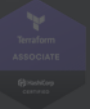


Simplifying Code Distribution with Databricks Asset Bundles



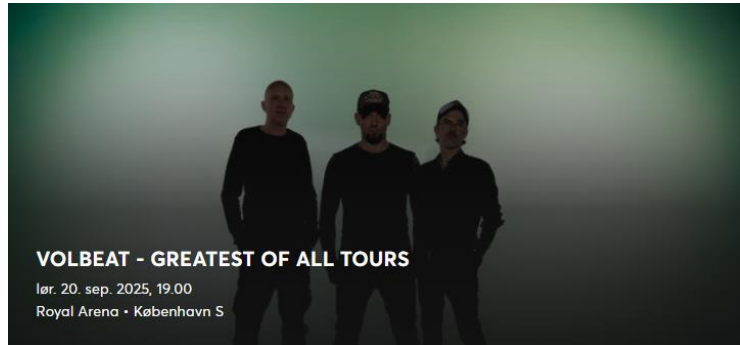
Premium sponsors



Standard sponsors



Raffle Prizes





Falek Miah

Principal Data Engineering
Consultant



15+ Years Microsoft Data Analytics



Intensive Data Engineering &
Analytics Experience



Data, Cloud & DevOps Enthusiast



Microsoft Azure, Databricks (Spark),
Terraform (HashiCorp) certified





Session Scope

Session Scope

- ❖ Good **software engineering is critical** for Data & AI teams
- ❖ These **principles should be adopted** for Data & AI pipeline & workloads
- ❖ Most **successful customers** apply software engineering best practices to their data solutions.
- ❖ To **build reliable and efficient solutions** by applying software engineering best practices:
 - ❖ Code Review
 - ❖ Testing
 - ❖ Automated Deployment
 - ❖ Environment Separation

In this session:

- Code Distribution
- Python Wheels
- Databricks Asset Bundles

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56

def __init__(self):
    self.file = ...
    self.fingerp...
    self.logduper...
    self.debug = ...
    self.logger...
    if path:
        self.fi...
        self.fi...
        self.fi...

@classmethod
def from_setti...
    debug = se...
    return cl...

def request_s...
    fp = seli...
    if fp in...
        retu...
    self.fir...
    if self...
        sel...

