OpenHatch

Release 2015.01.01

Jun 06, 2017

Contents

1 Contents **3**

2 Indices and tables **59**

OpenHatch values open communication. We believe documentation is an important part of a welcoming community. If you would like to see some of the projects that we are currently working on, please visit the OpenHatch website <http://openhatch.org>.

Trying to get your bearings with our documentation? Take a look at the *Project Overview* or *Contact Us* to ask questions. We also have a wiki with a lot of informal documentation and other useful things on it.

Confused or dismayed by our documentation? Please let us know so we can improve it! Make an issue on our issue tracker or *Contact Us* via email or IRC to chat with us about it. We take our documentation seriously and wish it to be helpful to you.

CHAPTER 1

Contents

Contributor's Guide

Confused? Contact us.

Project Overview

OpenHatch is an effort to help people get involved in free, open source software communities.

Our main website, openhatch.org, contains tools to find open source projects you can join, interactive lessons ("missions") to learn the skills needed to get involved, and a place to say what projects you work on or want to help.

We keep the code that runs the website on Github in the repository oh-mainline. The documentation can be found at readthedocs.org.

The website is a Python+Django app with jQuery and CSS and HTML on the frontend, and aims for high test coverage (mostly succeeding) and high usability (though it is not there yet). You can read more details about how the code is structured in this document, which we're working to improve.

The best way to contact us about the website is to send an email to our contributors list or find us at #openhatch on irc.freenode.net. (Other ways to contact us.)

Other elements of the OpenHatch project:

- The website is also powered by the "OpenHatch bugimporters," a separate Python-based codebase to download bugs from open source projects' bug trackers, based on Scrapy.
  - code: https://github.com/openhatch/oh-bugimporters
  - docs: http://oh-bugimporters.readthedocs.org/
  - main contact: http://lists.openhatch.org/mailman/listinfo/devel or #openhatch on irc.freenode.net
- The OpenHatch blog, a WordPress-based site where the community writes about great things going on in outreach and diversity
  - view it: http://openhatch.org/blog/

  - read about its theming: http://openhatch.readthedocs.org/en/latest/internals/wordpress.html
  - main contact: http://lists.openhatch.org/mailman/listinfo/publicity

- The OpenHatch wiki, where we store notes about events, future and past tech plans, and other general useful bits of text.
  - view it: https://openhatch.org/wiki/
  - read about its theming: FIXME, undocumented mostly
  - main contact: http://lists.openhatch.org/mailman/listinfo/publicity or #openhatch on irc.freenode.net
- Open Source Comes to Campus, a series of in-person outreach workshops, especially with Women in CS groups, to help university and college students get involved in free software
  - info: http://campus.openhatch.org/
  - main contact: hello@openhatch.org
  - planning list: http://lists.openhatch.org/mailman/listinfo/osctc-planning
- Outreach events such as the Boston Python Workshop for women and their friends that are "affiliated" with us.
  - General info: https://openhatch.org/wiki/Events/Affiliated
  - main contact: http://lists.openhatch.org/mailman/listinfo/events
- We host email lists for other groups working on efforts also aligned with our goals of diversity and outreach.
  - Big list of email lists: http://lists.openhatch.org/mailman/listinfo
  - Sample lists you'll find here: Women in Free Software India; Organizers of Columbia University Open Source Comes to Campus; etc.
- General thinking about how free software can be improved:
  - For everyone: http://lists.openhatch.org/mailman/listinfo/peers
  - Specific planning for the OpenHatch Board: http://lists.openhatch.org/mailman/listinfo/board and for 2014 fundraising http://lists.openhatch.org/mailman/listinfo/fundraising-2014

Projects that are sort of being "incubated" by OpenHatch, in that they're not fully ready yet, but are promising and exciting:

Oppia-based rewrite of the training missions:

- more about Oppia: https://code.google.com/p/oppia/
- code: https://github.com/openhatch/oh-missions-oppia-beta
- main contact: Tarashish (sunu) on http://lists.openhatch.org/devel or #openhatch on irc.freenode.net

"Greenhouse," a project to help open source projects greet new contributors:

- code: https://github.com/openhatch/oh-greenhouse
- main contact: Dave (daveeloo) on http://lists.openhatch.org/devel or #openhatch on irc.freenode.net
- Other lists used by the project: http://lists.openhatch.org/mailman/listinfo/greenhouse and http://lists.alioth.debian.org/mailman/listinfo/welcome-team

Key Features of openhatch.org

This document lists features that the openhatch.org site is supposed to have. These are the guiding priorities for the maintenance and development community.

The “List of features” section below is supposed to state high-level goals that site visitors should expect to achieve when using the website.

To the extent that our work together on the OpenHatch codebase provides value to the world, I think it makes sense to think about what users can expect from OpenHatch.

List of features

OpenHatch Friends and Fans

As a general fan of OpenHatch, I want to learn more about OpenHatch’s activities and the organization (including how to get in touch, and how to get involved) by visiting the front page of the website.

Site visitors

As a site visitor, I want to be reassured that people actually use the site.

Open Source Contributor

As a prospective open source contributor, I want to visit the OpenHatch site and be able to find ways to get involved in open source.

Note: Implementation detail: Right now we focus on helping people find “bitesize” bugs in open source projects.

We’ve gotten feedback that “bitesize bugs” are OK, but that we could do a better job of contextualizing the bugs as being within a particular project, and that many visitors are motivated by the open source project they’d contribute to first, and the task they’d do second.

As a prospective open source contributor, I want to visit the OpenHatch website and learn skills related to getting involved in open source.

Note: We achieve this via the training missions.

As an open source contributor, I want to visit the OpenHatch site and make a profile listing the projects I’ve contributed to.

Open Source Project Maintainers

As an open source project maintainer, I want to visit the OpenHatch site and be able to configure my project to show up in collection of bitesize bugs one can browse.

As an open source project maintainer, I want to provide a friendly face for my project on the OpenHatch site, so that prospective contributors feel welcomed into the project and reach out to me as needed to become contributors.

Getting Started

To get your own instance of OpenHatch running, follow these steps and then get in touch with us.

The code is written in Python. It uses the Django toolkit and tries to stick to good software testing practices. If you have Python experience, you should be able to get hacking pretty quickly even if you don't know Django or testing.

First steps

Getting the source code

OpenHatch source code can be seen through a web interface at https://github.com/openhatch/oh-mainline

To make contributions, you will need to do acquire the source code of www.openhatch.org. Complete these one-time tasks in the following order:

1. Make a new Github account on https://www.github.com if you don't already have one.
2. Fork the oh-mainline Github repository located here at https://github.com/openhatch/oh-mainline. Click on the fork button located on the upper right corner of the project page. Now you have your own personal copy of the oh-mainline repository.
3. Install git the version control system. If you have already done so, skip to the next step.
4. Clone your personal copy of the oh-mainline repository to your computer by typing this command into your terminal

   ```
   $ git clone https://github.com/<YOUR_GITHUB_USERNAME>/oh-mainline.git
   ```

It will take up to five minutes, depending on your Internet connection. it's kind of a big repository. (90 megabytes, or so.)

Installing and running a local development site

Once you have the repository, read the *Installation* documentation or open up the *oh-mainline/docs/getting_started/installation.rst* file in any text editor and follow the instructions.

Read it, and follow the few short steps to getting your local site going. It should take about 5 minutes.

Next steps

Get in touch

We really recommend that you get in touch with us. (It's not quite mandatory, but we'll all be happier if you do)

1. Join the Devel mailing list and say hello.
2. Visit *the #openhatch IRC channel in freenode*.

OpenHatch holds development meetings on IRC; our goal is to hold these meetings weekly. The meetings are announced on *devel@lists.openhatch.org*. Please join us on IRC and share your ideas or ask questions.

Read more documentation

Before you start hacking OpenHatch, we strongly advise you to watch Learning new codebase talk by Justin Lilly given during DjangoCon 2010.

You can find more tips about hacking OpenHatch in the Category:Hacking_OpenHatch!

You can find things to work on by browsing our issue tracker or asking us!

Start contributing!

We mark issues that are particularly good for new contributors with the "bitesize" keyword on our issue tracker. You can find the open easy issues here.

If you find an issue you like and it isn't assigned to anyone, assign it to yourself and start hacking. If it is assigned to someone already, but it looks like they haven't gotten around to working on it, leave a note on the ticket saying that you are interested in taking it (you can also try asking on IRC).

When you are ready to submit a contribution for an issue, follow the guidelines at *How we handle contributions*.

If you ever feel like you are getting stuck or could use some design feedback, don't hesitate to ask for help on the IRC channel, on the devel mailing list, or on the issue ticket. Attending the weekly development meetings on IRC is a great time to ask for help or recommendations on issues to work on.

Getting unstuck

Doing searches:

When doing searches for particular keywords in the GitHub repository, the "vendor" directory will most likely return lots of hits, making your search more difficult.

GitHub provides a little known method for excluding specific directories from the search. In the Search textfield, enter:

```
YOUR_SEARCH_WORD -path:vendor
```

The -path:vendor parameter will exclude the vendor directory from your search and will maximize the probability of getting meaningful hits.

It is also possible to do a search locally from the command line, by going to the mysite/ directory and using:

```
git grep YOUR_SEARCH_WORD
```

That will limit the search to your local code.

Installation

OpenHatch is currently designed to run on Python versions 2.6.0 to 2.7.8. OpenHatch site does not currently support Python 3 or above. We hope to do so in the future.

Note: These installation instructions are tested nightly on Ubuntu 12.04 and Debian stable. Last verified on Windows XP 11/7/2013, Mac OS X 10.9.5 October 29, 2014, and Mac OS X 10.10 December 15, 2014.

Overview

This repository contains (primarily) Python code written on top of Django and other Python modules. We bundle a copy of all of the essential dependencies for oh-mainline to run so that you can get started immediately (there is no need to download and configure additional software from other sources).

It should take you about 15 minutes to get the OpenHatch site running locally on your computer.

Here are the basic steps you'll follow for installation:

- *Open a command prompt*

- *Get the code from the GitHub repository*
- *Set up the database*
- *Run the site*

After running your own instance of the OpenHatch website, you can play with the code from an interactive shell on your computer.

If you want to work on core backend features, like the bug importer, or let your local site rescale images, please see Advanced Installation documentation to learn about optional dependencies and automated testing.

Essentials

Open up a command prompt

Note: Understanding how to open a command prompt for your operating system is an important prerequisite to master before continuing with the remaining installation instructions.

For the rest of these instructions, you have to open a command prompt:

- On a *Linux* or similar system, find a program with "terminal" or "konsole" in the name. Run it.
- On a *Mac*, click the search icon in the top-right of the screen and search for Terminal. This should find the Terminal program, stored in /Applications/Utilities. Run it.
- On a *Windows* computer, you'll need to use Git Bash. To do so, download and install the .exe at this link. (It will ask you a bunch of questions. You can accept the defaults.) Once that is installed, launch Git Bash by going to: *Start -> All Programs -> Git -> Git Bash*

Get the code from the GitHub repository

If you already have an *oh-mainline* directory on your computer, then you already have the source code. You may skip to the next step, *Set up the database*.

If you're reading this installation instruction file on the web, then you will need to clone the repository from GitHub to your local computer.

Step 1: Open a command prompt on your computer

Step 2: Create a new directory on your computer:

```
mkdir localhatch
```

Step 3: Change to the new directory:

```
cd localhatch
```

Step 4: On your personal Github account, fork the OpenHatch repository at https://github.com/openhatch by clicking on the "Fork" button on the right-hand side. Github now takes you to your forked repository of the OpenHatch upstream repository.

Step 5: On the command prompt, clone the repository from your fork of the GitHub OpenHatch code to your local computer:

```
git clone https://github.com/<YOUR_GITHUB_USERNAME>/oh-mainline.git
```

If your commands are executed successfully, you may continue to the next step.

Note: For most Django projects, you would need to install the dependencies at this point (using *pip install -r requirements.txt*), but for the OpenHatch project, these packages have been bundled for your convenience in the *vendor* directory, so they don't need to be installed separately.

Set up the database

Before you run the commands in the this section, make sure you have changed your **present working directory** to the *oh-mainline* directory.:

```
cd oh-mainline
```

Your local OpenHatch site will store data in a SQLite database.

