MySQL Fabric＆OpenStackを使った柔軟な拡張性

- MySQL FabricによるHAと拡張性、OpenStackとの連携

2015/02/23

Shinya Sugiyama / 杉山真也

MySQL Principal Sales Consult, MySQL Global Business Unit

SAFE HARBOR STATEMENT

以下の事項は、弊社の一般的な製品の方向性に関する概要を説明するものです。
また、情報提供を唯一の目的とするものであり、いかなる契約にも組み込むことはできません。
以下の事項は、マテリアルやコード、機能を提供することをコミットメントするものではない為、
購買決定を行う際の判断材料になさらないで下さい。

オラクル製品に関して記載されている機能の開発、リリースおよび時期については、
弊社の裁量により決定されます。

Agenda

1 MySQL Fabric概要

- MySQL Fabric基本機能のご紹介
- MySQL Fabricによる読み込み・書き込みのスケール(デモ)

2 OpenStackとMySQLの連携

- OpenStack概要
- OpenStackリポジトリーとしてのMySQL
- OpenStackにおけるDBaaS (Trove)概要
- MySQL FabricとOpenStack NovaによるDBのプロビジョニング

3 参考情報：MySQL Fabricサポート

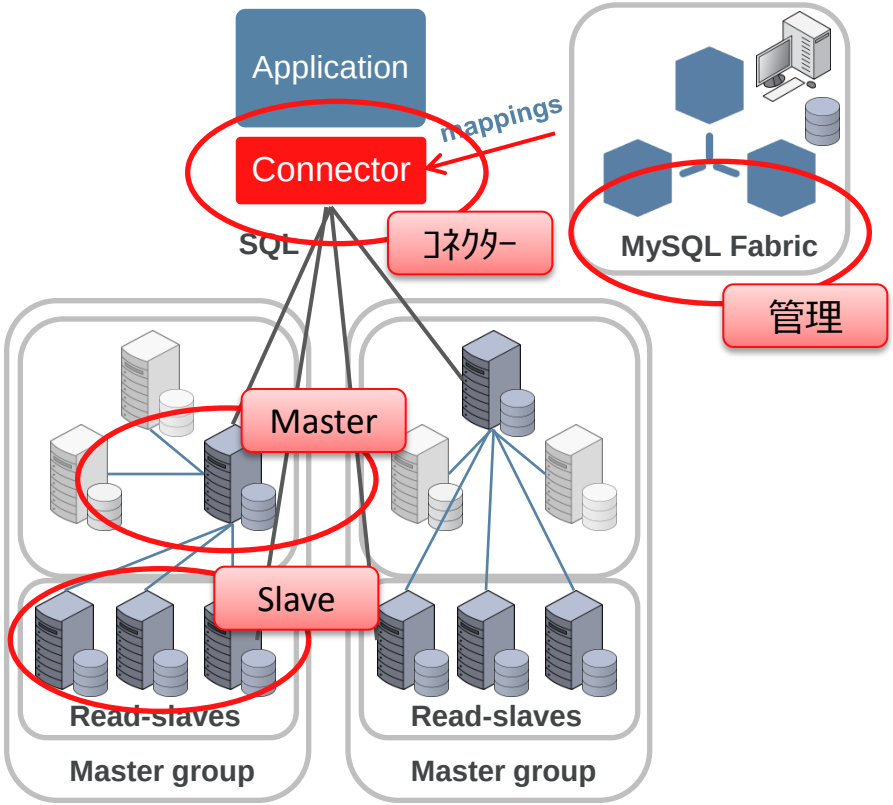
MySQL Fabric概要

高可用性とデータ
読み・書き込みシャーディング

MySQL Fabricとは?

- MySQLサーバー群を管理する
  統合型のフレームワーク
- 高可用性とデータ・シャーディング
  による拡張性を実現する事が可能
- OpenStack(NOVA)との連携可能
- MySQL Utilitiesの一部として提供
  (2014-05-27　ver.1.4.3～)
  - GTIDモードによるレプリケーション
    機能を活用している
    (MySQL 5.6.5以降で使用可能)

参照：MySQL Fabric の特徴と利点
http://www-jp.mysql.com/products/enterprise/fabric/features.html



MySQL Fabric configurationファイルの設置

Platform	Package	Location
Microsoft Windows	mysql-utilities-1.5.3-win32.msi	
Ubuntu Linux 12.04	mysql-utilities_1.5.3-1ubuntu12.04_all.deb	/etc/mysql/fabric.cfg
Debian Linux 6.0	mysql-utilities_1.5.3-1debian6.0_all.deb	/etc/mysql/fabric.cfg
Red Hat Enterprise Linux 6 / Oracle Linux 6	mysql-utilities-1.4.3-1.el6.noarch.rpm	/etc/mysql/fabric.cfg

参照：8.6.1. MySQL Fabric configuration file location
http://dev.mysql.com/doc/mysql-utilities/1.5/en/fabric-cfgref-path.html

例) /etc/mysql/fabric.cfg

```
[admin@Fabric01 ~]$ cat /etc/mysql/fabric.cfg
[DEFAULT]
prefix =
sysconfdir = /etc
logdir = /home/mysql/fabric

[storage]
address = localhost:63300
user = fabric
password = fabric
database = fabric
auth_plugin = mysql_native_password
connection_timeout = 6
connection_attempts = 6
connection_delay = 1

[servers]
user = root
password = root
unreachable_timeout = 5

[protocol.xmlrpc]
address = localhost:32274
#address = 192.168.56.110:32274
threads = 5
```

This section contains information that the MySQL Fabric node uses for the connection to the backing store.
(構成データ保存用データベース)

This section contains information that MySQL Fabric uses to connect to the servers being managed.
(Fabricが管理対象サーバーへ接続する為に利用)

This section contains information about how the client connects to a MySQL Fabric node and configuration parameters for the XML-RPC protocol on the server.
(Fabric管理サーバーへXML-RPCで接続する為に利用)

詳細は此方を参照下さい: 8.6. Configuring MySQL Fabric

http://dev.mysql.com/doc/mysql-utilities/1.5/en/fabric-cfgref.html

MySQL Fabricと管理データベース

```
# Configure Fabric data node (63300)
# mysqlfabric help commands
mysqlfabric manage setup
mysqlshow -ufabric -pfabric -h127.0.0.1 -P63300 fabric
```

