

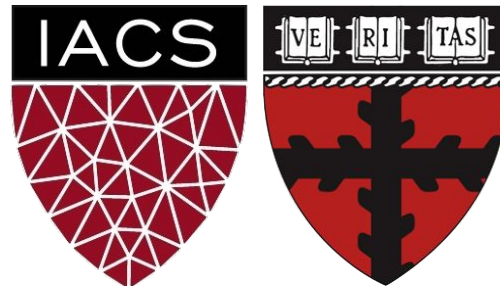
# Lecture 12: Virtual Environments & Virtual Machines

Advanced Practical Data Science, MLOps

AC295

Pavlos Protopapas

Institute for Applied Computational Science, Harvard



# Outline

---

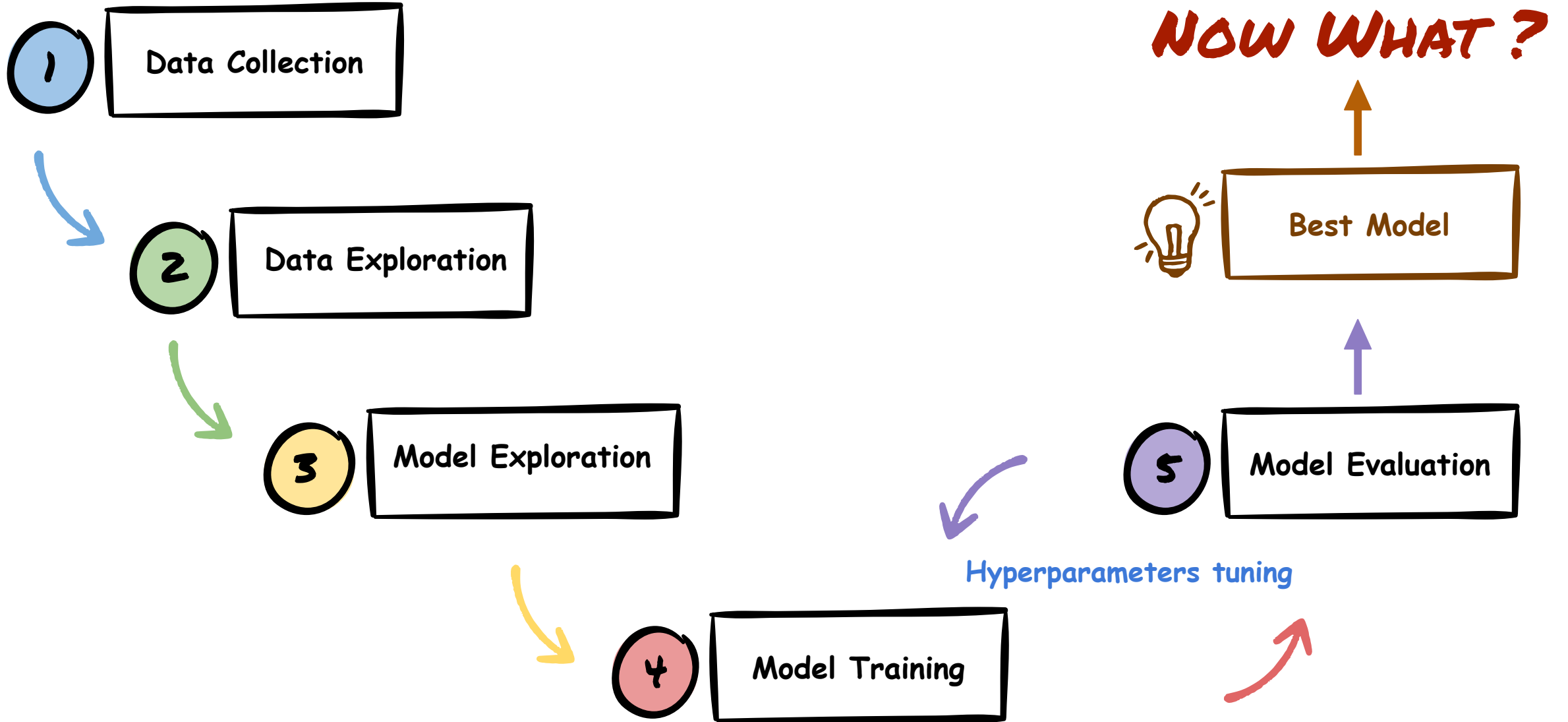
1. Where we are
2. Motivation
3. Virtual Machines
4. Virtual Environments

# Outline

---

1. **Where we are**
2. Motivation
3. Virtual Machines
4. Virtual Environments

# Where we are: Deep Learning Flow



# Leaderboard - Best Model

 Mushroom Classification Leaderboard 

Total Models Submitted: 591

|   | User                       | Score<br>*    | Model Size<br>(Mb) ** | Accuracy<br>*** | Total<br>Parameters | Accuracy<br>(Reported) | Loss<br>(Reported) |
|---|----------------------------|---------------|-----------------------|-----------------|---------------------|------------------------|--------------------|
| 1 | archclojure@gmail.com      | <b>0.8844</b> | 8.73                  | 84.33%          | 2,340,358           | 80.24%                 | 2.25               |
| 2 | neil_sehgal@g.harvard.edu  | <b>0.8799</b> | 22.33                 | 85.00%          | 6,001,686           | 79.29%                 | 1.09               |
| 3 | mbavare21@gmail.com        | <b>0.8774</b> | 22.64                 | 84.67%          | 6,084,054           | 86.05%                 | 2.50               |
| 4 | kevinhare@g.harvard.edu    | <b>0.875</b>  | 22.63                 | 84.33%          | 6,084,054           | 81.87%                 | 1.49               |
| 5 | anita.mahinpei@gmail.com   | <b>0.8681</b> | 8.72                  | 82.00%          | 2,340,358           | 75.37%                 | 1.44               |
| 6 | kxyang@ucdavis.edu         | <b>0.8659</b> | 22.33                 | 83.00%          | 6,001,686           | 77.56%                 | 0.49               |
| 7 | wangyuanbiao2016@gmail.com | <b>0.8563</b> | 26.19                 | 82.00%          | 7,047,750           | 91.18%                 | 0.69               |
| 8 | bharpe@college.harvard.edu | <b>0.8494</b> | 8.73                  | 79.33%          | 2,340,358           | 75%                    | 2.43               |

