Bolt

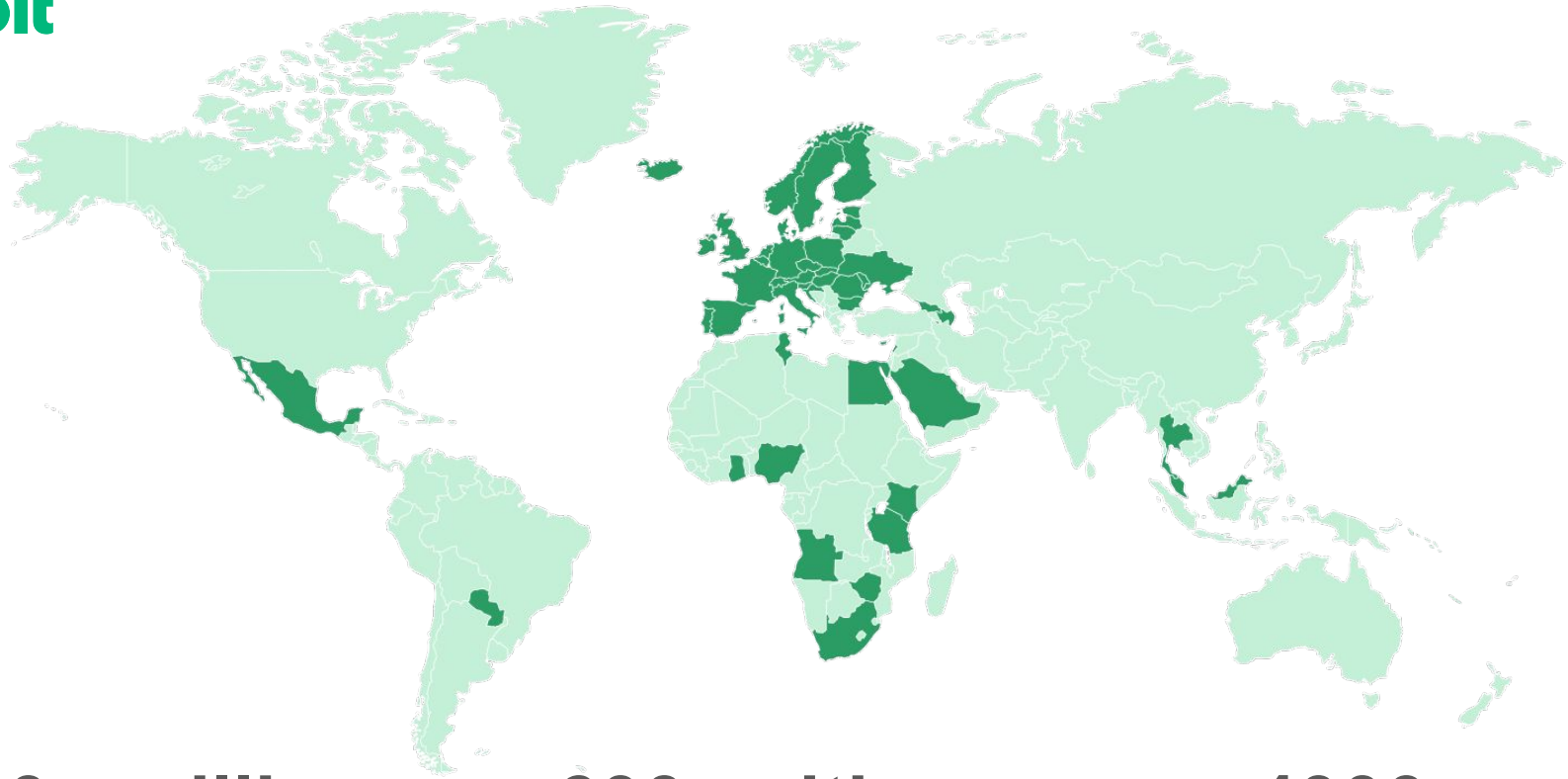
I Like To Move IT, Move IT

Replication in TiDB & MySQL

Leandro Morgado, Senior DBRE

FOSDEM 2025 - Brussels

Bolt



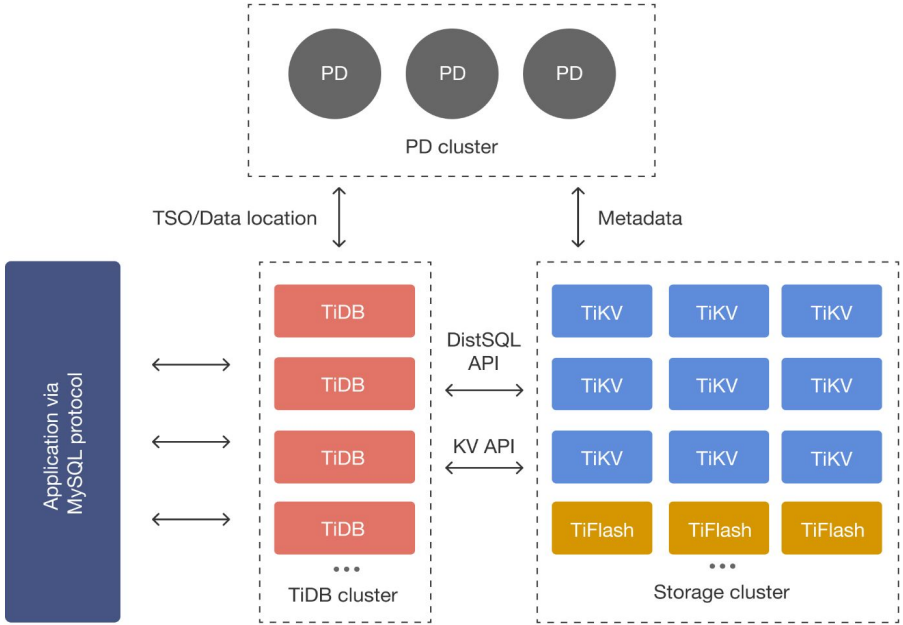
200+ million
