Partition Tolerance in NoSQL

# Introduction

Documenting the journey of experimenting partition tolerance in NoSQL databases for CP and AP models. There is a CAP theorem in NoSql databases that say in case of partition tolerance either of the consistency or availability can be achieved but not both.

**C**

**A** **P**

I am going to work on MongoDB that is a CP model and Riak which support AP model. The databases will be hosted on amazon ec2 instances.

# MongoDB on EC2 Instances:

* In amazon management console, set up the free tier ubuntu instance and install the mongo DB on an instance.
* Create the amazon AMI image.
* Launch 4 more instance using the image.
* Name the instances as primary, secondary1, secondary2, secondary3, secondary4.

Following the steps, you can setup instances with mongodb running.

|  |  |
| --- | --- |
| Launch Instance | Ubuntu Server 16.04 LTS (HVM). |
| Instance Type | T2.micro |
| VPC | Cmpe-281 |
| Assign auto IP | Disable *Assign Elastic IP to the instance.* |
| security group | mongodb cluster  Add Inbound rules. Open ports 22, 27017 |

|  |
| --- |
|  |
|  |
|  |

## Install Mongo DB on Instance:

* sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 9DA31620334BD75D9DCB49F368818C72E52529D4
* echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu xenial/mongodb-org/4.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb.list
* sudo apt update
* sudo apt install mongodb-org

## Generate MongoDB KeyFile:

* openssl rand -base64 741 > keyFile
* sudo mkdir -p /opt/mongodb
* sudo cp keyFile /opt/mongodb
* sudo chown mongodb:mongodb /opt/mongodb/keyFile
* sudo chmod 0600 /opt/mongodb/keyFile

## Configure MongoDB:

## 

* sudo vi /etc/mongod.conf
* replace bindIp with 0.0.0.0
* uncomment security and write keyFile : /opt/mongodb/keyFile.
* Uncomment replication and write replSetName : cmpe281

#### 

## Mongod Service:

* sudo vi /etc/systemd/system/mongod.service. Add the following text in it.

*[Unit]*

*Description=High-performance, schema-free document-oriented database After=network.target*

*[Service]User=mongodb*

*ExecStart=/usr/bin/mongod --quiet --config /etc/mongod.conf*

*[Install]*

*WantedBy=multi-user.target*

Now Enable Mongo Service

* sudo systemctl enable mongod.service
* sudo service mongod restart
* sudo service mongod status

## Create Replica Set:

* Create AMI image of the instance in aws.

AMI: mongodb

* Launch 4 instances using that image and allocate the elastic ip with each. Name the instances as primary,secondary1,secondary2,secondary3,sscondary4.
* Edit /etc/hosts in each instance and add the ip addresses of all instances that needs to be in replication set.

54.145.195.42 primary

3.81.242.72 secondary1

3.209.66.95 secondary2

3.85.252.30 secondary3

52.202.192.206 secondary4

* On primary node hit the following command to initiate the replica set.

rs.initiate({ \_id: "cmpe281", members: [ {\_id:0, host:"primary:27017"}, {\_id:1, host:"secondary1:27017"}, {\_id:2, host: "secondary2:27017"}, {\_id:3, host:"secondary3:27017"}, {\_id:4, host:"secondary4:27017"} ] })

* you can check that every node is in replica set with rs.status().
* On every secondary node, run rs.slave() to make secondary nodes slaves to get replicated from primary.

# MongoDB with no partition:

I have created a java program to write the data to mongoDB. The java program will run to insert documents in primary node. We will see that the data will get replicated to all nodes through secondary1 to secondary4.

MongoDb Java Client

**writes**

**Reads**

Primary

Replication

Replication

Secondary1

Replication

Replication

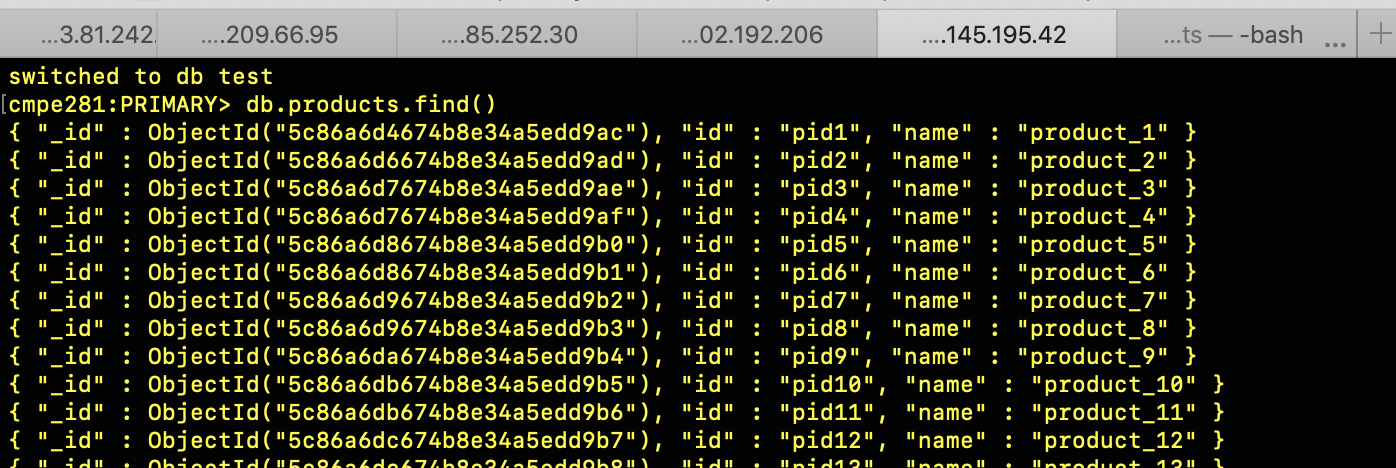
Secondary 4

Secondary2

Secondary3



## Master:



## Slave

## 