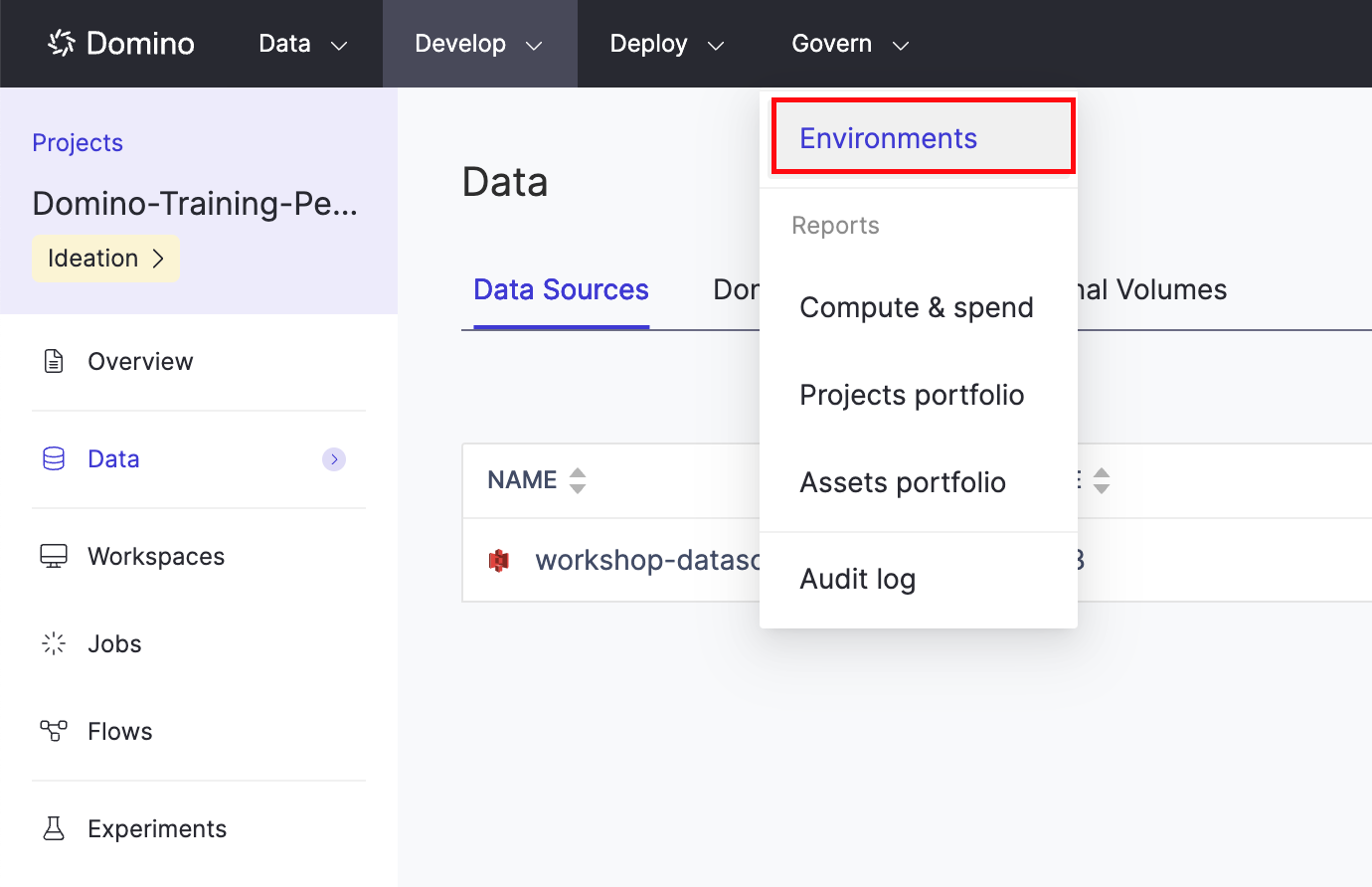
# Section 2

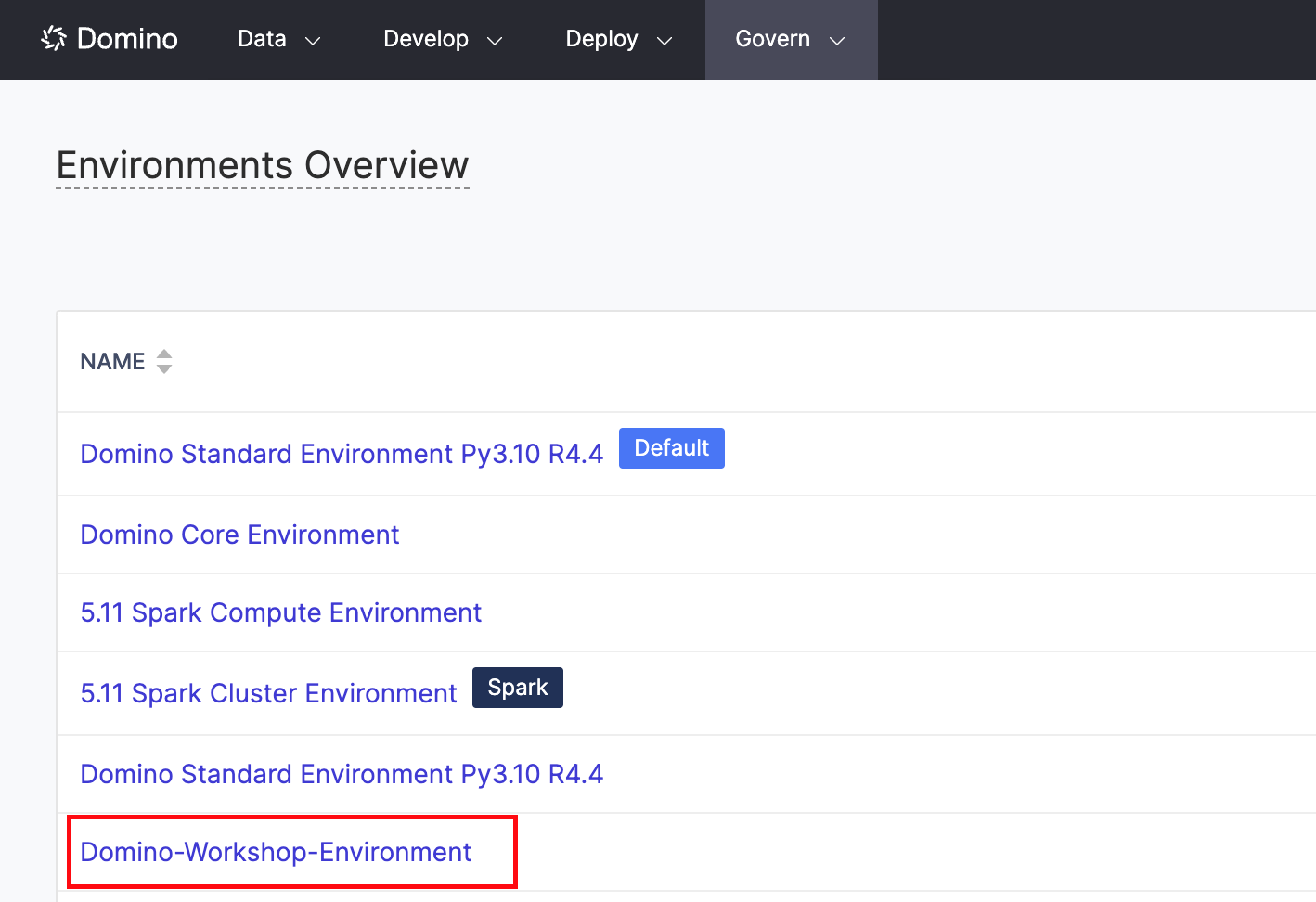
## Develop Model

### Lab 2.1 - Inspect Compute Environment

From the top menu, click on **Govern** and select **Environments**.



Select **Domino-Workshop-Environment**



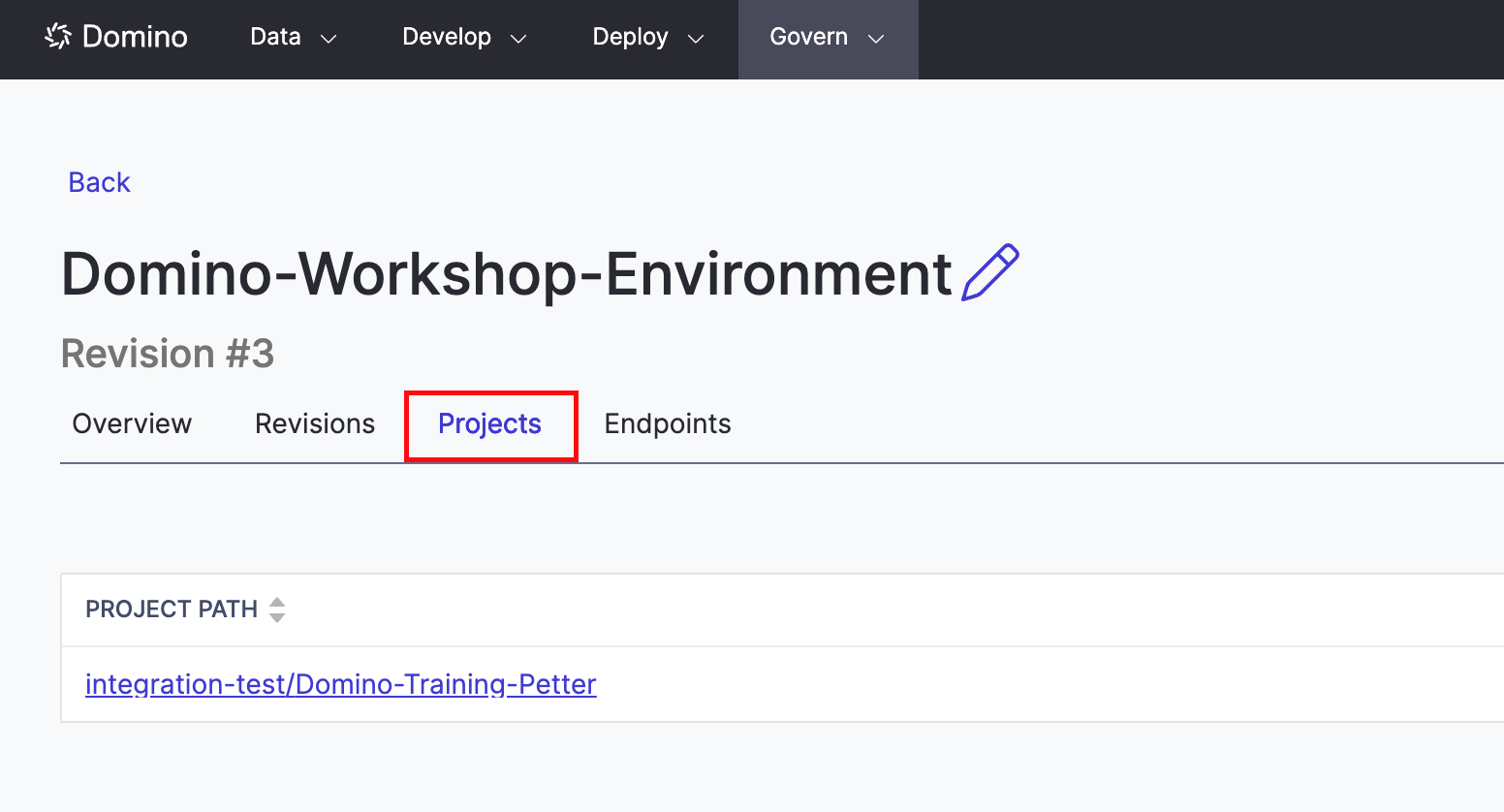
Inspect the Docker file to understand the packages installed, configurations specified, kernels installed, etc.

