Kerberos – A ‘Hello World’ example on CentOS/RHEL 7

Kerberos is a network authentication protocol that’s designed to allow machines to securely authenticate one another over a public network.

In a typical Kerberos setup, there is a single Kerberos server and lots of kerberos clients. The Kerberos server is often referred to as the KDC server, where KDC is short for Key Distribution Center. Before you can implement kerberos, there are 2 key requirements that needs to be met:

**Announcement**

I have released my new course on Udemy, [**Kubernetes By Example**](https://www.udemy.com/course/kubernetes-by-example/). Sign up now to get free lifetime access!

* all boxes must have fully qualified domain names
* time needs to be accurately synced across all boxes

The best way to understand how kerberos works is to go through a working example. You can follow this example with this [Kerberos vagrant project](https://github.com/Sher-Chowdhury/CentOS7-kerberos-demo). In this vagrant project we have the following scenario:

**+------------------------------------------------+**

**| |**

**| Kerberos Server |**

**| kdc.codingbee.net |**

**| (192.168.10.100) |**

**| |**

**+-------+--------------------------------+-------+**

**| |**

**| |**

**v v**

**+----------------------------+ +----------------------------+**

**| | | |**

**| krb client 1 | ssh | krb client 2 |**

**| krb-client1.codingbee.net | <-----------> | krb-client1.codingbee.net |**

**| (192.168.10.101) | | (192.168.10.102) |**

**| | | |**

**+----------------------------+ +----------------------------+**

The goal is that we want to ssh from **krb-client-1** to **krb-client-2** (and vice versa) using Kerberos authentication instead of entering a password, or use ssh keys. So the first thing to do is to create the above setup. With vagrant this is easy, all you do is a ‘vagrant up’. But if you were to do this manually then it would mean creating the Kerberos Server, and then the Kerberos Clients and then make sure that they can all ping each other.

Kerberos Server Setup

We can break down the installation process into 4 stages:

1. Setup valid fqdns
2. Setup Kerberos Server
3. Register trusted entities to the Kerberos Database
4. Configure firewalld to accept kerberos related traffic
5. Locally test our setup – configure ssh

Setup the Kerberos Server

To start with, we need to mimic valid fqdns by inserting the following entries into the **/etc/hosts** file:

**192.168.10.100 kdc.codingbee.net**

**192.168.10.101 krb-client1.codingbee.net**

**192.168.10.102 krb-client2.codingbee.net**

To start with we need to install the necessary rpms:

**$ yum install krb5-server krb5-workstation pam\_krb5**

**krb5-server** – this is the main kerberos server side software.

**krb5-workstation** – this is the keberos client side software. It’s optional to install it on the kerberos server, but is useful for Kerberos troubleshooting purposes.

**pam\_krb5** – this is the pam krb5 module. ssh does it’s authentication via pam, hence we need to make pam krb5 aware.

Next we need to update the following file:

**[root@kdc ~]# cat /var/kerberos/krb5kdc/kdc.conf**

**[kdcdefaults]**

**kdc\_ports = 88**

**kdc\_tcp\_ports = 88**

**[realms]**

**EXAMPLE.COM = {**

**#master\_key\_type = aes256-cts**

**acl\_file = /var/kerberos/krb5kdc/kadm5.acl**

**dict\_file = /usr/share/dict/words**

**admin\_keytab = /var/kerberos/krb5kdc/kadm5.keytab**

**supported\_enctypes = aes256-cts:normal aes128-cts:normal des3-hmac-sha1:normal arcfour-hmac:normal camellia256-cts:normal camellia128-cts:normal des-hmac-sha1:normal des-cbc-md5:normal des-cbc-crc:normal**

**}**

Basically we need to specify a realm:

**[root@kdc ~]# cat /var/kerberos/krb5kdc/kdc.conf**

**[kdcdefaults]**

**kdc\_ports = 88**

**kdc\_tcp\_ports = 88**

**[realms]**

**CODINGBEE.NET = {**

**master\_key\_type = aes256-cts**

**default\_principle\_flags = +preauth**

**acl\_file = /var/kerberos/krb5kdc/kadm5.acl**

**dict\_file = /usr/share/dict/words**

**admin\_keytab = /var/kerberos/krb5kdc/kadm5.keytab**

**supported\_enctypes = aes256-cts:normal aes128-cts:normal des3-hmac-sha1:normal arcfour-hmac:normal camellia256-cts:normal camellia128-cts:normal des-hmac-sha1:normal des-cbc-md5:normal des-cbc-crc:normal**

**}**

Note, you have to match the case, i.e. you need to write it in upper case.

