

File: CM.cpp — MSEdemo

```

FOLDERS
+ MSEdemo
  + Makefile
  + sim.dat
+ dist
+ doc
+ ReadMe
+ ReadMeCOC
+ ReadMeCap
+ ReadMeHarvest
+ Scripts
+ Test
+ misc
+ src
  MSEDemo.R
  MSEDemoWin.R
  MSEView.txt
+ misc
  CM.cpp
  CM.h
  CM.admb.R
  OperatingModel.cpp
  OperatingModel.h
  read.admb.R
  Scenario.cpp
  Scenario.h
+ .gitignore
  Makefile
  ReadMe.md

```

```

248:     time t{finish};
249:     elapsed_time=difftime(finish,start);
250:     hour=long(elapsed_time)/3600;
251:     minute=(long(elapsed_time)%3600)/60;
252:     second=(long(elapsed_time)%3600)%60;
253:     cout<<endl<<endl<<endl;
254:     cout<<"--Start time: "<<t.time().start()<<endl;
255:     cout<<"--Finish time: "<<t.time().finish()<<endl;
256:     cout<<"--Run time: ";
257:     cout<<hour<<" hours, "<<minute<<" minutes, "<<second<<" seconds";
258:     cout<<endl<<endl;
259: 
260: // Check calculate A-O-E
261: // calcReferencePoints();
262: 
263: 
264: FUNCTION run_PME
265: {
266:     // This is the entire management strategy evaluation routine.
267:     // So far I use 3 class objects to this via OOP.
268:     // 1) The scenario class: -parameters & data for operating model
269:     // 2) The HarvestControlRule class: Use FORTY_TEN, FIXED_HARVEST
270:     // 3) The Operating model class: call .runMSEscenario to run the
271:     // 4) The operating model calls myrefPoints.h to calculate PMS
272: 
273:     // Scenario class
274:     Scenario cScenario1{agek,n_pyr,reed,value(b0),value(h),value(s),
275:                         value(q),value(sig),value(tau),value(fit),
276:                         value(wt),it,ct};
277: 
278:     // Harvest control rule
279:     // int e_hcr = HarvestControlRule::FORTY_TEN;
280:     // int e_hcr = HarvestControlRule::FIXED_ESCAPEMENT;
281:     // int e_hcr = HarvestControlRule::FIXED_HARVEST_RATE;
282:     // int e_hcr = HarvestControlRule::CONDITIONAL_CONSTANT_CATCH;
283:     int e_hcr = n_hcr;
284:     HarvestControlRule c_hcr(e_hcr);
285: 
286:     // Estimator class (allow user defined estimator)
287:     EstimatorClass cEstimator(sEstimator);
288:     // cEstimator.runEstimator();
289: 
290:     // Operating model class
291:     OperatingModel cOM(cScenario1,cEstimator,e_hcr);
292:     cOM.runMSEscenario(cScenario1);
293: 
294: 
295:     ofstream ofs("CM.rep",ios::app);
296:     ofs<<"t_bo:" << cOM.get_bo() << endl;
297:     ofs<<"t_may:" << cOM.get_may() << endl;
298:     ofs<<"t_fmay:" << cOM.get_fmay() << endl;
299:     ofs<<"t_say:" << cOM.get_say() << endl;
300:     ofs<<"t_bit:" << cOM.get_bit() << endl;
301: 
302: 
303: 
304: 
```

(Finished in 0.1s)

git branch: master, index: 5129, working: 7e 527f, Line 294, Column 5

Doxygen: Main

[Home](#) [Downloads](#) [Documentation](#) [Extensions](#) [Support](#)

Doxygen

About

- [Downloads](#)
- [Changelog](#)
- [Documentation](#)
- [Get Involved](#)
- [Wish list](#)
- [Examples](#)
- [Links](#)
- [Extensions](#)
- [Support](#)
- [Donate](#)

Doxygen

Generate documentation from source code

Doxygen is the de facto standard tool for generating documentation for popular programming languages such as C, Objective-C, C++, C#, Java, Python, PHP, Ruby, Visual Basic, Fortran, VHDL, Tcl, and to some extent C++.

Doxygen can help you in three ways:

1. It can generate an on-line documentation of documented source files. There is also compressed HTML, and Unix man pages, which are much easier to keep the documentation up-to-date.
2. You can [configure](#) doxygen to extract the information it needs to find your way in large source distribution trees. This means of include dependency graphs, inheritance hierarchies, etc., automatically.
3. You can also use doxygen for creating non-documentation files such as makefiles, configuration files, and so on.

Doxygen is developed under Mac OS X and Linux, and runs on all major platforms as well. Furthermore, executables for Windows, Mac OS X, and Linux are available.

Doxygen license

Copyright © 1997-2013 by Dimitri van Heesch

smartell

Search or type a command GitHub

Explore List Blog

News Feed Pull Requests

GitHub Bootcamp If you are still new to things, we've provided a few walkthroughs to get you started.

Set Up Git A quick guide to help you get started with Git.

Create A Repository Create the place where your commits will be stored.

2 days ago **cgrandin** pushed to **master** at **smartell/MSEdemo** 3286131 Added read.admb.R

2 days ago **cgrandin** pushed to **master** at **smartell/MSEdemo** 836f68a Add src/Makefile, R code uses read.admb.R now

2 days ago **cgrandin** pushed to **master** at **smartell/MSEdemo** f2cedes Remove src/Makefile

Using Doxygen & GitHub

Steve Martell
IPHC

Doxxygen

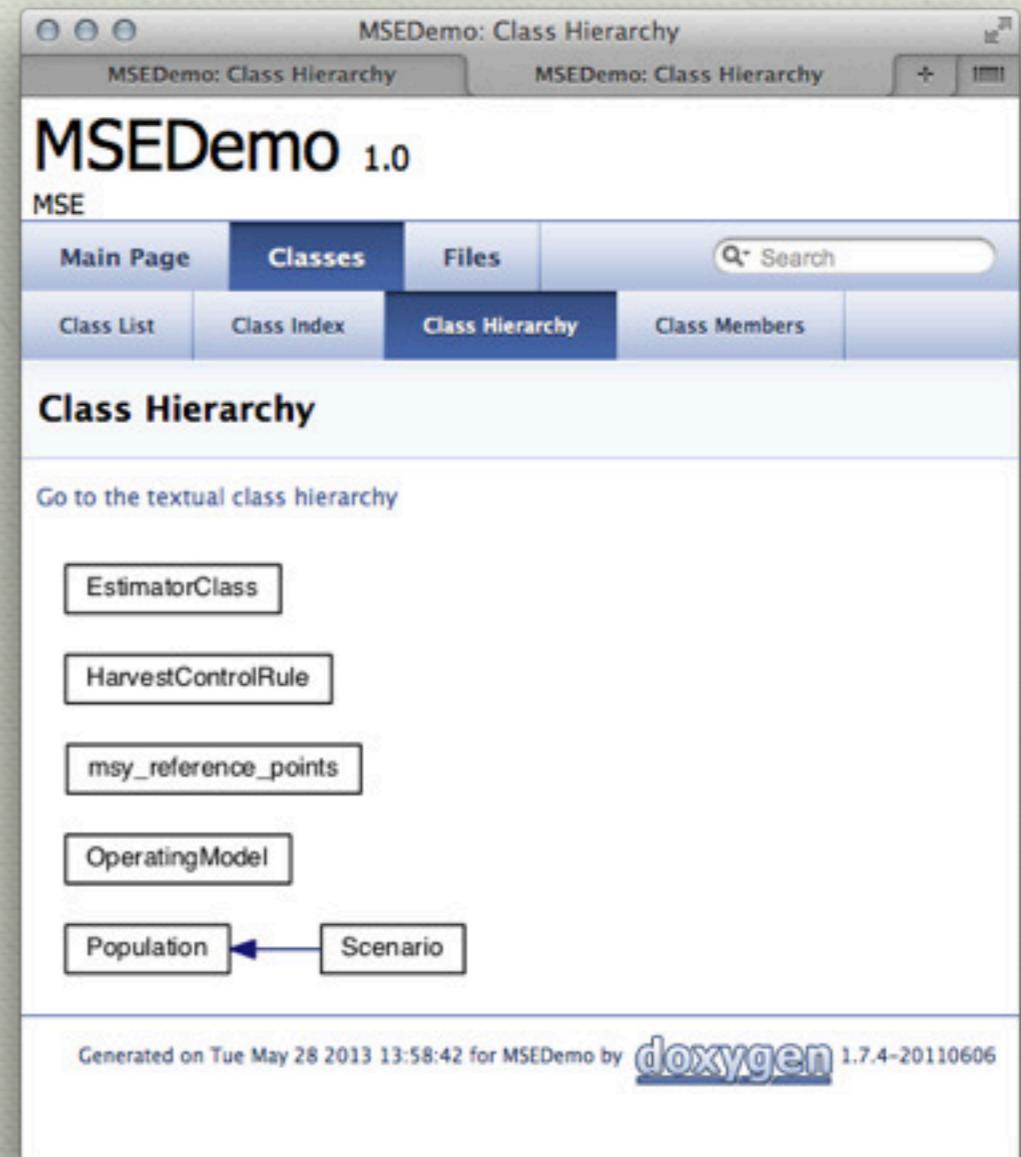
- ◆ Tool for generating documentation from annotated C++ code
- ◆ Widely used, including the ADMB-source code

