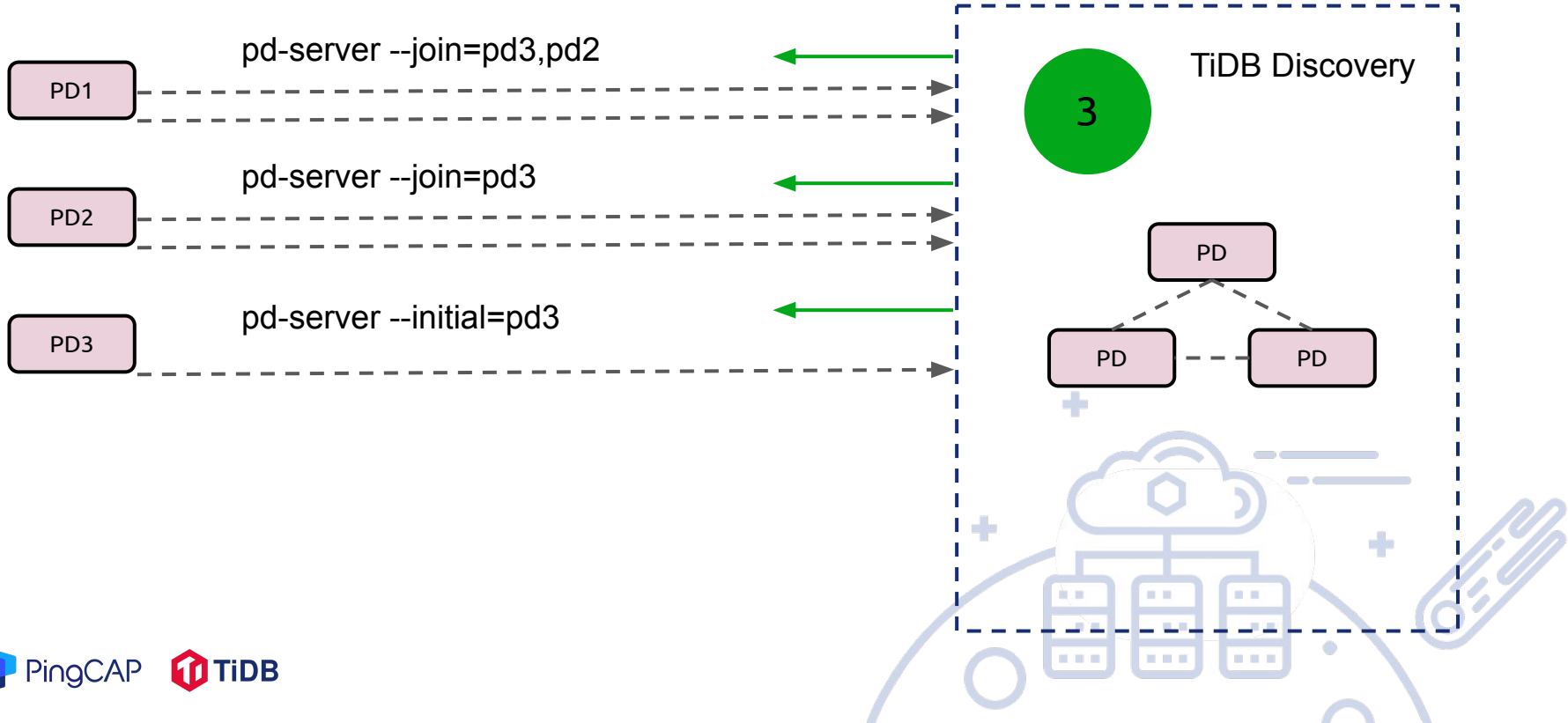
PD Startup



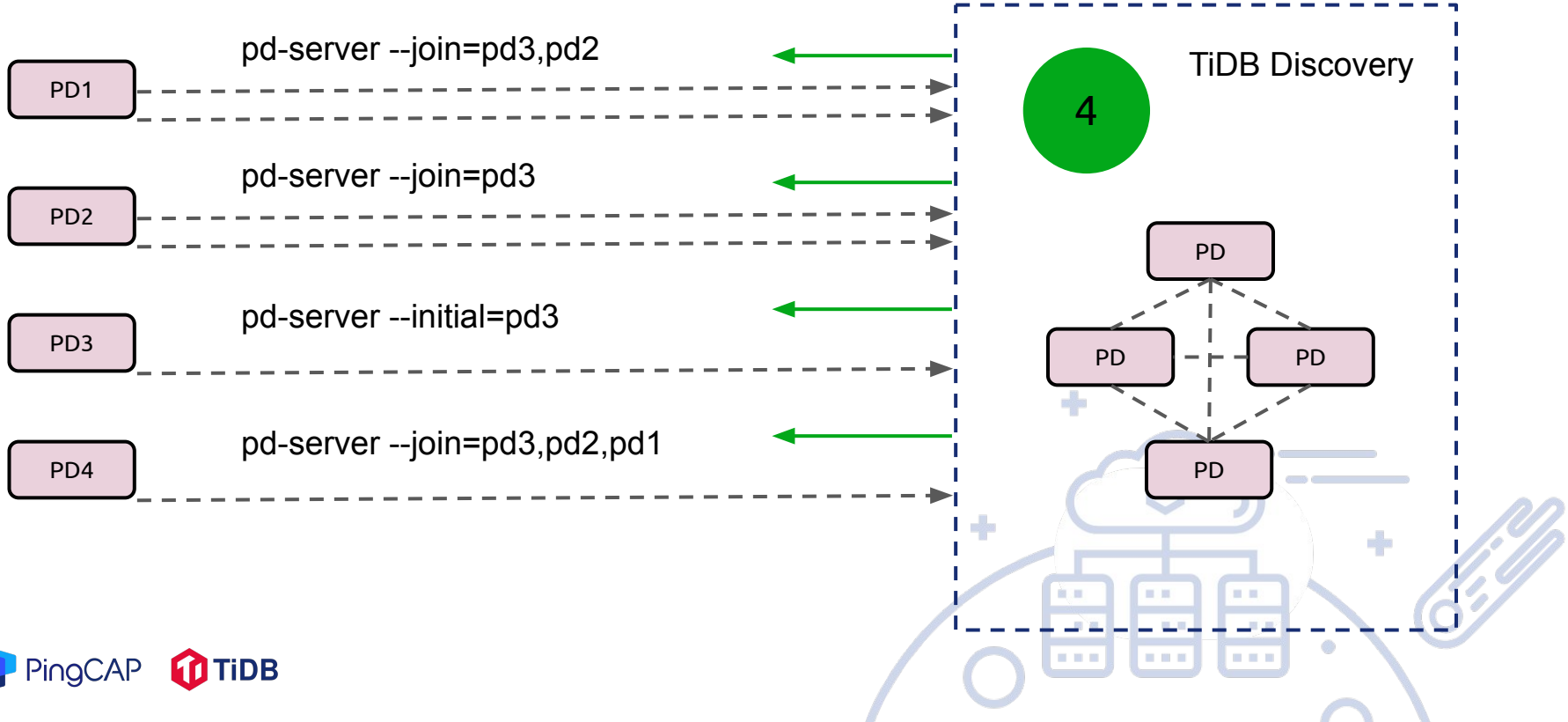
PD Startup


PD1
PD2
PD3
pd-server --join=pd3
pd-server --initial=pd3
TiDB Discovery
3
PD
PD
PD