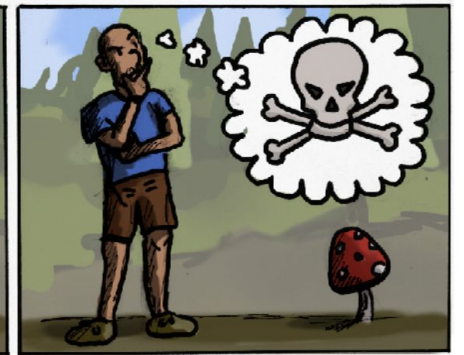
# Outline

---

1. Where we are
- 2. Motivation**
3. Virtual Machines
4. Virtual Environments

# We want to build a 🍄 Mushroom Finder App

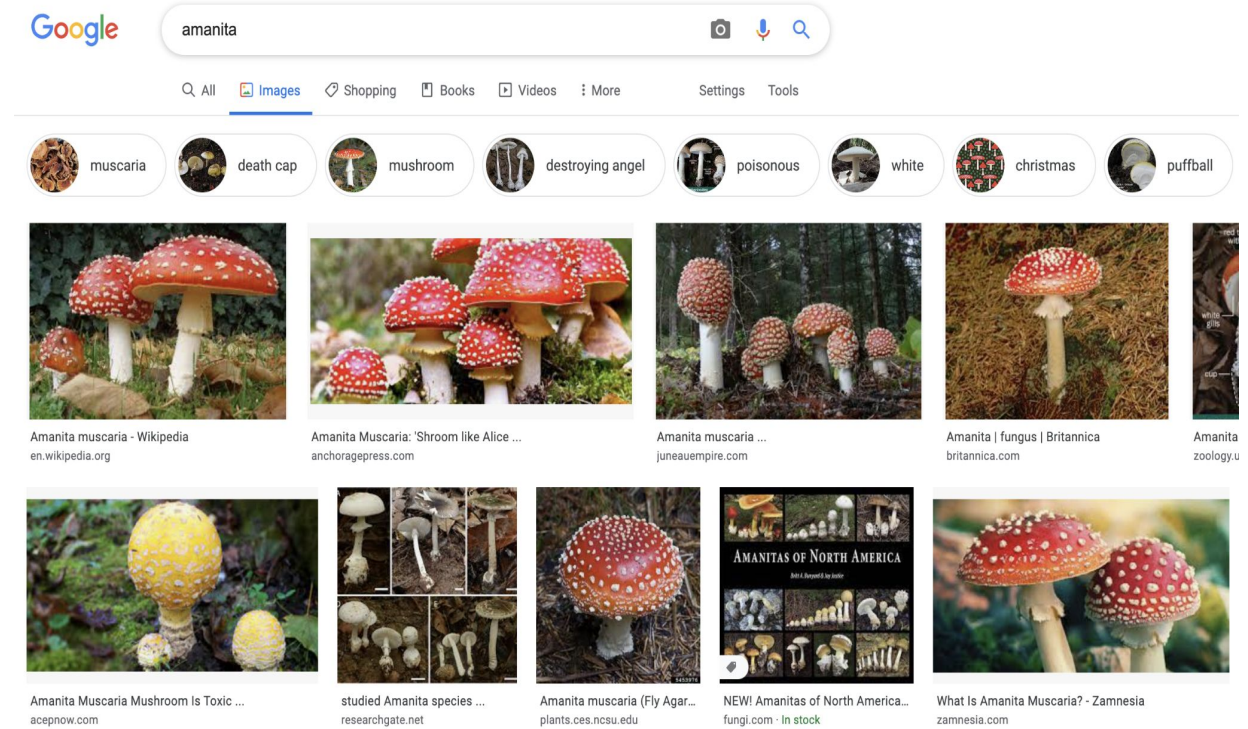
- Pavlos likes to go the forest for mushroom picking
- Some mushrooms can be poisonous
- Help build an app to identify mushroom type and if poisonous or not



Credit: Nikolas Protopapas

# Mushroom App: Data Collection

- Collect images from Google
- For our demo we downloaded images for mushrooms **oyster**, **crimini**, **amanita (Poisonous)**



Python Script





# Mushroom App: Models

- Identify our problem task
- Try various model architectures
- Transfer Learning
- Hyperparameters tuning
- In class competition

## Mushroom Classification Leaderboard 🍄

Total Models Submitted: 591

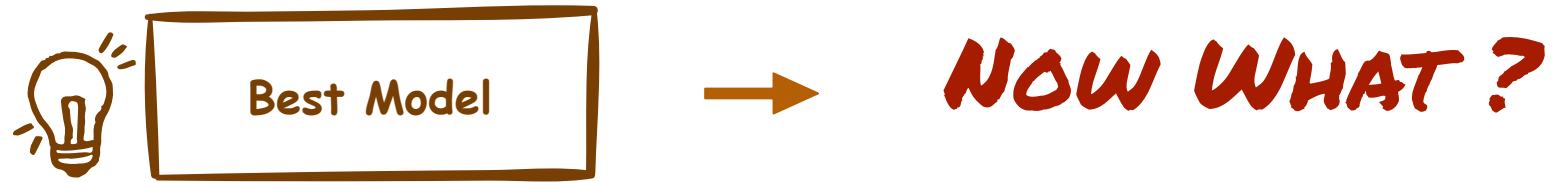
| User                         | Score * | Model Size (Mb) ** | Accuracy *** | Total Parameters | Accuracy (Reported) | Loss (Reported) |
|------------------------------|---------|--------------------|--------------|------------------|---------------------|-----------------|
| 1 archclojure@gmail.com      | 0.8844  | 8.73               | 84.33%       | 2,340,358        | 80.24%              | 2.25            |
| 2 neil_sehgal@g.harvard.edu  | 0.8799  | 22.33              | 85.00%       | 6,001,686        | 79.29%              | 1.09            |
| 3 mbavare21@gmail.com        | 0.8774  | 22.64              | 84.67%       | 6,084,054        | 86.05%              | 2.50            |
| 4 kevinhare@g.harvard.edu    | 0.875   | 22.63              | 84.33%       | 6,084,054        | 81.87%              | 1.49            |
| 5 anita.mahinpei@gmail.com   | 0.8681  | 8.72               | 82.00%       | 2,340,358        | 75.37%              | 1.44            |
| 6 kxyang@ucdavis.edu         | 0.8659  | 22.33              | 83.00%       | 6,001,686        | 77.56%              | 0.49            |
| 7 wangyuanbiao2016@gmail.com | 0.8563  | 26.19              | 82.00%       | 7,047,750        | 91.18%              | 0.69            |
| 8 bharpe@college.harvard.edu | 0.8494  | 8.73               | 79.33%       | 2,340,358        | 75%                 | 2.43            |

```
+ Code + Text
[ ]
trainable_parameters execution_time loss accuracy
2 2,388,227 3.24 mins 82.65 88.48%
3 2,306,051 3.24 mins 42.84 87.27%
1 164,355 2.31 mins 82.45 86.67%
4 22,170 3.24 mins 42.84 79.39%
64.24%
63.03%
63.03%
```

Colab

# Mushroom App: Best Model

---



# Mushroom App

- We want to build an app to take a photo of a mushroom and it helps us identify the type of mushroom
- How do we build the app?



**Type: amanita (93.54%)**

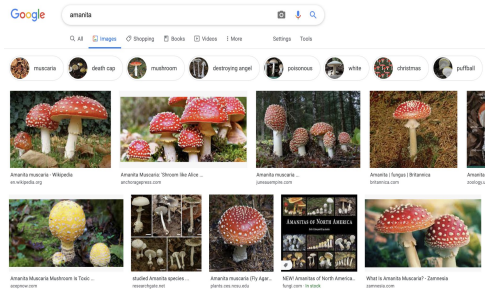
# How do we build an App?

---

- Collaborate with team to **design** and **develop**
- Expose best model as an **API**
- Build a **frontend** using HTML & javascript
- **Deploy** app to a cloud provider
- <http://awesome-mushroom-app.com> [Go live]

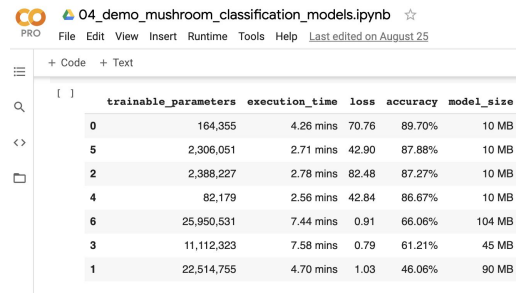
# How do we build an App?