GitHub.com

- ◆ Code repository
- ◆ FREE FOR OPEN SOURCE PROJECTS
- ◆ Based on Git (DVCS)

Hosting API on github

- ◆ Github allows you to create custom html pages.



- ◆ Doxygen produces html
(and latex) output based on
documentation in the code.

```
/**\brief Function that runs the user defined estimator.\author Steve Martell*/  
void EstimatorClass::runEstimator()  
{  
    adstring arg;  
    arg = m_model+" -ind MSE.dat -nox -est > NUL";  
    system(arg);  
    // cout<<"Finished running the estimator"<<endl;  
}
```

Member Function Documentation

void EstimatorClass::runEstimator ()

Function that runs the user defined estimator.

Author:

Steve Martell

Definition at line 23 of file [EstimatorClass.cpp](#).

```
{  
    adstring arg;  
    arg = m_model+" -ind MSE.dat -nox -est > NUL";  
    system(arg);  
    // cout<<"Finished running the estimator"<<endl;  
}
```

Here is the caller graph for this function:

EstimatorClass::runEstimator

OperatingModel::runMSEscenario

How to use Doxygen with Github

- ◆ Install Doxygen on your local machine.
- ◆ Install git on your local machine.
- ◆ Create a github account (github.com).
- ◆ The follow the next 6 steps ...

Step 1: create project on github

Check this box →

Create a New Repository

GitHub, Inc. github.com/new

Owner: smartell / Repository name: APIDemo ✓

Great repository names are short and memorable. Need inspiration? How about [shiny-tribble](#).

Description (optional): Demo for hosting [Doxygen](#) Code on Github

Public
Anyone can see this repository. You choose who can commit.

Private
You choose who can see and commit to this repository.

Initialize this repository with a README
This will allow you to `git clone` the repository immediately.

Add .gitignore: **None**

Create repository

GitHub
[About us](#)
[Blog](#)
[Contact & support](#)
[GitHub Enterprise](#)
[Site status](#)

Applications
[GitHub for Mac](#)
[GitHub for Windows](#)
[GitHub for Eclipse](#)
[GitHub mobile apps](#)

Services
[Gauges: Web analytics](#)
[Speaker Deck: Presentations](#)
[Gist: Code snippets](#)
[Job board](#)

Documentation
[GitHub Help](#)
[Developer API](#)
[GitHub Flavored Markdown](#)
[GitHub Pages](#)

More
[Training](#)
[Students & teachers](#)
[The Shop](#)
[Plans & pricing](#)
[The Octodex](#)

Step 2: Clone project



A screenshot of a terminal window titled "stevenmartell1 — bash — 66x17". The window shows the following command-line session:

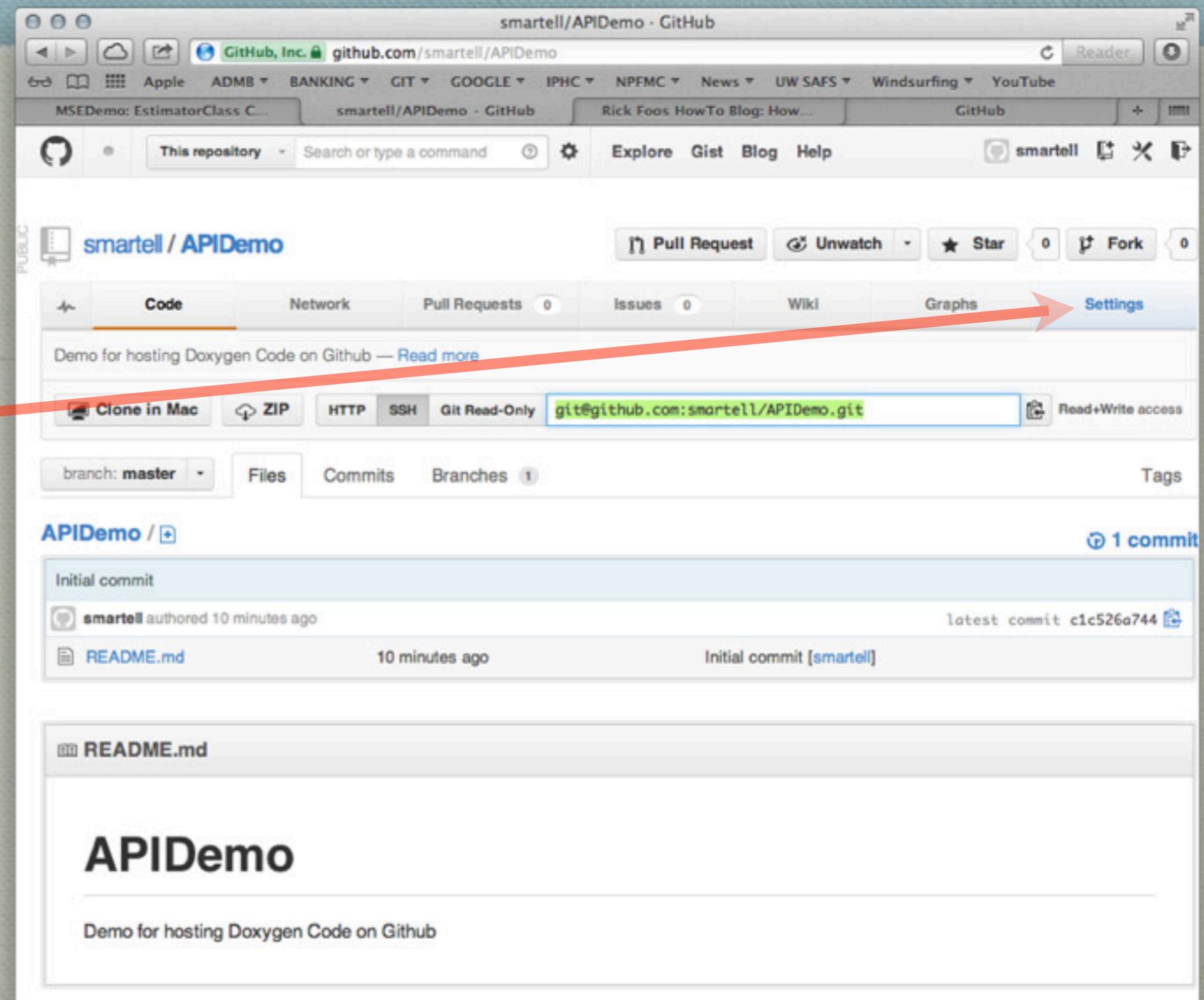
```
bash-3.2$ mkdir APIDemo
bash-3.2$ git clone git@github.com:smartell/APIDemo.git APIDemo/
Cloning into 'APIDemo'...
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Receiving objects: 100% (3/3), done.
bash-3.2$ cd APIDemo/
bash-3.2$ ls
README.md
bash-3.2$
```

