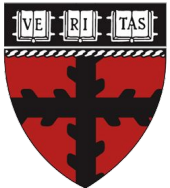


Lecture 2: Virtual Machines & Virtual Environments

AC215

Pavlos Protopapas
SEAS/Harvard



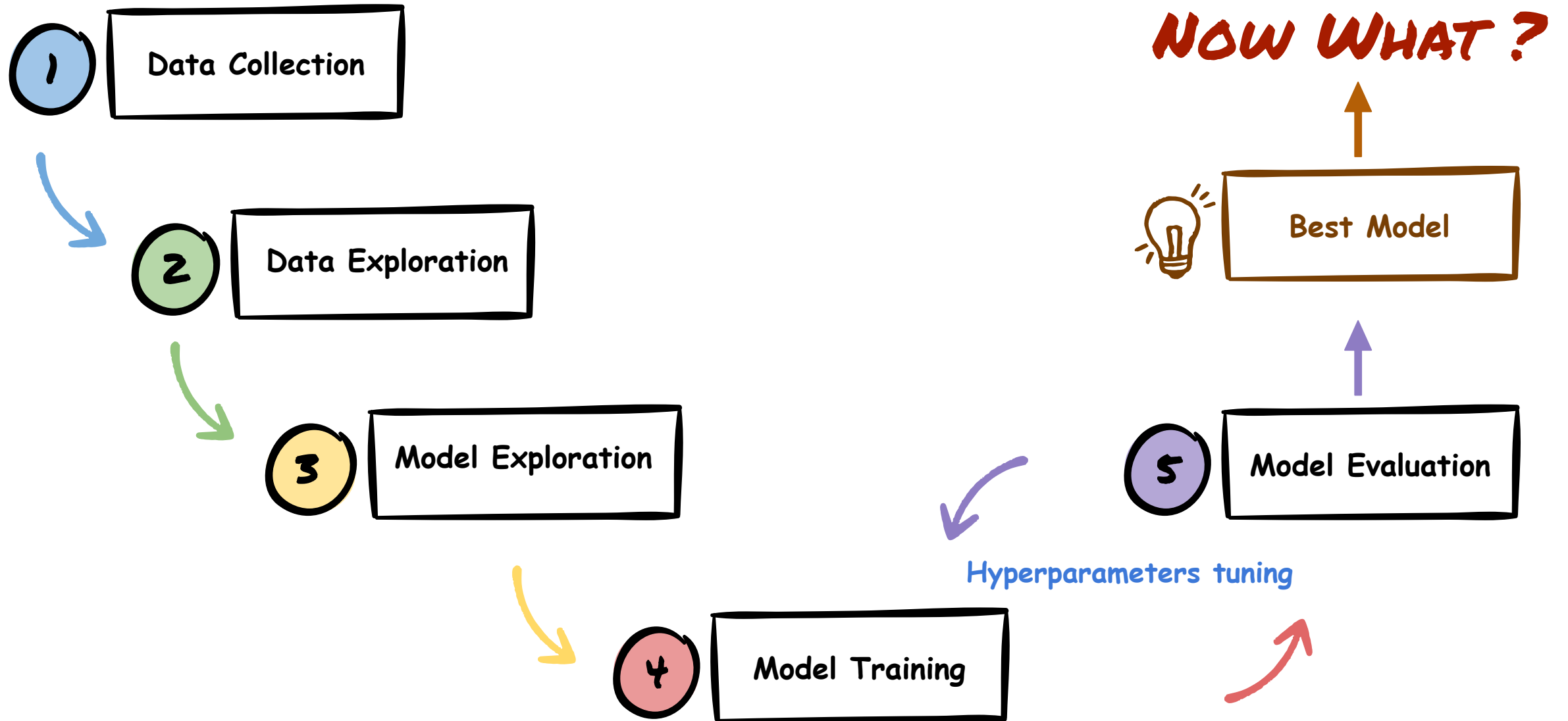
Outline

1. Motivation
2. Virtual Machines
3. Virtual Environments

Outline

- 1. Motivation**
2. Virtual Machines
3. Virtual Environments

Motivation: Deep Learning Flow

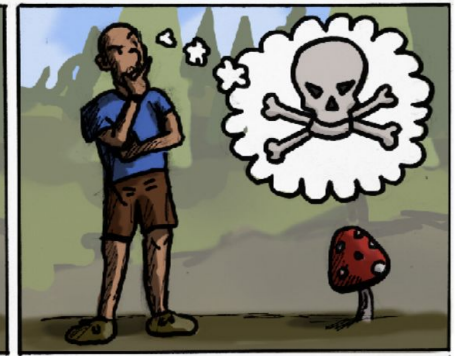


Motivation: Best Model

trainable_parameters	execution_time	loss	accuracy	model_size	learning_rate	batch_size	epochs	optimizer	name
2,306,051	2.97 mins	42.87	90.91%	10 MB	0.001	32	10	SGD	tthub_mobilenetv2_train_base_True
82,179	3.19 mins	42.79	90.30%	10 MB	0.001	32	10	SGD	tthub_mobilenetv2_train_base_False
164,355	3.91 mins	70.97	89.09%	10 MB	0.001	32	15	SGD	mobilenetv2_train_base_False
2,388,227	2.95 mins	82.03	88.48%	10 MB	0.001	32	10	SGD	mobilenetv2_train_base_True
11,112,323	6.85 mins	0.79	67.88%	44 MB	0.010	32	25	SGD	4_block
25,950,531	8.19 mins	0.74	66.67%	104 MB	0.010	32	25	SGD	2_block
22,514,755	4.78 mins	1.07	41.21%	90 MB	0.010	32	15	SGD	vgg_style