Data Collection



Python Script

Data Exploration  
Model Exploration  
Model Training  
Model Evaluation



|   | trainable_parameters | execution_time | loss  | accuracy | model_size |
|---|----------------------|----------------|-------|----------|------------|
| 0 | 164,355              | 4.26 mins      | 70.76 | 89.70%   | 10 MB      |
| 5 | 2,306,051            | 2.71 mins      | 42.90 | 87.88%   | 10 MB      |
| 2 | 2,388,227            | 2.78 mins      | 82.48 | 87.27%   | 10 MB      |
| 4 | 82,179               | 2.56 mins      | 42.84 | 86.67%   | 10 MB      |
| 6 | 25,950,531           | 7.44 mins      | 0.91  | 66.06%   | 104 MB     |
| 3 | 11,112,323           | 7.58 mins      | 0.79  | 61.21%   | 45 MB      |
| 1 | 22,514,755           | 4.70 mins      | 1.03  | 46.06%   | 90 MB      |

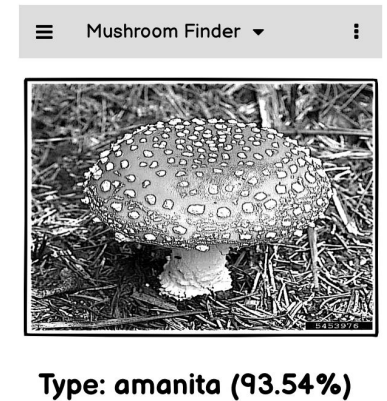
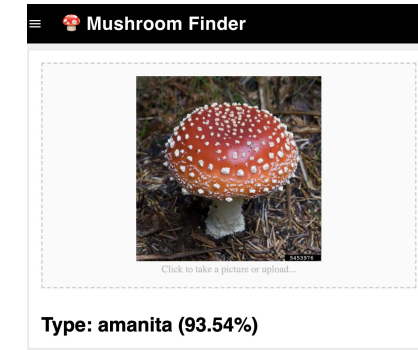
Colab



Rest API

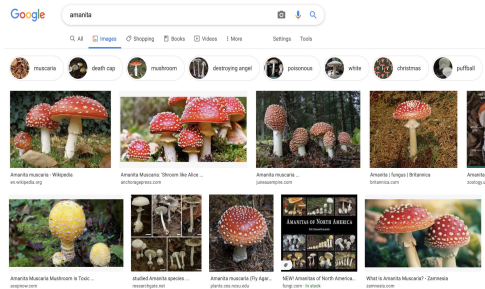
Best Model

IDE / Code Editor



# How do we build an App?

Data Collection



Python Script

- Data Exploration
- Model Exploration
- Model Training
- Model Evaluation

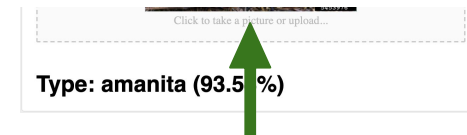
04\_demo\_mushroom\_classification\_models.ipynb

|   | trainable_parameters | execution_time | loss  | accuracy | model_size |
|---|----------------------|----------------|-------|----------|------------|
| 0 | 164,355              | 4.26 mins      | 70.76 | 89.70%   | 10 MB      |
| 5 | 2,306,051            | 2.71 mins      | 42.90 | 87.88%   | 10 MB      |
| 2 | 2,388,227            | 2.78 mins      | 82.48 | 87.27%   | 10 MB      |
| 4 | 82,179               | 2.56 mins      | 42.84 | 86.67%   | 10 MB      |
| 6 | 25,950,531           | 7.44 mins      | 0.91  | 66.06%   | 104 MB     |
| 3 | 11,112,323           | 7.58 mins      | 0.79  | 61.21%   | 45 MB      |
| 1 | 22,514,755           | 4.70 mins      | 1.03  | 46.06%   | 90 MB      |

Colab

Mushroom Finder

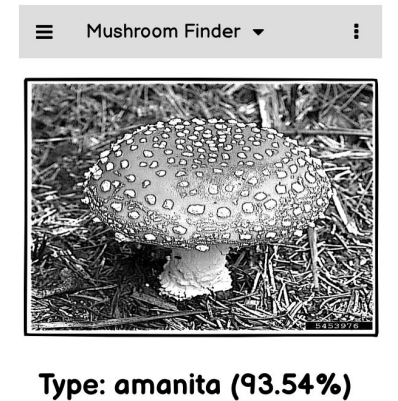
# PRODUCTIONIZING MODEL!



Rest API

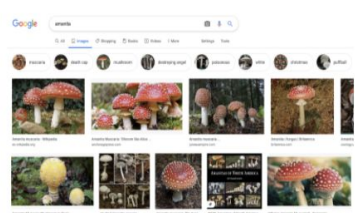
Best Model

IDE / Code Editor



# Challenges

## Development



Python Script

Python: pipenv  
Chromium: Mac install  
OS: Mac



```
cs109b_sec8.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Code Text
1 | trainable_parameters execution_time loss accuracy model_size
2 | 2,388,227 3.24 mins 82.65 88.48% 10 MB
3 | 2,306,051 3.24 mins 42.84 87.27% 10 MB
4 | 164,355 2.31 mins 82.45 86.67% 10 MB
5 | 82,179 2.26 mins 42.91 79.39% 10 MB
6 | 25,950,531 7.23 mins 1.14 64.24% 104 MB
7 | 11,112,323 7.86 mins 0.92 63.03% 45 MB
8 | 22,514,755 8.01 mins 0.80 63.03% 90 MB
9 |
10 |
11 | best_model = 'models/'+view_metrics.iloc[0]['name']+'.hdf5'
12 | print(best_model)
models/mobilenetv2_train_baseTrue_161905420.hdf5
```

Colab

Python: Colab provided env  
OS: Linux

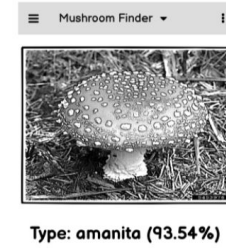


Rest API

Best Model

IDE / Code Editor

Python: pipenv  
OS: Mac



## Deployment



Python: pipenv  
OS: Linux

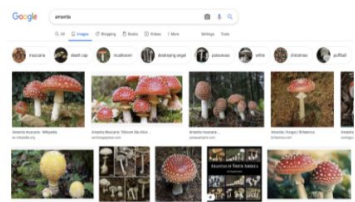
Server



One developer  
Using a Macbook

# Challenges - Multiple Developers

## Development



Python Script

Python: pipenv  
Chromium: Mac install,  
Windows install  
OS: Mac, Windows



Multiple developers, Using Mac and Windows OS

```
cs109b_sec8.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Code Text
[ ] trainable_parameters execution_time loss accuracy model_size
2 2,388,227 3.24 mins 82.65 88.48% 10 MB
3 2,306,051 3.24 mins 82.84 87.27% 10 MB
1 164,355 2.31 mins 82.45 86.67% 10 MB
4 82,179 2.26 mins 82.91 79.39% 10 MB
6 25,950,531 7.23 mins 1.14 64.24% 104 MB
0 11,112,323 7.86 mins 0.92 63.03% 45 MB
5 22,514,755 8.01 mins 0.80 63.03% 90 MB
[ ] best_model = 'modela/'+view_metrics.iloc[0]['name']+'.hdf5'
2 print(best_model)
modela/mobilenetv2_train_baseTrue_161905420.hdf5
```

