Lecture 11&12: ML Workflow Management

AI-5

Productionizing AI (MLOps)

Pavlos Protopapas, Shivas Jayaram

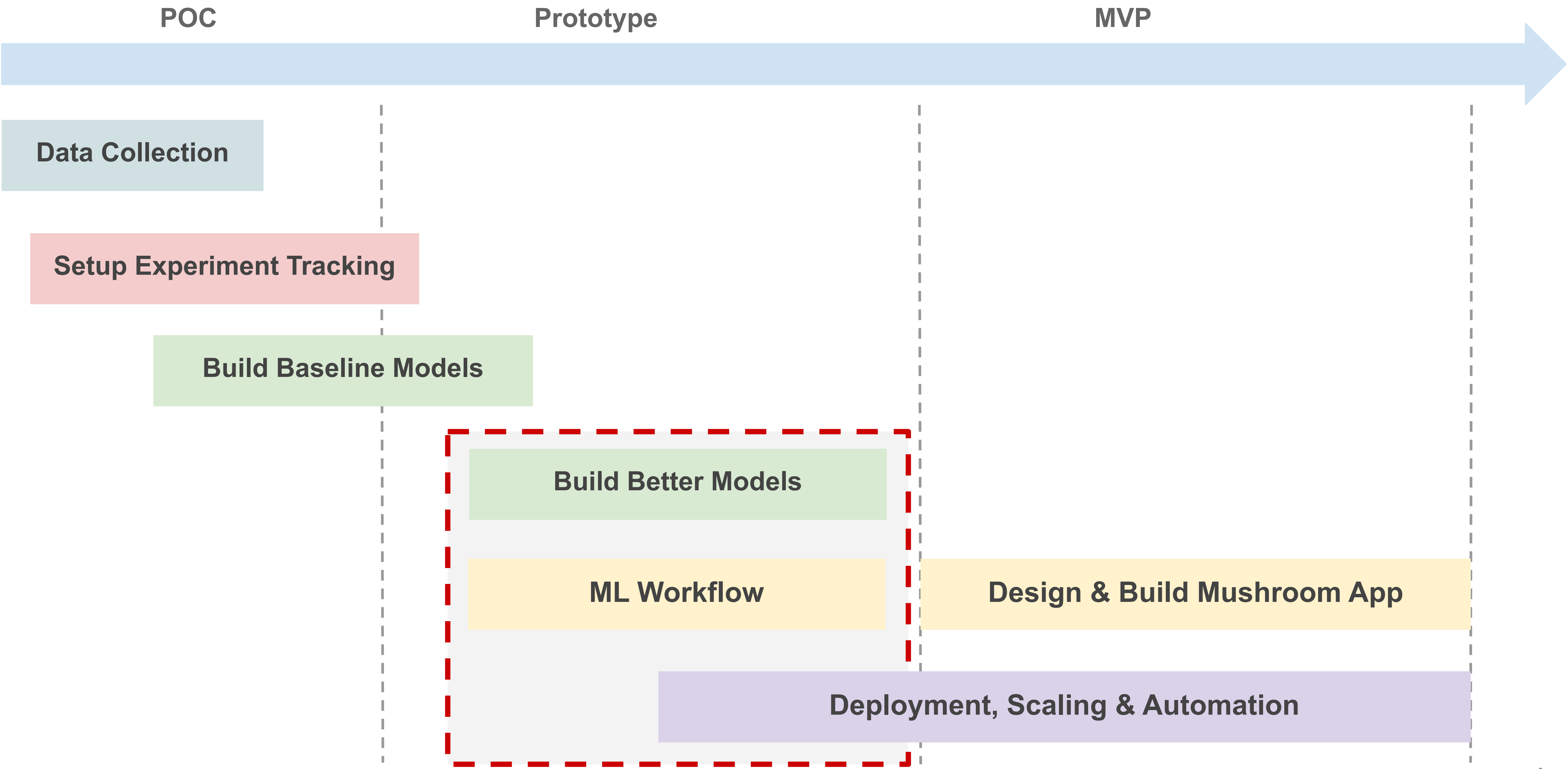
Outline

1. Recap
2. Serverless: Cloud Functions
3. Serverless: Cloud Run
4. Serverless: Model Deployment
5. ML Workflow Management
6. Vertex AI Pipelines

