Backpacking with Code: Software Portability for HTC

Andrew Owen

Slides adapted from Christina Koch, Rachel Lombardi

OSG User School 2025

Goals For This Session

- Describe what it means to make software “portable.”
- Compare and contrast software portability techniques.
- Choose the best portability technique for your software.
- Build a portable software environment.
  - Follow steps to build a container
  - Compile code on a Linux computer

Introduction


Imagine that we asked you to
go home and bake a cake…

An Analogy

Photo by jschantz on flickr, CC-BY

Running software on your own computer is like cooking in your own kitchen.

On Your Computer

- You know what is there.
  - All the software you need is already installed.
- You know where everything is (mostly).
- You have full control.
  - You can add new programs when and where you want.

OSPool: Other People's Computers

The Challenge

Running code on someone else's computer is like cooking in someone else's kitchen.

Photo by F Deventhal on Wikimedia, CC-BY

On Someone Else's Computer

- What's already there?
  - Is R installed? Or Python? What about the packages you need?
- If the software you need is installed, do you know where it is or how to access it?
- Are you allowed to change whatever you want?

The Solution

- Imagine going camping or backpacking – what do you need to do to cook anywhere?
- Similarly: take your software with you to any computer.
- This is what it means to make software portable.

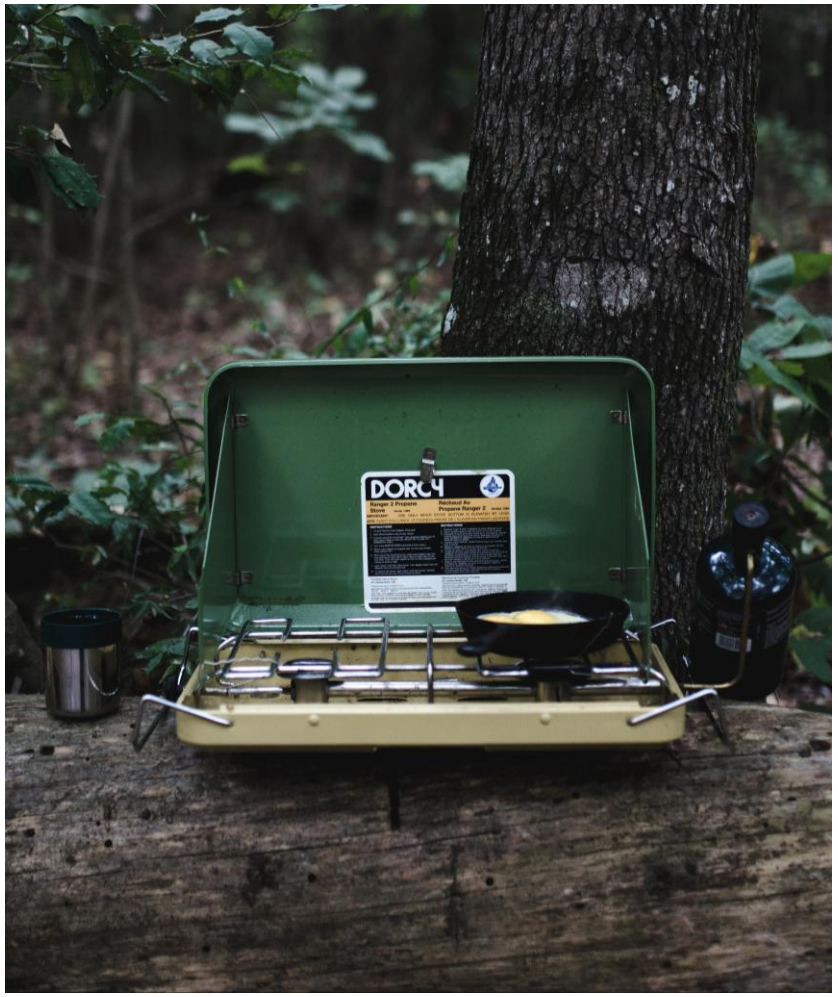


Photo by andrew welch on Unsplash

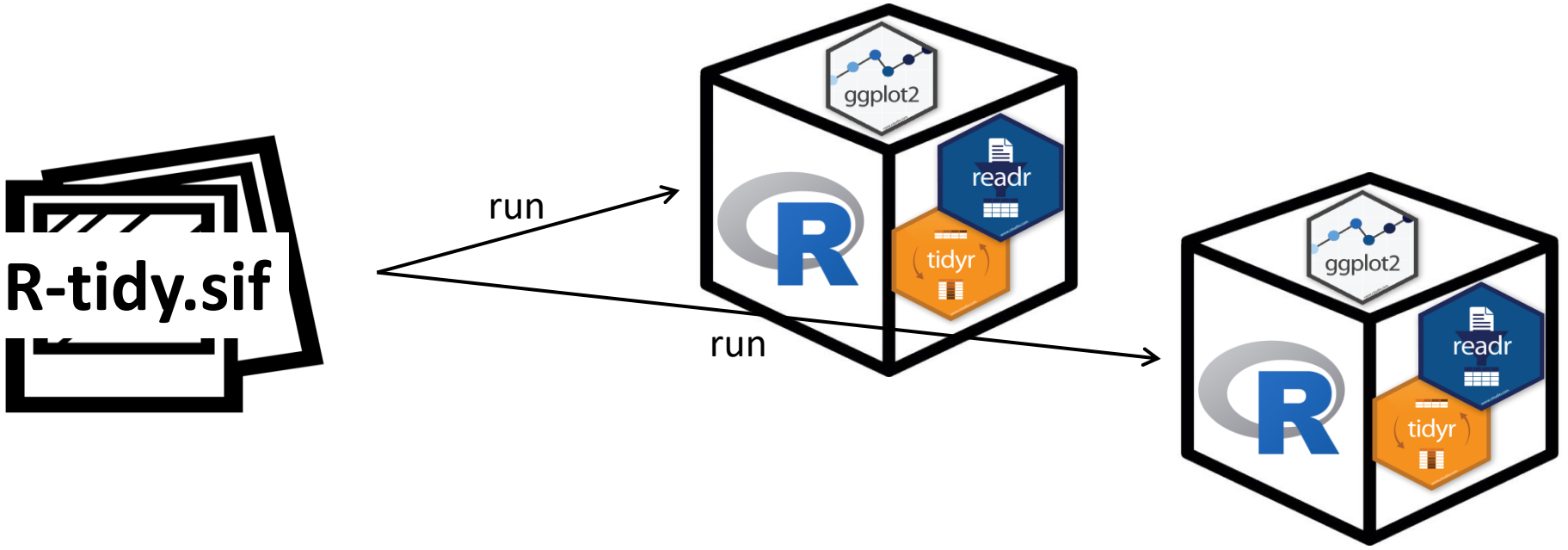
Option 1: Containers

Returning to Our Analogy…

Using a container is like bringing along a whole kitchen.

Containers

Containers are a tool for capturing an entire “environment” (software, libraries, operating system) into an “image” that can be run and used as the environment for a job



Credit: polaroid photos by Nick Bluth from the Noun Project

Why Use a Container?

Containers provide

- portability
- user control
- ease of use

and require no more technical skill than other methods of installing software!

Words for Containers

container = the actively running ~thing~ that can run commands

container image = is the set of files that can be used to create the container

