TUGAS 4 BASIS DATA LANJUT
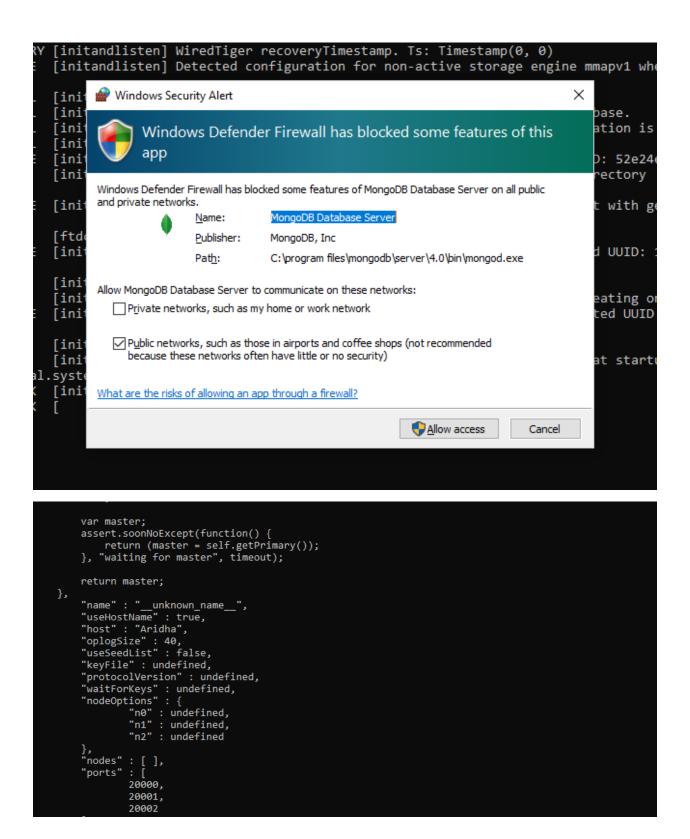
NAMA   : Widyatna Dahniah

NIM   : 17050623025

Prodi   : D3 Manajemen Informatika

REPLACE MONGODB

```
> replicaSet = new ReplSetTest({"nodes" : 3})
Starting new replica set __unknown_name__
{
        "kDefaultTimeoutMS" : 600000,
        "getReadConcernMajorityOpTimeOrThrow" : function (conn) {
"use strict";

        const majorityOpTime = _getReadConcernMajorityOpTime(conn);
        if (friendlyEqual(majorityOpTime, {ts: Timestamp(0, 0), t: NumberLong(0)})) {
            throw new Error("readConcern majority optime not available");
        }
        return majorityOpTime;
    },
        "nodeList" : function () {
"use strict";

        var list = [];
        for (var i = 0; i < this.ports.length; i++) {
            list.push(this.host + ":" + this.ports[i]);
        }

        return list;
    },
        "getNodeId" : function (node) {
"use strict";

        if (node.toFixed) {
            return parseInt(node);
        }

        for (var i = 0; i < this.nodes.length; i++) {
```

```
> replicaSet.startSet()
ReplSetTest starting set
ReplSetTest n is : 0
{
        "useHostName" : true,
        "oplogSize" : 40,
        "keyFile" : undefined,
        "port" : 20000,
        "noprealloc" : "",
        "smallfiles" : "",
        "replSet" : "__unknown_name__",
        "dbpath" : "$set-$node",
        "restart" : undefined,
        "pathOpts" : {
                "node" : 0,
                "set" : "__unknown_name__"
        },
        "setParameter" : {
                "writePeriodicNoops" : false,
                "numInitialSyncAttempts" : 1,
                "numInitialSyncConnectAttempts" : 60
        }
}
ReplSetTest Starting....
Resetting db path '/data/db/__unknown_name__-0'
2019-04-25T11:52:41.194+0700 I -        [js] shell: started program (sh6564):  C:\Program Files\MongoDB\Server\4.0\bin\mongod.exe --oplogSize 40 --port 20000 --
```

```
RY [initandlisten] WiredTiger recoveryTimestamp. Ts: Timestamp(0, 0)
E [initandlisten] Detected configuration for non-active storage engine mmapv1 whe
```

```
      [ini
      [ini                                                    base.
      [ini                                                    ation is
      [ini
E     [ini                                                    D: 52e24
      [ini                                                    rectory
E     [ini                                                    t with g
      [ftd
E     [ini                                                    d UUID:
      [ini
      [ini                                                    eating or
E     [ini                                                    ted UUID
      [ini
      [ini                                                    at startu
al.syste
X     [ini
X     [
```

Windows Security Alert ✕

🛡️ **Windows Defender Firewall has blocked some features of this app**

Windows Defender Firewall has blocked some features of MongoDB Database Server on all public and private networks.