管理データベース作成
mysqlfabric manage setup

MySQL Backing Store

```
mysql> show tables from fabric;
+-------------------+
| Tables_in_fabric  |
+-------------------+
| checkpoints       |
| error_log         |
| group_replication |
| group_view        |
| groups            |
| log               |
| machines          |
| permissions       |
| proc_view         |
| providers         |
| role_permissions  |
| roles             |
| servers           |
| shard_maps        |
| shard_ranges      |
| shard_tables      |
| shards            |
| user_roles        |
| users             |
+-------------------+
19 rows in set (0.00 sec)
```

```
mysql> select * from group_view;
+----------+---------------+--------------+
| group_id | promote_count | demote_count |
+----------+---------------+--------------+
| global   |             1 |            0 |
| shard1   |             1 |            0 |
| shard2   |             1 |            0 |
+----------+---------------+--------------+
3 rows in set (0.08 sec)

mysql> select * from servers;
+--------------------------------------+-----------------+------+--------+--------+----------+
| server_uuid                          | server_address  | mode | status | weight | group_id |
+--------------------------------------+-----------------+------+--------+--------+----------+
| 2dc929e4-b899-11e4-bbed-080027d65c57 | 127.0.0.1:63301 |    3 |      3 |      1 | global   |
| 2e7510f9-b899-11e4-bbed-080027d65c57 | 127.0.0.1:63302 |    1 |      2 |      1 | global   |
| 2eefc9f6-b899-11e4-bbed-080027d65c57 | 127.0.0.1:63303 |    1 |      2 |      1 | global   |
| 2f8c8c02-b899-11e4-bbed-080027d65c57 | 127.0.0.1:63304 |    3 |      3 |      1 | shard1   |
| 30359409-b899-11e4-bbed-080027d65c57 | 127.0.0.1:63305 |    1 |      2 |      1 | shard1   |
| 30ccd42e-b899-11e4-bbed-080027d65c57 | 127.0.0.1:63306 |    1 |      2 |      1 | shard1   |
| 316bea5a-b899-11e4-bbed-080027d65c57 | 127.0.0.1:63307 |    3 |      3 |      1 | shard2   |
| 31fe0dbc-b899-11e4-bbed-080027d65c57 | 127.0.0.1:63308 |    1 |      2 |      1 | shard2   |
| 32a746f9-b899-11e4-bbed-080027d65c57 | 127.0.0.1:63309 |    1 |      2 |      1 | shard2   |
+--------------------------------------+-----------------+------+--------+--------+----------+
```

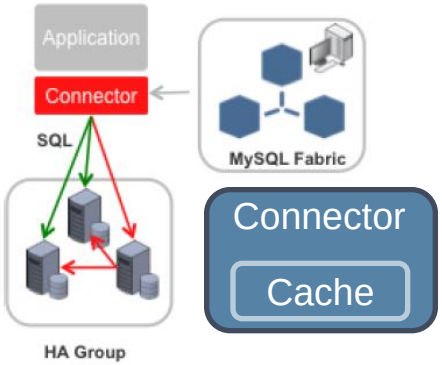
Fabric対応Connector

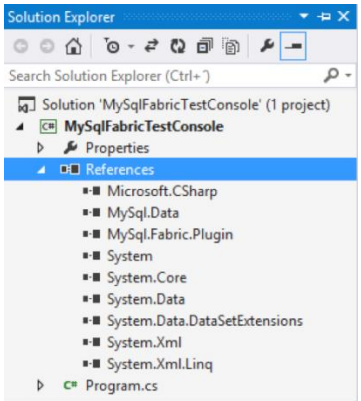
Python	Python requires version 1.2.0
Java	Connector/J 5.1.30 and later
PHP	version 1.6 and later
.NET	Connector/Net 6.9.4 or newer supports
C	Connector/C 6.2.0 or newer *1

*1 Connector/CはまだLab版になります。

```
import mysql.connector
from mysql.connector import fabric
```

参照：Using Connector/Python with MySQL Fabric





参照：6.7 MySQL Fabric Support

可用性グループの作成とサーバーの追加

「フェールオーバー」と通信経路の自動再構成による高可用性・「読み込みシャーディング」による拡張性

```
# Create group "global" and add my MySQL instances
mysqlfabric group create global
mysqlfabric group add global 127.0.0.1:63301
mysqlfabric group add global 127.0.0.1:63302
mysqlfabric group add global 127.0.0.1:63303

# Control state of Fabric:
mysqlfabric group lookup_servers global
mysqlfabric group health global

# Promote one of your servers at master, <UUID> is optional.
mysqlfabric group promote global

# Activate automatic failover
mysqlfabric group activate global
```

グループの定義(作成)
mysqlfabric group create **グループ名**

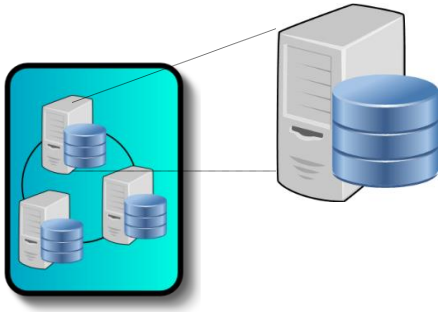
グループにサーバーを追加
mysqlfabric group add **グループ名** **サーバ名**

サーバーを一台Primary(master)に昇格
mysqlfabric group promote **グループ名**

グループを有効にしてHA機能を稼働
mysqlfabric group activate **グループ名**

High-Availability Group Concept

- **抽象概念**
  - サーバーセット
  - サーバー属性
- **コネクター属性**
  - 接続情報
  - モード：読み取り専用、読み書き
  - 重み付け: 負荷分散
- **管理属性**
  - 状態:サーバの状態/役割



```
1  import mysql.connector
2  from mysql.connector import fabric
3
4  import time
5
6  def add_employee(conn, emp_no, first_name, last_name):
7      try:
8          conn.set_property(group="global", mode=fabric.MODE_READWRITE)
9          cur = conn.cursor()
10         cur.execute(
11             "INSERT INTO employees VALUES (%s, %s, %s)",
12             (emp_no, first_name, last_name)
13         )
```

State: Primary
Mode: Read-Write
Host: localhost:13008

Master 書き込み

Slave 読み込み

重み付け

server_uuid	address	status	mode	weight
83f2dd4f-46a1-11e4-94c4-e82aea9348c9	localhost:1300X	SECONDARY	READ_ONLY	1.0
8426dd92-46a1-11e4-94c4-e82aea9348c9	localhost:13008	PRIMARY	READ_WRITE	1.0

MySQL Replication & MySQL Fabric HA

& how this effects failover

- MySQLのレプリケーションは、HAグループで使用される初期実装です
  - PRIMARY = レプリケーションのマスターは全ての書き込みを受け取る
- Failover
  1) Fabricがマスターにおける障害を検知
  2) スレーブからマスター候補を選択しマスターに昇格します
  3) 変更を更新し状況を保存
  4) ファブリック対応のコネクタに状態変化をプッシュ

MySQL Fabric 基本機能Short Demo

1) Fabricグループ構成の確認

mysqlfabric group lookup_servers global

2) FabricグループにSlaveの追加 (対応例:参照増加)

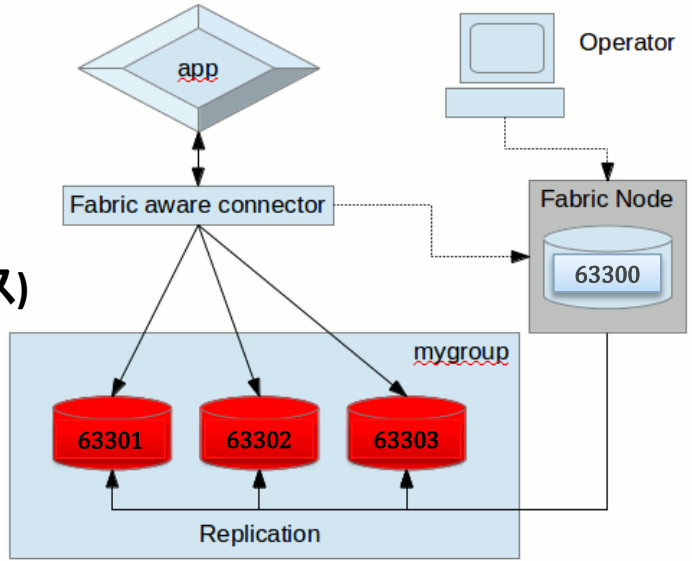
mysqlfabric group add global 127.0.0.1:63304
mysqlfabric group add global 127.0.0.1:63305