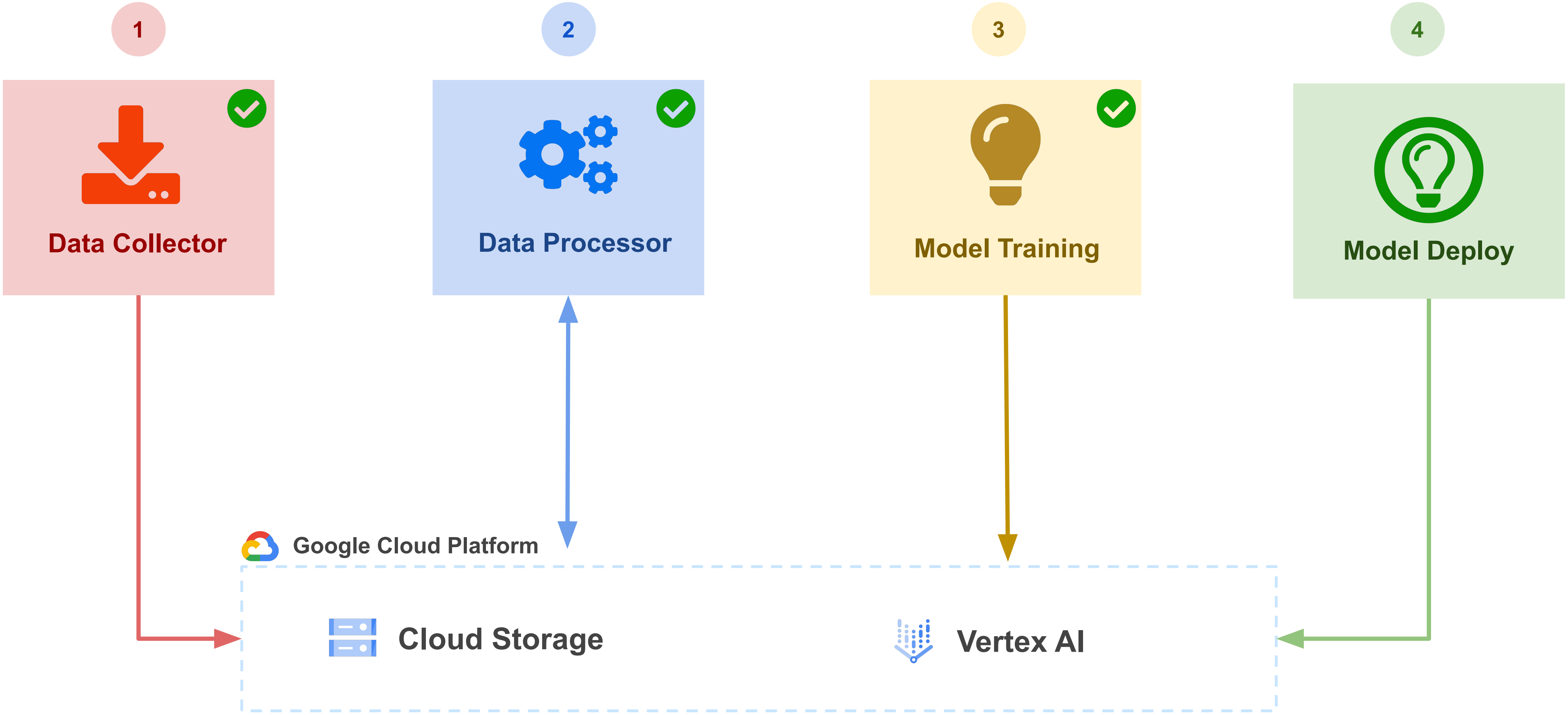
Outline

1. **Recap**
2. Serverless: Cloud Functions
3. Serverless: Cloud Run
4. Serverless: Model Deployment
5. ML Workflow Management
6. Vertex AI Pipelines

Recap: Mushroom App Status



Mushroom App Development



Protopapas, Jayaram

Outline

1. Recap
2. **Serverless: Cloud Functions**
3. Serverless: Cloud Run
4. Serverless: Model Deployment
5. ML Workflow Management
6. Vertex AI Pipelines

Serverless

What is serverless?

