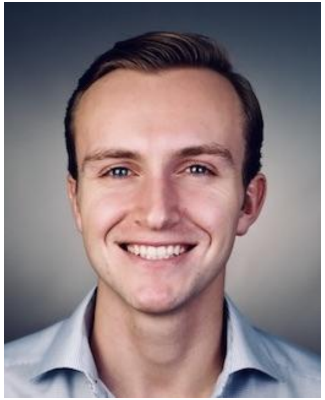
Notebook to Cloud

Yue Sun

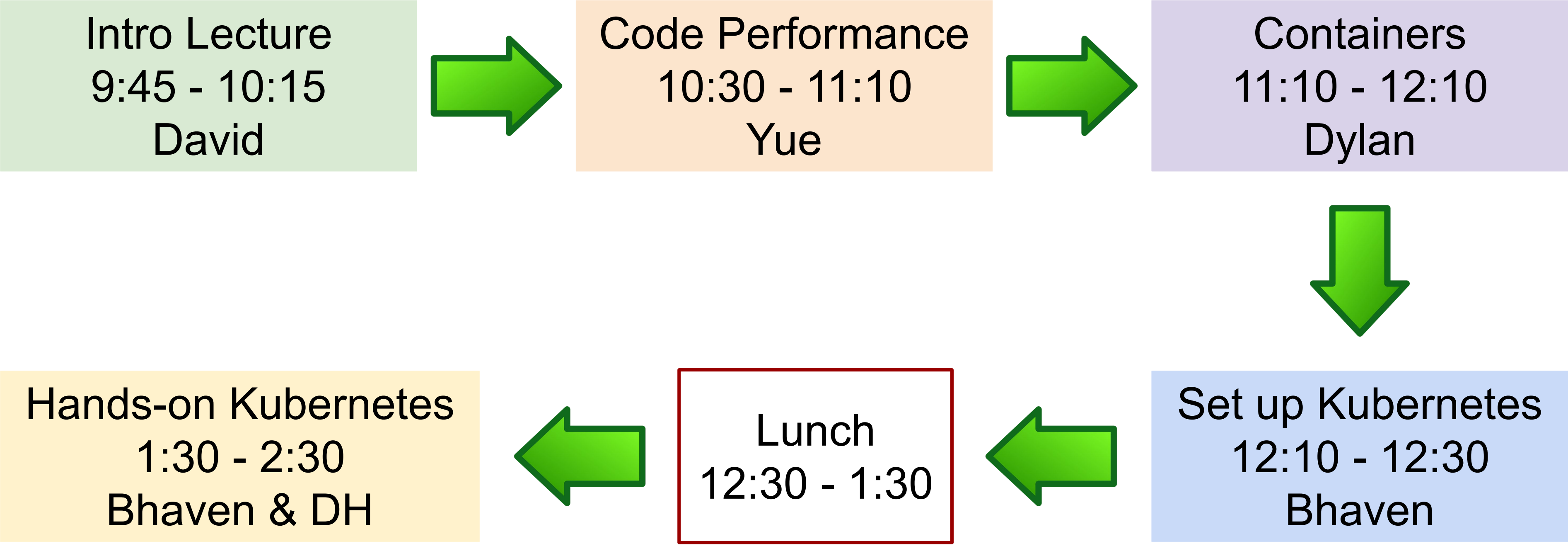


Dylan Randle

Bhaven Patel

DH Lee

Workshop Overview for Session 1



Intro Lecture
9:45 - 10:15
David

Code Performance
10:30 - 11:10
Yue

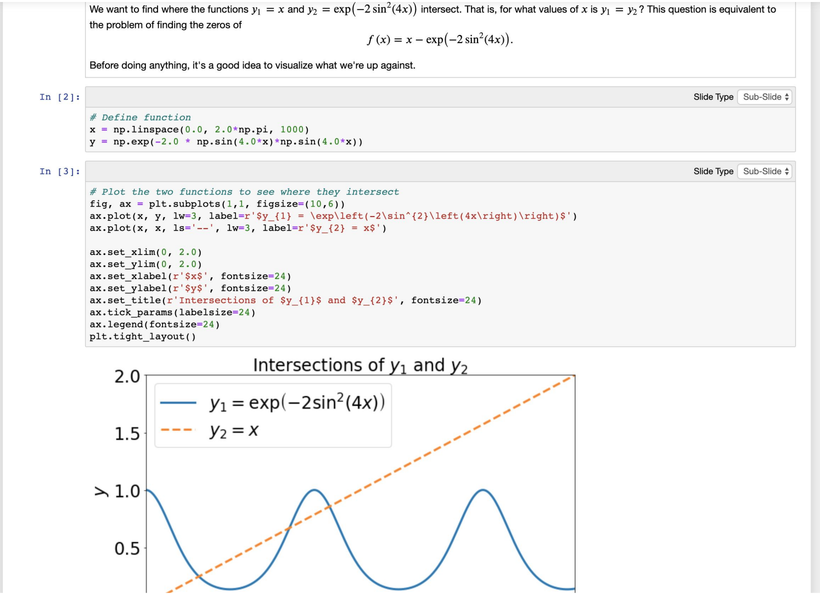
Containers
11:10 - 12:10
Dylan

Set up Kubernetes
12:10 - 12:30
Bhaven

Lunch
12:30 - 1:30

Hands-on Kubernetes
1:30 - 2:30
Bhaven & DH

Why Notebooks?



We want to find where the functions $y_1 = x$ and $y_2 = \exp\left(-2\sin^2(4x)\right)$ intersect. That is, for what values of x is $y_1 = y_2$? This question is equivalent to the problem of finding the zeros of

$$f(x) = x - \exp\left(-2\sin^2(4x)\right).$$

