Gitlab CI with Docker and Sphinx Documentation

Release 0.1

Dan DeRusha

Jun 28, 2018

Contents:

CHAPTER 1

Setting up Docker

Docker is a platform for developers and sysadmins to develop, deploy, and run applications with containers. The use of Linux containers to deploy applications is called containerization. Containers are not new, but their use for easily deploying applications is.

Containerization is increasingly popular because containers are:

- Flexible: Even the most complex applications can be containerized.
- Lightweight: Containers leverage and share the host kernel.
- Interchangeable: You can deploy updates and upgrades on-the-fly.
- Portable: You can build locally, deploy to the cloud, and run anywhere.
- Scalable: You can increase and automatically distribute container replicas.
- Stackable: You can stack services vertically and on-the-fly.

1.1 Downloading Docker

Please visit the Docker Download link to download the latest Stable Docker Community Edition. There are downloads for use on local hardware (macOS, Win), Cloud (AWS and Azure), or Server (CentOS, DEbian, Fedora, or Ubuntu) I chose the macOS version.

1.2 Installing Docker

1. Open the installer you just downloaded from Docker's site, and drag the application to the Applications Directory.
2. Double click on Docker.app and follow the directions posted on Docker's site. They are good.

1.3 Getting Started after Docker install

Again, Docker's site is well documented. Click here for some commands to get you started.

1.4 Kubernetes

Kubernetes is only available in Docker for Mac 17.12 CE and higher, on the Edge channel. Kubernetes support is not included in Docker for Mac Stable releases. To find out more about Stable and Edge channels and how to switch between them, see General configuration.

Docker for Mac 17.12 CE (and higher) Edge includes a standalone Kubernetes server that runs on your Mac, so that you can test deploying your Docker workloads on Kubernetes.

CHAPTER 2

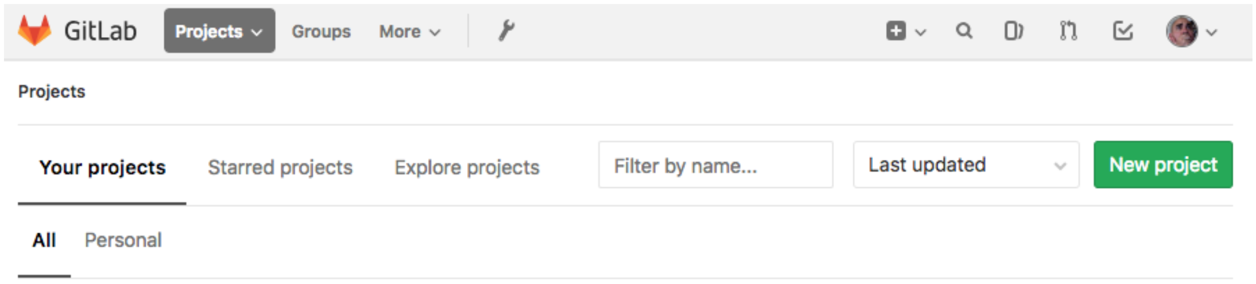
Gitlab setup

We set our sever up 3 years ago and did not document the steps :(BUT there is a article posted on Techrepublic.com that has all the steps covered. Jack even included instructions for adding SSH keys so your client can talk to the GitLab server.

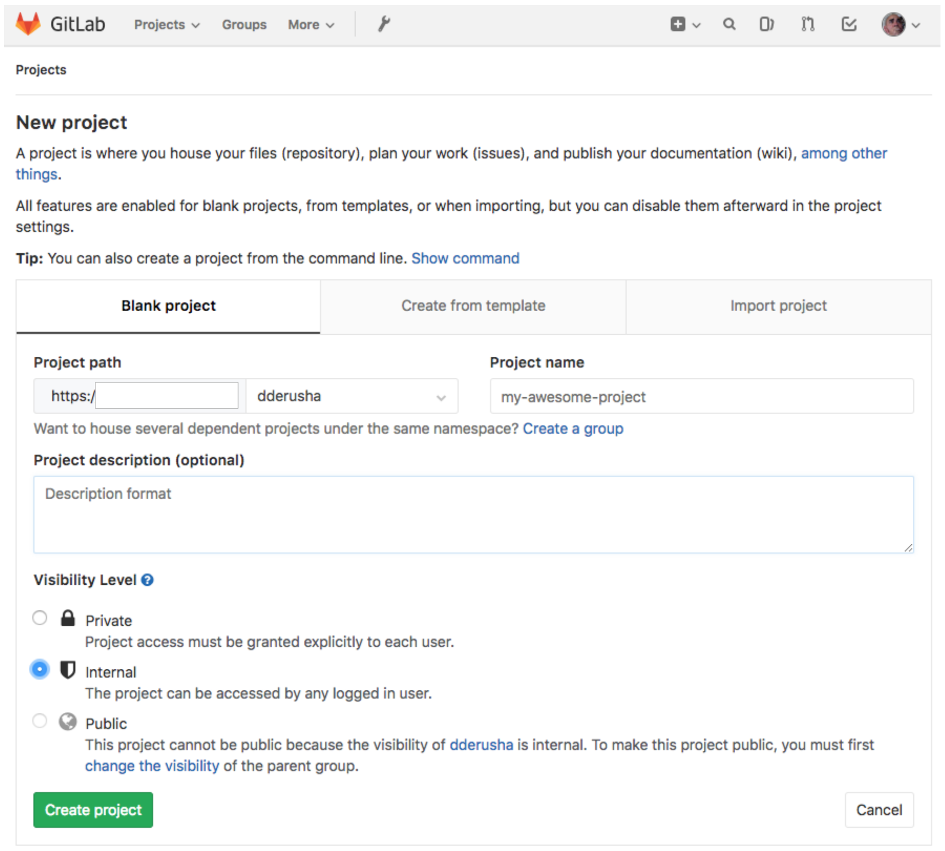
2.1 Gitlab prep for CI

At this point you should have a functioning internal GitLab server. Next step is to create a project.

