

# Lecture 9: Advanced Training Workflows

AI-5

Productionizing AI (MLOps)

Pavlos Protopapas, Shivas Jayaram

# Outline

---

1. Recap
2. Experiment Tracking
3. Tutorial: Experiment Tracking
4. Vertex AI, Serverless Training
5. Tutorial: Serverless Training
6. Multi GPU Training

# Outline

---

1. **Recap**
2. Experiment Tracking
3. Tutorial: Experiment Tracking
4. Vertex AI, Serverless Training
5. Tutorial: Serverless Training
6. Multi GPU Training

# Recap: Motivation

## The 3 components for better Deep Learning



### More Data

- Extraction
  - Transformation
  - Labeling
  - Versioning
  - Storage
- 
- Processing
  - Input to Training



### Better/Faster Models

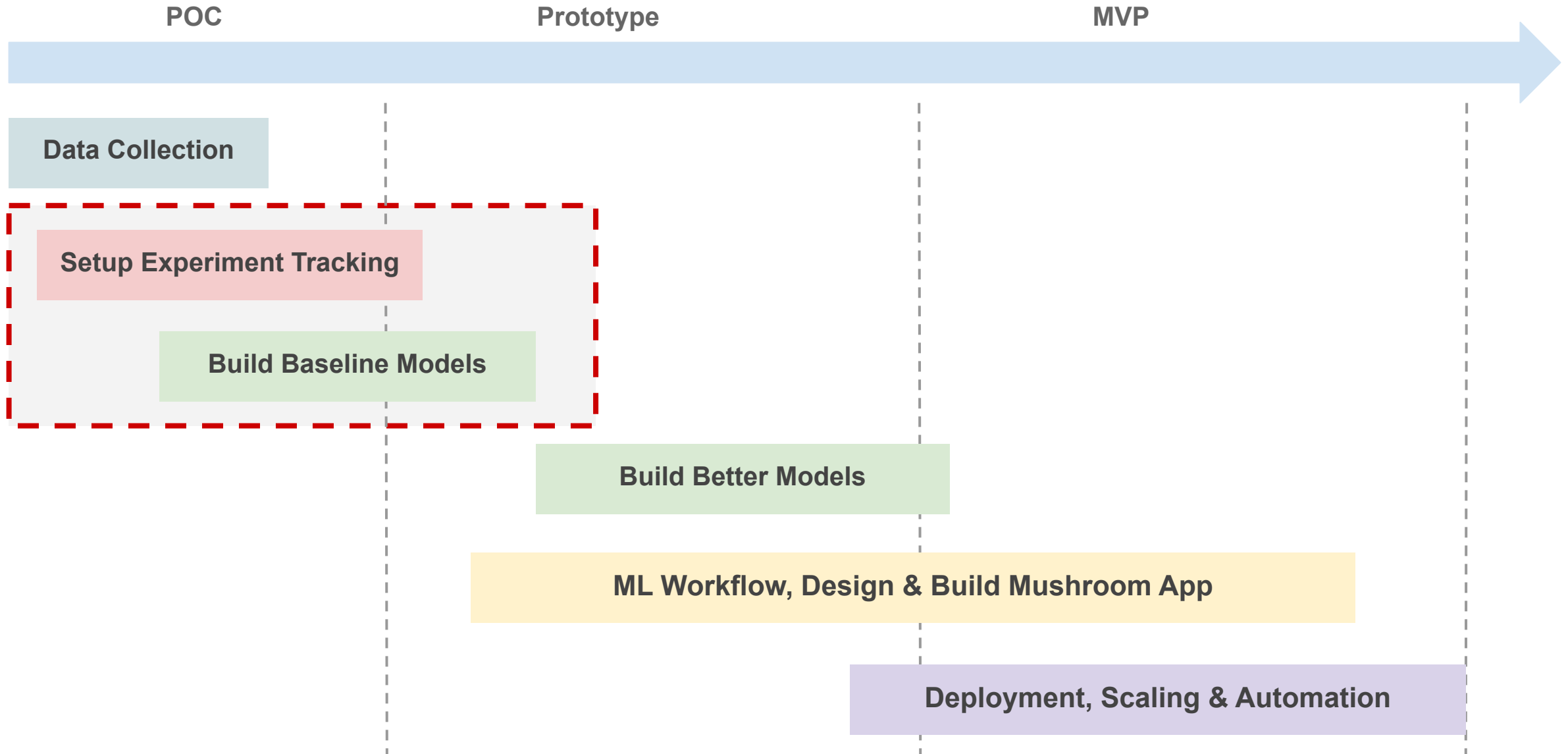
- SOTA Models
- Transfer Learning
- Distillation
- Compression