Before doing anything, it's a good idea to visualize what we're up against.

In [2]: Slide Type Sub-Slide

```
# Define function
x = np.linspace(0.0, 2.0*np.pi, 1000)
y = np.exp(-2.0 * np.sin(4.0*x)*np.sin(4.0*x))
```

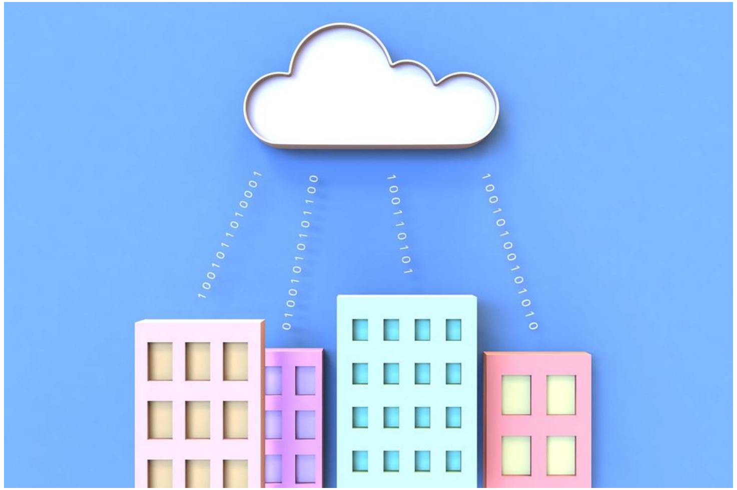
In [3]: Slide Type Sub-Slide

```
# Plot the two functions to see where they intersect
fig, ax = plt.subplots(1,1, figsize=(10,6))
ax.plot(x, y, lw=3, label=r'$y_{1} = \exp\left(-2\sin^{2}\left(4x\right)\right)$')
ax.plot(x, x, ls='--', lw=3, label=r'$y_{2} = x$')

ax.set_xlim(0, 2.0)
ax.set_ylim(0, 2.0)
ax.set_xlabel(r'$x$', fontsize=24)
ax.set_ylabel(r'$y$', fontsize=24)
ax.set_title(r'Intersections of $y_{1}$ and $y_{2}$', fontsize=24)
ax.tick_params(labelsize=24)
ax.legend(fontsize=24)
plt.tight_layout()
```

