

# Lecture 2: Virtual Machines & Virtual Environments

AC215

Pavlos Protopapas  
SEAS/Harvard



# Outline

---

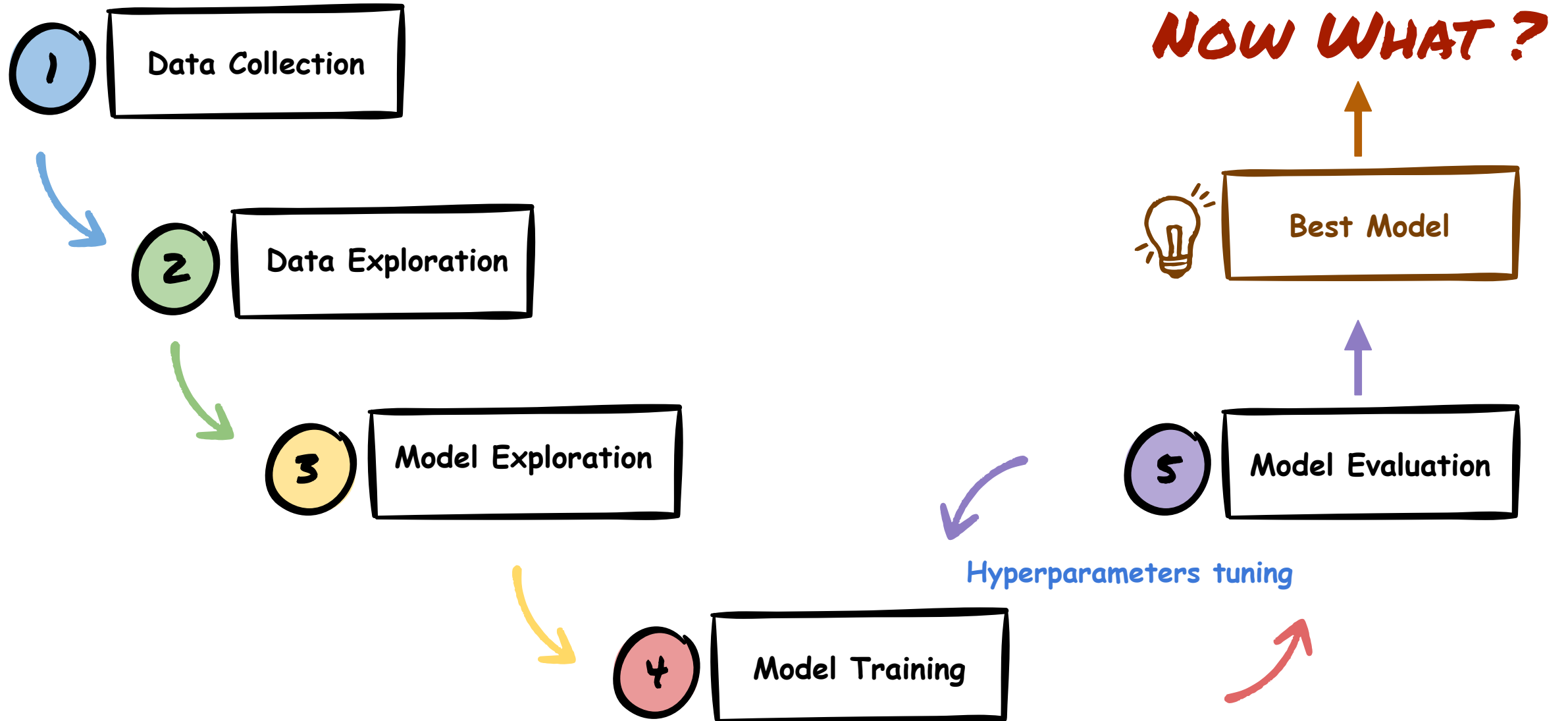
1. Motivation
2. Virtual Machines
3. Virtual Environments

# Outline

---

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

# Motivation: Deep Learning Flow



# We want to build a 🧀 Cheese App

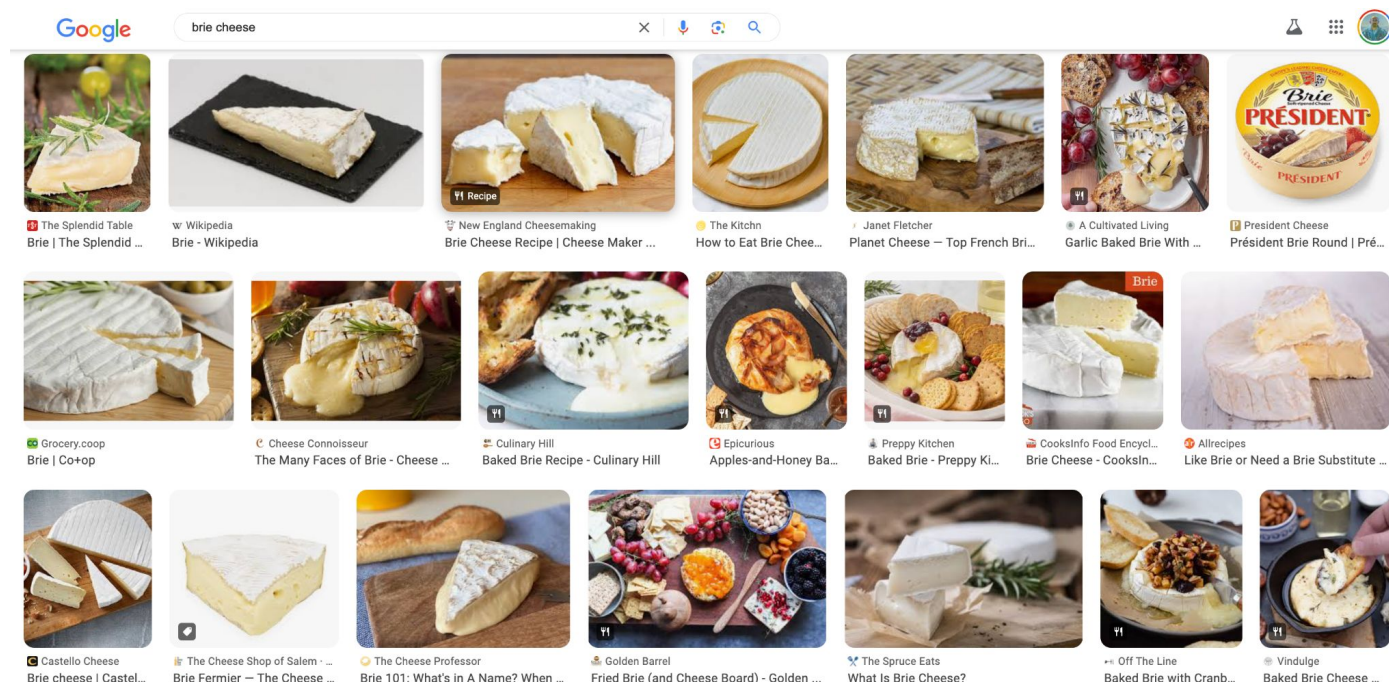
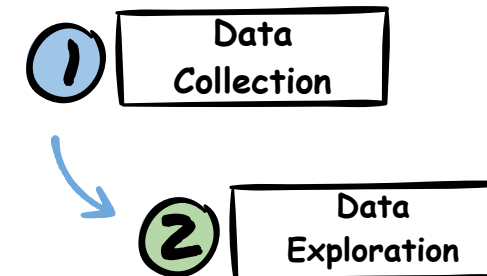
- Pavlos likes cheeses and when he throws parties he always have cheese for his guests.
- He wants to build an app to identify cheese types and learn what cheeses go with each other, how to pair it with wines, the way this particular cheese is made, a history of the cheese etc
- Project Summary