Run this command to create the database and add tables for our dependencies:

```
python manage.py syncdb --migrate --noinput
```

Note: We have to pass *–noinput* to request that Django not ask you questions. This is due to a bad interaction between Django's superuser creation system and our custom profiles. *–migrate* creates an empty database, with zero users and zero projects, ready for you to fill with data as you use your local version of the site. If you want your site to have a database filled with data like what is on the main OpenHatch.org site, you can import a data snapshot. See Importing data snapshots for more info about that.)

This will print out *lots* of text. Once all of the text is printed, you should see something like the output listed in *Output Samples* below. Afterwards, your database tables should be ready. You're ready to run the site.

If you are using Windows and do not have Python installed, you may get the error "Python: command not found." Follow these instructions to install Python.

Run the site

Before you run the commands in the this section, make sure you have changed your **present working directory** to the *oh-mainline* directory.

Run this command which will start a web server locally on your computer:

```
python manage.py runserver
```

As long as the "runserver" is running, you can visit your local version of the OpenHatch site in a web browser. So, try surfing to:

http://localhost:8000/

Note: Your local version of OpenHatch does not contain any user data in its SQLite database. You may add users manually through the user interface. If your development needs require a large amount of prepopulated data, you can find information about Importing data snapshots in the Advanced Installation documentation.

You're done

Hooray! That's it for the essentials. You have everything you need to get the site going, and to start making changes.

Now is a good time to find us on IRC or the email list and say hello! We can help you make the changes you want to. *Contact Us*!

If you want to read about some optional dependencies, open up Advanced Installation documentation. You can also read about how to maintain your local site in the Maintenance documentation.

Output Samples

Here is a sample output from `python manage.py syncdb --migrate --noinput`:

```
Synced:
  > ghettoq
  > django.contrib.auth
  > django.contrib.contenttypes
  > django.contrib.sessions
  > django.contrib.sites
  > django.contrib.webdesign
  > django.contrib.admin
  > registration
  > django_authopenid
  > django_extensions
  > south
  > django_assets
  > invitation
  > voting
  > reversion
  > debug_toolbar
  > sessionprofile
  > model_utils
  > djkombu
Migrated:
  - mysite.search
  - mysite.profile
  - mysite.customs
  - mysite.account
  - mysite.base
  - mysite.project
  - mysite.missions
```

Testing Basics

OpenHatch strives to follow best practices for testing. One common practice in the Python community is Test Driven Development (TDD). In TDD, a developer will write a test for a new feature before creating the feature's source code.

Running the OpenHatch test suite

You may run the test suite to see if all tests pass before you begin making changes to the code. To run the test suite,:

```
python manage.py test
```

The test suite begin running all of the tests and will display the test progress in the console window.

Running the test suite without warnings

You may run the test suite and turn off warnings, such as "deprecation warnings", being output to your screen. To run the test suite without warnings,:

```
python -Wignore manage.py test
```

The test suite will display its progress on the console but will not display any warnings.

Running a subset of tests

If you are working on a particular area of the source code, you may find it helpful to run a subset of the tests. You may pass an argument after the `python manage.py test` command.

Currently, you may pass one or more of the following arguments: *account*, *base*, *missions*, *project*, *search*, and *customs*. For example,:

```
python manage.py test missions
```

will run all the tests related to the OpenHatch missions.

Controlling detail of test output

You can use `--verbosity` or `-v` to specify the amount of notification and debug information that should be printed to the console.

- 0 means minimal output.
- 1 means normal output (default).
- 2 means verbose output.
- 3 means very verbose output.

For example,:

```
python manage.py test -v2
```

will run all the tests and display a more verbose output.

Additional testing information

The Internals section of this documentation contains more detailed information about the test suite, advanced testing, and continuous integration.

If you'd like to learn more about testing, we strongly recommend going through Ned Batchelder's blog post Getting Started Testing.

Documentation Basics

You can read the most up to date documentation online at this link: http://openhatch.readthedocs.org/en/latest/index.html

Source files

The documentation source files can be found in the docs/ folder of the oh-mainline repository: https://github.com/openhatch/oh-mainline/tree/master/docs

reStructuredText and Sphinx

The documentation for OpenHatch is built using Sphinx and deployed at readthedocs. You can learn more about the Sphinx, which uses reStructuredText (.rst files) format, and Sphinx deploy commands.

Style

We encourage you to help improve the OpenHatch documentation. We have a *Documentation Style Guide* which gives an overview of our basic documentation style and guidelines.

Changing or Adding Documentation

Before making any changes, we recommend taking a moment to read the *Documentation Style Guide*.

Making changes to documentation via pull request

To alter the documentation, you'll want to clone the github repository. (Not sure what cloning is? Read our version of Git Basics.)

Once you've got a local copy, you can edit the files in the docs/ directory to make changes. You may find the official Sphinx reStructuredText primer useful for that.

To see the changes rendered locally, you can run the render_docs.py script found in the tools folder of the oh-mainline repository:

```
python tools/render_docs.py
```

You will find the documentation rendered into html format inside the docs/html folder of the oh-mainline repository. You can view it in your browser and check that you like your changes before submitting them. (Again, see Git Basics for help submitting your changes.)

Once you submit your changes as a pull request and they have been merged by a maintainer, they will appear in the openhatch/oh-mainline repository. The openhatch.readthedocs.org/ files will update automatically via a github web hook.

Note: If you've create a new file or edited/deleted a "toctree", you may get an error "WARNING: document isn't included in any toctree". This means a file is not referenced by a table of contents anywhere. Consider adding it to one. See Sphinx guide or reference.)

Making changes to documentation via readthedocs/Github editor

If you're having trouble navigating the documentation by opening and editing files locally, you can also try paging through the readthedocs. Each page should have an 'Edit on Github' link in the righthand corner. When you click this link, Github will automatically create a fork of the project for you (if one does not automatically exist). Once you finish editing, make sure to submit a pull request.

How we handle contributions

We use git/Github to handle contributions. If you're new to git, you may appreciate this guide.

As a contributor

Creating a pull request

Get the latest version of master

Before creating a pull request, update the master branch of your local repository with the latest version of the OpenHatch-owned repository. In git, you can achieve this by developing on a branch and rebasing your branch commits on top of master with git rebase master. You can also use git rebase -i master for an interactive rebase, in which you can reorder and edit commits. We prefer rebasing to merging because rebasing preserves a linear commit history, which can be easier to keep track of and reason about.

Test your changes

1. Add unit tests with your functionality changes or additions.
2. Use docstrings and comments where appropriate. Spell-check your additions. Try to apply pep8 standards.
3. Test your changes on a local instance of the website. Prove to yourself that your changes address the issue they are supposed to address.
4. Run the test suite, and make sure your unit tests pass and all tests that passed before your changes still pass.
5. Use a tool like PyChecker to check for bugs.

Generate a pull request

Generate a pull request by pushing your changes to your personal remote. You can then create a pull request to the OpenHatch repository. In the commit message, include the issue the pull request addresses. For example: "Closes: http://openhatch.org/bugs/issue398"

Submitting a pull request

1. Add a link to the pull request in the issue ticket at https://openhatch.org/bugs.
2. Change the issue status to "need-review".
3. Join IRC and say that you have an issue ready for review.

The pull request will be checked for code style errors (such as pep8 violations) by the lint-review bot. To know more about the bot, see Checking coding style errors in pull requests with lint-review.

If the reviewer says it's ready to go, your request will get merged in short order. If the reviewer has feedback he/she wants addressed, make the necessary revisions and start back at the "Check/test your changes" section.

Permit us to share your work

1. Join our Devel email list by entering your email address into the form at http://lists.openhatch.org/mailman/listinfo/devel
2. Send an email to devel@lists.openhatch.org with a message like:

   > The work I contribute to OpenHatch is work I have permission to share. I permit anyone to re-use it under the terms of the Affero GPL, version 3 or later. Additionally, contributions in the docs/ directory can be shared under the terms of CC Zero 1.0.

As a reviewer

Apply the pull request to your local repository

Find the URL of the pull request by going to the main pull request page on Github and clicking on the link named 'command line'. Github will give you instructions, including the URL of the pull request. Follow all of the instructions except the last one, which tells you to push back to the origin.

Review the pull request for correctness and cleanliness

Things to think about:

1. Does the pull request make sense? Does it look readable?:

   ```
   git log -p
   ```

2. If the author hasn't already done this: tell the author "Please email devel@lists.openhatch.org saying that you're okay with your work being under the Affero GPL, version 3. If you're willing, it is preferable that you say 'the Affero GPL, version 3 or later, at your option'."
3. If you have revisions you'd like to see made, change the issue status to "in-progress", re-assign the issue to the pull request submitter if it isn't already, and leave your review feedback on the pull request.
4. After leaving the revisions in the comments, you may optionally leave a note to the author regarding expectations on when or if the pull request will be worked on further. You may use the below example:

   ```
   To add an arbitrary (but perhaps useful for planning) time
   bounded-ness, is this addressing the above something you'd be
   interested in doing over the next 3 days? If not, I can take care of
   it after that. If you're interested in being the one to do so, but you
   know you need more than 3 days is not long enough, that is fine; just
   say so, and we're happy to wait for you to perfect these changes.
   ```

Push and deploy

If you want to deploy the changes, and you have push access to the repository, you can do so by following the steps listed in the section labeled Deployment.

If you don't have push access, you will need to rope someone else in for this. Anyone in the Login team can do a push as well as deploy access. Asheesh Laroia (paulproteus) is the traditional person to do this, but it's good to ask someone else so they get practice!

Things to know:

- If you push to origin/master, Travis CI will test it.
- Once you're happy, you can run the deploy script, but note that will push the current HEAD to origin/master.

```
cd mysite
./scripts/deploy
```

- When you deploy, check a page or two to make sure things are okay.

For more details on how we use Continuous Integration and Travis CI, see Continuous integration.

Technical Guide to Development and Documentation

If you've made it this far without saying hello you should definitely do so!

Documentation Style Guide

Note: OpenHatch's Documentation Style Guide is still a work in process. We like Kenneth Reitz's excellent Guide Style Guide for its concise and consistent style guidelines. We refer you to this guide until ours is posted.

Layout of OpenHatch's source code

This section should help developers get a better understanding of OpenHatch's `oh-mainline` repository.

This section is a basic overview. Additional details can be found in the `LAYOUT` file in the root directory of `oh-mainline`.

Directory structure

docs/ This directory contains documentation files for OpenHatch and is rendered at RTD ReadtheDocs <http://openhatch.readthedocs.org>.

downloads/ This directory may be used by deployment for temporary storage.

[FUTURE: It may be possible to remove this directory in a later release.]

htmlcov/ This directory stores reports created by the coverage testing tool.

mysite/ This directory contains the OpenHatch website and all the "Django apps" that are part of it. Each subdirectory is an app.

Each of the apps has some tests, views, and frequently models and forms. Their file paths are:

/tests.py

/views.py

/forms.py

/models.py

You can read more about tests, views, forms, and models in the official Django tutorial:

https://docs.djangoproject.com/en/1.5/intro/tutorial01/

customs/ This directory contains "import/export" code like the support for loading and saving snapshots of the OpenHatch database, downloading data from bug trackers, and scanning other websites for information about OpenHatch members.

profile/ This app contains code on information about OpenHatch users.

account/ This app (mostly) contains code to let a user edit their information.

missions/ This is the Django app where the training missions live.

search/ This Django app contains the views and models necessary to display the volunteer opportunity finder, also known as bug search.

The apps also use other Django features, or Django add-ons. Here is a list by filename and a URL reference to further info:

/templatetags.py https://docs.djangoproject.com/en/1.5/howto/custom-template-tags/

/migrations/ http://south.aeracode.org/

/api.py http://django-tastypie.readthedocs.org/

/fixtures.json https://docs.djangoproject.com/en/1.5/howto/initial-data/

/management/commands https://docs.djangoproject.com/en/1.5/howto/custom-management-commands/

/view_helpers.py http://lists.openhatch.org/pipermail/devel/2013-March/003151.html

/templates/ https://docs.djangoproject.com/en/1.5/topics/templates/

tools/ This directory contains helper tools that make things easier for a contributor (for example, a script for rendering docs).

vendor/ This directory contains code from other projects that we rely on. (For more information, look at http://kitsune.readthedocs.org/en/latest/vendor.html .)