Scroll down to **Pluggable Workspaces Tools**. This is the area where IDEs are made available for end users.

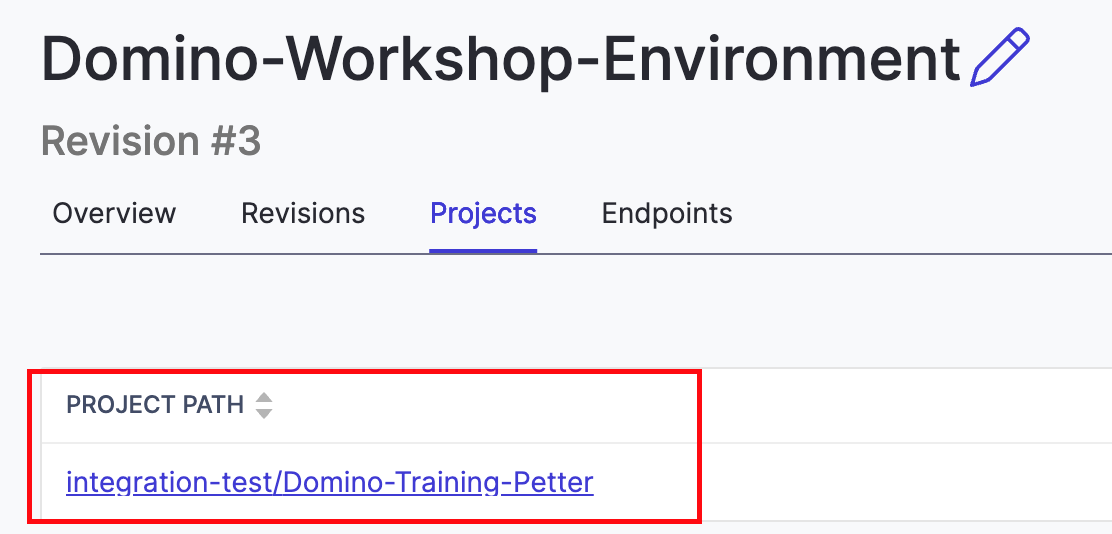
Scroll down to the **Run Setup Scripts** section.

Here, we have a script that executes upon the startup of workspace sessions or jobs (pre-run script) and a script that executes upon the termination of a workspace session or job (post-run script)

Finally, navigate to the **Projects** tab. You should see all the projects leveraging this computing environment.

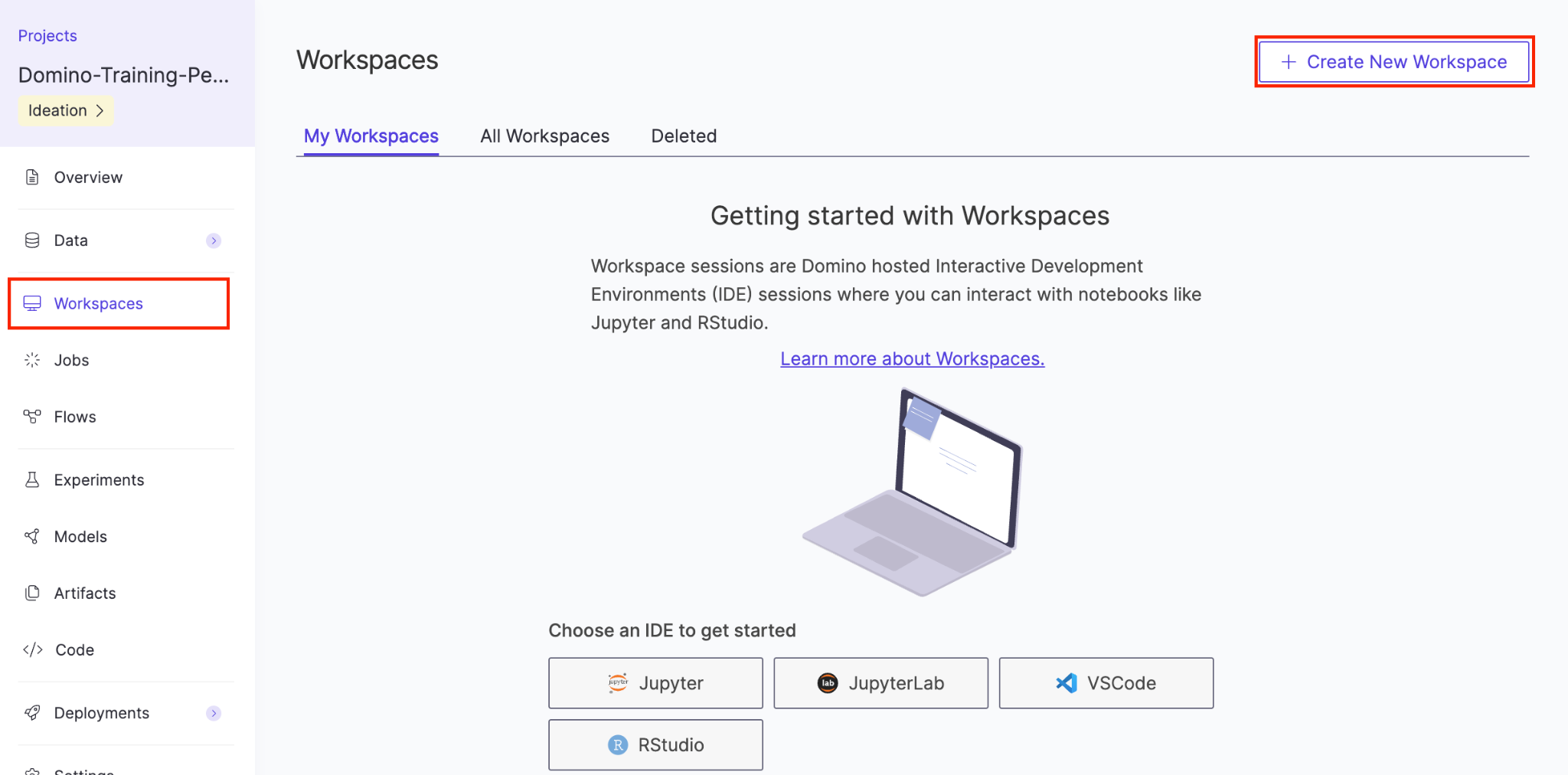


Under **PROJECT PATH**, click on your project {your-username/Domino-Training-name} name to prepare for the next lab.



### Lab 2.2 - Exploring Workspaces

Click into the **Workspaces** tab on the left, then in the top right corner, click **Create New Workspace**.



#### Workspace Name

Type a name for the Workspace in the cell

#### Workspace Environment

Ensure that **Domino-Workspace-Environment** is selected.