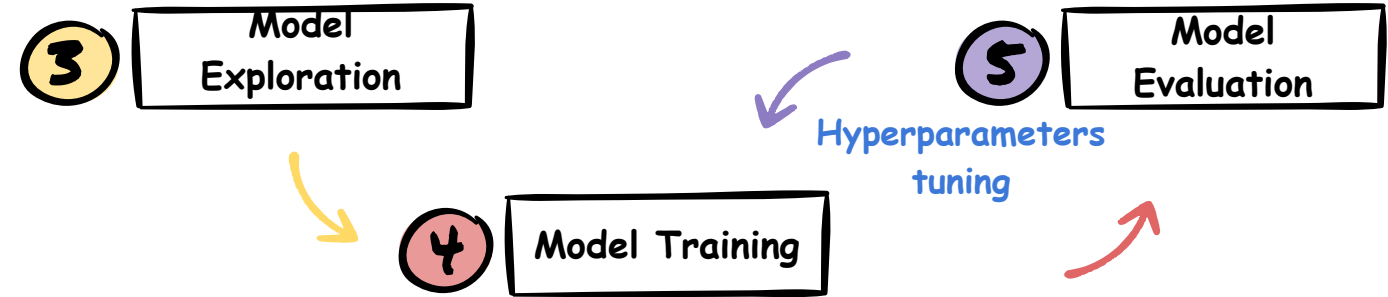
# Cheese App: Data

- Collect images from Google/Bing
- For our app we downloaded images for cheeses **brie**, **parmigiano**, **gouda**, **gruyere**
- Images are organized into 4 labels





# Cheese App: Models



- Identify our problem task
- Try various model architectures
- Transfer learning
- Hyperparameter tuning
- Experiment tracking

trainable_parameters	execution_time	loss	accuracy
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%

# Cheese App: Choose the Best Model

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	tfhub_mobilenetv2_train_base_True
82,179	3.19 mins	42.79	90.30%	10 MB	0.001	32	10	SGD	tfhub_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

Selection is based on multiple criteria, including accuracy, memory usage, execution time, and other key performance metrics

**NOW WHAT?**



# Steps to build an App?

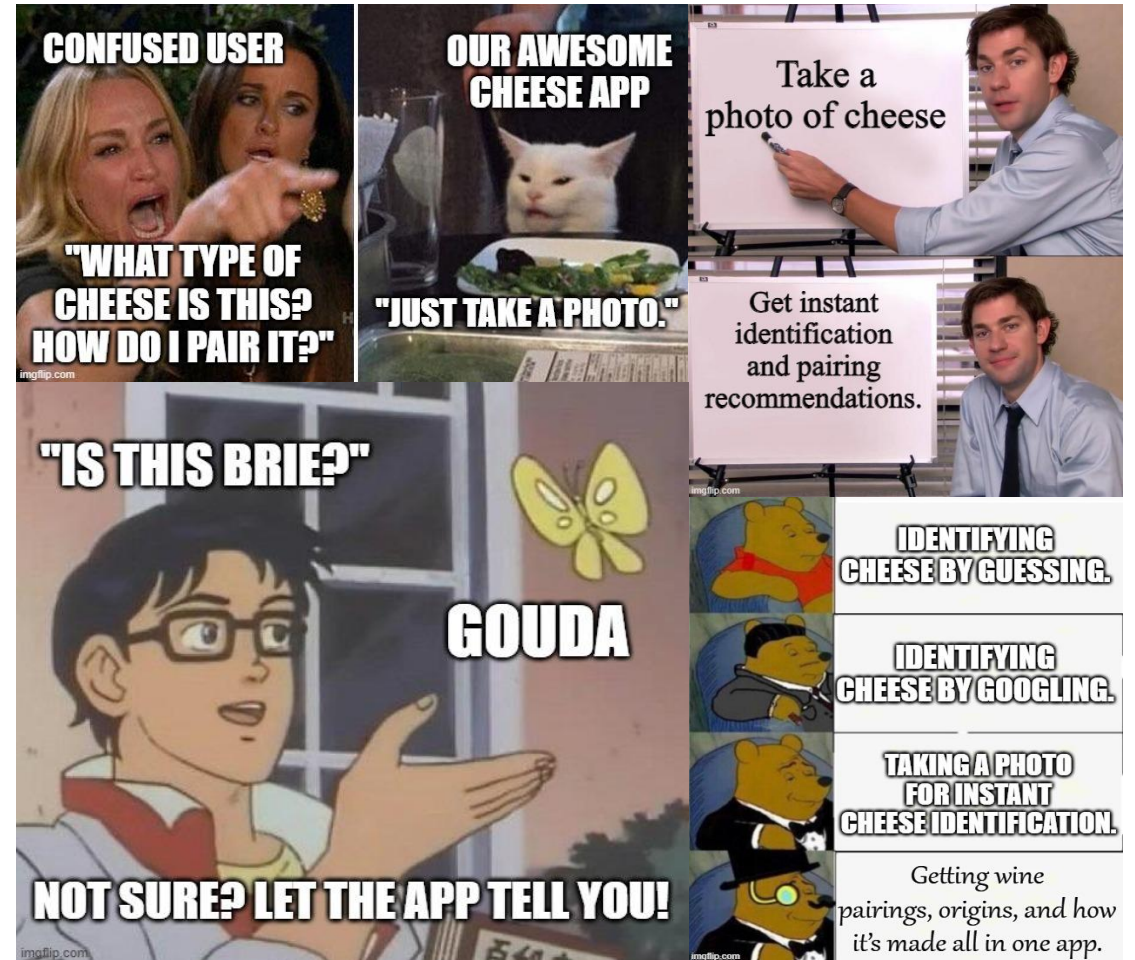
→ *NOW WHAT ?*

- Expose best model as an API
- Build a **frontend**
- **Integrate** model prediction API into the app
- **Deploy** app to a cloud provider
- Go live: <http://formaggio.me>

# Cheese App: The App

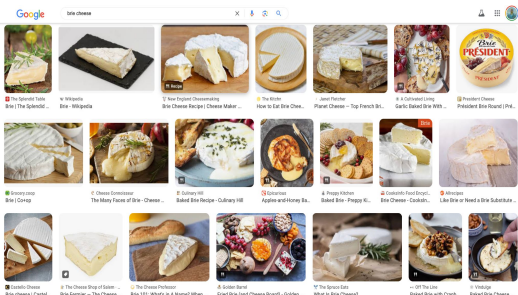
- We want to build an app to take a photo of a cheese and it helps us identify the type of cheese
- We also want the app to respond to prompts like “Where is Brie made?”, “What wine should I pair with Brie?”, “How is it made?”, and “Where can I buy it?”.

## Technical Specifications



# How do we build an App?

Data Collection



Python Script



01\_tutorial\_mushroom\_classification\_models.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

