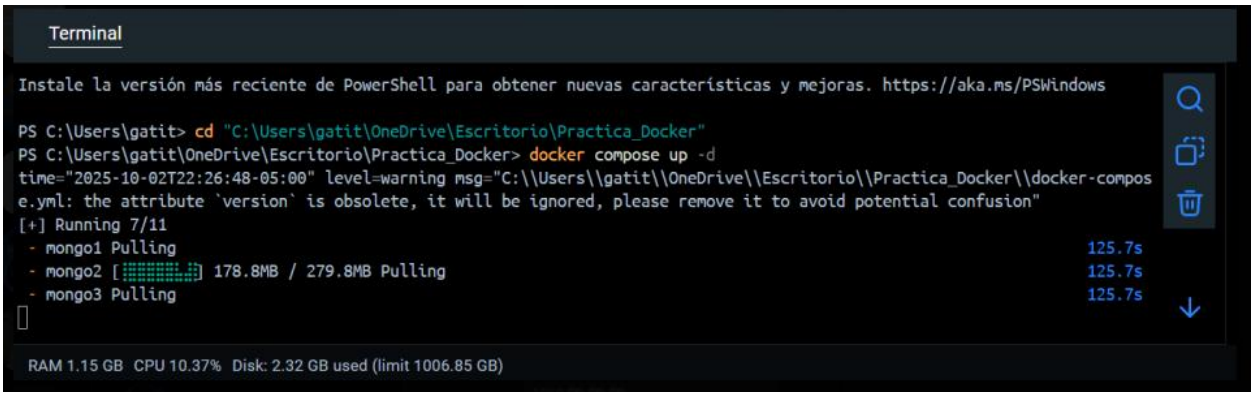
UNIVERSIDAD TECNICA DE AMBATO

INTEGRANTES:

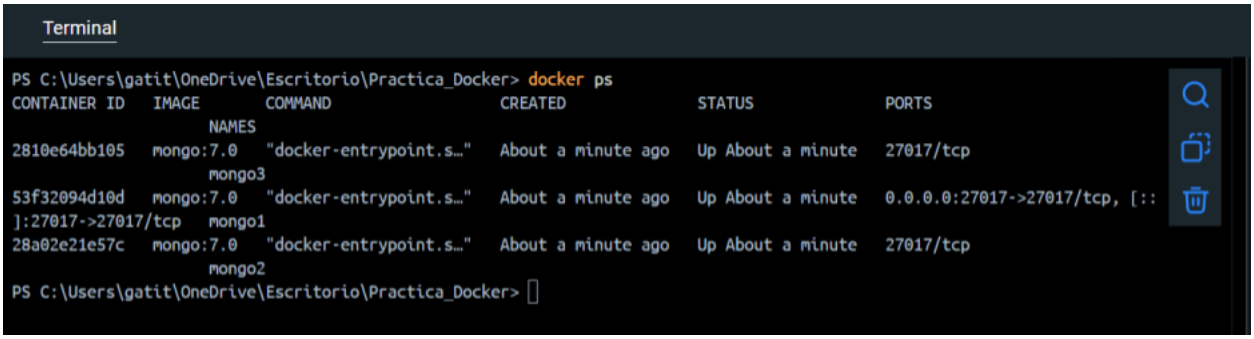
- Aldas Ismael
- Caguasango Alex
- Gómez Luis
- Paredes Luis

