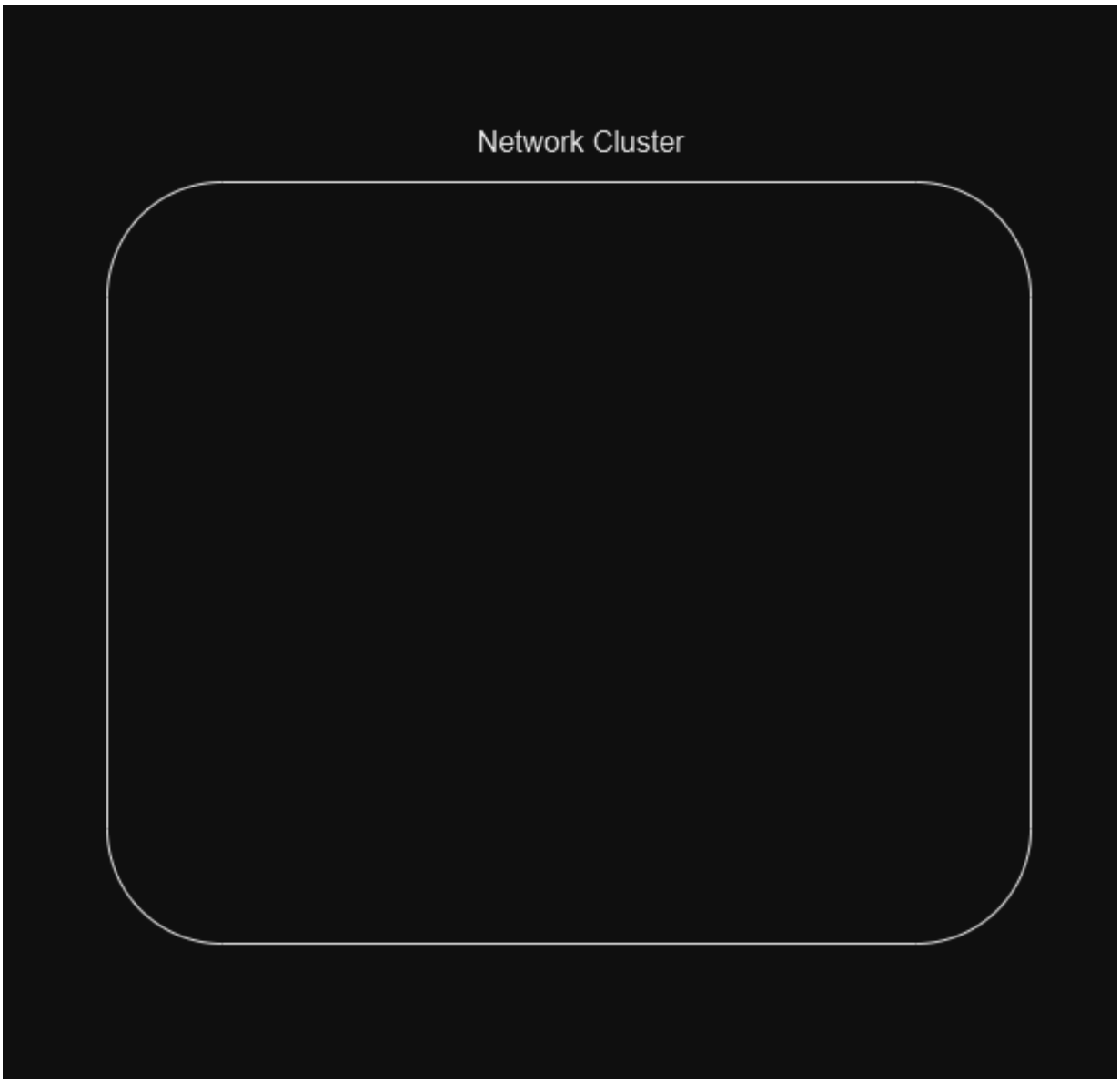
C. Criação da rede + nós + replicaset

Criação da rede, aonde vamos colocar os nós do cluster

```
docker network create cluster
```

