Local Git Repository

(The **ssh user@ip** command is used to establish an SSH (Secure Shell) connection to a remote server or device using the specified username (**user**) and IP address (**ip**).

```
ssh user1@162.15.8.51)
```

Admin or Remote System Setup

Install Git: First, ensure that Git is installed on your Ubuntu desktop machine. If it's not installed, you can install it using the following command:

```
sudo apt-get update
sudo apt-get install git
```

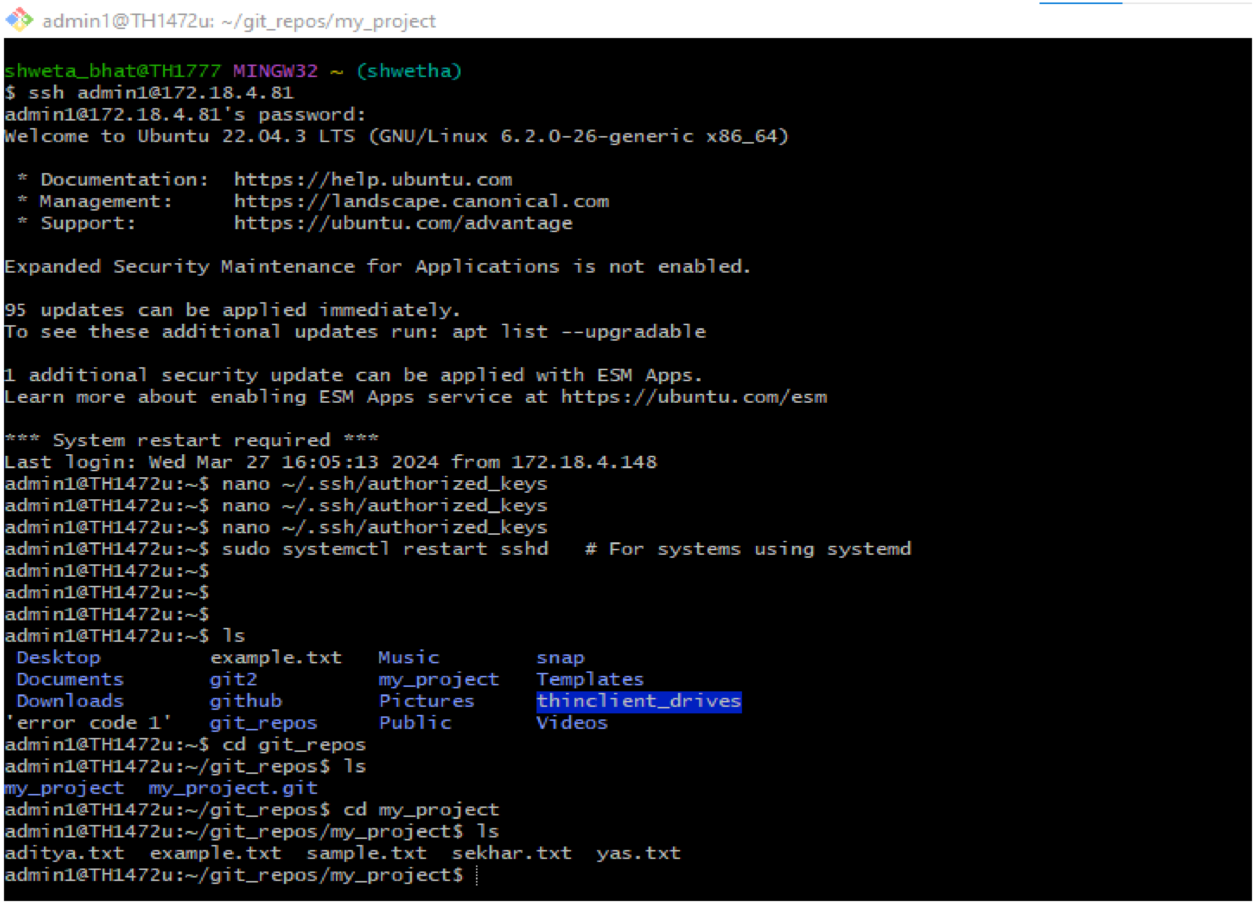
Create a Directory for Git Repositories: Choose a directory where you want to store your Git repositories. For example, you can create a directory named **git-repos** in your home directory:

```
mkdir  git-repos
```

Initialize a Bare Git Repository: Inside the **git-repos** directory, initialize a bare Git repository. A bare repository doesn't have a working directory and is typically used as a central repository for sharing code.

```
cd  git-repos
git init --bare my-project.git
```

Replace **my-project.git** with the desired name for your repository.



```
shweta_bhat@TH1777 MINGW32 ~ (shwetha)
$ ssh admin1@172.18.4.81
admin1@172.18.4.81's password:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 6.2.0-26-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

Expanded Security Maintenance for Applications is not enabled.

95 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

1 additional security update can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

*** System restart required ***
Last login: Wed Mar 27 16:05:13 2024 from 172.18.4.148
admin1@TH1472u:~$ nano ~/.ssh/authorized_keys
admin1@TH1472u:~$ nano ~/.ssh/authorized_keys
admin1@TH1472u:~$ nano ~/.ssh/authorized_keys
admin1@TH1472u:~$ sudo systemctl restart sshd   # For systems using systemd
admin1@TH1472u:~$
admin1@TH1472u:~$
admin1@TH1472u:~$
admin1@TH1472u:~$ ls
 Desktop          example.txt   Music        snap
 Documents        git2          my_project   Templates
 Downloads        github        Pictures     thinclient_drives
'error code 1'    git_repos     Public       Videos
admin1@TH1472u:~$ cd git_repos
admin1@TH1472u:~/git_repos$ ls
my_project  my_project.git
admin1@TH1472u:~/git_repos$ cd my_project
admin1@TH1472u:~/git_repos/my_project$ ls
aditya.txt  example.txt  sample.txt  sekhar.txt  yas.txt
admin1@TH1472u:~/git_repos/my_project$
```

Step 3: **Set Up SSH Key Authentication**

Ensure that SSH key authentication is set up for the users who will access the Git repository. Each user should generate an SSH key pair and provide you with their public key. Add these public keys to the authorized_keys file of the Git user.

```
sudo mkdir -p /home/git/.ssh

sudo nano /home/git/.ssh/authorized_keys
```

Paste the public keys of authorized users into the authorized_keys file, one per line. Save and exit the editor.

