# Development Tools

## Git

## Git Workflow

#### Branches

**master** - This branch only contains finished releases. The code should always be in a production ready state, meaning that it is ready to be deployed with no more development or unit testing required.

**develop** - This branch is the branch with the latest delivered changes which are ready for release. This can also be known as a continuous integration branch in which multiple new features can be tested for compatibility with each other.

**feature** - There are feature branches for each different deliverable within the scope of the project.

**release** - This branch is created off the develop branch and merged back into master branch. Release branches represent all code from developed features which are tested and complete and ready for deployment. Release branches should be named *release* followed by the version number.

This branching scheme has been inspired by git workflow

#### Init

git init

git remote add origin [git@github.com](mailto:git@github.com):peteehb/platinum-analytics.git

git fetch

git pull origin master

#### Committing

A commit command is issued when a developer wants to store any changes made. The developer adds the changed files to the commit and finishes with a message to describe those changes. A push command is used to sync the local branch with the remote branch on github. Before issuing a push request, a pull request is issued which retrieves any changes which may have been made by other users to the same branch.

git pull

git add .

git commit -am "Commit Message"

git push

#### Branching

Branching in GIT allows the developer to test new features without changing the develop branch. The developer has no commitment to add the changes should they fail to meet the requirement of the feature. Any successfully developed and tested feature can be merged back into the develop branch.

git branch -b new-feature develop

#### Merging

When development is complete on a branch the changes are merged back into the develop branch by either opening a pull request or merging the branch with the merge command.

git checkout develop

git pull --no-ff

git merge new-feature

git push

When working in a team it is usually better to create pull requests instead of merging feature branches. The main advantage of using pull requests is that they give a way for other team members and managers to review code before it is accepted into the main branch. Github shows every change made to files, displaying updates and also changes which will cause conflicts.

#### Conflicts

Conflicts are collisions which arise "when two branches change the same part of the same file and then those two branches are merged." [help.github.com Resolving conflicts from the commandline] To resolve conflicts..resolve conflicts

#### Release a Version

When all of the development in the current scope of the project is completed a version can be released. A version branch is created off the master branch with a given version number. This branch is then tagged, which creates the release.

git checkout -b release-1.2 develop

git checkout master

git merge --no-ff release-1.2

git tag -a v1.2

git checkout develop

git develop

git merge --no-ff release 1.2

git push

## SSH

Ssh was used to run commands on the raspberry pis. It allows remote login to the rpi, returning a console in which to run commands.

By placing my laptop's public into */home/pi/.ssh/authorized\_keys* I am able to login to the pi without entering a password. This process is repeated with the pi's keys placed on the db server. This facilitates the usage of scp to sync local files onto the db server, which also requires no input from the user.

SSH is also used to access the application and database servers to run commands to initialize these components.

## SCP

Used to copy files from the Raspberry Pis to the application server.

SyncData.py

scp '/opt/mon/logs/\*' app\_user@172.168.10.100

## VIM

Vim is a text editor found on almost all Linux systems. It is used to edit text files such as configurations on the Raspberry Pis, APP and DB servers.

## CRON JOBS

A cron job is used to launch a python script which starts the bluetooth monitoring on the Raspberry Pis. This script is run on startup so the user does not need to interact with the pi at all once it has been powered on.

sudo crontab -e

@reboot /opt/platinum/mon/launch\_bluetooth\_mon.sh

@reboot /opt/platinum/mon/sync\_local\_data.sh

#### launch\_bluetooth\_mon.sh

source /opt/mon/.virtualenvs/bin/activate

python StartBluetoothMonitor

#### StartBluetoothMonitor.py

# Check if pi has internet connection

conn\_established=check\_connection()

monitor = BluetoothMonitor(conn=conn\_established)

monitor.\_\_begin\_\_()

def check\_connection():

conn\_established=False

try:

conn = connect\_to\_internet()

except Exception, e:

print(e)

return conn\_established

Another cron job is scheduled to run a python script to check if an internet connection has come available. If it has detected a wifi signal of a known network and established a connection to that network, the python script copies all local SensorReadings csv files to the application server.

# Setup