### Faster Hardware

- Scaling data processing
- GPU, TPU
- Multi GPU Server Training

# Recap: Project Workflow



# Outline

---

1. Recap
- 2. Experiment Tracking**
3. Tutorial: Experiment Tracking
4. Vertex AI, Serverless Training
5. Tutorial: Serverless Training
6. Multi GPU Training

# Experiment Tracking

---

## Why

- Organize your work (data collection/model training)
- Reproducibility
- Logging

# Experiment Tracking

---

## What

- Environments
- Scripts (code)
- Data (version, train/validate/test split)
- Data pre-processing logic
- Model hyper parameters / configurations
- Evaluation metrics
- Model weights
- Performance results
- Sample predictions



# Training Code

```
# Training Params
learning_rate = 0.001
batch_size = 32
epochs = 10

# Data
train_data, validation_data = get_dataset(...)

# Model
model = build_model(...)

# Train
training_results = model.fit(...)
```

# Training Code using WandB

```
# Training Params
```

```
learning_rate = 0.001
```

```
batch_size = 32
```

```
epochs = 10
```

```
# Data
```

```
train_data, validation_data = get_dataset(...)
```

```
# Model
```

```
model = build_model(...)
```

```
# Initialize a W&B run
```

```
wandb.init(...)
```

```
# Train
```

```
training_results = model.fit(..., callbacks=[WandbCallback()])
```

```
# Close the W&B run
```

```
wandb.run.finish()
```