"Launching" or "running" a container is the process of using the "container image" to create the running "container"

"Building" is the process of creating the "container image" (and often uses a pre-existing container image as the basis for the new one)

Technology for Containers

https://apptainer.org/

+ Open source software
+ Does not require root to install
+ Beginner friendly syntax
- Fewer features
- No distribution system

https://www.docker.com/

+ Modern app interface (Docker Desktop)
+ Widely used distribution system (DockerHub, analogous to GitHub)
- Non-intuitive syntax
- Requires root to install
- Commercial software

Technology for Containers

https://apptainer.org/

https://www.docker.com/

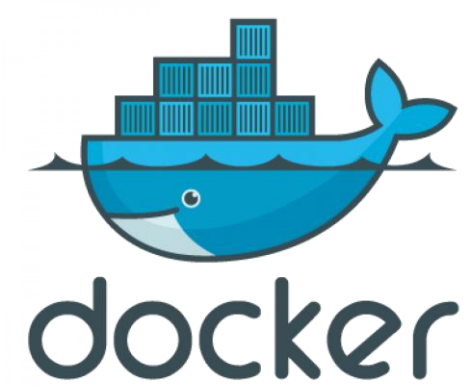
For beginners and those who only want to run the container on the OSPool

For advanced users and those who want to deploy container to many places

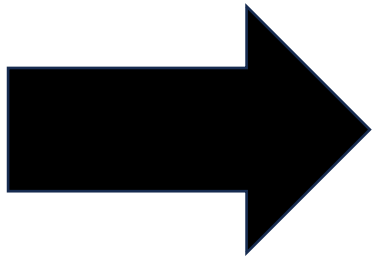
Use Existing Containers

- OSG provided: https://portal.osg-htc.org/documentation/htc_workloads/using_software/available-containers-list/
- OSG user provided (just a list, no descriptions): https://github.com/opensciencegrid/cvmfs-singularity-sync/blob/master/docker_images.txt
- Docker Hub: https://hub.docker.com/

Explore Containers



python:3.10



```
Apptainer> python3 --version
Python 3.10.14
```

```
$ apptainer shell docker://python:3.10
```

Demo

Print Python version on Access Point

```
ap40$ python3 --version
Python 3.9.19
```

Build and run an apptainer container running Python 3.10

```
$ apptainer shell docker://python:3.10
```

Print Python version inside Apptainer container

```
Apptainer> python3 --version
Python 3.10.14
```


Create Apptainer Definition File

Create a file container.def with the following contents:

```
Bootstrap: docker
From: python:3.13

%post
   python3 -m pip install cowsay
```


Create Apptainer Definition File

Create a file container.def with the following contents:

```
Bootstrap: docker
From: python:3.13

%post
    python3 -m pip install cowsay
```

Tells Apptainer to use the DockerHub "python" container with the tag of "3.13"*

The following commands should be used to modify the container

Install the "cowsay" Python package

*In the case of the "python" container, the "tag" equals the version of python inside.