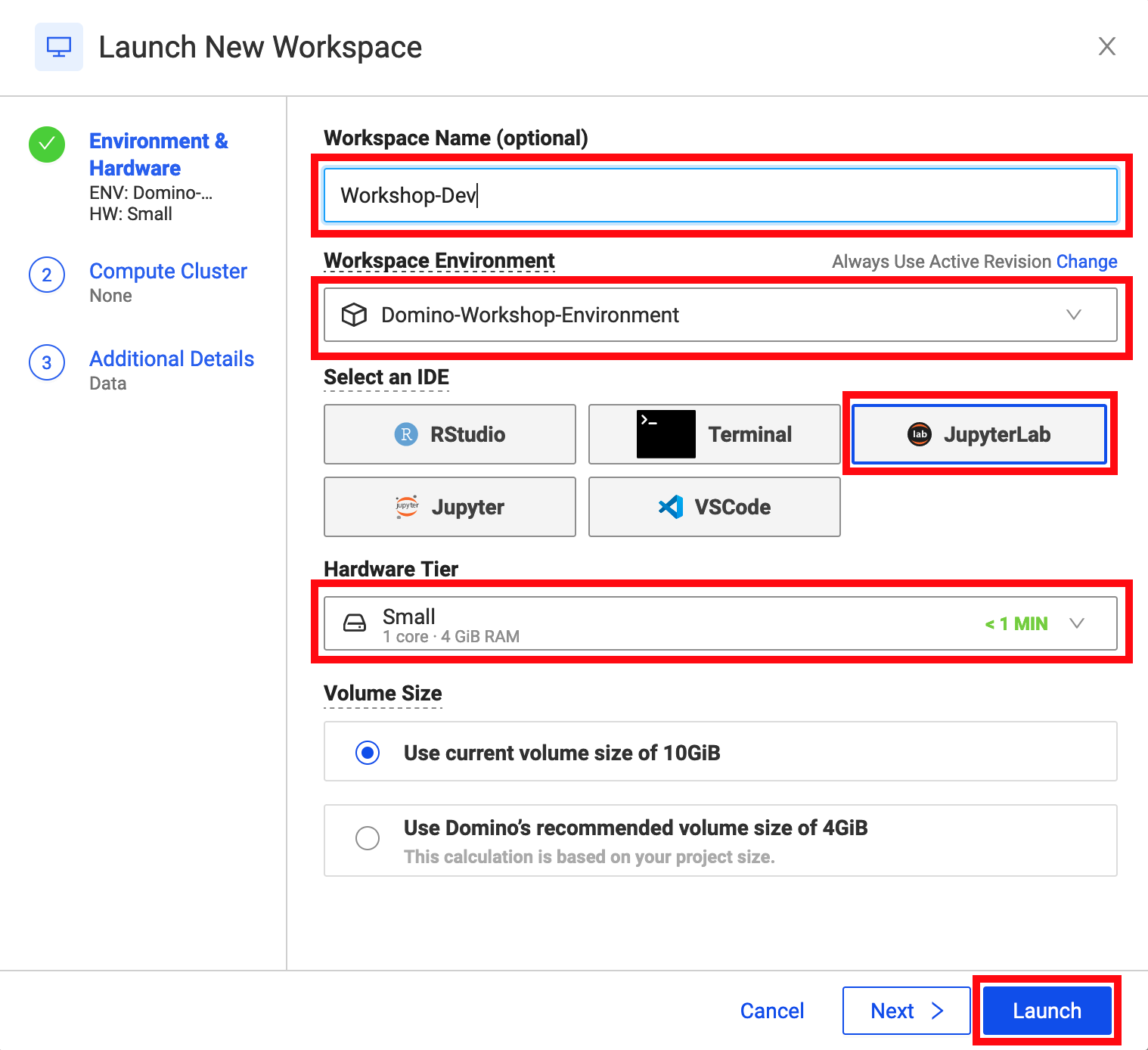
#### Select an IDE

Select **JupyterLab**

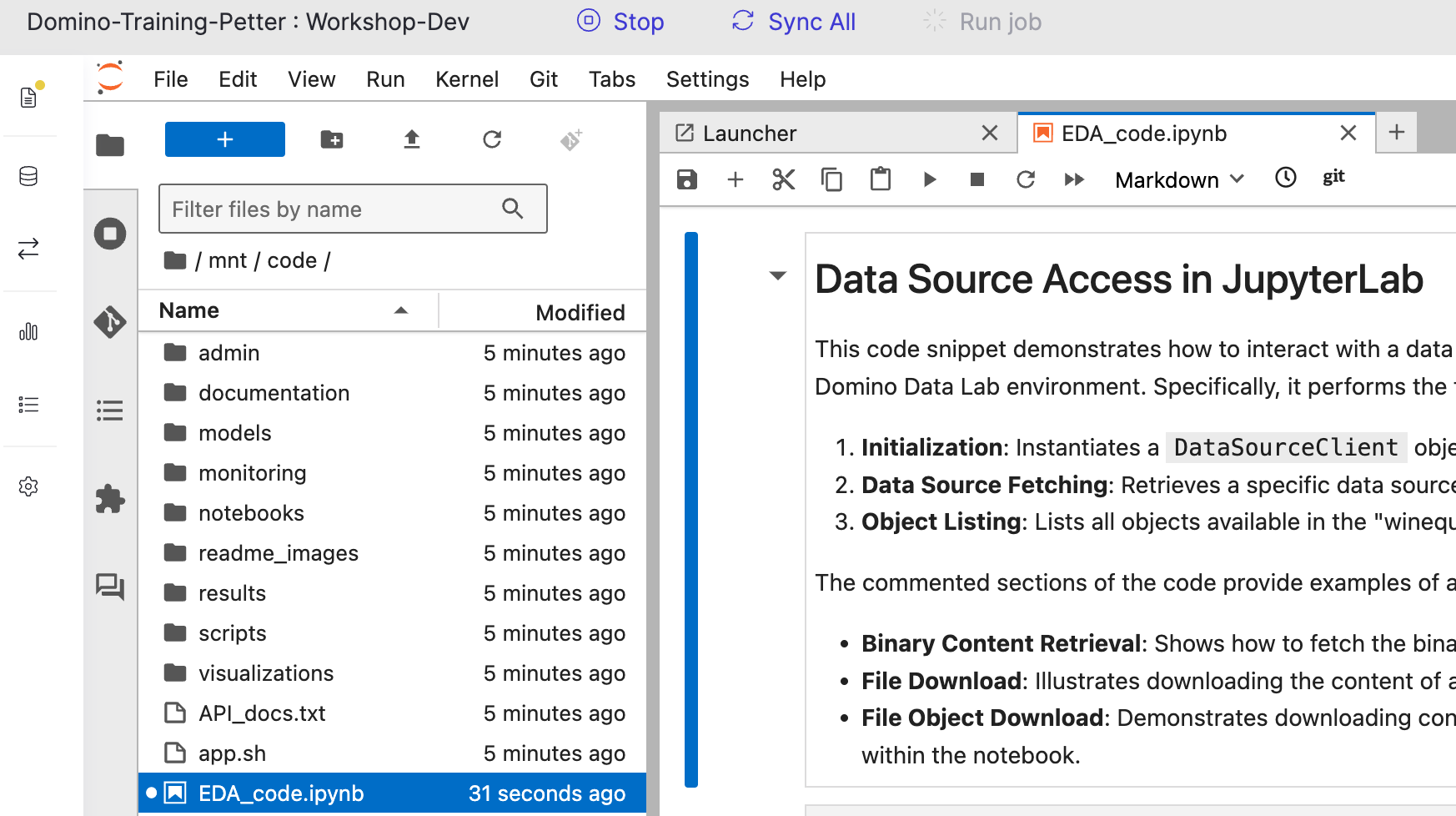
#### Hardware Tier

Ensure that **Small** is selected.

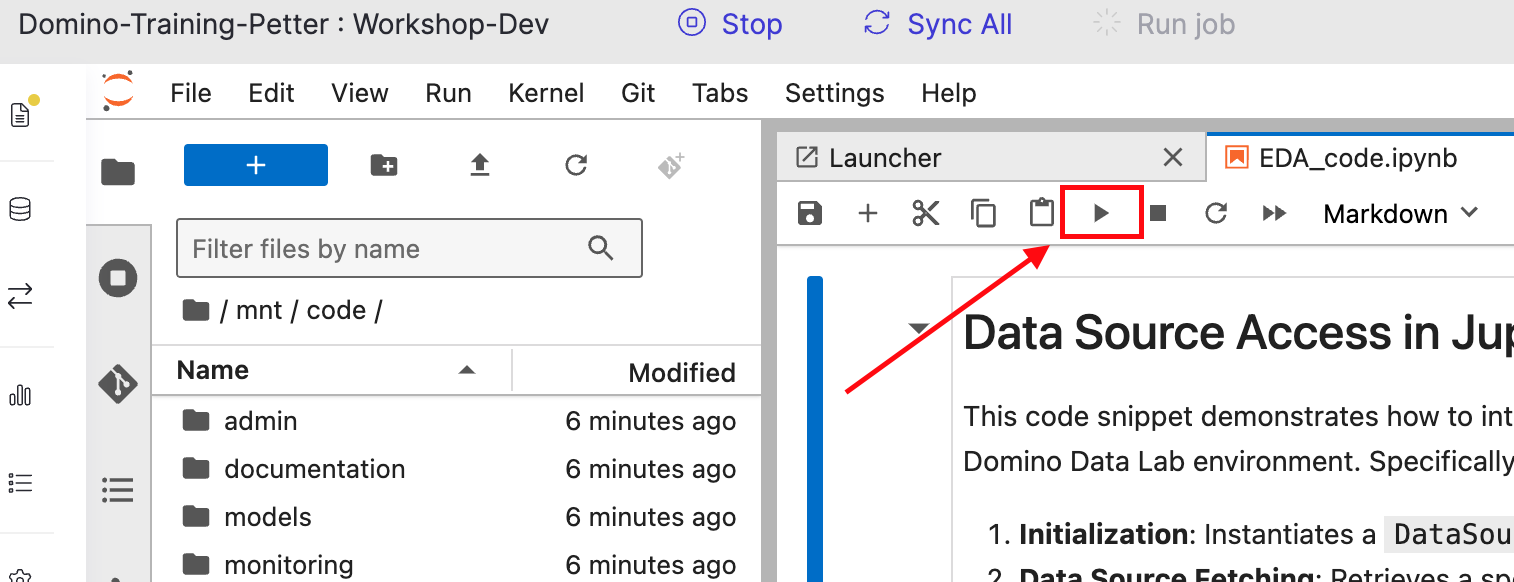
Click **Launch**



Once the workspace is launched, **Open** the notebook called **EDA\_code.ipynb**.