- Login to your Gitlab server.
- Click the green button to create a new Project.



1. Name your project
2. add a project description
3. Select a visability level. **I picked Internal so anyone can contribute from our team.**
4. Click create project



Projects

New project

A project is where you house your files (repository), plan your work (issues), and publish your documentation (wiki), among other things.

All features are enabled for blank projects, from templates, or when importing, but you can disable them afterward in the project settings.

Tip: You can also create a project from the command line. Show command

Blank project | Create from template | Import project

Project path: https:/ dderusha

Project name: my-awesome-project

Want to house several dependent projects under the same namespace? Create a group

Project description (optional)

Description format

Visibility Level

Private
Project access must be granted explicitly to each user.

[✓ Internal]
The project can be accessed by any logged in user.

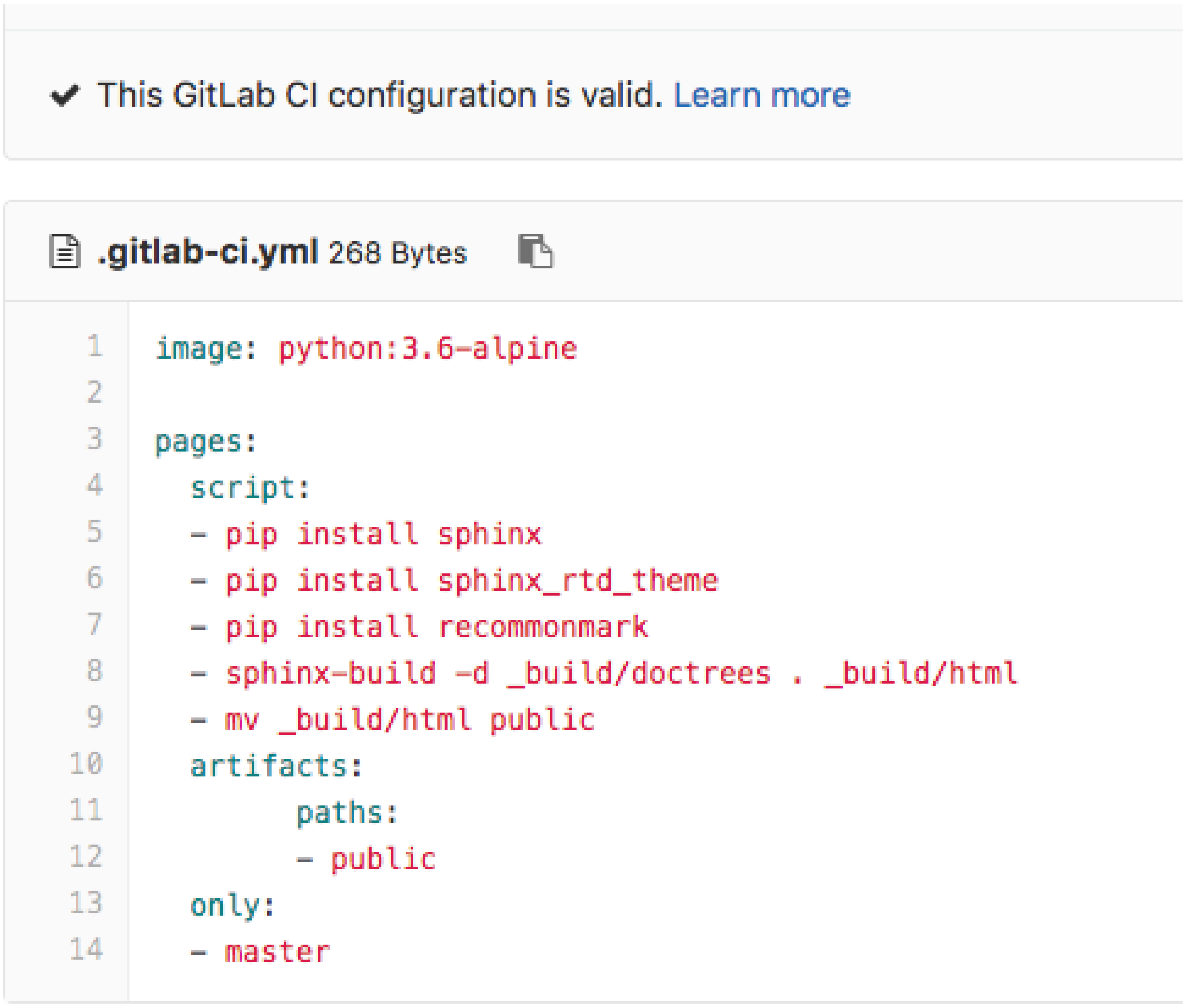
Public
This project cannot be public because the visibility of dderusha is internal. To make this project public, you must first change the visibility of the parent group.

2.2 Creating .gitlab-ci.yml file

Now we have a project started and we can add a configuation file. The .gitlab-ci.yml file needs to be in the root of the project.

Note: YAML files are sensitive to indentation and spacing. Do not use tab to create spaces.

`Touch` and create a file called .gitlab-ci.yml


✔ This GitLab CI configuration is valid. Learn more

.gitlab-ci.yml 268 Bytes

```
image: python:3.6-alpine

pages:
  script:
  - pip install sphinx
  - pip install sphinx_rtd_theme
  - pip install recommonmark
  - sphinx-build -d _build/doctrees . _build/html
  - mv _build/html public
  artifacts:
        paths:
        - public
  only:
  - master
```

As you can see at the top of the image shows the .yml file is configured correctly. Super, it's setup correctly, but how do we format it?