Note, we also uncommented master\_key\_type and inserted the ‘default\_principle\_flags = +preauth’ line. That is optional, and is done so to add a bit more security hardening. But it also breaks backward compatibility to kerberos 4 and older.

Then we do the same thing to the following small file:

**[root@kdc ~]# cat /var/kerberos/krb5kdc/kadm5.acl**

**\*/admin@EXAMPLE.COM \***

Which means it ends up looking like this:

**[root@kdc ~]# cat /var/kerberos/krb5kdc/kadm5.acl**

**\*/admin@CODINGBEE.NET \***

Next we edit the following file:

**[root@kdc ~]# cat /etc/krb5.conf**

**# Configuration snippets may be placed in this directory as well**

**includedir /etc/krb5.conf.d/**

**[logging]**

**default = FILE:/var/log/krb5libs.log**

**kdc = FILE:/var/log/krb5kdc.log**

**admin\_server = FILE:/var/log/kadmind.log**

**[libdefaults]**

**dns\_lookup\_realm = false**

**ticket\_lifetime = 24h**

**renew\_lifetime = 7d**

**forwardable = true**

**rdns = false**

**# default\_realm = EXAMPLE.COM**

**default\_ccache\_name = KEYRING:persistent:%{uid}**

**[realms]**

**# EXAMPLE.COM = {**

**# kdc = kerberos.example.com**

**# admin\_server = kerberos.example.com**

**# }**

**[domain\_realm]**

**# .example.com = EXAMPLE.COM**

**# example.com = EXAMPLE.COM**

Here we need to activate this commented parts, and once again replace the generic information, i.e. it should look like:

**[root@kdc ~]#**

**[root@kdc ~]# cat /etc/krb5.conf**

**# Configuration snippets may be placed in this directory as well**

**includedir /etc/krb5.conf.d/**

**[logging]**

**default = FILE:/var/log/krb5libs.log**

**kdc = FILE:/var/log/krb5kdc.log**

**admin\_server = FILE:/var/log/kadmind.log**

**[libdefaults]**

**dns\_lookup\_realm = false**

**ticket\_lifetime = 24h**

**renew\_lifetime = 7d**

**forwardable = true**

**rdns = false**

**default\_realm = CODINGBEE.NET**

**default\_ccache\_name = KEYRING:persistent:%{uid}**

**[realms]**

**CODINGBEE.NET = {**

**kdc = kdc.codingbee.net**

**admin\_server = kdc.codingbee.net**

**}**

**[domain\_realm]**

**.codingbee.net = CODINGBEE.NET**

**codingbee.net = CODINGBEE.NET**

Now as part of our hello-world ssh test, we need to create a test user account, so in our example we’ll create the following:

**[root@kdc ~]# useradd krbtest**

This user actually needs to be created on the client boxes only. However we’re creating it on the kerberos server so that we can do some sanity testing locally.

Now we’re ready to create the Kerberos server’s internal database, to do this run:

**[root@kdc ~]# kdb5\_util create -s -r CODINGBEE.NET**

This command can take several minutes. You need to choose a password when prompted.

