Open Source Conference 2022 Online/Spring

RDBMSとNoSQLのメリットを併せ持つクラウドネイティブなNewSQLデータベース「TiDB」をKubernetesで動かしてみよう！

Executive Technical Advisor / Makoto Hasegawa

PingCAP

WHO am I

Working at // PingCAP(JP)

Job Title // Executive Technical advisor

Hobby // Playing bass

Currently //

Develop and maintain private OpenStack cloud.

Develop and maintain Kubernetes as a Service platform.

Kubernetes organization member (sig-docs-ja)

CKA / CKAD / CKS

Makoto Hasegawa

Hi

Twitter // @makocchi

Facebook // makocchi0923

PingCAP について

PingCAP

- ☑ NewSQL製品「TiDB」を主に開発
- ☑ ワールドワイドにビジネスを展開中 (1600社以上の採用実績)
- ☑ CNCF に「TiKV」及び「Chaos Mesh」を寄与

29,000+	1,500+	1,500+	2
Github Stars	企業数	貢献者	CNCFに寄付したプロジェクト数

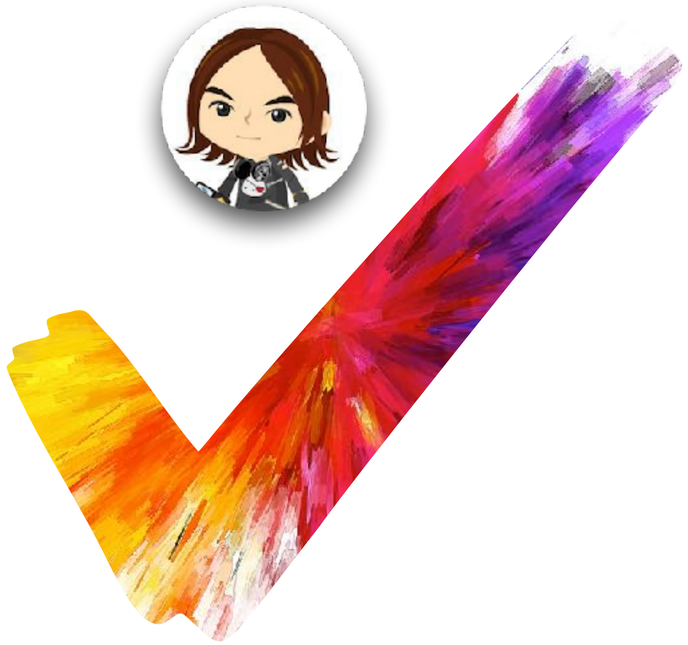
PingCAP

- ☑TiKV は CNCF で「Graduated」プロジェクトと認定されています
- ☑Chaos Mesh は現在「Incubating」ステータスとなっています

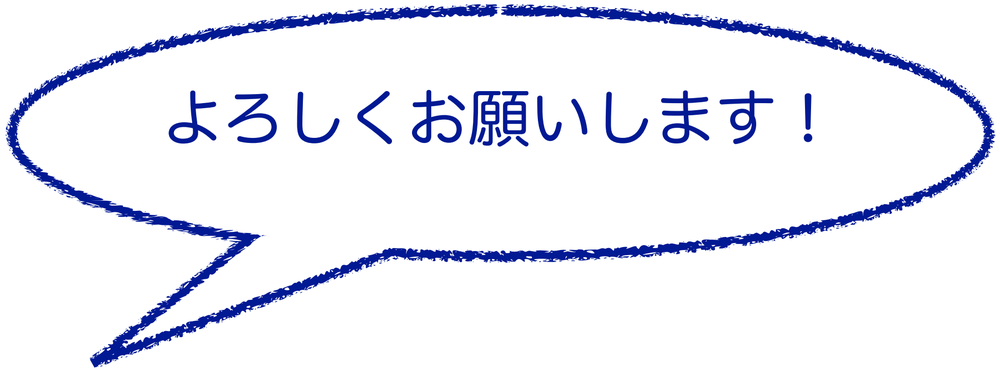
PingCAP

☑2021年4月に日本法人が発足しました 🎉🎉🎉

💪 本日のアジェンダ💪



- ☑TiDB について理解しよう
- ☑Kubernetes で TiDB を構築する際のポイント
- ☑Kubernetes で TiDB を運用する際のポイント
- ☑まとめ



TiDB と Kubernetes の組み合わせ方について

一緒に学んでいきましょう👌

TiDB とは

https://github.com/pingcap/tidb

- ☑TiDB は HTAP ワークロードをサポートするオープンソースの NewSQLデータベースです
- ☑TiDB は MySQL と互換性があり、水平方向のスケーラビリティ、強力な一貫性、および高可用性を備えています
- ☑主に PingCAP 社によって開発されています

※ TiDB の Ti は Titanium の Ti

※ HTAP は Hybrid Transactional and Analytical Processing

TiDB の製品・プロジェクト

フルマネージドサービス


エンタープライズ

サブスクリプション

ベンダーロックインなし
24時間365日サポート
技術トレーニングとコンサルティング

コミュニティエディション

TiDB の特徴

- ☑ Horizontal Scalability (水平拡張)
- ☑ MySQL Compatible Syntax (MySQL 互換)
- ☑ Distributed Transactions (分散トランザクション)
- ☑ Cloud Native (クラウドネイティブ志向)
- ☑ Minimize ETL (OLTP と OLAP のサポート)
- ☑ High Availability (高可用性)

