Software Engineering Theory and Practice

UNIVERSITY OF PORTSMOUTH

School of Computing

Title	Software Engineering Theory and Practice
Module Coordinator	Steven Ossont
Email	steven.ossont@port.ac.uk
Code	M30819
Moodle	https://moodle.port.ac.uk/course/view.php?id=11429

Schedule and Deliverables

Item	Value	Format	Outcomes	Deadline
Portfolio10	1%	GitHub Repo	Pass/Fail	07 May 2021, 23:00

Notes and Advice

- The Extenuating Circumstances procedure is there to support you if you have had any circumstances (problems) that have been serious or significant enough to prevent you from attending, completing or submitting an assessment on time.
- The UNION Extenuating Circumstances Extenuating Circumstances procedure
- ASDAC are available to any students who disclose a disability or require additional support for their academic studies with a good set of resources on the ASDAC Moodle site
- The University takes plagiarism seriously. Please ensure you adhere to the plagiarism guidelines. Examination Regulations (http://regulations.docstore.port.ac.uk/ExamRegs12AssessmentOffences.pdf).
- Any material included in your coursework should be fully cited and referenced in APA format (seventh edition). Detailed advice on referencing is available from http://referencing.port.ac.uk/
- Any material submitted that does not meet format or submission guidelines, or falls outside of the submission deadline could be subject to a cap on your overall result or disqualification entirely.
- If you need additional assistance, you can ask your personal tutor, learning support ana.baker@port.ac.uk and xia.han@port.ac.uk or your lecturers.

Git commands

> Your repository will be copied for marking automatically at the deadline. EDITS after the deadline are automatically ignored.

When you update the `CheckList.md` file. This will trigger an action to inspect the work for this part of the Portfolio.(You can also manually run the `PortfolioAdvisor` if you need to see if changes solve any errors)

Files, external to this repo and any images imported via URL will be ignored (Even if they are stored in GitHub).

Large blocks of text that use the 'CODE' formatting will be ignored. This includes using triple "' (Unless it is code or something sensible). If it looks like you are using "' to contain Markdown to avoid the lint checker, it will be ignored. Code blocks should be labeled with the code language so that the syntax highlighter is enabled.

Here is a helpful Markdown link: https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet#code

At no point in this Assessment should you use the `Add file` button on the GitHub webpage – Pretend it does not exist

Marking is performed on a Ubuntu machine. This means that everything is case sensitive, where it may not be on Windows.

> Clone, Edit, Commit, Push (Preferably on the command line)

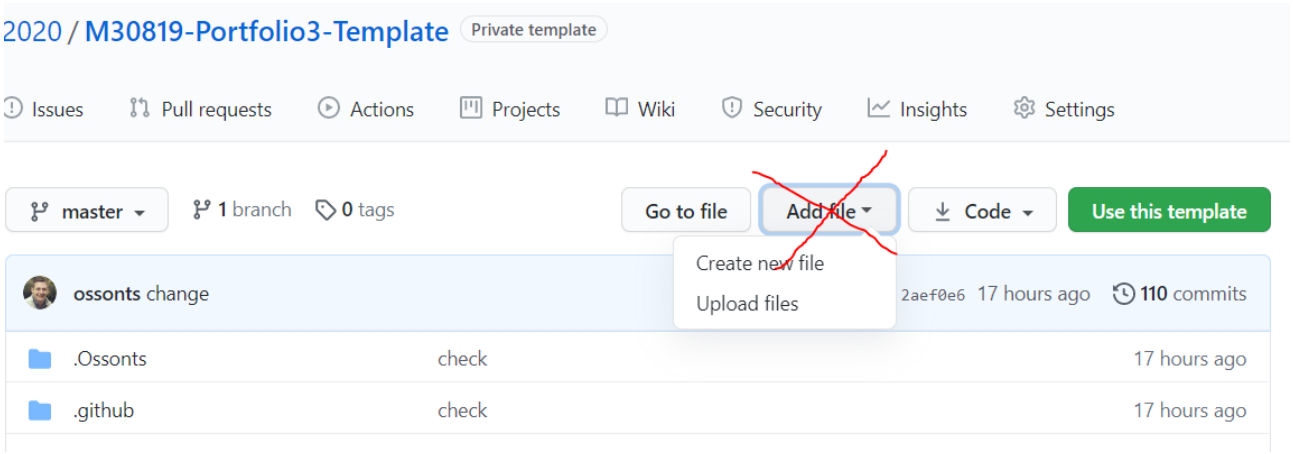

Figure 1: AddFile.png

Objectives

- Installing applications on Linux / VM
- Understand Docker at a basic level
- Learn the basics of Singularity
- Operate a container