3) 特定のSlaveをマスターに昇格 (対応例:メンテナンス)

mysqlfabric group promote global --slave_id=xxx

4) 疑似障害と可用性確認 (対応例:マスター障害)

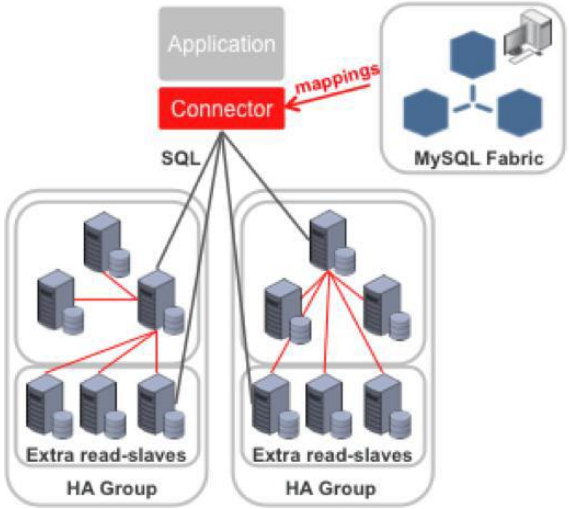
kill <pid>



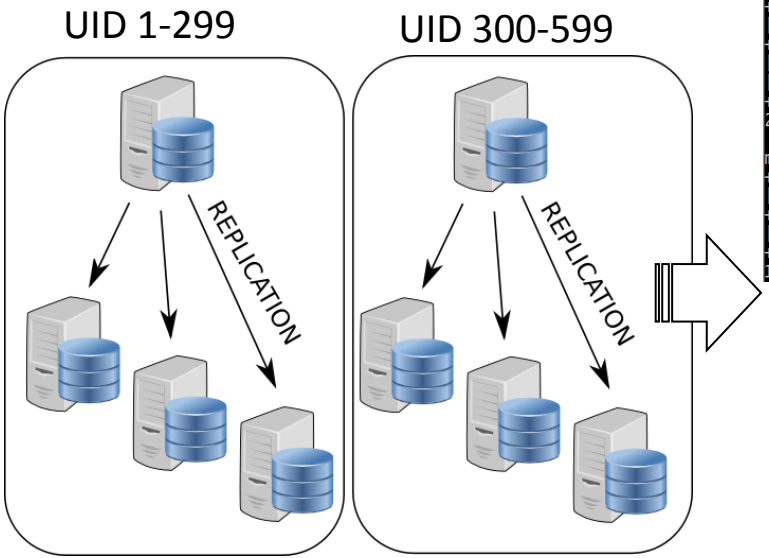
MySQL Fabricによる書き込みのスケールアウト

Sharding with Fabric

- 書き込みスケーラビリティ　より多くの書き込みを処理することが可能
- 大規模なデータセット　大き過ぎるデータベース/単一サーバーに収まらないデータ
- 性能改善　小さなインデックスサイズ/ワーキングセットに分割



Single Master in each HA Group.




```
mysql> select * from shard_ranges;
+-----------------+-------------+----------+
| shard_mapping_id | lower_bound | shard_id |
+-----------------+-------------+----------+
|                1 | 1           |        1 |
|                1 | 300         |        2 |
+-----------------+-------------+----------+
2 rows in set (0.00 sec)

mysql> select * from shard_tables;
+-----------------+---------------+-------------+
| shard_mapping_id | table_name     | column_name |
+-----------------+---------------+-------------+
|                1 | test.employees | emp_no      |
+-----------------+---------------+-------------+
1 row in set (0.00 sec)
```

キー範囲分割
UID 600-899
UID 900-1199
UID 1200-1499
……

可用性グループと書き込み処理の分散

アプリケーションでの分割キー(Range, Hash)　/　シャード再構成, シャード全体の更新も可能

```
mysqlfabric group create shard1
mysqlfabric group add shard1 127.0.0.1:63304
mysqlfabric group add shard1 127.0.0.1:63305
mysqlfabric group add shard1 127.0.0.1:63306
mysqlfabric group promote shard1
mysqlfabric group activate shard1
```

```
mysqlfabric group create shard2
mysqlfabric group add shard2 127.0.0.1:63307
mysqlfabric group add shard2 127.0.0.1:63308
mysqlfabric group add shard2 127.0.0.1:63309
mysqlfabric group promote shard2
mysqlfabric group activate shard2
```

1) **グループの定義**
2) **サーバ追加**
3) **マスター設定**
4) **HAグループを有効にする**

シャーディング定義にテーブルを追加

mysqlfabric sharding add_table shard_mapping_id table_name column_name

シャーディングを追加

mysqlfabric sharding add_shard shard_mapping_id groupid_lb_list

```
# Add sharding to employee
mysqlfabric sharding add_table 1 test.employees emp_no
mysqlfabric sharding add_shard 1 shard1/1,shard2/300 --state=enabled
```

Connector API: Shard Specific Query

- Provide **tables** in query
  - Property**:** tables
  - Fabric will compute map

ここでは、employeesとtitlesテーブルが対象

- Provide **sharding key**
  - Property**:** key
  - Fabric will compute shard

Keyは, emp_no

```
conn.set_property(tables=['employees.employees','employees.titles'] key=emp_no)
cur = conn.cursor()
cur.execute("INSERT INTO employees VALUES(%s,%s,%s)",(emp_no,first_name,last_name))
cur.execute("INSERT INTO titles(emp_no, title, from_date)"
            " VALUES (%s, %s, CURDATE())",(emp_no, 'Intern'));
conn.commit()
```

Connector API: Global Update

- Provide tables in query
  - **Property:** tables
  - Fabric will compute map
  - (Likely to not be needed)

- Set global scope
  - **Property:** scope
  - Query goes to global group

範囲の指定

```
conn.set_property(tables=['employees.titles'], scope='GLOBAL')
cur = conn.cursor()
cur.execute("ALTER TABLE employees.titles ADD nickname VARCHAR(64)")
```