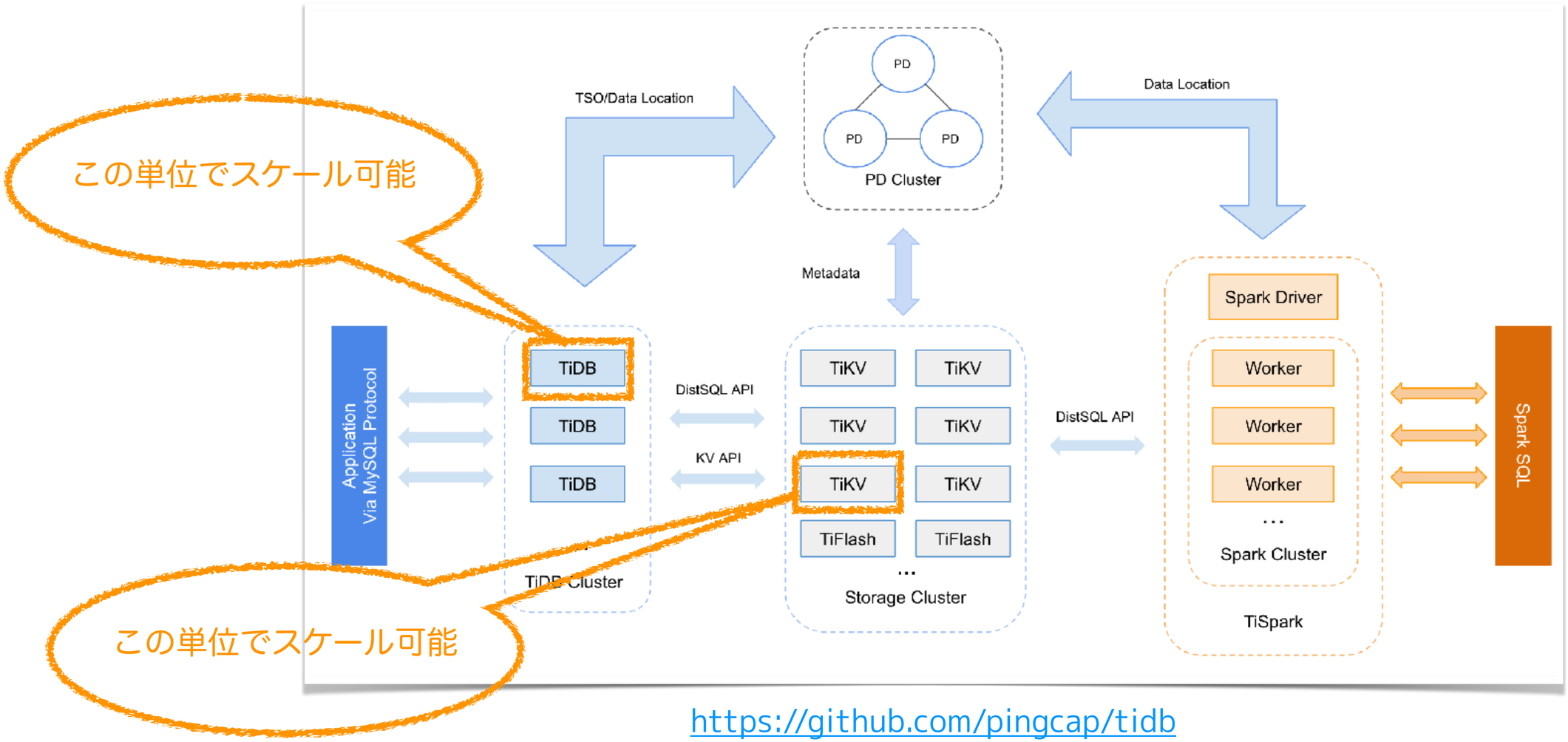
TiDB の特徴

- ☑ Horizontal Scalability (水平拡張)
- ☑ MySQL Compatible Syntax (MySQL 互換)
- ☑ Distributed Transactions (分散トランザクション)
- ☑ Cloud Native (クラウドネイティブ志向)
- ☑ Minimize ETL (OLTP と OLAP のサポート)
- ☑ High Availability (高可用性)

TiDB の特徴 (Horizontal Scalability)

- ☑ TiDB はアーキテクチャ上 SQL を処理するコンポーネントとストレージ部分が分離されています
- ☑ それぞれの単位(SQL、ストレージ)で水平に拡張が可能です(単純に増やすだけ！)
- ☑ 無停止で水平拡張可能
- ☑ 従来のデータベースでは垂直にしか拡張できない(リソースを追加で割り当てる)場合が多いと思いますが、垂直の拡張は限界があります

TiDB の特徴 (Horizontal Scalability)



https://github.com/pingcap/tidb

TiDB の特徴

- ☑ Horizontal Scalability (水平拡張)
- ☑ MySQL Compatible Syntax (MySQL 互換)
- ☑ Distributed Transactions (分散トランザクション)
- ☑ Cloud Native (クラウドネイティブ志向)
- ☑ Minimize ETL (OLTP と OLAP のサポート)
- ☑ High Availability (高可用性)

TiDB の特徴 (MySQL Compatible Syntax)

- ☑ MySQL 5.7 と互換があるように作られています
- ☑ 既存の MySQL のクライアントやライブラリはそのまま使うことが可能
- ☑ ただし完全に対応しているわけではないので注意が必要
  https://docs.pingcap.com/tidb/stable/mysql-compatibility
- ☑ ほとんどの場合はアプリケーションに手を加えることなくデータベースを入れ替えることができます

TiDB の特徴 (MySQL Compatible Syntax)

☑MySQL 8.0 互換はもう少し時間がかかりそう・・？

MySQL 8.0 Compatibility #7968

Open · 12 of 49 tasks · morgo opened this issue on 20 Oct 2018 · 5 comments

morgo commented on 20 Oct 2018 • edited — Member

This issue is to track what is required to move from MySQL 5.7 to MySQL 8.0 compatibility.

