Git Setup

So you are a Sakai Trunk Committer and Need to Use GIT:

1. Make sure you have git setup and have a github account and have your name and email properly set as git global variables on your workstation.
2. Fork the central Sakai repository to your own github account
    a. We will use the following references
        i. local = local git clone on your workstation, this where everyone typically does their work
        ii. origin = this is your local fork of the sakai project on github, you clone this repo on your local workstation
        iii. upstream = this is main sakai project github project everyone forks this project into their github account
3. Clone your fork onto your computer
4. Sync sakai master with your local forks master to receive updates
    a. Never work on your local master, generally this branch should always be the same as what is in sakai's master and if you make commits here this complicates things. If you mistakenly commit work to your local master, it's easy to fix

```
master> git branch SAK-XYZ # make a copy of what we've done so far in a branch

master> git reset --hard upstream/master # match current branch to upstream

master> git checkout SAK-XYZ # resume work on feature branch
```

    b. Sync sakai master with your forks master

```
master> git remote add upstream https://github.com/sakaiproject/sakai
```
(you will do this only once)

```
master> git pull upstream master
```
(you will update your local fork frequently)

```
master> git push origin master
```
(after you update your local fork don't forget to push to your github fork)

5. Use your own copy of git to make your changes and test, etc. You can commit and push as to your own repo as much as you like.
    a. Make sure you always work on a local branch, We recommend using the name of the JIRA as the branch name.

```
master> git checkout -b SAK-12345
SAK-12345>
(make your changes then commit them, if you make multiple commits its best to squash them into one logical commit)

SAK-12345> git push origin SAK-12345
(this creates a new branch called SAK-12345 in your fork on github that contains everything in your local branch SAK-12345, notice they have the same name this is important)
```

6. When you are ready to "commit to trunk" create a pull request from your repo against the sakai master branch in the sakaiproject repository.
    a. If the changes are code you would have committed to the old SVN trunk (i.e. just normal development) - merge your own pull request in to Sakai's master branch.
    b. If the changes you are making are worthy of further review or touching an area of Sakai that you do not generally work in - have someone else review and merge your pull request.
    c. If after review changes need to be made all you need to do is make the changes and commit in the SAME branch you issued the pull request (PR) from and then push those changes to your fork and it will automatically update the PR that was made to sakai's master.
        i. Use the rebase interactive mode to find all commits you've made since you branched off your master

```
SAK-12345> git rebase -i master
```

```
pick b76d157 commit2
pick a931ac7 commit1

# Rebase df23917..a931ac7 onto df23917
#
# Commands:
#  p, pick = use commit
#  r, reword = use commit, but edit the commit message
#  e, edit = use commit, but stop for amending
#  s, squash = use commit, but meld into previous commit
#  f, fixup = like "squash", but discard this commit's log message
#
# If you remove a line here THAT COMMIT WILL BE LOST.
# However, if you remove everything, the rebase will be aborted.
#
```

ii. Squash commit1 into commit2 by changing the prefix on the appropriate line:

```
pick b76d157 commit2
s a931ac7 commit1
```

iii. Change the commit messages if necessary in the following editor screen
iv. Push the changes back to the branch in your repo

```
SAK-12345> git push --force origin SAK-12345
```

v. Now your changes are all squashed into one commit, your repo branch is updated with your local code, and your PR is automatically updated without closing it and opening a new one!

- In general we trust Sakai committers to know the difference between (6.a) and (6.b) - as time progresses, we will get better at this. We will be giving trunk committers broad access to the sakai master repo and will rely on the fact that we know when to merge our own requests and when to request someone else to merge the requests.
- The super great news is that now folks who do not have trunk commit can issue pull requests to the sakai master repo and we can get those changes merged in a more natural way.
- We are continuing to use JIRA to do issue tracking - we will not be using the github issue tracking.
- Commit messages should a prefix of JIRA issue(s) that this commit applies to. As an example:

```
SAK-123 Improved the performance of site cache

The TTL on the cache of 30 seconds was too low for production deployments, upped to 1 hour....
```

How do I issue a Pull Request for the 11.x branch exclusively, or another branch aside from master?