customers

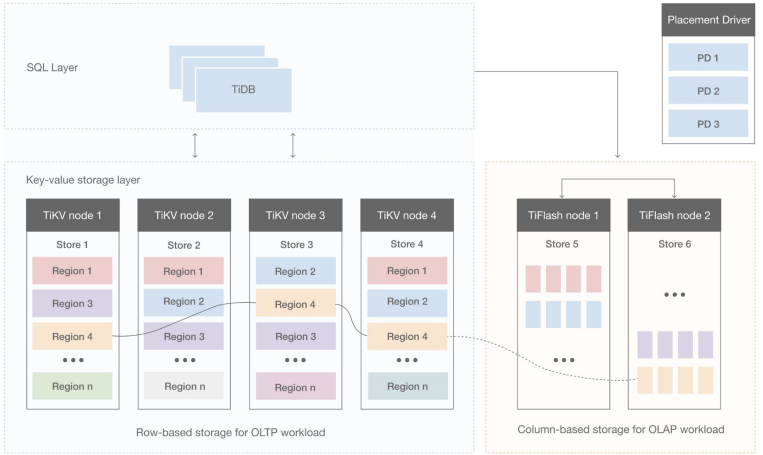
600+ cities
across **50 countries**

4000+
employees

What is TiDB?

Distributed MySQL compatible database



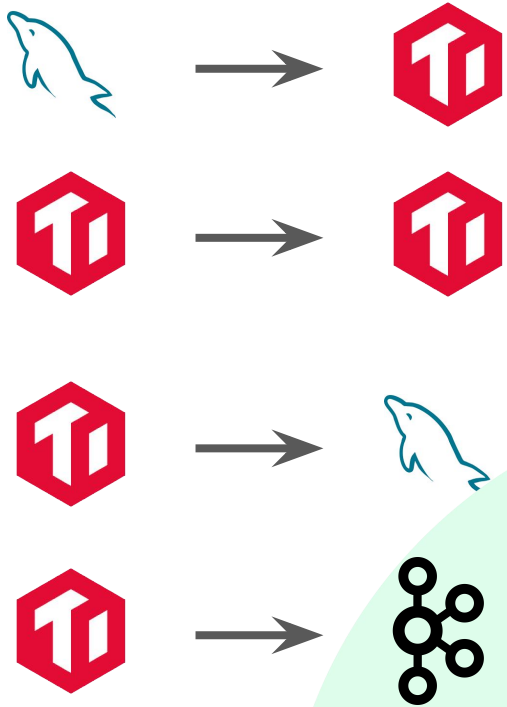


Why TiDB?

