PERCONA

TRAINING

Backups and Recovery

http://www.percona.com/training/

Backups and Recovery

BACKUP TYPES

Backup Types

- Binary Dump (`mongodump`)
- Filesystem Snapshots
- Binary copy/Offline
- Hot Backups
- Percona Backup for MongoDB (PBM)

Backups and Recovery

BINARY DUMP

Binary Dump (`mongodump`)

- `mongodump`
  - Full, per database, collection or query based
  - Excludes content of `local`
  - Logical i.e. documents + indexes information
  - Dump source may differ based on `--host` and/or `--readPreference`

Binary Dump (mongodump)

- Pros:
  - portable
  - non-blocking
  - inline compression (--gzip)
- Cons:
  - Huge restore time (easily 3-4x backup time)
  - Increased outage/recovery window
  - Eats memory, can push working set out of memory

Binary Dump (`mongorestore`)

- `mongorestore`
  - Rebuilds indexes based on metadata included in `mongodump`
  - Excludes `system.profile` on restore, but recreates the collection if it doesn't exist
  - Include, exclude, rewrite filters

Restore Strategies

- Restoring a dump with consistency (oplog)
- Restore Sharded dump issues
  - On a running production system, you can only capture an approximation of point-in-time snapshot
- Point In Time Recovery (PITR)
  - `mongodump --oplog`
    - only supported for full dumps
  - `mongorestore --oplogReplay --oplogLimit`

Exercises - backup/restore prep

1. Stop any running mongod processes

2. Start the replica set

```
screen -S rs1 -d -m mongod --replSet myReplSet --dbpath /mongodb/data/rs1 \
  --port 27017 --oplogSize 200 --wiredTigerCacheSizeGB 0.25 --keyFile /var/lib/mongo/keyfile
screen -S rs2 -d -m mongod --replSet myReplSet --dbpath /mongodb/data/rs2 \
  --port 27018 --oplogSize 200 --wiredTigerCacheSizeGB 0.25 --keyFile /var/lib/mongo/keyfile
screen -S rs3 -d -m mongod --replSet myReplSet --dbpath /mongodb/data/rs3 \
  --port 27019 --oplogSize 200 --wiredTigerCacheSizeGB 0.25 --keyFile /var/lib/mongo/keyfile
```

Exercises - mongodump

1. Insert test data

```
$ mongo admin -u root -p percona
> use training
> db.testcol.insert({a: 1});
> db.testcol.insert({a: 2});
> db.testcol.insert({a: 3});
> db.testcol.createIndex({a: 1})
```

2. Take backup

```
mongodump --db=training --collection=testcol --username=root \
 --authenticationDatabase=admin --password=percona
```

3. Drop some data

```
> use training
> db.testcol.remove({a: 2})
```

Exercises - mongorestore

4. Find the oplog position where the drop happened

```
> use local
> db.oplog.rs.find({ "o": { "drop" : "testcol_2"}}, { ts: 1 })
{ "ts" : Timestamp(1613501536, 1) }
```

5. Restore the data to a different collection

```
$ mongorestore --nsFrom=training.testcol --nsTo=training.testcol_restore \
 --username=root --authenticationDatabase=admin --password=percona dump/
```

6. Verify the restore is successful

```
> use training
> db.testcol_restore.find()
```

Backups and Recovery

FILESYSTEM SNAPSHOTS

Filesystem Snapshots

- LVM2 (Support thin provisioning)
- ZFS (Supports incremental snapshots)
- Storage Appliance
- Cloud i.e. EBS
  - Beware warm up times!

