Purdue CYAN Lab Documentation

Release 1.0

Read the Docs

Feb 05, 2024

DOCUMENTATION

Part I

Documentation

CHAPTER

ONE

GETTING STARTED

See also:

Part of the following information is reproduced from *Read the Docs* Getting Started with Sphinx

Sphinx is a powerful documentation generator that has many great features for writing technical documentation including:

- Generate web pages, printable PDFs, documents for e-readers (ePub), and more all from the same sources
- You can use reStructuredText or Markdown to write documentation
- An extensive system of cross-referencing code and documentation
- Syntax highlighted code samples
- A vibrant ecosystem of first and third-party extensions

If you want to learn more about how to create your first Sphinx project, read on.

1.1 Quick start

Assuming you have Python already:

```
$ pip install sphinx
```

Create a directory inside your project to hold your docs:

```
$ cd /path/to/project
$ mkdir docs
```

Run `sphinx-quickstart` in there:

```
$ cd docs
$ sphinx-quickstart
```

This quick start will walk you through creating the basic configuration; in most cases, you can just accept the defaults. When it's done, you'll have an `index.rst`, a `conf.py` and some other files. Add these to revision control.

Now, edit your `index.rst` and add some information about your project. Include as much detail as you like. Build them to see how they look:

```
$ make html
```

Your `index.rst` has been built into `index.html` in your documentation output directory (typically `_build/html/index.html`). Open this file in your web browser to see your docs.

1.2 Building this documentation

To access this documentation offline, install the following python packages. We recommend using a python virtual environment (i.e., *venv*).

```
$ pip install sphinx
$ pip install sphinx-prompt
$ pip install sphinx-rtd-theme
```

Clone the CYAN Lab documentation repository in your project directory to hold your docs:

```
$ cd /path/to/project
$ git clone https://github.com/purduecyan/rtfm
$ cd rtfm
$ make html
```

The documentation will be built into your `build/html/` directory. Open the `index.html` file in your web browser to see your docs.

1.3 Updating Git Submodules

To update all git submodule repositories in the source folder:

```
$ cd source
$ git submodule update --remote
```

Next, add, commit and push the files to remote for changes to take effect.

1.4 External resources

Here are some external resources to help you learn more about Sphinx.

- Sphinx documentation
- An introduction to Sphinx and Read the Docs for technical writers

CHAPTER

TWO

ADDING CODE

2.1 Code Snippets

You can include code snippets using `code-block` directive as shown below:

```
.. code-block:: python

    # Some Python code
    print("Hello World!")
```

The above `code-block` will generate a code block with syntax highlighting for the specified language.

```
# Some Python code
print("Hello World!")
```

You can also use the `prompt` directive to display CLI commands. For example, to create a Bash prompt, use

```
.. prompt:: bash $

    sudo apt update && sudo apt upgrade
```

This will create a code block with a Bash prompt as shown below:

```
$ sudo apt update && sudo apt upgrade
```

2.2 Add Code Files

To display code directly from files, you can use the `literalinclude` directive.

```
.. literalinclude:: code/sample.py
  :language: python
  :linenos:
  :caption: sample.py
```

This will display the `sample.py` file located in the `code` directory and use Python syntax highlighting as shown below:

Listing 1: sample.py

```
class MyClass:
    """A simple example class"""
```

(continues on next page)

(continued from previous page)

```
    i = 12345

    @staticmethod
    def f():
        return 'hello world'
```

2.3 Add Git Repos

2.3.1 Add submodules

To add Git repositories to the documentation, navigate to the `source` folder and use

```
$ cd source
$ git submodule add <remote_url> <destination_folder>
```

2.3.2 Commit changes

Adding a Git submodule will stage your submodule. You should now commit your submodule by using the `git commit` command.

```
$ git commit -m "Added the submodule to the project."
$ git push
```

2.3.3 Update submodules

To update/pull a submodule, use the `git submodule update` command.

```
$ git submodule update --init --recursive
```

CHAPTER THREE

AUTHORS

This page lists the maintainers and contributors of the Purdue CYAN Lab documentation. Please email **nadig [AT] purdue [DOT] edu** to contribute to this project.

