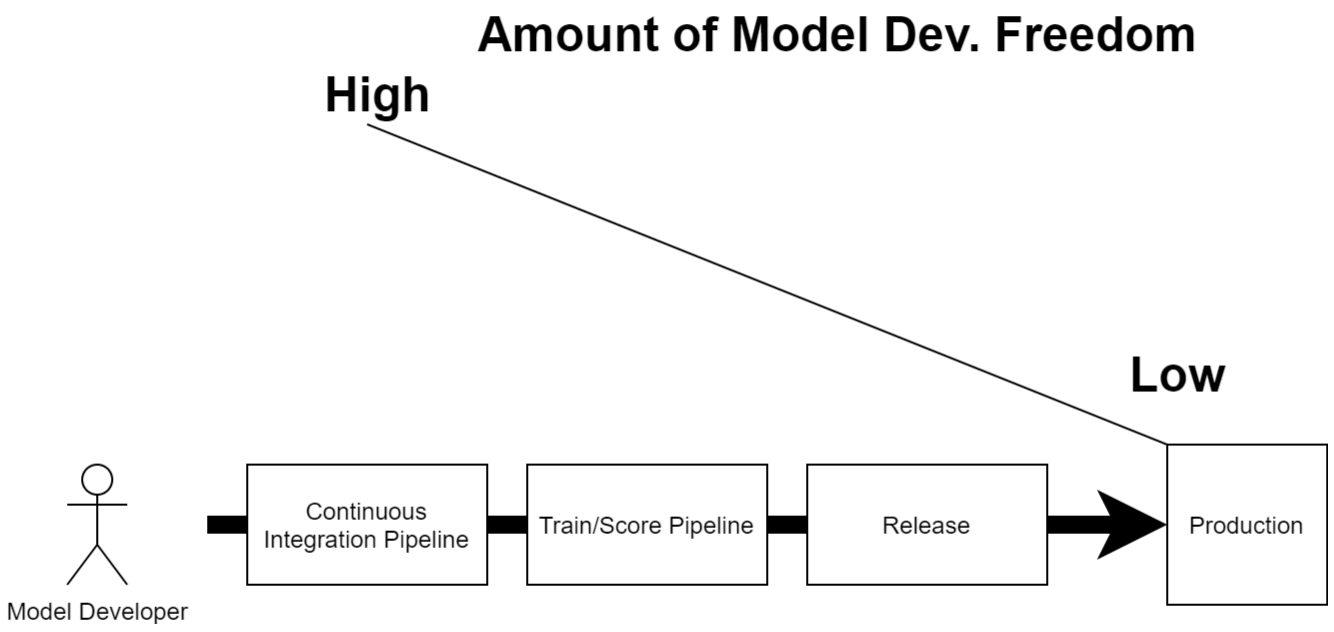
Proposal for MLOps - Separation of Responsibilities