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

Serverless

Types of serverless:

- Cloud Function
- Cloud Run
- Training Job (Vertex AI)
- Model Deployment (Vertex AI)
- Pipeline (Vertex AI)

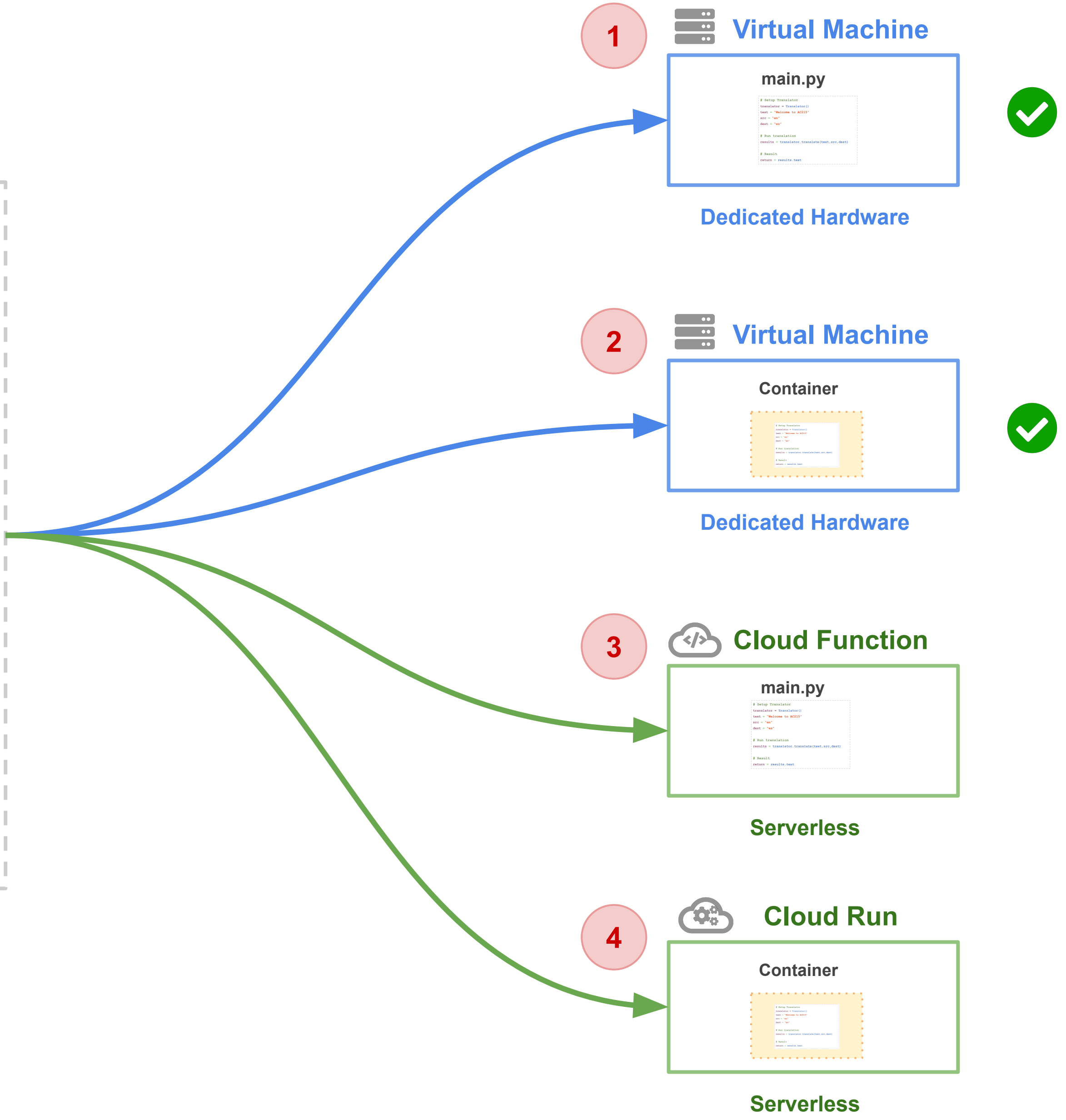
Deployment Options

Simple Translate App

```
# Setup Translator
translator = Translator()
text = "Welcome to AI5"
src = "en"
dest = "es"

# Run translation
results = translator.translate(text,src,dest)

# Result
return = results.text
```



Cloud Function

What is a could function?

- Run your code in GCP with no servers or containers
- Pay only for function execution time
- Scale out easily

Tutorial: Cloud Function

Steps to deploy an app as a **Cloud Function**

- Go to https://console.cloud.google.com/functions.
- Enable GCP APIs.
- Create a python code file.
- Deploy code as Cloud Function.
- For detailed instructions, please refer to the following link
  - Running App as Cloud Function. (https://github.com/dlops-io/serverless-deployment#running-app-as-cloud-function)

Outline

1. Recap
2. Serverless: Cloud Functions
3. **Serverless: Cloud Run**
4. Serverless: Model Deployment
5. ML Workflow Management
6. Vertex AI Pipelines

Cloud Run

What is cloud run?

