Domino for Practitioners

Rev 3 Training Day
May 4, 2022
Dr. Melanie Veale, Field Data Scientist

AGENDA

Project Setup	Settings and Collaboration
	Git in Domino
	Data in Domino
Running Code	Workspaces and Jobs
	Environments
	On-demand Compute Clusters
Publishing	Model APIs
	Web Applications
	Jobs and Scheduled Reports
	Launchers
Operationalizing	Use Domino programmatically
	Integrated Model Monitoring

DOMINO

End-User Course Objectives

Upon Completion of this Course, Learners will be able to:

Describe how to set up a Domino Project.

Explain the differences between running code in Workspaces and Jobs.

Articulate the benefits and use cases for On-Demand Distributed Computing.

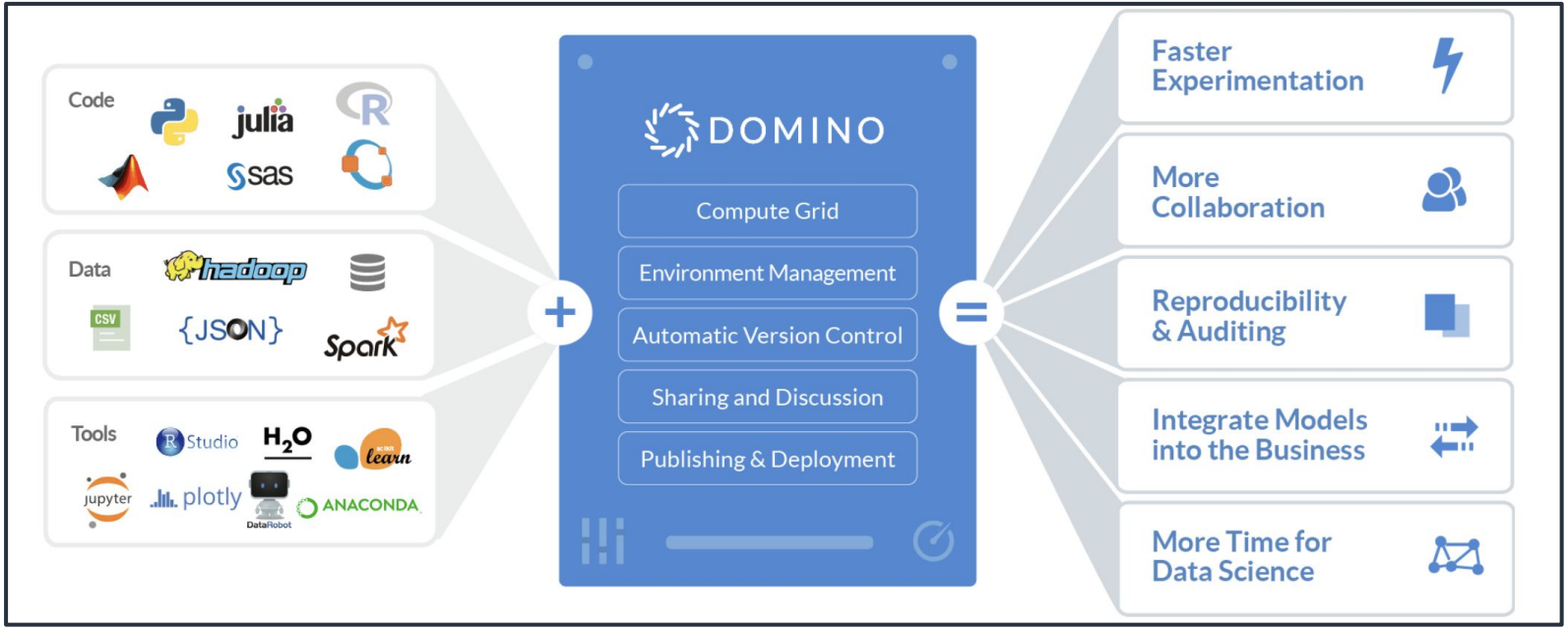
Describe the workflows of publishing and monitoring Model APIs.

Discuss Domino Web Apps, Scheduled Reports and Launchers.

Identify the different ways to interact with Domino programmatically.

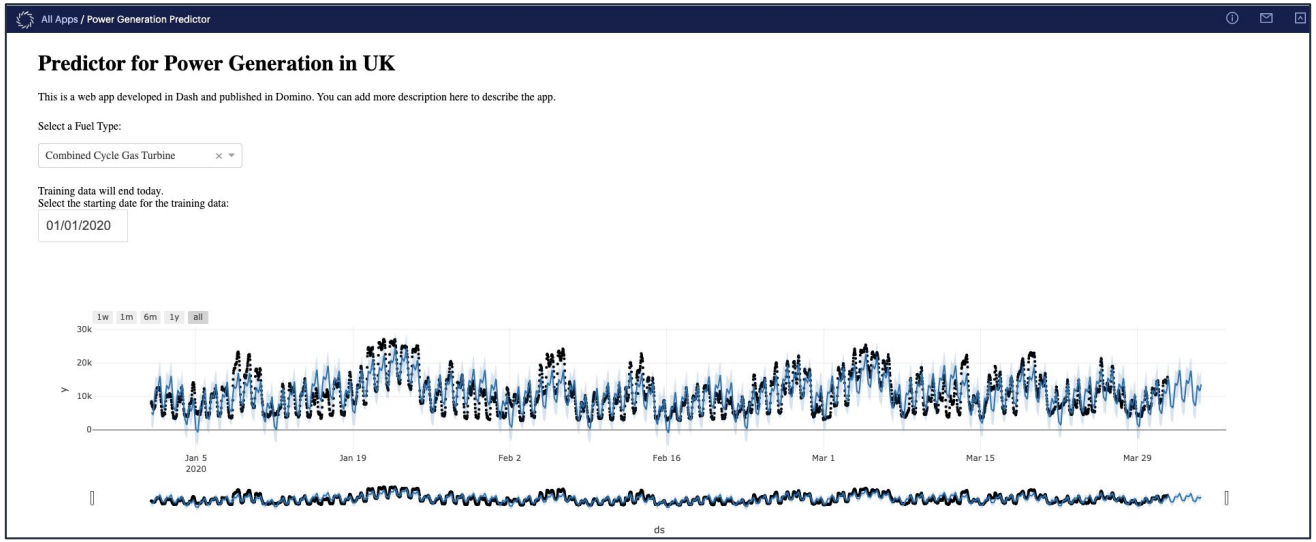
What is Domino?

Domino makes it easy to access scalable compute, update and share environments, keep track of your experiments and file changes, and easily deploy models



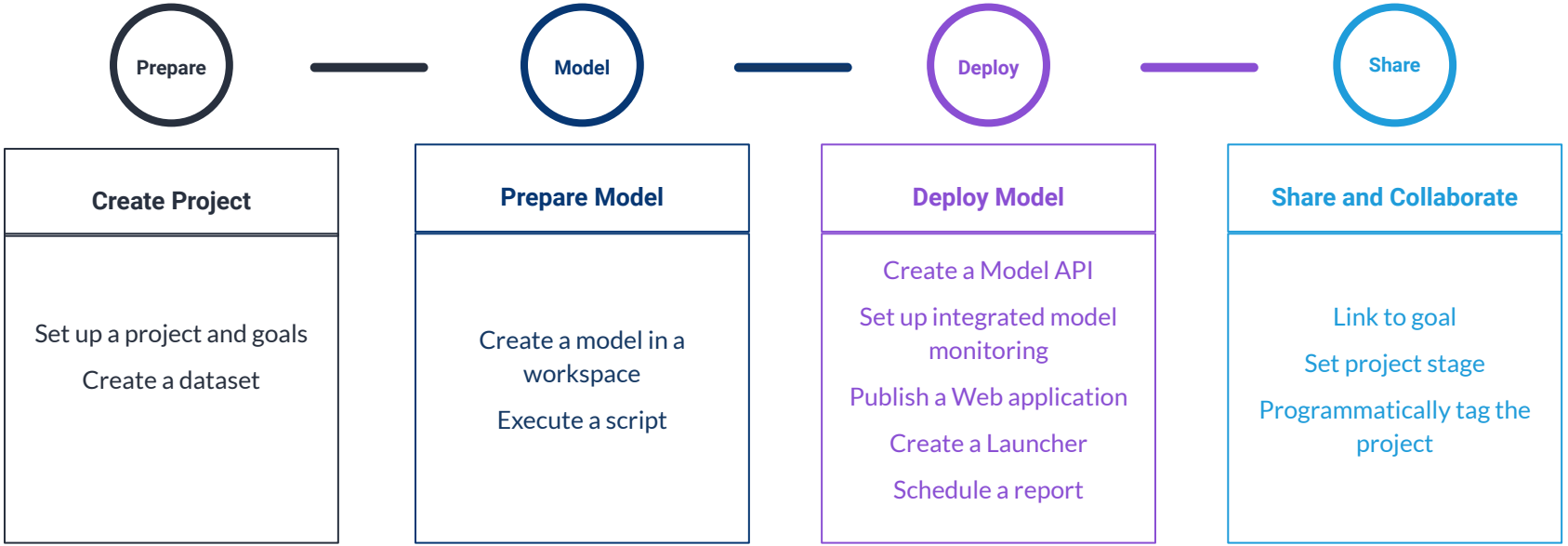
Data Science Project (Python)

- The activities will use real-world data from the Balancing Mechanism Reporting Service to predict future electricity generation
- This tutorial is also covered in depth in our documentation



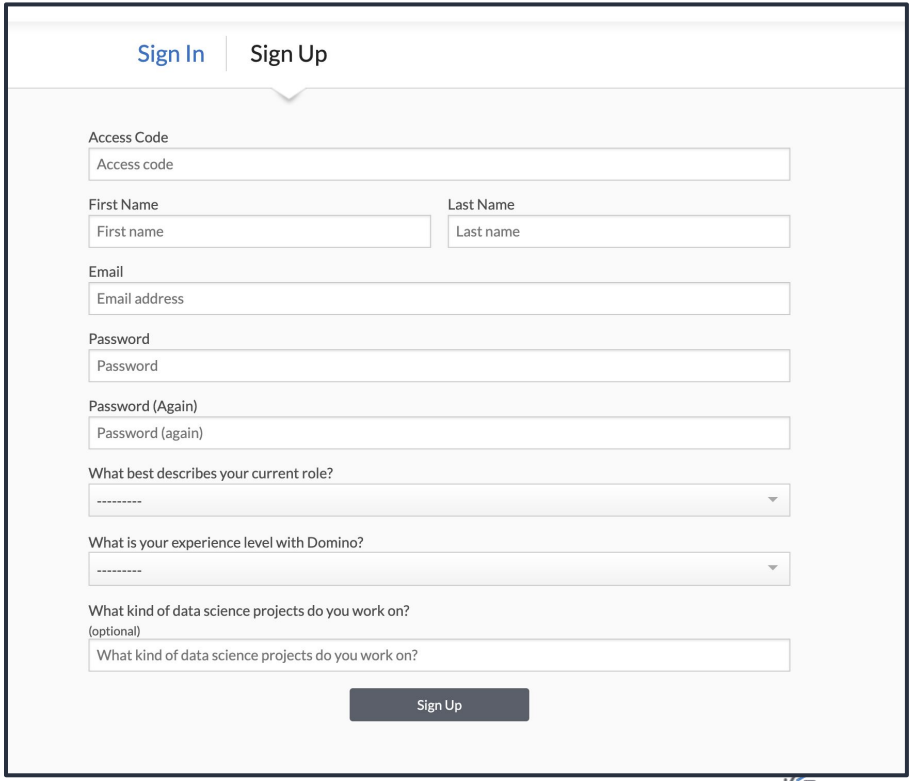
DOMINO

Training Objectives (Including Online Courses)



Self-Service Online Training

- Domino 101, 201, and Publishing are available as online self-service trainings
- Go to learn.dominodatalab.com and select ‘Sign Up’
- Create an account using this access code: 2sypw-iehkb44g



Course Goals

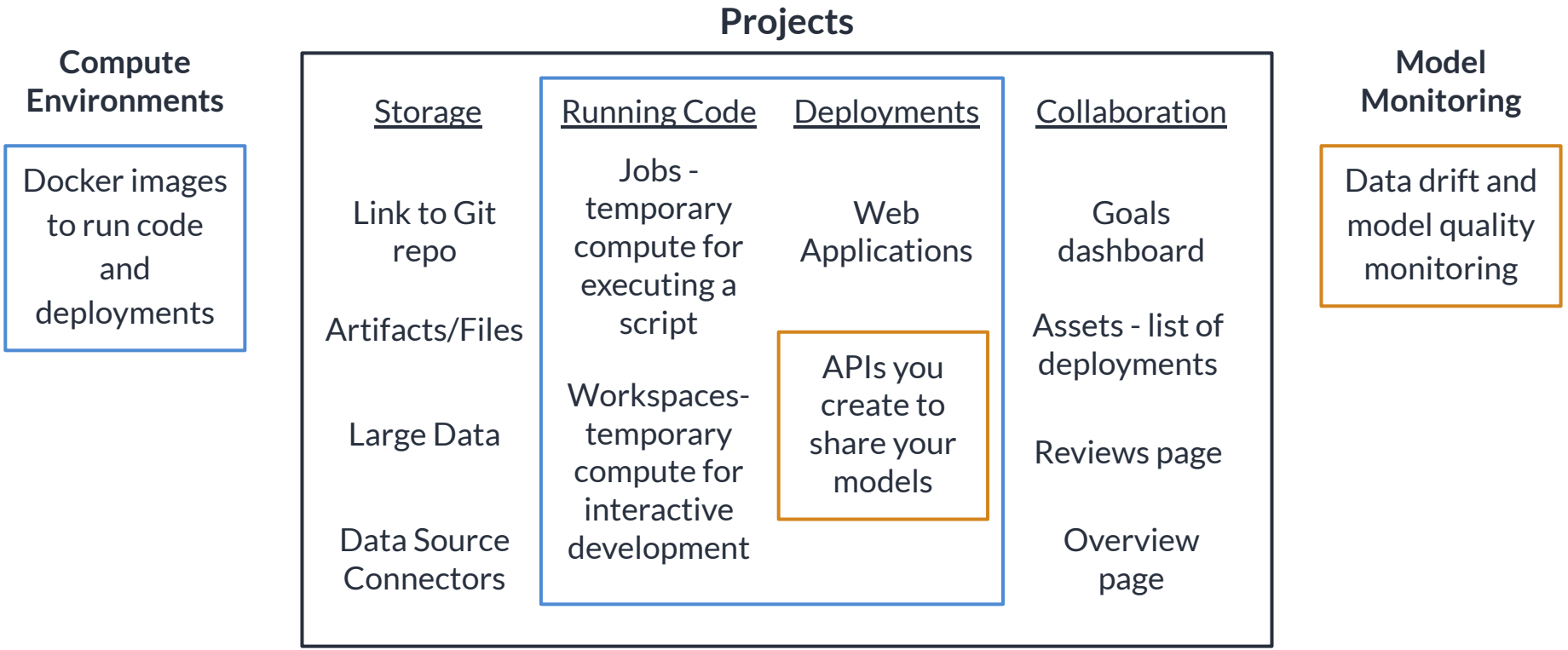
- Get comfortable clicking around in Domino
- Know where to find help and information

Project Setup

Projects: The Main Unit of Organization

- Projects hold all of your files and revision history
- You will run code inside of a project
- Project settings are basic options on how you run your code:
    - Compute environments (software your project needs to run)
    - Hardware tiers (compute resources your project runs on)
    - Access control - who can see and collaborate on your projects

Domino Overview



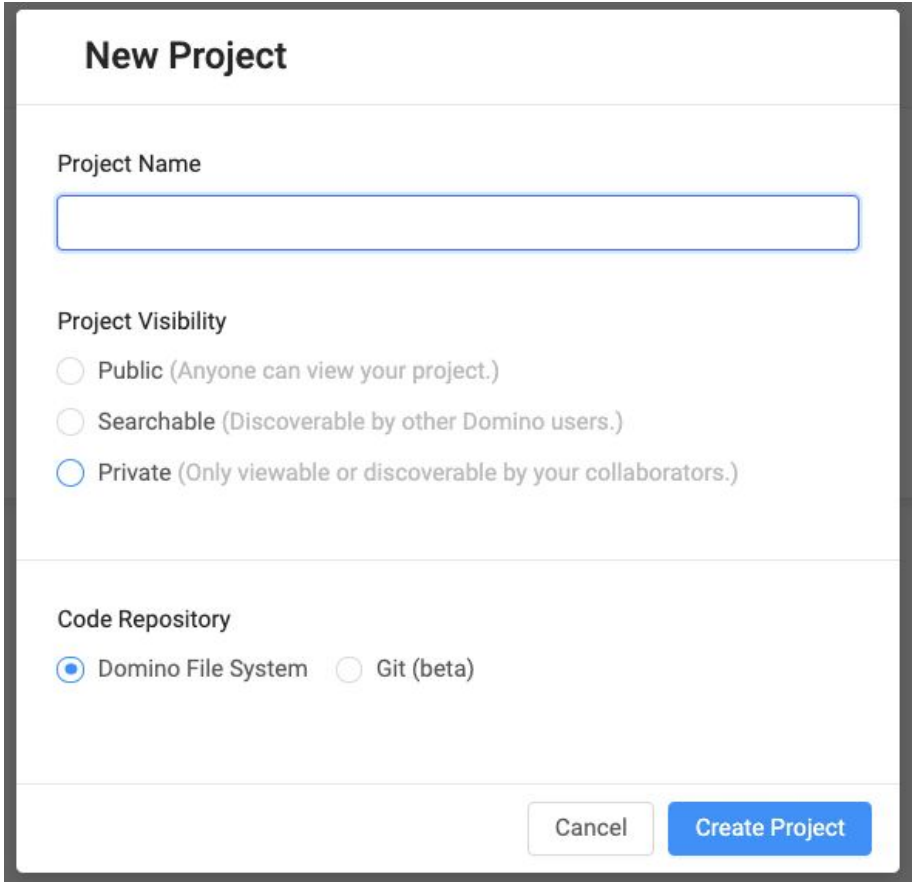
Project Types

Domino File System (Classic)

- Store code files directly in the Domino project

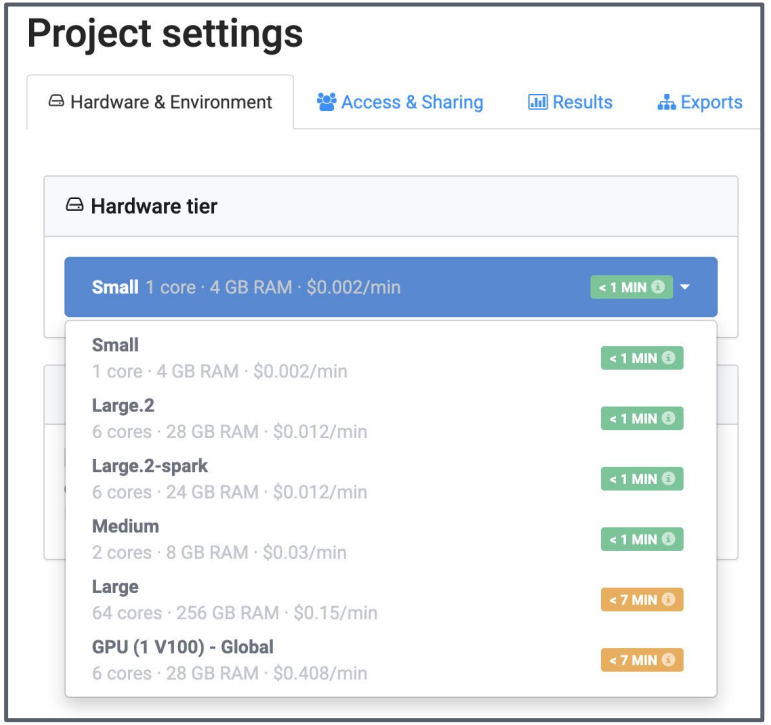
Git-Based

- Provides a full Git experience for your code



Project Settings: Hardware Tiers

- Compute resources for that job or workspace
- Your Domino admin defines these - can be custom for organizations
- Choose a hardware tier that will meet the needs of your workflow

