https://medium.com/@calvin.hsieh/steps-to-install-mongodb-on-aws-ec2-instance-62db66981218
https://stackoverflow.com/questions/16174591/mongo-count-the-number-of-word-occurrences-in-a-set-of-documents
https://aws.amazon.com/quickstart/architecture/mongodb/
~~MongoDB Atlas~~
~~Raspberry pi~~
~~Overhead memory on desktop machines~~
~~Docker hub~~

Install single unit MongoDB on laptop and run a word counter on it. **.count()**

1. brew services start mongodb-community@4.2
2. ps aux | grep -v grep | grep mongod
3. **In a new window** mongo

show dbs → show databases
show collections → show collections
use myDB → begin a new database from Mongo's client

To insert: db.*collectionName*.insert({name: 'Dillon Thoma', year: 'Senior'})

- To query: ~~select * from *collectionName*~~
    - db.*collectionName*.find()
    - db.*collectionName*.find({}).pretty();
    - db.Greetings.find({},{_id:0}).pretty();

https://scalegrid.io/blog/mongodb-performance-running-mongodb-map-reduce-operations-on-secondaries/

https://docs.mongodb.com/manual/tutorial/install-mongodb-on-ubuntu/ → install on ubuntu

1) ssh -i /path/my-key-pair.pem ec2-user@ec2-198-51-100-1.compute-1.amazonaws.com
copy public dns before doing step 1.
run the following command before doing step 1/
chmod 400 /path/my-key-pair.pem
2) sudo su -

3) passwd ubuntu
4) nano /etc/ssh/sshd_config
 # change PasswordAuthentication "no" to "yes"
 # change PermitRootLogin to “yes”
5) sudo service ssh restart
 # now you have access to the public ip.
sudo apt-get update
6) sudo apt-get install build-essential
7) sudo apt-get install mongodb
8) which mongo → shows where mongo was installed
9) sudo mkdir -p /data/db/ → for data
10) sudo mkdir /data/configdb → for config files
11) sudo chown -R ubuntu /data/configdb/ → change ownership of config server files
12) /etc/hosts → add private ip and host name

```
var map = function() {
    var summary = this.summary;
    if (summary) {
        // quick lowercase to normalize per your requirements
        summary = summary.toLowerCase().split(" ");
        for (var i = summary.length - 1; i >= 0; i--) {
            // might want to remove punctuation, etc. here
            if (summary[i])  {      // make sure there's something
               emit(summary[i], 1); // store a 1 for each word
            }
        }
    }
}
var reduce = function( key, values ) {
    var count = 0;
    values.forEach(function(v) {
        count +=v;
    });
    return count;
}

db.so.mapReduce(map, reduce, {out: "word_count"})
db.word_count.find().sort({value:-1})
```

Machine Type	**Components Installed**	**Description**	**IP Address**	**Hostname**
App Server 1	Application, Mongos	This server will server dual role of app server as well as the mongos server	3.230.2.27	appserver01
App Server 2	Application, Mongos	This server will server dual role of app server as well as the mongos server	3.226.47.160	appserver02
Mongo Config 1	Mongo Config Server	Used as mongodb config server	18.234.139.218	mongoconfig01
Mongo Config 2	Mongo Config Server	Used as mongodb config server	3.221.161.215	mongoconfig02
Shard 1 Primary	Mongo DB	Used as primary DB server in shard 1	3.91.34.195	mongosh01db01
Shard 1 Secondary	Mongo DB	Used as secondary DB server in shard 1	3.235.53.32	mongosh01db02
Shard 2 Primary	Mongo DB	Used as primary DB server in shard 2	34.231.171.233	mongosh02db01

Shard 2 Secondary	Mongo DB	Used as secondary DB server in shard 2	18.215.63.181	mongosh02db02