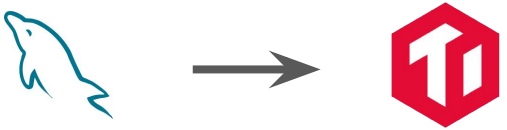
- Uses MySQL protocol => highly compatible with MySQL
- MySQL authentication compatible, GRANT works.
- Native CDC (Kafka) with TiCDC
- Monitoring and alerting are built in
- Native online schema (DDL) changes

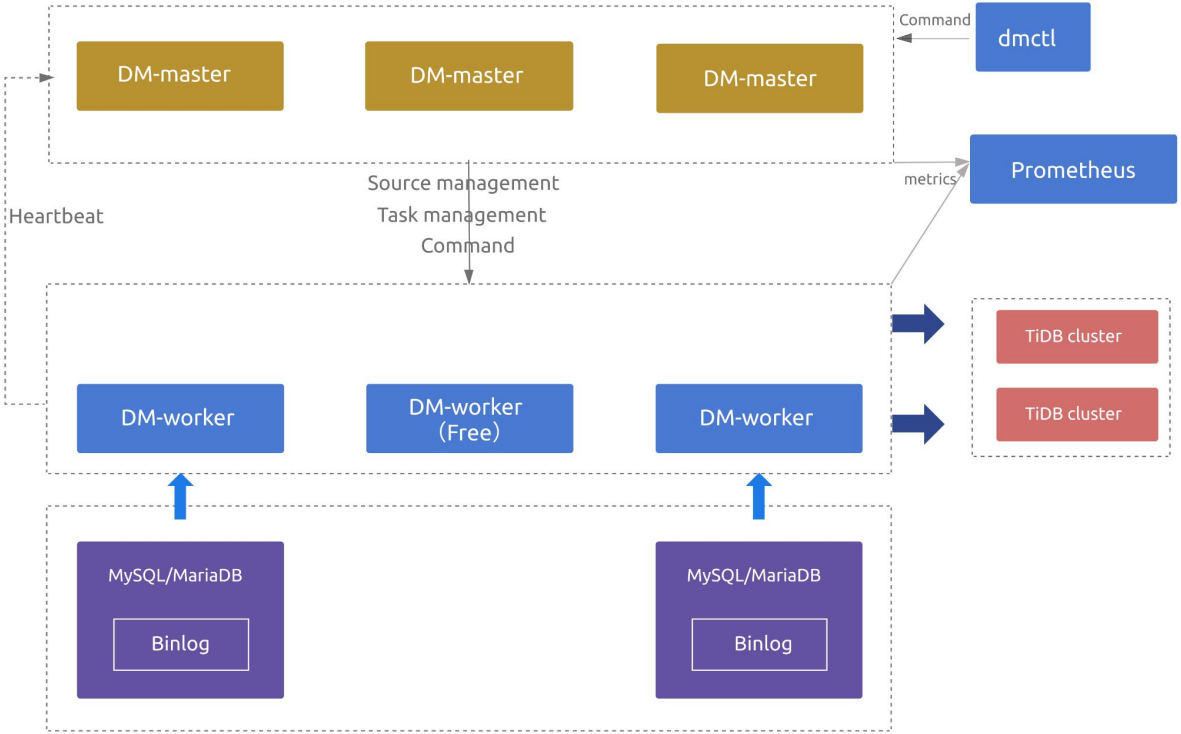
Replication Scenarios

- MySQL→TiDB with DM
- TiDB→TiDB with TiCDC
  - Same AWS region
  - Different AWS regions
- TiDB→MySQL with TiCDC
- TiDB→Kafka with TiCDC

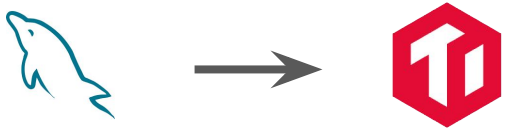


DM Architecture





Setup a DM cluster



- `tiup dm template` config with list of servers
- `tiup dm deploy $name $version ./topology.yaml` to deploy DM cluster
- Create a source config file with upstream IP, port, user, encrypted password (`tiup dmctl --encrypt 'pw'`)
- Add the upstream source to DM cluster

```
tiup dmctl --master-addr=$DM_MASTER_ADDR:8261
operate-source create mysql-source.yaml
```

Example topology & source files

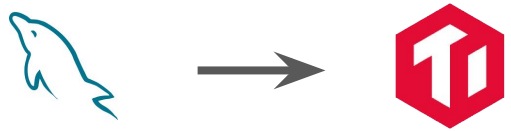
topology.yaml

```yaml
global:
  user: "tidb"
  ssh_port: 22
  deploy_dir: "/home/tidb/dm/deploy"
  data_dir: "/home/tidb/dm/data"
  # arch: "amd64"

master_servers:
  - host: dm-master-1.domain
  - host: dm-master-2.domain
  - host: dm-master-3.domain
worker_servers:
  - host: dm-worker-1.domain
  - host: dm-worker-2.domain

monitoring_servers:
  - host: 172.19.0.100

grafana_servers:
  - host: 172.19.0.100

alertmanager_servers:
  - host: 172.19.0.100
```

mysql-source.yaml

```yaml
source-id: "mysql-01"
from:
  host: "mysql-upstream.domain"
  user: "root"
  password: "fCxfQ9XKCezSzuCD0Wf5dUD+LsKegSg="
  # encrypt with `tiup dmctl --encrypt "123456"`
  port: 3306
```

