Steps for Mongo Replica set on AWS:

Spin up 3 brand-new **Ubuntu 16.04 LTS** instances in the [EC2 console](https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#Instances:sort=tag:Name), making sure to set up each one in a different availability zone, for increased availability in case of service outage in one AZ. Provision enough storage to fit your data size, and select the appropriate instance types for each replica set member. Also, create an EC2 key pair so that you can SSH into the instances.

Set the Hostname

SSH into each server and set its hostname so that when we initialize the replica set, members will be able to understand how to reach one another:

sudo bash -c 'echo DNSNAME > /etc/hostname && hostname -F /etc/hostname'

sudo bash -c 'echo ec2-13-56-84-60.us-west-1.compute.amazonaws.com > /etc/hostname && hostname -F /etc/hostname'

sudo bash -c 'echo ec2-13-57-127-153.us-west-1.compute.amazonaws.com > /etc/hostname && hostname -F /etc/hostname'

sudo bash -c 'echo ec2-13-57-66-238.us-west-1.compute.amazonaws.com > /etc/hostname && hostname -F /etc/hostname'

### For High Efficiency:

### Increase OS Limits

MongoDB needs to be able to create file descriptors when clients connect and spawn a large number of processes in order to operate effectively. The default file and process limits shipped with Ubuntu are not applicable for MongoDB.

Modify them by editing the limits.conf file:

sudo nano /etc/security/limits.conf

Add the following lines to the end of the file:

Spin up 3 brand-new Ubuntu 14.04 LTS instances in the EC2 console, making sure to set up each one in a different availability zone, for increased availability in case of service outage in one AZ. Provision enough storage to fit your data size, and select the appropriate instance types for each replica set member. Also, create an EC2 key pair so that you can SSH into the instances.Next, create a file called 90-nproc.conf in /etc/security/limits.d/:

sudo nano /etc/security/limits.d/90-nproc.conf

Paste the following lines into the file:

\* soft nproc 32000

\* hard nproc 32000

Repeat this step on all replica set members.

### Disable Transparent Huge Pages

Transparent Huge Pages (THP) is a Linux memory management system that reduces the overhead of Translation Lookaside Buffer (TLB) lookups on machines with large amounts of memory by using larger memory pages.

However, database workloads often perform poorly with THP, because they tend to have sparse rather than contiguous memory access patterns. You should disable THP to ensure best performance with MongoDB.

Run the following commands to create an init script that will automatically disable THP on system boot:

sudo nano /etc/init.d/disable-transparent-hugepages

Paste the following inside it:

#!/bin/sh

### BEGIN INIT INFO

# Provides: disable-transparent-hugepages

# Required-Start: $local\_fs

# Required-Stop:

# X-Start-Before: mongod mongodb-mms-automation-agent

Spin up 3 brand-new Ubuntu 14.04 LTS instances in the EC2 console, making sure to set up each one in a different availability zone, for increased availability in case of service outage in one AZ. Provision enough storage to fit your data size, and select the appropriate instance types for each replica set member. Also, create an EC2 key pair so that you can SSH into the instances.Make it executable:

sudo chmod 755 /etc/init.d/disable-transparent-hugepages

Set it to start automatically on boot:

sudo update-rc.d disable-transparent-hugepages defaults

Repeat this step on all replica set data members.

### Turn Off Core Dumps

MongoDB generates core dumps on some mongod crashes. For production environments, you should turn off core dumps since generating them can take minutes or even hours in case your workload is large.

sudo nano /etc/default/apport

Find:

enabled=1

Replace with:

enabled=0

### Configure the Filesystem

Linux by default will update the last access time when files are modified. When MongoDB performs frequent writes to the filesystem, this will create unnecessary overhead and performance degradation. We can disable this feature by editing the fstab file:

sudo nano /etc/fstab

Add the noatime flag directly after defaults:

LABEL=cloudimg-rootfs / ext4 defaults,noatime,discard 0 0

#### **Read Ahead Block Size**

In addition, the default disk read ahead settings on EC2 are not optimized for MongoDB. The number of blocks to read ahead should be adjusted to approximately 32 blocks (or 16 KB) of data. We can achieve this by adding a crontab entry that will execute when the system boots up:

sudo crontab -e

Choose nano by pressing 2 if this is your first time editing the crontab, and then append the following to the end of the file:

@reboot /sbin/blockdev --setra 32 /dev/xvda1

Make sure that your EBS volume is mounted on /dev/xvda1. Save the file and reboot the server:

sudo reboot

Repeat this step on all replica set data members.

