

End-to-end machine learning operations (MLOps) with Azure Machine Learning

This document serves as my personal tutorial guide, detailing the steps I executed to implement a complete Machine Learning Operations (MLOps) pipeline for a diabetes prediction model. The workflow covers my approach to environment setup, model training, and continuous delivery (CI/CD) on Azure Machine Learning (AML). In this learning path I have completed 7 challenges to complete the task.

Challenge 0: Convert a notebook to production code

My Understanding:

In this challenge I converted a Jupyter notebook to production ready code. I understood that when I store these scripts then it is easier to automate the code execution so I can parameterize scripts to easily reuse the code for retraining.

In this challenge I have created a new public repo by navigating to <https://github.com/MicrosoftLearning/mslearn-mlops> and selecting use this template feature.

And in that in the experimentation folder, I found a Jupyter Notebook that trains a Classification model and the data used by this notebook is in the experimentation/data. And in the src/model folder I found train.py that is used for training

Challenge 1: Create an Azure Machine Learning job

My Understanding:

In this challenge I understood that to automate machine learning workflows I can define machine learning tasks in scripts. And to execute those workflows which have Python scripts, I must use Azure Machine Learning jobs. And Azure Machine Learning jobs store all metadata of a workflow, including input parameters and output metrics. By running scripts as jobs, it's easier to track and manage machine learning models

In this challenge, I found a YAML file to define the job. And I created an Azure Machine Learning workspace and a compute instance and ran job using CLI (v2).

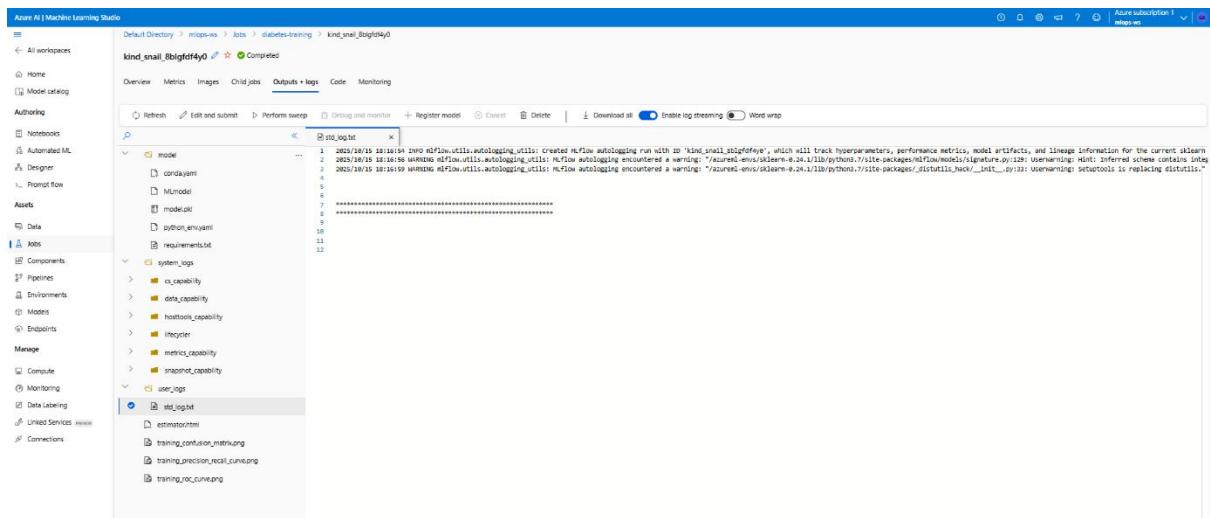
Result: A successfully completed job in the Azure Machine Learning workspace. The job should contain all input parameters and output metrics for the model you trained.

Azure Machine Learning Job details:

The screenshot shows the Azure Machine Learning Studio interface. On the left, the navigation sidebar is visible with sections like All workspaces, Home, Model catalog, Authoring, Assets, Data, and Jobs (which is selected). The main content area displays the properties of a completed job named 'kind_snail_8blgfd4y0'. The job status is 'Completed'. It was created on Oct 15, 2025, at 11:40 PM, started at 10:25 AM, and took 4m 0.6980s. The command run was 'python train.py --training_data \${inputs.training_data} --reg_rate \${inputs.reg_rate} # Correct parameter syntax'. The environment used is 'AzureML-sklearn-0.24-ubuntu18.04-py37-cpu49'. The registered model is 'Diabetes_Model1'. The Git repository is 'https://github.com/Ganesh-Esc/MLOps-by-Microsoft'. The job type is 'Command'. The inputs section shows 'training_data' as a Data asset named 'diabetes-dev-folder1'. The outputs section shows 'mlflow_log_model_725690645' as a Model asset. The tags section includes 'estimator_class : sklearn.linear_model.LogisticRegression' and 'estimator_name : LogisticRegression'. The parameters section lists various configuration options for the Logistic Regression model.

