Open Source Education - developer’s guide

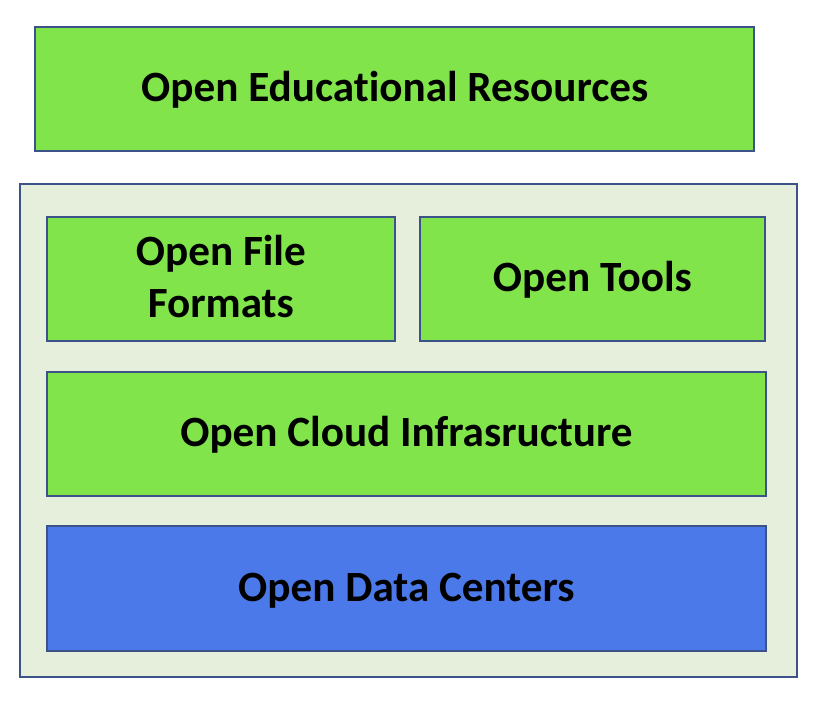
# Introduction

This project is an initiative towards making education open source, and to achieve highly interactive teaching and presentation material.

# OPE Philosophy

* Leverage modern open source technologies to create an open environment and platform in which

1. Educators can create, publish and operationalize high quality open source materials
2. Students require no more than access to a web browser to make use of the materials



* Loosen the grip that publishers have on the distribution of professional educational content
* Put the power to build, publish, and collaboratively develop such content in the hands of the educators who create it
* Provide tooling that makes the use of this power a manageable endeavor
* Meant to provide a templated environment for authoring and publishing open source content
* Provide structure in a way that is not perspective and not reliant on proprietary formats of any kind
* First class support for collaboration, replication and continual improvement of open source educational content

## The platform

The open ownership model starts with high-performance, open data centers providing the hardware resources. We exploit Linux to enable the use of cloud platforms on top of this hardware. Linux serves as the foundation, allowing us to build a rich environment of tools and service to support a novel approach to educational material.

## The Environment

Communities collaborating in the transparent authoring, editing and use of Course materials: Textbooks, Lectures, Lab manuals.

# Implementation

From the implementation perspective, the project has two significant components.

1. Infrastructure - Provide required support for the textbook content, building

source-to-image builder image/templates for the jupyter notebooks.