TALLER DOCKER

Ejecutar y levantar el clúster de MongoDB

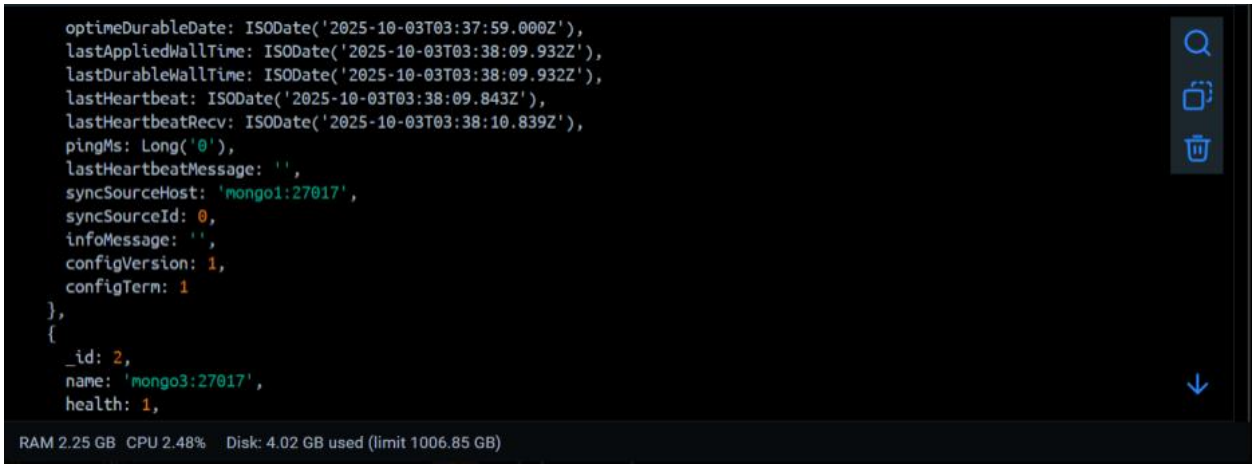


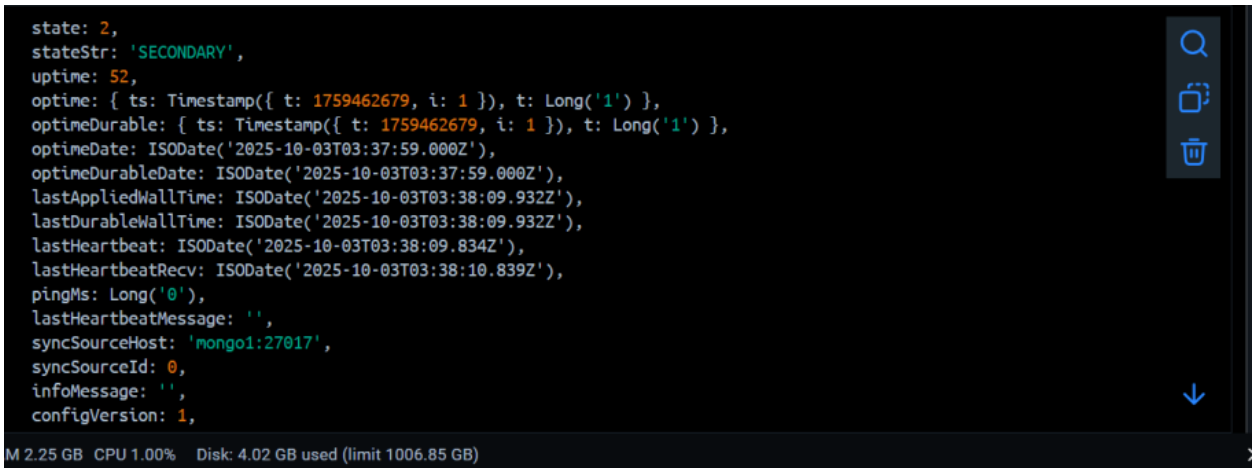
Verificar que los contenedores estén activos



Verificar que se haya configurado correctamente


electionTime: Timestamp({ t: 1759462649, i: 1 }),
electionDate: ISODate('2025-10-03T03:37:29.000Z'),
configVersion: 1,
configTerm: 1,
self: true,
lastHeartbeatMessage: ''
},
{
_id: 1,
name: 'mongo2:27017',
health: 1,
state: 2,
stateStr: 'SECONDARY',
uptime: 52,
optime: { ts: Timestamp({ t: 1759462679, i: 1 }), t: Long('1') },
optimeDurable: { ts: Timestamp({ t: 1759462679, i: 1 }), t: Long('1') },
optimeDate: ISODate('2025-10-03T03:37:59.000Z'),
RAM 2.24 GB CPU 0.50% Disk: 4.02 GB used (limit 1006.85 GB)