	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

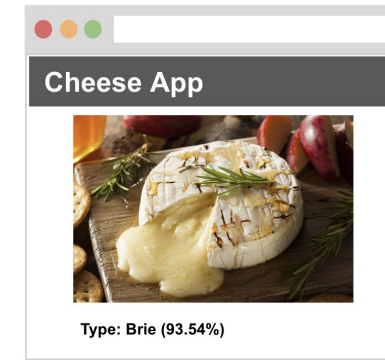


Rest API

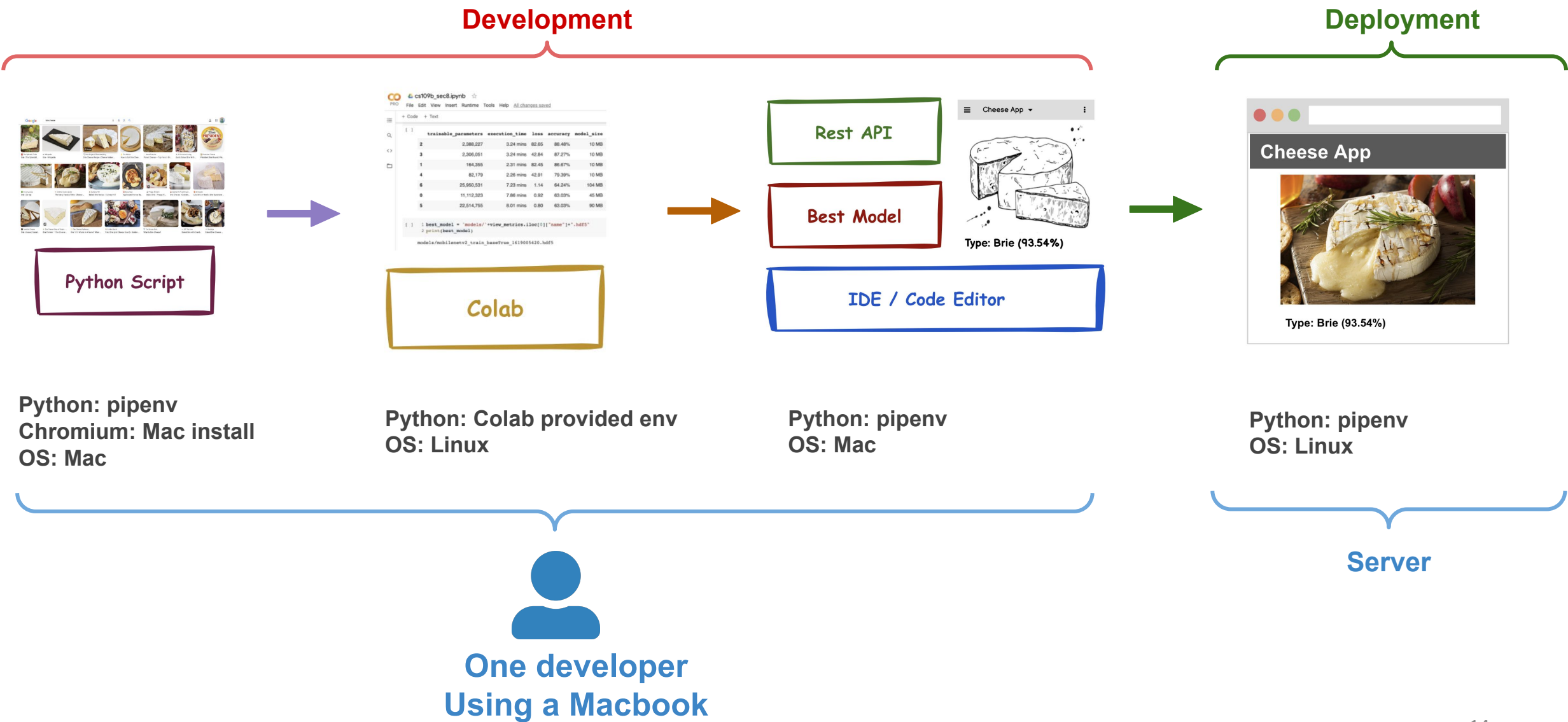
Best Model

IDE / Code Editor

Data Exploration  
Model Exploration  
Model Training  
Model Evaluation

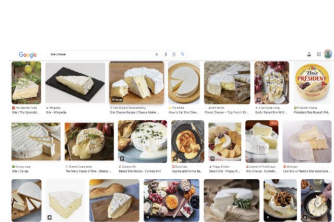


# Challenges



# Challenges - Multiple Developers

## Development



Python Script



	trainable_parameters	execution_time	loss	accuracy	model_size
2	2,388,227	3.24 min	82.85	88.48%	10 MB
3	2,306,051	3.24 min	42.84	87.27%	10 MB
1	164,355	2.31 min	82.45	86.67%	10 MB
4	82,179	2.26 min	42.91	79.39%	10 MB
6	25,950,531	7.23 min	1.14	64.24%	104 MB
0	11,112,323	7.86 min	0.82	63.03%	45 MB
5	22,514,755	8.01 min	0.80	63.03%	90 MB