- Run your containerized apps with no servers.
- Run containers as **service** or **job**.
- Only pay when your code is running
- Scale out easily

Tutorial: Cloud Run

Steps to deploy an app in **Cloud Run**

- Go to https://console.cloud.google.com/run.
- Enable GCP APIs.
- Deploy Docker Image in Cloud Run.
- For detailed instructions, please refer to the following link
    - Running App in Cloud Run. (https://github.com/dlops-io/serverless-deployment#running-app-in-cloud-run)

Outline

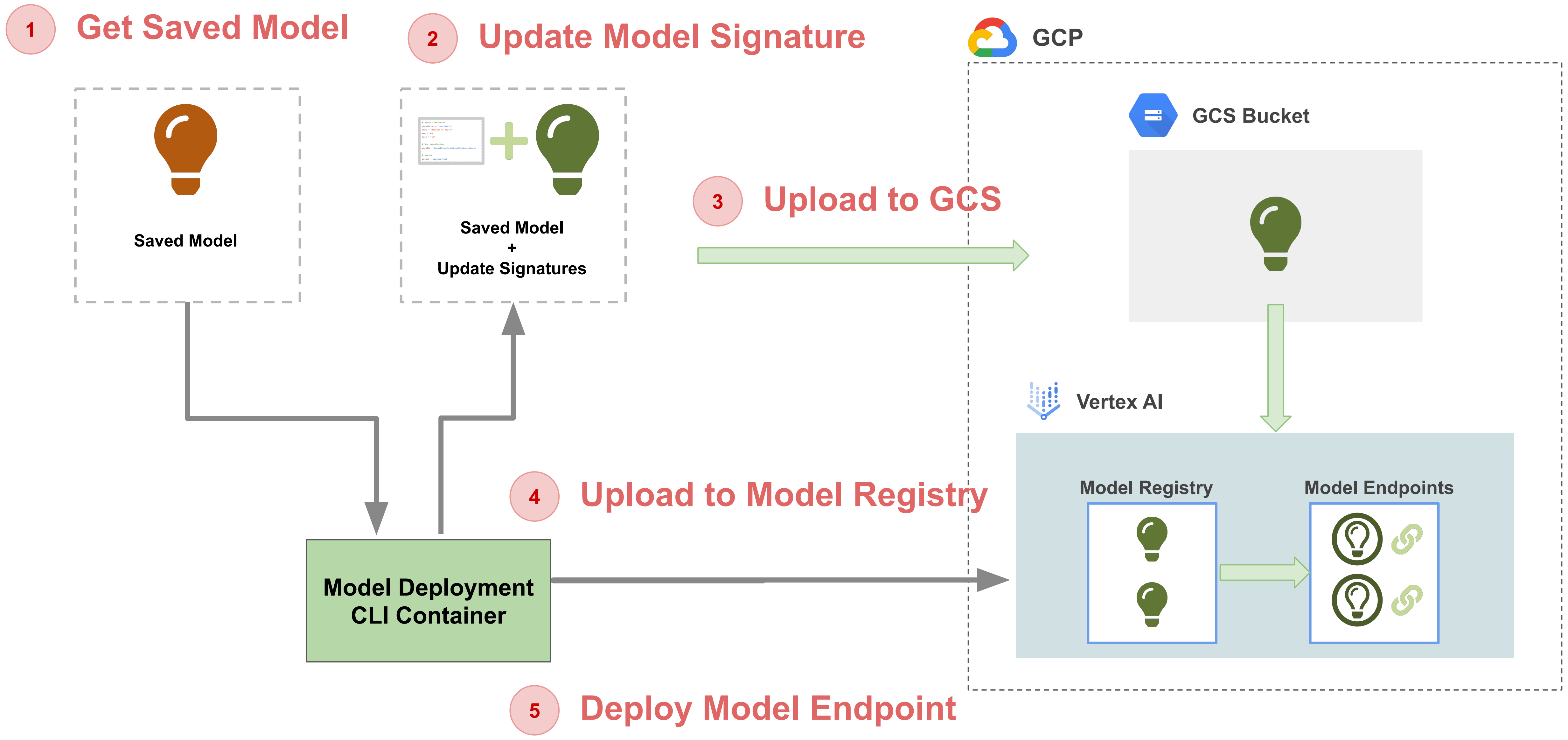
1. Recap
2. Serverless: Cloud Functions
3. Serverless: Cloud Run
4. **Serverless: Model Deployment**
5. ML Workflow Management
6. Vertex AI Pipelines

Serverless Model Deployment

What is serverless model deployment?