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

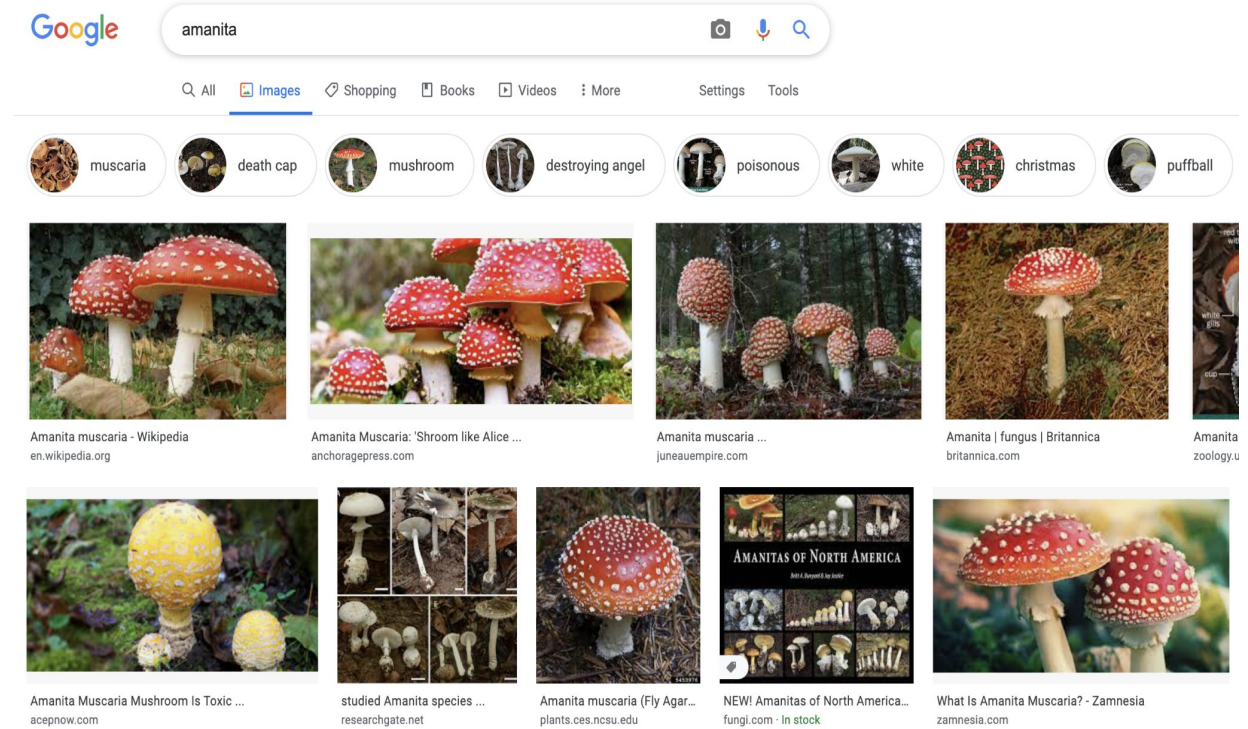


Credit: Nikolas Protopapas



Mushroom App: Data

- Collect images from Google
- For our demo we downloaded images for mushrooms **oyster**, **crimini**, **amanita (Poisonous)**
- Images organized into 3 labels



Python Script

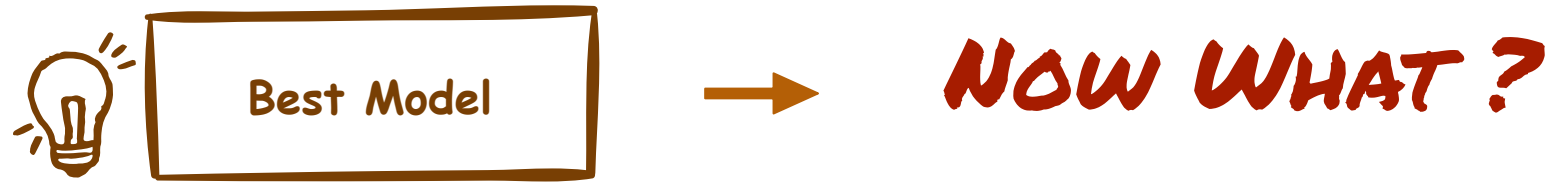
Mushroom App: Models

- Identify our problem task
- Try various model architectures
- Transfer Learning
- Hyperparameters tuning
- Experiment Tracking

<code>trainable_parameters</code>	<code>execution_time</code>	<code>loss</code>	<code>accuracy</code>
2,306,051	2.97 mins	42.87	90.91%
82,179	3.19 mins	42.79	90.30%
164,355	3.91 mins	70.97	89.09%
2,388,227	2.95 mins	82.03	88.48%
11,112,323	6.85 mins	0.79	67.88%
25,950,531	8.19 mins	0.74	66.67%
22,514,755	4.78 mins	1.07	41.21%

[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?



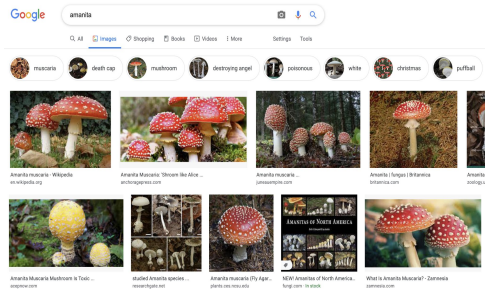
Type: amanita (93.54%)

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
- **Integrate** model prediction API into the app
- **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

O1_tutorial_mushroom_classification_models.ipynb

	trainable_parameters	execution_time	loss	accuracy	model_size
5	2,306,051	2.97 mins	42.87	90.91%	10 MB
4	82,179	3.19 mins	42.79	90.30%	10 MB
2	164,355	3.91 mins	70.97	89.09%	10 MB
6	2,388,227	2.95 mins	82.03	88.48%	10 MB
1	11,112,323	6.85 mins	0.79	67.88%	44 MB
0	25,950,531	8.19 mins	0.74	66.67%	104 MB
3	22,514,755	4.78 mins	1.07	41.21%	90 MB

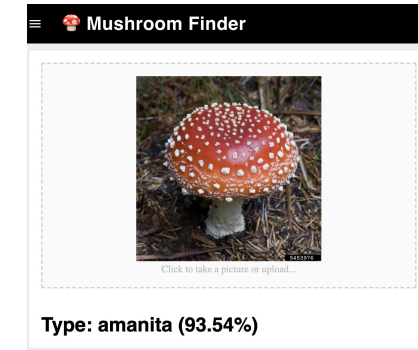
Colab



Rest API

Best Model

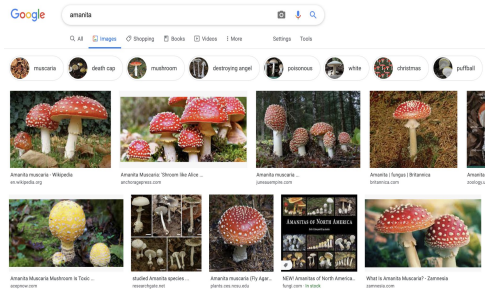
IDE / Code Editor



Type: amanita (93.54%)

How do we build an App?

Data Collection



Python Script

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

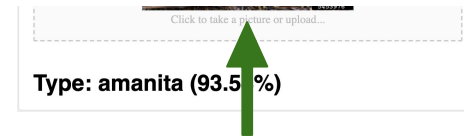
O1_tutorial_mushroom_classification_models.ipynb

	trainable_parameters	execution_time	loss	accuracy	model_size
5	2,306,051	2.97 mins	42.87	90.91%	10 MB
4	82,179	3.19 mins	42.79	90.30%	10 MB
2	164,355	3.91 mins	70.97	89.09%	10 MB
6	2,388,227	2.95 mins	82.03	88.48%	10 MB
1	11,112,323	6.85 mins	0.79	67.88%	44 MB
0	25,950,531	8.19 mins	0.74	66.67%	104 MB
3	22,514,755	4.78 mins	1.07	41.21%	90 MB

Colab

Mushroom Finder

PRODUCTIONIZING MODEL!



