A Simple Git Server

rev 1.0 - 2025-02-06 - Initial version

Table of Contents

Introduction....2
Server Configuration....4
Create a git User on the Server....4
Create a git User on a Mac Server....4
Create a git User on Linux as the Server....5
Configure a Fixed IP Address on the Server....6
Configure a Fixed IP Address on the Mac....6
Configure a Fixed IP Address on Linux....6
Setup Remote Login....7
Setup Remote Login on the Mac Server....7
Setup Remote Login on a Linux Server....7
Create the Root Folder (once) on the Server....7
Create a New Repository on the Mac or Linux Server....8
Clone an Existing Repos from Github....8
List the Available Repos....8
Configure the Client Computers....9
Using a Repo....11
Back Up the Server....12

Introduction

This document describes a method for configuring a minimal **git** data server on a **Mac** (or **Linux**) computer.

The **git** data server will share **repositories** via **ssh** on a **local network** (router) but will not be accessible via the internet. An internet-accessible solution is out of scope of this document at this time.

> The procedure should be similar in concept for a **Linux** server, but this is not fully demonstrated in this document, and may vary from one variant or distribution to another.

Target audience

The solo developer or a small garage team requiring free private repositories from one or more client computers over a local network. For a more capable solution, see Gitea, for instance.

Features

With this setup you get all of the **git** features across multiple computers on your local network.

> Since this is not a full blown service such a Github, Gitlab, you are not getting the wide range of internet-based collaborative GUI features available from these services. However, of course you can use popular GUIs, including Visual Studio Code and the GUIs listed here.

Prerequisites

- A Mac or Linux computer to use as the git data server.
  - The same computer can act as a client as well.

- The following command-line applications on the Mac or Linux computer:
  - **git**
  - **ssh**

Topology

The central git data (bare) repositories will live under a user account named **git** on the git data server. The repositories will be accessible using **ssh** from the clients (instead of **https**), as long as they are on the same local network.

Server Configuration

Here we define a server as a Mac or Linux computer playing the role of a **git** data server, accessible via **ssh**. The server may be a laptop playing the dual role of git data server and git client computer. This chapter describes how to configure the server side.

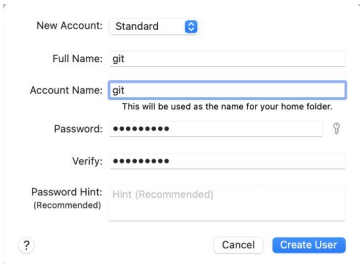
Create a git User on the Server

This section shows how to create a **git** user on a choice of Mac or Linux. The **git** user on the server will hold all the *bare repositories* to be shared with one or more client computers. This **git** user on the server provides the central storage (the data hub) for the shared repo data.

Create a git User on a Mac Server

To create the **git** user on the Mac.

- Login as an administrator on the Mac.
- Under **Systems Settings > Users & Groups** and click **Add Account...**
- Select a **Standard** account, enter **git** as the full name and account name, enter a password and click **Create User**. The password is for account access on the Mac and will not be used in **git** operations.



- Click **Create User**

The role of the **git** user on the server is to share the repository data with the client computers. However, by default, a new user on a Mac may have access to folders of other users on the computer. The following procedure applies anytime you create a new user for any purpose on a Mac:

> Let's say your main user is called **me**, right-click on **/Users/me** in the Finder and select **Get Info**, then open **Sharing and Permissions.** To keep the **/Users/me** data private to user **me**, make sure that **everyone** has **No Access**, and remove the **Staff** group, since the **git** user is part of **Staff**. Repeat for other users if necessary. To

learn more about setting folder permissions, please refer to the documentation for your Mac.

Create a git User on Linux as the Server

Create the user

```
sudo adduser git
```

To switch to the **git** user

```
su - git
```

To exit from the git user and back to the initial user, type `exit`.

To restrict the access to some other user account, for instance to ensure that the **git** user can't access files from that account

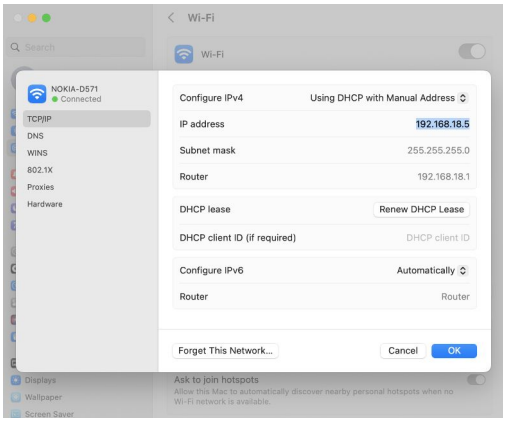
```
sudo chmod 700 /home/<some_other_account>
```

Configure a Fixed IP Address on the Server

A fixed IP address is best if you are using other computers as clients. This way the client computers can freeze the IP address in their hosts files for consistent access via a host name.

Configure a Fixed IP Address on the Mac

- Open **System Settings > Network** and click **Details...** to the right of your WiFi router's name, then select **TCP/IP** and choose **Using DHCP with Manual Address** under Configure IPv4, and finally enter your **IP** address (you pick the one that was provided by the DHCP).