Add a replication task

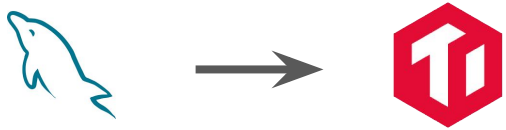


```
export DM_MASTER_ADDR=dm-master-1.domain:8261
tiup dmctl start-task testdm-task.yaml
```

testdm-task.yaml

```yaml
name: testdm
task-mode: all # load, dump and sync

target-database:
  host: "tidb-downstream.domain"
  port: 4000
  user: "root"
  password: "2c35Kx/9i59wV...." # encrypted with dmctl.

mysql-instances:
  - source-id: "mysql-01"
    block-allow-list:  "ba-rule1"

block-allow-list:
  ba-rule1:
    do-dbs: ["testdm"]
```

Verify a task

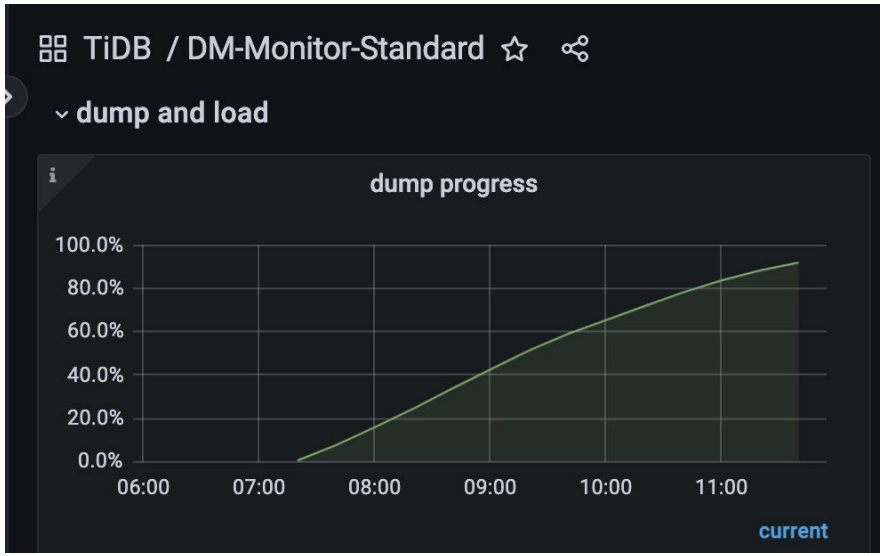


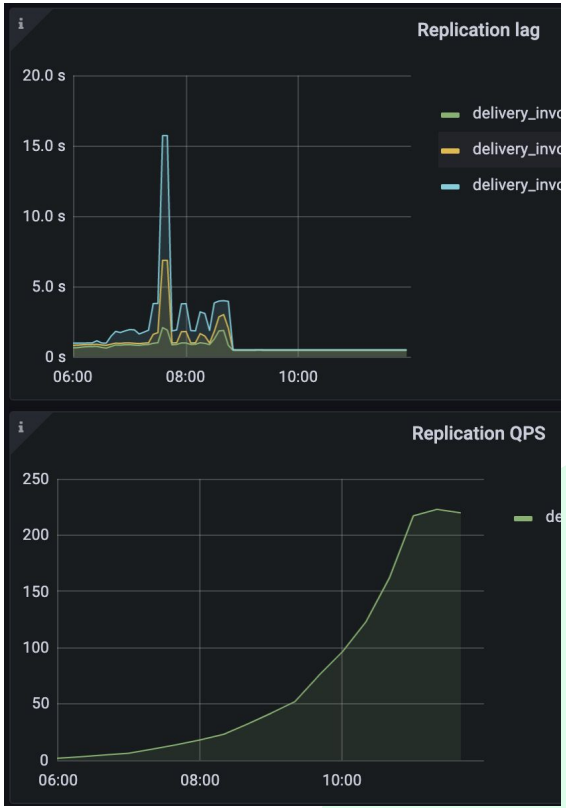
```
tiup dmctl query-status testdm
```

```
"sourceStatus": {
    "source": "mysql-01",
    "worker": "dm-worker-1.domain-8262",
},
"subTaskStatus": [
    {
        "name": "testdm",
        "stage": "Running",
        "unit": "Sync",
        ...
        "sync": {
        ...
            "secondsBehindMaster": "0",
```