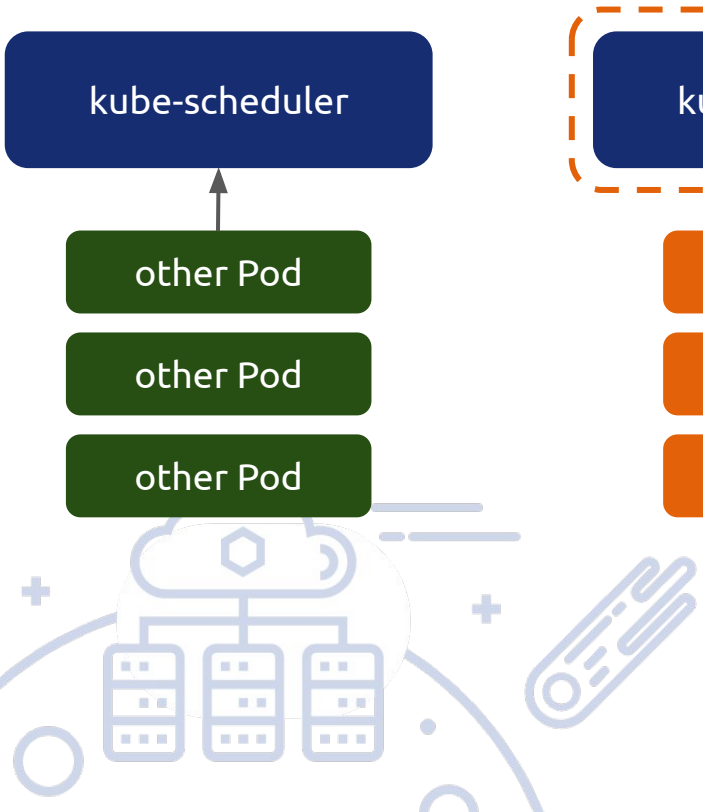
PD Startup

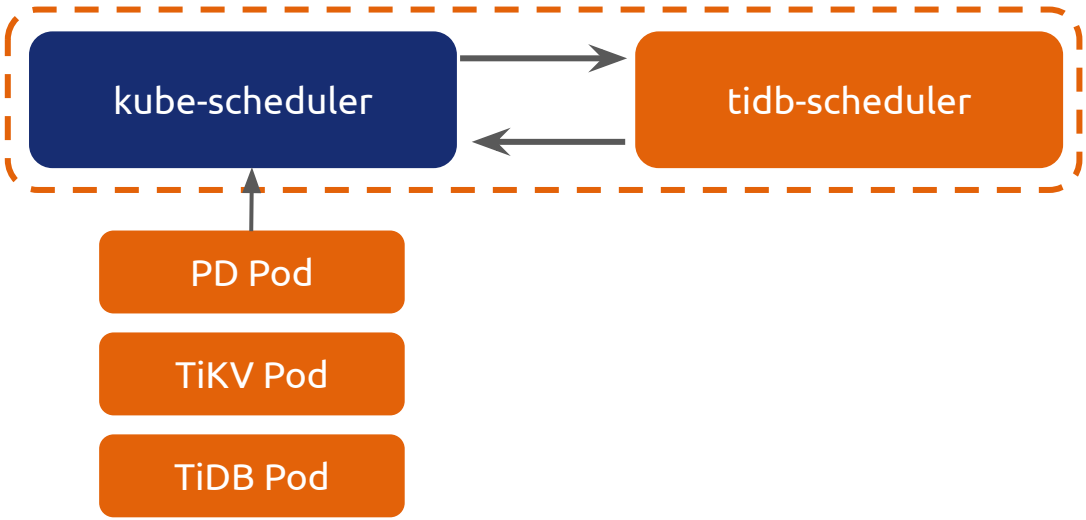

pd-server --join=pd3,pd2
PD1
pd-server --join=pd3
PD2
pd-server --initial=pd3
PD3
TiDB Discovery
3
PD
PD
PD

PD Startup


pd-server --join=pd3,pd2
PD1
pd-server --join=pd3
PD2
pd-server --initial=pd3
PD3
pd-server --join=pd3,pd2,pd1
PD4
4
TiDB Discovery
PD
PD
PD
PD