Colab



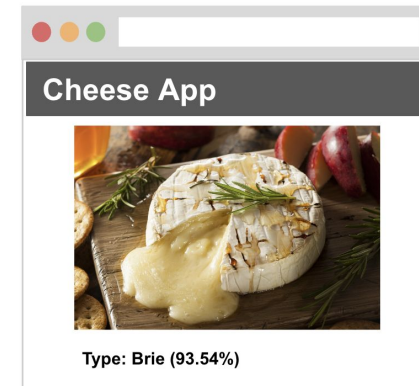
Rest API

Best Model

IDE / Code Editor



## Deployment



Python: pipenv  
Chromium: Mac install  
OS: Mac

Python: Colab provided env  
OS: Linux

Python: pipenv  
OS: Mac

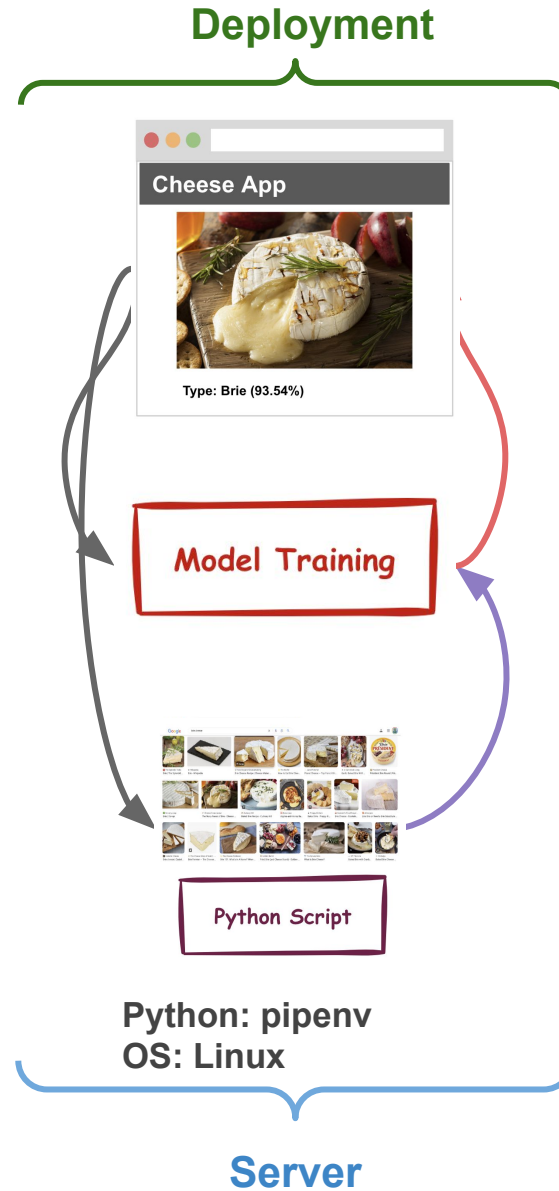
Python: pipenv  
OS: Linux

Server



Multiple developers, Using Mac and Windows OS

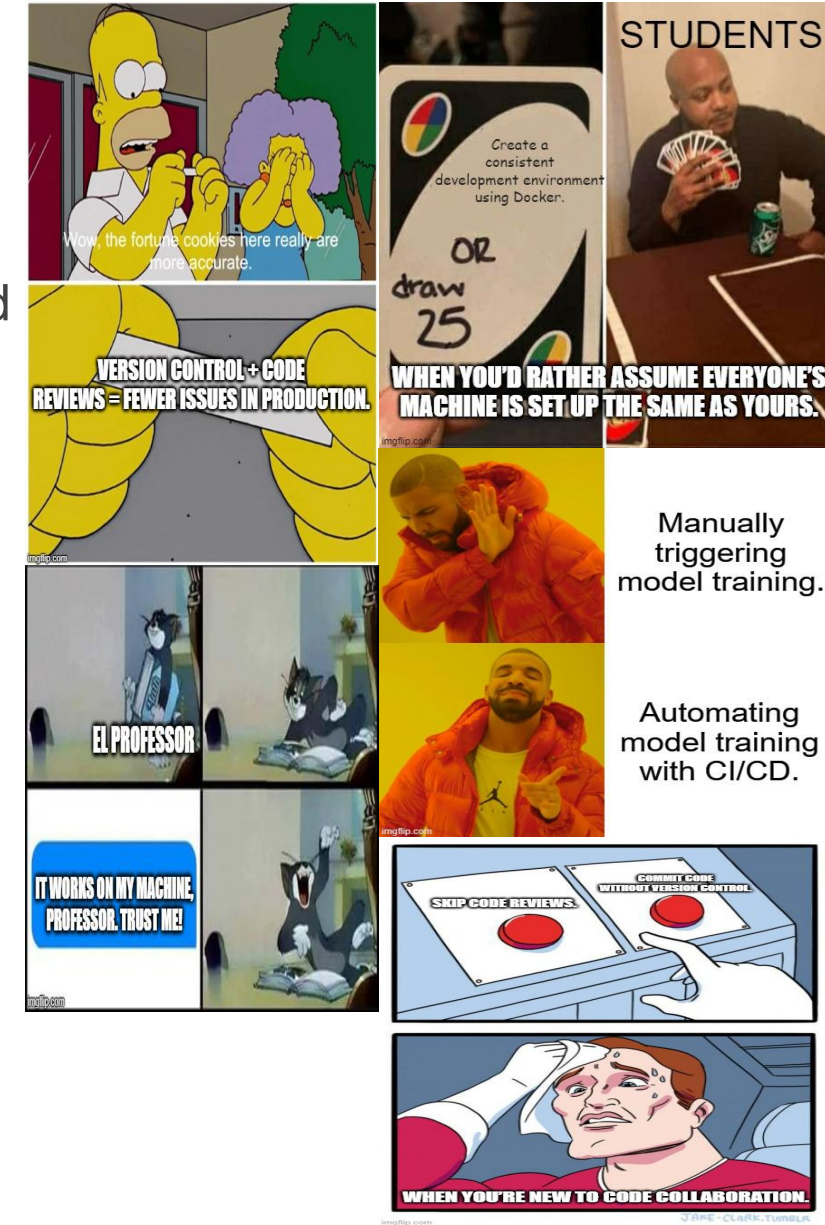
# Challenges - Multiple Developers + Automation





# Challenges / Solutions

- Onboarding Procedures for New Team Members:
- Required Installations for Specific Operating Systems:  
All team members have the necessary software and tools installed on their systems.
- Guidelines for Code Collaboration:  
Version control and code reviews.
- Methods for Sharing Datasets and Models
- Automation of Data Gathering and Model Training
- Resolving “It Works on My Machine” Issues



## **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

# Tools

---

- Virtual Machines
- Virtual Environments
- Containers



We will cover these next week

# Tools

---

- **Virtual Machines**
- Virtual Environments
- Containers

# Motivation

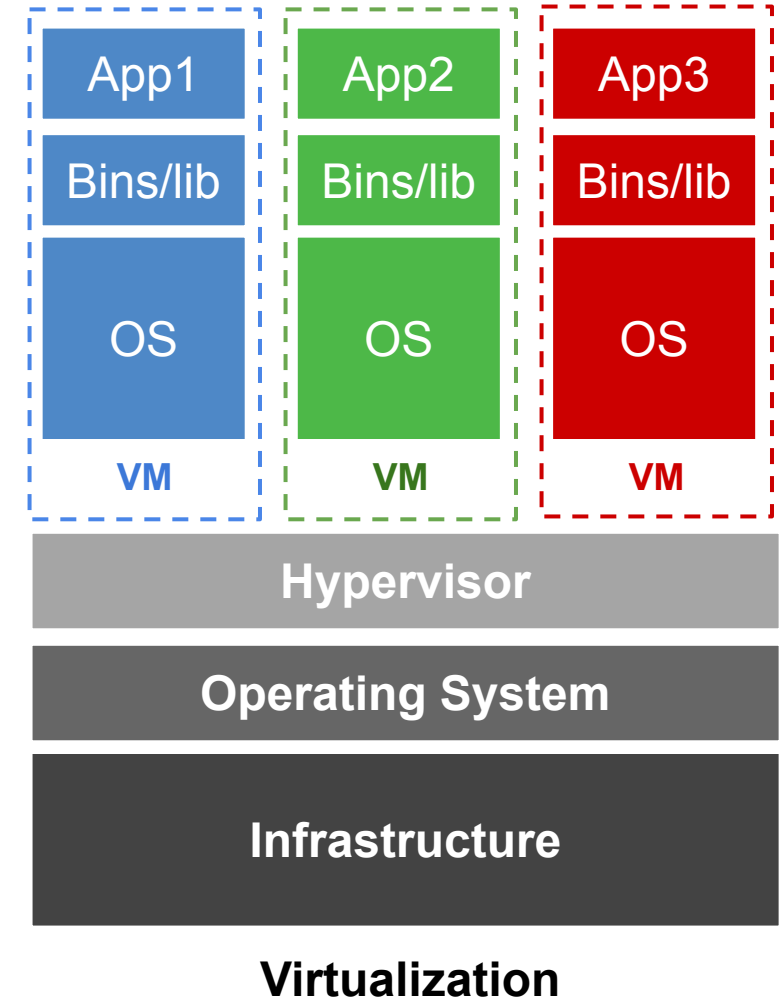
---

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

## Virtual Machines!

# What is a Virtual Machine?

- Virtual machines **mimic** real hardware like CPUs and hard drives.
- **Hypervisor** is a software that manages (creates and runs) virtual machines on a physical machine.
- **Unlimited** VMs can be run, subject to hardware limits.
- The main OS on the physical machine is called 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 shouldn't 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.

# Virtual Machines Tutorial (T1)

---

France, Italy, Switzerland, Spain, and the Netherlands are renowned for their rich traditions and diverse varieties of cheese. Given the global nature of these cheese-producing countries, our app will need to support multiple languages to cater to a wide audience.

Our initial application will be a straightforward translation tool, utilizing Python to integrate the Google Translate library. This will serve as a foundation for future examples, where we will explore different tools for implementation and deployment.



# Virtual Machines Tutorial (T1)

---

## 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 (T1)

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

Google Cloud

Overview

Solutions

Products

Pricing

Resources

Contact Us



Docs

Support



Console



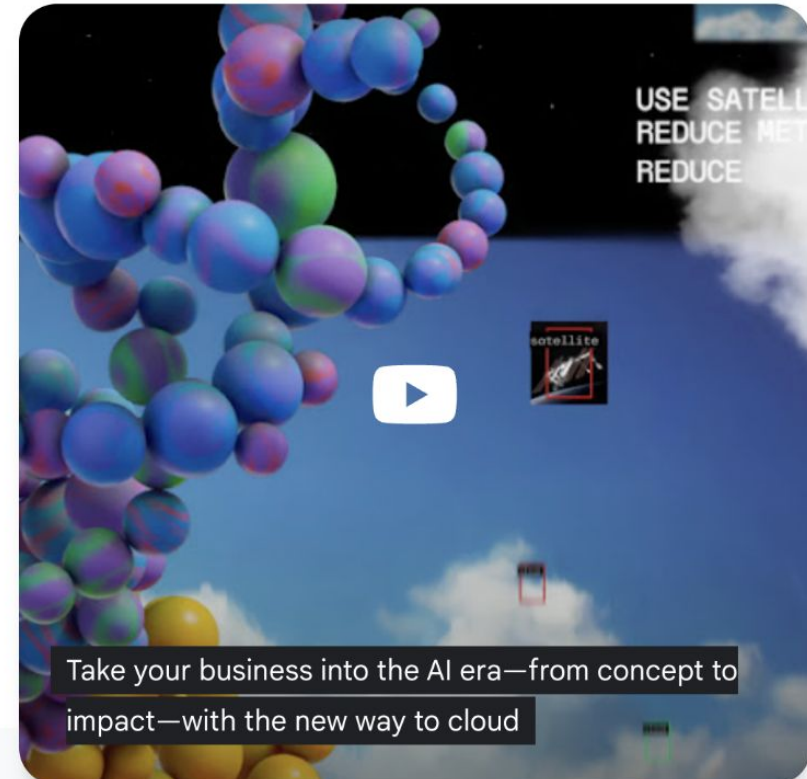
Try Gemini 1.5 models to build with up to a 2M token context window →

## Build what's next in generative AI

See what you can build with up to a 2M token context window using our newest and most advanced Gemini 1.5 models.