Informational Files

These informational files are found in the root directory of `oh-mainline`.

README.rst Read the README! Read it first.

It points to our main documentation; this LAYOUT file is just a quick thumbnail view of what different files in here are.

(Aside: The ".rst" extension indicates reStructuredText format is used.)

LICENSE This file explains what permissions you have, if you want to re-use source code you find in this repository.

CREDITS This file gives credit for files used by OpenHatch.

LAYOUT This file (the one that you are viewing now) gives an overview of the project high-level directory and file structure.

dotfiles

In general, dotfiles provide configuration details.

.coveragerc coverage testing configuration

.gitattributes git

.gitignore Files ignored by git

.travis.yml Travis continuous integration configuration

Other files and executable files

manage.py This is the well-known and widely-loved Django management script.

Procfile A file used when deploying the site.

requirements.txt This file indicates packages (i.e. ones that are not pure Python code and contain compiled code) that are installed in a different manner than packages found in the vendor directory.

run_importer.sh This shell script is used for deployment and running of scraping of projects for suitable bugs for contributors. [FUTURE: This file may be relocated to a different place.]

setup.py This file lists the dependencies of the OpenHatch codebase.

Advanced installation

This file contains information on things that you don't have to do! If you're a completionist or really just like installing dependencies or reading the OpenHatch documentation, keep reading.

Overriding local settings

If you wish to override the default settings, you may create a separate file with individual settings you wish to change. There is a hook at the end of the in **mysite/settings.py** that allows contributors to override individual settings. To override settings, create a new file in the mysite directory and name it **local_settings.py** . You can place any settings you wish to override in this file.

Automated testing

The OpenHatch code comes with automated tests that you can run to make sure that it is set up To execute all tests, run this command:

```
python manage.py test
```

For more about tests visit: http://openhatch.org/wiki/Automated_testing

Postfix, postmap and testing

The code for site creates a configuration file for an email service, Craigslist-style, that lets all users have an anonymous inbound email address that goes to them. In particular, the code configures a Postfix-based alias map for this. When that alias map changes, we notify Postfix by calling postmap.

If the postmap binary (/usr/sbin/postmap) is not available on the system, it is better not to try running that binary during testing. So before tests we check for presence of the postmap binary and log a warning if it is not present on the system.

Optional dependencies

You will probably see some warnings when you run the site, providing you information about extra dependencies.

These extra dependencies require compiled code, AKA Python C extensions. Depending on your operating system, you might install these using a GUI installer, the program "pip", or a package manager like apt-get.

For each dependency, we specify how to get it with pip *or* apt-get. If you have a Debian or Ubuntu system, use the apt-get instructions. Otherwise, try pip. (And if it doesn't work, ask for help quickly.)

Re-scaling images

When you add a profile photo, and at other times, the site attempts to rescale the image to fit into the visual constraints of the page. Django and the OpenHatch code work together with PIL (the Python Imaging Library) to transform images.

PIL requires some C dependencies, so the site can function without it. If you want image rescaling to work, you must install PIL.

To do that, run one of these commands:

```
$ sudo apt-get install python-imaging
$ pip install PIL
```

Bug import dependencies

If you want to modify the code that downloads bugs (AKA "volunteer opportunities") from other projects, you need these dependencies:

lxml: An XML and HTML parsing library

```
$ sudo apt-get install python-lxml
$ pip install lxml
```

Bug Importers

If you want to use the customs bug importers, they will need to be installed. You can do this in one of the following ways:

- pip install https://github.com/openhatch/oh-bugimporters.git # (readonly)
- Clone the repo into a folder at the same level as oh-mainline.

Training missions: System tools

Most of the training missions work fine without installing any extra dependencies. There are two exceptions.

The Subversion training mission requires that you have the 'svnadmin' tool installed. To get it on Debian or Ubuntu, do:

```
$ sudo apt-get install subversion
```

Subversion repositories for the svn training missions are stored in mysite/missions-userdata/svn. This directory must be available via svnserve for users to be able to do the svn missions. See mysite/missions-userdata/svn/README to read how to set up svnserve.

On Windows and Mac, the code currently can't find svnadmin.

The git training mission expects to find "git" on your system path. On Debian/Ubuntu systems, do:

```
$ sudo apt-get install git-core
```

Maintenance

You may want to read about how to maintain an OpenHatch site. maintenance.rst tells you about that.

Working with git

OpenHatch has a long history of helping contributors build existing and learn new skills. This document section has information that OpenHatch contributors have found useful when working with git. We encourage you to share helpful git resources by adding your favorites to this file and creating a pull request.

Git Commands	What it does
git **clone** <repo>	Used to clone the repo `git clone <repo> | <name>`
git **commit**	Commit an applied change on the given branch
git **remote**	Track a remote branch
`git revert <commit-SHA-1>`	Revert changes in a commit
`git fetch <remote> <branch>`	The git fetch command imports commits or tags from a remote repository into your local repo.
git pull	Updates your repo. Shorthand for `git fetch` followed by `git merge FETCH_HEAD` Recommended to use it with –rebase.
git log	Lists commits made in the current branch of the repo. Check this. Hacks: `git log --pretty=format:"<%h> [%an] %d%Creset %s"`
git rebase : • `git rebase <base>` • `git rebase -i <base>` • `git rebase -i HEAD~NUM` • `git rebase -i bbc643cd^` • `git rebase --abort` • `git reflog`	• Rebasing is the process of moving a branch to a new base commit • Interactive rebase • Modifying to a head • Modify to specified commit bbc643cd • Abort a rebase • Tracks the changesets to the tip of branch

Issues with Pull Requests

Helpful tips for new contributors

1. Get into OpenHatch workflow .
2. **Understand** your problem first.
3. Learn to **go to a certain commit**.
   - **Tip:** Each commit has a hash value(SHA-1) which is fixed. Switch to a commit.

4. If you need to **modify a single commit** as requested by the maintainer in the pull request **revert the old commit with a new commit and push it** mentioning the changes you have made:

    - **Tip:** Please **understand** reset and revert . The ideal solution is to revert previous commit , edit it and push it for changes that are **not published** or *force push* it for changes that **are published**. You can also undo a commit and *force push* the changes made.
    - **Note:** Commits **do not** technically **change** when we force push the commit, reset or modify them. But the **hash gets updated**, and the new hashed commit gets tagged to that branch. You can easily find you previous **commit(s) on github** too just by adding `/commit/<branch>` to the repo address. It will show you the **remote git log**. See the hash get changed after rewriting the commit.

5. Sometimes you have **two or more commits on your pull request**, which is usually not desired by maintainers. The solution is to do an **interactive rebase** and squash the previous **N** commits:

    - **Tip:** Please **understand** interactive rebase. Checkout the branch the pull request represents, count the **number[N]** of commits you need to squash from the pull requests or `git log` on that branch, then `git rebase -i HEAD~N`. N (**number of commits before**) is usually 2 if you want to **squash 2 commits**. Change pick to squash for all but one line. Save the configuration. Then force push the new commit `git push -f origin <branch>`.

6. If you have **unwanted commits** attached to the pull request or **history is broken** then you need to do an **interactive rebase** :

    - **Tip:** It is better to tell others that you are having such problem as it needs rewriting history. Please **understand** rebasing and please see your logs. The **solution** is an interactive rebase **Command**: `git rebase -i HEAD~N` (N gives the number of the revisions up in that branch). Try to pick N carefully count the commits as maintainers will ask you to give the finalized result in a single commit.
    - The **interactive rebase** on that branch will show the commands that runs on that branch and finally shows on the pull request. Delete the commits you don't need and keep the ones you need. Squash the needed commits to a single commit.
    - **Note:** Branches are technically a **tagging system** to commits that have a **hash value**, they are **relative**, even the master branch. It can operate with commands like squash, pick and many others. The commands are played from **top to bottom** on each commit and finally shown in the pull request. See the options.

Resolving Merge Conflicts

Conflicts are essentially two or more commits from different branches which **overlap** because they have **different content** on the same revision. Understanding Merge conflict, you can manually resolve it with git.

Tools like kdiff3 helps you pick that **content of the commit** which you want to get merged in the final commit. A tutorial on using **kdiff3**. Kdiff3 can be configured for all types of version control ranging from **git, svn or mercurial**.

Note: The merge conflicts can also be resolved with an interactive rebase.

Tips on Installing kdiff3

Ubuntu:

```
$ sudo apt-get install kdiff3
```

Mac:

```
$ brew install kdiff3
```

Configure kdiff3:

Recent Git versions have built-in support for kdiff3.

```
$ git config --global merge.tool kdiff3
```

This makes `git mergetool` launch kdiff3.

Note: We recommend that you refer to the kdiff3 documentation for the latest installation instructions.

Become a git Expert

If you like git a lot and use it often, use tools like **git-extras** and **aliases** to increase your **productivity**.

Alias

Saves you Keystrokes. These scripts are added to your `.bashrc`, `.zshrc` or any file you want to source. Open `.bashrc` on your favorite text editor. Follow the instructions.

Examples

```
alias gc="git commit -m "$1""

alias shortform="the longer version of the command"
```

git-extras

Get **40 extra commands** that you may find helpful but are missing in git, **i.e git-undo, git-summary, git-changelog, git-effort.**

Installation

Ubuntu:

```
$ sudo apt-get install git-extras
```

Mac:

```
$ brew install git-extras
```

Usage

- `git-summary` # gives the status of the **hours and duration you are actually working on a git project**.
- `git-effort` # shows your **file stats** on the project.
- `git-undo` # undo a git commit.
- `git-extras` # shows the list of commands.

Advanced testing

Note: Twill is going away in the OpenHatch code base and is being replaced by WebTest (yay!).

The purpose of this page is to show you how to write automated tests within the OpenHatch codebase.

If you already know how software testing works, skip to the section *Details specific to OpenHatch*.

Tests: An overview

You can run the many tests that are part of the OpenHatch code:

```
$ python manage.py test
```

During the test run, you'll see a bunch of dots. Dots mean success.

Tip You really should write a test if you add new functionality. This page explains how and when to write new tests and how to run the tests we have.

What a basic test looks like

Imagine this is in `mysite/base/views.py`:

```
def multiply(x, y):
    return x * y
```

Then this would be in `mysite/base/tests.py`:

```
import mysite.base.views

class TestMultiplication(django.test.TestCase):
    def test_return_one(self):
        self.assertEqual(35, mysite.base.views.multiply(7, 5))
```

When a test fails

When a test fails you will see:

- **FAILED** followed by the test_name
- the Traceback
- the failure summary (e.g. **FAILED (failures=2, errors=1, skipped=9)**)

To force a failure, maybe you are just curious to see what it will look like, you can add to the test code:

```
self.assertTrue(False)
```

This assertion will fail and so will the test containing this code.

General testing tips

Read the official Django testing guide

The official guide on Django testing is quite good. It says:

> *The best part [about writing tests for Django code] is, it's really easy.*

OpenHatch contributors use the Django "unit test" style of writing tests.

How to write code that is easy to test

If you are writing a function, have it "accept arguments" for its data, rather than having it calculate the input itself. For example:

Good:

```
def multiply(x, y):
    return x * y
```

Less good:

```
def multiply(x):
   y = settings.MULTIPLICATION_FACTOR
   return x * y
```

It's okay to rely on things like system settings and database content, but in general if your functions are simpler, they are easier to test.

Details specific to OpenHatch

We regularly run Automated Testing

OpenHatch's Automated Testing is run by Jenkins, with the interface on the virtual machine donated by GPLHost @ http://vm3.openhatch.org:8080/

Where to write your tests

In general, add tests to the same Django app as you are editing. For example, if you made changes to *base/views.py*, then add a test in *base/tests.py*.

The test files are kind of 'sprawling'. It doesn't really matter where within the *tests.py* file you add your test. I would suggest adding it to the end of the file.

The OpenHatch test case helper class

Note: Twill is going away in the OpenHatch code base and is being replaced by WebTest (yay!).

