GITTLE

Pythonic Git for Humans

Python + Git = <3

WHO AM I ?

Aaron O'Mullan
Co-Founder & CEO @ FriendCo.de

- Email : aaron@friendco.de
- FriendCode : @AaronO
- GitHub : @AaronO
- Twitter : @AaronOMullan

FriendCode

PYTHON BACKGROUND

PAST USAGE :

- Web with Django/Flask
- NLP and Machine Learning
- Scrapers & Web Crawlers
- Many, many other things ...

CURRENT USAGE :

- Web APIs @ FriendCode
- Backend Services (with Gittle) @ FriendCode
- Various small scripts & tools

WHY ?

Why implement Git in Python ?

NEED FOR AWESOMENESS :

- Git is Awesome
- Python is Awesome
- Automating Git isn't so Awesome

TO SOLVE MY OWN PROBLEMS AT FRIENDCODE :

- Automate git repo management (push/pull, commit, etc ...)
- Scriptable and usable from Python
- Easy to use & good interoperability in a SOA environment

GOALS

- No/Little boilerplate
- Data Driven
- Simple Data Model, easily serializable
- Pure Python for the "lolz"

GIT BASICS

- Objects / Entities
  - Repository : A Project
  - Commit : A group of modifications
  - Remote : A remote repository for centralized storage
  - Blob : A version of a specific file
  - Tree : A snapshot of blobs and their file/folder structure
  - SHA : SHA1 hash, used as an identifier for the other objects
  - Refs : Named pointers to SHAs ('HEAD', 'origin/master')
- Frequent Actions
  - Commit : Create a commit object
  - Pull/Push : Send/Fetch modifications to/from a remote

SAMPLES

```
# Import Gittle
from gittle import Gittle

# The "Gittle" class is the repo
# It will automatically detect if this repo is a bare one or not
repo = Gittle('path_to_local_repo')

# Get list of objects
repo.commits
# Get list of branches
repo.branches
# Get list of modified files (in current working directory)
repo.modified_files

# Get diff between latest commits
repo.diff('HEAD', 'HEAD~1')
```

CLONING

BASIC:

```
from gittle import Gittle

Gittle.clone(
    '/tmp/gittle',
    'git://github.com/FriendCode/gittle.git'
)
```

BARE (NO WORKING DIRECTORY):

```
from gittle import Gittle

Gittle.clone(
    '/tmp/gittle.git',
    'git://github.com/FriendCode/gittle.git',
    bare=True
)
```

COMMITING

```
from gittle import Gittle

# Open local repository
repo = Gittle('~/git/pyohio')

# Stage single file
repo.stage('slide1.txt')

# Stage multiple files
repo.stage(['other1.txt', 'other2.txt'])

# Do the commit
repo.commit(
    name="Aaron O'Mullan",
    email='aaron@friendco.de',
    message="""Finished first PyOhio slide"""
)
```

BRANCHES

```python
from gittle import Gittle

# Open our repository
repo = Gittle('~/git/pyohio')

# Create branch off master
repo.create_branch('dev', 'master')

# Print a list of branches
print(repo.branches)

# Remove a branch
repo.remove_branch('dev')

# Print a list of branches
print(repo.branches)
```

DIFFS

BETWEEN TWO REFS/SHASREPO.DIFF()

```
from gittle import Gittle

# Gittle supports the X~N notation which is the nth parent commit to the X ref/sha
Gittle('some_repo').diff('HEAD', 'HEAD~1')
```

DIFF WITH WORKING DIRECTORY

```
from gittle import Gittle

# Gittle supports the X~N notation which is the nth parent commit to the X ref/sha
Gittle('some_repo').diff_working('HEAD')
```

GIT SERVER

BASIC (READ ONLY):

```
from gittle import GitServer

GitServer('/', 'localhost').serve_forever()
```

READ/WRITE :

```
from gittle import GitServer

GitServer('/', 'localhost', perm='rw').serve_forever()
```