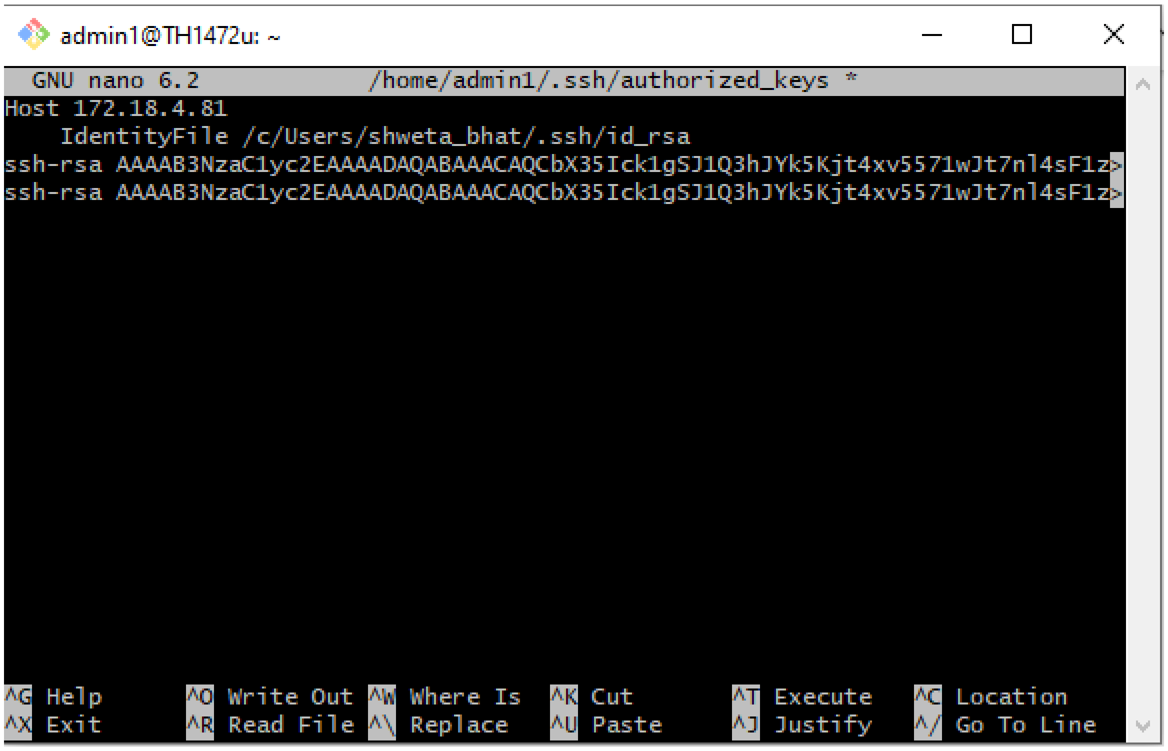
After pasting all the public keys, save the changes and exit the text editor (nano):

Press Ctrl + O to write the changes to the file.

Press Enter to confirm the filename.

Press Ctrl + X to exit the text editor.

To configure SSH authentication on the remote host system using the SSH key (/c/Users/shweta_bhat/.ssh/id_rsa) from your client system, you need to perform the following steps. Keep in mind that you'll need appropriate permissions on the remote host to complete these actions.



```
  GNU nano 6.2              /home/admin1/.ssh/authorized_keys *
Host 172.18.4.81
    IdentityFile /c/Users/shweta_bhat/.ssh/id_rsa
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQCbX35Ick1gSJ1Q3hJYk5Kjt4xv5571wJt7nl4sF1z>
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQCbX35Ick1gSJ1Q3hJYk5Kjt4xv5571wJt7nl4sF1z>

^G Help       ^O Write Out ^W Where Is  ^K Cut        ^T Execute    ^C Location
^X Exit       ^R Read File ^\ Replace   ^U Paste      ^J Justify    ^/ Go To Line
```

Copy Public Key to Remote Host:

(from client system after generating ssh key in their system they can add into host system by using ssh-copy-id command below , but password required)

First, you need to copy the public key (id_rsa.pub) corresponding to your private key (id_rsa) from the client system to the remote host. You can do this using the ssh-copy-id command or manually copying and appending the public key to the remote host's authorized_keys file.

Using ssh-copy-id (if available on the client system):

```
ssh-copy-id -i /c/Users/shweta_bhat/.ssh/id_rsa.pub user@remote_host
```

user@remote_host like (admin@163.14.6.41)

you can manually copy and append the public key:

in remote system (admin) directly :

file will be in this path

~/.ssh/authorized_keys'

Step 4: **Grant Permissions to the Repository**

Set permissions to allow users to read from and write to the Git repository.

```
sudo chown -R git:git ~/git_repos/my_project.git
sudo chmod -R 775 ~/git_repos/my_project.git
```

Client Side Commands (git bash)

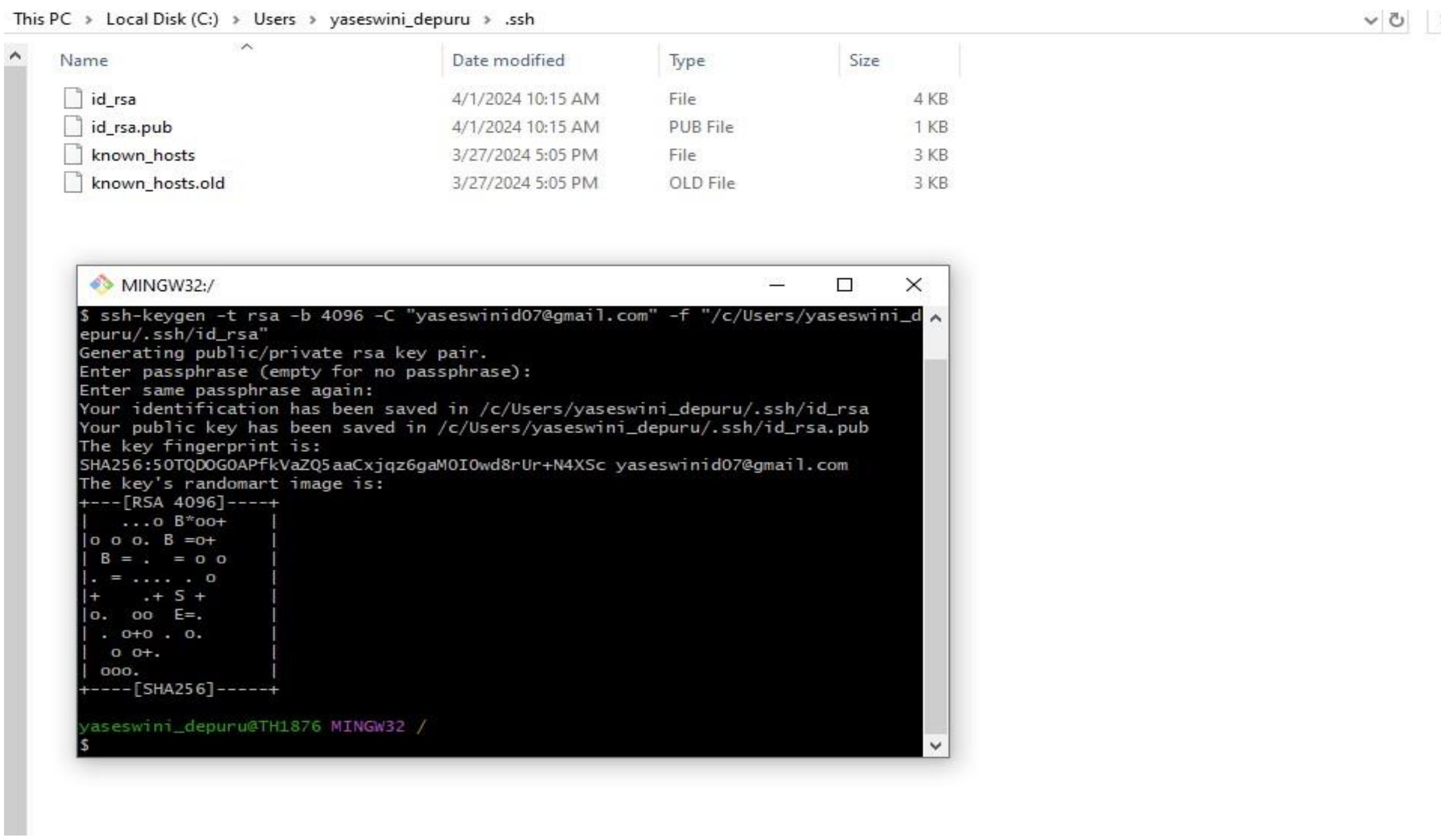
Generating an SSH key pair

In Git Bash, use the ssh-keygen command to generate an SSH key pair. Specify the desired path for the private key (id_rsa) using the -f option. For example, to generate the key pair and save it to /c/Users/user_name/.ssh/id_rsa, run the following command:

.ssh file will be in the system in the above path

```
ssh-keygen -t rsa -b 4096 -C "your_email@example.com" -f "/c/Users/shweta_bhat/.ssh/id_rsa"
```

Replace "your_email@example.com" with your actual email address. This email address is used as a comment in the SSH key.