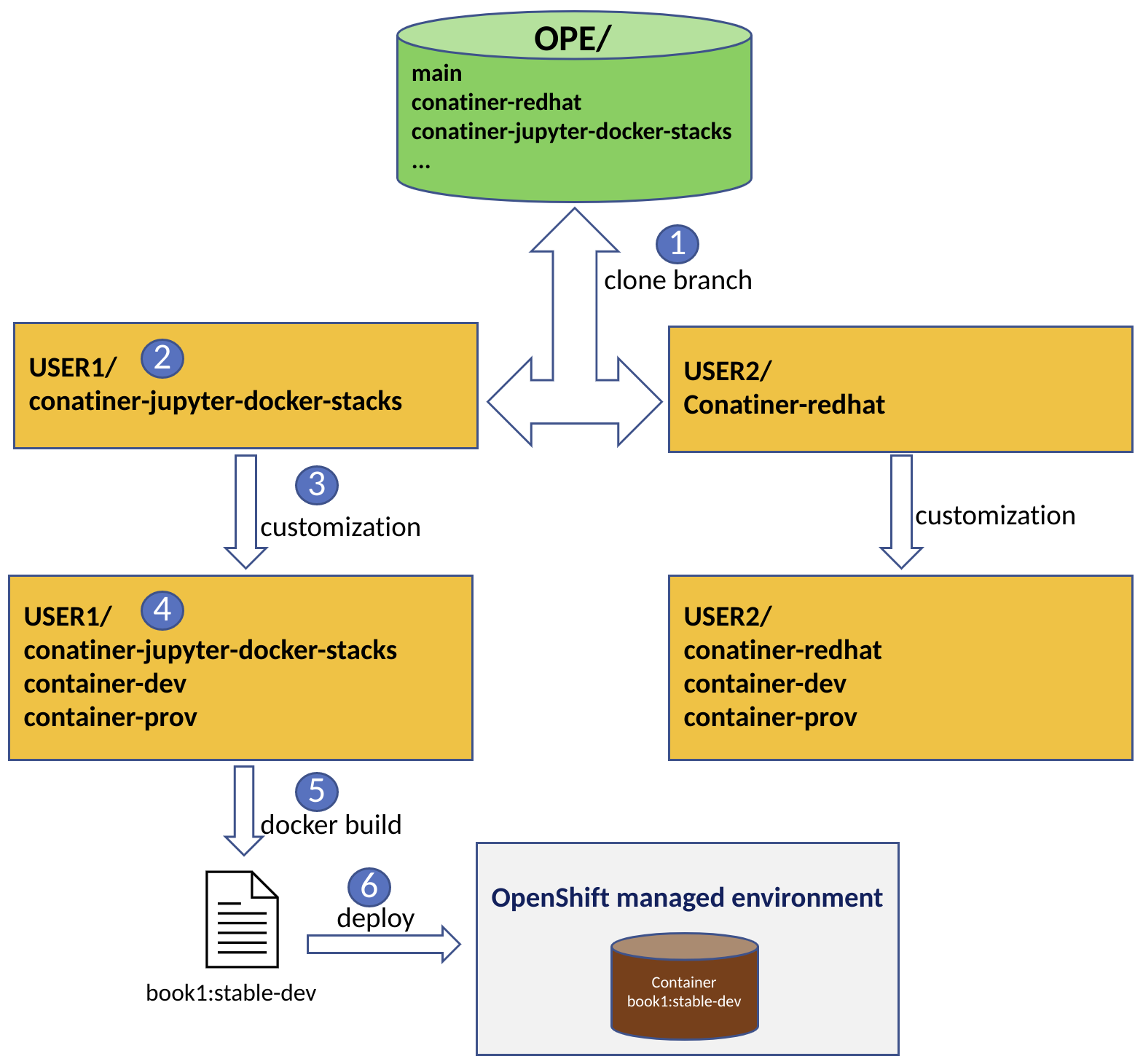
1. Textbook content - Actual teaching material based on JupyterBook with rich

presentations etc.