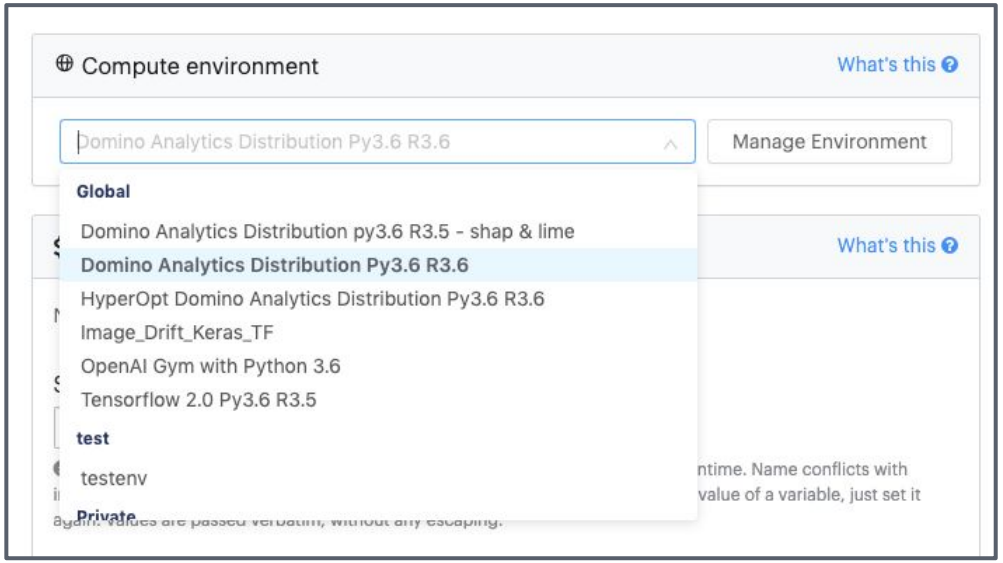


Project Settings: Compute Environments

Specification for the container where your job or workspace will run

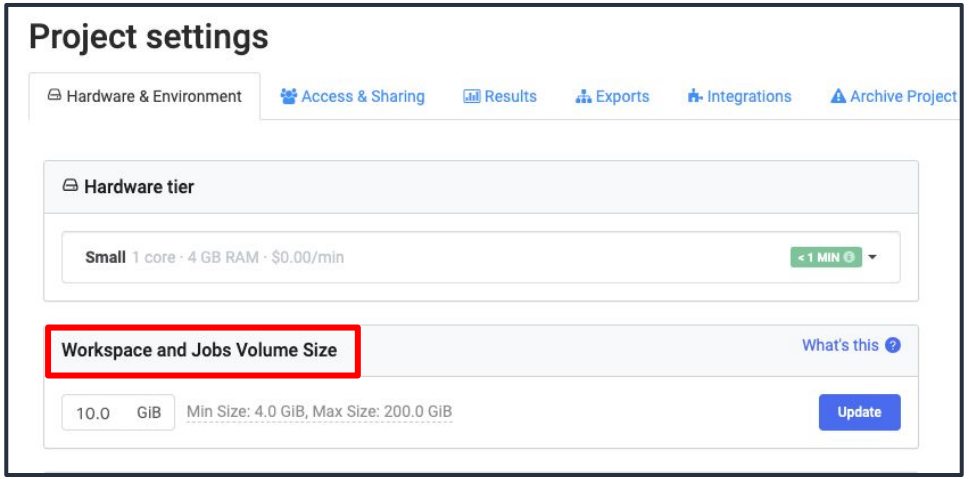
Automatically versioned and easily shared

…More on this later!



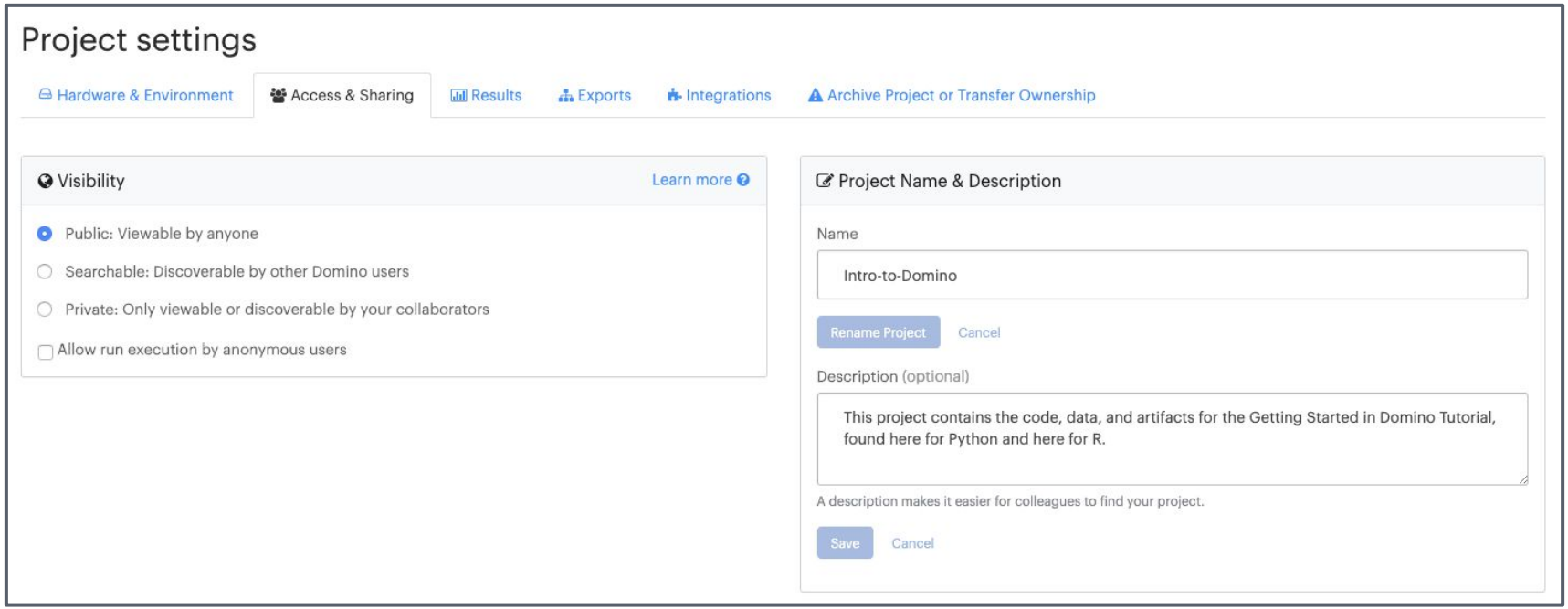
Project Settings: Volume Size

- Controls the amount of storage available to that workspace
- Can be updated in project settings
- Default limits are shown - this is configurable by your Admin



Project Settings: Access and Sharing

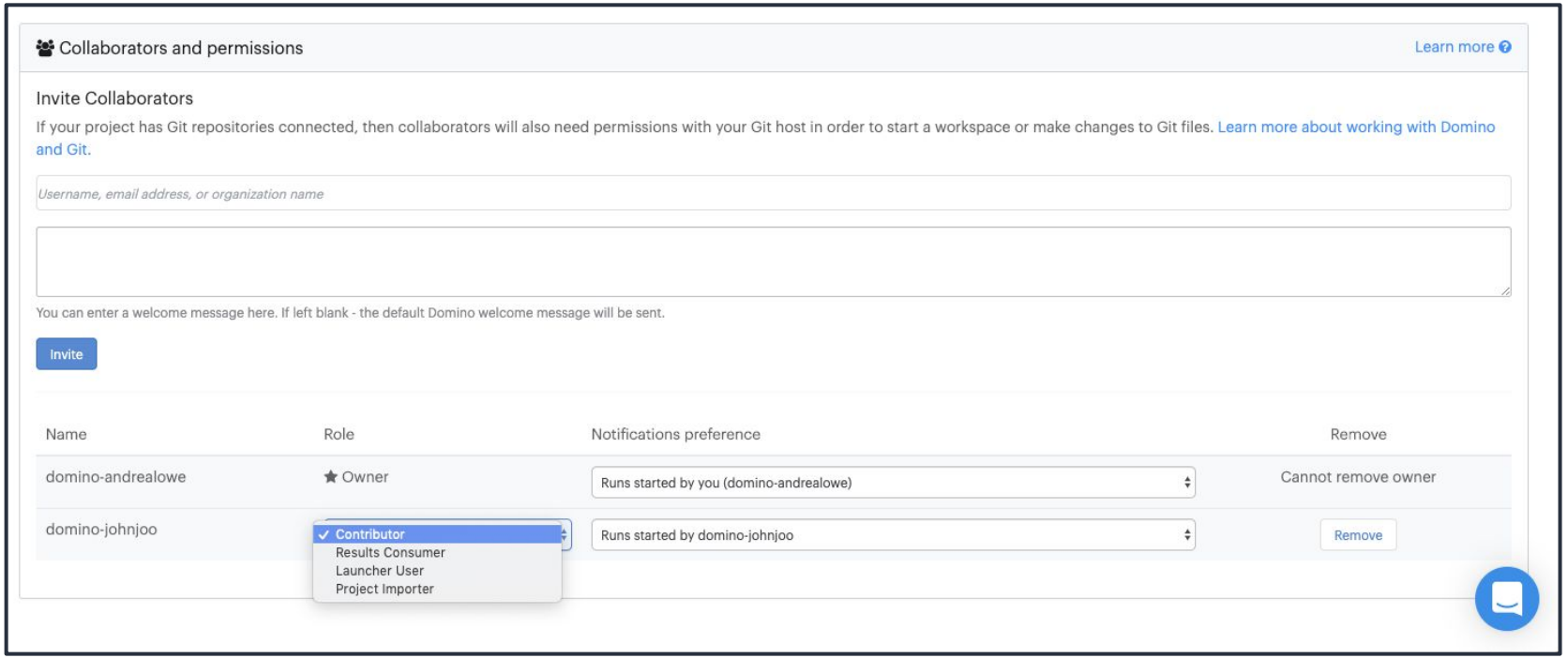
Set the visibility of your project



DOMINO

Project Settings: Access and Sharing

Add colleagues or groups (via organizations) to your project and set permission levels



Project Settings: Permissions

Collaborators on your projects can have different permissions levels

- Contributor - read/write files, start runs
- Results Consumer - read files and access published apps
- Launcher User - run Launchers and access those results
- Project Importer - import the project
- Owner - archive, change collaborator types, alter settings

Organizations in Domino

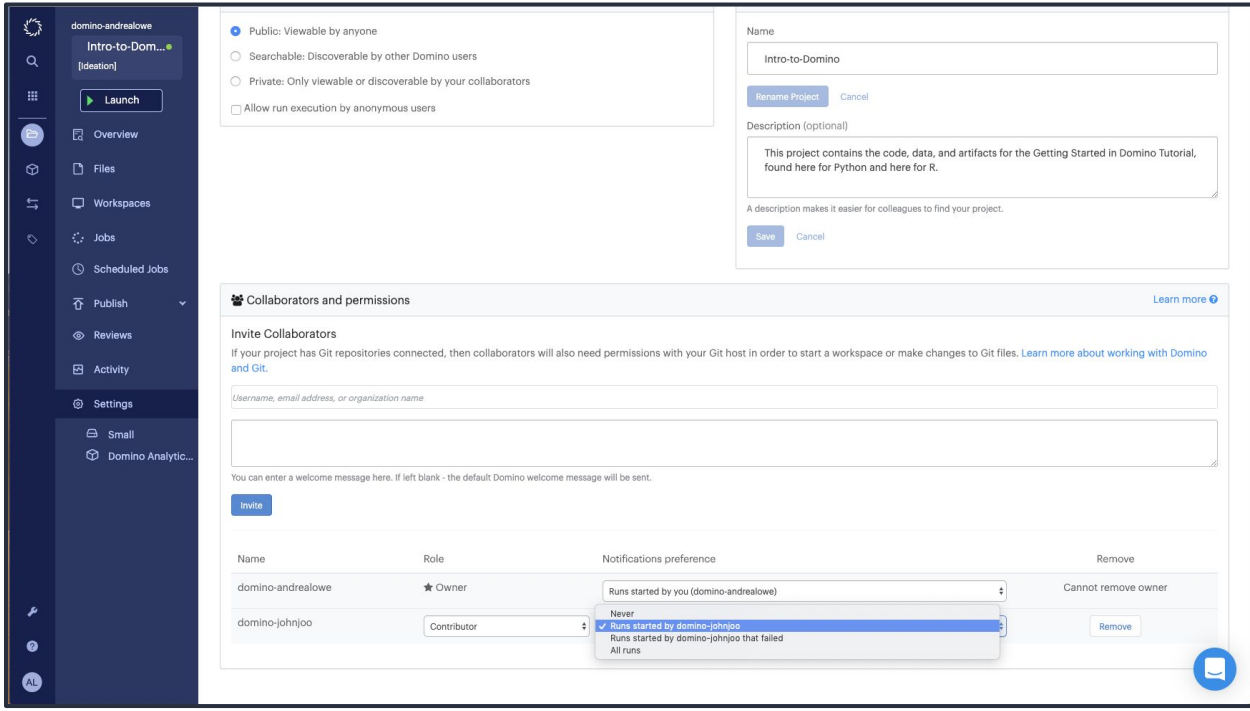
Domino lets you group users into Organizations

Use organizations to:

- Permission projects to many users at once and easily add/remove collaborators from multiple projects
- Assign *owner* level permissions to a group of users
- Share compute environments
- Restrict hardware tiers to specific groups (controlled by Admins)

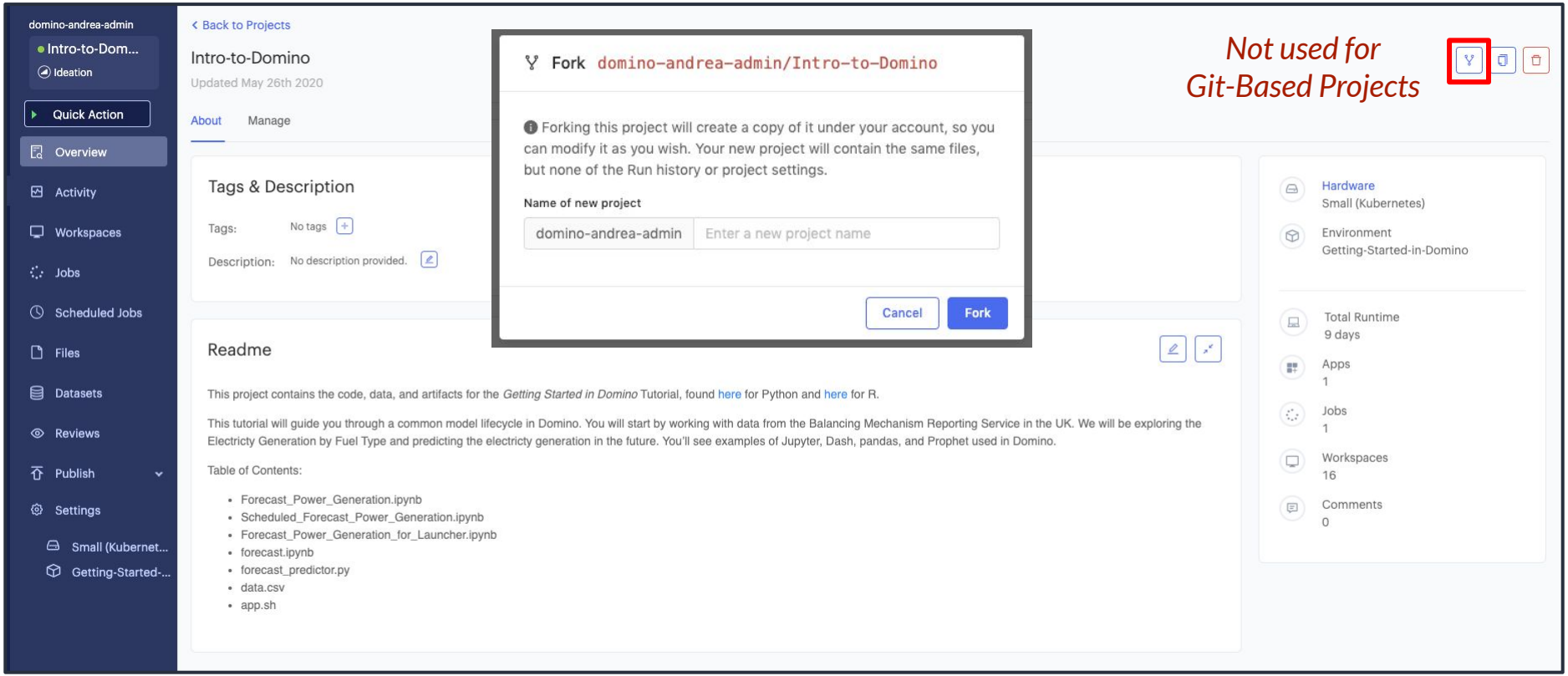
Project Settings: Emails

Configure email settings about Job activity by you and your collaborators



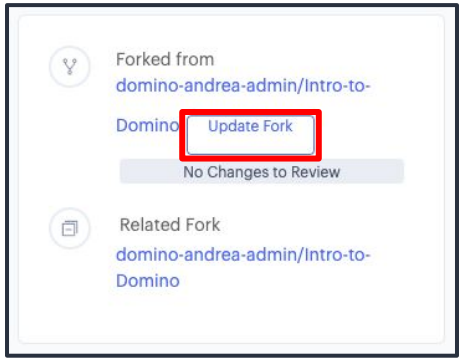
DOMINO

Projects: Fork/Merge



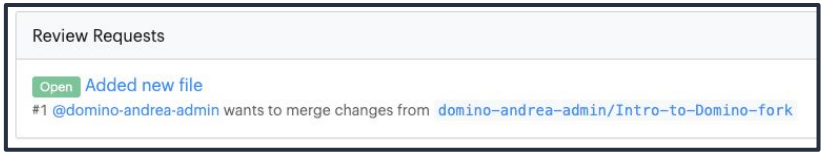
Projects: Fork/Merge

Bring changes from original project into your forked project:



Request to merge changes from forked project into original:

Review changes from forked versions of your project:

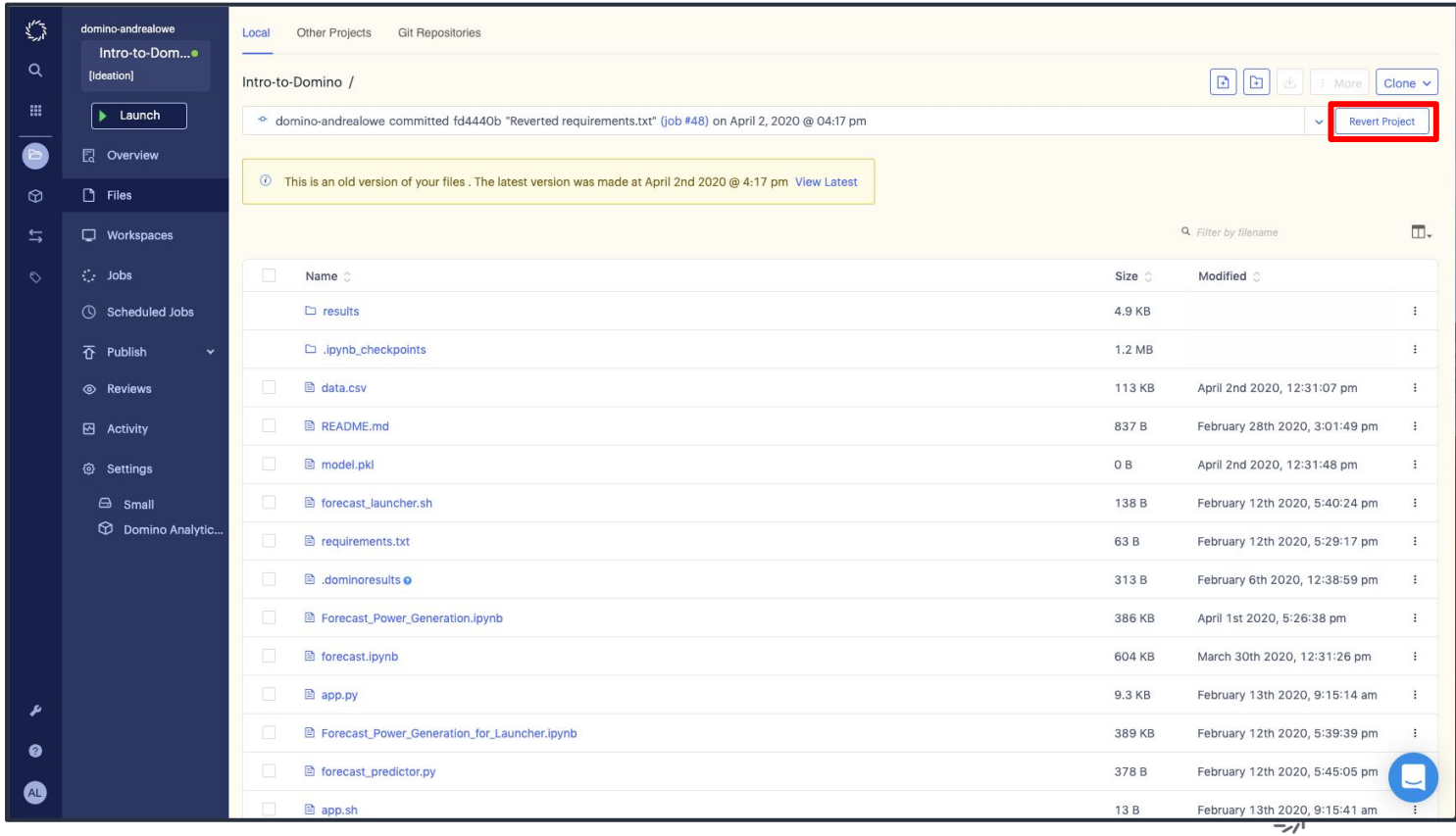


DOMINO

Projects: Versioning and Reproducibility

You can revert your project or any individual file to any previous state

In Git-Based Projects this page is labeled *Artifacts*

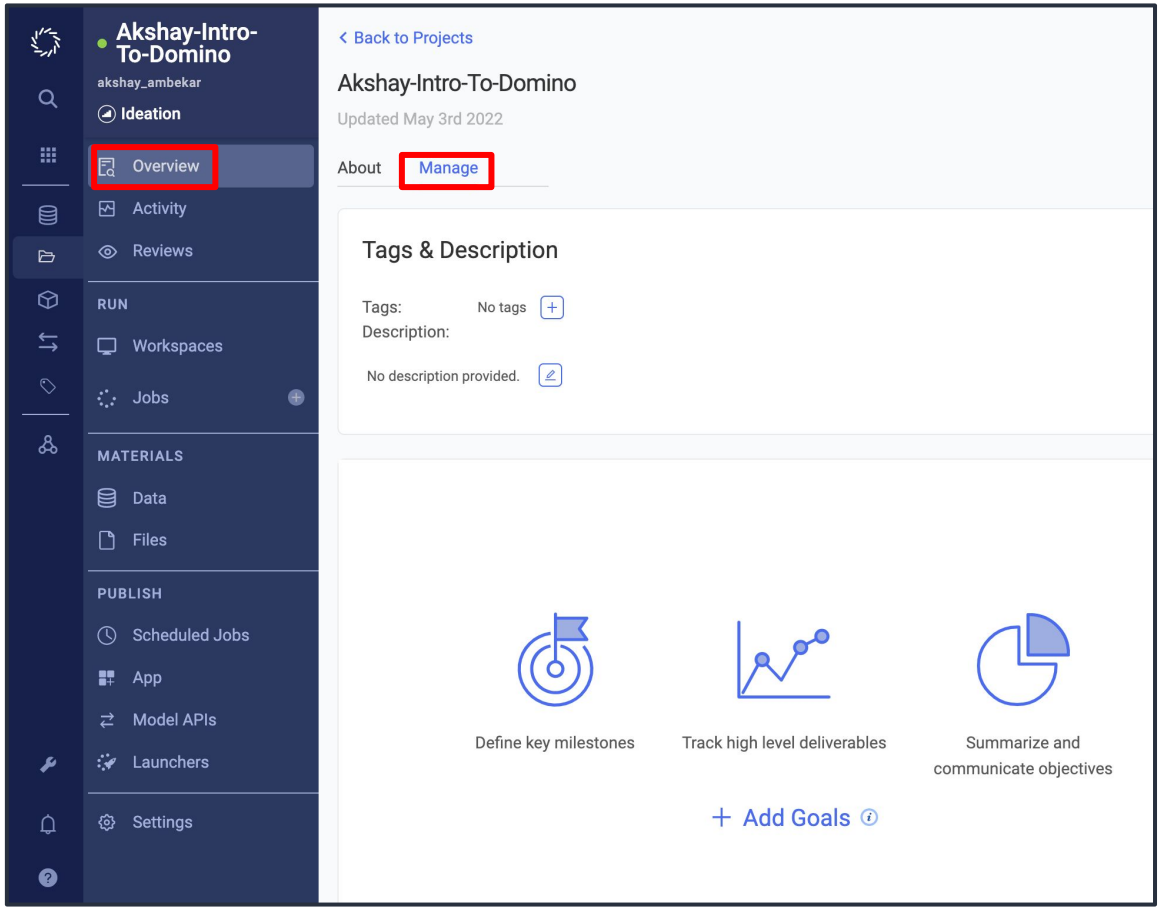


Projects: Goals

Define goals and link code, files, or artifacts to them

Add comments and discussion around each goal

Track stages of goals separately from the project



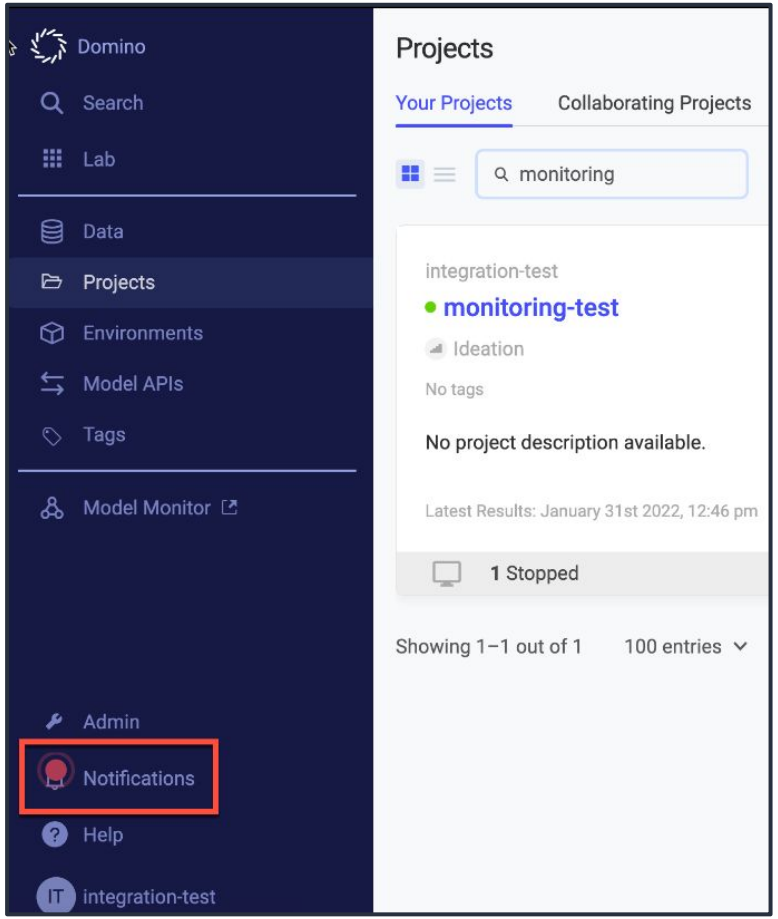
Notifications

Your Admins can send you notifications in Domino

These are different from the Project run notifications

The indicator color matches criticality:

- Default
- Critical

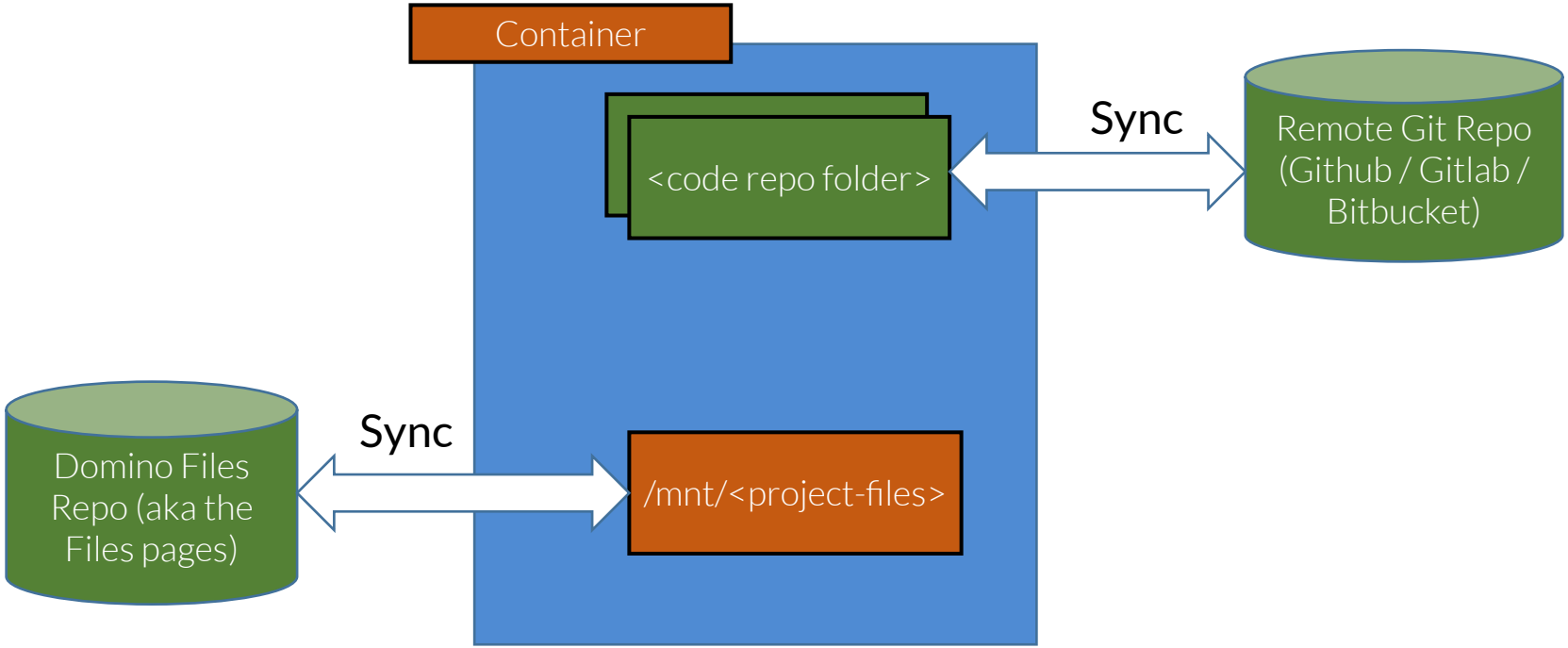


Activity: Fork a Project

1. Sign into temporary training Domino instance and open the project:
   https://rev.workshop.domino.tech/u/melanie_veale/Intro-To-Domino/overview
2. Fork the project from the Overview page and name it '<your name>-Domino-Training'
3. Navigate to settings page:
   a. Make sure project is set to private
   b. Note Environment and Hardware Tier
4. Add a goal to the project called 'Train and serialize model'

Git in Domino

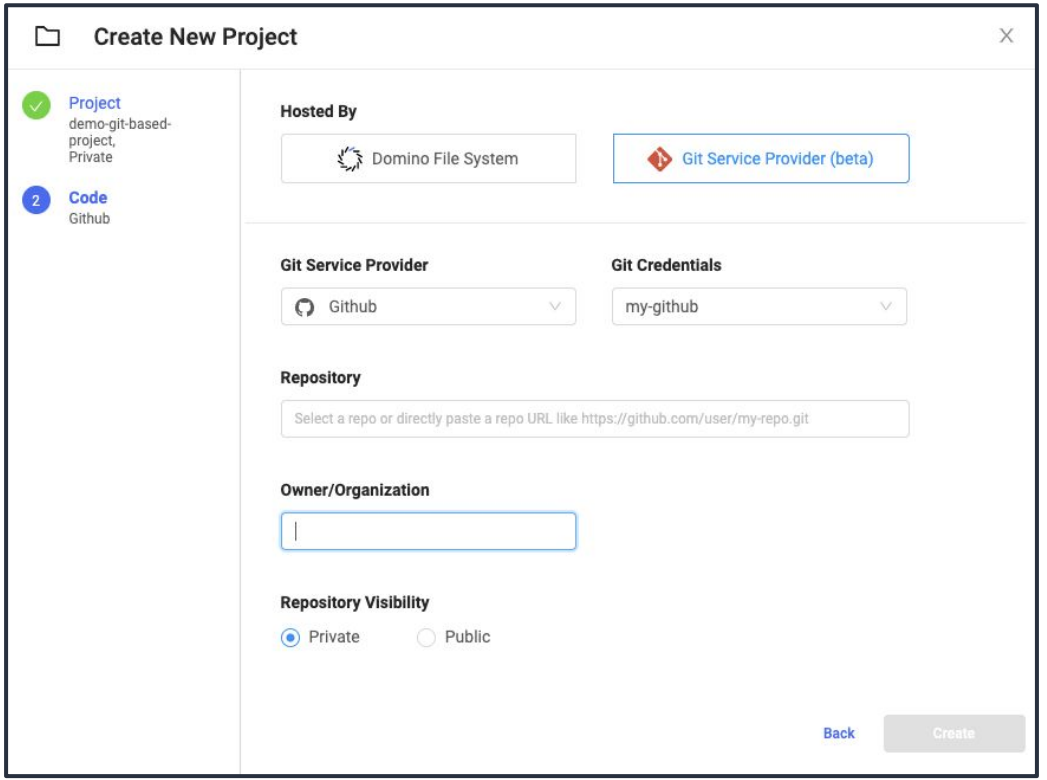
Integration with Git: Separation of Version Control Systems



Git Based Projects

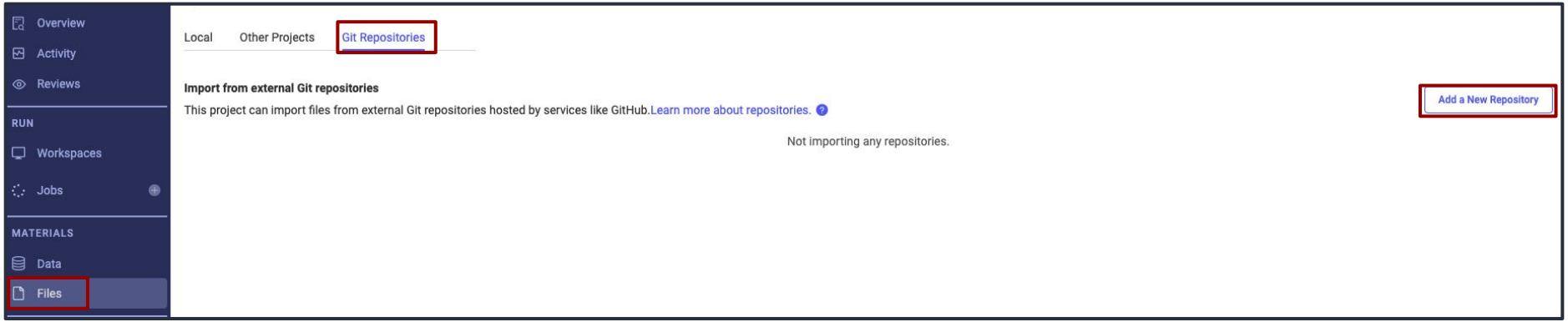
Users can build a project on an existing git repo

A new GitHub or GitLab repository can also be created directly



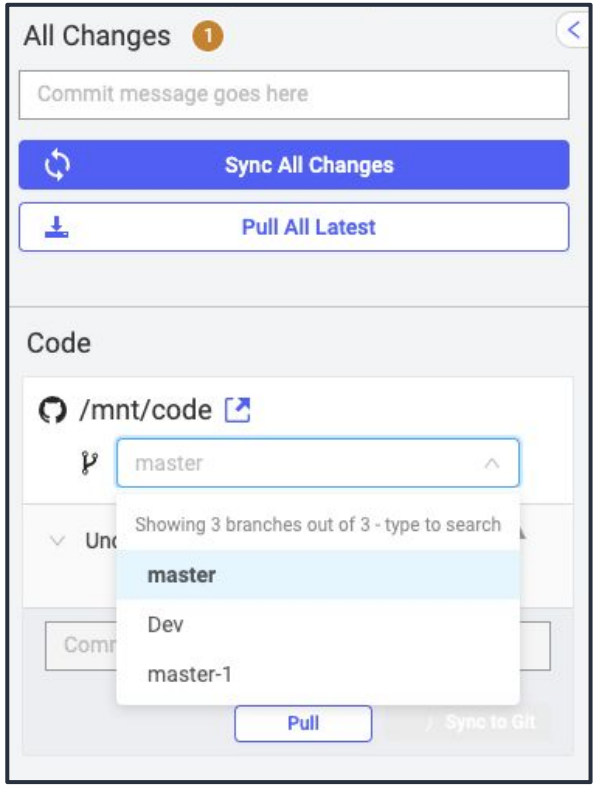
Import Git Repos

Any number of git repos can be imported into a standard Domino project or a git-based project



