**Tool Migration**

**Debian Wheezy**

**Erich Huebner**

**hubzero.org**

**March 2013**

# 1. Requirements

[\_\_] A [VHUB.org](https://vhub.org/register) account.

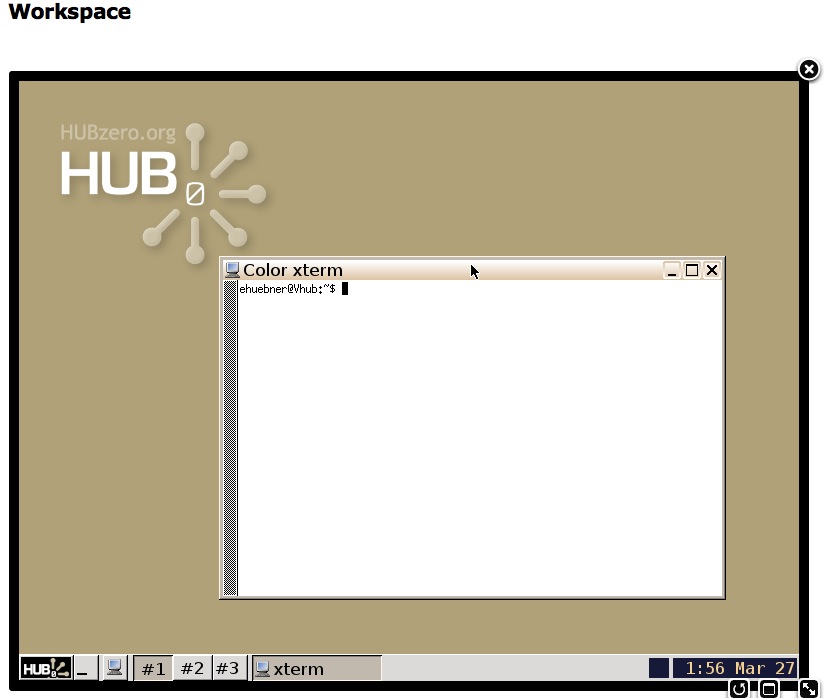
[\_\_] Access to the tool that is to be updated.

(added as a developer, hub admins added to the “apps” group).

[\_\_] Basic knowledge of the “nano” text editor. [Nano Manual](http://www.nano-editor.org/dist/v2.2/nano.html)

# 2. Setting up

Open a new tab to http://stage.vhub.org/myhub and open a new workspace.



# 3. Checking out the latest code

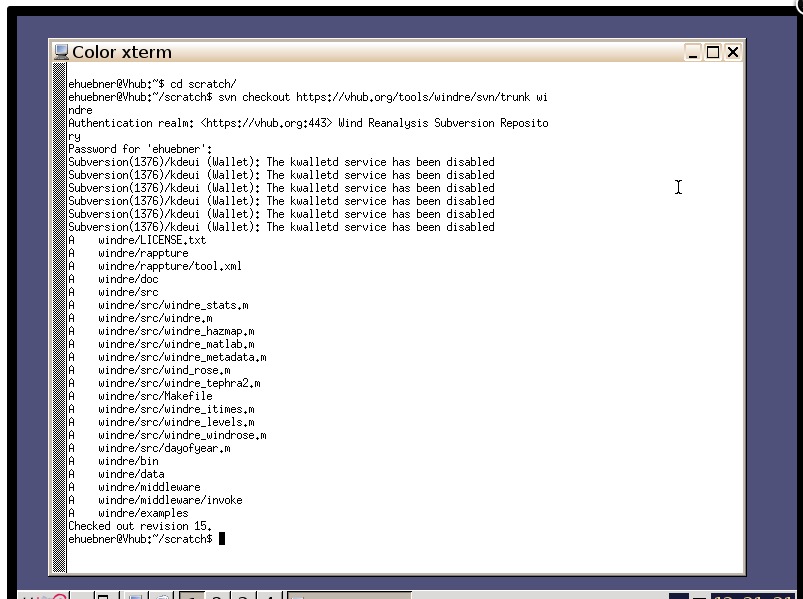
In your workspace, create a directory called “scratch”. This directory will hold a local copy of the tool's subversion source code repository where the new changes will be applied. Check out the tool's source code. The tool code source is now replicated in your scratch directory.

1 $ mkdir scratch

2 $ cd scratch

3 $ svn checkout [https://vhub.org/tools/”toolname”/svn/trunk](https://vhub.org/tools/) “toolname”

4 $



The example above shows the steps necessary to check out a local copy of a tool's source code repository. In line 1, the scratch directory is created. The scratch directory gives us a clean area to work while updating the tool. Next, we enter the scratch directory in line 2. Lastly, the "svn checkout" command is used, in line 3, to retrieve a local copy of the tool's source code repository. While checking out the repository, the developer may be prompted for their username and password. The [https://vhub.org](https://vhub.org/) username and password should be used. Note that the command in line 3 is checking out the tool's repository from [https://vhub.org](https://vhub.org/) instead of [https://stage.vhub.org](https://stage.vhub.org/). Contribtool on [https://stage.vhub.org](https://stage.vhub.org/) is setup to work with tool source code repositories from [https://vhub.org](https://vhub.org/). All source code changes should be committed to the [https://vhub.org](https://vhub.org/) repositories.

