Welcome to AI Studio by HP

AI Studio is a centralized desktop application specifically developed for data scientists and engineers like you. AI Studio lets you connect to multiple data-stores across local and cloud networks, so you can access the correct data and packages, wherever they are.

Compute locally without interruption to manage development, data, and model environments. Dive in now to learn more about how you can use the AI Studio platform to harness your data in a more meaningful way!

The community site is a hub for our members to engage, share ideas, and collaborate to help improve the AI Studio experience. Visit the community page to join and get the most out of AI Studio!

AI Studio User Guide EAP 2 (v1.2.2)

System Requirements

Z by HP AI Studio requires Windows 10 (build 19041 and higher), Windows 11, or Ubuntu 22.04 OS, AMD Ryzen™ 9 or higher processors, minimum 50 GB of available storage, 16GB of RAM, and Internet access. To enable GPU compute, Z by HP AI Studio requires any NVIDIA® GPU compatible with driver version 528.89 or newer. A minimum of 8GB of VRAM is recommended.

> **Note**: AI Studio supports up to 3 compatible NVIDIA GPUs on Windows. Work is underway to unlock up to four GPUs on a compatible workstation.

Hardware Recommendations:

- GPU: NVIDIA® GPU card with CUDA® Compute capability driver version 528.89 or newer

Software Requirements:

- WSL 2 (Windows Subsystem for Linux) must be enabled on setups running Windows OS. Refer to WSL Recommended Setup for more details.

Installation Requirements:

- Internet connection
- Local administrative / root user permissions
- An active AI Studio user license

Installing & Uninstalling AI Studio

AI Studio Installation

To install AI Studio on Windows:

1. Download and run the AI Studio installer here. If you haven't enabled virtualization in the BIOS, an error message will prompt you to do so before continuing.

   Tip: If git is not already installed on your machine, the installer will guide you to do so. Features that depend on git are disabled in the app for users who choose not to install.

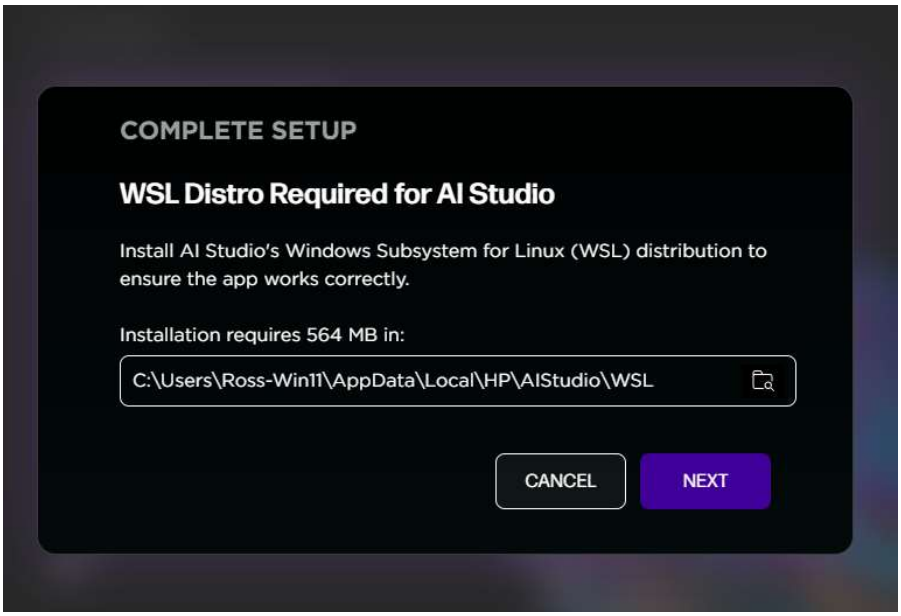
2. Complete the wizard to finish installing AI Studio.

3. Click on the AI Studio desktop icon to start using AI Studio.

   Note: The application automatically checks for a compatible existing distro and leverages it to install and configure the necessary images. If none exists, it will add one for you on your first start up.

4. On your first start up, you'll have the option to choose where to download the WSL distro. Click **Next** to install the AI Studio WSL distro to the default folder or click the folder icon and use the file browser to select a different folder.

   Note: WSL may time out if AI Studio is left open in the background for too long.



5. ~~(Recommended) Click the link(s) in the installation wizard to download and install WSL and the latest (64-bit) git version control system if you don't already have it, then click **Next**.~~

6. ~~Complete the wizard to finish installing AI Studio.~~

   ~~**Tip**: If prompted, allow the installer to make changes to your device.~~

~~To start using AI Studio, select the AI Studio desktop icon that's created when the installation is complete, or search for it manually in the Windows search bar.~~

To install AI Studio on Ubuntu (Early Access Only):

1. Download and run the AI Studio installer here.

2. After finishing the download, open the terminal in the same directory as your downloaded file and run the command:

   sudo apt install ./AIStudioSetup[app version].deb

   Note: Apt may give a warning that says "Download was performed unsandboxed as root". This is normal and does not mean the install failed.

3. Restart your Ubuntu machine to make sure your user is added to the AI Studio permissions group.

   Note: The first time you start the app on Ubuntu, AI Studio will ask for permission to set up nvidia-container-toolkit if you don't already have it.

   You can open AI Studio by searching for it in the applications menu, or by typing *AIStudio* in the terminal.

Uninstallation

To uninstall AI Studio on Windows:

There are two ways to uninstall AI Studio from your machine.

1. Run the AIStudioSetup.msi file to open the installer.

2. Click **Next**, then click Remove to delete AI Studio from your computer.

   OR

1. Type Add or remove programs into the Windows search bar and select it in System Settings.
2. Search for AI Studio in the Apps and Features search bar or scroll to select it manually.
3. Select the options icon, then select Uninstall.

To uninstall AI Studio on Ubuntu (Early Access Only):

From the terminal, run the command:

```
sudo apt remove hpaistudio
```

Note: This will not delete your user files. Uninstalling AI Studio on Ubuntu leaves your user files intact.

Networking and Sync Protocols

AI Studio uses AWS and Google cloud services to support container syncing across account machines. Workstation daemon services and IT guards need to be configured to allow ingress and egress of these services to enable collaboration features of AI Studio.

AWS and Google services enable syncing of MongoDB, Git, and Syncthing peer-to-peer applications to support collaboration features in AI Studio.

Navigating the Projects Page [Needs Review]

Whether you're a returning AI Studio user or this is your first time, you can find and manage your projects on a single pane of glass, directly from the Projects Page.

To create a new project:

1. From the Projects home page, select ***New Project***.

2. Enter a name and description for your project, then choose a privacy status for the project.

3. (Optional) Enter the URL for the Git Repository you wish to connect to the project.

4. Select a local folder to download your repository to, then click **Continue**.

   Tip: Add tags to your project to make it easier for collaborators to find and organize them.

5. (Optional) Use the assets catalog to select an asset to connect to from your list of configured assets or follow the steps for Connecting Data To a Project.

   Note: Assets are objects so they can be models, notebooks, scripts, or datasets. You can also click **Continue** without adding asset information to connect data to your project later from the *Assets* tab.

6. Choose a workspace template, then click **Create Project** to create your new project.

Dive into an existing project directly from the home page by clicking on the project tile. You can narrow homepage results with filters and customize how to sort them to find your projects in a snap.

Tip: Clicking the icon in the upper-right corner of a project tile triggers the project options dropdown box to appear.

To use your project:

From the **Projects** home page, choose the project you wish to use.

- Manage and run the workspaces from the *Overview* tab.
- Monitor experiment runs in TensorBoard or MLflow from the *Monitor* tab.
- Deploy your models locally from the *Deployments* tab.
- Import models and datasets from AWS S3, Azure Blob Storage, or the GCP Object Store from the *Assets* tab.

Connecting Data to a Project [Needs Review]

Use local data or use your AWS or Azure credentials to connect data to your existing projects from AWS or Azure Blob remote storage.

To connect data to a new project:

1. When you finish adding project details, connect your data by simply selecting **Add** next to the asset(s) you want to add. Then, click **Create Project**.

2. If there are no available assets, click **Create new Asset** to connect to a new one.

    - Name and describe your asset, then specify your asset type and the connection source (i.e., local, AWS, or Azure Blob Storage).

        - **Local**: Enter the path or browse your local directory to locate the asset, then click **Save**.

        - **AWS**: Enter the S3 URI associated with your asset, the bucket type, region (optional for private assets), and description. Then, click **Create Asset**.
            Tip: Public AWS S3 buckets don't require credentials and authentication.

        - **Azure Blob Storage:** Enter the Blob URI associated with your asset, the resource type, and an asset description. Then, click **Create Asset**.

