Generalization of Model Pipeline Tasks

Table of Content

- Introduction
- Sample Repository structure
- Steps during pipeline execution (Batch Model)
- Overall approach and goals
    - Choices of invoking the task code from Azure Pipeline
- High-Level workflow
- Parameters/variables passing
- FAQs

Introduction

This page provides a high-level overview of the workflow that will be followed when templated model repository comes into picture.

Sample Repository structure

The code block below shows a draft repository structure of the templated model repository

Templated Repo Structure

```
eac_model_template
|
|-src
|   |-batch_model
|   |    |-import_notebook.py
|   |    |-create_start_cluster.py
|   |    |-install_libraries.py
|   |    |-create_start_job.py
|   |-rest_model
|   |    |-deploy_to_acs.py
|   |    |-deploy_to_aks.py
|   |-utils
|   |    |-get_cluster_id.py
|   |-scripts
|   |    |-create_repo.sh
|   |    |-apply_branch_policies.sh
|   |    |-create_team.sh
|-pipelines
|   |-ci_pipeline.yml
|   |-train_pipeline.yml
|   |-evaluate_pipeline.yml
|   |-deploy_pipeline.yml
|   |-base_variables.yml
|- tests
|   |-unit_test.py
|-demo_model
|        |-model_variables.yml
|   |-model_modules
|               |-requirements.txt xor environment.yml
|       |-training.py
|       |-scoring.py
|-docs
|   |-getting_started.md
```

Steps during pipeline execution (Batch Model)

1. Import notebooks
2. Create cluster
3. Start cluster
4. Install libraries
5. Create job

6. Run job

Overall approach and goals

Starts with the part of Azure Databricks related tasks.

- Use Python for the tasks like API calls, pipelining operations, etc.
- Better long-term maintenance
  - Unit testable
- Better integration with the AzureML Python SDK that will be used in the future
- To minimize the effort for new model onboarding

Choices of invoking the task code from Azure Pipeline

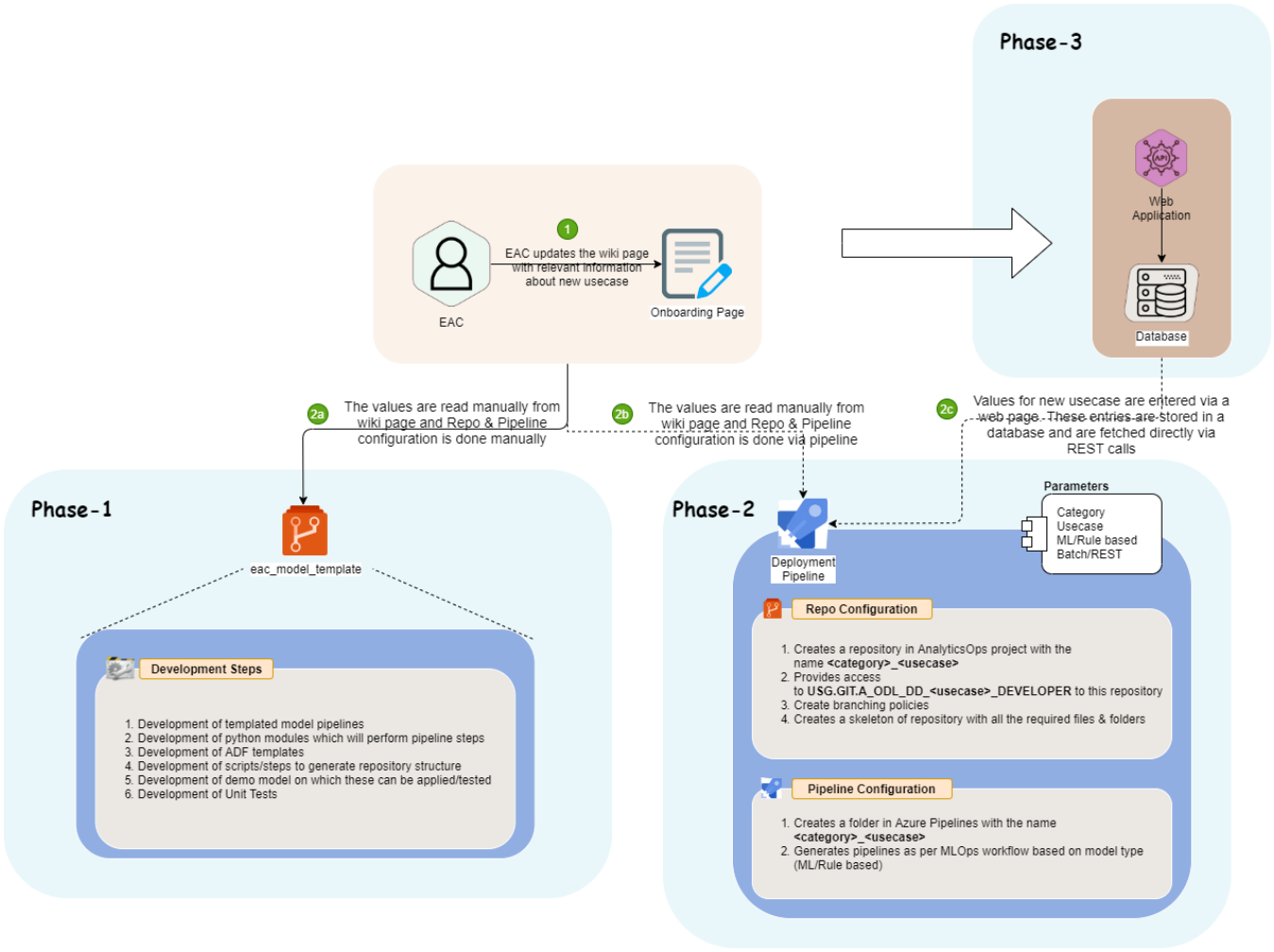
- Separate scripts for different tasks, e.g., *python import_notebooks.py --host $(adb.host) –token $(auth.token)*
  - The set of scripts should be part of the template repo, and be included in each model's repo
- A single CLI that groups different tasks,
  - e.g.,
    - `mlocli notebooks import_notebooks --host $(adb.host) –token $(auth.token)`
    - or group with other services like AzureML, ***mlocli azureml deploy –arg arg***
  - The CLI can be part of the template repo or as a standalone project, and be included or installed as a requirement in each model's repo

As an iterative approach, we can start with the scripts for different tasks and then develop towards CLI.

High-Level workflow

The below diagram shows the phase-wise approach for the implementation of the Generalized model repository. There are in total 3 phases.

- **Phase-1 -** Focus remains on the development of templated pipelines and reusable modules. The Repo and Pipeline configuration is done manually here.
- **Phase-2 -** Workflow remains the same as in phase-1 only the Repo and Pipeline configuration is done via a pipeline.
- **Phase-3 -** This phase focuses on having a web app where instead of updating the values on wiki, the updates are made to the page which stores the information in a database. This will make things a lot easier for eg. - when we fetch the values for the required parameters.



Parameters/variables passing

- The ADB host and token, etc. will be passed within the pipeline to Python script
- Other configs (those will be modified and controlled by different data science teams, e.g., model name, model version) will be read directly from config files, either JSON or YAML

For more details on variables please refer to the "Variables" section of this link.

FAQs

Where will this repository be hosted?

It will be present in AnalyticsOps project.