

mlflow

Platform for Machine Learning Lifecycle

Jules S. Damji

@2twitme

Outline – Introduction to MLflow: How to Use MLflow Projects – Module 1

- MLFlow Component
 - MLflow Projects
 - Concepts and Motivations
 - Executing MLflow projects in (DCE)
 - Building MLFlow Projects
 - Explore MLflow UI
 - Tutorials & Exercises
- Q & A

<https://github.com/dmatrix/olt-mlflow>

MLflow Components

mlflow

Tracking

Record and query experiments: code, data, config, and results

mlflow

Projects

Package data science code in a format that enables reproducible runs on many platform

mlflow

Models

Deploy machine learning models in diverse serving environments

new

mlflow

Model Registry

Store, annotate and manage models in a central repository

[databricks.com
/mlflow](https://databricks.com/mlflow)



mlflow.org



github.com/mlflow



twitter.com/MLflow

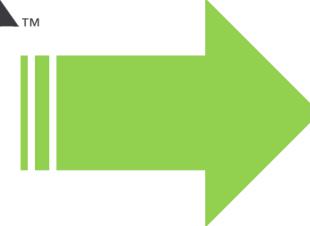
O'REILLY®

MLflow Projects Motivation

Diverse set of tools



TensorFlow



Diverse set of environments

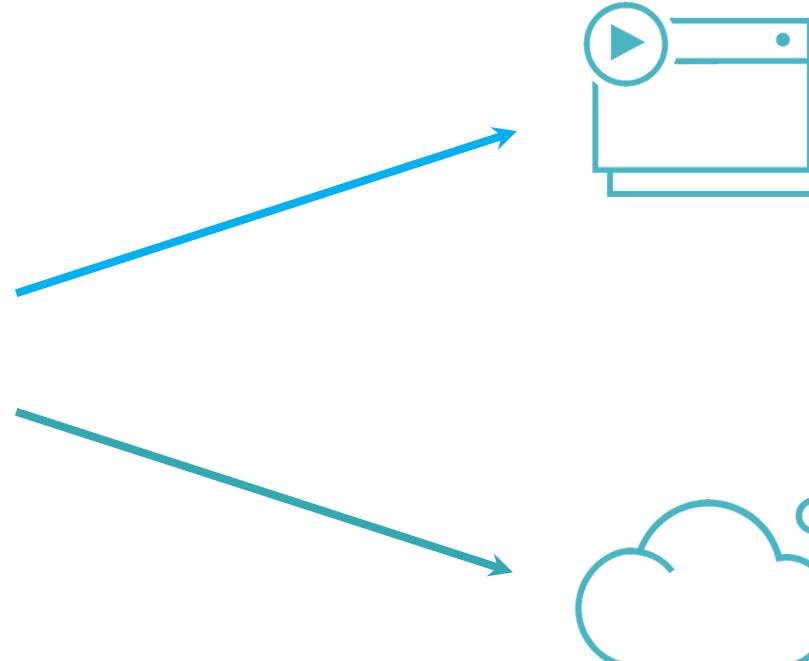
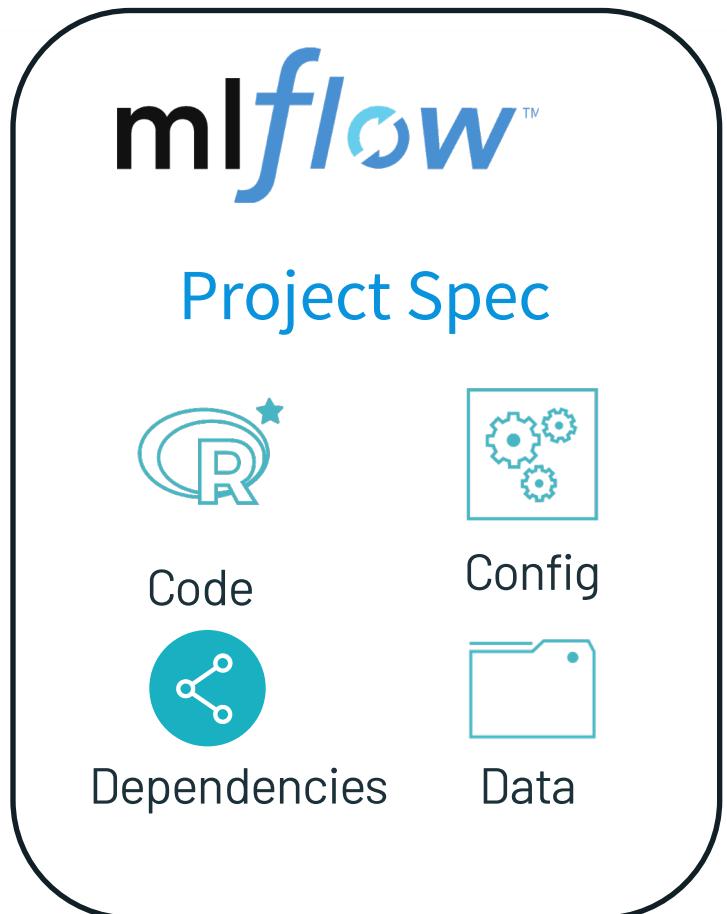