MongoDB Atlas: running a MongoDB replica set in the AWS cloud environment

1. Configure hostname of each server
    a. /etc/hostname
    b. /etc/hosts
2. Install MongoDB on all servers
    a. wget -qO - https://www.mongodb.org/static/pgp/server-4.2.asc | sudo apt-key add -
    b. echo "deb [arch=amd64,arm64] https://repo.mongodb.org/apt/ubuntu bionic/mongodb-org/4.2 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-4.2.list
    c. sudo apt-get update
    d. sudo apt-get install -y mongodb-org → install latest version of mongo
    e. sudo systemctl start mongod → start mongo
    f. sudo systemctl status mongod → check status
    g. sudo systemctl stop mongod → stop mongo
3. Configure shard 1 replica set “rs0”
    a. Edit /etc/hosts to necessary nodes
        i. *3.91.34.195* mongosh01db01 *3.235.53.32* mongosh01db02
    b. Edit /etc/mongod.conf for necessary nodes
        i. Change bindIp to 0.0.0.0 → bindIp: 0.0.0.0
        ii. Add configuration for rs0 → replication: replSetName: rs0
    c. sudo service mongod restart
4. Configure replica set
    a. Connect to MongoDB using command: mongo rs.initiate() rs.add(“mongosh01db01”)
    b. Connect using command: mongod
        i. sudo mongod → mongo

Mongo Script: https://gist.github.com/Greeshu/a5833afa286147d7672e975c798e8691

1. Launch the instances

- Launch 3 brand new Ubuntu Server 18.04 LTS instances in EC2 console.
- Make sure each instance is in different availability zone
- Create new security group, `mongodb-cluster`
    - Configure all three instances to use it
    - Allow SSH on port 22 from your IP only
    - Allow port 27017 from the `mongodb-cluster` security group and your IP
    - So that both your IP and the replica set members have access to each other's mongod process listening on port 27017
- Label each instance you created as follows (replace example.com with your own domain name):
- Data - db1.example.com
- Data - db2.example.com
- Arbiter - arbiter1.example.com

2. Request 3 Elastic IPs

Attach the requested IPs to each instance, so your replica members will maintain the same public IP throughout the lifetime.

3. Setup DNS Records

Go to your domain's DNS console and add `CNAME` records for db1, db2, arbiter1. For each record, enter each instance's Public DNS hostname, visible in the EC2 instances dashboard.

1. 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` ec2-54-164-33-14.compute-1.amazonaws.com `> /etc/hostname && hostname -F /etc/hostname'`

Make sure to modify db1.example.com and set it to each server's DNS hostname.

2. 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:

```
* soft nofile 64000
* hard nofile 64000
* soft nproc 32000
* hard nproc 32000
```

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
```

3. 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

# Default-Start:     2 3 4 5

# Default-Stop:      0 1 6

# Short-Description: Disable Linux transparent huge pages

# Description:       Disable Linux transparent huge pages, to improve
```

```
#                     database performance.
### END INIT INFO

case $1 in
  start)
    if [ -d /sys/kernel/mm/transparent_hugepage ]; then
      thp_path=/sys/kernel/mm/transparent_hugepage
    elif [ -d /sys/kernel/mm/redhat_transparent_hugepage ]; then
      thp_path=/sys/kernel/mm/redhat_transparent_hugepage
    else
      return 0
    fi

    echo 'never' > ${thp_path}/enabled
    echo 'never' > ${thp_path}/defrag

    unset thp_path
    ;;
esac
```

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
```

4. Configure the File System

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
```

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
```

5. Reboot

Reboot the instance

```
sudo reboot
```

Verify Server Configuration

After rebooting, you can check whether the new hostname is in effect by running:

```
hostname
```

Check that the OS limits have been increased by running:

```
ulimit -u # max number of processes
```

```
ulimit -n # max number of open file descriptors
```

The first command should output `32000`, the second `64000`.

Check whether the Transparent Huge Pages feature was disabled successfully by issuing the following commands:

```
cat /sys/kernel/mm/transparent_hugepage/enabled
```

```
cat /sys/kernel/mm/transparent_hugepage/defrag
```

For both commands, the correct output resembles:

```
always madvise [never]
```

Check that `noatime` was successfully configured:

```
cat /proc/mounts | grep noatime
```

It should print a line similar to:

```
/dev/xvda1 / ext4 rw,noatime,discard,data=ordered 0 0
```

In addition, verify that the disk read-ahead value is correct by running:

```
sudo blockdev --getra /dev/xvda1
```

It should print `32`.

Verify the configuration for all replica set members.

Install MongoDB

Run the following commands to install the latest stable 3.4.x version of MongoDB:

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv
0C49F3730359A14518585931BC711F9BA15703C6

echo "deb [ arch=amd64 ] http://repo.mongodb.org/apt/ubuntu
trusty/mongodb-org/3.4 multiverse" | sudo tee
/etc/apt/sources.list.d/mongodb-org-3.4.list



sudo apt-get update

sudo apt-get install -y mongodb-org
```

These commands will also auto-start `mongod`, the MongoDB daemon. Repeat this step on all replica set members.

