SSH Setup

Setting up an SSH Key

This next step works with msysGit (Git for Windows) or on any *NIX system.

```
ssh-keygen -t rsa -C "yourname@yourcompany.com"
```

The `-t` flag is the algorithms used to create the key. The `-C` flag is the comment attached to the key. The comment can serve as a reminder of which system you use this with if you are going to generate more than one key pair to partition your Git SSH key from other SSH-authenticated systems and servers.

Other algorithms, like DSA, can also be used for SSH authentication and are compatible with Git, since Git is merely using the operating system's underlying SSH capabilities.

```
ssh-keygen -t dsa -C "yourname@yourcompany.com"
```

Lengthier instructions for SSH key generation can be found at the excellent GitHub page.

The decision whether or not to use a passphrase for your SSH keys can also be found on GitHub.

Sharing the Public portion of the Key

Most Git services use half of the key we just generated for authenticating instead of the typical username and password. In security terms, SSH keys are quite a bit stronger than usernames.

Keep the private half of the key (`id_rsa`) protected. Give away the public half (`id_rsa.pub`) liberally. You could even store it in a directory service if desired.

Authorizing the key on another server

If you are in control of a server on which you'll be storing Git repos, you can authorize your account to automatically sign in. While logged in to the remote server, put the contents of a user's `id_rsa.pub` file on a single line (*absolutely no linebreaks!*) on a file named `~/.ssh/authorized_keys`.

Similarly, key strings are copied-and-pasted to web based repositories like GitHub via the user interface. Copy the contents of `id_rsa.pub` to the clipboard and paste it into the appropriate textbox in the web UI of GitHub.

Testing SSH

If you wish to test if you have passwordless (key authentication) working correctly, just SSH to the server. It will use your `id_rsa` and `id_rsa.pub` files automatically if they live in `~/.ssh/`. You should not see any prompt for a password.

```
ssh SERVERNAME
```

In the case of a gitolite server, it will report your repository permissions before terminating.

```
hello mccm06, the gitolite version here is v1.5.5-68-g3cf2970
the gitolite config gives you the following access:
     R   W  gitolite-admin
    @R  @W  testing
    @R   W  testinglessaccess
Connection to mybigserver closed.
```