This PC > Local Disk (C:) > Users > yaseswini_depuru > .ssh

Name	Date modified	Type	Size
id_rsa	4/1/2024 10:15 AM	File	4 KB
id_rsa.pub	4/1/2024 10:15 AM	PUB File	1 KB
known_hosts	3/27/2024 5:05 PM	File	3 KB
known_hosts.old	3/27/2024 5:05 PM	OLD File	3 KB

MINGW32:/

```
$ ssh-keygen -t rsa -b 4096 -C "yaseswinid07@gmail.com" -f "/c/Users/yaseswini_d
epuru/.ssh/id_rsa"
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/yaseswini_depuru/.ssh/id_rsa
Your public key has been saved in /c/Users/yaseswini_depuru/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:5OTQDOGOAPfkVaZQ5aaCxjqz6gaMOIOwd8rUr+N4XSc yaseswinid07@gmail.com
The key's randomart image is:
+---[RSA 4096]----+
|    ...o B*oo+   |
|o o o. B =o+     |
| B = .  = o o    |
|. = .... . o     |
|+    .+ S +      |
|o.  oo  E=.      |
| . o+o . o.      |
|  o o+.          |
| ooo.            |
+----[SHA256]-----+

yaseswini_depuru@TH1876 MINGW32 /
$
```

Copy Public Key to Remote Host:

First, you need to copy the public key (id_rsa.pub) corresponding to your private key (id_rsa) from the client system to the remote host. You can do this using the ssh-copy-id command or manually copying and appending the public key to the remote host's authorized_keys file.

Using ssh-copy-id (if available on the client system):

++++++++++++++++++++++++++++++++++

ssh-copy-id -i /c/Users/shweta_bhat/.ssh/id_rsa.pub user@remote_host

++++++++++++++++++++++++++++++++++++

while cloning using ssh-agent :

Open git bash in the folder where you want to clone project

+++++++++++++++++++++++++++++++

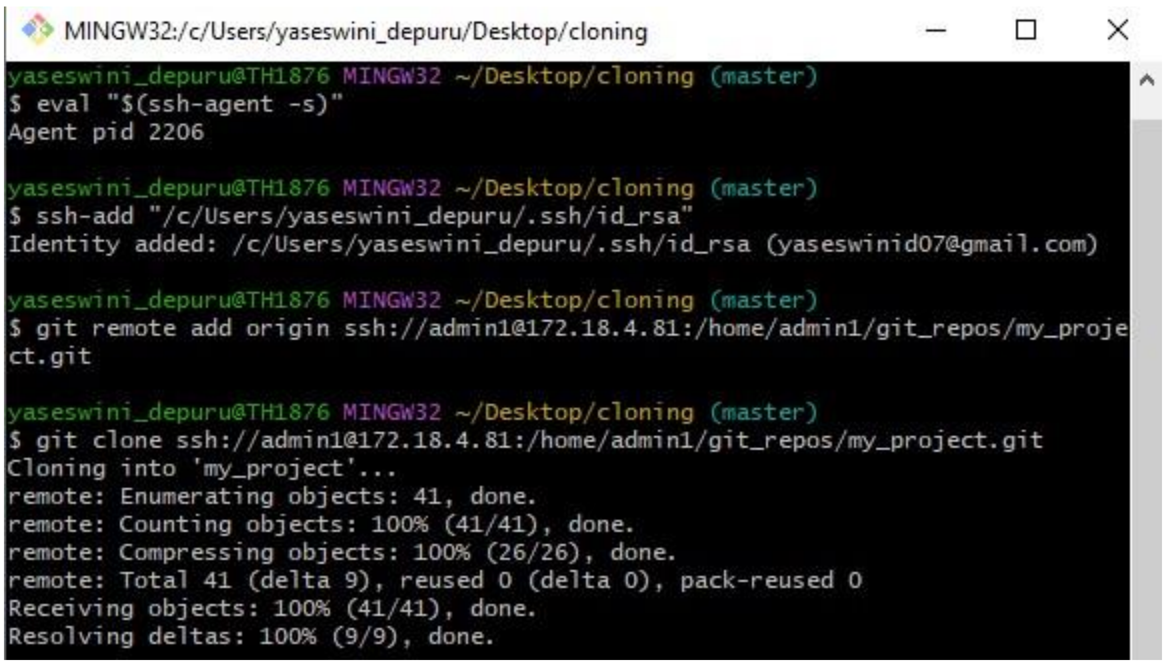
eval "$(ssh-agent -s)"

ssh-add "/c/Users/shweta_bhat/.ssh/id_rsa"