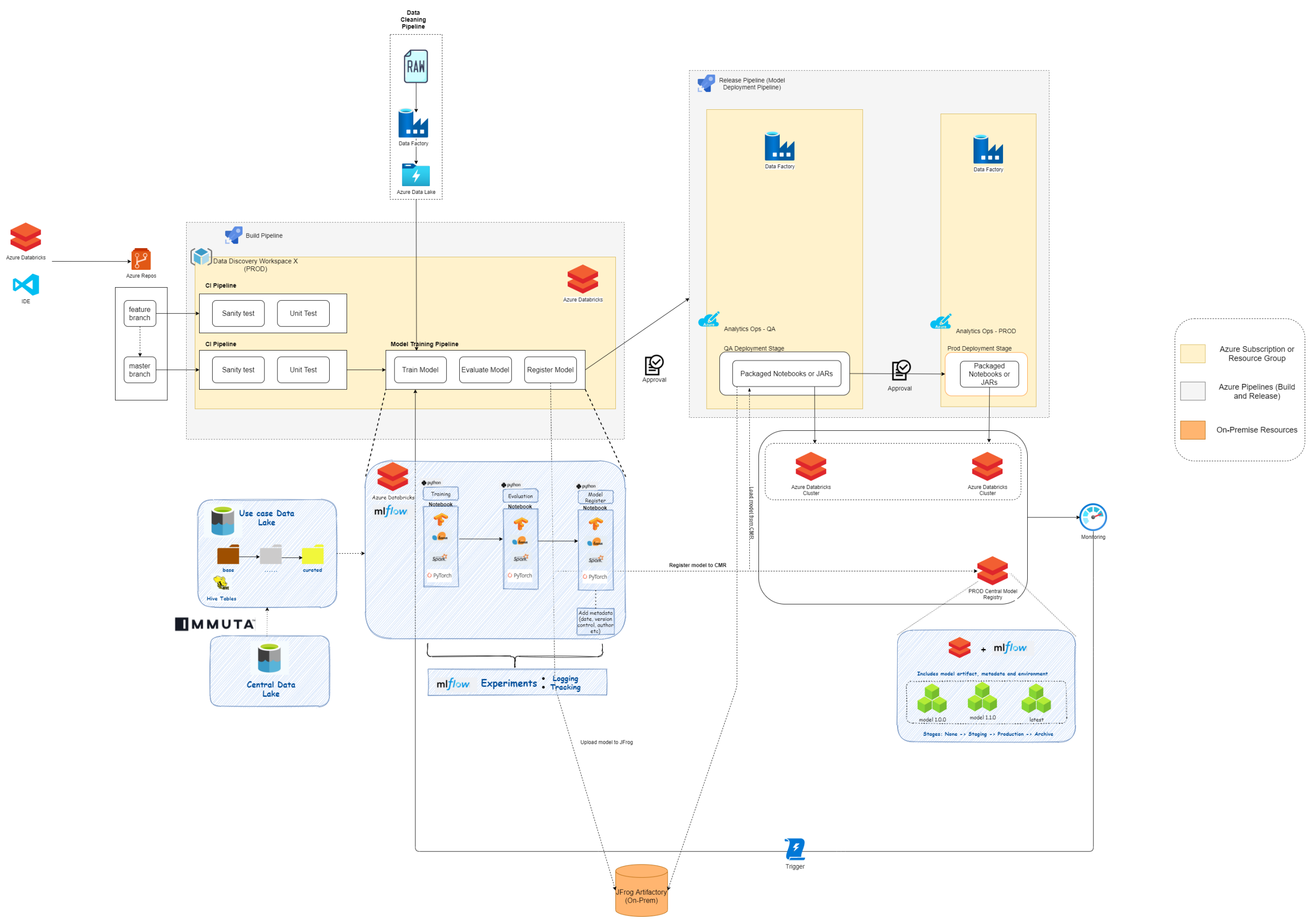
This page houses drafts which give further design details to responsibilities that must be held by GBI rather than the end users.. focus especially on the RELEASE pipeline as well as train/evaluation

Philosophy

Freedom and openness is a cornerstone of Swedbank. So too must it be here in MLOps at EAC. That being said, some guardrails need to be in place to care for models so that they land in a good way on production. One idea is the funnel of freedom gets tighter from left to right.. and human-based approvals are in place for now in order to allow some review of work done in less structured areas such as Data Discovery Workspaces.



ML Model CICD Workflow (Draft Proposal)



Templates

- Naming convention on templates - suggestion "template-<ci>-<job>-???"
- Use ***.yml*** instead of *.yaml* as extension
- Use **underscores** instead of dashes in filenames

Variables

As a general design, we should keep the variable or data structures separated from function (pipeline). In this section, we will present the variables only not associated with a specific task.

Side note:

On variable VS variable groups

- Variable will be repo or pipeline depend which is the use case for model variables.
- On the opposite side variable groups may very well suit GBI enforced choices since those should be defined by their commonality