def request...
    return
```

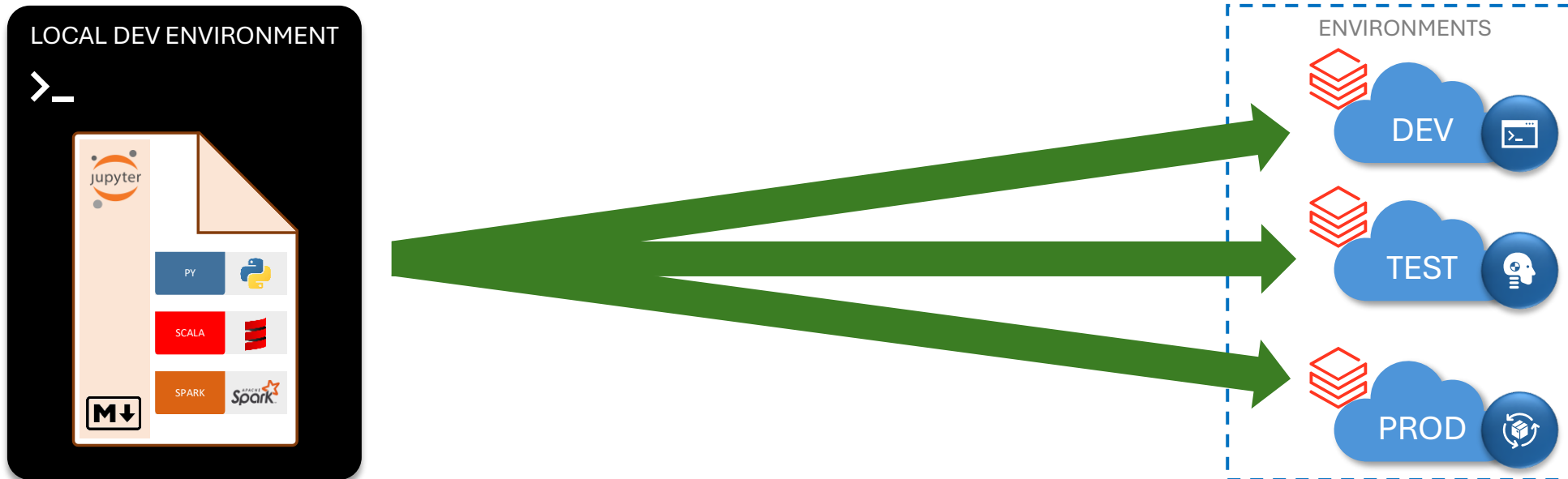


Code Distribution

Code Distribution



- ❖ Share Code Easily
- ❖ Improves Collaboration
- ❖ Ensures Consistency
- ❖ Facilitates Version Control
- ❖ Reduce Errors
- ❖ More Accessible, Maintainable & Scalable



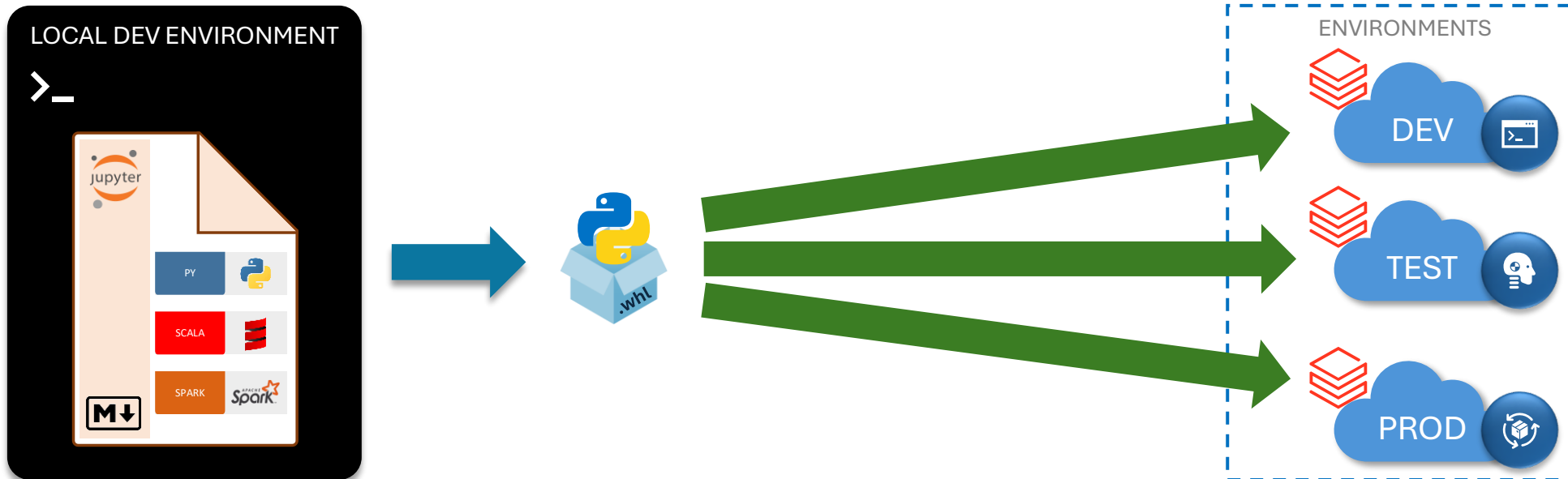
A background image of a terminal window with a dark background and light-colored text, tilted at an angle, suggesting code execution or development.

Python Wheel

Python Wheel

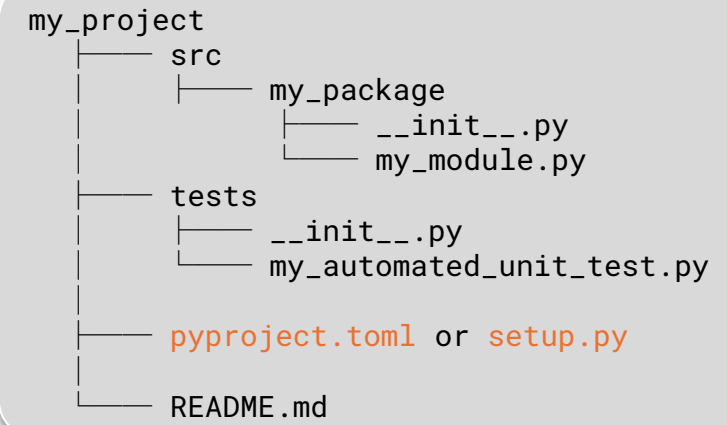
- ❖ Pre-Compiled Package
- ❖ Binary Format
- ❖ Single Objects Easy to Deploy
- ❖ Strong Community Support

- ❖ Keeps Code Structured
- ❖ Reuse Code Across Multiple Projects
- ❖ Manage Dependencies for Consistent Code
- ❖ Effective Testing Options
- ❖ Integrate with CI/CD Deployment Pipeline



Python Wheel - Configuration

❖ Project Directory Structure



