Production-Ready, Reproducible, Secure, Cross-Cloud Machine Learning Pipeline

Engineer: Michael Maio

Last updated: 8/17/2025

# Overview

The repo at <https://github.com/MichaelMaio/energy_forecasting> contains a working machine learning pipeline that addresses the following hypothetical scenario that a power company might need to deal with.

**Problem**: Electric car usage is increasing in a certain neighborhood, causing a gradual, year-over-year increase in the hourly peak kilowatt load reported by a sensor on the local transformer.

**Question**: Assuming the power consumption trend remains unchanged, how long before the transformer becomes overloaded?

**Solution**: Build a machine learning pipeline that can process recent trend data and forecast when the transformer may eventually become overloaded, informing the necessary schedule for a preventative upgrade.

This machine learning pipeline uses:

1. Python for the scripting.
2. Docker containers to encapsulate training, promotion, and prediction jobs.
3. MLflow for model management.
4. GitHub Actions to trigger a pipeline deployment whenever a change is merged into main (such as when a data scientist adjusts the model).
5. Terraform to create and update AzureML infrastructure from code.
6. A managed identity to keep everything secure.

#1 through #3 allow for the entire pipeline to be run locally for quick feedback on changes before deploying to the cloud. No Azure required.

#5 allows for other cloud providers, such as AWS or GCP, to be swapped in as needed.

## Caveat

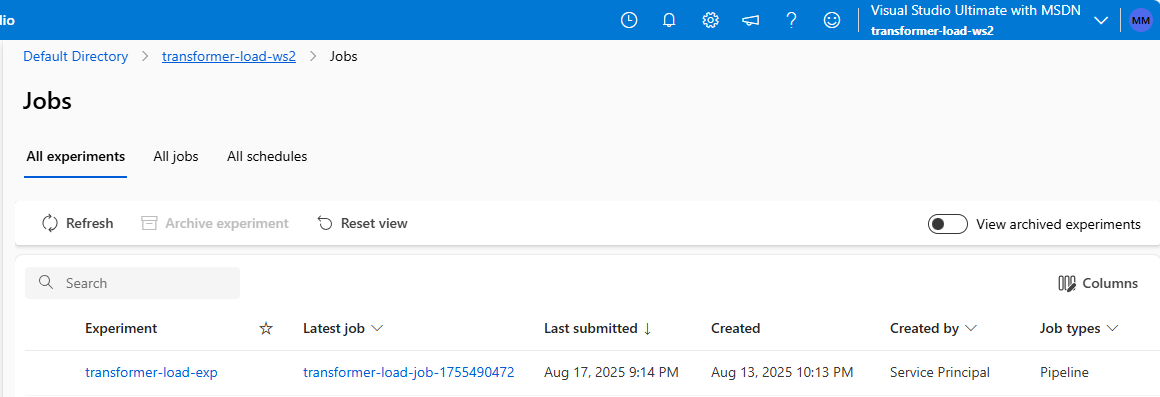
I only spent a few days on this so, though it is a working pipeline and ran multiple jobs successfully (screenshots below), it is of course bare bones and just a proof-of-concept. There are a bunch of features in AzureML and MLflow that I didn’t make use of due to lack of time. Additionally, it’s not the most interesting data science problem since it has very few features that go into the predictions. But it serves the larger purpose of exploring machine learning pipeline infrastructure.

If I had a few weeks or months, I could create a rich, production quality pipeline. But this pipeline is at least production-***ready*** in terms of the way the system is designed and the technologies used. And since it addresses a hypothetical problem for the purposes of experimentation rather than an actual problem that a business is trying to solve, I’ll be spending limited time on it.

