How to setup ssh in windows for Deployment(s)

Table of Contents

Introduction and Steps

Introduction

Setting up SSH based security to access your server is a much more effective way than the use of a manual root password. Cracking the security system of a node depending on SSH keys is nearly impossible since it secures your node in a more sophisticated way by the use of encoded keys.

About SSH keys

Use of SSH keys favors a very boosted form of security against the brute forces attacking a virtual private server. Use of passwords, independent of their complex nature is always vulnerable towards security threats. SSH keys provide a whole new level of security which is safe and impregnable. SSH keys are basically generated in pairs (i.e public key and a private key). One can associate the public key with any server, and only the client in possession of the private key can have access to the decrypted data.

Step 1: Pre-requisite

To perform the below steps in a Windows server, you need to have `OpenSSH Server` installed.

Fill the form with the necessary details and submit. Once the request is completed, verify it in Add/Remove Programs.

Step 2: Generate the SSH keys

To establish a secure connection between your local machine and the remote server, you need to **generate** an SSH key pair.

On your local machine, execute the following command in the terminal:

command
`ssh-keygen -t rsa -b 4096 -C "service_account_name"`

Step 3: Removing Carriage Returns

In case of the above command is executed in the Windows Server, we have to **remove carriage return** characters. This can be achieved using the below PowerShell commands.

PowerShell

```
Get-Content -Raw C:\path\to\SSH_KEY | % {$_ -replace "`r", ""} | Set-Content C:\path\to\SSH_KEY_Cleaned
Move-Item C:\path\to\SSH_KEY_Cleaned C:\path\to\SSH_KEY
```

Step 4: Validate the Key File Locally

To ensure that the private key is correctly formatted and functional, validate it using the following command:

command

```
ssh-keygen -y -f C:\path\to\SSH_KEY
```

This command should output the public key corresponding to the private key.

Check the File Format

The key file should start with:

vbnet

```
-----BEGIN OPENSSH PRIVATE KEY-----
```

and end with:

vbnet

```
-----END OPENSSH PRIVATE KEY-----
```

Ensure Key Pair Match

Make sure the **id_rsa** private key matches the public key added to `C:\Users\<UserName>\.ssh\authorized_keys` on the remote server.

Step 5: Configure `authorized_keys` and `known_hosts` on Windows

Adding to `authorized_keys`

Open the public key file (**id_rsa.pub**) in a text editor (e.g., Notepad).

Copy the entire contents of the public key file.

Manually add the public key to the `C:\Users\<UserName>\.ssh\authorized_keys` file on `the remote server.

command

```
echo "your_copied_key" >> C:\Users\<UserName>\.ssh\authorized_keys
```

Note: Once after successful creation of **authorized_keys**, edit the file in notepad and remove the double quotes at the starting and ending of the key.

Adding to known_hosts

To add a remote server to your **known_hosts** file on Windows:

Open Command prompt and run the following command:

Note: Replace the `remote_host` by with the ipaddress of the server.

command
`ssh-keyscan -H remote_host >> C:\Users\<UserName>\.ssh\known_hosts`

Step 6: Upload the Private Key to Azure DevOps

For use in Azure DevOps, upload the private key (**id_rsa**) as a secure file. Go through the link on how to Use secure files.

- Navigate to Pipelines
- Select **Library** in Azure DevOps
- Select **Secure Files**
- Click on **+ Secure File** and upload your private key (**id_rsa**).