optimeDurableDate: ISODate('2025-10-03T03:37:59.000Z'),
lastAppliedWallTime: ISODate('2025-10-03T03:38:09.932Z'),
lastDurableWallTime: ISODate('2025-10-03T03:38:09.932Z'),
lastHeartbeat: ISODate('2025-10-03T03:38:09.843Z'),
lastHeartbeatRecv: ISODate('2025-10-03T03:38:10.839Z'),
pingMs: Long('0'),
lastHeartbeatMessage: '',
syncSourceHost: 'mongo1:27017',
syncSourceId: 0,
infoMessage: '',
configVersion: 1,
configTerm: 1
},
{
_id: 2,
name: 'mongo3:27017',
health: 1,
RAM 2.25 GB CPU 2.48% Disk: 4.02 GB used (limit 1006.85 GB)

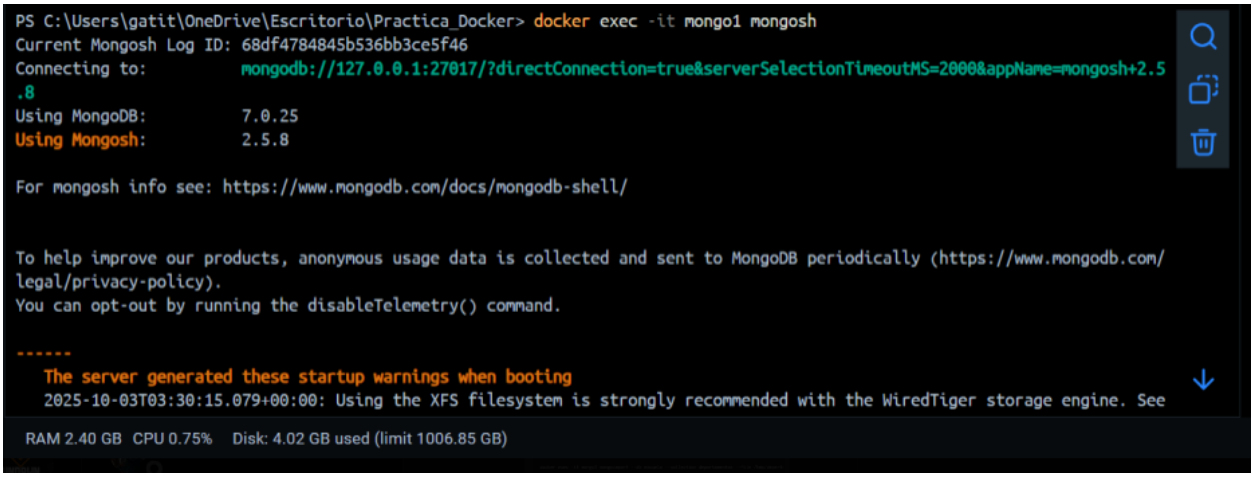

state: 2,
stateStr: 'SECONDARY',
uptime: 52,
optime: { ts: Timestamp({ t: 1759462679, i: 1 }), t: Long('1') },
optimeDurable: { ts: Timestamp({ t: 1759462679, i: 1 }), t: Long('1') },
optimeDate: ISODate('2025-10-03T03:37:59.000Z'),
optimeDurableDate: ISODate('2025-10-03T03:37:59.000Z'),
lastAppliedWallTime: ISODate('2025-10-03T03:38:09.932Z'),
lastDurableWallTime: ISODate('2025-10-03T03:38:09.932Z'),
lastHeartbeat: ISODate('2025-10-03T03:38:09.834Z'),
lastHeartbeatRecv: ISODate('2025-10-03T03:38:10.839Z'),
pingMs: Long('0'),
lastHeartbeatMessage: '',
syncSourceHost: 'mongo1:27017',
syncSourceId: 0,
infoMessage: '',
configVersion: 1,
M 2.25 GB CPU 1.00% Disk: 4.02 GB used (limit 1006.85 GB)

