

Week 6: collaboration tools

**NRSC 7657 Workshop in Advanced Programming for
Neuroscientists**

course business

- We're going to start group programming time during class this week.
- Questions for later:
 - (1) what is the purpose of your project
 - (2) what are you currently working/stuck on.

Collaboration

- As we'll all learn this afternoon, everyone is writing great code for their projects already (right?)

Collaboration

- As we'll all learn this afternoon, everyone is writing great code for their projects already (right?)
- If you want to share it with your lab-mates, or include it with your paper, or you've made a stand-alone tool that would be useful to the field, there are several ways to do so.

Collaboration

- As we'll all learn this afternoon, everyone is writing great code for their projects already (right?)
- If you want to share it with your lab-mates, or include it with your paper, or you've made a stand-alone tool that would be useful to the field, there are several ways to do so.
 - The best way (cost/benefit) depends on on your code and your audience (lab, other code-savvy users, non-coders, etc)

Collaboration

- As we'll all learn this afternoon, everyone is writing great code for their projects already (right?)
- If you want to share it with your lab-mates, or include it with your paper, or you've made a stand-alone tool that would be useful to the field, there are several ways to do so.
 - The best way (cost/benefit) depends on on your code and your audience (lab, other code-savvy users, non-coders, etc)
 - **No matter what, documentation is a key to effective code sharing**

Collaboration

Running code **not** on your computer

```
3  ✓ class ThingWeMade:
4  ✓     def __init__(self):
5      |         self.thing='neuron'
6      |         self.start()
7
8  ✓     def start(self):
9      |         self.thing2='neuron'
10
11  ✓    def fire(self):
12      |         self.spike = self.thing+' fires an action potential'
```



Collaboration

Running code **not** on your computer

```
3  ∨ class ThingWeMade:
4  ∨      def __init__(self):
5      |         self.thing='neuron'
6      |         self.start()
7
8  ∨      def start(self):
9      |         self.thing2='neuron'
10
11 ∨      def fire(self):
12      |         self.spike = self.thing+' fires an action potential'
```



Collaboration

Running code **not** on your computer

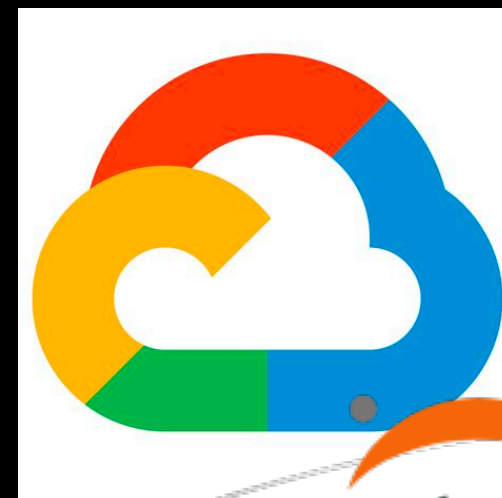
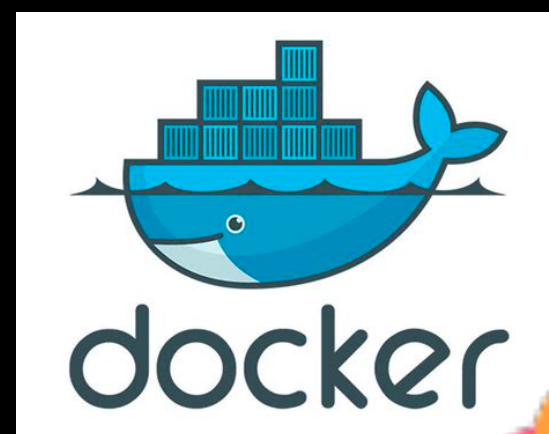
```
3  ✓ class ThingWeMade:
4  ✓      def __init__(self):
5          self.thing='neuron'
6          self.start()
7
8  ✓      def start(self):
9          self.thing2='neuron'
10
11 ✓      def fire(self):
12          self.spike = self.thing+' fires an action potential'
```



Collaboration

Running code **not** on your computer

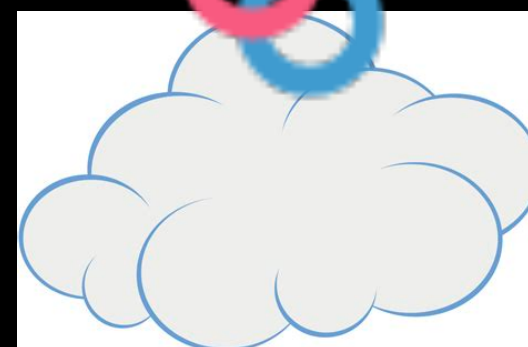
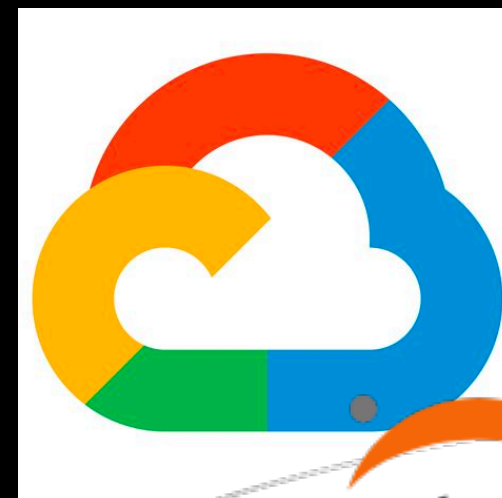
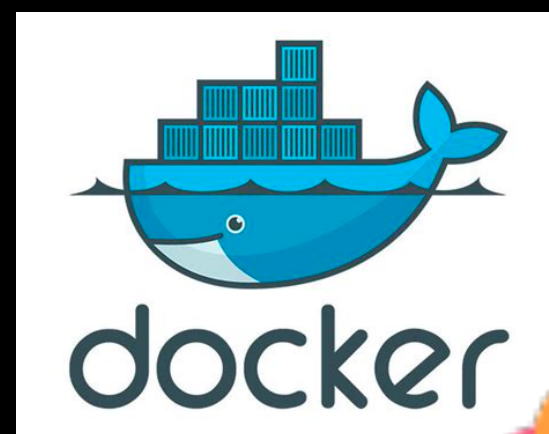
```
3  class ThingWeMade:
4      def __init__(self):
5          self.thing='neuron'
6          self.start()
7
8      def start(self):
9          self.thing2='neuron'
10
11     def fire(self):
12         self.spike = self.thing+' fires an action potential'
```