Rest API

Best Model

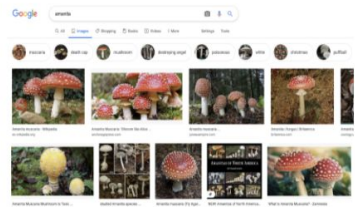
IDE / Code Editor



Type: amanita (93.54%)

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 = 'modela/'+view_metrics.iloc[0]['name']+'.hdf5'
12 | print(best_model)
modela/mobilenetv2_train_baseTrue_161905420.hdf5
```

Colab

Python: Colab provided env
OS: Linux

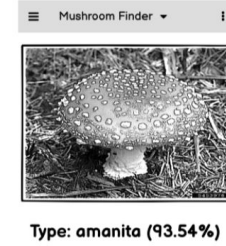


Rest API

Best Model

IDE / Code Editor

Python: pipenv
OS: Mac



Deployment



Python: pipenv
OS: Linux

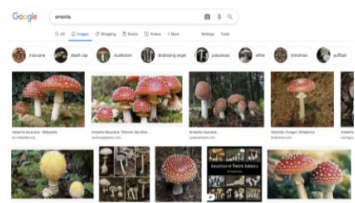
Server



One developer
Using a Macbook

Challenges - Multiple Developers

Development



Python Script



```
cs109b_sec8.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Code Text
[ ] trainable_parameters execution_time loss accuracy model_size
2 2,388,327 3.24 mins 82.65 88.48% 10 MB
3 2,306,051 3.24 mins 42.84 87.27% 10 MB
1 164,355 2.31 mins 82.45 86.67% 10 MB
4 82,179 2.26 mins 42.91 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



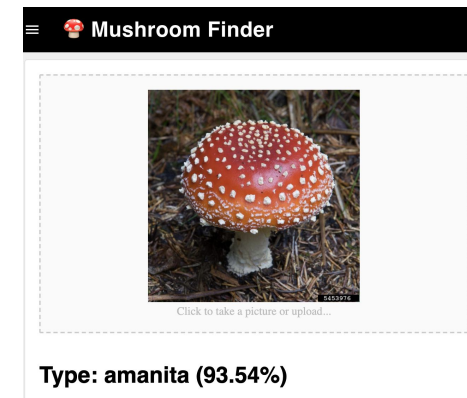
Rest API

Best Model

IDE / Code Editor



Deployment



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

Python: Colab provided env
OS: Linux

Python: pipenv
OS: Mac, Windows

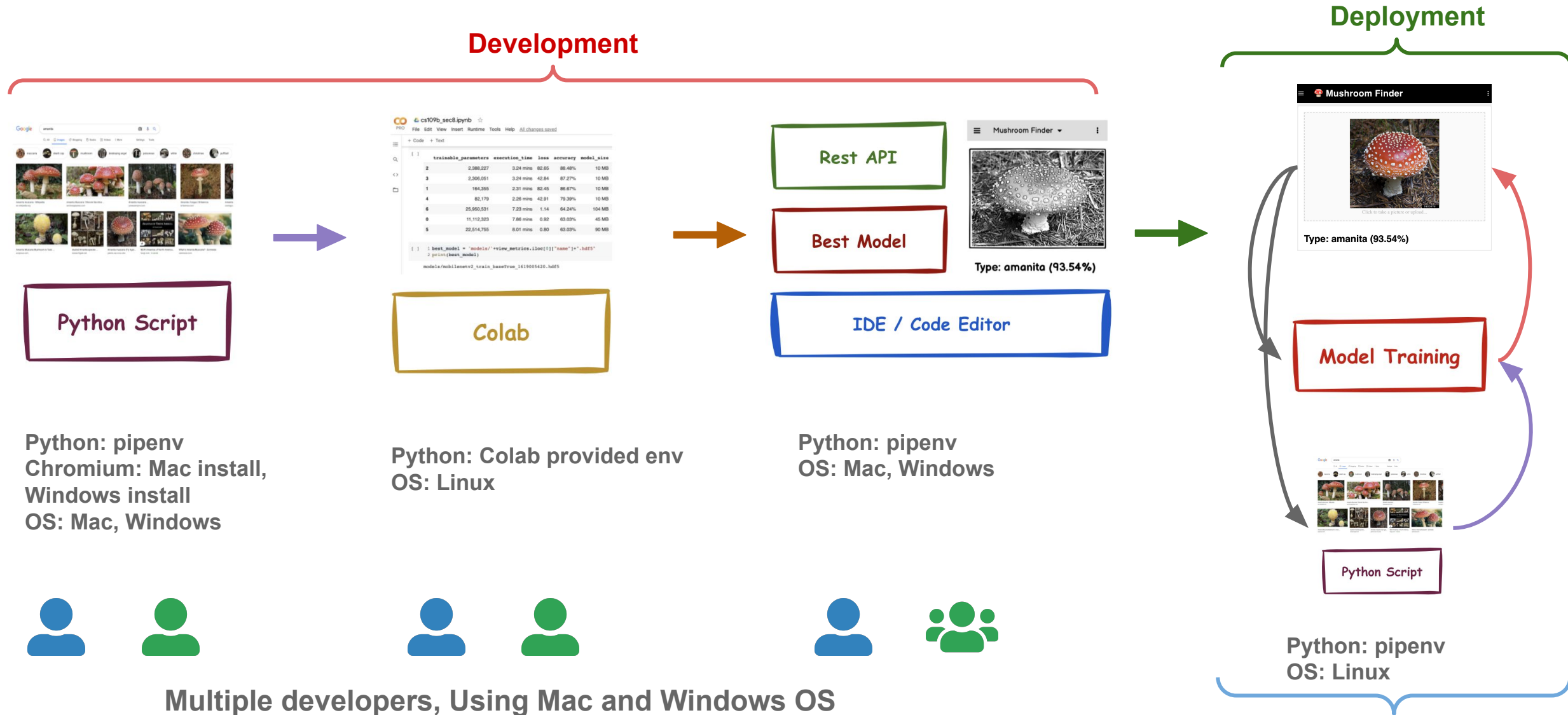
Python: pipenv
OS: Linux

Server



Multiple developers, Using Mac and Windows OS

Challenges - Multiple Developers + Automation



Challenges / Solutions

Challenges:

- Required Installations for Specific Operating Systems
- Guidelines for Code Collaboration
- Methods for Sharing Datasets and Models
- Automation of Data Gathering and Model Training
- Onboarding Procedures for New Team Members
- Resolving "It Works on My Machine" Issues `_(ツ)_/`

Solutions:

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

Tools

- Virtual Machines
- Virtual Environments
- Containers

Outline

1. Motivation
- 2. Virtual Machines**
3. Virtual Environments

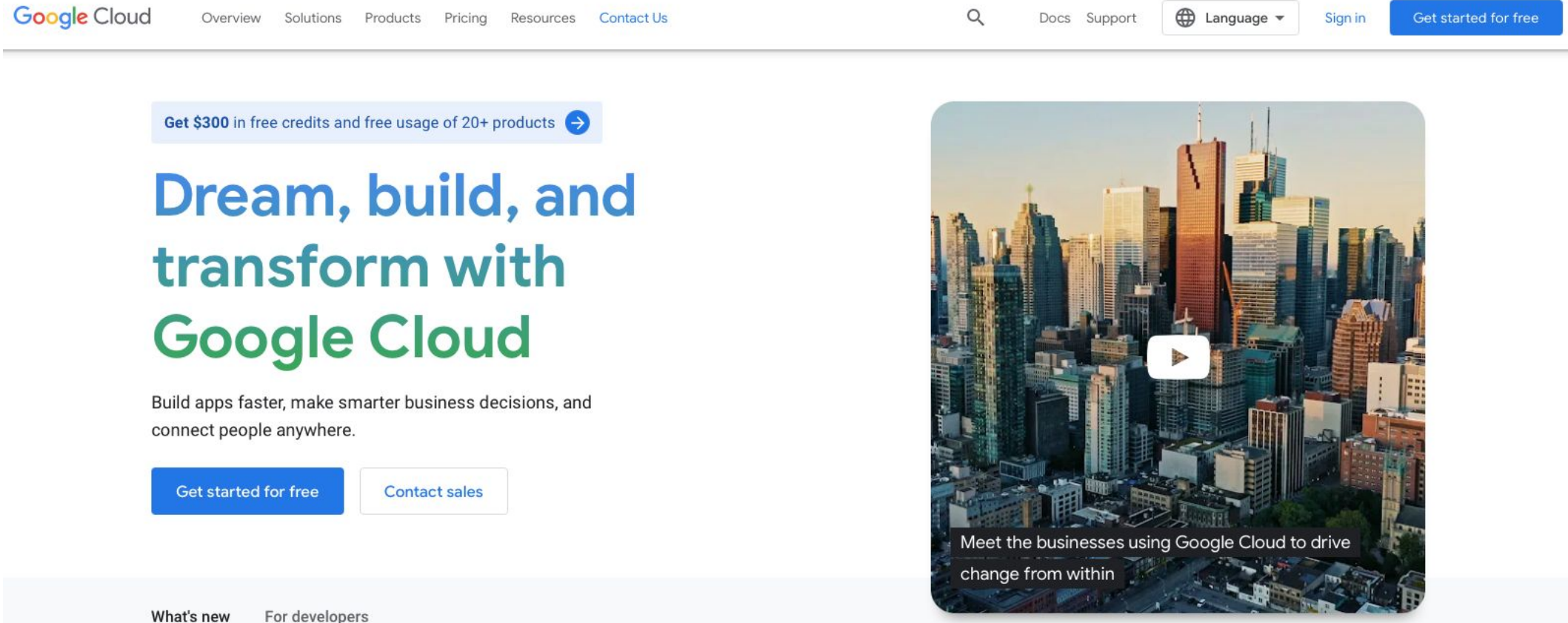
Running the Simple-Translate App on a **Virtual Machine**

To achieve this, follow the steps below:

- Create a Virtual Machine Instance.
- SSH into the Virtual Machine.
- Install Required Dependencies: git, Python.
- Download and Execute the Simple-Translate Python Script.
- For detailed instructions, please refer to the following link:
[Installing App on VM Manually](https://github.com/dlops-io/simple-translate#installing-app-on-vm-manually).
(<https://github.com/dlops-io/simple-translate#installing-app-on-vm-manually>)

Virtual Machines Tutorial

Google Cloud Platform: <https://cloud.google.com>



The screenshot shows the Google Cloud website homepage. At the top, there is a navigation bar with the Google Cloud logo on the left and links for Overview, Solutions, Products, Pricing, Resources, and Contact Us in the center. On the right side of the navigation bar, there is a search icon, links for Docs and Support, a Language dropdown menu, a Sign in link, and a blue button labeled "Get started for free".

Below the navigation bar, there is a promotional banner with the text "Get \$300 in free credits and free usage of 20+ products" followed by a right-pointing arrow icon. The main heading reads "Dream, build, and transform with Google Cloud", with "Dream, build, and transform with" in blue and "Google Cloud" in green. Below the heading, there is a sub-headline: "Build apps faster, make smarter business decisions, and connect people anywhere." At the bottom of this section, there are two buttons: a blue "Get started for free" button and a white "Contact sales" button with a blue border.

On the right side of the page, there is a large image of a city skyline at sunset. A white play button icon is overlaid on the image. At the bottom of the image, there is a text overlay that reads "Meet the businesses using Google Cloud to drive change from within".

At the bottom of the page, there is a light blue footer bar with the text "What's new" and "For developers" on the left side.

Virtual Machines Tutorial

Go to Navigation Menu

Google Cloud Platform | Preparing For Class | Search products and resources | 23 | CUSTOMIZE

Navigation menu | ACTIVITY | RECOMMENDATIONS

Google Cloud Next '21 is live. Join us now at 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 Tutorial

Select compute engine

The screenshot displays the Google Cloud console interface for the project 'ac215-project'. The left-hand navigation menu is expanded to show the 'VIRTUAL MACHINES' section, which includes 'VM instances', 'Instance templates', 'Sole-tenant nodes', and 'Machine images'. A blue arrow points from the text 'Select compute engine' to the 'VM instances' option. The main dashboard area shows a 'Welcome' message, project information (Project number: 129349313346, Project ID: ac215-project), and quick access buttons for 'Create a VM', 'Run a query in BigQuery', 'Create a GKE cluster', and 'Create a storage bucket'. Below this, there is a 'Quick access' section with buttons for 'APIs & Services', 'IAM & Admin', 'Billing', 'Compute Engine', 'Cloud Storage', 'BigQuery', 'VPC network', and 'Kubernetes Engine'. A 'View all products' button is also visible at the bottom of the quick access section.

Virtual Machines Tutorial

Select Virtual Machines

The screenshot shows the Google Cloud Platform interface for managing VM instances. The top navigation bar includes the Google Cloud Platform logo, a user profile dropdown, 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 categories like 'Virtual machines', 'Storage', 'Instance groups', and 'VM Manager'. The main content area features a filter input and a table with columns for 'Status', 'Name', 'Zone', 'Recommendations', 'In use by', 'Internal IP', 'External IP', and 'Connect'. Below the table is a large 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 Tutorial

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 has a blue header with the Google Cloud Platform logo and a search bar. The main content area is divided into a left sidebar and a main configuration area.

Left Sidebar:

- To create a VM instance, select one of the options:**
 - New VM instance** (selected): Create a single VM instance from scratch.
 - 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 Area:

- GENERAL-PURPOSE** (selected), COMPUTE-OPTIMIZED, MEMORY-OPTIMIZED, GPU
- Machine types for common workloads, optimized for cost and flexibility
- Series: E2
- CPU platform selection based on availability
- Machine type: e2-medium (2 vCPU, 4 GB memory)
- Resource summary:
 - vCPU: 1 shared core
 - Memory: 4 GB
- CPU PLATFORM AND GPU** (expanded)
- Display device**: Enable to use screen capturing and recording tools.
 - Enable display device
- Confidential VM service**: Enable the Confidential Computing service on this VM instance.
 - Enable the Confidential Computing service on this VM instance.
- Container**: Deploy a container image to this VM instance.
 -
- Boot disk**:
 - Disk type: New balanced persistent disk
 - Disk size: 10 GB
 - Image: Debian GNU/Linux 10 (buster)
 -
- Identity and API access**:
 - Service accounts: Compute Engine default service account
 - Access scopes:
 - Allow default access
 - Allow full access to all Cloud APIs
 - Set access for each API

Monthly estimate: \$25.46. That's about \$0.03 hourly. Pay for what you use: No upfront costs and per second billing.

