



# Creación de Replica Set

**Manuel Fernando Cordoba Gonzalez**  
**Mario Juan Sebastian Reyes Casas**

## Requisitos que debe tener CentOS para crear un clúster exitosamente:

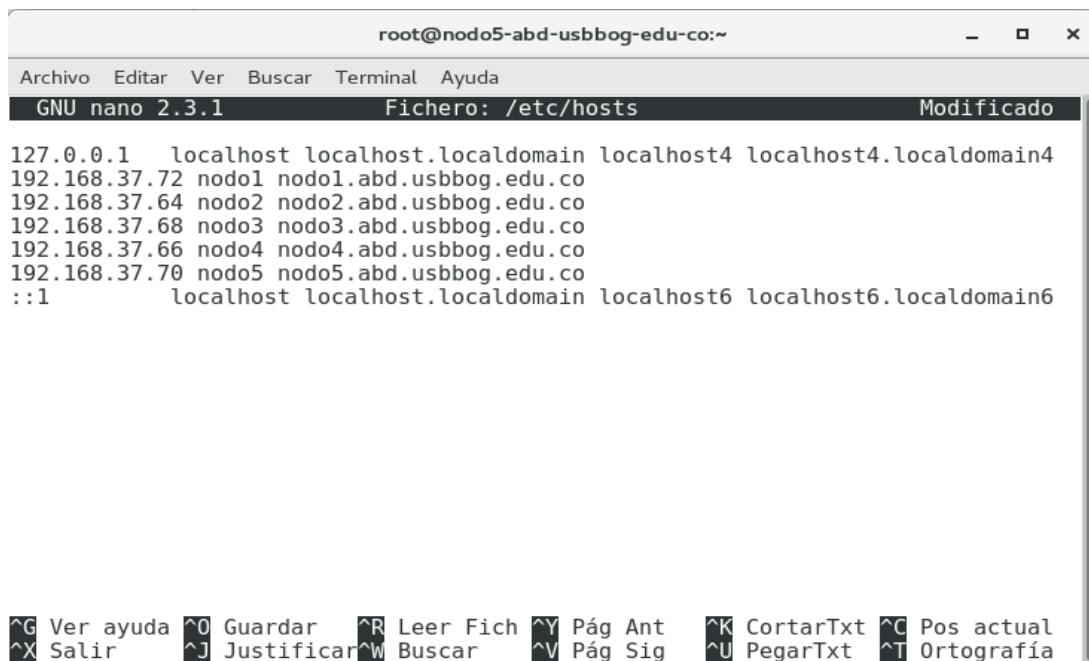
- **Firewalld desactivado:** Es una solución de firewall completa que administra dinámicamente el nivel de confianza de las conexiones e interfaces de red. Le brinda control total sobre el tráfico permitido o no permitido hacia y desde el sistema. En escenarios reales no es recomendable desactivarlo sin embargo por cuestiones de aprendizaje se desactivará.
- **SELinux desactivado:** Es una extensión de seguridad de CentOS, un módulo de seguridad para el kernel Linux que proporciona el mecanismo para configurar políticas de control de acceso; sin embargo, estas mismas políticas pueden ocasionar problemas en nuestras conexiones.
- **Tener instalado el repositorio de MongoDB:** Esto es lo más importante si se va a trabajar con MongoDB.

## 1. CONFIGURACIÓN PARA TODOS LOS NODOS

Como primer paso debemos añadir las IP de los nodos que vayamos a usar, para esto se debe ejecutar el siguiente comando en consola y acceder al archivo "hosts":

```
Archivo Editar Ver Buscar Terminal Ayuda
[root@nodo5-abd-usbbog-edu-co ~]# nano /etc/hosts
```

Una vez en el editor de archivo debemos añadir los nodos incluyendo el nodo actual, usando la siguiente sintaxis: *IP\_del\_nodo DNS Dominio*



```
root@nodo5-abd-usbbog-edu-co:~
GNU nano 2.3.1          Fichero: /etc/hosts          Modificado

127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
192.168.37.72 nodo1 nodo1.abd.usbbog.edu.co
192.168.37.64 nodo2 nodo2.abd.usbbog.edu.co
192.168.37.68 nodo3 nodo3.abd.usbbog.edu.co
192.168.37.66 nodo4 nodo4.abd.usbbog.edu.co
192.168.37.70 nodo5 nodo5.abd.usbbog.edu.co
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6

^G Ver ayuda  ^O Guardar   ^R Leer Fich ^Y Pág Ant  ^K CortarTxt ^C Pos actual
^X Salir      ^J Justificar^W Buscar    ^V Pág Sig  ^U PegarTxt  ^T Ortografía
```

Una vez añadidos todos los nodos se guardan (ctrl+O) y se cierra el editor (ctrl+X)

Se deberá crear una carpeta para el almacenamiento para esto, se ubica en la raíz y se crea la carpeta data que tendrá dentro la carpeta db.

```
[root@nodo5-abd-usbbog-edu-co /]# mkdir data
[root@nodo5-abd-usbbog-edu-co /]#
[root@nodo5-abd-usbbog-edu-co /]# mkdir /data/db
```