Choosing a base container

Search for your software + "container".

- Most big-name softwares have official containers online

Once identified, find the container "address", usually represented in the form of a docker pull command:

```
docker pull user/repository:tag
```

Do not use the latest tag - choose an explicit one

What to install and where

- "Common" tools/libraries/packages are rarely included in "official" containers; be sure to add what you need/want
- Dynamic items (scripts, input data) should not be included in the container
- Recommend that user software is installed in the /opt folder

Build the Container Image

Build the container image file using the provided definition file

```
apptainer build container.sif container.def
```

desired filename *definition file*

Apptainer Build Output

Output details what Apptainer is doing:

1. Download the base container image
2. Launch the base container image
3. Execute the commands from the %post section
4. Create the new container image file

When finished, should see a message like

INFO: Build complete: container.sif

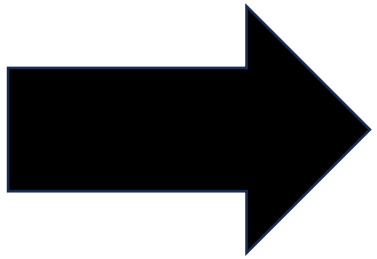
Test the Apptainer Image

Test the container.sif container by launching it with

```
apptainer shell -e container.sif
```

If you can get the "help" or "version" text for your desired program, should be good to go.

Test the Apptainer Image



```
Apptainer> cowsay "hello"
 _____
| Hello |
 =====
        \
         \
           ^__^
           (oo)\_______
           (__)\       )\/\
               ||----w |
               ||     ||
```

```
$ apptainer shell cowsay.sif
```

Using Containers in Jobs

For today:

```
container_image = cowsay.sif

...usual submit options...
```

That's it!

Using Containers in Jobs

Well… all other times:

```
container_image = osdf:///ospool/ap40/username/cowsay.sif

...usual submit options...
```

More on OSDF tomorrow!

Differences between local & cluster use

With HTCondor,

containers are always run ***as the user*** and ***never as root***

But…

official containers often assume you are running ***as root***!

When testing a container, ***test as the user***.

Common issues

Building Docker on a Mac (with ARM)
→ Make sure to include option --platform linux/amd64

Custom command not found
→ Manually add install location to your PATH at start of execution

Cannot create directory at /.cache
→ Manually set HOME or other environment variables at start of execution

Cannot write file at /app/data, etc.
→ Change program to use relative path for execution, not absolute

Learn How to Build/ Use Containers

- OSG User Documentation:
  https://portal.osg-htc.org/documentation/

- Videos:
  - https://portal.osg-htc.org/documentation/support_and_training/training/materials/
  - https://www.youtube.com/watch?v=awSLTflAIJ8

Option 2: Bring Along Software Files

Back to the Kitchen Analogy…

A backpacking approach instead of a portable kitchen

–

Only bring the absolute minimum!

Ways to Prepare Software Files

- Download pre-compiled software files
- Compile software yourself
  - Generate a single binary file
  - Create an installation with multiple binary files contained in a single folder

We always need a “compiled” file that is compatible with the version of Linux used by the execution point

