Centrify Server Suite 2016

Centrify Identity and Access Management for MongoDB

May 2016

Centrify Server Suite 2015.1

Mongo DB version 3.2

Legal notice

Contents

Benefits of integrating with Centrify

Centrify Server Suite is an enterprise-class solution that supports NoSQL MongoDB big data environments. Together, Centrify and MongoDB allow you to use your organization's existing Active Directory infrastructure to deliver access control and privilege management.

By installing the Centrify agent on NoSQL nodes you can provide single sign-on identity and access management for the users who will log on to clusters in the big data environment with their Active Directory credentials, by implementing Kerberos using GSSAPI.

The Centrify agent reduces identity-related risks by enforcing access controls and least-privilege security across nodes and clusters, and increases regulatory compliance through control over user access and the ability to trace activity back to an individual user.

Preparing for integration with Centrify

The following sections describe how to install and configure MongoDB for integration with the Centrify agent. For detailed instructions on preparing your NoSQL environment, go to http://www.mongodb.com/

Basic prerequisites

- Active Directory must be installed and at least one domain controller must be available.
- You should have a Windows workstation joined to the domain where you can run administrative consoles.
- You should have appropriate Centrify Server Suite Enterprise Edition software installed or available to be installed. For example, to install MongoDB on a machine running RedHat Enterprise Linux 6, you would download centrify-suite-2015.1-rhel: http://centrify.com/support/customer-support-portal/download-center/#2015.1

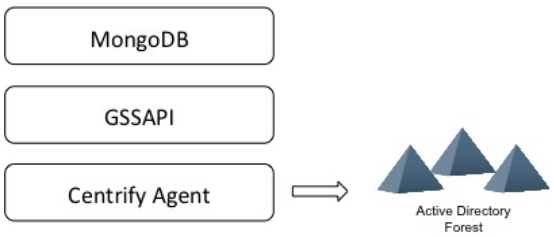
  You can request a free trial of Centrify Server Suite by filling out the http://www.centrify.com/free-trial/server-suite-form/ on the Centrify website and specifying `NoSQL` in the Comments field.
- You should have Centrify Server Suite documentation available for reference.

  You can download documentation from http://community.centrify.com/t5/custom/page/page-id/Centrify-Documentation after you register your free trial and set up your Centrify account.

- You should have the latest version of MongoDB software available. MongoDB Enterprise 3.2 or later can be downloaded from https://www.mongodb.com/lp/download/mongodb-enterprise?jmp=nav

Integrating MongoDB and Centrify

The following sections describe how to install and integrate your MongoDB environment with Centrify on a Linux machine running RedHat Enterprise Linux (RHEL) 6.0. The Centrify agent uses the Generic Security Service Application Program Interface (GSSAPI) to communicate securely with MongoDB using Kerberos authentication.



Note You must log on as root, or use the `sudo` command to complete the installation.

Install MongoDB

Please got to http://docs.mongodb.org/manual/tutorial/install-mongodb-enterprise-on-red-hat/ for detailed installation instructions. The instructions below assume you are installing the latest version of MongoDB, Enterprise version 3.2.

1 Create the following file so that you can install MongoDB using `yum`.

```
/etc/yum.repos.d/mongodb-enterprise.repo
```

Use the following repository file to install the latest, stable MogoDB release:

```
[mongodb-enterprise]
name=MongoDB Repository
baseurl=https://repo.mongodb.org/yum/redhat/$releasever/mongodb-enterprise/stable
gpgcheck=0
enabled=1
```

2 Install MongoDB.

```
yum install -y mongodb-enterprise
```

3 Disable SELinux.

```
$vi /etc/selinux/config
SELINUX=disabled
```

4 Reboot your machine.

Install the Centrify agent on each node

Installing the Centrify agent on each MongoDB node will allow users in your Active Directory domain to access the node using their Active Directory credentials.

1 Download the appropriate `.tgz` file.

2 Unzip and extract the agent package.

For example:

```
gunzip centrify-suite-2015.1-rhel4-x86_64.tgz
tar -xvf centrify-suite-2015.1-rhel4-x86_64.tgz
```

You should now see two files; an `.rpm` file and an executable file.