Git Integration

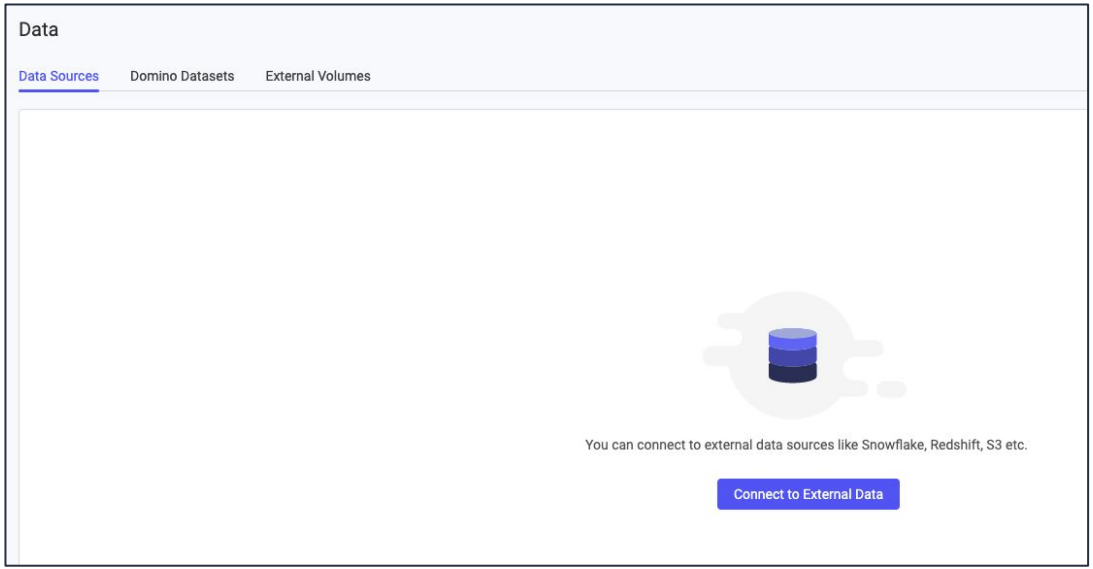
- Make sure to add authentication before setting up git integration
- You can change branches and resolve conflicts directly in Domino
- Branch updates and status will be reflected in your Workspaces
- See details in Domino 101

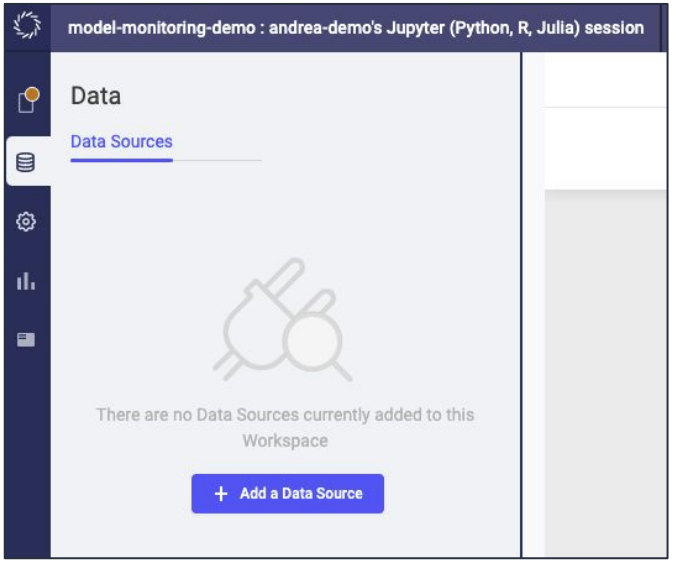


Data in Domino

Domino Data Connectors

Create a data source in projects or workspaces

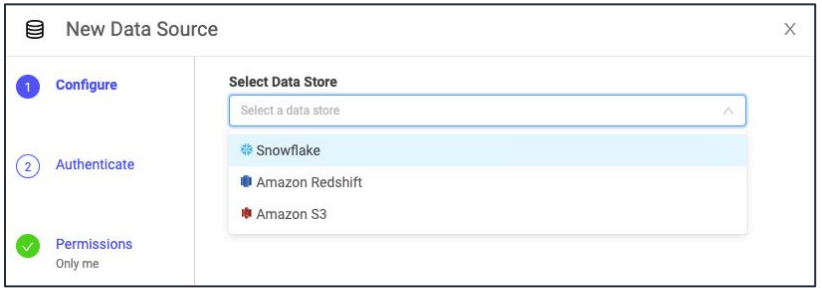




Create a Data Source

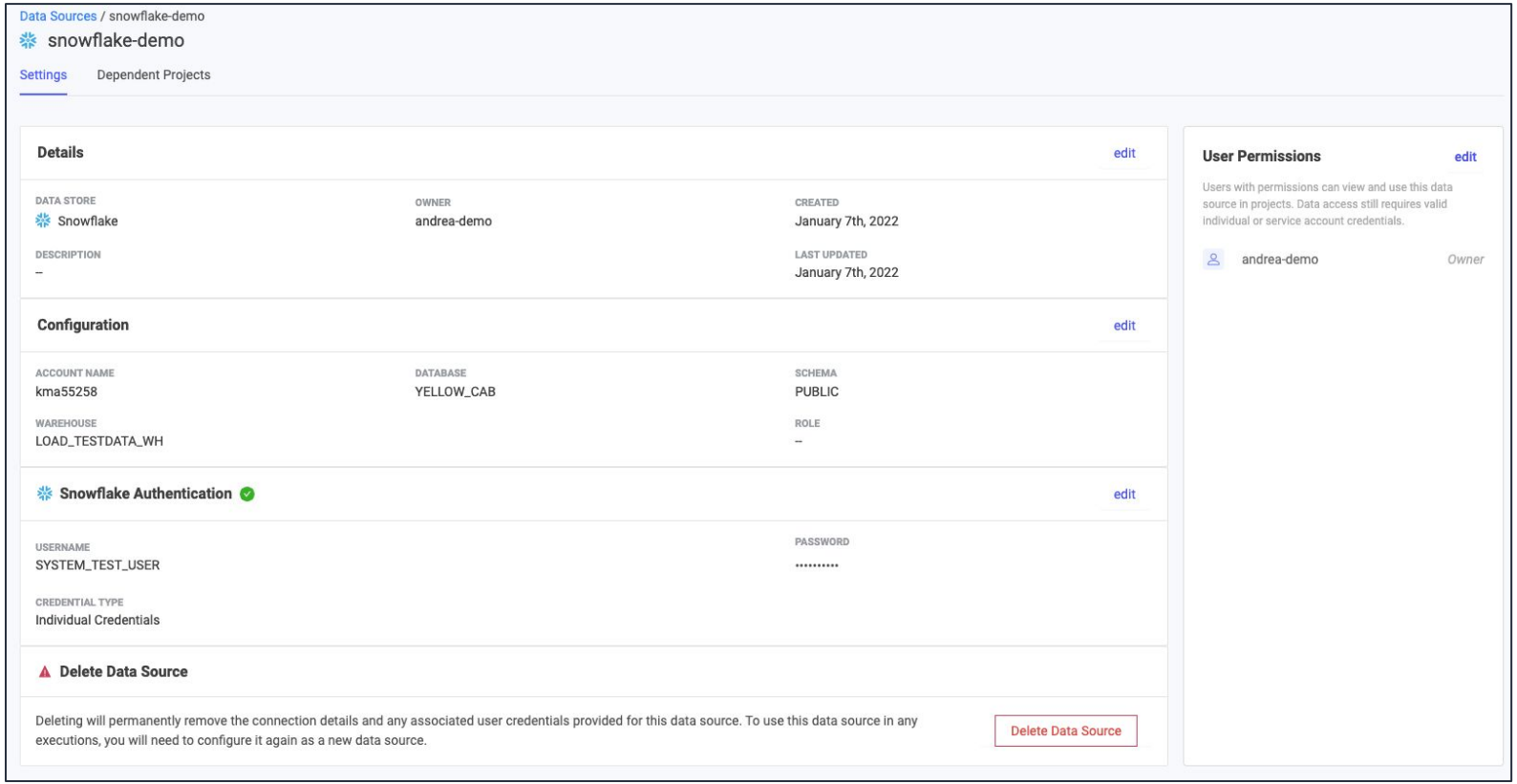
Choose from:

- Snowflake
- AWS S3 and Redshift
- MySQL
- Microsoft SQL Server (MSSQL)
- PostgreSQL
- Oracle Database
- Azure Data Lake Storage
- Google Cloud Storage



DOMINO

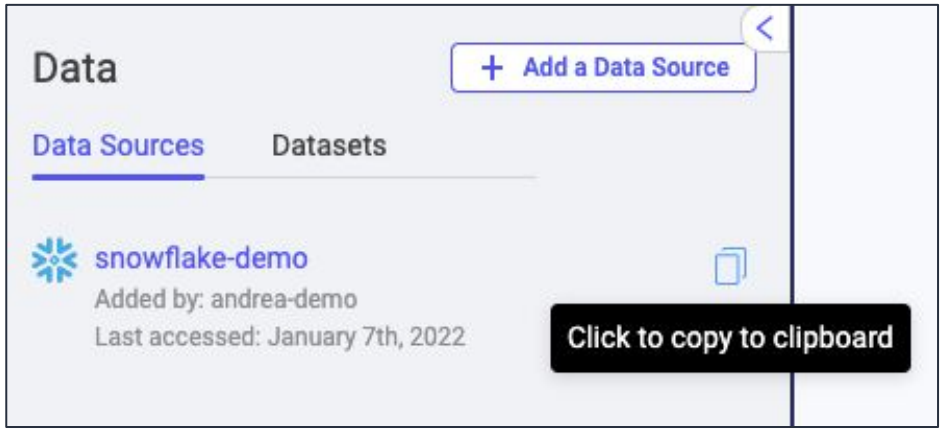
Create a Data Source



Use Data Connectors

Workspaces will provide an option to copy code for connecting to the data source

Use similar code in your Job scripts, see here for more details




```
from domino.data_sources import DataSourceClient

# instantiate a client and fetch the datasource
instance
ds =
DataSourceClient().get_datasource("snowflake-demo")

# res is a simple wrapper of the query result
res = ds.query("select * from {{TABLE NAME}} limit
100")

# to_pandas() loads the result into a pandas
dataframe
df = res.to_pandas()
```

Data Source Permissions

Admins can add service accounts

Admins can make data sources accessible to all, while users must add individuals or organizations

Permissions are checked as follows:

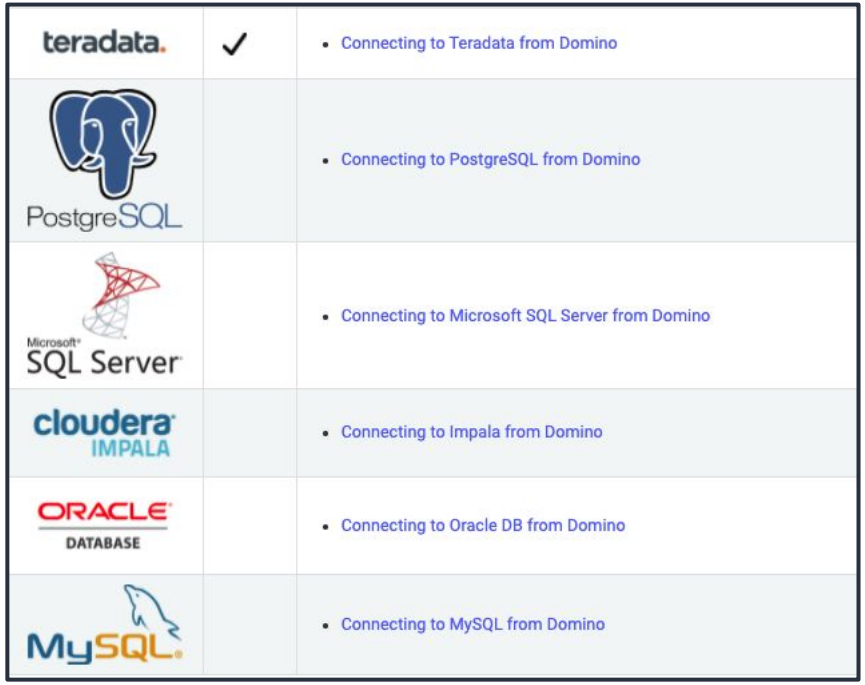
- **Workspaces and Jobs** - user who started the execution
- **Launchers** - user who started the launcher regardless of who created the launcher
- **Domino Apps** - user who published the app regardless of who is accessing the app
- **Model API** - no user identity, it is possible to inject an API key into an execution through an environment variable, and then use it explicitly when retrieving a data source

Other Data Connections

To connect to other outside data sources you'll need:

- Network connectivity
- Package or driver
- Credentials

See detailed instructions here, or check out the Domino 101 data section



Domino Environment Variables

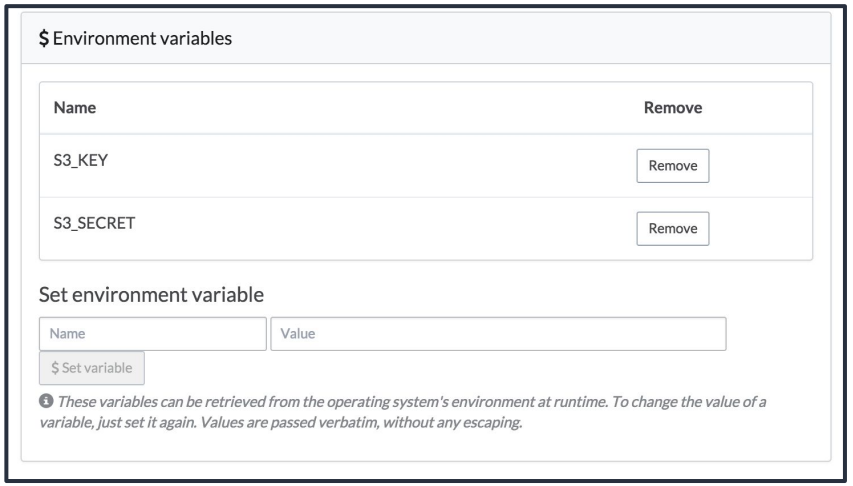
Use environment variables to store credentials securely.

Project level environment variables

- Accessible to anyone with access to project
- Added via Project Settings page

User level environment variables

- Accessible to only you when executing Runs
- Added via Account Settings page



Example use in R:

```
Sys.getenv("S3_KEY")
```

Example use in Python:

```
import os
os.environ["S3_KEY"]
```

DOMINO

Datasets vs Project Files

Project Files:

- Files are synced whenever a code session is launched or stopped
- Large or many files can slow down these launch/sync times
- Limits based on project volume size
- Can only import data by importing entire project
- Automatically versioned

Datasets:

- Stored in EFS
- Mounted to the machine for use during executions, no copying
- No copying means no additional wait for launching and syncing
- Large file size and large numbers of files allowed
- Manually versioned

Datasets Overview

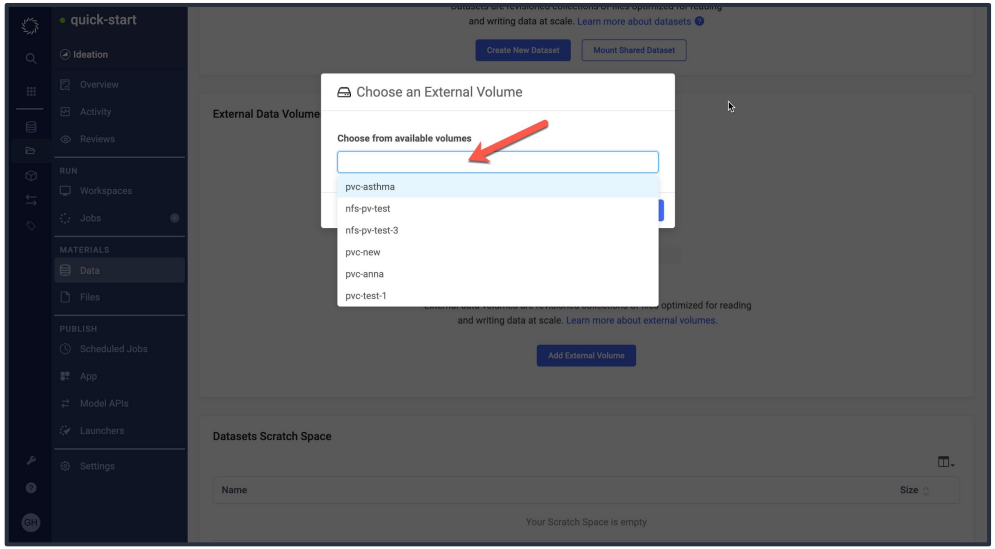
Each project will start with a default dataset

Local Datasets are always read/write - contents can be modified from the browser, Workpaces, Jobs, Apps, and Launchers

Shared Datasets are read only

External Data Volumes Overview

- Domino projects can access data stored in external data volumes
    - Currently supports generic volumes as well as NFS, EFS, and SMB
- Setup must be done by an Admin
- Access is controlled separately from projects - can be limited to specific users or organizations



Running Code in Domino
Workspaces

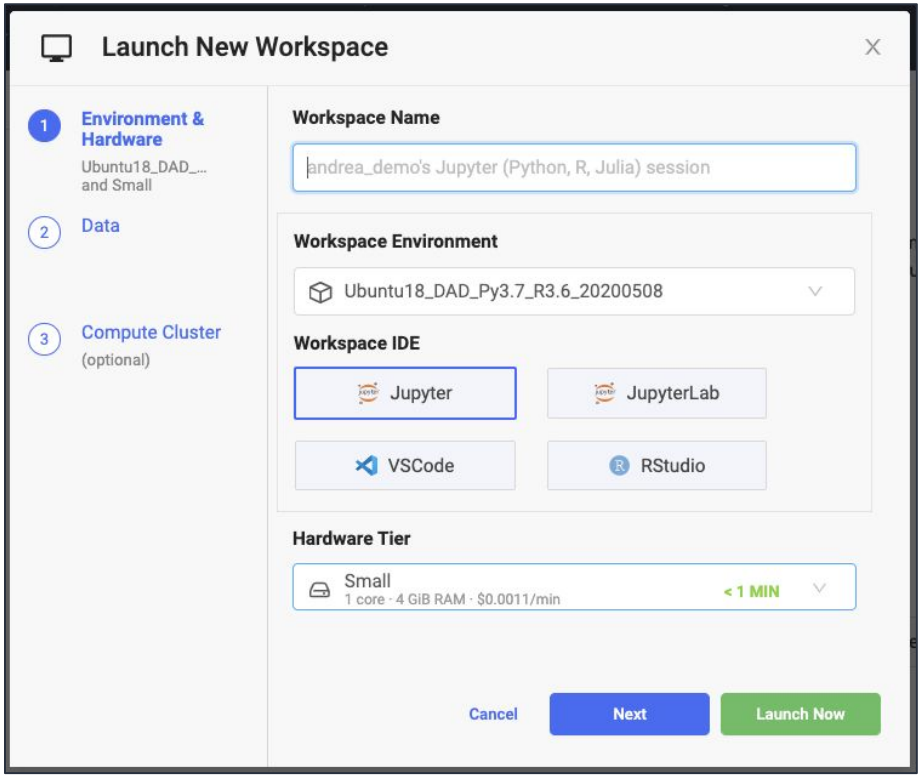
Run Code in Domino

	Workspaces	**Jobs**
For example	RStudio Jupyter Notebook SAS Studio	my_script.py another_script.R wrapper_script.sh
Saving to Domino	Manual	Automatic
Resource shutdown	Manual	Automatic

Configure a Workspace

Configure your workspace at launch:

- Name (optional)
- Hardware Tier
- Environment
- Workspace IDE
- Data
- On-Demand Spark Cluster