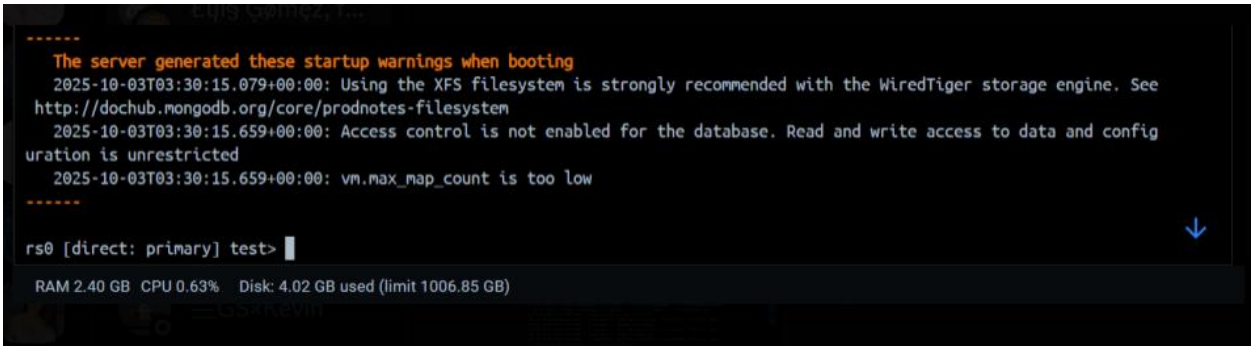

configTerm: 1
}
],
ok: 1,
'$clusterTime': {
clusterTime: Timestamp({ t: 1759462689, i: 1 }),
signature: {
hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
keyId: Long('0')
}
},
operationTime: Timestamp({ t: 1759462689, i: 1 })
}
PS C:\Users\gatit\OneDrive\Escritorio\Practica_Docker>
RAM 2.25 GB CPU 2.50% Disk: 4.02 GB used (limit 1006.85 GB)

Copiamos los archivos. JSON al contenedor antes de importarlos

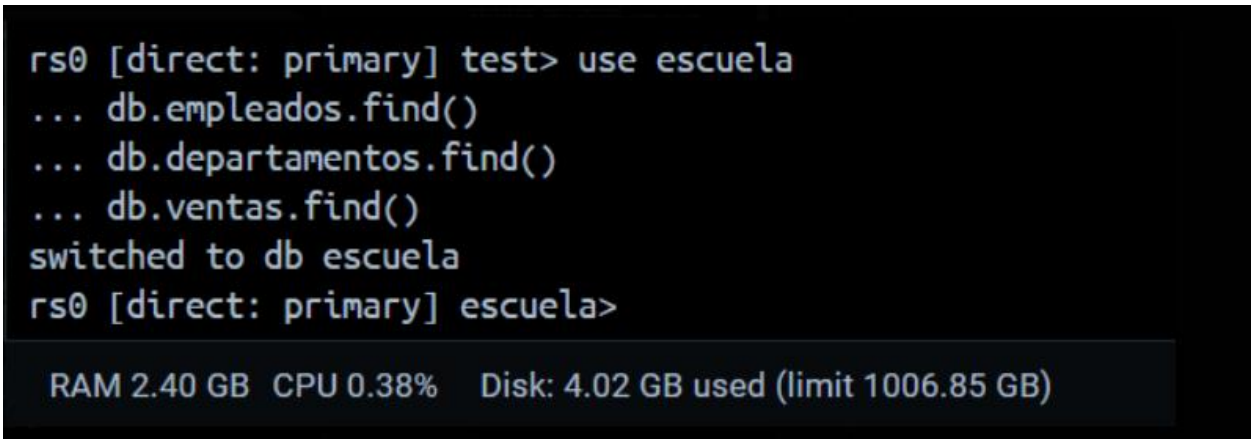
Ejecutamos **mongonimport** dentro del contenedor a utilizar con la ruta /**tmp**

Ingresamos a la Shell de **MongoDB**





Una vez dentro, con el comando **use escuela (que será el nombre de la base de datos),** se podrá visualizar los datos que están guardados.