Step 3: create html directory

```
stevenmartell1 — bash — 66x17
bash
/Users/stevenmartell1/Documents/APIDemo
bash-3.2$ mkdir html
bash-3.2$ echo "html/" > .gitignore
bash-3.2$ git add .
bash-3.2$ git commit -m".gitignore and html/ folder"
[master 3fb6244] .gitignore and html/ folder
 1 file changed, 1 insertion(+)
   create mode 100644 .gitignore
bash-3.2$ git push origin master
Counting objects: 4, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 300 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
To git@github.com:smartell/APIDemo.git
  c1c526a..3fb6244  master -> master
bash-3.2$ |
```

Step 4: create gh-pages

Choose
Settings



Step 4: create gh-pages

- Choose Settings
- Choose Automatic Page Generator

The screenshot shows the GitHub repository settings page for 'smartell/APIDemo'. The 'Features' section is visible, containing options for Wikis (checked), Restrict edits to Collaborators only (unchecked), and Issues (checked). Below this is the 'GitHub Pages' section, which includes instructions for creating a site using the generator, publishing manually, and a link to more info. A prominent red arrow points to the 'Automatic Page Generator' button. At the bottom, there's a 'Danger Zone™' section for making the repository private and a 'Transfer Ownership' section.

Features

Wikis
GitHub Wikis are the simplest way to let others contribute content. Any GitHub user can create and edit pages to use for documentation, examples, support or anything you wish.

Restrict edits to Collaborators only
Public Wikis will still be readable by everyone.

Issues
GitHub Issues adds lightweight issue tracking tightly integrated with your repository. Add issues to milestones, label issues, and close & reference issues from commit messages.

GitHub Pages

Create a beautiful site for your project with our [GitHub Pages](#) generator.

Author your content in our markdown editor, select a theme, then publish.

To publish a page manually, push an HTML or [jekyll](#) site to a branch named `gh-pages`. [More info.](#)

Automatic Page Generator

Danger Zone™

Make this repository private
Please upgrade your plan to make this repository private.