# 4. Editing the invoke script

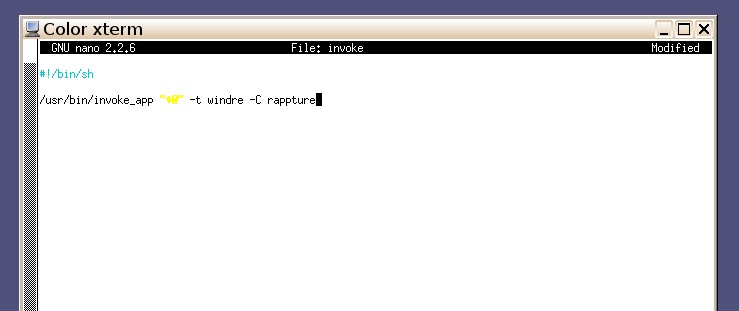
The invoke script must now call the “invoke\_app” script. We don’t antipiate that this will change going forward.

5 $ cd ”toolname”/middleware

6 $ nano invoke

7 $

8 $



The example above shows the steps necessary to update the path of the invoke script. In line 5, we enter the middleware directory. And in line 6, we use our favorite editor to access the invoke file. Next, change the path of invoke\_app to be: “/usr/bin/invoke\_app”. For Rappture tools, please add “–C rappture” to the end of this line. Finally, save and overwrite the current file.

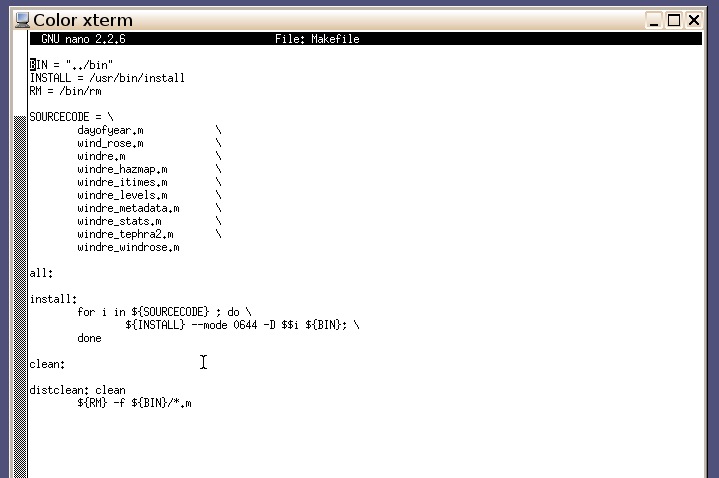
# 5. Testing your changes

Lets look at the Makefile. This is an essential part of installing a tool, which we will do in a couple steps to test the new changes to the tool in the workspace.

9 $ cd ../src

10 $ nano Makefile

11 $



The example above shows the steps necessary to access the Makefile. In line 9, we enter the src directory. And in line 10, we use our favorite editor to access the Makefile.

We are looking for four targets: all, install, clean, and distclean to be present, although not all may be used. Save and overwrite the current file if any changes are made.

-------------------------------------------------------------------------------------------------

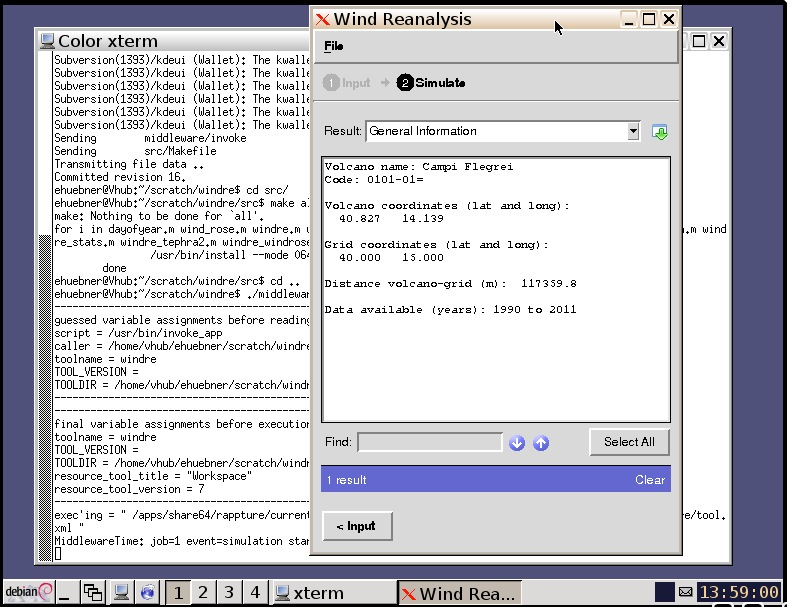
Now, let’s install the code using the Makefile. This will only install the tool locally in your scratch folder and does not affect the svn code repository on the server.