```
my_project
├── src
│   └── my_package
│       ├── __init__.py
│       └── my_module.py
├── tests
│   ├── __init__.py
│   └── my_automated_unit_test.py
├── pyproject.toml or setup.py
└── README.md
```

Python Wheel - Configuration



❖ Project Directory Structure

❖ Dependency Management:

❖ SetupTools

- ❖ Traditional approach
- ❖ Manual setup
- ❖ Needs multiple files – (setup.py, requirements.txt)
- ❖ Basic dependency handling

❖ Poetry

- ❖ Modern and recommended approach
- ❖ Commands to setup
- ❖ Easier to read, manage and more repeatable
- ❖ Better dependency handling

```
# Create New Project  
py -m poetry new my_project
```

```
# Add Dependencies/ Packages  
py -m poetry add pytest
```

```
my_project  
├── src  
│   └── my_package  
│       ├── __init__.py  
│       └── my_module.py  
├── tests  
│   ├── __init__.py  
│   └── my_automated_unit_test.py  
├── pyproject.toml or setup.py  
└── README.md
```

```
[project]  
name = "my_project_app"  
version = "0.1.0"  
description = "A example python package"  
authors = [  
    {name = "me", email = "me@email.co.uk"}  
]  
readme = "README.md"  
requires-python = ">=3.11"  
dependencies = [  
    "pytest"  
]  
  
[tool.poetry]  
packages = [  
    {include = "my_project_app", from = "src"}  
]  
  
[build-system]  
requires = ["poetry-core>=2.0.0,<3.0.0"]  
build-backend = "poetry.core.masonry.api"
```



Demo

- Develop Python Wheel

```
self.debug = debug
self.logger = logging.getLogger(__name__)
if path:
    self.file = open(os.path.join(job_dir, 'fingerprint.log'), 'a')
    self.file.seek(0)
    self.fingerprints.update({
        path: fingerprint

    })

    @classmethod
    def from_settings(cls, settings):
        debug = settings.getbool('debug')
        return cls(job_dir(settings), debug)

    def request_seen(self, request):
        fp = self.request_fingerprint(request)
        if fp in self.fingerprints:
            return True
        self.fingerprints.add(fp)
        if self.file:
            self.file.write(fp + '\n')

    def request_fingerprint(self, request):
        return request_fingerprint(request)
```



Python Wheel – Deployment

- ❖ Manual Deployment via UI
- ❖ Automated Deployment using CI/CD Pipelines
- ❖ Deploy via Databricks Asset Bundles (DAB)

Configuration Notebooks (0) Libraries Event log Spark UI Driver logs Metrics Apps Spark compute UI - Master

Filter libraries

| Status | Name |
|--------------------------|----------|
| <input type="checkbox"/> | com.data |
| <input type="checkbox"/> | com.micr |
| <input type="checkbox"/> | pyodbc |

Install library [Send feedback](#)

Library Source ⓘ

☐ Workspace ☐ File Path/ADLS ☒ PyPI ☐ Maven ☐ CRAN ☐ DBFS

⚠ We recommend specifying an exact version of the library with == to prevent regressions. [Learn more](#)

Package

PyPI package (simplejson or simplejson==3.8.0)

Index URL ⓘ

Optional

Cancel Install

```
stages:
- stage: Build
  displayName: "Build Wheel"
  jobs:
    - job: BuildWheel
      displayName: "Build Python Wheel"
      steps:
        - task: UsePythonVersion@0...

        - script: |
            python -m pip install --upgrade pip
            pip install poetry
            poetry install --no-dev
            poetry build
            displayName: 'Build Python Package with Poetry'

        - task: PublishBuildArtifacts@1...