Palavra	Função
docker	chama o Docker
network create	Indica que estamos criando um rede docker
cluster	nome da rede (você pode escolher)

-> Essa rede vai permitir que os nós se vejam

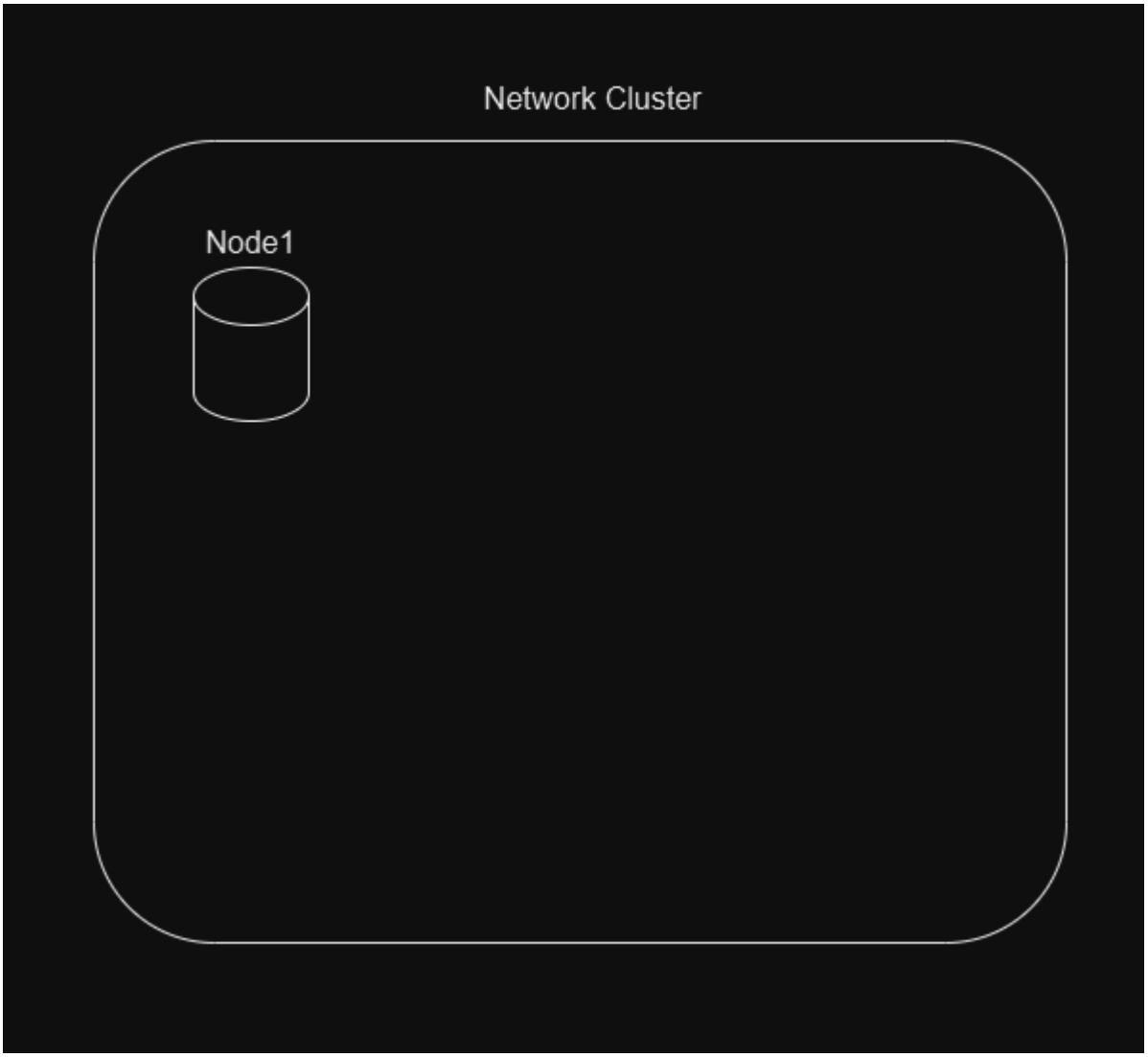


Criação nó 1

```
docker run -d --rm -p 27017:27017 --name node1 --network cluster
mongodb/mongodb-community-server:latest --replSet replicaSet --bind_ip
localhost,node1
```