Colab

Python: Colab provided env  
OS: Linux



Rest API

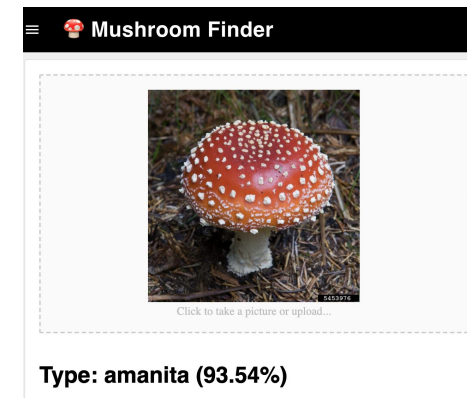
Best Model

IDE / Code Editor

Python: pipenv  
OS: Mac, Windows



## Deployment

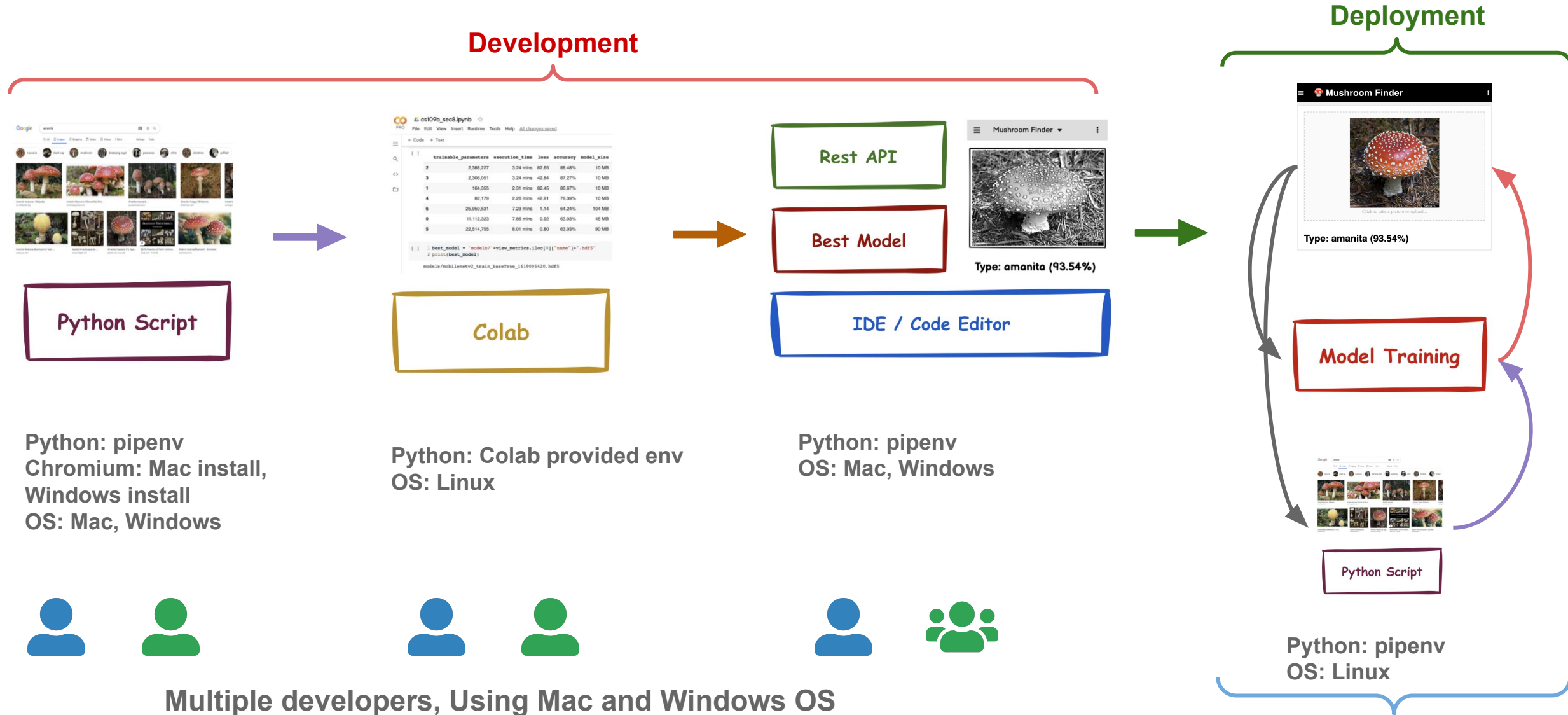


Python: pipenv  
OS: Linux

Server



# Challenges - Multiple Developers + Automation



# Challenges / Solutions

---

## Challenges:

- OS specific installations required
- How to collaborate code/frameworks?
- Automate data collection / model training
- New team member onboarding
- “It works on my machine” ㄟ(ツ)ㄟ

## Solutions:

- Isolate development into environments that can be shared
- Develop in an common OS regardless of developers host OS
- Track software/framework installs

# Tools

---

- Virtual Machines
- Virtual Environments
- Containers

# Outline

---

1. Where we are
2. Motivation
- 3. Virtual Machines**
4. Virtual Environments

# Virtual Machines Demo

<https://cloud.google.com/billing/docs/how-to/edu-grants>

[Google Cloud: Cloud Computing Services](#)

The screenshot shows the Google Cloud website homepage. At the top left is the Google Cloud logo. The navigation menu includes links for 'Why Google', 'Solutions', 'Products', 'Pricing', 'Getting Started', and 'Contact Us'. On the right side, there are icons for search, a document, 'Docs', 'Support', a 'Language' dropdown, and a 'Console' button with a globe icon. A yellow banner across the top of the main content area reads: 'New customers get \$300 in free credits to spend on Google Cloud. All customers get free usage of 20+ products. [See offer details.](#)'

The main content area features a large heading: 'Accelerate your transformation with Google Cloud'. Below this is the subtext: 'Build apps faster, make smarter business decisions, and connect people anywhere.' There are two buttons: a blue 'Contact sales' button and a white 'Go to console' button.

To the right is a carousel slide for 'Google Cloud Next '21'. The slide text reads: 'Join our global digital experience, streaming live now'. Below the text is a 'Watch now for free' link. The slide also includes the text: 'Choose from 100+ free sessions, demos, and moments to learn from Google experts.' The slide has left and right navigation arrows and a progress indicator at the bottom.

Annotations on the screenshot include a black circle around the 'Console' button in the top right navigation bar, and a black arrow pointing from the bottom right towards the 'Console' button.

# Virtual Machines Demo

Go to Navigation Menu

The screenshot displays the Google Cloud Platform console interface. At the top, there is a blue header bar with the text "Google Cloud Platform" and "Preparing For Class". Below the header, there is a navigation menu with options like "ACTIVITY" and "RECOMMENDATIONS". A red box highlights the "Navigation menu" button, and a red arrow points to it from the text "Go to Navigation Menu". The main content area shows a notification about Google Cloud Next '21, followed by three panels: "Project info" (showing project name, ID, and number), "API APIs" (showing a graph of requests per second), and "Google Cloud Platform status" (showing "All services normal").

Google Cloud Platform | Preparing For Class

Navigation menu | ACTIVITY | RECOMMENDATIONS

Search products and resources

Google Cloud Next '21 is live. Join us now at [g.co/cloudnext](https://g.co/cloudnext). DISMISS

**Project info**

- Project name: Preparing For Class
- Project ID: preparing-for-class
- Project number: 406706675980

[ADD PEOPLE TO THIS PROJECT](#)

**API APIs**

Requests (requests/sec)

| Requests (requests/sec) |
|-------------------------|
| 1.0                     |
| 0.8                     |
| 0.6                     |
| 0.4                     |
| 0.2                     |

No data is available for the selected time frame.

**Google Cloud Platform status**

All services normal

[Go to Cloud status dashboard](#)

**Billing**

Estimated charges: USD \$0.00  
For the billing period Oct 1 - 12, 2021

# Virtual Machines Demo

## Select compute engine

The screenshot displays the Google Cloud Platform (GCP) console interface. The top navigation bar is blue and contains the text "Google Cloud Platform" on the left, a search bar with the placeholder "Search products and resources" in the center, and user profile icons on the right. A left-hand navigation menu is open, listing various services. The "Compute Engine" option is highlighted with a mouse cursor, and a black arrow points from this menu item to the "Project info" section of the main dashboard. The main dashboard area is divided into several panels: "Project info" (showing project name "Preparing For Class", ID "preparing-for-class", and number "406706675980"), "APIs" (showing a line graph for "Requests (requests/sec)" with a note "No data is available for the selected time frame."), "Google Cloud Platform status" (reporting "All services normal"), "Billing" (showing "Estimated charges USD \$0.00"), "Monitoring" (with options like "Create my dashboard"), and "Trace" (showing "No trace data from the past 7 days").