Name: MongoDB Database Server
Publisher: MongoDB, Inc
Path: C:\program files\mongodb\server\4.0\bin\mongod.exe

Allow MongoDB Database Server to communicate on these networks:

☐ Private networks, such as my home or work network

☑ Public networks, such as those in airports and coffee shops (not recommended because these networks often have little or no security)

What are the risks of allowing an app through a firewall?

🛡️Allow access   Cancel

```
        var master;
        assert.soonNoExcept(function() {
            return (master = self.getPrimary());
        }, "waiting for master", timeout);

        return master;
    },
        "name" : "__unknown_name__",
        "useHostName" : true,
        "host" : "Aridha",
        "oplogSize" : 40,
        "useSeedList" : false,
        "keyFile" : undefined,
        "protocolVersion" : undefined,
        "waitForKeys" : undefined,
        "nodeOptions" : {
                "n0" : undefined,
                "n1" : undefined,
                "n2" : undefined
        },
        "nodes" : [ ],
        "ports" : [
                20000,
                20001,
                20002
```

```
> replicaSet.initiate()
{
        "replSetInitiate" : {
                "_id" : "__unknown_name__",
                "protocolVersion" : 1,
                "members" : [
                        {
                                "_id" : 0,
                                "host" : "Aridha:20000"
                        }
                ]
        }
}
d20000| 2019-04-25T12:02:10.322+0700 I REPL     [conn1] replSetInitiate admin command received from client
d20000| 2019-04-25T12:02:10.340+0700 I REPL     [conn1] replSetInitiate config object with 1 members parses ok
d20000| 2019-04-25T12:02:10.354+0700 I REPL     [conn1] ******
d20000| 2019-04-25T12:02:10.354+0700 I REPL     [conn1] creating replication oplog of size: 40MB...
d20000| 2019-04-25T12:02:10.354+0700 I STORAGE  [conn1] createCollection: local.oplog.rs with generated UUID: 40725cc7-6554-4e5a-9cc0-83d744e96c8d
d20000| 2019-04-25T12:02:10.403+0700 I STORAGE  [conn1] Starting OplogTruncaterThread local.oplog.rs
d20000| 2019-04-25T12:02:10.404+0700 I STORAGE  [conn1] The size storer reports that the oplog contains 0 records totaling to 0 bytes
d20000| 2019-04-25T12:02:10.404+0700 I STORAGE  [conn1] Scanning the oplog to determine where to place markers for truncation
d20000| 2019-04-25T12:02:10.580+0700 I REPL     [conn1] ******
d20000| 2019-04-25T12:02:10.580+0700 I STORAGE  [conn1] createCollection: local.system.replset with generated UUID: 6de3a410-eff1-4afb-9377-37d79c7a09b1
d20000| 2019-04-25T12:02:10.581+0700 I COMMAND  [monitoring keys for HMAC] command admin.system.keys command: find { find: "system.keys", filter: { purpose: "HM
AC", expiresAt: { $gt: Timestamp(0, 0) } }, sort: { expiresAt: 1 }, $readPreference: { mode: "nearest", tags: [] }, $db: "admin" } planSummary: EOF keysExamined
:0 docsExamined:0 cursorExhausted:1 numYields:0 nreturned:0 reslen:231 locks:{ Global: { acquireCount: { r: 2 }, acquireWaitCount: { r: 1 }, timeAcquiringMicros
```

```
> for (i=0; i<1000; i++) { primaryDB.coll.insert({count: 1}) }