Filesystem Snapshots

- Without journaling enabled

```
db.fsyncLock()
# take snapshot on another session
db.fsyncUnlock()
```

- Example of LVM snapshot

```
lvcreate --size 100M --snapshot --name mdb-snap01 /dev/vg0/mongodb
```

Backups and Recovery

OFFLINE BACKUPS

Offline Backups

- Shutdown `mongod`
- Shutdown `SECONDARY`
- Rarely used
  - Older versions, incompatible filesystem, etc

Backups and Recovery

HOT BACKUPS

Hot Backups

- Percona Server-only feature
- Creates a physical data backup on a running server without notable performance and operating degradation
- WiredTiger only
- Targets:
  - directory
  - TAR file
  - S3-compatible storage

```
use admin
db.runCommand({ createBackup: 1, backupDir: "/my/backup/data/path" })
db.runCommand({ createBackup: 1, archive: <path_to_archive>.tar })
db.runCommand({ createBackup: 1, s3: {bucket: "backup20190510", path: <some_optional_path>} })
```

Hot Backups - restore

1. Stop the service

2. Wipe the datadir

3. Copy back the files

4. Adjust perms

5. Start the service

If you try to restore the node into the existing replica set and there are no oplog entries to roll it forward, initial sync will happen

Exercise - Hot Backup

1. Log into one of the instances

2. Take a hot backup to /tmp

```
use admin
db.runCommand({ createBackup: 1, backupDir: "/tmp/backup" })
```

3. Inspect the files

```
sudo ls -la /tmp/backup
```

Backups and Recovery

Percona Backup for MongoDB (PBM)

Percona Backup for MongoDB

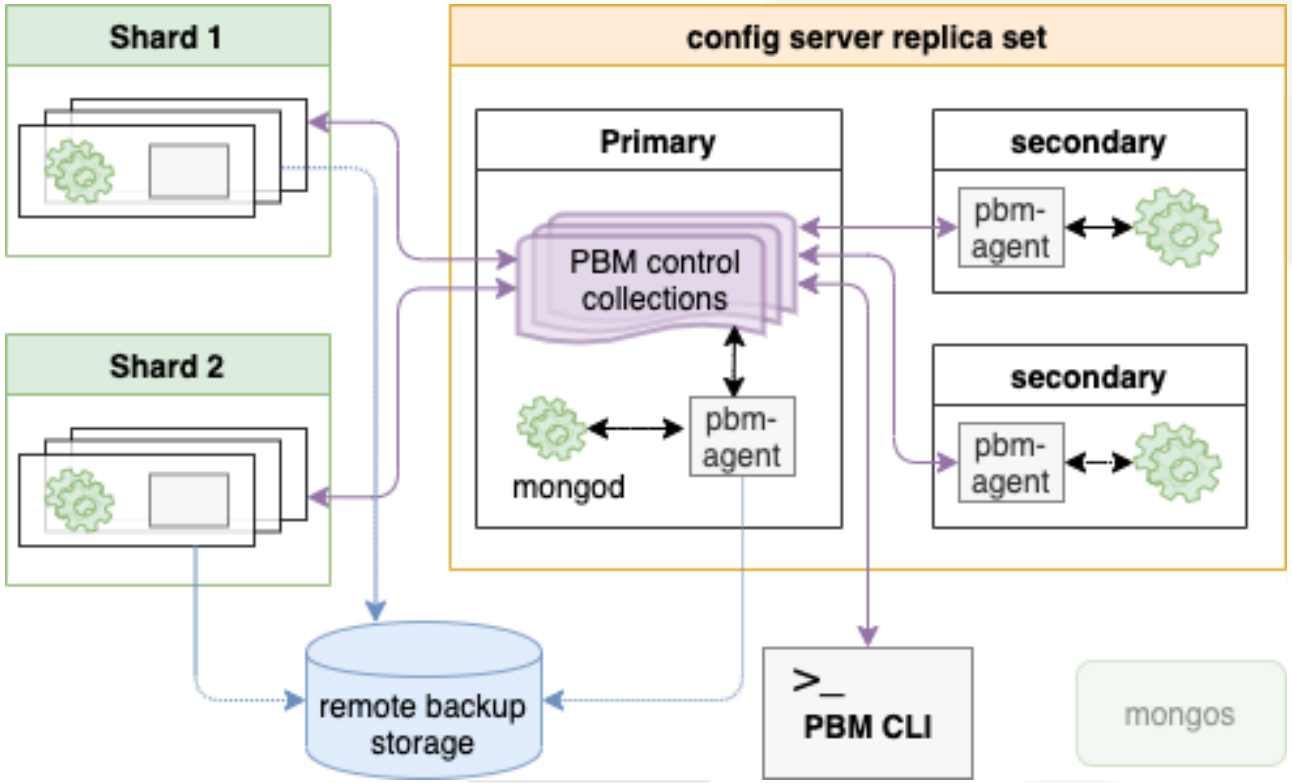
Distributed low-impact solution for achieving consistent backups of MongoDB sharded clusters and replica sets