# Virtual Machines Demo

## Select Virtual Machines

The screenshot shows the Google Cloud Platform interface for VM instances. The top navigation bar includes the Google Cloud Platform logo, the user's profile (Preparing For Class), and a search bar. The main header is 'Compute Engine' with a sub-header 'VM instances' and several action buttons: CREATE INSTANCE, IMPORT VM, REFRESH, START / RESUME, STOP, SUSPEND, RESET, DELETE, and CREATE SCHEDULE. A left sidebar lists various services under 'Virtual machines', 'Storage', 'Instance groups', 'VM Manager', and 'Settings'. The main content area features a filter bar and a table with columns: Status, Name, Zone, Recommendations, In use by, Internal IP, External IP, and Connect. Below the table is a central graphic with a globe and the text 'VM Instances' and 'Compute Engine lets you use virtual machines that run on Google's infrastructure. Create micro-VMs or larger instances running Debian, Windows, or other standard images. Create your first VM instance, import it using a migration service, or try the quickstart to build a sample app.' At the bottom of the graphic are two buttons: 'CREATE INSTANCE' and 'TAKE THE QUICKSTART'.



# Virtual Machines Demo

## Select all defaults

The screenshot shows the Google Cloud Platform interface for creating a new VM instance. The page is titled "Create an instance" and features a navigation bar at the top with "Google Cloud Platform" and "Preparing For Class" on the left, and a search bar on the right. The main content area is divided into a left sidebar and a main configuration panel.

**Left Sidebar:** Contains four options for creating a VM instance:

- New VM instance:** Create a single VM instance from scratch (selected).
- New VM instance from template:** Create a single VM instance from an existing template.
- New VM instance from machine image:** Create a single VM instance from an existing machine image.
- Marketplace:** Deploy a ready-to-go solution onto a VM instance.

**Main Configuration Panel:**

- Machine types:** Includes tabs for GENERAL-PURPOSE, COMPUTE-OPTIMIZED, MEMORY-OPTIMIZED, and GPU. The description states: "Machine types for common workloads, optimized for cost and flexibility."
- Series:** A dropdown menu set to "E2".
- Machine type:** A dropdown menu set to "e2-medium (2 vCPU, 4 GB memory)".
- Resource Allocation:** A table showing "vCPU" as "1 shared core" and "Memory" as "4 GB".
- CPU PLATFORM AND GPU:** A section with a dropdown arrow.
- Display device:** A section with the description "Enable to use screen capturing and recording tools." and an unchecked checkbox "Enable display device".
- Confidential VM service:** A section with the description "Enable the Confidential Computing service on this VM instance." and an unchecked checkbox.
- Container:** A section with the description "Deploy a container image to this VM instance" and a "DEPLOY CONTAINER" button.
- Boot disk:** A section with a "CHANGE" button and a table of disk settings:

|           |                              |
|-----------|------------------------------|
| Disk type | New balanced persistent disk |
| Disk size | 10 GB                        |
| Image     | Debian GNU/Linux 10 (buster) |
- Identity and API access:** A section with a "CHANGE" button and a "Service accounts" dropdown menu set to "Compute Engine default service account".
- Access scopes:** A section with three radio button options:
  - Allow default access
  - Allow full access to all Cloud APIs
  - Set access for each API

**Monthly estimate:** Located on the right side of the configuration panel, it shows a total of "\$25.46" and a note: "That's about \$0.03 hourly. Pay for what you use: No upfront costs and per second billing." Below this is a "DETAILS" link.

# Virtual Machines Demo

Wait for instance to start and click on ssh

Google Cloud Platform

Preparing For Class

Search products and resources

Compute Engine

VM instances

CREATE INSTANCE IMPORT VM REFRESH START / RESUME STOP SUSPEND RESET DELETE CREATE SCHEDULE

Virtual machines

VM instances

Instance templates

Sole-tenant nodes

Machine images

TPUs

Committed use discounts

Migrate for Compute Engi...

Storage

Disks

Snapshots

Images

Instance groups

Instance groups

Health checks

VM Manager

INSTANCES INSTANCE SCHEDULE

VM instances are highly configurable virtual machines for running workloads on Google infrastructure. [Learn more](#)

Filter Enter property name or value

| Status                              | Name ↑     | Zone          | Recommendations | In use by | Internal IP       | External IP    | Connect |
|-------------------------------------|------------|---------------|-----------------|-----------|-------------------|----------------|---------|
| <input checked="" type="checkbox"/> | instance-1 | us-central1-a |                 |           | 10.128.0.7 (nic0) | 34.132.242.220 | SSH     |

Related actions

DISMISS

- View billing report**  
View and manage your Compute Engine billing
- Monitor VMs**  
View outlier VMs across metrics like CPU and network
- Explore VM logs**  
View, search, analyze, and download VM instance logs
- Set up firewall rules**  
Control traffic to and from a VM instance
- Patch management**  
Schedule patch updates and view patch compliance on VM instances

# Virtual Machines Demo

And here is your virtual machine

```
ssh.cloud.google.com/projects/preparing-for-class/zones/us-central1-a/instances/instance-1?authuser=0&hl=en_US&projectN...  
Connected, host fingerprint: ssh-rsa 0 B3:0F:76:49:9A:9A:D5:DD:7C:CC:3B:2B:2E:5B  
18:DB:C0:2C:D0:B3:EE:98:31:F3:10:8E:02:54:CC:E4:72:BE  
Linux instance-1 4.19.0-17-cloud-amd64 #1 SMP Debian 4.19.194-3 (2021-07-18) x86  
_64  
  
The programs included with the Debian GNU/Linux system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.  
rootopapas@instance-1:~$ █
```

git clone <https://github.com/dlops-io/simple-translate.git>

# Why should we use virtual machines?

---

## Motivation

- All team members want an identical machine with same OS
- Should be easy create and delete instances
- All softwares installations need to be same across team members
- Team members need to run the same experiments in Isolation!

**Virtual Machines!**

# Why should we use virtual machines?

---

## Advantages

- Full autonomy: it works like a separate computer system; it is like running a computer within a computer.
- **Very secure**: the software inside the virtual machine cannot affect the actual computer.
- Lower costs: buy one machine and run multiple operating systems.
- Used by all Cloud providers for on demand server instances.

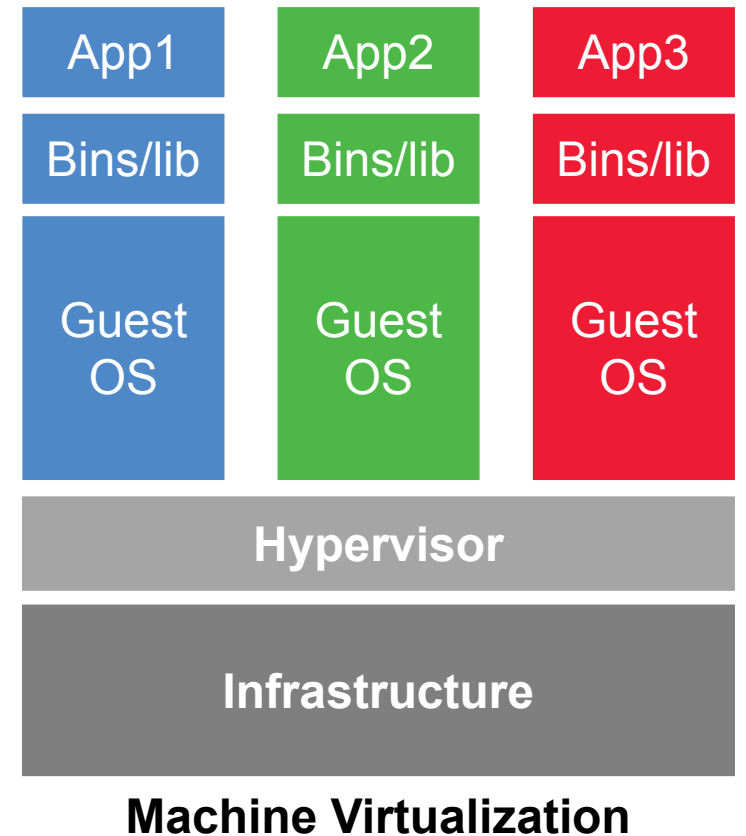
## Softwares used for Virtualization

- VirtualBox
- VMWare
- Parallels

# Why should we use virtual machines?

## Advantages

- virtual machines have their own virtual hardware: CPUs, memory, hard drives, etc.
- you need a **hypervisor** that manages different virtual machines on server
- hypervisor can run as **many** virtual machines as we wish
- operating system is called the "**host**" while those running in a virtual machine are called "**guest**"
- You can install a completely different operating system on this virtual machine



# Why should we use virtual machines?

---

## Limitations

- Uses hardware in local machine
- Not very portable since size of VMs are large
- There is an overhead associated with virtual machines
  - Guest is not as fast as the host system
  - Takes a long time to start up
  - It may not have the same graphics capabilities

# Outline

---

1. Where we are
2. Motivation
3. Virtual Machines
4. **Virtual Environments**



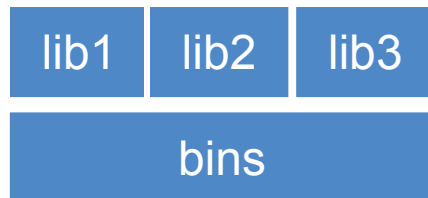
# Why should we use virtual environment?