To connect data to an existing project:

1. From the project's *Overview* page, select the ***Assets*** tab and navigate to the *Project Assets* window.

2. Click **Catalog** to connect an existing asset to the project from the Assets Catalog.

3. If the asset you wish to add is already listed, click **Add** to upload it into your project.

    Tip: Click on the more options icon to edit the asset before you add it to your project.

4. If your desired asset hasn't been connected, click **New Asset** to connect to a new one.

    - Name and describe your asset, then specify your asset type and connection source (i.e., local, AWS, or Azure Blob Storage).

        - **Local**: Enter the path or navigate the file browser to locate the asset, then click **Save**.

- **AWS**: Enter the S3 URI associated with your asset, the bucket type, region (optional for private assets), and description. Then, click **Create Asset**.

- **Azure Blob Storage**: Enter the Blob URI associated with your asset, the resource type, and an asset description. Then, click **Create Asset**.

Inviting Team Members

After you create and share a project, you're ready to start collaborating. Team Managers can invite, accept, and resend pending invitations to assign team members to available seats. Contributors can invite users, but invitees need Team Manager approval before they're assigned to an account seat.

To invite team members:

1. Navigate to the **Account** tile.

2. Choose the *Team Settings* tab, then select ***Invite new members***.

3. Enter their email address in the provided field, then click **Invite**.

4. Click ***Add more team members*** to increase your team's size, then repeat steps 1 – 4 until your team is complete.

5. After verification that the invited user's account is activated, enter their role to delegate responsibility among collaborators.

   Tip: As a Team Manager, you can select the **Account** tile and navigate to the *Team Settings* tab to edit the roles of your other team members.

Team Managers can add or remove team members at any time, from *Team Settings*. Team managers can also send a reminder or delete members with a pending invitation status. Anyone can invite teammates to collaborate on a project, provided there are available seats.

Using a Predefined Workspace [Needs Review]*

Workspaces created with AI Studio templates sync with collaborators in shared projects, so team members can be certain they're using workspaces in the same environments.

From the *Project Overview* page:

1. Select **New Workspace**.

2. Choose the workspace environment that is most compatible with your machine(s) and project requirements.

    - **Minimal (CPU Ready):** For basic libraries, small workspaces offer lightning-fast startup times, but sacrifice compute and power to do so.

    - **Data Science (CPU Ready):** The recommended setup for most users. Medium workspaces have the most balance of power and speed.

    - **Deep Learning (GPU Ready)**: For the most complex experiments, this is the most powerful workspace AI Studio can offer.

    - **Local GenAI (GPU Ready)**: For working with LLMs, RAG pipelines, and Gen AI agents. Gen AI workspaces are an optional add-on and require an additional license to use.

    - [Link to base images section] **NVIDIA's NGC Catalog (GPU Ready)**: Leverage hundreds of models and containers to transform AI Studio into your hub for optimized GPU Software solutions.

      Note: If you choose not to download the required workspace images, you'll need to do so before you can run your workspace for the first time.

    Tip: Click ***learn more*** for more details about your workspace.

3. Name your new workspace.

4. Click on **Create Workspace** to run your workspace in a notebook tab.

    Note: Download the required base image and libraries before running Data Science or Deep Learning workspaces.

Adding Custom Libraries [Needs Review](Custom Workspace Renamed)

For larger projects that need more nuanced customization and powerful compute, you can set up your workspace with custom libraries. Custom workspaces sync with collaborators in shared projects, so teams know they're using workspaces in the same environments.

Tip: Customize base AI Studio templates by uploading a **requirements.txt** file.

To create your environment:

1. Choose a project and navigate to *My Workspaces* on the project's home page.

2. Click on **New Workspace.**

3. Choose the workspace environment that is most compatible with your machine(s) and project requirements.

   Tip: Check out our guide on the supported NGC containers [link to base images section].

4. (Recommended) Download the Deep Learning or Data Science base image before adding those libraries to the workspace.

   Note: If images are not downloaded during workspace creation, they must be downloaded before starting the workspace.

5. Name your workspace and specify your project's graphical requirements.

6. Click **Add Custom Libraries** to add any custom Python libraries you want to use in your workspace.

   Note: Workspaces with custom libraries have two kernels available to run a jupyter notebook in (*conda env:base* or conda env:*aistudio*). When you launch a workspace, the default kernel is set to conda env:base for all custom workspaces. Users must validate their custom libraries, then manually change the kernel to conda env:aistudio to use custom libraries with the workspace.

7. Paste applicable libraries from a compatible base image in the space provided, then click **Validate libraries** to test your package.

   Note: You can use your own libraries, but they may be incompatible with the libraries necessary to run the required base images. Validating libraries is optional but assists in troubleshooting.

8. Click **Create Workspace** to save your new workspace.

9. Click on the play icon in your workspace tile to instantiate your environment.

> **Warning**: The blue progress bar indicates that the application is still building the container for your workspace. This process should take about 5 minutes, depending on your device's capabilities. **DO NOT** navigate away from the window or select another workspace until the progress bar vanishes.

AI Studio projects can support more than one workspace to accommodate projects that require an ensemble of models to complete your team's work.

Reusing a Workspace [Needs Review]

Once you create a workspace, it appears as a tile on the project's ***Overview*** *page*. Projects can run multiple workspaces concurrently; each workspace notebook appears as a separate tab on the Projects page.

To reuse a workspace:

From your project's home page, navigate to *My Workspaces* and select the workspace you want to reuse.

2. Click the play icon to start your workspace.

   Tip: Select the more options icon and click **Edit Workspace** to view details about your workspace.

3. Click the terminal icon to open your workspace notebook. The workspace notebook should run in a new tab beside *Project Setup*.

   Note: Starting a workspace creates a container, while stopping one closes the associated container. Building a container may take a few minutes, depending on your machine's capabilities.

Now that your workspace is running, you can start testing and collaborating on models directly from AI Studio.

Workspace Sharing [Needs Review]

Share assets and file updates with collaborators with real-time workspace sharing in AI Studio.

> **Tip**: View the documentation on Networking and Sync Protocols to ensure your team's system configurations are set up to support AI Studio.

~~To share data from the file browser:~~

1. ~~From the *Setup & Documentation* tab, click **Setup**, then navigate to the File Browser and select the shared folder.~~
2. ~~Click **Upload files** to add files or folders.~~
3. ~~Select the files you wish to share in the file explorer, then click **Save.**~~

To share data from a notebook tab:

1. Select the shared folder from the notebook file browser.
2. Click **Upload files** and select the files you wish to share.
3. Click **Open** to add the files to your shared folder, then click **Save**.

> **Note**: Avoid simultaneously modifying files in the datasync tree (i.e. user A and user B both editing notebook X).

Files placed in a project's shared folder leverage Syncthing to synchronize online collaborators' workspaces and must sync completely before other users can access them. Syncing larger assets can take several minutes to appear for your collaborators. Synchronization only occurs for a user's last active account – other accounts don't sync in the background.

AI Studio Workspace Images [Needs Review]*

Workspaces are one of the core elements of AI Studio. Each workspace runs in a separate container, isolated from the host application except when sharing project files and folders. Workspaces in AI Studio leverage IPython notebooks or Python scripts through JupyterLab to help users develop their experiments in a centralized, secure, and efficient way.

Base Images

Base Images are predefined environments with a pre-installed Python version and a set of libraries designed to help you dive into AI Studio right away. AI Studio offers two different base images to accommodate the scale of your workspaces: one that's GPU enabled and one that's CPU-only.

CPU-only images are smaller, faster to load, and are most useful when performing basic computations and traditional DS algorithms. CPU-only images have most of the same libraries as the more robust GPU-enabled options, so users that don't have a GPU that's compatible with AI Studio are still likely to find them helpful. GPU enabled images support cuda packages and attach notebooks to the device to run on GPUs without additional configuration by the user.

Minimal Image:

The CPU-only minimal image comes with the minimal set of libraries necessary to run Data Science experiments in AI Studio. It runs JupyterLab 4.0.1 with Python 3.10.11, based on the minimal-notebook:python-3.10 image from Jupyter Docker hub.

Deep Learning Image:

The Deep Learning image is specifically tailored for Deep Learning and Neural Network Architectures (NNA). It encompasses an extensive configuration designed to optimally run your most intricate experiments. High computational power is necessary, making the Deep Learning image well-suited for tasks involving image analytics and Large Language Models (LLMs). It includes all the features found in the Data Science image, with upgraded libraries like TensorFlow and PyTorch. The Deep Learning image also comes pre-configured with GPU capabilities for which we recommend reserving at least 4GB of memory.

Data Science Image:

The Data Science image represents the most standard yet powerful configuration, designed for quick setup and enhanced performance. Inspired by scipy image on Jupyter Docker hub, this image encompasses all the features of the minimal configuration and further enhances it with Data Visualization tools like Seaborn, Altair, and scikit-image. Recommended for usage with a minimum of 4GB of memory.