# Different stages in developing a new textbook template

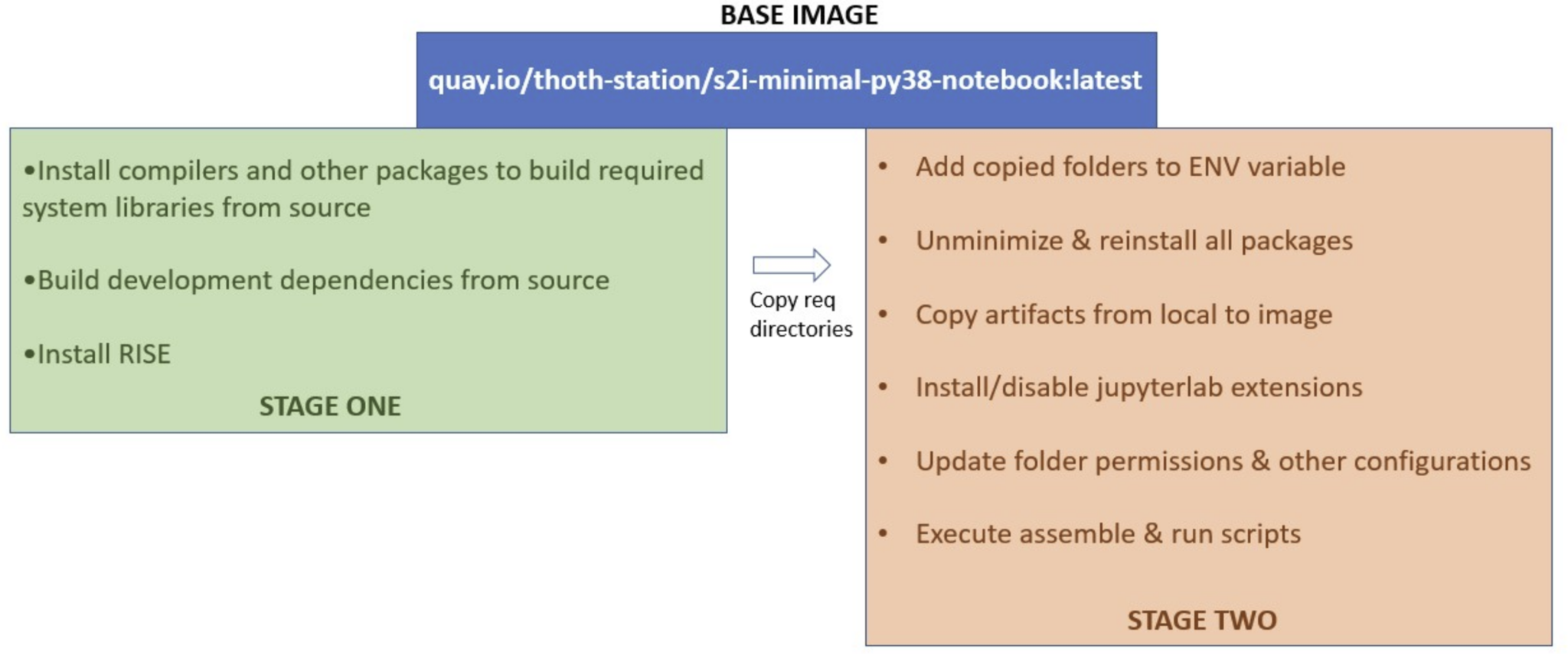
[OPE repo](https://github.com/OPEFFORT/ope) contains the base fedora-based and ubuntu based templates as branches for the jupyter notebook builder-image. This is the starting point and the subsequent steps are as below:

1. Choose the required base template i.e fedora or ubuntu based images. Clone the respective branch to your local repository.
2. Update configurations related to target image registry, docker image tags etc. Add new layers to the Dockerfile to include additional libraries and other configurations to meet the requirements.
3. Create new branches for every deploying environment and update the **ope\_uid** file with that of the environment user id.
4. Build the docker image with the respective customization (CUST) option. Export to docker image registries and make the repository publicly visible.
5. Add the new image details in the OpenShift environment and deploy.



# Jupyter notebook templates

We have created a fedora and an ubuntu based jupyter notebook template. The container template is created over two stages to achieve better organization and to reduce size.



Stage one:

* Add libraries required to install/build development dependencies

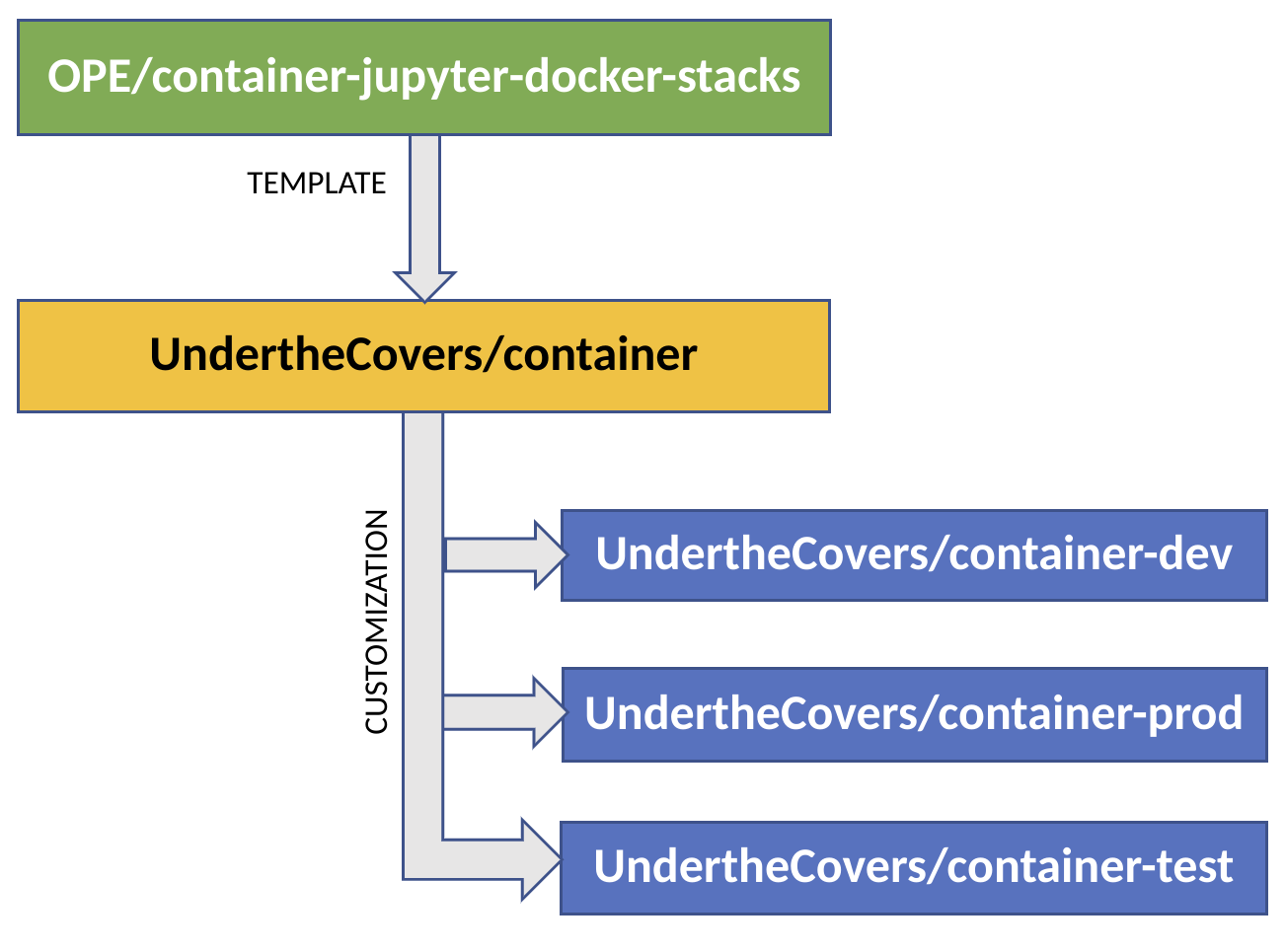
Stage two:

* Copy the final binary folders from stage-one.
* Include artifacts(executables, configs) from local
* Enable jupyter extensions
* Folder permissions