Palavra	Função
docker run	Cria e inicia um novo container
-d	Executa o container em segundo plano (modo "detached")
--rm	Remove automaticamente o container quando ele for parado

Palavra	Função
-p 27017:27017	Mapeia a porta 27017 do host para a porta 27017 do container (porta padrão do mongoDB)
--name node1	Dá o nome "node1" para o container
--network cluster	Conecta o container á rede Docker chamada "cluster" criada anteriormente com o "docker network create cluster"
mongodb/mongodb-community-server:latest	Imagem usada para o container
--replSet replicaSet	comando para aplicar o replica set com o nome "replicaSet"
--bind_ip localhost,node1	fala pro mongoDB aceitar as conexões de localhost e do nome do container (node1)



Criação nó 2

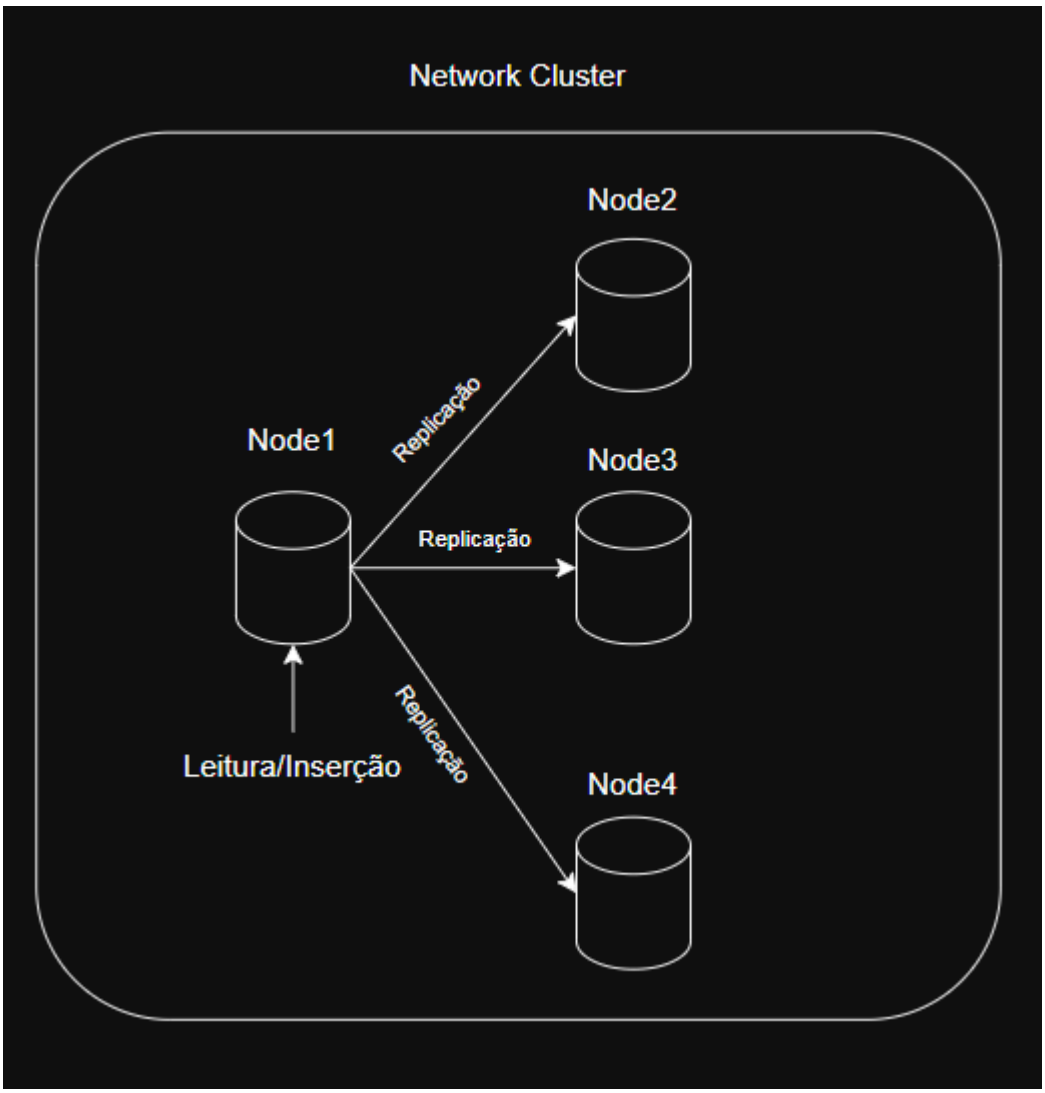
```
docker run -d --rm -p 27018:27017 --name node2 --network cluster
mongodb/mongodb-community-server:latest --replSet replicaSet --bind_ip
localhost,node2
```