[DETAILS](#)

Virtual Machines Tutorial

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

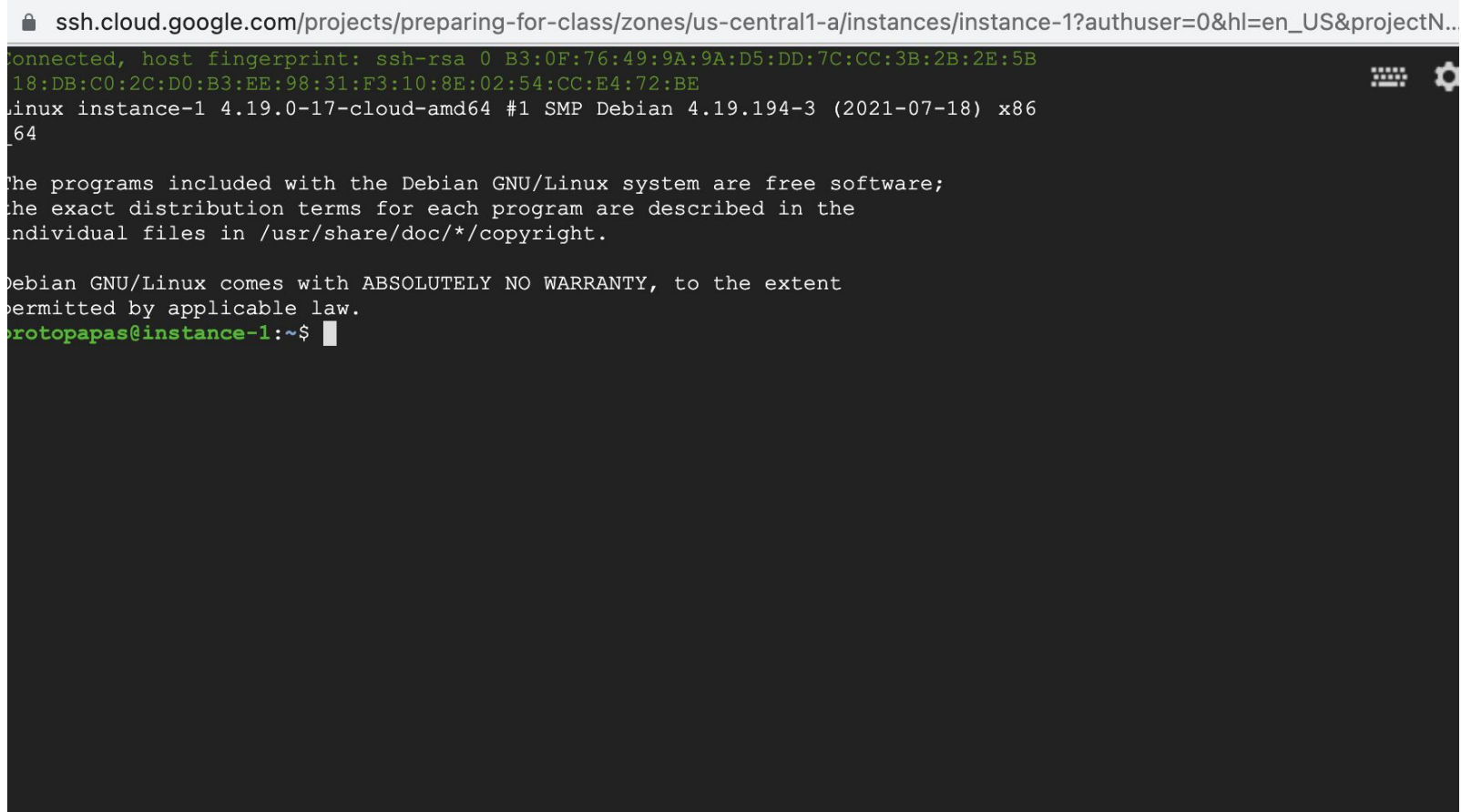
Related actions

- 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

DISMISS

Virtual Machines Tutorial

And here is your virtual machine

A screenshot of a terminal window showing an SSH connection to a Google Cloud instance. The terminal output includes the host fingerprint, the system information (Debian 4.19.194-3), and the user prompt.

```
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.  
protopapas@instance-1:~$
```

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

Why should we use virtual machines?

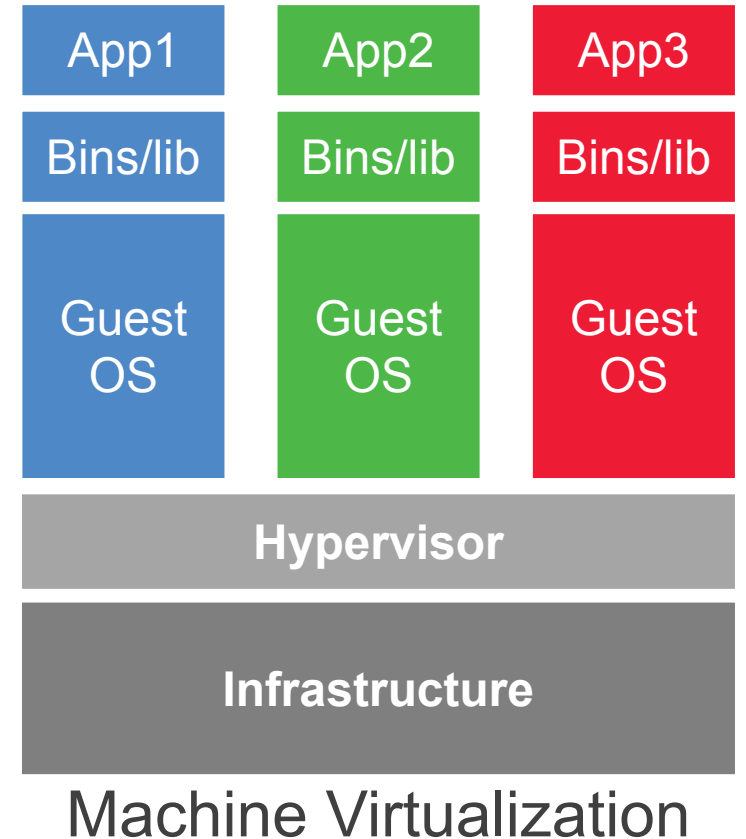
Motivation

- **Uniform Operating Environments:** Desire for a standardized OS across all team member workstations.
- **Consistent Software Configuration:** Requirement for identical software setups across the team.
- **Effortless Instance Management:** The need for simple procedures to instantiate and terminate VMs.

Virtual Machines!

Key Components of Virtual Machines & Hypervisors?

- Virtual machines **mimic** real hardware like CPUs and hard drives.
- **Hypervisors** manage these virtual machines on a server.
- **Unlimited** VMs can be run, subject to hardware limits.
- The main OS is the "**host**," and VMs run "**guest**" OS.
- Guest VMs can have different operating systems.



Why should we use virtual machines?