Note: There is a list of reserved keywords that cannot be used to name a job.

- `image`
- `services`
- `stages`
- `types`
- `before_script`
- `after_script`
- `varibales`
- `cache`

A job is defined by a list of paramaters telling the job what to do. In our case, we want to automate our document building. Now all the pieces are coming together.

- We have a gitlab server to host our projects containing code and documentation.
- We configured docker
- We setup our runner

We need to tell the runner what to "automate", then the runner configures the docker contaniner with our .gitlab-ci.yml setup file. Every job needs to have a script, everything else is optional. In our case we use `image` to define what OS to load in our docker container. We chose alpine 3.6 which is a 5MB linux image. You can find tons of other Official repositories on Dockers site.

> `Pages:` can be any word you want to describe the job such as "poopmonster" or "job1". The educated call it arbitrary. The `script:` builds a fresh new envirnoment everytime. Previously these steps were manual. The CI also takes the built files and reloads them so all changes are immediatly available.
>
> `only:` is a list of git refs for which job is created.
>
> `artifacts:` Artifacts are a listing of files and directories where the successful job gets placed.

For more information about GitLab's CI/CD configuration Click here.

CHAPTER 3

Resources

Here are some links that I found useful.

CHAPTER 4

Setting up the CI Runner for Gitlab

CI is Continuous intergration. Allows multiple people to work on the same project code in real time. Users create code, and upload/merge to the CI server many times throughout the workday. We are going to use a mac OS runner with our Gitlab server.

Note: Install the runner on a different host than your GitLab server.

1. Using terminal, download the binary package.

   ```
   sudo curl --output /usr/local/bin/gitlab-runner https://
   gitlab-runner-downloads.s3.amazonaws.com/latest/binaries/
   gitlab-runner-darwin-amd64
   ```

2. Make the binary executable.

   ```
   sudo chmod +x /usr/local/bin/gitlab-runner
   ```

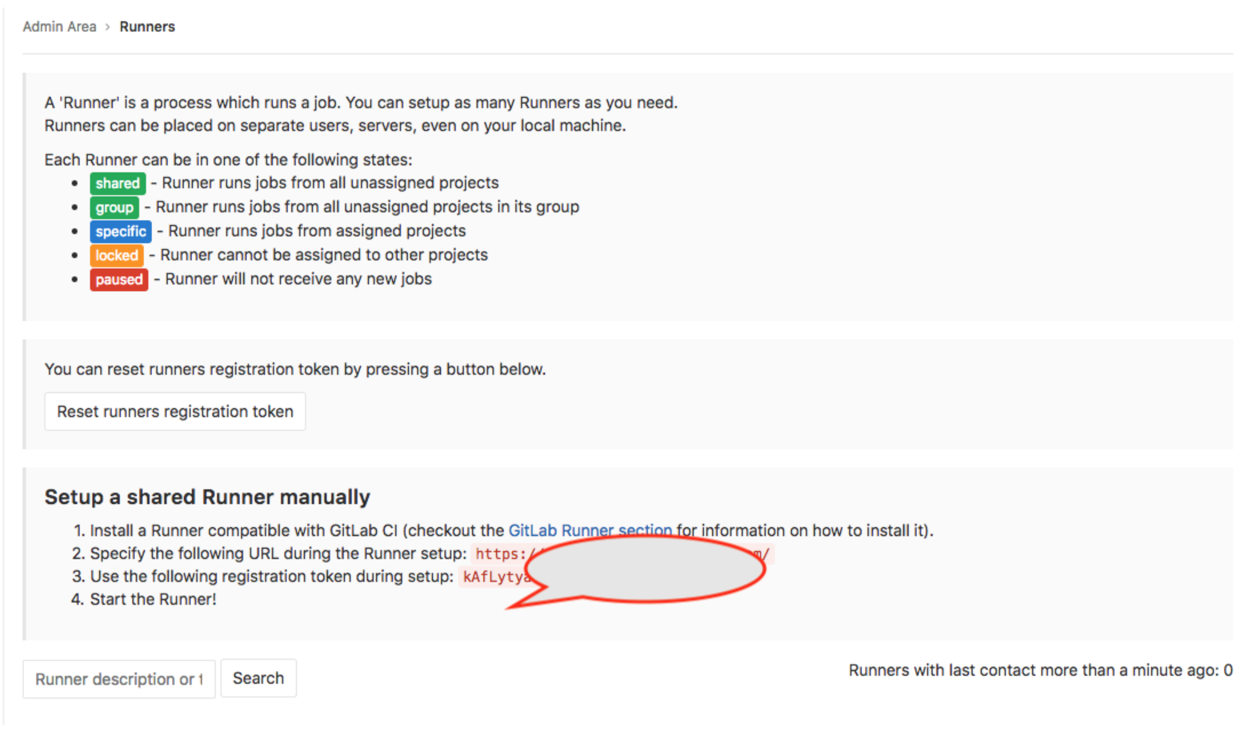
The following commands need to be run as the user that's going to run the runner.

4.1 Install & Register the runner

Registering the runner binds the runner to your gitlab server. Make sure you install the runner on a different server than gitlab. You can only add the runner to your gitlab server if you are the admin of the GitLab server. Change to your home dir, install the runner and start the service.

- `cd ~`
- `gitlab-runner install`
- `gitlab-runner start`

The registration token is available at the `admin/runners` page.