- Deploy our trained model for predictions with no servers.
- Setup **online** or **batch** prediction modes
- For **online** predictions there is an ongoing cost
- Access GPU or CPU hardware for inference
- Scale out easily
- Alert: Continuous cost to keep endpoint up

Serverless Model Deployment



Serverless Model Deployment: Model Signature

Why do we need to update the model signature?

- Make model input to accept a raw image
- Perform data preprocessing steps prior to model inference
- Combine data preprocessing & model inference in one endpoint

Serverless Model Deployment: Update Model Signature

```python
# Preprocess Image
def preprocess_image(bytes_input):
    resized = ...
    return resized

# Define tf functions
@tf.function(input_signature=[tf.TensorSpec([None], tf.string)])
def preprocess_function(bytes_inputs):
    decoded_images = tf.map_fn(
        preprocess_image, bytes_inputs, dtype=tf.float32, back_prop=False
    )
    return {"model_input": decoded_images}

@tf.function(input_signature=[tf.TensorSpec([None], tf.string)])
def serving_function(bytes_inputs):
    images = preprocess_function(bytes_inputs)
    results = model_call(**images)
    return results

# Update model signature and save
tf.saved_model.save(
    prediction_model,...,
    signatures={"serving_default": serving_function},
)
```

Define preprocessing function

Define @tf.function for new model signature

Save Model with the new model signature

Tutorial: Serverless Model Deployment

Steps to perform **Serverless Model Deployment** on mushroom classification model:

- Create a GCS bucket to store saved model.
- Update Model Serving Signature
- Upload Model to Vertex AI Model Registry.
- Deploy Model as an Endpoint.
- For detailed instructions, please refer to the following link
  - Serverless Model Deployment. (https://github.com/dlops-io/model-deployment)
  - View Model Endpoints. (https://console.cloud.google.com/vertex-ai/online-prediction/endpoints)
  - View Model Registry. (https://console.cloud.google.com/vertex-ai/models)

Outline

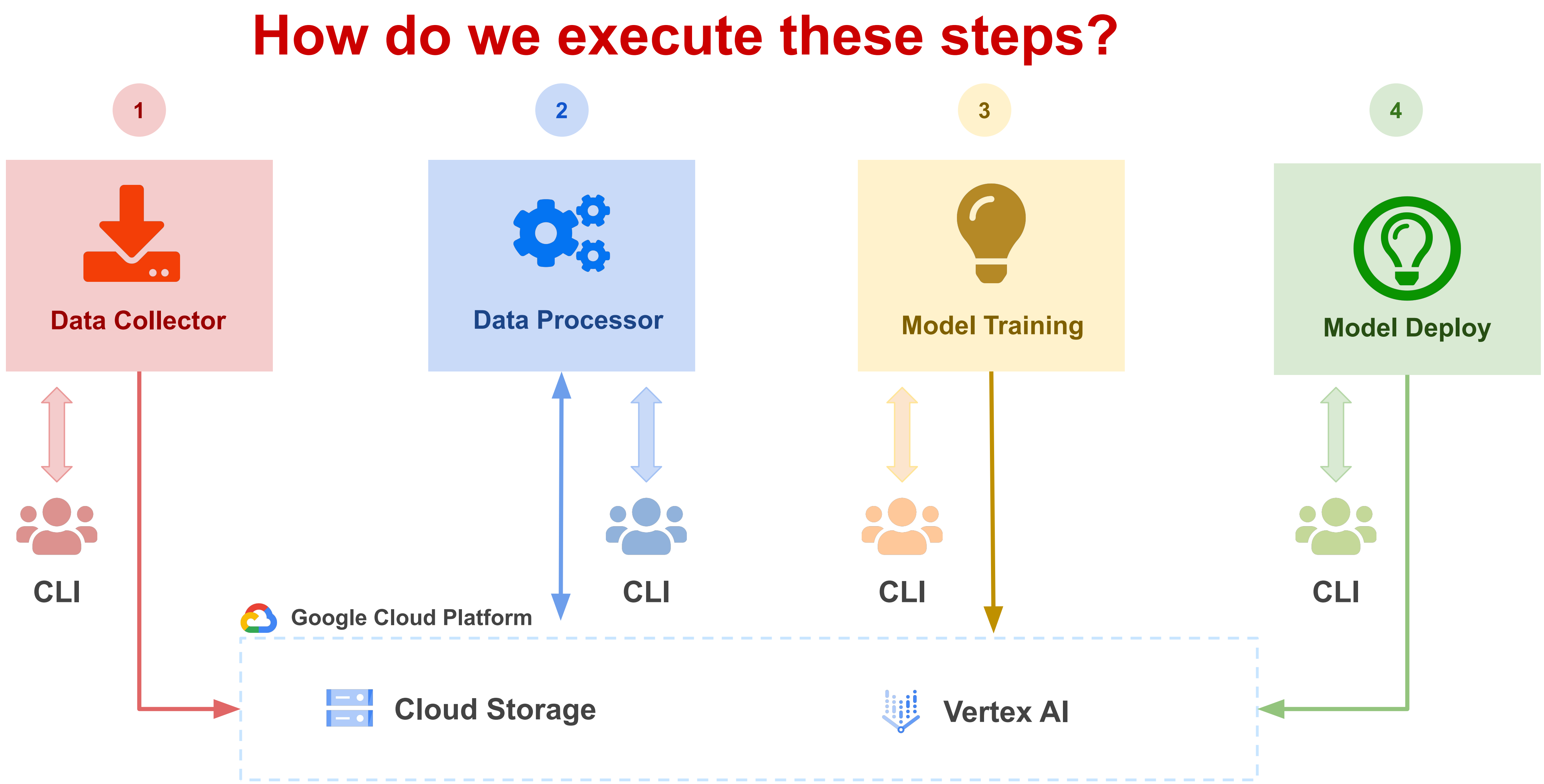
1. Recap
2. Serverless: Cloud Functions
3. Serverless: Cloud Run
4. Serverless: Model Deployment
5. **ML Workflow Management**
6. Vertex AI Pipelines