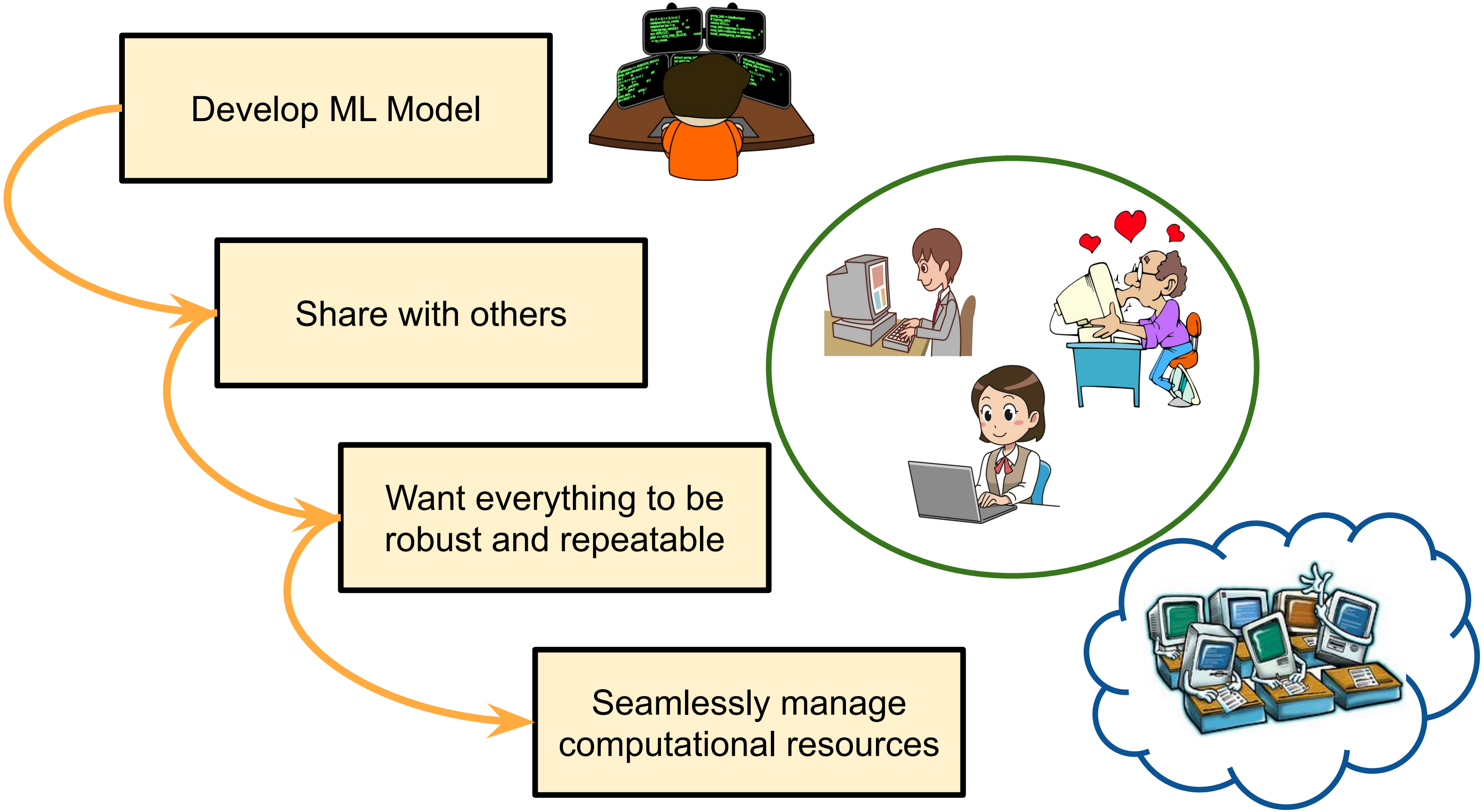
- Easy to use
- Great for prototyping
- Excellent for documentation and examples

Why Cloud?



- *Many* resources beyond what you have locally
  - CPUs
  - GPUs
  - Storage
  - ...
- Resources maintained by experts
- Easier to reproduce results
  - Collaborators have replicable platforms
- Easier to host applications
  - Virtual machines
  - Containers
  - ...