Local GenAI Image (EAP only):

Local GenAI image is our official environment for language application and GenAI use cases. Besides the libraries for machine learning support, it comes with several libraries to create and use Language Model and processing chains, such as LangChain, Transformers, and ChromaDB. Moreover, it has libraries to load your model locally (llama-cpp) and also connectors to different cloud services (OpenAI, HuggingFace, Google and Amazon). It also comes with Galileo libraries pre-installed, with full support to features such as Evaluate, Protect and Observe, as well as a set of templates for different use-cases of GenAI.

NGC Catalog Containers

NVIDIA's NGC Catalog includes dozens of containers, which you can leverage in AI Studio with the same steps you'd use to add a base image to a workspace. Any of the NGC containers and models that are available in AI Studio during workspace creation and asset management are supported and available to all users.

> **Note**: Some of the models and containers listed on NVIDIA's NGC Catalog website may not be compatible with your projects in AI Studio.

Three images we have examples for and that run optimally on AIS include:

- NeMo Framework: NVIDIA NeMo™ is an end-to-end platform for development of custom generative AI models.
- RAPIDS Base: Execute end-to-end data science and analytics pipelines entirely on GPUs. Use this image if you want to use RAPIDS as a part of your pipeline. Visit rapids.ai for more information.
- RAPIDS Notebooks: Execute end-to-end data science and analytics pipelines entirely on GPUs. Use this image if you want to explore RAPIDS through notebooks and examples. Visit rapids.ai for more information.

Check out our guide on Using a Predefined Workspace for more details on how to add an NGC container like the ones above to your AI Studio workspace.

> **Tip**: When you're creating a workspace or adding an asset in AI Studio, you can double-click on an NGC container's name to open NVIDIA's documentation about the container in your web browser.

The following libraries come pre-installed:

Minimal	Deep Learning	Data Science	Local GenAI	Name	Description
	x	x		altair==5.0.1	

					A declarative statistical visualization library for Python with a simple, friendly, and consistent API. Built on top of the powerful Vega-Lite JSON specification.
	x	x		bokeh==3.3.0	An interactive visualization library for modern web browsers. It provides elegant and streamlined construction of versatile graphics and offers high-performance interactivity across large or streaming asset.
	x	x		ipympl==0.9.3	A Jupyter extension that improves chart visualization in matplotlib.
x	x	x	x	matplotlib==3.8.2	A comprehensive library for creating static, animated, and interactive visualizations in Python.
x	x	x	x	mlflow==2.6.0	A platform used to streamline ML development.

					It lets users log and track ML experiments; pack and register ML models; and version ML projects, among other useful features.
x	x	x	x	numpy==1.26.3	A fundamental package for scientific computing with Python. It provides a powerful N-dimensional array object; sophisticated broadcasting functions; and tools for seamless C/C++ and Fortran code integration. It also offers useful tools for linear algebra, Fourier transform, and random number capabilities.
x	x	x	x	pandas==2.2.0	A package that provides fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data

					both easy and intuitive.
x	x	x		patsy==0.5.6	A library for describing statistical models (especially linear models, or models that have a linear component).
x	x	x	x	pillow==10.2.0	A Python Imaging Library fork that adds image processing capabilities to the Python interpreter.
	x	x		plotly-express==0.4.1	A higher-level wrapper to the plotly data visualization library. It enables interactive data visualizations.
	x		x	pytorch	A widely-used Deep Learning library.
	x	x		scikit-image==0.21.0	An open-source image processing library for the Python programming language. It includes algorithms for segmentation, geometric transformations, color space manipulation,

					analysis, filtering, morphology, feature detection, and more.
x	x	x	x	scikit-learn==1.4.0	A Python module for machine learning built on top of SciPy. It provides a wide range of algorithms for classification, regression, clustering, feature extraction, anomaly detection and others.
x	x	x	x	scipy==1.12.0	An open-source library for mathematics, science, and engineering. It builds on NumPy and provides additional functionality for optimization, integration, interpolation, eigenvalue problems, signal and image processing, statistical functions, and more.
x	x	x		statsmodel==0.14.0	A Python library that

					provides classes and functions for estimating statistical models, conducting statistical tests, and exploring data.
x	x	x	x	tensorboard==2.13.0	A library that logs information through ML model training. It also adds an interface for visualizing and understanding the evolution of logged training.
	x			tensorflow	A widely-used Deep Learning library.
	x	x	x	tqdm==4.65.0	A library that displays progress bars in loops and iterations in Python. It lets users track the progress of lengthier tasks visually.
x	x	x	x	zstandard==0.19.0	A fast and efficient compression library. It offers competitive compression rates at significantly higher speed when compared to

					standard compression libraries.
			x	langchain	LangChain is a framework for developing applications powered by large language models (LLMs).
			x	langchain-core	defines the base abstractions for the LangChain ecosystem.
			x	langchain-community	LangChain Community contains third-party integrations that implement the base interfaces defined in LangChain Core, making them ready-to-use in any LangChain application.
			x	langchain-openai	This package contains the LangChain integrations for OpenAI through their openai SDK.
			x	langchain-huggingface	This package contains the LangChain integrations for huggingface

					related classes.
			x	langchain-google-vertexai	This package contains the LangChain integrations for Google Cloud generative models.
			x	langchain-aws	This package contains the LangChain integrations with AWS.
			x	openai	The OpenAI API provides a simple interface to state-of-the-art AI models for natural language processing, image generation, semantic search, and speech recognition.
			x	google-cloud-aiplatform	Google Vertex AI is an integrated suite of machine learning tools and services for building and using ML models with AutoML or custom code
			x	boto3	Boto3 is the Amazon Web Services (AWS) Software Development

					Kit (SDK) for Python, which allows Python developers to write software that makes use of services like Amazon S3 and Amazon EC2.
			x	promptquality	Accelerate AI System Evaluations with Galileo Evaluate.
			x	sentence-transformers	This framework provides an easy method to compute dense vector representations for sentences, paragraphs, and images.
			x	transformers	Transformers provides thousands of pretrained models to perform tasks on different modalities such as text, vision, and audio.
			x	datasets	Datasets is a library for easily accessing and sharing datasets.
			x	evaluate	Evaluate is a library that makes evaluating and

					comparing models and reporting their performance easier and more standardized.
			x	rouge-score	This is a native python implementation of ROUGE, designed to replicate results from the original perl package.
			x	chromadb	the open-source embedding database.
			x	galileo-protect	Galileo Protect is a proactive GenAI security solution that acts as an LLM Firewall. It's designed to protect your system from harmful inputs and safeguard your users from potentially problematic outputs.
			x	galileo-observe	Galileo Observe is a monitoring tool designed for generative AI applications in production. It helps you understand how users interact with your

					application and identify potential issues.

Cloning a GitHub Repository [Needs Review]

Use AI Studio to share data and collaborate with your teammates without changing the workflows your team is most familiar with. You can connect AI Studio to your GitHub account by cloning a repository into your workspace to collaboratively manage your code.

To clone your repository:

1. From the ***Projects Overview*** page, select the ***Project Setup*** tab, navigate to setup, then click **Clone GitHub Repository**.

2. Type your GitHub Repository URL in the space provided.

3. Specify the local folder into which to download the cloned repository.

   Warning: Your file path should **not** include spaces. If the repo fails to clone, check your local file path to ensure it doesn't contain any forbidden characters.

4. Click **Add GitHub Repository** to clone the repository.

   Note: When you add your cloned repository, AI Studio redirects you to the GitHub domain page to authenticate your account. Allow GitHub to access the necessary data from your account, then return to the application to start using data on the cloned repository.

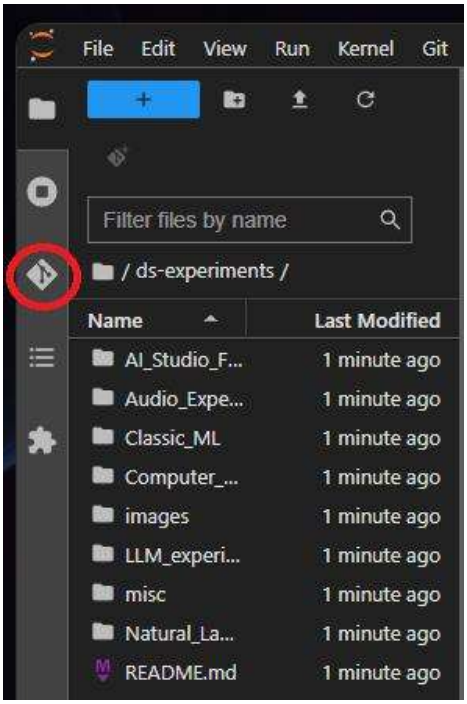
5. Restart your workspace if it's open to apply changes.

   Tip: You can access and manipulate data from cloned repositories with the notebook tab that appears when you run your workspace, exactly like you would with a local or remote AWS connection.