In *mysite/base/tests.py* there is a TwillTests class. It offers the following convenience methods:

- *login_with_client*

- *login_with_twill*

The subversion missions test cases

When running or testing the subversion mission locally, subversion (svn and svnadmin) must be installed on the local system. If subversion is not installed, the tests will not be run.

Settings information related to subversion, such as path locations, can be found in the *settings.py*.

About fixtures

Note: Twill is going away in the OpenHatch code base and is being replaced by WebTest (yay!).

To run your tests

What Django app did you write your test in? Let's pretend it was in the `base` module. To run all the tests in `base`:

```
$ python manage.py test base
```

To run just a few specific tests

You can run just one test. For example, a test named *base.Feed*:

```
$ python manage.py test base.Feed
```

Or you can run two (or more) tests:

```
$ python manage.py test base.Feed base.Unsubscribe.test_unsubscribe_view
```

The structure here is *app.class.method*. If you want to just run your own new test, you can do so.

Mocking and patching

Note: This section is important, but we haven't written it yet. Please consider helping us write this section. See Documentation

Testing with Twill, versus the Django test client

Note: Twill is going away in the OpenHatch code base and is being replaced by WebTest (yay!).

To make a long story short:

- The Django test client is good at introspecting how the function worked internally.
- Twill tests are good because they let you say "Click on the link called 'log in'".

Deployment

This is a quick-and-dirty page explaining how to deploy new versions of the OpenHatch code.

Prerequisites

- You must be part of the *Login Team* (so your SSH key is available in Github and you're in the openhatch-committers group, and also that your SSH key is in the deploy@linode.openhatch.org account's .ssh/authorized_keys)
- You must be at a computer with that SSH key
- Deploying takes about 3 minutes, maybe less if things go well. (If there are database migrations to run, it can take dramatically longer.)

How the deploy script works

You need to have these programs installed: **ssh**, **git**.

The script does two things:

- Pushes the current local master branch to Heroku.
- SSHes to the two linodes, where it runs mysite/scripts/deploy_myself.sh which updates the site.

Recommended way to use the deploy script

```
# Make sure .git/config has these 5 lines
[remote "origin"]
    url = git@github.com:openhatch/oh-mainline.git
[remote "heroku"]
    url = https://git.heroku.com/openhatch-production.git
    fetch = +refs/heads/*:refs/remotes/heroku/*

git fetch  # get the latest

git checkout origin/master -b deploy_me  # create a deploy_me branch

# Then get the patch file with e.g. wget, and do:
# Import the patch into current branch, probably called deploy_me
git am /path/to/the/patch.file

git log  # and sanity-check it

# If you like it, do:
cd mysite
./scripts/deploy
```

It's really important to make the separate branch so that you don't accidentally push random local work into the live site.

Notes about the deployment

Here are some relevant details of how web requests get routed to the OpenHatch code.

- Web requests hit CloudFlare, which proxies them to Heroku (for openhatch.org and www.openhatch.org) or linode.openhatch.org (for other OpenHatch sites, like wiki.openhatch.org).
- linode.openhatch.org has an nginx that handles some requests itself, and dispatches others to Apache.
- In production, we use a mysite/local_settings.py file that imports mysite/deployment_settings.py and overrides the Django SECRET_KEY, DATABASE_URL, and a few other settings.

Other sites we host

The OpenHatch organization hosts some other websites, including bostonpythonworkshop.com and corp.openhatch.org. For information about that, read the documentation on the wiki about static site hosting.

Deploying to Heroku

Overview

Heroku is a service that provides web application hosting. They have a free-of-cost tier. If you want to create a web URL for the changes you've made to your version of the OpenHatch site, deploying that code to Heroku is an easy, no-cost way to do that.

The steps are all listed below. Keep reading to start following them. Note that many of the instructions require typing commands into a command prompt.

Install the Heroku toolbelt and log in

To use the Heroku service, you'll need to create an account on their website and install software that makes it easy to interact with their service.

Read their instructions to do that. Be sure to configure your SSH key with the service. (If you need help with that, read their docs or find OpenHatch people on IRC.) Finally, make sure you have run the "heroku login" command.

If you don't see instructions for your operating system, look in this page's *Troubleshooting* section.

Create a Heroku app

On the Heroku service, individual sites are called "apps". You'll need to create an app corresponding to the code you want to deploy there. At the time of writing, you are permitted to create an unlimited number of apps for free. Therefore, I personally recommend creating an app whose name is similar to the branch name on your computer.

This app name appears in public as part of the domain name, so choose something you don't mind other people reading! (If you leave out the app name, Heroku will pick a random cute name for your app.)

On your computer, within a terminal, change directory into your clone of oh-mainline. You'll use the "heroku" command to create your app. (In the example here, I've named my app "openhatch". Substitute your own app name!) So, type something like this:

```
$ heroku create openhatch
```

You should see this output:

```
Creating openhatch... done, stack is cedar
http://openhatch.herokuapp.com/ | git@heroku.com:openhatch.git
Git remote heroku added
```

Now push your local git repo to Heroku with this command:

```
$ git push heroku master
```

You should see this output:

```
Initializing repository, done.
Counting objects: 70870, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (23526/23526), done.
Writing objects: 100% (70870/70870), 78.90 MiB | 103 KiB/s, done.
Total 70870 (delta 43536), reused 70870 (delta 43536)

-----> Removing .DS_Store files
-----> Python app detected
-----> No runtime.txt provided; assuming python-2.7.4.
-----> Preparing Python runtime (python-2.7.4)
-----> Installing Distribute (0.6.36)
-----> Installing Pip (1.3.1)
-----> Installing dependencies using Pip (1.3.1)
       Downloading/unpacking psycopg2 (from -r requirements.txt (line 2))

...

       Successfully installed psycopg2 PIL
       Cleaning up...

-----> Discovering process types
       Procfile declares types -> web

-----> Compressing... done, 70.5MB
-----> Launching... done, v5
       http://openhatch.herokuapp.com deployed to Heroku

To git@heroku.com:openhatch.git
 * [new branch]      master -> master
```

Because of the large size of the OpenHatch git repository, the first git push may take a few minutes.

As you make local changes, you can just use the same "git push" command to update the code on Heroku. Note that if you end up rewriting history, you may need to add a plus sign to the above command, e.g.:

```
$ git push heroku +HEAD:master
```

Set up the database

Now that your version of the OpenHatch code is on Heroku, you'll have to initialize the database that Heroku automatically created for you.

Now, initialize the database with:

```
$ heroku run python manage.py syncdb --noinput
```

More information from Heroku will scroll by. You may notice

```
( INFO     Some parts of the OpenHatch site may fail because the lxml
  library is not installed. Look in ADVANCED_INSTALLATION.mkd for
  information about lxml )
```

At this point, this is not an issue.

You'll also need to run the migrate command:

```
$ heroku run python manage.py migrate
```

Visit your app on the web

Now you can go to the Heroku URL for your app. If you're not sure what that URL is, you can type:

```
$ heroku apps:info
```

Look for the "Web URL" at the bottom of the output, and visit that in your web browser.

Now, celebrate! Your OpenHatch instance is on the web. Go get yourself a strawberry smoothie (making substitutions as necessary for your dietary restrictions).

Troubleshooting

- If Heroku doesn't have instructions for your operating system, and you have a package manager, try installing *git* and *rubygems* from your package manager. For example, on Fedora and other systems that use yum, you could type:

  ```
  $ sudo yum install git rubygems
  ```

  Then run:

  ```
  $ sudo gem install heroku
  ```

- You can verify the status of your application with:

  ```
  $ heroku ps
  === web (1X): `./mysite/manage.py runserver 0.0.0.0:$PORT`
  web.1: up 2014/01/04 13:48:55 (~ 17m ago)
  ```

- From time to time things might not work as expected. In those times, Heroku provides with a nice log facility:

  ```
  $ heroku logs
  ```

  More documentation on how to use Heroku's log facility and processes is available to you.

Maintenance tasks

The OpenHatch web app has some built-in features to help you maintain an instance. This file lists those features and how to use them.

Importing data snapshots

If you want your site to have a database filled with data like what is on the main OpenHatch.org site, you can import a data snapshot.

See https://openhatch.org/wiki/Importing_a_data_snapshot for more information about that. You can read our privacy policy at https://openhatch.org/policies-etc/.

How to run the bug importer

This can be done via cron job.

Run the hourly tasks related to profiles

There's a management command that runs necessary maintenance tasks. At time of writing, it tries to keep our cache of recommended bugs more or less up to date with the state of the bug table in our database.

To run this maintenance task hourly, run these commands:

```
# The following use of GNU screen is helpful for running maintenance tasks,
# but is not necessary.
screen -RD   # Create an instance of screen, or attach to an existing one.

# Type Ctrl-a, c to open a new screen
while (true); do ./manage.py profile_hourly_tasks; sleep 1h; done

# Type Ctrl-a, d to hide ("detach from") the screen
```

Adding jQuery UI components

When you want to add a jQuery UI component, go to http://jqueryui.com/download

Select the following options, plus yours:

- Sortable
- Accordion
- Tabs
- Progressbar

Note that the site will automatically select any dependencies (like jQuery UI's "Core").

First, under "Theme", select "No Theme". Under "Version", select "1.7.2 (stable release, for jQuery 1.3.2). Then click your little cartoon hand on the Download button.

Unzip the file in /tmp/, and just extract the file js/jquery-ui-1.7.2.custom.min.js, and cp it to mysite/static/js/. That will overwrite the existing jQuery UI bundle.

Be sure to check using git diff that the change you've introduced in git's view of that file is exactly what you expect.

Finally, don't forget to add your component to the list above, so the next person does the right thing.

Editing the website's CSS

The CSS files for OpenHatch repository can be found in the static folder of the oh-mainline repository: https://github.com/openhatch/oh-mainline/tree/master/mysite/static

These CSS files has been written in Less format. Not sure what Less is? Read the official documentation.

Adding a new bug tracker via git

(You can also add a bug tracker via the website.)

Clone the repository

You will need to have a local copy of our git repository. You can read about that in the *Getting Started*.

You will also need a local copy of a sister project, "oh-bugimporters". You can get that from https://github.com/openhatch/oh-bugimporters .

Overview of steps

You will need to achieve all the following things:

- Adjust oh-bugimporters so that it generates output with data from your bug tracker of choice.
- Ensure the data imports properly, by running:

  python manage.py import_bugimporter_data < output_from_bugimporters.jsonlines
- Ensure the web UI shows an option for the new kind of bug tracker. To do that, take a look at mysite/customs/models.py.

In a little more detail

- Clone oh-bugimporters to your local machine, and oh-mainline in parallel, such that both folders are on the same level in the directory hierarchy.
- Initialize oh-bugimporters with:

  ```
  virtualenv env
  ```

  and:

  ```
  env/bin/python setup.py develop
  ```

  See also oh-bugimporters/docs/intro.rst for more infos about how to setup the subproject of oh-bugimporters for development and testing.
- Now add your new tracker to the bugimporter folder, by using one of the existing variants as template. If possible, add proper tests too.
- Create a testfile with basic queries and then run a command like:

  ```
  env/bin/scrapy runspider bugimporters/main.py -a input_filename=/tmp/input-
  ↪configuration.yaml -s FEED_FORMAT=json -s FEED_URI=/tmp/results.json -s LOG_
  ↪FILE=/tmp/scrapy-log -s CONCURRENT_REQUESTS_PER_DOMAIN=1 -s CONCURRENT_
  ↪REQUESTS=200
  ```

  in the oh-bugimporters folder. Here /tmp/input-configuration.yaml is the prepared input file with the following basic content (may vary, depending on the bugtracker's implementation):

  ```
  meta: {limit: 500, next: null, offset: 0, previous: null, total_count: 1}
  objects:
  - base_url: !!python/unicode 'http://scons.tigris.org/issues'
    bitesized_text: !!python/unicode 'Easy'
    bitesized_type: !!python/unicode 'key'
    bugimporter: !!python/unicode 'tigris'
    custom_parser: !!python/unicode ''
    documentation_text: !!python/unicode 'documentation'
    documentation_type: !!python/unicode 'subcomp'
  ```

```
  existing_bug_urls: []
  get_older_bug_data: null
  queries: [!!python/unicode 'http://scons.tigris.org/issues/xml.cgi']
  tracker_name: !!python/unicode 'SCons'
```

After the run, check the log files /tmp/scrapy-log and /tmp/results.json for correct results.