Once that’s done, you need to enable and start the following deamons:

**$ systemctl enable krb5kdc**

Configure firewalld to accept Kerberos related traffic

In terms of firewalld, you just need to add the kerberos and kadmin services:

**$ systemctl start firewalld.service**

**$ firewall-cmd --add-service=kerberos --permanent**

**$ firewall-cmd --add-service=kadmin --permanent**

**$ systemctl restart firewalld.service**

**$ systemctl enable firewalld.service**

Register trusted entities to the Kerberos Database

Next we need to start populating the Kerberos Database with entities that makes up the realm. We do this by running the **kadmin.local** which starts an interactive shell:

**[root@kdc ~]# kadmin.local**

**Authenticating as principal root/admin@CODINGBEE.NET with password.**

**kadmin.local:**

The first entity to add should be the root level kerberos entity, which is done by running the following in the kadmin.local session:

**kadmin.local: addprinc root/admin**

Then choose a password when prompted. Make a note of this password because it is need when (remotely) adding any additional entities in the future. Also notice how we used the ‘root/’ notation. You can think of this as meaning that we’re referring to a table name called ‘root’ inside the internal kdb database.

Now we need to register the krbtest user account to the kerberos database, so that it becomes trusted.

**kadmin.local: addprinc krbtest**

Then choose a password for this account when prompted. Notice this time we didn’t use ‘{table-name}/’ notation. I think that’s because when not specified it uses the ‘user/’ as the default.

Then add the kerberos server itself as a trusted object into the kerberos database:

**kadmin.local: addprinc -randkey host/kdc.codingbee.net**

Here, we’re adding a new entry to the ‘host’ table. Next get the keytab file:

**kadmin.local: ktadd host/kdc.codingbee.net**

The ktadd essentially creates the following file in the background:

**$ file /etc/krb5.keytab**

Kerberos works using symmetric encryption. This file stores the keys for encryption/decryption.

Quick tip: to find out what other commands you can run in this interactive shell, run:

**kadmin.local: ?**

Finally do a quick check of what’s in the database:

**kadmin.local: listprincs**

Finally quit out of the kadmin interactive session:

**kadmin.local: quit**

Locally test our setup – configure ssh

Next we’ll configure ssh so that it start’s relying on kerberos to do the authentication. This is optional, since in our scenario, we don’t need to ssh from any of the client boxes to the Kerberos server. But we are going to do this for early sanity checking purposes only.

First we’ll configure the ssh daemon side config file **/etc/ssh/sshd\_config**. All you need to do is ensure the following line is present:

**GSSAPIAuthentication yes**

Then restart the sshd deamon:

**[root@kdc ~]# systemctl restart sshd**

Now the sshd deamon will attempt to authenticate using kerberos. However we still need to configure the ssh client side to initiate an kerberos based authentication. To do this we need edit **/etc/ssh/ssh\_config**. Basically the following lines needs to be present:

**GSSAPIAuthentication yes**

**GSSAPIDelegateCredentials yes**

These lines are likely to already be present but commented out. If so then need to uncomment them and make sure they are both set to yes.

Finally we need to apply the following setting:

**[root@kdc ~]# authconfig --enablekrb5 --update**

Now we are ready to do a quick sanity check to test our setup.

First switch to the test user:

**[root@kdc ~]# su - krbtest**

Now if we try to ssh into itself, we get a password prompt:

**[krbtest@kdc ~]$ ssh kdc.codingbee.net**

**The authenticity of host 'kdc.codingbee.net (127.0.1.1)' can't be established.**

**ECDSA key fingerprint is SHA256:wkwlYMdcolcOB+zOiw7cMsAb+RKTn1Tim01WJGrZ21Y.**

**ECDSA key fingerprint is MD5:7c:f6:cc:1a:85:f0:fa:78:02:37:61:27:f3:fb:cc:9e.**

**Are you sure you want to continue connecting (yes/no)? yes**

**Warning: Permanently added 'kdc.codingbee.net' (ECDSA) to the list of known hosts.**

**krbtest@kdc.codingbee.net's password:**

So far we have failed to ssh in without the password. That’s because we don’t have an existing token to use. So we need to generate a new token. Before we generate this token, lets first check that there are no active tokens:

**[krbtest@kdc ~]$ klist**

**klist: Credentials cache keyring 'persistent:1001:1001' not found**

To get a token, we first need to initialise kerberos:

**[krbtest@kdc ~]$ kinit**

**Password for krbtest@CODINGBEE.NET: {enter password that you chose when registering krbtest in the krb database}**

If you entered the correct password, then you don’t get any reponse. So to check if this has worked run the klist command again, it should now look something like this:

**[krbtest@kdc ~]$ klist**

**Ticket cache: KEYRING:persistent:1001:1001**

**Default principal: krbtest@CODINGBEE.NET**

**Valid starting Expires Service principal**

**02/25/2018 19:25:26 02/26/2018 19:25:26 krbtgt/CODINGBEE.NET@CODINGBEE.NET**

Now if we try again, we it should connect without any password prompt:

**[krbtest@kdc ~]$ ssh kdc.codingbee.net**

**Last login: Sun Feb 25 19:21:24 2018**

**[krbtest@kdc ~]$**

Success!