- stage: Deploy
  displayName: "Deploy to Databricks"
  dependsOn: Build
  jobs:
    - job: DeployWheel
      displayName: "Deploy Python Wheel to Databricks"
      steps:
        - task: DownloadBuildArtifacts@0...

        - task: AzureKeyVault@2...

        - script: |
            pip install databricks-cli
            WHEEL_PATH=$(ls $(Build.ArtifactStagingDirectory)/python-wheel/*.whl)
            databricks fs cp $WHEEL_PATH dbfs:/FileStore/wheels/ --overwrite
            env:
              DATABRICKS_HOST: $(databricksHost)
              DATABRICKS_TOKEN: $(databricksToken)
            displayName: "Upload Wheel to DBFS"

        - script: |
            WHEEL_FILE=$(ls $(Build.ArtifactStagingDirectory)/python-wheel/*.whl |
            databricks libraries install --cluster-id $(databricksClusterID) --whl
```

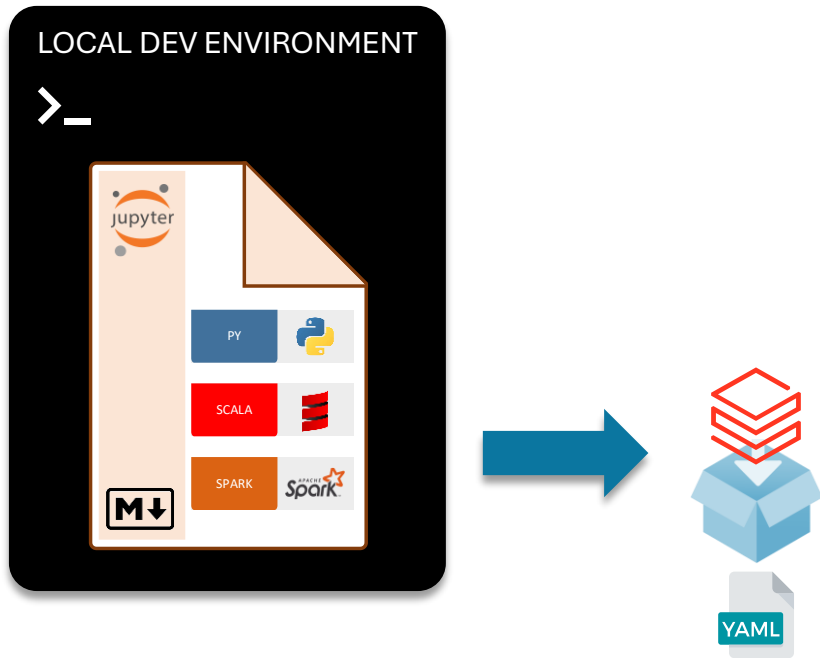


Databricks Asset Bundles (DAB)

Databricks Asset Bundles (DAB)



- ❖ Similar to Like Python Wheels

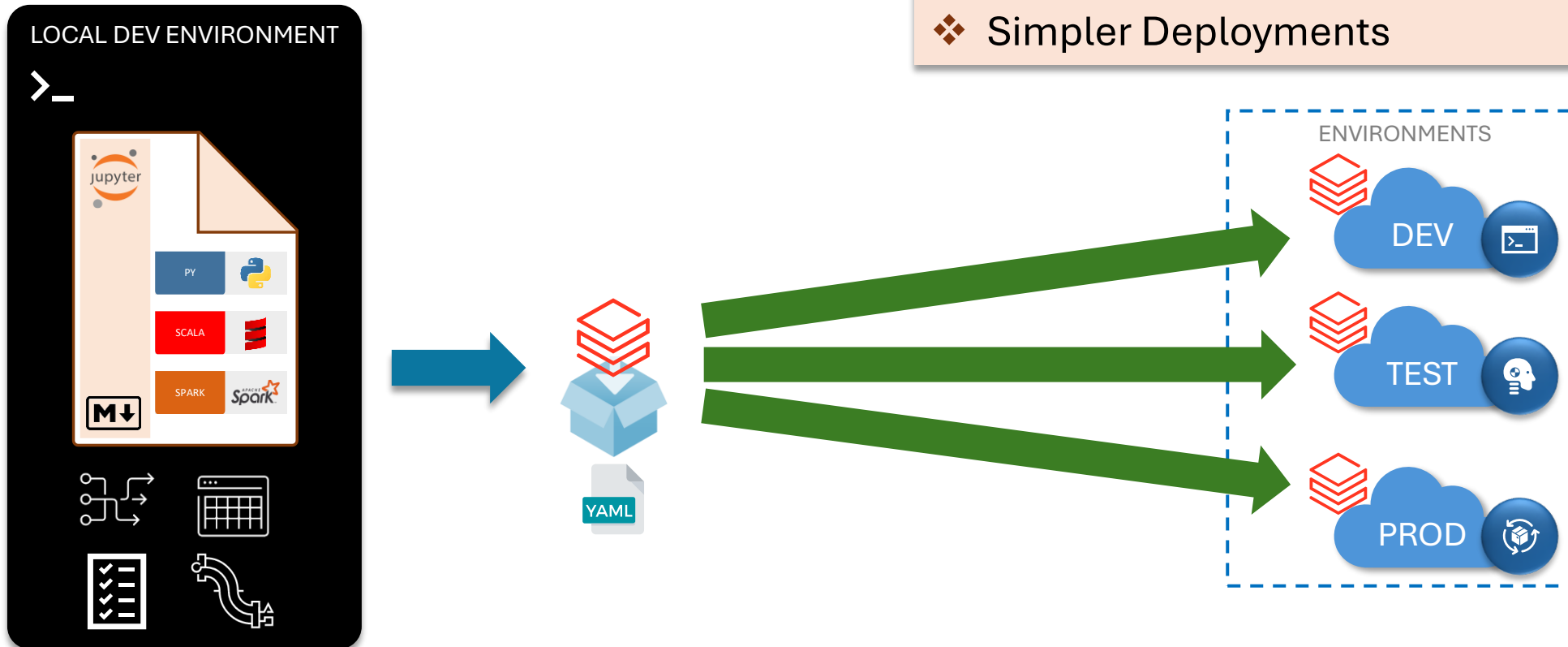




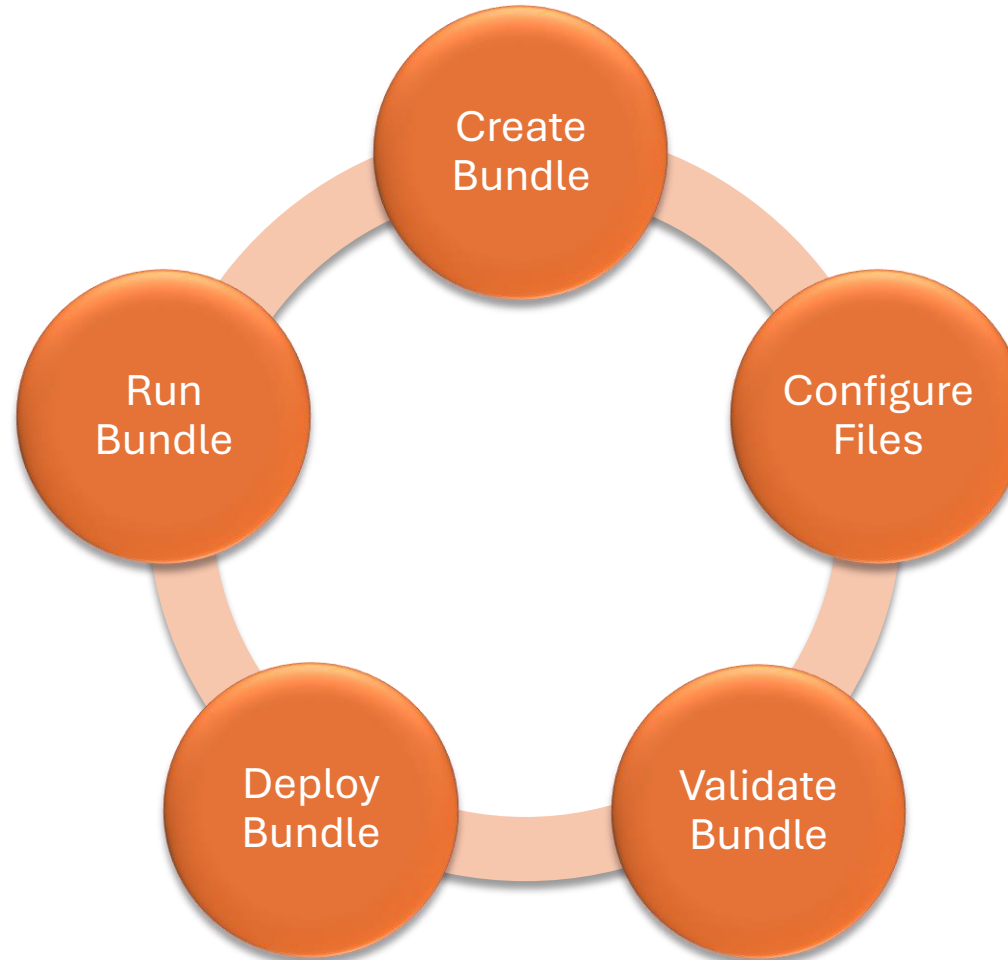
Databricks Asset Bundles (DAB)

- ❖ Similar to Like Python Wheels
- ❖ Deploy Workflows Pipelines, Jobs, DLT etc.

- ❖ Infrastructure-as-Code (IaC) & YAML
- ❖ Configure & Deploy a single Package
- ❖ Ensures Consistency
- ❖ Improves Collaboration
- ❖ Simpler Deployments

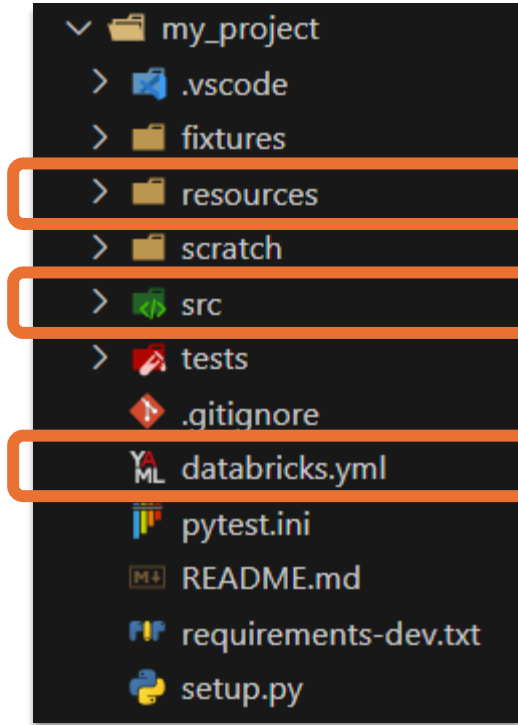


Databricks Asset Bundles – Overview



Uses Databricks CLI
To Validate, Deploy, and Run bundles

Databricks Asset Bundles – Create Bundle



Holds **Databricks Assets** like jobs and pipelines

Stores the **source code** (notebooks, python scripts, DLT etc.)