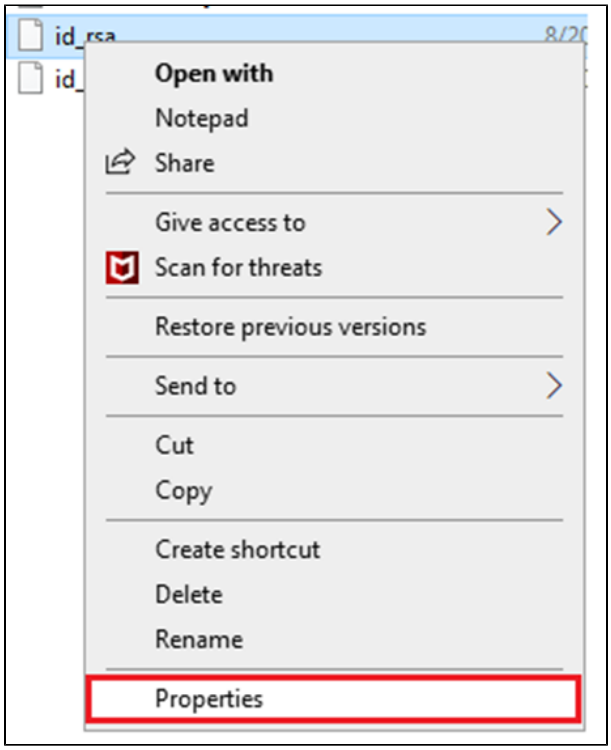
Step 7: Disable inheritance

a. id_rsa

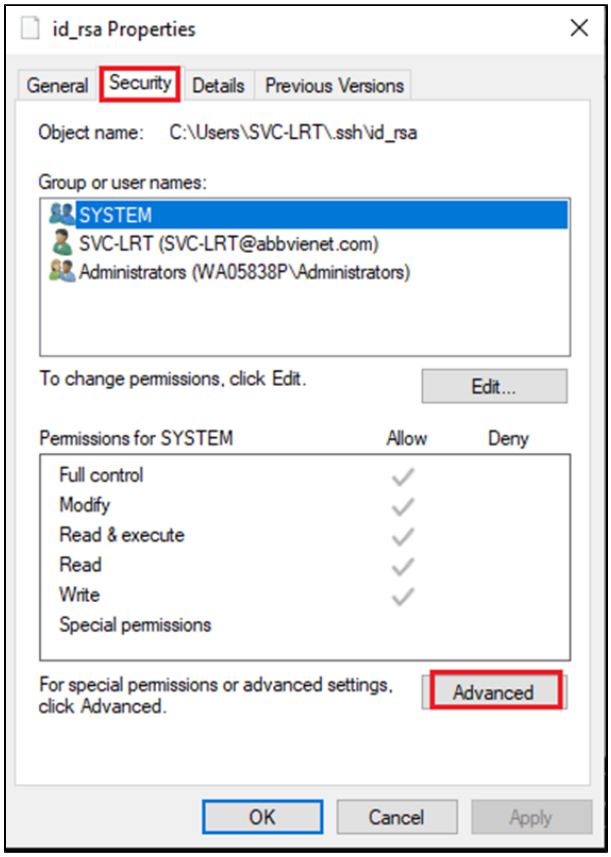
Turning off inheritance is often a safer and easier way to restrict access to a specific sub-folder.

To Disable inheritance

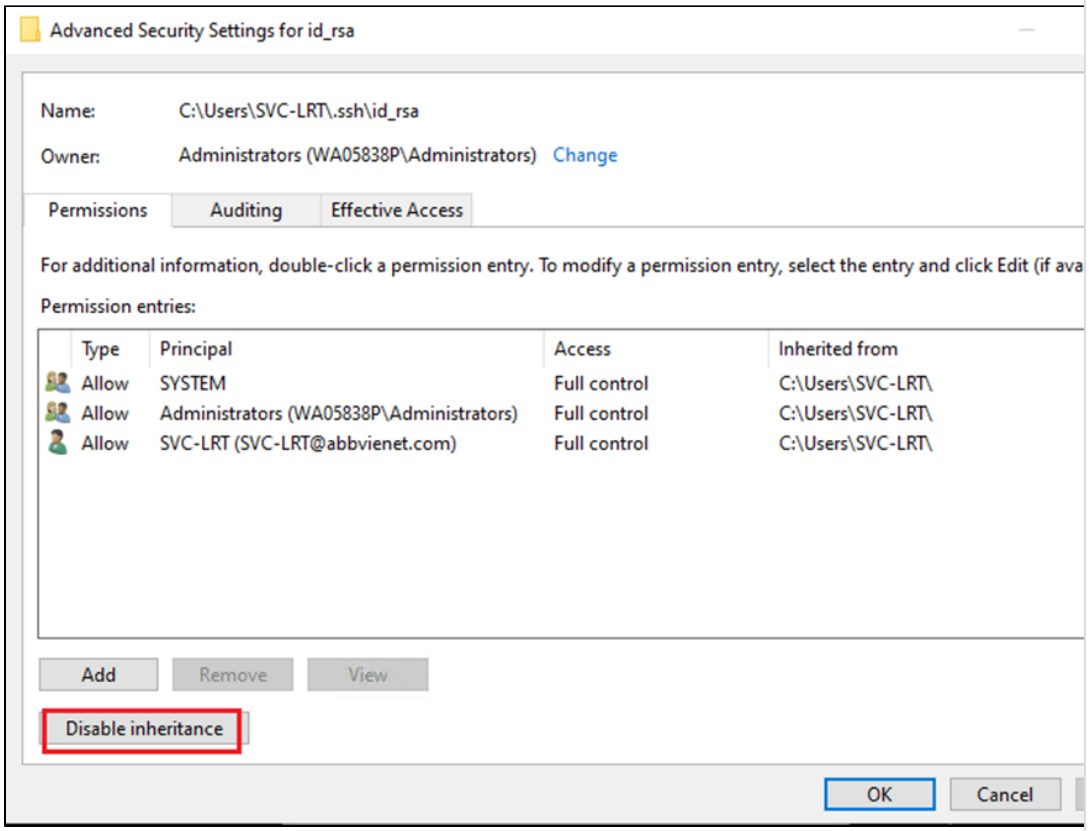
- Right Click and select the Properties of **id_rsa** file