Portfolio 10 Part 1

Create a file with the filename `Student.id`, add your Student ID to the content of this file. Note:

- Invalid ID = No marks
- The file extension is `.id` other file extensions e.g. `.txt` are not permitted
- The content of the file should be your ID ONLY, e.g. `UP1234567`
- You may have a return character at the end of the line `\n` (Note this is not two ASCII text characters)
- You need the `UP`
- Filename is case sensitive
- File contents are case sensitive
- File should contain one line of text only

- Markdown formatting is NOT permitted e.g. `*` or `-`

When you have completed this part of the portfolio :

- Add any files that are needed for this part (if any), to your GitHub repo E.g. new files that you created
- Put an `X` in the `CheckList.md` to indicate this part of the Portfolio is complete
- Commit and **push** your changes to GitHub
- Manually run the `PortfolioAdvisor` action, and ensure it passes. (Must PASS)
- Navigate to the `PortfolioAdvisor` action on GitHub.com and read the feedback / errors / comments
- Manually run the `Pandoc` action, and check the resulting PDF is as expected. (Must PASS)
- Manually run the `MarkdownChecker` action, this will be superseded by the super-linter. (Must PASS)
- Check you have an `X` in the `CheckList.md` to indicate this part of the Portfolio is complete
- Commit and **push** any changes you have made to pass the `PortfolioAdvisor` to GitHub
- Be sure to address any errors from the GitHub actions
- You do not need to wait for the `PortfolioAdvisor` to complete, if it is running slow. Proceed and come back and look at the checker errors/warnings when it completes.

Portfolio 10 Part 2

If you have an application/code that you need to run; you can simply install it on your local machine and run. This has limitations, especially if you require libraries or programmes to be installed as part of the OS; or if you need multiple OS/versions of OS; or if the install is complex/time consuming.

The solution to making this smoother is to use a container (similar to a light weight virtual machine). The main container technology used today is called Docker and for example GitHub actions supports Docker containers. This way you can run Testing/CI/CD that requires libraries or tooling that GitHub does not support.

You can find the Docker Handbook here: https://docker.farhan.info/

1. Great a file called `Docker.md` on the root of this repo. Add to your `.pandoc.yml`
2. Add 5 markdown bullet points to your `Docker.md` file. List 5 reasons you might want to use a Docker container. Read this introduction to containerization and Docker: https://docker.farhan.info/introduction-to-containerization-and-docker

   You may even want to install docker
3. Add a sixth bullet point to your `Docker.md` file. List one key disadvantage to using a Docker container

When you have completed this part of the portfolio :

- Add any files that are needed for this part (if any), to your GitHub repo E.g. new files that you created
- Put an `X` in the `CheckList.md` to indicate this part of the Portfolio is complete
- Commit and **push** your changes to GitHub
- Manually run the `PortfolioAdvisor` action, and ensure it passes. (Must PASS)
- Navigate to the `PortfolioAdvisor` action on GitHub.com and read the feedback / errors / comments
- Manually run the `Pandoc` action, and check the resulting PDF is as expected. (Must PASS)
- Manually run the `MarkdownChecker` action, this will be superseded by the super-linter. (Must PASS)
- Check you have an `X` in the `CheckList.md` to indicate this part of the Portfolio is complete
- Commit and **push** any changes you have made to pass the GitHUb actions
- Be sure to address any errors from the GitHub actions
- You do not need to wait for the `PortfolioAdvisor` to complete, if it is running slow. Proceed and come back and look at the checker errors/warnings when it completes.

Portfolio 10 Part 3

Read the whole of Part 3 before typing commands; then pick the solution that best suits your setup.

There is another container technology called **Singularity**.

1. Install Singularity **v3.X** on a machine (v2.x is in many package managers but will **not** work for your needs). Source code: https://github.com/hpcng/singularity. You may want to consider the following [Complete install took 17min - come to a drop in if you have issues]
   - On a **Linux machine**; Linux VM; WSL

Install from source: https://sylabs.io/guides/3.7/admin-guide/admin_quickstart.html#installation-from-source

First you will need to install `Go` (be sure to check your architecture)

```
$ export VERSION=1.15.11 OS=linux ARCH=amd64
$ wget https://dl.google.com/go/go$VERSION.$OS-$ARCH.tar.gz
$ sudo tar -C /usr/local -xzvf go$VERSION.$OS-$ARCH.tar.gz
$
```

Ensure you can run `go`:

```
$ echo 'export GOPATH=${HOME}/go' >> ~/.bashrc
$ echo 'export PATH=/usr/local/go/bin:${PATH}:${GOPATH}/bin' >> ~/.bashrc
$ source ~/.bashrc
$ go version
go version go1.15.11 linux/amd64
$
```

> `go` version 1.16.X may cause issues later. Stick to version 1.15.X if you can. If you already have `Go` installed, only change version if you have problems later.