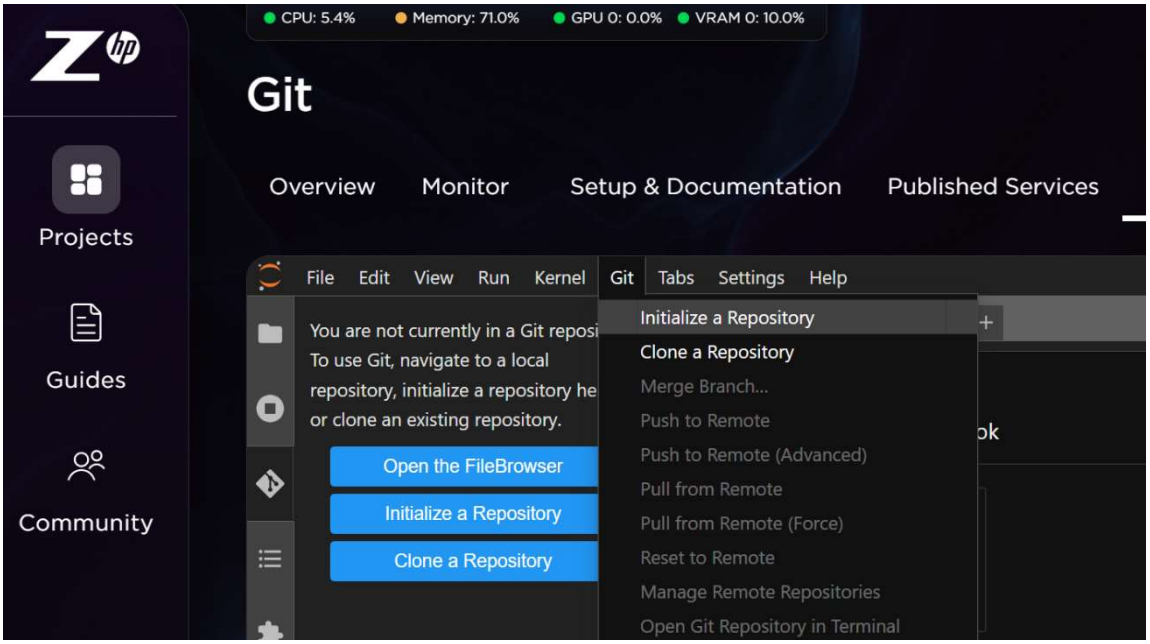
To push code changes to the repository:

1. Run the workspace to open a notebook tab.

2. Click on the git action icon or use the Git menu.



3. Select **Initialize a Repository**, then click **Yes** to begin initialization.



Now you can now use the Jupyter extension to push and pull code updates to the associated repository.

Managing Project Assets [Needs Review]

Use the asset manager to view and manage a project's associated assets directly from the ***Assets*** tab. You can use the asset manager to connect a completely new asset. Or, you can edit, download, or remove any asset you or your collaborators connect to the project.

To connect new data to a project:

1. From the project's overview page, navigate to the *Assets* tab.

2. From the Project Assets window, click **Add Asset**, then click **New Asset**.

3. Name and describe your asset, then specify your connection source.
   - **Local**: Enter the path to your local asset, then click **Create Asset**.
   - **AWS S3**: Enter the S3 URI associated with your asset, the bucket type, region, and description, then click **Create Asset**.
   - **Azure Blob Storage**: Enter the Blob URI associated with your asset, the resource type, and an asset description. Then, click **Create Asset**.

   Tip: Add tags to your assets to make it easier for collaborators to find and organize them.

4. Click **Add** beside the new asset to add it to your project, then click **Add Assets** to connect the selected asset(s).

   Tip: Click the more options icon at the end of the asset row to edit, upload, download, or remove the asset.

Using the File Browser

Maximize your project’s workflow efficiency by using the file browser to view and manage a project's associated files.

About the File Browser

AI Studio's file browser is a convenient file manager that lets data scientists examine a project's file paths without running a workspace. You can share updates with collaborators in real-time by simply uploading the files to a shared folder. The file browser only displays content that's relevant to the open project to help prevent accidental data loss or corruption of unrelated files.

Model Deployment [Needs Review]

Model Services are embedded utilities that let users deploy registered models for local inference. Locally published model services are hosted on localhost as a browser tab. Each tab represents a different notebook and you can simultaneously run as many models as your machine can handle.

To deploy a new model service:

1. From the ***Project Overview*** page, select the *Deployments* tab, then click **Deploy Your Model**.
2. Name your new service and select one of the registered models from your connected MLflow account.

   > **Note**: Only models saved to MLflow can be deployed. See Registering a New Model for more information.

3. Choose your model version and GPU configuration.
4. Select a workspace to deploy the service on, then click **Deploy**.

   > **Note**: You can't add models to a running workspace, so add any models you might find useful **before** you run it.

When you create a new service, it automatically becomes viewable from the *Deployments* tab. Use the more options icon at the end of each row to edit or delete a published service at any time.

> **Tip**: Your project's GPU, CPU, VRAM, and memory consumption appear in the corner of the screen, so you can visualize the effects of the tests you run in real-time.

~~Local model deployment lets users test trained models and AI services locally, avoiding scaling and production issues that could block your productivity. AI Studio leverages MLflow's model registering and serving features to let any model that follows MLflow standards be deployed as a service in the application. Running as a local container with a Rest API available and accessible with swagger.~~

~~To deploy a model:~~

~~1. Log and register the model for MLflow in your python code.~~

~~2. From the Published Services tab, create a new Service.~~

~~3. Select the model's name and version as they're registered in MLflow.~~

4. Type the name you want for the service.

5. Click Run to run the service you created.

Spawning the container to run the service might take a few minutes, but the service icon will turn green and a service URL will populate when it's ready. By clicking on this URL, a swagger interface for the model's Rest API will open in a browser window.

Using the Monitor Page

AI Studio runs the tools you select natively, so you can use popular data science solutions in a single application. AI Studio supports the open-source tools ML Flow (pytorch) and Tensorboard (TensorFlow).

To begin monitoring:

1. From your project's home page, select and run the workspace you want to use.

2. Select a Jupyter Notebook from the root folder.

3. Import the necessary MLFlow or Tensorboard packages into the notebook.

4. Embed MLFlow or Tensorboard hooks for tool to capture, track, and manage metrics and artifacts.

   Tip: See the quick guide use case for an example of how to work with MLFlow and Tensorboard. Visit MLflow and Tensorboard documentation sites for detailed information on these tools.

5. Run the notebook you selected.

6. Navigate to the *Monitor* tab and use the integrated application to monitor your data as usual.

   Tip: Click the icon in the upper-right corner of the DS tool panel to enter full screen view.

You have real-time access to your project's CPU, GPU, VRAM, and memory consumption by the Z by HP logo so you can check on your running models at any time. AI Studio periodically recommends configuration improvements based on your tool usage trends.

Account Switching

AI Studio users with multiple accounts can switch between them from any screen with a few clicks.

To switch accounts:

1. Click on your user icon in the upper-right corner of the screen.

2. Click on the account you want to switch to from the dropdown menu.

Users must be online to switch between accounts. Account switching requires an internet connection to sync and sync only occurs for a user's last active account – other accounts do not sync in the background. If the app becomes unresponsive when one or more of the synchronizing users are offline, log out, then log back into AI Studio to restore your session.

> **Tip**: An alert will inform users if AI Studio is unable to switch accounts. To restore account switching, stop all the project's open workspaces and try again.

Compute

AI Studio represents compute tabularly, providing you with an instant overview of the device(s) capabilities of each account user.

To view your team's available compute:

1. Navigate to the **Account** tile and select the ***Compute*** tab.

2. View details about the compute available to each device used by you and your team.

The email address associated with each device appears to the left of a device's name. The table displays the device's available cores and RAM to the right of the name. If applicable, this view also shows the GPU name and the GPU RAM attributed to each device.

Team Settings

You can view and search for current active members of your team under the **My team** tab.

Pending invitations appear under the **Pending** tab until invitees accept or decline the invitation sent to their email address.

You can also use the **Pending** tab to view the date an invite was sent, or you can resend an invitation to or delete pending invitees by clicking on the more options icon and selecting **Set Reminder**. Invited user requests created by collaborators require administrative approval to allocate an account seat. Requested invitations will appear under the **Requests** tab.

To manually add users to an existing team:

1. From the *Team Settings* page in the ***Account*** tile, click on **Invite new members**.
2. Enter their email address to invite them to your team.
3. Click on **Add more team members** and repeat step 2 for each user you want to add.
4. When you're finished adding members, click on **Invite** to send account invitation emails. Once users activate their account and accept the invitation, AI Studio adds them to your team as a contributor.

AI Studio sends each new member an invitation to join the team via email. Team managers can remove users from their team or edit individual user roles at any time by clicking on the more options icon beside the member you wish to manage.