ML Workflow Management

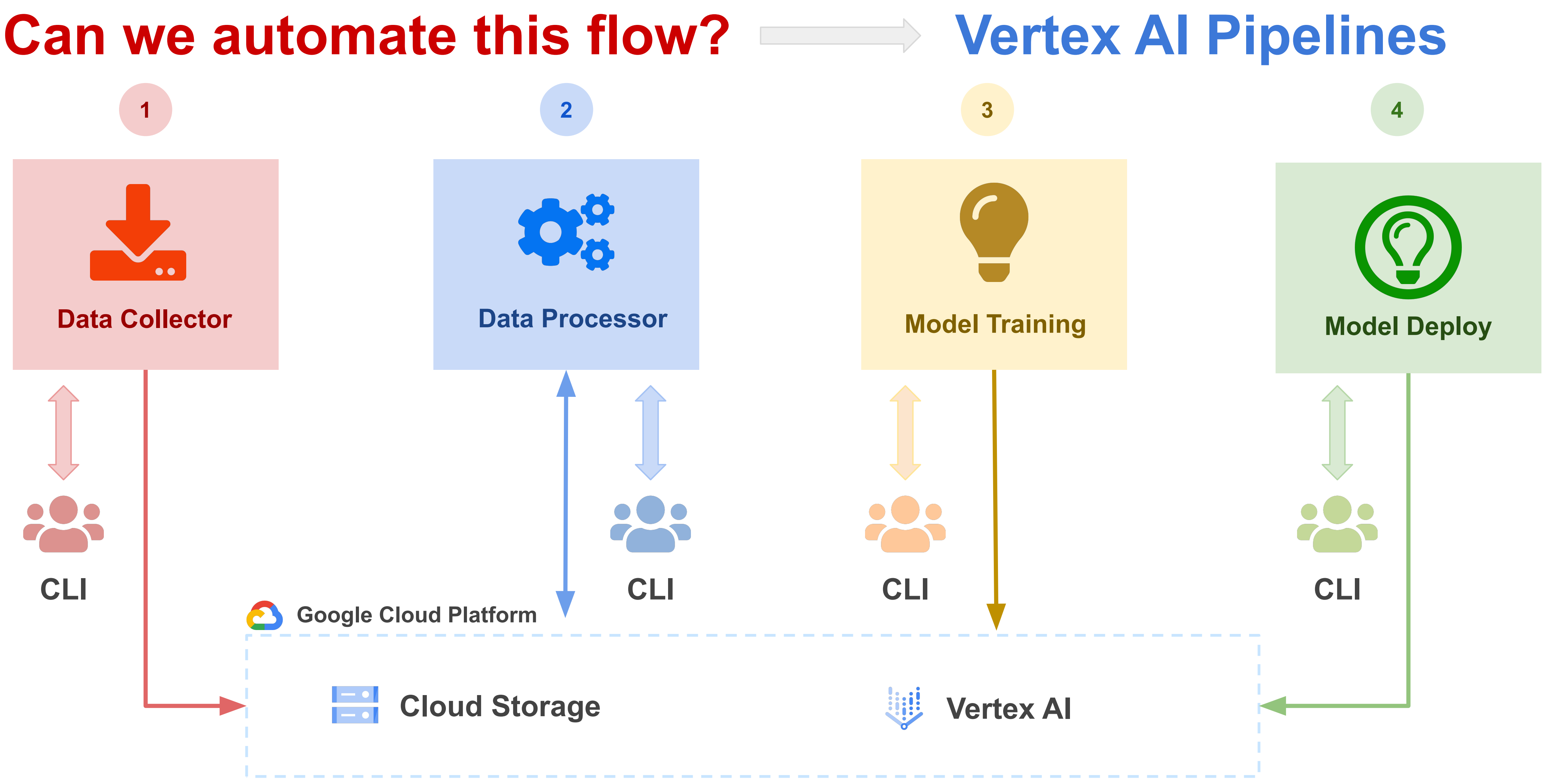
What is ML workflow management?