A Dream Scenario



Develop ML Model

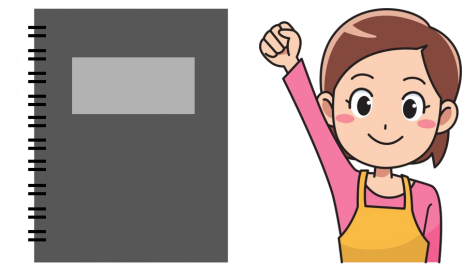
Share with others

Want everything to be robust and repeatable

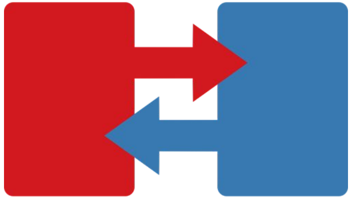
Seamlessly manage computational resources

A Common Scenario

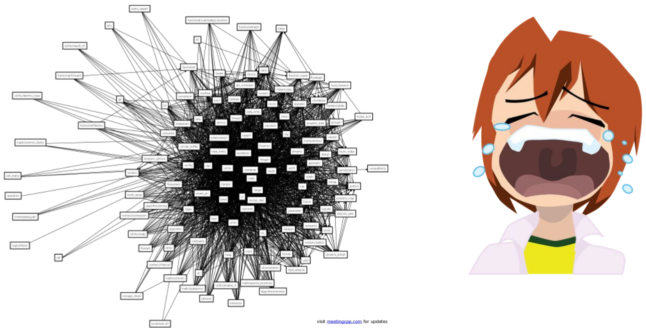
Develop code base in Jupyter notebooks



Pass around to colleagues and friends



Tweak depending on versions and platforms people are using

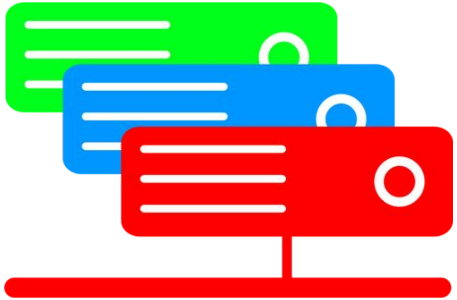


A Modern Approach

Prototype, document, and examples in Jupyter notebooks

1

```
Apple //e
]10 LOAD COFFEE
]20 DRINK COFFEE
]30 IF MUG > EMPTY
THEN GOTO 20
]40 GOTO 10
]RUN
```



Large code-bases written in scripts

Managed with version control

2

Ship package in lightweight containers

3

Deploy to the cloud

4

Automatically manage containers in cloud

5

Towards Collaboration: What Can Go Wrong

- Congrats! You’ve developed a nice application.
- But you can’t work in isolation:
  - Need help and input from colleagues
  - Want to have other people use your awesome app
- What can go wrong?
- Turns out, a lot!
  - Different software versions (e.g. Python 2.7 vs. 3.7)
  - Different operating systems (Windows vs. Linux vs. MacOS)
  - Different packages required than are available locally

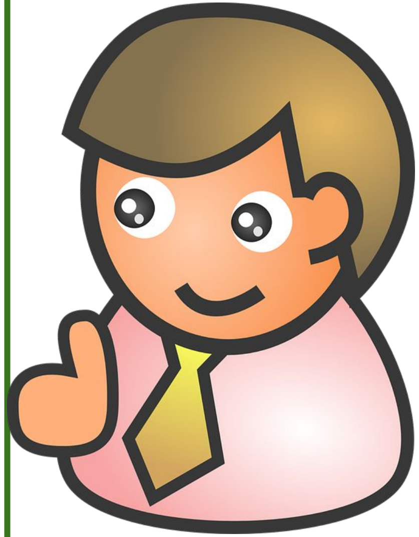
Achieving Collaborative Isolation

- We develop on specific platforms using specific versions of software and dependencies
- How can we make sure everyone works with the same environment?
    - Virtual environment --- Still depends on operating system
    - Virtual machines --- Full isolation
    - Containers --- Only virtualize OS (not the hardware)