MySQL Fabric 書き込みSharding Short Demo

1) Fabricグループ構成の確認

2) FabricにSharding Groupを追加 (対応例:データ増加による遅延)

mysqlfabric group create shard#

mysqlfabric group add shard# サーバー

mysqlfabric sharding add_table 1 test.employees emp_no

mysqlfabric sharding add_shard 1 shard1/1,shard2/300

3) Fabricグループ構成の確認

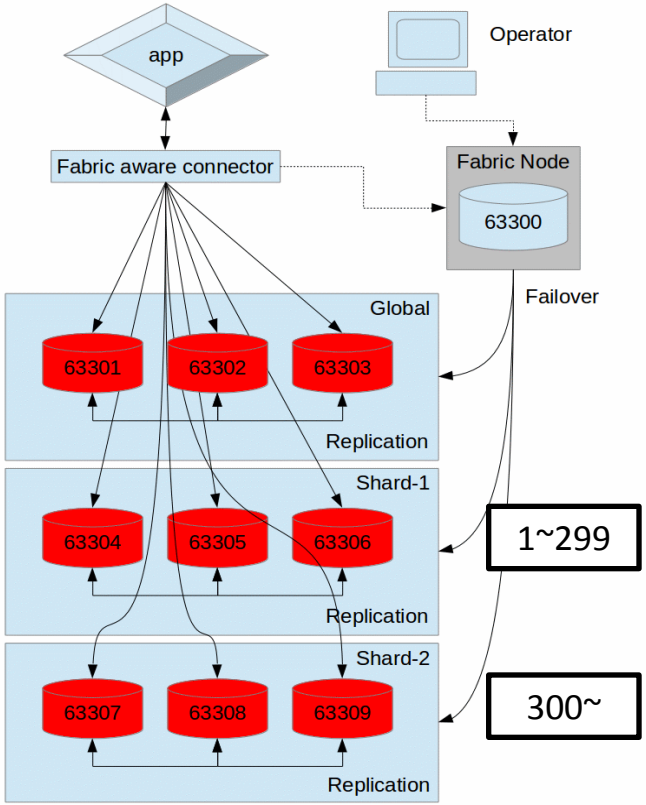
shard1 (Server:63304,63305,63306)

shard2 (Server:63307,63308,63309) 300以上

mysqlfabric dump sharding_information

mysqlfabric group lookup_servers グループ名

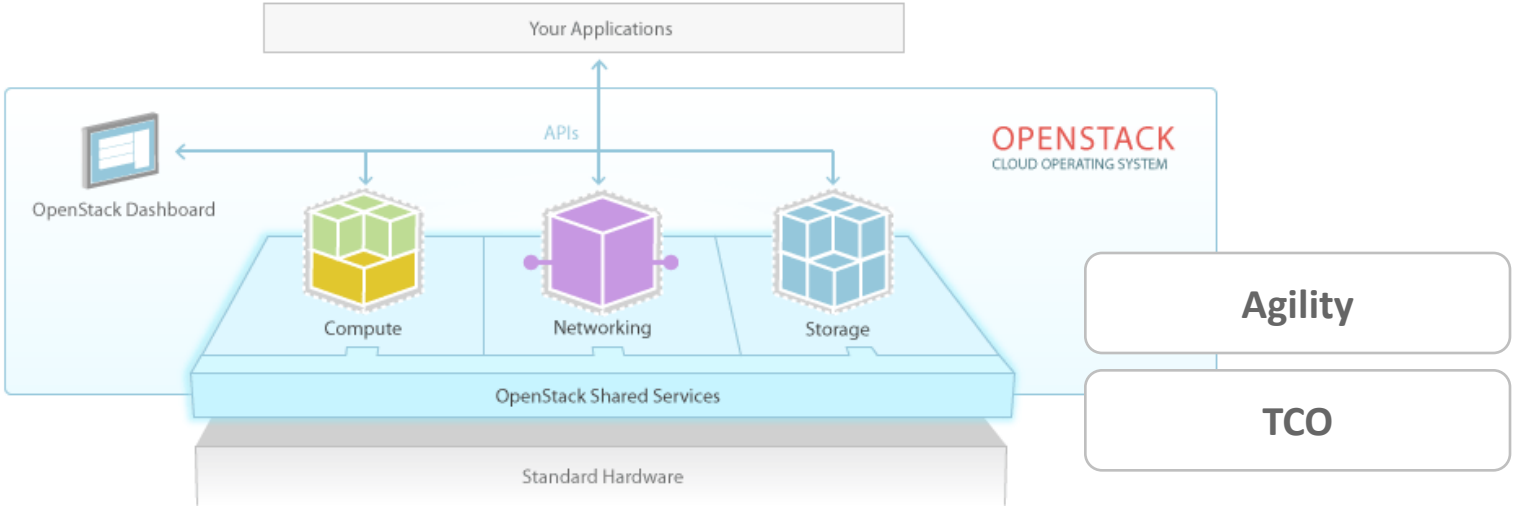
4) 書き込みデータの分散状況確認



OpenStackとMySQLの連携

OpenStackとは?

OpenStackは2010年にRackspaceとNASAによって発足され,
大規模・マルチテナントクラウド環境への導入を目的としたオープンソース
クラウド・オペレーティングシステムです。コンピュート・ストレージ・ネットワーク・
アイデンティティ管理・ オーケストレーション等を提供する分散サービス群で
構成されています。現在は、200社以上が参加して、Oracleも参加しています。



OpenStack Foundation

A much wider ecosystem …

Platinum Members (8)

Corporate Sponsors (63)

and more …

Gold Members (19)

and more …

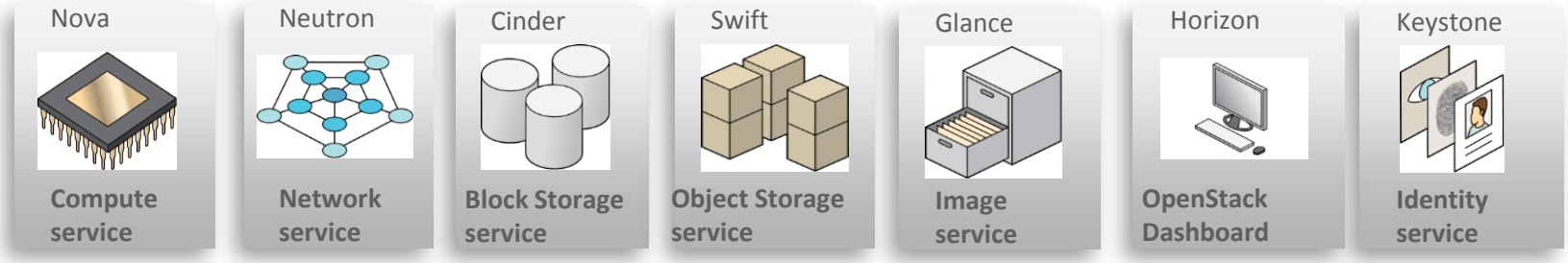
Supporting Organizations (248)