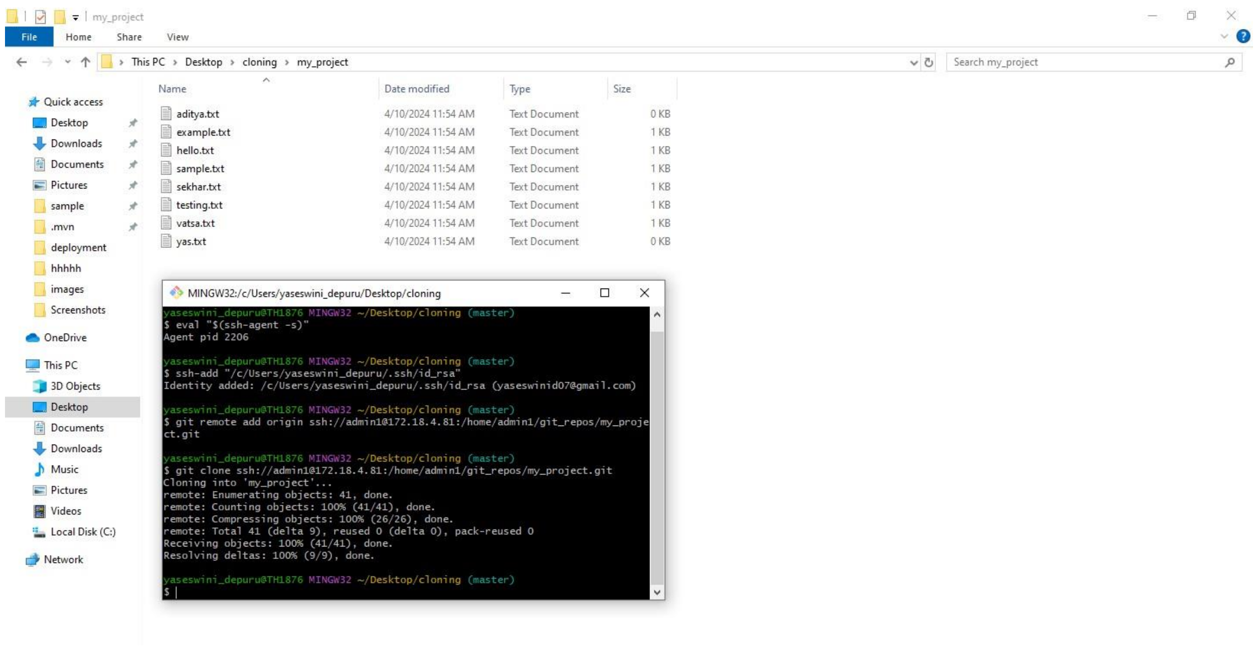
++++++++++++++++++++++++++++

git remote add origin ssh://username@server_ip:/path/to/repos/myrepo.git

git clone ssh://admin1@172.18.4.81:/home/admin1/git_repos/my_project.git



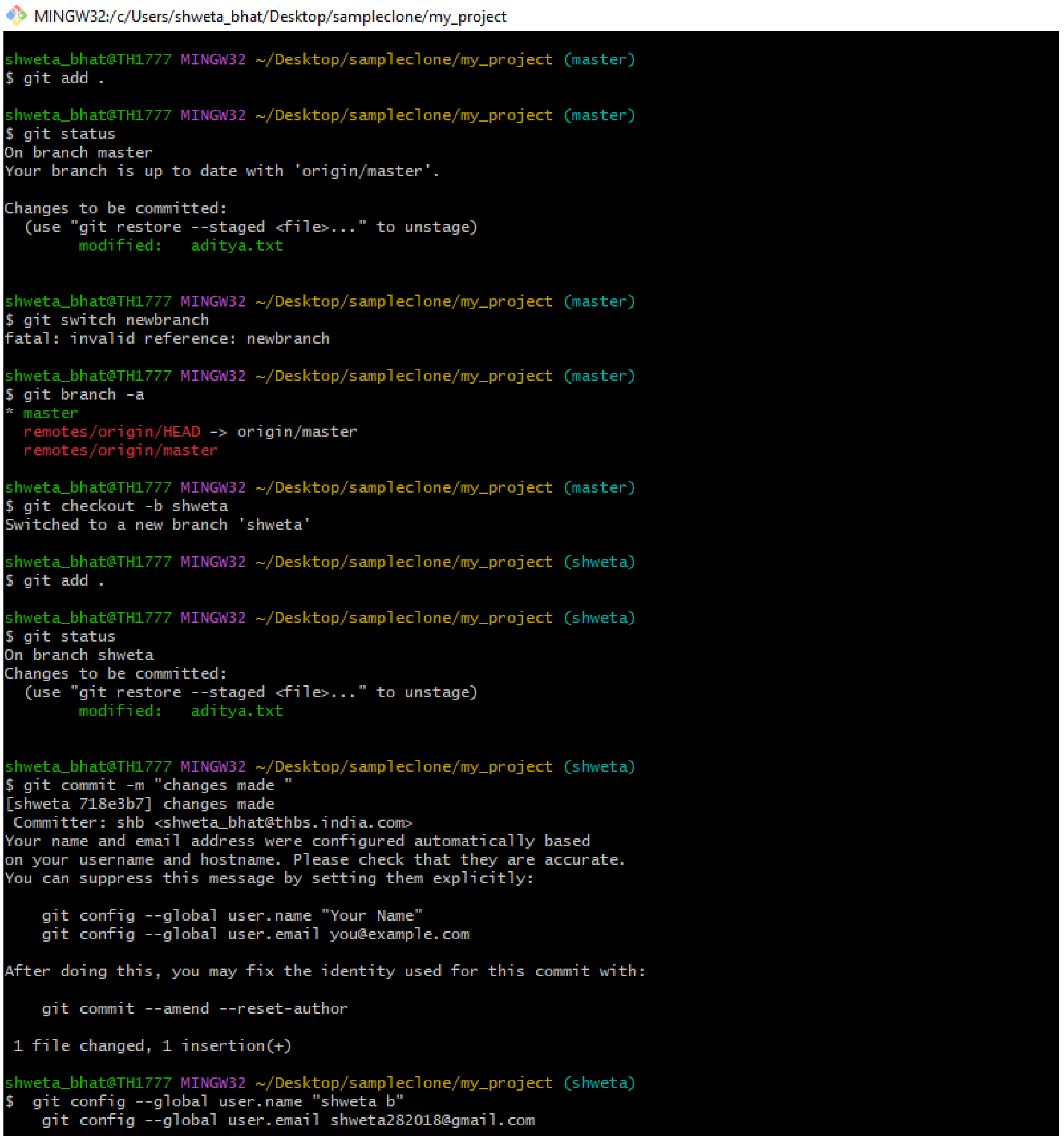
```
yaseswini_depuru@TH1876 MINGW32 ~/Desktop/cloning (master)
$ eval "$(ssh-agent -s)"
Agent pid 2206

yaseswini_depuru@TH1876 MINGW32 ~/Desktop/cloning (master)
$ ssh-add "/c/Users/yaseswini_depuru/.ssh/id_rsa"
Identity added: /c/Users/yaseswini_depuru/.ssh/id_rsa (yaseswinid07@gmail.com)

yaseswini_depuru@TH1876 MINGW32 ~/Desktop/cloning (master)
$ git remote add origin ssh://admin1@172.18.4.81:/home/admin1/git_repos/my_proje
ct.git

yaseswini_depuru@TH1876 MINGW32 ~/Desktop/cloning (master)
$ git clone ssh://admin1@172.18.4.81:/home/admin1/git_repos/my_project.git
Cloning into 'my_project'...
remote: Enumerating objects: 41, done.
remote: Counting objects: 100% (41/41), done.
remote: Compressing objects: 100% (26/26), done.
remote: Total 41 (delta 9), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (41/41), done.
Resolving deltas: 100% (9/9), done.
```