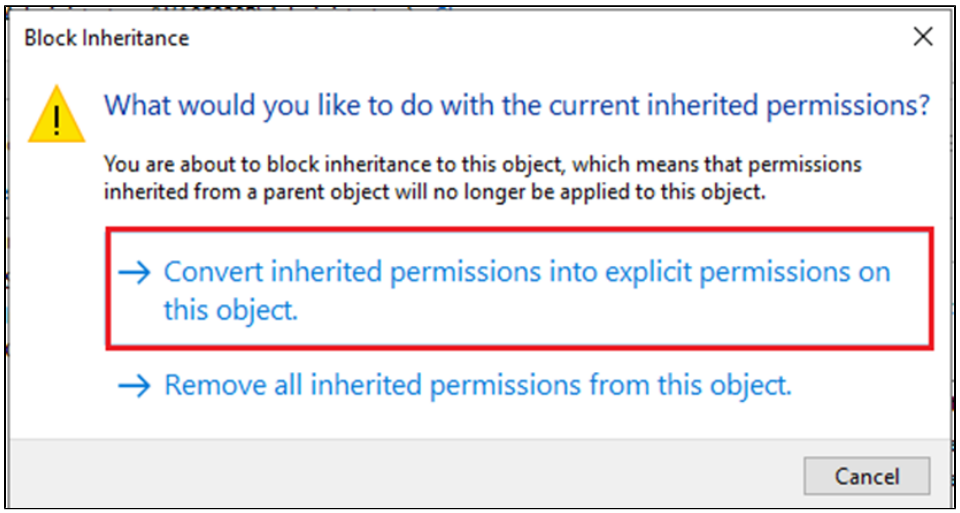
- Select **Security** tab and click on **Advanced**



