ZenPack Development Procedures

November 2010

David Buler

David Buler
nyitguy127001@gmail.com

Summary

This document is designed to supplement the Zenoss Development Guide and consolidate numerous forum documents that describe how to create and publish a ZenPack. It does not provide details about the content of a ZenPack, but is restricted to the mechanics of the development process.

This guide mainly applies to ZenPacks that will be published to the Zenoss community. The majority of topics covered are not necessary for ZenPacks whose value is limited to the local organization.

Contribution Consent Form

The Zenoss community can only publish submissions from users that have submitted a Contribution Form, found at http://community.zenoss.org/community/developers. If you intend to submit your ZenPack to the community, start by submitting this form.

1. ZenPack Development Overview

At the time of this writing, the community ZenPack repository was being moved from the Subversion (SVN) platform to GitHub. Whereas SVN provided individual repositories for each ZenPack, GitHub merges all ZenPacks into a single repository. Therefore, when creating a new ZenPack or modifying an existing ZenPack, it is necessary to work with the entire repository as a whole.

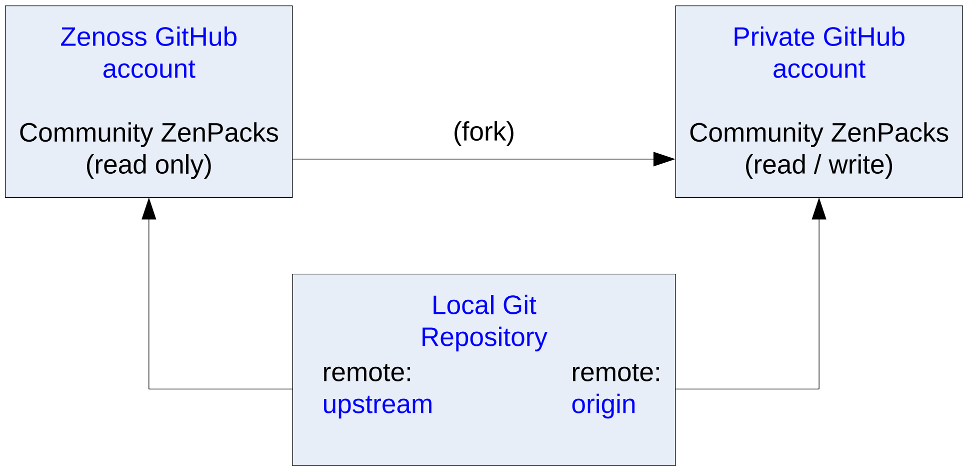


Figure 1: Development Repositories

Figure 1 illustrates the three repositories used in ZenPack development. The **Community ZenPacks** repository that resides in the Zenoss GitHub account is read only. However, it is the master repository and acts as the source for updates. After creating a private account on GitHub, a fork or copy of the Zenoss **Community ZenPacks** repository can be created in the private account. Since private accounts have read/write access, the repository can be modified. Finally, the local Git repository on the development server is configured with pointers to remote repositories. The Zenoss **Community ZenPacks** repository is configured with the alias *upstream* and the private **Community ZenPacks** repository with the alias *origin*.

1.1. Basic workflow for publishing a new ZenPack

1. Develop and test the new ZenPack in a branch on the local repository.
2. Merge the latest files from the *upstream* repository to the local repository.
3. Push the local repository to the *origin* repository.
4. Send a pull request to the Zenoss repository administrator.

2. Configuring the Development Environment

2.1. Install Git software on the development computer

See the main Git download page to download and compile the Git software manually or find a precompiled package. This link describes different methods of installing Git.

2.2. Create a local Git repository

Log in as the **zenoss** user to perform these steps. Create a directory to be used for the Git repository. This directory can be located anywhere. For clarity, it will be referred to as $LOCALREPO throughout this document.

```
$ mkdir $LOCALREPO
```

Next, initialize the Git repository:

```
$ cd $LOCALREPO
$ git init
```

2.3. Create a GitHub account

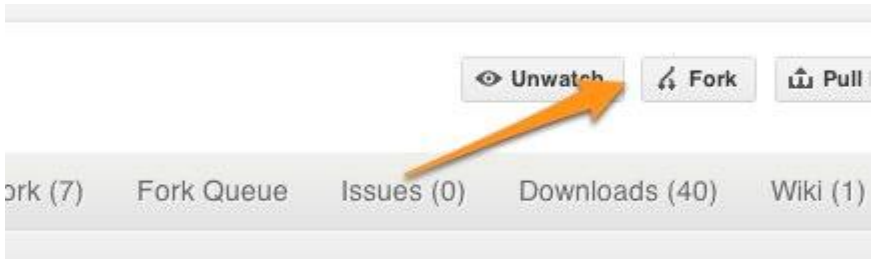
Visit https://github.com/plans and click on the **Create a free account** button. Set up your private account.