Repeat for all replica set members.

MongoDB Setup

Create `keyFile`

The `keyFile` stores the password used by each node. The password allows each node to authenticate to each other, allowing them replicate changes between each other. This password should be long and very complex. We'll use the openssl command to ensure our password is complex.

```
openssl rand -base64 741 > keyFile
```

Create the directory where the key will be stored

```
sudo mkdir -p /opt/mongodb
```

Copy the file to the new directory

```
sudo cp keyFile /opt/mongodb
```

Set the ownership of the keyfile to mongodb.

```
sudo chown mongodb:mongodb /opt/mongodb/keyFile
```

Set the appropriate file permissions.

```
sudo chmod 0600 /opt/mongodb/keyFile
```

Copy the `KeyFile` for all replica set members.

Setup `mongod.conf`

Now it's time to configure MongoDB to operate in replica set mode, as well as allow remote access to the server.

```
sudo nano /etc/mongod.conf
```

Find and remove `bindIp: 127.0.0.1`, or prefix it with a `#` to comment it out:

```
# network interfaces

net:

  port: 27017

#  bindIp: 127.0.0.1  # remove or comment out this line
```

Find the commented out `security` section and uncomment it. Use the path of the keyFile created earlier:

```
security:

  keyFile: /opt/mongodb/keyFile
```

Find the commented out `replication` section and uncomment it. Add the following below, replacing `example-replica-set` with a name for your replica set:

```
replication:

  replSetName: my-replica-set
```

IMPORTANT use the same `replSetName` for ALL replica members

Create `mongod.service`

```
sudo nano /etc/systemd/system/mongod.service
```

Write the following to the file:

```
[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
```

Enable `mongod.service`

```
sudo systemctl enable mongod.service
```

Restart MongoDB to apply our changes.

```
sudo service mongod restart
```

Repeat for all replica set members.

Initialize the Replica Set

Be sure you have everything setup properly in all replica set members by this point.

Connect to one of the MongoDB instances (preferably `db1`) using SSH 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
```

Initialize the replica set:

```
rs.initiate()
```

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

Create Admin Account

The default MongoDB configuration is wide open, meaning anyone can access the stored databases unless your network has firewall rules in place.

Create an admin user to access the database.

```
mongo
```

Select `admin` database.

```
use admin
```

Create `admin` account.

```
db.createUser( {
    user: "johndoe",
    pwd: "strongPassword",
    roles: [{ role: "root", db: "admin" }]
});
db.auth("johndoe", "strongPassword") > db.grantRolesToUser("johndoe", [ { role: "read", db: "admin" } ])
```

It's recommended to not use special characters in the password to prevent issues logging in

Adding Replica Members

Add the second data member to the replica set:

```
rs.add("ec2-75-101-236-111.compute-1.amazonaws.com")
```

```
mongod --bind_ip localhost,34.228.156.209
```

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)
```

Be sure to replace `example.com` with your own domain name.

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.

https://gist.github.com/calvinh8/c99e198ce5df3d8b1f1e42c1b984d7a4#3-setup-dns-records

Initialize the replica set: sudo rs.initialize()

```
example-replica-set:PRIMARY> rs.initiate()
{
        "ok" : 0,
        "errmsg" : "not authorized on admin to execute command { replSetInitiate: undefined }",
        "code" : 13,
        "codeName" : "Unauthorized"
}
```

https://stackoverflow.com/questions/23943651/mongodb-admin-user-not-authorized

DNS

https://www.cyberciti.biz/faq/howto-linux-bsd-unix-set-dns-nameserver/

https://cleanbrowsing.org/articles/changing-dns-nameservers-via-terminal