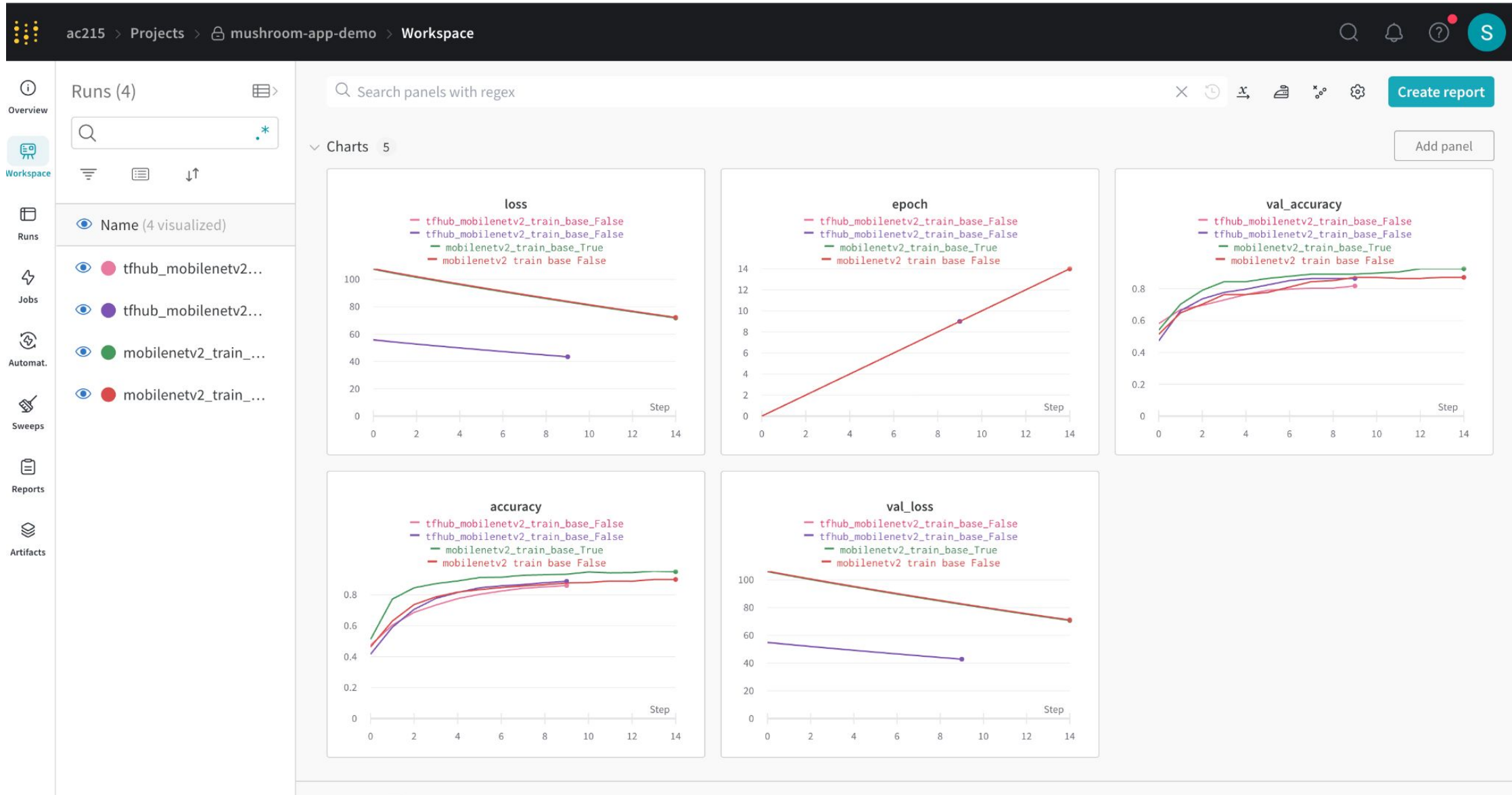
Initialize wandb run



Add a callback to monitor model

Let wandb know to finish the run

# Experiment Tracking using WandB



# Tutorial: Experiment Tracking

---

Goals of the tutorial are

- Explore models for mushroom classifications
- [https://colab.research.google.com/drive/1H7KuMWpzSm8f\\_a\\_8sA1JAylTKISTEqpz?usp=sharing](https://colab.research.google.com/drive/1H7KuMWpzSm8f_a_8sA1JAylTKISTEqpz?usp=sharing)
- Experiment Tracking using Weights & Bias  
<https://colab.research.google.com/drive/1tY5CcNP24kSnaac40pvcc2CPBoz0G6GF?usp=sharing>

# Outline

---

1. Recap
2. Experiment Tracking
3. Tutorial: Experiment Tracking
4. **Vertex AI, Serverless Training**
5. Tutorial: Serverless Training
6. Multi GPU Training