- Helps us efficiently manage end-to-end ML tasks from data collection to model deployment
- Helps orchestrate various and automated pipeline execution
- Manages collaboration, integration, and scalability

ML Workflow: Mushroom App



ML Workflow: Mushroom App



Protopapas, Jayaram

Outline

1. Recap
2. Serverless: Cloud Functions
3. Serverless: Cloud Run
4. Serverless: Model Deployment
5. ML Workflow Management
6. **Vertex AI Pipelines**

Vertex AI Pipelines

What is Vertex AI Pipelines?

- Vertex AI is machine learning platform offered by Google in GCP.
- Vertex AI Pipelines helps you to automate, monitor, and govern your ML components by orchestrating your ML workflow in a serverless manner

Building Vertex AI Pipelines

```python
# Import Kubeflow Pipelines
from kfp import dsl
# Define Components
@dsl.component
def square(x: float) -> float:
    return x**2
@dsl.component
def add(x: float, y: float) -> float:
    return x + y
@dsl.component
def square_root(x: float) -> float:
    return x**0.5
# Define Pipeline
@dsl.pipeline
def sample_pipeline(a: float = 3.0, b: float = 4.0) -> float:
    a_sq_task = square(x=a)
    b_sq_task = square(x=b)
    sum_task = add(x=a_sq_task.output, y=b_sq_task.output)
    return square_root(x=sum_task.output).output
```

Import kubeflow pipeline SDK

Define pipeline components

Define Pipeline, an orchestration of how you want your component tasks to run

Building Vertex AI Pipelines

Define Pipeline, an orchestration of how you want your component tasks to run

```
...

# Define Pipeline
@dsl.pipeline
def sample_pipeline(a: float = 3.0, b: float = 4.0) -> float:
    a_sq_task = square(x=a)
    b_sq_task = square(x=b)
    sum_task = add(x=a_sq_task.output, y=b_sq_task.output)
    return square_root(x=sum_task.output).output

# Build yaml file for pipeline
compiler.Compiler().compile(
    sample_pipeline, package_path="sample-pipeline.yaml"
)
```

Compile pipeline into a yaml file

Running Vertex AI Pipelines

```
# Initialize GCP
import google.cloud.aiplatform import aip

# Create a Pipeline Job in Vertex AI
job = aip.PipelineJob(
    display_name=DISPLAY_NAME,
    template_path="sample-pipeline1.yaml",
    pipeline_root=PIPELINE_ROOT,
    enable_caching=False,
)

# Run the Pipeline Job
job.run()
```