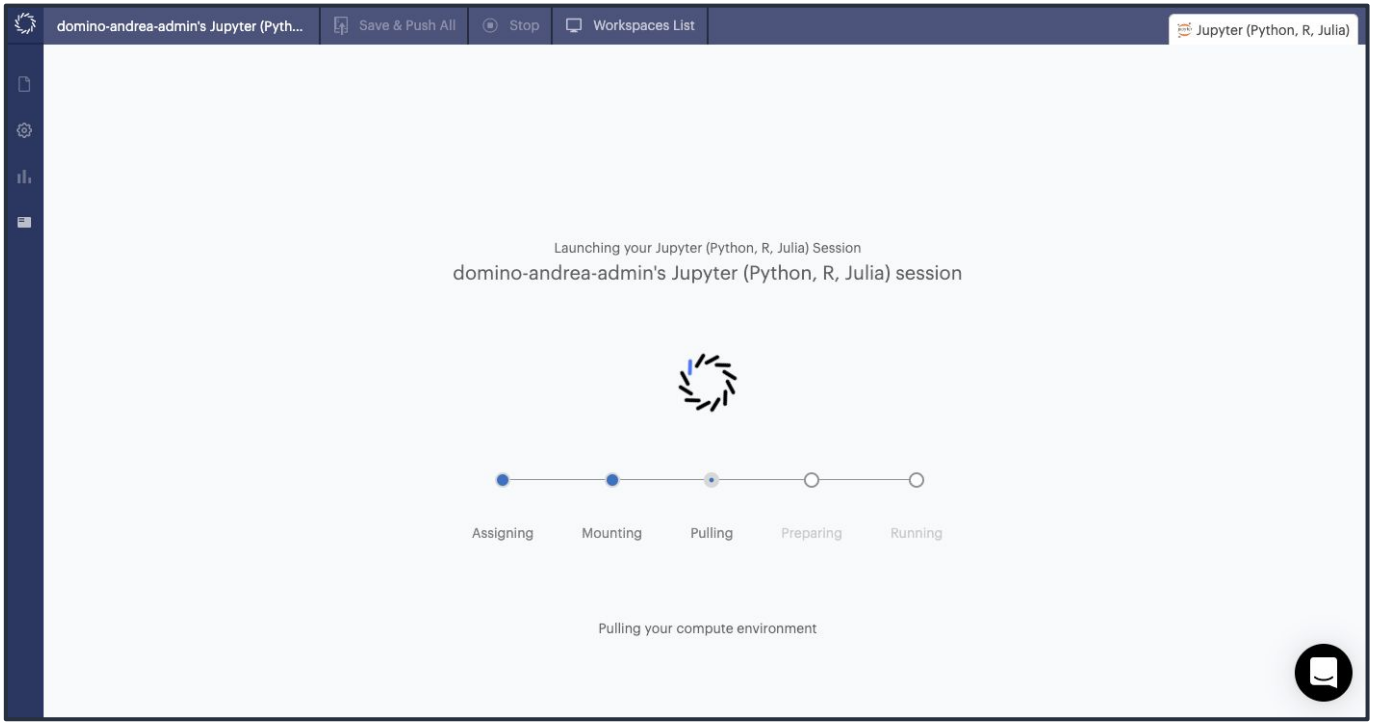


DOMINO

Using a Workspace

While the workspace is starting up, you can see:

- Status of your starting execution
- Logs for your starting execution



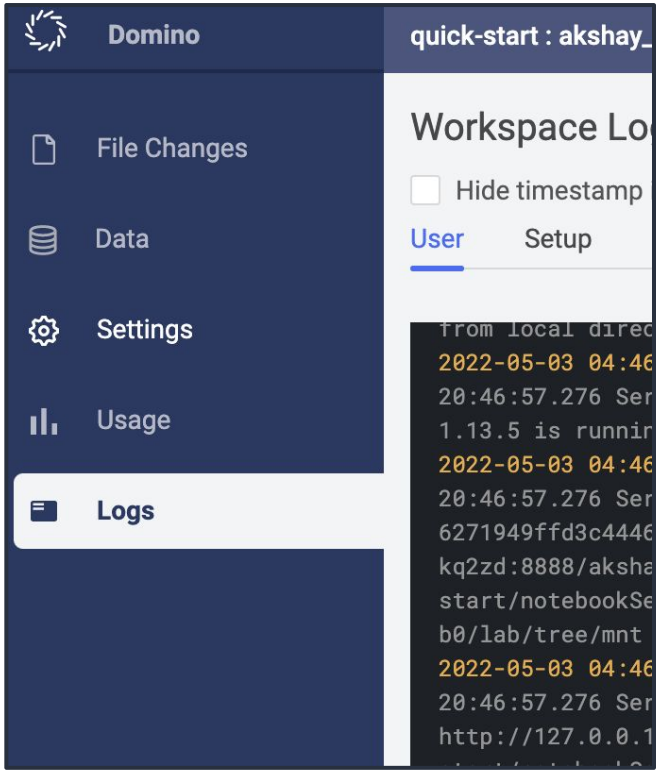
Workspace Launch Process

Stage	What's Happening	Wait Time Depends On...
Assigning	Finding resources for your session	Availability of machines and launch time if needed
Mounting	Attaching persistent volume (files) to pod	Internal processes
Pulling	Copying compute environment (Docker image)	Level of customization to compute environment
Preparing	Launch Domino processes and move files into volume	Size/number of files

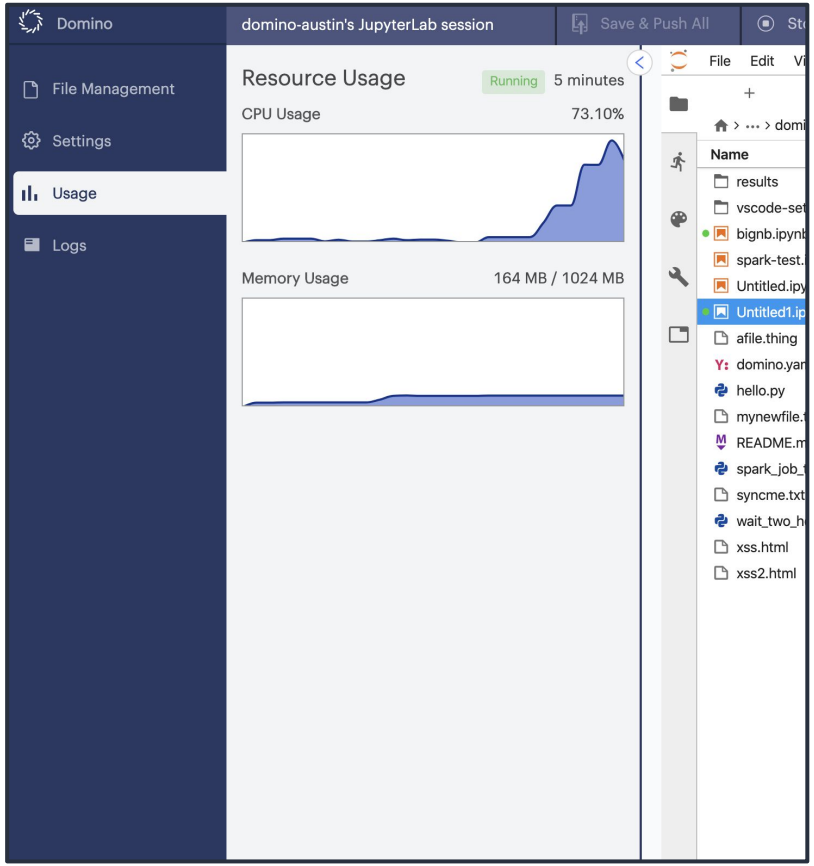
Workspace Navigation Bar

On the left of the screen are the panes for

- File Changes, where changes to your files are tracked and synced
- Data, where your data sources, datasets and snapshots are tracked
- Settings, where you can see and edit your workspace configuration
- Usage, where you can view how your workspace is consuming resources
- Logs, where you can see the setup and user logs your workspace is generating



Workspace Navigation Bar: Resource Usage



In the Usage pane, you'll find graphs of your workspace CPU Usage and Memory Usage over time.

DOMINO

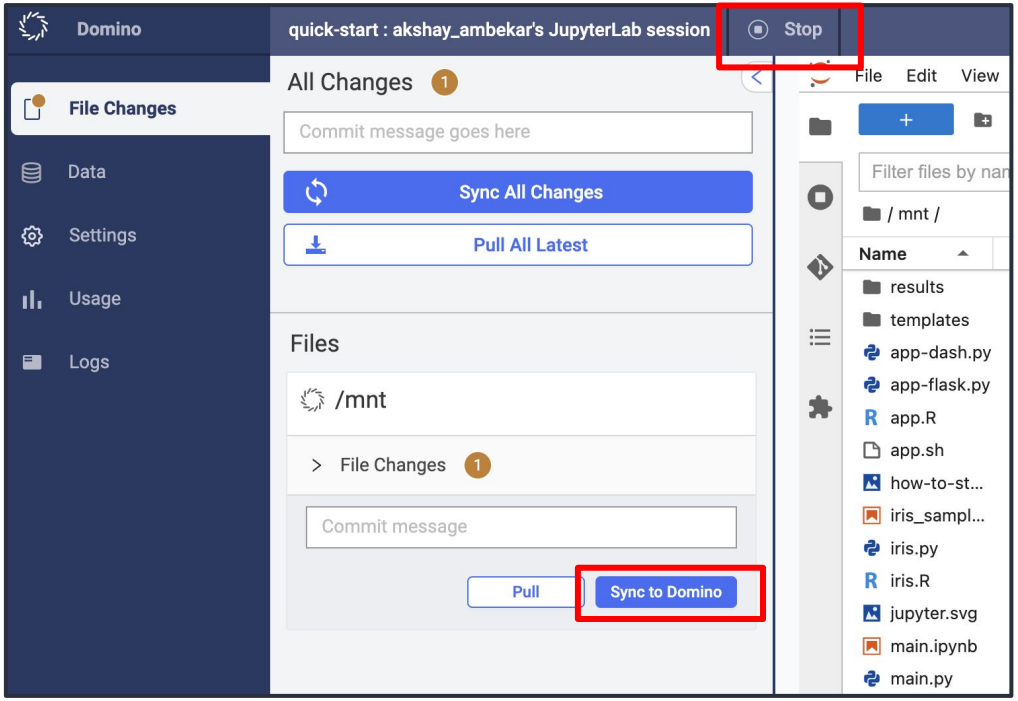
Workspace Sync and Stop

Sync work in left navigation bar

Important - *Stop* session when finished to save on costs

Closing the browser tab does *not* stop a workspace

You can Stop workspaces without committing files



What Restarts?

Persists between Sessions:

- Everything in the *Files* page (Classic Project) or *Code* and *Artifacts* folders (Git-Based Project)
- Workspace settings (Hardware tier, spark settings, volume size, etc)

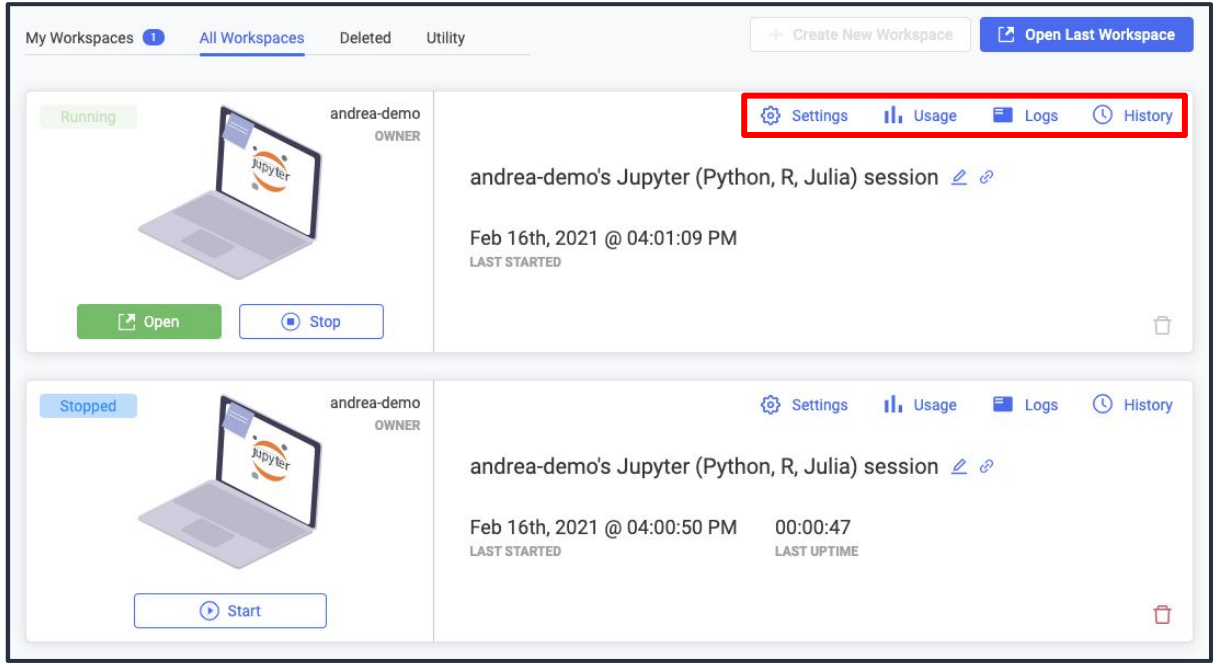
Does not persist between Sessions:

- Any objects in memory
- Files outside of the */mnt* directory (*Files* page) including pip installed packages
- Compute Environment - it will load the latest version on restart

Workspace Dashboard

The workspaces dashboard will show all stopped and running workspaces

See the state of the environment, files, and hardware for every workspace



Workspace Limits

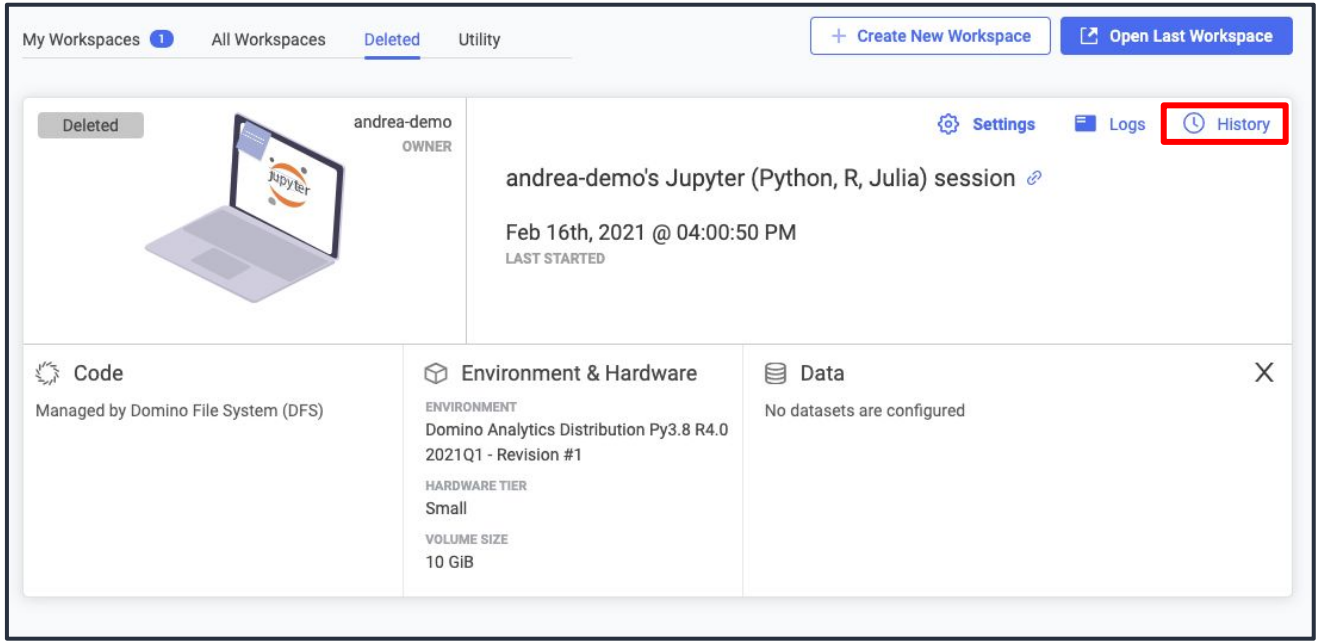
- Configurable by your Admin
- Defaults:
    - Four workspaces per *user* per *project*
    - Sixteen workspaces *total* per user

Deleting a Workspace

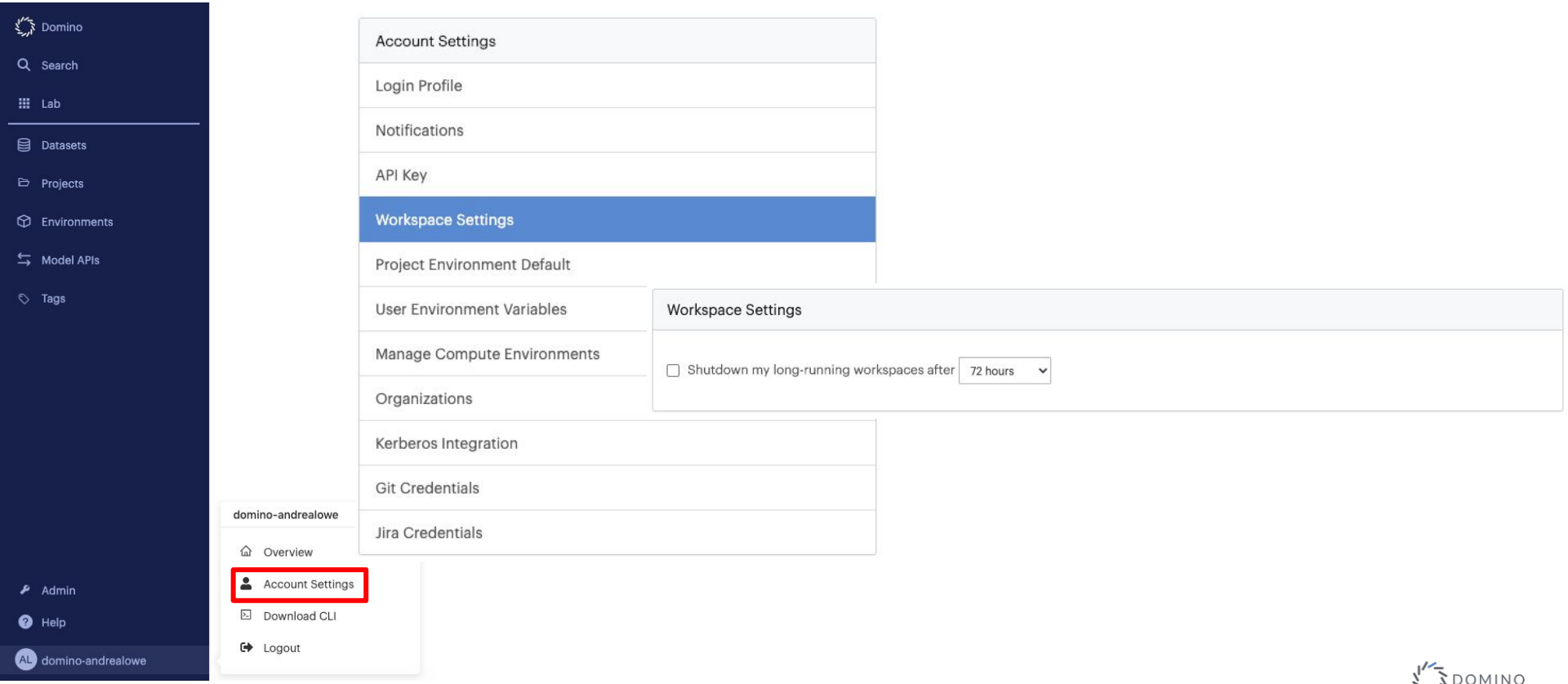
Deleting a workspace stops it permanently

Important: Sync work before deleting!