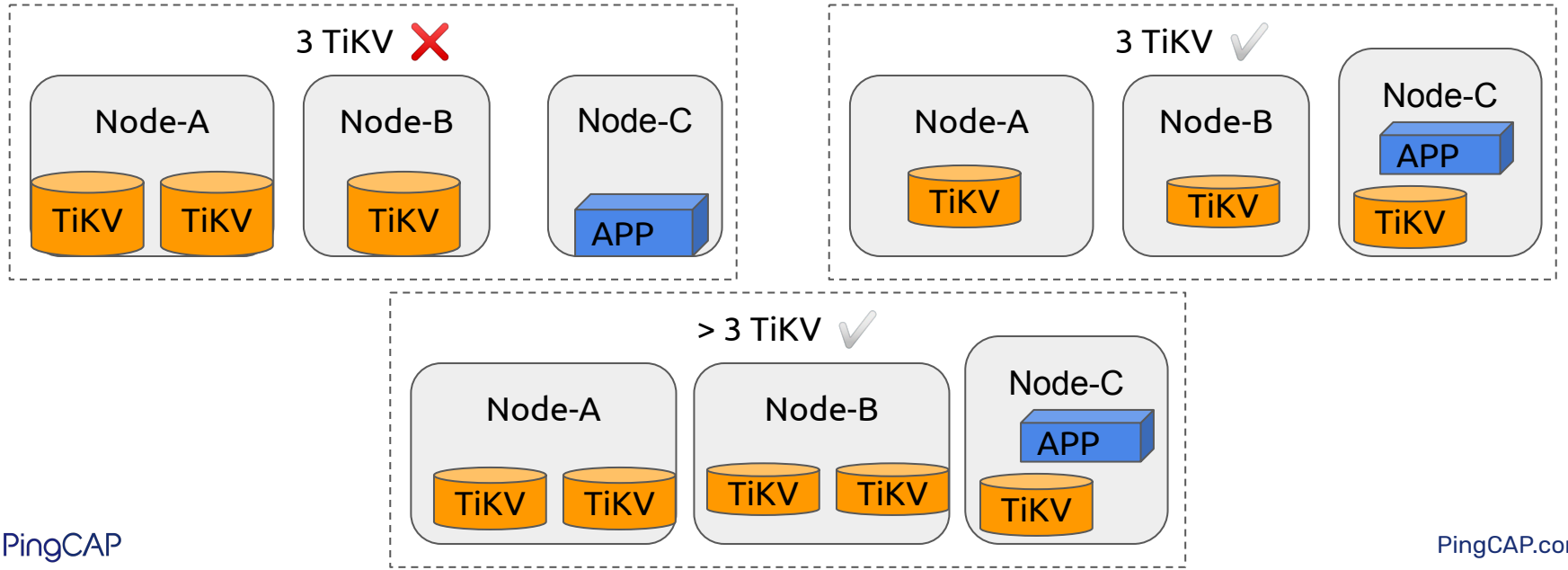
TiDB Scheduler





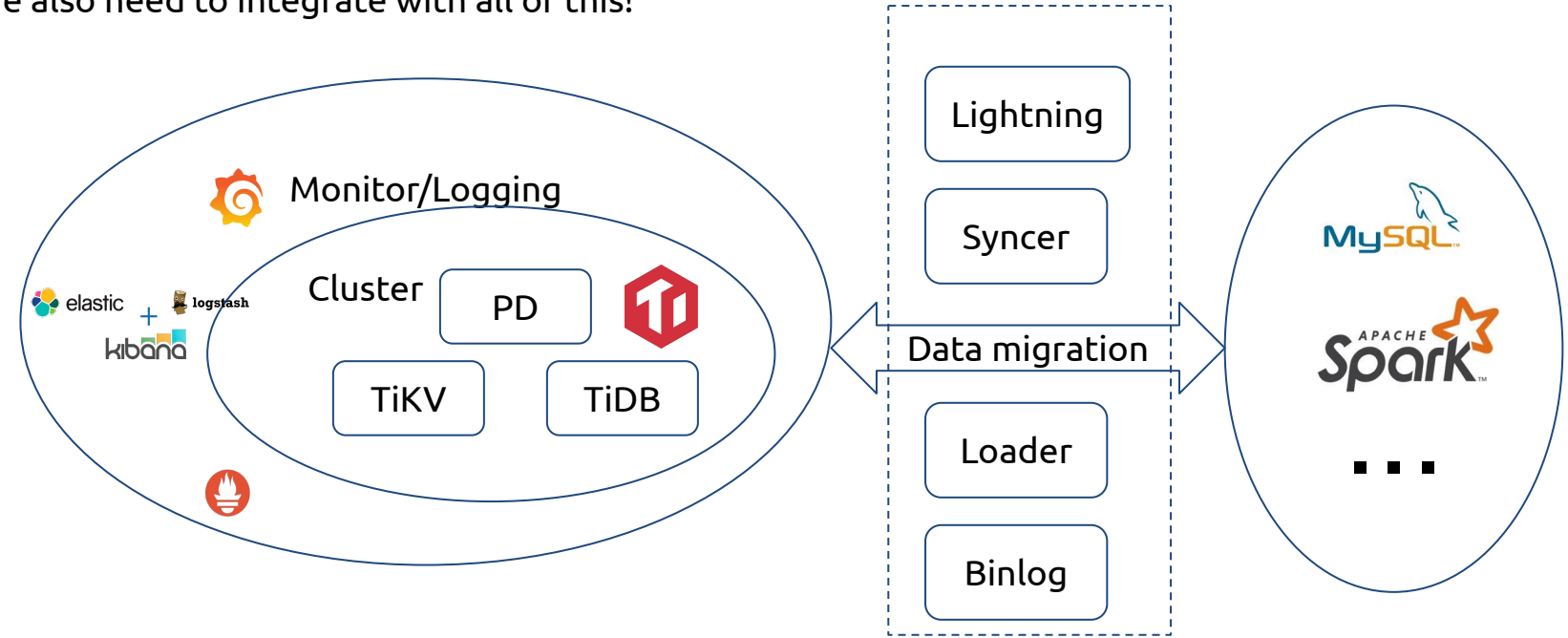
How we schedule stateful app

- Schedule consider existing pods topology



TiDB ecosystem

We also need to integrate with all of this!