mlflow™
Projects

Package data science
code in a format that
enables reproducible runs
on any platform

Challenge: ML results difficult to reproduce

MLflow Projects



Local Execution



Remote Execution



databricks

MLflow Projects Environments

- MLflow Supports three software environments
 - Environments to Execute your Entry point code



- Conda for Python Packages
- Docker for non-Python Packages
- Current System environment

1. Example MLflow Project File

```
my_project/
├── MLproject
|
└── main.py
    └── model.py
...

```

```
name: tutorial

conda_env: conda.yaml

entry_points:
  main:
    parameters:
      batch_size: {type: int, default: 10}
      epochs: {type: int, default: 100}
    command: "python train_keras.py {batch_size} {epochs}
```

```
$ mlflow run git://<my_project>.git -P epochs=200
mlflow.run("git://<my_project>", parameters={..})

mlflow run . -e main -P epochs=200
```

2. Example Conda.yaml

```
my_project/
└── MLproject
    ├── conda.yaml
    ├── main.py
    └── model.py
    ...

```

```
channels:
- defaults
- conda-forge
dependencies:
- python=3.7.5
- pip
- pip:
  - mlflow
  - keras==2.3.1
  - tensorflow==2.0.0
name: mlflow-env
```

MLflow Projects

Packaging format for reproducible ML runs

- Any code folder or GitHub repository
- MLproject file with project configuration

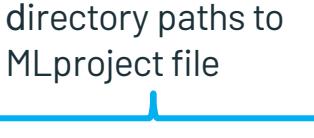
Defines dependencies for reproducibility

- Conda (+ R, Docker, ...) dependencies can be specified in MLproject
- Reproducible in (almost) any environment

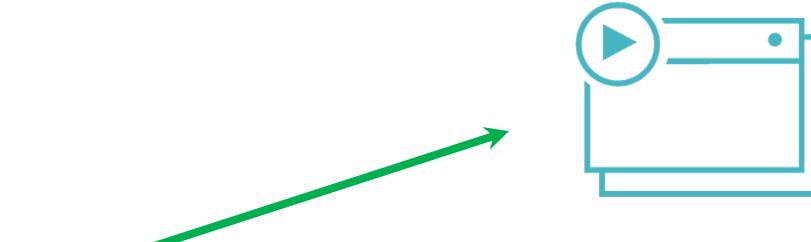
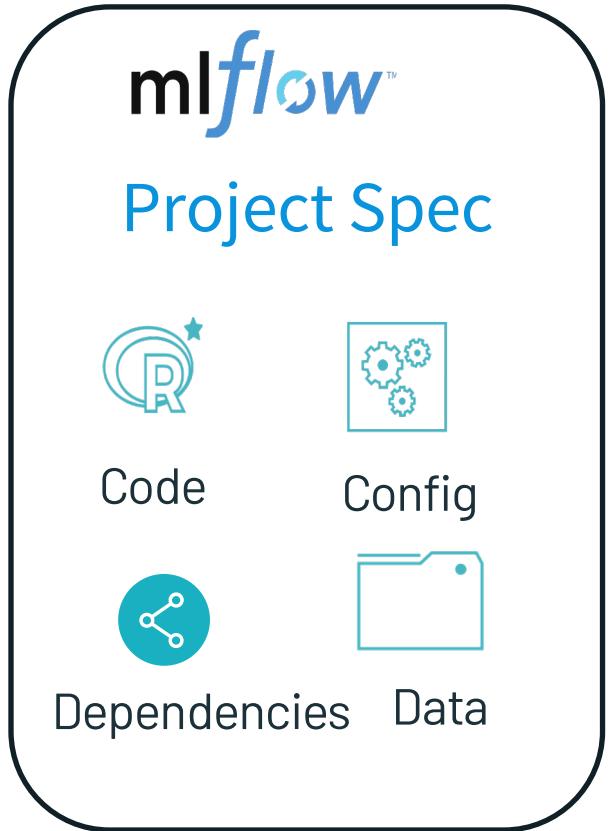
Execution API for running projects