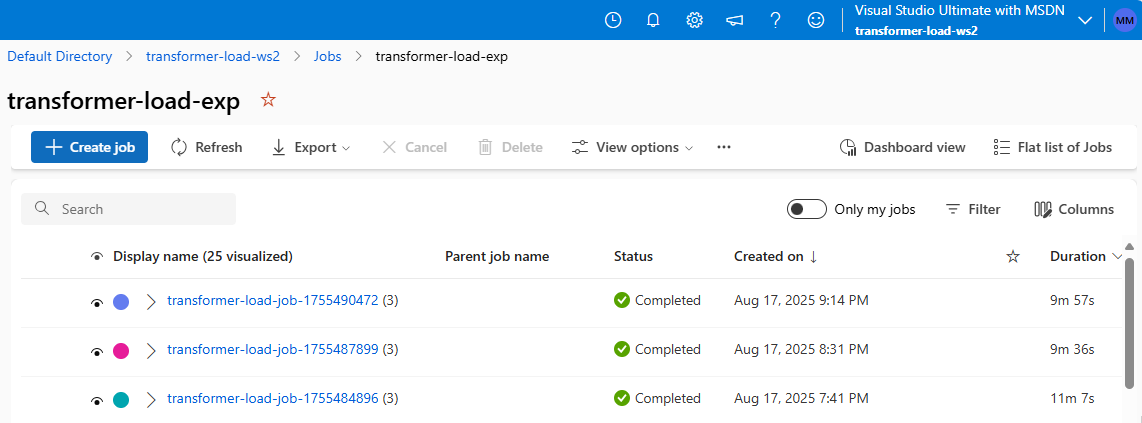
# System Design Diagram (coming soon)

# AzureML In Action

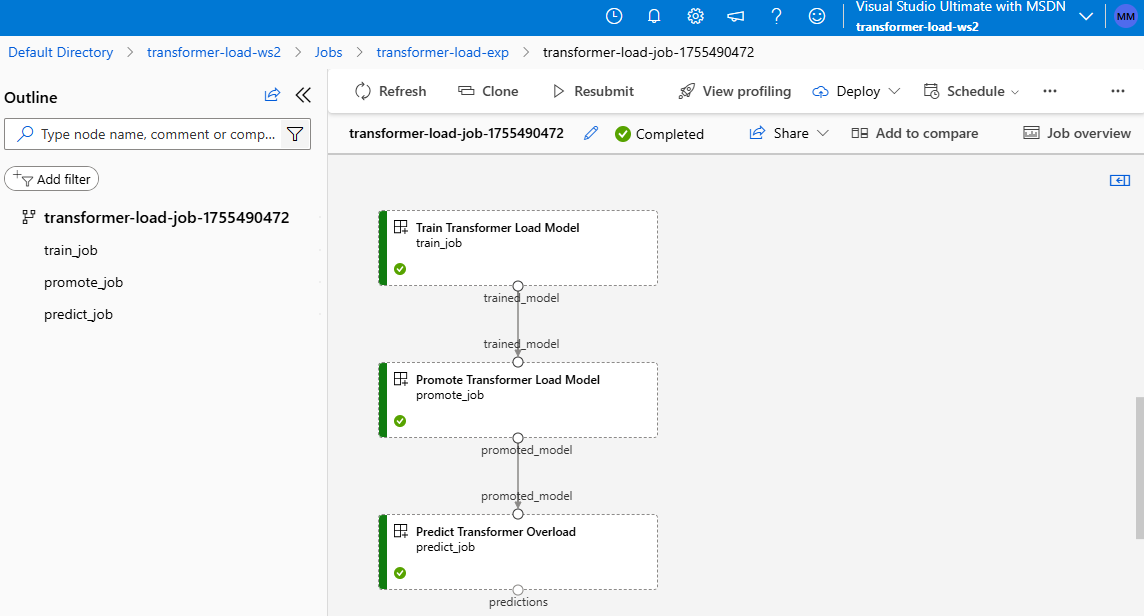
This is the starting point: a high-level view of the experiment in Azure AI’s Machine Learning Studio.



Drilling into the experiment shows a list of its jobs. Each job represents a different deployment of the pipeline that trains the model, promotes the model (if it passed testing), and uses the model to make predictions.

