```
Microsoft Windows [Version 10.0.22621.1702]
(c) Microsoft Corporation. All rights reserved.

C:\Users\gamerpc>mongosh --host localhost --port 27019
Current Mongosh Log ID: 64893883bb69865858dd0199
Connecting to:          mongodb://localhost:27019/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+1.6.1
Using MongoDB:          6.0.3
Using Mongosh:          1.6.1

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

------
   The server generated these startup warnings when booting
   2023-06-13T22:32:35.285-05:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
   2023-06-13T22:32:35.285-05:00: This server is bound to localhost. Remote systems will be unable to connect to this server. Start the server with --bind_ip <address> to specify which IP addresses it should s
erve responses from, or with --bind_ip_all to bind to all interfaces. If this behavior is desired, start the server with --bind_ip 127.0.0.1 to disable this warning
------

------
   Enable MongoDB's free cloud-based monitoring service, which will then receive and display
   metrics about your deployment (disk utilization, CPU, operation statistics, etc).

   The monitoring data will be available on a MongoDB website with a unique URL accessible to you
   and anyone you share the URL with. MongoDB may use this information to make product
   improvements and to suggest MongoDB products and deployment options to you.

   To enable free monitoring, run the following command: db.enableFreeMonitoring()
   To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
------

test> conn = new Mongo("localhost:27001");
mongodb://localhost:27001/?directConnection=true&serverSelectionTimeoutMS=2000
test> shard1DB = conn.getDB("admin");
admin
test>

test> configShard1 = {
...   _id: "shardrs1",
...   members: [
...     { _id: 0, host: "localhost:27001" },
...     { _id: 1, host: "localhost:27002" },
...     { _id: 2, host: "localhost:27003" }
...   ]
... };
{
  _id: 'shardrs1',
  members: [
    { _id: 0, host: 'localhost:27001' },
    { _id: 1, host: 'localhost:27002' },
    { _id: 2, host: 'localhost:27003' }
  ]
}
test>

test> shard1DB.runCommand({replSetInitiate: configShard1});
{ ok: 1 }
test> conn = new Mongo("localhost:27004");
mongodb://localhost:27004/?directConnection=true&serverSelectionTimeoutMS=2000
test> shard2DB = conn.getDB("admin");
admin
test>

test> configShard2 = {
...   _id: "shardrs2",
...   members: [
...     { _id: 0, host: "localhost:27004" },
...     { _id: 1, host: "localhost:27005" },
...     { _id: 2, host: "localhost:27006" }
...   ]
... };
{
  _id: 'shardrs2',
  members: [
    { _id: 0, host: 'localhost:27004' },
    { _id: 1, host: 'localhost:27005' },
    { _id: 2, host: 'localhost:27006' }
  ]
}
test>

test> shard2DB.runCommand({replSetInitiate: configShard2});
{ ok: 1 }
test> db = (new Mongo("localhost:27001")).getDB("torneo_deportivo");
torneo_deportivo
shardrs1 [direct: primary] torneo_deportivo> for (let i = 0; i < 150000; i++) {
...   db.Jugadores.insert({
...     "Id": i,
...     "Nombre": "Jugador " + i,
...     "Edad": Math.floor(Math.random() * 40) + 18,
...     "Posición": "Posición " + i,
...     "Equipo": "Equipo " + i
...   });
... }
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
Stopping execution...
shardrs1 [direct: primary] torneo_deportivo>

shardrs1 [direct: primary] torneo_deportivo> for (let i = 0; i < 150000; i++) { db.Jugadores.insertOne({ "Id": i, "Nombre": "Jugador " + i, "Edad": Math.floor(Math.random() * 40) + 18, "Posición": "Posición "
+ i, "Equipo": "Equipo " + i }); }
Stopping execution...
shardrs1 [direct: primary] torneo_deportivo>
```

```
shardrs1 [direct: primary] torneo_deportivo> for (let i = 0; i < 150000; i++) { db.Jugadores.insertOne({ "Id": i, "Nombre": "Jugador " + i, "Edad": Math.floor(Math.random() * 40) + 18, "Posición": "Posición "
+ i, "Equipo": "Equipo " + i }); }
Stopping execution...