2.4. Fork the Community-Zenpacks repository

More information about forking GitHub repositories can be found at http://help.github.com/forking.

1. Log in to your GitHub account at https://github.com/login.
2. Navigate to the Community-Zenpacks repository at https://github.com/zenoss/Community-Zenpacks.

3. Click on the **Fork** button.



You will be presented with information about the fork.

2.5. Configure SSH access to your private GitHub account

This procedure assumes that OpenSSH is installed on the development computer. More information on this topic can be found at http://help.github.com/troubleshooting-ssh.

On the development computer, log in as the **zenoss** user. Open a console window.

Verify that the **zenoss** user has a public key.

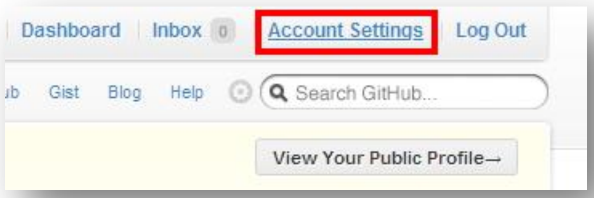
```
$ ls ~/.ssh
```

If the *id_rsa.pub* file does not exist, create a public key pair. Press <ENTER> three times to use the default settings:

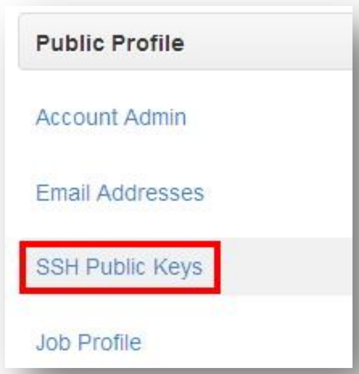
```
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/zenoss/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/zenoss/.ssh/id_rsa.
Your public key has been saved in /home/zenoss/.ssh/id_rsa.pub.
The key fingerprint is:
[long data string here]
```

Copy the contents of the *~/.ssh/id_rsa.pub* file to the clipboard.

Return to your private GitHub account. Select **Account Settings** in the top, right-hand corner:



From the left-hand pane, select **SSH Public Keys**:



In the middle of the screen, select **Add another public key**.

Name the key in the **Title** field. Paste the contents of the *id_rsa.pub* file in the **Key** field. Click the **Add key** button.

2.6.Create links to GitHub repositories

A link to the Zenoss Community-ZenPacks repository will be necessary to update the local repository. A link to the fork in your private account will allow updates to be saved and eventually merged with the Zenoss Community-ZenPacks repository.

Create the read-only link to the Zenoss Community-ZenPacks repository. Name the alias *upstream*:

```
$ git remote add upstream git://github.com/zenoss/Community-Zenpacks.git
```

Create the read/write link to the forked copy of Community-ZenPacks in your private account. Fill in your GitHub user name in place of [GIT_USER]

```
$ git remote add origin git@github.com:[GIT_USER]/Community-Zenpacks.git
```

2.7.Pull down Community-ZenPacks to local repository

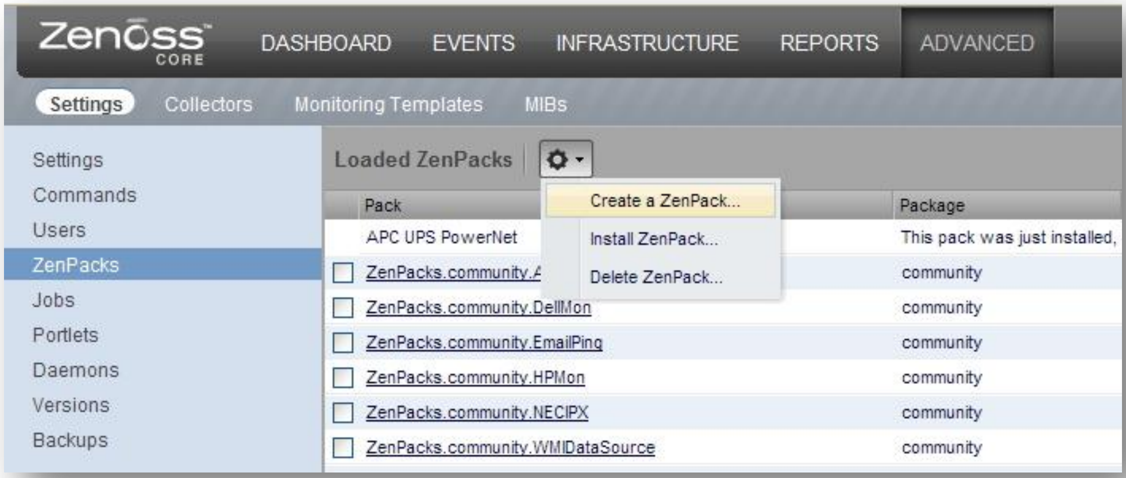
Finally, pull all the current files from the Zenoss Community-ZenPacks repository to the local repository:

```
$ git fetch upstream
$ git merge upstream/master
```