d20000| 2019-04-25T12:55:34.363+0700 I STORAGE  [conn19] createCollection: test.coll with generated UUID: aec8e6ee-3ba4-460d-9f9f-5247f62a2baa
d20000| 2019-04-25T12:55:34.637+0700 I COMMAND  [conn19] command test.coll appName: "MongoDB Shell" command: insert { insert: "coll", ordered: true, lsid: { id: UUID(
"05a5809d-5793-47de-b1db-ac3ef90526c9" ) }, $clusterTime: { clusterTime: Timestamp(1556170303, 1), signature: { hash: BinData(0, 0000000000000000000000000000000000000000
00), keyId: 0 } }, $db: "test" } ninserted:1 keysInserted:1 numYields:0 reslen:230 locks:{ Global: { acquireCount: { r: 3, w: 3 } }, Database: { acquireCount: { w: 2,
W: 1 } }, Collection: { acquireCount: { w: 2 } } } protocol:op_msg 274ms
d20001| 2019-04-25T12:55:34.730+0700 I STORAGE  [repl writer worker 10] createCollection: test.coll with provided UUID: aec8e6ee-3ba4-460d-9f9f-5247f62a2baa
d20002| 2019-04-25T12:55:34.730+0700 I STORAGE  [repl writer worker 10] createCollection: test.coll with provided UUID: aec8e6ee-3ba4-460d-9f9f-5247f62a2baa
d20001| 2019-04-25T12:55:35.142+0700 I REPL     [repl writer worker 10] applied op: command { ts: Timestamp(1556171734, 1), t: 1, h: 3867649606046479784, v: 2, op: "c
", ns: "test.$cmd", ui: UUID("aec8e6ee-3ba4-460d-9f9f-5247f62a2baa"), wall: new Date(1556171734636), o: { create: "coll", idIndex: { v: 2, key: { _id: 1 }, name: "_id
_", ns: "test.coll" } } }, took 414ms
d20002| 2019-04-25T12:55:35.274+0700 I REPL     [repl writer worker 10] applied op: command { ts: Timestamp(1556171734, 1), t: 1, h: 3867649606046479784, v: 2, op: "c
", ns: "test.$cmd", ui: UUID("aec8e6ee-3ba4-460d-9f9f-5247f62a2baa"), wall: new Date(1556171734636), o: { create: "coll", idIndex: { v: 2, key: { _id: 1 }, name: "_id
_", ns: "test.coll" } } }, took 547ms
d20000| 2019-04-25T12:55:35.469+0700 I NETWORK  [conn18] end connection 127.0.0.1:53399 (8 connections now open)
WriteResult({ "nInserted" : 1 })
> primaryDB.coll.count()
1000
```

```
> conn2 = new Mongo("localhost:20002")
d20002| 2019-04-25T13:16:54.607+0700 I NETWORK  [listener] connection accepted from 127.0.0.1:53412 #14 (4 connections now open)
d20002| 2019-04-25T13:16:54.608+0700 I NETWORK  [conn14] received client metadata from 127.0.0.1:53412 conn14: { application: { name: "MongoDB Sh
ell" }, driver: { name: "MongoDB Internal Client", version: "4.0.6" }, os: { type: "Windows", name: "Microsoft Windows 10", architecture: "x86_64
", version: "10.0 (build 17134)" } }
connection to localhost:20002
> secondaryDB = Conn2.getDB("test")
2019-04-25T13:17:17.993+0700 E QUERY    [js] ReferenceError: Conn2 is not defined :
@(shell):1:1
> secondaryDB = conn2.getDB("test")
test
> secondaryDB.coll.find()
Error: error: {
        "operationTime" : Timestamp(1556171736, 650),
        "ok" : 0,
        "errmsg" : "not master and slaveOk=false",
        "code" : 13435,
        "codeName" : "NotMasterNoSlaveOk",
        "$clusterTime" : {
                "clusterTime" : Timestamp(1556171736, 650),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        }
}
```

```
> conn1 = new Mongo("localhost:20000")
d20000| 2019-04-25T12:44:11.262+0700 I NETWORK  [listener] connection accepted from 127.0.0.1:53405 #19 (9 connections now open)
d20000| 2019-04-25T12:44:11.262+0700 I NETWORK  [conn19] received client metadata from 127.0.0.1:53405 conn19: { application: { name: "MongoDB Sh