Collaboration

Running code **not** on your computer

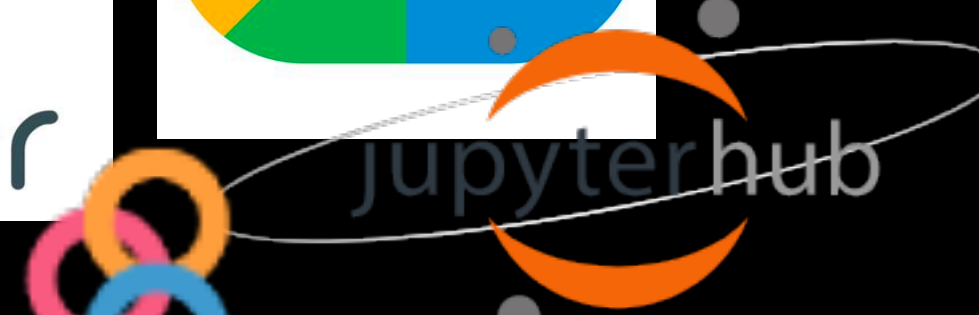
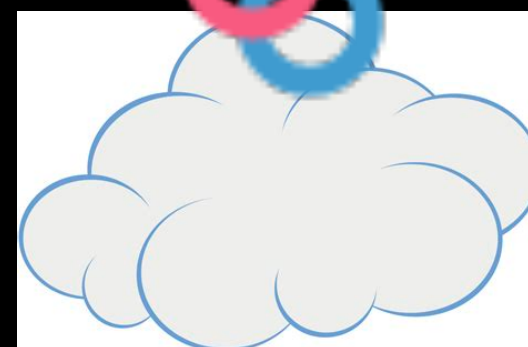
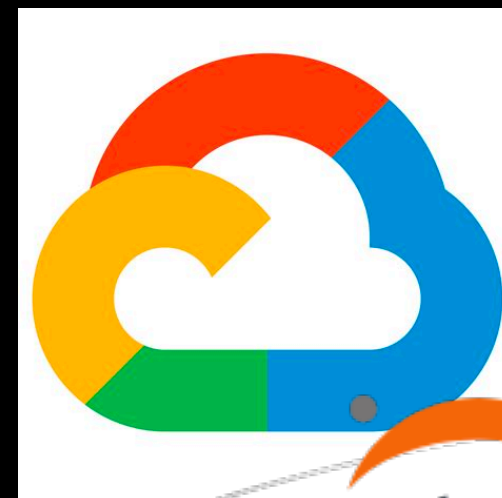
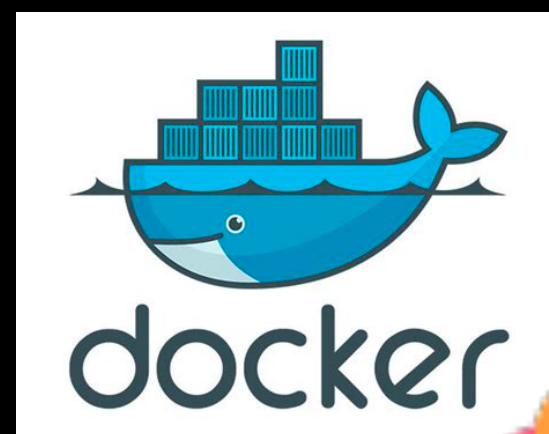
```
3  ✓ class ThingWeMade:
4  ✓      def __init__(self):
5          self.thing='neuron'
6          self.start()
7
8  ✓      def start(self):
9          self.thing2='neuron'
10
11  ✓      def fire(self):
12          self.spike = self.thing+' fires an action potential'
```



Collaboration

Running code **not** on your computer

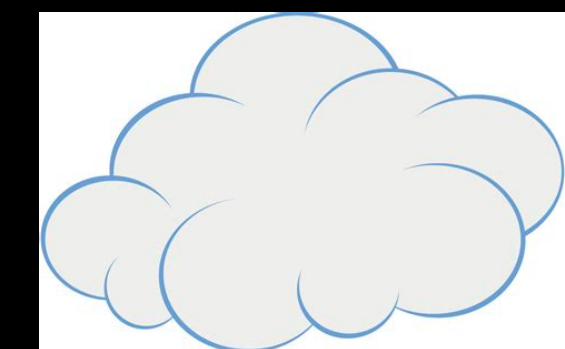
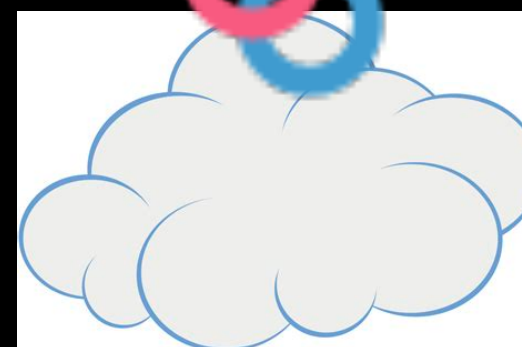
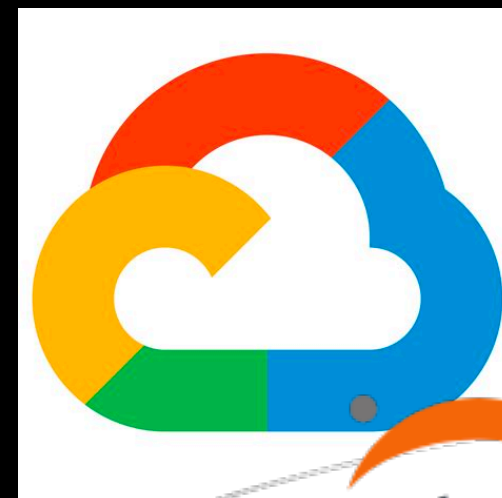
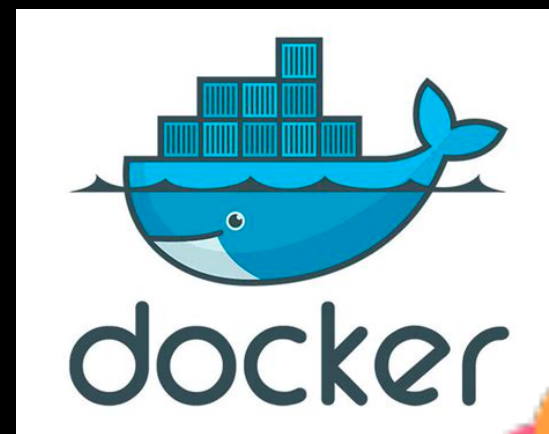
```
3  class ThingWeMade:
4      def __init__(self):
5          self.thing='neuron'
6          self.start()
7
8      def start(self):
9          self.thing2='neuron'
10
11     def fire(self):
12         self.spike = self.thing+' fires an action potential'
```



Collaboration

Running code **not** on your computer

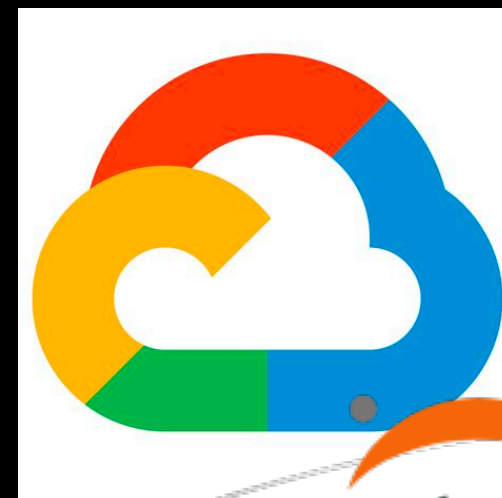
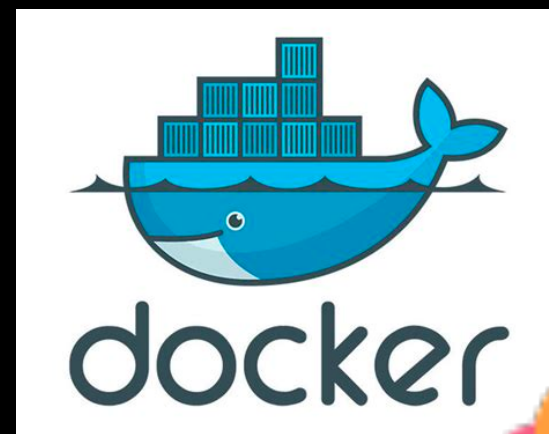
```
3  class ThingWeMade:
4      def __init__(self):
5          self.thing='neuron'
6          self.start()
7
8      def start(self):
9          self.thing2='neuron'
10
11     def fire(self):
12         self.spike = self.thing+' fires an action potential'
```



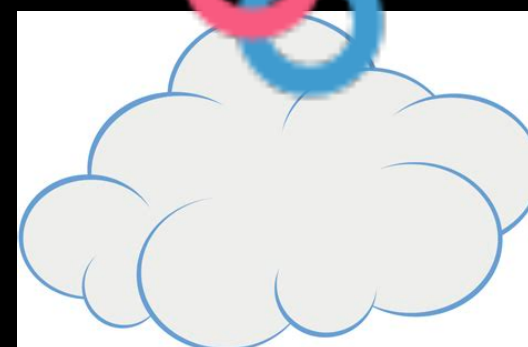
Collaboration

Running code **not** on your computer

```
3  class ThingWeMade:
4      def __init__(self):
5          self.thing='neuron'
6          self.start()
7
8      def start(self):
9          self.thing2='neuron'
10
11     def fire(self):
12         self.spike = self.thing+' fires an action potential'
```



jupyterhub



Collaboration

Running code **not** on your computer



- Easiest way: share your scripts/notebooks. This includes .m files.
 - Github: version control, so people can stay up to date if others update the code.
- Con: need to manage version control (python, MATLAB) and dependency management (python)
- Con: hardware limited to target machine, which may not be able or be efficient at running your code.
- Pro: hardware is owned by the user, so it is managed by them and not limited.