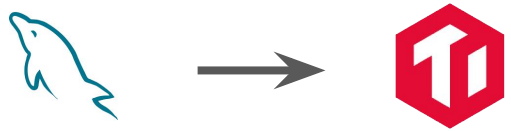
```
"subTaskStatus": [
    {
        "name": "testdm",
        "stage": "Running",
        "unit": "Dump",
        "result": null,
        "unresolvedDDLLockID": "",
        "dump": {
            "totalTables": "32",
            "completedTables": 28,
            "finishedBytes": 18436885770 22,
            "finishedRows": 11016880995,
            "estimateTotalRows": 12743422990,
            "bps": "139726561",
            "progress": "99.37 %"
        },
        "validation": null
    }
```

Grafana monitoring





DM TIPS



- **sync-diff-inspector** for data consistency check
- Can copy binlogs to DM worker to avoid purged binlogs
- Ignore replicating gh-ost tables on upstream

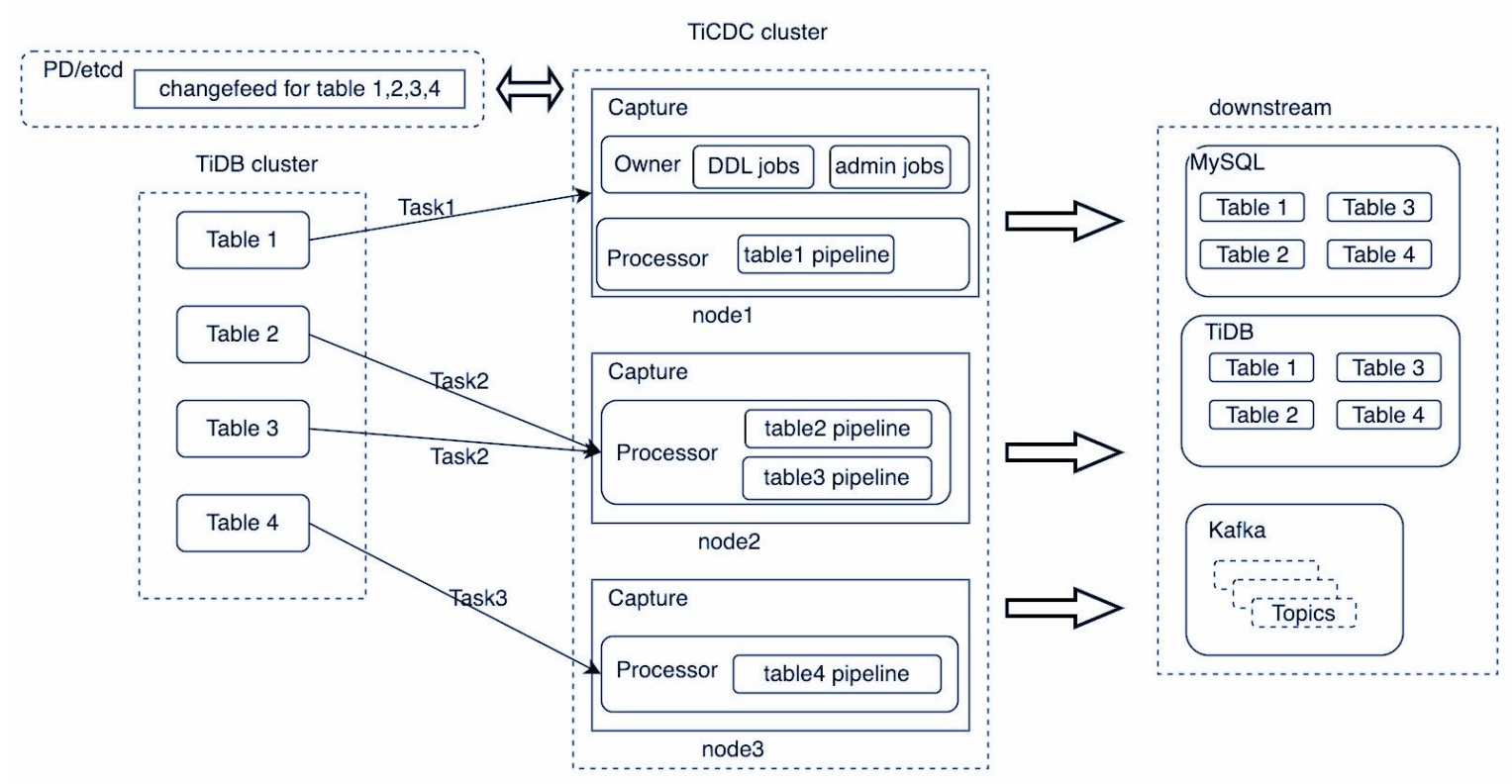
```
online-ddl: true
online-ddl-scheme: "gh-ost"
```