Download singularity:

```
$ wget https://github.com/hpcng/singularity/releases/download/v3.7.3/singularity-3.7.3.tar.gz
$ tar -xzf singularity-3.7.3.tar.gz
$
```

then install singularity :

```
$ cd singularity
$ ./mconfig
$ make -C ./builddir
$ sudo make -C ./builddir install
$
```

… singularity is now installed.

- **WSLv2** on Windows (Remember WSL on Windows can simplify things).

  The linux instructions above work on WSL2. >You need **WSLv2**. https://askubuntu.com/questions/1177729/wsl-am-i-running-version-1-or-version-2 >*In WSL1,* `uname -r` *ends with -Microsoft, in WSL2, it ends* `-microsoft-standard-WSL2` *… lower case "m"!*

- Allegedly it is possible on a **Mac** (Although a VM would be simpler)

  https://sylabs.io/singularity-desktop-macos/

- There is a **Windows** option (Virtualbox + Vagrant)

  This would be considered a **last resort** and worthy of those who like a challenge https://sylabs.io/guides/3.0/user-guide/installation.html#windows. If you are considering the Windows option please reconsider and install WSLv2 https://docs.microsoft.com/en-us/windows/wsl/install-win10#manual-installation-steps

Only proceed once you have singularity installed.

2. Create a file called `Singularity.md` on the root of this repo. Add to your `.pandoc.yml`.
3. Document the commands you used to install Singularity on your machine. (Similar to the notes above for installing on a Linux VM – these notes may be handy in the future if you need to install Singularity fast!) [Use a Markdown heading; Fenced code; and use the fenced code type `shell`]
4. Find out which version of Singularity you are running. Add both the command **and** output to your `Singularity.md` file. [Use a Markdown heading; Fenced code; and use the fenced code type `shell`]

   For example:

   ```
   $ singularity --version
   singularity version 3.7.3
   $
   ```

5. Read this comparison article: introduction to Docker vs Singularity https://pythonspeed.com/articles/containers-filesystem-data-processing/

When you have completed this part of the portfolio :

- Add any files that are needed for this part (if any), to your GitHub repo E.g. new files that you created
- Put an `X` in the `CheckList.md` to indicate this part of the Portfolio is complete
- Commit and **push** your changes to GitHub
- Manually run the `PortfolioAdvisor` action, and ensure it passes. (Must PASS)
- Navigate to the `PortfolioAdvisor` action on GitHub.com and read the feedback / errors / comments
- Manually run the `Pandoc` action, and check the resulting PDF is as expected. (Must PASS)
- Manually run the `MarkdownChecker` action, this will be superseded by the super-linter. (Must PASS)
- Check you have an `X` in the `CheckList.md` to indicate this part of the Portfolio is complete
- Commit and **push** any changes you have made to pass the `PortfolioAdvisor` to GitHub
- Be sure to address any errors from the GitHub actions
- You do not need to wait for the `PortfolioAdvisor` to complete, if it is running slow. Proceed and come back and look at the checker errors/warnings when it completes.

Portfolio 10 Part 4

1. Read this document on building a container. https://singularity.lbl.gov/docs-build-container

   You can think of the singularity container `file system` as being a file systems that is *overlayed* on top of your existing folder structure. The first command to learn is `build` it can take a variety of file types/recipes and can output a variety of container types.