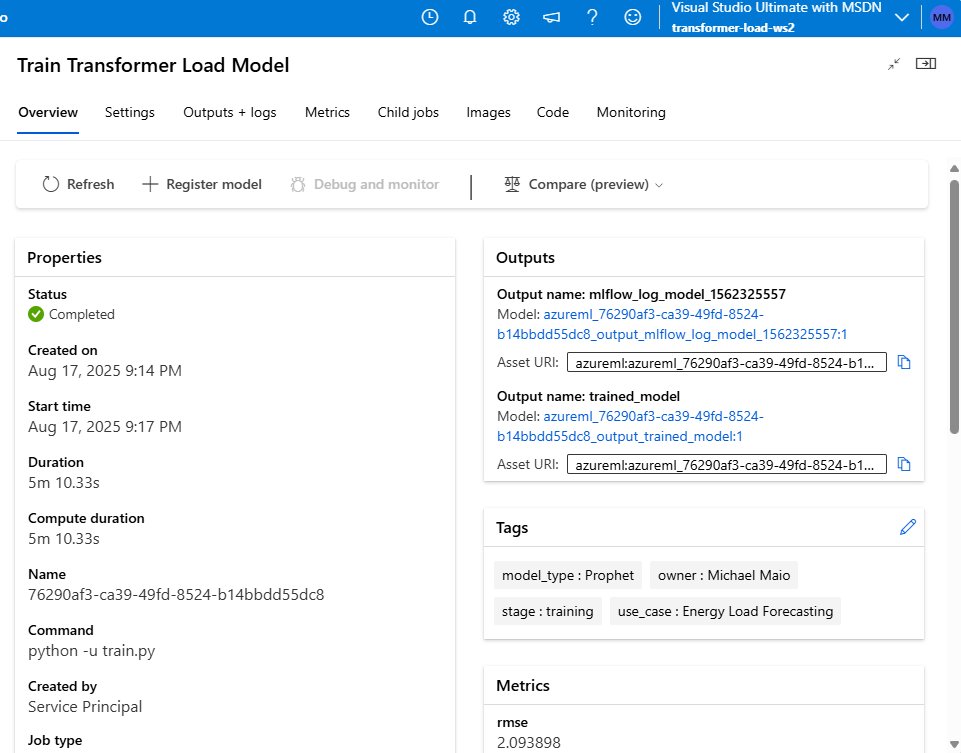


Drilling into the latest job reveals a list of sub-jobs and how they are wired together. Below you can see that the sub-job which trains the model outputs a “trained\_model” to the job that promotes the model, which then outputs a “promoted\_model” to the job that uses the model to output “predictions”.

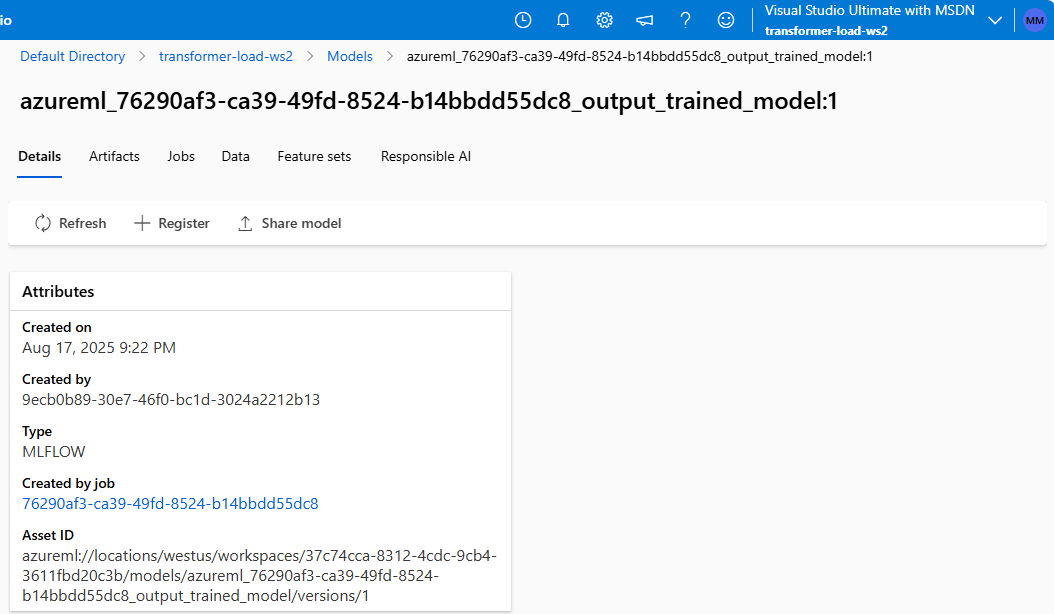


You can drill into each sub-job to view all kinds of details about it. Below you can see that the first sub-job, “Train Transformer Load Model”, did the following:

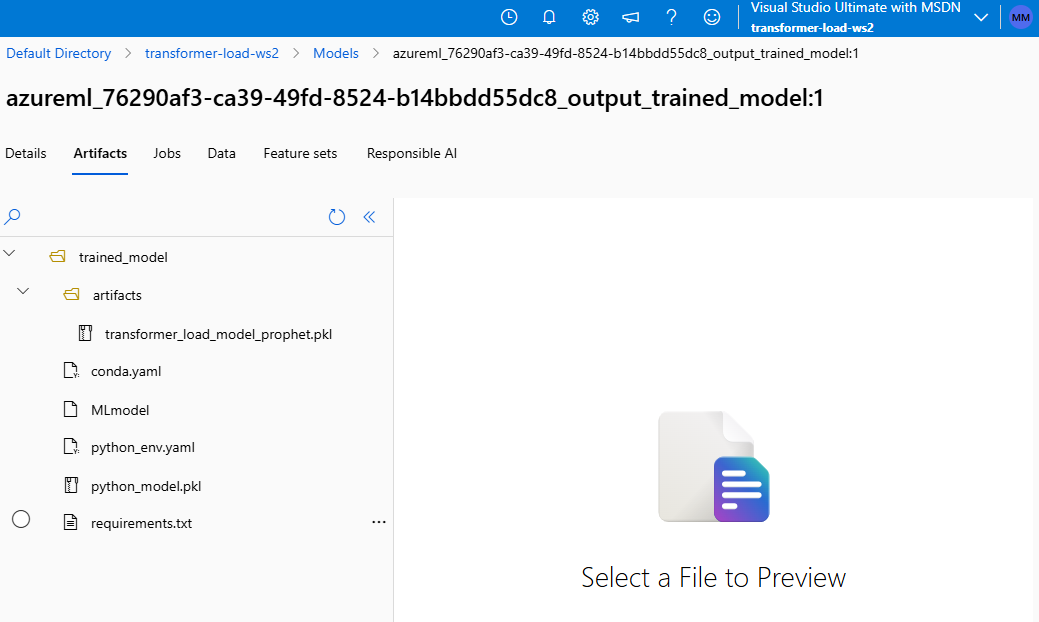
1. It output the model; once when MLflow logged the model and once to pass the model along to the promotion job.
2. It applied some informative tags.
3. It reported the metric “rmse” (aka Root Mean Squared Error), indicating how well the model performed during testing.



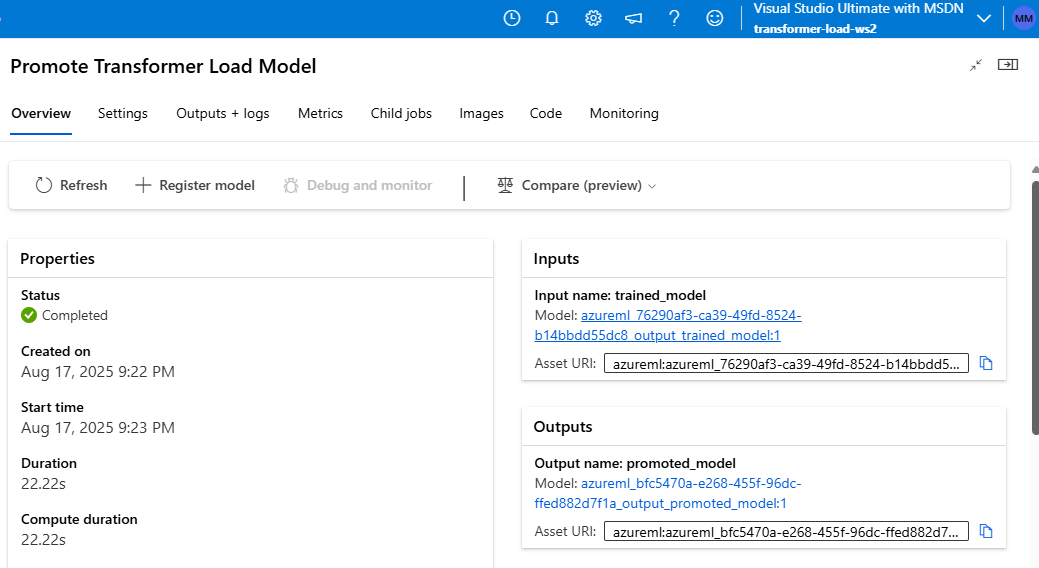
You can drill into one of the model links to get more information on the model.