Kerberos Client Setup

As part of our example scenario we have 2 client boxes, so we need to repeat the following steps on both boxes in order to get our scenario working.

To start with, we need to mimic valid fqdns by inserting the following entries into the **/etc/hosts** file:

**192.168.10.100 kdc.codingbee.net**

**192.168.10.101 krb-client1.codingbee.net**

**192.168.10.102 krb-client2.codingbee.net**

Next we need to install the necessary rpms:

**$ yum install krb5-workstation pam\_krb5**

Then edit the **/etc/krb5.conf**. It needs to look identical to the **/etc/krb5.conf** file that’s in the kerberos server. So simply copy and pasting will do the trick.

Now run the following command:

**[root@krb-client1 ~]# kadmin**

**Authenticating as principal root/admin@CODINGBEE.NET with password.**

**Password for root/admin@CODINGBEE.NET:**

Troubleshooting:

[root@krb-client1 etc]# kadmin

Authenticating as principal root/admin@CODINGBEE.NET with password.

Password for root/admin@CODINGBEE.NET:

kadmin: Communication failure with server while initializing kadmin interface

[root@krb-client1 etc]# ssh root@192.168.10.100 date && date

root@192.168.10.100's password:

Sun Aug 30 02:05:26 EDT 2020

Sun Aug 30 11:19:27 EDT 2020

Another reason, kadmind is not started at server. Start it:

systemctl start kadmin.service

systemctl enable kadmin.service

Here you’ll get prompted to enter the root kerberos password. This is the password you chose when setting up the kerberos server. After that you will get taken to the remote version of the kadmin.local session. Next let’s see what entities are registered on the central kerberos database:

**kadmin: list\_principals**

**K/M@CODINGBEE.NET**

**host/kdc.codingbee.net@CODINGBEE.NET**

**kadmin/admin@CODINGBEE.NET**

**kadmin/changepw@CODINGBEE.NET**

**kadmin/kdc.codingbee.net@CODINGBEE.NET**

**kiprop/kdc.codingbee.net@CODINGBEE.NET**

**krbtest@CODINGBEE.NET**

**krbtgt/CODINGBEE.NET@CODINGBEE.NET**

**root/admin@CODINGBEE.NET**

At the moment our box isn’t registered so let’s register it:

**kadmin: addprinc -randkey host/krb-client1.codingbee.net**

This command essentially generates a long random string that’s going to be stored on the kerberos database. This is symmetric encryption key, so we need to install a local copy of this key to. We create this local key by generating the keytab file:

**kadmin: ktadd host/krb-client1.codingbee.net**

Now if we list principles:

**kadmin: list\_principals**

**K/M@CODINGBEE.NET**

**host/kdc.codingbee.net@CODINGBEE.NET**

**host/krb-client1.codingbee.net@CODINGBEE.NET**

**kadmin/admin@CODINGBEE.NET**

**kadmin/changepw@CODINGBEE.NET**

**kadmin/kdc.codingbee.net@CODINGBEE.NET**

**kiprop/kdc.codingbee.net@CODINGBEE.NET**

**krbtest@CODINGBEE.NET**

**krbtgt/CODINGBEE.NET@CODINGBEE.NET**

**root/admin@CODINGBEE.NET**

**kadmin:**

Now we can see our recently added entry. Now we exit out:

**kadmin: quit**

Next we configure ssh:

First we’ll configure the ssh daemon side config file **/etc/ssh/sshd\_config**. All you need to do is ensure the following line is present:

**GSSAPIAuthentication yes**

Then restart the sshd deamon:

**[root@kdc ~]# systemctl restart sshd**

Now the sshd deamon will attempt to authenticate using kerberos. However we still need to configure the ssh client side to initiate an kerberos based authentication. To do this we need edit **/etc/ssh/ssh\_config**. Basically the following lines needs to be present:

**GSSAPIAuthentication yes**

**GSSAPIDelegateCredentials yes**

These lines are likely to already be present but commented out. If so then need to uncomment them and make sure they are both set to yes.

