



Módulo 3: MLFlow Model Registry

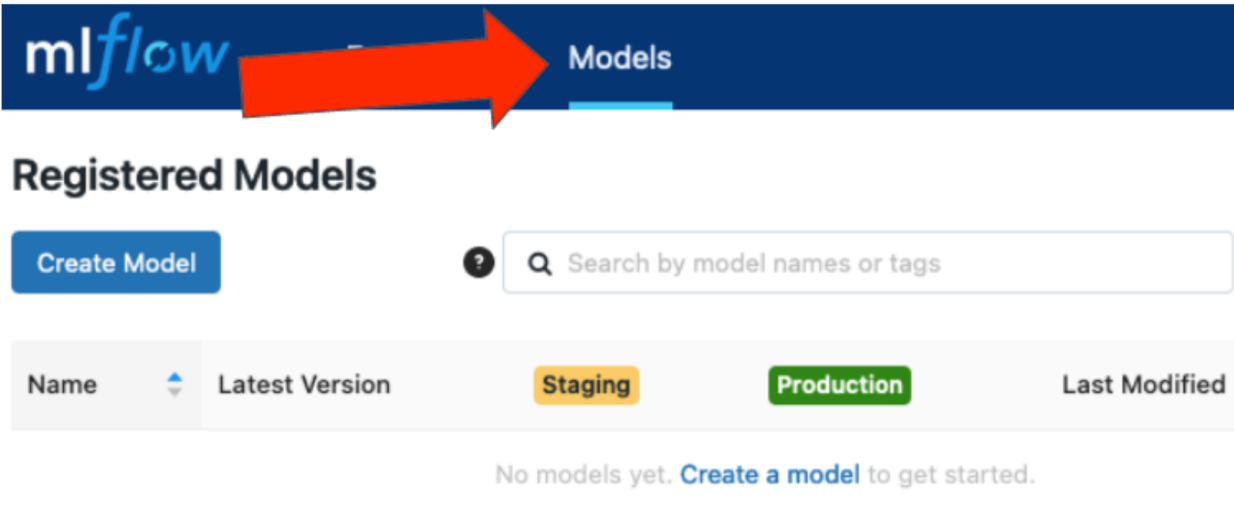
Introduction to MLFlow Model Registry

MLFlow Model Registry proporciona acceso a modelos para colaborar a través de una interfaz o con el módulo de MLFlow Client. Además, brinda una manera de gestionar el ciclo de vida de los modelos a través del versionado y el staging.

Se compone de 4 conceptos:

- **Model:** MLFlow Models que es creado y se lleva un registro en MLFlow Tracking a través de una run o un experimento.
- **Registered Model:** Un modelo es elegible para ser registrado con MLFlow Model Registry, lo que permite llevar un versionado y asignarle una stage.
- **Model Version:** Se incrementa cada vez que se registra un modelo con el mismo nombre.
- **Model Stage:** El modelo se puede asignar a una etapa como Staging, Production, Archived.

Working with the Model Registry



Using MLflow client module

```
# Import from MLflow module
from mlflow import MlflowClient

# Create an instance
client = MlflowClient()

# Print the object
client
```

```
<mlflow.tracking.client.MlflowClient object at 0x101d55f30>
```

Registering a model

```
# Create a Model named "Unicorn"  
client.create_registered_model(name="Unicorn")
```

```
<RegisteredModel: creation_timestamp=1679404160448, description=None,  
last_updated_timestamp=1679404160448, latest_versions=[], name='Unicorn',  
tags={}>
```

Model UI

Registered Models

Create Model

?