Preferences

AI Studio users can further streamline their data science workflows by personalizing their account preferences.

To change the default asset download folder:

1. From the **Account** tile, click on the gear icon in the top-right of the screen to open the preferences menu.

2. Click on the icon beside the default download folder value to open a file browser.

3. Navigate to the desired folder and double-click it to set it as the default download folder for new assets.

4. Click on **Save Changes** to finalize your adjustments.

To change your default AWS profile:

1. From your web browser, navigate to the AWS profile you want to set as default and copy the S3 path.

2. In the account preferences menu, paste the copied text into the AWS Profile value.

3. Click on **Save Changes** to finalize your adjustments.

Get Help [Needs Review]

AI Studio provides online documentation and community support for anyone interested. AI Studio subscribers can also rely on a dedicated support team to help resolve conflicts in a pinch.

To contact support from the app:

1. Click on the **Support** tile.
2. Specify the operating system on which you encountered the problem.
3. Select your country from the dropdown menu to generate the proper country code.
4. Provide a seven-digit phone number support can use to contact you.
5. Specify the product you're requesting support for.
6. Enter a brief description of the problem, what you were doing when it occurred, and how the observed behavior differs from what is expected.
7. Select the checkbox to attach the app's crash logs to the ticket.
8. Click on **Submit** to reach out to support. To provide more feedback about your AI Studio experience, select **Send Feedback** to go to the AI Studio ideas portal.

To contact support from the web:

1. Visit the AI Creator Community home page and search the discussion forums to see if your question has been answered there.
2. Click Support to check out the product FAQs.
3. If you still need help, make a new post to ask the community or scroll to the bottom of the page for links to more support.

P2P Data Sharing

AI Studio uses peer-to-peer networking to synchronize data in the background in these three circumstances:

- Data recorded by MLflow(User runs MLFLOW experiment in Jupyter notebook)

  Note: If two users are in the same account and runs MLFlow experiment in Jupyter notebook both the users should be able to respective results of Mlflow experiment for a project.
  Note: This data includes model parameters, experiment results and model data.

- Data output to Tensorboard

- Assets in the *shared* directory (see below)

Synchronization operates asynchronously in the background while AI Studio is running. All members of an account contribute to a mesh p2p network. As files are added to paths monitored for synchronization, they are synchronized across all account members.

Note: Two or more account members must be online to sync.

Synchronized paths (Windows)

- %LOCALAPPDATA%\HP\AIStudio\[account ID]\projects\[project ID]\mlflow
- %LOCALAPPDATA%\HP\AIStudio\[account ID]\projects\[project ID]\tensorboard
- %LOCALAPPDATA%\HP\AIStudio\[account ID]\projects\[project ID]\shared

Synchronized paths (Linux XDG_STATE_HOME unset)

- $HOME/.state/hp/aistudio/[account ID]/projects/[project ID]/mlflow
- $HOME/.state/hp/aistudio/[account ID]/projects/[project ID]/tensorboard
- $HOME/.state/hp/aistudio/[account ID]/projects/[project ID]/shared

Synchronized paths (Linux XDG_STATE_HOME set)

- $XDG_STATE_HOME/hp/aistudio/[account ID]/projects/[project ID]/mlflow
- $XDG_STATE_HOME/hp/aistudio/[account ID]/projects/[project ID]/tensorboard
- $XDG_STATE_HOME/hp/aistudio/[account ID]/projects/[project ID]/shared

Warning: Simultaneously editing files in shared paths (e.g. storing a jupyter notebook in a shared tree to collaborate) is not advised. HP recommends using AI Studio's git integration features to develop code collaboratively.

Limitations & Known Issues

- WSL may time out if AI Studio is left open in the background too long, interfering with expected workspace behavior. Log out and close the application, then reopen AI Studio and log back in to reset WSL and restore normal functionality. (WSL Timeout Issue)

- For some users, ML flow and Tensorboard take longer than expected to sync across collaborators who are working together in a shared folder. This is, in part, due to a bug with the Syncthing GUI that causes a delay before indicating that the machines are in sync with each other. (KI)

-

Admin: Purchasing or Renewing an Account [blocked]

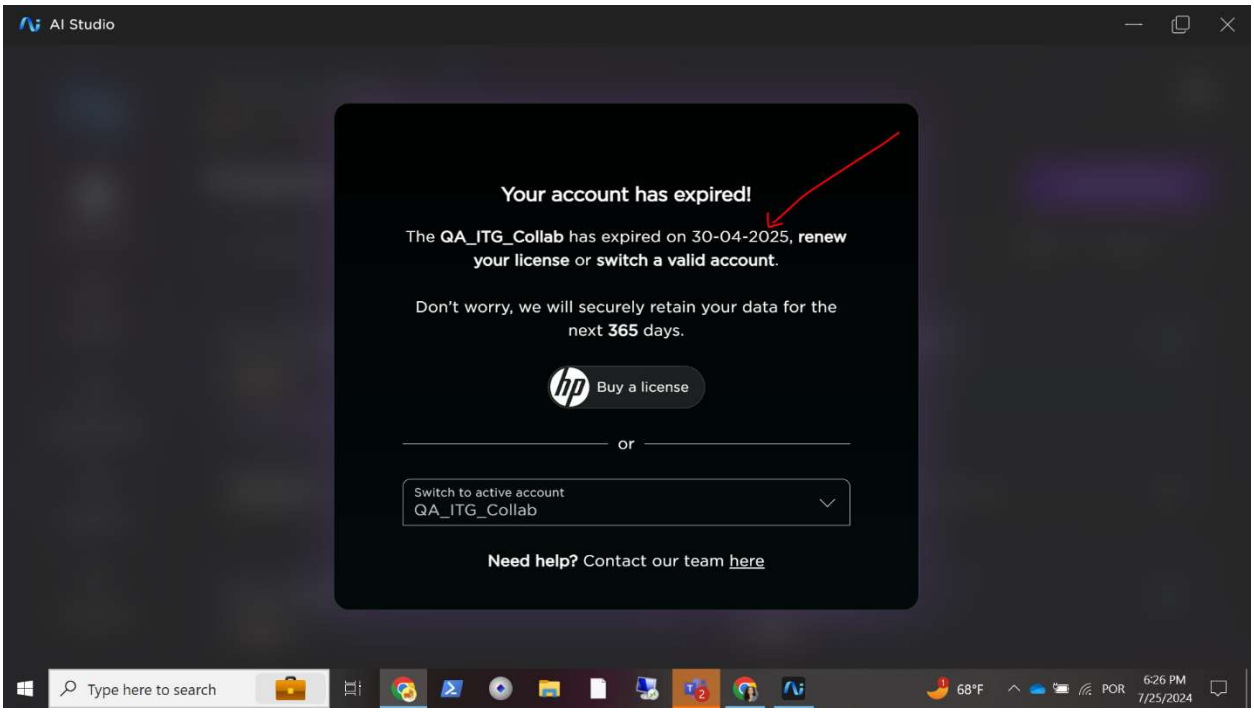
To purchase or renew an account:

1. Click the ‘Upgrade your plan’ link in the top left corner of your screen if your account is expring or has expired or click your profile icon and then the ‘Buy new account’ button anytime.

2. Complete form to confirm the account owner contact information and number of seats required

3. Click ‘Contact Sales’

4. HP sales will contact the account owner to support the request

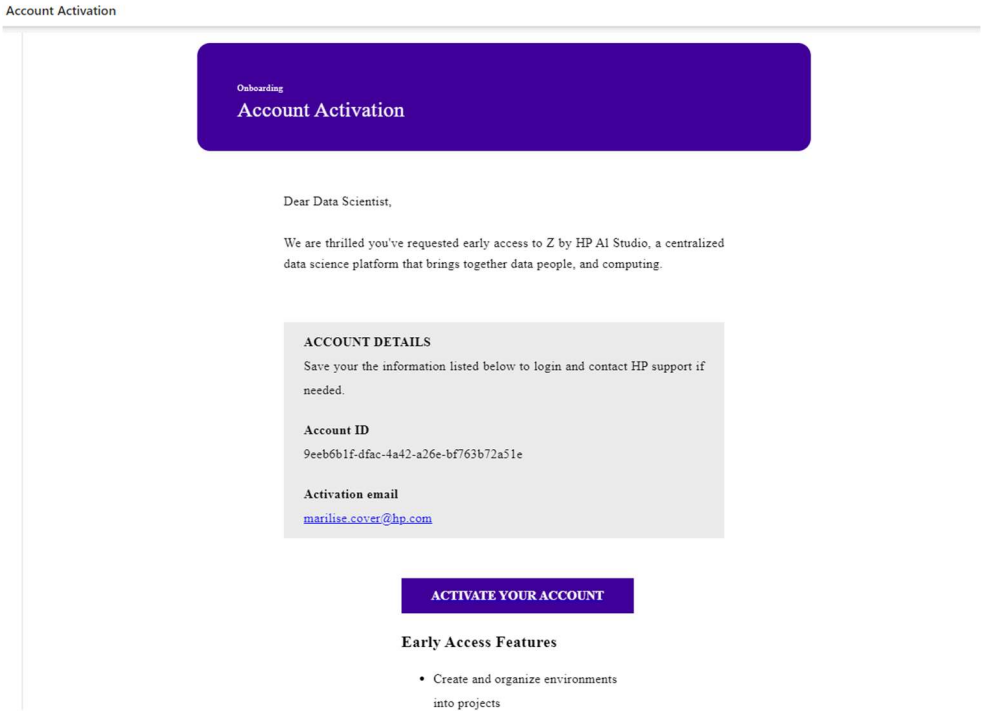


Admin: Activating an Account

The first user to activate their account on a new license is designated as the Account Owner. Account Owners have unique responsibilities, like configuring the account and inviting other Team Managers.