shardrs1 [direct: primary] torneo_deportivo>

shardrs1 [direct: primary] torneo_deportivo> for (let i = 0; i < 15; i++) { db.Jugadores.insertOne({ "Id": i, "Nombre": "Jugador " + i, "Edad": Math.floor(Math.random() * 40) + 18, "Posición": "Posición " + i,
 "Equipo": "Equipo " + i }); }
{
  acknowledged: true,
  insertedId: ObjectId("64893a12e39fbe0fed6e2a61")
}
shardrs1 [direct: primary] torneo_deportivo> for (let i = 0; i < 15; i++) {
...   db.Entrenadores.insertOne({
...     "Id": i,
...     "Nombre": "Entrenador " + i,
...     "Edad": Math.floor(Math.random() * 40) + 30,
...     "Equipo": "Equipo " + i
...   });
... }
{
  acknowledged: true,
  insertedId: ObjectId("64893a28e39fbe0fed6e2a70")
}
shardrs1 [direct: primary] torneo_deportivo> for (let i = 0; i < 15; i++) {
...   db.Árbitros.insertOne({
...     "Id": i,
...     "Nombre": "Árbitro " + i,
...     "Edad": Math.floor(Math.random() * 20) + 25
...   });
... }
{
  acknowledged: true,
  insertedId: ObjectId("64893a3be39fbe0fed6e2a7f")
}
shardrs1 [direct: primary] torneo_deportivo> for (let i = 0; i < 15; i++) {
...   db.Encuentros.insertOne({
...     "Id": i,
...     "Fecha": new Date(),
...     "Hora": "Hora " + i,
...     "Equipolocal": "Equipo Local " + i,
...     "Equipovisitante": "Equipo Visitante " + i
...   });
... }
{
  acknowledged: true,
  insertedId: ObjectId("64893a50e39fbe0fed6e2a8e")
}
shardrs1 [direct: primary] torneo_deportivo> for (let i = 0; i < 15; i++) {
...   db.Resultados.insertOne({
...     "Id": i,
```



```
...     "Id": i,
...     "Encuentroid": i,
...     "Goleslocal": Math.floor(Math.random() * 5),
...     "Golesvisitante": Math.floor(Math.random() * 5)
...   });
... }
{
  acknowledged: true,
  insertedId: ObjectId("64893a6be39fbe0fed6e2a9d")
}
shardrs1 [direct: primary] torneo_deportivo> for (let i = 0; i < 15; i++) {
...   db.tablaposiciones.insertOne({
...     "Id": i,
...     "Equipo": "Equipo " + i,
...     "Puntos": Math.floor(Math.random() * 100)
...   });
... }
{
  acknowledged: true,
  insertedId: ObjectId("64893a7ee39fbe0fed6e2aac")
}
shardrs1 [direct: primary] torneo_deportivo> shardrs1 = new Mongo("localhost:27001");
mongodb://localhost:27001/?directConnection=true&serverSelectionTimeoutMS=2000
shardrs1 [direct: primary] torneo_deportivo> shardrs1db = shardrs1.getDB("Torneo_deportivo");
Torneo_deportivo
shardrs1 [direct: primary] torneo_deportivo> db.jugadores.count();
DeprecationWarning: Collection.count() is deprecated. Use countDocuments or estimatedDocumentCount.