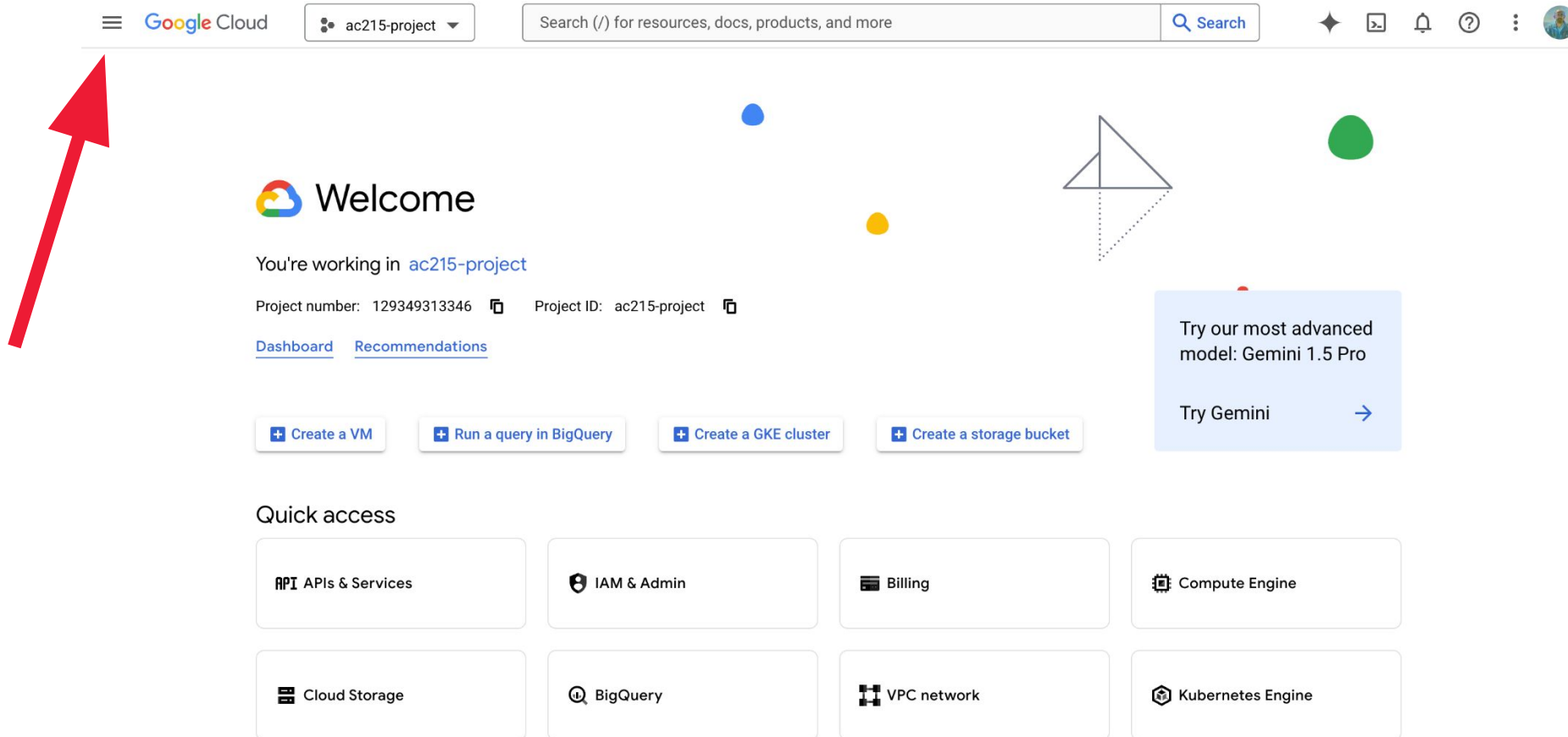
Try it in Vertex AI

Contact sales



# Virtual Machines Tutorial (T1)

## Go to Navigation Menu



The screenshot shows the Google Cloud console dashboard for the 'ac215-project'. A red arrow points to the navigation menu icon (three horizontal lines) in the top left corner. The dashboard includes a search bar, project information, a 'Welcome' message, project details, quick access links, and a 'Quick access' section with various service tiles.

Google Cloud ac215-project Search (/) for resources, docs, products, and more Search

Welcome

You're working in [ac215-project](#)

Project number: 129349313346 Project ID: ac215-project

[Dashboard](#) [Recommendations](#)

[+ Create a VM](#) [+ Run a query in BigQuery](#) [+ Create a GKE cluster](#) [+ Create a storage bucket](#)

Try our most advanced model: Gemini 1.5 Pro

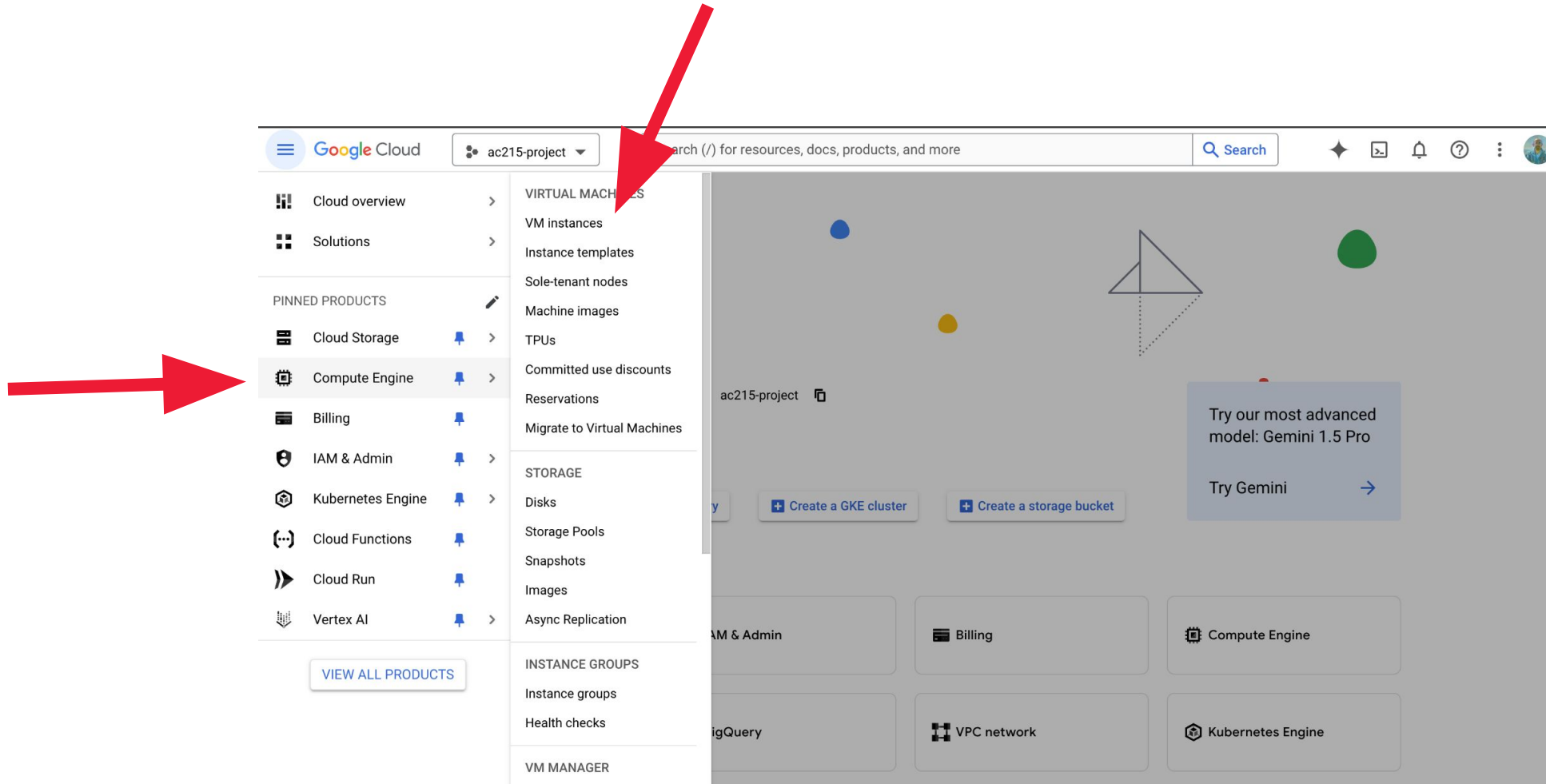
Try Gemini →

Quick access

APIs & Services	IAM & Admin	Billing	Compute Engine
Cloud Storage	BigQuery	VPC network	Kubernetes Engine