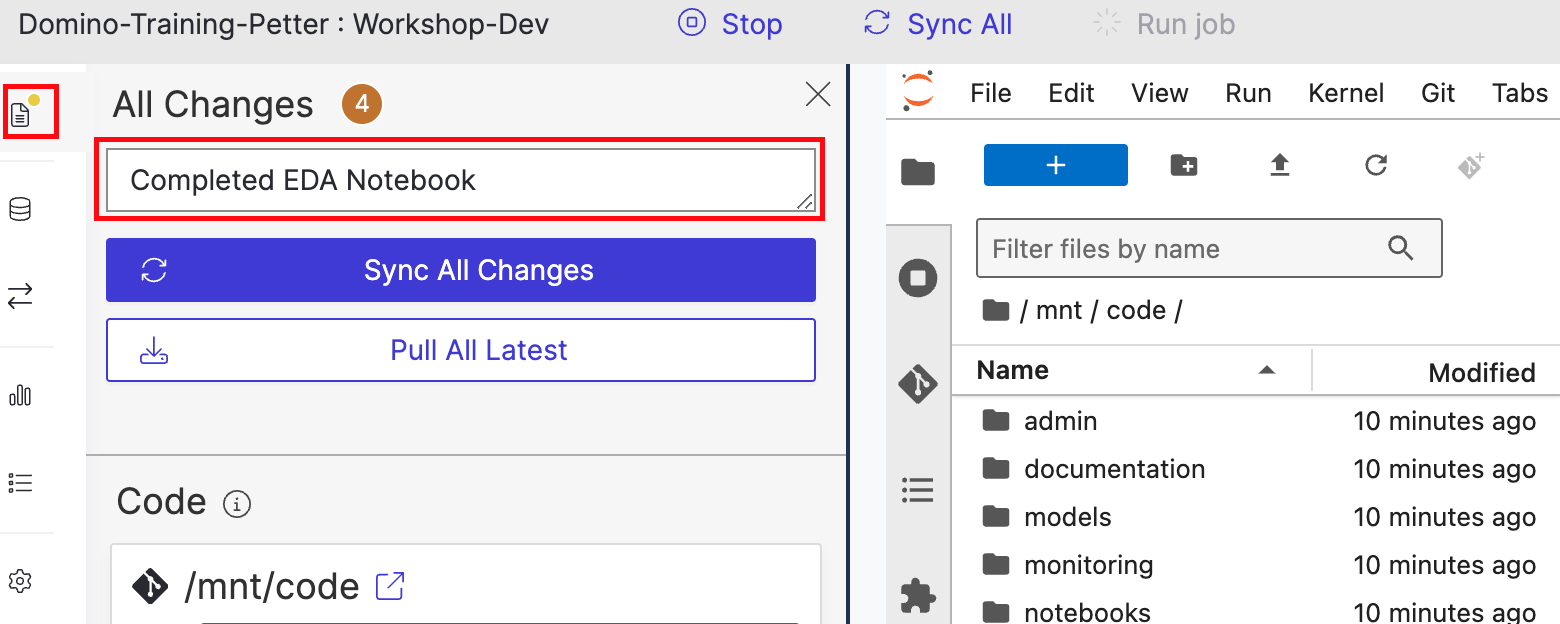


Read through the notebook and execute all cells one by one using the play button.



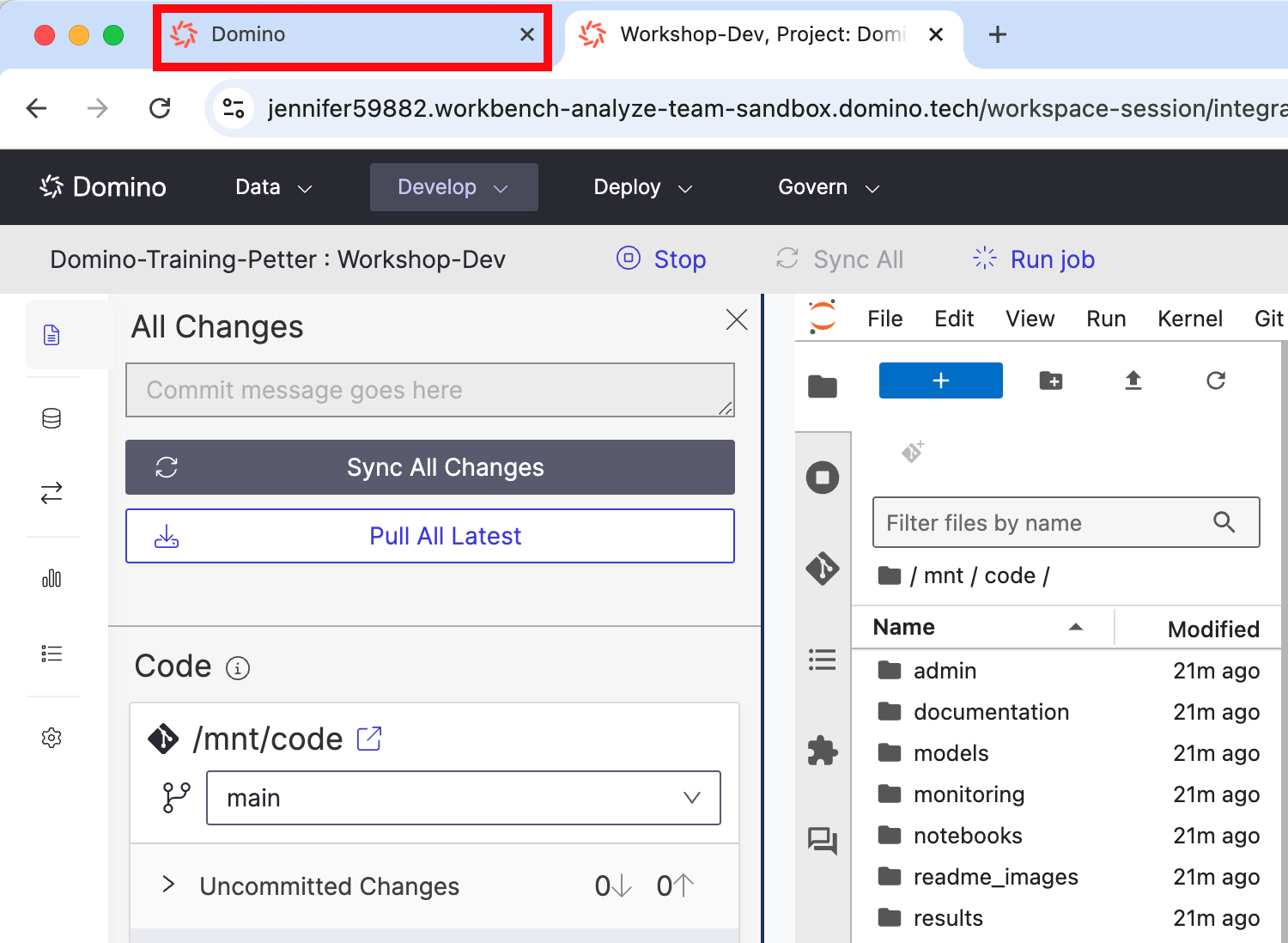
### Lab 2.3 - Syncing Files

Now that we've finished working on our notebook and writing data back to our project, we want to sync our latest work. To do so, click on the File Changes tab in the top left corner of your screen.

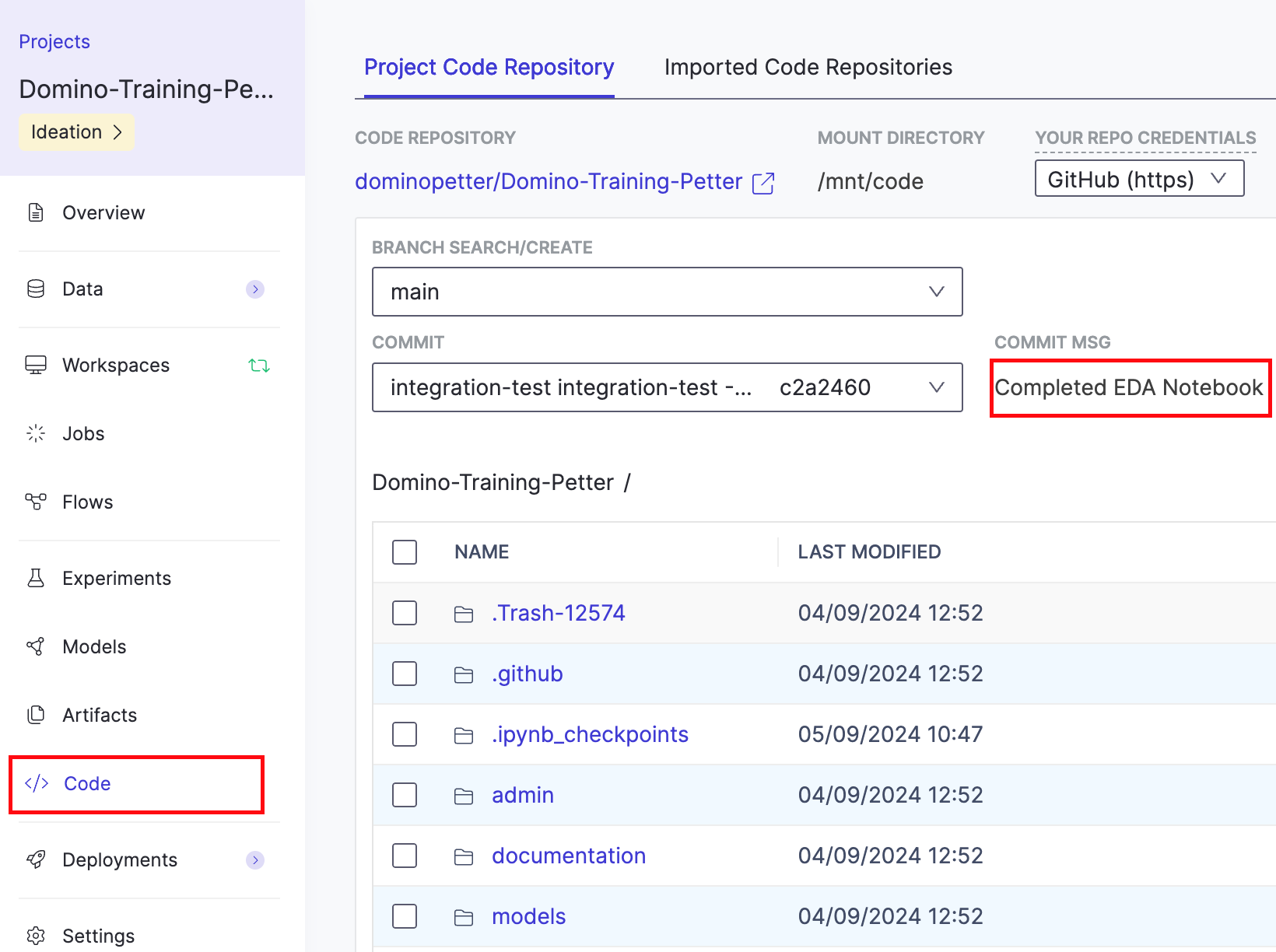


Enter an informative and brief commit message, such as **Completed EDA Notebook,** and click to **Sync All Changes**.