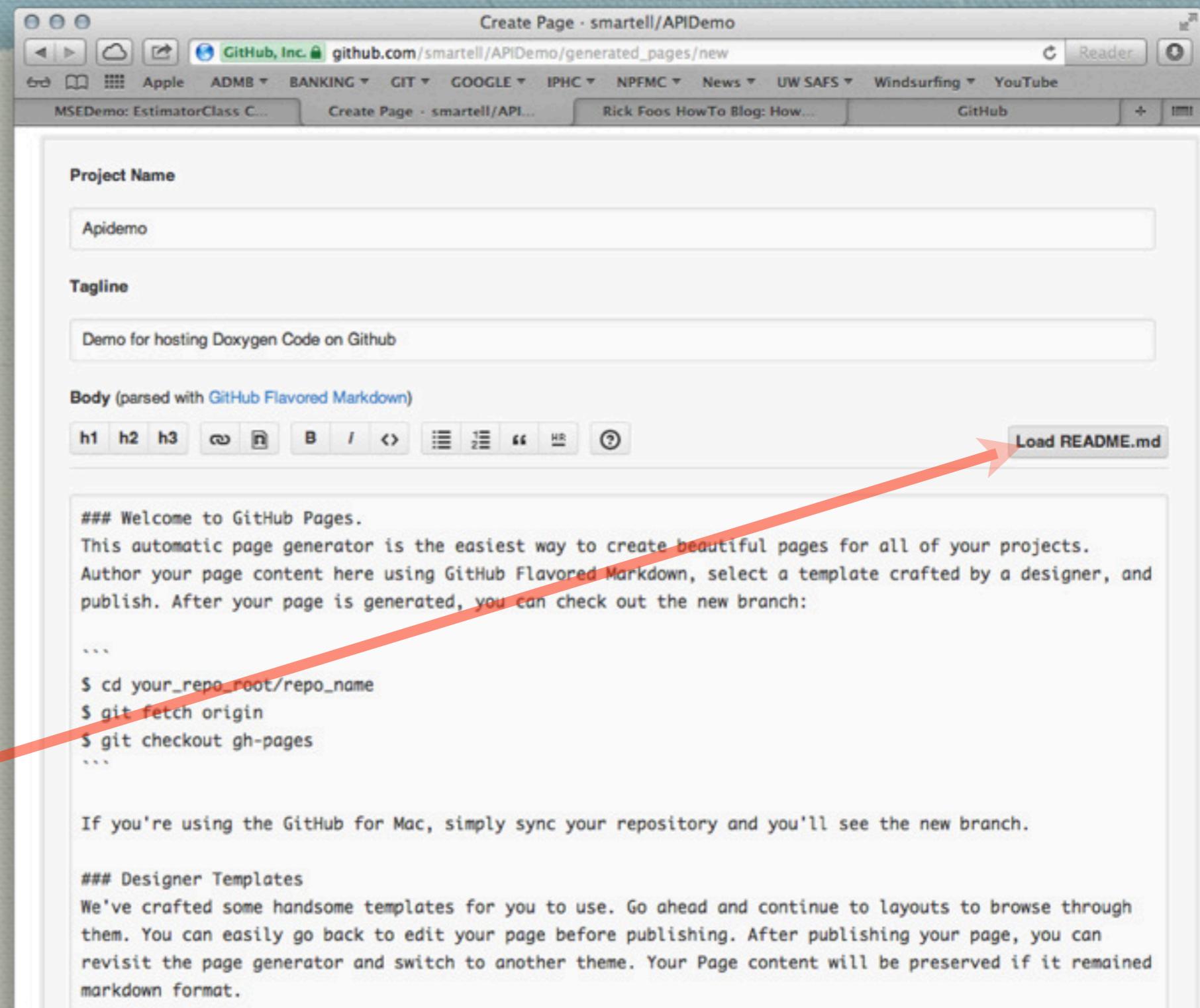
Transfer Ownership

Step 4: create gh-pages

◆ Choose Settings

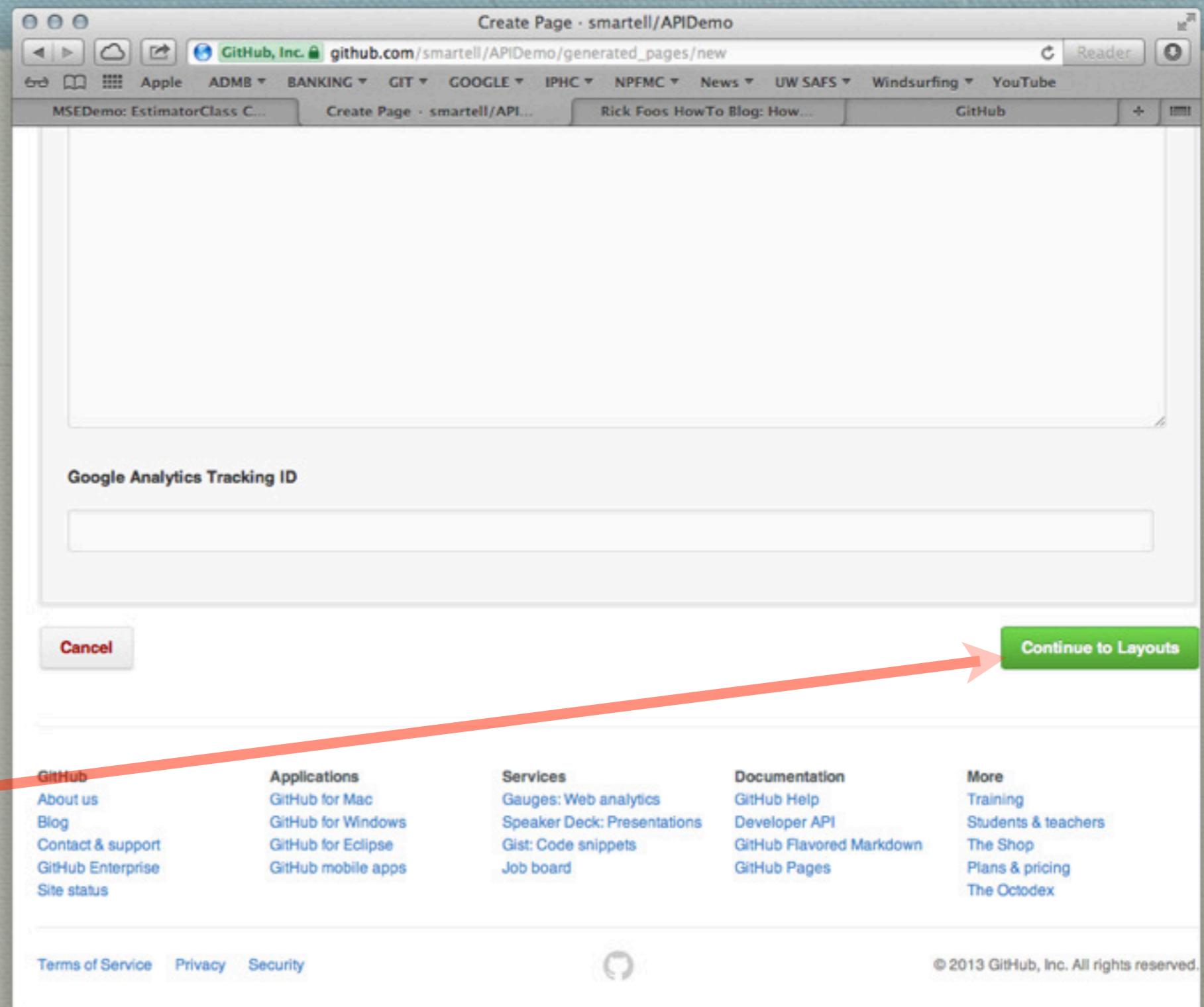
◆ Choose Automatic Page Generator

◆ Choose Load README.md



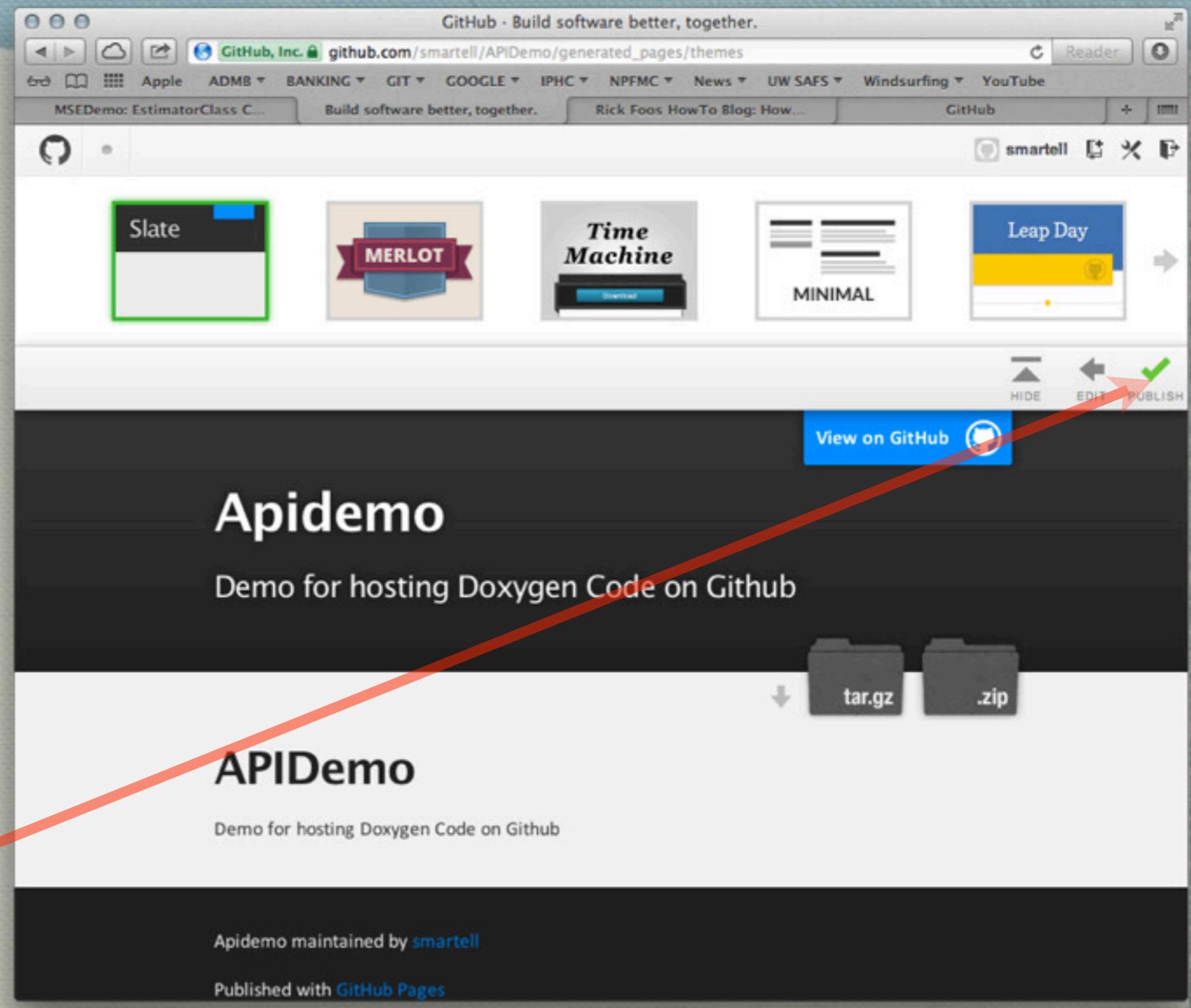
Step 4: create gh-pages

- ◆ Choose Settings
- ◆ Choose Automatic Page Generator
- ◆ Choose Load README.md
- ◆ Choose Continue to Layouts