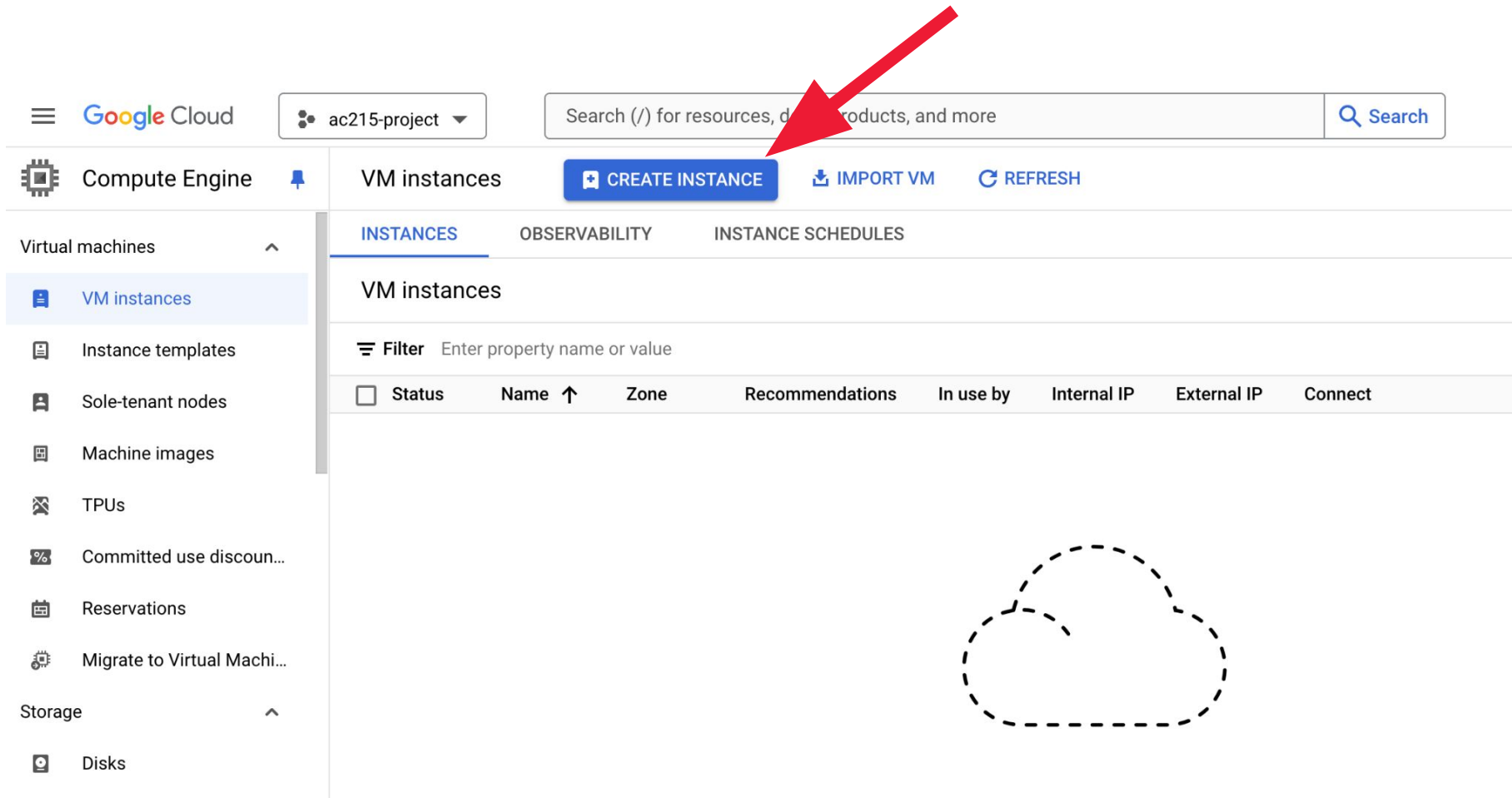
# Virtual Machines Tutorial (T1)

Select compute engine and VM instances



# Virtual Machines Tutorial (T1)

You should see this (or similar). Click on CREATE INSTANCE



The screenshot shows the Google Cloud Console interface for the 'ac215-project'. The left sidebar contains a navigation menu with categories like 'Compute Engine' and 'Storage'. Under 'Compute Engine', 'VM instances' is selected. The main content area shows the 'VM instances' page with tabs for 'INSTANCES', 'OBSERVABILITY', and 'INSTANCE SCHEDULES'. A red arrow points to the 'CREATE INSTANCE' button. Below the tabs, there is a filter bar and a table header with columns: Status, Name, Zone, Recommendations, In use by, Internal IP, External IP, and Connect. A dashed cloud icon is drawn in the bottom right corner of the console area.

Google Cloud

ac215-project

Search (/) for resources, documentation, and more

Search

Compute Engine

VM instances

CREATE INSTANCE

IMPORT VM

REFRESH

INSTANCES

OBSERVABILITY

INSTANCE SCHEDULES

VM instances

Filter Enter property name or value

Status	Name	Zone	Recommendations	In use by	Internal IP	External IP	Connect
--------	------	------	-----------------	-----------	-------------	-------------	---------

Storage

Disks

# Virtual Machines Tutorial (T1)

Select all defaults

Google Cloud Platform

Preparing For Class

Search products and resources

Create an instance

To create a VM instance, select one of the options:

New VM instance

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

GENERAL-PURPOSE

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)

vCPU

1 shared core

Memory

4 GB

CPU PLATFORM AND GPU

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.

Container

Deploy a container image to this VM instance

DEPLOY CONTAINER

Boot disk

Disk type

New balanced persistent disk

Disk size

10 GB

Image

Debian GNU/Linux 10 (buster)

CHANGE

Identity and API access

Service accounts

Service account

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

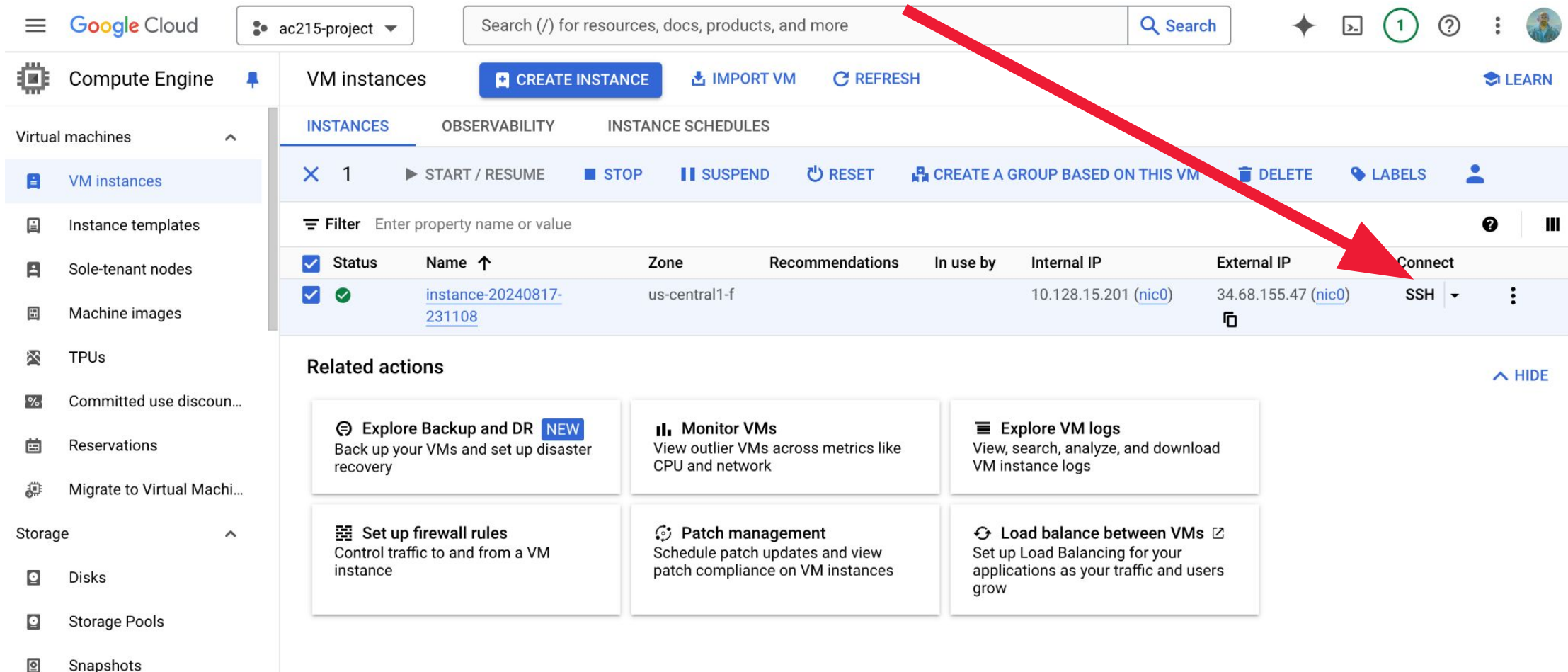
Pay for what you use: No upfront costs and per second billing