3 Install the Centrify Agent.

For example: `rpm -Uvh centrifydc-5.2.3-rhel4-x86_64.rpm`

4 Install the Centrify Enterprise Agent.

```
./install.sh -ent-suite
```

5 Ensure that you can join to your Active Directory domain.

In /home/centrify/suite5.2.3, run the command line program `adcheck`.

```
[root@mrhel6-4 suite5.2.3]# ./adcheck-rhel4-x86_64 dd-demo.test
```

Note You should not see an error prompt if the machine is able to connect to an available domain controller.

6 Join to the Active Directory domain.

In the following example, the Active Directory domain is in the zone `Analytics`, the user is `tim`, and `-V` displays debugging information.

```
[root@mrhel6-4 suite5.2.3]# adjoin -z Analytics -u tim -V dd-demo.test
```

When you are prompted for a password, enter the domain controller password (in this example, tim's password).

Note If you would like to leave a zone, use the command-line program `adleave -f`.

Configure MongoDB and Kerberos

By integrating the Centrify agent with MongoDB, you can centrally create, secure, and distribute the service accounts and Kerberos key table (`keytab`) files that you require for distributed computing. The service accounts are stored securely in Active Directory with the domain controller acting as the Kerberos key distribution center (KDC).

Create the Kerberos keytab file

1 Create the Kerberos `keytab` file and service account. In the following example, the account surname, `mongod2`, is a unique service account on the domain controller.

```
adkeytab -V -n -u tim -K /etc/mongodb.keytab -U mongodb/mrhel6-4.dd-demo.test@DD-DEMO.TEST -P mongodb/mrhel6.4.dd-demo.test -c ou=NoSQL -S mongod2 mongod2
```

The parameters are defined in the following ways:

-n: create a new account.

-u: user name.

-K: the path of the keytab file.

-U: user principal name.

-P: service principal name.

-c: the parent container that houses any child NoSQL zones.

-S: account surname.

Get the Kerberos ticket (TGT/TGS) and authorize Kerberos

1 Get the TGT.

```
[root@mrhel6-4 suite5.2.3]# /usr/share/centrifydc/kerberos/bin/kinit -kt/etc/mongodb.keytab mongodb/mrhel6-4.dd-demo.test@DD-DEMO.TEST
```

2 Change ownership of the keytab file. This is the keytab file that mongodb will use to authenticate the user.

```
[root@mrhel6-4 suite5.2.3]# chown mongod:mongod /etc/mongodb.keytab
```

3 Get the TGS.

```
[root@mrhel6-4 suite5.2.3]# /usr/share/centrifydc/kerberos/bin/kinit -kt /etc/mongodb.keytab -S mongodb/mrhel6-4.dd-demo.test@DD-DEMO.TEST mongod2
[root@mrhel6-4 suite5.2.3]# /usr/share/centrifydc/kerberos/bin/klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: mongod2@DD-DEMO.TEST
```

In this example, the default principal name `mongod2@DD-DEMO.TEST` is the user that has been authenticated through Kerberos.

4 Restart the service.

Configure MongoDB

1 Log in to mongodb as a root user, and start the mongod daemon.

```
$sudo service mongod start
```

2 Create a basic collection.

In NoSQL environments, a **collection** an array that is similar to a table used in relational database structures.

In the following example, our collection is called “inventory”:

```
$ mongo
> use inventory
> db.inventory.insert({“id”:10,”name”:”Cartman”,”grades”:70})
```

While not necessary, completing this step will allow you to create a sample collection that you can assign to a user in this example.