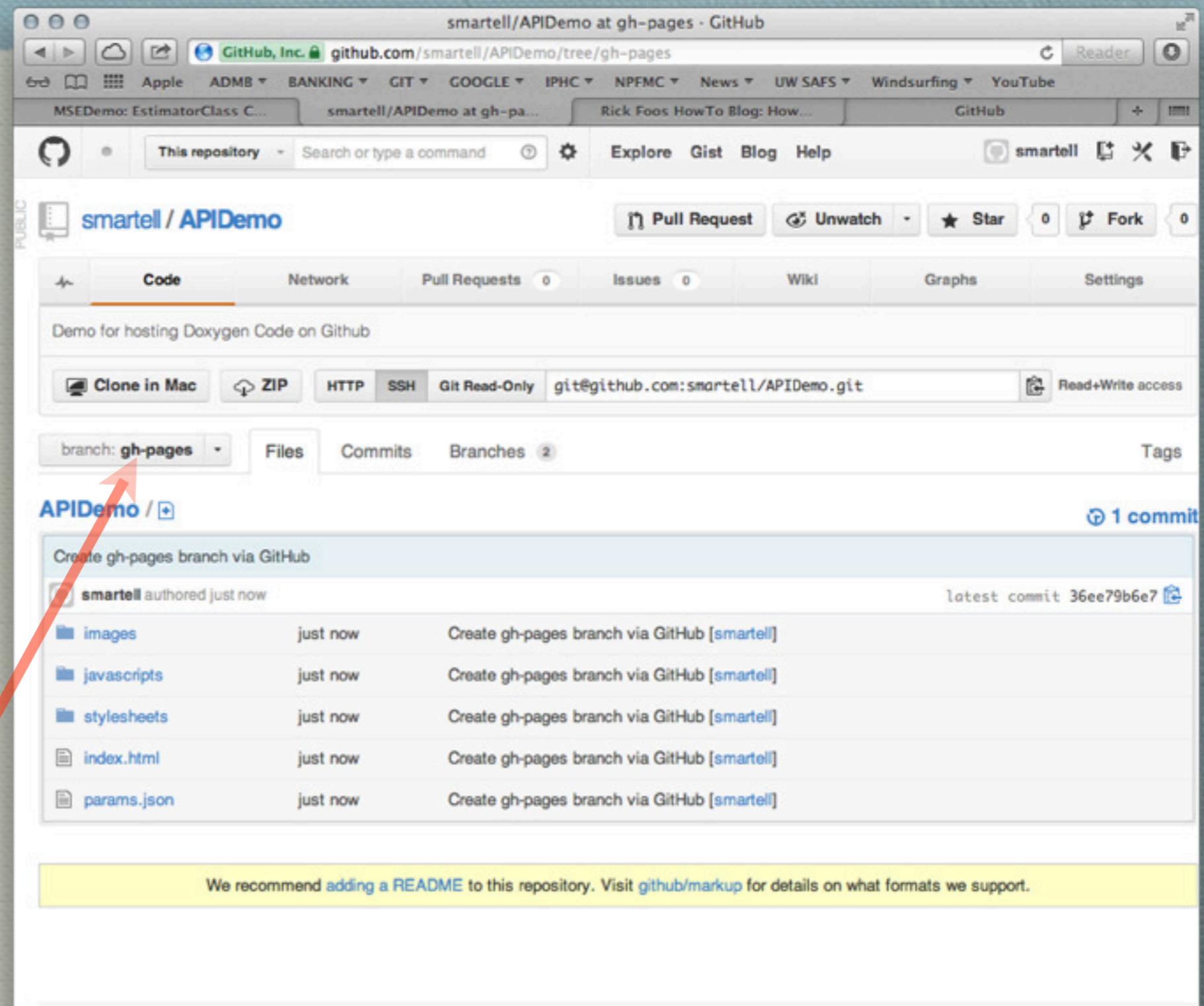
Step 4: create gh-pages

- ◆ Choose Settings
- ◆ Choose Automatic Page Generator
- ◆ Choose Load README.md
- ◆ Choose Continue to Layouts
- ◆ Publish



Step 4: create gh-pages

- ◆ Choose Settings
- ◆ Choose Automatic Page Generator
- ◆ Choose Load README.md
- ◆ Choose Continue to Layouts
- ◆ Publish
- ◆ View gh-pages branch



Step 5: Clone repo in html directory

```
stevenmartell1 — bash — 66x17
bash
bash-3.2$ cd html/
bash-3.2$ git clone git@github.com:smartell/APIDemo.git .
Cloning into '.'...
remote: Counting objects: 20, done.
remote: Compressing objects: 100% (16/16), done.
remote: Total 20 (delta 2), reused 3 (delta 0)
Receiving objects: 100% (20/20), 22.80 KiB, done.
Resolving deltas: 100% (2/2), done.
bash-3.2$ git checkout origin/gh-pages -b gh-pages
Branch gh-pages set up to track remote branch gh-pages from origin
.
Switched to a new branch 'gh-pages'
bash-3.2$
```

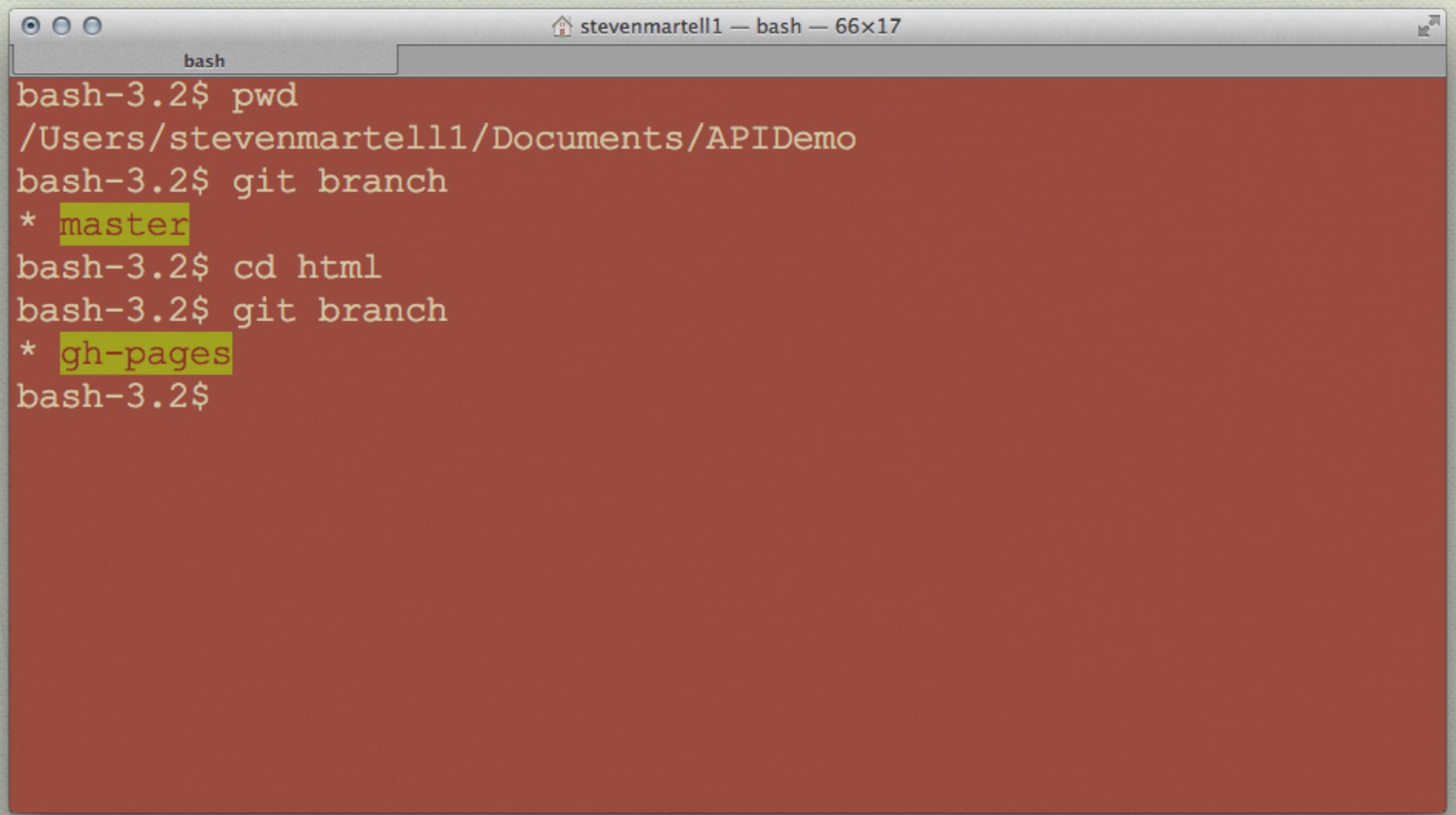
Step 5 (cont.): delete master branch & html files produced by github

```
stevenmartell1 — bash — 66x17
bash
bash-3.2$ git branch
* gh-pages
  master
bash-3.2$ git branch -d master
warning: deleting branch 'master' that has been merged to
          'refs/remotes/origin/master', but not yet merged to HEAD.
Deleted branch master (was 3fb6244).
bash-3.2$ pwd
/Users/stevenmartell1/Documents/APIDemo/html
bash-3.2$ ls
images           javascripts      stylesheets
index.html       params.json
bash-3.2$ rm -r ***!
bash-3.2$ ls
images           javascripts      stylesheets
bash-3.2$ rm -rf images/ javascripts/ stylesheets/
bash-3.2$ |
```