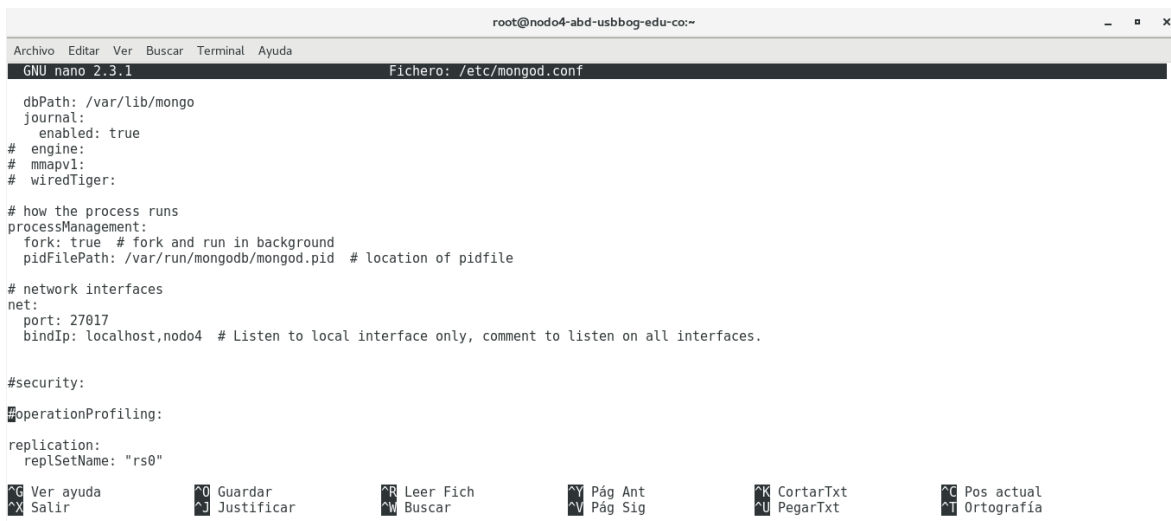
Accedemos al archivo mongod.conf ejecutando el siguiente comando.

```
[root@nodo5-abd-usbbog-edu-co db]# nano /etc/mongod.conf
```

Una vez en el editor

- Nos dirigimos al apartado *net* y después de *bindIp*: escribimos *localhost,(IP o DNS del nodo actual)*
- Se comenta el apartado *replication*: quitando el *#* del inicio, después una línea más abajo se presiona la tecla tab y se escribe *replSetName: "rs0"*

**Nota: En este caso se le dio el nombre a la réplica rs0, este nombre es arbitrario, sin embargo, este nombre deberá respetarlo y ponerlo en todas las configuraciones de los nodos.**



```
root@nodo4-abd-usbbog-edu-co:~
GNU nano 2.3.1          Fichero: /etc/mongod.conf

dbPath: /var/lib/mongo
journal:
  enabled: true
# engine:
# mmapv1:
# wiredTiger:

# how the process runs
processManagement:
  fork: true # fork and run in background
  pidFilePath: /var/run/mongodb/mongod.pid # location of pidfile

# network interfaces
net:
  port: 27017
  bindIp: localhost,nodo4 # Listen to local interface only, comment to listen on all interfaces.

#security:
#operationProfiling:
replication:
  replSetName: "rs0"

^G Ver ayuda      ^O Guardar      ^R Leer Fich    ^Y Pág Ant      ^K CortarTxt    ^C Pos actual
^X Salir          ^J Justificar   ^W Buscar      ^V Pág Sig      ^U PegarTxt     ^T Ortografia
```

Una vez añadidos modificado se guardan (ctrl+O) y se cierra el editor (ctrl+X)

## 2. CONFIGURACIÓN NODO MAESTRO

Ahora iniciamos el servicio con el comando *systemctl start mongod* y después entramos con el comando *mongo*

Una vez en el servicio ejecutamos el comando *rs.initiate()* **solo en un nodo** para inicializar el nodo actual como el master (primary).

**Nota: Asegurarse de que el valor de 'OK' sea 1, si no es así consultar en la documentación oficial o en Google para posibles soluciones.**

Ya teniendo nuestro nodo maestro añadimos nuestros demás nodos (slaves)

```
}
rs0:PRIMARY> rs.add("nodo3:27017")
{ "ok" : 1 }
rs0:PRIMARY> rs.add("nodo2:27017")
{ "ok" : 1 }
rs0:PRIMARY> rs.add("nodo4:27017")
{ "ok" : 1 }
rs0:PRIMARY> rs.add("nodo5:27017")
{ "ok" : 1 }
rs0:PRIMARY>
```

---

### 3. CONFIGURACIÓN NODOS SLAVES

Por ultimo ejecutamos el comando `rs.slaveOk()` para permitir que la conexión actual permita que las operaciones de lectura se ejecuten en miembros secundarios.

```
rs0:SECONDARY> rs.slaveOk()
rs0:SECONDARY> db.workers.find()
{ "_id" : ObjectId("5d4fa553184bd648abd3c1cd"), "nombre" : "carle", "apellido" : "narcotico" }
{ "_id" : ObjectId("5d5da83e33299b1f914976cf"), "nombre" : "carle", "apellido" : "narcotico" }
rs0:SECONDARY> █
```

**Nodo secundario** (slave): Es el nodo que mantiene una copia de la dataset del **nodo primario**(master), sin embargo, el **nodo secundario** solo tendrá permiso de lectura nunca de escritura. Si intentamos insertar una colección estando en un nodo secundario nos arroja el siguiente aviso:

```
rs0:SECONDARY> db.try.insert({"nombre" : "carlos", "telefono" : "00021"})
WriteResult({ "_writeError" : { "code" : 10107, "errmsg" : "not master" } })
```