3. Developing a ZenPack

The primary reference for developing ZenPacks is the Zenoss Developer's Guide. Jane Curry has also published an excellent supplement to the Developer Guide: Creating Zenoss ZenPacks. For the purpose of this document, discussion of ZenPack development will be restricted to interactions with the local Git repository.

3.1.Creating a ZenPack



Pack	Package
APC UPS PowerNet	This pack was just installed,
ZenPacks.community.A	community
ZenPacks.community.DellMon	community
ZenPacks.community.EmailPing	community
ZenPacks.community.HPMon	community
ZenPacks.community.NECIPX	community
ZenPacks.community.WMIDataSource	community

1. Navigate to **Advanced > Settings > ZenPacks**.
2. From the Action menu, select **Create a ZenPack...**
3. In the **Create a new ZenPack** dialog, provide a name. See Section 3.2.1 in the Zenoss Developer's Guide for details regarding ZenPack names. As an example, we will create **ZenPacks.community.MyZenPack**.

3.2. Create branch for ZenPack development

It is a good idea to develop a ZenPack in its own branch.

First, update the master branch with content from upstream.

```
$ cd $LOCALREPO
$ git checkout master
$ git fetch upstream
$ git merge upstream/master
```

Now, create a separate branch and switch to it:

```
$ git branch MyZenPack
$ git checkout MyZenPack
```

3.3. Relocate the ZenPack to the local repository

The Zenoss Developer's Guide says the following about locating ZenPack source code outside of Zenoss:

> For any non-trivial ZenPacks we recommend maintaining the source code somewhere other than $ZENHOME/ZenPacks. There are a couple reasons for this:
>
> - Performing a **zenpack --remove** deletes the ZenPack's directory from $ZENHOME/ZenPacks. If you do not have the files copied in another location you can easily lose all or some fo your work.
> - If your ZenPack source is maintained in a version control system it is frequently easier to keep the code within a larger checkout directory elsewhere on the filesystem.

Copy the new ZenPack to the local repository directory:

```
$ cp -r $ZENHOME/ZenPacks/ZenPacks.community.MyZenPack $LOCALREPO
```

Reinstall the ZenPack at the new location:

```
$ cd $LOCALREPO
$ zenpack --link --install ZenPacks.community.MyZenPack
INFO:zen.ZenPackCMD:Previous ZenPack exists with same name
ZenPacks.community.FileManager
INFO:zen.ZenPackCMD:installing zenpack ZenPacks.community.FileManager;
launching process
$ zopectl restart
```