3 Add a user who will be able to access the “inventory” collection, with a role that allows read/write permissions, in this case, the role is called “readWrite.”

```
> use $external
switched to db $external
> db createUser({user: “mongod2@DD-DEMO.TEST”,roles: [{ role: “readWrite”,
db: “inventory” }]})
> use $external
switched to db $external
> db.createUser({user: "mongod2@DD-DEMO.TEST",roles: [ { role: "readWrite",
db: "inventory" }]})


Successfully added user: {
        "user" : " mongod2@DD-DEMO.TEST ",
        "roles" : [
                {
                        "role" : "readWrite",
                        "db" : "inventory"
                }
        ]
}
```

4 Exit the service.

Configure MongoDB with Kerberos

1 In `/etc/sysconfig/mongod`, add the following line:

```
export KRB5_KTNAME=/etc/mongodb.keytab
```

2 In `/etc/mongod.conf`, comment out the `bind_ip` directive.

```
#bind_ip=127.0.0.1
```

For Kerberos single sign-on to work, you must bind over the network. If you do not bind over the network, the local host name will be used instead of the actual system host name, and authentication will fail.

3 In `/etc/sysconfig/mongod`, re-enable authentication using GSSAPI.

```
security:
authorization: enabled
setParameter:
authenticationMechanisms:GSSAPI
```

4 Restart the mongod service.

```
$ service mongod restart
```

If the service fails to restart, you can view the mongodb log by issuing the following command:

```
$ tail -n 100 /var/log/mongodb/mongod.log
```

If there is a port address conflict, kill the mongodb process before attempting to restart.

Test the MongoDB and Kerberos configuration

1 Get the TGT and TGS. See "Get the Kerberos ticket (TGT/TGS) and authorize Kerberos" on page 5.

After getting the TGS, ensure that you obtained the correct principal name. In our example, this is `mongod2@DD-DEMO.TEST`.

2 Start the MongoDB shell using the machine host name and domain name.

```
$ mongo mrhel6-4.dd-demo.test
MongoDB shell version: 3.2
connecting to: mrhel6-4.dd-demo.test/test
>
```

3 Verify that the authentication is successful.

```
>db.getSiblingDB("$external").auth({mechanism: "GSSAPI",user:'mongod2@DD-DEMO.TEST',})
1
```

If the Kerberos/GSSAPI authentication is successful, a "1" is returned, as in the example above.

Now that you have verified the success of the authentication procedure, you can access your collection database. For example, to access the collection "inventory" with a read-only role, you would enter the following:

```
> use inventory
switched to db inventory
> db.inventory.find()
```

This will return the contents of “inventory.”

```
{ “_id” : ObjectId(“55fb467ad75cefe1c6cce2b5”), “id” : 10, “name” :
“Cartman”,”grades” :70}
```

If the authentication was not successful, the following authentication error will be displayed:

```
Error: error: { “$err” : “not authorized for query on inventory.inventory”,
“code” : 13}
```

Log in as a different Active Directory user

If you log on to the machine as a user other than root, the TGT will be updated automatically, and the new user will be able to access MongoDB database collections.

For example:

1 Log on to the machine as tim, the new user.

```
ssh tim@172.27.9.154
password: tim's Active Directory password

[tim@mrhel6-4 ~]$ /usr/share/centrifydc/kerberos/bin/klist
Ticket cache: FILE:/tmp/krb5cc_1379927162
Default principal: tim@DD-DEMO.TEST

Valid starting     Expires            Service principal
09/17/15 18:14:15  09/18/15 04:14:15  krbtgt/DD-DEMO.TEST@DD-DEMO.TEST
renew until 09/24/15 18:14:15
09/17/15 18:15:39  09/18/15 04:14:15  mongodb/mrhel6-4.dd-demo.test@DD-
DEMO.TEST
renew until 09/24/15 18:14:15
```

The principal is changed to tim@DD-DEMO.TEST.

2 Log in to MongoDB as tim.

```
[tim@mrhel6-4 ~]$ mongo
MongoDB shell version: 3.2
connecting to: test
> db.getSiblingDB("$external").auth({mechanism: "GSSAPI",user:'tim@DD-
DEMO.TEST',})
1
> use inventory
switched to db inventory
> db.inventory.find()
{ "_id" : ObjectId("55fb467ad75cefe1c6cce2b5"), "id" : 10, "name" :
"Cartman", "grades" : 70 }
```

>

The new user is authenticated and able to access MongoDB.

For additional information on securing MongoDB nodes, go to https://docs.mongodb.org/manual/security/