and more …

Oracle OpenStack for Oracle Linux & VM 提供開始

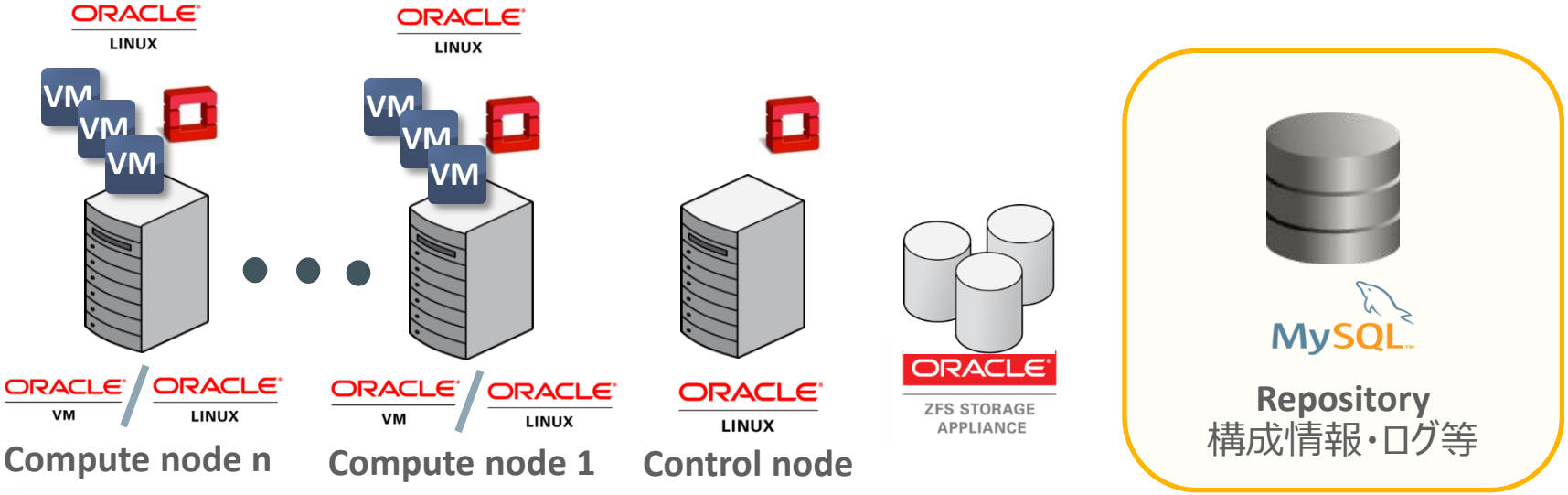
Press Release: 2014 10 08

- Oracle Linux と Oracle VMによるOpenStackクラウド本番環境の構築と管理
  - 高速、信頼性、スケーラブル、安全性
  - Linux, Windows, Solaris上で本番環境のワークロードを実現
- 無料ダウンロードおよび利用可能
- 「Oracle Linux and Oracle VM Premier Subscription」
  「Oracle Premier Support for Systems」が追加費用なしでサポートサービス提供
- OpenStackリリースIcehouseベースで以下を提供



詳細 : http://www.oracle.com/jp/technologies/linux/overview/index.html

MySQLは構成管理データベースとして広く利用されている



MySQL database

抜粋： OpenStack.org

Controller setup
Node setup

Most OpenStack services require a database to store information. These examples use a MySQL database that runs on the controller node. You must install the MySQL database on the controller node. You must install MySQL client software on any additional nodes that access MySQL.

openstack CLOUD SOFTWARE

Highly available MySQL

SIDEBAR ◂ PREV | UP | NEXT ▸

CONTENTS SEARCH

OPENSTACK MANUALS ▸ OPENSTACK HIGH AVAILABILITY GUIDE - CURRENT

- Preface
- 1. Introduction to OpenStack High Availability
- HA using active/passive
  - 2. The Pacemaker cluster stack
  - 3. Cloud controller cluster stack
    - Highly available MySQL
    - Highly available RabbitMQ
  - 4. API node cluster stack
  - 5. Network controller cluster stack
- HA using active/active
- A. Community support

Highly available MySQL

OpenStack HAマニュアルにMySQL DRBD構成詳細が記載されています

Configure DRBD
Creating a file system
Prepare MySQL for Pacemaker high availability
Add MySQL resources to Pacemaker
Configure OpenStack services for highly available MySQL

MySQL is the default database server used by many OpenStack services. Making the MySQL service highly available involves

- Configure a DRBD device for use by MySQL,
- Configure MySQL to use a data directory residing on that DRBD device,
- Select and assign a virtual IP address (VIP) that can freely float between cluster nodes,
- Configure MySQL to listen on that IP address,
- Manage all resources, including the MySQL daemon itself, with the Pacemaker cluster manager.

Note

MySQL/Galera is an alternative method of configuring MySQL for high availability. It is likely to become the preferred method of achieving MySQL high availability once it has sufficiently matured. At the time of writing, however, the Pacemaker/DRBD based approach remains the recommended one for OpenStack environments.

参照：Highly available MySQL http://docs.openstack.org/high-availability-guide/content/s-mysql.html

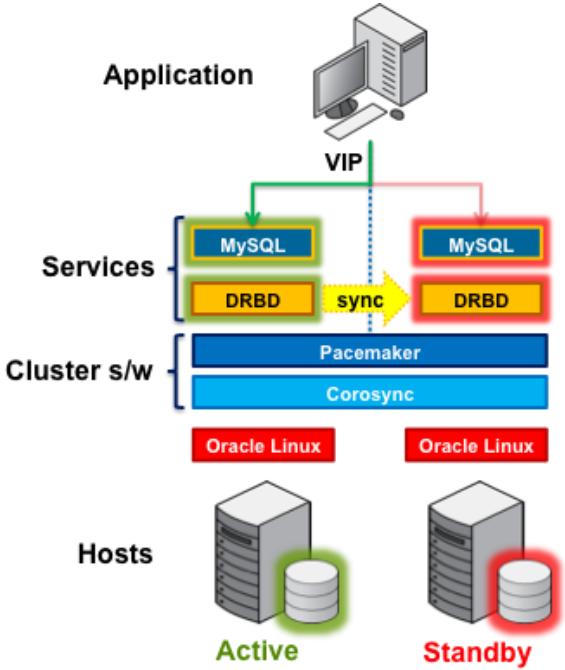
High Availability with DRBD

- 分散ストレージを利用するため、共有ディスクやSAN不要
- 同期レプリケーションによってデータを失うリスクを回避
- オープンソースで実績の多いソリューション
- OpenStack関連サービスはMySQLに設定されたVIPに接続

```
sql_connection = mysql://glancedbadmin:<password>@192.168.42.101/glance
```

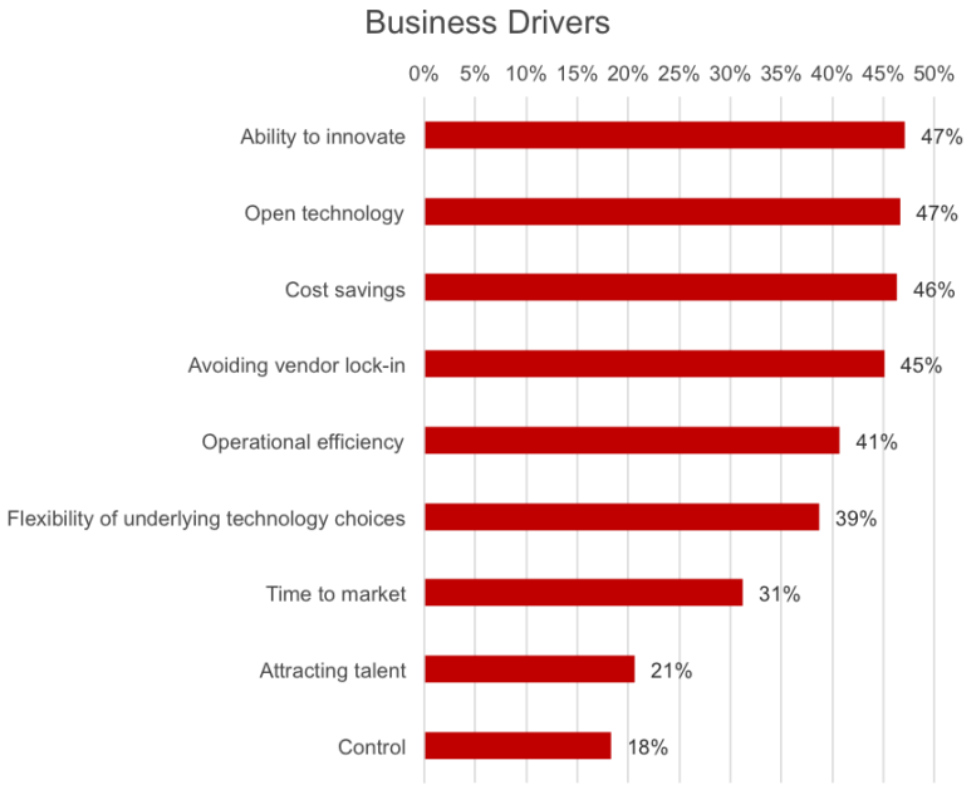
Oracle Linux ＋ DRBD Stack

- Oracle認定構成：フルスタックサポート
- Oracle Linux Unbreakable Enterprise Kernel R2にDRBDが統合
- Oracle Linux 6.2以上で使用可能
- クラスタリングとフェイルオーバーにて、PacemakerとCorosyncを使用
- 追加機能として、MySQL DTrace probesを利用可能