Select *History* for more settings and logs from previous session



Set Automatic Time Limits for Workspaces



DOMINO

Activity: Develop a Model

1. Navigate to the Workspaces tab in your training project
2. Select the 'JupyterLab' button (optionally name the workspace)
3. Open the Forecast_Power_Generation.ipynb
4. Run all cells in order
    a. Note how we install a new package directly into the notebook in cell
    b. Observe how the serialized model is exported at the end
5. **Sync** your files, then **Stop** your session
6. Navigate to the Files tab and note the updated files
7. Link your model.pkl file to the goal we set up earlier

Running Code in Domino
Jobs

Domino Jobs Overview

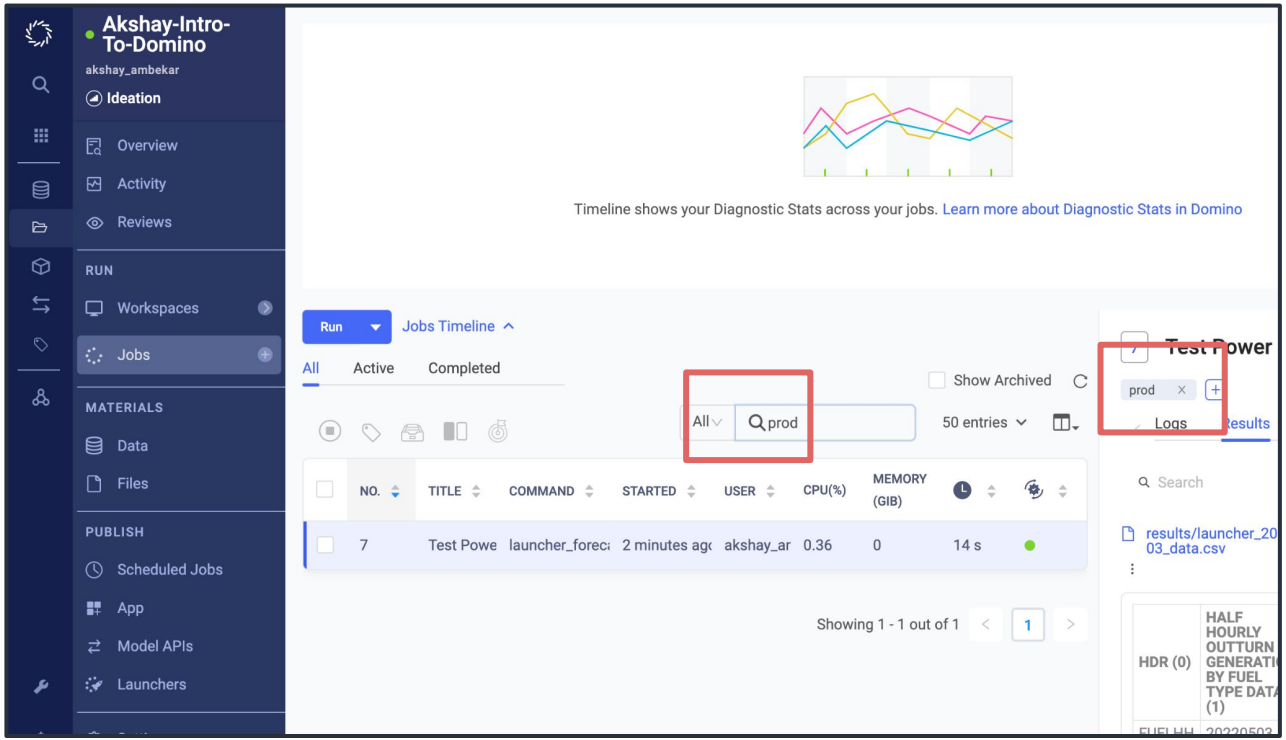
- Execute R, Python, or Bash scripts
- Domino keeps a snapshot of all files the code creates or modifies
  - Results between Jobs can be compared
  - Custom metrics can be tracked in the dashboard
- Multiple Jobs can be started at once and these are run in parallel
- Run Jobs from the Jobs dashboard or the code file itself

Jobs Dashboard

Dominostats can be used to track custom metrics

Jobs Dashboard

Tag Jobs to create separate dashboards for different experiments

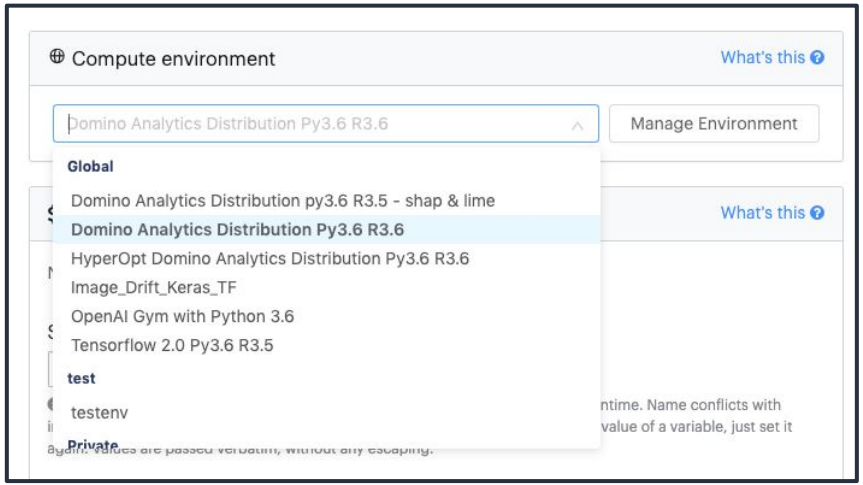


DOMINO

Environments

Environments in Domino

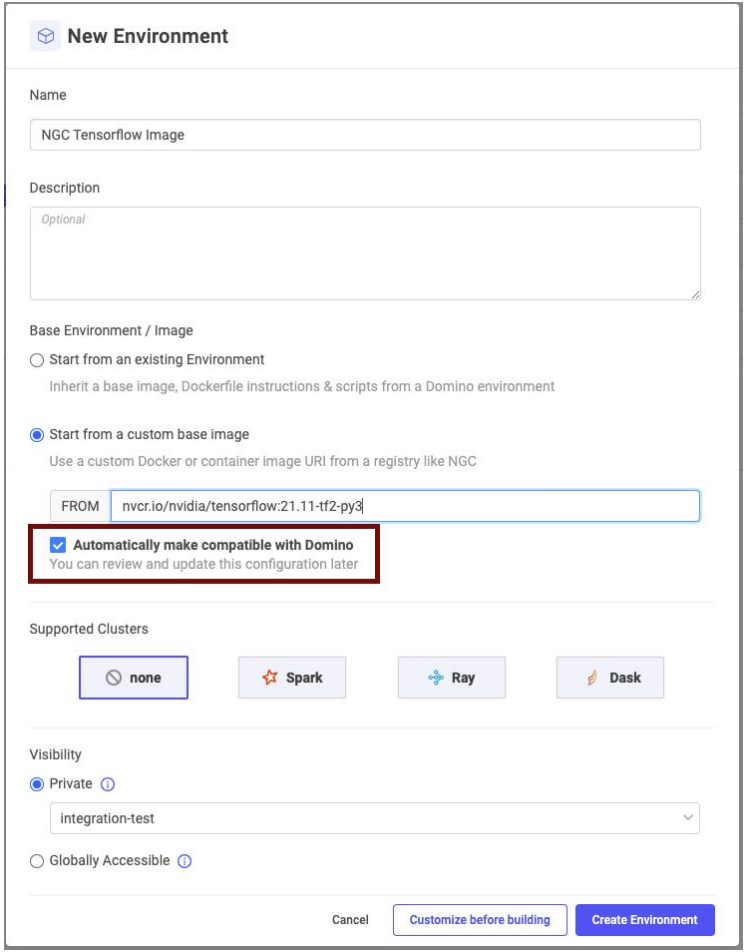
- An environment is a Domino abstraction on top of a Docker image, that provides additional flexibility and versioning
- When Domino starts your run, it creates an isolated Docker container based on the environment associated with your project



Environments in Domino

Domino comes with a set of Standard Environments, and it's easy to customize or create your own:

- Input Dockerfile instructions to add to an existing environment or
- Use a custom image, such as NVIDIA NGC containers, and have Domino make it automatically compatible with Workspaces and Jobs
- Learn more in Domino 201



Activity: Change your Compute Environment

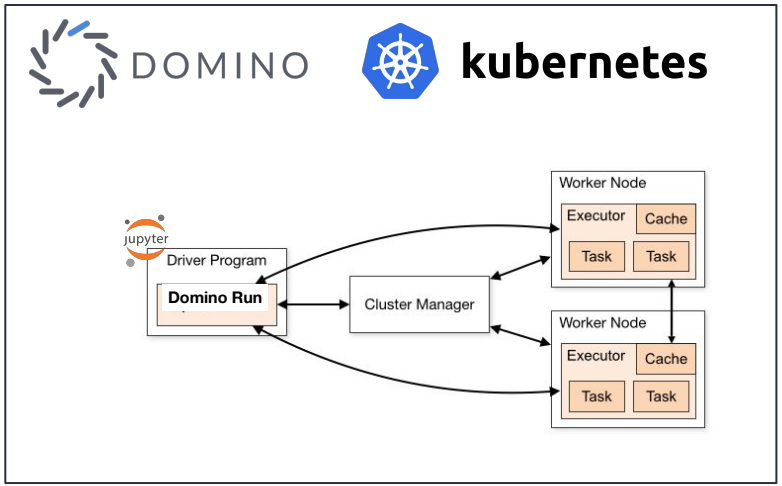
1. Navigate to the Settings for your training project
2. Update the Compute environment to ‘Intro to Domino Power Forecasting’
3. (Optional) Navigate back to Workspaces and note your existing workspace still uses the default environment. You have two options to update:
    a. Edit the Settings for that individual workspace
    b. Create a new workspace, and note it picks up the new environment

On-Demand Distributed Computing

Spark in Domino

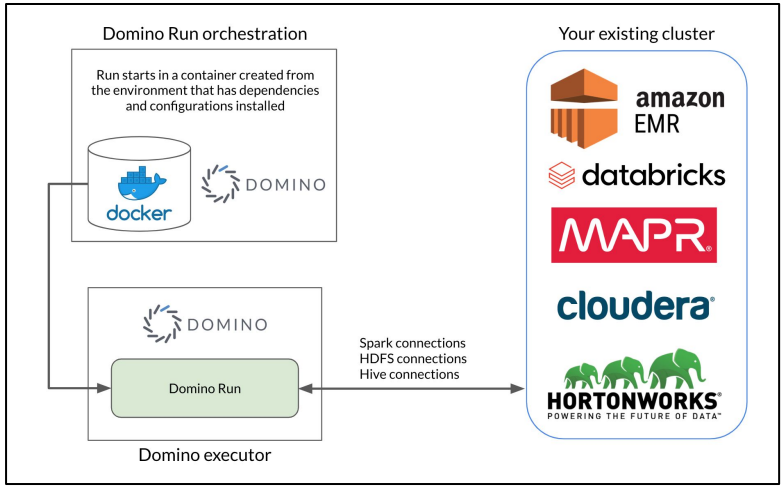
On-Demand Clusters

Natively orchestrated directly on Domino controlled infrastructure



External Clusters

Leverage existing long-lived Spark deployments in the enterprise



Spark on Demand in Domino: Use Cases

Distributed Machine Learning

Parallelize compute heavy workloads such as distributed training or hyper-parameter tuning

Take advantage of powerful machine learning algorithms from Spark MLlib

Interactive Exploratory Analysis

Efficiently load large data sets in distributed manner

Explore and understand the data using a familiar interface with Spark SQL

Featurization and Transformation

For experienced Spark practitioners

Sample, aggregate, and re-label large data sets

Optimal performance may require a practitioner who is skilled in tuning Spark

Ray on Demand in Domino: Use Cases

Distributed Multi-Node Training

Take existing PyTorch and Tensorflow models and scale them across multiple machines to dramatically reduce training times

Ray is suitable for both distributed CPU and GPU training

Hyperparameter Optimization

Launch a distributed hyperparameter sweep with just a few lines of code.

Take advantage of a large number of advanced parameter search algorithms.

Reinforcement Learning

Take advantage of a number of built-in reinforcement learning algorithms, along with a general framework for developing your own.

Dask on Demand in Domino: Use Cases

Working with Large Datasets

Great for scaling up Python data analysis or transformation where processing requires more resources than can be provided by a single machine.

Performed with minimal modification and without having to switch over to a different ecosystem like Spark.

Distributed Training

Dask provides a simple way to parallelize existing scikit-learn models.

For larger memory-bound workloads, there are Dask specific ML libraries (e.g. Parallel Meta-estimators, Incremental Hyperparameter Optimizers)

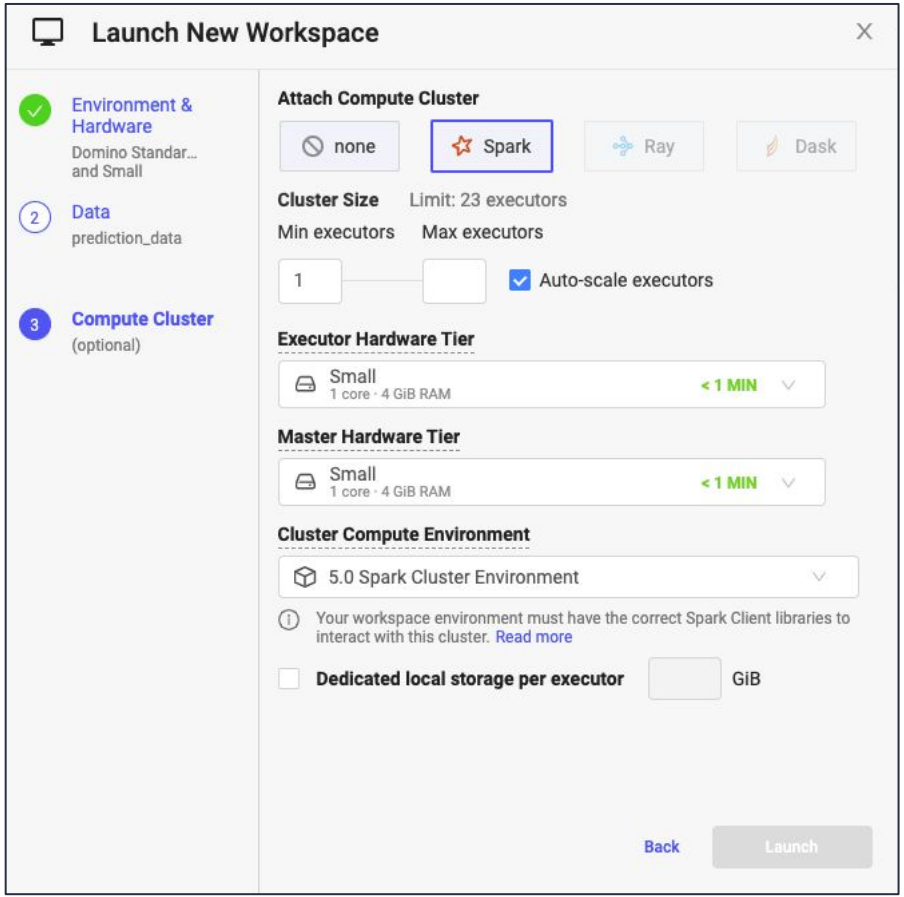
Custom Distributed Computations

The low-level Dask scheduling APIs can be used to build custom algorithms that can benefit from parallelism.

You control the business logic while Dask handles task dependencies, network communication, workload resilience, diagnostics, etc.

On-Demand Distributed Computing Overview

- Available only to the current execution
- Use with data residing outside the cluster - i.e. in Domino datasets or an S3 buckets
- Web UIs for the associated cluster are available in the same workspace
- Learn more in Domino 201
- Clusters can be auto-scaled



Model APIs

APIs: The Universal Translator

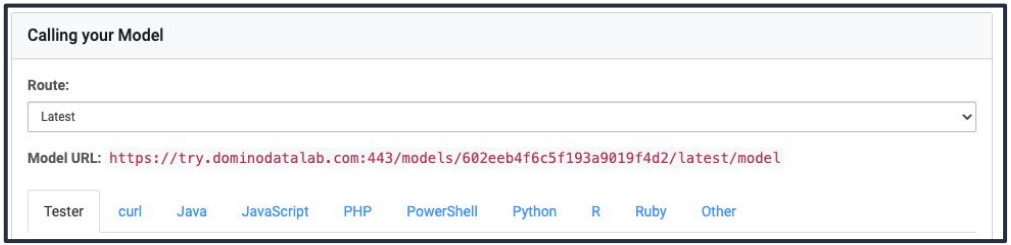
- A developer can create an application in the language of their choice AND still use your R or Python model
- Domino model APIs serve your code as a low-latency web service
- You don't need any API design experience to turn your Python or R code into a Model API
- Details in our Publishing course

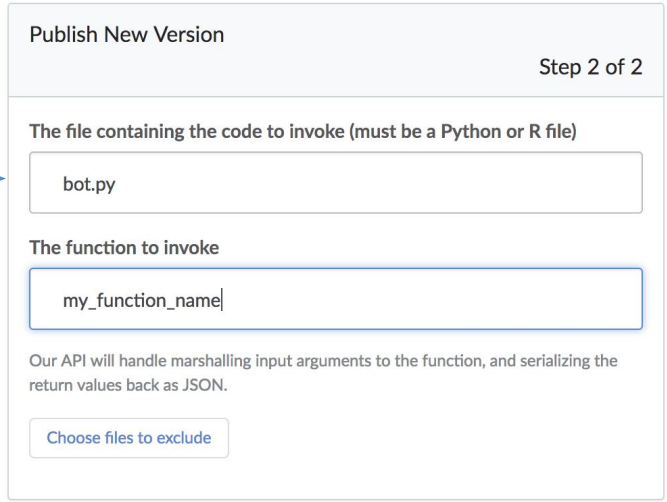
Build Model APIs in Domino

Script:

```
load model

function (input):
    result = call model (input)
    return result
```





DOMINO

Host Domino Model APIs Externally

Model APIs can be exported to run in NVIDIA Fleet Command, Sagemaker, or any external container registry

1. Build Model API in Domino,
2. Use Domino's REST APIs to export
3. Stop model instances to remove from Domino cluster

Activity: Create and Publish a Model API

1. Navigate to the ‘Publish/Model APIs’ page in your project
2. Select ‘New Model’
3. Name the model ‘<your name>-Power-Generation-Prediction’ (description optional) and select ‘Next’
4. Under file enter ‘model_forecast.py’, and under the function to invoke enter ‘predict’
    a. Select ‘Create Model’
    b. Peruse the model code while waiting for the model to deploy
5. Test your model after it is running under the Model ‘Overview’ tab with the following sample

```
{
  "data": {
    "year": 2022,
    "month": 6,
    "day": 7
  }
}
```

6. (optional) Copy your model URL and token into the Model_caller.ipynb notebook to call your model

Web Applications

Web Applications

- Each Domino project can host one web app
- Detailed instructions are available for Dash, Flask, Shiny, Streamlit, and Django
- See our Publishing course for more

Web Apps in Domino

App Files

/mnt/...