- For the website part (note how we're switching to the oh-mainline folder now) you have to initialize your local installation of OpenHatch with the command:

```
python manage.py syncdb --migrate --noinput
```

- Once you have changed the files mysite/customs/forms.py and models.py to add your new tracker type, you have to recreate the migration scripts for the customs folder. So call:

```
python manage.py schemamigration customs --auto
```

  See also the page https://openhatch.org/wiki/Making_schema_changes for more infos on managing and updating schema changes.

- Now you can start the local OpenHatch site with:

```
python manage.py runserver
```

  and direct your browser to it at http://localhost:8000 .

- Add a user and your project, and setup the new bug tracker for it, as you would do normally.
- Ensure that the base folder for temporary import files is writable for your current user. The default folder as used in ./run_importer.sh is:

```
/var/web/inside.openhatch.org/crawl-logs
```

- Patch the import script ./run_importer.sh and change the URL for the OpenHatch site from "https://openhatch.org/..." to "http://localhost:8000". Otherwise, the run_importer script tries to download and update all bugs that are currently tracked at the real website...which might take a little while.
- Run the import script:

```
./run_importer.sh
```

  and wait for it to finish. Then reload the browser page and check that the bugs have indeed been imported properly.

- If you mixed things up, you can reset the database completely at any time with:

```
python ./manage.py reset_db --router=default
```

  This will leave you with a blank OpenHatch instance, without any users, projects or bugs. Then rinse and repeat the steps above...

If you get stuck, please email the list or ping paulproteus or others in IRC!

Submit a patch

This is the easiest part. See *How we handle contributions*!

Developer Notes

This section of the documentation is intended to provide developers with technical information that is helpful to know. This information will typically only be necessary for the short term (less than 6 months) as developers collaborate on larger issues that require a longer development timeframe.

We make this information available since it will likely be useful beyond the scope of one individual issue in our GitHub issue tracker.

January 2015 Notes

DataImportAttempts (DIAs)

DataImportAttempt is a model that contains metadata about when the profile importer (the thing that attempts to automatically fill in your list of projects you've contributed to, by letting you type in a query like 'asheesh@ahseesh.org', and then the code will trigger some background HTTP GETs to other services (see – an attempt to import data!) and log a note about this attempt in a model called DataImportAttempt. We no longer do automatic profile importing, so we don't need DIAs.

Twill

Twill being phased out of our tests and is being replaed with the use of webtest.

Ohloh

For project icons, we will be moving away from the use of Ohloh (now know as OpenHub). Ohloh is not used for any other purpose in the OpenHatch codebase. We will refactor the way project icons are fetched so we get them a different way than through Ohloh.

Operations Guide

Here's more information about our project structure that may be useful.

Monitoring

The basics

- *linode.openhatch.org* is the main OpenHatch box, which runs the website.
- *linode2.openhatch.org* is the secondary server for OpenHatch. It hosts Nagios!
- *vm3.openhatch.org* is a third server, hosted at GPLHost, that runs the Jenkins continuous integration server.
- The Nagios configuration is owned by a user called *nagios* on *linode2.openhatch.org*.

Access

- We use ssh keys for login.
- If you want SSH access to that account, file a bug requesting it, and attach an SSH key. You should hear back within 2 days; if you don't hear back by then, try to find paulproteus or jesstess on IRC.
- Then you can do:

```
ssh nagios@linode2.openhatch.org
```

- You'll know it's working if you are logged in. If you see a "Password:" prompt, then it is not working.

Notifications

- Nagios notifications go to monitoring@lists.openhatch.org. Anyone can subscribe to this list or read its archives.

Making changes

In brief, here's what you need to know:

- Edit files in ~nagios/
- Once you know what changes you want to make, create a local branch with those changes:

```
git checkout -b my_changes
```

- As you make changes, make meaningful commits. Also, tell "git commit" to use your identity:

```
git commit --author="Some Body <some.body@example.com>"
```

- After you have made the changes, ask someone to review them and merge the changes to *master*.
- **Rationale**: If you stick to the above process, it is fairly easy to roll back to the "master" branch of the Nagios configuration.
- **History**: We came up with this process during issue332.

Viewing the web interface, and handling the daemon

- On *linode2*, *~nagios/secrets/* contains the mailman and Nagios web interface passwords.
- View the Nagios web interface at http://linode2.openhatch.org/nagios3/
- To restart the Nagios daemon, run

```
sudo /etc/init.d/nagios3 restart
```

In case of emergency

- See *Emergency operations for the OpenHatch server*. People with ssh keys set up for the Linode Shell (Lish) can reboot the box and have other limited emergency capabilities.

TODOs

- Send Nagios notifications to IRC (*#openhatch-auto*?)?
- Make the Nagios web interface world-viewable.
- Version the monitoring configurations.
- Send SMS alerts to people who want them.
- Add historical trending (Munin)?

Related

- See also *Emergency operations for the OpenHatch server*
- See also the page about the *Login Team*

Continuous Integration

Overview

The OpenHatch code has a suite of tests. It's important that when we deploy the code changes to the website that all tests are passing.

Continuous integration helps our developers see if their code changes are passing all tests or are failing a test and additional code changes are needed.

Travis CI

Travis CI is a hosted, distributed "continuous integration" system (read more on Wikipedia about Travis CI). The GitHub page for the *oh-mainline* indicates whether our tests currently are passing.

Using Travis CI

There are multiple ways that Travis CI communicates the source code's current build status and whether tests are passing:

- The first is the "build" badge on the *oh-mainline* GitHub page displayed at the top of the README. Clicking on the "build" badge will display Travis CI's status page for OpenHatch.
- OpenHatch's Travis CI status page can be directly found at https://travis-ci.org/openhatch/oh-mainline.
- GitHub also provides information on every pull request about Travis CI's testing and status related to the individual pull request. This is very helpful for developers and reviewers.

Note: Currently, Travis CI is showing that our tests are not passing when tested with a MySQL database. Details can be found in the OpenHatch issue tracker. We hope to have this issue resolved soon.

Configuration for Travis CI

The *.travis.yml* file in the *oh-mainline* directory contains configuration information used by Travis CI.

Jenkins

Jenkins is a "continuous integration" tool (read more on Wikipedia). It wakes up once an hour, checks the git repository for new commits, and runs the test suite. For additional information about Jenkins, read more on Jenkins.

Status information about continuous integration projects can be found on OpenHatch's Jenkins dashboard : http://vm3.openhatch.org

Jenkins configuration

There are a number of "projects" in Jenkins. Different ones run different suites of tests in the OpenHatch codebase. They include or exclude different Django apps from the OpenHatch codebase.

For example,

- Test the "installation" instructions
  - This tests the OpenHatch developer instructions for building OpenHatch.
- Test the "customs" app
  - The tests for the customs app often go out to the network and can break if the remote servers change their APIs.
- Test the "search" app
  - The volunteer opportunity finder ("search") tests can take a while to run, so we separate them out.
- Test all apps except customs and search
  - This is the catchall that tests the rest of the code.

Status information about continuous integration projects can be found on OpenHatch's Jenkins dashboard.

Jenkins administration

Right now, only Raffi and Asheesh can modify the configuration of Jenkins.

Anyone can enqueue a run of the test suite by clicking a "Build" link within a Jenkins project. That's a good thing.

Future work

It would be super nice if, whenever there was a commit to GitHub master that passed all the tests, it would be automatically deployed.

Backups of the live site

Overview

We have a free, donated account from rsync.net that lets us store 50GB of data there.

We use duplicity (as per the rsync.net official document). We do full backups weekly and incrementals daily. We encrypt these backups.

The only server essential to continued operation of the site is linode.openhatch.org. The other servers do unimportant things that do not keep state. It would be convenient to have backups for them, but it is not essential, so for now I suggest we simply skip it.

Details

We use this script to run backups. It runs via root's crontab, and emails the results to Asheesh daily.

- do_backup.sh: in git

Restoring

duplicity has a built-in "verify" feature, which checksums the data, but that doesn't help us ensure that our backup was complete.

Therefore, weekly, we automatically restore and test the virtual machine, via a Jenkins job. http://openhatch.org/bugs/issue530 describes that.

More info about encryption

This backup is encrypted with a GPG key that has been emailed to hello @openhatch.org on Thu, Jan 26.

Emergency operations for the OpenHatch server

The main OpenHatch server is a virtual machine hosted by linode.com.

What to do when the site isn't working

- Check if SSH is alive

  ```
  telnet linode.openhatch.org 22
  ```

  You should get a banner message. If so, things are not *so* bad. Someone with root (like Asheesh/paulproteus) can probably SSH in and figure out what's going on.

- If SSH is not alive, and the website is down... **Find Asheesh**, if possible. Otherwise, well, you might want to know about Lish.

Lish: Emergency reboots, and more

If you can't load the website, and if the Linode doesn't even respond to SSH, then people with access can connect over the Linode Shell and read console messages or reboot the virtual machine.

If you want to help us by being part of an emergency crew who can reboot it, see the next section.

PLEASE do not reboot the machine without getting in touch with paulproteus (Asheesh), unless it's *clearly* a good idea to reboot it!

- http://library.linode.com/troubleshooting/using-lish-the-linode-shell
- Lish via SSH
  - ssh linode22043@atlanta76.linode.com
  - Lish listens on ports 22, 443, and 2200

To get your key in the list

- File a bug, and assign it to paulproteus.
  - The subject should be, “Add my SSH key to lish for linode.openhatch.org”
  - Explain who you are and why it is a good thing for you to be able to see the “physical console” of the virtual machine.
- You should hear an answer back within 2 days.

WordPress theming

Overview

The OpenHatch blog is powered by WordPress. This Django-based codebase has some minimal hooks that enable us to style the WordPress blog by making changes to this Django codebase.

Details

On a local instance, if you visit http://localhost:8000/+theme-stubs/wordpress/index , you will see an amusing absurdity: a Django template has been rendered, but the template blocks have been filled with placeholder strings.

The purpose of this page is to provide a machine-readable version of our theme which can, in turn, be processed by a separate engine to be turned into a WordPress theme.

(In the future, we may use this to generate a MediaWiki theme... and maybe a Roundup theme? Who knows.)

It is controlled by the template in mysite/base/templates/base/wordpress_index.html.

One thing to note: When exporting the page for use with WordPress, make sure your settings are configured to set DEBUG to False, or else every WordPress user will get a copy of the Django Debug Toolbar. This is not actually a problem, just an amusing fact. (TODO: When django-debug-toolbar gets this pull request landed <https://github.com/django-debug-toolbar/django-debug-toolbar/pull/303>, we can use that in the instructions.)

Editing the Wordpress CSS

If you wish to change the blog’s appearance, you may need to edit the CSS file here: https://github.com/openhatch/oh-mainline/blob/master/mysite/static/css/blog-style.css

Related

- See also https://github.com/paulproteus/oh-wordpress-theme , the project with code and documentation on generating a fully-functional WordPress theme from this page.

Front-end style guide

This **style guide** covers the interaction of HTML, CSS, and JavaScript on OpenHatch’s main website.

The “front end” refers to what people see in their web browsers. We create that experience using HTML, CSS, and JavaScript. We tend to use the jQuery library so we write less JavaScript, and we try to follow good conventions.

This document contains links to high-quality style suggestions from others, and also names some common problems that have occurred in the OpenHatch code in the past.

NOTE: Most of the HTML/CSS advice applies to the upcoming site redesign, so it may not cross-apply well to the live site for the moment.

HTML/CSS

Colors

Main background: lightest grey, #f8f8f8, with light-hatch.png background image. Header and footer: dark grey, #333, with dark-hatch.png background image.

Default text color: darkest grey, #222; black is used sometimes for emphasis.

Links: orange, #FF6D3D; white; darkest grey, #222;

Links don't ever change color; on mouseover, they get underlined.

Borders: translucent light blue, rgba(100, 200, 255, .3); dashed light grey, #e4e4e4;

Module interiors are slightly translucent white: rgba(255,255,255,.6); Occasionally (e.g. the front page) a module can have a full-white interior for emphasis.