Collaboration

python sharing tools





- jupyter notebooks are rendered by GitHub automatically
- nbviewer.jupyter.org is another, more feature rich way to render notebooks over the internet (not on your localhost: like a regular jupyter server)
 - This is still viewing, not executing the notebook.

Collaboration

nbviewer



 **Jupyter**
nbviewer

JUPYTERFAQ

NRSC7657

Name

CU-NRSC-7657's repositories

Week1

Week2

Week3

Week4

Week5

.DS_Store

LICENSE

NRSC 7657 Workshop in Advanced Programming for Neuroscientists Syllabus.pdf

README.md

Collaboration



- mybinder.org

The screenshot shows a JupyterLab interface with the following elements:

- Header:** "jupyter PerformanceProfiling (autosaved)" with a Python logo, "Visit repo", and "Copy Binder link" buttons.
- Menu Bar:** File, Edit, View, Insert, Cell, Kernel, Widgets, Help.
- Toolbar:** Includes icons for saving, adding, deleting, and running cells, along with "Download", "GitHub", and "Binder" buttons. A "Memory: 124.6 MB / 2 GB" indicator is on the right.
- Code Cell (In [5]):**

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import os,sys,glob
```
- Text Cell:** Contains the word "iteration" in bold, followed by the text "Load some data to iterate over:" and "Serial execution of cells".
- Code Cell (In [6]):**