3.4. Add ZenPack to repository

```
$ git add ZenPacks.community.MyZenPack
```

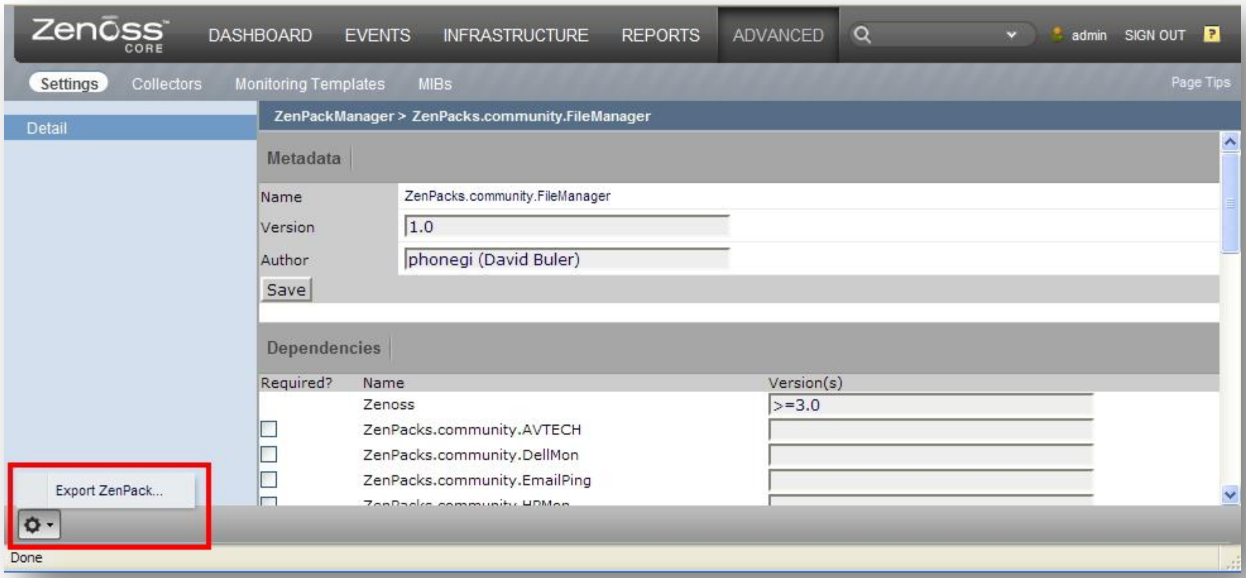
At this point, the ZenPack can be developed to completion. Use the development documents noted at the beginning of this section for more information. See the References section for links to Git documentation.

4. Publishing a ZenPack

OK, so you've finally completed your ZenPack and want to share it with the Zenoss Community. The ZenPack Workflow document recommends a few tests before publishing.

4.1. Create the ZenPack egg file

Fill in and save all Metadata and Dependency information, then export the ZenPack by selecting the **Action** button, then **Export ZenPack…**



ZenPackManager > ZenPacks.community.FileManager

Metadata

Name	ZenPacks.community.FileManager
Version	1.0
Author	phonegi (David Buler)

Save

Dependencies

Required?	Name	Version(s)
	Zenoss	>=3.0
☐	ZenPacks.community.AVTECH	
☐	ZenPacks.community.DellMon	
☐	ZenPacks.community.EmailPing	

4.2. Basic ZenPack Testing

After creating the egg, remove the ZenPack and install it from the egg. I prefer to do this at the command prompt so I can see if any errors occur:

```
$ zenpack --remove ZenPacks.community.MyZenPack
$ zenpack --install $ZENHOME/export/ZenPacks.community.MyZenPack-1.0-py2.6.egg
$ zopectl restart
```

Now, walk though each piece of the ZenPack – collectors, daemons, etc… to verify that everything works as it should. If there are problems, remove the ZenPack and reinstall the one in the repository. Fix any problems. Repeat until everything is working.

Take screenshots of the added functionality so they can be incorporated into the document that will describe the ZenPack.

Remove the ZenPack from the command line:

```
$ zenpack --remove ZenPacks.community.MyZenPack
```

Log back into the Zenoss system and verify that there are no artifacts or problems after removing the ZenPack.

4.3. Update *MyZenPack* branch from *upstream*

After the ZenPack has been proven clean, perform a final commit and tag:

```
$ cd $LOCALREPO
$ git checkout MyZenPack
$ git commit -m "Zenoss.community.MyZenPack 1.0"
$ git tag -a MyZenPack1.0 -m "Zenoss.community.MyZenPack 1.0"
```

Switch to the master branch and perform an update:

```
$ cd $LOCALREPO
$ git checkout master
$ git merge upstream/master
```

If you were updating an existing ZenPack, it is possible (unlikely) that the *upstream* repository has had commits that may conflict with your changes. You can compare commits to the *master* branch that are not in the *MyZenPack* branch by executing:

```
$ git log --oneline master ^MyZenPack
```

Merge the *master* branch into the *MyZenPack* branch:

```
$ git checkout MyZenPack
$ git merge master
```

4.4.Push changes to private GitHub account

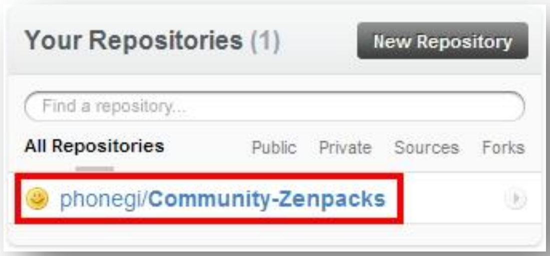
Once the *MyZenPack* branch of the local repository has been updated with the new ZenPack, push the changes up to the forked **Community-Zenpacks** in your private GitHub account.

```
$ git push --tags origin MyZenPack
```