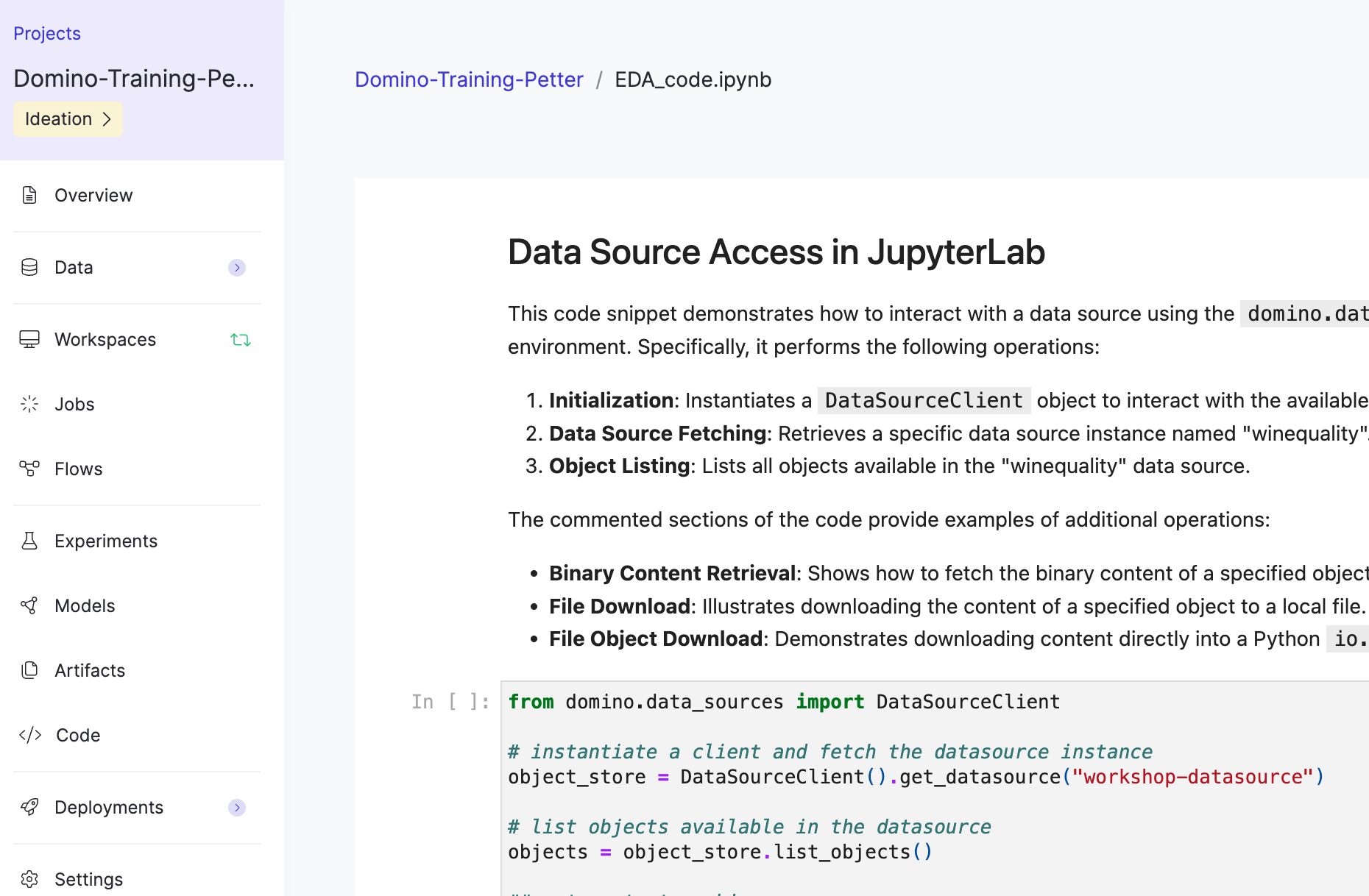
Click the **Domino Tab** next to your **Workspace Tab**, as shown below.



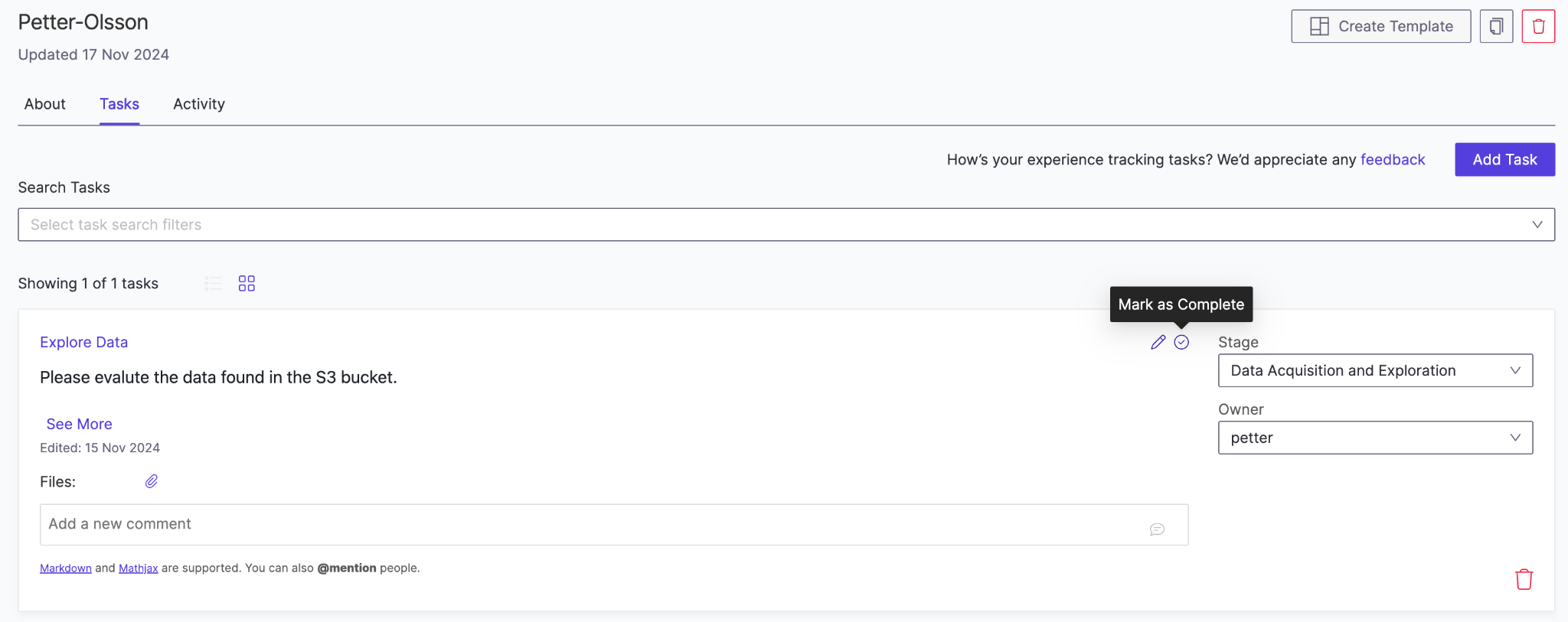
Select **Code**. The latest commit is reflected, and you can see the **EDA\_code.ipynb** in your file directory.



Click on the **EDA\_code.ipynb** file to view it.



Now navigate to **Overview** and **Tasks** andmark the task as complete.