To activate an account:

1. Follow the link in the second email you receive from *aistudio@hp.com*.



2. From the entitlements portal, click **New Account**

3. Enter your team's Account, User, and Agent information in the provided fields.

   Note: Use the *Admin Emails* field for the account users you wish to assign to the role of Team Manager. Team Managers can reassign user roles after account configuration from inside AI Studio.

4. Click **Save** to finish configuring your team.

Once you sign in to AI Studio you can finish setting up your team and invite users to collaborate.

Note: If you're already signed in, sign out your HP account before activating another one.

Contributor: Activating an Account

When you're invited to a new team on AI Studio, you'll receive an email to alert you and help you get started. After a quick setup, you'll be steps away from harnessing your team's true computational potential with AI Studio.

To activate an account:

1. From your email inbox, open the message from aistudio@hp.com and click the link to **Activate your account**.
2. From your browser, sign in with your HPID and download the latest version of the application.
3. Run the installer and follow the setup wizard to install AI Studio.
4. Run the application and sign in again if prompted.

   Note: Depending on your browser settings, a second sign-in may not be necessary.

Now you're ready to get started with AI Studio.

Note: If you're already signed in, sign out your HP account before activating another one.

Admin: Changing User Roles

Team Managers have certain permissions in AI Studio that other contributing team members do not. They can add or remove team members, edit a team member's role, and approve requests to join the team.

To change a user's role:

1. From the ***Account*** tile, navigate to *Team Settings* and find the user you want to edit.

2. Click on the **more options** button on the right of the user's row.

3. Click **Edit role.**

4. Select the new role you want to assign to the user, then click **Save** to change the user's role.

   Note: Contributors promoted to Team Manager may be unable to remove users until their next log in.

Admin: Approving Invite Requests [draft]

To approve an invite request:

5. From the ***Account*** tile, navigate to *Team Settings* and click **Requests**.
6. Approved requests will send an activation email to collaborators and assign an account seat
7. Approved requests will appear in **Pending** until the invitation is activated.
8. ??

Registering a New Model

New models are automatically registered in AI Studio when you run a model with compatible hooks embedded in the code. Currently, only hooks embedded for Tensorflow or Pytorch are registerable.

NOTE: Only models registered in MLflow can be deployed for local inference on the *Deployments* tab.

To set a Pytorch hook for ML Flow:

1. Name the hook.
2. Define the hook's model metrics (scikit learn).
3. Start the MLflow run.
4. Use ???
5. Use cuda
6. Run the model.

Tip: Visit MLflow's website for detailed documentation on MLflow features.

To set a Tensorflow hook for Tensorboard:

1. Open your script in a notebook tab in a (conda:envi root) kernel.
2. Import the libraries necessary for Tensorboard by entering them into your script's code.
3. Clean out any old log files in the kernel.
4. Run the model.

Tip: Visit Tensorboard's website for detailed documentation on MLflow features.

If the hooks are properly embedded, they will appear in the *Deployments* tab and can be deployed to your workspaces. AI Studio leverages Swagger's API tools to expose your models to a RESTful API that lets you create, save, and experiment with your data in new ways.

Jobs (draft)*

[from RN draft] With Jobs, AI Studio continues to work for you even after you've punched out for the day. Maximize your productivity by freeing up your on-clock time for tasks that need your active attention. With Jobs, users can: [Hyperlink bullets to related docs?]

- Schedule a job run to begin on a certain date and time.
- Specify a job's run frequency.
- View a job's status, duration, and logs to determine its health.
- Allocate hardware and workspace assets to a job.

Creating a New Job [Needs Review]

To create a new job:

1. Click on **Create New Job**.

   Note: You need to clone the git repository you want to run the job on if you haven't already before you can continue.

2. Name, date, and schedule your job.

   Note: Enter time in *ddmmyyy* format.

3. Choose a workspace to run the job on.

4. Select a Branch / Tag or Commit from the dropdown menu.

   Note: These outputs depend on the git repository associated with the job.

5. Enter the script you want your job to execute in the script path field.

   Note: You must select a Branch / Tag or Commit to make this field editable.

6. Add any arguments you wish to apply to the job in the **Script Arguments** window.

7. Click on **Review and Create** to run a quick check to make sure you've satisfied all the fields before moving on to the next step.

8. Choose the GPU(s) you want to allocate to the Job.

9. Add the variables you want to set in the job runner at runtime.

10. Review your job, then click **Create Job** to save your configuration.

    - Weird UI bug is making the Jobs panel ‘twitch’ every 10 or so seconds (v1.35.3)

Job Management [Needs Review]

The ***Jobs*** window has UI features designed to make scheduling, finding, and running jobs quick and easy. Use the filter icon to

Use the more options icon at the end of a job's row to:

- ### Edit a Job
  Use **Edit Job** from the more options menu to update your configuration.

- ### Activating / Deactivate a Job
  Use **(De)Activate Job** from the more options menu to activate or deactivate your job.

  > **Warning**: If you deactivate your job, the scheduled job will not start and you will not be able to start the job until you manually reactivate the job.

- ### Delete a Job
  Use **Delete Job** to remove the Job from your list.

- ### Run a Job
  Use **Run Now** from the more options menu to immediately deploy your job. You can also deploy your model by clicking the start icon.

  > **Note**: Scheduling a job for a time in the past runs the job immediately after it's created. You can schedule jobs for any past hour that is not before the current day.

- ### View a Previous Run
  Use **See Last Run** to view the python and system logs, output artifacts, job details, and job results associated with the job you ran most recently.

Registering a Model

This guide outlines the steps necessary to register a model to the MLflow Model Registry, which provides an organized structure for model lifecycle management, including versioning, annotations, and staging or production-ready deployment.

By following these steps, you can successfully register, manage, and deploy machine learning models using MLflow's Model Registry. The Model Registry not only simplifies model management but also ensures traceability and consistency across versions, making it an indispensable tool for machine learning operations (MLOps). Let's get started with registering a model.

Step 0: Install Required Dependencies

To use MLflow and its model registry, you first need to install the MLflow library. AI Studio workspace images automatically do this for you. To re-install, start a workspace and open your terminal or command prompt and run the following command:

```
pip install --upgrade mlflow
```

This ensures that MLflow and all of its necessary dependencies are installed and up-to-date.

Step 1: Train and Log the Model

Before registering a model, it must be logged using MLflow's logging methods. For example, if you are using Scikit-learn, you can follow this process to embed hooks in your notebook:

1. Start an MLflow Run: This is a container for tracking parameters, metrics, and artifacts (like models).
2. Train a Model: Train your machine learning model using your preferred framework.
3. Log the Model: Use mlflow..log_model() to log your model to the tracking server.

```
import mlflow.sklearn

from sklearn.ensemble import RandomForestRegressor

# Example: Train and log a RandomForest model

with mlflow.start_run():

    model = RandomForestRegressor(n_estimators=100)

    model.fit(X_train, y_train)
```

```
# Log the model
      mlflow.sklearn.log_model(model, "random_forest_model")
```

In this case, the RandomForestRegressor model is trained and then logged to MLflow as a Scikit-learn model flavor.

Step 2: Register the Model

Now that the model has been logged, it can be registered to the Model Registry.

MLflow offers two ways to register models:

1. Register While Logging: You can log and register the model simultaneously by specifying the registered_model_name parameter:

   ```
   mlflow.sklearn.log_model(model, "random_forest_model",
   registered_model_name="RandomForestModel")
   ```

2. Register After Logging: If you have already logged a model and wish to register it afterward, use mlflow.register_model(). First, obtain the model's URI and then register it:

   ```
   model_uri = "runs://random_forest_model"
   mlflow.register_model(model_uri, "RandomForestModel")
   ```

Both approaches add the model to the Model Registry, where it can be versioned, annotated, and transitioned between stages (e.g., Staging, Production).