```
a = 2
```
- Code Cell (In [7]):**

```
b = 2
```
- Code Cell (In [4]):**

```
c = a + b
```
- Footer:** A text label "Serial execution of some lines, through iteration".
- Buttons:** A "dashboard" button is located to the right of the In [7] cell.

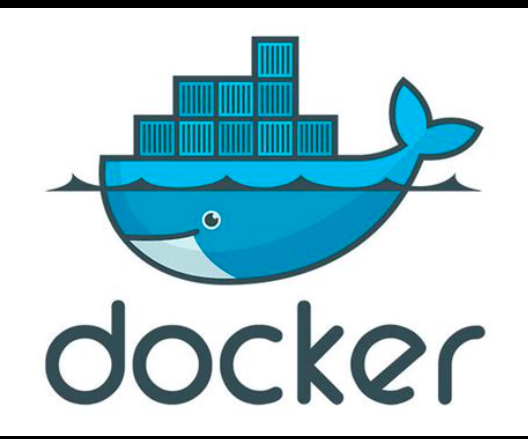
Collaboration

Containerization: docker

- Both nbviewer and mybinder.org use docker containers and virtual machines
 - nbviewer for creating a machine to render the code
 - binder uses docker to create a virtual machine that can run the code
- Because python (and jupyter) use tools that are integrated with the OS (python kernels), cloud servers (running server OSs) can more easily (and automatically) be configured for python

Collaboration

Containerization: docker

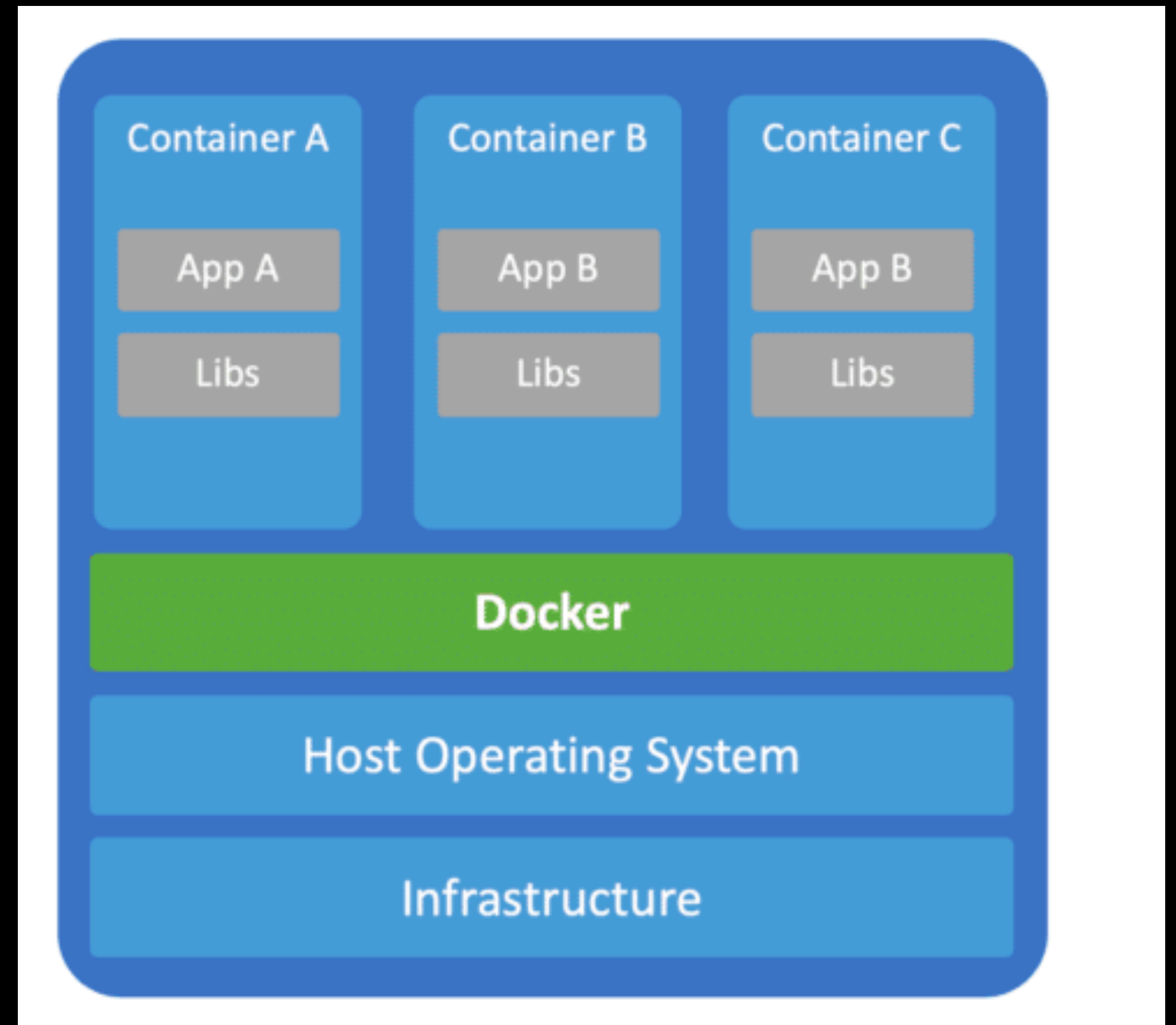


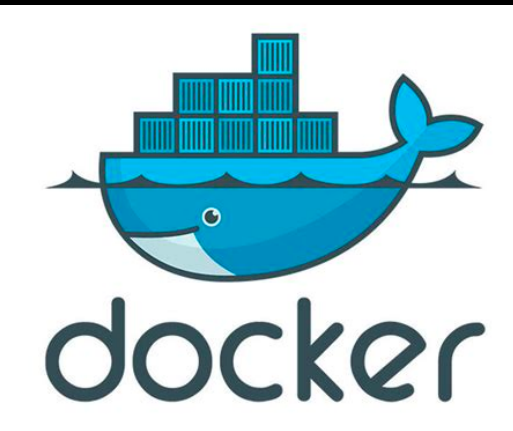
- Both nbviewer and mybinder.org use docker containers and virtual machines
 - nbviewer for creating a machine to render the code
 - binder uses docker to create a virtual machine that can run the code
- Because python (and jupyter) use tools that are integrated with the OS (python kernels), cloud servers (running server OSs) can more easily (and automatically) be configured for python

Collaboration

Containerization: docker

- What is Docker?
 - A platform to deliver self-sufficient packages of code from a host to a client.
 - Platform as a Service (PaaS)
 - A package (called an image) contains **all** of what is needed - binaries, libraries, config, runtime that can run your code as one process.

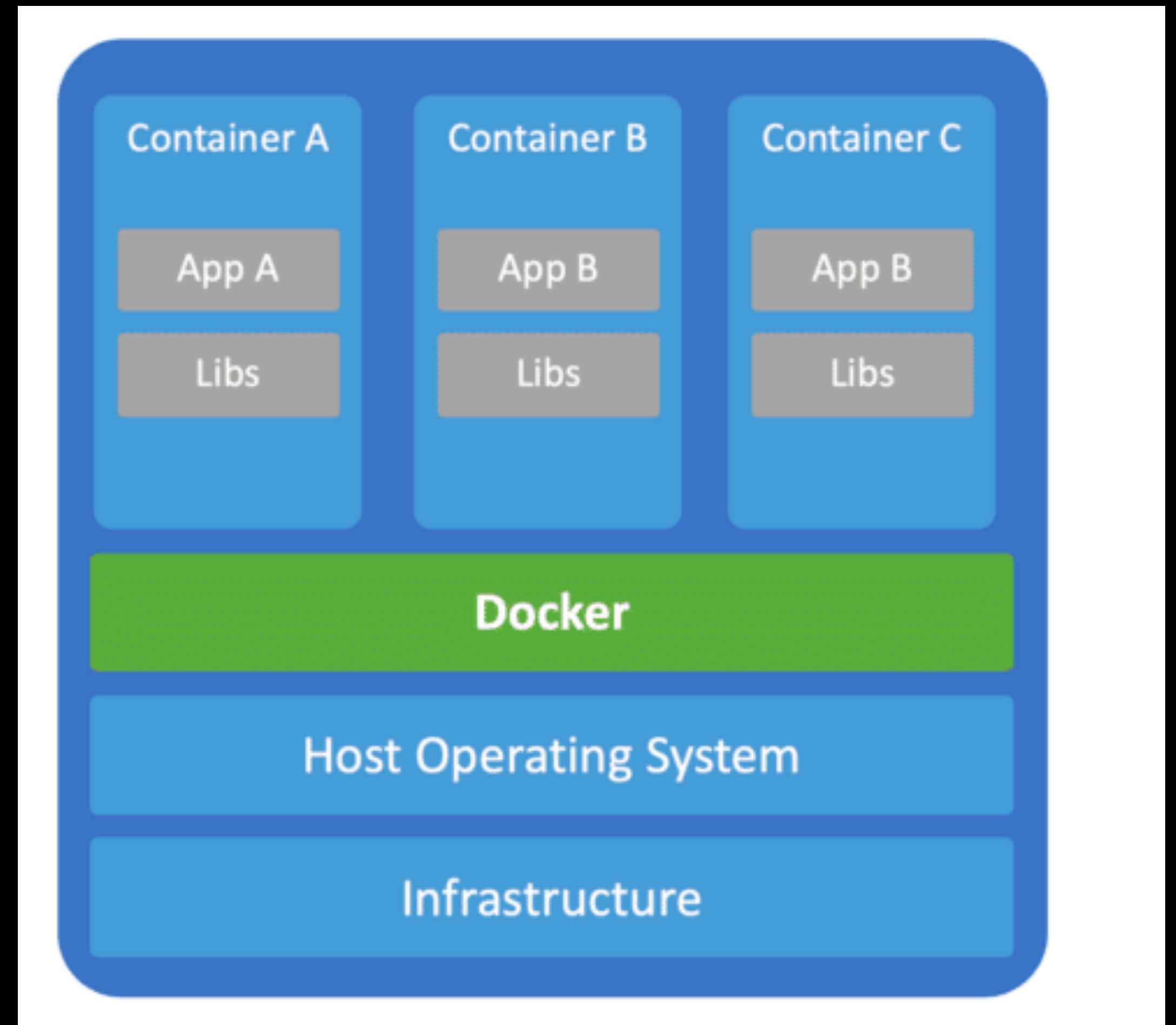




Collaboration

Containerization: docker

- What is Docker?
 - A platform to deliver self-sufficient packages of code from a host to a client.
 - Platform as a Service (PaaS)
 - A package (called an image) contains **all** of what is needed - binaries, libraries, config, runtime that can run your code as one process.



Collaboration

Containerization: docker + binder



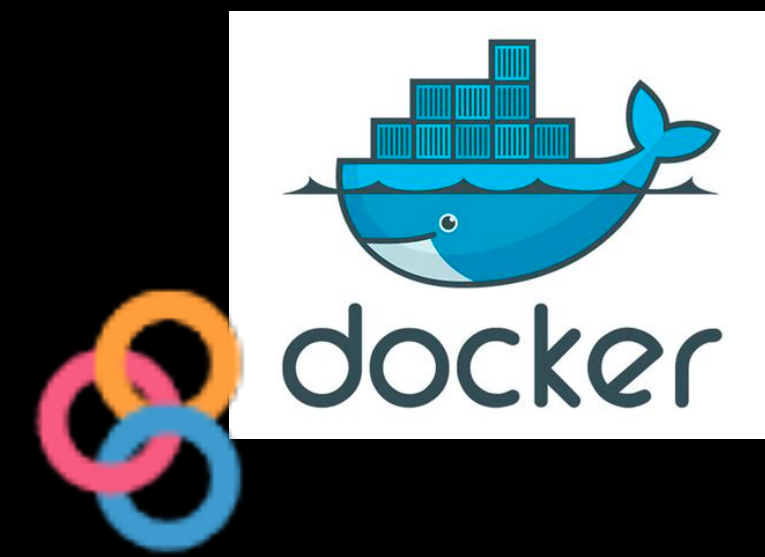
<https://github.com/alan-turing-institute/the-turing-way/blob/master/workshops/boost-research-reproducibility-binder/workshop-presentations/zero-to-binder-python.md#whats-happening-in-the-background---part-1>

While you wait, BinderHub (the backend of Binder) is:

- Fetching your repo from GitHub
- Analysing the contents
- Creating a Docker file based on your repo
- Launching that Docker image in the Cloud
- Connecting you to it via your browser

Collaboration

Containerization: docker + binder



<https://github.com/alan-turing-institute/the-turing-way/blob/master/workshops/boost-research-reproducibility-binder/workshop-presentations/zero-to-binder-python.md#whats-happening-in-the-background---part-1>

While you wait, BinderHub (the backend of Binder) is:

- Fetching your repo from GitHub
- Analysing the contents
- Creating a Docker file based on your repo
- Launching that Docker image in the Cloud
- Connecting you to it via your browser

Collaboration



- mybinder.org

The screenshot shows a JupyterLab interface with the following elements:

- Header:** "jupyter PerformanceProfiling (autosaved)" with a Python logo, "Visit repo", and "Copy Binder link" buttons.
- Menu Bar:** File, Edit, View, Insert, Cell, Kernel, Widgets, Help.
- Toolbar:** Includes icons for saving, adding, deleting, and running cells, along with "Download", "GitHub", and "Binder" buttons. A "Memory: 124.6 MB / 2 GB" indicator is on the right.
- Code Cell (In [5]):**

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import os,sys,glob
```
- Text Cell:** Contains the word "iteration" in bold, followed by the text "Load some data to iterate over:" and "Serial execution of cells".
- Code Cell (In [6]):**

```
a = 2
```
- Code Cell (In [7]):**

```
b = 2
```
- Code Cell (In [4]):**