Using Pre-Compiled Code

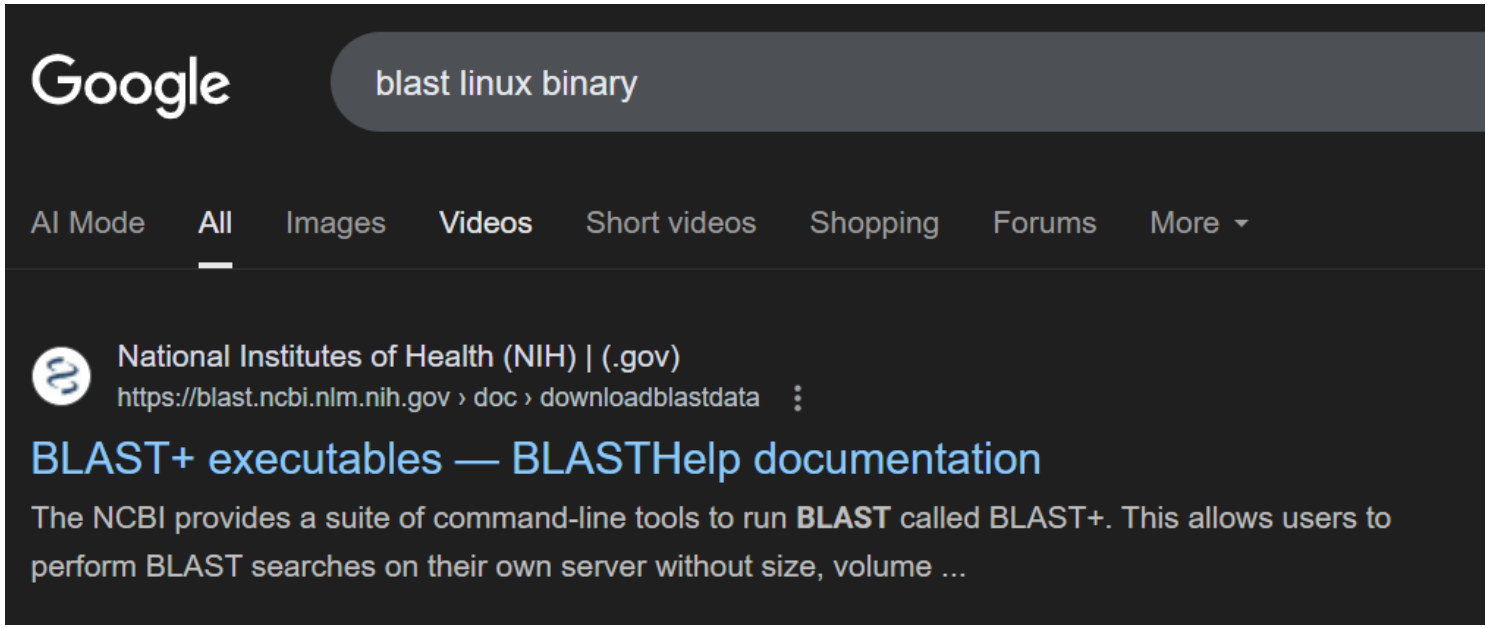
Some software providers have already done the hard work for you!

You just need to

1. Find the right file
2. Download and extract the file

Find Pre-Compiled Code

- Search for your software and "Linux binary"
- Look for a **.tar.gz** file with keyword **Linux** and **x86_64**