DETAILS

32

# Virtual Machines Tutorial (T1)

Wait for instance to start and click on ssh



The screenshot shows the Google Cloud console interface for VM instances. A red arrow points from the top search bar to the 'SSH' button in the 'Connect' column of the instance table.

**Google Cloud** ac215-project Search (/) for resources, docs, products, and more

**Compute Engine** VM instances [CREATE INSTANCE](#) [IMPORT VM](#) [REFRESH](#) [LEARN](#)

**Virtual machines**

- VM instances
- Instance templates
- Sole-tenant nodes
- Machine images
- TPUs
- Committed use discount
- Reservations
- Migrate to Virtual Machine

**INSTANCES** **OBSERVABILITY** **INSTANCE SCHEDULES**

1 [START / RESUME](#) [STOP](#) [SUSPEND](#) [RESET](#) [CREATE A GROUP BASED ON THIS VM](#) [DELETE](#) [LABELS](#) [?](#) [⋮](#)

**Filter** Enter property name or value

<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Status	Name ↑	Zone	Recommendations	In use by	Internal IP	External IP	Connect	<a href="#">?</a>	<a href="#">⋮</a>
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		<a href="#">instance-20240817-231108</a>	us-central1-f			10.128.15.201 ( <a href="#">nic0</a> )	34.68.155.47 ( <a href="#">nic0</a> )	SSH	<a href="#">?</a>	<a href="#">⋮</a>

**Related actions** [HIDE](#)

- [Explore Backup and DR](#) **NEW**  
Back up your VMs and set up disaster recovery
- [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
- [Load balance between VMs](#) [ⓘ](#)  
Set up Load Balancing for your applications as your traffic and users grow

# Virtual Machines Tutorial (T1)

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

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



# Tools

---

- Virtual Machines
- **Virtual Environments**
- Containers

# What are virtual environments

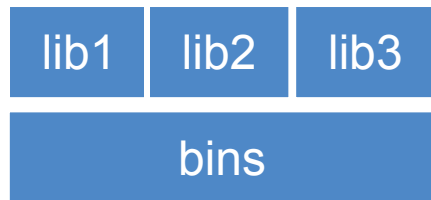
---

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 their Jupyter notebooks from the Anaconda prompt. Whenever they installed a module, it was placed in one of the following folders: `bin`, `lib`, `share`, or `include`. They could then import the module and used it without any issue.



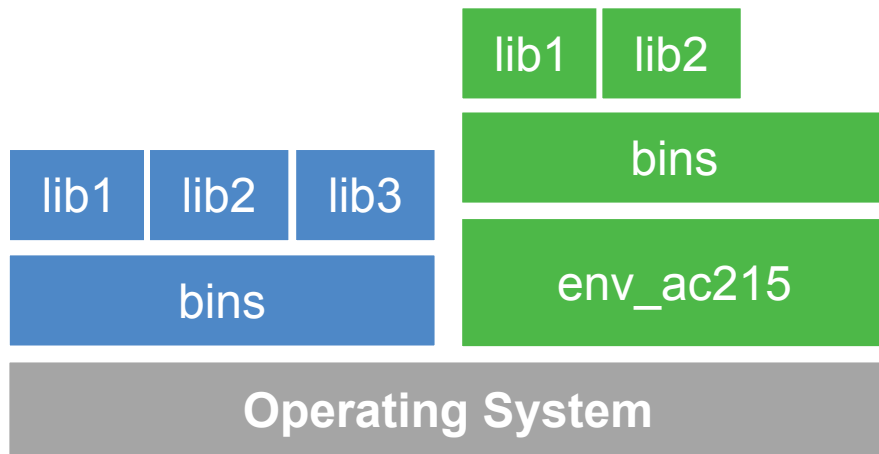
Operating System

**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, they employ a layer of **abstraction** known as a virtual environment. This helps them keep modules organized and prevents issues while developing new projects.

