# Initial Setup For Development On An Existing Project

## Clone the application using git bash

**git clone <app>**

## Application Configuration

Copy the configuration file example from the project parent directory to your config directory and rename it e.g.

copy StudentFacultyGradeEntry\_configuration.example  to C:\Users\<username>\.grails\StudentFacultyGradeEntry\_configuration.groovy

Then check inside the file for local configurations that are required

## Retrieve submodules

Navigate to the application branch you will be developing on

**git checkout <currentDevBranch>**

**git submodule update --init**

## Compilation and Database Upgrade

**grails clean all**

**grails compile all**

[**http://confluence.sungardhe.com/display/banner/Banner+XE+Common+Upgrade+App+and+Plugins+Developer+Reference+Page**](http://confluence.sungardhe.com/display/banner/Banner+XE+Common+Upgrade+App+and+Plugins+Developer+Reference+Page)

**grails banner-baseline-dbupgrade**

**grails banner-xe-dbupgrade**

**grails seed-data all**

# Local Application Development

Navigate to the branch being developed on. Eg.

**git checkout <currentDevBranch>  
git submodule update --init**

This creates a new branch <currentDevBranch> that is a local version of remote branch <origin/currentDevBranch>

**git pull** here will execute a git fetch on <origin/currentDevBranch> and ensure your local branch is up-to-date with this remote version.

Development will typically take place on your own personal feature branch. To create this:

**git checkout –b myFeatureBranch**

Develop and commit changes on this branch

# Post Development Merge

Once development is completed on <myFeatureBranch> the changes need to be made part of <currentDevBranch>

Ensure <currentDevBranch> is up-to-date:  
**git checkout <currentDevBranch>  
git pull**

For a clean commit history, ensure <myFeatureBranch> now actually branches from <currentDevBranch> at the latest commit:  
**git checkout <myFeatureBranch>**  
**git rebase <currentDevBranch>**

Merge <myFeatureBranch> commits back to <currentDevBranch>  
**git checkout <currentDevBranch>  
git merge <myFeatureBranch**

Delete the <myFeatureBranch> that is now no longer needed  
**git branch –d <myFeatureBranch>**

# Update Remote Repository

**git push origin <currentDevBranch>**

# Local Plugin Development

Same as for application development  
Ensure your application is updated with new plugin version if appropriate

# Squashing Commits For A Cleaner Repository

You may have multiple commits on your feature branch throughout development but only want to push 1 finalised commit to the repository as other developers may not be interested in your intermediate stages. You can ‘squash’ all of your development commits into 1 final commit before pushing.

See here : <http://m037138.ellucian.com:8082/job/devgit1%20documentation/Interactive_rebase_aka_Squashing_commits/>