ell" }, driver: { name: "MongoDB Internal Client", version: "4.0.6" }, os: { type: "Windows", name: "Microsoft Windows 10", architecture: "x86_64
", version: "10.0 (build 17134)" } }
connection to localhost:20000
> testReplSet:PRIMARY
2019-04-25T12:44:35.395+0700 E QUERY    [js] ReferenceError: PRIMARY is not defined :
@(shell):1:13
> testReplSet:PRIMARY>
... testReplSet:PRIMARY> primaryDB = conn1.getDB("test")
2019-04-25T12:45:21.712+0700 E QUERY    [js] SyntaxError: missing ; before statement @(shell):2:11
> testReplSet:PRIMARY> testReplSet:PRIMARY> primaryDB = conn1.getDB("test")
2019-04-25T12:45:32.072+0700 E QUERY    [js] SyntaxError: missing ; before statement @(shell):1:32
> testReplSet:PRIMARY> primaryDB = conn1.getDB("test")
2019-04-25T12:46:28.713+0700 E QUERY    [js] ReferenceError: invalid assignment left-hand side @(shell):1:12
> primaryDB = conn1.getDB("test")
test
> primaryDB.isMaster()
{
        "hosts" : [
                "DESKTOP-KPS9R2U:20000",
                "DESKTOP-KPS9R2U:20001",
                "DESKTOP-KPS9R2U:20002"
        ],
        "setName" : "__unknown_name__",
        "setVersion" : 2,
        "ismaster" : true,
        "secondary" : false,
        "primary" : "DESKTOP-KPS9R2U:20000",
        "me" : "DESKTOP-KPS9R2U:20000",
        "electionId" : ObjectId("7fffffff0000000000000001"),
        "lastWrite" : {
                "opTime" : {
                        "ts" : Timestamp(1556170303, 1),
                        "t" : NumberLong(1)
                },
                "lastWriteDate" : ISODate("2019-04-25T05:31:43Z"),
                "majorityOpTime" : {
                        "ts" : Timestamp(1556170303, 1),
                        "t" : NumberLong(1)
                },
                "majorityWriteDate" : ISODate("2019-04-25T05:31:43Z")
```

```
> conn2.setSlaveOk()
>
> secondaryDB.coll.find()
{ "_id" : ObjectId("5cc14bd6faef356afab69fd4"), "count" : 1 }
{ "_id" : ObjectId("5cc14bd6faef356afab69fcc"), "count" : 1 }
{ "_id" : ObjectId("5cc14bd6faef356afab69fcb"), "count" : 1 }
{ "_id" : ObjectId("5cc14bd6faef356afab69fce"), "count" : 1 }
{ "_id" : ObjectId("5cc14bd6faef356afab69fd0"), "count" : 1 }
{ "_id" : ObjectId("5cc14bd6faef356afab69fd6"), "count" : 1 }
{ "_id" : ObjectId("5cc14bd6faef356afab69fd5"), "count" : 1 }
{ "_id" : ObjectId("5cc14bd6faef356afab69fcd"), "count" : 1 }
{ "_id" : ObjectId("5cc14bd6faef356afab69fd2"), "count" : 1 }
{ "_id" : ObjectId("5cc14bd6faef356afab69fca"), "count" : 1 }
{ "_id" : ObjectId("5cc14bd6faef356afab69fd1"), "count" : 1 }
{ "_id" : ObjectId("5cc14bd6faef356afab69fd9"), "count" : 1 }
{ "_id" : ObjectId("5cc14bd6faef356afab69fda"), "count" : 1 }
{ "_id" : ObjectId("5cc14bd6faef356afab69fd3"), "count" : 1 }
{ "_id" : ObjectId("5cc14bd6faef356afab69fcf"), "count" : 1 }
{ "_id" : ObjectId("5cc14bd6faef356afab69fd7"), "count" : 1 }
{ "_id" : ObjectId("5cc14bd6faef356afab69fd8"), "count" : 1 }
{ "_id" : ObjectId("5cc14bd6faef356afab69fdf"), "count" : 1 }
{ "_id" : ObjectId("5cc14bd6faef356afab69feb"), "count" : 1 }
{ "_id" : ObjectId("5cc14bd6faef356afab69fee"), "count" : 1 }
Type "it" for more
> secondaryDB.coll.count()
1000
>
```