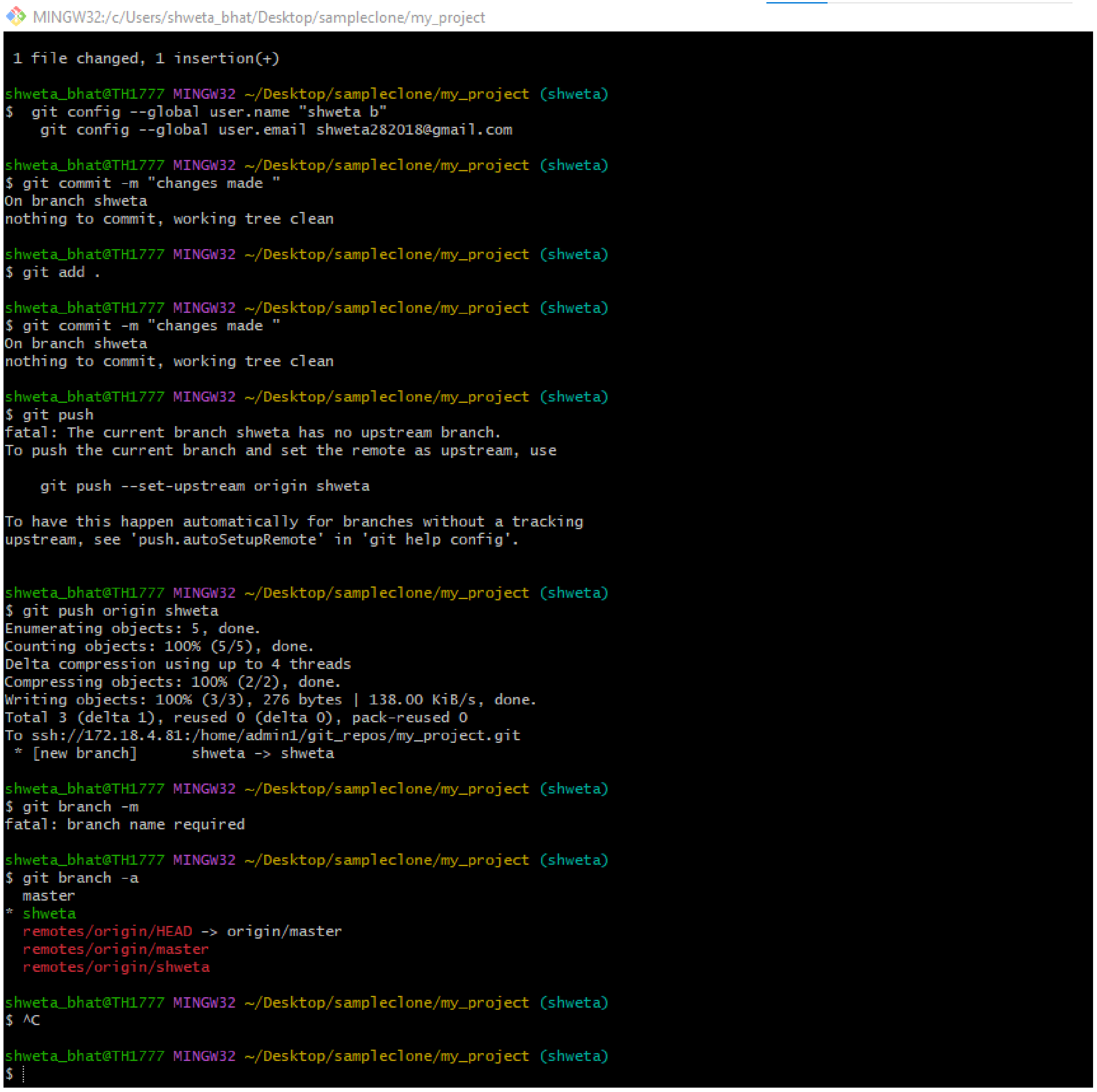
Name	Date modified	Type	Size
aditya.txt	4/10/2024 11:54 AM	Text Document	0 KB
example.txt	4/10/2024 11:54 AM	Text Document	1 KB
hello.txt	4/10/2024 11:54 AM	Text Document	1 KB
sample.txt	4/10/2024 11:54 AM	Text Document	1 KB
sekhar.txt	4/10/2024 11:54 AM	Text Document	1 KB
testing.txt	4/10/2024 11:54 AM	Text Document	1 KB
vatsa.txt	4/10/2024 11:54 AM	Text Document	1 KB
yas.txt	4/10/2024 11:54 AM	Text Document	0 KB

```
yaseswini_depuru@TH1876 MINGW32 ~/Desktop/cloning (master)
$ eval "$(ssh-agent -s)"
Agent pid 2206

yaseswini_depuru@TH1876 MINGW32 ~/Desktop/cloning (master)
$ ssh-add "/c/Users/yaseswini_depuru/.ssh/id_rsa"
Identity added: /c/Users/yaseswini_depuru/.ssh/id_rsa (yaseswinid07@gmail.com)

yaseswini_depuru@TH1876 MINGW32 ~/Desktop/cloning (master)
$ git remote add origin ssh://admin1@172.18.4.81:/home/admin1/git_repos/my_proje
ct.git

yaseswini_depuru@TH1876 MINGW32 ~/Desktop/cloning (master)
$ git clone ssh://admin1@172.18.4.81:/home/admin1/git_repos/my_project.git
Cloning into 'my_project'...
remote: Enumerating objects: 41, done.
remote: Counting objects: 100% (41/41), done.
remote: Compressing objects: 100% (26/26), done.
remote: Total 41 (delta 9), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (41/41), done.
Resolving deltas: 100% (9/9), done.

yaseswini_depuru@TH1876 MINGW32 ~/Desktop/cloning (master)
$ |
```

After cloning my_project from the remote to client system , open the git bash in the my_project folder , execute git commands .


MINGW32:/c/Users/shweta_bhat/Desktop/sampleclone/my_project

```
shweta_bhat@TH1777 MINGW32 ~/Desktop/sampleclone/my_project (master)
$ git add .

shweta_bhat@TH1777 MINGW32 ~/Desktop/sampleclone/my_project (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   aditya.txt


shweta_bhat@TH1777 MINGW32 ~/Desktop/sampleclone/my_project (master)
$ git switch newbranch
fatal: invalid reference: newbranch

shweta_bhat@TH1777 MINGW32 ~/Desktop/sampleclone/my_project (master)
$ git branch -a
* master
  remotes/origin/HEAD -> origin/master
  remotes/origin/master

shweta_bhat@TH1777 MINGW32 ~/Desktop/sampleclone/my_project (master)
$ git checkout -b shweta
Switched to a new branch 'shweta'

shweta_bhat@TH1777 MINGW32 ~/Desktop/sampleclone/my_project (shweta)
$ git add .

shweta_bhat@TH1777 MINGW32 ~/Desktop/sampleclone/my_project (shweta)
$ git status
On branch shweta
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   aditya.txt


shweta_bhat@TH1777 MINGW32 ~/Desktop/sampleclone/my_project (shweta)
$ git commit -m "changes made "
[shweta 718e3b7] changes made
 Committer: shb <shweta_bhat@thbs.india.com>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:

    git config --global user.name "Your Name"
    git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 1 file changed, 1 insertion(+)

shweta_bhat@TH1777 MINGW32 ~/Desktop/sampleclone/my_project (shweta)
$  git config --global user.name "shweta b"
    git config --global user.email shweta282018@gmail.com
```

