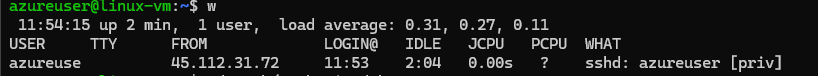
DevOps Practice notes and lab-

1. Linux command “w” to check how many users are logged in to the server.

Create a new Linux server and login as administrator.

Run w to check how many users are logged in to the server.

>> w



Create a new user and login by that user.

>> sudo adduser new\_username

A computer screen with white text

Description automatically generated

Login as new created user “new\_username”. Here I mentioned the username as “new\_username” we can define any user name such as “Ram”.

>> ssh -i key\_location username@serverip



We failed to build the connection. Because the user contains “.ssh” directory and from there it pulls the key. To authenticate.

It’s already will be available for admin to check

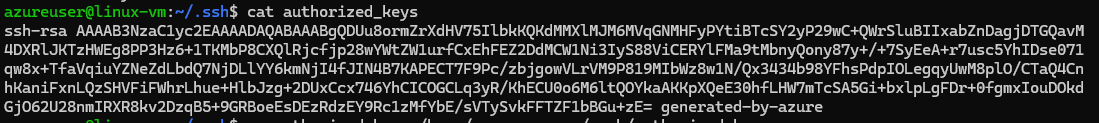
>> .ssh

>> ls

>> cat authorized\_keys

A black background with different colored letters

Description automatically generated



Copy this key to new created user directory, as we can’t login to new users directory, we can do it by login as root user or by using sudo cmd.

Login as root

>> sudo su

Then create .ssh directory and authorized\_keys file and pest the key

>> cd home

>> cd new\_username/

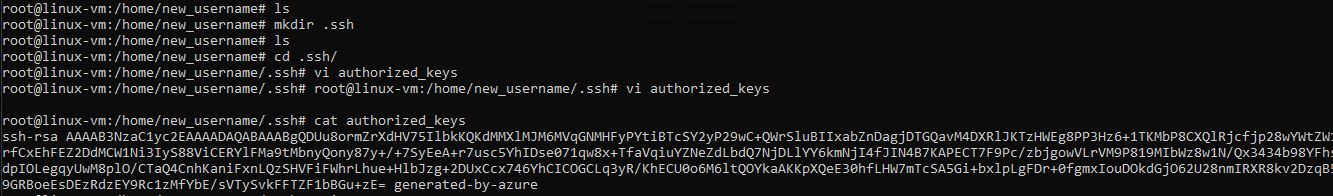
>> cd /.ssh/

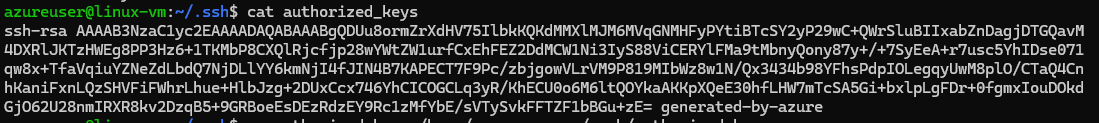
>> mkdir .ssh

>> cd .ssh

>> vi authorized\_keys

Write the content for key





Now try to login, login works.

A computer screen with white text

Description automatically generated

Now run w and check

A screen shot of a black screen

Description automatically generated

It shows all connected sessions, their network ip’s etc..

Question-

I did not understand the use of this. i have shared the key to new created user. user downloaded and stored the key on his local server. and using -i to provide the file path to authenticate and login. than what is the use of changing again in user directory .ssh folder?

Ans-

Thank you for the clarification! If the new user already has the private key stored on their local machine and is using ssh -i /path/to/private\_key user@hostname to log in, then the setup you're describing (creating .ssh and adding the public key to authorized\_keys) isn't strictly necessary for the login process itself. The .ssh folder and authorized\_keys file are typically used on the **server side**, not the client side.

Let me break this down and explain when and why you would need to create .ssh and set up authorized\_keys:

**The Flow of SSH Authentication:**

1. **User Downloads and Stores the Private Key**:
   * The **new user** (client) has the **private key** stored on their local machine (for example, ~/.ssh/id\_rsa on their local machine). This is correct and the way SSH key-based authentication works.
   * The user uses the command:

bash

Copy code

ssh -i /path/to/private\_key new\_user@hostname\_or\_ip

to authenticate themselves.

1. **Server-Side Setup (Authorized Keys)**:
   * On the **server** side (the remote machine you're logging into), the **public key** associated with that private key should be placed in the ~/.ssh/authorized\_keys file of the user account you're trying to log into (in this case, new\_user).