0
shardrs1 [direct: primary] torneo_deportivo> db.jugadores.countDocuments();
0
shardrs1 [direct: primary] torneo_deportivo> for (let i = 0; i < 15; i++) { db.Encuentros.insertOne({ "Id": i, "Fecha": new Date(), "Hora": "Hora " + i, "Equipolocal": "Equipo Local " + i, "Equipovisitante": "
Equipo Visitante " + i }); }
{
  acknowledged: true,
  insertedId: ObjectId("64893b5ee39fbe0fed6e2abb")
}
shardrs1 [direct: primary] torneo_deportivo> db.jugadores.countDocuments();
0
shardrs1 [direct: primary] torneo_deportivo> db.Encuentros.countDocuments();
30
shardrs1 [direct: primary] torneo_deportivo> db.Entrenadores.countDocuments();
15
shardrs1 [direct: primary] torneo_deportivo> db.Árbitros.countDocuments();
15
shardrs1 [direct: primary] torneo_deportivo> db.Resultados.countDocuments();
15
shardrs1 [direct: primary] torneo_deportivo> db.tablaposiciones.countDocuments();
15
shardrs1 [direct: primary] torneo_deportivo> db.jugadores.countDocuments();
```

```
0
shardrs1 [direct: primary] torneo_deportivo> for (let i = 0; i < 15; i++) {
...   db.Jugadores.insertOne({
...     "Id": i,
...     "Nombre": "Jugador " + i,
...     "Edad": Math.floor(Math.random() * 40) + 18,
...     "Posición": "Posición " + i,
...     "Equipo": "Equipo " + i
...   });
... }
{
  acknowledged: true,
  insertedId: ObjectId("64893bc8e39fbe0fed6e2aca")
}
shardrs1 [direct: primary] torneo_deportivo> db.jugadores.countDocuments();
0
shardrs1 [direct: primary] torneo_deportivo> db.Jugadores.countDocuments();
50728
shardrs1 [direct: primary] torneo_deportivo> db = (new Mongo("localhost:27002")).getDB("torneo_deportivo");
torneo_deportivo
shardrs1 [direct: secondary] torneo_deportivo> for (let i = 0; i < 15; i++) {
...   db.Jugadores.insertOne({
...     "Id": i,
...     "Nombre": "Jugador " + i,
...     "Edad": Math.floor(Math.random() * 40) + 18,
...     "Posición": "Posición " + i,
...     "Equipo": "Equipo " + i
...   });
... }
MongoServerError: not primary
shardrs1 [direct: secondary] torneo_deportivo> rs.conf()
{
  _id: 'shardrs1',
  version: 1,
  term: 1,
  members: [
    {
      _id: 0,
      host: 'localhost:27001',
      arbiterOnly: false,
      buildIndexes: true,
      hidden: false,
      priority: 1,
      tags: {},
      secondaryDelaySecs: Long("0"),
      votes: 1
    },
    {
```

```
    },
    {
      _id: 1,
      host: 'localhost:27002',
      arbiterOnly: false,
      buildIndexes: true,
      hidden: false,
      priority: 1,
      tags: {},
      secondaryDelaySecs: Long("0"),
      votes: 1
    },
    {
      _id: 2,
      host: 'localhost:27003',
      arbiterOnly: false,
      buildIndexes: true,
      hidden: false,
      priority: 1,
      tags: {},
      secondaryDelaySecs: Long("0"),
      votes: 1
    }
  ],
  protocolVersion: Long("1"),
  writeConcernMajorityJournalDefault: true,
  settings: {
    chainingAllowed: true,
    heartbeatIntervalMillis: 2000,
    heartbeatTimeoutSecs: 10,
    electionTimeoutMillis: 10000,
    catchUpTimeoutMillis: -1,
    catchUpTakeoverDelayMillis: 30000,
    getLastErrorModes: {},
    getLastErrorDefaults: { w: 1, wtimeout: 0 },
    replicaSetId: ObjectId("648938969d31e4bcdf964f2c")
  }
}
shardrs1 [direct: secondary] torneo_deportivo> Shard1 = new Mongo("localhost:27001")
mongodb://localhost:27001/?directConnection=true&serverSelectionTimeoutMS=2000
shardrs1 [direct: secondary] torneo_deportivo> Sh.enablesharding(“Torneo_deportivo”)
Uncaught:
SyntaxError: Unexpected character '“'. (1:18)

> 1 | Sh.enablesharding(“Torneo_deportivo”)
    |                   ^
  2 |

shardrs1 [direct: secondary] torneo_deportivo> Sh.status()
ReferenceError: Sh is not defined
```

```
shardrs1 [direct: secondary] torneo_deportivo> sh.status()
Warning: MongoshWarning: [SHAPI-10003] You are not connected to a mongos. This command may not work as expected.