## Ubuntu based template

Source code base : <https://github.com/AbiShanna/UndertheCovers/tree/container>

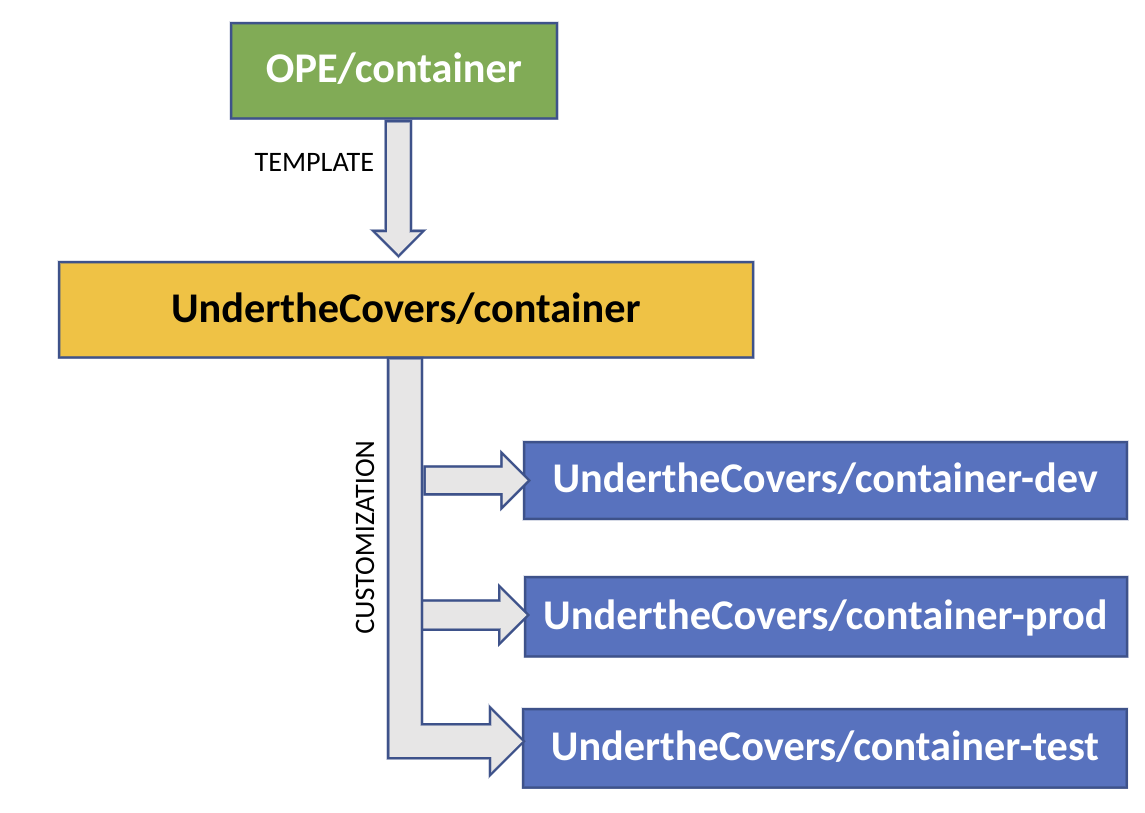
The Ubuntu based template is created using the open source [Jupyter Book project](https://jupyterbook.org/) and the [executablebooks/cookiecutter-jupyter-book template](https://github.com/executablebooks/cookiecutter-jupyter-book).



## Fedora based template

Source code base - <https://github.com/AbiShanna/ope/tree/container-redhat>

This generic template is built using Red Hat’s Univerbal Base Image (UBI). The base image [s2i-minimal-jupyter-notebook](https://github.com/thoth-station/s2i-minimal-notebook) is customized with multiple layers of softwares to support author, build and publish textbooks, extensions to notebook and configurations that can be deployed in an OpenShift managed cloud environment.



To customize and author materials, one can clone the OPE/container branch and can from there create more branches based on the local repo for each

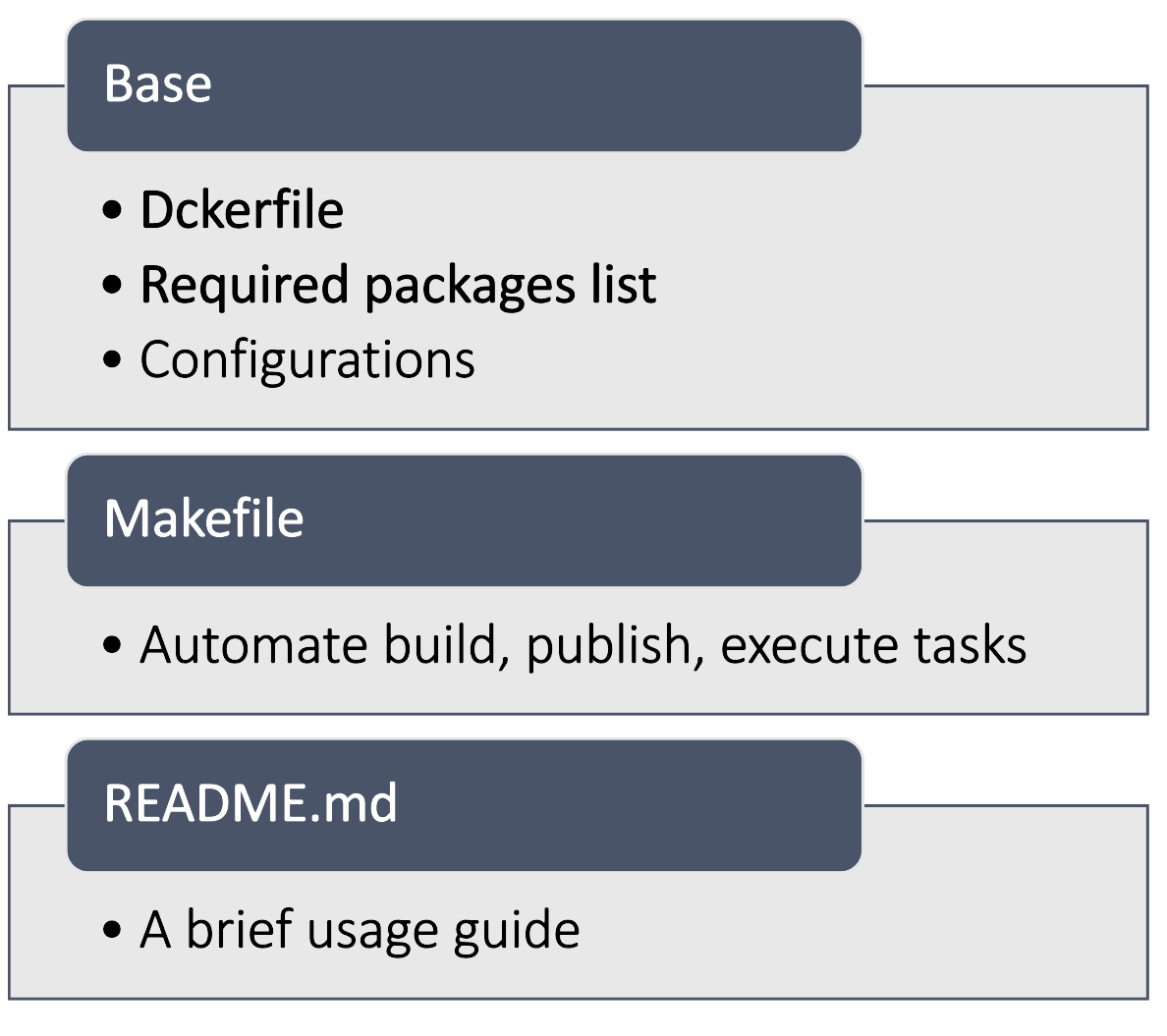
environment like development, testing, production etc.