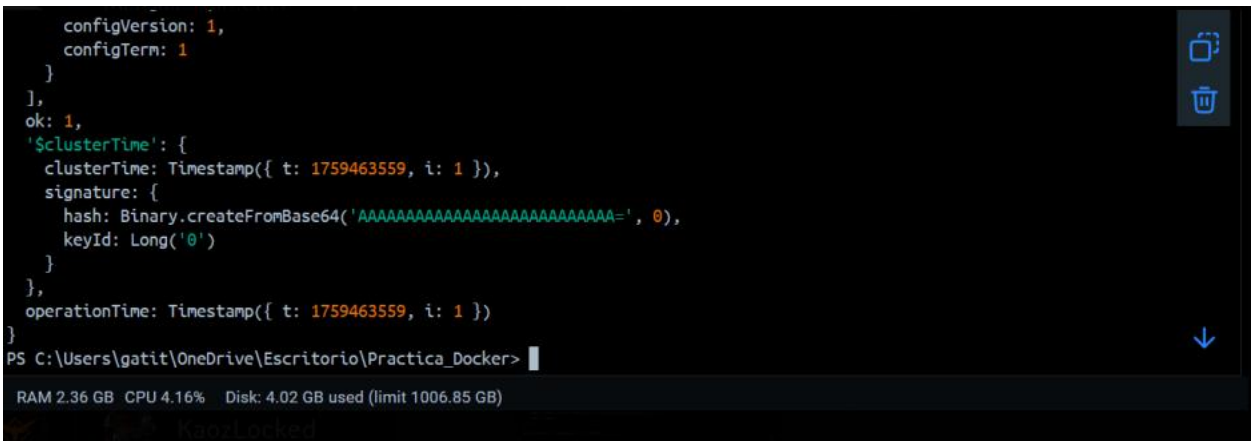


Luego revisamos el estado de REPLICA SET inicializado.


appliedOpTime: { ts: Timestamp({ t: 1759463559, i: 1 }), t: Long('1') },
durableOpTime: { ts: Timestamp({ t: 1759463559, i: 1 }), t: Long('1') },
lastAppliedWallTime: ISODate('2025-10-03T03:52:39.912Z'),
lastDurableWallTime: ISODate('2025-10-03T03:52:39.912Z')
},
lastStableRecoveryTimestamp: Timestamp({ t: 1759463529, i: 1 }),
electionCandidateMetrics: {
lastElectionReason: 'electionTimeout',
lastElectionDate: ISODate('2025-10-03T03:37:29.793Z'),
electionTerm: Long('1'),
lastCommittedOpTimeAtElection: { ts: Timestamp({ t: 1759462638, i: 1 }), t: Long('-1') },
lastSeenOpTimeAtElection: { ts: Timestamp({ t: 1759462638, i: 1 }), t: Long('-1') },
numVotesNeeded: 2,
priorityAtElection: 1,
electionTimeoutMillis: Long('10000'),
numCatchUpOps: Long('0'),
newTermStartDate: ISODate('2025-10-03T03:37:29.888Z'),
RAM 2.36 GB CPU 0.00% Disk: 4.02 GB used (limit 1006.85 GB)


wMajorityWriteAvailabilityDate: ISODate('2025-10-03T03:37:30.407Z')
},
members: [
{
_id: 0,
name: 'mongo1:27017',
health: 1,
state: 1,
stateStr: 'PRIMARY',
uptime: 1346,
optime: { ts: Timestamp({ t: 1759463559, i: 1 }), t: Long('1') },
optimeDate: ISODate('2025-10-03T03:52:39.000Z'),
lastAppliedWallTime: ISODate('2025-10-03T03:52:39.912Z'),
lastDurableWallTime: ISODate('2025-10-03T03:52:39.912Z'),
syncSourceHost: '',
syncSourceId: -1,
infoMessage: '',
RAM 2.36 GB CPU 0.63% Disk: 4.02 GB used (limit 1006.85 GB)