Step 3: View and Manage the Model

Once registered, you can view and manage your model from the MLflow UI or programmatically. To access the MLFlow UI, simply click the Monitor tab in your project.

- Via the MLflow UI: You can access the model by navigating to the "Models" section in the MLflow UI. Here, you can see all versions of the model, assign stages (e.g., Staging, Production), and add descriptions or annotations.

- Programmatically: You can interact with the Model Registry programmatically using the MLflow client API.

Example of listing all versions of a model:

```
from mlflow.tracking import MlflowClient client = MlflowClient() versions =
client.get_registered_model("RandomForestModel").latest_versions for version in
versions: print(f"Version: {version.version}, Stage: {version.current_stage}")
```

Step 4: Transition Model Stages

MLflow provides a robust mechanism to manage model versions and their lifecycle. You can move models across different stages: None, Staging, Production, or Archived.

For example, to move a model version to Production:

```
client.transition_model_version_stage(
        name="RandomForestModel",
        version=1,
        stage="Production"
)
```

This helps manage models that are ready for deployment and tracks their lifecycle status.

Step 5: Load a Registered Model

Once the model has been registered, it can be loaded for inference using mlflow.pyfunc.load_model() or the specific flavor you logged. For example:

```
model = mlflow.pyfunc.load_model(model_uri="models:/RandomForestModel/Production")
```

You can replace "Production" with "Staging" or specify a particular version.

How to Save a Model as a Python Function

MLflow provides the ability to log a Python function as a model using the python_function model flavor. This is a flexible format that allows any Python model or code to be registered and deployed. Here's how you can do it:

Function-Based Model

If you have a simple function that you want to log, you can use the mlflow.pyfunc.log_model() method. For example:

```
import mlflow
import pandas as pd

# Define a simple predict function
def predict(model_input):
 return model_input.apply(lambda x: x * 2)

# Log the function as a model
with mlflow.start_run():
 mlflow.pyfunc.log_model("model", python_model=predict, pip_requirements=["pandas"])
 run_id = mlflow.active_run().info.run_id

# Load the model and perform inference
model = mlflow.pyfunc.load_model(f"runs:/{run_id}/model")
x_new = pd.Series([1, 2, 3])
prediction = model.predict(x_new)
print(prediction)
```

This logs the predict function as a model, which can then be loaded and used for inference.

Class-Based Model

If you need more flexibility or power, such as custom preprocessing or complex logic, you can create a class that implements the predict method and log it:

```
import mlflow

import pandas as pd

class MyModel(mlflow.pyfunc.PythonModel):

def predict(self, context, model_input):
    return [x * 2 for x in model_input]


# Log the class-based model

with mlflow.start_run():

      mlflow.pyfunc.log_model("model", python_model=MyModel(),
      pip_requirements=["pandas"]) run_id = mlflow.active_run().info.run_id

# Load the model and perform inference

model = mlflow.pyfunc.load_model(f"runs:/{run_id}/model")

x_new = pd.Series([1, 2, 3])
```

```
print(model.predict(x_new))
```

The class-based approach allows you to define more complex models with custom logic.

Loading and Using the Python Function Model

After saving the model as a Python function (either function-based or class-based), you can load and use it just like any other MLflow model.

```
# Load the Python function model

model = mlflow.pyfunc.load_model("runs://model")

# Use it to predict on new data

result = model.predict(input_data)
```

This is particularly useful when integrating custom Python models into the MLflow ecosystem, allowing them to be deployed just like traditional machine learning models.

> **Tip**: Once your models are registered to MLFlow, you can publish them for local inference using the deployment tab in your project.

Model Flavors Supported by MLflow

MLflow supports a wide variety of model "flavors" or formats, such as:

- Scikit-learn (mlflow.sklearn)
- TensorFlow (mlflow.tensorflow)
- PyTorch (mlflow.pytorch)
- XGBoost (mlflow.xgboost)
- Python Function (mlflow.pyfunc) A generic format for Python models.

Each flavor has specific logging methods (e.g., mlflow.sklearn.log_model()) to log models into MLflow's Model Registry. You can choose the flavor based on your framework to ensure compatibility during registration and inference.

Example Model Training and Publishing [Needs Review]

In this example, we'll show you how to train a model with TensorFlow and register it with MLflow in AI Studio. Our goal is to build a breast cancer classification model, track experiments, and publish the results. We'll use a public dataset to demonstrate each step of the process: starting with data exploration, then training, and finally using MLflow for registration and publishing.

Step 1: Install Required Libraries

If you're using a custom workspace, install the necessary requirements by either adding them to a requirements file or manually installing them with these commands:

```
import pandas as pd
import numpy as np
import mlflow.tensorflow
```

You will need `mlflow`, `tensorflow`, `pandas`, `seaborn`, and other related libraries. If not already installed, you can use:

```
pip install mlflow tensorflow pandas seaborn
```

You can also use predefined workspaces that already have many of these dependencies installed, including MLflow, to help you get started faster.

Step 2: Load and Explore the Data

We'll begin by loading the breast cancer dataset:

```
df = pd.read_csv('../data/cancer_classification.csv')
```

Check the dataset's information to understand the feature types and view missing values with the following commands:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   mean radius              569 non-null    float64
 1   mean texture             569 non-null    float64
```

```
 2   mean perimeter           569 non-null    float64
 3   mean area                569 non-null    float64
 4   mean smoothness          569 non-null    float64
 5   mean compactness         569 non-null    float64
 6   mean concavity           569 non-null    float64
 7   mean concave points      569 non-null    float64
 8   mean symmetry            569 non-null    float64
 9   mean fractal dimension   569 non-null    float64
 10  radius error             569 non-null    float64
 11  texture error            569 non-null    float64
 12  perimeter error          569 non-null    float64
 13  area error               569 non-null    float64
 14  smoothness error         569 non-null    float64
 15  compactness error        569 non-null    float64
 16  concavity error          569 non-null    float64
 17  concave points error     569 non-null    float64
 18  symmetry error           569 non-null    float64
 19  fractal dimension error  569 non-null    float64
 20  worst radius             569 non-null    float64
 21  worst texture            569 non-null    float64
 22  worst perimeter          569 non-null    float64
 23  worst area               569 non-null    float64
 24  worst smoothness         569 non-null    float64
 25  worst compactness        569 non-null    float64
 26  worst concavity          569 non-null    float64
 27  worst concave points     569 non-null    float64
 28  worst symmetry           569 non-null    float64
 29  worst fractal dimension  569 non-null    float64
 30  benign_0__mal_1          569 non-null    int64
dtypes: float64(30), int64(1)
memory usage: 137.9 KB
```

To view descriptive statistics for a summary of each feature:

```
df.describe().transpose()
```

	count	mean	std	min	25%	50%	75%	max
mean radius	569.0	14.127292	3.524049	6.981000	11.700000	13.370000	15.780000	28.11000
mean texture	569.0	19.289649	4.301036	9.710000	16.170000	18.840000	21.800000	39.28000
mean perimeter	569.0	91.969033	24.298981	43.790000	75.170000	86.240000	104.100000	188.50000
mean area	569.0	654.889104	351.914129	143.500000	420.300000	551.100000	782.700000	2501.00000
mean smoothness	569.0	0.096360	0.014064	0.052630	0.086370	0.095870	0.105300	0.16340
mean compactness	569.0	0.104341	0.052813	0.019380	0.064920	0.092630	0.130400	0.34540
mean concavity	569.0	0.088799	0.079720	0.000000	0.029560	0.061540	0.130700	0.42680
mean concave points	569.0	0.048919	0.038803	0.000000	0.020310	0.033500	0.074000	0.20120
mean symmetry	569.0	0.181162	0.027414	0.106000	0.161900	0.179200	0.195700	0.30400
mean fractal dimension	569.0	0.062798	0.007060	0.049960	0.057700	0.061540	0.066120	0.09744
radius error	569.0	0.405172	0.277313	0.111500	0.232400	0.324200	0.478900	2.87300
texture error	569.0	1.216853	0.551648	0.360200	0.833900	1.108000	1.474000	4.88500
perimeter error	569.0	2.866059	2.021855	0.757000	1.606000	2.287000	3.357000	21.98000
area error	569.0	40.337079	45.491006	6.802000	17.850000	24.530000	45.190000	542.20000
smoothness error	569.0	0.007041	0.003003	0.001713	0.005169	0.006380	0.008146	0.03113
compactness error	569.0	0.025478	0.017908	0.002252	0.013080	0.020450	0.032450	0.13540
concavity error	569.0	0.031894	0.030186	0.000000	0.015090	0.025890	0.042050	0.39600
concave points error	569.0	0.011796	0.006170	0.000000	0.007638	0.010930	0.014710	0.05279
symmetry error	569.0	0.020542	0.008266	0.007882	0.015160	0.018730	0.023480	0.07895
fractal dimension error	569.0	0.003795	0.002646	0.000895	0.002248	0.003187	0.004558	0.02984
worst radius	569.0	16.269190	4.833242	7.930000	13.010000	14.970000	18.790000	36.04000
worst texture	569.0	25.677223	6.146258	12.020000	21.080000	25.410000	29.720000	49.54000
worst perimeter	569.0	107.261213	33.602542	50.410000	84.110000	97.660000	125.400000	251.20000
worst area	569.0	880.583128	569.356993	185.200000	515.300000	686.500000	1084.000000	4254.00000
worst smoothness	569.0	0.132369	0.022832	0.071170	0.116600	0.131300	0.146000	0.22260
worst compactness	569.0	0.254265	0.157336	0.027290	0.147200	0.211900	0.339100	1.05800
worst concavity	569.0	0.272188	0.208624	0.000000	0.114500	0.226700	0.382900	1.25200
worst concave points	569.0	0.114606	0.065732	0.000000	0.064930	0.099930	0.161400	0.29100
worst symmetry	569.0	0.290076	0.061867	0.156500	0.250400	0.282200	0.317900	0.66380
worst fractal dimension	569.0	0.083946	0.018061	0.055040	0.071460	0.080040	0.092080	0.20750
benign_0__mal_1	569.0	0.627417	0.483918	0.000000	0.000000	1.000000	1.000000	1.00000

Step 3: Exploratory Data Analysis (EDA)

Understanding data distribution is an important step. We'll use `seaborn` and `matplotlib` commands to explore the example dataset.

Countplot of Target Variable

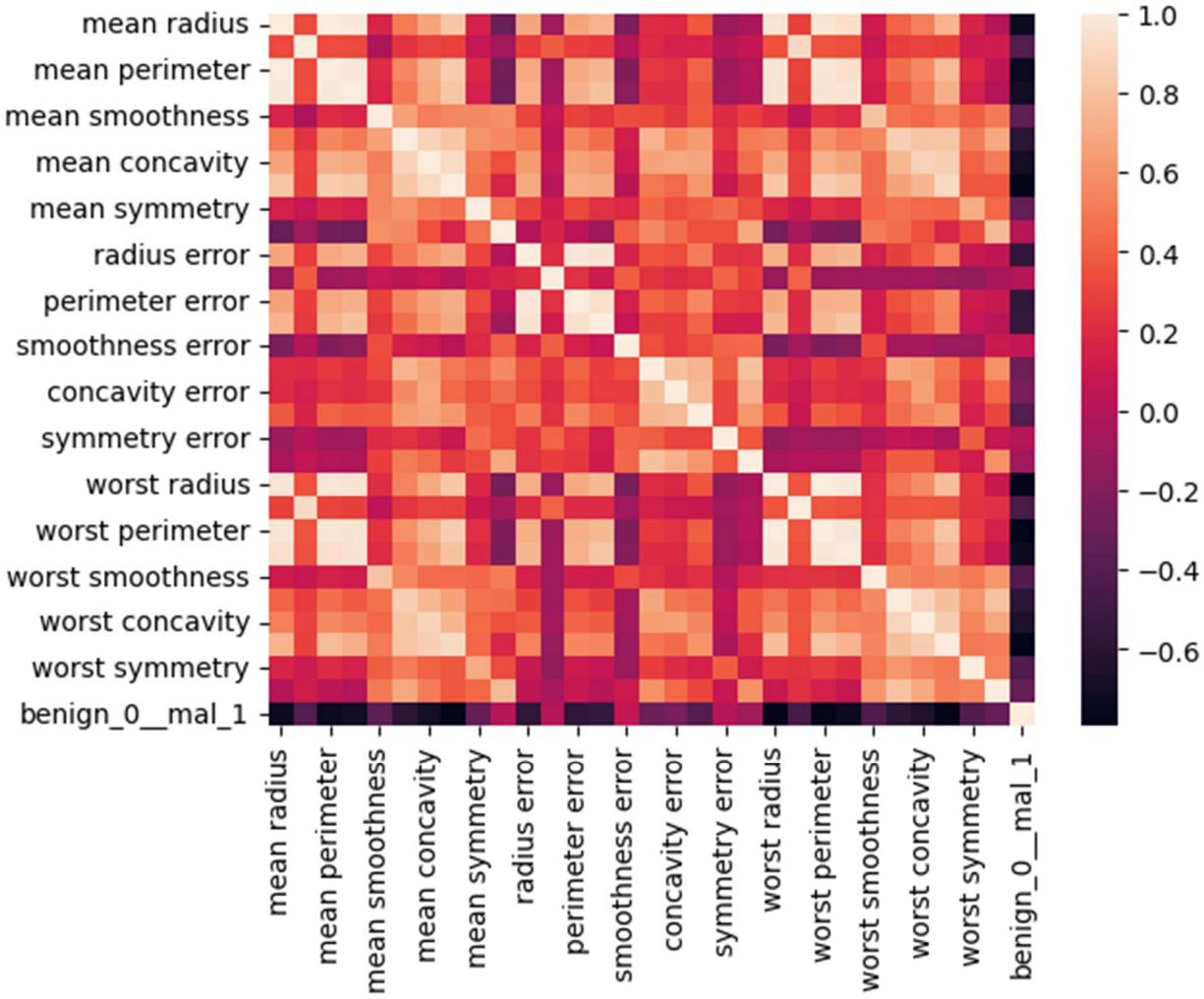
Plotting the count of benign and malignant cases helps visualize class distribution:

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.countplot(x='benign_0__mal_1', data=df)
plt.show()
```

Correlation Heatmap

Understanding feature correlation is key to building an effective model:

```
sns.heatmap(df.corr(), annot=True)
plt.show()
```



Step 4: Train-Test Split

Before we train the model, split the dataset into training and testing sets:

```
from sklearn.model_selection import train_test_split

X = df.drop('benign_0__mal_1', axis=1).values
y = df['benign_0__mal_1'].values

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.25, random_state=101)
```

Step 5: Data Scaling

Neural networks perform better with scaled data, so we'll scale features using `MinMaxScaler`:

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Step 6: Train a TensorFlow Model

To create and train a simple TensorFlow neural network model:

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout

model = Sequential([
    Dense(30, activation='relu'),
    Dropout(0.5),  # Add dropout to prevent overfitting
    Dense(15, activation='relu'),
    Dropout(0.5),  # Add dropout to prevent overfitting
    Dense(1, activation='sigmoid')
])
```

```
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])

# Use EarlyStopping to prevent overtraining
early_stopping =
tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=5,
restore_best_weights=True)

model.fit(X_train, y_train, epochs=50, validation_data=(X_test,
y_test), callbacks=[early_stopping])
```

To avoid overfitting, we use techniques like **Dropout layers**, which randomly deactivate neurons during training to help the model with generalization. Use **Early Stopping** to stop training when validation loss stops improving, preventing the model from overtraining and saving you time and resources.

For the purposes of this example, we'll continue on without using those overfitting reduction techniques and move on to how to log and register the model with MLFlow.

Step 7: Log the Model with MLflow

We can use MLflow to log our TensorFlow model and help us keep track of different versions and monitor performance:

```
mlflow.tensorflow.autolog()

with mlflow.start_run():
    model.fit(X_train, y_train, epochs=50,
validation_data=(X_test, y_test))
```

Using `mlflow.autolog()` automatically logs metrics, parameters, and the model itself.

Step 8: Register the Model in MLflow

Once you log the model, you can register it to the MLflow Model Registry to version the model and track its lifecycle.

To register while logging:

Use the `registered_model_name` parameter:

```
mlflow.tensorflow.log_model(model, "breast_cancer_model",
registered_model_name="BreastCancerModel")
```

To register after logging:

Use the following parameter:

```
model_uri = "runs:/<run-id>/breast_cancer_model"
mlflow.register_model(model_uri, "BreastCancerModel")
```

You can transition the model between stages like *Staging* and *Production* for use in deployment. Once registered, you can also publish the model in the ***Deployments*** tab for easy access and integration.

Step 9: Publish and Monitor the Model

After registering your model in MLflow, navigate to the ***Monitor*** tab in AI Studio to view its details, including versioning, performance metrics, and transition stages. This feature allows you to track your model's lifecycle from experimentation through to production and facilitates seamless deployment and monitoring.

Summary

In this example, we covered the entire workflow for training a breast cancer classification model using TensorFlow, logging it with MLflow, and registering it for easy management and deployment. Using MLflow's capabilities to track and register models ensures efficient model management and deployment across environments.

AI Studio provides a streamlined environment for working with MLflow, TensorFlow and many other machine learning tools, making it easy to track experiments, manage models, and publish them for use in production systems.

For more information on MLflow and its capabilities, you can visit the official MLflow documentation.

Feel free to extend this workflow to more complex models or datasets to meet your specific use cases.