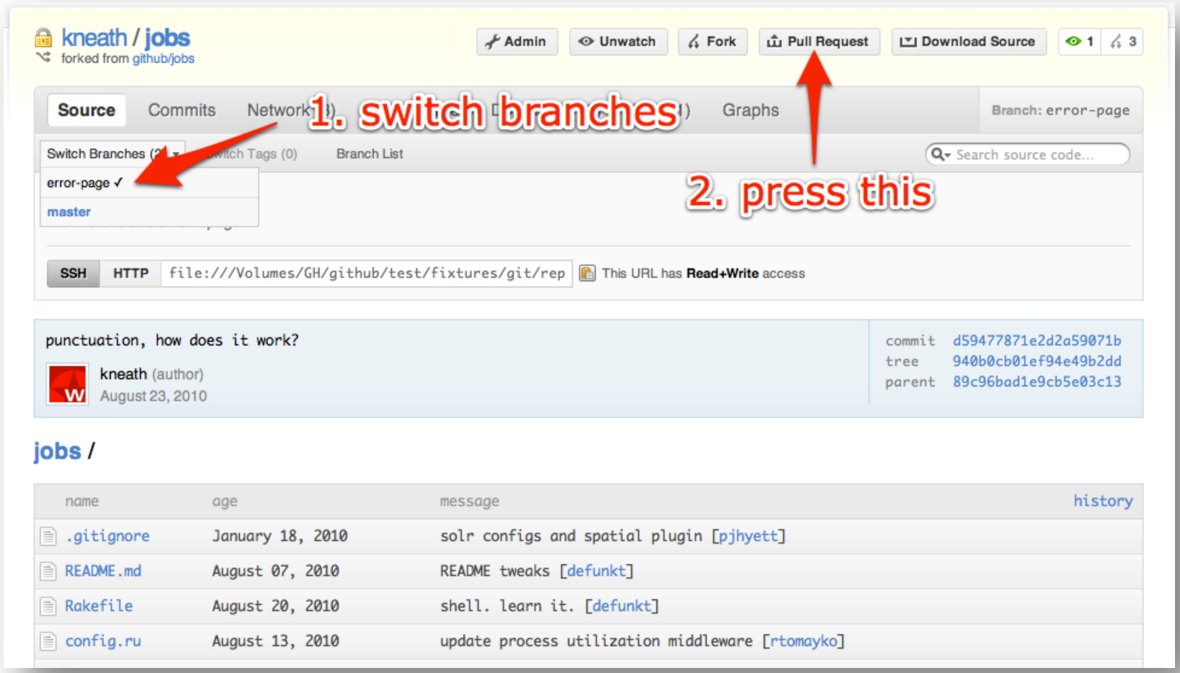
4.5.Send a pull request to the Zenoss Administrators

More information about pull requests can be found here.

1. Log in to your private GitHub account.
2. On the right-hand side of the screen under **Your Repositories**, select the **[username]/Community-Zenpacks** repository.



3. As illustrated below, select the *MyZenPacks* branch and then **Pull Request**.



punctuation, how does it work?

kneath (author)
August 23, 2010

commit d59477871e2d2a59071b
tree 940b0cb01ef94e49b2dd
parent 89c96bad1e9cb5e03c13

jobs /

name	age	message	history
.gitignore	January 18, 2010	solr configs and spatial plugin [pjhyett]	
README.md	August 07, 2010	README tweaks [defunkt]	
Rakefile	August 20, 2010	shell. learn it. [defunkt]	
config.ru	August 13, 2010	update process utilization middleware [rtomayko]	

4. Review Commits and Files Changed, add a description of your changes or ZenPack, and click the **Send Pull Request** button.

4.6.Document the new ZenPack

1. Open the ZenPack Template.
2. Click on the **Edit document** link.
3. Toggle HTML mode ON.
4. Copy the entire contents to the clipboard.
5. Go to the ZenPacks Documentation page.
6. Click on the **Create a document** link.
7. Select **Write a New Document**.
8. Toggle HTML mode ON.
9. Paste the contents of the clipboard into the body of the document.
10. Edit and Save the document.

4.7.Notify the Zenoss Team

Send an email to community@zenoss.com containing a link to the new document describing your ZenPack and inform them that a pull request has been sent.

References

Zenoss Developer's Guide
Jane Curry's Creating Zenoss ZenPacks
Git User's Manual, Git Documentation, Git Quick Reference
Git forking, Git pull requests
Zenoss Community - Git instructions
Zenoss Community - ZenPack Workflow