electionTime: Timestamp({ t: 1759462649, i: 1 }),
electionDate: ISODate('2025-10-03T03:37:29.000Z'),
configVersion: 1,
optimeDurableDate: ISODate('2025-10-03T03:52:39.000Z'),
lastAppliedWallTime: ISODate('2025-10-03T03:52:39.912Z'),
lastDurableWallTime: ISODate('2025-10-03T03:52:39.912Z'),
lastHeartbeat: ISODate('2025-10-03T03:52:40.374Z'),
lastHeartbeatRecv: ISODate('2025-10-03T03:52:40.823Z'),
pingMs: Long('0'),
lastHeartbeatMessage: '',
syncSourceHost: 'mongo1:27017',
syncSourceId: 0,
infoMessage: '',
configVersion: 1,
configTerm: 1
},
{
_id: 2,
name: 'mongo3:27017',
health: 1,
RAM 2.36 GB CPU 0.38% Disk: 4.02 GB used (limit 1006.85 GB)



EJECUCION DE CONSULTAS.

Para la ejecución de consultas, se debe verificar que el entorno este levantado correctamente , que todos sus contenedores estén funcionando correctamente y tener replica set inicializado , sin olvidarse de que los datos deben estar importados.Con todo esto preparado entramos a la consola interactiva de Mongo.

Ingresamos a la Base de Datos

Agregamos los pipelines de forma directa

1.Empleado con salario más alto

```
rs0 [direct: primary] escuela> db.empleados.aggregate([
...   { $sort: { salario: -1 } },
...   { $limit: 1 }
... ])
[ { _id: 6, nombre: 'Frank', salario: 2000, departamento_id: 2 } ]
rs0 [direct: primary] escuela>
```

RAM 2.51 GB CPU 3.23% Disk: 4.03 GB used (limit 1006.85 GB)

2.Departamentos sin empleados asignados

```
rs0 [direct: primary] escuela> db.departamentos.aggregate([
...   { $lookup: {
...       from: "empleados",
...       localField: "_id",
...       foreignField: "departamento_id",
...       as: "empleados"
...   }},
...   { $addFields: { count: { $size: "$empleados" } }},
...   { $match: { count: 0 } }
... ])
[ { _id: 4, nombre: 'Logística', empleados: [], count: 0 } ]
rs0 [direct: primary] escuela>
```

RAM 2.51 GB CPU 11.34% Disk: 4.03 GB used (limit 1006.85 GB)

3. Sucursal que mas ventas tiene por mes

```
rs0 [direct: primary] escuela> db.ventas.aggregate([
...   { $group: { _id: { mes: "$_id.mes", sucursal: "$_id.sucursal" }, total: { $sum: "$total" } } },
...   { $sort: { "_id.mes": 1, total: -1 } },
...   { $group: { _id: "$_id.mes", topSucursal: { $first: "$_id.sucursal" }, maxTotal: { $first: "$total" } } }
... ])
...
[
  { _id: '2025-09', topSucursal: 'Quito', maxTotal: 39000 },
  { _id: '2025-08', topSucursal: 'Guayaquil', maxTotal: 42000 }
]
rs0 [direct: primary] escuela>
```

- Empleados con mayor salario promedio en la empresa
  (Con ventanas)

```
rs0 [direct: primary] escuela> db.empleados.aggregate([
...   {
...     $setWindowFields: {
...       partitionBy: null,
...       output: {
```


promEmpresa: 1183.3333333333333
},
{
_id: 6,
nombre: 'Frank',
salario: 2000,
departamento_id: 2,
promEmpresa: 1183.3333333333333
},
{
_id: 3,
nombre: 'Carla',
salario: 1500,
departamento_id: 2,
promEmpresa: 1183.3333333333333
}
]
RAM 2.52 GB CPU 0.25% Disk: 4.03 GB used (limit 1006.85 GB)

- Misma consulta (Sin ventanas)

```
rs0 [direct: primary] escuela> // Primero calcular el promedio global
... var promEmpresa = db.empleados.aggregate([
...   { $group: { _id: null, prom: { $avg: "$salario" } } }
... ]).toArray()[0].prom;
...
... // Luego filtrar con $expr
... db.empleados.aggregate([
...   { $match: { $expr: { $gt: ["$salario", promEmpresa] } } }
... ])
[
  { _id: 1, nombre: 'Ana', salario: 1200, departamento_id: 1 },
  { _id: 6, nombre: 'Frank', salario: 2000, departamento_id: 2 },
  { _id: 3, nombre: 'Carla', salario: 1500, departamento_id: 2 }
]
rs0 [direct: primary] escuela>
```