   You can find some singularity libraries here https://cloud.sylabs.io/library

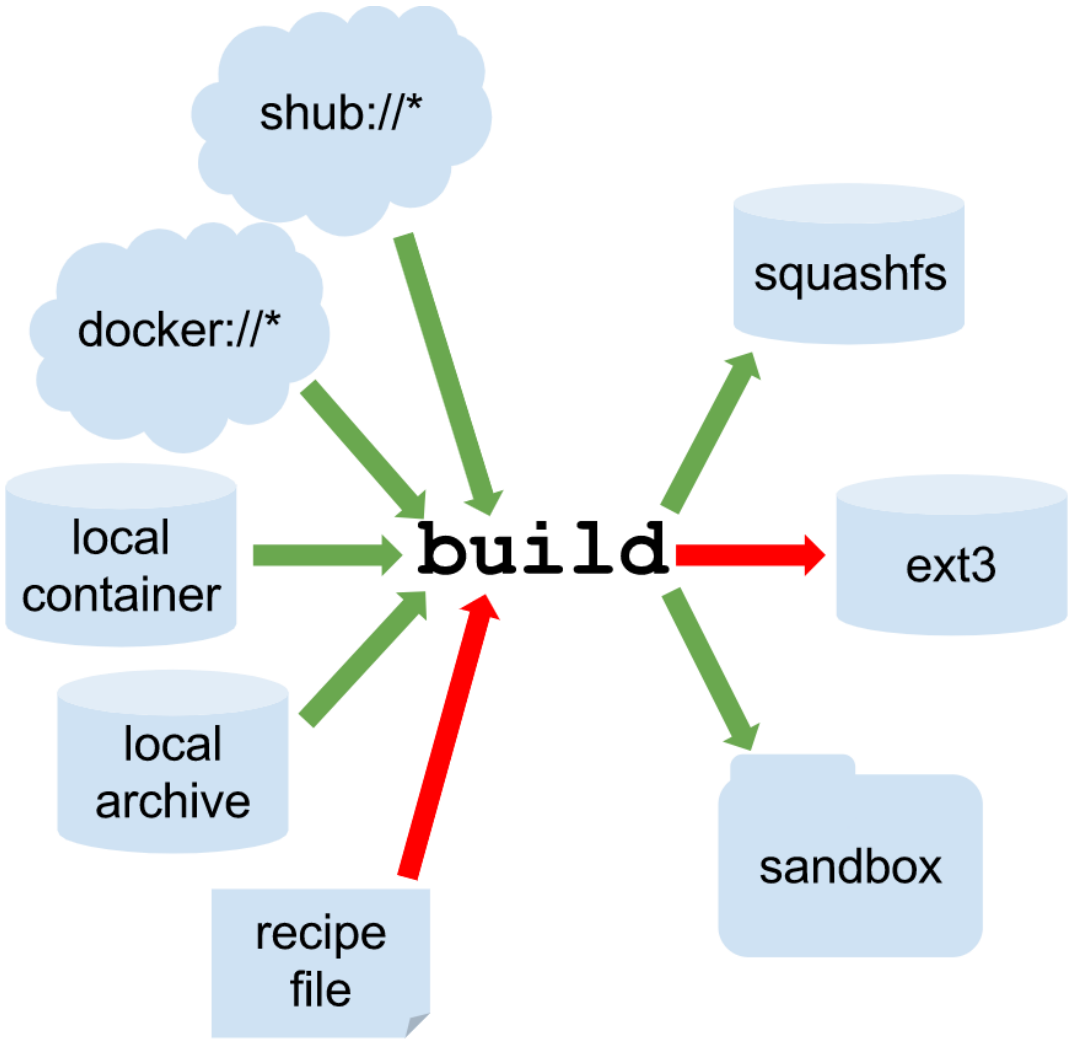

Figure 2: build image

2. There are many ways to build a container, let us have a look at a simple way to get `alpine` OS https://alpinelinux.org/about/ up and running in a container.

   In this example we have Ubuntu running with WSLv2 **(host)** and intend to run alpine linux in a **container**.

   > Run singularity `build` commands as `sudo` when you are creating a container.

   Build an alpine container using the command below. This will look up the default library image for Alpine v3 [We use Alpine as it is small and really good fun]:

   ```
   $ singularity build  alpine.img  library://library/default/alpine:3
   $
   ```

3. The command above will generate you a container image file called `alpine.img`. It is 2.7M. Run `ls` on your new container and add the `ls` command **and** output to your `Singularity.md` file. [Use a Markdown heading; Fenced code; and use the fenced code type `shell`]

   Here is an example:

   ```
   $ ls -alh alpine.img
   -rwxr-xr-x 1 so so 2.7M Apr 22 15:11 alpine.img
   $
   ```

4. You can run this image simply by executing the image file (or using the full singularity command). Type `exit` to leave the container. Repeat the example below using the image you have just generated. Add the

commands and output to your `Singularity.md` file [Use a Markdown heading; Fenced code; and use the fenced code type `shell`]

For example:

```
$ ./alpine.img
Singularity> exit
$ singularity shell alpine.img
Singularity> exit
$
```

Add, commit and push your `alpine.img` image to git.