- Allows merging sharded DB upstream to single schema
- Filters allow some tricks,eg: migrate
  - Create BIGINT PK table on TiDB
  - Filter CREATE TABLE
- Archive data ignoring DELETEs

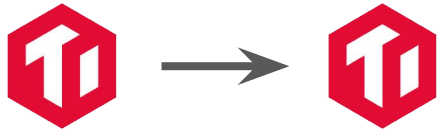
task.yaml

```yaml
filters:
  filter-rule-1:
    schema-pattern: "testdb"
    table-pattern: "testtable"
    events: ["create table"]
    action: Ignore
```

TiCDC Architecture



Deploy TiCDC



- Add CDC to existing TiDB:

  `tiup cluster scale-out <cluster-name> cdc.yaml`
- Can be deployed when creating new cluster
- Easy to test locally:

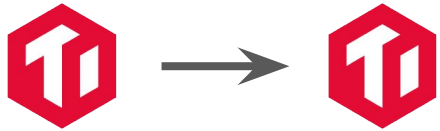
  `tiup playground --ticdc 1`
- Table needs valid index:

  PK or UNIQUE NOT NULL
- Network latency < 100ms

cdc.yaml

```yaml
cdc_servers:
  - host: cdc-1.domain
    gc-ttl: 86400
    data_dir: "/cdc-data"
  - host: cdc-2.domain
    gc-ttl: 86400
    data_dir: "/cdc-data"
```

Same AWS region



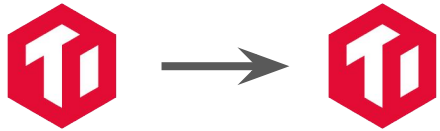
- Standby TiDB cluster
- Useful for TiDB version upgrades, quickly switch to old version

- Make a full backup of upstream TiDB (BR tool)
- Restore BR backup to downstream, get TSO

```
3/03/23 09:02:36.598 +00:00] [INFO] [collector.go:69] [\"Full Backup success summary\"]
 [BackupTS=440287101507338448] [total-kv=50694139] [total-kv-size=11.82GB] [average-spe
```

- Increase GC life time temporarily (eg: 24h)
  ⇔ MySQL's binlog expire_logs_days

Create Changefeed

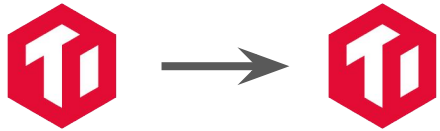


```
tiup cdc cli changefeed create \
    --server=http://cdc-1.domaion:8300 \
    --sink-uri="mysql://username:pw@downstream-tidb:3306/" \
    --changefeed-id="simple-task"
    --start-ts="440287101507338448"
    --config cdc_filter.yaml
```

```
[filter]
rules = ['*.*', '!test.*'] # Don't replicate test schema

[[filter.event-filters]]
matcher = ["archive.*"]   # Archive tables
ignore-event = ["delete"] # Ignore deletes
```

cdc_filter.yaml

Verifying a CDC task

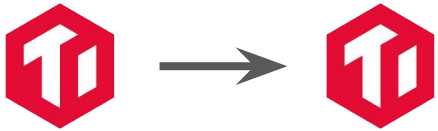


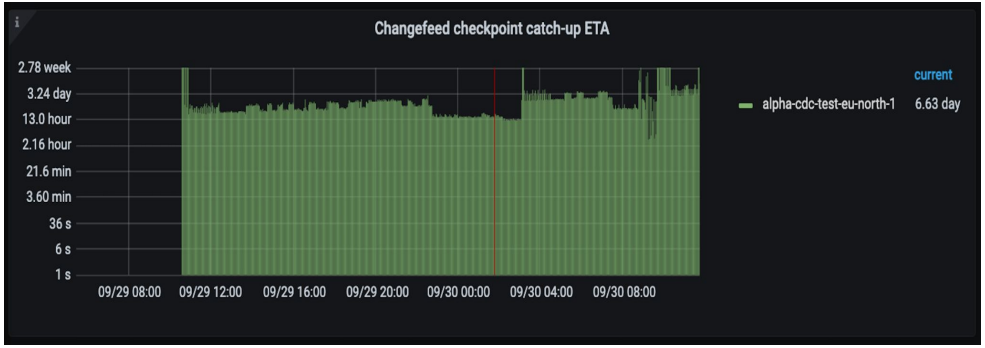
```
tiup cdc cli changefeed list \
--server=http://cdc-1.domain:8300
```