12 $ make all install

13 $ cd ..

14 $ ./middleware/invoke

15 $



The example above shows the steps necessary to install the updates to your tool. In line 12, we use the make command to call the entries in the Makefile for “all” and “install”. Next in line 13, change up one directory. And in line14, start the tool by executing the invoke script.

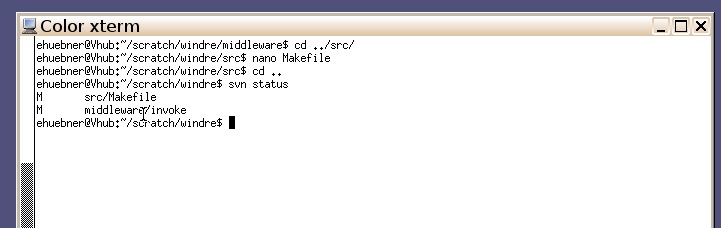
# 6. Checking your code into the repository

Next, let’s check the status of any files that we have modified as they relate to the code repository. We will also commit the code and create a new revision that will be ready to install on production.

16 $ cd ..

17 $ svn status

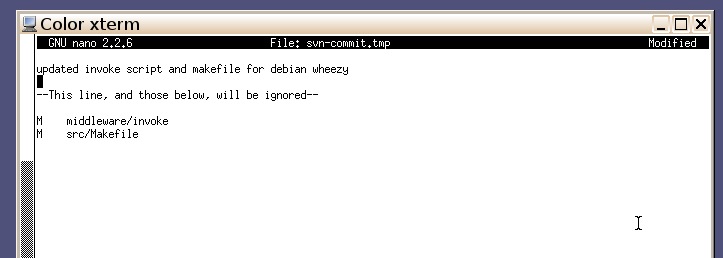
18 $



The example above shows the steps necessary to check the status of your changes as compared to the code repository. In line 16, we change up one directory to the top level of the tool directory. In line 17, the command ‘svn status’ shows us what files have changed signified by the “M” in the first column. The two in the example are expected. If you see a ‘?’ instead, those files are not recognized by the repository and need to be removed or added as needed. (see [SVN documentation](http://subversion.apache.org/docs/))

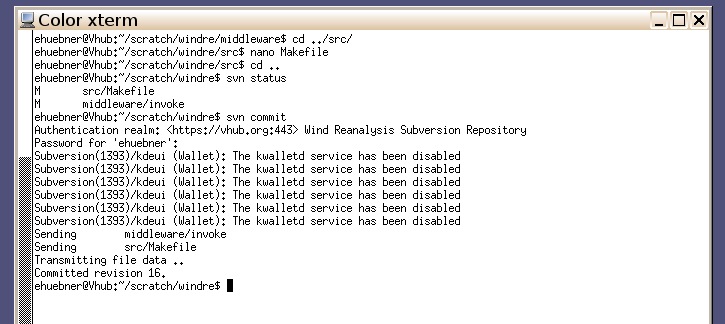
19 $ svn commit

20 $



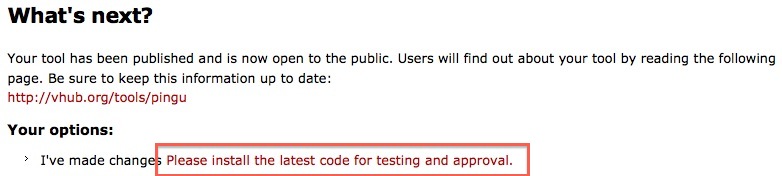
The example above shows the steps necessary to commit the updates to the svn repository. In line 19, we use the “svn commit” to start this process. An svn-commit.tmp file is created and opened in your default editor. Enter a comment describing your changes for this commit in the first and second line of the file. You will also see the two files that are to be updated in this file. Save and over write the file. You may be prompted for their username and

password. The [https://vhub.org](https://vhub.org/) username and password should be used. The output will finalize (if successful) with “Committed revision “some number”. (See image below)



# 7. Changing the status of the tool

In your web browser navigate to <http://stage.vhub.org/contribtool>, login when prompted. Click on your tool’s link. Find the link shown below, “Please install the lasted code for testing and approval” and click on it to specify you have uploaded new code.



# 8. Your tool has now been updated for Wheezy

Any updates to the tool may also be done at this time. The next step is for the HUBzero team to install the new version.

Thank you for updating with us.