Finally we need to apply the following setting:

**[root@kdc ~]# authconfig --enablekrb5 --update**

Now you need to repeat the same steps for the krb-client2.codingbee.net box. After that you will be ready to

Test the Kerberos setup

Now we are ready to put our setup to the test. To do this test, log into one of the client boxes. In my case I’ll use. krb-client2.codingbee.net

We first need to create a test user account, so in our example we’ll create the following:

**[root@krb-client1 ~]# useradd krbtest**

First switch to the test user:

**[root@kdc ~]# su - krbtest**

**[root@kdc ~]# su - krbtest**

Now if we try to ssh into itself, we get a password prompt:

**[krbtest@krb-client1 ~]$ ssh krb-client2.codingbee.net**

**The authenticity of host 'krb-client2.codingbee.net (192.168.10.102)' can't be established.**

**ECDSA key fingerprint is SHA256:wkwlYMdcolcOB+zOiw7cMsAb+RKTn1Tim01WJGrZ21Y.**

**ECDSA key fingerprint is MD5:7c:f6:cc:1a:85:f0:fa:78:02:37:61:27:f3:fb:cc:9e.**

**Are you sure you want to continue connecting (yes/no)? yes**

**Warning: Permanently added 'krb-client2.codingbee.net,192.168.10.102' (ECDSA) to the list of known hosts.**

**krbtest@krb-client2.codingbee.net's password:**

So far we have failed to ssh in without the password. That’s because we don’t have an existing token to use, which you can check like this:

**[krbtest@krb-client1 ~]$ klist**

**klist: Credentials cache keyring 'persistent:1001:1001' not found**

To get a token, we first need to initialise kerberos:

**[krbtest@kdc ~]$ kinit**

**Password for krbtest@CODINGBEE.NET: {enter password that you chose when registering krbtest in the krb database}**

If you entered the correct password, then you don’t get any reponse. So to check if this has worked run the klist command again, it should now look something like this:

**[krbtest@kdc ~]$ klist**

**Ticket cache: KEYRING:persistent:1001:1001**

**Default principal: krbtest@CODINGBEE.NET**

**Valid starting Expires Service principal**

**02/25/2018 19:25:26 02/26/2018 19:25:26 krbtgt/CODINGBEE.NET@CODINGBEE.NET**

This time we have a token, but not it expires after exactly 24 hours.

Now if we try again, we it should connect without any password prompt:

**[krbtest@krb-client1 root]$ ssh krb-client2.codingbee.net**

**Last login: Sun Feb 25 17:17:20 2018 from 192.168.10.101**

**[krbtest@krb-client2 ~]$**

Success!

Note: if skipped doing ‘kinit’, and instead ran the ssh command directly, then you will get a password prompt. If you then entered the password that you set during the principal adding of the krbtest user, then it will login in successfully and and create the 24 hour token for you.

Recommended reading

http://www.roguelynn.com/words/explain-like-im-5-kerberos/

[post-content post\_name=rhsca-quiz]

In our quiz we’ll call our kerberos server as **kdc.codingbee.net** and our kerberos client as krb-**client1.codingbee.net**.

What are the 2 pre-requisites before setting up Kerberos in an infrastructure?

What are the main steps of setting up a Kerberos server?

What is command to install the necessary rpms on the Kerberos Server?

What files needs to be edited on the Kerberos Server?

What parts do you need to change in /var/kerberos/krb5kdc/kdc.conf?

What part do you need to change in /var/kerberos/krb5kdc/kadm5.acl?

What part do you need to change in /etc/krb5.conf?

What is the command to create test user account 'krbtest'?

What's the command to create the internal kdc database?

What is the command to enable and start the kerberos daemons?