- CLI / Python / R / Java
- Supports local and remote execution
 - mlflow run –help (CLI)
 - mlflow run <https://github.com/dmatrix/jsd-mlflow-examples.git#keras/imdbclassifier> (CLI)
 - mlflow.run (<project_uri>, parameters={}) or mlflow.projects.run(<project_uri>, parameters={}) (API)

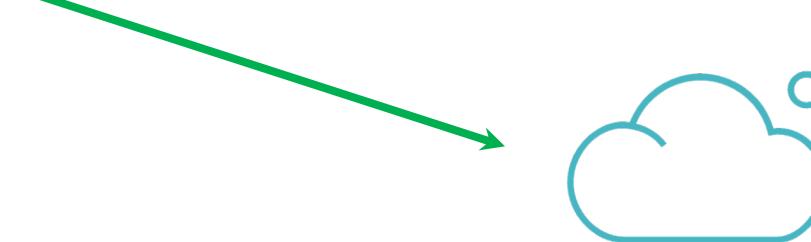
directory paths to
MLproject file



MLflow Project Execution



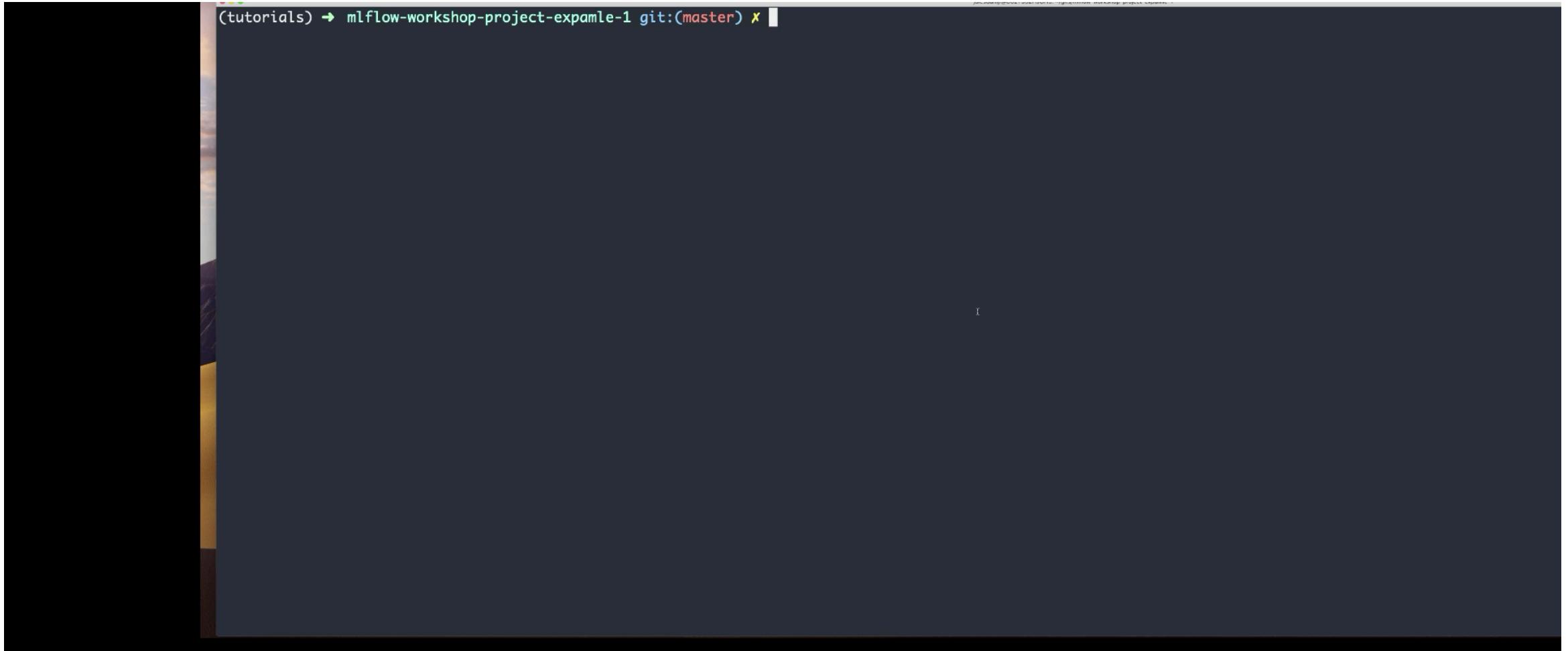
Local Execution



Remote Execution



MLflow Project Local Execution



A screenshot of a terminal window with a dark background. The title bar shows '(tutorials) ➔ mlflow-workshop-project-expamle-1 git:(master) x'. In the main area, the command 'mlflow run' is being typed, with the 'r' key partially visible.

MLflow Project Remote Execution



MLflow Project Remote Execution: Databricks

```
{ "new_cluster":  
  { "spark_version": "7.0.x-scala2.12",  
    "node_type_id": "i3.xlarge",  
    "aws_attributes": {"availability": "ON_DEMAND"},  
    "num_workers": 4 },  
  "libraries": [  
    { "pypi": {  
        "package": "tensorflow" }  
    }  
  ]  
}
```

1

Databricks Cluster Config Specification
dbr_project_spec.json

2

```
mlflow run git://github.com/mlflow/mlflow-example.git -b databricks --backend-config dbr_project_spec.json
```

MLflow Project Remote Execution: Kubernetes (Experimental)

```
FROM continuumio/miniconda:4.5.4

RUN pip install mlflow>=1.0 \
    && pip install azure-storage-blob==12.3.0 \
    && pip install numpy==1.14.3 \
    && pip install scipy \
    && pip install pandas==0.22.0 \
    && pip install scikit-learn==0.19.1 \
    && pip install cloudpickle
```