Variables

```
#GBI Variable group:
#Authentication parameters for each environment:
WORKSPACE_URL_QA
WORKSPACE_URL_PROD
OAUTH2_TOKEN_URL
OAUTH2_ID
OAUTH2_SECRETS_NAME
SPN_QA
SPN_PROD

#Those are cluster Variable Valid across multiple environment
DATABRICKS_CLUSTER_NAME_QA (non empty string)(if null use a default value)
DATABRICKS_CLUSTER_NAME_PROD (non empty string)(if null use a default value)
DATABRICKS_CLUSTER_NAME_TST (non empty string)(if null use a default value)



#CLuster parameters
CLUSTER_SPARK_VERSION (string)(if null use a default value)
CLUSTER_NODE_TYPE_ID (string)(if null use a default value)
CLUSTER_DRIVER_NODE_ID (integer)(if null use a default value)
CLUSTER_AUTOTERMINATION_MINUTES (integer)(if null use a default value)


#We will probably have a default location
MODEL_SRC_PATH (string)(if null use a default value) (optional)
NOTEBOOK_PATH (string)(if null use a default value) (optional)

CMR_SCOPE
CMR_Prefix


#Model Variables:
#Since those values are "opened", I added a specification to those values in case we want to verify the
correctness.
# Main key of the model
MODEL_NAME (non empty string) (no default value) # probably the same as USE_CASE
TEAM_NAME (non empty string) (no default value)
ENVIRONMENT_FILE (non empty string) (Valid values : conda xor pip, default : conda) (optional)
CLUSTER_MIN_WORKER_NODE  (int) (min and max need to be defined) (optional)
CLUSTER_MAX_WORKER_NODE (int) (min and max need to be defined) (optional)
MODEL_TYPE (enum) (valid values: Supervised, Unsupervised, rulebased) (default if not set : Supervised)
(optional)
ARTIFACTORY_REPOSITORY_MODEL (non empty string)(optional) # if specified and if the model does need to go in
artifactory.
IMAGE (non empty string)(optional) # In case we have a specific image to use to use the model. I can imagine
cases where building the image with conda or pip takes so long time that creating a shortcut could help.


# Those are needed to the scheduling part
# The dataset part is still unclear because, we do not know what dataset information is needed for the
scheduling part: either a dataset package from Metamodel or a dataset definition from ADF.
DATASET_NAME (non empty string) (no default value)
# The dataset template file is a legacy from AMMF which specify the table. It will be used to pass parameters
to the model at run time.
DATASET_TEMPLATE_FILE (non empty string) (no default value)
# From my understanding, the ADF part needs to exists and is created with the web interface. Once automated we
may not need this part.
ADF_JOB (non empty string) (no default value)
# Business information
FINANCIAL_ID
CATEGORY (non empty string)
```