What is the command to open up the firewall rules?

What is the command to enter the kerberos interactive terminal?

What tasks needs to performed in the kerberos interactive session?

What is the command to set a password for the main admin account?

What is the command to register system test user account?

What is the command to register host machine?

What is the command to pull down copies of symmetric key files?

What is the command to view help info?

What is the command to exit out of the kdc terminal?

Where is the symmetric key file stored?

Which ssh config file needs to be changed?

What setting needs to be present in the /etc/ssh/sshd\_config file?

What setting needs to be present in the /etc/ssh/ssh\_config file?

What command needs to be run in order to load in these ssh config changes?

what is the command to enable krb5 based authentication?

What are the steps you need to take to locally test this?

What are the steps you need to do to set up a krb client?

what is the command to install the client rpms needs to be installed on the client?

What is the command to enter the kerberos interactive session?

what tasks do you need to run in the kerberos session?

what is the command to list resources?

What is the command to register the client?

What is the command to pull down the symmetric keys?

What ssh related configurations needs to be made?

what is the command to load and activate these changes:

# Kerberos - Principal (Account)

[> Software Security](https://datacadamia.com/security/security) [> (Authentication|Access control|Identification) - AuthN](https://datacadamia.com/security/auth/auth) [> Kerberos](https://datacadamia.com/security/auth/kerberos/kerberos)

### Table of Contents

* [1 - About](https://datacadamia.com/security/auth/kerberos/principal#about)
* [2 - Articles Related](https://datacadamia.com/security/auth/kerberos/principal#articles_related)
* [3 - Syntax](https://datacadamia.com/security/auth/kerberos/principal#syntax)
* [4 - Example](https://datacadamia.com/security/auth/kerberos/principal#example)
* [5 - Type](https://datacadamia.com/security/auth/kerberos/principal#type)
* [6 - Documentation / Reference](https://datacadamia.com/security/auth/kerberos/principal#documentationreference)

## 1 - About

A Kerberos [principal](https://datacadamia.com/security/auth/principal) is a unique identity to which Kerberos can assign [tickets](https://datacadamia.com/security/auth/kerberos/ticket).

Kerberos defines two different types of accounts (or Principals):

* [User Principal Name (UPN)](https://datacadamia.com/security/auth/kerberos/upn),
* and [Service Principal Name (SPN)](https://datacadamia.com/security/auth/kerberos/spn).

In a Microsoft environment, they would be named:

* Active Directory users
* and computers.

## 2 - Articles Related

* [Kerberos - (Ticket|Credentials)](https://datacadamia.com/security/auth/kerberos/ticket)
* [Kerberos - ticket-granting ticket (TGT)](https://datacadamia.com/security/auth/kerberos/tgt)
* [Kerberos - klist](https://datacadamia.com/security/auth/kerberos/klist)
* [Kerberos - Service principal name](https://datacadamia.com/security/auth/kerberos/spn)
* [Kerberos - KeyTab (Key Table)](https://datacadamia.com/security/auth/kerberos/keytab)
* [Kerberos - kinit](https://datacadamia.com/security/auth/kerberos/kinit)
* [Kerberos - Ticket Granting Server (TGS)](https://datacadamia.com/security/auth/kerberos/tgs)
* [Kerberos - Client](https://datacadamia.com/security/auth/kerberos/client)
* [Kerberos - User Principal Name (UPN)](https://datacadamia.com/security/auth/kerberos/upn)
* [Security - Principal](https://datacadamia.com/security/auth/principal)
* [More ...](https://datacadamia.com/security/auth/kerberos/principal?do=backlink)

## 3 - Syntax

Traditionally, a principal is divided into three (components|parts):

primary[/instance][@REALM]

where:

* The primary is for :
  + a user, the same as your username.
  + a [service](https://datacadamia.com/security/auth/kerberos/service), the name of the service
* The instance is an optional string that qualifies the primary. The instance is for:
  + a user, usually null (but a user might also have an additional principal, with an instance called admin, which he/she uses to administrate a database. The principal jennifer@ATHENA.MIT.EDU is completely separate from the principal jennifer/admin@ATHENA.MIT.EDU, with a separate password, and separate permissions.
  + a host, the instance is the fully qualified hostname, e.g., daffodil.mit.edu
* The [realm](https://datacadamia.com/security/auth/kerberos/realm) is the Kerberos [realm](https://datacadamia.com/security/auth/kerberos/realm). In most cases, your Kerberos realm is your domain name, in **upper-case** letters. (default to the default realm for the context in which it is being used ie the machine nico.gerardnico.com would be in the realm GERARDNICO.COM. )

Principals can have an arbitrary number of (components|parts). Each (components|parts) is separated by a component separator, generally `/'.

