# **Puppet creating and managing user accounts with SSH access:-**

Security and access control

To have good security and access control practices, we need to use the following policies:

1. Everyone who needs access to a machine has his/her own user account with an SSH key (not a password).
2. Access to special-purpose accounts, such as those used to deploy and run applications, or a database, is controlled by authorizing specific SSH keys, rather than using passwords.
3. Accounts that need certain, specific superuser privileges can get them via the sudo mechanism.
4. The root account is not accessible via the network (but there is secure, out-of-band access to the system console).
5. Third parties, such as contractors and support staff, get temporary access with limited privileges, which can be revoked once a job is finished.

Setting up policies listed above, while highly desirable from a security point of view, is time-consuming to do by hand and difficult to maintain. If a new user arrives, someone has to add and configure his account on every server. If a user leaves, the accounts have to be removed or locked everywhere.

Puppet can make it quicker and easier to manage user accounts securely across a large network. We can add or remove individual and shared accounts, control their access via SSH, manage their privileges via sudo, and have the changes immediately applied to every machine under Puppet's control, all without logging into a single server.

Puppet provides a couple of ways to help us manage users. The user resource type controls user accounts, and the ssh\_authorized\_key resource type controls SSH access to accounts. We can use Puppet to control user privileges by managing the **sudoers** file.

Creating a user

Edit our **manifests/site.pp** file as follows:

node 'puppet-agent' {

include user

}

Also, we need to edit our **modules/user/manifests/init.pp** file:

class user {

user { 'testuser':

ensure => present,

comment => ' user',

home => '/home/testuser,

managehome => true

}

}

Run puppet:

ubuntu@puppet-agent:~$ sudo puppet agent --test

Info: Retrieving plugin

Info: Caching catalog for puppet-agent.ec2.internal

Info: Applying configuration version '1419727849'

Notice: /Stage[main]/User/User[testuser]/ensure: created

Notice: Finished catalog run in 0.22 seconds

ubuntu@puppet-agent:~$ cd /home

ubuntu@puppet-agent:/home$ ls

testuser ubuntu

Puppet's user resource type creates a user or modifies it if the user already exists. The following line declares a user whose login name is 'testuser':

user { 'testuser':

The user should be present:

ensure => present,

We can also specify here some information about the user:

comment => ' user',

The comment attribute sets the user's full name.

home => '/home/testuser',

The home attribute sets the path to the user's home directory. Puppet will not create this directory for us unless we also set the managehome attribute:

managehome => true,

So the manifest says that a user named 'testuser' should exist, whose full name is ' user', and that the home directory should be '/home/testuser', and that that directory should exist. Note that we have not specified a password for the user, and as a result 'testuser' will not yet be able to log in. Although Puppet can set passwords for users, SSH authentication is recommended.

Removing a user

To remove a user from the system altogether, use the **ensure => absent** attribute:

user { 'testuser':

ensure => absent,

}

When we run Puppet, the 'testuser' account will be removed though testuser's home directory and any files he owned will remain.

Access control

Now that we've created the user's account, we now need to provide a secure way for a user to log in. We can do this using the SSH protocol.

In this section, the Puppet master will put a public key of my labtop into **authorized\_keys** of Puppet agent so that I can login to the agent node from my laptop computer via ssh.

Puppet can manage SSH public keys and authorize them for user accounts, using the **ssh\_authorized\_key** resource type.

We'll need our own SSH public key for this. If we already have one on our own computer, display the contents:

If we don't have an SSH key, we can generate one for this exercise:

k@laptop:~/.ssh$ ssh-keygen

Generating public/private rsa key pair.

Enter file in which to save the key (/home/testuser/.ssh/id\_rsa):

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

Your identification has been saved in /home/testuser/.ssh/id\_rsa.

Your public key has been saved in /home/testuser/.ssh/id\_rsa.pub.

The key fingerprint is:

bf:af:d1:85:af:a6:5f:f9:19:ad:cf:94:df:7d:21:d1 testuser@laptop

The key's randomart image is:

+--[ RSA 2048]----+

| |

| |

| . |

| o E |

| S . o |

| . . + oo|

| o . =o+|

| o.o =B|

| +\*+ .oO|

+-----------------+

Now display the **id\_rsa.pub** file to see the public key:

k@laptop:~$ cat ~/.ssh/id\_rsa.pub

ssh-rsa AAAAB3...jjQfJ7 testuser@laptop

The key itself is the long string of numbers and letters, without the **ssh-rsa** part at the beginning, or the **testuser@laptop** part at the end. It's this string we'll put into the Puppet manifest in the next step.

Edit our **modules/user/manifests/init.pp** file as follows using our own key string as the value for key:

class user {

user { 'testuser':

ensure => present,

comment => ' user',

home => '/home/testuser',

managehome => true

}

ssh\_authorized\_key { 'testuser\_ssh':

user => 'testuser',

type => 'rsa',

key => 'AAAA...GjjQfJ7',

}

}

Run Puppet:

root@puppet-agent:~# puppet agent --test

Info: Retrieving plugin

Info: Caching catalog for puppet-agent.ec2.internal

Info: Applying configuration version '1419754797'

Notice: /Stage[main]/User/Ssh\_authorized\_key[testuser\_ssh]/key: key changed 'AAA...iQ8JkZV1F' to 'AAAA...jjQfJ7'

Notice: Finished catalog run in 0.03 seconds

Actually, we got the output after changing the pub key.

At this point, Puppet has added the key to the file **/home/testuser/.ssh/authorized\_keys** on Puppet agent node. When we try to log in to testuser's account via SSH, the system will look in this file to see if our private key matches any of the public keys listed there. Assuming it does, we'll be able to log in to Puppet agent node of AWS via ssh:

k@laptop:~/.ssh$ ssh testuser@54.88.104.246

Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.13.0-36-generic x86\_64)

\* Documentation: https://help.ubuntu.com/

System information as of Sat Dec 27 19:11:48 UTC 2014

System load: 0.0 Processes: 104

Usage of /: 11.8% of 7.74GB Users logged in: 0

Memory usage: 11% IP address for eth0: 172.31.43.38

Swap usage: 0%

...

$ uname -a

Linux puppet-agent 3.13.0-36-generic #63-Ubuntu SMP Wed Sep 3 21:30:07 UTC 2014 x86\_64 x86\_64 x86\_64 GNU/Linux

$ cd /home

$ ls

tesuser ubuntu