See the complete list.

Essential

- [x] Common Table Expressions (regular and recursive) support common table expression (CTE) #17472
- [x] Window Functions Support window functions for TiDB #4807 docs/design: add proposal for window functions #8117
- [x] `utf8mb4` as default (rename `utf8` as in TiDB)
- [x] Role Based Authentication
- [ ] `SKIP LOCKED` Support `SKIP LOCKED` Syntax #18207

Nice to Have

- [ ] Lateral derived tables (8.0.14) Details
- [x] Invisible Indexes UCP: Support Invisible Indexes #9246
- [ ] `JSON_TABLE` function + add remaining JSON functions that were backported to 5.7
- [ ] multi-valued indexes on json arrays / json_overlaps() and member of()

TiDB の特徴

- ☑ Horizontal Scalability (水平拡張)
- ☑ MySQL Compatible Syntax (MySQL 互換)
- ☑ Distributed Transactions (分散トランザクション)
- ☑ Cloud Native (クラウドネイティブ志向)
- ☑ Minimize ETL (OLTP と OLAP のサポート)
- ☑ High Availability (高可用性)

TiDB の特徴 (Distributed Transactions)

- ☑TiDB では内部でデータをチャンク(Region)に分割して分散配置します
- ☑一貫性を保つ為に 2PC(2-phase commit) 等、様々な技術が採用されています
- ☑2PC 等の技術については下記の資料に詳しく書いてあります

「TiDBのトランザクション」

https://www.slideshare.net/akiomitobe/tidb-249788408

TiDB の特徴

- ☑ Horizontal Scalability (水平拡張)
- ☑ MySQL Compatible Syntax (MySQL 互換)
- ☑ Distributed Transactions (分散トランザクション)
- ☑ Cloud Native (クラウドネイティブ志向)
- ☑ Minimize ETL (OLTP と OLAP のサポート)
- ☑ High Availability (高可用性)

TiDB の特徴 (Cloud Native)

- ☑クラウド環境(Public や Private、そして hybrid)に親和性が良い作りになっています(複数のコンポーネントで構成されている点等)
- ☑それぞれのコンポーネントは自由にスケールさせることができます
- ☑TiDB のストレージ部分は TiKV というソフトウェアで実現されていますが、TiKV は CNCF によって管理されています(Guraduated)

TiDB の特徴

- ☑ Horizontal Scalability (水平拡張)
- ☑ MySQL Compatible Syntax (MySQL 互換)
- ☑ Distributed Transactions (分散トランザクション)
- ☑ Cloud Native (クラウドネイティブ志向)
- ☑ Minimize ETL (OLTP と OLAP のサポート)
- ☑ High Availability (高可用性)

TiDB の特徴 (Minimize ETL)

- ☑TiDB は OLTP と OLAP を同時にサポート(HTAP)しています
- ☑ETL に読ませるために、わざわざ分析用のデータに変換する必要がありません (別途 TiFlash というコンポーネントを動かす必要があります)
- ☑あと実は MySQL のインターフェイスだけではなくて Spark のインターフェイスも使うことができます (TiSpark)

※ OLTP : On Line Transaction Processing
※ OLAP : On Line Analytical Processing
※ HTAP : Hybrid Transactional Analytical Processing

TiDB の特徴

- ☑ Horizontal Scalability (水平拡張)
- ☑ MySQL Compatible Syntax (MySQL 互換)
- ☑ Distributed Transactions (分散トランザクション)
- ☑ Cloud Native (クラウドネイティブ志向)
- ☑ Minimize ETL (OLTP と OLAP のサポート)
- ☑ High Availability (高可用性)

TiDB の特徴 (High Availability)

- ☑ TiDB を構成するコンポーネント達はそれぞれ冗長化が可能です
- ☑ TiDB は内部の TiKV にデータを保存していますが、Raft を使ってチャンクを冗長して配置しています
- ☑ TiKV 側では障害が発生しても Raft によって Leader Election が行われ、自動でアクセス先が切り替わります
- ☑ アクセス頻度が高いノード(ホットスポット)が発生した場合は自動的にデータが移されて負荷分散されます

TiDB の特徴(再掲)

- ☑ Horizontal Scalability (水平拡張)
- ☑ MySQL Compatible Syntax (MySQL 互換)
- ☑ Distributed Transactions (分散トランザクション)
- ☑ Cloud Native (クラウドネイティブ志向)
- ☑ Minimize ETL (OLTP と OLAP のサポート)
- ☑ High Availability (高可用性)

TiDB に弱点は無いのでしょうか？

個人的に思う TiDB の弱点

- ☑TiDB は分散されてデータが配置されており、SQL を処理するコンポーネントもストレージ部分と切り離されている性質上、どうしてもトランザクションにネットワーク等の latency が乗ってくることになります
- ☑つまり数 ms 以内でクエリを返さなければならない要件のシステムには不向き💦
- ☑また、データは通常 3 つに冗長されて保存されるのでストレージはそれなりに膨らむでしょう
- ☑Stored Procedure や Trigger や UDF 等はまだ対応していないのもちょっと残念なポイント