## 4 - Example

* Concatenation of HTTP/, the user logon name, and the realm (it must be in uppercase)

HTTP/server3.wasteched30.torolab.ibm.com@WASTECHED30.TOROLAB.IBM.COM

* host ??

HOST/server3.wasteched30.torolab.ibm.com@WASTECHED30.TOROLAB.IBM.COM

## 5 - Type

* KRB\_NT\_PRINCIPAL: a user principal name type.
* KRB\_NT\_SRV\_HST: a [service](https://datacadamia.com/security/auth/kerberos/service) with host name as instance ([telnet, rcommands](https://datacadamia.com/security/auth/kerberos/application)) name type.
* KRB\_NT\_SRV\_INST: a [service](https://datacadamia.com/security/auth/kerberos/service) and other unique instance ([krbtgt](https://datacadamia.com/security/auth/kerberos/tgt" \l "krbtgt" \o "security:auth:kerberos:tgt)) name type.
* KRB\_NT\_SRV\_XHST: a [service](https://datacadamia.com/security/auth/kerberos/service) with host as remaining components name type.
* KRB\_NT\_UID: a unique ID name type.
* KRB\_NT\_UNKNOWN: unknown name type.

See [KerberosPrincipal](https://docs.oracle.com/javase/9/docs/api/javax/security/auth/kerberos/KerberosPrincipal.html" \o "http://docs.oracle.com/javase/9/docs/api/javax/security/auth/kerberos/KerberosPrincipal.html)

## 6 - Documentation / Reference

* [What is a Kerberos Principal?](http://web.mit.edu/kerberos/krb5-1.5/krb5-1.5.4/doc/krb5-user/What-is-a-Kerberos-Principal_003f.html)

Another guide:

# Installing Kerberos on Redhat 7

This installation is going to require 2 servers one acts as kerberos KDC server and the other machine is going to be client. Lets assume the FQDN's are (here cw.com is the domain name, make a note of the domain name here):

* Kerberos KDC Server: kdc.cw.com
* Kerberos Client: kclient.cw.com

Important: Make sure that both systems have their hostnames properly set and both systems have the hostnames and IP addresses of both systems in /etc/hosts. Your server and client must be able to know the IP and hostname of the other system as well as themselves.

Pre-Requisites:

Setup and install NTP

yum -y install ntp

ntpdate 0.rhel.pool.ntp.org

systemctl start ntpd.service

systemctl enable ntpd.service

RHEL 7 comes with systemd as the default service manager. Here is a handy guide for mapping service and chkconfig command [here](http://fedoraproject.org/wiki/SysVinit_to_Systemd_Cheatsheet)

Packages required:

* KDC server package: krb5-server
* Admin package: krb5-libs
* Client package: krb5-workstation

Configuration Files:

* /var/kerberos/krb5kdc/kdc.conf
* /var/kerberos/krb5kdc/kadm5.acl
* /etc/krb5.conf

Important Paths:

* KDC path: /var/kerberos/krb5kdc/

## Installing & Configuring KDC Server:

yum -y install krb5-server krb5-libs

Primary configuration file is 'krb5.conf':

* Ensure the default realm is set your domain name in capital case

Sample '/etc/krb5.conf'

[libdefaults]

default\_realm = CW.COM

dns\_lookup\_realm = false

dns\_lookup\_kdc = false

ticket\_lifetime = 24h

forwardable = true

udp\_preference\_limit = 1000000

default\_tkt\_enctypes = des-cbc-md5 des-cbc-crc des3-cbc-sha1

default\_tgs\_enctypes = des-cbc-md5 des-cbc-crc des3-cbc-sha1

permitted\_enctypes = des-cbc-md5 des-cbc-crc des3-cbc-sha1

[realms]