Step 5 (cont.): add README.md and push up to github gh-pages

```
stevenmartell1 — bash — 66x17
bash
bash-3.2$ echo "# html README file" > README.md
bash-3.2$ git add .
bash-3.2$ git commit -m "html ReadmE added"
[gh-pages 9358032] html ReadmE added
 1 file changed, 1 insertion(+)
   create mode 100644 README.md
bash-3.2$ git push origin gh-pages
Counting objects: 4, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 301 bytes, done.
Total 3 (delta 1), reused 0 (delta 0)
To git@github.com:smarrell/APIDemo.git
 36ee79b..9358032  gh-pages -> gh-pages
bash-3.2$ cd ..
bash-3.2$ |
```

At this stage you have a repo within a repo (that ignores the html folder).



The screenshot shows a terminal window titled "stevenmartell1 — bash — 66x17". The window title bar includes a house icon and the user name "stevenmartell1". The terminal itself has a dark red background. The session starts with the command "pwd" which outputs the path "/Users/stevenmartell1/Documents/APIDemo". Then, "git branch" is run, showing a single branch named "master". Next, "cd html" is executed, changing the directory to "html". Finally, another "git branch" command is run, showing a new branch named "gh-pages". The "master" branch from the previous step is also listed.

```
bash-3.2$ pwd
/Users/stevenmartell1/Documents/APIDemo
bash-3.2$ git branch
* master
bash-3.2$ cd html
bash-3.2$ git branch
* gh-pages
bash-3.2$
```

Step 6: Run Doxygen. First create configuration file (Doxyfile)

```
stevenmartell1 — bash — 66x17
bash
bash-3.2$ pwd
/Users/stevenmartell1/Documents/APIDemo
bash-3.2$ mkdir docs
bash-3.2$ cd docs/
bash-3.2$ Doxygen -s -g

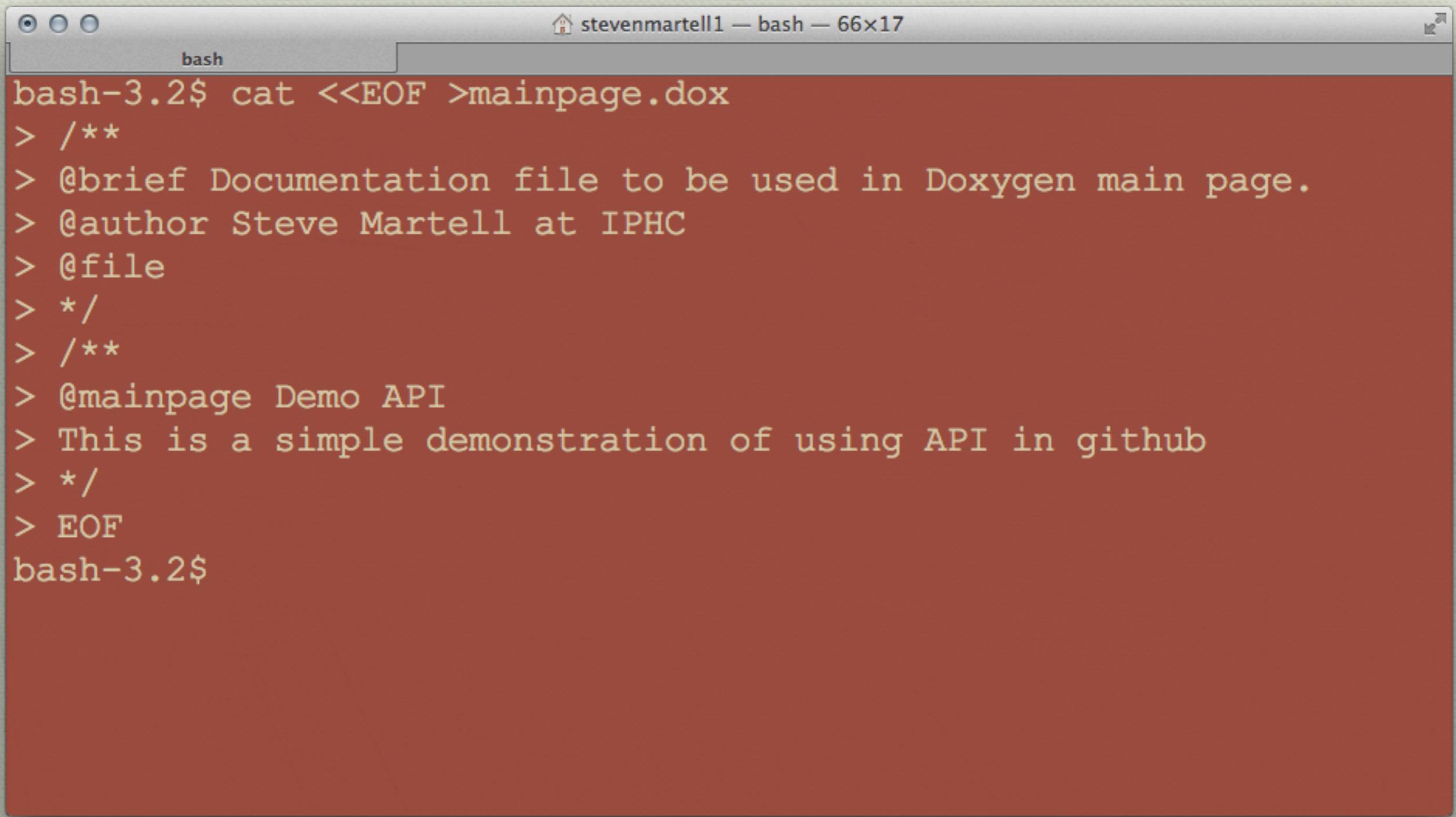
Configuration file `Doxyfile' created.

Now edit the configuration file and enter
doxygen Doxyfile
to generate the documentation for your project

bash-3.2$
```

Step 6: Run Doxygen.