- Supports Percona Server for MongoDB and MongoDB Community v3.6 or higher
- Doesn't work in standalone mode
- Requires an agent running on each node

Percona Backup for MongoDB

- Simple command-line management utility
- Integrated with MongoDB authentication
- Use S3-compatible storage or volume mounted on all nodes (e.g. NFS)
- Point in time restore

Percona Backup for MongoDB



Percona Backup for MongoDB

- Perform a backup

```
pbm backup
```

- List available backups

```
pbm list
2019-09-10T07:04:14Z
2019-09-09T07:03:50Z
2019-09-08T07:04:21Z
2019-09-07T07:04:18Z
```

- Restore a backup

```
pbm restore 2019-06-09T07:03:50Z
```

Backups and Recovery

SHARDED CLUSTERS

Backing up a sharded cluster

- Stop balancer process
- Lock one secondary member of each replica set
- Backup one config server
- Backup a replica set member for each shard
- Unlock
- Re-enable the balancer

Restoring up a sharded cluster

- Restore config servers
- Restore shards
- Restart mongos nodes

Backup/Restore a sharded cluster

- Too complex? PBM does it for you

Exercises - PBM

1. Install pbm

```
yum install percona-backup-mongodb
```

2. Configure the credentials for PBM agent

```
tee /etc/sysconfig/pbm-agent << EOF
PBM_MONGODB_URI="mongodb://pbmuser:secretpwd@localhost:27017"
EOF
```

3. Start PBM agent

```
systemctl start pbm-agent
```

Exercises - PBM

4. Prepare a file to configure the backup location

```
tee /etc/pbm-agent-storage-local.conf << EOF
storage:
  type: filesystem
  filesystem:
    path: /backup
EOF
```

5. Setup the backup location

```
sudo mkdir /backup
sudo chown pbm: /backup
pbm config --file=/etc/pbm-agent-storage-local.conf \
  --mongodb-uri "mongodb://pbmuser:secretpwd@localhost:27017"
```

Exercises - PBM

6. Take a backup

```
pbm backup --mongodb-uri mongodb://pbmuser:secretpwd@localhost:27017/?replicaSet=r
```

7. List the backups

```
pbm list --mongodb-uri mongodb://pbmuser:secretpwd@localhost:27017
```

8. Connect to the primary and delete the data from one collection

```
> use training
> db.testcol.drop()
```

Exercises - PBM

9. Restore the backup

```
pbm restore 2019-12-12T11:21:10Z --mongodb-uri mongodb://pbmuser:secretpwd@localhc
```

10. Verify the collection is back in place

```
> db.testcol.find()
```

Backups and Recovery

CLONING

Cloning a dataset

- Generate a separate environment with the same data
- The recommended way is to restore into standalone mode
- Then initiate as the first member of a new replica set
- Sharded clusters require more advanced surgery
  - https://www.percona.com/blog/2020/04/28/restoring-a-mongodb-sharded-cluster-to-a-different-environment/

Cloning a dataset - steps

1. Put the restored files into the datadir (remember perms)

2. Comment out replica set configuration from mongod.conf to start in standalone mode

```
#replication:
  #replSetName: testRPL
```

3. Start mongod

```
service mongod start
```

4. Drop the `local` database

```
use local
db.dropDatabase()
```

Cloning a dataset - steps

5. Edit mongod.conf and set the new replica set info

```
replication:
  replSetName: testRPL2
```

6. Restart the service

```
service mongod restart
```

7. Initiate the new replica set

```
mongo
> rs.initiate()
```

PERCONA
TRAINING

Questions