5. Inspect your alpine image and put the command and metadata into your `Singularity.md` file. [Use a Markdown heading; Fenced code; and use the fenced code type `shell`] For example:

```
$ singularity inspect alpine.img
org.label-schema.build-arch: amd64
org.label-schema.build-date: Thursday_22_April_2021_15:11:55_BST
org.label-schema.schema-version: 1.0
org.label-schema.usage.singularity.deffile.bootstrap: library
org.label-schema.usage.singularity.deffile.from: library/default/alpine:3
org.label-schema.usage.singularity.version: 3.7.3
$
```

When you have completed this part of the portfolio :

- Add any files that are needed for this part (if any), to your GitHub repo E.g. new files that you created
- Put an `X` in the `CheckList.md` to indicate this part of the Portfolio is complete
- Commit and **push** your changes to GitHub
- Manually run the `PortfolioAdvisor` action, and ensure it passes. (Must PASS)
- Navigate to the `PortfolioAdvisor` action on GitHub.com and read the feedback / errors / comments
- Manually run the `Pandoc` action, and check the resulting PDF is as expected. (Must PASS)
- Manually run the `MarkdownChecker` action, this will be superseded by the super-linter. (Must PASS)
- Check you have an `X` in the `CheckList.md` to indicate this part of the Portfolio is complete
- Commit and **push** any changes you have made to pass the `PortfolioAdvisor` to GitHub
- Be sure to address any errors from the GitHub actions
- You do not need to wait for the `PortfolioAdvisor` to complete, if it is running slow. Proceed and come back and look at the checker errors/warnings when it completes.

Portfolio 10 Part 5

The image in the previous part would be good if it contained **everything** we wanted, but usually you would want to build a container and customize the applications/data files/scripts. The `alpine.img` we built is readonly (default).

1. Rebuild the image, this time using the sandbox option. [Read the docs]

```
$ sudo singularity build --sandbox  alpine  library://library/default/alpine:3
$
```

Do not give this build the same name as before (E.g. remove the file extension – otherwise it will get hard to know if we are talking about the image `file` or the `folder` with the same name)

Note this time instead of getting a single file you generate a folder with all of the files that will make up the image. (much easier for editing)

```
$ ls  alpine/
bin  environment  home  media  opt   root  sbin
srv  tmp  var  dev  etc  lib   mnt   proc  run
singularity  sys  usr
$
```

Add the command and output of `ls -al alpine/` to your `Singularity.md` file. [Use a Markdown heading; Fenced code; and use the fenced code type `shell`]

Do not add this image **folder** to Git [Remove if you did]

2. The sandbox version of your container is writable. Please install Python2 (not v3) in your container. Remember alpine uses a different package manager than you may be familiar with (Ubuntu). Here is an example:

```
$ sudo singularity shell --writable alpine
Singularity> python
/bin/sh: python: not found
Singularity> python2
/bin/sh: python2: not found
Singularity> python3
/bin/sh: python3: not found
Singularity> apk add --update --no-cache python2
fetch http://dl-cdn.alpinelinux.org/alpine/v3.11/main/x86_64/APKINDEX.tar.gz
fetch http://dl-cdn.alpinelinux.org/alpine/v3.11/community/x86_64/APKINDEX.tar.gz
(1/9) Installing libbz2 (1.0.8-r1)
(2/9) Installing expat (2.2.9-r1)
(3/9) Installing libffi (3.2.1-r6)
(4/9) Installing gdbm (1.13-r1)
(5/9) Installing ncurses-terminfo-base (6.1_p20200118-r4)
(6/9) Installing ncurses-libs (6.1_p20200118-r4)
(7/9) Installing readline (8.0.1-r0)
(8/9) Installing sqlite-libs (3.30.1-r2)
(9/9) Installing python2 (2.7.18-r0)
Executing busybox-1.31.1-r9.trigger
OK: 46 MiB in 23 packages
Singularity>
```

Now if we run Python it executes Python2; the only python installed in your container.

```
Singularity> python
Python 2.7.18 (default, May  3 2020, 20:31:45)
[GCC 9.2.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
[CTRL+D]
Singularity> exit
$
```

Control-D will exit Python shell and `exit` command will exit singularity shell.

