**Overview of what your VMs:**

**VM1**: Primary shard and possibly a mongos router

**VM2**: Secondary shard

**VM3**: Config server and possibly a mongos router

Before starting keep in mind that we are creating both replicas and shard server:

**mongosh --host vm\_ip --port 27017** is the command used to run replicas and **mongos --host vm\_ip --port 27017** is used to run shard server after we complete configuration. These commands may have been mentioned below but you must use full command for connectivity. Also use different PORTS for each VM this time.

Let’s go through each VM step by step:

**STEP:1**

**VM3** (This is not a primary VM as in our case. Remember that I have taken VM1 as primary):

Mongod.conf (leave rest of the settings as it is):

**sharding:**

**clusterRole: "configsvr"**

**replication:**

**replSetName: "csReplSet"**

**net:**

**port: 27019**

**bindIp: 0.0.0.0 # or specific IP of VM3**

**storage:**

**dbPath: /data/configdb**

csReplSet was named rs0 in our case.

make sure you check for data path correctly also use spaces and not tabs as this file uses YAML format and is indentation sensitive (plus case sensitive of course.

**STEP:2**

Then intiate mongod server:

**sudo systemctl start mongod**

(you can use restart, enable, stop etc instead of start whenever needed)

then initiate server replica set with mongosh:

**mongosh --port 27019**

**STEP:3**

**rs.initiate({**

**\_id: "csReplSet",**

**configsvr: true,**

**members: [{ \_id: 0, host: "vm3\_ip:27019" }]**

**})**

Notice that the inner parameters represent what you want to configure our focus is just to enable configsvr for now. Set other parameters vicely

Now leave this running and lets configure VM1 and VM2 for shrad server:

**STEP:4**

**VM1 and VM2:**

Add in mongod.conf of both VM1 and 2:

**sharding:**

**clusterRole: "shardsvr"**

**replication:**

**replSetName: "rs0"**

**net:**

**port: 27017**

**bindIp: 0.0.0.0 # or specific IPs of VM1 and VM2**

**storage:**

**dbPath: /data/db**

restart as mentioned in the command above.

Now we initialize shell only in VM1 which is primary now:

**mongosh --port 27017**

**STEP:5**

**rs.initiate({**

**\_id: "rs0",**

**members: [**

**{ \_id: 0, host: "vm1\_ip:27017" },**

**{ \_id: 1, host: "vm2\_ip:27017" }**

**]**

**})**

You can follow above command but it cases errors sometimes as we have to use that add command before which we did yesterday already in your VM3(primary).

**rs.add("your\_ip\_for\_first:27017")**

**rs.add("your\_ip\_for\_second\_macine:27017")**

can add either way but latter is more safe from errors.

**STEP:6**

Now you can either take **VM1 or VM3** as your mongos server:

Create a file /etc/mongos.conf:

**sharding:**

**configDB: csReplSet/vmip:27019**

**port: 27017**

**bindIp: 0.0.0.0**

**STEP:7**

use IP of your VM either you are using VM1(primary) or VM3

net:

add this to configuration of VM:

**mongos --config /etc/mongos.conf**

after running config, just start the shell

**mongosh --port 27017 --host vmip**

**STEP:8**

our mongos is set up and ready here you can use this to add other two VMs in your shard after running this in the shell (In case you setup mongos in VM3, add VM1 and 2):

**sh.addShard("rs0/vm1\_ip:27017")**

**sh.addShard("rs0/vm2\_ip:27017")**

then enable sharding:

**sh.enableSharding("databaseName")**

**sh.shardCollection("databaseName.collectionName", { shardKey: 1 })**

check status of your sharding by using the command:

**sh.status()**

**Network Errors Are Most Important:**

if you ran into some kind of connection errors then run this command for firewall:

**sudo ufw allow 27017**

**sudo ufw allow from 0.0.0.0 to any port 27017**

<https://stackoverflow.com/questions/66944769/cant-connect-to-mongodb-remotely-after-opening-ubuntu-firewall-and-mongod-conf>

**AGAIN I AM SAYING THAT ALL THE ISSUES WE FACED WERE EITHER SYNTAX ERRORS ARE PORT MISCONFIGURATION(CONNECTIVITY). MAKE SURE YOU DON”T RUN INTO ANY OF THESE AND DEBUG AND CHECK SERVER AFTER EVERY STEP.**

**FOR TESTING YOUR CLUSTER:**

* Create A collection and then add some dummy data using mongosh in the test database.
* Go to other VM and try to find that data there (using findOne or findMany) You can see these basic commands of mongosh everywhere on the internet.
* If not so then simply check status using rs.status() or sh.status().

**Just to let you know what mistakes can be made:**

* YAML files are sensitive to spaces and indentation. Using tabs instead of spaces or incorrect indentation levels can cause MongoDB to fail to start.
* If VMs cannot communicate with each other, ensure that the required ports (27017 for mongod, 27019 for config servers, and any other specified port) are open on each VM's firewall.
* Ensure the bindIp is set to 0.0.0.0 or to the correct internal network IPs to allow connections from other VMs.
* When setting up mongos, ensure the configDB string is correct and points to the active config servers with their right IPs and port.
* Always ensure mongos is pointing to an up and running config server set. If mongos can’t connect to the config servers, it won’t function.
* The MongoDB user (usually mongodb on Ubuntu) needs to have read/write permissions to the data paths specified in dbPath for both the shard and config servers.
* MongoDB logs are present in the log files in mongo folder in the directory where you installed the MongoDB. Always check for error diagnosis.