Advantages

- **Complete Autonomy:** it works like a separate computer system; it is like running a computer within a computer.
- **Enhance Security:** the software inside the virtual machine cannot affect the actual computer.
- **Cost-Effectiveness:** Purchase a single machine and run multiple operating systems.
- **Widely Adopted:** Utilized by all major cloud providers for on-demand server instances.

Software for Virtualization

- VirtualBox
- VMWare
- Parallels

Why should we use virtual machines?

Limitations

- **Local Hardware Dependency:** Relies on the hardware resources of the host machine.
- **Limited Portability:** Large file sizes can impede easy transfer or deployment.
- **Resource Overhead:** Additional computational and memory resources are required to operate.
- **Reduced Performance:** The guest system typically runs slower than the host environment.
- **Slow Initialization:** Extended startup times compared to native systems.
- **Graphics Constraints:** May lack the graphical capabilities of the host system.

Outline

1. Motivation
2. Virtual Machines
3. **Virtual Environments**

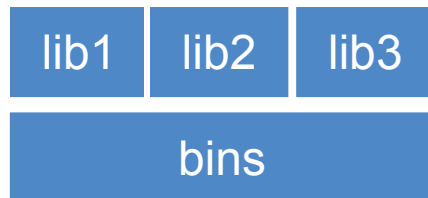
What are virtual environments

A virtual environment is an isolated Python setting in which the interpreter can execute libraries and scripts independently of other virtual environments.

- Consider a virtual environment as a **directory** containing the following **components**:
 - *`site_packages/`*: A directory where third-party libraries are installed.
 - *Symlinks*: Links to system executables.
 - *Scripts*: These ensure that the code utilizes the interpreter and site packages specific to the virtual environment.

Why should we use virtual environment?

Maggie took CS109B and used to run her Jupyter notebooks from the Anaconda prompt. Whenever she installed a module, it was placed in one of the following folders: `bin`, `lib`, `share`, or `include`. She could then import the module and used it without any issue.



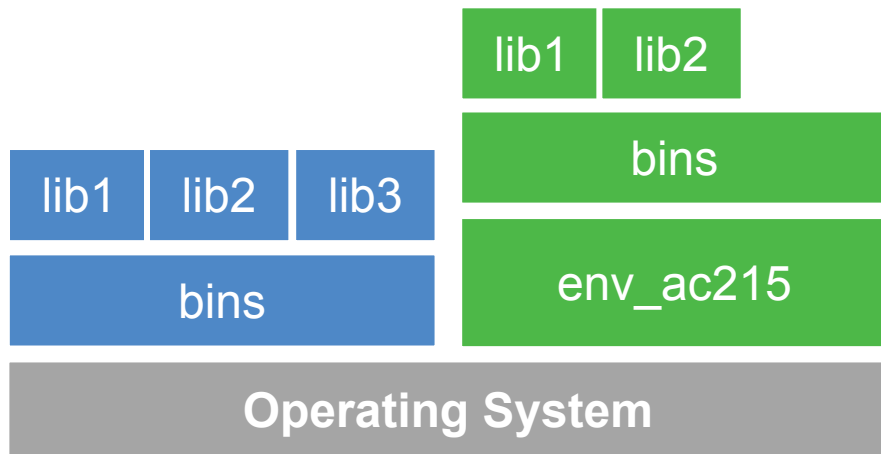
Operating System

Maggie

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

Why should we use virtual environment?

Maggie begins taking AC215 and decides that **isolating** the new coding environment from previous ones would be beneficial to avoid package conflicts. To achieve this, she employs a layer of **abstraction** known as a virtual environment. This helps her keep modules organized and prevents issues while developing new projects.

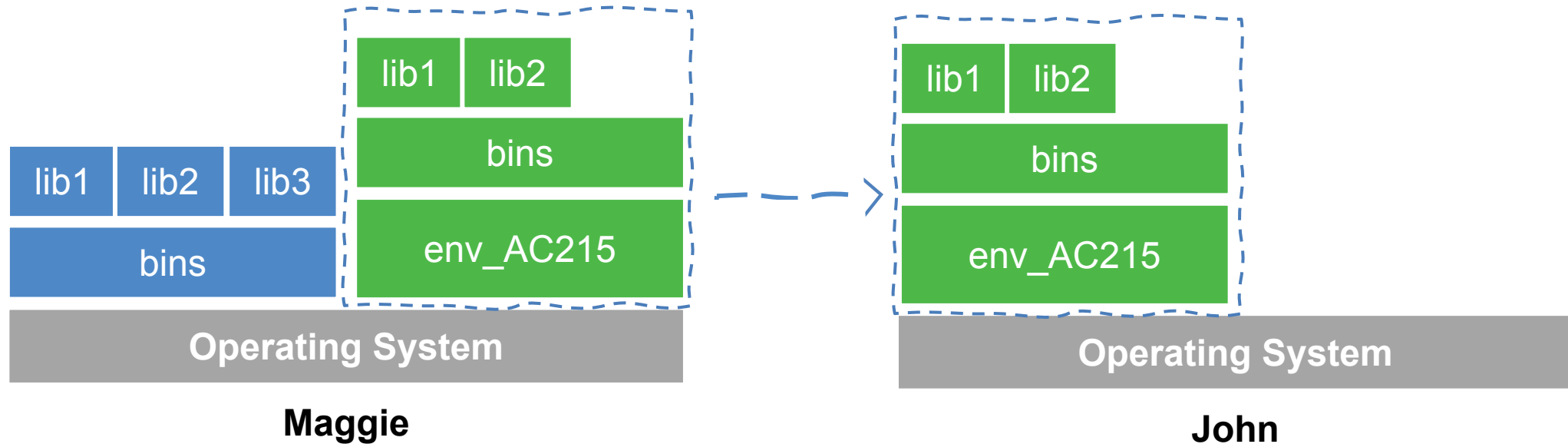


Maggie

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

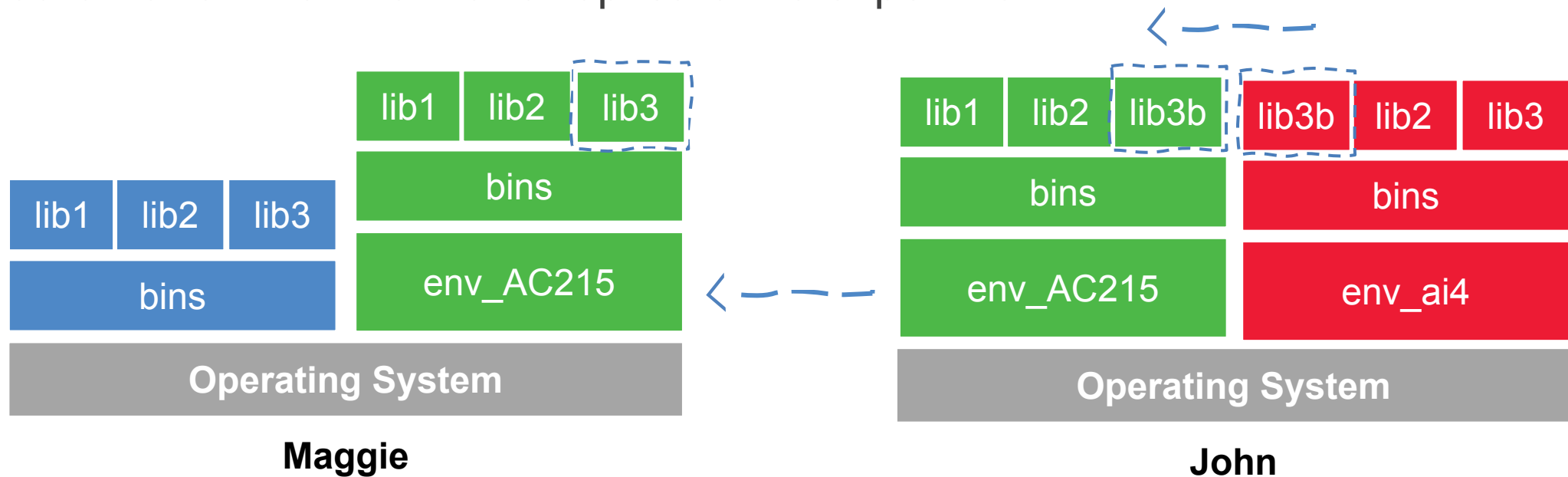
Why should we use virtual environment?