Virtual Environments

- ✓ Requirement A
- ✓ Requirement B
- ✓ Requirement C

- List requirements in special file
- Automatically install dependencies



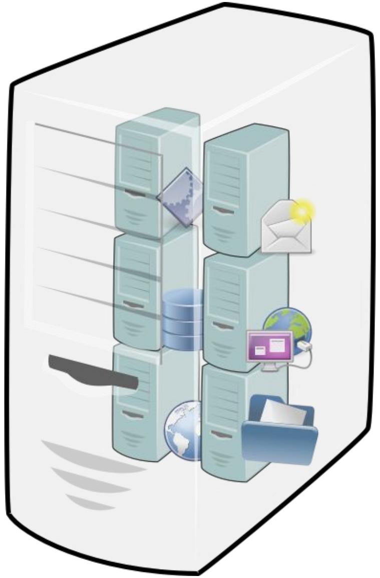
- Create virtual environment
- Install package
- Dependencies installed too

- ✓ Requirement A
- ✓ Requirement B
- ✓ Requirement C

Comments on Virtual Environments

- Many options for virtual environments
  - virtualenv
  - conda
- Very useful for working with fellow developers
- Convenient, lightweight method to achieve code portability
- Not as helpful when deploying a package to a larger audience
- They do not provide complete isolation
  - Still depends on your operating system
  - Uses global packages and dependencies of the operating system

The Other End of the Spectrum: Virtual Machines



- A virtual machine (VM) is a file that acts like a separate computer system.
- You can install a completely different operating system on this virtual machine.
    - e.g. You run MacOS; create a VM and install Windows on it
- VMs have their own virtual hardware
    - CPUs, memory, hard drives, etc.
- Software inside the VM can't affect the actual computer
    - Sandboxing
    - Safe place to test virus-infected software

Comments on Virtual Machines

- They can help lower costs and can be more efficient
  - e.g. no need to spend money on physical hardware and cooling systems
- Best Virtual Machines of 2019
  - VMware
  - VirtualBox
- There is overhead associated with VMs
  - They may not be as fast as the host system
  - They may not have the same graphics capabilities
- They can take some time to start up (order of minutes)

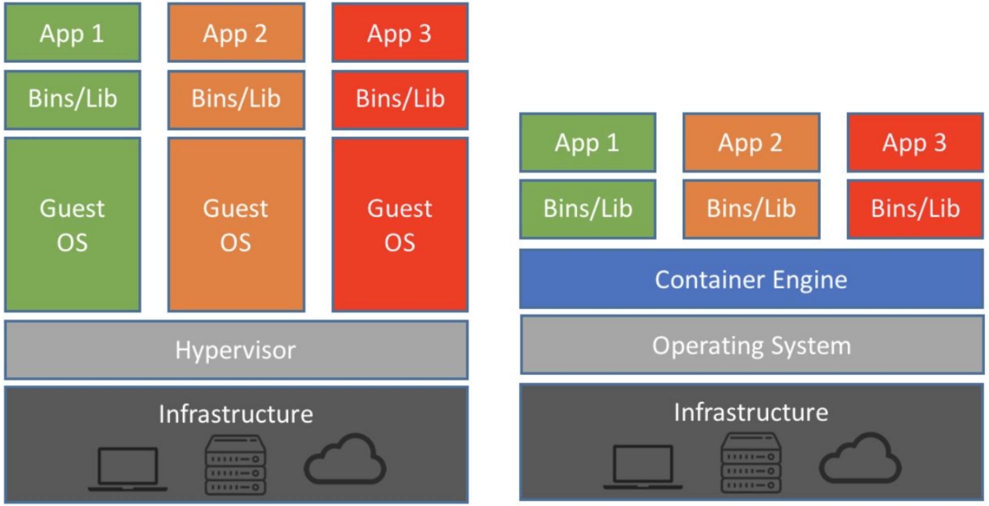
} Powerful but heavyweight

Amazon Web Services (AWS)