Admin Area › Runners

A 'Runner' is a process which runs a job. You can setup as many Runners as you need.
Runners can be placed on separate users, servers, even on your local machine.

Each Runner can be in one of the following states:

- shared - Runner runs jobs from all unassigned projects
- group - Runner runs jobs from all unassigned projects in its group
- specific - Runner runs jobs from assigned projects
- locked - Runner cannot be assigned to other projects
- paused - Runner will not receive any new jobs

You can reset runners registration token by pressing a button below.

Reset runners registration token

Setup a shared Runner manually

1. Install a Runner compatible with GitLab CI (checkout the GitLab Runner section for information on how to install it).
2. Specify the following URL during the Runner setup: `https:/`[redacted]`m/`
3. Use the following registration token during setup: `kAfLytya`[redacted]
4. Start the Runner!

Runner description or t | Search

Runners with last contact more than a minute ago: 0

Register

- `gitlab-runner register`
- Enter your Gitlab server url.
- `https://my.server.com`
- Enter your runner token.
- `myS3crT0k`
- Enter a description for the runner. If you can't think of anything clever, it can be changed later in the UI.
- Next enter tags for the runner. Add as many as you would like, separated by commas.
- Enter the runner executor, we used `docker`
- Lastly we need to set the default image docker will use to build the envirnoment. We are using `alpine:latest` or alpine 3.6.

For other configurations and OS's check out Gitlab's Registering Runners Documentation.

CHAPTER 5

Getting started with Sphinx

While attending and presenting at Mac Admin and Developer conference, MacAD.uk there was a session Your code should document itself! Embedding documentation into your Python projects that interested me. Awesome content presented by Bryson, I knew someday this would be useful. Working on a project, people were asking for documentation explaing our processes. Like any good project, we didn't have any centralized methodical represenation of our thoughts, why not use Sphinx?

5.1 Sphinx

Sphinx is a tool that makes it easy to create intelligent and beautiful documentation, written by Georg Brandl and licensed under the BSD license.

5.2 Install Sphinx

```
pip install sphinx
```

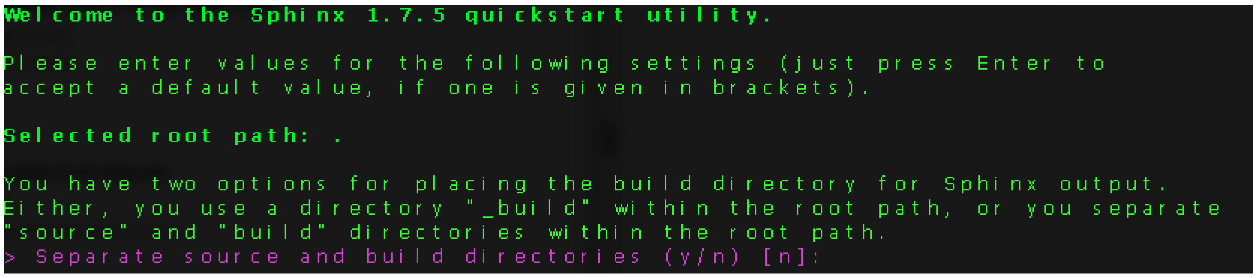
Make a directory somewhere to house your Sphinx projects. `mkdir /place/to/store/sphinx/docs`

Change into your new directory. `cd /place/where/you/stored/sphinx/docs`

Next start your project by running `sphinx-quickstart`

5.3 Setting up your Sphinx project

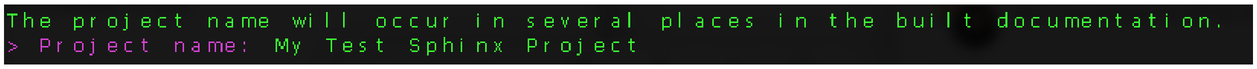
After runnin `sphinx-quickstart` you should see the following output.



```
Welcome to the Sphinx 1.7.5 quickstart utility.

Please enter values for the following settings (just press Enter to
accept a default value, if one is given in brackets).

Selected root path: .

You have two options for placing the build directory for Sphinx output.
Either, you use a directory "_build" within the root path, or you separate
"source" and "build" directories within the root path.
> Separate source and build directories (y/n) [n]:
```

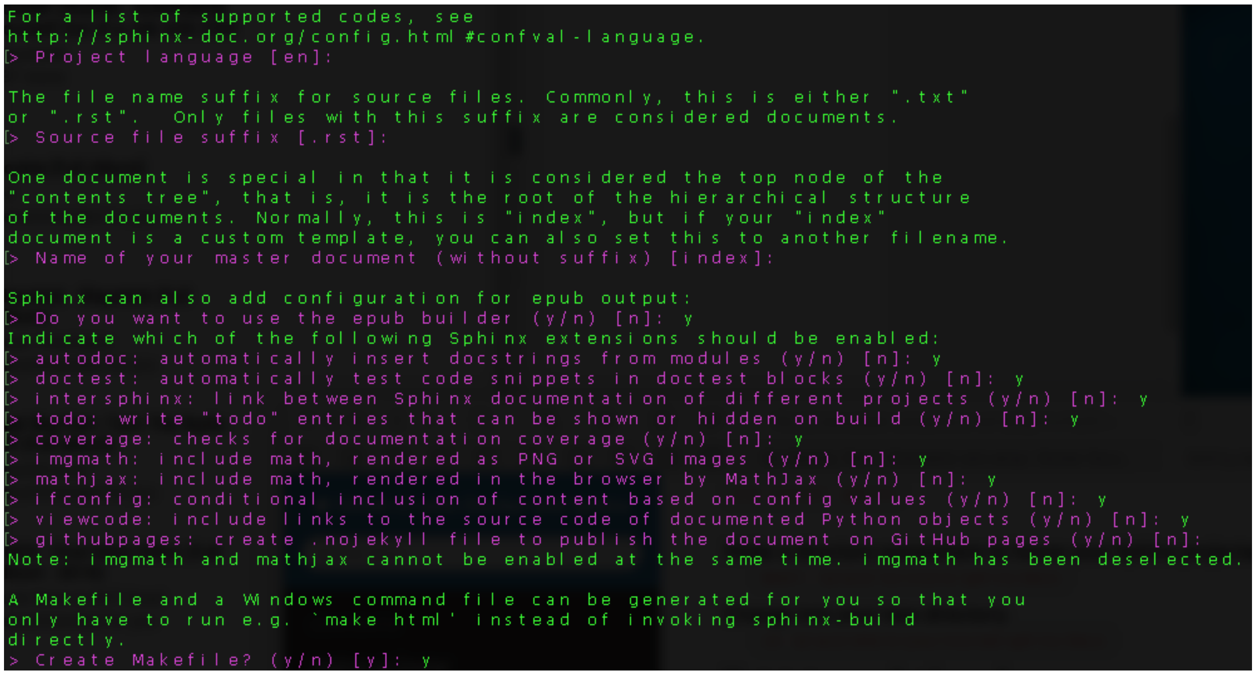
the default setting is in `[]` , press return for the default setting.