Installing Mongo

## **Step 1 — Adding the MongoDB Repository**

MongoDB is already included in Ubuntu package repositories, but the official MongoDB repository provides most up-to-date version and is the recommended way of installing the software. In this step, we will add this official repository to our server.

Ubuntu ensures the authenticity of software packages by verifying that they are signed with GPG keys, so we first have to import they key for the official MongoDB repository.

* sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv EA312927

Issue the following command to create a list file for MongoDB.

* echo "deb http://repo.mongodb.org/apt/ubuntu xenial/mongodb-org/3.2 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-3.2.list

After adding the repository details, we need to update the packages list.

* sudo apt-get update

## **Step 2 — Installing and Verifying MongoDB**

Now we can install the MongoDB package itself.

* sudo apt-get install -y mongodb-org

This command will install several packages containing latest stable version of MongoDB along with helpful management tools for the MongoDB server.

In order to properly launch MongoDB as a service on Ubuntu 16.04, we additionally need to create a unit file describing the service. A unit file tells systemd how to manage a resource. The most common unit type is a service, which determines how to start or stop the service, when should it be automatically started at boot, and whether it is dependent on other software to run.

We'll create a unit file to manage the MongoDB service. Create a configuration file named mongodb.service in the /etc/systemd/system directory using nano or your favorite text editor.

* sudo nano /etc/systemd/system/mongodb.service

Paste in the following contents, then save and close the file.

instances./etc/systemd/system/mongodb.service

[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

Next, start the newly created service with systemctl.

* sudo systemctl start mongodb

While there is no output to this command, you can also use systemctl to check that the service has started properly.

* sudo systemctl status mongodb

The last step is to enable automatically starting MongoDB when the system starts.

* sudo systemctl enable mongodb

The MongoDB server now configured and running, and you can manage the MongoDB service using the systemctl command (e.g. sudo systemctl mongodb stop, sudo systemctl mongodb start)..

Initialize the Replica Set

Connect to one of the MongoDB instances (preferably db1) to initialize the replica set and declare its members. Note that you only have to run these commands on one of the members. MongoDB will synchronize the replica set configuration to all of the other members automatically.

Connect to MongoDB via the following command:

mongo db1.example.com

Initialize the replica set:

rs.initiate()

The command will automatically add the current member as the first member of the replica set.

Add the second data member to the replica set:

rs.add("db2.example.com")

And finally, add the arbiter, making sure to pass in true as the second argument (which denotes that the member is an arbiter and not a data member).

rs.add("arbiter1.example.com", true)

Verify Replica Set Status

Take a look at the replica set status by running:

rs.status()

Inspect the members array. Look for one PRIMARY, one SECONDARY, and one ARBITER member. All members should have a health value of 1. If not, make sure the members can talk to each other on port 27017 by using telnet, for example.

Setup Log Rotation

By default, MongoDB will fill up the /var/log/mongodb/mongod.log file with gigabytes of data. It will be very hard to work with this log file if we do not set up log rotation in advance.

Install logrotate as follows:

sudo apt-get install logrotate

Configure log rotation for MongoDB:

sudo nano /etc/logrotate.d/mongod

Paste the following contents:

/var/log/mongodb/\*.log {

daily

rotate 5

compress

dateext

missingok

notifempty

sharedscripts

copytruncate

postrotate

/bin/kill -SIGUSR1 `cat /var/lib/mongodb/mongod.lock 2> /dev/null` 2> /dev/null || true

endscript

}

This will set up daily log rotation for mongod.log as well as send the SIGUSR1 signal to mongod when the log file is rotated so that it starts writing to the new log file.

Now, Mongo Db replica set is configured and we can use it by this URL:

var mongoURL = "mongodb://ec2-13-56-84-60.us-west-1.compute.amazonaws.com,ec2-13-57-127-153.us-west-1.compute.amazonaws.com/bonapettit?replicaSet=example-replica-set";