Kubernetes で TiDB を構築する際のポイント

そもそも TiDB には・・・

「tiup」という超絶便利な構築ツールがある💪

tiup とは

https://github.com/pingcap/tiup

- ☑ TiDB をインストール、管理するためのツール
- ☑ TiDB の構成情報を yaml で定義し、tiup に読み込ませることで TiDB を自動で構築することが可能
- ☑ 構築するだけではなくて sysctl の設定やチューニング等もやってくれる
- ☑ 構成ノードに対して ssh を行い操作するので ssh ができる環境が必要
- ☑ なにげに TPC-C のベンチマークもかけることが可能

tiup とは

https://github.com/pingcap/tiup

構成情報の yaml はこんな感じ 👉

```yaml
global:
  user: "tidb"
  ssh_port: 22
  deploy_dir: "/tidb-deploy"
  data_dir: "/tidb-data"
server_configs: {}
pd_servers:
  - host: 10.0.1.4
  - host: 10.0.1.5
  - host: 10.0.1.6
tidb_servers:
  - host: 10.0.1.7
  - host: 10.0.1.8
  - host: 10.0.1.9
tikv_servers:
  - host: 10.0.1.1
  - host: 10.0.1.2
  - host: 10.0.1.3
monitoring_servers:
  - host: 10.0.1.4
grafana_servers:
  - host: 10.0.1.4
alertmanager_servers:
  - host: 10.0.1.4
```