Import Google Cloud SDK

Create a pipeline job

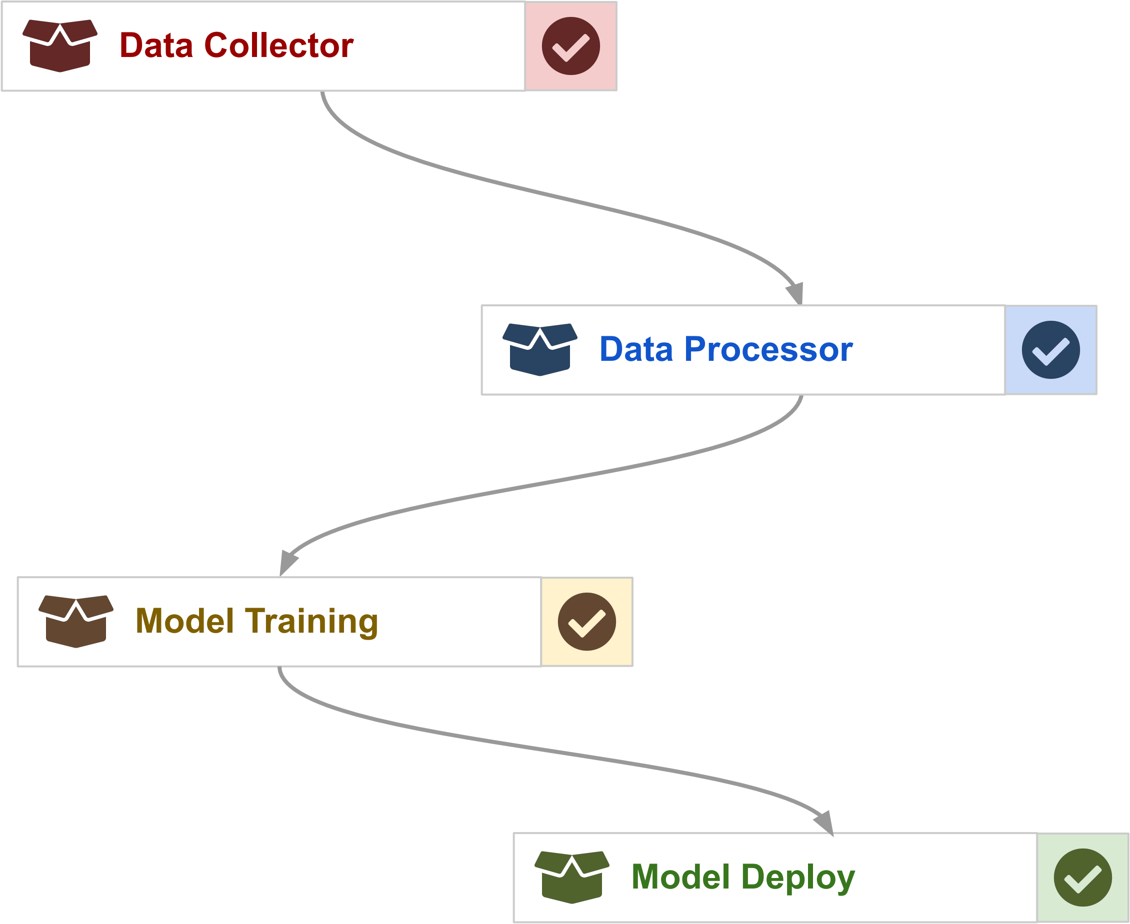
Run pipeline job in Vertex AI

Building Vertex AI Pipelines

Steps to build pipelines for your custom containers

- Make your containers callable
- Build & Push Container Images to a Container Registry
- Define a sequence of steps using a directed acyclic graph (DAG)

Building Vertex AI Pipelines



1. Download images
2. Uploads to GCP

1. Verify images
2. Check for duplicates
3. Convert to TF Records

1. Train model
2. Save model

1. Upload to Registry
2. Deploy model Endpoint

Making Container Callable

Dockerfile

```
# Use the official Debian-hosted ...
FROM python:3.9-slim-buster
.
.
.
# Add the rest of the source code.
RUN --chown=app:app . /app
# Entry point
ENTRYPOINT ["pipenv", “shell”]
```

Dockerfile

```
# Use the official Debian-hosted ...
FROM python:3.9-slim-buster
.
.
.
# Add the rest of the source code.
RUN --chown=app:app . /app
# Entry point
ENTRYPOINT ["/bin/bash","./docker-entrypoint.sh"]
```

Change entrypoint to a shell file

Making Container Callable

Development mode:
Authenticated to GCP
pipenv shell to test cli inside container

docker-entrypoint.sh

```
#!/bin/bash
args = "$@"

if [[ -z ${args} ]];
then
    # Authenticate gcloud using service account
    gcloud auth activate-service-account --key-file $GOOGLE_APPLICATION_CREDENTIALS
    # Set GCP Project Details
    gcloud config set project $GCP_PROJECT
    pipenv shell
else
  pipenv run python $args
fi
```

Production mode:
Run container using "docker run ... cli.py –search"

Tutorial: Vertex AI Pipelines

Steps to build **Vertex AI Pipelines** on the mushroom app ML workflow components:

- Make Containers Callable.
- Build & Push Image.
- Build ML Pipeline.
- Run Pipeline in Vertex AI
- For detailed instructions, please refer to the following link
  - Mushroom App Workflows. (https://github.com/dlops-io/ml-workflow#mushroom-app-ml-workflow-management)
  - View Vertex AI Pipelines. (https://console.cloud.google.com/vertex-ai/pipelines/runs)

THANK YOU