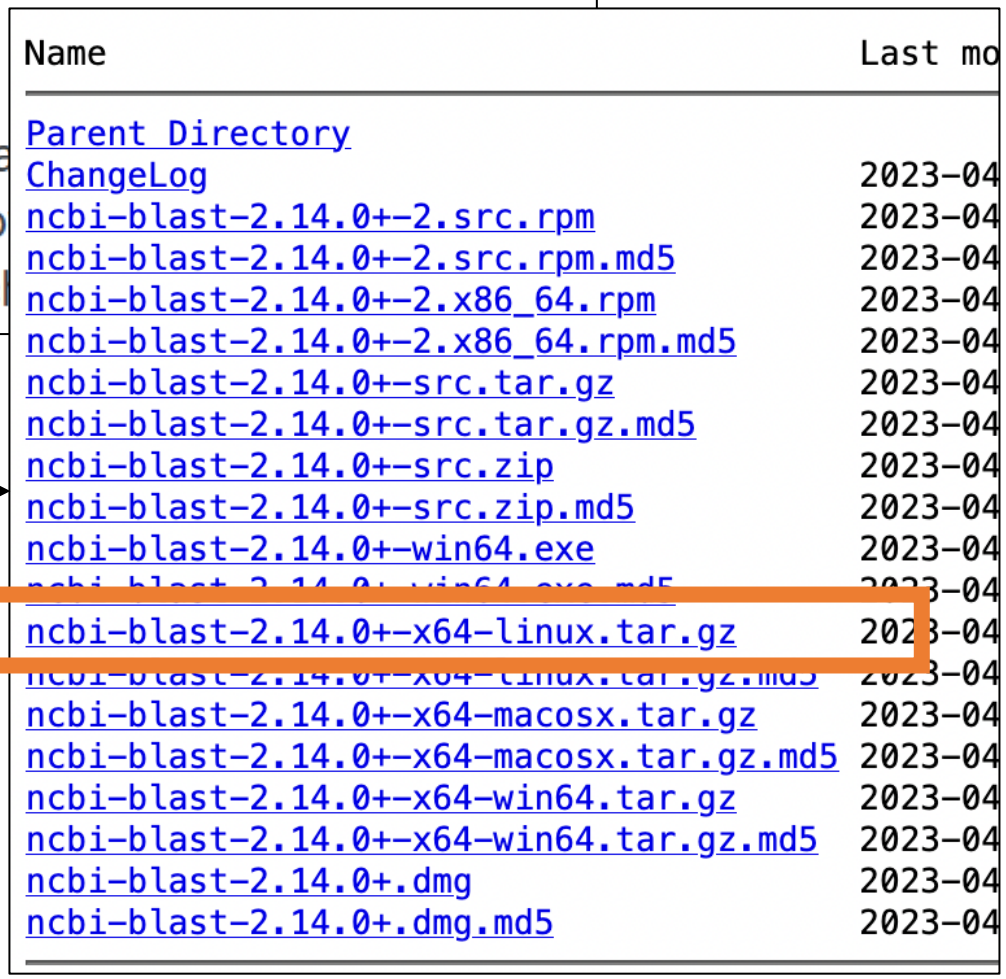
Find Pre-Compiled Code

What are the next steps?

Download and install BLAST+. Installers and source code are availa

https://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/ Do

need,(see database section below), or create your own. Start searc

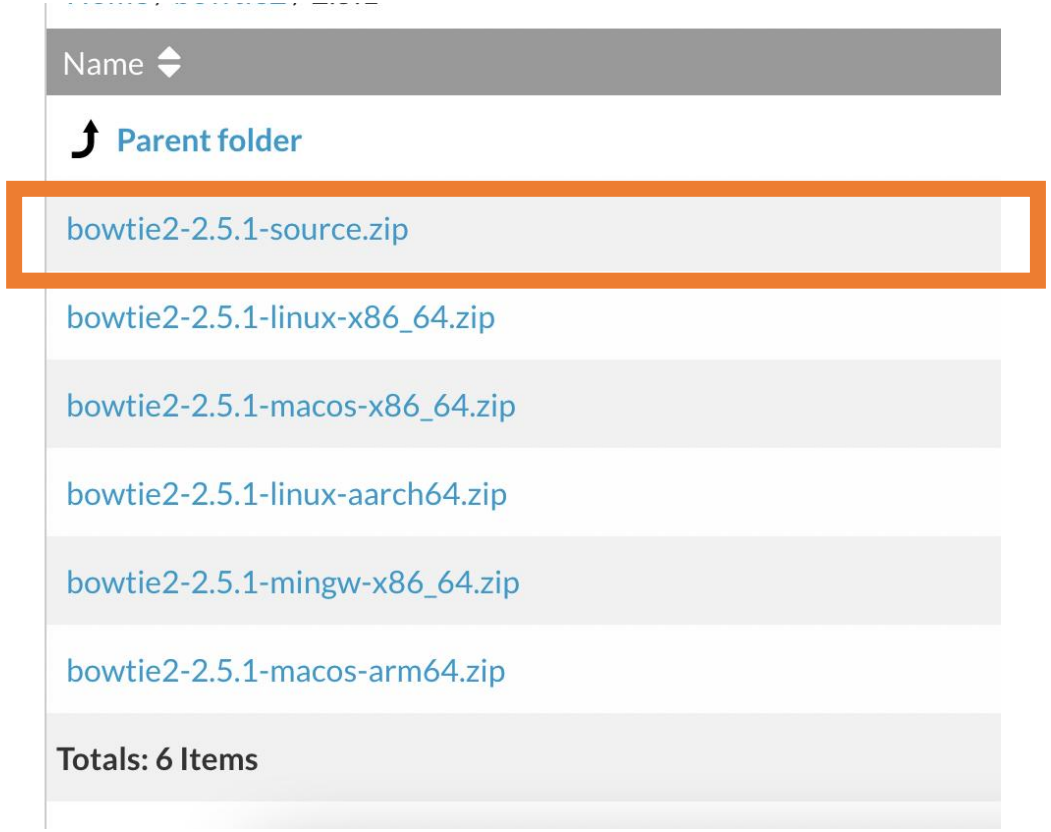


Name	Last mo
Parent Directory	
ChangeLog	2023-04
ncbi-blast-2.14.0+-2.src.rpm	2023-04
ncbi-blast-2.14.0+-2.src.rpm.md5	2023-04
ncbi-blast-2.14.0+-2.x86_64.rpm	2023-04
ncbi-blast-2.14.0+-2.x86_64.rpm.md5	2023-04
ncbi-blast-2.14.0+-src.tar.gz	2023-04
ncbi-blast-2.14.0+-src.tar.gz.md5	2023-04
ncbi-blast-2.14.0+-src.zip	2023-04
ncbi-blast-2.14.0+-src.zip.md5	2023-04
ncbi-blast-2.14.0+-win64.exe	2023-04
ncbi-blast-2.14.0+-x64-linux.tar.gz	2023-04
ncbi-blast-2.14.0+-x64-macosx.tar.gz	2023-04
ncbi-blast-2.14.0+-x64-macosx.tar.gz.md5	2023-04
ncbi-blast-2.14.0+-x64-win64.tar.gz	2023-04
ncbi-blast-2.14.0+-x64-win64.tar.gz.md5	2023-04
ncbi-blast-2.14.0+.dmg	2023-04
ncbi-blast-2.14.0+.dmg.md5	2023-04