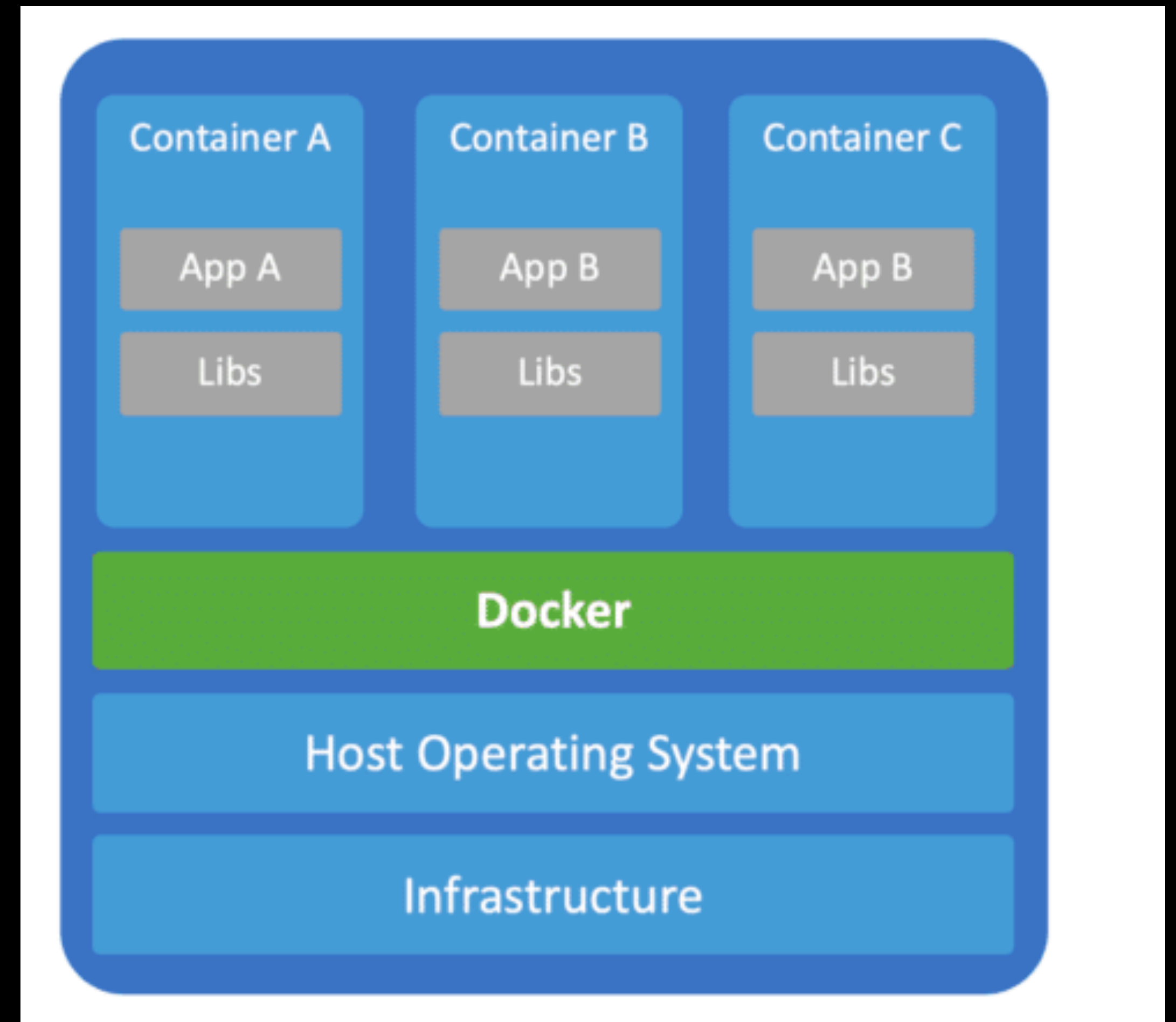
```
c = a + b
```
- Footer:** A text label "Serial execution of some lines, through iteration".
- Buttons:** A "dashboard" button is located to the right of the In [7] code cell.

