Building the Amanzi User Guide

This document describes the steps to build and view the Amanzi User Guide. Assumptions are that you have successfully built Amanzi, and installed the required tools.

Required Software Tools

Not sure we'll put too many details in here, probably just say install Conda and then use pip to install the extra contributed packages. We probably should figure out what parts of the Pyhon conda distribution we really use though (i.e., python, matplotlib, sphinx, etc.)

The required software tools are anaconda (python2) and pip. The specific Python conda distribution used in the Amanzi code is 2.7.

Step-by-Step Instructions to Building the User Guide

Load the Amanzi build environment. This will load all of the specific export paths that you will need for the Amanzi install.

> . /amanzi/tools/init/amanzi-developer-env.sh

The only other environment variable you need to load/change is the AMANZI_INSTALL_DIR.

> export AMANZI_INSTALL_DIR=/n/swdev/amanzi/install/Ubuntu-16.04-x86_64/stable-0.87

Now, tell Amanzi to load the correct module for anaconda/python.

> module load anaconda-python2/5.1.0

Change to the location where you are planning to build the source code, or where you built the source code in previous examples. An example is below.

> cd /home/kbennett/center/amanzi/amanzi-0.87-ug/

Change to the user_guide subdirectory

> cd doc/user_guide

Now, you are ready to build the user guide.

You need to set the following environment variables to build the Amanzi User Guide.

All of these are shown as examples in the Makefile help, which details information on environmental variables and more. To access the Makefile help, type

> make help

Required Steps

The below three environmental variables are required.

AMANZI_INSTALL_DIR	The prefix of the Amanzi installation: $AMANZI_INSTALL_DIR/bin/amanzi is the binary that is used to run the tests in the User Guide. *THIS IS SET by modules, if loaded on your machine.*
AMANZI_SECTION_OPTIONS	Specifies which parts of the User Guide to build *MUST be done after export AMANZI_SECTION_OPTIONS=--install.*
PYTHONPATH	Update your PYTHONPATH to include the path to custom python modules amanzi_xml and prettytable.

Check for the variables by running

> env | grep AMANZI_UG_DIR

If you see it listed, you have set it correctly.

Setting the below environmental variables are optional.

AMANZI_INCLUDE_TODOS	(True|False) True specifies that TODO elements are included in the html or pdf document.

Shows three variables related to your PYTHONPATH:

AMANZI_UG_DIR	The full path to the User Guide source you are building. This is the source we assume you want to get the amanzi_xml and prettytable modules from. *NOT SET by modules, if loaded on your machine.*
AMANZI_PYTHON	These are the paths you need to add to your PYTHONPATH environment variable.
PYTHONPATH	This shows your current PYTHONPATH with the new paths for Amanzi prepended to it.

You can cut and paste directly from either AMANZI_PYTHONPATH to set your PYTHONPATH. An example of the PYTHONPATH is shown below.

> export

AMANZI_PYTHONPATH=/Volumes/data/ascem/amanzi/repos/amanzi-0.87/tools/amanzi_xml:/Volumes/data/ascem/amanzi/repos/amanzi-0.87/tools/prettytable

To set the AMANZI_SECTION_OPTIONS environment variable, which is used in the build, you should at the least set this equal to --install. Other options are listed under make help.

> export AMANZI_SECTION_OPTIONS=--install

If you want to check what you have set the variable to you can view it by typing

> echo $AMANZI_SECTION_OPTIONS

If you only want to update changes made to the verification section of the user guide, set

> export AMANZI_SECTION_OPTIONS=--verification

Now you are ready to set up the install of the User Guide. You can do this by typing,

> make html

This builds the User Guide in _build/html, and the top-level page is index.html.

Remote Files

If your files are on a separate server:

To view the user guide on your local machine, i.e. on Firefox, you will need to install or have installed FUSE (MacOS build is here: https://osxfuse.github.io/). Download both the osxfuse-3.7.1.dmg package, and the sshfs-2.5.0.pkg. Install these on your machine.

Make a directory that you will use to store the linked files. For example,

> mkdir user_guide

Now run sshfs and connect to where you built the User Guide on the servers. Point to the folder you generated in the previous step.

> sshfs es40:/home/kbennett/center/amanzi/user_guide/amanzi-0.87/doc/user_guide/ user_guide

Go to Firefox, and type in the url bar,

file://///Users/kbennett/Documents/center/ascem/amanzi/user_guide

Go into the _build directory, html directory, and then index.html. You should get the following page in your Firefox window.