この構成情報を元に
TiDB のクラスターを自動で作成してくれる！

TiDB には「tiup」という構築ツールがある💪

しかし tiup は対象が vm の場合しか対応していない😢

※vmだけではなくて物理の場合も同様

TiDB には「tiup」という構築ツールがある💪

しかし tiup は対象が vm の場合しか対応していない😢

※vmだけではなくて物理の場合も同様

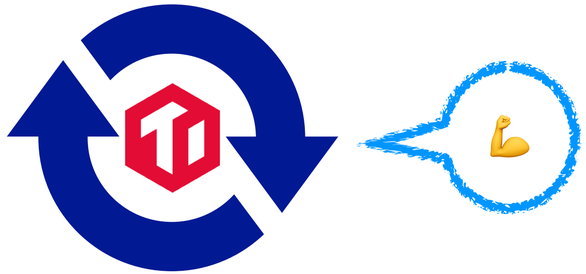
Kubernetes での構築に対応していない😭

そこで登場するのが

tidb-operator

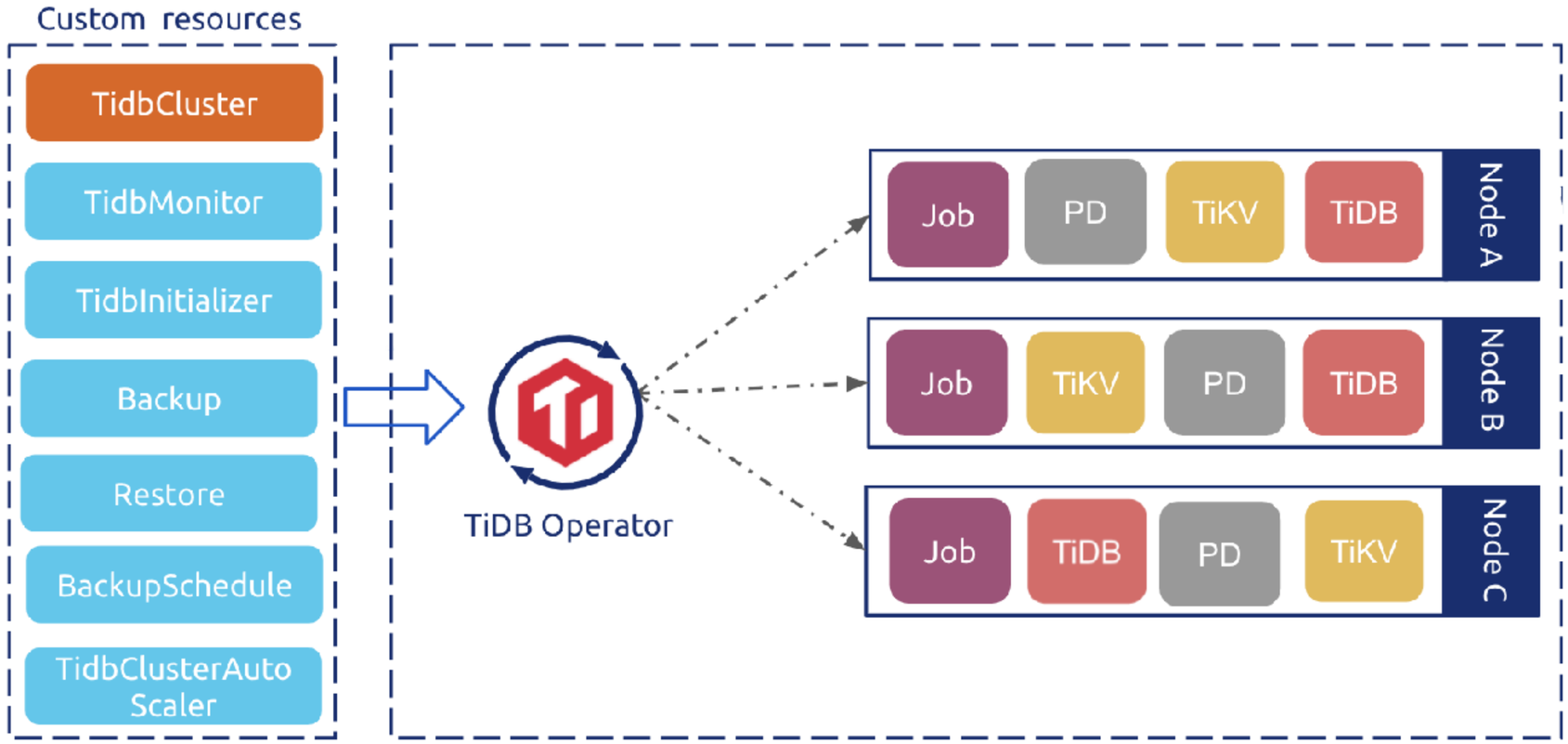
すごいぞ tidb-operator

- ☑tidb-operator は Kubernetes 上で動くコントローラー
- ☑tiup で使用する topology.yaml と同じような内容を Kubernetes のカスタムリソース TidbCluster で定義してあげる
- ☑tidb-operator は定義された通りに Kubernetes のクラスター上に TiDB のクラスターを自動で構築してくれる
- ☑構築だけではなく、バックアップやリストアも可能



すごいぞ tidb-operator

Kubernetes のカスタムリソースとオペレーターの関係



https://docs.pingcap.com/tidb-in-kubernetes/stable/architecture

tiup と tidb-operator

tiup

```
$ tiup cluster deploy tidb-test v5.2.2 ./topology.yaml
```

tiup cluster deploy 相当のことは kubectl apply 等で TidbCluster のリソースを定義するだけで OK

クラスター名やバージョンも TidbCluster で定義してあげます

tidb-operator

```
$ kubectl apply -f tidb_cluster.yaml
```

TidbCluster の定義の仕方

TidbCluster

```
apiVersion: pingcap.com/v1alpha1
kind: TidbCluster
metadata:
  name: tidb-test
spec:
  version: v5.2.2
  pd:
    baseImage: pingcap/pd
    replicas: 3
    requests:
      storage: "10Gi"
    config: {}
  tikv:
    baseImage: pingcap/tikv
    replicas: 3
    requests:
      storage: "10Gi"
    config: {}
  tidb:
    baseImage: pingcap/tidb
    replicas: 3
    config: {}
```

こんな感じで
定義してあげれば OK

TidbCluster の定義の仕方

TidbCluster

```
apiVersion: pingcap.com/v1alpha1
kind: TidbCluster
metadata:
  name: tidb-test
spec:
  version: v5.2.2
  pd:
    baseImage: pingcap/pd
    replicas: 3
    requests:
      storage: "10Gi"
    config: {}
  tikv:
    baseImage: pingcap/tikv
    replicas: 3
    requests:
      storage: "10G
    config: {}
  tidb:
    baseImage: pi
    replicas: 3
    config: {}
```

topology.yaml

```
global:
  user: "tidb"
  ssh_port: 22
  deploy_dir: "/tidb-deploy"
  data_dir: "/tidb-data"
server_configs: {}
pd_servers:
  - host: 10.0.1.4
  - host: 10.0.1.5
  - host: 10.0.1.6
tidb_servers:
  - host: 10.0.1.7
  - host: 10.0.1.8
  - host: 10.0.1.9
tikv_servers:
  - host: 10.0.1.1
    10.0.1.2
    3
servers:
  host: 10.0.1.4
```

なんとなく書き方は似ている
Kubernetes の方は
IP アドレスを意識しないで OK

Kubernetes で TiDB を構築する際のポイント

Kubernetes で TiDB を構築する際のポイント

tidb-operator を使って tiup のように

TiDB のクラスターを自動で作成しよう💪

tidb-operator の導入方法

- ☑ tidb-operator は Helm でのインストールが提供されています
- ☑ 公式のドキュメントも Helm の手順が書かれています
- ☑ 現時点では CRD の定義が Helm Chart に含まれていないので、手動で作成してあげる必要があります
- ☑ Helm で tidb-operator を入れると実体として tidb-controller-manager と tidb-scheduler が稼働することになります

Kubernetes で TiDB を運用する際のポイント

Affinity/AntiAffinity

- ☑ 各コンポーネント(PD/TiDB/TiKV,etc)がきちんとバラけて配置されるように PodAntiAffinity を設定しましょう
- ☑ 逆に戦略的に同居させたい場合は PodAffinity を使ってもいいかもしれません

TidbCluster.yaml

```
...
spec:
  pd:
    baseImage: pingcap/pd
    replicas: 3
    requests:
      storage: "10Gi"
    affinity:
      podAntiAffinity:
        preferredDuringSchedulingIgnoredDuringExecution:
          - weight: 10
            podAffinityTerm:
              labelSelector:
                matchLabels:
                  app.kubernetes.io/component: "pd"
              topologyKey: “region”
```

tidb-scheduler を使う

- ☑ Affinity の設定が面倒くさい！そんなアナタには tidb-scheduler に任せましょう
- ☑ tidb-scheduler が動いていても、明示的に使う設定にしなければ使われません
- ☑ 各コンポーネントをいい感じに分散して配置するように Kubernetes の scheduler を拡張したものになります
- ☑ 例えば PD の場合こんな感じで配置されます

