

Portable Workflows II

CMSE 890-602

Portable workflows: Scripting languages

Considerations:

- Dependencies (interpreter version, shell version)
- Environments (conda, pip)
- Containers

Documenting dependencies

- Ensure the minimum and *maximum* compatible versions are listed
 - This can be the same version!
- Provide lists of required packages
 - Follow appropriate styles e.g. requirements.txt for pip
- Provide installation instructions for compatible operating systems
 - Which ideally is something like “conda env create -f env.yml”

Environments

- Include:
 - requirements.txt, a list of pip packages that are required, for pure Python workflows
 - Environment YAML file for conda
 - Package lists for R
 - Software lists for everything else
- Note that the conda environment YAML can include:
 - Conda packages
 - Pip packages (in a separate block)
 - R packages (for an R environment)

Containers

1. Choose a compatible base container OS
2. Use that to install dependencies
3. Install the environment into the container

These steps can apply to all dependencies of the workflow too!

REMINDER: Singularity will automatically mount useful local paths, Docker *does not*. NextFlow handles the Docker paths automatically.

Container setup: Docker

```
FROM debian:bullseye-slim
```

← Base image

```
LABEL image.author.name "Your  
Name Here"  
LABEL image.author.email  
"your@email.here"
```

← Author info

```
RUN apt-get update
```

← Get updates to the image OS

```
RUN apt-get install  
<software>
```

← Install dependencies

```
ENV PATH=$PATH:/usr/games/
```

← Add specific path to the environment

```
RUN conda create -f env.yaml
```

← Install environment
(after installing conda)

Building a Docker container

- Install Docker!
- Run `sudo docker build -t image-name path/to/Dockerfile`
- Make sure you have administrative privileges!
 - Assume all the other docker commands have sudo in front

Sharing a Docker container

- Create a dockerhub account at hub.docker.com
- Create a repository for your container
 - Make sure it is a useful name
 - Can be public or private (but you have to be able to authenticate to download your private containers)
- Run `docker login` and follow the prompts
- Tag your locally built image with your username
 - `docker tag image-name username/repository-name:versiontag`
- Push this tag to your repository
 - `docker push username/repository-name:versiontag`
- Now you can access your container anywhere!
 - Including on the HPC with singularity

Container setup: Singularity/Apptainer

```
Bootstrap: docker
From: debian:bullseye-slim

%environment
export PATH=$PATH:/usr/games/

%labels
AUTHOR <your name>

%post
apt-get update
apt-get install <software>
conda create -f env.yaml
```

Use dockerhub

Base image OS

Add specific path to the environment

Author info

Get updates to the image OS

Install dependencies

Install environment
(after installing conda)

Portable workflows: Snakemake

- Snakemake supports conda environments for each rule
 - Environments can be frozen with specific package choices
- Use containers with `--sdm apptainer`
 - Container image is referenced under the “container” rule directive
 - Can specify a single container for the entire workflow
 - No *direct* Docker support, all via Singularity/Apptainer!
 - But Docker containers are preferred
- Snakemake can *automatically* generate a container with

```
snakemake --containerize > Dockerfile
```

- This contains all necessary environments for each rule
 - Used with the global directive “containerized”
 - Requires conda environments for each rule
- <https://snakemake.readthedocs.io/en/stable/snakefiles/deployment.html#containerization-of-conda-based-workflows>

Snakemake style

- Snakemake has a recommended directory structure to follow
 - Git repository template available!
<https://github.com/snakemake-workflows/snakemake-workflow-template/generate>
- There is the built-in linter to check your Snakefile follows standard style
 - `snakemake --lint`

Portable workflows: NextFlow

- NextFlow supports conda and containers
- Uses different command line options for Docker and Singularity
- Typically prefers a single container with all dependencies
- Can combine a container with a conda env like SnakeMake
 - Requires manual Dockerfile creation
- As with SnakeMake, can apply a container to every process or per-process
 - Global: use NextFlow config file `process.container`
 - Per-process: use process directive `container`
 - Need to set the container type in the config with `docker.enabled` or `singularity.enabled = true`

Nextflow style

- Not actually any official or strictly defined style!
- There are various tutorials and recommendations
 - <https://training.nextflow.io/>
 - https://nf-co.re/docs/usage/tutorials/nf_core_usage_tutorial

Many containers or one container?

Many

Pros:

- Individual software easy to update
- Reusable across other workflows
- Easy to parallelize

Cons:

- Many files
- Possible repeated data

One

Pros:

- Just one file
- Easy setup
- No repetition

Cons:

- Hard to parallelize
- Hard to update without rebuilding entire container

In-class assignment

Please fill out the official course evaluation when you receive it! You should have an email with a personalized link.

Fill out this (anonymous) feedback form:

<https://forms.gle/EWeSdskWpuedbGpF8>

Semester project due midnight Dec 12th!

“Final exam” period 10am-12pm Dec 9th in my office,
Room 1450