This screenshot shows the Metrics tab for the same job. The metrics listed are training_accuracy_score, training_f1_score, training_log_loss, training_precision_score, training_recall_score, training_roc_auc_score, and training_score. Their values are displayed as follows:

| Metric | Value |
|--------------------------|-----------|
| training_accuracy_score | 0.6928571 |
| training_f1_score | 0.6058648 |
| training_log_loss | 0.6071673 |
| training_precision_score | 0.6880648 |
| training_recall_score | 0.6928571 |
| training_roc_auc_score | 0.6214107 |
| training_score | 0.6928571 |



Challenge 2: Trigger the Azure Machine Learning job with GitHub Actions

My Understanding:

By using CLI I can run Machine Learning job from anywhere. Using a platform like GitHub will allow us to automate Azure Machine Learning jobs. And to trigger the job to run, I can use GitHub Actions.

In the .github/workflows folder, I found the 02-manual-trigger.yml file. This file defines a GitHub Action which can be manually triggered. The workflow checks out the repo onto the runner, installs the Azure Machine Learning extension for the CLI (v2), and logs in to Azure using the AZURE_CREDENTIALS secret.

I created a service principal, using the Cloud Shell in the Azure portal, which has contributor access to the resource group I created.

Created a GitHub secret in your repository and named it AZURE_CREDENTIALS and copy and paste the output of the service principal to the **Value** field of the secret.

Result:

A successfully completed Action in your GitHub repo, triggered manually in GitHub.

A step in the Action should have submitted a job to the Azure Machine Learning workspace.

A successfully completed Azure Machine Learning job, shown in the Azure Machine Learning workspace.

Challenge 3: Trigger GitHub Actions with feature-based development

My Understanding:

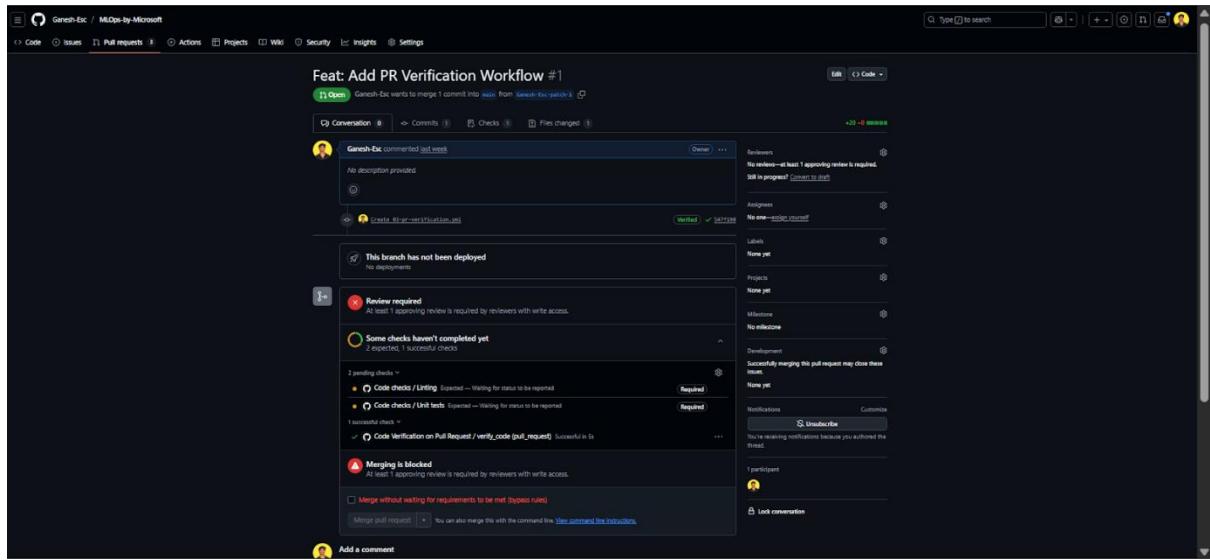
Triggering a workflow by pushing directly to the repo is **not** considered a best practice. Preferably, I must review any changes before you build them with GitHub Actions.