If you find yourself working on a bug that is only intended for the 11.x branch (i.e., it's already fixed in master or doesn't affect master), you'll need to issue a Pull Request against the 11.x branch exclusively. To do so, you'll need to perform the following steps:

1. On your local clone, checkout the master branch and ensure it is up to date with upstream

```
master> git checkout master

master> git pull upstream master && git push origin master
```

2. Depending on some settings on your local system, you may also have to fetch upstream to get a reference to the upstream 11.x branch (or any other upstream branches you don't have a reference to locally)

```
master> git fetch upstream

remote: Counting objects: 24, done.
remote: Compressing objects: 100% (22/22), done.
remote: Total 24 (delta 0), reused 24 (delta 0), pack-reused 0
Unpacking objects: 100% (24/24), done.
From github.com:sakaiproject/sakai
 * [new branch]      11.x       -> upstream/11.x
```

3. Then, you checkout (create) the branch locally and set it up to push to origin, so you can stay in sync with the 11.x branch like you do with master

```
master> git checkout 11.x

Branch 11.x set up to track remote branch 11.x from upstream.
Switched to a new branch '11.x'

master> git push -u origin 11.x
```

```
Counting objects: 1, done.
Writing objects: 100% (1/1), 290 bytes | 0 bytes/s, done.
Total 1 (delta 0), reused 0 (delta 0)
To git@github.com:<yourGitHubAccount>/sakai.git
 * [new branch]      11.x -> 11.x
Branch 11.x set up to track remote branch 11.x from origin.
```

4. To update the 11.x branch from upstream, perform the following (like you would with master, just replacing the master branch name with the 11.x branch name, or any other branch name from upstream)

   ```
   master> git checkout 11.x
   ```

   ```
   master> git pull upstream 11.x && git push origin 11.x
   ```

5. To create a local branch for development against only 11.x, you create a new branch as you would normally, but instead of being based on master (which is default), you want to base it on the 11.x branch:

   ```
   master> git checkout -b SAK-12345 11.x
   ```

6. Once you have a local feature branch based on 11.x and you've done some development and committed your work, you'll want to push it to your origin so you can issue a Pull Request against the upstream 11.x branch

   ```
   master> git push -u origin SAK-12345
   ```

7. Your 11.x specific feature is now ready for you to issue the Pull Request!

Reviewing Pull Requests

If you are reviewing someone else's pull request and would like to check out a copy of the Sakai repo that includes the pull request, use the following command:

```
master> git fetch upstream pull/326/head:SAK-29127
```

```
master> git checkout SAK-29127
```

This will pull in the code for pull request #326 and create a branch in your repository named SAK-29127 to allow you to clone the repo as it appears with the pull request.

Typical git workflow would go like this (assuming everything is setup as explained above):

- go to master and get the most recent updates
  - `git checkout master`
  - `git pull upstream master && git push origin master`
- create a new feature branch to work in (remember never work in master)
  - `git checkout -b SAK-xxxxx`
- switch to the feature branch if needed
  - `git checkout SAK-xxxxx`
- work on your local feature, creating as many commits as needed
  - `git commit -a -m "SAK-xxxxx: Issue title"`
- once you are happy with your changes and believe it is ready for inclusion in sakai's project (upstream) squash your commits into 1 commit
  - `see 6c above`
- push your changes to origin in a branch typically with the same name
  - `git push origin SAK-xxxxx`
- next issue a pull request (PR) from your feature branch in origin to upstream master (please include a link to the JIRA ticket in your PR description)
  - `on github.com`
  - `or from the command line with the hub tool: hub pull-request`

- if review is needed this is the time for others to review your changes
    - if further changes are required you switch to your local branch make the changes, commit, squash the commits, and then push them to the same branch in origin that the PR was issued from and PR will be updated automatically.
    - repeat this step until changes are satisfactory
- if review is complete then the PR is merged
- optional branch can be deleted in origin when PR is merged

Security Issue?

If it's a issue related to security follow this workflow: Security issues - Jira & Github

Tips & Tricks

- How to Undo Almost Anything with Git

- If your local branch (master/11.x) gets out of sync with upstream master and just doesn't seem to work right anymore run through this (You may want to save this branch first)
- http://stackoverflow.com/a/1628334/3708872

```
git checkout master

git fetch upstream

# For master (use 11.x for 11.x)
git reset --hard upstream/master
```