※ KeystoneやCeilometer等でデータが増加する環境では、**MySQL Cluster**で可用性とスケールアウトをサポート

OpenStack Survey (用途)



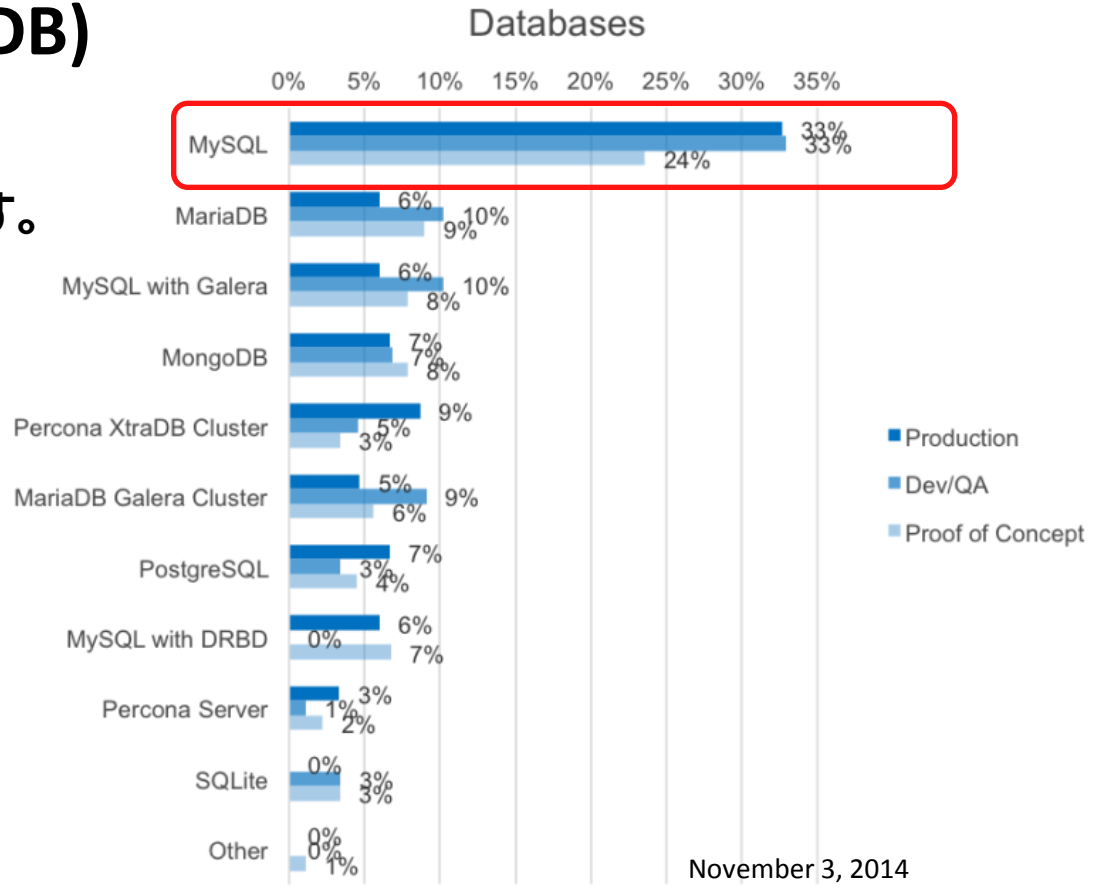
Top 5 Workloads

1. Web Services
2. QA Testing Environment
3. Databases
4. Up to the user
5. Continuous integration/Automated
6. Enterprise Applications
7. Benchmarks/Stress Testing
8. Management and Monitoring
9. Storage/Backup
10. Research

参照：OpenStack User Survey Insights: November 2014

OpenStack Survey (DB)

MySQLはOpenStackにおいても,
最も利用されているデータベースです。



参照:[OpenStack User Survey Insights: November 2014]

DBaaS(Data Base as a Service) in OpenStack

TroveはOpenStack上で”Database as a Service”を実現する為のコンポーネントです。
RDBMSだけでなくNoSQLにも対応しDBのスケールアウトを柔軟にサポート。

Mission Statement

The OpenStack Open Source Database as a Service Mission: To provide scalable and reliable Cloud Database as a Service provisioning functionality for both relational and non-relational database engines, and to continue to improve its fully-featured and extensible open source framework.

Description

Trove is Database as a Service for OpenStack. It's designed to run entirely on OpenStack, with the goal of allowing users to quickly and easily utilize the features of a relational or non-relational database without the burden of handling complex administrative tasks. Cloud users and database administrators can provision and manage multiple database instances as needed. Initially, the service will focus on providing resource isolation at high performance while automating complex administrative tasks including deployment, configuration, patching, backups, restores, and monitoring.

抜粋:https://wiki.openstack.org/wiki/Trove#Mission_Statement

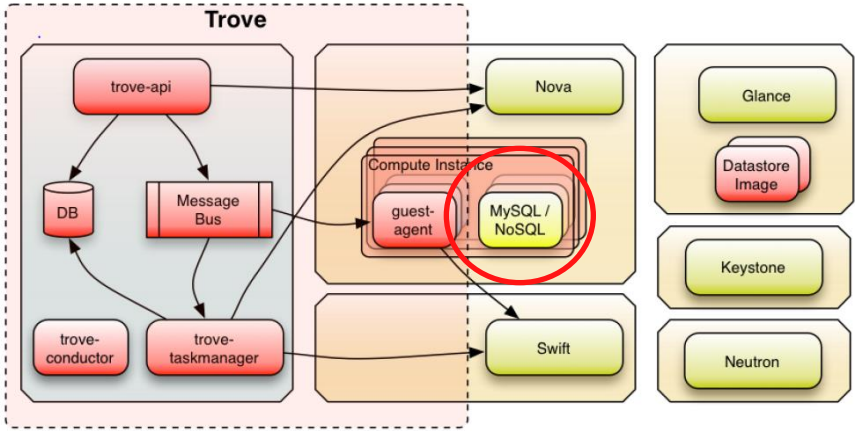
名称	リリース日
Austin	2010/10/21
Bexar	2011/2/3
Cactus	2011/4/15
Diablo	2011/9/22
Essex	2012/4/5
Folsom	2012/9/27
Grizzly	2013/4/4
Havana	2013/10/17
Icehouse	**2014/4/17**
Juno	2014/10/16