Q Search by model names or tags

Search

Clear

Name	Latest Version	Staging	Production	Last Modified	Tags
Unicorn	—	—	—	2023-03-21 09:09:20	—

110 / page

Searching registered models

```
# Search for registered models
client.search_registered_models(filter_string=MY_FILTER_STRING)
```

- Identifiers
 - name - of the model
 - tags - tags associated with model
- Comparators
 - `=` - equal to
 - `!=` - not equal to
 - `LIKE` - case-sensitive pattern match
 - `ILIKE` - case-insensitive pattern match

Example search

```
# Filter string
unicorn_filter_string = "name LIKE 'Unicorn%'"

# Search models
client.search_registered_models(filter_string=unicorn_filter_string)
```

```
[<RegisteredModel: creation_timestamp=1679404160448, description=None,
last_updated_timestamp=1679404160448, latest_versions=[], name='Unicorn',
tags={}>,
<RegisteredModel: creation_timestamp=1679404276745, description=None,
last_updated_timestamp=1679404276745, latest_versions=[], name='Unicorn 2.0',
tags={}>]
```

Registering Models

Un modelo registrado proporciona un control de versiones más similar a las aplicaciones de Software tradicionales; este control de versiones también brinda

una forma de llevar un seguimiento de los cambios a la vez que una manera de cómo revertir dichos cambios.

Ways to register models

```
# Existing MLflow Models
mlflow.register_model(model_uri, name)
```

`model_uri`

- local filesystem
- tracking server

```
# During training run
mlflow.FLAVOR.log_model(name,
    artifact_uri,
    registered_model_name="MODEL_NAME")
```

`registered_model_name="MODEL_NAME"`

Registering model example

```
# Import mlflow
import mlflow

# Register model from local filesystem
mlflow.register_model("./model", "Unicorn")

# Register model from Tracking server
mlflow.register_model("runs:/run-id/model", "Unicorn")
```

```
Registered model 'Unicorn' already exists. Creating a new version of this model...
2023/03/24 14:34:26 INFO mlflow.tracking._model_registry.client:
Waiting up to 300 seconds for model version to finish creation.
Model name: Unicorn, version 1
Created version '1' of model 'Unicorn'.
<ModelVersion: creation_timestamp=1679682866413, current_stage='None',
description=None, last_updated_timestamp=1679682866413, name='Unicorn',
run_id=None, run_link=None, source='./model', status='READY', status_message=None,
tags={}, user_id=None, version=1>
```

```
Registered model 'Unicorn' already exists. Creating a new version of this model...
2023/03/24 14:36:56 INFO mlflow.tracking._model_registry.client:
Waiting up to 300 seconds for model version to finish creation.
Model name: Unicorn, version 2
Created version '2' of model 'Unicorn'.
<ModelVersion: creation_timestamp=1679683016297, current_stage='None',
description=None, last_updated_timestamp=1679683016297, name='Unicorn',
run_id='2e974508b68b45ceb114657c6e97fef5', run_link=None,
source='./mlruns/1/2e974508b68b45ceb114657c6e97fef5/artifacts/model',
status='READY', status_message=None, tags={}, user_id=None, version=2>
```

Registered Models

Create Model

Name	Latest Version
Insurance	—
Insurance2	—
Test Scores	—
Unicorn	Version 2
Unicorn 2.0	—

Unicorn

Created Time: 2023-03-21 09:09:20

> Description [Edit](#)



> Tags

▼ Versions

All

Active 0

Compare

<input type="checkbox"/>	Version	Registered at
<input type="checkbox"/>	 Version 2	2023-03-24 14:36:56
<input type="checkbox"/>	 Version 1	2023-03-24 14:34:26

Logging model

```
# Import modules
import mlflow
import mlflow.sklearn
from sklearn.linear_model import LogisticRegression

# Model
lr = LogisticRegression()
lr.fit(X, y)

# Log model
mlflow.sklearn.log_model(lr, "model", registered_model_name="Unicorn")
```