Google Cloud

Microsoft Azure

Containers



Machine Virtualization

Containers

- Only virtualize the OS
- Give impression of separate OS
  - Much cheaper than VMs
- e.g. Create container on Mac, but install Linux OS
  - Container still works on Mac
  - Inside container it’s like Linux
- Benefits:
  - Lightweight
  - Quick start-up time
  - Pseudo-isolation
  - Run many at once on a system

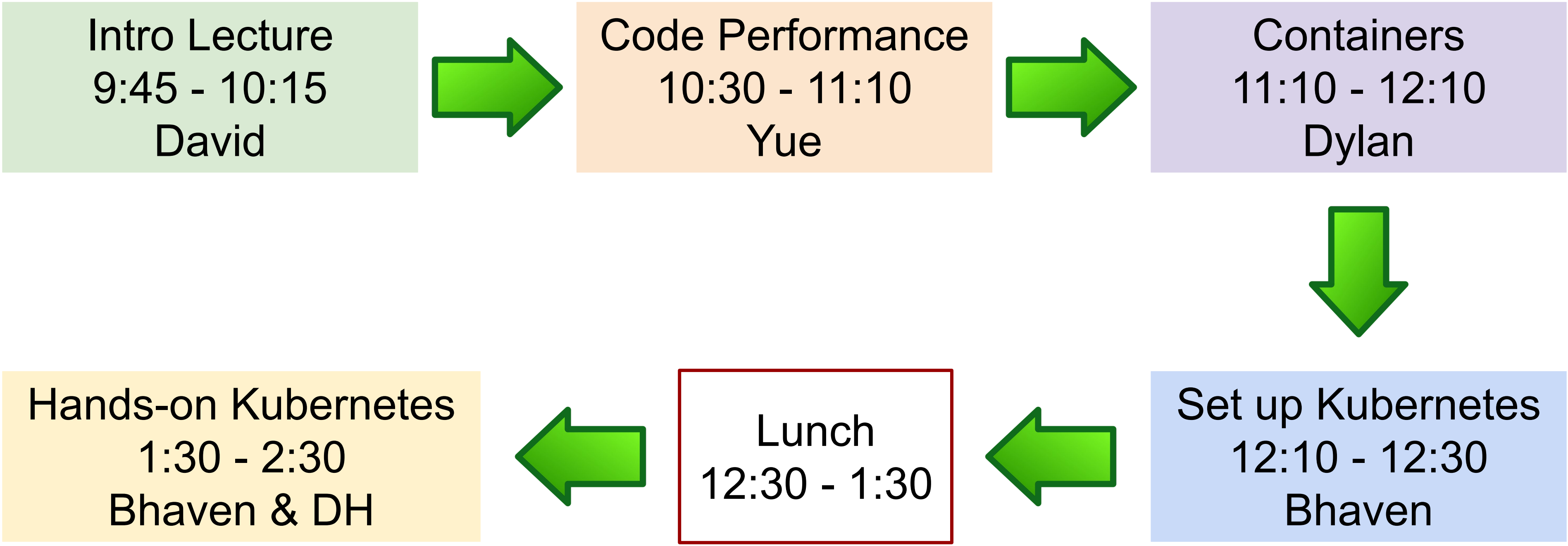
To the Cloud: Managing Containers with Kubernetes

- What if you have a bunch of containers?
  - Either working together or
  - Independent but taking up resources or
  - Both
- K8s does all of the container management for you!
- K8s is an open-source platform for container management developed by Google.
- It allows users to define rules for how container management should occur, and then it handles the rest!

There is so much more...

- We’re leaving a lot on the table today
  - Containers for high performance computing: Singularity
  - Amazon Sagemaker : Fully automated machine learning service
  - Much, much more
- Goals for today:
  - Perspective on efficient development practices
  - Best practices for deploying models - containerization
  - Deployment to the cloud and container management
  - First contact with AWS
- DH will summarize the big ideas at the very end

Workshop Overview for Session 1



Intro Lecture
9:45 - 10:15
David

Code Performance
10:30 - 11:10
Yue

Containers
11:10 - 12:10
Dylan

Set up Kubernetes
12:10 - 12:30
Bhaven

Lunch
12:30 - 1:30

Hands-on Kubernetes
1:30 - 2:30
Bhaven & DH