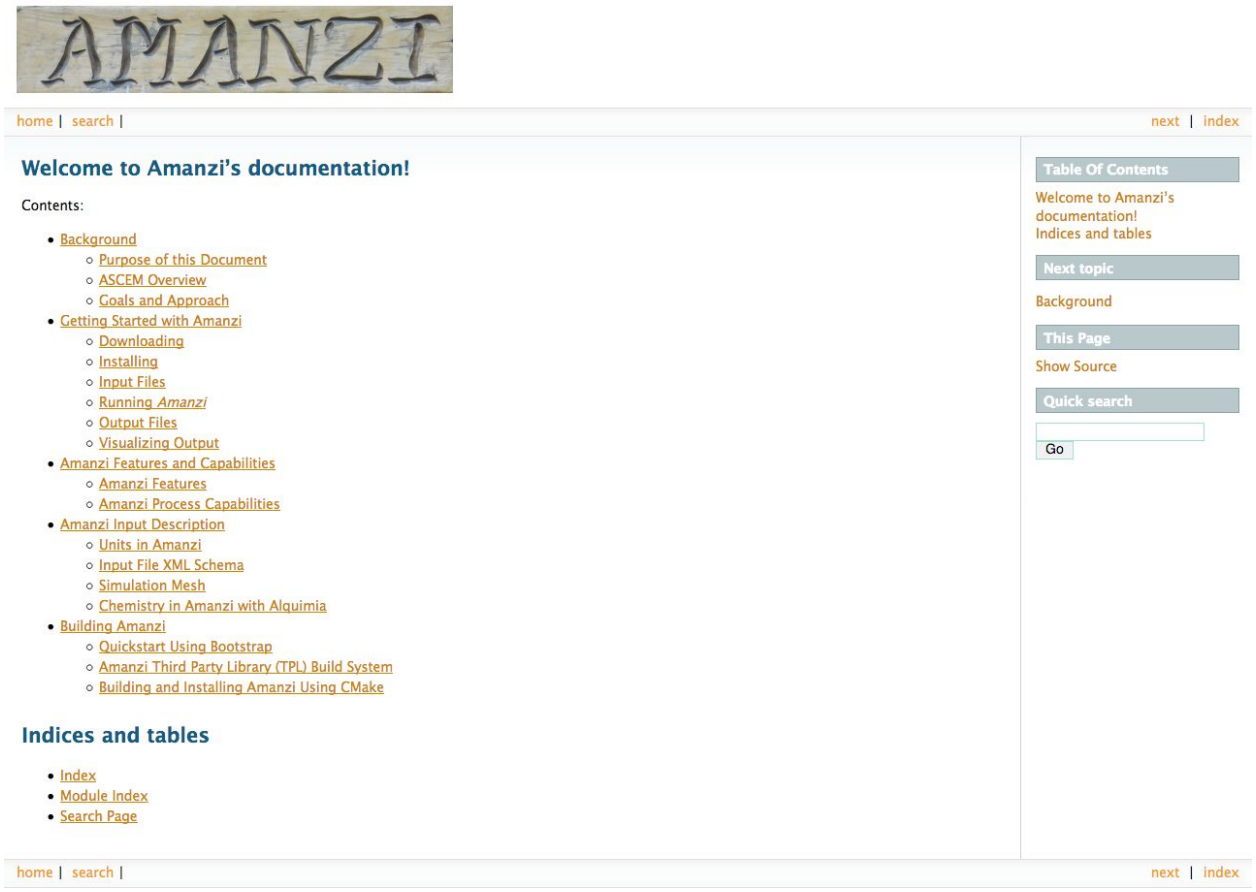
Local Files

If your files are local:

Go to Firefox, and type in the url bar,

file:////Users/kbennett/Documents/center/ascem/amanzi/amanzi/doc/user_guide

Go into the _build directory, html directory, and then index.html. You should get the following page in your Firefox window.



Publishing the User Guide

The User Guide is published under the amanzi organization on github.com. The naming convention for Users and Organizations on github are explained in their documentation:

https://help.github.com/articles/user-organization-and-project-pages/

The workflow for publishing is explained in

https://daler.github.io/sphinxdoc-test/includeme.html

Possible use of Travis to automatically push updates to the documentation is here.

https://github.com/Syntaf/travis-sphinx

Layout

If the gh-pages branch does not exist in your repo, and you are tasked with getting the documentation started, then refer to the instructions below in the "Creating the gh-pages branch" subsection. If the gh-pages branch already exists, you are in the right place, continue reading.

Within the gh-pages branch, the file structure should be laid out to support multiple versions of the documentation (User Guide, etc.). We'll work this out in the sphinx-test repository, and then when things stabilize we can create a gh-pages branch under amanzi and copy things over.

From doc/web_front/_build/html, copy into the top-level:

_images/
_sources/
_static/
downloads.html
genindex.html
index.html
search.html
searchindex.js

0.87/
    UserGuide
    - other directories or files could be at this level (e.g., Theory Guide).

0.87.1/
    UserGuide
    - other directories or files could be at this level (e.g., Theory Guide).

0.88-dev/
    UserGuide
    - other directories or files could be at this level (e.g., Theory Guide).

UserGuide -> 0.87.x/UserGuide (link to the latest stable version).

Workflow

Creating the gh-pages branch

We needed to create an empty branch to get things rolling. To test this concept out for our User Guide development I executed the following commands in the github.com/amanzi/sphinx-test repository.

```
git symbolic-ref HEAD refs/heads/gh-pages
rm .git/index
git clean -fdx
```

The first command sets the reference .git/HEAD to point to the new branch. This “symbolic” reference is changed when you create (or switch to) a branch using checkout as well. But we would drag the files and the history of the master branch with used checkout. Next, since we want to make this new branch truly empty we remove the .git/index file. Finally, we force the removal all untracked directories and files.

See git documentation for details:

https://git-scm.com/docs/git-symbolic-ref

https://git-scm.com/docs/git-clean

At this stage we have a completely empty branch of the repository that we started working in, which in this case was our sphinx-test repository. Next we simply copy in the files we want github to serve and commit those to the branch. From this point on this branch behaves like any other branch, it’s just that its history is now free from the history of the master (and it’s normal branches).