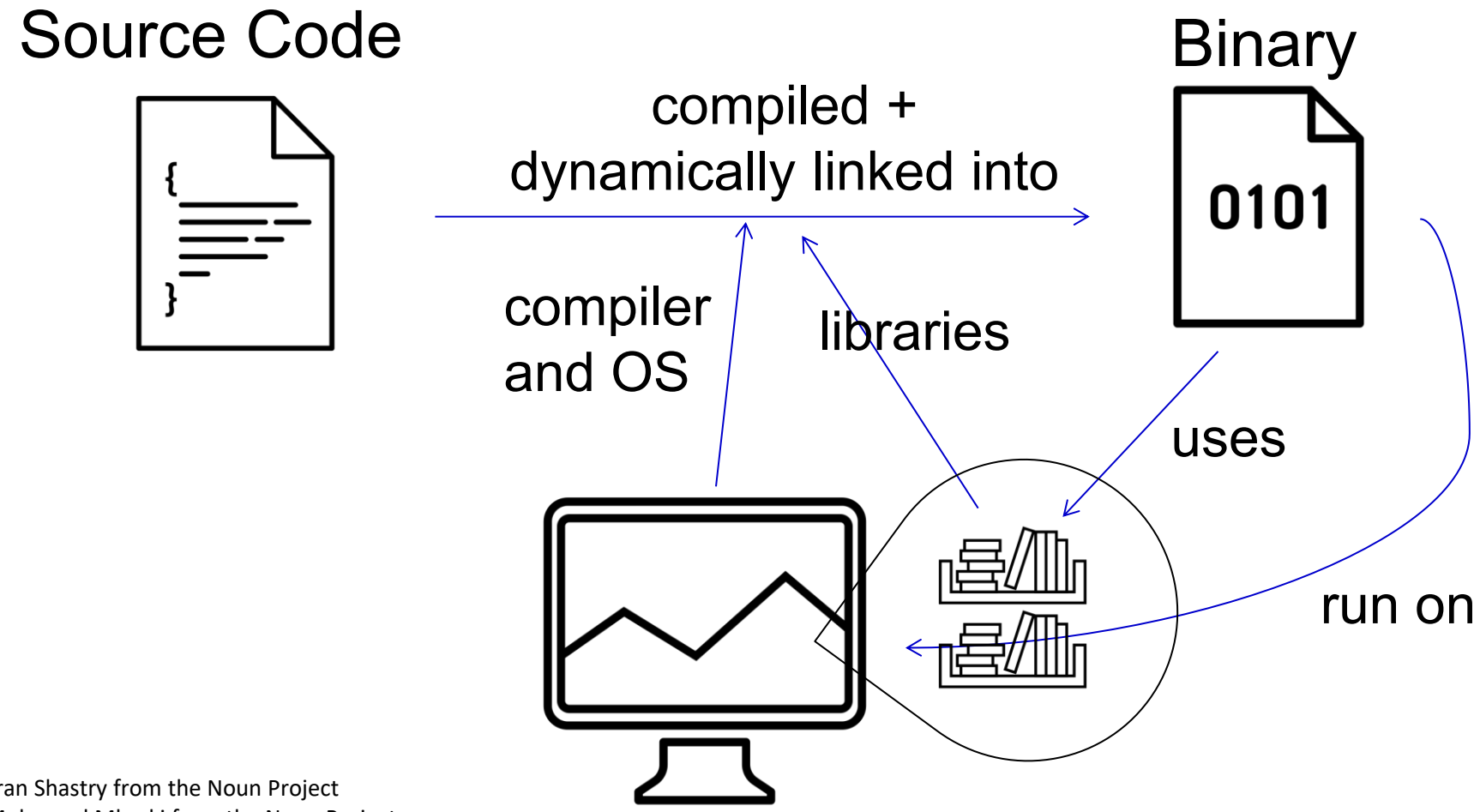
The exact process is different for each software!

Compiling from Source Code

- Finding source code can be similar process
- Once downloaded, usually need to extract the code
- May need to clone a git repository