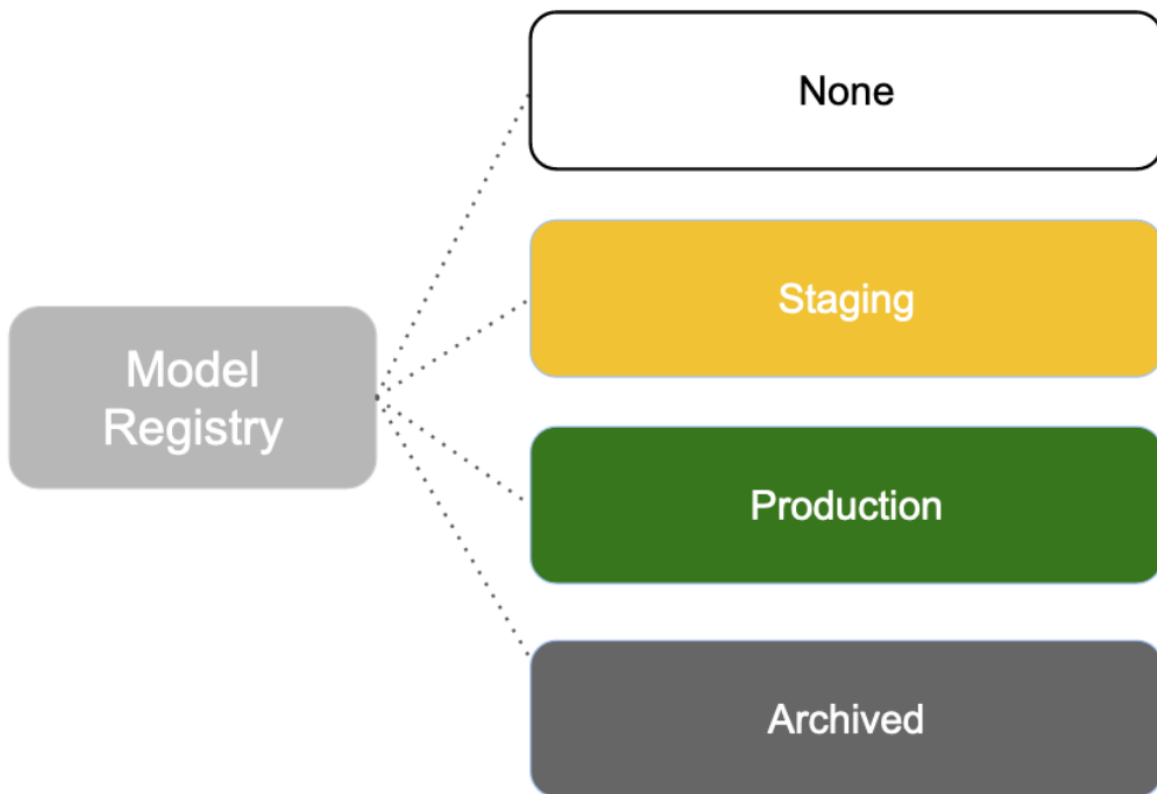
```
Registered model 'Unicorn' already exists. Creating a new version of this model...
2023/03/24 17:31:10 INFO mlflow.tracking._model_registry.client:
Waiting up to 300 seconds for model version to finish creation.
Model name: Unicorn, version 3
Created version '3' of model 'Unicorn'.
<mlflow.models.model.ModelInfo object at 0x14734d330>
```

Model Stages

Para hacer la transición a través de los diferentes ambientes de desarrollo de software, MLFlow proporciona Model Stages. Un Model Stage puede asignarse a una versión de un modelo registrado. Hay unos predefinidos: None, Staging, Production, Archived. Un modelo puede pasar un stage a otro pero solo puede tener asignado un único stage a la vez.

Predefined Stages

Model Stages son usados para representar las diferentes fases en el ciclo de vida del modelo. Cada stage puede considerarse un conjunto de reglas definidas en una organización de cómo lidiar con los entornos de desarrollo de software.



None: Es el stage por defecto y significa que el modelo todavía no ha recibido un stage.

Staging: Se asigna cuando un modelo va a través del testing y la evaluación.

Production: Se asigna cuando un modelo ha pasado todas las pruebas y está listo para ser usado en producción.

Archived: Se asigna cuando un modelo ya no es utilizado.