3.1 Maintainers

- Deepak Nadig, Purdue University, http://web.ics.purdue.edu/~nadig/.

3.2 Contributors

CHAPTER FOUR

CONTRIBUTE

- Source Code: https://github.com/purduecyan/rtfm
- Issue Tracker: https://github.com/purduecyan/rtfm/issues

CHAPTER

FIVE

CHANGELOG

Commit: 7f08cf4 — **Updated submodules.** — On (2024-02-03) — By <*Deepak Nadig*>

Commit: a8f56f4 — **Added sphinx-copybutton and sphinx-code-tabs.** — On (2024-02-03) — By <*Deepak Nadig*>

Commit: 9a50713 — **Updated Changelog.** — On (2024-01-28) — By <*Deepak Nadig*>

Commit: 4f62d90 — **Updated submodules.** — On (2024-01-28) — By <*Deepak Nadig*>

Commit: 5757b0d — **Updated configs** — On (2024-01-28) — By <*Deepak Nadig*>

Commit: cce3d2e — **Updated configs** — On (2024-01-28) — By <*Deepak Nadig*>

Commit: 74cf425 — **Updated submodule URLs** — On (2024-01-28) — By <*Deepak Nadig*>

Commit: 2c05b47 — **Added Multipass examples** — On (2024-01-28) — By <*Deepak Nadig*>

Commit: 88aaa80 — **Added two submodules - Vagrant and Cloud-init** — On (2024-01-26) — By <*Deepak Nadig*>

Commit: d14d933 — **Added PDF parts** — On (2022-03-10) — By <*Deepak Nadig*>

Commit: 4e8dd48 — **Updated documentation for adding code** — On (2022-03-10) — By <*Deepak Nadig*>

Commit: 2210e49 — **Changed .gitmodule URLs to https.** — On (2021-12-27) — By <*Deepak Nadig*>

Commit: d9a9251 — **Update submodule config.** — On (2021-12-27) — By <*Deepak Nadig*>

Commit: 7e189f1 — **Added CCI and SDN submodules (CNIT 481 Courses).** — On (2021-12-27) — By <*Deepak Nadig*>

Commit: 929459c — **Removed .idea files** — On (2021-12-23) — By <*Deepak Nadig*>

Commit: 5e492a9 — **Updated changelogs** — On (2021-12-23) — By <*Deepak Nadig*>

Commit: 1f1b4c2 — **Updated .readthedocs.yaml to build all formats** — On (2021-12-23) — By <*Deepak Nadig*>

Commit: 91cf6ab — **Added .readthedocs.yaml** — On (2021-12-23) — By <*Deepak Nadig*>

Commit: 29f1dc0 — **Updated links** — On (2021-12-23) — By <*Deepak Nadig*>

Commit: 78b8780 — **Updated conf.py** — On (2021-12-23) — By <*Deepak Nadig*>

Commit: 49afb9d — **Documentation ver. 1.0** — On (2021-12-23) — By <*Deepak Nadig*>

Commit: 3fddcc9 — **Added .gitignore** — On (2021-12-22) — By <*Deepak Nadig*>

Commit: 9de8a42 — **Initial commit** — On (2021-12-22) — By <*CYAN Lab | Purdue University*>

CHAPTER SIX

SUPPORT

For support requests, please create an issue at https://github.com/purduecyan/rtfm/issues.

Part II

Research Resources

CHAPTER

SEVEN

INFRASTRUCTURE

This page contains links for setting up various research and infrastructure resources used at CYAN Lab.

7.1 Multipass

7.1.1 Introduction

Multipass allows launching and managing lightweight virtual machines by leveraging cloud-init to customize the initial configuration of the virtual machines.

