Difference Engine GitHub How-to

Use cases:
1. Download an existing app to your local machine
2. Add a feature to a project

LEGEND

0 Steps are numbered

Notes are on the righthand side

Bash/Terminal commands `$ look like this`

<variable-strings-are-in-brackets>

1 Download exisiting app onto your local machine.

Note: 'qa' is usually known as 'master' on GitHub.
We've renamed it to match the Heroku server name.

```
$ git clone <url-of-the-app>
```

Do all the rails admin procedures in the Readme

2 Add a feature to a project

2a Pull the most up-to-date version of qa from GitHub

```
$ git checkout qa
$ git pull origin qa
$ git checkout -b <my-feature-branch>
```

Switch to qa branch which already exists from clone (qa is the default branch)
This will overwrite your existing qa branch with the most recent, GitHub version of qa
This simultaneously creates and moves you to your new branch, <my-feature-branch>

You can now write code on <my-feature-branch>

2b Save your code locally after each complete piece of your feature is done

```
$ git add --all
$ git commit -m'commit message here'
```

Saves code to repo on your local machine
Commits and adds descriptive comments to your code (on local repo)

Useful git commands

```
$ git status
$ git branch
```

Displays useful info about the state of your local repo
Tells you which branch you're on

2c Push your feature to GitHub when it's complete

```
$ git push origin <my-feature-branch>
```

Your branch is now up on github for all to see (pushes your new branch from local repo and creates a new branch on remote GitHub repo)

2d Go to the branches tab on GitHub Select your new branch, <my-feature-branch>, and create a pull-request by clicking the button to the right of it.

GitHub

New pull request

You can now leave comments about your branch and complete the pull request by clicking the green button.

Create pull request

2e Ask one of your teammates to review and merge your feature

2f The teammate (Reviewer) goes to GitHub to the Pull request tab and selects your new feature:<my-feature-branch>

2g The Reviewer pulls down <my-feature-branch> to their local machine and tests with the app.

```
$ git fetch origin
$ git checkout -b my-feature-branch origin/my-feature-branch
```

2h 3 possible outcomes:

Happy case #0
Code is good and no merge conflicts
Click the merge button on the GitHub GUI

GitHub

Merge pull request

Sad case #1
Code is good but there are merge conflicts
The original author of the <my-feature-branch> pulls qa down to local machine

```
$ git checkout qa
$ git pull origin qa
$ git checkout <my-feature-branch>
$ git merge qa
```

Switches to qa (default) branch on your local machine
Pulls down most recent remote version of qa (default) from github
Switches to <my-feature-branch>
This merges up-to-date version of qa onto the <my-feature-branch>
Terminal will tell you where conflicts are so you can resolve them by discussing or pair programming between the author of <my-feature-branch> and the author of the conflicting code

```
$ git push origin <my-feature-branch>
```

Push your merge-resolved feature to GitHub
Go back to step 2g

Sad case #2
Code is incomplete or buggy (doesn't meet requirements of Trello card)

The Reviewer notifies the code Author of the problem and the Author fixes it on their local machine.
Go back to step 2g

```
$ git add --all
$ git commit -m'commit message here'
$ git push origin <my-feature-branch>
```

See either Happy case #0 or Sad case #1
(repeat until Happy case #0)

2i After <my-feature-branch> has been merged on GitHub, delete it from your local machine and from GitHub (to minimize confusion and clutter).

```
$ git branch -d <my-feature-branch>
```

Deletes local copy of <my-feature-branch>

GitHub branches tab

Go to the branches tab on GitHub and click the trash can icon to the right of the remote branch you want to delete.

OR you can do this from the command line

```
$ git push origin --delete my-feature-branch
```