Add the command and output of `python --version` in your container; to your `Singularity.md` file. [Use a Markdown heading; Fenced code; and use the fenced code type `shell`]

3. You can see the files that are in your image by looking in the folder: For example:

```
$ ls -al alpine/usr/bin/pyth*
lrwxrwxrwx 1 root root     7 Apr 22 15:32 alpine/usr/bin/python -> python2
lrwxrwxrwx 1 root root     9 Apr 22 15:32 alpine/usr/bin/python2 -> python2.7
-rwxr-xr-x 1 root root 13992 May  3  2020 alpine/usr/bin/python2.7
$
```

Add the command and output of `ls -al alpine/usr/bin/pyth*`; to your `Singularity.md` file. [Use a Markdown heading; Fenced code; and use the fenced code type `shell`]

4. Unlike Docker; singularity has access to your local (outside of container) files by default.

   Other than when building a container as sudo (root), you should run the container as local user. This prevents privilege escalation.

In this repo there is an example Python script that you **cannot change**. `HelloWorld.py` was written in python2 and will only run in python2. There are breaking changes between Python2 and Python3. One such change is that the `print` method now requires brackets. It was strange having one method that does not require brackets but all others do, so now it is fixed!

Have a look at the script. Here is the python script being executed:

```
$ python2 HelloWorld.py
```

```
Hello Old World!
$ python3 HelloWorld.py
  File "HelloWorld.py", line 1
    print "Hello Old World!"
          ^
SyntaxError: Missing parentheses in call to 'print'. Did you mean print("Hello Old World!")?
$
```

If you want to run this script you need python2; but it is not a good idea to install Python2. So … containers to the rescue.

Do not install Python2 on your machine!

Instead use the alpine container to execute the script. Use the `singularity run` command [Read the docs]. Add the command and output to your `Singularity.md` file. [Use a Markdown heading; Fenced code; and use the fenced code type `shell`]

For example:

```
$ singularity run alpine python HelloWorld.py
Hello Old World!
$
```

> The command above runs a container called `alpine`, executing the `python` executable inside the container, passing rest of the command as parameters.

When you have completed this part of the portfolio :

- Add any files that are needed for this part (if any), to your GitHub repo E.g. new files that you created
- Put an `X` in the `CheckList.md` to indicate this part of the Portfolio is complete
- Commit and **push** your changes to GitHub
- Manually run the `PortfolioAdvisor` action, and ensure it passes. (Must PASS)
- Navigate to the `PortfolioAdvisor` action on GitHub.com and read the feedback / errors / comments
- Manually run the `Pandoc` action, and check the resulting PDF is as expected. (Must PASS)
- Manually run the `MarkdownChecker` action, this will be superseded by the super-linter. (Must PASS)
- Check you have an `X` in the `CheckList.md` to indicate this part of the Portfolio is complete
- Commit and **push** any changes you have made to pass the `PortfolioAdvisor` to GitHub
- Be sure to address any errors from the GitHub actions
- You do not need to wait for the `PortfolioAdvisor` to complete, if it is running slow. Proceed and come back and look at the checker errors/warnings when it completes.

Portfolio 10 Part 6

1. Once we have all the applications installed in our image we can convert it into a single read only file. It is bad practiced to generate containers adhoc; the process must be repeatable. For this we need a recipe (definition file). We will build a definition file for our alpine and python image.

   Read: https://sylabs.io/guides/3.0/user-guide/definition_files.html#overview

   Create a file `alpine.def` (add/commit/push)

2. Update your `alpine.def` file to make sure the definition file bootstraps a `library` called `alpine:3` .

   For example:

   ```
   $ cat alpine.def
   bootstrap: library
   from: alpine:3
   $
   ```

3. Build this definition file to make a single image (not as a sandbox).

   > Note call this image `alpine2.img` (Do not overwrite your image from before)

   ```
   $ sudo singularity build alpine2.img alpine.def
   INFO:    Starting build...
   INFO:    Using cached image
   INFO:    Verifying bootstrap image /root/.singularity/cache/library/sha256.03883ca565b32e58fa0a496
   ```

```
WARNING: integrity: signature not found for object group 1
WARNING: Bootstrap image could not be verified, but build will continue.
INFO:    Creating SIF file...
INFO:    Build complete: test
$
```

4. Install python2 in your container; this time use the definition file. (edit `alpine.def`). Rebuild the container.

   ```
   $ cat alpine.def
   bootstrap: library
   from: alpine:3
   %post
     apk add --update --no-cache python2
   ```

   Rebuild the container, this time you will see Python gets installed inside the container.