PD のreplica数	最大で 1 ノードに PD をいくつ配置するのか
1	1
2	1
3	1
4	1
5	2

ノード障害時に
過半数以上死なないように
配置してくれる

tidb-scheduler を使う

- TiKV の場合も同じようにいい感じに分散してくれますが、ノードが 3 つ以上の場合のみ正常に動くようです (3より少ないとスケジューリングしてくれなかった)

TiKV のreplica数	各ノードにどのように配置されるのか(3ノードの場合)
3	1,1,1
4	2,1,1
5	2,2,1
6	2,2,2
7	3,2,2
8	3,3,2

tidb-scheduler を使う

☑ ちなみに 4 ノードになるとどうなるかというと、こうなります

TiKV のreplica数	各ノードにどのように配置されるのか(3ノードの場合)
3	1,1,1,0
4	1,1,1,1
5	2,1,1,1
6	2,2,1,1
7	2,2,2,1
8	2,2,2,2

tidb-scheduler を使う

☑ ちなみにちなみに、3 ノードから途中で 4 に増やすとこうなります

TiKV のreplica数	各ノードにどのように配置されるのか(3から4ノードの場合)
3	1,1,1
4	2,1,1
5	2,2,1
6	2,2,2
7	2,2,2,1
8	2,2,2,2

ここで新しいノードを
増やすと
新しいノードで作られる

tidb-scheduler を使う

- tidb-scheduler を使うためには tidb-operator をインストールする時に Helm で「--set scheduler.create=true」を指定します
  - ただしデフォルトで create=true なので、もし必要なければ create=false で明示的に消してあげる必要があります
- TidbCluster の定義内の spec.schedulerName を設定してあげることで一括で設定することができます

TidbCluster.yaml

```
apiVersion: pingcap.com/v1alpha1
kind: TidbCluster
metadata:
  name: sample
spec:
  version: v5.2.2
  schedulerName: tidb-scheduler
  pd:
   ...
```

tidb-scheduler を使う

- ☑ tidb-scheduler を使うためには tidb-operator をインストールする時に Helm で「--set scheduler.create=true」を指定します
  - ☑ ただしデフォルトで create=true なので、もし必要なければ create=false で明示的に消して
- ☑ TidbCluster の定義 で設定することがて

```
apiVersion:
kind: TidbC
metadata:
  name: sam
spec:
  version:
  scheduler
  pd:
    ...
```

```
apiVersion: pingcap.com/v1alpha1
kind: TidbCluster
metadata:
  name: sample
spec:
  version: v5.2.2
 pd:
    schedulerName: tidb-scheduler
    ...
  tikv:
    schedulerName: default-scheduler
    ...
  tidb:
    schedulerName: tidb-scheduler
    ...
```

コンポーネントで
個別に変更することも
可能

tidb-scheduler を使う

- デフォルトでは分散のトポロジーのキーは「kubernetes.io/hostname」のラベルによって行われます
- このキーは TidbCluster の Annotation で任意に変更することが可能です

TidbCluster.yaml

```
apiVersion: pingcap.com/v1alpha1
kind: TidbCluster
metadata:
  name: sample
  annotations:
    pingcap.com/ha-topology-key: topology.kubernetes.io/zone
spec:
  version: v5.2.2
 pd:
   ...
```

例えば zone レベルで
分散させたい時

Advanced StatefulSet を使う

- もともと Kubernetes では StatefulSet という仕組みがあり、tidb-operator も基本的にはこの StatefulSet を使って tidb や tikv のコンポーネントを作成します
- StatefulSet は Pod を指定された数だけシーケンシャルに作成します (mypod-0, mypod-1 ,,, mypod-N)
- Pod が増える時は Pod の名前の末尾の数字が増えていき、Pod が減る時は末尾の数字が大きいものから減っていきます

Advanced StatefulSet を使う

- ☑ StatefulSet だと、例えばこんな時困ることになります

Advanced StatefulSet を使う

- ☑ つまり、通常の StatefulSet だとこういう状態にすることはできません

- ☑ 特定の番号の Pod (今回の例だと 1) だけ欠番にして通常通り運用したい・・・
- ☑ そんな願いを叶えてくれるのが Advanced StatefulSet になります

Advanced StatefulSet を使う

- ☑ Advanced StatefulSet は advanced-statefulset-controller を動かすことで使用可能になります
- ☑ tidb-operator をインストールする時に Helm の設定を変更することで追加でインストールすることができます(デフォルトではインストールされない)
- ☑ もしくは下記のリポジトリから手動でインストールすることもできます

  https://github.com/pingcap/advanced-statefulset/blob/master/manifests/deployment.yaml

- ☑ 別途 CRD も必要になりますので、kubectl で作成しておきます

```
$ kubectl apply -f https://raw.githubusercontent.com/pingcap/tidb-operator/master/manifests/advanced-statefulset-crd.v1.yaml
```

Advanced StatefulSet を使う

- Advanced StatefulSet は「kind: StatefulSet」はそのままで、apiVersionを「apps.pingcap.com/v1alpha1」にすることで使うことができます
- それ以外の部分は既存(apps/v1)の StatefulSet と全く同じです

sample-asts.yaml

```yaml
apiVersion: apps.pingcap.com/v1alpha1
kind: StatefulSet
metadata:
  name: sample-asts
spec:
  selector:
    matchLabels:
      app: nginx
  serviceName: "nginx"
  replicas: 4
...
```

