BioHPC Git Training Demo Script

First, ensure that git is installed on your machine, and you have configured an ssh key. See the main slides for instructions.

To follow this demo script open a terminal on Linux or Mac, or a 'Git Bash' session on Windows.

Lines that begin with a $ sign and are in a `console font` should be typed in to the terminal. Don't type the $ - it represents the prompt that you see in the terminal window. Lines that begin with a # are comments. Other lines are output that you should expect to see if things work correctly.

1 – Create a Project on the BioHPC git service

- Login to the BioHPC git service at https://git.biohpc.swmed.edu using your BioHPC account.
- Click the green 'New Project' button, give the project a name 'demo1', choose the privacy level you want and click 'Create Project'.
- You'll be taken to a screen that shows the first steps to create a git repository and start work in this project.

2 – Initialize a git repository on your machine, add a README file, commit and push

```
# Make a directory for our new git repository and change into it
$ mkdir demo1
$ cd demo1

# Initialize the git repository
$ git init
Initialized empty Git repository in c:/Users/dtrudg/demo1/.git/


# List the content of the directory – notice the .git folder which is hidden
$ ls -lah
total 8.0k
drwxr-xr-x    1 dtrudg   Administ        0 Mar 17 18:44 .
drwxr-xr-x   48 dtrudg   Administ      12k Mar 17 18:44 ..
drwxr-xr-x    1 dtrudg   Administ     4.0k Mar 17 18:44 .git

# .git is where git stores configuration and the version information
# don't delete the .git directory or mess with the contents
$ ls -lah .git
total 5.5k
drwxr-xr-x    1 dtrudg   Administ     4.0k Mar 17 18:44 .
drwxr-xr-x    1 dtrudg   Administ        0 Mar 17 18:44 ..
-rw-r--r--    1 dtrudg   Administ       23 Mar 17 18:44 HEAD
-rw-r--r--    1 dtrudg   Administ      157 Mar 17 18:44 config
-rw-r--r--    1 dtrudg   Administ       73 Mar 17 18:44 description
drwxr-xr-x   11 dtrudg   Administ     4.0k Mar 17 18:44 hooks
drwxr-xr-x    3 dtrudg   Administ        0 Mar 17 18:44 info
```

```
drwxr-xr-x    4 dtrudg   Administ        0 Mar 17 18:44 objects
drwxr-xr-x    4 dtrudg   Administ        0 Mar 17 18:44 refs
```

```
# Let’s create a README file
# Use the vi editor – press i to insert text, ESC to stop writing
# and :wq to save and quit
$ vi README
```

```
# Now show the repository status. Our README isn’t going to be commited yet
$ git status
```

```
On branch master

Initial commit

Untracked files:

  (use "git add <file>..." to include in what will be committed)

      README

nothing added to commit but untracked files present (use "git add" to track)
```

```
# Now let’s mark the readme file to be commit it.
# We add it to the index – which is where a commit is assembled
$ git add README
```

```
# Now we should see it’s going to be committed if we check git status
$ git status
```

```
On branch master

Initial commit

Changes to be committed:

  (use "git rm --cached <file>..." to unstage)

        new file:   README
```

```
# Okay, let’s make the commit. Use –m to provide the commit message
# The commit will have a hash identifier, that uniquely identifies it
# Here the hash is 7c0188a
$ git commit -m "First Commit"
```

```
[master (root-commit) 7c0188a] First Commit

1 file changed, 13 insertions(+)

      create mode 100644 README
```

```
# Now we want to push our repository to the BioHPC git server.
# In git a remote is a location that we can push to
# Let’s add our git project as a remote, called origin
$ git remote add origin git@git.biohpc.swmed.edu:david.trudgian/demo1.git
```

```
# We can see the remotes setup for our repository with the git remote command
$ git remote
origin
```

```
# Now we need to push our code to the remote repository. The first time we
# do this we tell git where to push it. We use the –u option so that this is
# set as a default, and we can just say git push in future.
$ git push -u origin master
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 324 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To git@git.biohpc.swmed.edu:david.trudgian/demo1.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.

# Now have a look the web interface at https://git.biohpc.swmed.edu to see
# your repository online
```