This is where the .ssh directory and authorized\_keys file come into play.

**Why You Would Need to Set Up .ssh/authorized\_keys on the Server:**

* **When you provide a user with an SSH key pair (private and public keys),** the **public key** needs to be stored on the **server** in the authorized\_keys file under the **correct user account** (e.g., new\_user on the server).
* The **private key** is kept on the user's local machine, and they use -i with ssh to authenticate. The **server** checks if the private key matches the stored public key in authorized\_keys to authenticate the user.

**So, if the new user already has the private key:**

* The **only thing that needs to be done** on the server (remote machine) is ensuring that the **public key** (that corresponds to the private key the user has) is properly stored in the ~/.ssh/authorized\_keys file of the user account they are trying to log into (in this case, new\_user).

Create a new user and test-

Created a new user ram now try to login by Ram

**A screen shot of a computer

Description automatically generated**

Same issue as previous client. Check for .ssh on ram

A screen shot of a computer program

Description automatically generated

Now again we need to add the key public on the server for ram. Like this we need to do for all users but it’s not good we need to place the key in some common directory that all users can access. So we should not do separately for all users.

For password authentication we don’t need all those.

Here the thing is while we are creating the linux server we are defining a username and generating the key so by default the public key is placed in that user’s directory so the user can login by using private key.

But while we are just adding other users we are not generating the key so manually we are placing the public key for that user so that the user can login by using his private key.

Hence, we can enable key auth by 2 different ways. 1 is by using the script. But, in script as well it make changes in each users directory and add public key.

If we want to do it in a different way that is creating a shared directory in shared root level and storing the public key there ad also configuring the changes that each user key will verify authentication from that directory. So in this way the users those have valid key can login.

Step 1-

**Create a centralized authorized\_keys directory**: Create a directory, e.g., /etc/ssh/authorized\_keys.d/, where you can store the shared public keys.

>> sudo mkdir -p /etc/ssh/authorized\_keys.d

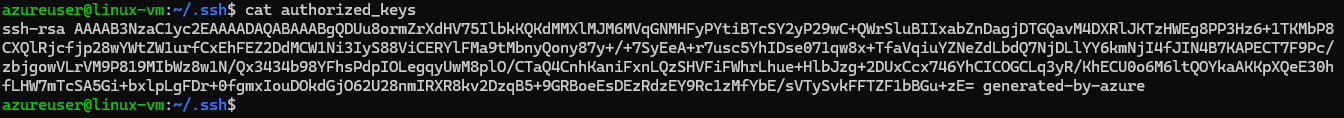
A screen shot of a computer

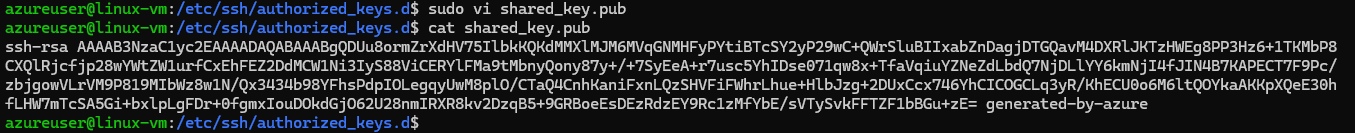
Description automatically generated

Step 2- **Add the public key to the centralized location**: Save the public key to a file in the new directory:

>> echo "ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEArbitraryPublicKey..." | sudo tee /etc/ssh/authorized\_keys.d/shared\_key.pub

I did it in a different way, I goes for vi cmd copy my existing public key and placed.





Step 3-

**Configure SSH to use this directory**: You’ll need to modify the SSH configuration to include this directory. Open /etc/ssh/sshd\_config and ensure the following line is added:

Note we need to add 2 lines here

>> PubkeyAuthentication yes

>> AuthorizedKeysFile /etc/ssh/authorized\_keys.d/\*.pub

Note- the file we need to change is sshd\_config. Not ssh\_config.

This tells SSH to look for public keys in the /etc/ssh/authorized\_keys.d/ directory for all users, not just in their individual ~/.ssh/authorized\_keys files.

A computer screen with white text

Description automatically generated

A screen shot of a computer

Description automatically generated

Note- in this I am adding the authorized key file location so by default it will see the public key here to check authentication.  
  
If I want only the users added in admin group or any other group only can have auth by this public key. We can add alternative cmd in sshd\_config.