Advanced StatefulSet を使う

- ☑ Advanced StatefulSet は消したい Pod を annotation で記述します

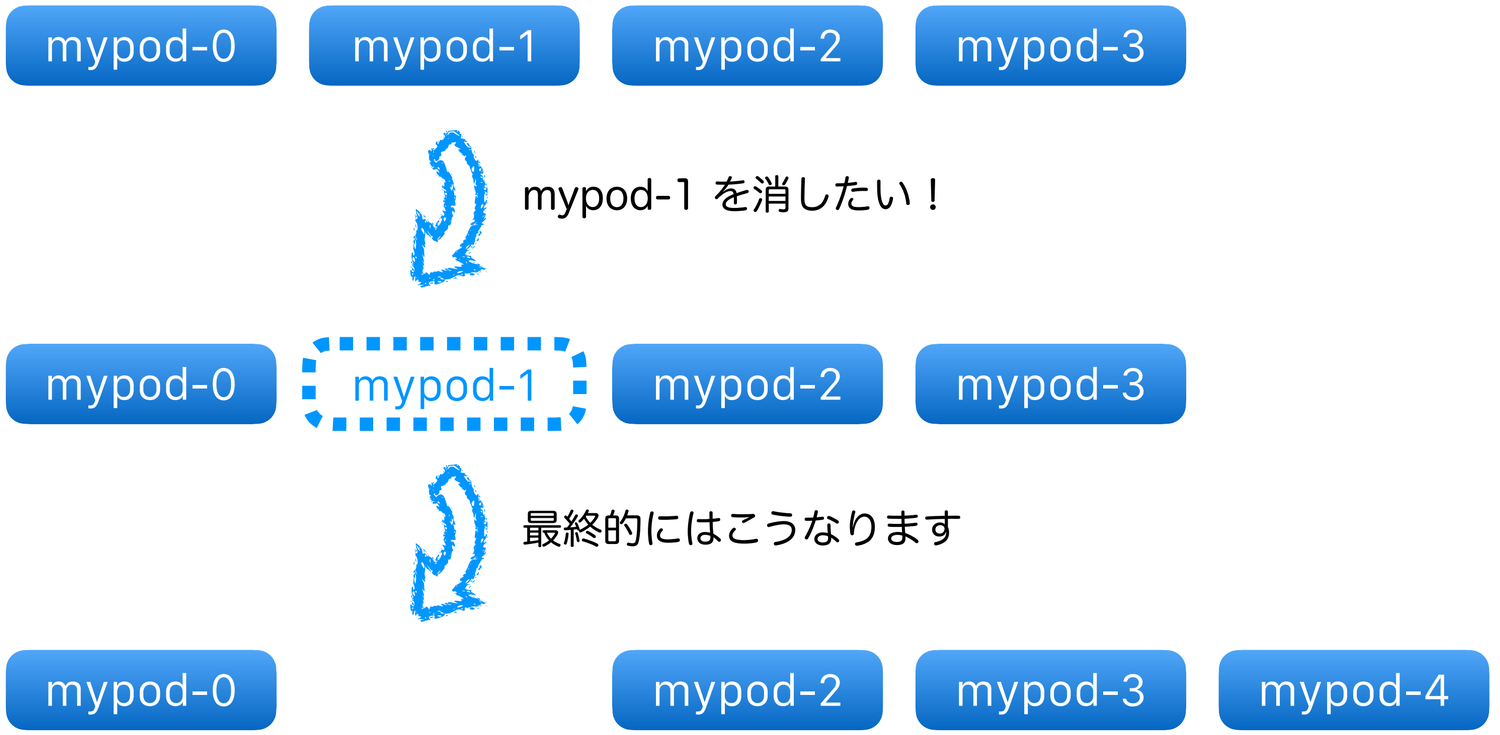
sample-asts.yaml

```yaml
apiVersion: apps.pingcap.com/v1alpha1
kind: StatefulSet
metadata:
  name: sample-asts
  annotations:
    delete-slots: '[1]'
spec:
...
```

これで
mypod-1 は
自動的に消えます

Advanced StatefulSet を使う

- ☑ Advanced StatefulSet は消したい Pod を annotation で記述します



- ☑ 1 を歯抜けの状態にしつつ、合計 Pod の数を合わせる為に 4 が作られます

Advanced StatefulSet を使う

- tidb-operator は通常は StatefulSet を使ってコンポーネントを作るので、Advanced StatefulSet を使ってコンポーネントを作成するようにしてあげる必要があります
- tidb-operator は FeatureGate の機能を持っており、Advanced StatefulSet を使う/使わないを切り替えることができます
- Advanced StatefulSet を使うようにするには tidb-operator で 「--features=AdvancedStatefulSet=true」を引数に指定してあげる必要があります
- FeatureGate で有効にすれば、あとは通常通り TidbCluster のリソースを作るだけ！

Advanced StatefulSet を使う

- tidb-operator 経由で Advanced StatefulSet を作成した場合は直接 Advanced StatefulSet に対して Annotation を追加するのではなくて、TidbCluster リソースに対して Annotation を付与して制御します
- Advanced StatefulSet は tidb-operator によって管理されているので直接操作すると不整合になり、tidb-operator が正常に状態を管理できなくなります

TidbCluster.yaml

```yaml
apiVersion: pingcap.com/v1alpha1
kind: TidbCluster
metadata:
  name: sample
  annotations:
    tikv.tidb.pingcap.com/delete-slots: '[0]'
spec:
  version: v5.2.2
 pd:
   ...
```

Annotation を追加するのは TidbClusterの方

Advanced StatefulSet を使う

- TidbCluster に対して特定の Annotation を付与してあげれば tidb-operator が各 Advanced StatefulSet に対して「delete-slots」の Annotation を付与してくれます