1

Create a Dockerfile

2

```
docker build -t mlflow-docker-example -f Dockerfile .
```

3

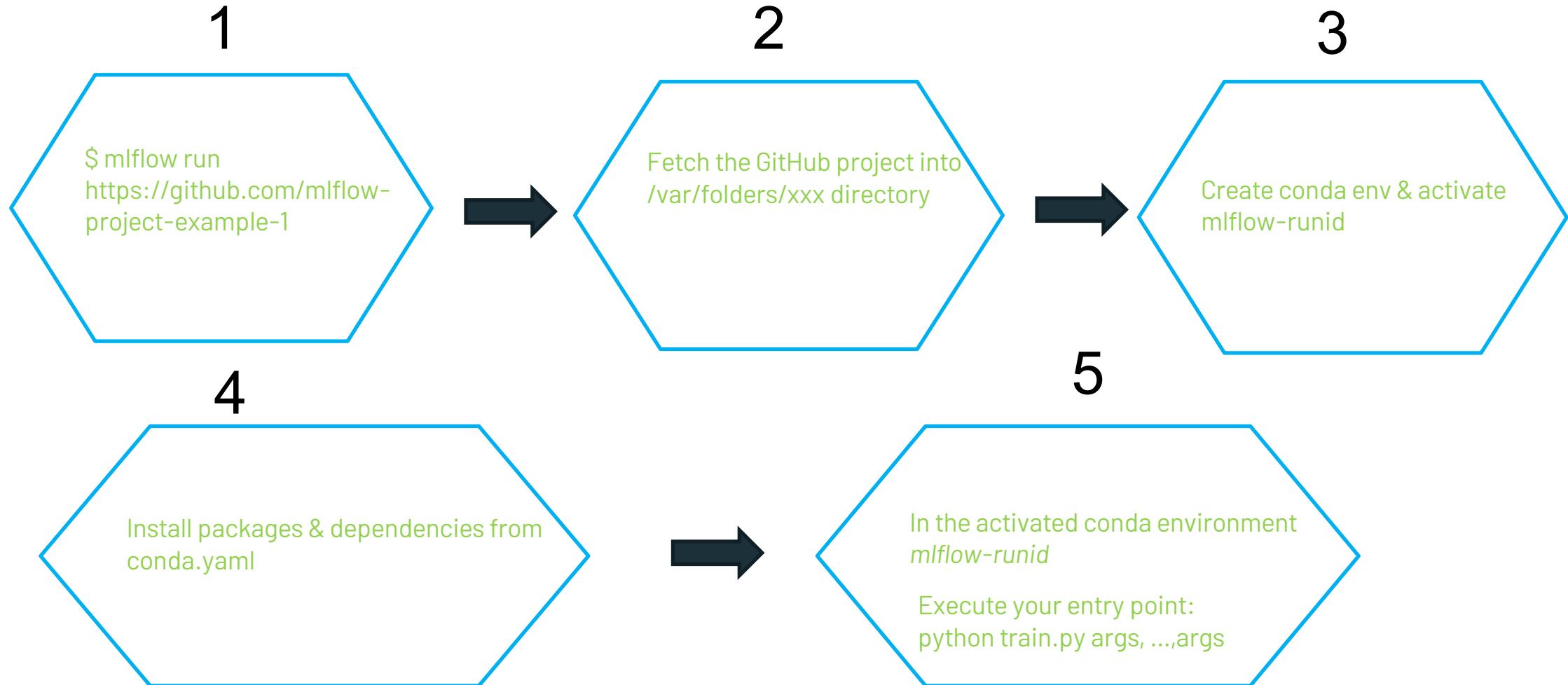
Create MLproject

4

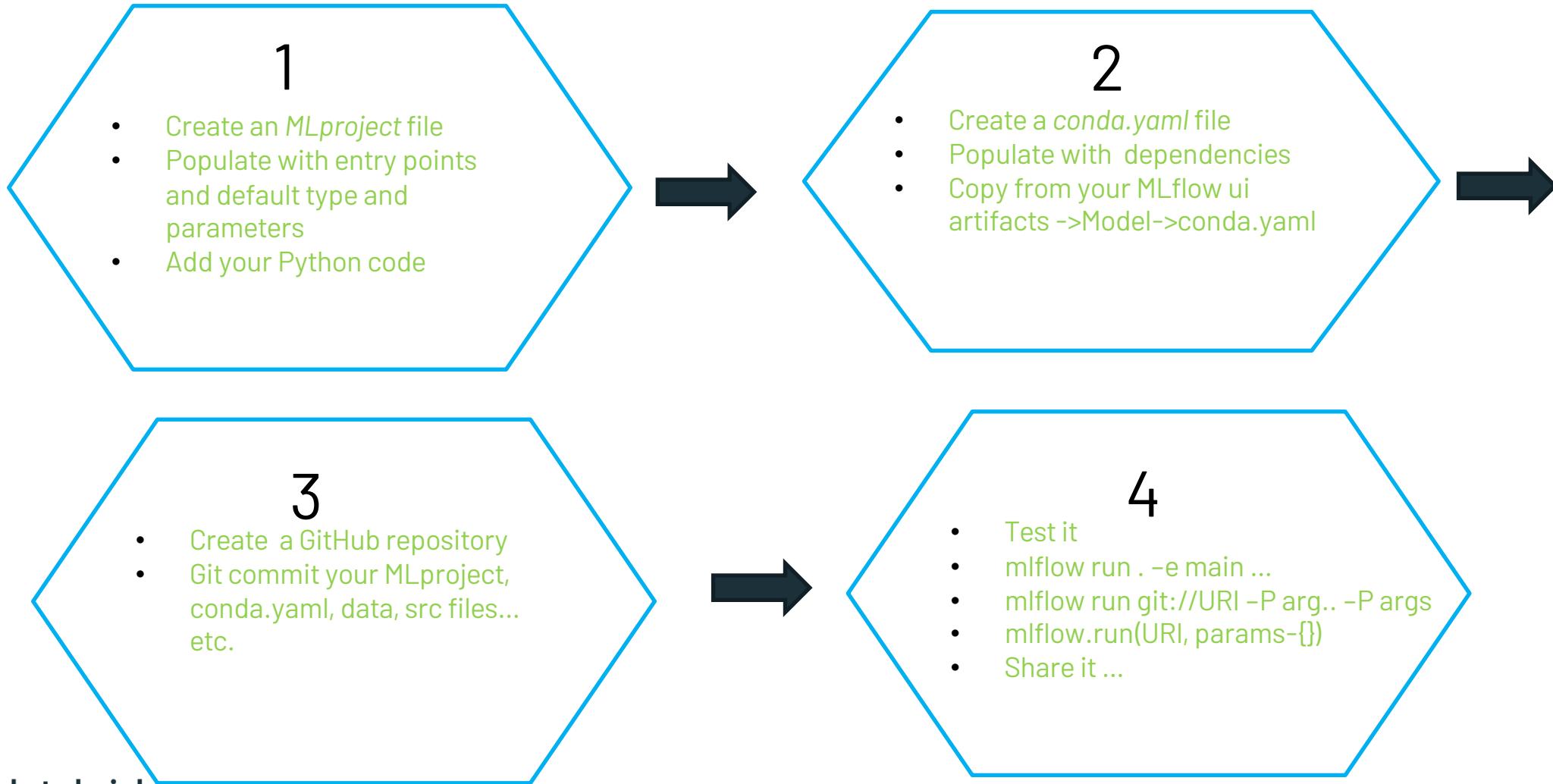
```
mlflow run . -e main
```

<https://github.com/mlflow/mlflow/tree/master/examples/docker>

Anatomy of MLflow Project Execution



How to build an MLflow Project



MLflow Projects Usage

- Reproducibility and Experimentation ✓

- Reproduce experiments in a target environment
 - Experiment with different hyperparameters
 - Use of CLI and Programmatic interface