3 – Add some additional files and commit changes

```
# Let's write a simple script, and add it to our repository

$ vi script.sh
$ git add script.sh
$ git commit -m 'Added new script'

# With more than one commit let's look at the log, which shows a history of
# our code
$ git log

# Now we'll change our script and commit the changes. Notice we have to git add
# the file that we changed in order to commit the changes.
$ vi script.sh
$ git status
$ git add script.sh
$ git status

# And add some notes in another file
$ vi notes.txt
$ git add notes.txt
$ git status
$ git commit -m 'Added notes'

# We can add multiple files and directories at once
$ mkdir dir1
$ cd dir1
$ date > file1
$ date > file2
$ cd ..
$ mkdir dir2
$ hostname > filea
$ hostname > fileb
$ cd ..
$ git status
$ git add dir1 dir2
$ git commit –m 'Added directories'

# We can remove a file
$ git rm dir1/file2
$ git commit 'Removed dir1 file2'

# We can examine and undo our changes to a file
$ vi script.sh
$ git status
$ git diff

# We checkout the latest committed version (HEAD) of script.sh undoing our change
$ git checkout HEAD
$ git status

# Now let's check the commit log, and push all of our changes to the server
$ git log
$ git push
```

4 – Create a branch for a new feature, to work on it independently from the master branch

You might want to work on a new experimental feature in your code, without disrupting the master branch – which needs to stay stable, and be modified for bugfixes. We can work on our experimental feature safely by creating a branch. Once we're happy with the new feature it can be merged into the master branch.

```
$ git branch

# The git branch command without arguments shows us our local branches
# The * indicates our current branch. The –r option will show branches available
# on configured remote repositories

$ git branch
$ git branch -r

# Let's create a branch called newstuff to add our new features
$ git branch newstuff

# Now we should see it in the branch list – but we're not working on it yet
$ git branch

# We need to checkout the branch to switch to it
$ git checkout newstuff

# Let's modify our script in this branch, adding our experimental feature
$ vi script.sh

# Now commit the changes
$ git add script.sh
$ git commit –m 'New feature work'
```

```
# We can look at the log in a graph format, and include all of our branches
$ git log --graph –all --decorate

# Finally let’s push our branch to the BioHPC remote repository
$ git push –u origin newstuff
```

5 – Go back to the master branch to make a bug fix

Lots of people use our stable master, so we need to fix a bug on it.

```
# We checkout master. Our work in the newstuff branch dissappears
# from the visible files, but still exists in the git repository!
$ git checkout master
$ git branch

# Make our bug fix and commit it
$ vi script.sh

# Show the difference between our files and the last commit
$ git diff

# Now commit the changes
$ git add script.sh

$ git commit –m 'bug fix'

# Now if we look at the log graph we can see our branches have diverged
# We have two branches with differents sets of changes
$ git log --graph --all --decorate

# Push our bug fix to the remote
$ git push
```

6 – Merge our work from the ‘newstuff’ branch into master

We can continue to switch between branches and work on our new feature in the newstuff branch. When we’re done and ready to bring the feature into our master branch we need to merge the changes from newstuff into master. From the master branch we do:

```
$ git merge newstuff
```

If there are no conflicts this does a fast-forward commit – no work for us! If there are conflicts….

```
$ git status
$ vi script.sh

# In the editor the conflicting portions are wrapped in special lines, e.g.

<<<<<<< HEAD

echo "This is a bad script."

=======
echo "This is a great script."
>>>>>>> newstuff
```

```
# The top section is from HEAD – our current branch. The bottom section is the
# conflicting text from the newstuff branch.
# We need to edit the file, to leave only the code that we want to keep
```

```
# Now we use git add to mark the conflict resolved
$ git add script.sh
```

```
# Check that all conflicts are resolved
$ git status
```

```
# Now we commit the merge. Don't specify a message – git automatically
# generates a detailed message for us.
$ git commit
```

```
# Now the tree shows that we've merged newstuff into master
$ git log --graph --all --decorate
```

```
# Send it all to the remote repository
$ git push
```

7 – Cloning an existing repository

```
# Go back to our home directory
$ cd
```

```
# We'll clone a repository that already exists on the server
$ git clone git@git.biohpc.swmed.edu:hatawang/biohpc-public.git
```

```
# Let's go into it and take a look around
$ cd biohpc-public
$ git status
$ git log
```

8 – Pulling Changes From Other Developers

Note – you won't have the repositories for this section on your machine, but it does show the commands to use.

# Let's go into another repository. I made some changes from another machine, and pushed them.
# Now we need to get the changes to our local repository on this machine

```
$ cd ~/demo_pull
$ git status
$ git log
```

```
# We can get the changes and merge them into our local version in one step
$ git pull
```

```
# Sometimes we might want to check changes before incoporating them in our code
# We fetch them, and merge them separately – which lets us look at them
# before we decide to merge them with our local version:

$ cd ~/demo_fetch
$ git fetch
$ git diff origin/master
$ git merge origin/master
```