Rob Menning
Procedure for working with Git environment branching strategy in enterprise setting.

STEP	LOCAL GIT	LOCAL FILES	REMOTE ('origin') GIT	WORKFLOW	ENVIRONMENT
0. Clone repository to local.	#if have not already cloned cd to organization directory $git clone <URL>	Working copy files match origin master.	Retrieve <URL> of repository using 'Clone or download', 'code' button.		Prod environment matches master.
1. Create a new branch called feature1.	$cd <repo> $git checkout master $git pull --rebase origin master $git checkout -b feature1	Working copy pointing at feature1.			
2. Make changes for feature in working copy files. Commit to feature1 branch.	$git checkout feature1 $git diff $git status $git add [<file> | .] $git commit –m <comment>	feature1 is in modified then staged, then committed states. Working copy pointing at feature1.		Modify local files to implement feature.	
3. Pull from origin master.	$git checkout master $git pull --rebase origin master	Local master equals origin master. Working copy pointing at master.		Resolve merge conflicts.	
4. Merge master into feature1.	$git checkout feature1 $git merge master #if conflicts, fix then $git diff $git status $git add [<file> | .] $git commit –m <comment>	feature1 and master equal. Working copy pointing at feature1.		Resolve merge conflicts.	
5. Pull from origin dev.	$git checkout dev $git pull --rebase origin dev	Local dev equals origin. Working copy pointing at dev.		Resolve merge conflicts.	
6. Merge feature1 into dev.	$git checkout dev $git merge feature1	feature1 and dev equal. Working copy pointing at dev.		Resolve merge conflicts.	

7. Push dev to origin dev.	$git checkout dev $git push origin dev		After push feature1 will be in origin dev branch.	**Deploy to dev environment.** Testing.	**Dev environment matches dev.**
8. Pull from origin master.	$git checkout master $git pull --rebase origin master	Local master equals origin master. Working copy pointing at master.		Resolve merge conflicts.	
9. Merge master into feature1.	$git checkout feature1 $git merge master #if conflicts, fix then $git diff $git status $git add [<file> | .] $git commit –m <comment>	feature1 and master equal. Working copy pointing at feature1.		Resolve merge conflicts.	
10. Pull from origin stage.	$git checkout stage $git pull --rebase origin stage	local stage equals origin. Working copy pointing at stage.		Resolve merge conflicts.	
11. Merge feature1 into stage.	$git checkout stage $git merge feature1	feature1 and stage equal. Working copy pointing at stage.		Resolve merge conflicts.	
12. Push stage to origin stage.	$git checkout stage $git push origin stage		After push feature1 will be in origin stage branch.	**Deploy to stage environment.** **UAT.**	**Staging environment matches staging.**
13. Pull from origin master.	$git checkout master $git pull --rebase origin master	Local master equals origin master. Working copy pointing at master.		Resolve merge conflicts.	
14. Merge master into feature1.	$git checkout feature1 $git merge master #if conflicts, fix then $git diff $git status $git add [<file> | .]	feature1 and master equal. Working copy pointing at feature1.		Resolve merge conflicts.	

	$git commit –m <comment>				
15. Push feature1 to feature1.	$git checkout feature1 $git push origin feature1	Working copy pointing at feature1	After push feature1 will be in origin feature1 branch.		
16. Create a pull request.			Create a pull request.	Final reviews, approvals.	
17. Complete the pull request merge into master.			Complete pull (merge) from feature1 to master.	**Deploy to production. Post deploy testing.**	**Prod environment matches master.**
18. Pull from origin master.	$git checkout master $git pull --rebase origin master	Local master equals origin master. Working copy pointing at master.			
After post deploy testing, clean up feature1.	$git brahch --all $git push --delete origin feature1 $git branch -d feature1 $git brahch --all	feature1 deleted locally.	feature1 deleted from origin.		