For the final project, Maggie collaborates with John and shares her working environment by distributing a `.yml` file for the Conda environment.



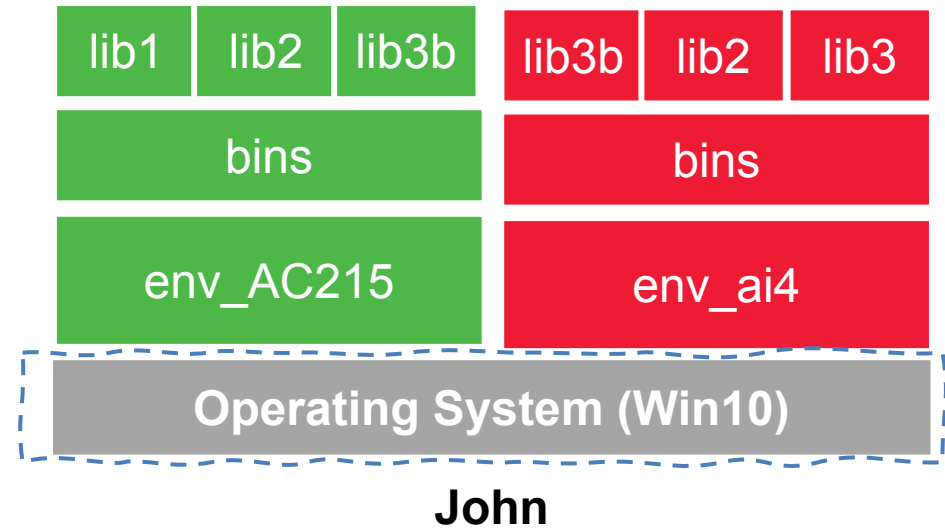
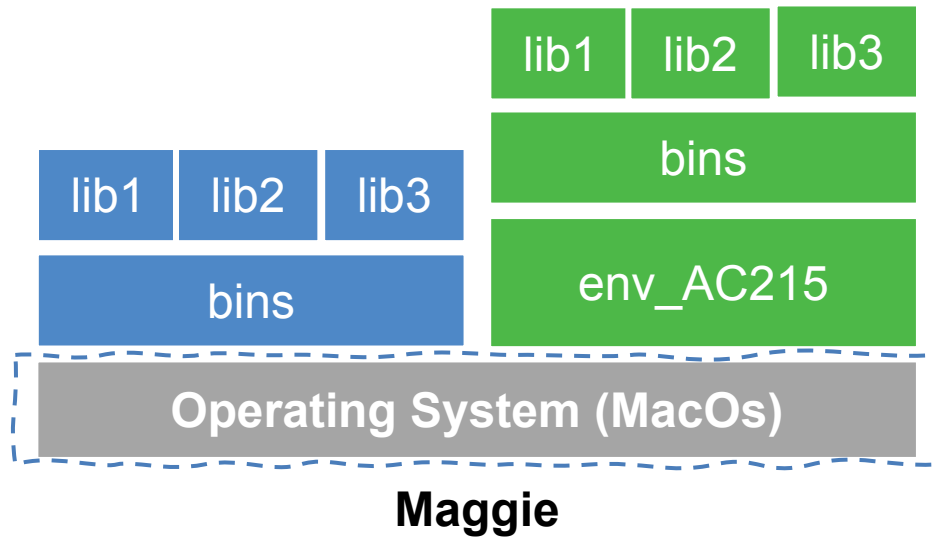
Why should we use virtual environment?

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.



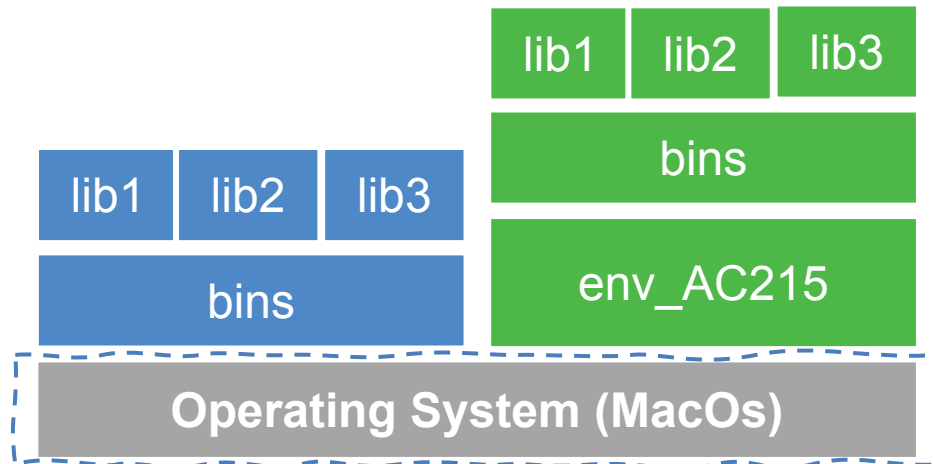
Why should we use virtual environment?

- What could go wrong?

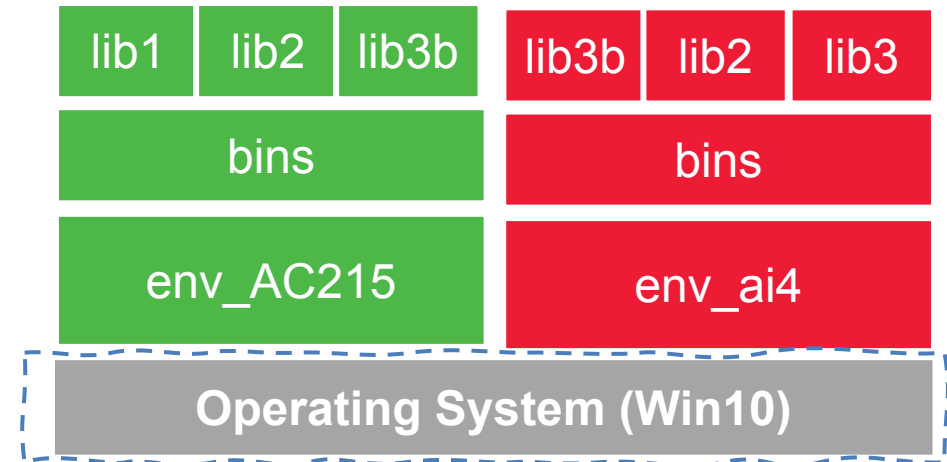


Why should we use virtual environment?