Transitioning models

Registered Models > Unicorn >

Version 3

Registered At: 2023-03-24 17:31:10

Stage: None ▾

Last Modified Transition to → Staging

Source Transition to → Production

> D Transition to → Archived

```
# Import MLFlow Client
from mlflow import MlflowClient
client = MlflowClient()

# Transition to Staging
client.transition_model_version_stage(
    name="Unicorn",
    version=3,
    stage="Staging"
)
```

```
<ModelVersion: creation_timestamp=1679693470034, current_stage='Staging',
description=None, last_updated_timestamp=1679699050734, name='Unicorn',
run_id='a1454f2865e449f8835f38f71e53e547', run_link=None,
source='./mlruns/1/a1454f2865e449f8835f38f71e53e547/artifacts/model',
status='READY', status_message=None, tags={}, user_id=None, version=3>
```

Model Deployment

Debido a que Model Registry proporciona un repositorio centralizado que ayuda a gestionar los modelos a través de todo el ciclo de vida de desarrollo, también brinda un proceso simplificado de despliegue de modelos.

Puede utilizarse las versiones o los stages.

- Model versions
 - Version +1
- Model stages
 - Staging
 - Production
 - Archived

Registered Models

Create Model

Name	Latest Version	Staging	Production
Insurance	Version 1	Version 1	—
Insurance2	Version 2	—	Version 2
Test Scores	—	—	—
Unicorn	Version 3	Version 3	Version 1

Ways to deploy models

Load model

```
# MLflow flavor
mlflow.FLAVOR.load_model()
```

Serve model

```
# MLflow serve command-line
mlflow models serve
```

Models URI

Para especificar un despliegue de MLFlow Model Registry, se utiliza como convención el URI del modelo, que consiste en el nombre del modelo y la versión o stage.

Convention

```
models:/
```

Model version

```
models:/model_name/version
```

Model stage

```
models:/model_name/stage
```

Load models

```
# Import flavor
import mlflow.FLAVOR

# Load version
mlflow.FLAVOR.load_model("models:/model_name/version")

# Load stage
mlflow.FLAVOR.load_model("models:/model_name/stage")
```

```
# Import flavor
import mlflow.sklearn

# Load Unicorn model in Staging
model = mlflow.sklearn.load_model("models:/Unicorn/Staging")
# Print model
model
```

```
LogisticRegression()
```

```
# Inference
model.predict(data)
```

Serving models

```
# Serve Unicorn model in Production stage
mlflow models serve -m "models:/Unicorn/Production"
```

```
2023/03/26 15:07:00 INFO mlflow.models.flavor_backend_registry:
Selected backend for flavor 'python_function'
2023/03/26 15:07:00 INFO mlflow.pyfunc.backend: === Running command 'exec gunicorn
--timeout=60 -b 127.0.0.1:5000 -w 1 ${GUNICORN_CMD_ARGS} --
mlflow.pyfunc.scoring_server.wsgi:app'
[2023-03-26 15:07:00 -0400] [86409] [INFO] Starting gunicorn 20.1.0
[2023-03-26 15:07:00 -0400] [86409] [INFO] Listening at: http://127.0.0.1:5000
[2023-03-26 15:07:00 -0400] [86409] [INFO] Using worker: sync
[2023-03-26 15:07:00 -0400] [86410] [INFO] Booting worker with pid: 86410
```

Invocations endpoint



<http://localhost:5000/invocations>

- Formats
 - CSV
 - JSON

CSV format

```
pandas_df.to_csv()
```

JSON format

```
{
  "dataframe_split": {
    "columns": ["R&D Spend", "Administration", "Marketing Spend", "State"],
    "data": [["165349.20", 136897.80, 471784.10, 1]]
  }
}
```

Model prediction

```
# Send payload to invocations endpoint
curl http://127.0.0.1:5000/invocations -H 'Content-Type: application/json' -d
{
  "dataframe_split": {
    "columns": ["R&D Spend", "Administration", "Marketing Spend", "State"],
    "data": [["165349.20", 136897.80, 471784.10, 1]]
  }
}
```

```
[[104055.1842384]]
```