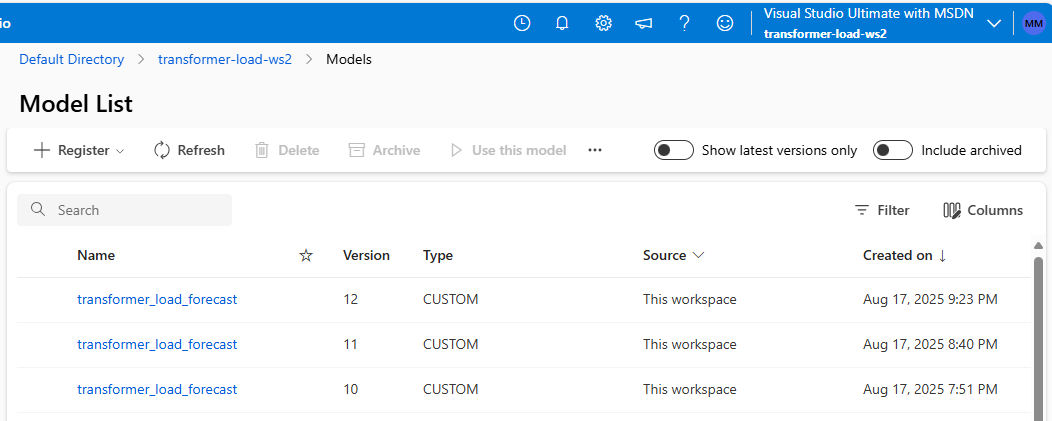


And drill into its artifacts.



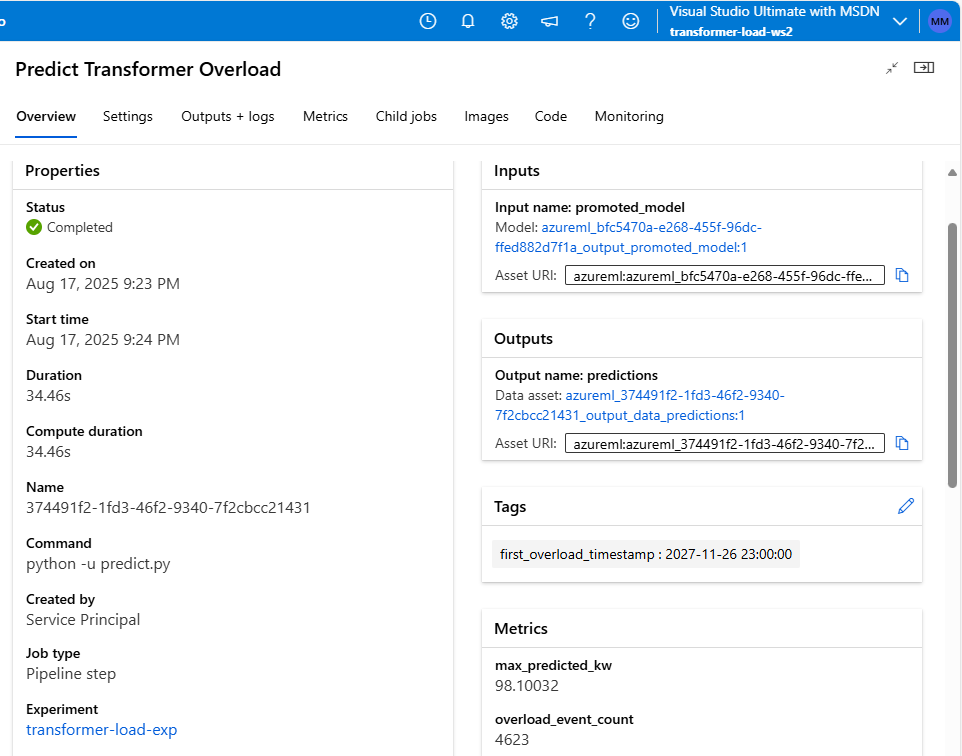
Moving on to the “Promote Transformer Load Model” sub-job, you can see that it output the “promoted\_model”, meaning the model passed testing during training and the Root Mean Squared Error of the model was sufficiently low for it to be useful in making predictions.



If you view the AzureML model registry for the workspace, you can see that the promotion sub-job registered the model since it passed testing.

Moving onto the “Predict Transformer Overload” sub-job, we can see that it created the following:

1. A tag reporting that the transformer is predicted to hit its first overload at 11pm on November 26th, 2027.
2. A metric predicting that the maximum load over the entire 5-year period will be about 98 kilowatts.
3. A metric predicting that the transformer will overload over 4,623 times in the next 5 years given current usage trends.



You can also drill into the “predictions” output and see the files that the prediction job uploaded, including:

1. The predicted transformer load in kilowatts for each hour during the next 5 years.
2. The number of times the transformer is predicted to overload during that period.
3. A chart of the predicted loads.