---

- Virtual environments help to make development and use of code more **streamlined**.
- Virtual environments keep dependencies in separate “**sandboxes**” so you can switch between both applications easily and get them running.
- Given an operating system and hardware, we can get the exact code environment set up using **different technologies**. This is key to understand the trade off among the different technologies presented in this class.

# Why should we use virtual environment?

- Maggie took CS109, she used to run her Jupyter notebooks from anaconda prompt. Every time she installed a module it was placed in the either of `bin`, `lib`, `share`, `include` folders and she could import it in and used it without any issue.



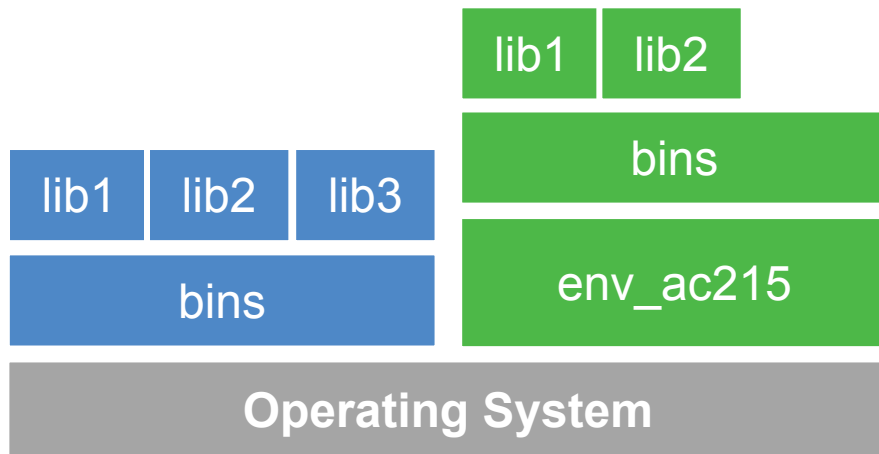
Operating System

**Maggie**

```
$ which python  
/c/Users/maggie/Anaconda3/python
```

# Why should we use virtual environment?

- Maggie starts taking AC215, and she thinks that it would be good to **isolate** the new environment from the previous environments avoiding any conflict with the installed packages. She adds a layer of **abstraction** called virtual environment that helps her keep the modules **organized** and avoid misbehaviors while developing a new project.

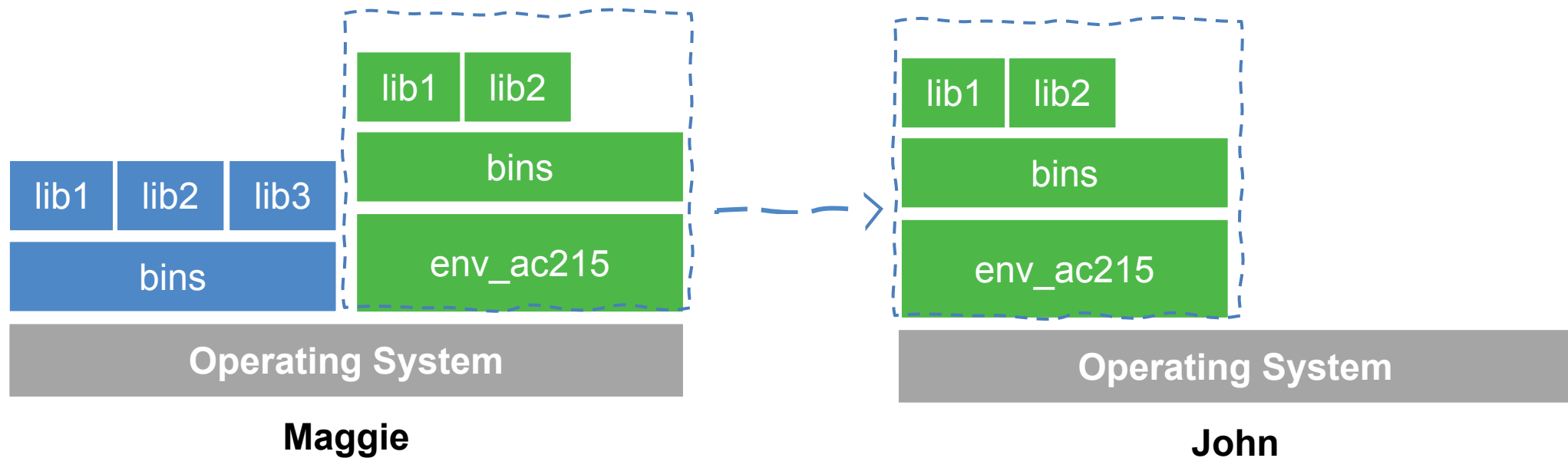


**Maggie**

```
$ which python  
/c/Users/maggie/Anaconda3/envs/env_ac215/python
```

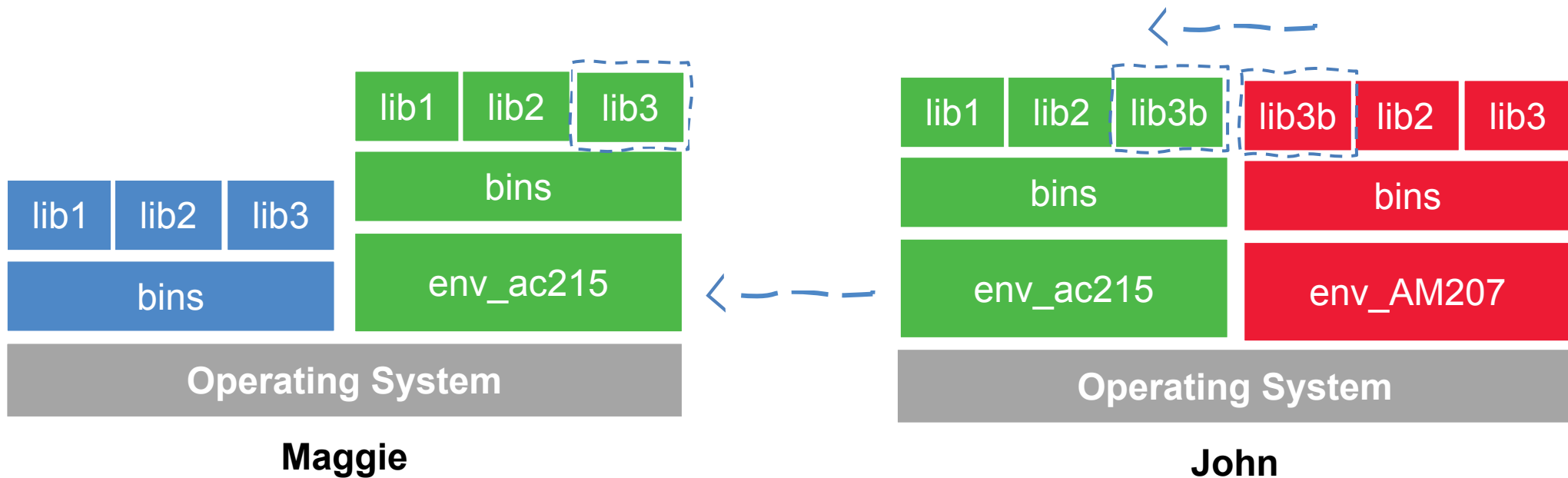
# Why should we use virtual environment?