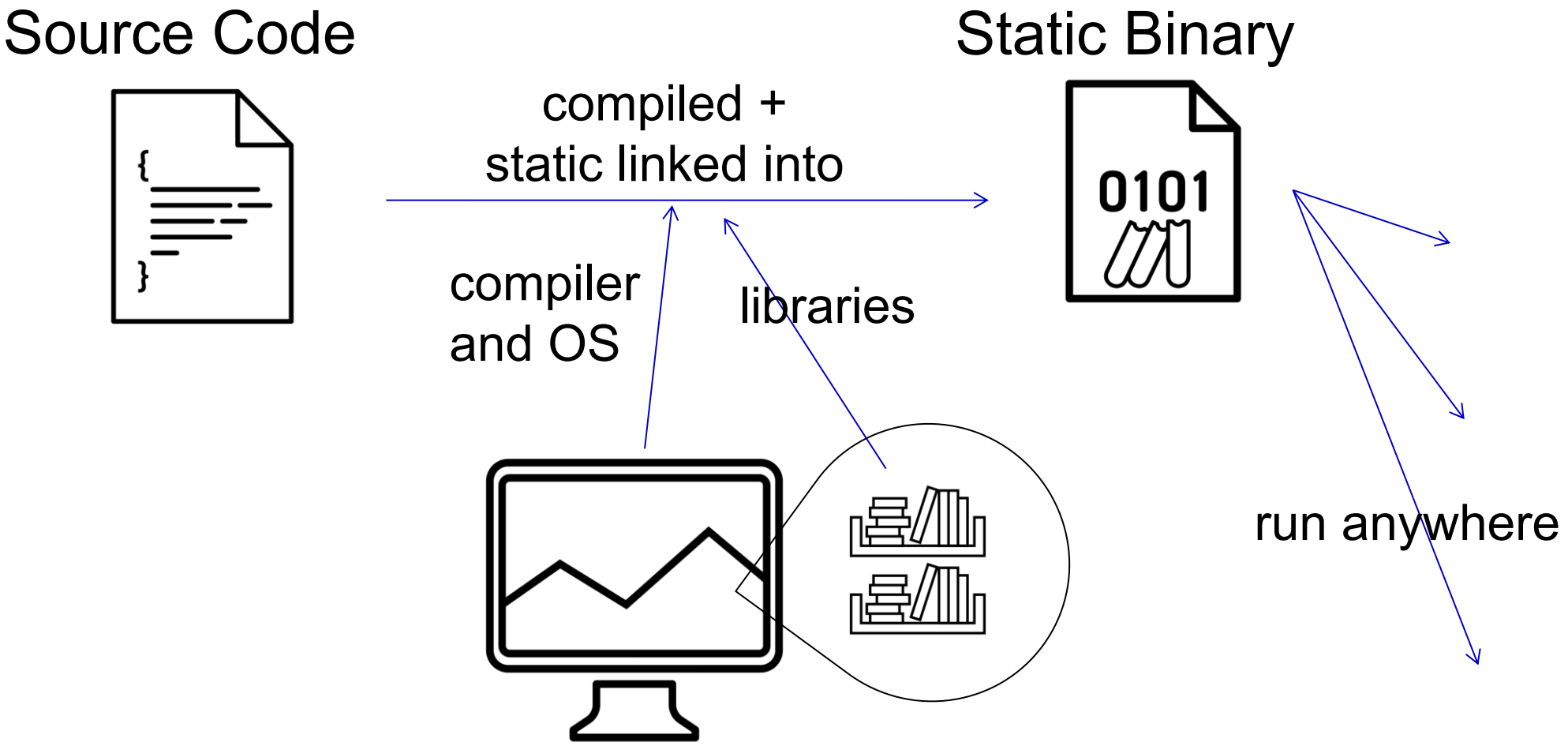


Compiling from Source Code



binary code by Kiran Shastry from the Noun Project
Source Code by Mohamed Mbarki from the Noun Project
Computer by rahmat from the Noun Project
books by Viral faisalovers from the Noun Project

Static Linking



Book by Aleksandr Vector from the Noun Project

Compiling from Source Code

Most common: a three-step build process

1. ./configure # or cmake # configures the build process
2. make # does the compilation and linking
3. make install # moves compiled files to specific location(s)

Installation options (like where to install) are usually set at the configure/cmake step