- Follow along with the promts that get outputted after your selections.
- Name the project
- Add the author and give the project a version number.



```
The project name will occur in several places in the built documentation.
> Project name: My Test Sphinx Project
```

After completing all the prompts, I suggest creating the Makefile.

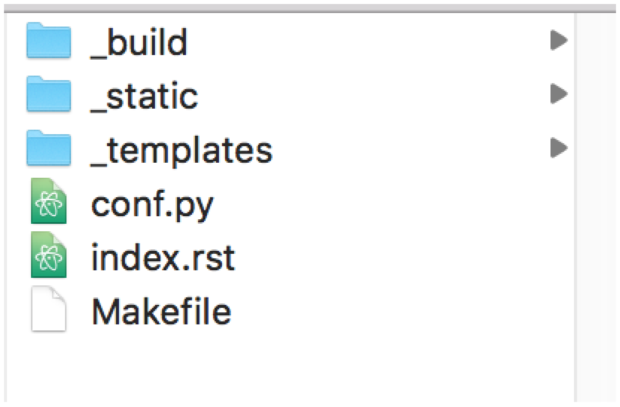


```
For a list of supported codes, see
http://sphinx-doc.org/config.html#confval-language.
> Project language [en]:

The file name suffix for source files. Commonly, this is either ".txt"
or ".rst".  Only files with this suffix are considered documents.
> Source file suffix [.rst]:

One document is special in that it is considered the top node of the
"contents tree", that is, it is the root of the hierarchical structure
of the documents. Normally, this is "index", but if your "index"
document is a custom template, you can also set this to another filename.
> Name of your master document (without suffix) [index]:

Sphinx can also add configuration for epub output:
> Do you want to use the epub builder (y/n) [n]: y
Indicate which of the following Sphinx extensions should be enabled:
> autodoc: automatically insert docstrings from modules (y/n) [n]: y
> doctest: automatically test code snippets in doctest blocks (y/n) [n]: y
> intersphinx: link between Sphinx documentation of different projects (y/n) [n]: y
> todo: write "todo" entries that can be shown or hidden on build (y/n) [n]: y
> coverage: checks for documentation coverage (y/n) [n]: y
> imgmath: include math, rendered as PNG or SVG images (y/n) [n]: y
> mathjax: include math, rendered in the browser by MathJax (y/n) [n]: y
> ifconfig: conditional inclusion of content based on config values (y/n) [n]: y
> viewcode: include links to the source code of documented Python objects (y/n) [n]: y
> githubpages: create .nojekyll file to publish the document on GitHub pages (y/n) [n]:
Note: imgmath and mathjax cannot be enabled at the same time. imgmath has been deselected.

A Makefile and a Windows command file can be generated for you so that you
only have to run e.g. `make html' instead of invoking sphinx-build
directly.
> Create Makefile? (y/n) [y]: y
```

Congrats!! your first sphinx project is now started!

5.4 Working with your new Sphinx project

Navagate to the directory you setup for your documentation, and you should see something similar:


```
_build
_static
_templates
conf.py
index.rst
Makefile
```

I use atom to edit the `conf.py` and `index.rst` files. There are plenty of editors you can choose from such as Sublime or BBedit.

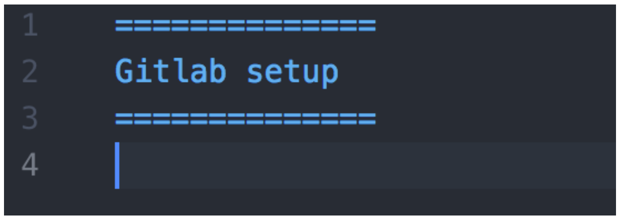
- Open terminal and `cd` to the project directory.
- Next let's create some `.rst` files to get our documentation started.
- While in the current project directory, `touch mysphinxdirections.rst` to create a Restructured Text file.

I'd create a couple of files to get started. For example, some docs used to create this project are called

- sphinxgettingstarted.rst
- GitLabsetup.rst
- runnersetup.rst

Avoid spaces, and try to keep the filename relevant to it's function.