- Shareability ✓

- Use other people's models and experiment via a MLFlow Project GitHub

- Flexibility ✓

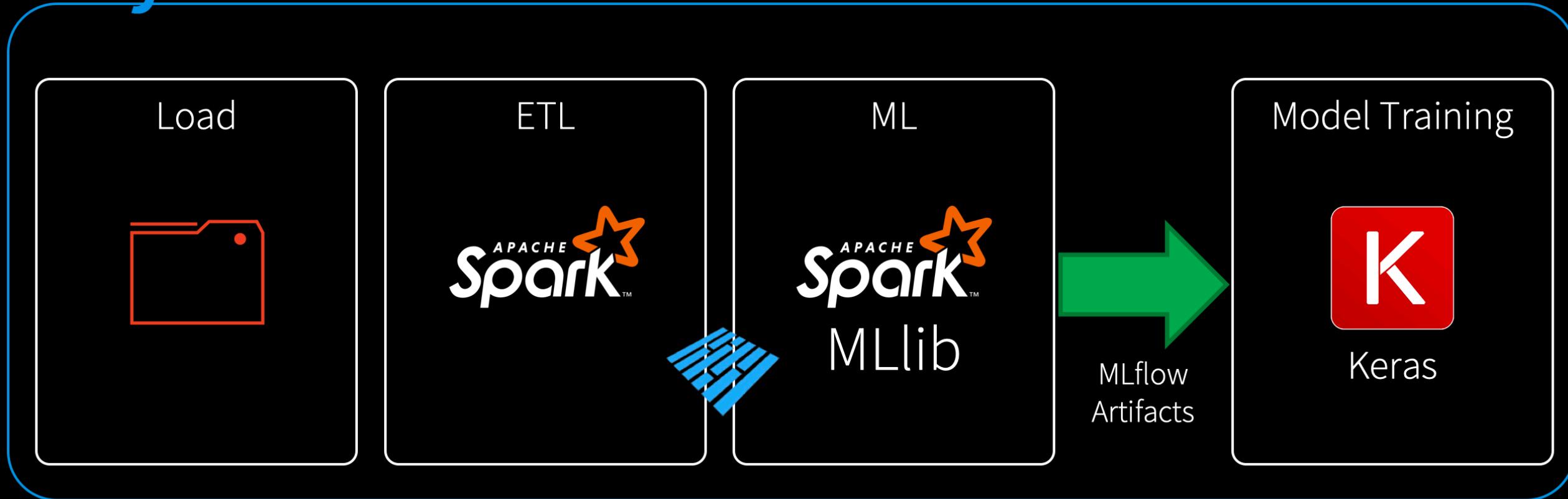
- Execute on three different software environments
 - Conda
 - Docker
 - System

- Extensibility

- Create complicated Project Workflows

MLflow Project: Create Multi-Step Workflow

mlflow



https://github.com/mlflow/mlflow/tree/master/examples/multistep_workflow

```
1 name: multistep_example
2
3 conda_env: conda.yaml
4
5 entry_points:
6     load_raw_data:
7         command: "python load_raw_data.py"
8
9     etl_data:
10        parameters:
11            ratings_csv: path
12            max_row_limit: {type: int, default: 100000}
13        command: "python etl_data.py --ratings-csv {ratings_csv} --max-row-limit {max_row_limit}"
14
15     als:
16        parameters:
17            ratings_data: path
18            max_iter: {type: int, default: 10}
19            reg_param: {type: float, default: 0.1}
20            rank: {type: int, default: 12}
21        command: "python als.py --ratings-data {ratings_data} --max-iter {max_iter} --reg-param {reg_param} --rank {rank}"
22
23     train_keras:
24        parameters:
25            ratings_data: path
26            als_model_uri: string
27            hidden_units: {type: int, default: 20}
28        command: "python train_keras.py --ratings-data {ratings_data} --als-model-uri {als_model_uri} --hidden-units {hidden_units}"
29
30 main:
31    parameters:
32        als_max_iter: {type: int, default: 10}
33        keras_hidden_units: {type: int, default: 20}
34        max_row_limit: {type: int, default: 100000}
35    command: "python main.py --als-max-iter {als_max_iter} --keras-hidden-units {keras_hidden_units}
36                                --max-row-limit {max_row_limit}"
```

MLflow Project Documentation



Search

Documentation > MLflow Projects

Edit on GitHub



Quickstart

Tutorials and Examples

Concepts

MLflow Tracking

- MLflow Projects

Overview

+ Specifying Projects

+ Running Projects

Iterating Quickly

Building Multistep Workflows

MLflow Projects

An MLflow Project is a format for packaging data science code in a reusable and reproducible way, based primarily on conventions. In addition, the Projects component includes an API and command-line tools for running projects, making it possible to chain together projects into workflows.

Table of Contents

- [Overview](#)
- [Specifying Projects](#)
- [Running Projects](#)
- [Iterating Quickly](#)
- [Building Multistep Workflows](#)

<https://mlflow.org/docs/latest/projects.html>

MLflow Projects

Tutorials: <https://github.com/dmatrix/olt-mlflow>

Thank you! 😊

Q & A

jules@databricks.com
@2twitme

<https://www.linkedin.com/in/dmatrix/>