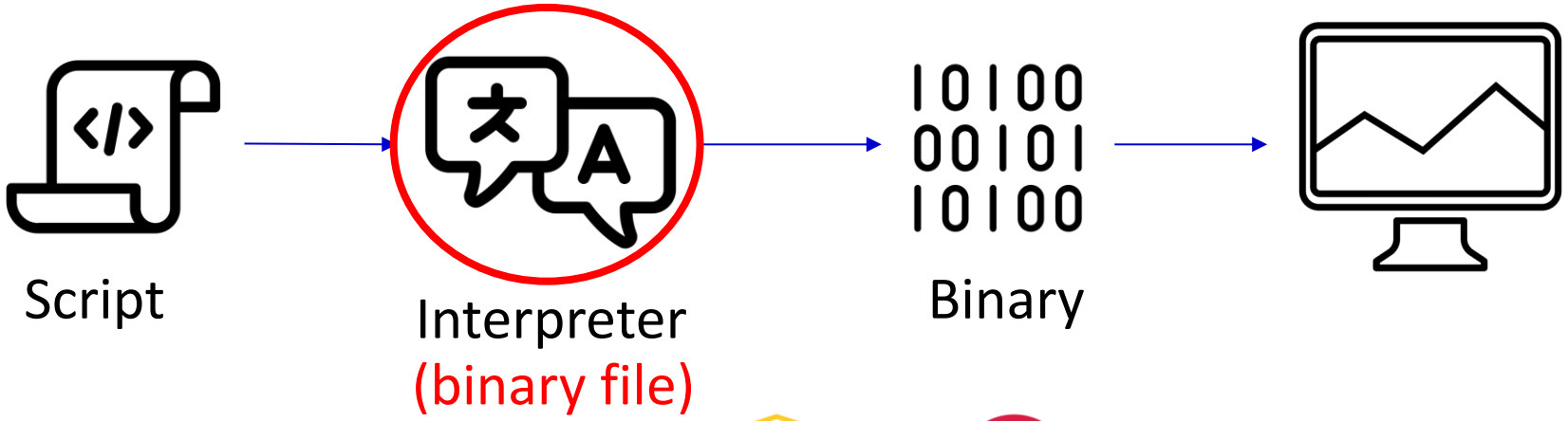
Compiling from Source Code

Advanced: Use a compiler (like gcc) directly

- Can use options to control compilation process
- We assume you know what you are doing!

What Kind of Code?

- Programs written in C, C++ and Fortran are typically compiled.
- For interpreted (scripting) languages like perl, Python, R, or Julia:
  - Don’t compile the scripts, but *do* use a compiled copy of the underlying language interpreter.



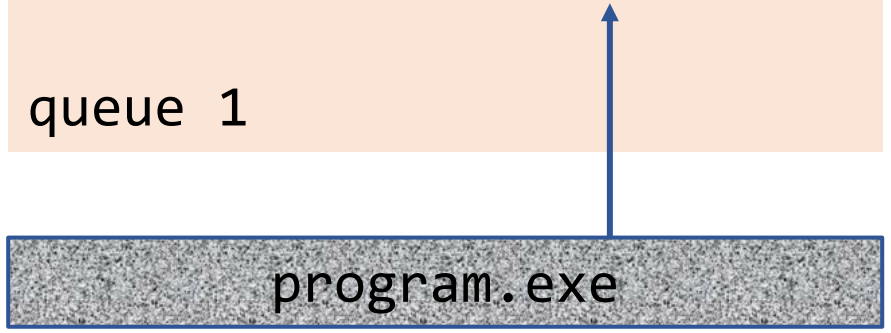
Using Software Files in Jobs

Executable

- Software must be a single compiled binary file or single script.

```
executable = program.exe

queue 1
```



Wrapper Script

- Software can be in any compiled format.

```
executable = run_program.sh
transfer_input_files =
             program.tar.gz
```

```
#!/bin/bash
# run_program.sh

tar –xzf program.tar.gz
program/bin/run in.dat
```

Why Bring Along Software Files

- **No Installation Required** (sometimes)
  - Software releases that are pre-compiled for Linux don’t need any compiling or installation!
- **No Docker/Apptainer Required**
  - Not all computers in the OSPool support containers
- **Lightweight**
  - Rely on the execution point's operating system for most things

Next Steps

Using Software in a HTC System

- Create or find software files:
  - Put them in a container (or find a container that has them already)
  - Download them in a tar.gz or .zip file
  - Make a tar.gz file with code you have built
- Declare necessary files, requirements in your submit file
- If needed, write a wrapper script to set up the environment when the job runs.

Choosing a Software Method

Containers

+ Easy to deploy
+ Consistent environment
+ Best for complex programs

- Building image can be difficult
- Image file can be large

Bring-along-files

+ Standalone binaries Just Work™
+ Lightweight

- Execution points have inconsistent environments
- Not suitable for programs with lots of dependencies

Installing Software

Software installation can be hard.

- Make sure to read the instructions for installing your software, especially the requirements
- The last error message you see may not be the same as the first error message that caused it..
- Test, test, test!

Work Time

1. Go through the introductory exercises
2. Then, choose an approach for *your* software and try to find or make a portable version for OSPool jobs.
3. Ask for help!

Additional guides are at portal.osg-htc.org/documentation

Acknowledgements

This work is supported by NSF under Cooperative Agreement OAC-2030508 as part of the PATh Project. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF.

Appendix: Container and Compiling Tips

Best Practices in Using Containers on OSG

- Don’t use the **latest** tag in images
- Use **version number**/specific names in the images
- Test images with **apptainer shell**
- **Unique** image name eliminates the risk of running a job using previous versions due to stashing.

Where to install software?

- Do not use $HOME, /root or /srv
  - Container will run as some user we do not know yet, so $HOME is not known and will be mounted over
  - /root is not available to unprivileged users
  - /srv is used a job cwd in many cases
- /opt or /usr/local are good choices