Try to avoid font-weight: bold; if possible; differentiate headers and so forth by size, or maybe color, instead.

Try to store styles in our CSS files or LESS files, rather than inline in the element.

Cartoons

Cartoons are always 141px high. They should always be flush with the module beneath them. They are only used on one-column pages (even if there are multiple-column areas farther down the page, the first module should be a full-width module).

Layout

There are two base template layouts for pages: one-column and two-columns. Two-columns has a left 1/3 column and a main 2/3 column, cut on the same lines as a three-columns outline.

The template that a page uses should be based on what the first module set on the page looks like. If you want to add more columns on a one-column page, just create the appropriate divs. If you want one or three columns on a two-column page, put them inside the {% more_content %} block.

Two column CSS layout:

```
<column column-left> <column column-right>
```

Three column CSS layout:

```
<column three-column> <column three-column> <column three-column three-column-last>
```

Modules and submodules

Modules have the following structure:

```
<module>
    <module-head>
    Optional.
    The name of the module goes here, inside a h3.
    Inside of the h3 tag, you can make the title a link, if you want.
    </module-head>
    <module-body>
    Has a white background, the module contents all go in here, including submodules.
    </module-body>
    <module-foot>
    Optional.
    Want one of those clever links below your module content on the bottom right? Put␣
↪it in here.
    If you want a link to also appear on the left, give it the "module-foot-left"␣
↪class.
    </module-foot>
</module>
```

Submodules have the following structure:

```
<submodule>
    <submodule-head>
    Optional.
    This is where the title for the submodule goes, if you want to differentiate it␣
↪somehow.
    Styling for this isn't standardized yet... see /missions/ for an example.
    Recommended to put the title inside an <h3>.
    </submodule-head>
    <submodule-body>
    Also optional.
    If you don't want to bother with a head/body distinction, you can just put␣
↪content straight inside the submodule.
    </submodule-body>
</submodule>
```

(all tags are the class names of the relevant divs, unless otherwise stated)

You may want to put your submodules inside a <submodule-container> with a <clearer>, if you're floating them but don't want other content to ride up.

CSS

Name IDs and classes using hyphens, not underscores or camelcase. (e.g. "#front-page", not "#front_page" or "#front-Page"). CSS file names should use the same convention. (Not all of them do, but hopefully that can eventually be corrected.)

Usage instructions for the code, pre and tt tags

If you're interested in modifying the 'Hints' sections of the training missions, here are some guidelines regarding the usage variations of the code, pre and tt tags to keep in mind.

The CSS properties for the pre tag and the code tag are such that a pre element has a newline before and after it and is on a newline itself and a code element does not have a newline before or after it but is on a newline itself. A tt element neither has a newline before or after it nor it is on a newline itself.

Note: As of HTML5, the tt tag has been deprecated.

Here are a few examples:

If you write:

```
<p>If you are on Linux, type: <opening tag>man diff<closing tag> at the command line.
↪</p>
```

If you replace the "<opening tag>" and the "<closing tag>" above in the code with pre tags, the output would be as follows:

```
If you are on Linux, type:

man diff

at the command line.
```

If you replace the "<opening tag>" and the "<closing tag>" above in the code with code tags, the output would be as follows:

```
If you are on Linux, type:
man diff
at the command line.
```

If you replace the "<opening tag>" and the "<closing tag>" above in the code with tt tags, the output would be as follows:

```
If you are on Linux, type: man diff at the command line.
```

Editing the training mission hints

While working on Issue 958, it was found that in the hints for training missions, the "low" hint sometimes had trouble laying out its child elements properly due to the CSS properties of its parent div and this caused the "low" hints to display weirdly. To fix this issue, a new CSS property for '#low' was added to mysite/static/css/missions/base.css. However, here are some guidelines for how to phrase things when editing the training mission hints:

The first sentence of a training mission hint should be a paragraph (p tag).

The first sentence of a training mission hint should be a full sentence.

Full sentences start with capital letters and end with periods.

JavaScript

This is a list of strategies for avoiding problems that have plagued OpenHatch code in the past. **Note:** that the OpenHatch code does not yet follow this guide. It ought to. Perhaps it can be a release goal in the future.

If it's not a link, don't make it a link

If there's no fallback for non-JS users, don't use the <A> tag.

Issue 478 covered a problem where a user was clicking on what appeared to be a link. Because there is no JavaScript equivalent for the functionality the user clicked, it simply should not be a link. (Though style-wise it may **look** the same to the user.)

Don't rely on "return false;"

It is easy to mistakenly use "return false;" at the end of a JavaScript callback when you really mean event.preventDefault(). You can read more about this problem.

Good references

This document by isobar looks great.

Web API

The basics

The OpenHatch Python code provides some basic APIs that can be used by JavaScript on the web. Currently, these APIs are only used by JavaScript within the OpenHatch site itself. In the future, people might want to re-use our data in off-site JavaScript apps, and we hope to enable that.

This page documents the data export APIs that exist.

API v1 Profile data

The URL /+api/v1/profile/ is a RESTful API base URL. It has one endpoint, portfolio_entry.

You can find out more about it interactively in your web browser, by visiting a URL like http://127.0.0.1:8000/+api/v1/profile/portfolio_entry/?format=json .

Additionally, if you have "cURL" (a common web page downloading tool), you can run this command from your computer's command prompt:

```
curl http://127.0.0.1:8000/+api/v1/profile/portfolio_entry/
```

Note that with curl (and with AJAX clients), you can (and should) omit ?format=json.

This exports data from the *PortfolioEntry* model in mysite/profile/models.py. You can read the detailed code and configuration behind it at mysite/profile/api.py. The API is powered by the Tastypie Django app.

More information:

- Django Tastypie documentation: http://django-tastypie.readthedocs.org/en/latest/toc.html

Checking Coding Style Errors in Pull Requests with lint-review

Overview

We use lint-review, an automated code linting bot that checks pull requests for code style errors (such as pep8 violations). It uses the Github API to fetch the changes, runs linters against them and comments on the pull request if any code style errors are there in the changes.

Configuration

The `.lintrc` file stores the configurations for the lint-review bot. You can find the `.lintrc` file for this project here.

Issue tracking using GitHub Issues

Overview

We use GitHub Issues for tracking issues. The Managing Projects section of GitHub Help provides useful information about filtering, sorting, assigning, and labeling issues.

Issue labels

We use *Labels* to categorize an issue's priority, status, and type as well as other helpful user information. The *Bitesize* label is used to indicate that the issue is suitable for a new contributor to the project.

Historical note on issue tracking

Prior to using *GitHub Issues*, we used two other issue trackers. To preserve historical information, issues that were imported from past trackers can be identified by the user creator (@imported-from-roundup in the oh-mainline repo and @bot-sunu in the oh-bugimporters repo).

Security Considerations for the OpenHatch Application

Policy

OpenHatch will make its best effort to protect our users, including their accounts and any identifying or personal information they store with us. To that end, we care deeply about security issues that result in compromises to:

- Authentication
- Authorization
- User accounts
- Cross-site scripting attacks
- SQL injection attacks
- any other vulnerability that affects our data or our users

There are certain classes of issues that we de-prioritize, because they do not materially affect what we are trying to protect as listed above. These classes include, but are not limited to:

- Open redirections in OpenHatch applications

Known Issues

- It is known the the OpenID registration flow has the possibility of an open redirect to another site. Since it would be non-trivial to exploit this vulnerability in a way that would compromise the user, we are erring on the side of having a less complicated code base and leaving the bug unpatched. If a future version of one of our vendor libraries patches the bug, we may upgrade the library to close the open redirect.

Sections to add:

- Adding a dependency
- A tour of the templates
- Automated testing

Tutorials

Here are some tutorials that may help you use the site.

Writing Training Missions

Training Missions are tools which help people learn the skills needed to contribute to open source projects without burning their fingers. They can be background information (Windows Setup), or more involved (Subversion Training, which creates repositories for each user to work on).

Missions are made up of some Python code ("views"), and some HTML ("templates"). OpenHatch uses some conventions to make developing new training missions as simple as possible.

Note: This document currently covers Simple (non-interactive) Training Missions. We are working to update it to describe interactive missions, as well.

Simple Training Missions

Simple Training Missions provide users with step by step documentation. As an example we'll start a new mission to document how to make missions (how meta!), cunningly named "Make a Mission".

To begin developing a new mission, we need to create two directories: one for the mission code, and one for the mission templates. Both are stored in sub-directories of the OpenHatch repository (`oh-mainline`; see *Getting Started* for more information).

To create the new directories:

```
$ cd oh-mainline/mysite/missions
$ mkdir makeamission
$ mkdir templates/missions/makeamission
```

The first directory we create is the code directory. The second is the template directory.

It will help other developers work on your code if you use the same name for both directories. Note that we're making directories named `makeamission`, not "Make a Mission". That's because the directory name needs to be a valid Python package name, which means no spaces or non-alphanumeric characters.

Within the code directory (`makeamission`), you'll need to create an empty file named `__init__.py`; this tells Python that this directory is a package. On Mac OS X and Linux, you can do this by running the `touch` command:

```
$ touch makeamission/__init__.py
```

On Windows, just create the file with your favorite (or second favorite) text editor.

Defining Steps

Missions are made up of a sequence of steps: each step has a little bit of Python code and an HTML template.

Note: It may be helpful to refer to existing Mission code as you work on yours: examples are a great way to learn. The *Windows Setup* mission (in the `setup` directories) is an especially simple example.

Create a file named `views.py` in your code directory; we'll start writing the Mission steps there.

The simplest type of Mission Step just renders a template. That's what we'll start with. In `views.py`, start with the following code:

```
from mysite.missions.base import MissionBaseView


class StepOne(MissionBaseView):
    url = '/'
    template_name = 'missions/makeamission/index.html'
    view_name = 'main-page'
    title = 'Setting up Your Mission'
```

This Python code defines the first step in our new Mission. There are a few things to note:

- `StepOne` is the name of the step's *class*. This needs to be unique for your steps in the Mission.
- `MissionBaseView` is the basic building block of Mission Steps and provides some helpers you can use in the template.
- The `url` defines what the URL of this mission will be **within the Mission**. This must begin with a `/`.
- The `view_name` (`main-page` in this case) allows us to refer to this view in other views.
- The `title` will be displayed in the sidebar of the Mission.

After you've added the first step, let's add a second step:

```
class StepTwo(MissionBaseView):
    url = '/templates'
    template_name = 'missions/makeamission/steptwo.html'
    title = 'Writing Mission Templates'
```

Note that here we've omitted `view_name`. When it's omitted OpenHatch uses the `url` setting (omitting the leading `/`) for the view name.

Step Templates

Each of the Steps we've defined refer to a template: these templates will contain the text content for each step. OpenHatch uses Django templates, which add some logical functionality to plain HTML.

Let's start with the first step, `StepOne`. In a new file (`templates/missions/makeamission/index.html`), add the following content:

```
{% extends 'missions/mission_base.html' %}
{% load base_extras %}

{% block mission_main %}
<div class='submodule fat'>
  <div class='head'>
    <h3>{{ title }}</h3>
  </div>
  <div class="body">

    <p>Real content here, please!</p>

    <p class="next_mission_link">
        <a href="{{ next_step_url }}">Go forward and make a template!</a></p>
  </div>
```

```
</div>

{% endblock mission_main %}
```

There are a few interesting things here:

- The first line tells OpenHatch that this page should be based on the common Mission template.
- `{{ title }}` and `{{ next_step_url }}` are substitutions: Mission views provide several conveniences so you don't have to repeat yourself. These include `title` (the step title), `next_step_url` (the URL of the next step), and `prev_step_url` (the URL of the previous step).

The template for the second step should be named `steptwo.html`, which is what you specified in the class (`StepTwo`) above.

Mission Information

Missions are made up of a sequence of steps, so we need to define what order those steps come in. Missions also have some information of their own, like their name and an identifier.