Start an app server

`app.sh`

DOMINO

Web Apps Access

Permission levels are separate from projects and include:

Anyone, including anonymous users

Users do not need a Domino account, but they do need network access

Anyone with an account

All users who have Domino accounts can view your App

Invited users (other users may request access)

All users who have Domino accounts and are logged in to Domino can request access to your App if it appears in the Launchpad

Invited users only

Only Domino users who are added by the App owner via the Invite People field can view the App. Users cannot request access

Take-home Activity: Create a Dash App

1. Peruse the files called ‘app.sh’ and ‘app.py’ in your training project
2. Navigate to Publish/App
3. Name your app ‘<your name> Power Generation in the UK’ and add an optional description
4. (optional) Decide whether to show your App in the launchpad, and what permissions it should have
5. Click ‘Publish’ on the bottom right
6. Once app is running select ‘View App’ to use it
7. **Stop** the app when you are finished

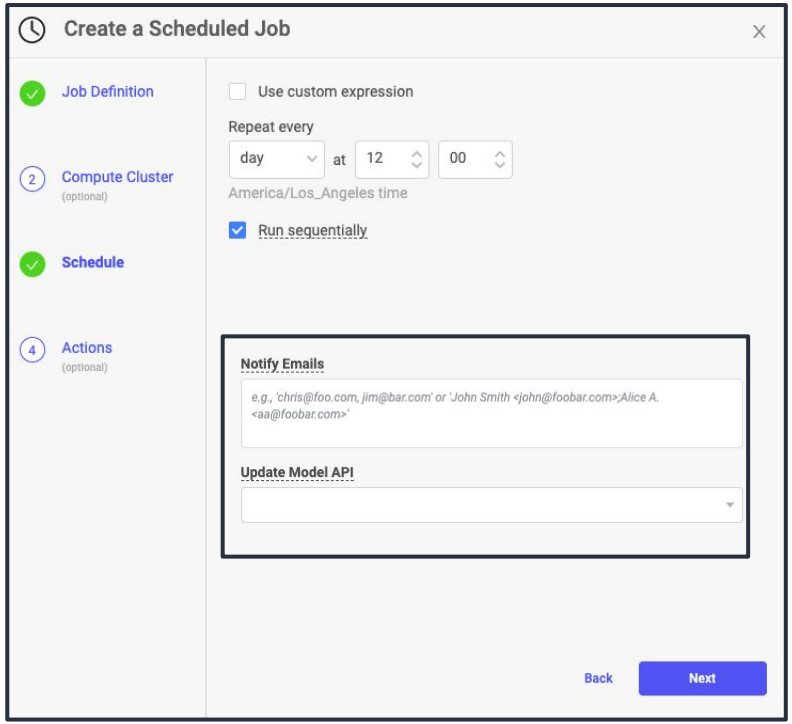
Scheduled Jobs

Scheduled jobs

- Schedule any Job in Domino to occur on a regular basis.
- Designate the recipients of the automatically generated email with the Job results attached

Useful for

- Refreshing data, i.e. data pump
- Scheduling regular reports
- Retraining models on regular basis



Take-home Activity: Create a Scheduled Report

1. Peruse the file called ‘scheduled_forecast.py’ in your training project
2. Navigate to the ‘Publish/Scheduled Jobs’
3. Under file name enter ‘scheduled_forecast.py’
4. Select a time in the next couple of minutes
5. Click ‘Schedule’
6. Verify Job runs at scheduled time and look at the results
7. Click the ⋮ next to the scheduled job and the select **Pause**

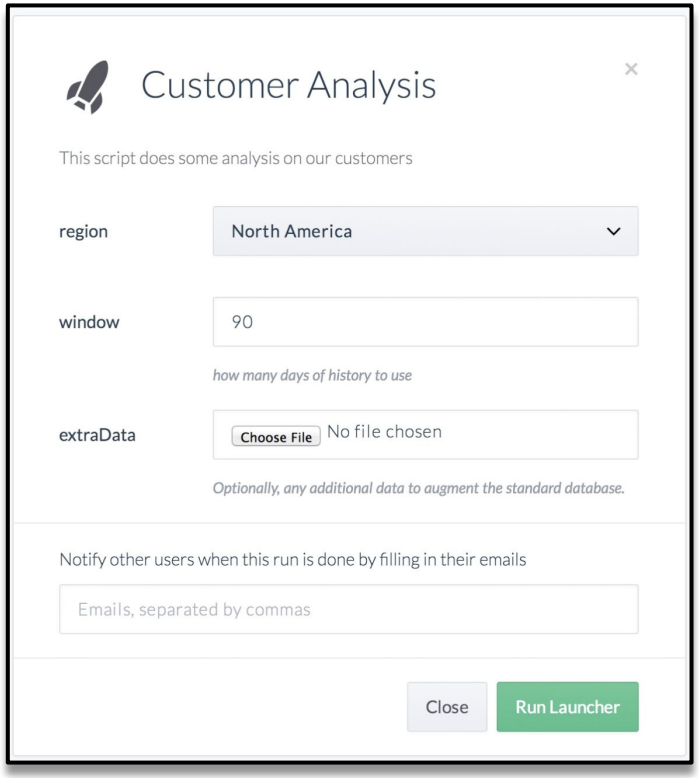
Launchers

Launchers Overview

Launchers are simple self service web forms on top of your script

Consumers of your project can upload data and specify parameters that are neatly consumed by your code

A separate permission level is available for Launcher Users, which limits them to only use Launchers and access Results.



Take-home Activity: Create a Launcher

1. Peruse the file called ‘launcher_forecast.py’ in your training project
    a. Note the comments with the list of allowed values for fuel_type
2. Navigate to the ‘Publish/Launchers’ tab and click New Launcher
3. Under command to run enter ‘launcher_forecast.py’
4. Click Add Parameter once to add the start_date parameter
    a. Change the name to date, type to “Date” and enter 01/01/2022 as default
5. Click Add Parameter again to add the fuel_type parameter
    a. Change the name to fuel_type, type to “Select” and enter a comma separated list of accepted values - CCGT, OIL, COAL, NUCLEAR, WIND, PS, OCGT, BIOMASS
6. Click Save Launcher, then go back to the Launchers list
7. Run the Launcher with a different start_date and fuel_type and see the results

Using Domino Programmatically

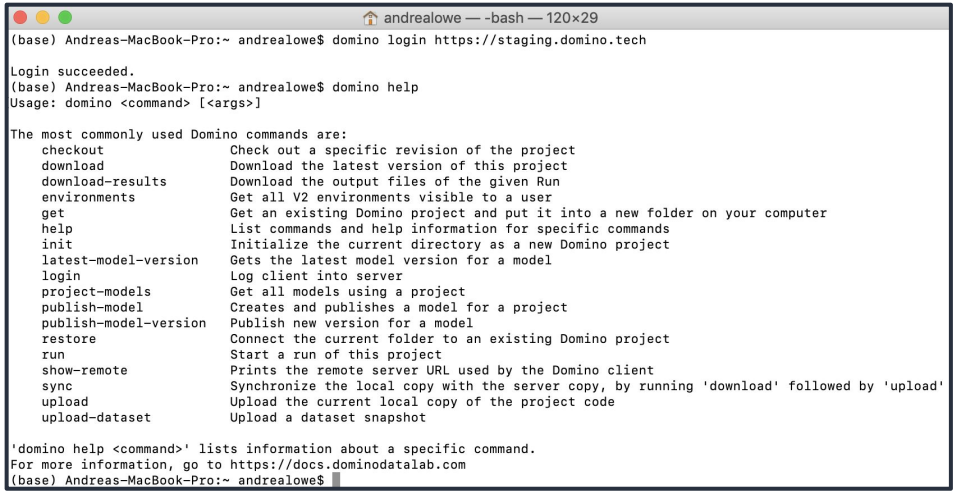
Domino Command Line Interface (CLI)

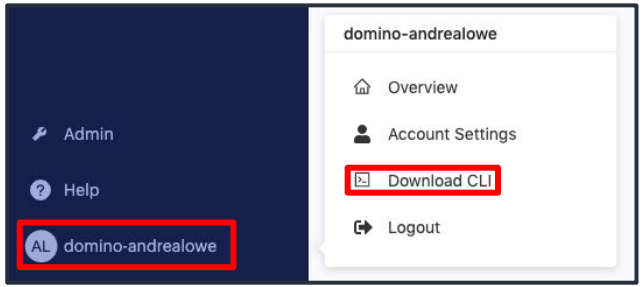
- Work locally but run your code in Domino
- Sync a local folder with a Domino project
- Easily upload many/large files to Domino

To start, download the CLI by clicking on your user name in the bottom left

After the download completes, login using the terminal (you may be given a link for authentication)

See list of possible commands here and in Domino 201





DOMINO

Domino API

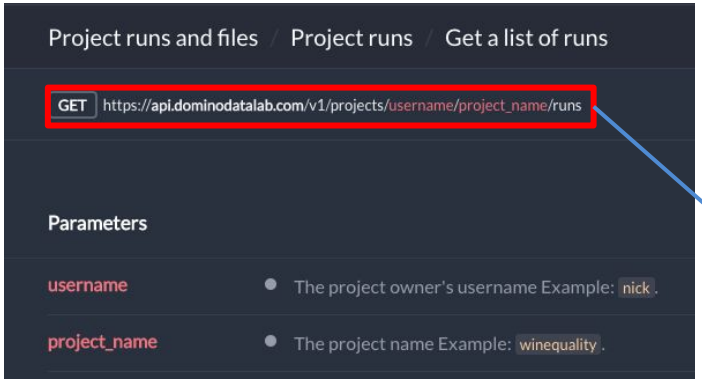
Much of Domino's functionality is available through a REST API

Available APIs can be found here (videos here) and can be used for workflows such as:

- Schedule extraction of new data and refresh models or datasets
- Parallelize your work with spread and collect flows
  - Start and inspect jobs programmatically
  - Automate meta-analysis across multiple experiments
- Get info about your work in Domino
  - Organizations, projects, models, and datasets
  - Admins can see compute time spend and other usage metrics

Domino API - How to Use

Domino's API:



Python script:

```
import requests
import json
import os

api_key = os.environ['DOMINO_USER_API_KEY']
username = os.environ['DOMINO_PROJECT_OWNER']
project_name = os.environ['DOMINO_PROJECT_NAME']

url = 'https://mycompany.domino.tech/v1/projects/{username}/{project_name}/runs'.format(username=username,
                                                                                        project_name=project_name)

headers = {'X-Domino-Api-Key': api_key}

r = requests.get(url, headers=headers)
```

- Replace 'api.dominodatalab.com' with your company's Domino URL
- Find your API key in account settings or reference it using the built-in environment variable if within a Domino workspace/job

Activity: Use the Domino API to Tag a Project

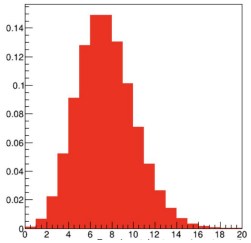
1. Peruse the file called ‘tag-project-api.py’
2. Use the Domino API to create a script to tag the project with ‘time-series’
    a. Refer to the API docs found here for the specification
    b. Keep the tag ‘time-series’ or change it to a different tag
3. Select ‘Run’ from the top right corner
    a. Note you can also run the same code from a workspace, either by pasting into a notebook or by running the script from a Jupyter(Lab) terminal
4. After the job has completed, navigate to the project overview page and look at the tags

Integrated Model Monitoring

Model Monitoring Approach

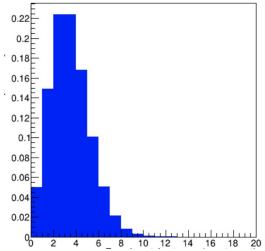
Training Data

Used as reference distributions



Prediction Data

Production data with features & predictions



Pred. Class
1
0
1
1

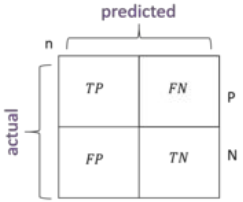
Ground Truth Data

Ground Truth labels created by experts or actual outcomes

Act. Class
1
1
0
1

Statistical Tests

Divergence Metrics

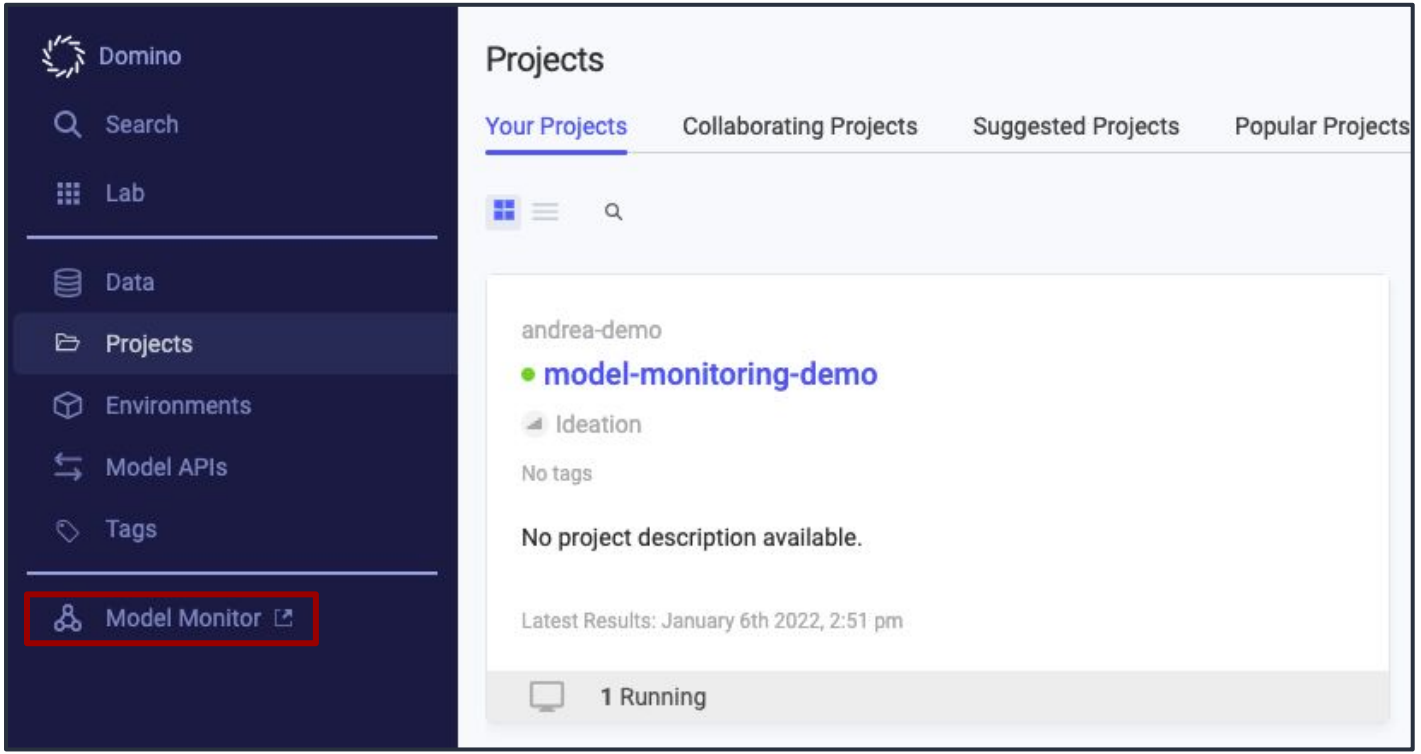


Confusion Matrix

Model Quality Metrics

DOMINO

Access in Domino Navigation Bar



Model Monitor Workflow

1

Register Model

Register model & training data to create reference patterns

2

Configure Checks

Configure tests, thresholds, and schedule for the checks

3

Ingest Production Data

Setup the system to frequently ingest Prediction and Ground Truth data

4

Receive Alerts

Receive alert emails when any of the scheduled checks fail

Integrated Model Monitoring

- Domino Model APIs can automatically send prediction data for monitoring
- Training sets can be created using the Domino Data API
- These are a versioned set of data, column info, and other metadata

```
from domino.training_sets import TrainingSetClient
tsv = TrainingSetClient.create_training_set_version(
       training_set_name="",
       df=training_df,
       key_columns=[],
       target_columns=[""],
       exclude_columns=[],
       meta={"experiment_id": "123456"},
       monitoring_meta=model.MonitoringMeta(**{
         "categorical_columns": [],
         "timestamp_columns": [],
         "ordinal_columns": []
       }),
       project_name=""
)
```

Set up Model APIs for Integrated Monitoring

Use Domino's client libraries to ingest prediction data for monitoring:

```
from domino_data_capture.data_capture_client import DataCaptureClient
data_capture_client = DataCaptureClient(feature_names, predict_names, metadata_names)
```

Then add to model function:

```
data_capture_client.capturePrediction(feature_values, predict_values,
metadata_values, event_id, timestamp, prediction_probability, sample_weight)
```

See R version here

Data Storage for Integrated Model Monitoring

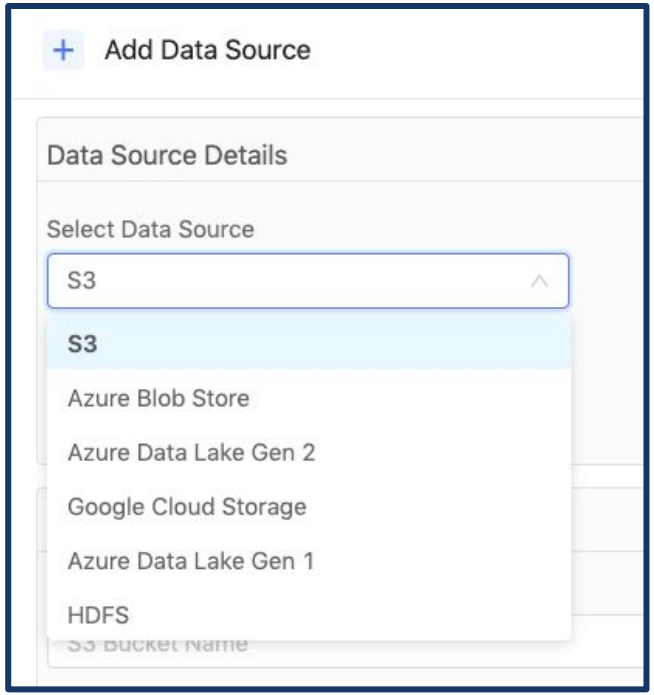
- Each instance of your model API will store one Parquet file per hour
  - Do not alter file names, these are stored as: `/domino/datasets/local/prediction_data/<model_version_id>/$$date$$=<date_in_utc>/$$hour$$=<hour_in_utc>/predictions_<uuid>.parquet`
- This prediction data is automatically consumed by the model monitor after setup
- By default, a daily cleanup job deletes data older than 30 days, or the oldest data beyond 300GB

Upload Data for Any Model

DMM can be used with the following data sources:

- Amazon S3
- Azure Blob
- Azure Data Lake Gen 1
- Azure Data Lake Gen 2
- Google Cloud Storage
- HDFS