Collaboration

Containerization: docker + binder

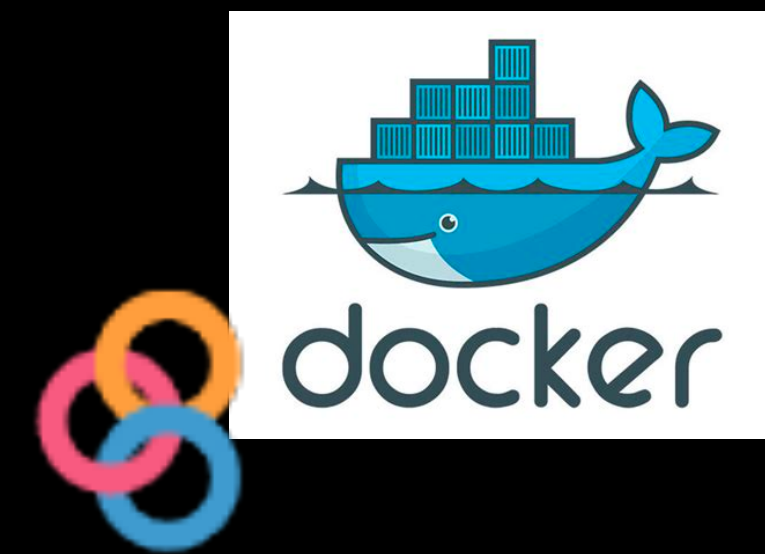


- What is the hardware

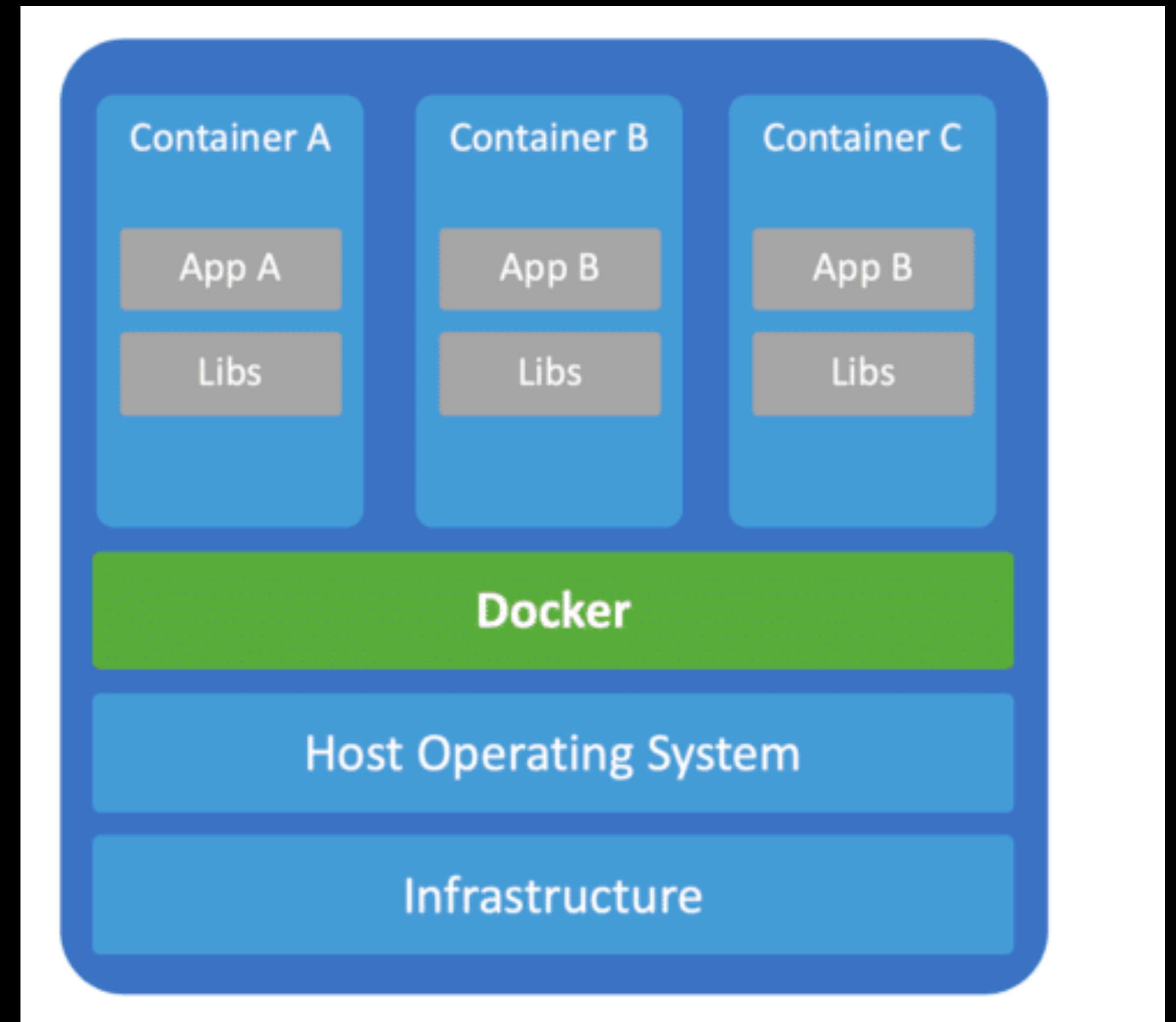


Collaboration

Containerization: docker + binder



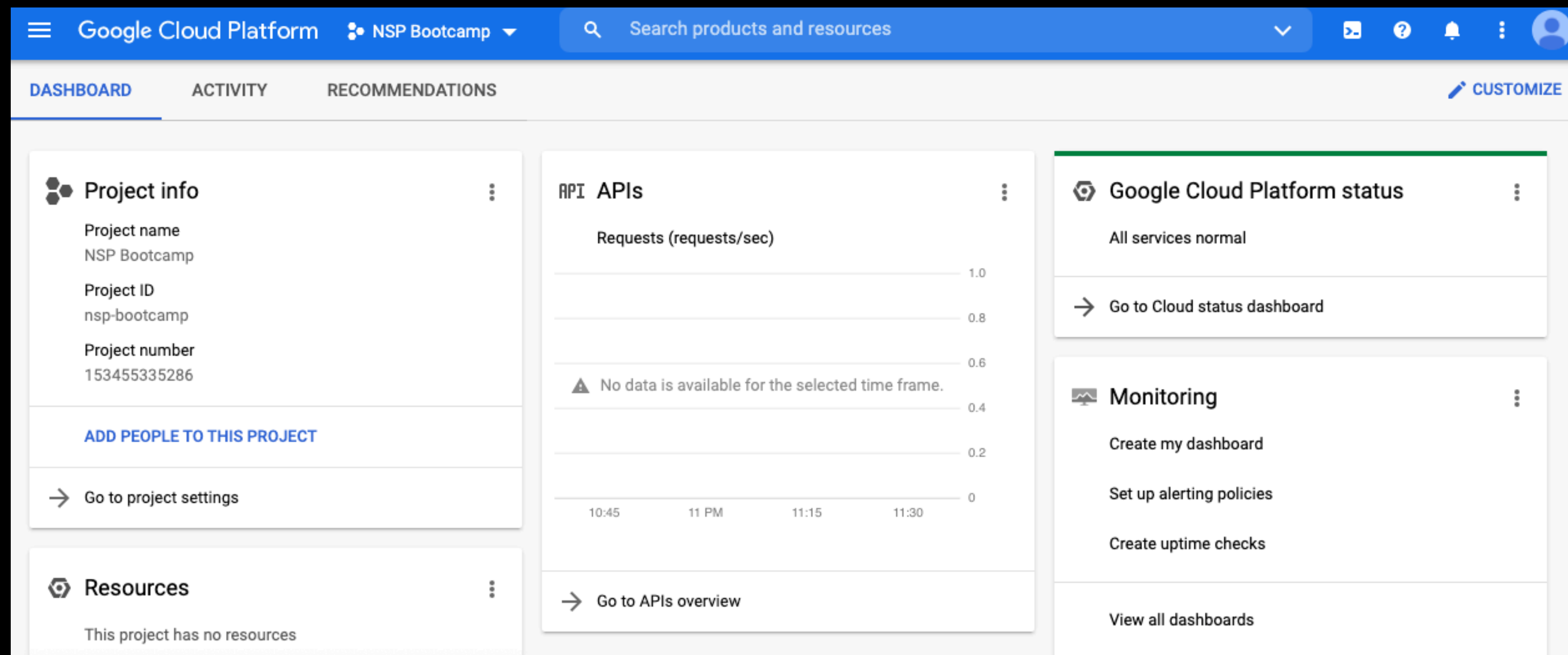
- What is the hardware



Collaboration

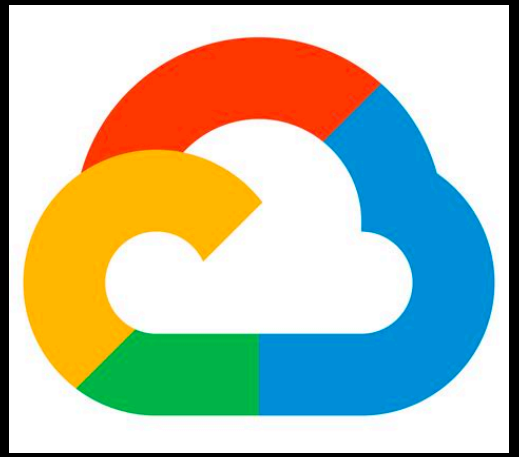
GCP

- Google Cloud Platform
- Access to Google's compute resources



Collaboration

GCP



- Google Cloud Platform
- Access to Google's compute resources

The screenshot shows the Google Cloud Platform (GCP) dashboard for a project named 'NSP Bootcamp'. The interface includes a top navigation bar with the GCP logo, project name, a search bar, and user profile. Below this is a secondary navigation bar with tabs for 'DASHBOARD', 'ACTIVITY', and 'RECOMMENDATIONS', along with a 'CUSTOMIZE' link. The main content area is divided into several panels:

- Project info:** Displays project details such as Project name (NSP Bootcamp), Project ID (nsp-bootcamp), and Project number (153455335286). It includes a link to 'ADD PEOPLE TO THIS PROJECT' and a button to 'Go to project settings'.
- Resources:** A section indicating that 'This project has no resources'.
- API APIs:** A panel showing a line chart for 'Requests (requests/sec)'. The chart area displays a message: 'No data is available for the selected time frame.' Below the chart is a button to 'Go to APIs overview'.
- Google Cloud Platform status:** A panel showing 'All services normal' and a button to 'Go to Cloud status dashboard'.
- Monitoring:** A panel with options to 'Create my dashboard', 'Set up alerting policies', 'Create uptime checks', and 'View all dashboards'.

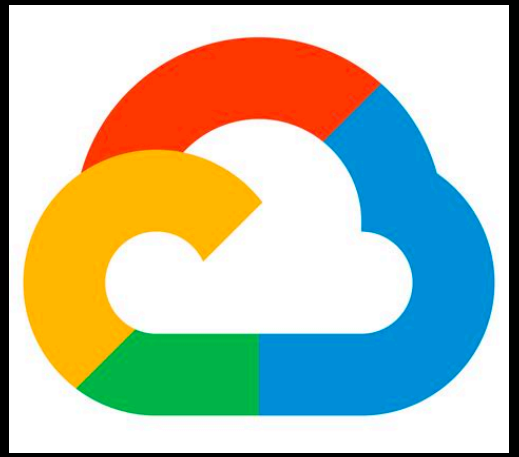
Collaboration

Other platforms

Google Cloud Platform	Amazon Web Services ^[12]	Microsoft Azure ^[13]	Oracle Cloud ^[14]
Google Compute Engine	Amazon EC2	Azure Virtual Machines	Oracle Cloud Infra OCI
Google App Engine	AWS Elastic Beanstalk	Azure App Services	Oracle Application Container
Google Kubernetes Engine	Amazon Elastic Kubernetes Service	Azure Kubernetes Service	Oracle Kubernetes Service
Google Cloud Bigtable	Amazon DynamoDB	Azure Cosmos DB	Oracle NoSQL Database
Google BigQuery	Amazon Redshift	Azure Synapse Analytics	Oracle Autonomous Data Warehouse
Google Cloud Functions	AWS Lambda	Azure Functions	Oracle Cloud Fn
Google Cloud Datastore	Amazon DynamoDB	Azure Cosmos DB	Oracle NoSQL Database
Google Cloud Storage	Amazon S3	Azure Blob Storage	Oracle Cloud Storage OCI