We'll define the sequence of steps and the metadata by adding the following to our mission's `views.py`:

```
class MakeAMission(Mission):

    mission_id = 'make-a-mission'
    name = 'Writing New Missions'

    view_classes = (
        StepOne,
        StepTwo,
    )
```

We also need to modify the import at the top of that file to read:

```
from mysite.missions.base import Mission, MissionBaseView
```

Making it Accessible

The final step to writing your Mission is to make it accessible on the site by telling OpenHatch how to route the URLs. Django projects define URL routing in a file cunningly named `urls.py`. You can find this in the `mysite` directory. You can begin by opening `urls.py`. You'll need to tell it where the file `views.py` for your new mission lives by adding an `import` statement near the top of `urls.py` right after the `import` statements for the existing training missions like so:

```
import mysite.missions.makeamission.views
```

In `urls.py`, you'll also find a list of URL patterns – regular expressions which Django will use to match URLs and figure out where to send requests. Finally, add the new mission by adding a new item after the other missions:

```
(r'^missions/makeamission',
    include(mysite.missions.makeamission.views.MakeAMission.urls())),
```

Two important things to note:

- `makeamission` in the `include` and `import` statements above refer to the directory you created, so you'll need to make sure the name matches.

- `MakeAMission` is the name you give your Mission class.

Once you've added it to the URLs, you can start the server and visit http://localhost:8000/missions/makeamission/ to see your new mission!

Adding a new bug tracker

One of the pillars of OpenHatch is helping project owners get in touch with developers willing to help, by offering them easy to fix bugs where they can get started. In order to do this, OpenHatch crawls bugtrackers of many projects and features them in the site.

If you have a project that you want to add to OpenHatch you can do it very easily following these steps.

- Go to http://openhatch.org/customs/
- In the "Tracker Type" select box select the bug tracker that your project uses. If your bug tracker is not in that list, it may be a good a idea to contact us and let us know.
- Click on "Add a tracker" to add a new tracker
- You will be directed to a form where you need to fill specific information about your tracker. This form changes depending on the tracker type.

For Trac and Roundup:

- **Fill the new tracker form like this:**
  - Tracker name: Name of your project
  - Base url: This is the URL to the homepage of the Trac tracker instance. Remove any subpaths like 'ticket/' or 'query' from this.
  - Bug project name format: This field contains instructions of how to fill it, but if you are not sure, probably {tracker_name} is the best option.
  - Bitesized type: Choose the field that can help identify a bug as bite-sized. If you don't find the field you need in the select box, you may have to contact us to add it.
  - Bitesized text: The value that the Bitesized type contains for a bite-sized ticket.
  - Documentation type: Ditto Bitesized type.
  - Documentation text: Ditto Documentation text.
  - As appears in distribution: For most cases you can leave this field empty.
- **Click "Next" and you will be directed to another form:**
  - Url: Enter the URL of the CSV representation of a search result that only returns bitesized bugs for your project. Here is an example of how the value for that field looks for the GHC Tracker (Trac): hackage.haskell.org/trac/ghc/query?status=new&group=difficulty&format=csv&order=id&difficulty=Easy+%28less+than+
  - Description: Enter a description of the results that the search query returns.
- Click "Finish".

Tutorials we hope to add:

- Adding a field to the profile
- Making schema changes

Community Guide

About the OpenHatch community

The OpenHatch website is an open source, free software project. There are many ways that individuals and organizations can make contributions to it. Here's what you can expect when attempting to contribute code and documentation.

We're friendly and respectful

The existing community is proud of its reputation for being friendly, welcoming, and helpful. We want to accept code, documentation, design, and other contributions from as many people as possible – that can include you!

We have people of all backgrounds in the project. Diversity is one of our project's strengths.

Generally, to contribute directly to the project's growth, you can submit patches or Github.com pull requests. You can read more about that.

Community structure

Here are some sub-groups of the OpenHatch community:

- Contributor: This refers to anyone who makes a contribution that gets put in the main OpenHatch git repository. To become part of this, just show up with a patch or pull request. (We'll help you do that on the mailing list or IRC.)
- Github commit group: These are the people who can push changes to our main git repository on Github. To become part of this, submit a few high-quality code reviews and ask the project lead for this power.
- "Login team": These are the people who can SSH into the main OpenHatch server. To become part of this, submit a few high-quality code contributions and then ask the project lead.
- "Project lead": Asheesh Laroia is the project lead. You can contact him with questions about the project.

Anyone can contribute to OpenHatch, through submitting a patch or pull request.

You might be interested to know that OpenHatch is also a non-profit, with its own governance. The non-profit and the code development team share some people, but they do not share a power structure.

Contact Us

Talk to us on IRC (#openhatch on irc.freenode.net). Not familiar with IRC? See **Chat with us on IRC** below!

Join our contributors mailing list.

Or say hi to us on the project blog, Identi.ca, Twitter, and Facebook.

Chat with us on IRC

You can usually find us on #openhatch on irc.freenode.net - a friendly place to say hi and get answers to questions interactively.

If you don't know how to use IRC, you can use our installation guide or just click on this link to a web chat interface.

Also, there's an excellent FAQ at botbot.me

Are you trying to figure out who everybody is? Many of the people are part of the OpenHatch project.

Quick start

```
/server irc.freenode.net

/join #openhatch

/nick newnickname

/me waves hello

# The /me is an action message.
# Type /me 'does anything'
# Example:  /me waves hello
# What it looks like: * bossmom waves hello
```

More details: http://www.ircbeginner.com/ircinfo/ircc-commands.html

Login Team

The **Login Team** is the people who can SSH into the main deployment of OpenHatch.

In order to subscribe to the monitoring-private email list or have push access to Github, you have to be part of the login team.

Current members

- Jack Grigg (pythonian4000)
- Asheesh Laroia (paulproteus)
- Jessica McKellar (jesstess)
- John Morrissey (jwm)
- Karen Rustad (aldeka)
- Elana Hashman (ehashman)

How to join

- Email asheesh at asheesh.org...
  - Include a copy of the OpenHatch Servers Access & Usage Policies at *Login Team Agreement*
  - "Digitally sign" it by typing your name at the end of it (and/or PGP sign, if you like)
  - (Here's a summary of the document: You agree to be nice, respectful, and communicative. Asheesh can kick you out of the login team, but he will also try to be nice, respectful, and communicative. If you find a security hole, or otherwise discover a bad problem, you will contact other people and let us know. If you take reasonable actions like trying to diagnose the security problem, that's cool. If you exploit a security hole to do something that you are actually permitted to do, then that's pretty smart and crafty of you, and you should tell you did it so we know how cool you are.)
- Then if Asheesh agrees you should be on it, he'll let you on. You should hear back within four days.

What you get, when you're on the login team

- SSH key access to the deploy user account on linode and linode2.
- When we have a VM hosted by OSU OSL, you'll get a personal account and sudo permission.
- SSH key permission to push to the Github openhatch/oh-mainline repository.

How to deploy new versions

- See *Deployment*

Login Team Agreement

How to use this document

If you want to become part of the *Login Team*, you have to take this document, copy it into an email, and send it to asheesh at asheesh.org.

Inversely, this document serves as a record to visitors to the site: you can read it to find out what have people agree to in order to get shell access.

—

OpenHatch Servers Access & Usage Policies

Version 1.1

Version 1.0 of the OpenHatch Servers Access & Usage Policies becomes effective on April 30, 2011. Version 1.1 fixes some trivial typographic errors.

Introduction

This document describes the policies for people who have direct (e.g., SSH) login access to any account on the OpenHatch Servers. When you are granted this, you become part of a group of people we call the Login Team.

General statement

Privilege

Access to OpenHatch servers is a privilege, not a right or a commercial service, and Asheesh Laroia and the Login Team reserve the right to revoke this privilege at any time, without prior notice. An explanation will be given within 48 hours. Asheesh promises to try to have a reasonable conversation with you about the loss of access, including (if you ask for it) the possibility of gaining this privilege again.

Guarantees

There is no guarantee of service. Although the Login Team will do its best to assure that everything functions perfectly, they can't give any guarantees.

Penalties

If someone violates the rules set out in Policies section of this document, as Asheesh's sole discretion, then Asheesh will revoke that person's access. See also the "Privilege" section.

Rules and Guidelines

The following section defines OpenHatch's Rules and Guidelines.

- The rules are binding and may not be violated.
- The guidelines specify rules that may be violated if necessary but we would rather one did not.

The Rules

You will not carry out any willful, deliberate, reckless or unlawful act, interfere with the work of another developer or jeopardize the integrity of data, equipment, systems programs, or other stored information.

You will not use the OpenHatch servers for financial gain or for commercial purposes, including consultancy or any other work outside the scope of official duties or functions for the time being, without specific authorization from Asheesh to do so.

You will not use OpenHatch servers for any unlawful activities whatsoever.

You will not deliberately interfere with or alter the integrity of the OpenHatch servers, by doing any of the following:

- permitting another individual to use your shell login account;
- impersonating other individuals in communication without their permission;
- attempting to capture or crack passwords or encryption, unless you have permission from the account-holder to do that;
- destroying or altering data or programs or anything belonging to other users, unless you are you doing something a user has explicitly permitted you to do.

You will not restrict or deny access to the system by legitimate users.

You will not transmit threatening or harassing materials.

When you copy private data onto a non-Login Team computer, for example user passwords that you copy through a MySQL dump, you promise to keep them safe on the computer you copied them to. You will try to delete them reasonably quickly, and should try to use the public data snapshots wherever possible.

You do not have to delete session IDs that come to you via the monitoring-private list, but you should not share them (because sharing session IDs can permit people to impersonate users).

You will not use OpenHatch servers in a manner which constitutes net abuse.

You will not allow any bot net or other criminal infiltration of the OpenHatch servers through carelessness or abuse, to the best of your ability. If you think you found or created a security hole, you will tell Asheesh. He promises to be nice to you in his acknowledgment of the issue.

The Guidelines

Processes: Do not run any long running process without the permission of Login Team. If you come up with cool ideas of for processes to run, you should have a conversation with the rest of the Login Team first. Running servers of any sort (this includes IRC bots) without prior permission from Login Team is also forbidden. Avoid running processes that are abusive in CPU or memory. If necessary Login Team will clean up such processes without warning.

WWW pages: In general, web space on the OpenHatch Servers is provided specifically for the purpose of serving the webapps of OpenHatch itself, whose goal is to help OSS projects get new contributors. Using your login account to create private 'vanity' pages on OpenHatch Servers is discouraged. (Obviously, the OpenHatch web app itself provides a way to create vanity pages; you may use that.)

Homedir: If you receive an email notification that your homedir is large and that more free space is needed then please promptly take action. The Login Team may find it necessary to clean up without warning.

Tunneling: If you need to use the OpenHatch servers as the end of a tunnel, like an SSH tunnel, try to talk to Asheesh first to make sure your use is okay. (If you had to do it in a hurry, try to tell him afterward.)

Mail/News: Don't use the OpenHatch servers for reading email. If you want to forward your email address somewhere else, that's fine; create a .forward file. If a developer becomes unreachable for a prolonged time, his/her accounts, data and mail forwarding/filtering/etc may be disabled until the person reappears.

I HAVE READ AND AGREED TO THE FOLLOWING TERMS AND CONDITIONS

(Erase this line, and type your name here, if you agree)

Web Analytics Team

This is the set of people who have access to look at aggregate logs of our page views. Because we give people access to aggregated information, we expect it not to be personally identifiable information, so the privacy implications should not be huge.

Purpose

When design or code or documentation or publicity or other contributors are thinking about what changes to make to the website, it is often very helpful to be informed by data about how people are using the site.

Who may join, and how to join

Any reasonably trustworthy contributor (such as: people who have shown up to the #openhatch IRC channel and seem to get along fine with other members; people who have contributed code to the project; or so forth) is welcome to join.

We might take your access away if you seem to not have been using it for one month. That is just constant housekeeping, not because we have stopped trusting you. If we do that, we will try to notify you (but might forget to), and you are free to ask for it back.

To join:

Email devel at lists.openhatch.org with a brief message, such as:

I'd like to join the Web Analytics Team because I want to know more about __________ and think that our aggregate page logs will help with that.

You should expect someone to reply to you within about 4 days with a yes or a no.

Technology

Right now, we use Google Analytics. You can log in here:

- https://www.google.com/analytics/

People on the Login Team are permitted to have "Manage"-level access.