ADVANCED GIT

Gits beautiful and elegant internals

- Repository structure
  - HEAD
  - Objects
  - Refs (branches & tags)
  -
  - Index
- Protocols
  - SSH
  - TCP
  - Smart HTTP

HEAD

The HEAD file contains the current active reference (branch)

EXAMPLE :

```
ref: refs/heads/master
```

In the above example we see that HEAD points to "refs/heads/master" which is the local master branch

LOCATION :

```
.git/HEAD
```

OBJECTS

- Persists all objects on disk (blobs, trees, commits)
- Objects are stored as text
- Compressed using zlib
- Has a 40 character hexadecimal SHA

EXAMPLE :

SHA

```
fb6f1f81f713037c7412c218d6e321330404e913
```

Location

```
fb/6f1f81f713037c7412c218d6e321330404e913
```

REFS

Refs are named pointers to SHAs (usually commit objects), they are used to store branches and tags

EXAMPLE :

```
$ find refs/
refs/
refs/heads
refs/heads/master
refs/heads/dev
refs/heads/feature/super_improvement
refs/tags/
refs/tags/v1.0
refs/tags/v2.0
```

LOCATION :

```
.git/refs/
```

INDEX

- Only exists for working repositories
- Tracks modifications that have not yet been commited

LOCATION :

```
.git/index
```

SSH PROTOCOL

The Git SSH protocol operates by calling two main executables. Auth is handled by SSH (username/password or RSA key)

PUSHING :

```
git-receive-pack
```

PULLING :

```
git-upload-pack
```

GIT PROTOCOL

- Works over TCP
- In a similar way to SSH uses the following commands :
  - git-upload-pack
  - git-receive-pack
- Outdated (less and less used in practice)
- Authentication is inferior both SSH and Smart Git HTTP

SMART GIT HTTP

- Works over HTTP/HTTPS
  - Not blacklisted by firewalls
  - Builds on top of the common HTTP protocol, thus allowing to easily insert middleware, load balance, ...
- Uses HTTP Basic Auth
- More and more widespread

SMART GIT HTTP URLS

```
['post', v.rpcService,     '(.*?)/git-upload-pack$'],
['post', v.rpcService,    '(.*?)/git-receive-pack$'],

['get',  v.getInfoRefs,    '(.*?)/info/refs$'],
['get',  v.getTextFile,    '(.*?)/HEAD$'],
['get',  v.getTextFile,    '(.*?)/objects/info/alternates$'],
['get',  v.getTextFile,    '(.*?)/objects/info/http-alternates$'],
['get',  v.getInfoPacks,   '(.*?)/objects/info/packs$'],
['get',  v.getTextFile,    '(.*?)/objects/info/[^/]*$'],
['get',  v.getLooseObject, '(.*?)/objects/[0-9a-f]{2}/[0-9a-f]{38}$'],
['get',  v.getPackFile,    '(.*?)/objects/pack/pack-[0-9a-f]{40}\\.pack$'],
['get',  v.getIdxFile,     '(.*?)/objects/pack/pack-[0-9a-f]{40}\\.idx$']
```

The above url to handler patterns are extracted from my node-git-http project that implements a Smart Git Http server in Node (will open source).

WRAP UP

- Gittle is Beta
- Gittle is useable (used in production @ FriendCode)
- Gittle is a brilliant project to look at to understand Git

- Git is beautiful & elegant
  - Simple protocols
  - Simple datastore
  - Distributed
- Git is a standard unit of code storage
- Git is ruling the developer world thanks to GitHub and it's awesomeness, if you don't use Git use it now !

DEMO TIME !!!

THE END

Shameless Plugs

PROJECTS (OPEN SOURCE):

- TV.js (github.com/SamyPesse/TV.js). Apple TV "clone" using torrent streaming and RaspberryPi
- Yapp.JS (github.com/FriendCode/yapp.js). Modern client side framework for writing large JS applications
- And many other cool things :)

QUESTIONS ?