>>

Match Group admin

AuthorizedKeysFile /etc/ssh/authorized\_keys.d/\*.pub

Save and exit.

A screen shot of a computer

Description automatically generated

Step 4-

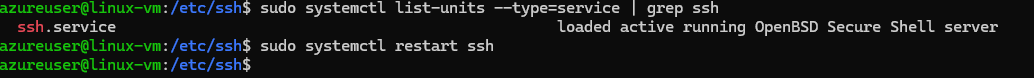
After making changes to /etc/ssh/sshd\_config, you must restart the SSH service to apply the changes:

>> sudo systemctl restart sshd



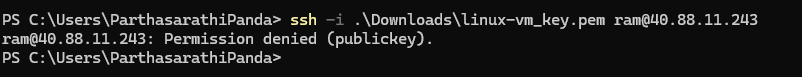
It did not worked for me because sshd service is not configured yet. So I can check the similar service running.

>> sudo systemctl list-units --type=service | grep ssh



Step 4-

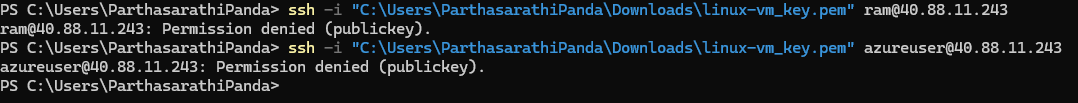
Try login failed



Reboot the server.

>> sudo reboot

But here now I am not able to login by any user. The issue is due to I missed to configure permissions on the created directory and on the public key file.



Now I am not able to login to the server how can I set permissions?

The only way I can do it by using serial console.

To login to serial console I can’t use key here. But while I had created the users like “ram” I mentioned password for them so by that userame and password I can login.

A screen shot of a computer

Description automatically generated

**Set the appropriate permissions for the authorized\_keys.d directory**:

>> sudo chmod 700 /etc/ssh/authorized\_keys.d

>> sudo chmod 644 /etc/ssh/authorized\_keys.d/\*.pub

A black background with white text

Description automatically generated

User can not execute the cmd if the user is not in sudo group.

Login as root

>> sudo su but the user is not able to use cmd sudo.

Now anyway we need to login to the server as a root user.

I did in by resetting password from azure portal. Go to reset option and even we have key auth change to password change mode and change password.

A screenshot of a computer

Description automatically generated

Now we can logged in as admin user

A screenshot of a computer screen

Description automatically generated

Now change the file permissions.

After file permission change I am able to login. But as I changed it to password type also I am able to login by password only.

I rechange it to key type by placing key.

A screenshot of a computer

Description automatically generated

And on vm I need to disable password authentication type to no.

>> sudo vi /etc/ssh/sshd\_config

Change password auth to no.

Ex-

PasswordAuthentication no

ChallengeResponseAuthentication no

>> cd /etc/ssh/sshd\_config.d

Go to all subfiles and change same

Now password auth is disabled but still not able to login by key.

It shows permission issue.

This was due to key corruption. I tried to generate the key manually and upload public key to azure and have the private key with mey. But that did not worked.

I regenerated key from azure and it worked.

Now I amble able to login by using key and also others users are able to login.

Also I reset the configuration.

A screenshot of a computer

Description automatically generated

It was done.

Note –

To generate key locally and upload public key to azure and have private key with us rather then generating from azure.

We can use below cmd-

>> ssh-keygen -t rsa -b 2048 -f location/key-name

Ex-

ssh-keygen -t rsa -b 2048 -f "C:\Users\ParthasarathiPanda\Downloads\new\_linux-vm\_key.pem"

To know in details while logging in where the actual validation failed use -v in cmd

Ex-

ssh -v -i "C:\Users\ParthasarathiPanda\Downloads\linux-vm\_key.pem" [azureuser@40.88.11.243](mailto:azureuser@40.88.11.243)

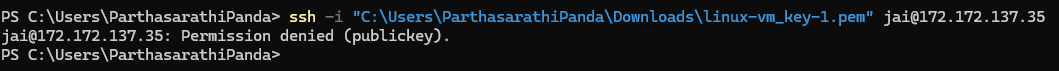
I feel it’s better idea to generate different key for different user while azure is providing that facility. That could be better way to handle.

Lets create a new user “jai”

>> sudo adduser jai

A screenshot of a computer program

Description automatically generated



Create key for the user

A screenshot of a computer

Description automatically generated

Download the key.

A screenshot of a computer

Description automatically generated

Tried login and successfully log in.

A screenshot of a computer

Description automatically generated