People on the Web Analytics team are permitted to have "Edit"-level acces or "View"-level access.

The team isn't technology-specific. If we add another web analytics tool, we should give these people a similar level of access.

Members

Manage level:

- Asheesh Laroia

Edit level:

- Britta Gustafson
- Susan Tan

Domain Team

The **Domain Team** is the team of people who can modify settings of the openhatch.org domain, such as what IP address it points to. The team also has the same power for all openhatch-owned domains.

Since the team has somewhat limited membership, one goal is to be responsive to requests from non-members who want to add or edit our DNS settings.

Membership

It is difficult to delegate access very widely for this, as anyone with control over the openhatch.org DNS can probably gain any other privilege they want. Membership is open to Login Team members or Board members who merit a special degree of trust.

Current members:

- Asheesh Laroia (paulproteus)
- Shauna Gordon-McKeon (shauna)

How to join

Email asheesh at asheesh.org with at least one sentence to answer why you need domain/DNS-level access for the thing you're working within OpenHatch.

You should expect a response within 4 days. If you don't get one, feel free to send another email that is CC: devel@lists.openhatch.org.

What you get, when you're on the team

- CloudFlare DNS password. (Username is asheesh@openhatch.org for now.)

Information useful to members

- openhatch.org DNS is hosted by cloudflare.com.
- The openhatch.org domain is configured in gandi.net. Currently that still uses Asheesh's personal account, but we should change that.
- CloudFlare sends emails to asheesh at openhatch.org. We should change that.

Collaboration tools

This document lists some communication and collaboration tools that people in the OpenHatch community use in order to work together effectively. The purpose of this part of the documentation is to help people who are new to these tools quickly become oriented, and to help us share tips on how to use these tools well.

Freenode IRC

Developers, event organizers, documentation writers, bug filers, and basically everyone in the OpenHatch community sometimes uses IRC as a real-time chat system.

See https://openhatch.org/wiki/Contact to read more about how to join us there.

Email lists

We use the email lists at http://lists.openhatch.org/ (hosted by the free-software Mailman package).

These lists are typically publicly archived, and typically are open for anyone to join. Archives are currently performed by the "pipermail" software tool, which is widely-used but not universally loved.

Etherpad

Etherpad is a real-time, rich-text capable, free software text editor.

People discussing a topic on IRC often use it to think aloud so they can write more than a few sentences at a time, and to get feedback on an idea that is a paragraph or longer.

The Etherpad software is widely available; we typically use the websites running Etherpad hosted by Mozilla or Wikimedia. To create a new Etherpad document, one procedure is:

- Think up the name for the document (for example, oh-just-testing)
- Visit https://etherpad.mozilla.org/oh-just-testing
- Wait for it to ask you, "Do you want to create this pad?" and answer yes.

This way, you can choose the name of the pad, rather than accepting the default behavior of having a randomly-generated name.

We typically treat Etherpads as temporary storing places for documents, since it is easy to misplace the link to the document. Therefore, it's usually a good idea to export a document as HTML from within the Etherpad interface, and send that HTML document to the devel list.

Google Docs

Google Docs is a free-of-cost collection of collaboration tools for writing text documents, spreadsheets, and other such things.

We sometimes use that, too.

Quotes database

This document explains our IRC quotes database.

The URL to the quotes DB

http://boiling-refuge-7775.herokuapp.com/

How and when to submit a new quote

Let's say something interesting happens on IRC. For example, if your friend stops by the IRC channel and says something amusing:

```
<friend> hey, I just realized
<friend> oh-mainline... the oh starts for OpenHatch...
<friend> but it's also, like -- oh! mainline!
```

and you want to put it in the IRC quotes database. Here are the steps you take:

- Copy the text to your clipboard on your computer.
- Visit the quotes database at http://boiling-refuge-7775.herokuapp.com/.
- Click *submit quote*
- Enter your quote in the DB, and click *Submit*
- See the glorious "Your quote has been submitted and is now pending approval!" message
- Send a private message to paulproteus to ask them to approve it.

How to approve quotes

- Visit the login page at http://boiling-refuge-7775.herokuapp.com/?m=userlogin
- Log in (if you need an account, ask paulproteus)
- Visit http://boiling-refuge-7775.herokuapp.com/?m= and see the list of quotes.
- Notice that some are very pale! Either approve or delete them.
- To moderate, visit http://boiling-refuge-7775.herokuapp.com/?m=panel and enter the 4-character short name into the *permaid* box. Choose either *do_approved* or *do_deleted* to mark it respectively as approved or deleted.

Ticket tracking

This document explains how we handle inbound email to the OpenHatch organization.

Our "ticket tracker" is a tool we use to reply to emails people send to OpenHatch. The "ticket tracker" is a fairly internal tool, although if you're excited about getting involved, we welcome you! Crucially, it is distinct from our "bug tracker," which tracks issues in the open source code of the OpenHatch project. By contrast, the ticket tracker helps us keep track of email conversations.

The URL to the ticket tracking site

http://tickets.openhatch.org/rt/

Our normal workflow

Somebody sends an email to hello@openhatch.org. By default, this ticket goes into the “General” queue and is not assigned to anybody.

To move the conversation forward:

A ticket responder can reply by email with a helpful message, which changes the ticket status from “new” to “open”, or they can write a reply specifically to support-comment at tickets.openhatch.org to write a message only viewable by ticket responders.

On the RT website, a ticket responder can click “reply” to reply with a helpful message, or they can click “comment” to write a message only viewable by ticket responders.

To keep the tickets organized:

On the RT website, go to a ticket and visit the “basics” menu option (in the grey bar) to put the ticket in the appropriate queue, assign the ticket to somebody, and assign a new status to the ticket.

When a ticket is resolved, we should assign it the status “resolved”.

In the future:

We plan to also be able to assign queues, ownership, and statuses by email (using CommandByMail). We have a ticket filed with the hosting provider to set this up.

Workflow notes:

There are some slight differences from the default RT workflow, which might be worth highlighting in case you’re comparing this with past RT experiences or the official docs:

- We’ve disabled the “Auto-Reply to requestor” scrip. We thought this seemed impersonal; additionally, Asheesh was concerned it would cause us to send replies to spam we receive.
- We’ve disabled the scrip that causes the requestor to receive an email when the ticket is marked as “resolved”. We thought this seemed impersonal.

Request Tracker Admin Team

These people have “privileged” accounts in RT and the password to the separate “root” account within RT:

- Britta
- Asheesh
- Shauna

How to help out

If you want to help answer people’s questions that they email to OpenHatch, please email us and we’ll give you an account with some degree of appropriate privilege to do that. We haven’t configured this yet, but we’re excited to figure it out!

hello@ migration

(This section written on Sun Feb 16. Hopefully, it will go away by the end of February.)

Right now, hello at openhatch.org forwards to a bunch of people.

In the near future (~1 week from now), we'll turn that forwarding off. People who were on the hello@ alias can get something similar to what they used to have by requesting an RT account.

The best way to request an RT account is to email hello@, as we already have things set up so that emails to hello@ will create a ticket.

Importing past mails

(This section written on Sun Feb 16. Hopefully, it will go away by the end of March.)

We haven't imported any past threads or emails yet.

For bulk import of historical archives, Asheesh hopes to get to this in the future (the next 2 to 6 weeks). He'll make an "archive" queue that hopefully contains one "ticket" per email thread. That way, new email helpers can read the archives within RT.

If you want to reply to a hello@ mail that isn't in RT yet, try to use the following workflow:

- Forward the most recent mail in the thread to support at tickets.openhatch.org.
- Modify the ticket so the person who sent it is the "Requestor."
- (Optional: Move it into the appropriate queue.)
- Reply to the ticket via your email, now that it has a ticket number.

That's a bit cumbersome. Sorry about that.

How the ticket tracker is administered

- tickets.openhatch.org points at an IP address of a VM run by Gossamer Threads.
- They are generously donating their hosted RT service, and for this, we plan to thank them on the OpenHatch sponsors page.
- From what I understand, we have a virtual machine that they administer. We can file support tickets with them, and we can also FTP into the machine and make changes if we want.
- hello at openhatch.org forwards to a few email addresses, including bountyarchive at rose.makesad.us; a procmail rule there forwards the email further into our RT instance. This is hackish and should be replaced in the future. (This remark written on 2014-02-16.)

Further thoughts

Maybe it would be interesting for us to CC: the ticket tracker on emails to sponsors, generally. That way, we'd have a shared archive.

How to run an OpenHatch sprint

This is a brief set of steps for anyone interested in getting people together to improve the OpenHatch codebase or web app project. It is based on The In-Person Event Handbook. We call these "sprints", after the terminology at pythonsprints.com.

Step 1: Figure out what you want to accomplish at the sprint

Some ideas for this can be found in the "What do you want to accomplish at this event?" section of The In-Person Event Handbook.

Step 2: Pick a date & venue

The date and venue go hand-in-hand, as you might have to reserve space. See the Bay Area Debian Shotgun Rules for good things to think about. Power and wifi and laptop-friendliness are especially important.

Step 3: Find funding (optional)

As of the time of writing (December 2013), the Python Software Foundation is happy to sponsor food. Visit the Python Sprints Call for Applications for information on how to apply. It's nice to be able to say that food is sponsored, so people are more likely to attend.

In Asheesh's opinion, if food sponsorship is asked-for but not yet confirmed, it's fine to say in the announcement email that it's pending confirmation.

Step 4: Figure out the maximum number of people we can support

Given the goals you set, perhaps we only have mentorship resources available for 4 newcomers or so. In that case, it's best to ask people to RSVP to you, perhaps by creating an event page on eventbrite.com or by asking people to send a personal email to the event organizer.

Step 5: Get the word out

You should definitely send an email to the OH-Dev mailing list. It's also a good idea to send (even if brief) personal emails to OpenHatch contributors in your city.

Write your announcement in the style recommended by the Bay Area Debian Shotgun Rules.

Consider also Tweeting about it, or getting the pythonsprints.com blog to send your announcement, or sending an announcement to the Python Meetup in your city. (It is OK not to do that if you suspect it would result in more RSVPs than the event can support.)

Step 6: Read the In-Person Event Handbook and follow its instructions

Now is a great time to read the rest of The In-Person Event Handbook and think about follow-up processes, validating documentation, and the preferred modes of communication for attendees physically at the sprint.

Food sponsorship logistics notes

Typically, the Python Software Foundation (or any food sponsor) wants to see an itemized receipt with the food and drink items on it. If the sprint is at a venue where people typically order one item at a time, you can simplify life by leaving a tab open and having just one person pay the bill. Then that person can be the one to get reimbursed.

If a person with an OpenHatch Foundation card is around (such as Asheesh or Shauna), a good idea is to have that card be the one that pays the tab. Then the owner of that card is the only one who has to deal with the reimbursement logistics.

THANKS

- The Free Software projects that we depend on - see the vendor/ directory for a list.
- Our contributors and consultants are listed at http://openhatch.org/projects/OpenHatch. If you've helped to create OpenHatch, you deserve to be listed there!

- The fantastic people who have helped us:
  - Nelson Pavlosky
  - Shotput Ventures, who mentored and funded us in the summer of 2009.
  - The Institute for Security, Technology, and Society and the Neukom Institute, both at Dartmouth, who sponsored Parker Phinney's internship during the Dartmouth winter 2010 term.
  - Google Summer of Code, who sponsored John Stumpo's internship during the summer of 2010.
  - Jerold Camacho, who reported a cross-site scripting issue in 2013.
- See CREDITS for other files we use.

Contributor's Guide

The Contributor's Guide provides an introduction to contributing to the OpenHatch project.

Technical Guide to Development and Documentation

The Technical Guide provides information to contributors interested in developing software or writing documentation for OpenHatch projects.

Operations Guide

The Operations Guide gives an overview of the infrastructure and processes that keep the OpenHatch website running efficiently.

Tutorials

We encourage our contributors to write tutorials and share their process knowledge with other contributors. The tutorials are intended to be focused 'how to' documents.

Community Guide

Many people help OpenHatch serve newcomers to open source. The Community Guide helps explain our teams, individuals, and activities that support our goal of being a welcoming and helpful community.

CHAPTER 2

Indices and tables

- genindex
- search