Developing Integrated Genome Browser

To develop IGB and the IGB API, the core IGB development team uses the **Fork-and-Branch Workflow** described in https://www.atlassian.com/git/tutorials/comparing-workflows/forking-workflow

Quick Summary: Developers fork the team repository on Bitbucket. When working on a bug fix or new feature, they create a topic branch specific to that task. They push their branch to their fork for review and testing. When all is done, they submit a pull request from the topic branch on their fork to the main branch on the team repository. When working on a task recorded in the IGB JIRA system, they include the corresponding JIRA number (e.g., IGBF-1234) in the branch name and every commit to that branch.

Step-by-Step Set-up Guide

Fork the code on Bitbucket - configure your fork

1. Sign up for a free account on Bitbucket. Use an "edu" address if you have one to get more "build minutes" with Bitbucket pipelines.
2. Set up ssh for git to avoid having to enter your password every time you interact with Bitbucket using git. See: Set up SSH for Git.
3. Using your Bitbucket account, fork the team repository: https://bitbucket.org/lorainelab/integrated-genome-browser
4. Configure your fork to link to team project management software JIRA.
    a. Log into Bitbucket
    b. Go to your fork home page
    c. Select "Settings > Links > Add new link"
        i. Choose Link type "Jira"
        ii. Enter Link url https://jira.bioviz.org/
        iii. Enter Link key IGBF
    d. Check that the links work - select "Commits"
    e. Look for commit messages containing the Link key "IGBF"
    f. Note that all link keys now link out to JIRA

Install Git

1. Windows users: Git for Windows

Clone your fork and add the team repository as "upstream"

1. Clone a copy of your forked IGB repository onto your computer using your favorite git client software.
2. Add the team repository as a remote called "upstream" (the convention.)

For example:

```
git remote add upstream git@bitbucket.org:lorainelab/integrated-genome-browser.git
```

Start work - make a branch

Before you start work on a new feature, bug fix, or other improvement, create a **new** branch for the changes you intend to make. This new branch is called a "topic branch" and should ***only*** address one specific, discrete feature or bug fix.

Important: If you are working on a task captured in the IGB JIRA project, include the JIRA issue number in the branch name. This enables the JIRA and Bitbucket sites to create links to each other.

For example:

```
git checkout -b IGBF-1234
```

where IGBF-1234 is the name of the new branch.

To test your set-up, trying compiling and running IGB.

To build and run IGB from the command line:

1. Install Apache maven (these directions from the Baeldung site are helpful)
2. Change into the project directory and type **mvn install**
3. Start IGB by running one of the "run_igb" scripts in the top level of the project.

Note: Following the upgrade of IGB to Java 21, it may not be possible to run IGB from within an IDE. This is due to the requirement to expose internal Java modules. You should be able to build IGB from within the IDE, but to run it please use one of the "run_igb" scripts from your terminal.

Edit code, commit to your clone, push to your fork

Edit your code, test it, commit your edits to your local copy, and then push them to your fork hosted on Bitbucket. If working on a JIRA issue, always include the JIRA ticket name in the commit.

For example:

```
git commit -m "IGBF-1234 Fix typo - covfefe not collusion"
```

Synchronize early & often with the main repository

If the main development branch changes, you must obtain those changes and test them with your branch.

To update your fork's copy of the main branch, check out your main branch on your clone and pull changes from the main branch from the team repository, aliased to "upstream." Then, push the changes to your fork.

For example:

```
git checkout main
git pull upstream main
git push origin main
```

If all goes well, your fork will then receive all the commits present on the main branch on the team repository. To check that it worked, just review the commit history on your fork and compare it to the team repository.

Rebase your branch

After updating your clone and fork with the latest changes to the main branch, you'll need to test how those new commits interact with your topic branch. You should use "rebase" commands to do this. This will move the "base" of your topic branch to the latest commit on the main branch.

To rebase your branch on the latest main, switch back to the main branch, update it with any new commits, check out your feature branch, and rebase.

For example, let's assume you have committed all your work to your topic branch - called IGBF-1234 in this example. Then run:

```
git checkout main
git pull origin main # assumes your fork is up-to-date
git checkout IGBF-1234
git rebase main
```

Next, push your newly rebased branch to "origin" (your fork) to update it:

```
git push origin IGBF-1234
```

Squash commits

If you have made multiple commits for a single Jira ticket it is usually best to squash the commits into a single commit before creating a pull request. See the Git cheat sheet for more information.

Make a pull request - PR

To request that your edits be incorporated into the team repository, you need to make a pull request (PR).

To make a PR using the Bitbucket UI:

- Log in to your Bitbucket account.
- Go to your fork and select "Branches"
- Under "Pull request" select "Create" next to your branch
- A form will appear. Fill in the fields:
    - Select **your fork** and **your branch** as the pull request source (left side).
    - Select the **team repository** (lorainelab) and **main branch** as the pull request target (right side)
    - Click **Create pull request**

Things you need to know about PRs:

- If you make changes to your branch (the source of the PR), those changes will be reflected in the PR. You do not need to create a new PR if you add new commits or otherwise modify your branch.
- You should always rebase onto the latest main branch before submitting a PR.
- Please squash all your commits into one single commit, unless you have a very good reason not to. This ensures that we can easily apply your changes to other branches if required. It also makes code review easier.
- The project admins get email notifications whenever someone submits a PR. However, if you do not hear anything about your PR, get in touch.

Testing your code changes

NOTE:
When you build IGB, existing jar files in the repositories bundle directory are allowed to persist.

This may effect any code changes you may be trying to test.

To be sure that the IGB build you are running reflects the code you currently have, clear all jar files in the bundles directory before building.

In Development:

- In top level of git repository:
    - $ rm *.jar bundles/
- Then build the project:
    - in Netbeans: right click > clean and build
    - in command line: mvn clean install

or

In IGB:

- Reset preferences to default by: Launching IGB > Preferences > Other Options > Reset Preferences to Default
- Delete the IGB folder in AppData located at C:\Users\<user>\AppData\Roaming\IGB (sometimes the AppData folder is "hidden", Go to the View tab in File Explorer and check the Hidden Items checkbox)
- Go to the IGB installation folder (defaults to C:\Program Files\IGB) and run the uninstall.exe application there.
- Now when you reinstall IGB, it should act as if IGB has never been installed.

Learning git

The IGB team highly recommends working through the tutorials in **Learning Git Branching** - https://learngitbranching.js.org/?locale=en_US. Also, when you rebase for the first (or second or third) time, it's helpful to repeat the tutorial on rebasing. Don't worry. You will get the hang of it.