MongoServerError: not primary and secondaryOk=false - consider using db.getMongo().setReadPref() or readPreference in the connection string
shardrs1 [direct: secondary] torneo_deportivo> rs.conf()
{
  _id: 'shardrs1',
  version: 1,
  term: 1,
  members: [
    {
      _id: 0,
      host: 'localhost:27001',
      arbiterOnly: false,
      buildIndexes: true,
      hidden: false,
      priority: 1,
      tags: {},
      secondaryDelaySecs: Long("0"),
      votes: 1
    },
    {
      _id: 1,
      host: 'localhost:27002',
      arbiterOnly: false,
      buildIndexes: true,
      hidden: false,
      priority: 1,
      tags: {},
      secondaryDelaySecs: Long("0"),
      votes: 1
    },
    {
      _id: 2,
      host: 'localhost:27003',
      arbiterOnly: false,
      buildIndexes: true,
      hidden: false,
      priority: 1,
      tags: {},
      secondaryDelaySecs: Long("0"),
      votes: 1
    }
  ],
  protocolVersion: Long("1"),
  writeConcernMajorityJournalDefault: true,
  settings: {
    chainingAllowed: true,
    heartbeatIntervalMillis: 2000,
    heartbeatTimeoutSecs: 10,
    electionTimeoutMillis: 10000,
    catchUpTimeoutMillis: -1,
```

```
}
shardrs1 [direct: secondary] torneo_deportivo> rs.initiate(config)
BSONError: cyclic dependency detected
shardrs1 [direct: secondary] torneo_deportivo> config = {
...   _id: "configsrv",
...   configsvr: true,
...   members: [
...     { _id: 0, host: "localhost:27019" },
...     { _id: 1, host: "localhost:27020" },
...     { _id: 2, host: "localhost:27021" }
...   ]
... };
{
  _id: 'configsrv',
  configsvr: true,
  members: [
    { _id: 0, host: 'localhost:27019' },
    { _id: 1, host: 'localhost:27020' },
    { _id: 2, host: 'localhost:27021' }
  ]
}
shardrs1 [direct: secondary] torneo_deportivo> rs.initiate(config);
MongoServerError: already initialized
shardrs1 [direct: secondary] torneo_deportivo>
```

```
shardrs1 [direct: secondary] test> primary = rs.status().members.filter(member => member.stateStr ===
"PRIMARY")[0].name;
localhost:27002
shardrs1 [direct: secondary] test> exit

C:\Users\gamerpc>mongosh --host localhost:27002
Current Mongosh Log ID: 648a774a0c22a01d99eed87b
Connecting to:          mongodb://localhost:27002/?directConnection=true&serverSelectionTimeoutMS=2000
&appName=mongosh+1.6.1
Using MongoDB:          6.0.3
Using Mongosh:          1.6.1

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

------
   The server generated these startup warnings when booting
   2023-06-14T21:18:28.611-05:00: Access control is not enabled for the database. Read and write acces
s to data and configuration is unrestricted
   2023-06-14T21:18:28.611-05:00: This server is bound to localhost. Remote systems will be unable to
connect to this server. Start the server with --bind_ip <address> to specify which IP addresses it sho
uld serve responses from, or with --bind_ip_all to bind to all interfaces. If this behavior is desired
, start the server with --bind_ip 127.0.0.1 to disable this warning
------

------
   Enable MongoDB's free cloud-based monitoring service, which will then receive and display
   metrics about your deployment (disk utilization, CPU, operation statistics, etc).

   The monitoring data will be available on a MongoDB website with a unique URL accessible to you
   and anyone you share the URL with. MongoDB may use this information to make product
   improvements and to suggest MongoDB products and deployment options to you.

   To enable free monitoring, run the following command: db.enableFreeMonitoring()
   To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
------

shardrs1 [direct: primary] test>
```