CW.COM = {

kdc = kdc.cw.com:88

admin\_server = kdc.cw.com:749

default\_domain = cw.com

}

[domain\_realm]

.cw.com = CW.COM

cw.com = CW.COM

[logging]

kdc = FILE:/var/log/krb5kdc.log

admin\_server = FILE:/var/log/kadmin.log

default = FILE:/var/log/krb5lib.log

Adjust /var/kerberos/krb5kdc/kdc.conf on the KDC:

default\_realm = CW.COM

[kdcdefaults]

v4\_mode = nopreauth

kdc\_ports = 0

[realms]

CW.COM = {

kdc\_ports = 88

admin\_keytab = /etc/kadm5.keytab

database\_name = /var/kerberos/krb5kdc/principal

acl\_file = /var/kerberos/krb5kdc/kadm5.acl

key\_stash\_file = /var/kerberos/krb5kdc/stash

max\_life = 10h 0m 0s

max\_renewable\_life = 7d 0h 0m 0s

master\_key\_type = des3-hmac-sha1

supported\_enctypes = arcfour-hmac:normal des3-hmac-sha1:normal des-cbc-crc:normal des:normal des:v4 des:norealm des:onlyrealm des:afs3

default\_principal\_flags = +preauth

}

Adjust /var/kerberos/krb5kdc/kadm5.acl on KDC:

\*/admin@CW.COM \*

## Creating KDC database to hold our sensitive Kerberos data

Create the database and set a good password which you can remember. This command also stashes your password on the KDC so you don’t have to enter it each time you start the KDC:

kdb5\_util create -r CW.COM -s

This command may take a while to complete based on the CPU power

Now on the KDC create a admin principal and also a test user (user1):

[root@kdc ~]# kadmin.local

kadmin.local: addprinc root/admin

kadmin.local: addprinc user1

kadmin.local: ktadd -k /var/kerberos/krb5kdc/kadm5.keytab kadmin/admin

kadmin.local: ktadd -k /var/kerberos/krb5kdc/kadm5.keytab kadmin/changepw

kadmin.local: exit

Let’s start the Kerberos KDC and kadmin daemons:

systemctl start krb5kdc.service

systemctl start kadmin.service

systemctl enable krb5kdc.service

systemctl enable kadmin.service

Now, let’s create a principal for our KDC server and stick it in it’s keytab:

[root@kdc ~]# kadmin.local

kadmin.local: addprinc -randkey host/kdc.cw.com

kadmin.local: ktadd host/kdc.cw.com

## Setup kerberos client

yum -y install krb5-workstation

Transfer your /etc/krb5.conf (which got created from above command) from the KDC server to the client. Hop onto the client server, install the Kerberos client package and add some host principals:

[root@client ~]# yum install krb5-workstation

[root@client ~]# kadmin -p root/admin

kadmin: addpinc --randkey host/client.example.com

kadmin: ktadd host/kdc.example.com

[Raw](https://gist.github.com/ashrithr/4767927948eca70845db/raw/75cc44ce085c732c4eba09f61739f5ad2ff00b11/kerberos_ssh.md)

[**kerberos\_ssh.md**](https://gist.github.com/ashrithr/4767927948eca70845db#file-kerberos_ssh-md)

# Setting up SSH to use Kerberos Authentication

**Pre-Req**: Make sure you can issue a kinit -k host/fqdn@REALM and get back a kerberos ticket without having to specify a password.

## Step1: Configuring SSH Server

Configure /etc/ssh/sshd\_config file to include the following lines:

KerberosAuthentication yes

GSSAPIAuthentication yes

GSSAPICleanupCredentials yes

UsePAM no

Now, restart the ssh daemon.

## Step2: Configure the SSH Client

Configure /etc/ssh\_config to include following lines:

Host \*.domain.com

GSSAPIAuthentication yes

GSSAPIDelegateCredentials yes

Note: make sure you change the domain to match your environment.