CONTENTS

1. A NOTE ON ENCRIPTION

1.1 Symmetric Key Encryption

There is no confusion about encryption, it is a method of coding the text so that people who has unauthorized access to the text will not be able to comprehend the text (hopefully). Generally a secret code (string) is used to encrypt the text. A simple example is an 'exclusive or' the text with the secret string. As we know we can just have an 'exclusive or' again on the converted string with the secret code to get back the original text. This is s very simple method of encryption and one can have more complex ones. It is to be noted that, if an encrypted message/text is send, the receiver should know the secrete code to decrypt the text. So there must be a way for the receiver to get the secret code. Anybody who have access to the secret code (string) can decode the encrypted text. People may even try a brute force method to arrive at the secrete code. The risk of brute force attack can be reduced by frequently changing the secret code.

1.2 Public Key/Private key Encryption

Since long people thought about a way of encryption method, with two keys, one for encrypting and another for decrypting, so that once encrypted with a key, you need the second key to decrypt it. In this scheme, one key can be distributed to all senders as public key and the second key, the private key, kept as secret by the receiver and it is not known to others. The present method to arrive at these keys pair is based on difficulty in arriving the integer factors of large integer numbers. Under this scheme there is no need for the sender to know the private key and knowledge of the public key will not help in decrypting the coded message. One disadvantage as of now is,it is more CPU intensive than the first method and generally used to authenticate the person and then arrive at a secrete string and communicate further. This is the scheme used in 'ssh'.

1.3 CA Certificates

Since public-keys are distributed freely, it may be required to verify that it is really from the original supplier. CA certification is a method to that end, and known public keys are signed by well known agencies to build a confidence on the public keys.

1.4 Hashing

Another concept related to this subject is hashing a file, though it is not encrypting. Hashing is a method to uniquely arrive at a string/code from a file. Though one can not get back a file from its hash, it can be used to check the integrity of a file. Different files always make different hash string and creating new hash string and comparing with a previous hash string can be used to check the integrity of a file.

2. ABOUT CONNECTING THROUGH SSH/SCP.

To ssh/scp in batch mode (ie without password prompt), following steps are required.(This is only possible if the remote is configured to support this authentication.Also it is just one method and other methods also exist.)

1. .ssh directory in $(HOME). It should not have write permission for others. ie it must be chmod 755

ie;

cd ~
mkdir .ssh
chmod 755 .ssh

Now under .ssh, run

ssh-keygen -t rsa

It will prompt for passphrase, which can be empty, otherwise it will be prompted for at use also. Passphrase is a mechanism to encrypt the private key, so that even if somebody got access to it, the passphrase is needed to use it. The problem with it is that, it will create problem in automated mode, so may be avoided in such cases.

It will create two files

1. id_rsa which is private key and should not be shared with others, other than extra ordinary situations. Only the owner should have permissions on this.
   ie it must be ***'chmod 600 id_rsa'***.
2. id_rsa.pub the public key which can be shared with others but must have only read permissions to others.
   ie ***'chmod 644 id_rsa.pub'***

The third file of importance is 'authorized_keys' which also must only have read permissions for others. ie ***'chmod 644 authorized_keys'***. This file is a collection of public keys of others who are allowed to login to the user/machine. One can certainly add own public key to this file so it is possible to ssh to own machine/account.

ie ***'cat id_rsa.pub >> authorized_keys'.***

One can add public keys of others/machines like this to allow logins from other users/machines to the account in the machine. Always remember about the permissions on the files.

Forth file is known_hosts. Initially one can just touch it. It must also be ***' touch known_hosts;chmod 644 known_hosts'.*** 'known_hosts' will get build up on making ssh to other user/machine.

If one need access to an account in a machine he has to append his public key to the authorized_keys of that account and also need to copy the public key of that account to own authorized keys.

Permissions of these files should be as mentioned.

3. ABOUT USAGE OF GIT

3.1 General Description

Git is a version control system which can have remote locations which can be used by others to get copy/clone of the repository.

If you are just starting your first git repository you may need to issue the following setup.

git config --global user.name "Your user name"
git config --global user.emal "your email"

If you need to create a git repository of a project/files, the following steps are the basic commands.

1. Either you create or new directory or an existing one with the required project related files.
2. Go to the directory and issue *git init*
3. git got a concept of 'branches' to maintain different versions or stages of a project. The default branch is 'master'. There are ways to create new branches, delete branches, merge branches. With *'git init'* one created by default the 'master' branch and by default moved to that branch.
   The next step is adding files and folders to the repository. *'git add <file>/<folder> <file> ....'* will add files/folders to the the current branch, ie to begin with 'master'. If everything needs to be added one can use; *'git add .'*

4. Adding alone is not sufficient, one need to commit it. ie , *'git commit -a -m "<info string>"*. '-a' flag is to commit all changes made to the files in the repository. If you don't do this the changes will not be reflected. If one has created new files/folders it will not be reflected in the repository unless it is explicitly added using *' git add ...'* and the commit it using *'git commit....'*

The reverse of add is
'git rm <file> .... <file>'

5. One can create new branch using *' git branch <branch name>'* and can be moved to desired branch using *'git checkout <branch name>'*.

Currently active branch and other branches can be seen by;
'git branch'

6. One can merge a branch to current branch using *'git merge <branch name>'* . This will work if there are no conflicts, for examples if branches modified same file and committed then a conflict will arise and it has to be resolved manually. The conflicts will be marked in the file and can be modified using an editor. Once conflicts are resolved manually, it can be committed

'git commit -a -m "<commit message>"'

However two different branches modified different files, it is not a problem. Merging both to the master(or to any other branch) will include the modifications from both.

eg.

git checkout master
git merge branch1
git merge branch2

Now master will have both modifications in branch1 and branch2 (provided they modify different files).