   ```
   $ sudo singularity build alpine2.img alpine.def
   INFO:    Starting build...
   INFO:    Using cached image
   INFO:    Verifying bootstrap image /root/.singularity/cache/library/sha256.03883ca565b32e58fa0a496
   WARNING: integrity: signature not found for object group 1
   WARNING: Bootstrap image could not be verified, but build will continue.
   INFO:    Running post scriptlet
   + apk add --update --no-cache python2
   fetch http://dl-cdn.alpinelinux.org/alpine/v3.11/main/x86_64/APKINDEX.tar.gz
   fetch http://dl-cdn.alpinelinux.org/alpine/v3.11/community/x86_64/APKINDEX.tar.gz
   (1/9) Installing libbz2 (1.0.8-r1)
   (2/9) Installing expat (2.2.9-r1)
   (3/9) Installing libffi (3.2.1-r6)
   (4/9) Installing gdbm (1.13-r1)
   (5/9) Installing ncurses-terminfo-base (6.1_p20200118-r4)
   (6/9) Installing ncurses-libs (6.1_p20200118-r4)
   (7/9) Installing readline (8.0.1-r0)
   (8/9) Installing sqlite-libs (3.30.1-r2)
   (9/9) Installing python2 (2.7.18-r0)
   Executing busybox-1.31.1-r9.trigger
   OK: 46 MiB in 23 packages
   INFO:    Creating SIF file...
   INFO:    Build complete: test
   $
   ```

   Git add; commit; push your `alpine2.img` image

5. Run a singularity `shell` using your new image; execute python inside your container (so it shows the version) and add the process to your`singularity.md` file. [Use a Markdown heading; Fenced code; and use the fenced code type `shell`]

   For example:

   ```
   $ singularity shell alpine2.img
   Singularity> python
   Python 2.7.18 (default, May  3 2020, 20:31:45)
   [GCC 9.2.0] on linux2
   Type "help", "copyright", "credits" or "license" for more information.
   >>>
   Singularity> exit
   $
   ```

6. You now have a custom image, with alpine OS installed and python2. Remember you can just execute your image (so you don't need the singularity command keyword). It will open a shell in your container by default. Instead of running a shell by default you can customize and run a script or executable by default. For example Python2 !

   Edit your image definition file so that it has a `%runscript` section.

```
$ cat alpine.def
bootstrap: library
from: alpine:3
%post
  apk add --update --no-cache python2
%runscript
  exec python "$@"
```

This example `runscript` will execute the command `python` (remember we just have python2 installed inside the container) and the `"$@"` will take any other parameters that are passes to the container and put them on the end of this command (e.g. pass any container parameters to the `python` executable).

Are you ready for this?

Now you can execute your container and pass a local file; it will execute inside the container; and return the output.

```
$ ./alpine2.img HelloWorld.py
Hello Old World!
```

Repeat the example above using **your** container. Add the command **and** output to your`singularity.md` file. [Use a Markdown heading; Fenced code; and use the fenced code type `shell`]

When you have completed this part of the portfolio :

- Add any files that are needed for this part (if any), to your GitHub repo E.g. new files that you created
- Put an `X` in the `CheckList.md` to indicate this part of the Portfolio is complete
- Commit and **push** your changes to GitHub
- Manually run the `PortfolioAdvisor` action, and ensure it passes. (Must PASS)
- Navigate to the `PortfolioAdvisor` action on GitHub.com and read the feedback / errors / comments
- Manually run the `Pandoc` action, and check the resulting PDF is as expected. (Must PASS)
- Manually run the `MarkdownChecker` action, this will be superseded by the super-linter. (Must PASS)
- Check you have an `X` in the `CheckList.md` to indicate this part of the Portfolio is complete
- Commit and **push** any changes you have made to pass the `PortfolioAdvisor` to GitHub
- Be sure to address any errors from the GitHub actions
- You do not need to wait for the `PortfolioAdvisor` to complete, if it is running slow. Proceed and come back and look at the checker errors/warnings when it completes.

Portfolio 10 Part 7

Singularity supports docker files; so you can take a Dockerfile and build a singularity image. For example these portfolios are executed in a Docker container via github.

The docker image is store here: https://github.com/orgs/M30819-2020/packages/container/package/setapdocker

In **this** repo you have the `Dockerfile` that is used to build the docker image. It is possible to use singularity to build an image based on a `Dockerfile`. Or there are tools that will convert to a singularity definition file.

Both docker (`Dockerfile`) and singularity (`Singularity.def`) native image definition/recipe files are supplied in this repo. Have a look at them both.

1. Build the `Singularity.def` image that is used to mark your repo. [Note it contains all the tooling, not the marking source code]

   The resulting image will be ~1.3G ; do **not** commit this image to your repo.