- Click **OK**

Configure a Fixed IP Address on Linux

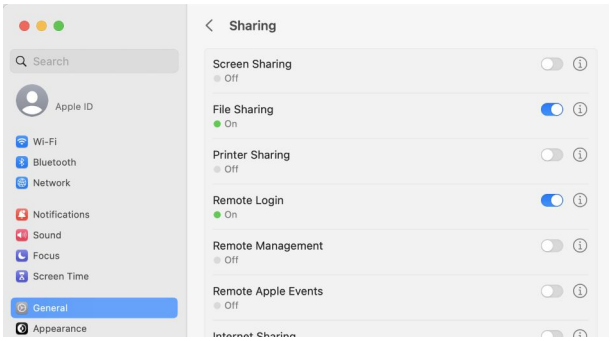
Please refer to your Linux documentation for how to configure a fixed IP address for the Linux computer acting as a server. This may differ depending on your Linux distribution and on whether you are using a Linux GUI or the command-line only.

Setup Remote Login

Setup Remote Login on the Mac Server

Remote login is required so that clients can use **ssh** to perform **git** operations.

- Under **System Settings > General > Sharing**, enable **Remote Login**



Setup Remote Login on a Linux Server

Refer to your Linux documentation for how to install or configure a ssh server such as `openssh-server`.

Create the Root Folder (once) on the Server

As mentioned previously, a bare repo acts as the database for a repository. For this reason, in the **git** account on the Mac or Linux server, we reserve a special folder as the root of all bare repositories, and we name this folder **~/<user>**.

Create a **~/<user>** folder (once) with read and write permissions. This is where all of the bare repos will live. In the following example we specify *peter* as the user.

```
cd ~git
mkdir peter
chmod a+w peter
```

Create a New Repository on the Mac or Linux Server

The concept is that anytime you need to create a new repo, you will create a *bare repo* under the ***~/<user>*** folder of the **git** user the Mac or Linux server.

> You can think of bare repos on the **git** server account as databases for the repos: they are not the working copies.

To create a test bare repository on the server, sign-in as the **git** user and type

```
cd ~/<user>
git init --bare test-repo.git
```

The above procedure will be repeated for each of the new repositories that you need.

Clone an Existing Repos from Github

In the **git** user account on the server

```
cd ~/<user>
```

then clone each of the repos as a mirror, as follows

```
git clone --mirror https://github.com/<some_github-user>/some-github-repo
```

> If, for simplicity, it is easier to clone from Github using an administrator account (for instance your Github credentials are already setup there), make sure to set the owner to **git** after cloning. This can be done recursively as follows from the administrator's account:
>
> ```
> cd /Users/git
> chown -R git: <user>
> ```

Basically, you need to end up with all bare repos owned by the **git** user.

See here for details about backing Github repositories. You may alternatively opt to download all of your Github data.

List the Available Repos

To list the available repositories for a user, do `ssh git@<host> ls <user>`

Example:

```
ssh git@server ls peter
```

There is no per-user permissions on the repos. If there are repos under different user names, they are available to all users.

Configure the Client Computers

You will need to perform the following configuration for each of the client Mac, Windows or Linux computer that you want to use as a git client. This includes on the Mac server if you intend to use it as a client.

Configuring the host name

On the client computer, in the host file (`c:/Windows/System32/drivers/etc/hosts` on Windows, or `/etc/hosts` on Mac), add the IP address of the Mac server, for instance `102.168.18.5 server`, for ease of use. The name "server" here is referred to as <*host*> in other parts of this document.

Create the SSH keys on the client computer

Create the SSH keys on the client computer. The procedure is identical for **Mac**, **Linux** and **Windows**.

```
ssh-keygen -t rsa -b 4096 -C "myclientinfo"
```

Where *myclientinfo* is a short descriptive note about the client, typically some description of the user name and computer.

Accept the default file name.

Enter no passkey. This is needed in order to not require passwords on every git operation.

This creates the **id_rsa** and **id_rsa.pub** key files under **/Users/<*client_user*>/.ssh**.

Register the public SSH key with the server

Copy the public SSH key file to the Mac server.

On **Mac** or **Linux** client computer

```
ssh-copy-id -i ~/.ssh/id_rsa git@<host>
```

On **Windows**, from the client computer

```
scp c:/Users/<client_user>/.ssh/id_rsa.pub git@<host>:~/.ssh
```

```
then on the Mac server
```

```
cat id_rsa.pub >> ~/.ssh/authorized_keys
```

```
chmod 600 ~/.ssh/authorized_keys
```

Repeat the above procedure for any other client computer.

Remark: the *authorized_keys* file on the git account of the server contain a copy of all the authorized public keys on the Mac server. Each corresponding private key remains on each of the corresponding client computer.

Using a Repo

The clients will access the repo via **ssh.** This is similar to accessing a repo on Github via *https*, but with **ssh** as the protocol instead.

For instance the following will clone a repo from any of the properly registered **ssh** clients:

```
git clone ssh://git@<host>/~/<user>/my-repo
```

The repos are organized by owners, represented by *<user>* in this document. For instance, if the server is named **server** in the host file of the client computer, and if the repositories are stored in the *peter* folder on the git server:

```
git clone ssh://git@server/~/peter/my-repo
```

Example, clone and make a first push

```
git clone ssh://git@server/~/peter/test-repo
# ...modify some files...
git push
```

Back Up the Server

Back up the **/Users/git** folder on the Mac server regularly. This will secure recent copies of the **/Users/git/<user>** folders containing the bare repositories, as well as the **/Users/git/.ssh** folder containing the public ssh keys of the authorized clients.