**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, add files, commit and push

```
I am John.

# At BioHPC cluster, make a directory for our new git repository and change into it
$ mkdir demo1

# copy some files to demo1

$ cp function1 function2 script.sh demo1
$ cd demo1

$ ls –laht

total 16K
drwxr-xr-x 2 s167891 biohpc_admin 4.0K Apr 10 16:53 .
-rw-r--r-- 1 s167891 biohpc_admin   87 Apr 10 16:53 function1
-rw-r--r-- 1 s167891 biohpc_admin    0 Apr 10 16:53 function2
-rw-r--r-- 1 s167891 biohpc_admin   36 Apr 10 16:53 script.sh
drwxr-xr-x 4 s167891 biohpc_admin 4.0K Apr 10 16:53 ..

# Initialize the git repository
$ git init
Initialized empty Git repository in /project/biohpcadmin/s167891/Gittraining/demo1/.git/


# List the content of the directory – notice the .git folder which is hidden
$ ls –lah
total 24K
drwxr-xr-x 8 s167891 biohpc_admin 4.0K Apr 10 16:53 .git
```

```
drwxr-xr-x 3 s167891 biohpc_admin 4.0K Apr 10 16:53 .
-rw-r--r-- 1 s167891 biohpc_admin   74 Apr 10 16:53 script.sh
-rw-r--r-- 1 s167891 biohpc_admin   87 Apr 10 16:53 function1
-rw-r--r-- 1 s167891 biohpc_admin    0 Apr 10 16:53 function2
drwxr-xr-x 4 s167891 biohpc_admin 4.0K Apr 10 16:53 ..
```

```
# .git is where git stores configuration and the version information
# don't delete the .git directory or mess with the contents
$ ls .git -a
.  ..  branches  config  description  HEAD  hooks  info  objects  refs
```

```
# Now show the repository status. Our file are not commited yet
$ git status
```

```
On branch master
Initial commit
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        function1
        function2
        script.sh
nothing added to commit but untracked files present (use "git add" to track)
```

```
# Now let's mark the files to be committed.
# We add all the files to stage – which is where a commit is assembled
$ git add .
```

```
# 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:   function1
        new file:   function2
        new file:   script.sh
```

```
# 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 b3f1c32
$ git commit –m 'first commit'
```

```
[master (root-commit) b3f1c32] first commit
 3 files changed, 15 insertions(+)
 create mode 100644 function1
 create mode 100644 function2
 create mode 100644 script.sh
```

```
$ git log
```

```
commit b3f1c323617bf874bbfb326b014f5f16152da05c
Author: Wei Guo <wei1.guo@utsouthwestern.edu>
Date:   Mon Apr 10 16:57:57 2017 -0500
```

```
    first commit
```

```
# 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:wei1.guo/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: 5, done.
Delta compression using up to 40 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (5/5), 406 bytes | 0 bytes/s, done.
Total 5 (delta 0), reused 0 (delta 0)
To git@git.biohpc.swmed.edu:wei1.guo/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
# the remote repository online
```

## 3 – Add some additional changes and files and commit changes

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

$ vi function3
$ git add function3
$ git commit –m 'Added function 3'

# 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

# We can examine and undo our changes to a file

$ git diff
$ git add script.sh
$ git status

On branch master
```

```
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   script.sh
```

```
# And add some notes in another file
$ vi notes.txt
$ git add notes.txt
$ git status
$ git commit –m ‘Added notes and scripts’

# We can remove a file
$ git rm notes.txtg\
$ git commit -m 'Removed notes.txt'

# We can examine and undo our changes to a file

$ vi script.sh

$ git diff

$ git add script.sh

# To unstage the change

$ git reset script.sh

# To undo our change
$ git checkout script.sh


$ git status

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

## 4 – Mary creates a branch for new algorithm, 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.

```
Now I am Marry

$ cd IamMary

$ ls –a

$ git status

$ git clone git@git.biohpc.swmed.edu:wei1.guo/demo1.git
```

**$ cd demo1**

**$ git status**

# 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**

**$ git branch -a**

# 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**

```
Counting objects: 3, done.
Delta compression using up to 40 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 345 bytes | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote:
remote: To create a merge request for newstuff, visit:
remote:
https://git.biohpc.swmed.edu/wei1.guo/demo1/merge_requests/new?merge_request%5Bsource_branch%5D=ne
wstuff
remote:
To git@git.biohpc.swmed.edu:wei1.guo/demo1.git
 * [new branch]      newstuff -> newstuff
Branch newstuff set up to track remote branch newstuff from origin.
```

## 5 – John fixes a bug in the master branch

**I am John now**

**$ cd demo1**

# 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 – John fetches newstuff from remote and merge 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:

```
I am John

$ cd demo1

# I will fetch Mary's newstuff branch to my local database for my review. This is
temporary.

$ git fetch origin newstuff

$ git diff origin/newstuff

# Now I want to use Mary's code in the master so that everyone can use the
algorighm.
```

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

```
$ git merge origin/newstuff

$ cat script.sh

$ git status
$ vi script.sh

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

<<<<<<< HEAD
```

```
date +%s
=======
date -u
>>>>>>> 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:david.trudgian/astrocyte_example_chipseq.git

# Let's go into it and take a look around
$ cd astrocyte_example_chipseq
$ 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 ~/demo1
$ 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 ~/demo1
$ git fetch
$ git diff origin/master
$ git merge origin/master
```