MySQL & OpenStack Trove

Community Edition Only

- Database as a ServiceをOpenStackで提供
- 複数データベースインスタンスのプロビジョニングと管理
- Single Tenant Database in Compute (Nova) Instance
- ユーザー/データベースの管理
- REST APIで全て管理する事が可能

- インスタンスの準備
- 複製作成
- インスタンスのサイズ変更
- ユーザーとDBの追加と権限の管理
- データベースのバックアップを管理
- データベース設定変更 (Group)



Provisioning by Trove

Community Edition Only

Trove用管理DB

```
$ mysql -u root -p
mysql> CREATE DATABASE trove;
mysql> GRANT ALL PRIVILEGES ON trove.* TO trove@'localhost' IDENTIFIED BY 'TROVE_DBPASS';
mysql> GRANT ALL PRIVILEGES ON trove.* TO trove@'%' IDENTIFIED BY 'TROVE_DBPASS'
```

```
+----+--------+-------------+-----------------------+--------+-------------+------+
| id | name | datastore | datastore_version | status | flavor_id | size |
+----+--------+-------------+-----------------------+--------+-------------+------+
```

TroveによるDBインスタンス作成

```
$ trove create 名前 2 --size=2 --databases DBNAME ¥
--users USER:PASSWORD --datastore_version mysql-5.6  --datastore mysql
```

Troveは標準でMySQLに対応。

詳細：https://wiki.openstack.org/wiki/Trove

Provisioning by MySQL Fabric with Nova

MySQL Fabricは、OpenStackなどのクラウドフレームワークと連携し、
ベアメタル及び仮想環境においてMySQL導入の自動化をサポートします。
また、柔軟にMySQLを準備しスケールアウトさせる事を可能にします。
新規サーバー利用開始時には、**透過的**にMySQLとレプリケーションを設定します。

参照負荷増加 参照処理の増加	**書き込み処理増加** 書き込み増・高負荷	**サーバー障害発生時** グループ内サーバー障害発生時
問題: 参照処理の高負荷状態によるレスポンス遅延	**問題:** 書き込みデータ増加による、全体的な処理遅延の発生	**問題:** サーバー障害による、サービス停止と機会損失
対応: 参照サーバーをコマンド1つで動的に追加	**対応:** テーブルの書き込みデータを複数サーバーにシャーディング	**対応:** スレーブサーバーをマスターへ自動的に昇格

mysqlfabricコマンド

```
-bash-4.2$ mysqlfabric provider list
Fabric UUID:  5ca1ab1e-a007-feed-f00d-cab3fe13249e
Time-To-Live: 1

  provider_id      type username                          url       tenant default_image default_flavor extra
------------- --------- -------- ---------------------------- ------------ ------------- -------------- -----
mysqlfabric01 OPENSTACK  my_user http://8.21.28.222:5000/v2.0/ my_user_role     mysql_img   nova_mysql01  None
```

```
[admin@Fabric01 ~]$ mysqlfabric help provider register
provider register provider_id username password url  [--tenant=NONE]
[--provider_type=OPENSTACK] [--default_image=NONE] [--default_flavor=NONE]
[--extra=NONE] [--synchronous]
Register a provider.
```

プロバイダー, APIアドレス, ユーザー,パスワード等の登録

```
[admin@Fabric01 ~]$ mysqlfabric help server create
server create provider_id  [--image=NONE] [--flavor=NONE] [--number_machines=1]
[--availability_zone=NONE] [--key_name=NONE] [--security_groups=NONE]
[--private_network=NONE] [--public_network=NONE] [--userdata=NONE] [--swap=NONE]
[--scheduler_hints=NONE] [--meta=NONE] [--datastore=NONE]
[--datastore_version=NONE] [--size=NONE] [--databases=NONE] [--users=NONE]
[--configuration=NONE] [--security=NONE] [--skip_store] [--wait_spawning]
[--synchronous]
Create a virtual machine instance:
```

登録済みプロバーダーから, イメージ指定したプロビジョニング等

※ コマンド実行には、OpenStack コマンドラインクライアントのインストールが必要です。

Server Provisioning – NovaとFabricの連携

	インスタンス名	イメージ名	IP アドレス	サイズ	キーペア	状態	アベイラビリティゾーン	タスク	稼働状態	稼働時間	アクション
☐	nova_mysql01	ol65	192.168.56.242	mysql1.micro | 256MB メモリー | 1 仮想 CPU | 8.0GB ディスク	-	Build	nova	Spawning	No State	0 分	Floating IP の割り当て

1 項目を表示中

```
> mysqlfabric provider register my_stack
mats xyzzy http://example.net:5000/v2.0/
my_project --provider_type=OPENSTACK

> mysqlfabric server create my_stack
--image name="Oracle Linux 7 amd64"
--flavor name=m1.small
--userdata=mysql-oracle-linux-init

> mysqlfabric server list my_stack
```

サーバーとMySQLのプロビジョニング

1. OpenStack Nova APIをCALL
   - サーバー準備
2. Slaveクローンの作成
3. レプリケーションの自動設定
4. カスタム操作を実行

※他のフレームワークは対応中(OpenStack Trove, AWS,…)
※ MySQLインストール済みイメージを予め作成し,Nova APIをCall後に,
mysqlfabric group add コマンドでサーバー追加する事も可。
※ Docker等の利用も選択肢の一つ。

MySQL Fabric Resources

- **Download and try**
  http://dev.mysql.com/downloads/fabric/
- **Documentation**
  http://dev.mysql.com/doc/mysql-utilities/en/fabric.html
- **MySQL Fabric on the web**
  http://www-jp.mysql.com/products/enterprise/fabric.html
- **Forum (MySQL Fabric, Sharding, HA, Utilities)**
  http://forums.mysql.com/list.php?144
- **Tutorial: MySQL Fabric - adding High Availability and Scaling to MySQL**
  http://www.clusterdb.com/mysql-fabric/mysql-fabric-adding-high-availability-and-scaling-to-mysql
- **MySQL Fabric-A Guide to Managing MySQL High Availability and Scaling Out**
  http://www.mysql.com/why-mysql/white-papers/mysql-fabric-product-guide
- **Webinar Replays (HA,Sharding,Java)**
  http://www.mysql.com/news-and-events/on-demand-webinars/#en-20-41
- **MySQL Fabric – adding Scaling to MySQL**
  http://www.clusterdb.com/mysql-fabric/mysql-fabric-add-scaling-to-mysql

まとめ

#	概要
1	MySQL Fabricは、レプリケーション、MySQL Utility, Connector を連携させたMySQLサーバー群を管理する統合型のフレームワークです。
2	MySQL Fabricは、高可用性(HA)とデータ・シャーディングによる 読み込みと書き込みの拡張性をサポートします。
3	MySQL Fabricは、OpenStackと連携しベアメタル、仮想環境において MySQLのProvisioningをサポートします。