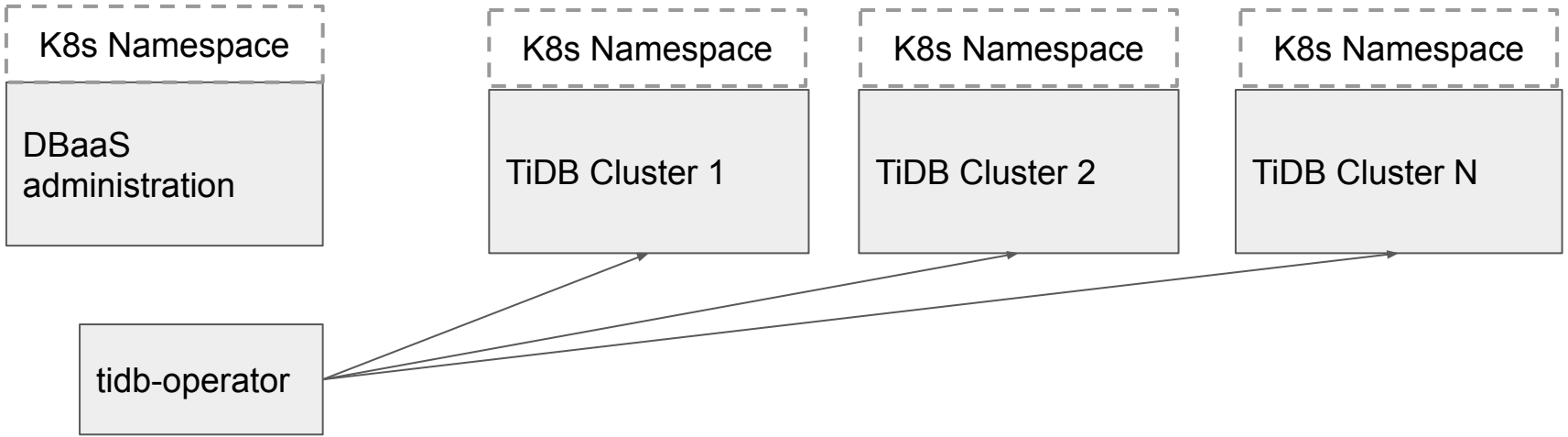
PingCAP

Part 3 - Making an aaS out of TiDB

K8s Namespace management

Each TiDB instance gets its own namespace.

We can offer access to the K8s namespace for their database clusters.