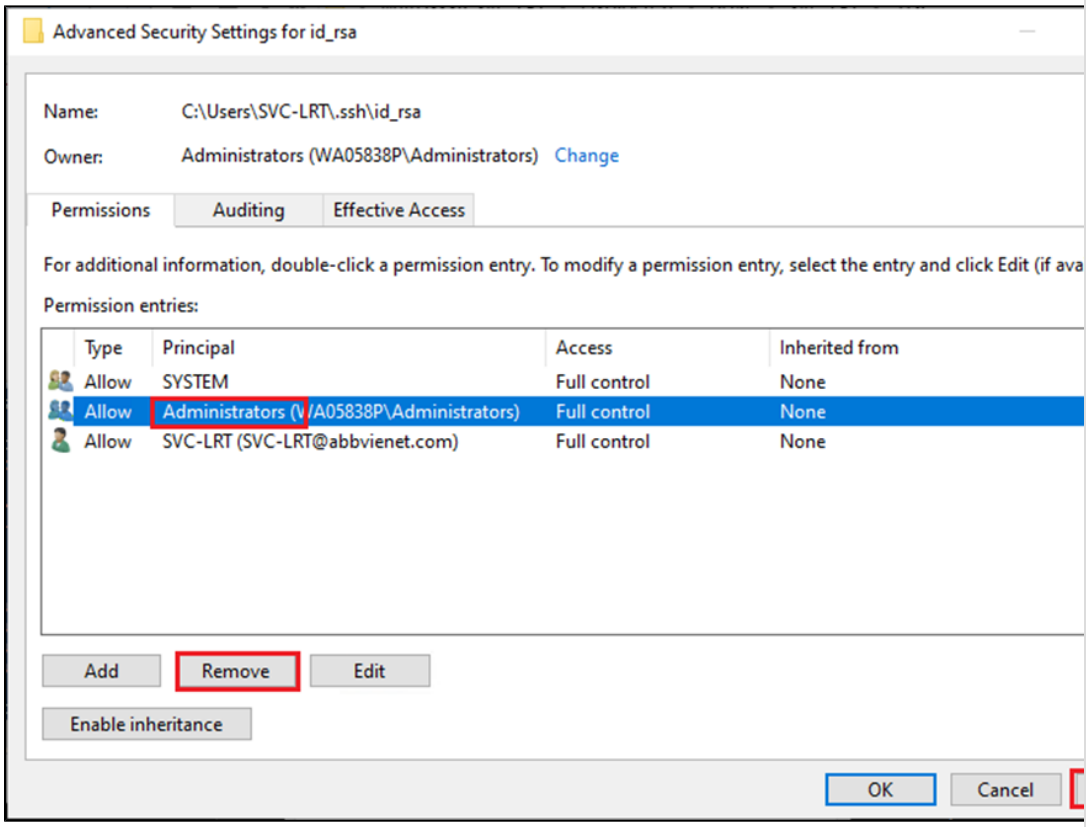
- When the below window appears, click on **Disable inheritance**



- Select as highlighted below



- Later, Select the **Administrator Remove Apply OK.**



b. authorized_keys

Repeat the steps under **7.a** for the file `authorized_keys`

c. Comment the code

It is required to comment out the following lines in `C:\ProgramData\ssh\sshd_config` file.

sshd_config

```
# Match Group administrators
#       AuthorizedKeysFile __PROGRAMDATA__/ssh/administrators_authorized_keys
```

d. Restart service

Once after completing the above step, restart the `OpenSSH SSH Server` service at `services.msc`.

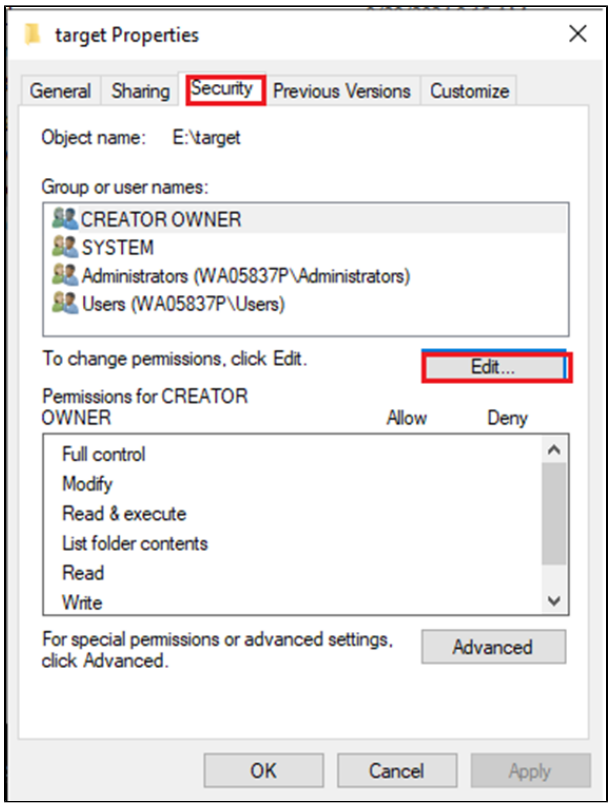
Step 8: Providing permissions for target folder

Note: Ignore this step, if your project doesn't have this kind of deployment process.