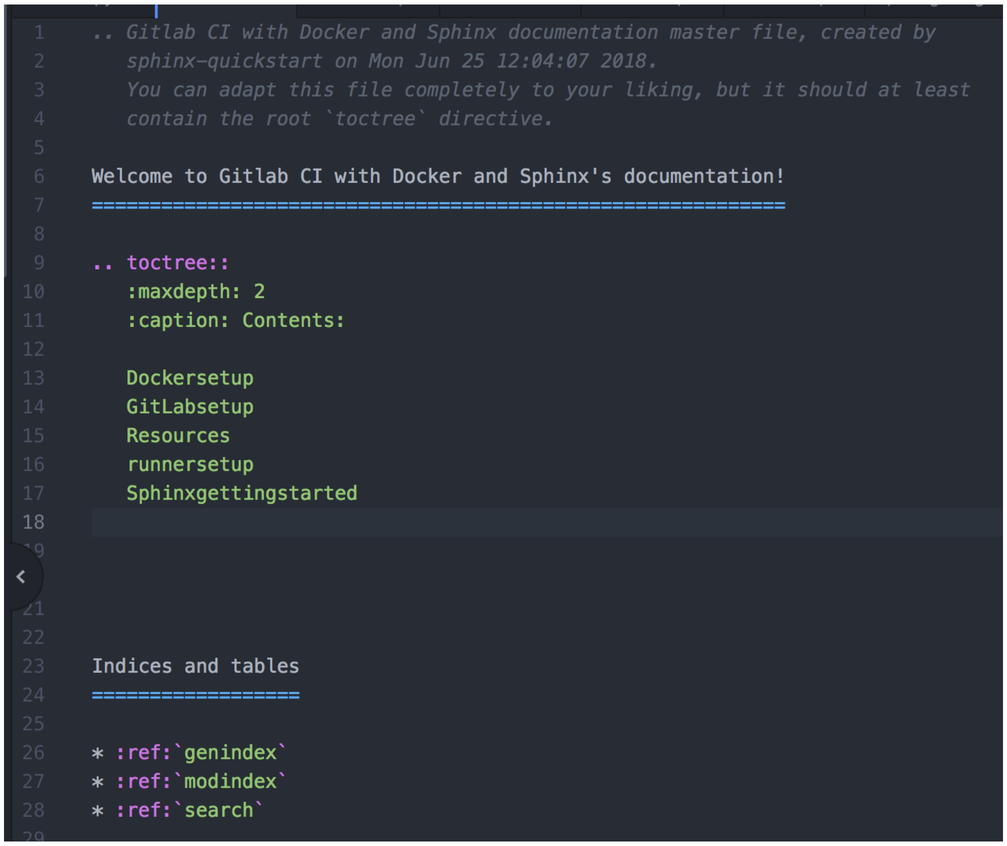
Open your editor and open your project. Open your `.rst` files and give them a title heading.


```
===============
Gitlab setup
===============
```

Make sure to save the files after adding the heading.

Next open your index.rst, and add the `.rst` files you touched in the terminal.

Note: Don't add the file extension `.rst` to the index list.



```
.. Gitlab CI with Docker and Sphinx documentation master file, created by
   sphinx-quickstart on Mon Jun 25 12:04:07 2018.
   You can adapt this file completely to your liking, but it should at least
   contain the root `toctree` directive.

Welcome to Gitlab CI with Docker and Sphinx's documentation!
============================================================

.. toctree::
   :maxdepth: 2
   :caption: Contents:

   Dockersetup
   GitLabsetup
   Resources
   runnersetup
   Sphinxgettingstarted




Indices and tables
==================

* :ref:`genindex`
* :ref:`modindex`
* :ref:`search`
```

At this point, we have

- Installed Sphinx.
- created a Sphinx project.
- created .rst files for our project.
- added .rst filenames to the index.rst

Now that we have some content configured, let’s build some documentation!

5.5 Building your Sphinx project

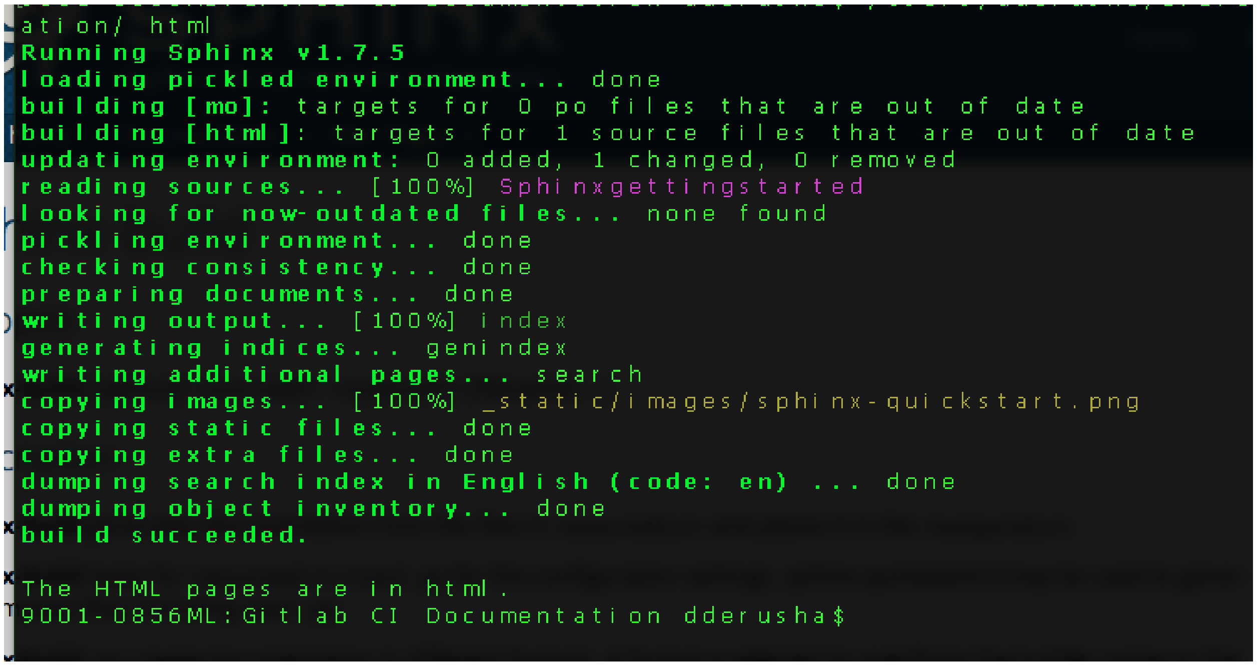
With the base of our project configured, we can generate some html content to view what we’ve started. Head back to your terminal application for the next step.

```
sphinx-build [options] <source directory> <output directory>
[filenames]
```