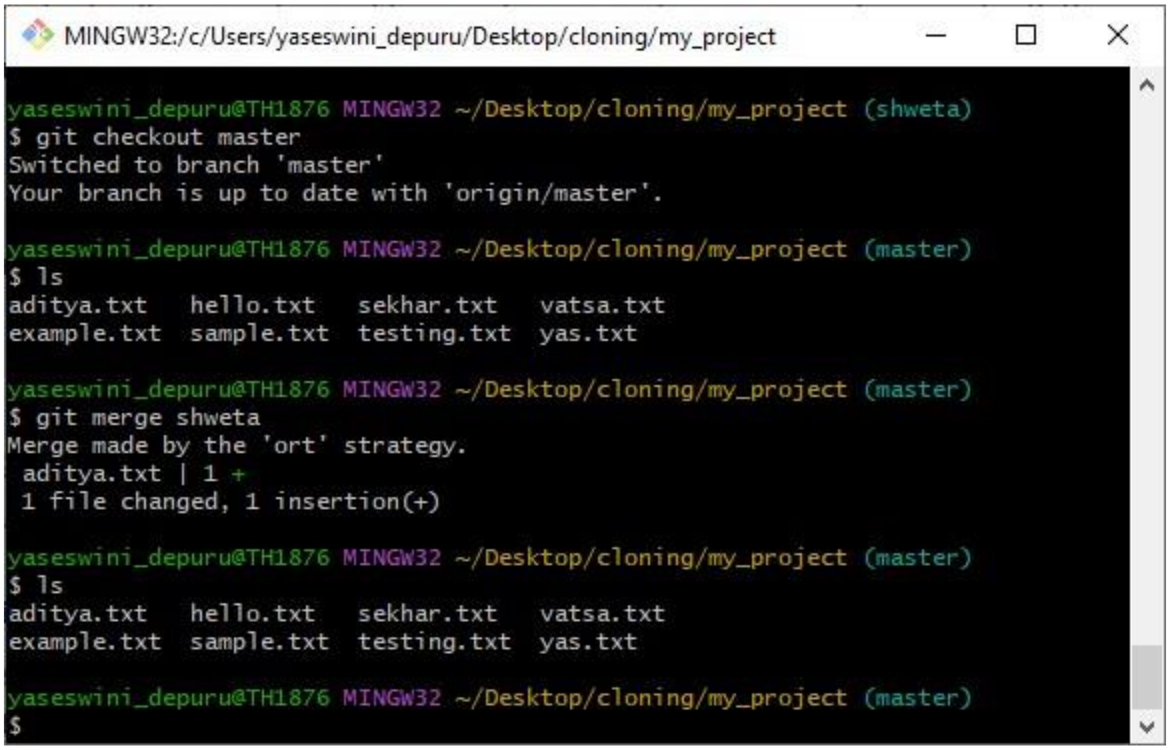

MINGW32:/c/Users/shweta_bhat/Desktop/sampleclone/my_project

```
 1 file changed, 1 insertion(+)

shweta_bhat@TH1777 MINGW32 ~/Desktop/sampleclone/my_project (shweta)
$  git config --global user.name "shweta b"
    git config --global user.email shweta282018@gmail.com

shweta_bhat@TH1777 MINGW32 ~/Desktop/sampleclone/my_project (shweta)
$ git commit -m "changes made "
On branch shweta
nothing to commit, working tree clean

shweta_bhat@TH1777 MINGW32 ~/Desktop/sampleclone/my_project (shweta)
$ git add .

shweta_bhat@TH1777 MINGW32 ~/Desktop/sampleclone/my_project (shweta)
$ git commit -m "changes made "
On branch shweta
nothing to commit, working tree clean

shweta_bhat@TH1777 MINGW32 ~/Desktop/sampleclone/my_project (shweta)
$ git push
fatal: The current branch shweta has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin shweta

To have this happen automatically for branches without a tracking
upstream, see 'push.autoSetupRemote' in 'git help config'.


shweta_bhat@TH1777 MINGW32 ~/Desktop/sampleclone/my_project (shweta)
$ git push origin shweta
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 276 bytes | 138.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
To ssh://172.18.4.81:/home/admin1/git_repos/my_project.git
 * [new branch]      shweta -> shweta

shweta_bhat@TH1777 MINGW32 ~/Desktop/sampleclone/my_project (shweta)
$ git branch -m
fatal: branch name required

shweta_bhat@TH1777 MINGW32 ~/Desktop/sampleclone/my_project (shweta)
$ git branch -a
  master
* shweta
  remotes/origin/HEAD -> origin/master
  remotes/origin/master
  remotes/origin/shweta

shweta_bhat@TH1777 MINGW32 ~/Desktop/sampleclone/my_project (shweta)
$ ^C

shweta_bhat@TH1777 MINGW32 ~/Desktop/sampleclone/my_project (shweta)
$
```

We can merge two existing branches

* git checkout branch_to_merge_into (master)

* git merge feature_branch(shweta)

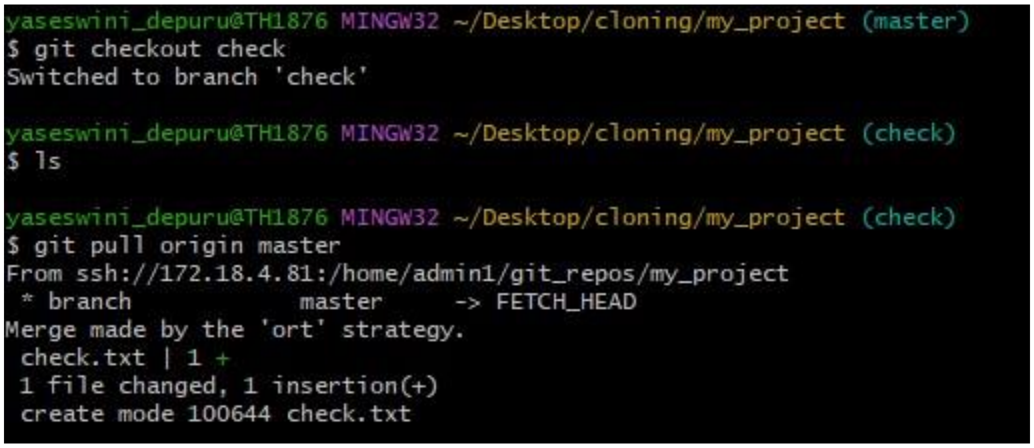


MINGW32:/c/Users/yaseswini_depuru/Desktop/cloning/my_project

```
yaseswini_depuru@TH1876 MINGW32 ~/Desktop/cloning/my_project (shweta)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

yaseswini_depuru@TH1876 MINGW32 ~/Desktop/cloning/my_project (master)
$ ls
aditya.txt   hello.txt   sekhar.txt   vatsa.txt
example.txt  sample.txt  testing.txt  yas.txt

yaseswini_depuru@TH1876 MINGW32 ~/Desktop/cloning/my_project (master)
$ git merge shweta
Merge made by the 'ort' strategy.
 aditya.txt | 1 +
 1 file changed, 1 insertion(+)

yaseswini_depuru@TH1876 MINGW32 ~/Desktop/cloning/my_project (master)
$ ls
aditya.txt   hello.txt   sekhar.txt   vatsa.txt
example.txt  sample.txt  testing.txt  yas.txt

yaseswini_depuru@TH1876 MINGW32 ~/Desktop/cloning/my_project (master)
$
```

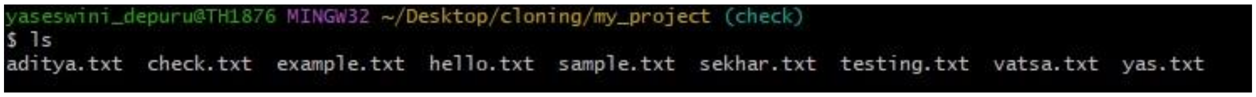
We can pull from one branch to another
* create a new branch - “git checkout -b check”

- Pull from the branch - “git pull origin master”



```
yaseswini_depuru@TH1876 MINGW32 ~/Desktop/cloning/my_project (master)
$ git checkout check
Switched to branch 'check'

yaseswini_depuru@TH1876 MINGW32 ~/Desktop/cloning/my_project (check)
$ ls

yaseswini_depuru@TH1876 MINGW32 ~/Desktop/cloning/my_project (check)
$ git pull origin master
From ssh://172.18.4.81:/home/admin1/git_repos/my_project
 * branch            master     -> FETCH_HEAD
Merge made by the 'ort' strategy.
 check.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 check.txt
```