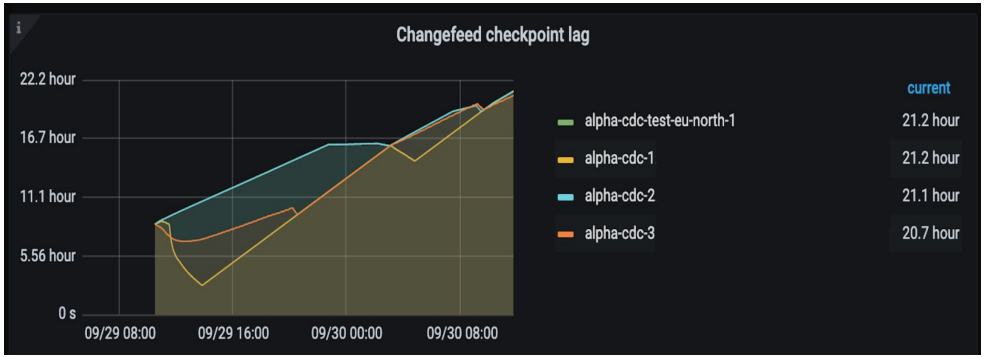
```
"id": "simple-task",
"namespace": "default",
"summary": {
  "state": "normal",
  "tso": 440292747891179670,
  "checkpoint": "2023-03-23 14:30:54.504",
  "error": null
}
```

```
tiup cdc cli changefeed query \
    --server=http://cdc-1.domain:8300 \
    --changefeed-id="simple-task"

{
  "upstream_id": 7213756783018473777,
  "namespace": "default",
  "id": "simple-task",
  "sink_uri": "mysql://username:pw@downstream-tidb:3306/",
  "config": {
    "memory_quota": 1073741824,
    "case_sensitive": true,
    "enable_old_value": true,
    "force_replicate": false,
    "ignore_ineligible_table": false,
    "check_gc_safe_point": true,
    "enable_sync_point": false,
    "bdr_mode": false,
    "sync_point_interval": 600000000000,
    "sync_point_retention": 86400000000000,
    "filter": {
      "rules": [
        "*.*",
        "!test.*"
      ],
      "event_filters": [
        {
          "matcher": [
            "archive.*"
          ],
          "ignore_event": [
            "delete"
          ],
          "ignore_sql": null,
          "ignore_insert_value_expr": "",
          "ignore_update_new_value_expr": "",
          "ignore_update_old_value_expr": "",
          "ignore_delete_value_expr": ""
        }
      ]
```

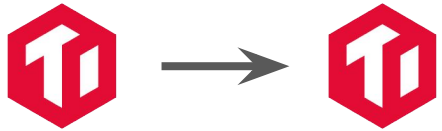
Monitoring CDC



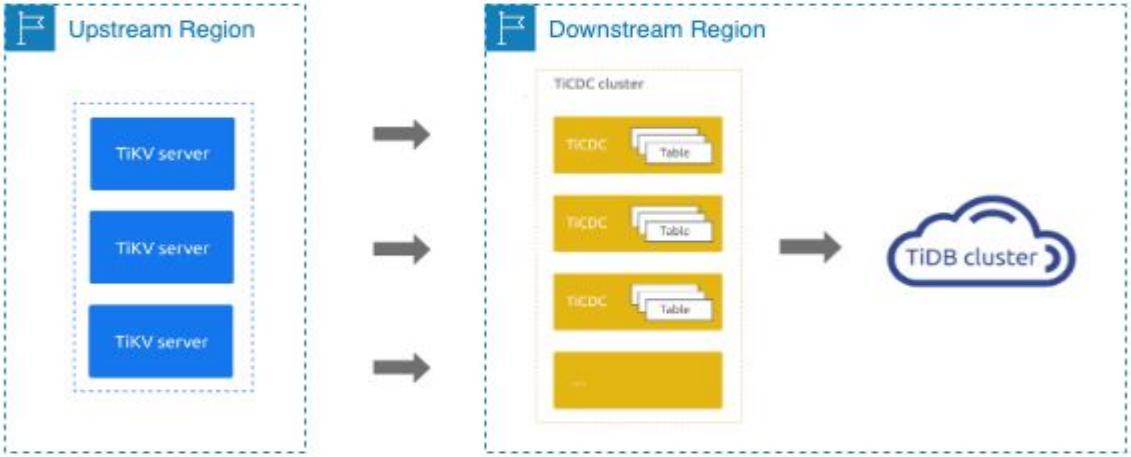




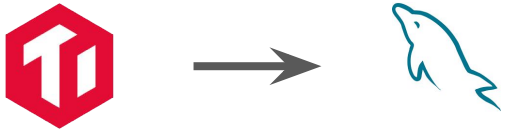
Different AWS regions



- Avoid single region failure with hot standby cluster
- Network latency can cause infinite lag for busy workload
- Solution is to deploy TiCDC cluster in downstream region
- Possible since version 6.5.0 LTS