If no output directory is listed, the output defaults to the source.

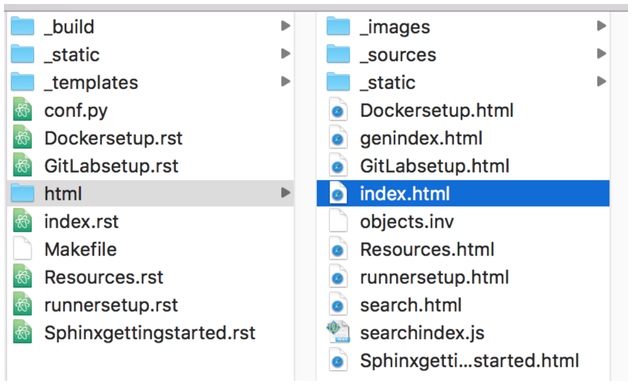
```
sphinx-build /path/to/my/sphinx/project/ html
```

will result in the following output.



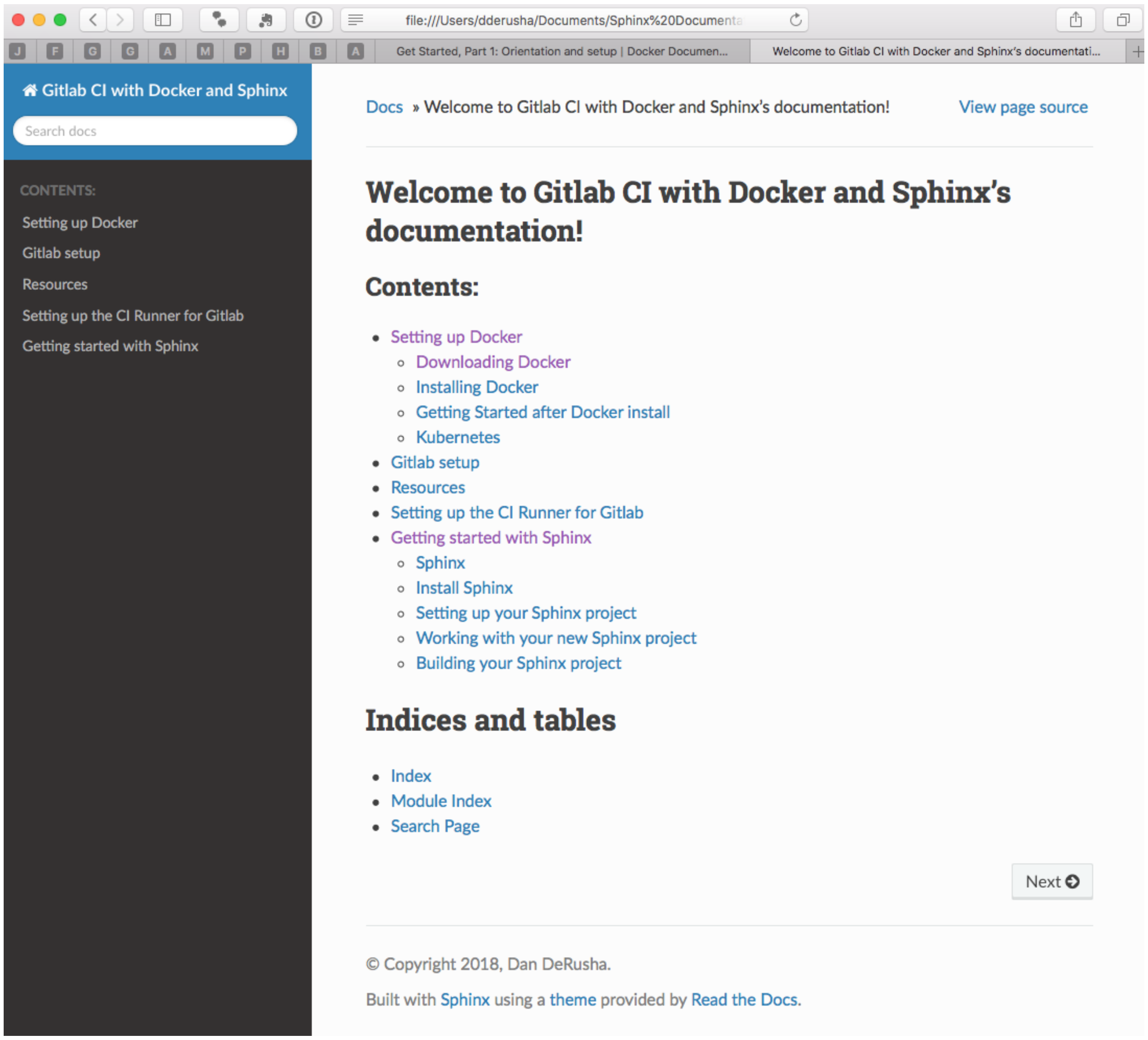
```
ation/ html
Running Sphinx v1.7.5
loading pickled environment... done
building [mo]: targets for 0 po files that are out of date
building [html]: targets for 1 source files that are out of date
updating environment: 0 added, 1 changed, 0 removed
reading sources... [100%] Sphinxgettingstarted
looking for now-outdated files... none found
pickling environment... done
checking consistency... done
preparing documents... done
writing output... [100%] index
generating indices... genindex
writing additional pages... search
copying images... [100%] _static/images/sphinx-quickstart.png
copying static files... done
copying extra files... done
dumping search index in English (code: en) ... done
dumping object inventory... done
build succeeded.

The HTML pages are in html.
9001-0856ML:Gitlab CI Documentation dderusha$
```