# Understanding the artifacts involved in the build step

Both the fedora and the ubuntu based repository has the following folder structure.

## Base

All the configuration files are placed in this folder.



## Makefile

## Dockerfile

The build command invokes the actual docker build process with a list of arguments passed from multiple configuration files. The assemble and run scripts are from the s2i-minimal-notebook image of thoth-station. The assemble script installs the provided list of python libraries, basic jupyterlab extensions and the run script is executed during the run time which creates the user account, sets the required folder structure etc.

# Notes specific to fedora image:

* Source-to-image:

[Source-to-image](https://github.com/openshift/source-to-image) is a framework for building reproducible container images from

source code. S2I produces ready-to-run images by injecting source code into a container image and creating a new image that runs the assembled application.



The jupyter notebook templates created will be ingested with the textbook content and the final containers will be made available for the students to use.

* These images are aimed to be deployed in an OpenShift managed cloud environment. The default home directory of the image will be ‘/opt/app-root/src’. The persistent volume provided by the environment is also by default mapped to the same ‘/opt/app-root/src’ directory.
* Understanding micropipenv:

Unlike the Ubuntu based image, where we list the python libraries with versions to be installed, the s2i process utilizes a new wrapper tool called *micropipenv* to handle the installations. ‘Pipfile’, ‘Pipfile.lock’ is parsed by Thamos tool. After checking it installs the listed libraries.

* Understanding the Thamos tool:
* Understanding the ‘generate\_container\_user’ script present under *‘/opt/app-root/etc’*

During the execution of ‘run’ script, if the current user is not root/default(1001), this script creates an entry to /etc/passwd with no home directory and sets the required permissions.

# Resources

* Open Education (OPE) repository - <https://github.com/OPEFFORT/ope>
* BU CAS CS210 course material repository of Prof. Jonathan Appavoo - <https://github.com/jappavoo/UndertheCovers>
* Basic docker commands:
  + To authenticate quay.io :

*docker login quay.io*

* + To get the size consumed by each layer of the docker image:

*docker history --format "{{.ID}}: {{.CreatedBy}}: {{.Size}}" --no-trunc <docker\_image>*

Some Git commands:

1. To add file from one git branch to another:

git checkout work

git checkout master -- utils

git add utils

git commit -m "Adding 'utils' directory from 'master' branch."

1. To add a branch from diff repo:

git remote add fork <url of fork>

git fetch fork <branch>

git checkout -b fork\_branch fork/<branch>