Collaboration

Other platforms




Google Cloud Platform	Amazon Web Services ^[12]	Microsoft Azure ^[13]	Oracle Cloud ^[14]
Google Compute Engine	Amazon EC2	Azure Virtual Machines	Oracle Cloud Infra OCI
Google App Engine	AWS Elastic Beanstalk	Azure App Services	Oracle Application Container
Google Kubernetes Engine	Amazon Elastic Kubernetes Service	Azure Kubernetes Service	Oracle Kubernetes Service
Google Cloud Bigtable	Amazon DynamoDB	Azure Cosmos DB	Oracle NoSQL Database
Google BigQuery	Amazon Redshift	Azure Synapse Analytics	Oracle Autonomous Data Warehouse
Google Cloud Functions	AWS Lambda	Azure Functions	Oracle Cloud Fn
Google Cloud Datastore	Amazon DynamoDB	Azure Cosmos DB	Oracle NoSQL Database
Google Cloud Storage	Amazon S3	Azure Blob Storage	Oracle Cloud Storage OCI

Collaboration

Google colab



 Plotting.ipynb

File Edit View Insert Runtime Tools Help [Cannot save changes](#)

Table of contents

Plotting

Figures and Axes

Axis

Artist

The plot object

a note on backends

matplotlib 3d plots

Seaborn

palettes

ipywidgets and interactive plotting

Bokeh

Section

+ Code + Text

Copy to Drive

RAM Disk

Editing

Plotting

some neuroscience data

25 June 2021

NRSC 7657

Daniel J Denman and John Thompson

University of Colorado Anschutz

We'll start with the workhorse of plotting, matplotlib. It has flexibility and deep functionality, and many other packages rely on it in some way. If they don't they are going to end up doing the drawing, managing what your graphics card does directly, often then relying on `PyQt`. All of these have to eventually converge there, as all things that have a graphical component do.[This is mostly trivia, but may come in handy]



Collaboration

Google colab

- there are time limits (at)
- Resource (GPU, memory, disk space on the VM)
- Time - 12 - 24 hours and then it will kick you off. Not for hardcore number crunching.



Collaboration

Deepnote

- Similar idea to Colab, but with project organization
- Most importantly: like a google doc, everyone viewing a Deepnote notebook sees the code and the results of execution live in their own browser.
- Not sure what the hardware is! (GCP, AWS, Azure??)

https://deepnote.com/project/NRSC7610-Python-Activities-JayFWIp7SGmIW-JFkMKIbg/%2FNRSC7610_ReceptiveField.ipynb

Collaboration

Why all of this cloud stuff?

- Your computer works...and many have a computer or 17 in the lab...

Collaboration

Why all of this cloud stuff?

1. Working with other people, especially not locally (colab, deepnote)

Collaboration

Why all of this cloud stuff?

1. Working with other people, especially not locally (colab, deepnote)
2. Publishing code along with manuscripts. Reproducibility!

Collaboration

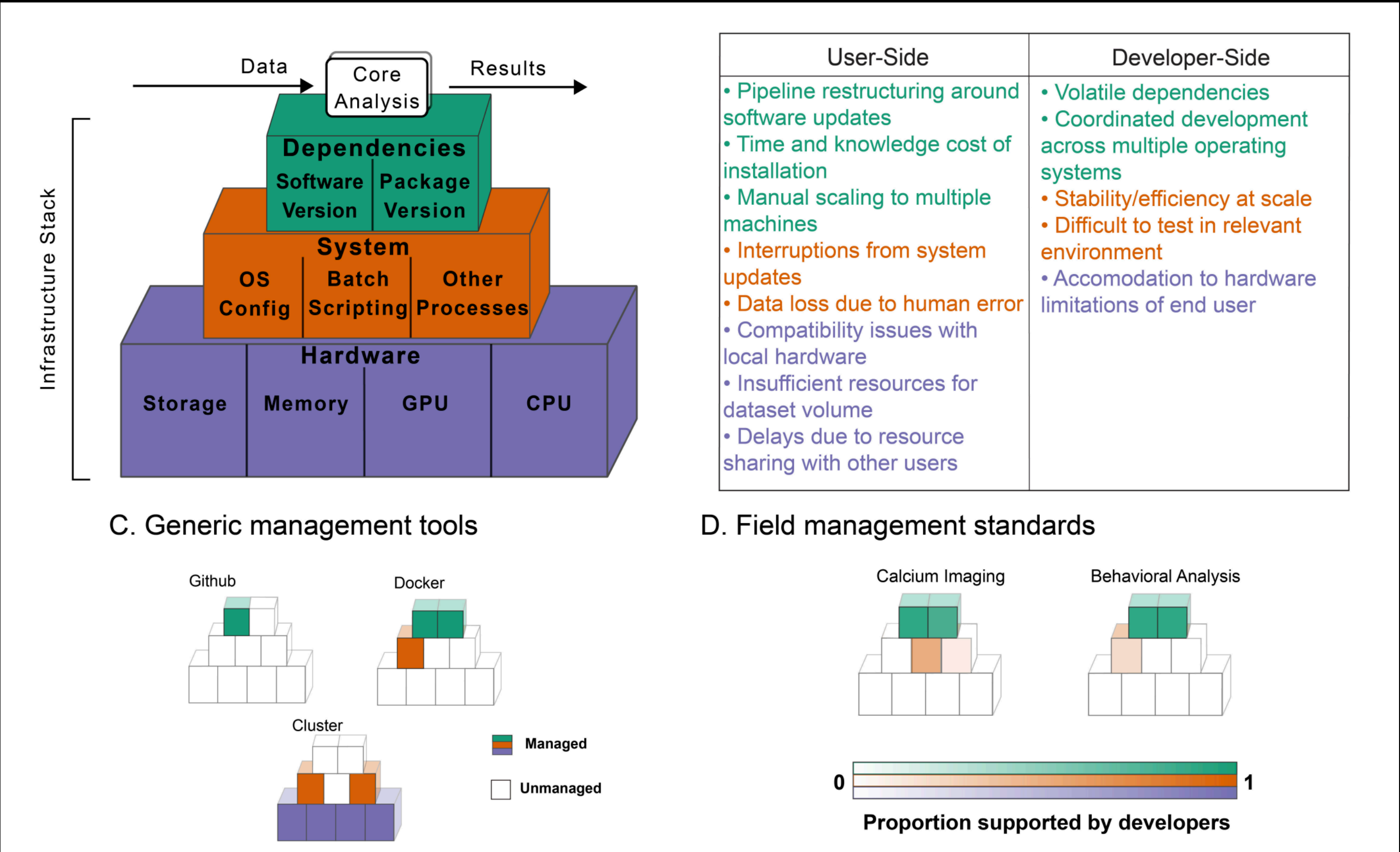
Why all of this cloud stuff?

1. Working with other people, especially not locally (colab, deepnote)
2. Publishing code along with manuscripts. Reproducibility!
3. SaaS - a future direction for some analysis, which runs on the cloud?

