Using GitHub

Ushahidi uses Git for version control, and collaborates using GitHub. This page is intended as a quick start for working with Ushahidi projects. However, if you're completely new to Git you may want to browse the Pro Git ebook and GitHub's help section-- both of which are excellent resources for learning how Git works.

This page assumes you have Git installed and configured, and have a GitHub account.

Strings highlighted in orange require modification on your part, and often identify places to insert your username, the name of the repo you're working with, and so on.
Strings highlighted in [green] are optional, and can be left off if you prefer. They just allow you to personalize your experience a bit more.

Setting Up a Workflow

You should begin by forking the Ushahidi project repository of your choice to your personal account. Register or sign in to GitHub, go to the project page that interests you (for example, Ushahidi_Web) and click the "Fork" button in the header. This will create a copy of the project for you to work with.

Next, open your command line interface and create a parent directory to hold your Git projects. For example:

```
cd ~
mkdir github
cd github
```

Next, clone your newly forked repo to your machine:

```
git clone git@github.com:your_username/repo_name.git [-b branch_name] [local_folder_name]
```

Then we'll want to add an "upstream" remote so we can keep our forked repo in sync with changes that are made to the original/official repo.

```
git remote add upstream git://github.com/ushahidi/repo_name.git
```

Congratulations, you now have a forked repo and you're ready to begin working!

Keeping In Sync

It's important to keep your forked repo up to date with changes from the "upstream"- that is, the original Ushahidi repository that you forked from. It's quick and easy to do this:

```
git fetch upstream
git merge upstream/develop
```

This will grab the latest changes and merge them into your local repository.

Checking Status

You can see where your local repo is in relation to your GitHub fork by using:

```
git status
```

This will give you a list of files that are uncommitted, or have been committed but are waiting to be pushed to your repo on GitHub.

Committing Files

When you've finished working on a new feature, fixing a bug, or otherwise completed a milestone you're ready to commit. This creates a marker in your repository's history, allowing people to see precisely what changes you made up until that point, and ideally your description summarizing those changes.

```
git add filename
git commit -m 'Description of changes.'
```

Alternatively you can use `git add .` to commit changes to multiple files at once, but please be careful in doing so: it's easy to accidentally commit unintended changes.

Pushing Changes to Your Repo

Ready to push your changes up to your GitHub account? Be sure you've committed all your changes (see above), then:

```
git push origin develop
```

Sharing Your Changes with Ushahidi

If you're a member of the Ushahidi organization, it's as simple as:

```
git push upstream develop
```

However, most developers won't fall under this category. In that case you'll want to suggest Ushahidi include your changes by creating a pull request. Pull requests allow us to evaluate incoming changes, ensure code quality and security, and gives us an opportunity to work with developers to polish code patches before they're committed to the repo. Ideally code to be pushed upstream should be in its own branch.

To create a pull request, go to your project's page on GitHub and press the 'Pull Request' button in the header. It's fairly straightforward from there, but you can visit the GitHub help page for a walk-through of the process. Pull requests can take a bit to be vetted; please be patient.

TODO: Add instructions for preparing a pull request branch

Advanced Topics

Fetch from Multiple Forks

```
git remote add herp git://github.com/herp/repo.git
git remote add derp git://github.com/derp/repo.git
git fetch --multiple herp derp
```

“Cherry Pick” Commits to Merge

```
git remote add example git://github.com/example/repo.git
git fetch example
git cherry-pick sha_of_example’s_commit
```

List Available Forks

```
git remote
```

Change Message of Previous Commit

DANGER – Only do this if you haven't pushed your commit to GitHub yet!

```
git commit --amend -m ‘NEW COMMIT MESSAGE’
```

Revert a Bad Commit

DANGER – Only do this with your forked or local repo! Doing this with the Ushahidi repo could be a sync nightmare.

```
git revert sha_of_shameful_commit
```

List Available Branches

> git branch

Create a New Branch

> git branch name