Criação nó 3

```
docker run -d --rm -p 27019:27017 --name node3 --network cluster
mongodb/mongodb-community-server:latest --replSet replicaSet --bind_ip
localhost,node3
```

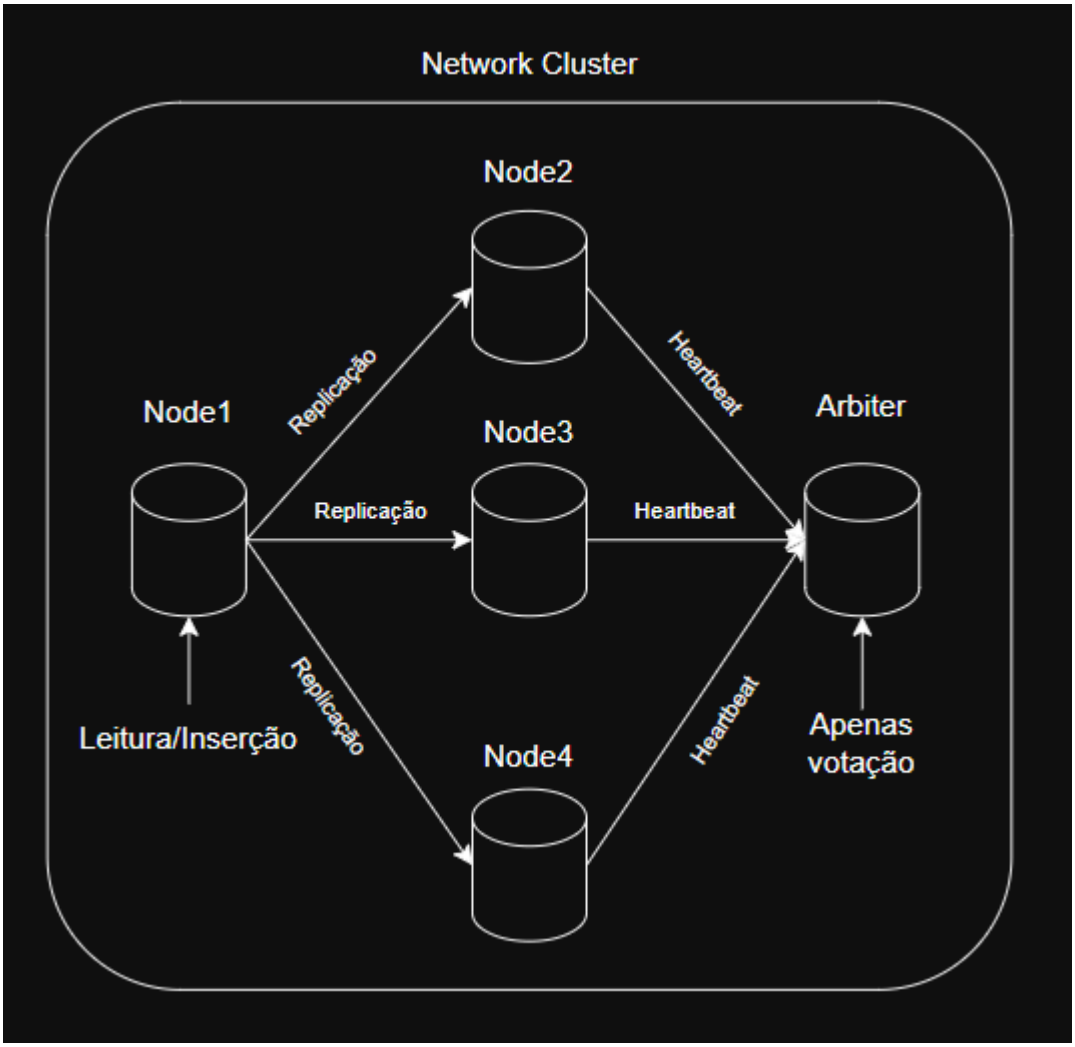
Criação nó 4

```
docker run -d --rm -p 27020:27017 --name node4 --network cluster
mongodb/mongodb-community-server:latest --replSet replicaSet --bind_ip
localhost,node4
```