   Execute a singularity `shell` in your SETAP container and execute the following:

   1. `latex -v`
   2. `gh --version`

   Add the command and output to your `Singularity.md` file. [Use a Markdown heading; Fenced code; and use the fenced code type `shell`]

   For example:

```
$ sudo singularity build  setap.img Singularity.def
$ singularity shell setap.img
```

```
$ gh --version
gh version 9.9.9 (2021-04-20)
https://github.com/cli/cli/releases/tag/v1.9.2
$
```

2. Do not commit the 1.3G setap.img to your repo. Calculate the MD5 of your `setap.img` image and create a file with the MD5 as the name and the extension `md5`.

   For example:

   ```
   $ md5sum setap.img
   26b80f90e25eb468efc782333932d272
   $ touch 26b80f90e25eb468efc782333932d272.md5
   $ ls *.md5
   26b80f90e25eb468efc782333932d272.md5
   $
   ```

   Git add commit push your `md5` file

When you have completed this part of the portfolio :

- Add any files that are needed for this part (if any), to your GitHub repo E.g. new files that you created
- Put an `X` in the `CheckList.md` to indicate this part of the Portfolio is complete
- Commit and **push** your changes to GitHub
- Manually run the `PortfolioAdvisor` action, and ensure it passes. (Must PASS)
- Navigate to the `PortfolioAdvisor` action on GitHub.com and read the feedback / errors / comments
- Manually run the `Pandoc` action, and check the resulting PDF is as expected. (Must PASS)
- Manually run the `MarkdownChecker` action, this will be superseded by the super-linter. (Must PASS)
- Check you have an `X` in the `CheckList.md` to indicate this part of the Portfolio is complete
- Commit and **push** any changes you have made to pass the `PortfolioAdvisor` to GitHub
- Be sure to address any errors from the GitHub actions
- You do not need to wait for the `PortfolioAdvisor` to complete, if it is running slow. Proceed and come back and look at the checker errors/warnings when it completes.

Portfolio 10 Completed

You must run these actions [**Run these actions manually now**]:

- MarkdownChecker [Must pass]
- Pandoc [Must pass]
- PortfolioAdvisor [Must pass]

[Banners above are **EXPERIMENTAL**; do not rely on them]

- Commit and push your changes to GitHub

Complete the checklist and submit by putting an `X` in the portfolio checklist to indicate the Portfolio is complete and ready to mark.

> If you want to re-submit your work, any time before the deadline. You can un-tick the Portfolio Completed checkbox; **commit & push;** then re-tick Portfolio Completed checkbox; **commit & push.**

Once you have done this step, the commit hash of your latest commit should be in `.Ossonts/latestSHA.log`

- Double check that this commit hash (URL is included in `.Ossonts/latestSHA.log`) is the version you want to submit.
- If this commit hash is **incorrect**; Re-submit your portfolio. (See above).
- If you commit hash was **NOT** correct please raise an issue as well as resubmitting to correct the SHA.

When you have completed this part of the portfolio :

- Add any files that are needed for this part (if any), to your GitHub repo E.g. new files that you created
- Put an `X` in the `CheckList.md` to indicate this part of the Portfolio is complete
- Commit and **push** your changes to GitHub
- Manually run the `PortfolioAdvisor` action, and ensure it passes. (Must PASS)
- Navigate to the `PortfolioAdvisor` action on GitHub.com and read the feedback / errors / comments
- Manually run the `Pandoc` action, and check the resulting PDF is as expected. (Must PASS)

- Manually run the `MarkdownChecker` action, this will be superseded by the super-linter. (Must PASS)
- Check you have an `X` in the `CheckList.md` to indicate this part of the Portfolio is complete
- Commit and **push** any changes you have made to pass the `PortfolioAdvisor` to GitHub
- Be sure to address any errors from the GitHub actions
- You do not need to wait for the `PortfolioAdvisor` to complete, if it is running slow. Proceed and come back and look at the checker errors/warnings when it completes.

Your repository will be copied for marking automatically at the coursework deadline OR when the Completed checklist item is ticked; whichever is earliest

Checklist

- ☐ Portfolio 10 Part 1
- ☐ Portfolio 10 Part 2
- ☐ Portfolio 10 Part 3
- ☐ Portfolio 10 Part 4
- ☐ Portfolio 10 Part 5
- ☐ Portfolio 10 Part 6
- ☐ Portfolio 10 Completed

Please accept as submitted and READY to be marked.