- Maggie collaborates with John for the final project and shares the environment she is working on through .yml file (for conda env).



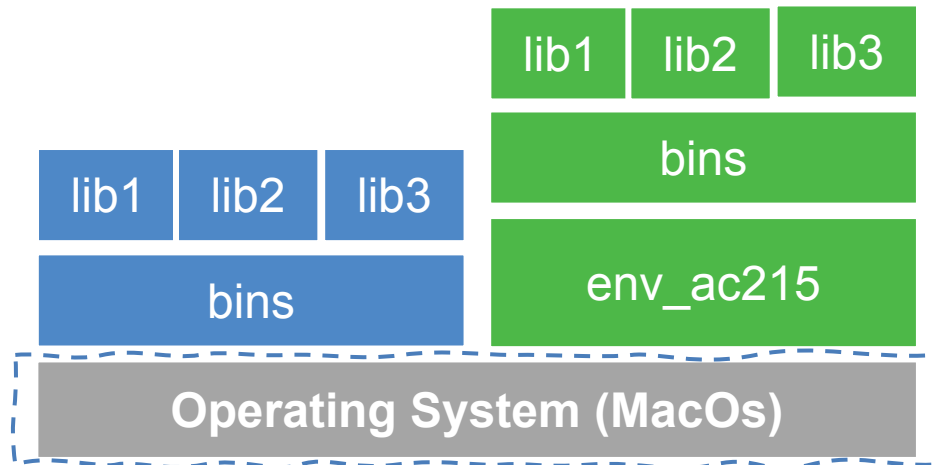
# Why should we use virtual environment?

- John experiments with a new method he learned in another class and adds a new library to the working environment. After seeing tremendous improvements, he sends Maggie back his code and a new .yml file (for conda env). She can now update her environment and replicate the experiment.

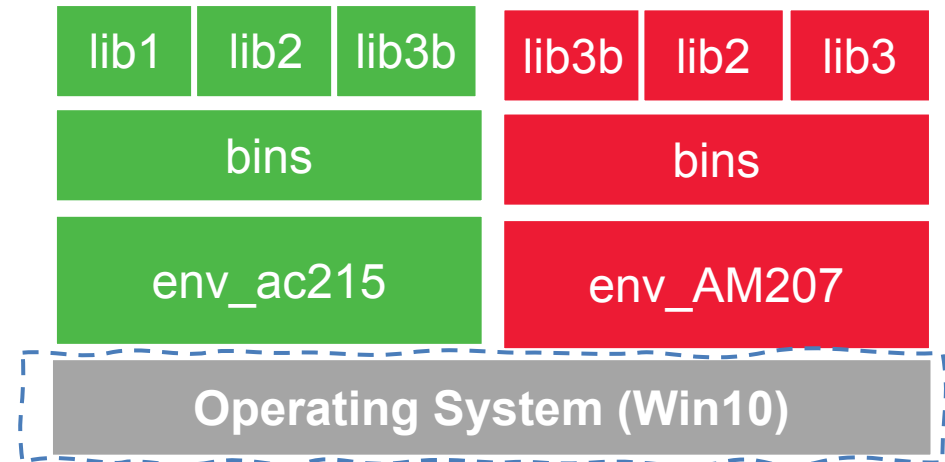


# Why should we use virtual environment?

- [What could go wrong?](#)
- Unfortunately, Maggie and John reproduce different results, and they think the issue relates to their operating systems. Indeed while Maggie has a MacOS, John uses a Win10.



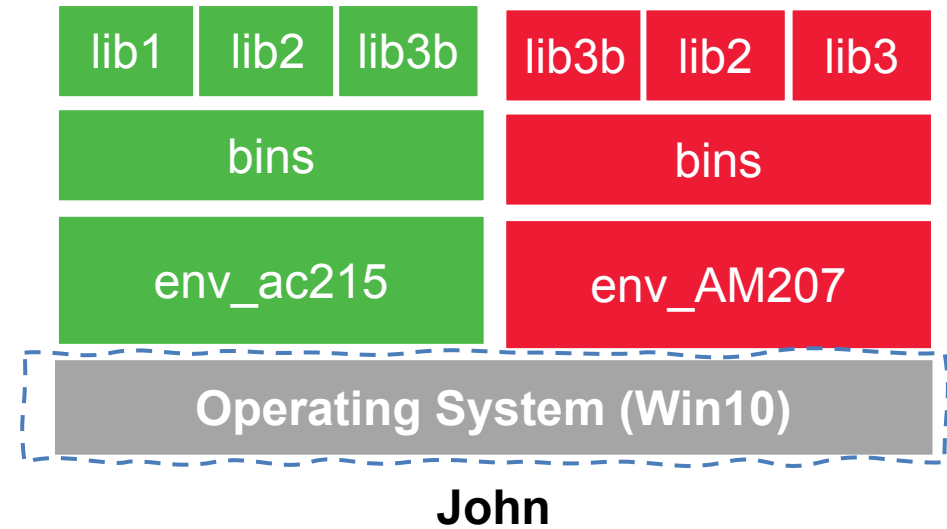
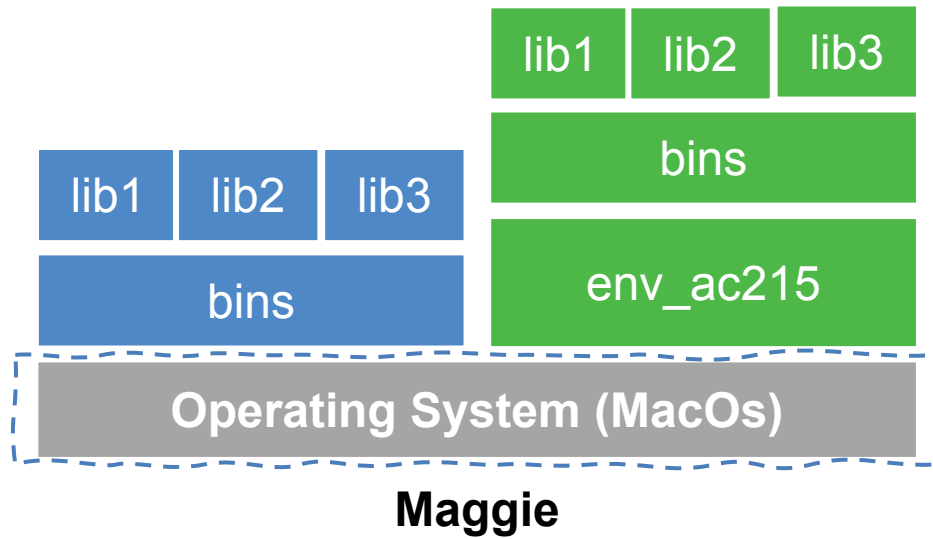
**Maggie**



**John**

# Why should we use virtual environment?

- What could go wrong?