RAM 2.52 GB CPU 0.62% Disk: 4.03 GB used (limit 1006.85 GB)

- Empleado con el salario mas alto

```
rs0 [direct: primary] escuela> db.empleados.aggregate([
...   { $sort: { salario: -1 } },
...   { $limit: 1 }
... ])
[ { _id: 6, nombre: 'Frank', salario: 2000, departamento_id: 2 } ]
rs0 [direct: primary] escuela>
```

RAM 2.52 GB CPU 0.76% Disk: 4.03 GB used (limit 1006.85 GB)

- Mostrar el salario promedio de un departamento para cada empleado

```
rs0 [direct: primary] escuela> db.empleados.aggregate([
...   {
...     $setWindowFields: {
...       partitionBy: "$departamento_id",
...       output: {
...         promDep: { $avg: "$salario" }
...       }
...     }
...   },
...   { $project: { nombre: 1, salario: 1, departamento_id: 1, promDep: 1 } }
... ])
[
  {
    _id: 1,
    nombre: 'Ana',
    salario: 1200,
    departamento_id: 1,
RAM 2.52 GB  CPU 2.14%  Disk: 4.03 GB used (limit 1006.85 GB)
```

```
    departamento_id: 2,
    promDep: 1433.3333333333333
  },
  {
    _id: 3,
    nombre: 'Carla',
    salario: 1500,
    departamento_id: 2,
    promDep: 1433.3333333333333
  },
  {
    _id: 5,
    nombre: 'Elena',
    salario: 700,
    departamento_id: 3,
    promDep: 700
  }
```

- Departamentos cuyo promedio salarial es mayo al promedio general

```
rs0 [direct: primary] escuela> db.empleados.aggregate([
...   {
...     $setWindowFields: {
...       partitionBy: "$departamento_id",
...       output: {
...         promDep: { $avg: "$salario" }
...       }
...     }
...   },
...   {
...     $setWindowFields: {
...       partitionBy: null,
...       output: {
...         promGlobal: { $avg: "$salario" }
...       }
...     }
...   },
...   },
...   { $match: { $expr: { $gt: ["$promDep", "$promGlobal"] } } },
...   { $group: { _id: "$departamento_id" } }
... ])
...
[ { _id: 2 } ]
rs0 [direct: primary] escuela>
```

- Departamentos cuyo promedio salarial es mayo al promedio general (Método sin ventanas)

```
rs0 [direct: primary] escuela> // Calcular promedio global
... var promGlobal = db.empleados.aggregate([
...   { $group: { _id: null, prom: { $avg: "$salario" } } }
... ]).toArray()[0].prom;
...
... // Promedio por departamento y comparación
... db.empleados.aggregate([
...   { $group: { _id: "$departamento_id", promDep: { $avg: "$salario" } } },
...   { $match: { promDep: { $gt: promGlobal } } }
... ])
...
[ { _id: 2, promDep: 1433.3333333333333 } ]
rs0 [direct: primary] escuela>
RAM 2.53 GB  CPU 1.63%  Disk: 4.03 GB used (limit 1006.85 GB)
```

- Método con $topN (si la versión lo soporta)

Prueba de Resiliencia

Detenemos un nodo

Con **rs.status** nos conectamos al REPLICA SET y verificamos que mongo3 está apagado

```
    lastHeartbeat: ISODate('2025-10-03T16:13:39.220Z'),
    lastHeartbeatRecv: ISODate('2025-10-03T16:13:37.611Z'),
    pingMs: Long('0'),
    lastHeartbeatMessage: '',
    syncSourceHost: '',
    syncSourceId: -1,
    infoMessage: '',
    electionTime: Timestamp({ t: 1759506482, i: 1 }),
    electionDate: ISODate('2025-10-03T15:48:02.000Z'),
    configVersion: 1,
    configTerm: 2
  },
  {
    _id: 2,
    name: 'mongo3:27017',
    health: 0,
    state: 8,
```