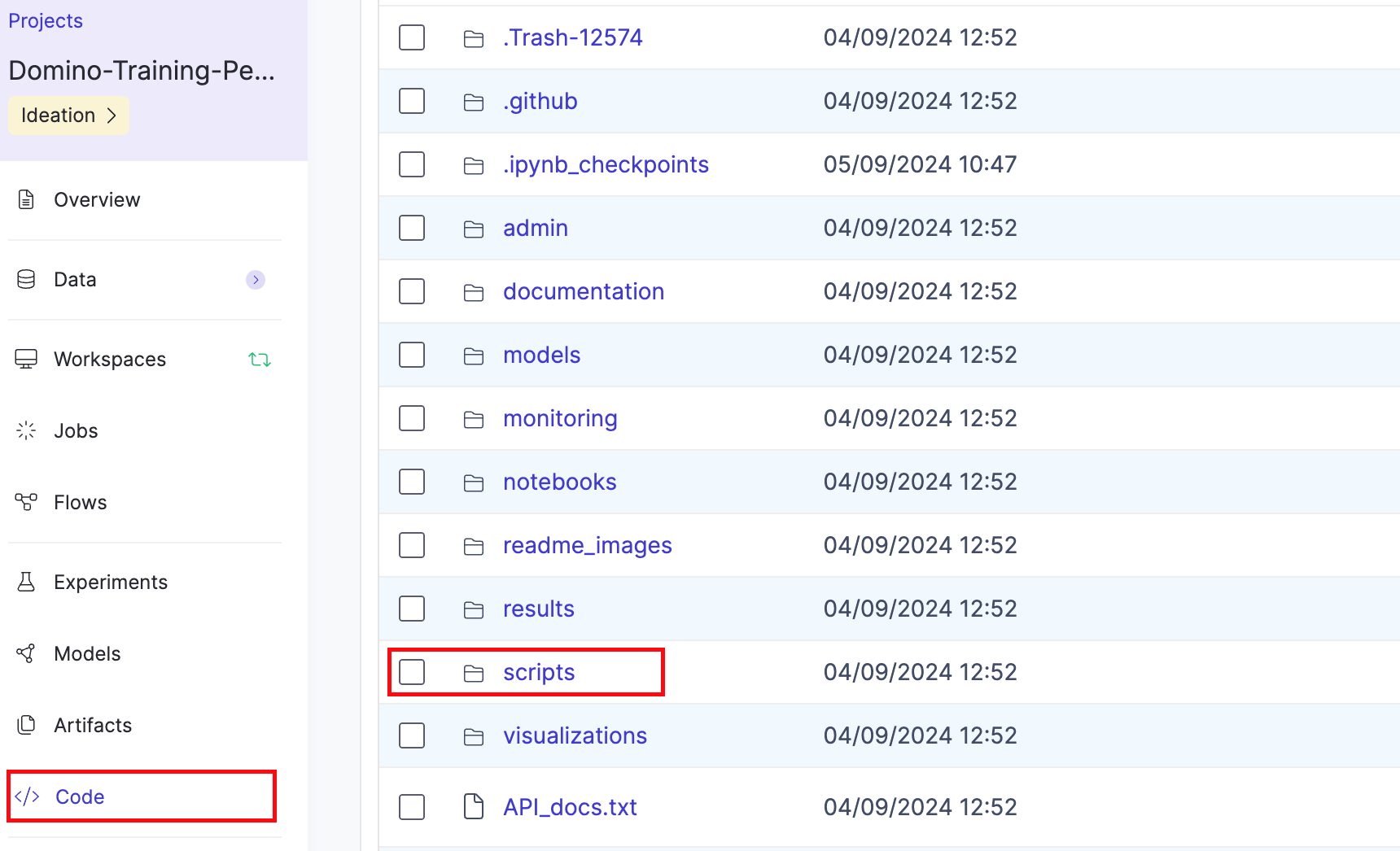


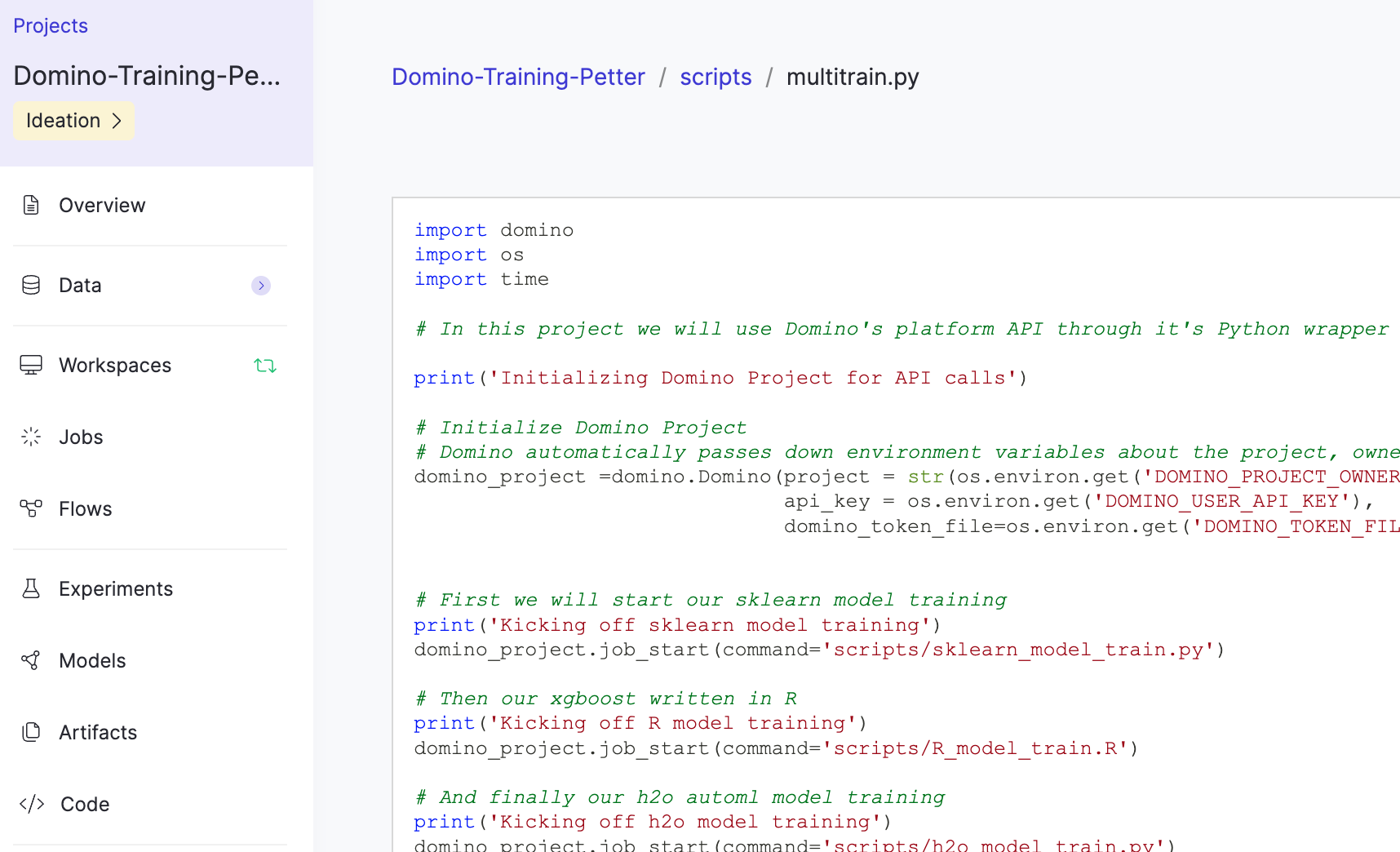
### Lab 2.4 - Run and Track Experiments

Now it's time to train our models!

We are taking a three-pronged approach and building a model in sklearn (Python), xgboost (R), and an auto-ml ensemble model (H2O).

First, navigate to your **Code** pane, enter the **Scripts** directory, and open **multitrain.py**.