**Criação do arbiter

```
docker run -d --rm --name arbiter --network cluster mongodb/mongodb-community-server:latest --replSet replicaSet --bind_ip localhost,arbiter
```

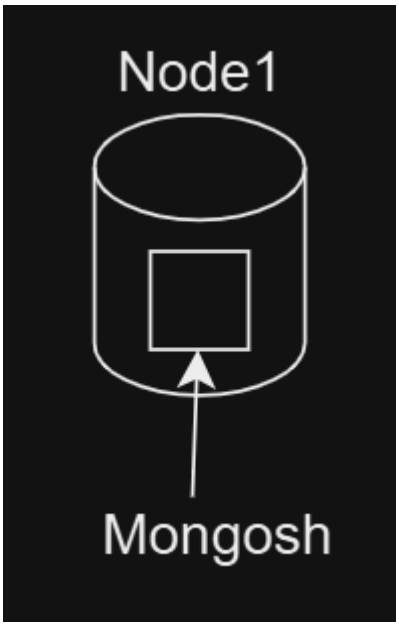


**Acessar a ferramenta mongosh disponível dentro do container Docker

```
docker exec -it node1 mongosh
```

palavra	Função
docker exec	Executa um comando dentro de um container já em execução
-it	Interativo + terminal (necessário para comandos que usam entrada/saída)
node1	nome do container que vai ser executado

palavra	Função
mongosh	comando a ser executado, no caso o Shell interativo do mongoDB



Verificar se a instancia está funcionando corretamente e obter informações sobre o nó

```
`db.runCommand ({hello:1})`
```

Palavra	Função
db	referencia ao banco de dados atual (mongosh)
runCommand()	executa comando administrativos diretamente no mongoDB
{ hello: 1}	comando para verificar status e metadados do nó atual

---

**definir replicaset (sem arbiter)

```
rs.initiate ({ _id: "replicaSet", members:[{_id:0, host: "node1"}, {_id:1,
host: "node2"}, {_id:2, host: "node3"}, {_id:3, host: "node4"}]})
```

**com arbiter

```
rs.initiate ({ _id: "replicaSet", members:[{_id:0, host: "node1"}, {_id:1, host: "node2"}, {_id:2, host: "node3"}, {_id:3, host: "node4"}, {_id:4, host: "arbiter", arbiterOnly: true}]})
```

Palavra	Função
rs.initiate()	Iniciar replica set
_id: "replicaSet"	nome do replica set (**Deve bater com o valor passado com --replSet lá na criação do nó)
members: [...]	Lista dos membros (nós) do replica set
{ _id: 0, host: "node1" }	Cada objeto define um nó com seu id e o nome do host

---

sair do mongosh

```
exit
```

**verificar status e membros da replica

```
docker exec -it node1 mongosh --eval "rs.status()"
```

Palavra	Função
docker exec	Executa um comando dentro de um container já em execução
-it	Interativo + terminal (necessário para comandos que usam entrada/saída)
node1	Nome do container onde o MongoDB está rodando
mongosh	usar Shell interativo do mongoDB
eval "rs.status()"	Executa o comando MongoDB rs.status() e exibe o resultado no terminal