付与する Annotation	対象コンポーネント
pd.tidb.pingcap.com/delete-slots	PD
tidb.tidb.pingcap.com/delete-slots	TiDB
tikv.tidb.pingcap.com/delete-slots	TiKV
tiflash.tidb.pingcap.com/delete-slots	TiFlash
dm-master.tidb.pingcap.com/delete-slots	DM Master
dm-worker.tidb.pingcap.com/delete-slots	DM Worker

Autoscaling したい

- 負荷に応じて Pod の数を増やしてスケールさせたい場合も多々あると思います
- Kubernetes は HorizontalPodAutoscaler (HPA) の仕組みがあり、負荷に応じて Pod の数を増減させる仕組みが備わっています
- 作成された StatefulSet(Advanced 含む)に対して HPA を設定することはできますが、前述の通り tidb-operator によって管理されているので tidb-operator 経由ではない操作によってリソースが書き換えられてしまうことは避けるべきです
- ではどうすれば？

Autoscaling したい

- そのために tidb-operator では TidbClusterAutoScaler というリソースで HPA のような挙動を実現する仕組みが備わっています (まだ成熟度的には alpha なので使う際は慎重に・・・)
- こちらの機能は Advanced StatefulSet と同じように FeatureGate によって有効/無効を切り替えられます(デフォルトは無効)
- TidbClusterAutoScaler を使うようにするには tidb-operator で 「--features=AutoScaling=true」を引数に指定してあげる必要があります
- また、リソース取得の為に TidbMonitor も同時に作成しておく必要があります (prometheus 等が一式自動で投入される)

Autoscaling したい

- TidbClusterAutoscaler は現時点では CPU のメトリクスを元にスケールする方法しかサポートしていません
- また TiKV と TiDB のコンポーネントしか対応していないです
- 下記のように定義してあげます

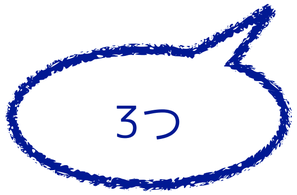
TidbClusterAutoScaler.yaml

```
apiVersion: pingcap.com/v1alpha1
kind: TidbClusterAutoScaler
metadata:
  name: auto-scaling-tikv
spec:
  cluster:
    name: sample
  tikv:
    rules:
      cpu:
        max_threshold: 0.8
        min_threshold: 0.2
  tidb:
...
```

対象のクラスター
名を指定して
threshold を設定します

全ての Pod の CPU 使用率が
0.8(80%) 以上なら増えて
0.2(20%) 以下なら減る

Kubernetes で TiDB を運用する際のポイント



Kubernetes で TiDB を運用する際のポイント

Affinity や tidb-scheduler で賢く Pod をスケジューリングしよう💪

Advanced StatefulSet を使ってより柔軟に StatefulSet を管理しよう

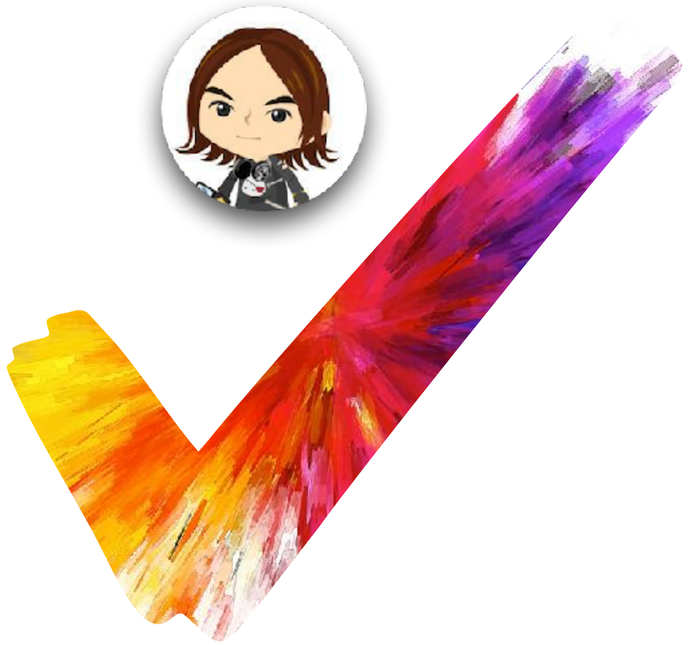
TidbClusterAutoScaler で Autoscaling も忘れずに！

3つ



本日のまとめ

まとめ



- ☑ TiDB は MySQL 互換の NewSQL で HTAP もサポートしているオープンソースのソフトウェア
- ☑ TiDB はクラウドネイティブな志向で作成されているので Kubernetes との相性がとても良い
- ☑ SQL とストレージが分離されているので、それぞれのコンポーネント毎にスケールすることが可能で効率がよい
- ☑ スケールは無停止で可能、手動シャーディングも必要なし
- ☑ 便利な tiup は Kubernetes に対して利用することができないが、tidb-operator を使えば様々な自動化の恩恵を受けることができる (クラスター構築、メトリクス環境、オートスケール、全て 1 発)

やってみよう！

Kubernetes で TiDB を動かしてみたくなりましたか？

是非触ってみてください！💪

Thank You!!

RDBMSとNoSQLのメリットを併せ持つクラウドネイティブなNewSQLデータベース
「TiDB」をKubernetesで動かしてみよう！

Executive Technical Advisor / Makoto Hasegawa

PingCAP