All the files are pulled from ‘master’ branch to ‘check’


```
yaseswini_depuru@TH1876 MINGW32 ~/Desktop/cloning/my_project (check)
$ ls
aditya.txt  check.txt  example.txt  hello.txt  sample.txt  sekhar.txt  testing.txt  vatsa.txt  yas.txt
```

Create Back up of the data available periodically to enable replication

Create a Scripts Directory:

mkdir ~/scripts

Navigate to the Scripts Directory:

cd ~/scripts

Create the Backup Script:

nano backup.sh

Script :

```
#!/bin/bash

# Define paths
repo_path="/path/to/your/git/repository"
backup_dir="/path/to/backup/directory"

#/home/admin1/git_repos/my_project
```

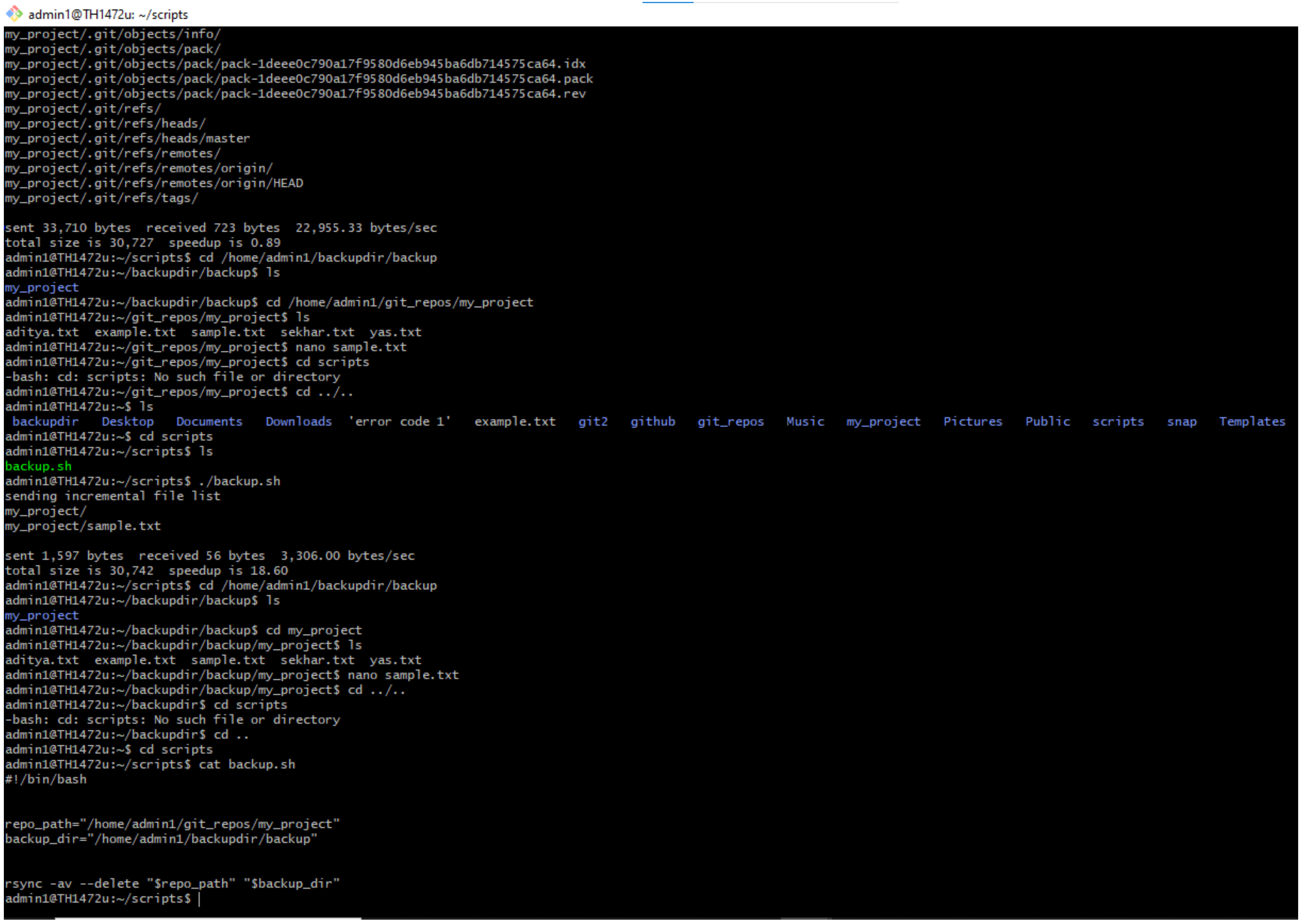
```
#/home/admin1/backupdir/backup

# Perform backup using rsync
rsync -av --delete "$repo_path" "$backup_dir"
```

Make the Script Executable:

```
chmod +x backup.sh
```

Run the Script :

```
./backup.sh
```



admin1@TH1472u: ~/scripts

```
my_project/.git/objects/info/
my_project/.git/objects/pack/
my_project/.git/objects/pack/pack-1deee0c790a17f9580d6eb945ba6db714575ca64.idx
my_project/.git/objects/pack/pack-1deee0c790a17f9580d6eb945ba6db714575ca64.pack
my_project/.git/objects/pack/pack-1deee0c790a17f9580d6eb945ba6db714575ca64.rev
my_project/.git/refs/
my_project/.git/refs/heads/
my_project/.git/refs/heads/master
my_project/.git/refs/remotes/
my_project/.git/refs/remotes/origin/
my_project/.git/refs/remotes/origin/HEAD
my_project/.git/refs/tags/

sent 33,710 bytes  received 723 bytes  22,955.33 bytes/sec
total size is 30,727  speedup is 0.89
admin1@TH1472u:~/scripts$ cd /home/admin1/backupdir/backup
admin1@TH1472u:~/backupdir/backup$ ls
my_project
admin1@TH1472u:~/backupdir/backup$ cd /home/admin1/git_repos/my_project
admin1@TH1472u:~/git_repos/my_project$ ls
aditya.txt  example.txt  sample.txt  sekhar.txt  yas.txt
admin1@TH1472u:~/git_repos/my_project$ nano sample.txt
admin1@TH1472u:~/git_repos/my_project$ cd scripts
-bash: cd: scripts: No such file or directory
admin1@TH1472u:~/git_repos/my_project$ cd ../..
admin1@TH1472u:~$ ls
 backupdir   Desktop   Documents   Downloads  'error code 1'   example.txt   git2   github   git_repos   Music   my_project   Pictures   Public   scripts   snap   Templates
admin1@TH1472u:~$ cd scripts
admin1@TH1472u:~/scripts$ ls
backup.sh
admin1@TH1472u:~/scripts$ ./backup.sh
sending incremental file list
my_project/
my_project/sample.txt

sent 1,597 bytes  received 56 bytes  3,306.00 bytes/sec
total size is 30,742  speedup is 18.60
admin1@TH1472u:~/scripts$ cd /home/admin1/backupdir/backup
admin1@TH1472u:~/backupdir/backup$ ls
my_project
admin1@TH1472u:~/backupdir/backup$ cd my_project
admin1@TH1472u:~/backupdir/backup/my_project$ ls
aditya.txt  example.txt  sample.txt  sekhar.txt  yas.txt
admin1@TH1472u:~/backupdir/backup/my_project$ nano sample.txt
admin1@TH1472u:~/backupdir/backup/my_project$ cd ../..
admin1@TH1472u:~/backupdir$ cd scripts
-bash: cd: scripts: No such file or directory
admin1@TH1472u:~/backupdir$ cd ..
admin1@TH1472u:~$ cd scripts
admin1@TH1472u:~/scripts$ cat backup.sh
#!/bin/bash


repo_path="/home/admin1/git_repos/my_project"
backup_dir="/home/admin1/backupdir/backup"


rsync -av --delete "$repo_path" "$backup_dir"
admin1@TH1472u:~/scripts$ |
```

Set Up Periodic Execution:

Edit Cron Jobs:

crontab -e

- It will ask for select any editor , select it .