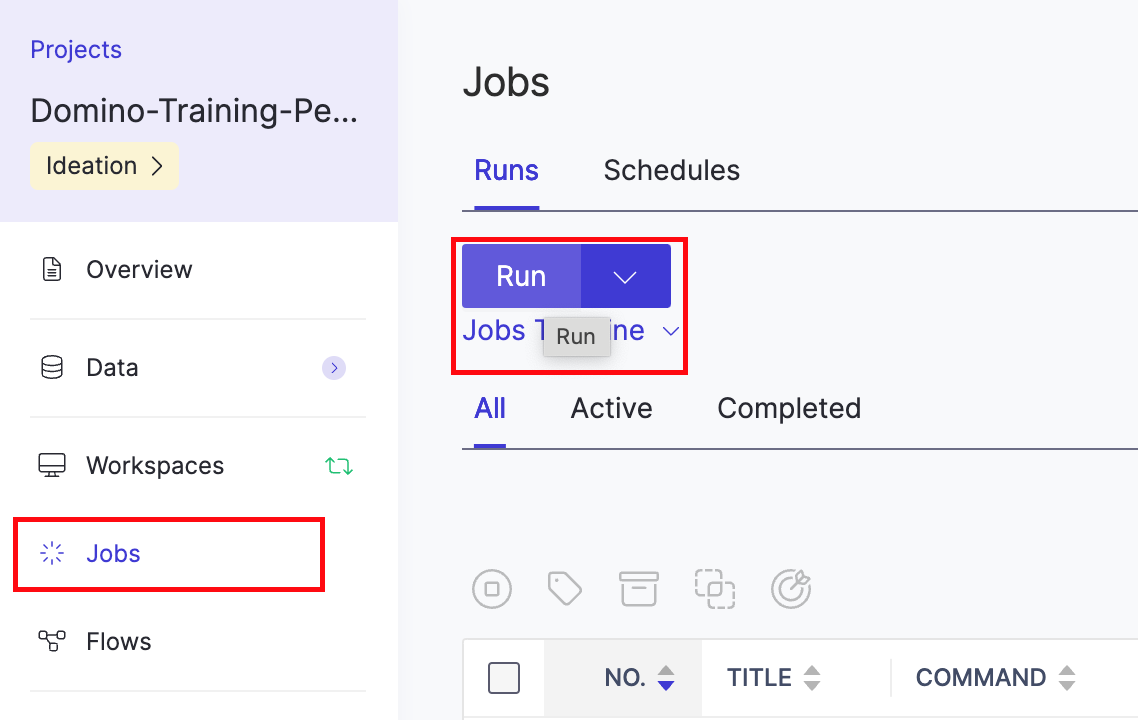




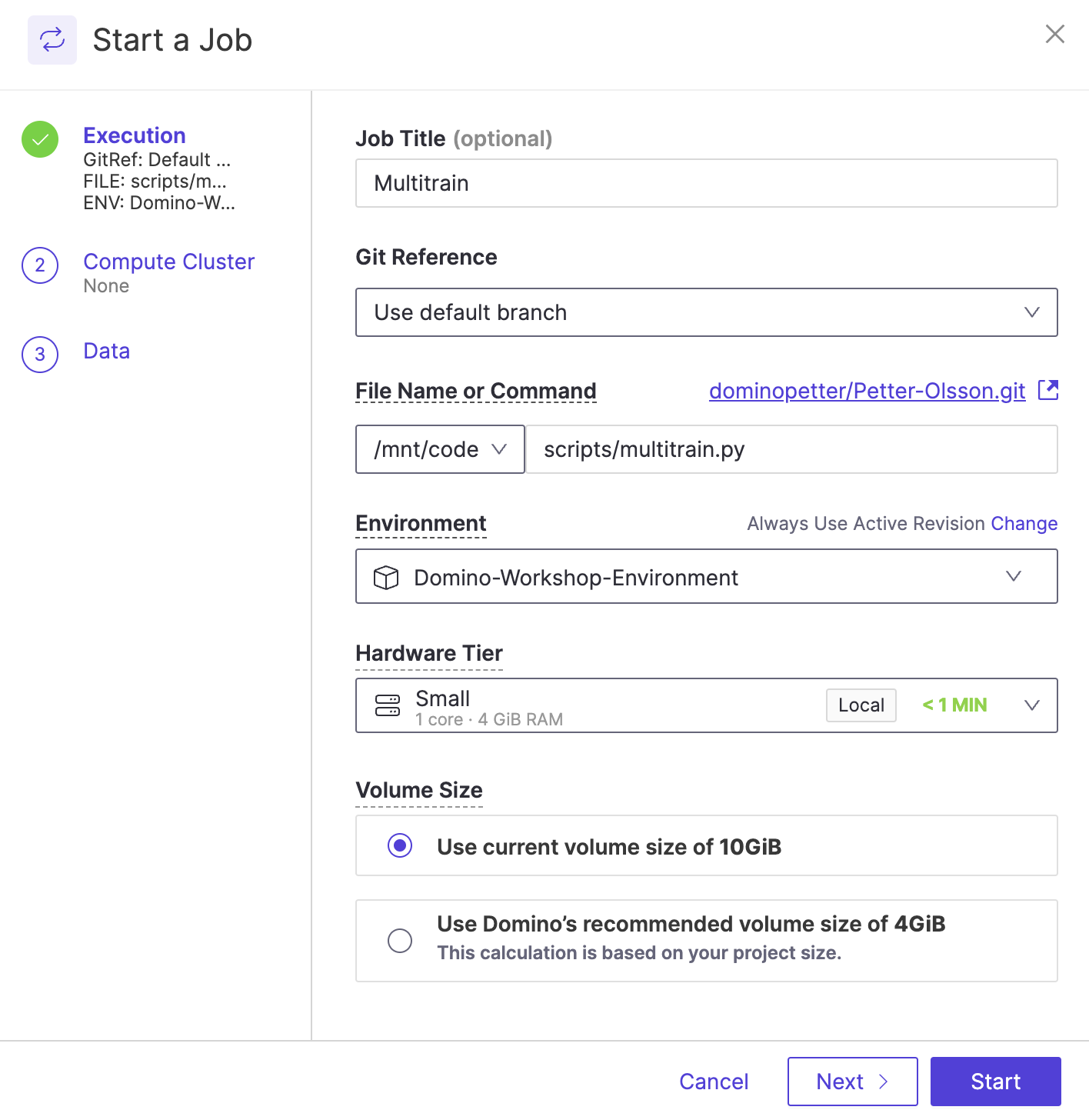
Check out the code in the script and comments describing the purpose of each line of code.

You can also check out any training scripts that **multitrain.py** will call.

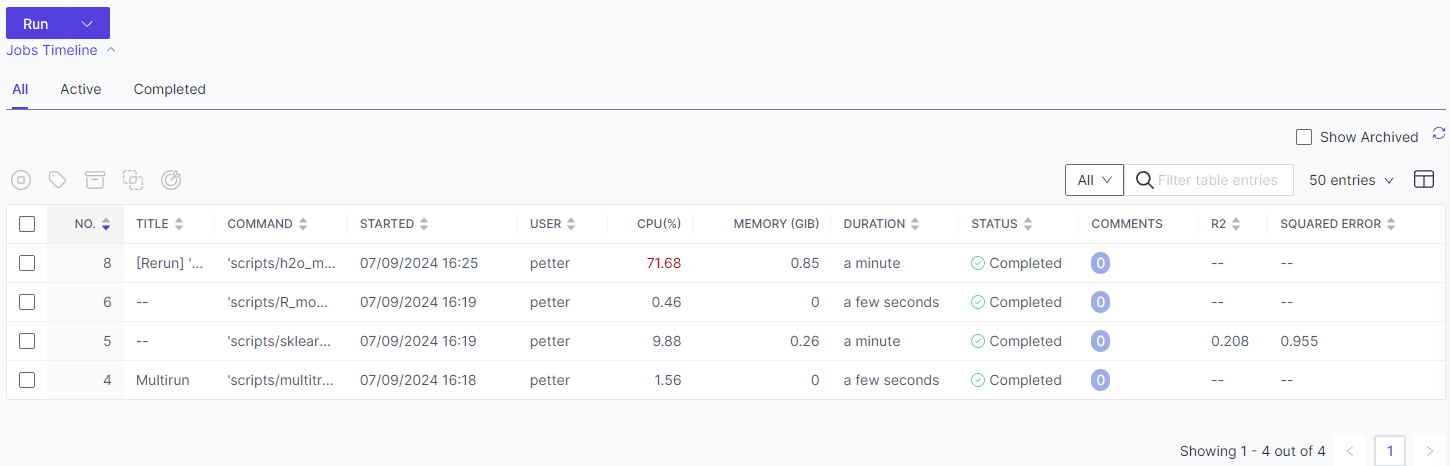
Navigate to the **Jobs** pane and click on **Run**.



Please fill in the information below. The **File Name or Command** will auto-populate when you start typing. When you are done, click Start.

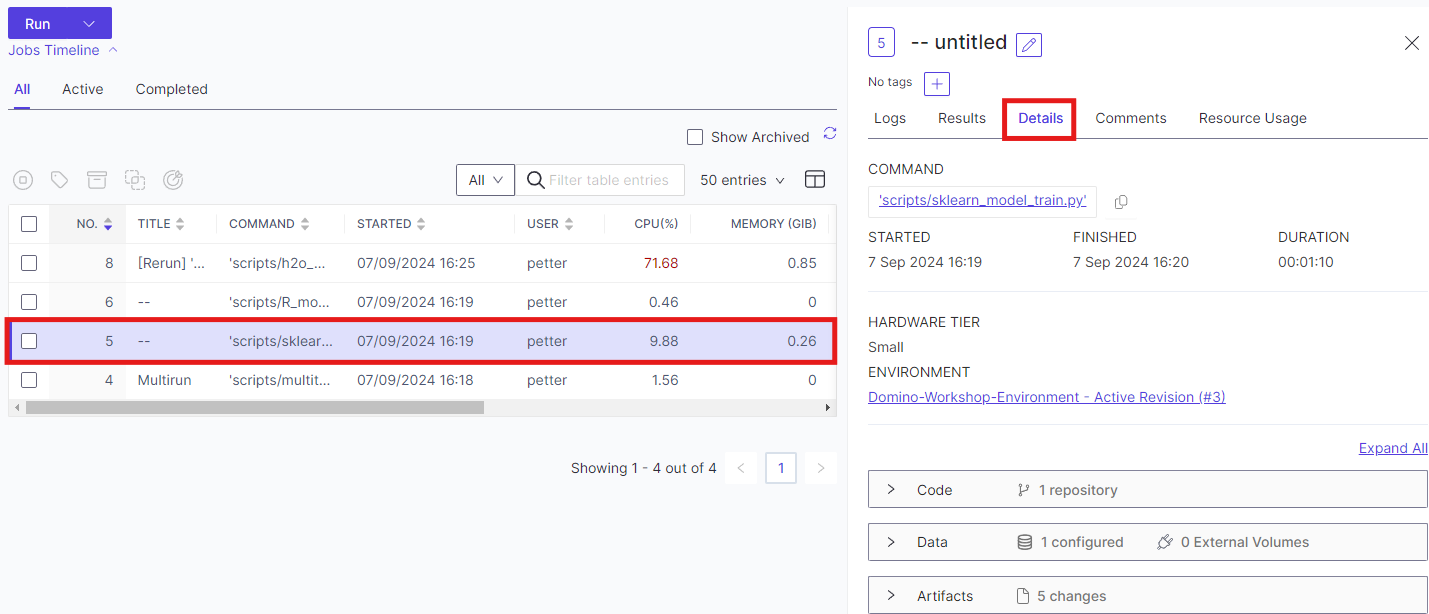


Your multirun script will spawn three executions, and you may see them in **starting**, **running,** or **completed** state.