CI Pipeline tasks

Creation of Templates on these major Steps

1 - Required to Do

2

3

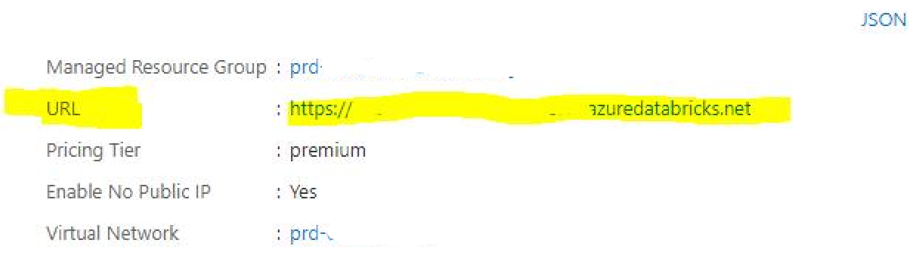
4

5 - Completely Optional

- ## Stage Model_CI
  - Job - Sanity Tests
    - **1 - (GBI) -** Verify Model Folder Structure - Doable with pipeline pure syntax
    - **1 - (GBI) -** Verify train.py and train function exist (optional for supervised learning) - Doable with pipeline pure syntax
    - **1 - (GBI) -** Verify score.py + evaluate and score functions (ASTs? https://docs.python.org/3/library/ast.html) - Need a python script to do that.
    - **? - (GBI) -** Verify Variable Groups? (Unclear right now what these are needed for at a data scientist level)
    - **(Model Dev) -** Pylint standards execution (as of now no central coding standard exists in Swedbank.. could start with simple PEP-8 or suggestions from customer, could vary customer to customer)
    - **3 - (GBI) -** Verify Pylint execution & publishing
    - **1 - (GBI) -** Check EAC or Azure Pipeline Config files & proper values for required variables (non-empty strings, bad inputs) to execute?? Doable with pipeline pure syntax
      - Do we provide prefilled key-value pairs for CMR? for Release Target Databricks Workspace?
    - **2 - (Model Dev) -** Ask or metadata related to Models Financial ID.. this can later automate requesting of service principle
    - **1 - (GBI) -** Check the presence of the service principal and show instructions if not filled in with instructions - Doable with pipeline pure syntax
  - Job - Unit Tests
    - **(Model Dev) -** Execute UnitTests (Consumers are expected to write and manage their tests and test results)
    - **1 - (GBI) -** Verify Execution Unit Tests & Publishing Doable with pipeline pure syntax
  - Job - Extra/StretchJobs
    - **4 - (Model Dev) -** Run Bandit ?
    - **1 - (GBI) -** Check library compatibility with Databricks clusters? (DS needs to input which cluster version of databricks they're going to use) - Python needed here
    - **2 - (Model Dev) -** Remind to call out for ML vs non-ML clusters? (via scanning dependencies?)
    - **1 - (GBI) -** Scan for hardcoded keys or tokens - pipeline pure syntax ? I am unsure about that.
    - **3 - (GBI) -** Scan for duplicate code across org?
    - **2 - (GBI) -** Verify README, Verify README has useful information

Model Training/Scoring tasks

- ## Variables Needed
  - **databricks.host** - seems to be set manually on manual execution of the pipeline
    How-to-set: when you execute the pipeline you must manually set this variable which you can get from the Azure portal GUI when you click on the databricks workspace as seen here



JSON

Managed Resource Group	: prd· [redacted]
URL	: https:// [redacted] ızuredatabricks.net
Pricing Tier	: premium
Enable No Public IP	: Yes
Virtual Network	: prd-. [redacted]

  - **(GBI) - NOTEBOOK_PATH** - seems to be set via a template file, seems to need to point to "/shared/_______" but unclear what this does
  - **workspaceUrl** - uses a sub-set of the databricks.host variable... but is currently hardcoded in places
  - **(GBI) - MODEL_SRC_PATH** - seems to be set via a template file, should point to where the model notebook source is located
  - **(GBI) - TRAIN_JOB_NAME (SCORE_JOB_NAME)** - Used to name the databricks job
  - **(GBI) -** CMR_SCOPE
  - **(GBI) -** CMR_Prefix

- Stage Train_Model (Alternatively "Score_Model")
    - (GBI) - Import the notebooks from git to databricks- Working on bash, conversion to python ongoing
    - (GBI) - Set the language(s) and deploy the notebooks - Working on bash, conversion to python ongoing
    - (Model Dev) - Create / Re-use Databricks Cluster (Users can configure which cluster types they need to use) - Working on bash, conversion to python ongoing
    - (GBI) - Start the cluster - Working on bash, conversion to python ongoing
    - (Model Dev) - Install library dependencies (Will be configurable by the consumer)
    - (GBI) - Create / Retrieve Scoring Job
- Stage Evaluate_Model
    - (GBI) - Verify presence of evaluation function or metrics
    - (Model Dev) - Execute evaluation routine and publish metrics (Possibly help compare results with other runs)
    - (GBI) - Prompt to auto update/publish details of model to IM
- Stage Register_Model
    - (GBI) - Verify model was accepted by data scientist for registration
    - (GBI) - Verify registration of model to Central Model Registry (MLFlow)
    - (GBI) - Execute registration of model or DBC to JFrog
    - (GBI) - Verify correct staging level with model developer (if model is registered, we can upgrade it's state via the model code

Rule-Based Model CICD Workflow (Draft Proposal)

The stages for those model will be the same as above except the training which is not needed. The variable MODEL_TYPE will help to identify the type and help to set up conditionality. This is an explicit process, we could do something implicit like : If there is not train.py then it must be a rule based therefore switch to rule based mode.

References

Variable groups for Azure Pipelines and TFS - Azure Pipelines | Microsoft Docs

cloudpoc_pipeline.yml - Repos (azure.com)

pipeline-variable-template.yaml - Repos (azure.com)

Model Scheduling with ADF - Design Proposal - Enterprise Analytics in Cloud (EAC) - Swedbank IT Wiki