Similar to adding files is the case of removing files. However if a common file is removed from one branch and another branch keeps it (and may be modified also) can lead to conflicts on merging and has to be modified manually and then committed.

7. One can remove a branch by issuing;

'git branch -d < branch name> .....<branch name>'

This will work only if branches to be deleted are merged with current branch. In case one needs to delete a branch unconditionally, issue;
'git branch -D <branch-name>'

8. git cloning

Cloning is making exact copy of a repository from a remote or another local repository. This is a way of collaborating work or can be even a branching out. It can be as issued as follows

'git clone <remote> <local_folder>'
or
'git clone <path to git repo> <local_folder>

Now the new user can modify the repo and commit it. Any stage original repo can get the modifications of the new repository by issuing;

'git pull <new_repo>'
The modifications of the new repo will be fetched and merged with currently active branch of the original. However it there are conflicts manual modifications may be required.
Another option just to check the modification of the new by the old is to fetch the new. ie;
'git fetch <new>'
and one can check the modifications by issuing
'git log -p HEAD..FETCH_HEAD'

9. There is a concept of 'remote repository' which can be in the same machine,

another machine in local network or a cloud repository. For transaction between local and remote one can use 'ssh' or 'https', which git decides using the url. One can have more than one remote repository. If the need is to create a 'remote' in local network (ie same machine or another machine in the local network) one may follow the steps below.

a. Create a new account for git admin in that machine, it may an existing account also.
b. login in to that machine and create a new folder with a meaningful name connecting the repository.
c. go to the new folder and issue;
   'git init --bare'
d. One may create the 'ssh' setting as explained earlier so that batch ssh possible.
e. come back to the original machine and move to the git folder of interest. Make sure that one can ssh to the 'remote'.
   Now issue;
   git remote add <remote_name> <remote_url>
   eg. ***git remote add origin git@192.168.1.20:myrepo***
   or
   git remote add gitorigin git@machine.name:myrepo
   again issue;
   git push -u <remote_name> master

Cloud machines eg github.com, sourceforge.com etc provide web interface to create a new account, set up ssh/https etc and initialize a new repository. Also they explain commands to be issued locally which are very much similar as above.
Any time the local version is modified, remote can be synced with local with push.

Others who need a copy of the repository can clone it.

ie ; ***git clone <remote_url>***

if the remote is updated and local needs to sync with remote

git pull <remote_url>

One can check the remotes by issuing;

'git remote -v show'

Though it is possible for different users (if allowed) to push their branches to a remote, it may also lead to conflicts. If such conflicts occure it may need to be resolved before pushing. This can be done by issuing a pull request, and care should be taken in such cases.

These are basic set of commands and there are more commands and options.

3.2 A Case Study

As a case study an old project 'kgwrite' was used. Kgwrite is a word processor of the type nroff/troff or Tex. One inserts commands in the text to instruct the actions and the processed output is generated in Postscript (or any supported image format) which can be converted to PDF using available tools (like ps2pdf). There are many word processors available now and the relevance of 'kgwrite' is arguable. However it still has some relevance in the generation of automated documents. Source of 'kgwrite' was under a folder 'kgwrite', and as the first step all unnecessary files are removed and and made a ***make clean***. Then the following commands are issued;

git init
git add .
git commit -m "First commit"

As explained earlier, created and set the $HOME/.ssh directory and logged into the github.com account and created a new repository by name 'kgwrite'. Also added the ssh public key to github.com so that ssh to github.com was possible. Then the following commands are issued;

git remote add sshgit git@github.com:tcbabu/kgwrite.git
git push -u sshgit master

Now I have created a remote repository of my project 'kgwrite' in github.com. And anybody can clone the project (since it is a public repository, otherwise other permissions are needed).I also thought of creating a remote in the local network. For this I used another account in the same machine by id 'abu', the present being 'babu'. I could have as well used another machine in the local network. I logged in as 'abu' and setup the 'ssh' as explained earlier. I also appended public key of 'babu' to 'authorized_keys' of 'abu', so that 'babu' can login to 'abu' without a password prompt. Also appended public key of 'abu' to 'authorized_keys' of 'babu', so that 'abu' also can login to 'babu' without a password prompt. Now login as 'abu' and issued the following command to create a remote in local network.

mkdir kgwrite;cd kgwrite
git init --bare

Now logged in as 'babu' and issued following commands;

cd kgwrite
git remote add local abu@hplaptop:kgwrite
git push -u local master

Now two remotes are created one in the local network and other in github.com. I checked my remotes by issuing ***git remote -v show***. Looking at the output I could verify my remotes.

Now logged in as another user 'thara' and issued the following command;

git clone abu@hplaptop:kgwrite kgwrite

User 'thara' had created a setup, so that 'thara' can login to 'abu' without password. By issuing the above command 'thara' had created a copy of 'kgwrite' from

'abu'. And by default 'thara's remote is 'abu'. In case it is needed to use github as the remote 'thara' could add that also as another remote.

By this example there are two remoted and to users. Ofcourse 'thara' could have cloned from 'babu' also. They can also make pull requests from each other, or they can always deal though a remote, by both using pull/push requests from a common remote.

4. ABOUT DOCKER

In a sense docker is a packing method like tar, rpm ,deb etc.
(to be written)