In this challenge, I created a GitHub Actions workflow which is triggered by the creation of a pull request. And also created a **branch protection rule** to block any direct pushes to the **main** branch.

To trigger the workflow, I did the following:

- Created a branch in the repo.
- Make a change and push it. For example, I changed the hyperparameter value.
- Created a pull request merge the new branch with the main.

Result:



Challenge 4: Work with linting and unit testing

My understanding:

Code quality can be assessed in two ways: linting and unit testing. Use linting to check for any stylistic errors and unit testing to verify your functions.

In this challenge I found that there are files in the test folder that perform linting and unit testing on the scripts. The flake8 lints the code to check for stylistic errors. The test_train.py performs unit tests on the code to check whether functions behave as expected.

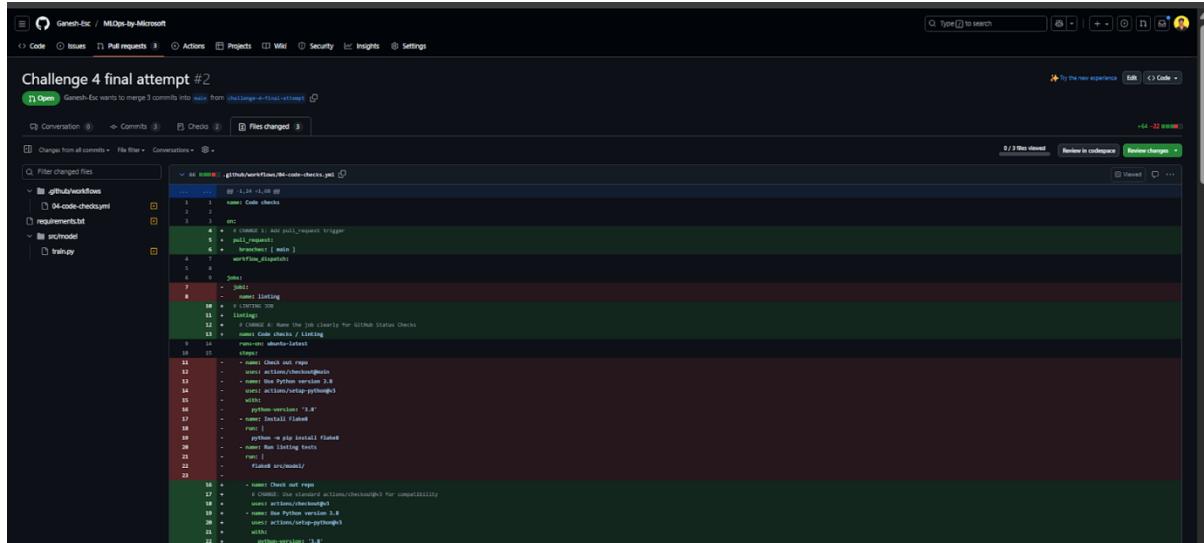
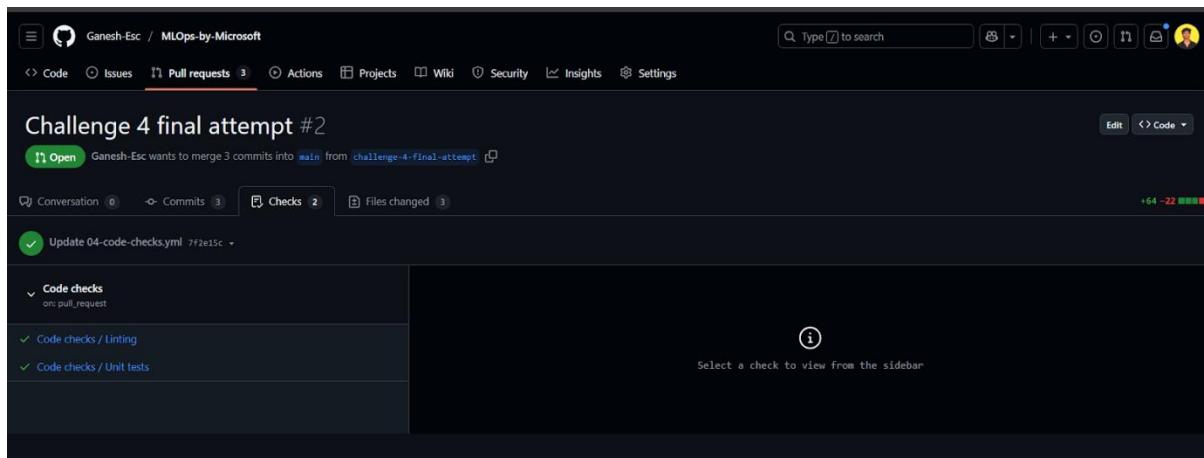
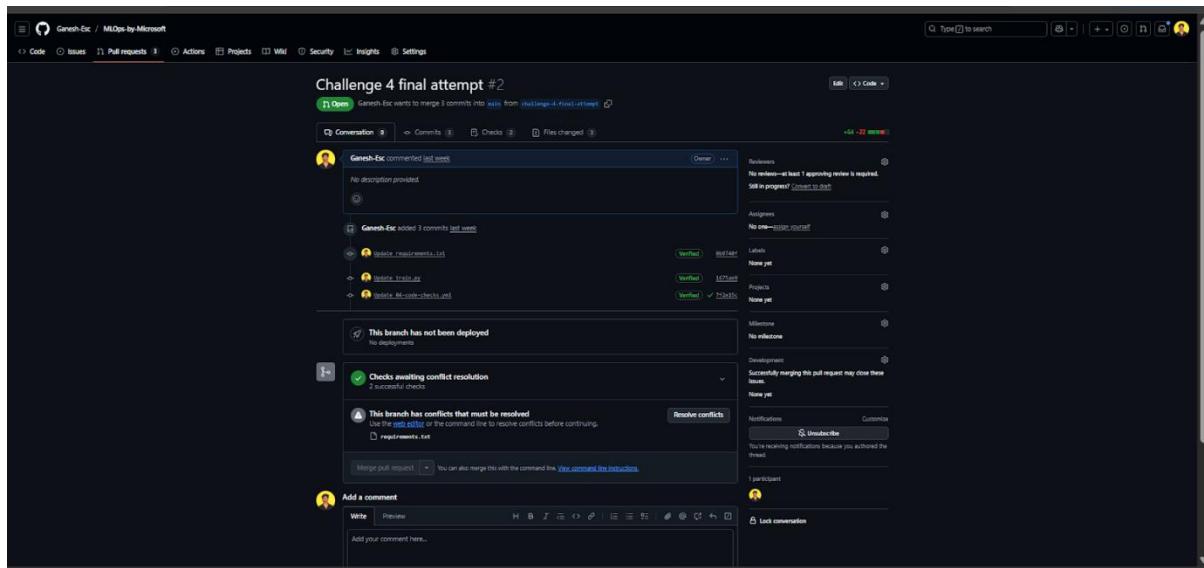
To achieve this, I triggered the code checks manually in the Actions tab of Github repository.