参考情報：MySQL Fabricサポート

	MySQL Editions		
	Standard SE	Enterprise EE	Cluster CGE
機能概要			
MySQL Database	✔	✔	✔
MySQL Connectors	✔	✔	✔
MySQL Replication	✔	✔	✔
MySQL Fabric		✔	✔
MySQL Partitioning		✔	✔
MySQL Utilities		✔	✔
Storage Engine: MyISAM, InnoDB	✔	✔	✔
Storage Engine: NDB (ndbcluster)			✔
MySQL Workbench SE/EE*	✔	✔	✔
MySQL Enterprise Monitor*		✔	✔
MySQL Enterprise Backup*		✔	✔
MySQL Enterprise Security (外部認証サポート) *		✔	✔
MySQL Enterprise Audit (ポリシーベース監査機能) *		✔	✔
MySQL Enterprise Encryption （非対称暗号化） *		✔	✔
MySQL Enterprise Scalability (スレッドプール) *		✔	✔
MySQL Enterprise High Availability (HAサポート) *		✔	✔
Oracle Enterprise Manager for MySQL*		✔	✔
MySQL Cluster Manager (MySQL Cluster管理) *			✔
MySQL Cluster Geo-Replication			✔

Community Editionで
Fabricをご利用出来ますが、
Enterprise版ではサポートも提供

ORACLE

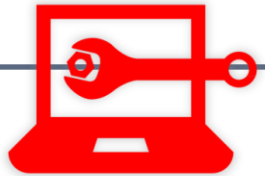
*商用版のみで利用可能な追加機能

	MySQL Editions		
	Standard SE	Enterprise EE	Cluster CGE
Oracle Premium Support			
24時間365日サポート	✔	✔	✔
インシデント数無制限	✔	✔	✔
ナレッジベース	✔	✔	✔
バグ修正&パッチ提供	✔	✔	✔
コンサルティングサポート	✔	✔	✔
オラクル製品との動作保証			
Oracle Linux	✔	✔	✔
Oracle VM	✔	✔	✔
Oracle Solaris	✔	✔	✔
Oracle Enterprise Manager		✔	✔
Oracle Golden Gate		✔	✔
Oracle Data Integrator		✔	✔
Oracle Fusion Middleware		✔	✔
Oracle Secure Backup		✔	✔
Oracle Audit Vault and Database Firewall		✔	✔

MySQL Enterprise Edition のサービスカテゴリー

管理ツール

- 監視
- バックアップ
- 開発サポート
- 管理全般
- マイグレーション



拡張機能

- 拡張性
- 高可用性
- セキュリティ
- 監査
- 暗号化

サポート

- 技術サポート
- コンサルティングサポート
- オラクル製品との
  動作保証

詳細：**MySQL Editions**　http://www-jp.mysql.com/products/

MySQL Enterprise Edition管理ツールと拡張機能概要

MySQL Enterprise Edition	
MySQL Enterprise Monitor	複数サーバの一括管理、クエリ性能分析 MySQLの運用管理と監視で最適化
MySQL Enterprise Backup	高速なオンラインバックアップ、ポイントインタイムリカバリ (例) Backup速度 49倍　Restore速度: 80倍
MySQL Enterprise Scalability	Thread Poolプラグインによる性能拡張性の向上
MySQL Enterprise Security	LDAPやActive Directoryとの外部認証と統合管理
MySQL Enterprise Audit	ユーザ処理の監査、Oracleと同じツールで管理も可能 ユーザーアクセス監査(Who, What, When, Where)
MySQL Enterprise Encryption	非対称暗号化(公開鍵暗号)の業界標準機能を提供 1024bit～　Public Key & Private Keyのペアで管理
Oracle Enterprise Manager for MySQL	Oracle Enterprise ManagerからMySQLを統合管理

運用サポート

24x365, インシデント対応無制限,コンサルティングサポート

レプリケーション・レビュー
パーティショニング・レビュー
スキーマ・レビュー
クエリー・レビュー
パフォーマンス・チューニング等

問題

相談

回答・サポート

```
/**
   Replication binlog transmitter (binlog dump) observer parameter.
*/
typedef struct Binlog_transmit_param {
  uint32 server_id;
  uint32 flags;
  /* Let us keep 1-16 as output flags and 17-32 as input flags */
  static const uint32 F_OBSERVE= 1;
  static const uint32 F_DONT_OBSERVE= 2;

  void set_observe_flag() { flags|= F_OBSERVE; }
  void set_dont_observe_flag() { flags|= F_DONT_OBSERVE; }
  /**
     If F_OBSERVE is set by any plugin, then it should observe binlog
     transmission even F_DONT_OBSERVE is set by some plugins.

     If both F_OBSERVE and F_DONT_OBSERVE are not set, then it is an old
     plugin. In this case, it should always observe binlog transmission.
   */
  bool should_observe()
  {
    return (flags & F_OBSERVE) || !(flags & F_DONT_OBSERVE);
  }
ram;
```

Get Started Today!

MySQL Enterprise Edition Trial

30日間トライアル

Oracle Software Delivery Cloud

http://edelivery.oracle.com/

製品パックを選択：“MySQL Database”

事例紹介：http://www.mysql.com/why-mysql/case-studies/#ja-5-0

製品マニュアル： http://dev.mysql.com/doc/index-enterprise.html

Contact a MySQL Sales Rep

[MySQL お問い合わせ窓口]
電話： 0120-065556
【受付時間】平日 9:00-12:00/13:00-18:00
(祝日及び年末年始休業日を除きます)
メール：MySQL-Sales_jp_grp@oracle.com
URL: http://www.mysql.com/about/contact/

MySQL関連情報

■ **MySQL Cluster 7.4.3 RCリリース　[2015-01-22]**
http://dev.mysql.com/downloads/cluster/7.4.html

■ **MySQL 5.7.5 DMRリリース　　[2014-09-25]**
A character-based ngram full-text parser that supports Chinese, Japanese, and Korean (CJK), and a word-based MeCab parser plugin that supports Japanese were introduced in MySQL 5.7.6, for use with InnoDB tables.
http://dev.mysql.com/doc/refman/5.7/en/full-text-plugins.html

■ **Oracle‘s MySQL image Coming Soon on Docker Hub Registry [2015/2]**
https://blogs.oracle.com/MySQL/entry/oracle_s_mysql_image_coming

■ **MySQL for Oracle DBA(Webinar)**
http://www.mysql.com/products/enterprise/em.html

有難うございました

Q&A

mysqlfabric group add **グループ名 サーバー名**した場合の挙動について？

マスターサーバーにバイナリーログが存在する場合の挙動

CHANGE MASTER TO MASTER_HOSTコマンドが実行されてマスターからバイナリーログを読み込みデータベース作成、テーブル作成、データのINSERT処理が実行され、データが複製されたスレーブサーバーが作成されます。
特に追加のアクションは必要ありません。データが同期された状態で、グループにサーバーが追加されます。

マスターサーバーのバイナリーログが既にPURGEされている場合の挙動

MASTER_AUTO_POSITION = 1でマスターから必要なログが読み取れず以下の様なエラーが発生します。

```
ServerError: Error trying to configure server (4c6e5d9d-bd51-11e4-9ab3-08002766cefe) as slave: Got fatal error 1236 from master when reading data from binary log: 'The slave is connecting using CHANGE MASTER TO MASTER_AUTO_POSITION = 1, but the master has purged binary logs containing GTIDs that the slave requires.'.
```

【対応】バックアップを取得し、準備したデータベースサーバーにリストア後に**mysqlfabric group add** コマンドを実行下さい。