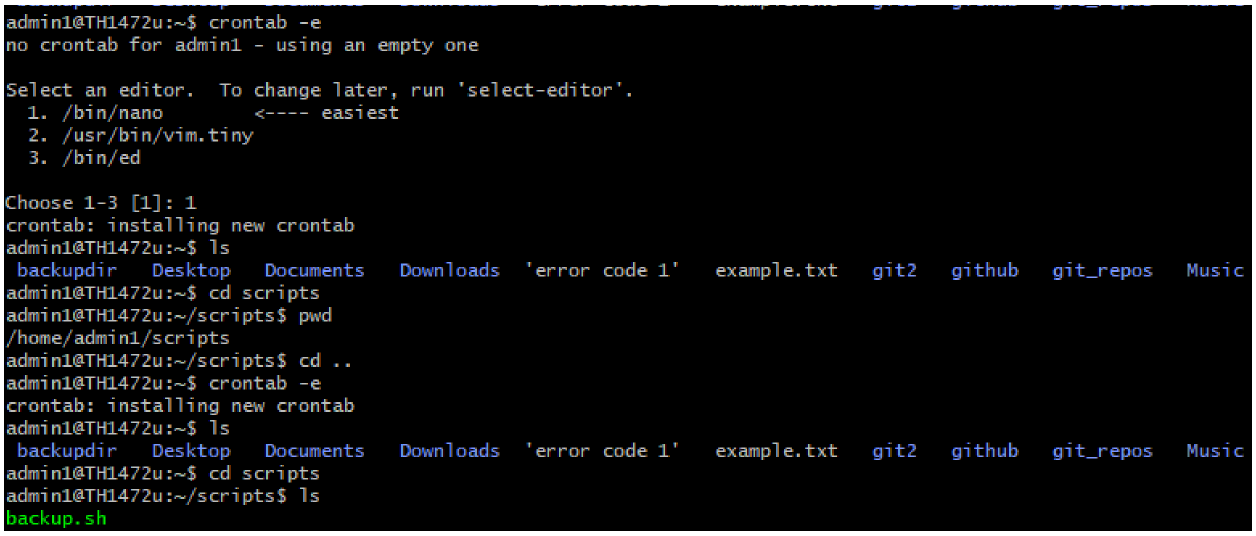
Add Cron Job Entry:

0 0 * * * /path/to/scripts/backup.sh

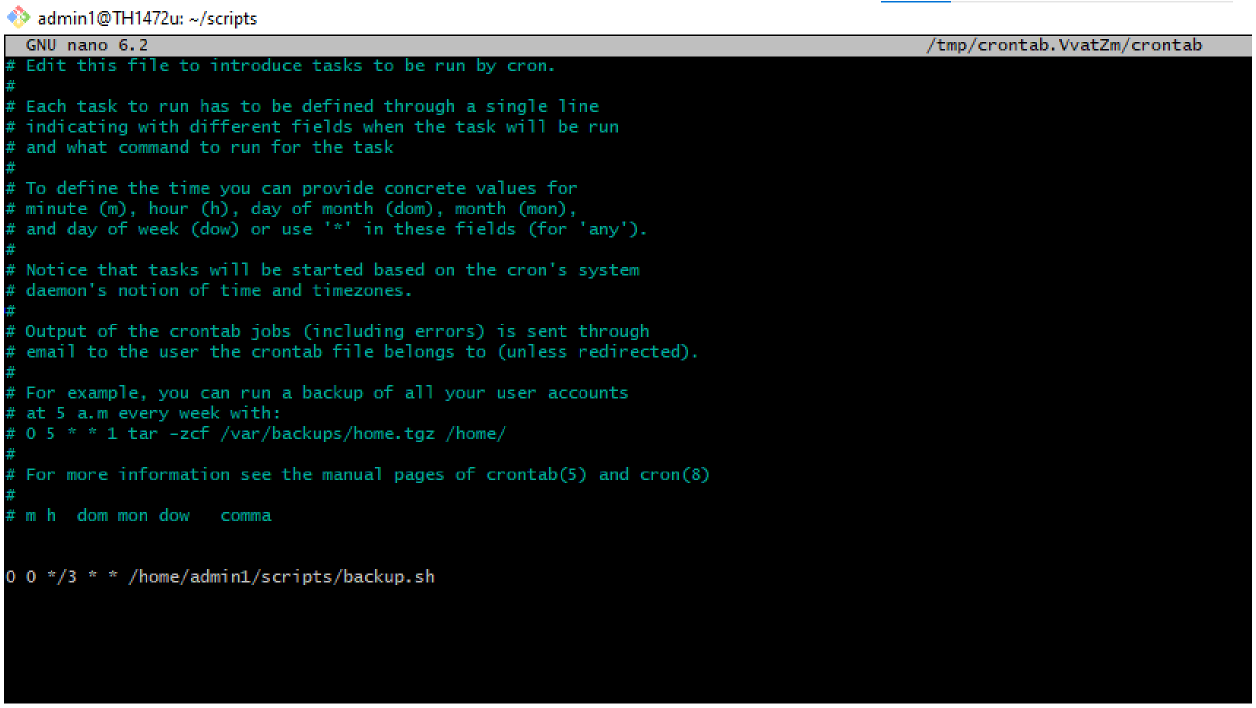
#0 0 */3 * * /home/admin1/scripts/backup.sh

Pattern :

minute hour day month day_of_week command_to_execute



```
admin1@TH1472u:~$ crontab -e
no crontab for admin1 - using an empty one

Select an editor.  To change later, run 'select-editor'.
  1. /bin/nano        <---- easiest
  2. /usr/bin/vim.tiny
  3. /bin/ed

Choose 1-3 [1]: 1
crontab: installing new crontab
admin1@TH1472u:~$ ls
 backupdir   Desktop   Documents   Downloads  'error code 1'   example.txt   git2   github   git_repos   Music
admin1@TH1472u:~$ cd scripts
admin1@TH1472u:~/scripts$ pwd
/home/admin1/scripts
admin1@TH1472u:~/scripts$ cd ..
admin1@TH1472u:~$ crontab -e
crontab: installing new crontab
admin1@TH1472u:~$ ls
 backupdir   Desktop   Documents   Downloads  'error code 1'   example.txt   git2   github   git_repos   Music
admin1@TH1472u:~$ cd scripts
admin1@TH1472u:~/scripts$ ls
backup.sh
```



admin1@TH1472u: ~/scripts

```
  GNU nano 6.2                                                        /tmp/crontab.VvatZm/crontab
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   comma


0 0 */3 * * /home/admin1/scripts/backup.sh
```