**Basics**

**>git init**  
**> git init**  
*Initializes a new git repository*.  
> git init --bare  
*Initializes a bare git repository. A bare repository has no working directory.*

**>git status***Shows the status of the repository. Staged files show in green. Unstaged files show in red.*

**>git commit**  
**> git commit***Commits staged files to the repository. Any unstaged files in the working directory will not be committed.*> git commit -am ‘<comment>’  
*Adds tracked but unstaged files to the staging area, and commits the staged files. This will not add new (untracked) files or delete existing (tracked) files.*

**>git add**  
**> git add .***Adds new (untracked) files and existing (tracked) files in the working directory to the staging area.***> git add -A .***The same as above, but it also will remove files that are tracked, but are no longer in the working directory.*

**>git log***Shows a log of commits.*

**>git show**  
**> git show <hash>***Shows information about the object specified.*

**>git reset**  
**> git reset***Removes all staged files from the staging area.***> git reset --hard***Removes all staged files from the staging area, and restores any unstaged files back to their state as of the last commit.*

**>git clean**  
**> git clean –dnx***Does a dry run of cleaning any untracked files from your working directory.***> git clean –dfx***Cleans any untracked files from your working directory.*

**Working in a Repository**

**>git branch**  
**> git branch***Shows branches in the repository.***> git branch –r***Shows remote branches.***> git branch –d <branch name>***Deletes the branch specified. If there are files that have not been merged to the current branch, this will not succeed. You will have to use –D instead.***> git branch –m <old name> <new name>***Renames a branch.***> git branch –set-upstream <remote>/<branch>***Sets up a new branch (local branch) to track to a remote branch.*

**>git checkout**  
**> git checkout <branch>***Switches the working directory to the specified branch. Unstaged files will remain unstaged.***> git checkout -b <branch>***Creates a new branch and performs a checkout to that branch.***> git checkout --track <remote>/<branch>***Tracks a new branch to the specified remote.*

**>git difftool**  
**> git difftool***Performs a diff on all changes.***> git difftool --stat***Shows a one-line diff on all changes.***> git difftool <branch>..<other branch>***Compares the two specified branches*

**>git merge  
> git merge <branch>***Merges the branch specified into the current branch*

**>git mergetool***Merges all changes.using a tool*

**>git rebase***Like merge, but it applies changes one commit at a time.*

**>git stash**  
**> git stash***Pushes unstaged changes into a special place.***> git stash pop***Gets the unstaged changes out of that special place.*

**Working with Remotes**

**>git clone**  
**> git clone <url> <repo name>***Creates a local repository of the remote specified. If repo name is not specified, it will have the same name as the remote project. By default, the remote will be named “origin.”***> git clone --bare <url> <repo name>***Save as above, except it creates the repository as a bare repository.*

**>git remote**  
**> git remote add <branch> <name> <url>***Adds a remote repository for the specified branch with the specified name.***> git remote add –t <branch> <name> <url>***Same as above, but also sets up the remote to track the specified branch.***> git remote show <remote>***Shows all remote branches, and which local branches track to the remotes.*

**>git fetch  
> git fetch***Downloads objects from the default remote. Does not automatically merge any branches.* **> git fetch <remote>**  
*Same as fetch, but for the specified remote.*

**>git pull**  
**> git pull***Fetches and merges objects from the default remote branch (tracked to the current branch).***> git pull <remote> <branch>***Same as above, but for the specified remote and branch.*

**>git push**  
**> git push***Packages all changes to the current branch and sends it to its tracked remote branch.***> git push <remote> <branch>***Same as above, except to the specified remote and branch.***> git push <remote> :<branch>***Deletes the branch on the specified remote.*

**Create a Topic Branch**

* Branch from Master
* If working alone:
  + Branch name is Feature #
* Else:
  + Branch name is Screen #
  + Create a topic branch for each feature associated with the screen
    - These are created as a topic branch off of the screen branch.

**Bugs**

* Branch from test
* Branch name is Bug #
* Merge changes to test
* When the changes are resolved (after verified in test), merge to master
  + You may need to create a temporary integration branch from master and integrate to that first

**Working on Topic Branches**



* Commit frequently
* Push as often as possible
* Merge from source branch as often as possible

**Complete a Topic Branch**

* Merge from master
* Push branch to server
* Initiate feature review
  + Analytics
  + Pull request
  + Code review
  + Second pair of eyes (someone else tests)
  + Automated test review
* Merge to master
* Delete topic branch

**QA Push**

* Create integration branch from test
* Merge from master to the integration branch
* Run tests
* Merge from integration branch to QA
* Delete integration branch
* Tag commit with incremental build number

**Demo Push**

* Create integration branch from demo
* Merge from test to the integration branch
* Run tests
* Merge from integration branch to demo
* Delete integration branch
* Tag commit with incremental build number

**Dev Branch**

* This is an early integration branch
* Should be maintained automatically
* No need to interact directly with dev