Contains the bundle **configuration & deployment**

`databricks bundle init`

Create
Bundle

Demo

- Create Databricks Asset Bundles



Databricks **Asset Bundles** vs Terraform



- ❖ Similarity to Terraform

 - ❖ DABs run Terraform code behind the scenes

- ❖ When should you use **Terraform vs Databricks Asset Bundles** for deployment?

 - ❖ DAB is to **manage** Databricks at **projects-level**

 - ❖ Terraform is **manage** Databricks at **infrastructure-level**

https://medium.com/@alexott_en/terraform-vs-databricks-asset-bundles-6256aa70e387



- ❖ **Infrastructure Setup** – Workspaces, Metastores, Catalogs, and Cloud resources.

- ❖ **Manages Access Control** – Users, groups, and resources permissions.

- ❖ **Great for Platform Teams** – Setting up the foundation & components for the DBX projects.



- ❖ **Project-Level Deployment** – Notebooks, Jobs, Workflows, Pipelines, and project configs.

- ❖ **Environment Promotion** – Promoting code & artifacts between environments

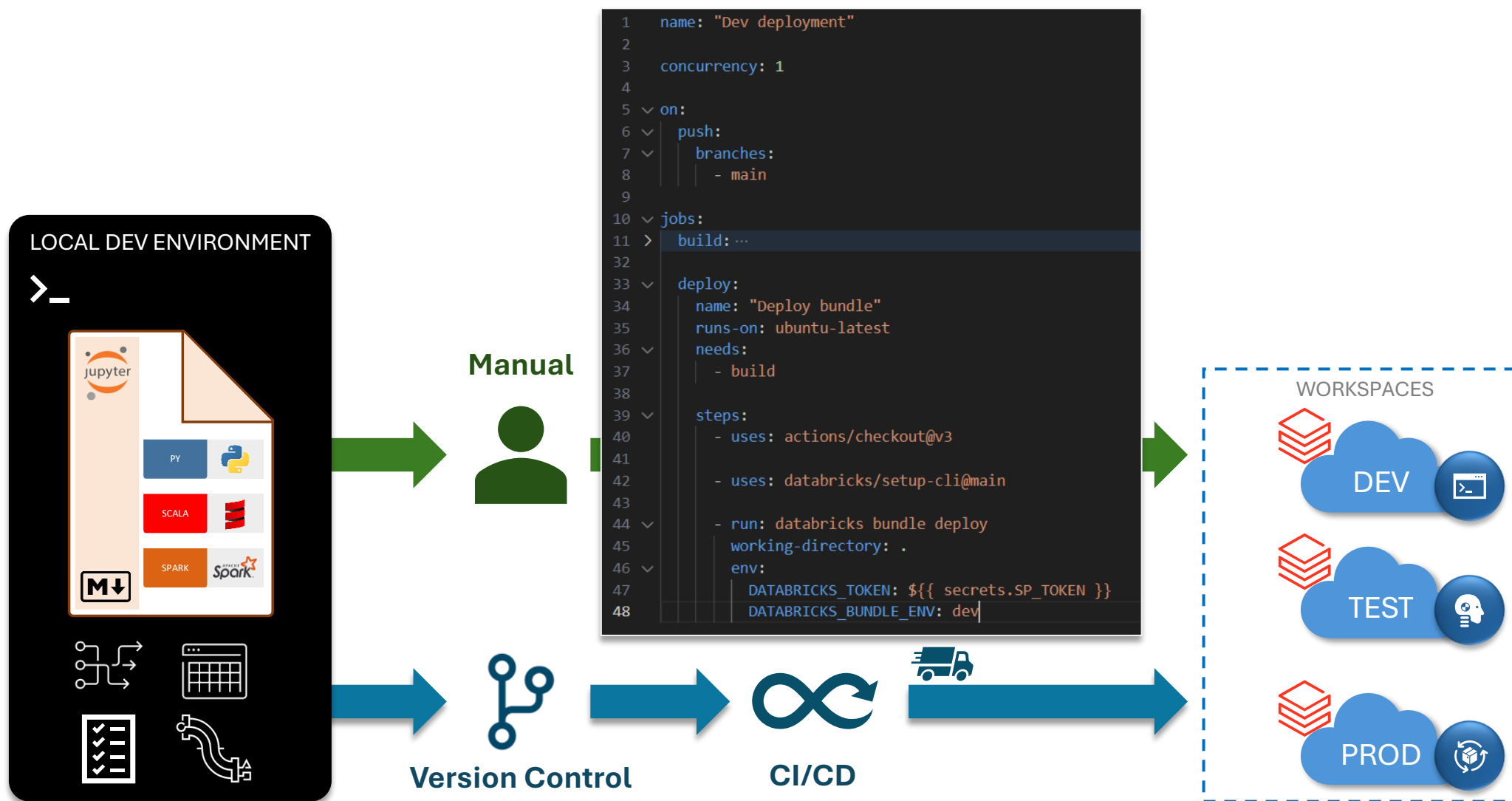
- ❖ **Great for Data & AI Teams** – Deploying and managing code without touching core infra.



Databricks Asset Bundles – Deployment

❖ Deployment methods

- ❖ Manually deploy using IDEs or terminals
- ❖ Automate deployment using CI/CD Pipelines





Final Thoughts & Takeaways

Final Thoughts & Takeaways



- ↻ Package your code
- ↻ Use Asset Bundles for end-to-end deployment
- ↻ Keep it clean and consistent
- ↻ Promote with confidence
- ↻ Know the deployment strategy & tools
- ↻ Automate deployments



Databricks
Asset
Bundles





Thank You