Create mainpage.dox



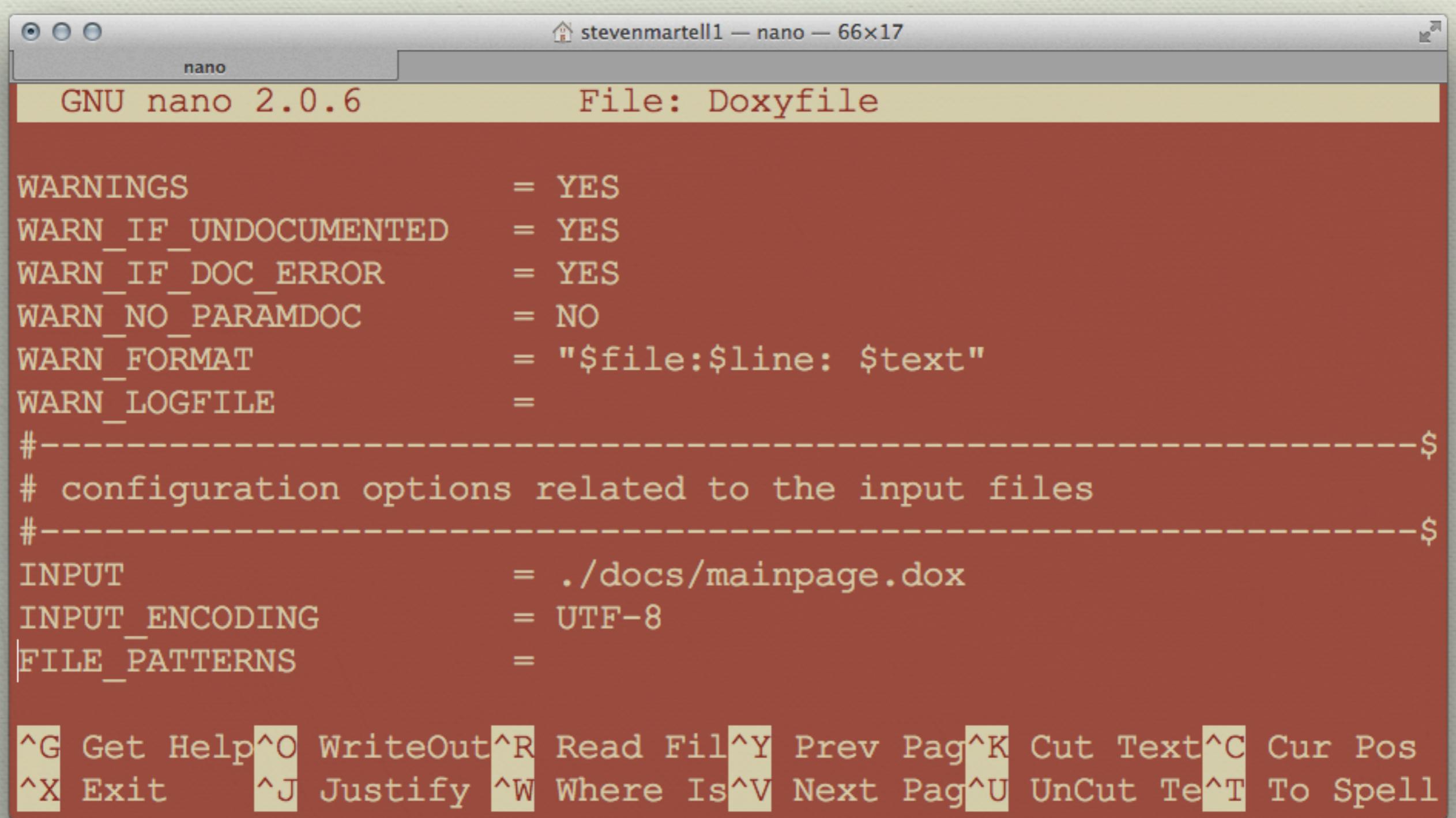
A screenshot of a terminal window titled "stevenmartell1 — bash — 66x17". The window shows a command being run in a bash shell:

```
bash-3.2$ cat <<EOF >mainpage.dox
> /**
> @brief Documentation file to be used in Doxygen main page.
> @author Steve Martell at IPHC
> @file
> */
> /**
> @mainpage Demo API
> This is a simple demonstration of using API in github
> */
> EOF
bash-3.2$
```

The terminal window has a light blue header bar and a dark orange body. The text is white on a black background within the terminal area.

Step 6: Run Doxygen.

Edit the doxyfile (INPUT = mainpage.dox)



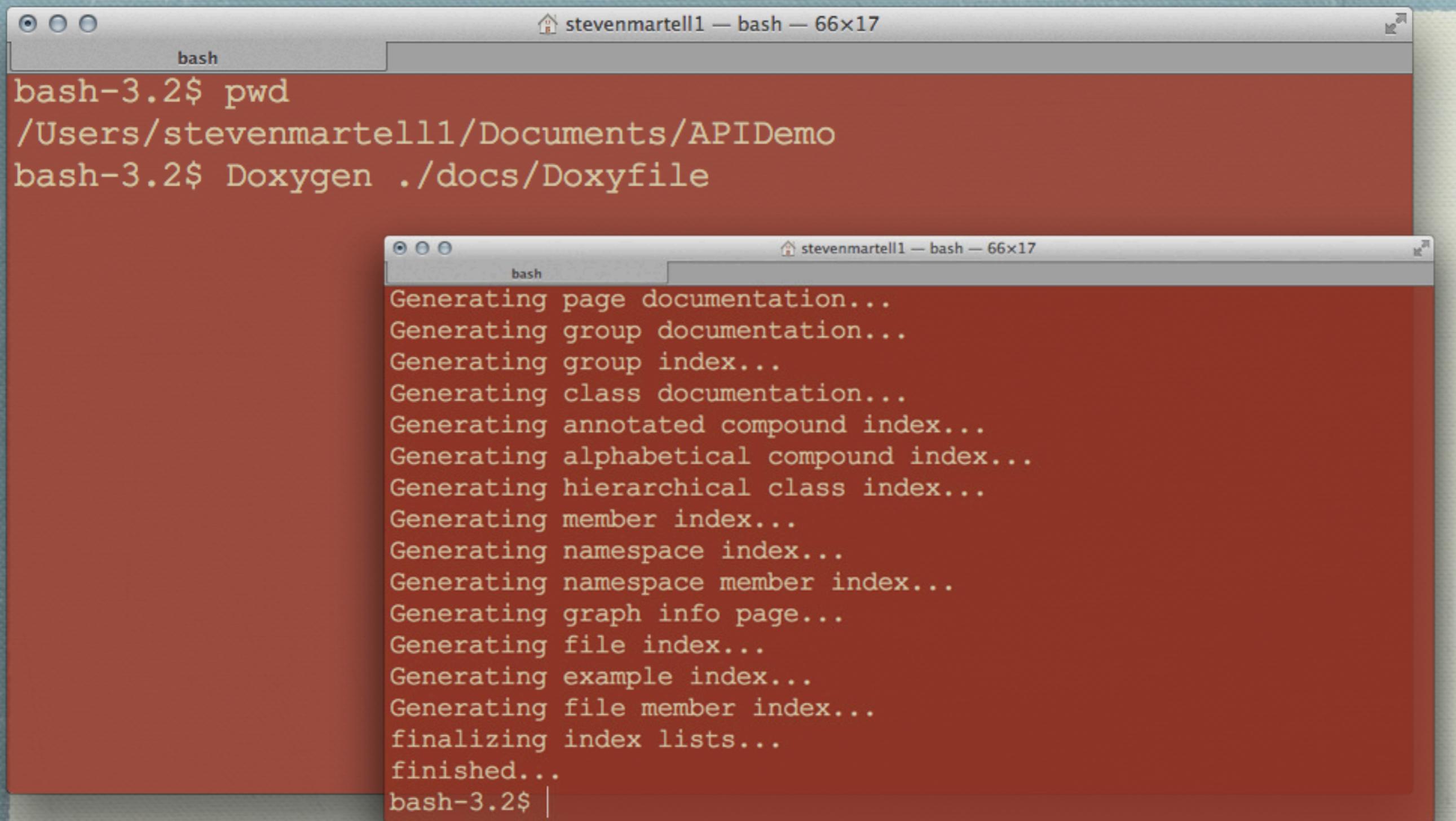
The screenshot shows a terminal window titled "stevenmartell1 — nano — 66x17". The title bar also includes "File: Doxyfile". The window content is a configuration file named "Doxyfile".

```
GNU nano 2.0.6          File: Doxyfile

WARNINGS                = YES
WARN_IF_UNDOCUMENTED    = YES
WARN_IF_DOC_ERROR        = YES
WARN_NO_PARAMDOC         = NO
WARN_FORMAT               = "$file:$line: $text"
WARN_LOGFILE              =
#-----$#
# configuration options related to the input files$#
#-----$#
INPUT                    = ./docs/mainpage.dox
INPUT_ENCODING            = UTF-8
FILE_PATTERNS             =
^G Get Help ^O WriteOut ^R Read Fil^Y Prev Pag^K Cut Text^C Cur Pos
^X Exit      ^J Justify ^W Where Is^V Next Pag^U UnCut Te^T To Spell
```