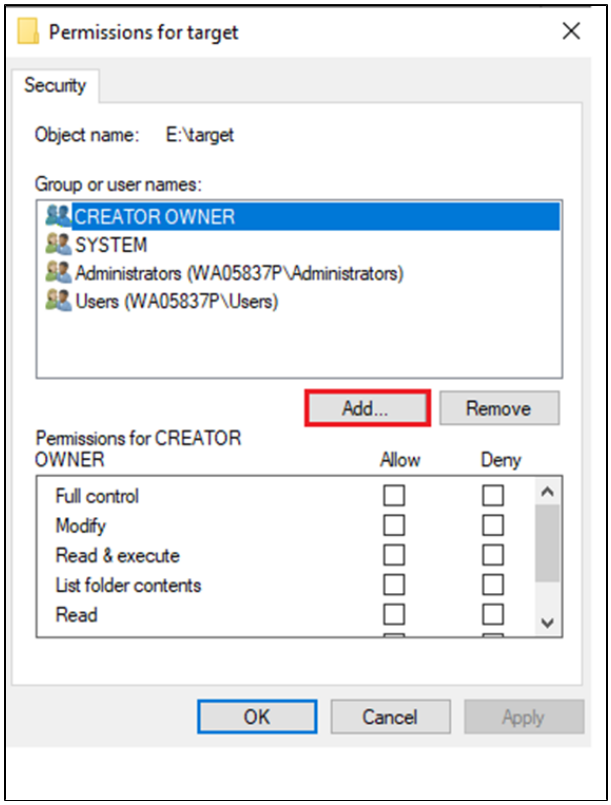
A **target** folder is created in the **E:** drive of the server. This folder is used to copy the artifact and execute the `deploy.ps1` file.

To make sure **SVC-LRT** should not be restricted through the pipeline, we have to provide necessary permissions to the **target** folder.

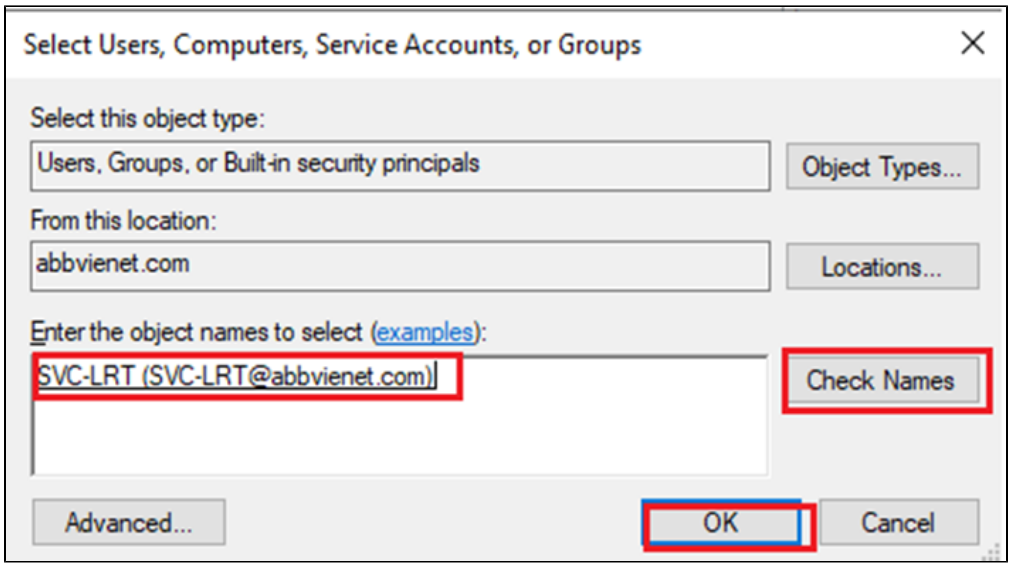
- Go to **E:** drive
- Right click on the **target** folder and select **Properties**
- Go to **Security** tab and click on **Edit**



- Select **Add** from the window appears



- Enter **SVC-LRT** and click on **Check Names**. Once the validation done and shows as below, click on **OK.**



- Select the **SVC-LRT** user and enable all the checkbox under **Allow**. Finally click **Apply**.