Collaboration

Software as a Service - SaaS

- <http://neurocaas.org/>



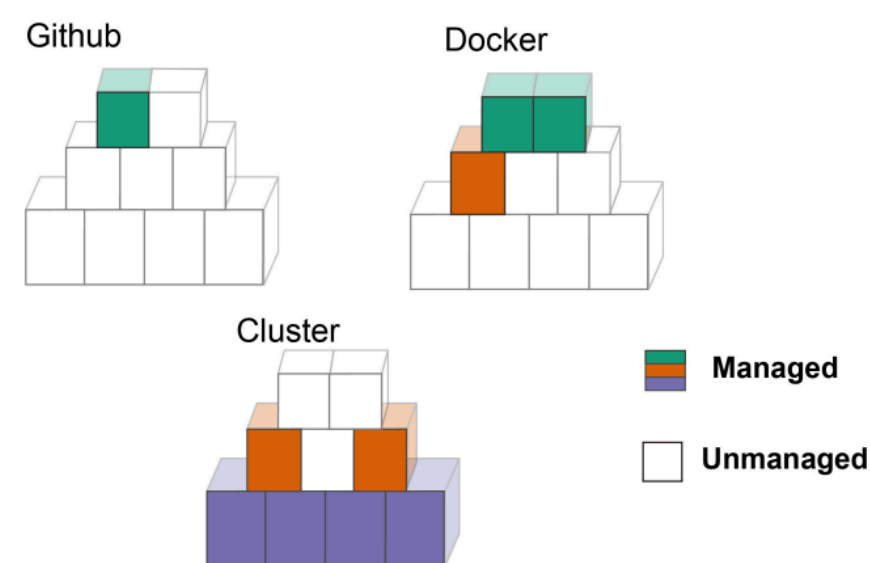
Collaboration

Software as a Service - SaaS

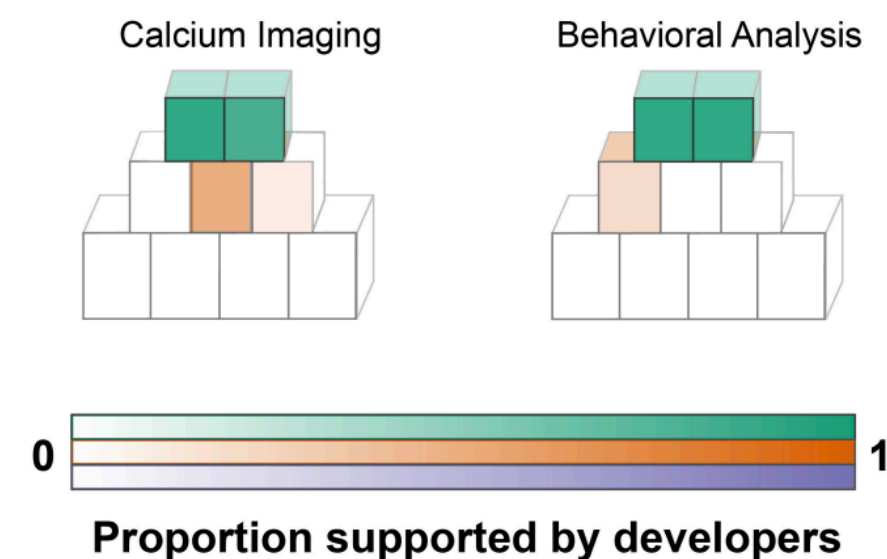
- <http://neurocaas.org/>

Infrastructure as Graduate Student (laGS)

C. Generic management tools

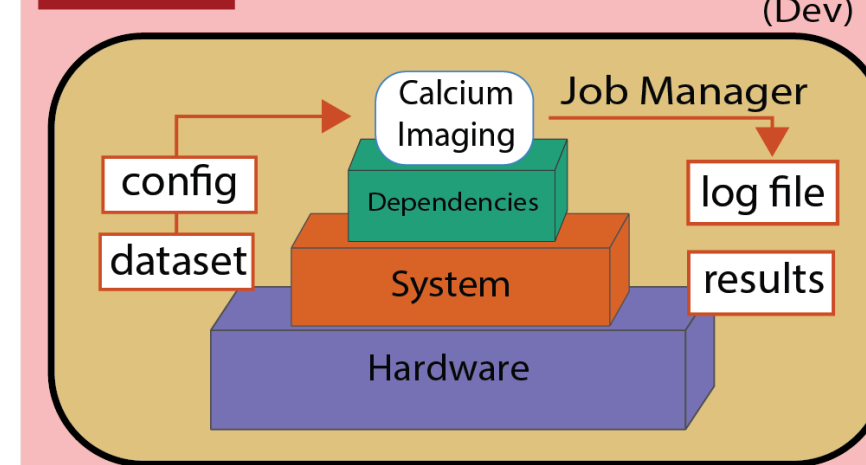


D. Field management standards

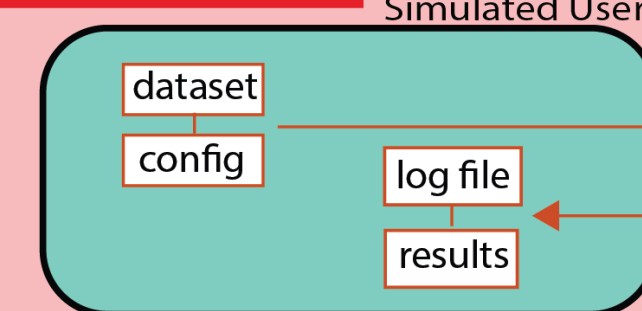


Infrastructure as Code (laC)

Choose NeuroCAAS Resources



Test with Simulated User



Publish to NeuroCAAS

NeuroCAAS Workflow

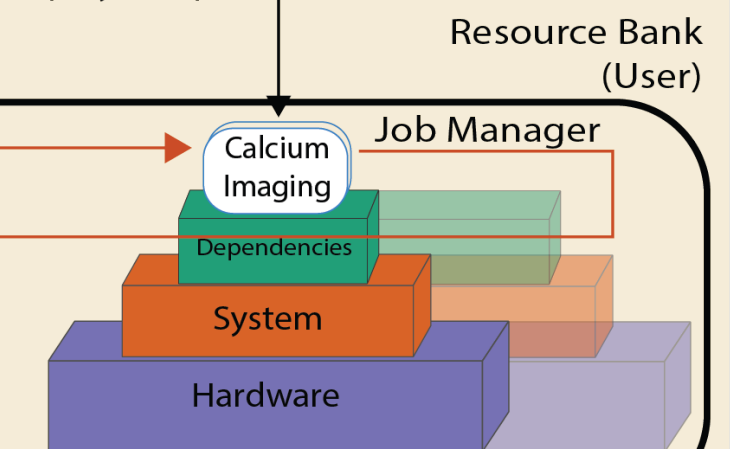
Write Blueprint

```
{NCAP Algorithm 1:
  {Resource: Software
    Name: NCAP Algorithm 1,
    Version: 1.0.0
    Workflow_Script: run.sh
    Dependencies: [Package1, Package2]}
  {Resource: Inputs
    Datatype: [.mp4,.mpg],
    Params: [param1, param2]}
```

```
{Resource: Operating System
  Type: Linux,
  Version: 16.04}
{Resource: Storage
  Capacity: 200 GB,
  Type: HDD}
{Resource: GPU
  ID: NVIDIA P100}
{Resource: CPU
  Cores: 16,
  Type: CPU_Type}
```

```
{Resource: Job Manager
  Messaging: Verbose,
  Workflow: Parallel (Simple),
  User Test: Passed}}
```

Deploy Blueprint



Collaboration

<http://neurocaas.org/>

- Make it easier to use these cloud stacks for neuroscientists
 - (So you don't have to do as much of, say, what we'll do next week)
- (1) data + (2) configuration file.

https://github.com/cunningham-lab/neurocaas/blob/master/experiments/DLC/batch_10.json

Collaboration

<http://neurocaas.org/>

- <https://www.biorxiv.org/content/10.1101/2020.06.11.146746v2>




bioRxiv

THE PREPRINT SERVER FOR BIOLOGY

bioRxiv posts many COVID19-related papers. A reminder: they have not been formally peer-reviewed and should not guide health-related behavior or be reported in the press as conclusive.

New Results

Neuroscience Cloud Analysis As a Service

 Taiga Abe, Ian Kinsella, Shreya Saxena, E. Kelly Buchanan, Joao Couto, John Briggs, Sian Lee Kitt, Ryan Glassman, John Zhou, Liam Paninski, John P. Cunningham

doi: <https://doi.org/10.1101/2020.06.11.146746>