Head back to the finder and locate your source directory and locate the new HTML directory.



```
_build                       _images
_static                      _sources
_templates                   _static
conf.py                      Dockersetup.html
Dockersetup.rst              genindex.html
GitLabsetup.rst              GitLabsetup.html
html                         index.html
index.rst                    objects.inv
Makefile                     Resources.html
Resources.rst                runnersetup.html
runnersetup.rst              search.html
Sphinxgettingstarted.rst     searchindex.js
                             Sphinxgetti...started.html
```

Double click on the index.html file, and a web browser will display your project.



file:///Users/dderusha/Documents/Sphinx%20Documenta

Get Started, Part 1: Orientation and setup | Docker Documen...

Welcome to Gitlab CI with Docker and Sphinx's documentati...

Gitlab CI with Docker and Sphinx

Search docs

CONTENTS:

- Setting up Docker
- Gitlab setup
- Resources
- Setting up the CI Runner for Gitlab
- Getting started with Sphinx

Docs » Welcome to Gitlab CI with Docker and Sphinx's documentation!

View page source

Welcome to Gitlab CI with Docker and Sphinx's documentation!

Contents:

- Setting up Docker
  - Downloading Docker
  - Installing Docker
  - Getting Started after Docker install
  - Kubernetes
- Gitlab setup
- Resources
- Setting up the CI Runner for Gitlab
- Getting started with Sphinx
  - Sphinx
  - Install Sphinx
  - Setting up your Sphinx project
  - Working with your new Sphinx project
  - Building your Sphinx project

Indices and tables

- Index
- Module Index
- Search Page

Next

© Copyright 2018, Dan DeRusha.

Built with Sphinx using a theme provided by Read the Docs.

shpinx-build man page and additional options

Doesn't that look purdy? I'm not a designer, and with minimal effort I have a resource that looks nice! Yes, yes that does look nice, why doesn't my project look like that?

5.6 Sphinx Themes

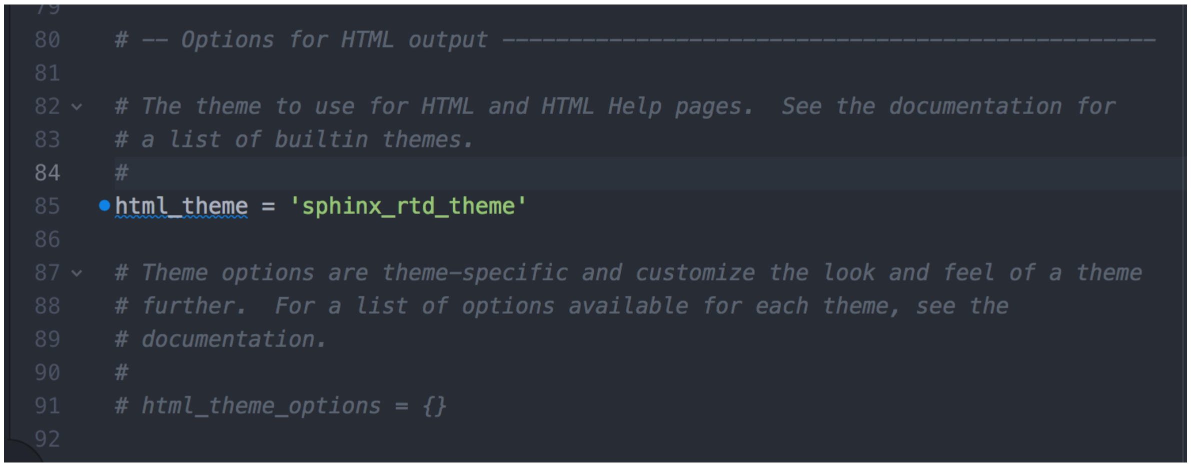
A theme is a collection of HTML templates, stylesheets and other files. A theme keeps the HTML appearance consistant throughout the project making it look good.

In our example project, we are using `sphinx_rtd_theme`.

5.7 Setting up a Theme

Open your text editor and navigate back to the source directory. Locate the `conf.py` and open it for modification. Sphinx has some built in themes you can use setting the `html_theme` config value in your `conf.py`.

- Search `conf.py` for `html_theme`, enter the value `sphinx_rtd_theme` and save the script.



```
# -- Options for HTML output ----------------------------------------------------

# The theme to use for HTML and HTML Help pages.  See the documentation for
# a list of builtin themes.
#
html_theme = 'sphinx_rtd_theme'

# Theme options are theme-specific and customize the look and feel of a theme
# further.  For a list of options available for each theme, see the
# documentation.
#
# html_theme_options = {}
```

- Open terminal and install the theme `pip install sphinx_rtd_theme`

Save all your work.

Re-build your project, and refresh your browser and now you should see the blue and black theme.

CHAPTER 6

Indices and tables

- genindex
- modindex
- search