Step 6: Run Doxygen.

Run Doxygen ./docs/Doxyfile



A screenshot of a terminal window titled "stevenmartell1 — bash — 66x17". The window shows the command "Doxygen ./docs/Doxyfile" being run, followed by a series of status messages from Doxygen indicating the generation of various documentation components.

```
bash-3.2$ pwd
/Users/stevenmartell1/Documents/APIDemo
bash-3.2$ Doxygen ./docs/Doxyfile
Generating page documentation...
Generating group documentation...
Generating group index...
Generating class documentation...
Generating annotated compound index...
Generating alphabetical compound index...
Generating hierarchical class index...
Generating member index...
Generating namespace index...
Generating namespace member index...
Generating graph info page...
Generating file index...
Generating example index...
Generating file member index...
finalizing index lists...
finished...
bash-3.2$ |
```

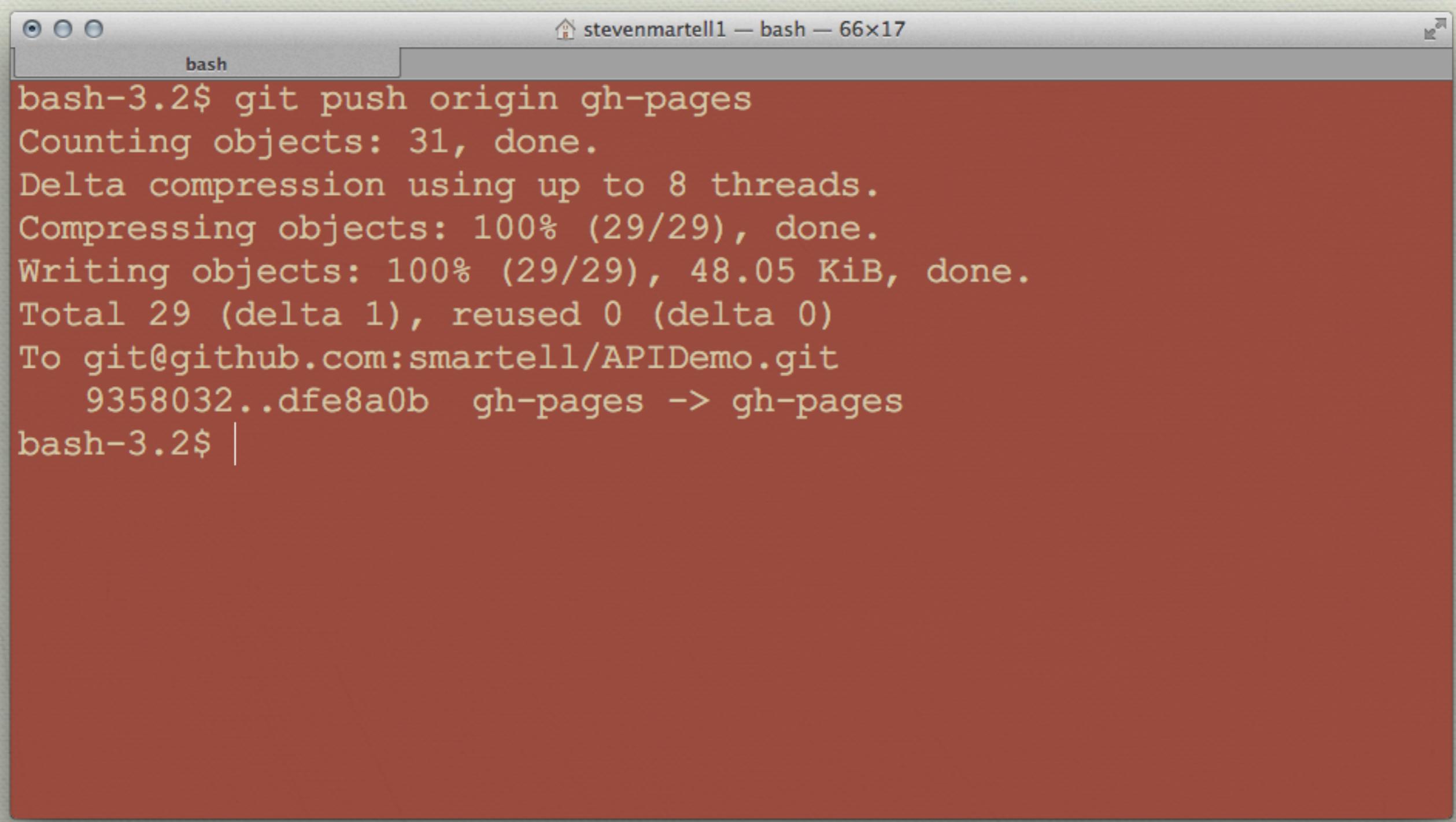
Step 6: Run Doxygen.

Add and commit html files

```
stevenmartell1 — bash — 66x17
bash
bash-3.2$ pwd
/Users/stevenmartell1/Documents/APIDemo
bash-3.2$ cd html/
bash-3.2$ git add .
bash-3.2$ git ci -m"Added Doxygen HTML output"
[gh-pages dfe8a0b] Added Doxygen HTML output
 24 files changed, 2096 insertions(+)
 create mode 100644 bc_s.png
 create mode 100644 closed.png
 create mode 100644 doxygen.css
 create mode 100644 doxygen.png
 create mode 100755 installdox
 create mode 100644 jquery.js
 create mode 100644 nav_f.png
 create mode 100644 nav_h.png
 create mode 100644 open.png
 create mode 100644 search/all_6d.html
```

Step 6: Run Doxygen.

Lastly push upto gitHub



A screenshot of a terminal window titled "stevenmartell1 — bash — 66x17". The window shows the command "git push origin gh-pages" being run and its progress. The output includes:

```
bash-3.2$ git push origin gh-pages
Counting objects: 31, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (29/29), done.
Writing objects: 100% (29/29), 48.05 KiB, done.
Total 29 (delta 1), reused 0 (delta 0)
To git@github.com:smarrell/APIDemo.git
  9358032..dfe8a0b  gh-pages -> gh-pages
bash-3.2$ |
```

Yer done, check out
smartell.github.io/APIDemo/