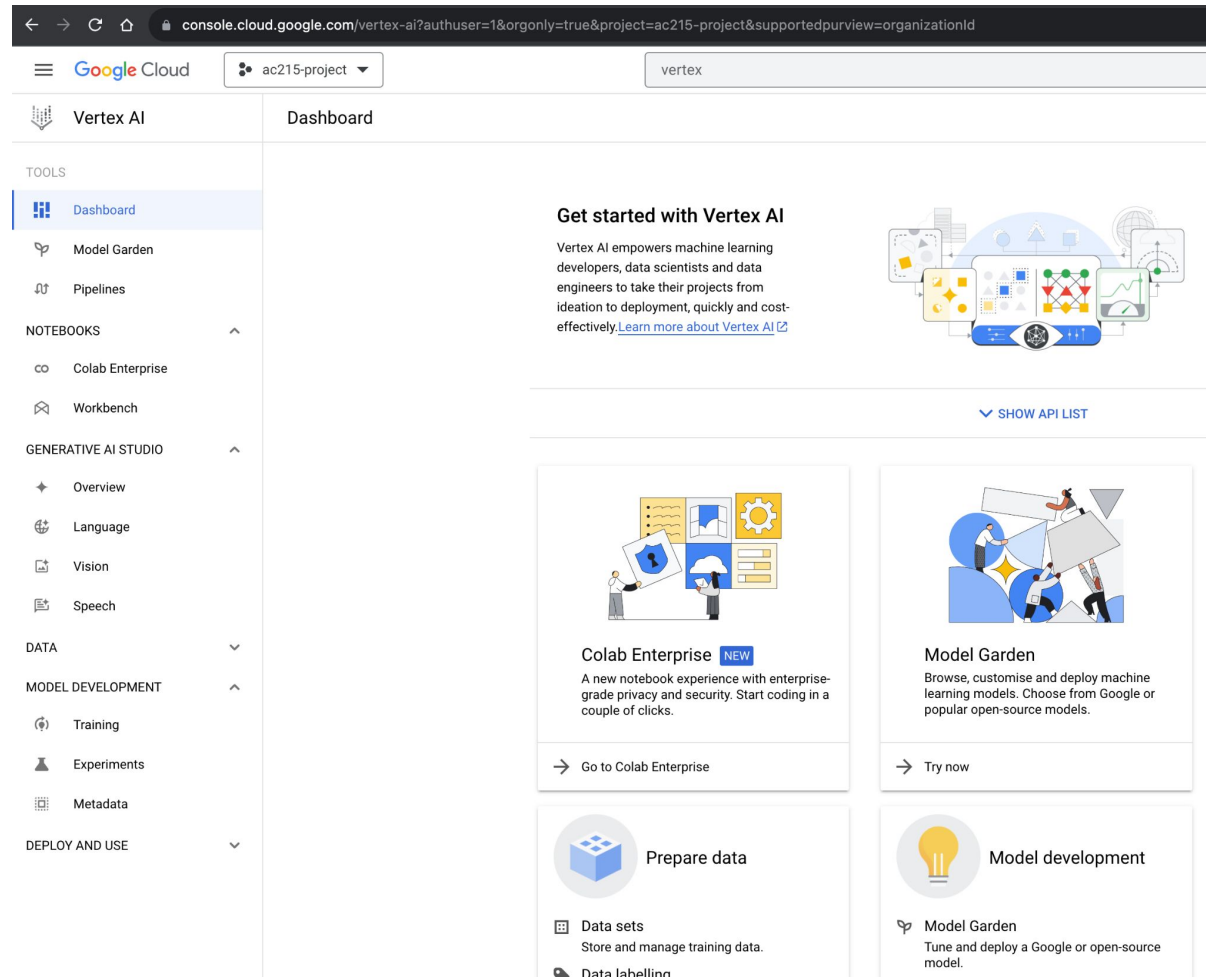
# Vertex AI

---

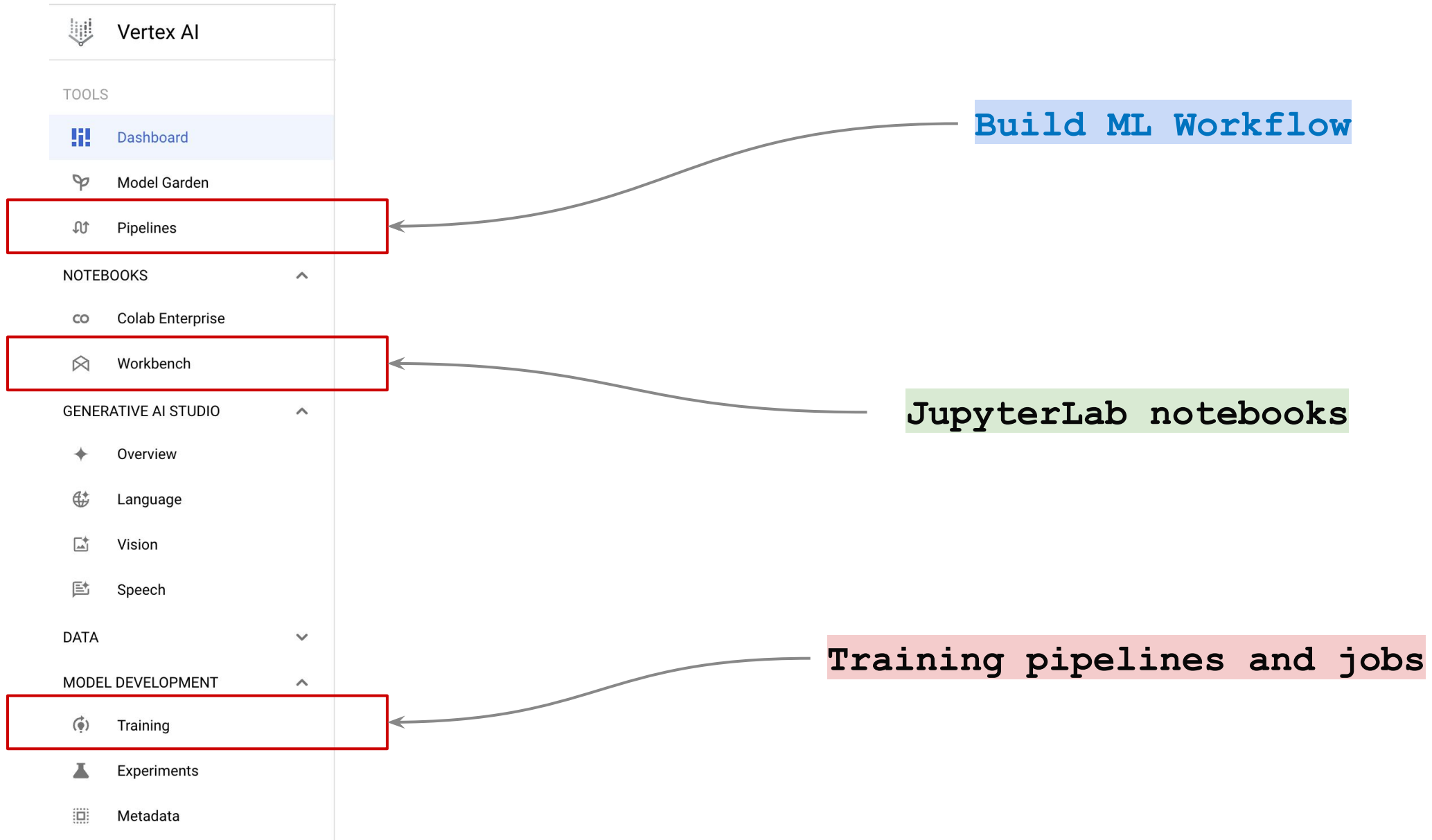
- **Vertex AI** is machine learning platform offered by Google in GCP.
- Vertex AI combines **data engineering**, **data science**, and **ML engineering workflows**, enabling your teams to **collaborate** using a common toolset and scale your applications using the benefits of Google Cloud.