Cloud-init Configuration

Create a cloud-init configuration file, for example, `cloud-config.yaml`, with your desired settings. Here's an example:

```
# cloud-config.yaml
# Cloud-init configuration for Multipass instance

# Set the hostname
hostname: my-instance

# Add user
users:
  - name: myuser
    sudo: ALL=(ALL) NOPASSWD:ALL
    groups: users, admin
    home: /home/myuser

# Install packages
packages:
  - git
  - python3

# Run commands on instance startup
runcmd:
  - echo "Hello from Multipass Cloud-init!"
```

Launching Multipass Instance

Launch a Multipass instance using the cloud-init configuration:

```
multipass launch --cloud-init cloud-config.yaml my-instance
```

Replace `cloud-config.yaml` with the actual path to your cloud-init configuration file.

Verify that your Multipass instance was launched with the specified configurations using:

```
multipass list
```

Alternatively, you can open a shell and monitor cloud-init progress. In a separate terminal, you can connect to the VM by running:

```
multipass shell my-instance
```

and then observe cloud-init progress using

```
tail -f /var/log/cloud-init-output.log
```

Conclusion

You've successfully used cloud-init with Multipass to customize the configuration of your virtual machine upon launch. Feel free to explore more cloud-init options to tailor the setup according to your requirements.

7.1.2 Cloud-init with Options

Multipass Launch Options

- `--name` *instance-name*: Specify the name of the instance.
- `--cpus` *count*: Set the number of CPUs for the instance.
- `--mem` *size*: Set the amount of memory for the instance.
- `--disk` *size*: Set the disk size for the instance.
- `--cloud-init` *file*: Provide a cloud-init configuration file.
- `--timeout` *duration*: Set the maximum time allowed for instance creation.

Example Command

Launch a Multipass instance named "my-instance" with 2 CPUs, 8GB of memory, 10GB disk, and using a cloud-init configuration file named "cloud-config.yaml". Set a timeout of 30 minutes:

```
multipass launch --name my-instance --cpus 2 --mem 8G --disk 10G --cloud-init cloud-
↪config.yaml --timeout 30m
```

Explanation

- `--name my-instance` sets the instance name to "my-instance".
- `--cpus 2` allocates 2 CPUs to the instance.
- `--mem 8G` allocates 8GB of memory to the instance.
- `--disk 10G` sets the disk size to 10GB.
- `--cloud-init cloud-config.yaml` provides the cloud-init configuration file.
- `--timeout 30m` sets a timeout of 30 minutes for instance creation.

Conclusion

You have successfully launched a Multipass instance with custom CPU, memory, and disk options using cloud-init, and a specified timeout.

7.2 Vagrant

Vagrant is a useful tool to automate and provision virtual machines. A vagrant repository is available at https://github.com/purduecyan/vagrant

To install Vagrant on an Ubuntu machine with VirtualBox as the VM provider, use:

```
$ sudo apt update
$ sudo apt install virtualbox virtualbox-guest-additions-iso vagrant
```

To initialize a Vagrant environment, create and navigate to a folder that will store your vagrant file and run

```
$ vagrant init
```

You can now edit the `Vagrantfile` to configure and provision your VM. To create the VM(s),

```
$ vagrant up
```

Once the VMs are created, you can login to a VM using `vagrant ssh`. You can also stop all running VMs withing a project using

```
$ vagrant halt
```

To delete all the VMs created, run

```
$ vagrant destroy
```

7.2.1 Additional resources

You can find more information about Vagrant at https://www.vagrantup.com.

Part III

Courses

CHAPTER

EIGHT

CNIT 481: CLOUD COMPUTING INFRASTRUCTURE

8.1 Cloud-init

8.2 Authors

This page lists the maintainers and contributors of the Purdue CYAN Lab documentation. Please email **nadig [AT] purdue [DOT] edu** to contribute to this project.

8.2.1 Maintainers

- Deepak Nadig, Purdue University, http://web.ics.purdue.edu/~nadig/.

8.2.2 Contributors

CHAPTER

NINE

CNIT 481: SOFTWARE DEFINED NETWORKS

9.1 SDN Labs

Coming soon.

9.2 Authors

This page lists the maintainers and contributors of the Purdue CYAN Lab documentation. Please email **nadig [AT] purdue [DOT] edu** to contribute to this project.

9.2.1 Maintainers

- Deepak Nadig, Purdue University, http://web.ics.purdue.edu/~nadig/.

9.2.2 Contributors