Dedicated Kubernetes for each datacenter

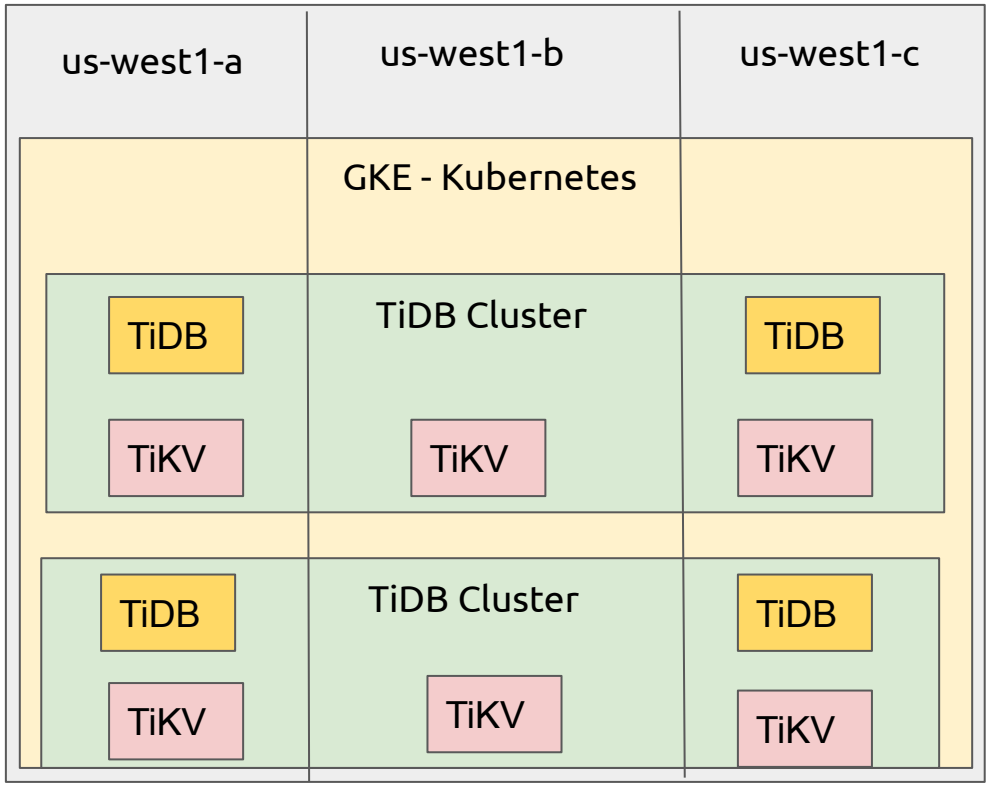
Each customer gets their own K8s cluster that can run multiple TiDB clusters.

Each datacenter operates a seprate K8s cluster.

On GKE, the K8s cluster is operated for free!

Region 1
TiDB Cluster 1
TiDB Cluster 2

Region 2
TiDB Cluster



PingCAP

Theory: run the operator on Kubernetes, add UI

An operator, what else do we need?

- Run tidb-operator
- Provide a button-click that does a helm install of a new TiDB Cluster

Okay, we will end up with more things

- Provide a nice UI
- How do we make sure users still have good visibility into what is going on in the system?

Practice: nothing works!

- Capacity management still needed
- Local SSD by default is worse than cloud disk
- Need a very recent version of Kubernetes with topology aware cloud Disk (we can use cloud disk for PD)
    - GKE is 1 version behind latest K8s, AWS is 2

Capacity Management

On prem

- Need a system to help the user provision new servers

Cloud: need auto scalers, existing autoscalers don’t work right

- Problem
  - They don’t provision new node types
  - May not understand local SSD properly
  - Won’t understand our requirements (high availability, isolation, local SSD)
- Solution
  - Write our own customized autoscaler

Local SSD

GKE, the flagship Kuberentes implementation, no documentation about the following

- Local SSD by default performs worse than cloud disk!
  - Must enable the nobarrier option
  - Must custom install the Linux guest environment (not available on GKE images)
  - SSD Disks are pre-mounted as 375 GiB, your pod cannot request a larger size
  - Failures are handled poorly: a node restart may not recognize the existing disk

Some of these issues will be helped by an alpha feature --local-ssd-volumes that has no timeline for stabilization to beta.

Networking and VPC issues

GCP VPC peering: limitations across projects (no DNS names)

- Solution: reserve an IP address for the load balancer

GCP internal load balancer cannot be accessed from another region

- Solution: no good solutions here. Use unstable IP addresses? Running your own HA load balancer in the cloud to work across VPCs is very difficult. Hope GCP fixes this soon.

Monitoring: re-use

- TiDB already provides Grafana dashboards
- Netdata for real-time metrics that are not TiDB specific
- Loki for logs
    - because we can re-use the Grafana interface
    - Horrible memory leak right now….

PingCAP

Thank You !

TiDB Community Slack Channel
https://pingcap.com/tidbslack/