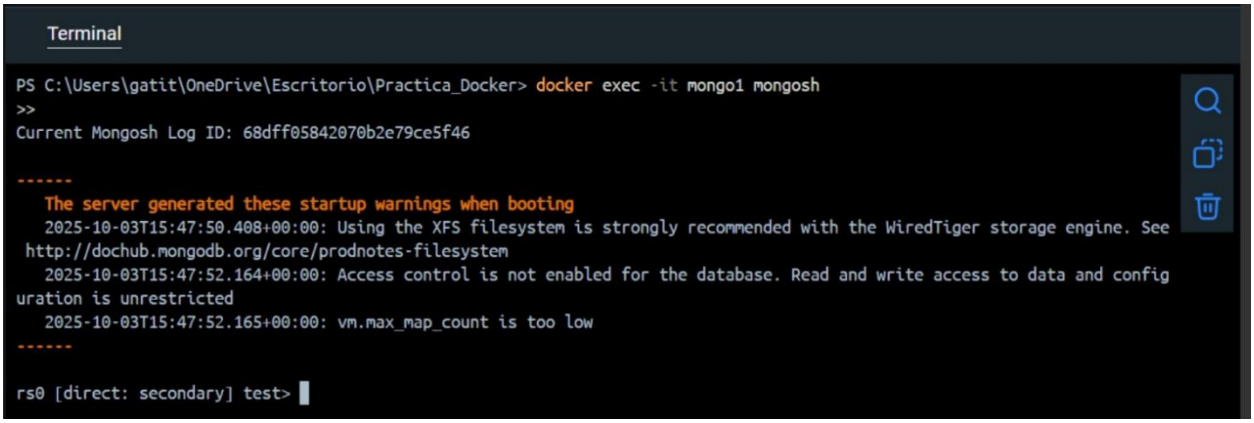
RAM 2.39 GB CPU 0.63% Disk: 4.03 GB used (limit 1006.85 GB)

```
    stateStr: '(not reachable/healthy)',
    uptime: 0,
    optime: { ts: Timestamp({ t: 0, i: 0 }), t: Long('-1') },
    optimeDurable: { ts: Timestamp({ t: 0, i: 0 }), t: Long('-1') },
    optimeDate: ISODate('1970-01-01T00:00:00.000Z'),
    optimeDurableDate: ISODate('1970-01-01T00:00:00.000Z'),
    lastAppliedWallTime: ISODate('2025-10-03T15:59:12.786Z'),
    lastDurableWallTime: ISODate('2025-10-03T15:59:12.786Z'),
    lastHeartbeat: ISODate('2025-10-03T16:13:34.294Z'),
    lastHeartbeatRecv: ISODate('2025-10-03T15:59:13.673Z'),
    pingMs: Long('0'),
    lastHeartbeatMessage: 'Error connecting to mongo3:27017 :: caused by :: Could not find address for mongo3:27017: Soc
ketException: onInvoke :: caused by :: Host not found (authoritative)',
    syncSourceHost: '',
    syncSourceId: -1,
    infoMessage: '',
    configVersion: 1,
```

RAM 2.39 GB CPU 3.94% Disk: 4.03 GB used (limit 1006.85 GB)

```
    configVersion: 1,
    configTerm: 2
  }
 ],
 ok: 1,
 '$clusterTime': {
  clusterTime: Timestamp({ t: 1759508012, i: 1 }),
  signature: {
   hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
   keyId: Long('0')
  }
 },
 operationTime: Timestamp({ t: 1759508012, i: 1 })
}
rs0 [direct: secondary] escuela>
```

Ejecutamos el contenedor del cliente de MongoDB



Realizamos la consulta

En este caso no sucederá nada ya que como el nodo primario esta activo las escrituras y lecturas continuaran sin ningún problema, únicamente habrá algún fallo si en una consulta se desea leer específicamente del nodo apagado o detenido.

De ser que el nodo primario se detenga, habrá un breve lapsus de pausa mientras uno de los nodos secundarios toma el rol , luego de esto el trabajo continuara con normalidad.