- *What could go wrong?*
- Unfortunately, Maggie and John are getting different results, which they suspect is due to their differing operating systems. Specifically, Maggie is using macOS, while John is on Windows 10.



Maggie



John

Why should we use virtual environment?

- **Streamlines** code development and usage.
- Isolates dependencies in separate "**sandboxes**" for easy switching between applications.
- Given an operating system and hardware, we can get the exact code environment set up using **different technologies**.

Virtual environments

Pros

- **Reproducible Research:** Enables consistent and replicable outcomes.
- **Explicit Dependencies:** Clearly defines all required software and packages.
- **Enhanced Engineering Collaboration:** Streamlines teamwork by standardizing environments.

Cons

- **Setup Challenges:** Initial environment configuration can be complex.
- **Lack of Isolation:** Does not completely isolate the working environment.
- **OS Compatibility Issues:** May not function consistently across different operating systems.

Creating Virtual Environments

- **virtualenv (python2) / venv (python3)**

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

- **mamba**

Fast (C++) replacement for the Conda package manager that aims to offer quicker dependency resolution and installation - **must do HW0 of CS109A**

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

How to use it:

- create an environment within your project folder `python3 -m venv your_env_name`
- it will add a folder called `environment_name` in your project directory
- activate environment: `source your_env_name/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
- Base virtual environment 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

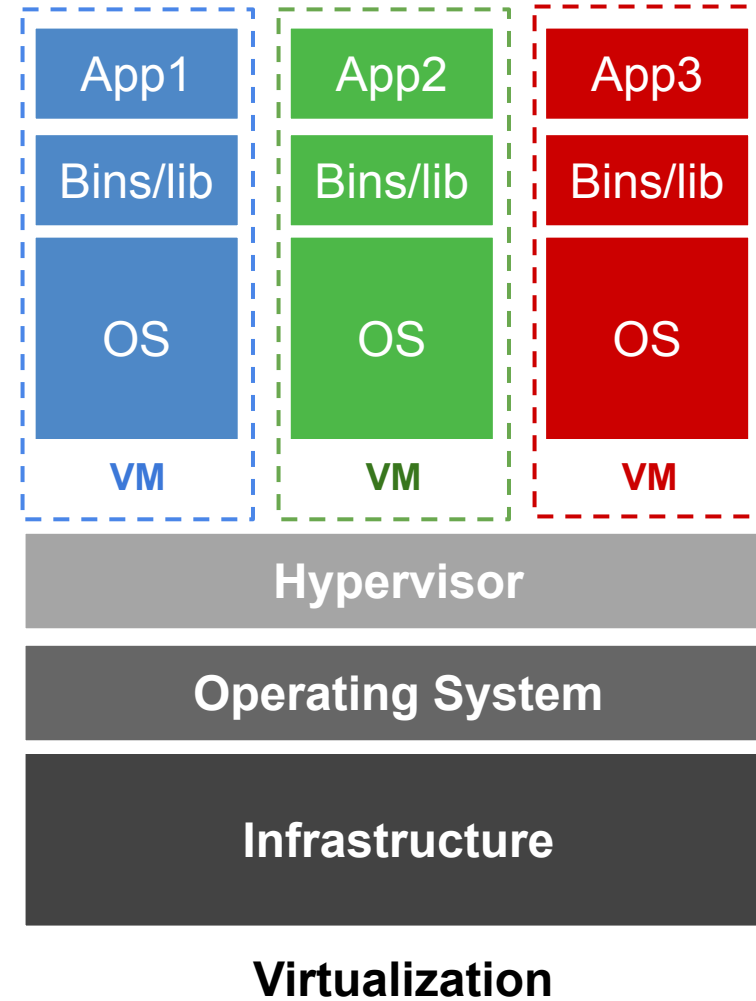
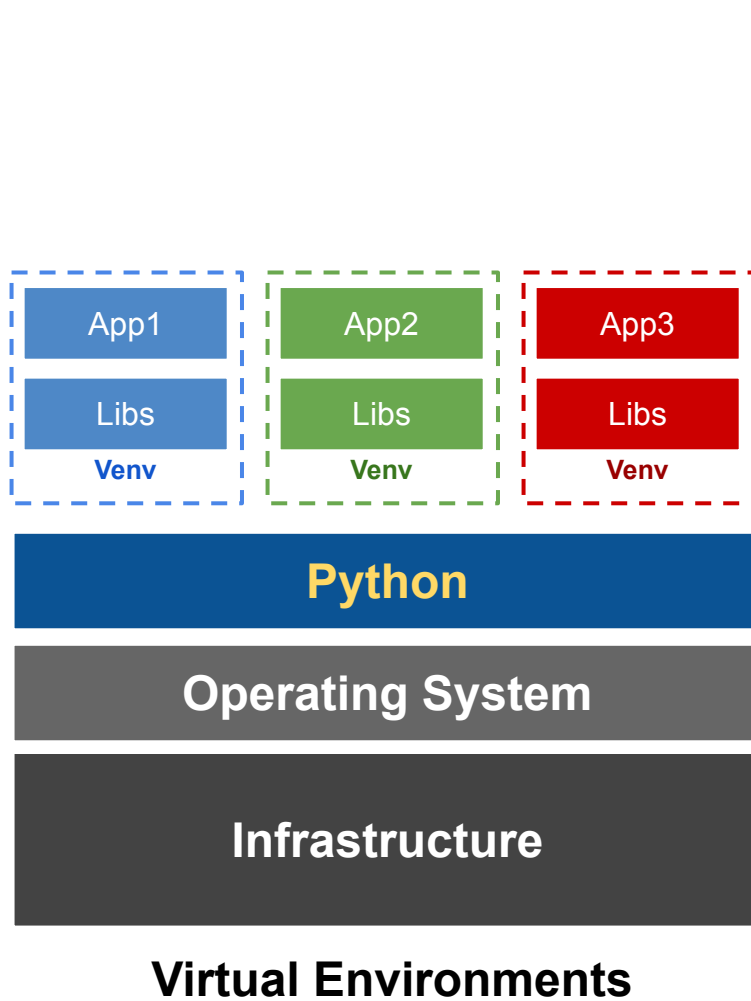
<https://jakevdp.github.io/blog/2016/08/25/conda-myths-and-misconceptions/>

- 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



Virtual Environment Tutorial

- Let us run the simple-translate app using Virtual Environment
- For this we will do the following:
 - Create a VM Instance
 - SSH into the VM
 - Install dependencies: git, python
 - Download and run the simple-translate python script
- Full instructions can be found [here](#)

THANK YOU