# Vertex AI

Vertex AI: <https://console.cloud.google.com/vertex-ai>



# Vertex AI





# Serverless Training

---

## **What is serverless training?**

- Execute training on an as-needed basis
- Access GPU hardware only for the “training” step in a pipeline
- No setup of servers required
- Brings down training cost

# Serverless Training

## 1 Move Code to Python File

### CO Notebook

```
def get_dataset():  
    ...  
def get_model_1():  
    ...  
def get_model_2():  
    ...  
  
# Data  
train_data, val_data = get_dataset(...)  
# Model  
model_1 = build_model_1(...)  
# Train  
training_results = model_1.fit(...)
```

### Python File

```
def get_dataset():  
    ...  
def get_model_1():  
    ...  
def get_model_2():  
    ...  
  
# Data  
train_data, val_data = get_dataset(...)  
# Model  
model = ...  
# Train  
training_results = model.fit(...)
```

Pass Arguments:

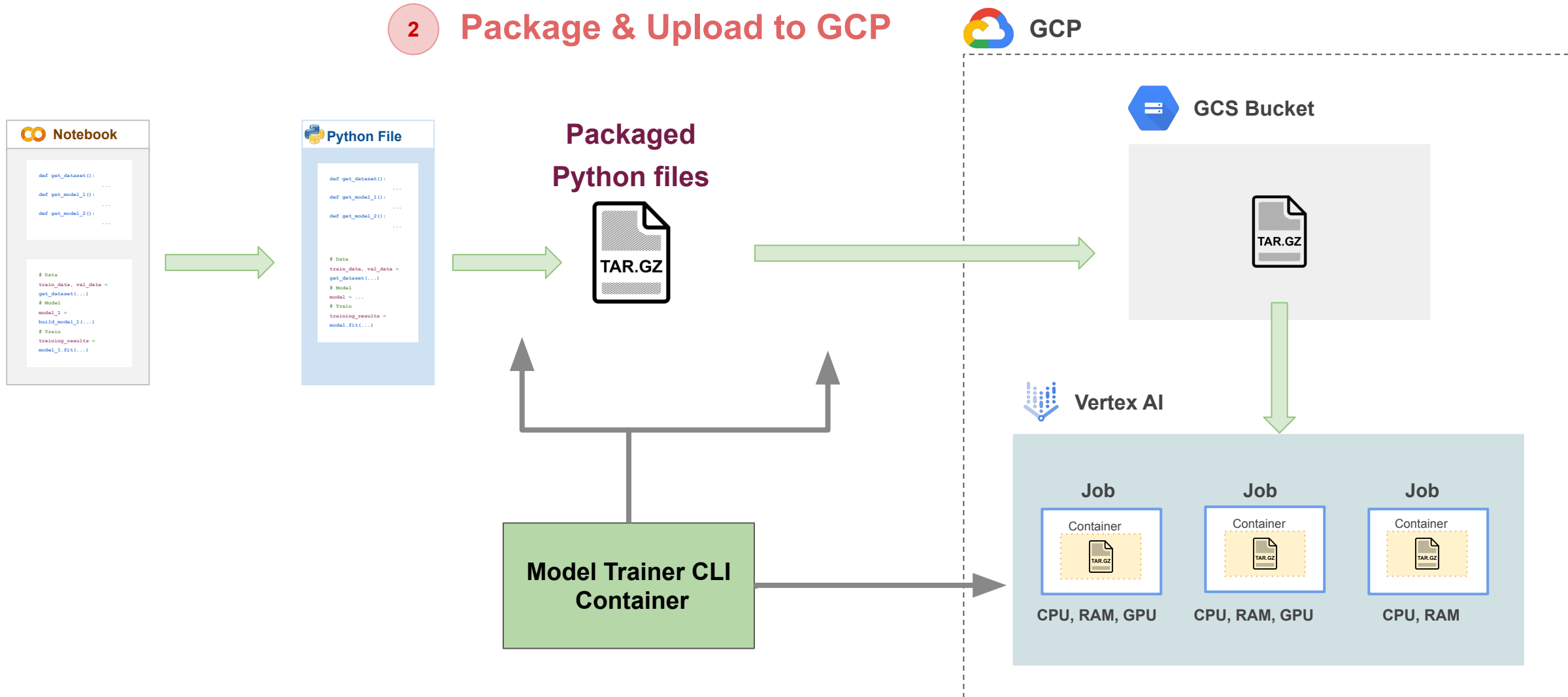
model=model\_1

epochs=25

batch\_size=32

# Serverless Training

## 2 Package & Upload to GCP



## 3 Create & Run Training Jobs

# Tutorial: Serverless Training

Steps to perform **Serverless Training** on mushroom classification models:

- Create a GCS bucket to store packaged python training code.
- Get Weights & Bias API Key for experiment tracking.
- Package & Upload python code.
- Create Jobs in Vertex AI.
- For detailed instructions, please refer to the following link
  - [Serverless Training](https://github.com/dlops-io/model-training). ( <https://github.com/dlops-io/model-training> )
  - [View Training Jobs](https://console.cloud.google.com/vertex-ai/training/custom-jobs). ( <https://console.cloud.google.com/vertex-ai/training/custom-jobs> )
  - [View Experiment Metrics](https://wandb.ai/home). ( <https://wandb.ai/home> )

# Outline

---

1. Recap
2. Experiment Tracking
3. Tutorial: Experiment Tracking
4. Vertex AI, Serverless Training
5. Tutorial: Serverless Training
6. **Multi GPU Training**

## How do we perform distributed training?

- What type of distribution:
  - Single Machine, Single GPU [One Device Strategy]
  - Single Machine, Multiple GPUs [Mirrored Strategy]
  - Multiple Machine, Multiple GPUs [Multi Worker Mirrored Strategy]
- Organize code to apply the appropriate Strategy
- Train as usual

# Training Code

```
# Training Params
learning_rate = 0.001
batch_size = 32
epochs = 10

# Data
train_data, validation_data = get_dataset(...)

# Model
model = build_model(...)

# Train
training_results = model.fit(...)
```

# Training Code for Multi GPU

```
# Training Params
```

```
batch_size = 32
```

```
...
```

```
# Create distribution strategy
```

```
strategy = tf.distribute.MirroredStrategy()
```

```
num_workers = strategy.num_replicas_in_sync
```

```
# Data
```

```
train_data, validation_data = get_dataset(..., batch_size=batch_size * num_workers)
```

```
# Wrap model creation and compilation within scope of strategy
```

```
with strategy.scope():
```

```
    # Model
```

```
    model = build_model(...)
```

```
# Train
```

```
training_results = model.fit(...)
```

Create distribution Strategy



Adjust dataset batch size

Create & Compile model in strategy scope



**THANK YOU**