And to trigger the workflow, I did the following:

Made a change and pushed it. And created a pull request, showing the integrated code checks.

Result:

Both the **Linting** and **Unit tests** checks are completed successfully without any errors. The successful checks should be shown in a newly created pull request.



Challenge 5: Work with environments

My Understanding:

There are many advantages to using environments in Machine learning projects. When we have separate environments for development, staging, and production, we can more easily control access to resources.

It's better to use environments to isolate workloads and control the deployment of the model.

In this Challenge

I created a development and production environment and added an approval check to the production environment

Removed the global repo AZURE_CREDENTIALS secret, so that each environment will only be able to use its own secret and added the **AZURE_CREDENTIALS** secret to each environment that contains the service principal output.

And then Created a new data asset in the workspace with the following configuration:

Name: *diabetes-prod-folder*

Path: The **data** folder in the **production** folder which contains a larger CSV file to train the model. The path should point to the folder, not to the specific file.

Also Created one GitHub Actions workflow, triggered by changes being pushed to the main branch, with two jobs:

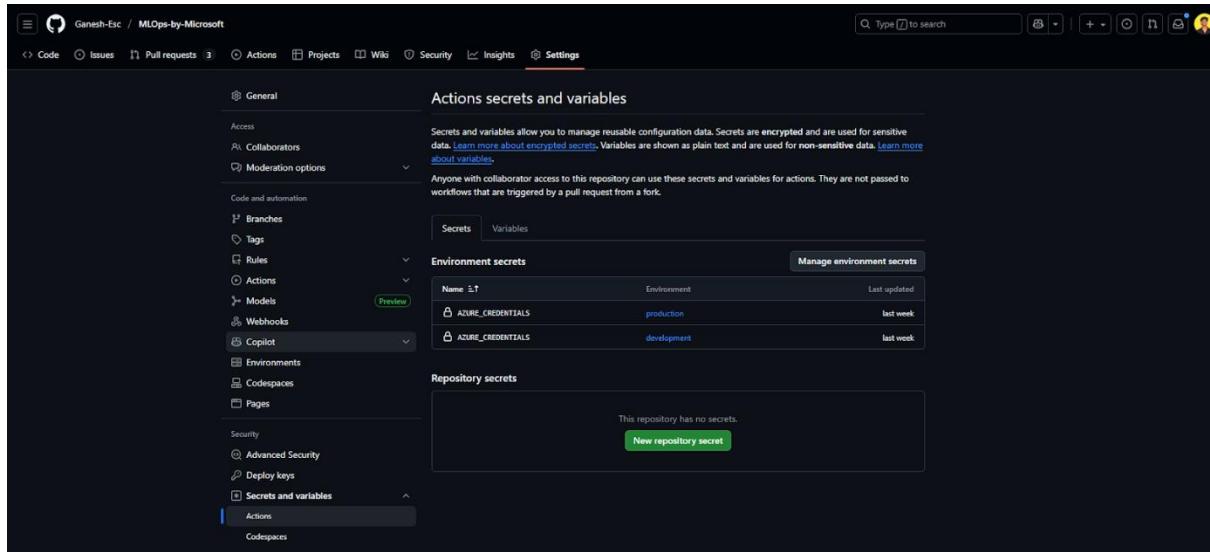
The **experiment** job that trains the model using the *diabetes-dev-folder* dataset in the **development environment**.

The **production** job that trains the model in the **production environment**, using the production data (the *diabetes-prod-folder* data asset as input).

Added a condition that the **production** job is only allowed to run when the **experiment** job ran *successfully*. Success means that the Azure Machine Learning job ran successfully too.

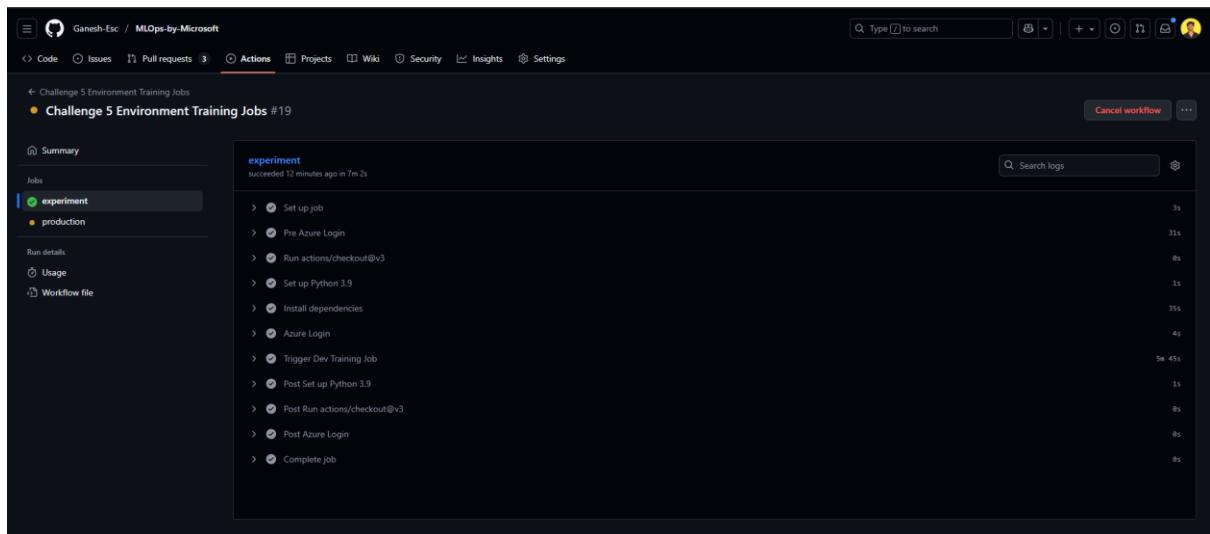
Result:

Show the environment secrets in the settings.



The screenshot shows the GitHub Actions settings page for the repository "MLOps-by-Microsoft". The left sidebar has "Actions" selected. Under "Environment secrets", there are two entries: "AZURE_CREDENTIALS" (production, last updated last week) and "AZURE_CREDENTIALS" (development, last updated last week). The "Repository secrets" section is empty, showing "This repository has no secrets." with a "New repository secret" button.