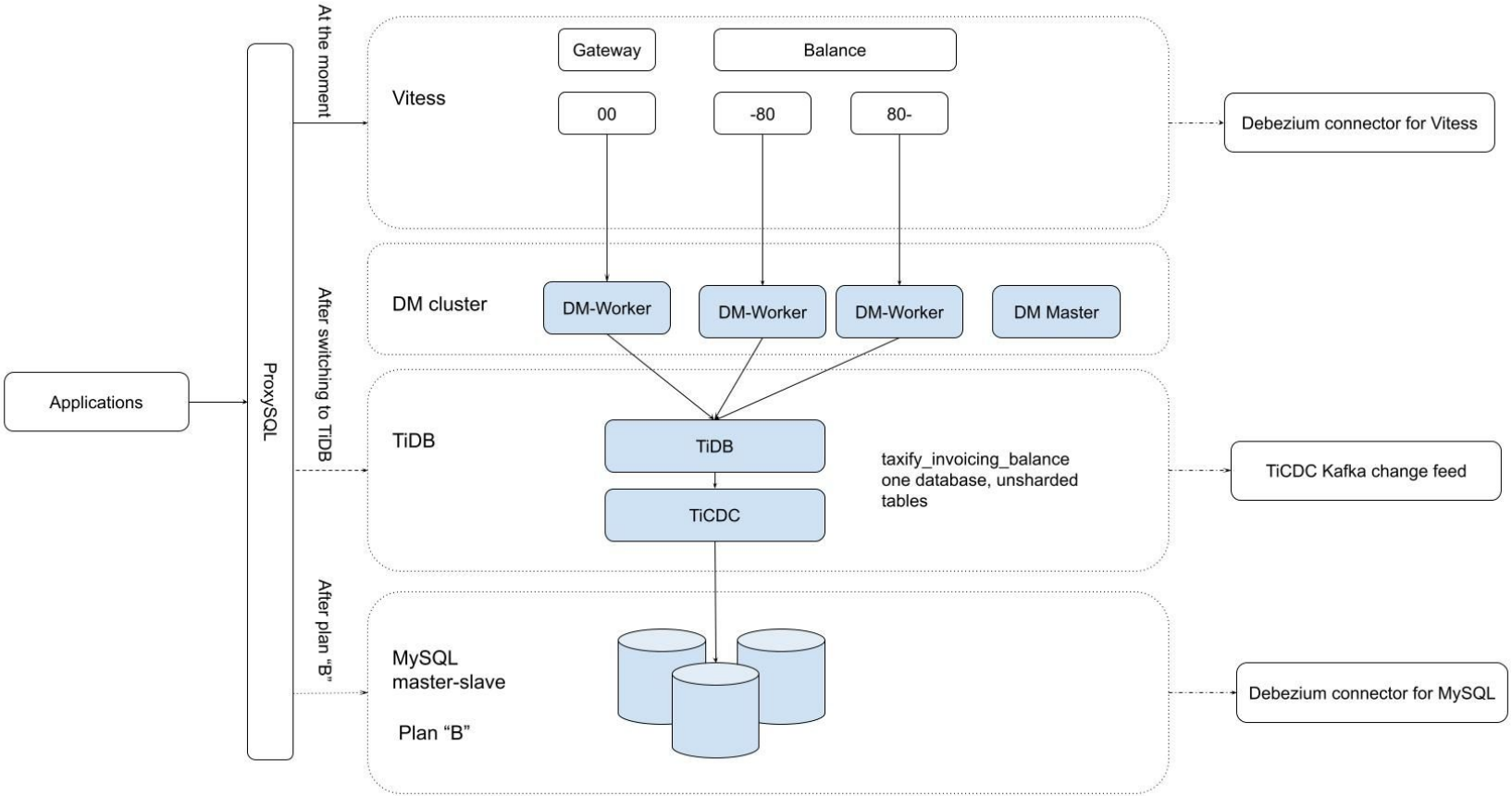


TiDB to MySQL



- TiCDC can also replicate to MySQL sink
- Use **Dumpling** to dump TiDB data, note the TSO
- Load data into MySQL
- Start CDC changefeed from TSO
- For low parallelism workload, reduce CDC **worker-count**
  Avoid MySQL DEADLOCKs

Mixed replication scenario



Conclusions

- TiDB has great tools to move and replicate data around 💯
- **DM** to import/replicate from MySQL 🚀
- **TiCDC** to export to MySQL, TiDB, Kafka

Bolt

Thank you!

Q & A