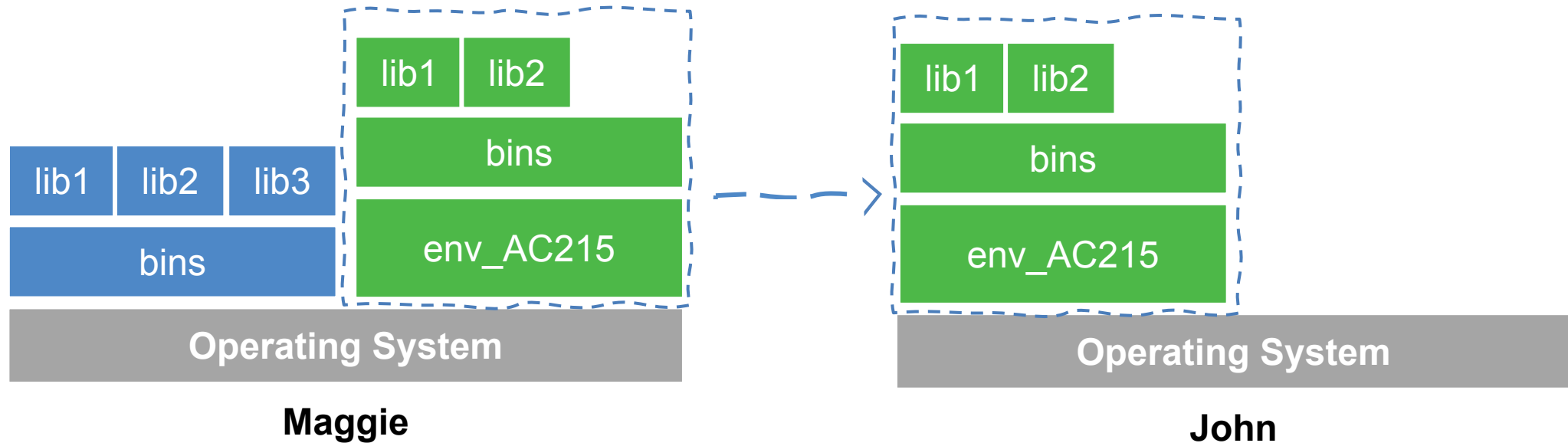


**Maggie**

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

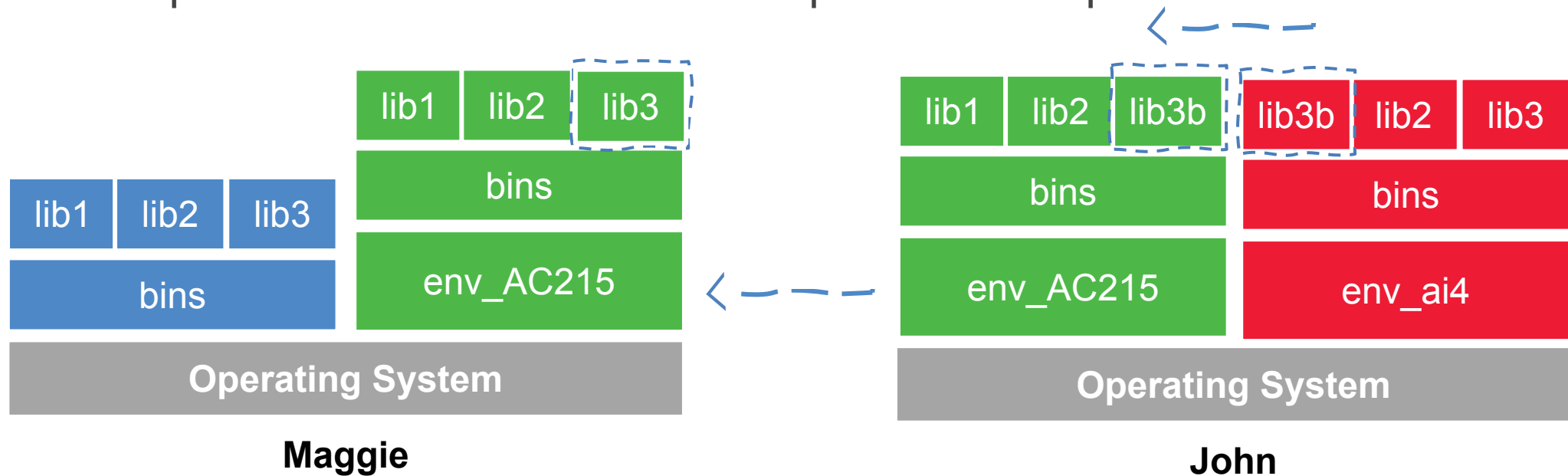
# Why should we use virtual environment?

For the final project, Maggie collaborates with John and shares their working environment by distributing a “configurations 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 “configuration file” (for conda env). They can now update their environment and replicate the experiment.



# 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

---

- **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

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

# venv

- 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`
- listing installed packages: `pip list`
- deactivate environment once done: `deactivate`
- removing a virtual environment: simply delete the `env_name` directory.
- to generate a list of all the Python packages that are currently installed in your environment: `pip freeze > requirements.txt`
- to install all the Python packages listed in a “configuration file”: `pip install -r requirements.txt`

`venv` does not have a built-in command to list all the virtual environments. You can use `pew` which needs to be installed

# 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:

- list all the environments:

```
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

```
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`

A YAML (YAML Ain't Markup Language) file is similar to a dictionary. We will encounter many of them in this course

# PipEnv

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

All packages after  
[dev-packages] in Pipfile are  
for development

Pipfile is human readable and  
specify the packages and  
version constraints  
Pipfile.lock not human readable  
and specifies the packages and  
their dependencies

# PipEnv

---

## How to use it:

- To install pipenv use: `pip install pipenv`
- Navigate to your project directory
- To create a new environment: `pipenv install`
- To activate the environment: `pipenv shell`
- To install a new package `pipenv install numpy`
- To sync from an existing Pipfile.lock: `pipenv sync`
- Exit the virtual environment: `exit`

# 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 Environment Tutorial (T2)

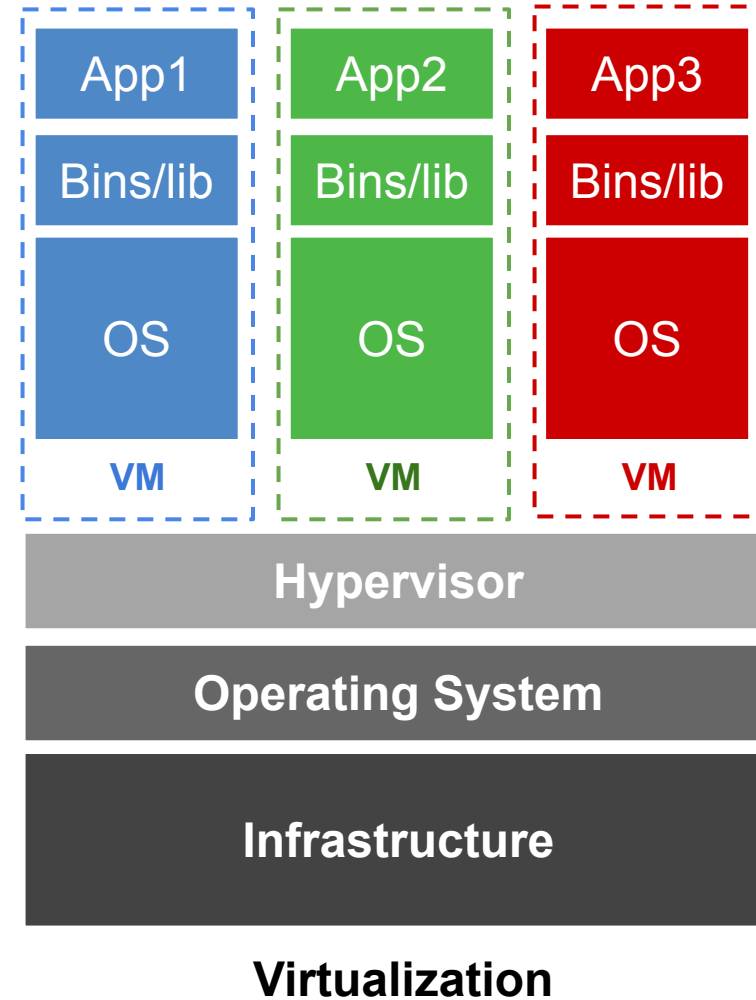
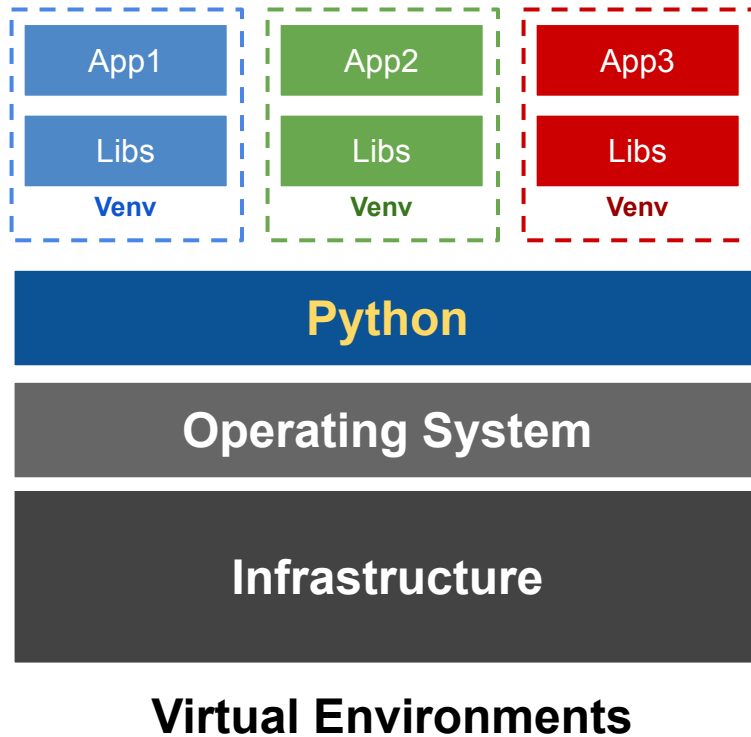
---

- 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](https://github.com/dlops-io/simple-translate?tab=readme-ov-file#installing-app-on-vm-using-pipenv-t2).

<https://github.com/dlops-io/simple-translate?tab=readme-ov-file#installing-app-on-vm-using-pipenv-t2>



# Summary: Virtual Environments vs Virtual Machine



**THANK YOU**