A successfully completed Actions workflow that contains two jobs. The production job needs the experimentation job to be successful to run.



The screenshot shows the GitHub Actions workflow logs for "Challenge 5 Environment Training Jobs" (run #19). The "experiment" job is highlighted. It succeeded 12 minutes ago in 7m 2s. The log details the steps: Set up job, Pre Azure Login, Run actions/checkout@v3, Set up Python 3.9, Install dependencies, Azure Login, Trigger Dev Training Job, Post Set up Python 3.9, Post Run actions/checkout@v3, Post Azure Login, and Complete job. Each step is marked with a green checkmark and its duration.

| Step | Duration |
|------------------------------|----------|
| Set up job | 3s |
| Pre Azure Login | 31s |
| Run actions/checkout@v3 | 8s |
| Set up Python 3.9 | 1s |
| Install dependencies | 35s |
| Azure Login | 4s |
| Trigger Dev Training Job | 56 45s |
| Post Set up Python 3.9 | 1s |
| Post Run actions/checkout@v3 | 8s |
| Post Azure Login | 8s |
| Complete job | 8s |

Show that the workflow required an approval before running the production workload.

The screenshot shows the Azure DevOps pipeline interface for a workflow named "Challenge 5 Environment Training Jobs #19".

Summary: Shows the status as "Waiting" with Ganesh-Esc as the trigger. A message indicates "Ganesh-Esc requested your review to deploy to production".

Workflow: The "05-environment-jobs.yml" file defines a workflow with two stages: "experiment" and "production". The "experiment" stage has a duration of 7m 2s. An arrow points from "experiment" to "production", with the label "production waiting for review".

Deployment protection rules: Shows a deployment protection rule where Ganesh-Esc requested review for the "production" environment.

Review pending deployments: A modal dialog titled "Review pending deployments" is open, showing a checkbox for "production" with the note "Review needed from Ganesh-Esc". It also includes a "Leave a comment:" input field and "Reject" and "Approve and deploy" buttons.

Show two successful Azure Machine Learning jobs, one trained with the *diabetes-dev-folder* as input and the other with the *diabetes-prod-folder* as input.

The screenshot shows the GitHub Actions pipeline run details for a job named '05-environment-jobs.yml'. The run was triggered via push last week and is labeled as 'Success' with a total duration of 20m 3s. The pipeline consists of two steps: 'experiment' (7m 1s) and 'production' (6m 37s). A deployment protection rule is present, with an event from Ganesh-Esc approved last week, targeting the 'production' environment.

Development Job details:

The screenshot shows the 'Overview' tab for a completed job named 'magenta_pummelo_0pm88jzlb'. The job was created on Oct 26, 2025 at 12:12 PM and completed at 12:15 PM. It took 2m 12.40s and had a compute duration of 2m 12.40s. The command used was '# FIX: Keep the command simple. Since "code: ." uploads the contents of src/ # the train.py file should be in the root of the uploaded code package, run "train.py --training-data".' The job is associated with an experiment named 'diabetes-training' and an environment named 'AzureML-sklearn-0.24-ubuntu18.04-py37-cpu:49'. The status is 'Completed'.

The screenshot shows the 'Outputs + logs' tab for the same job. The 'Outputs' section shows three log files: 'std_log.txt', 'metrics-capability.log', and 'lifecycle.log'. The 'Logs' section displays the content of the 'lifecycle.log' file, which contains log entries related to the execution of 'train.py' with specific parameters like '--reg_rate 0.01' and '--success_return_code: Zero'.

Production Job Details:

The screenshot shows two views of the Azure AI | Machine Learning Studio interface. The top view displays the 'Properties' tab for a completed production job named 'tidy_pencil_2r5kxftc54'. It shows details like status (Completed), created on Oct 26, 2025, at 12:22 PM, and a command log entry. The bottom view shows the 'Outputs + logs' tab, where a log file named 'lifecycle.log' is selected, displaying its contents.

Challenge 6: Deploy and test the model

My Understanding:

To get value from a model, we must deploy it. And we can deploy a model to a managed online or batch endpoint

In this challenge, I

Registered the model from the production job output in the Azure Machine Learning Studio.

Created a GitHub Actions workflow which deploys the latest version of the registered model. And the workflow should create an endpoint and deploy your model to the endpoint using the CLI (v2).

Result:

A model registered in the Azure Machine Learning workspace.

A successfully completed Action in your GitHub repo that deploys the model to a managed online endpoint.