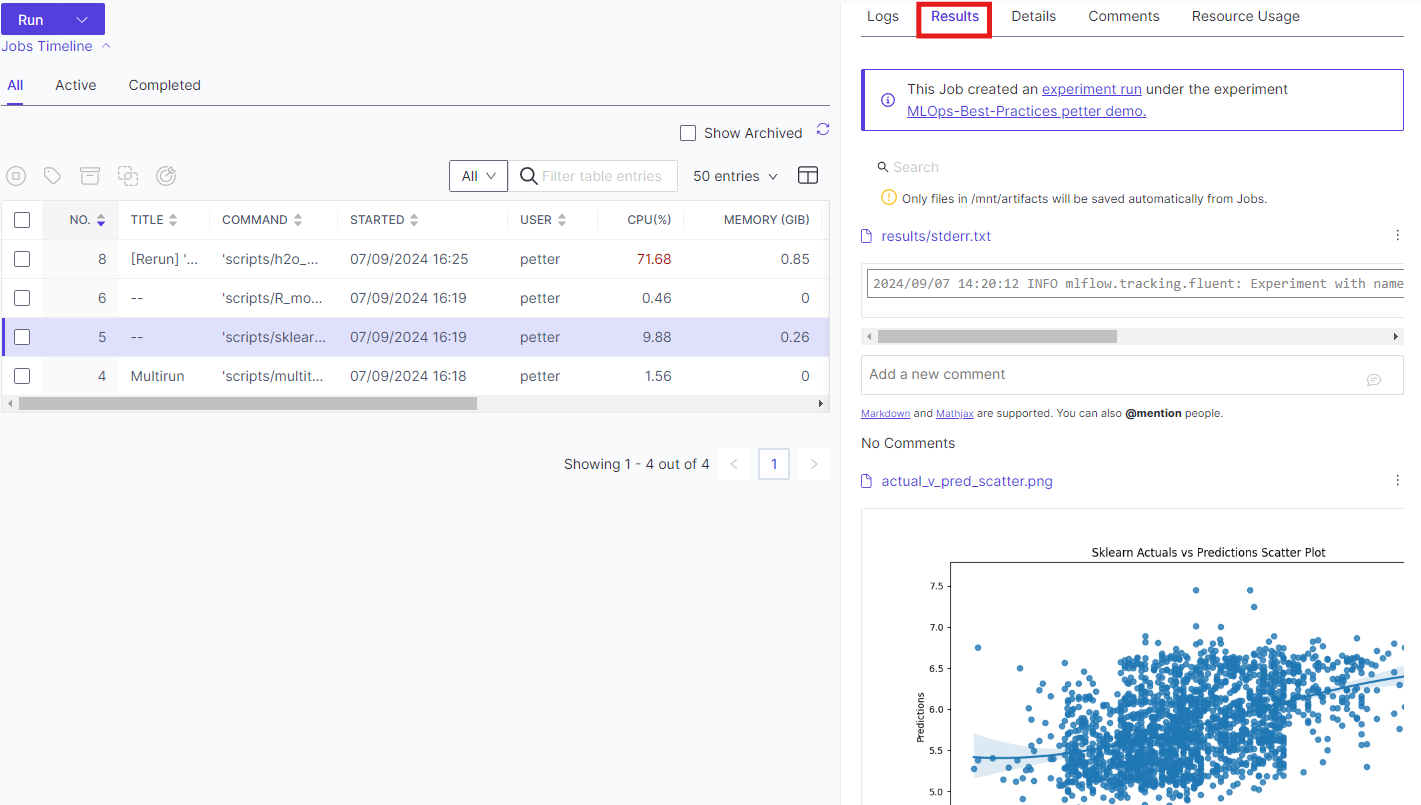


Click into the **sklearn\_model\_train.py** job run.

In the details tab of the job run, note that the **HARDWARE TIER** and **ENVIRONMENT** are tracked to document not only who ran the experiment and when but also what versions of the code, software, and hardware were executed.



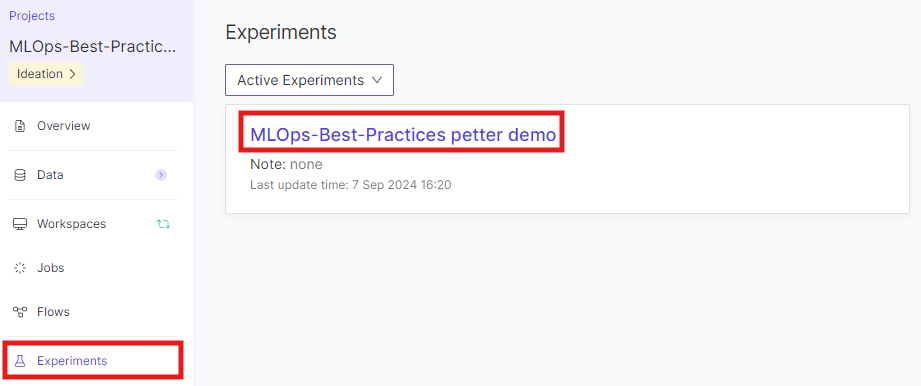
Click on the Job **Results** tab. Scroll down to view the visualizations and other outputs of the job.



We've trained three models, and it is time to select which one to deploy. Domino experiment management leverages **MLflow Tracking** to log experiment parameters, metrics, and artifacts easily. MLflow runs as a service in your Domino cluster, fully integrated within your workspace and jobs, and honors role-based access control. Existing MLflow experiments work right out of the box, requiring no code changes!

The jobs we just ran had MLFlow tracking to log the R^2 value and Mean Squared Error (MSE).

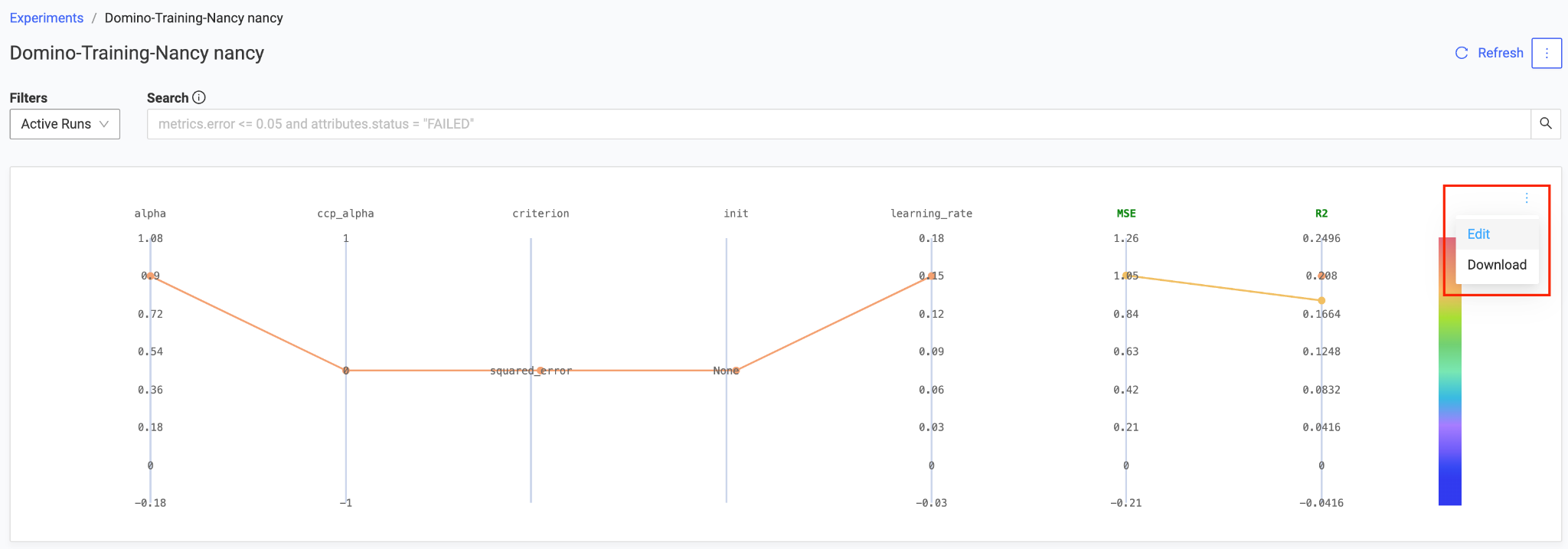
To view the experiments, click on the **Experiments** pane in your project. Here, you have one set of experiments against which all the jobs were logged.



**Click on the experiment name to see more details.**

Within the experiment, we can see three different runs corresponding to the three other jobs we created. Our code tags each with the framework used to make the model: H2O Automl, sklearn, and R. We also track the R2 value and Mean Squared Error (MSE). Our visualization currently shows only the R2 value. Let's update it to show both R2 and MSE to get a better view of our models.

Click on the three dots and choose **Edit.**

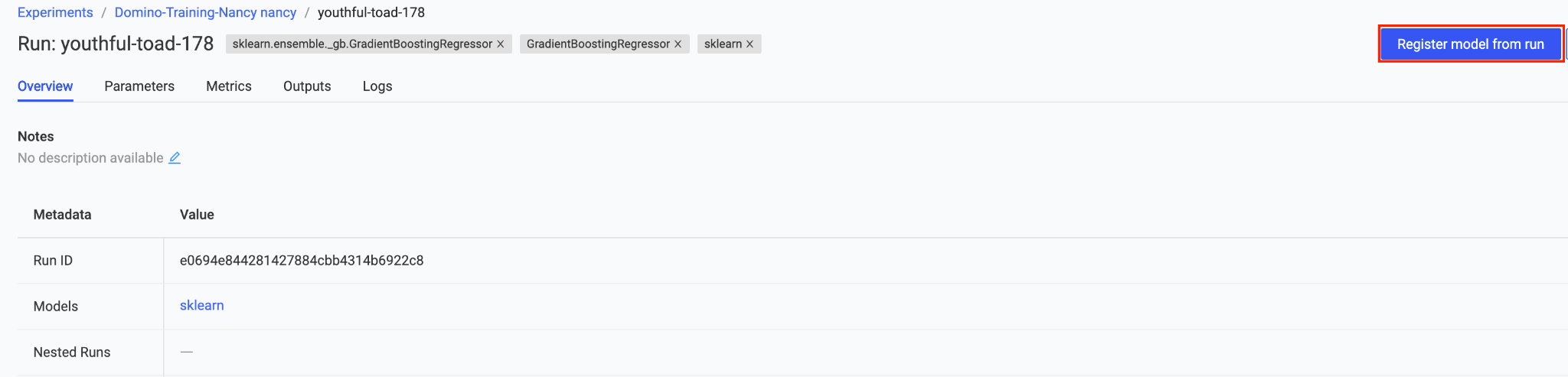


Now, click on **Target (Metrics)** and select **MSE** to add it to our visualization. Then click **Save**.

Our results show that the **sklearn model** is the best candidate for deployment. Click on the **sklearn** **GradientBoostingRegressor** Run to see the details.

Once on the overview page, you can see all the details, and tabs at the top provide even more information. Since we have decided to use this model, we will register it in our central model registry.

Click **Register model from run** in the top right corner.



Give your model a name, such as **FirstName-LastName**. In **Logged MLflow Model,** select ‘**model**’, give it a **short description,** and click **Register Model**.

End Section 2.