# Virtual environments

---

## Pros

- Reproducible research
- Explicit dependencies
- Improved engineering collaboration

## Cons

- Difficulty setting up your environment
- Not isolation
- Does not always work across different OS



# What are virtual environments then?

---

- A virtual environment is an isolated python environment where the interpreter can run libraries and scripts independently from other virtual environments
- Think of a virtual environment as a directory with the following **components**:
  - *site\_packages/* directory where third-party libraries are installed
  - *links* [really symlinks] to the executables on your system
  - some *scripts* that ensure that the code uses the interpreter and site packages in the virtual environment

# Creating Virtual Environments

---

- **VirtualEnv**

- The default way to create virtual environments in python

- **Conda**

- Is a package manager and environment manager for Data Scientists

- **PipEnv**

- Production-ready tool that aims to bring the best of all packaging worlds to the Python world

# VirtualEnv

---

- Virtual environments manager embedded in Python
- Incorporated into broader tools such as pipenv
- Allow to install modules using pip package manager

# VirtualEnv

---

## How to use it:

- create an environment within your project folder `virtualenv your_env_name`
- it will add a folder called `environment_name` in your project directory
- activate environment: `source env/bin/activate`
- install requirements using: `pip install package_name=version`
- deactivate environment once done: `deactivate`

# Conda

---

- Virtual environments manager embedded in Anaconda
- Allow to use both conda and pip to manage and install packages
- Virtual environments comes pre-installed with various engineering and data science packages

# Conda

---

## How to use it:

- create an environment

```
conda create --name your_env_name python=3.7
```

- it will add a folder located within your anaconda installation

```
/Users/your_username /anaconda3/envs/your_env_name
```

- activate environment `conda activate your_env_name` (should appear in your shell)
- install requirements using `conda install package_name=version`
- deactivate environment once done `conda deactivate`
- duplicate your environment using **YAML file** `conda env export > my_environment.yml`
- to recreate the environment now use `conda env create -f environment.yml`

# Conda

---

## How to use it:

- find which environment you are using

```
conda env list
```

- create an environment

```
conda create --name your_env_name python=3.7
```

- it will add a folder located within your anaconda installation

```
/Users/your_username/[opt]/anaconda3/envs/your_env_name
```

- activate environment

```
conda activate your_env_name (should appear in your shell)
```

- install requirements using

```
conda install package_name=version
```

- deactivate environment once done

```
conda deactivate
```

- duplicate your environment using YAML file `conda env export > my_environment.yml`

- to recreate the environment now use `conda env create -f environment.yml`

# PipEnv

---

- Built on top of *VirtualEnv*
- Fixes many shortcomings of *VirtualEnv*
- Distinguish **development** vs. **production** environments
- Automatically keeps track of packages and package dependencies using a `Pipfile` & `Pipfile.lock`



# PipEnv

---

## How to use it:

- Need to `pip install pipenv`
- To create a new environment run `pipenv install`
- Activate the environment by `pipenv shell`
- To install a new package `pipenv install numpy` or `pip install numpy` (this will not lock the package automatically)
- To sync from an existing Pipfile: `pipenv sync`

# More on Virtual environments

---

## Further readings

- Pipenv: Python Dev Workflow for Humans

<https://pipenv.pypa.io/en/latest/>

- For detailed discussions on similarities and differences among virtualenv and conda

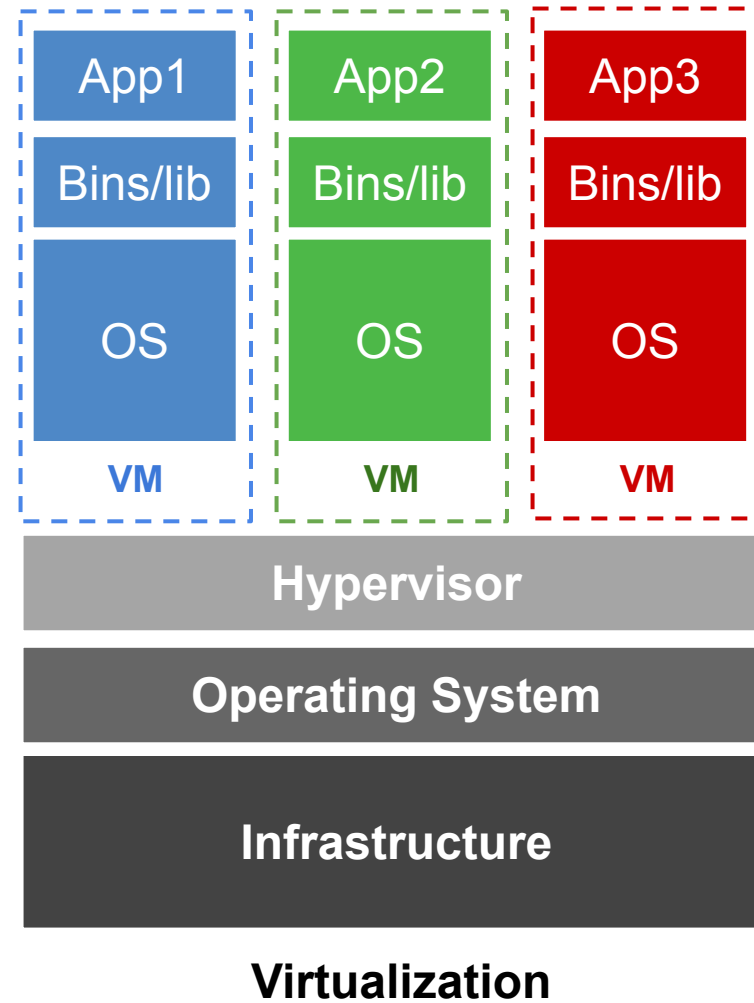
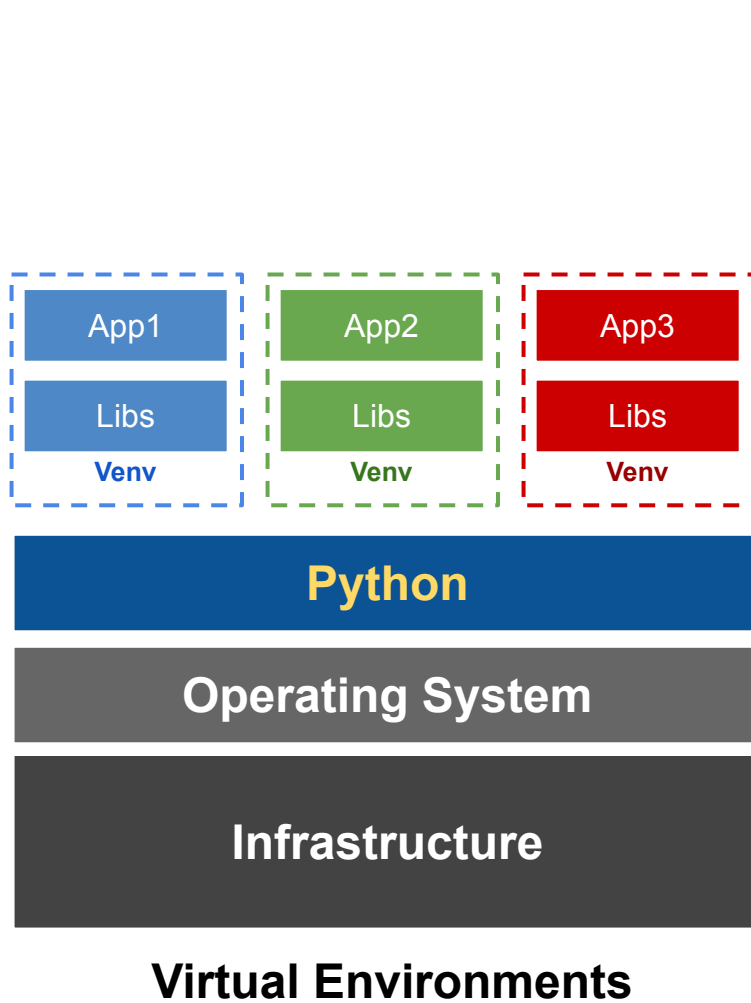
<https://jakevdp.github.io/blog/2016/08/25/conda-myths-and-misconceptions/>

- More on venv and conda environments

<https://towardsdatascience.com/virtual-environments-104c62d48c54>

<https://towardsdatascience.com/getting-started-with-python-environments-using-conda-32e9f2779307>

# Virtual Environments vs Virtual Machine



**THANK YOU**