# Table of Contents

# [Setup](#_Hlk439000995" \s "1,20,26,1,,Setup)

# [Workflow](#_Hlk439001020" \s "1,1893,1902,1,,Workflow)

# [Appendices](#_Hlk439001033" \s "1,2220,2231,1,,Appendices)

# Setup

Prepare an environment for developing your website locally and hosting it using Google App Engine.

**Google App Engine**

* Create a Google Account to be associated with your Google App Engine Account
  + NOTE: You do not need a new account per website; You may deploy multiple websites using the same Google/GAE account.
* Install Google App Engine SDK for Python to your computer (this computer will be where you are developing and deploying the website)
* ??? Authenticate your local GAE Session ???

**Git and GitHub**

* Create a GitHub account
* Install Git on your computer (this computer will be used for development)
* NOTE: Git provides a Cygwin shell, so separately installing Cygwin is not required

**Using Google App Engine to Deploy Website**

<http://www.fizerkhan.com/blog/posts/Free-Static-Page-Hosting-on-Google-App-Engine-in-a-5-minutes.html>

* Application id *id* means the website URL will be *id.appspot.com*
* Install GAE SDK for Python
* Create an application folder; [Example Application Folder Directory Structure](#_Hlk438995490" \s "1,524,571,2,,Example Application Folder Direc)
* [Example app.yaml File](#_Hlk438995332" \s "1,462,484,2,,Example app.yaml File)
* Test the static page: Run the development server by using **dev\_appserver.py ./** and use[**http://localhost:8080**](http://localhost:8080)to view the page
* Deploy the website: **appcfg.py .**

**Create a GitHub Repo**

<https://help.github.com/articles/adding-an-existing-project-to-github-using-the-command-line/>

* Creating git repo
  + Git init, git add, git commit,
* Adding GitHub repo
  + Git remote add origin <GitHub repo URL>
  + (Verify the new URL) git remote –v
  + git push origin master

# Workflow

After your tools and accounts are setup, the following provides a standard set of steps to work on the code for your website.

* General workflow: Plan, develop, test, return to plan/develop if fail, commit, deploy
* Git Branches: master, working
* You may use git bash as your Cygwin Shell

**Proposed Workflow**

* Determine/Document what will be worked on
* Deploy test web server: run

**dev\_appserver.py ./ > /dev/null 2>&1 &**

from the command line in the directory containing the app.yaml of the website files in your local working directory

* *Ensure working directory is up to date: git pull origin master*
* Develop
  + Make changes to code
  + View website changes at <http://localhost:8080>
  + Repeat as necessary
  + Commit as necessary to working branch and remote repo
  + Upon “final” changes, commit to master branch and remote repo, documenting what was successfully implemented and what may need to be done in the future
* If sufficient changes have been successfully implemented, it may warrant deploying the changes to the website
  + From the local working directory, run

**appcfg .**

* Verify changes successful on deployed website at *applicationid.appspot.com*

# Appendices

## Example Application Folder Directory Structure

**application\_folder/**

**- app.yaml # configuration file. we will see in next section**

**- public/ # public folder will contain static files**

**- index.html**

**- js/**

**- css/**

**- img/**

## Example app.yaml File

**application: coolmoon**

**version: 1**

**runtime: python27**

**api\_version: 1**

**threadsafe: yes**

**handlers:**

**- url: /(.+)**

**static\_files: public/\1**

**upload: public/(.\*)**

**- url: /**

**static\_files: public/index.html**

**upload: public/index.html**

**skip\_files:**

**- ^(.\*/)?app\.yaml**

**- ^(.\*/)?app\.yml**

**- ^(.\*/)?#.\*#**

**- ^(.\*/)?.\*~**

**- ^(.\*/)?.\*\.py[co]**

**- ^(.\*/)?.\*/RCS/.\***

**- ^(.\*/)?\..\***

**- ^(.\*/)?tests$**

**- ^(.\*/)?test$**

**- ^test/(.\*/)?**

**- ^COPYING.LESSER**

**- ^README\..\***

**- \.gitignore**

**- ^\.git/.\***

**- \.\*\.lint$**

**- ^fabfile\.py**

**- ^testrunner\.py**

**- ^grunt\.js**

**- ^node\_modules/(.\*/)?**