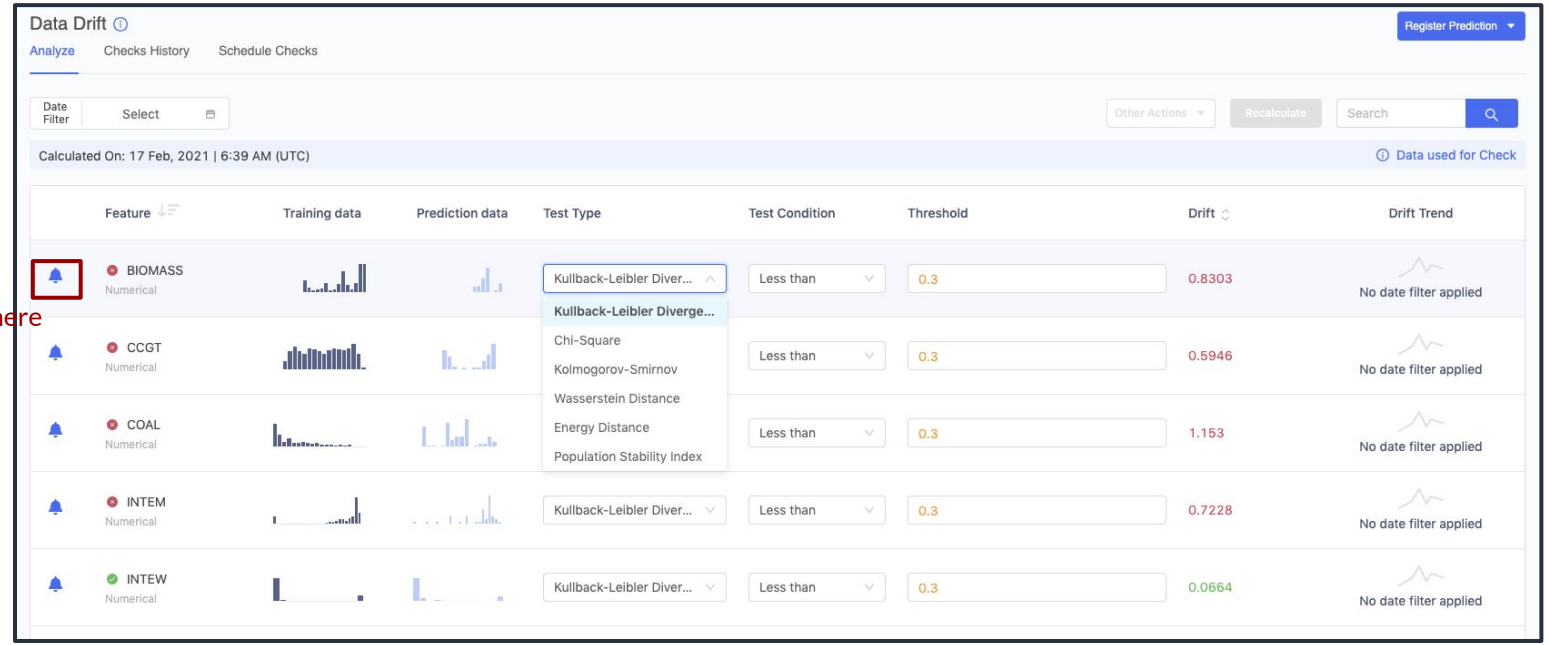
See docs for specifics on using each data type



Data Drift Monitoring

Set up preferred tests, thresholds, and disable alerts

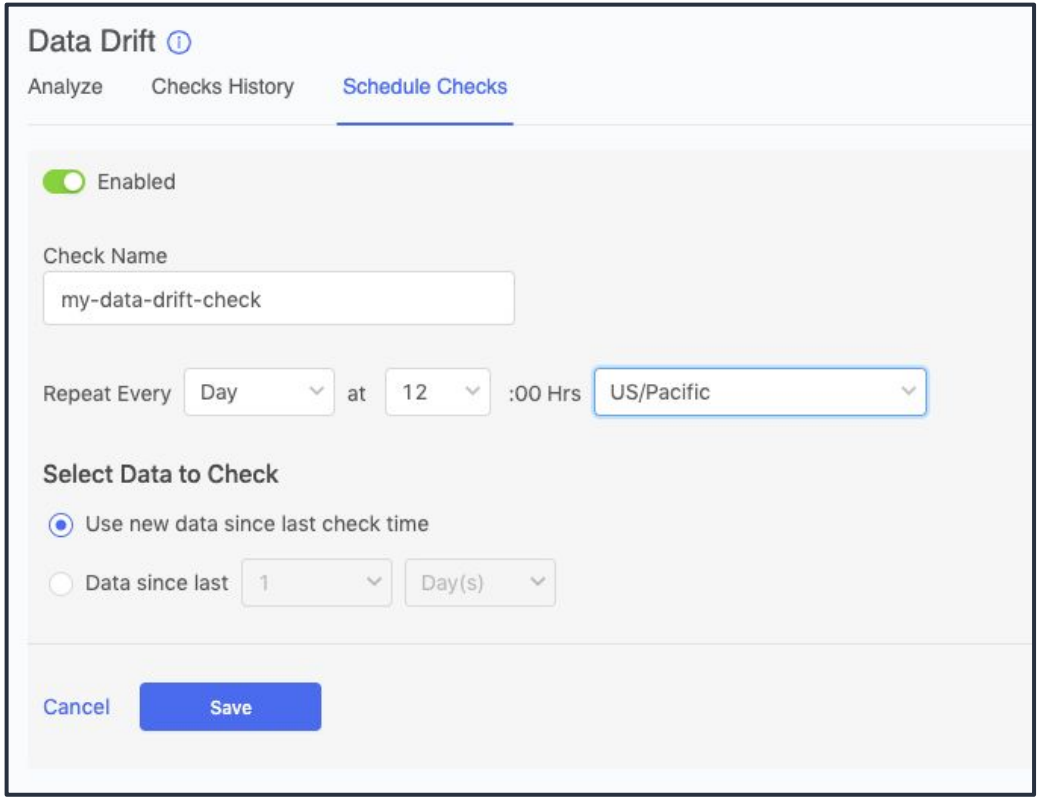
Disable alerts for specific features here



Schedule Data Drift and Model Quality Checks

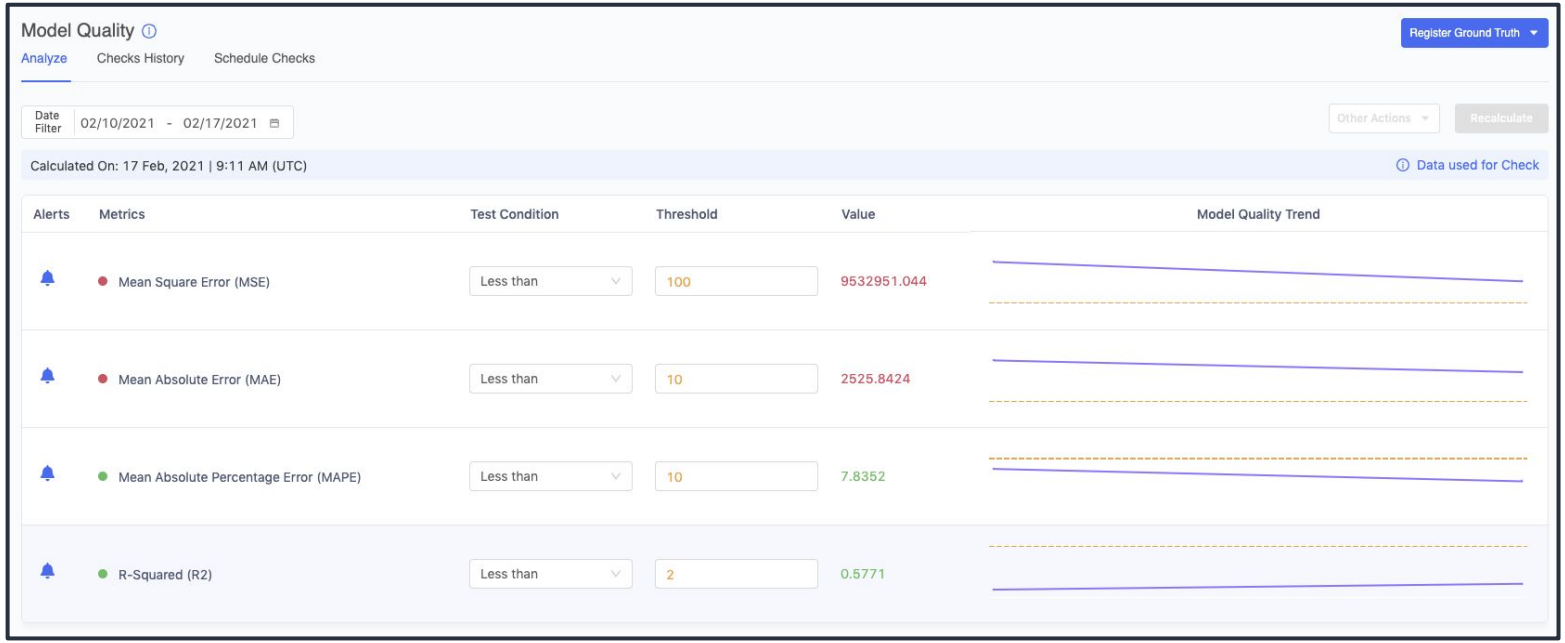
Automate notifications if data drift or model quality metrics degrade beyond a set threshold

Select Data to Check ensures that only predictions with timestamps in selected range are used



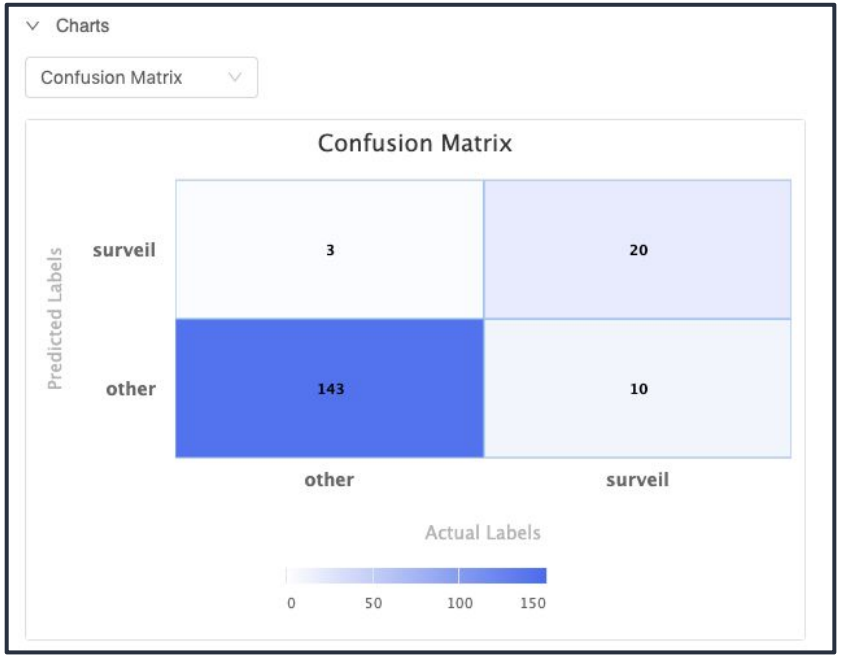
Model Quality Monitoring

You can set up preferred tests, thresholds, disable alerts, and filter on date



Model Quality Monitoring

Classification models will also show the confusion matrix and report



Charts

Classification Report

Class	Precision	Recall	F1	Support
other	0.9346	0.9795	0.9565	146
surveil	0.8696	0.6667	0.7547	30
Accuracy	-	-	0.9261	176
Weighted Avg	0.9235	0.9261	0.9221	176
Macro Avg	0.9021	0.8231	0.8556	176

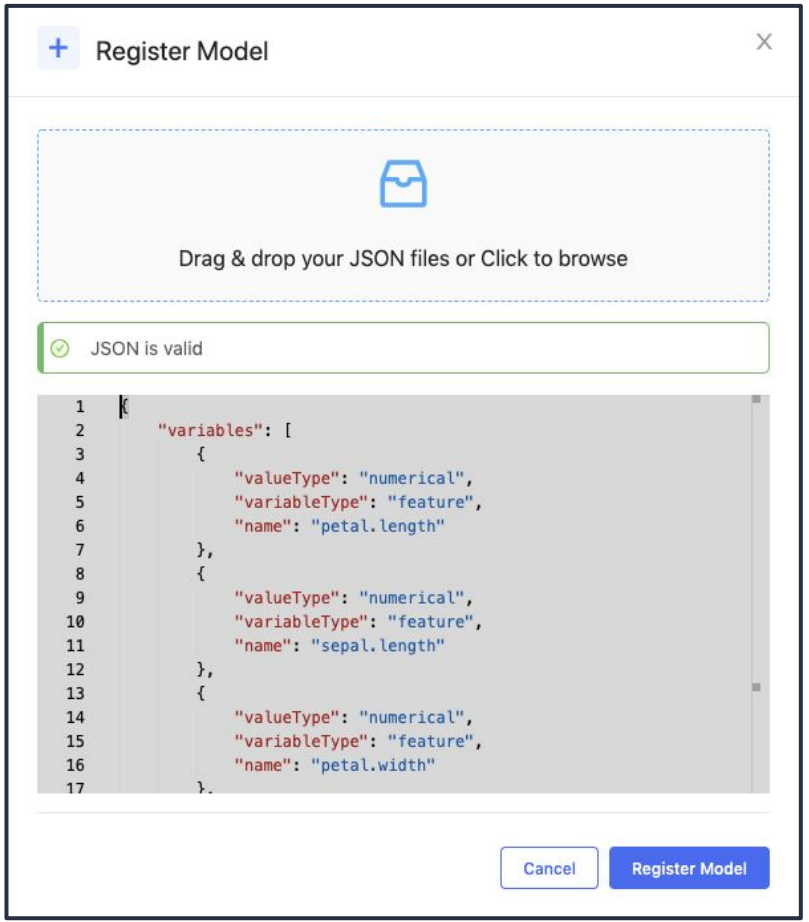
DOMINO

Register Model

Register model with a config file

The config file is a JSON object with info about the model

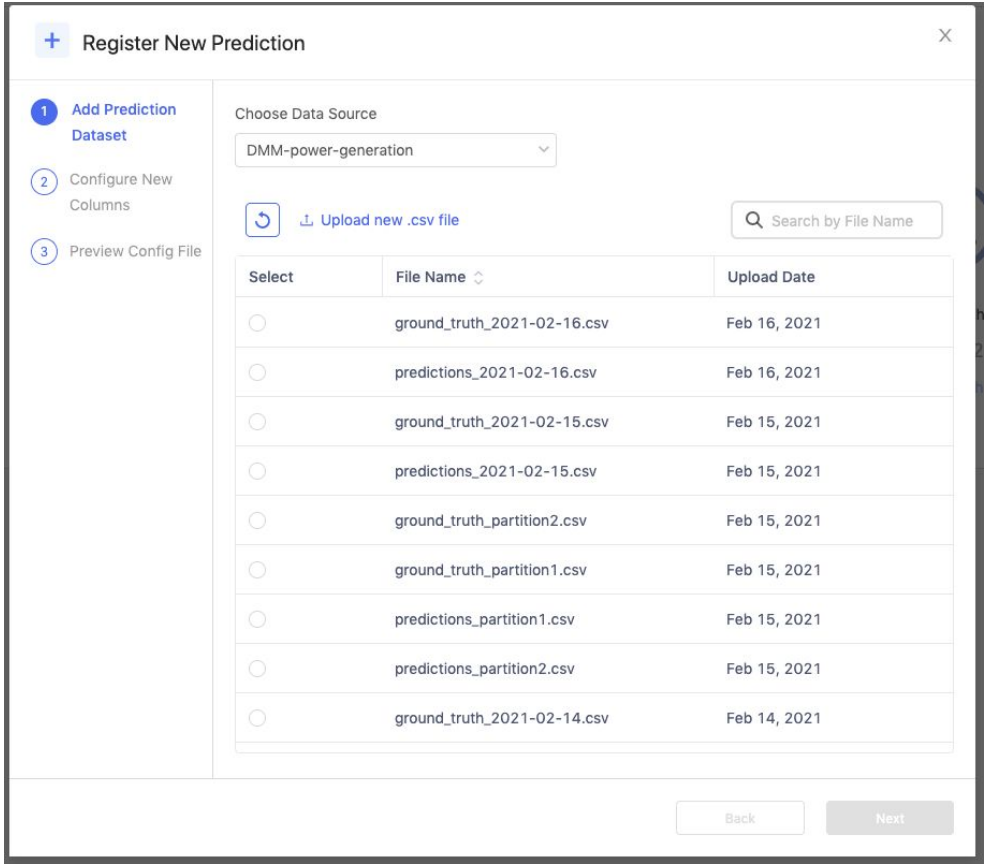
This can be customized to override binning method or provide pre-computed distributions



Register Predictions and Ground Truth

Choose data from any data source and configure any new columns

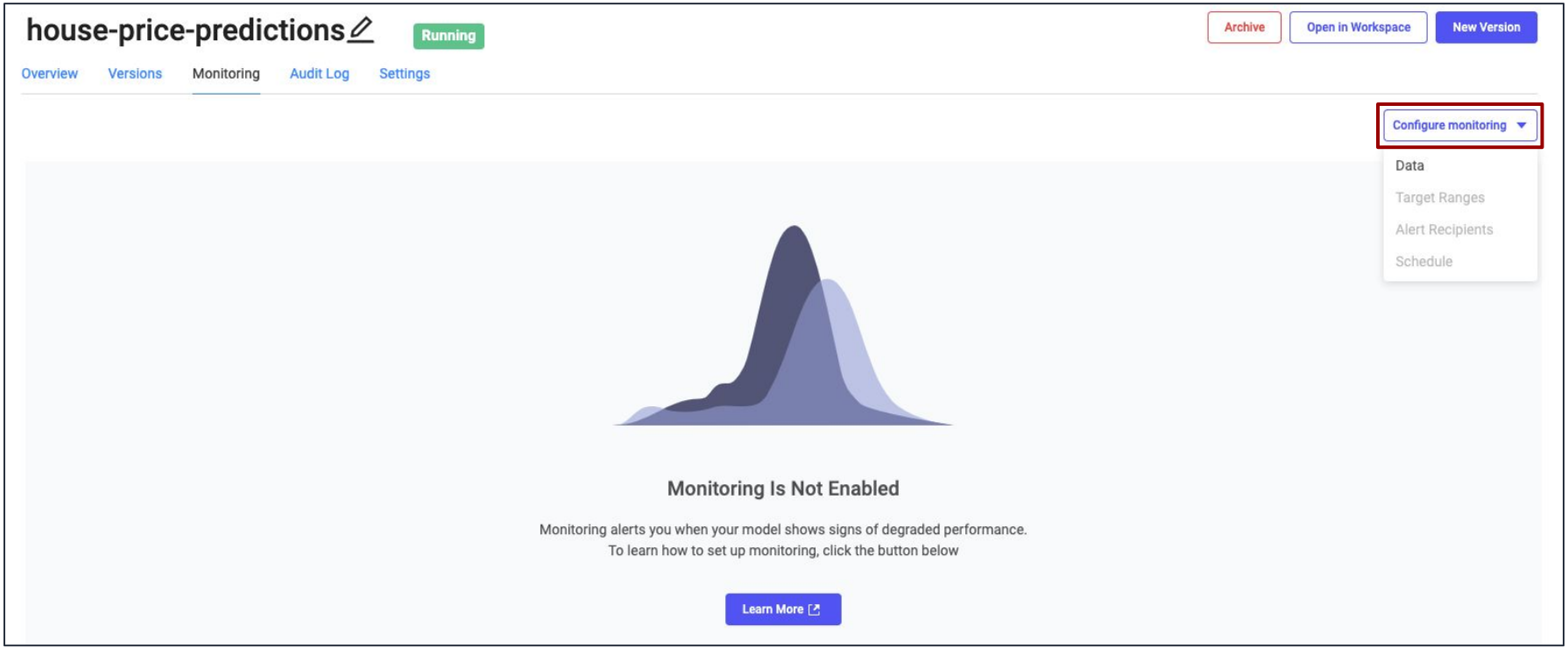
Map ground truth column to prediction column



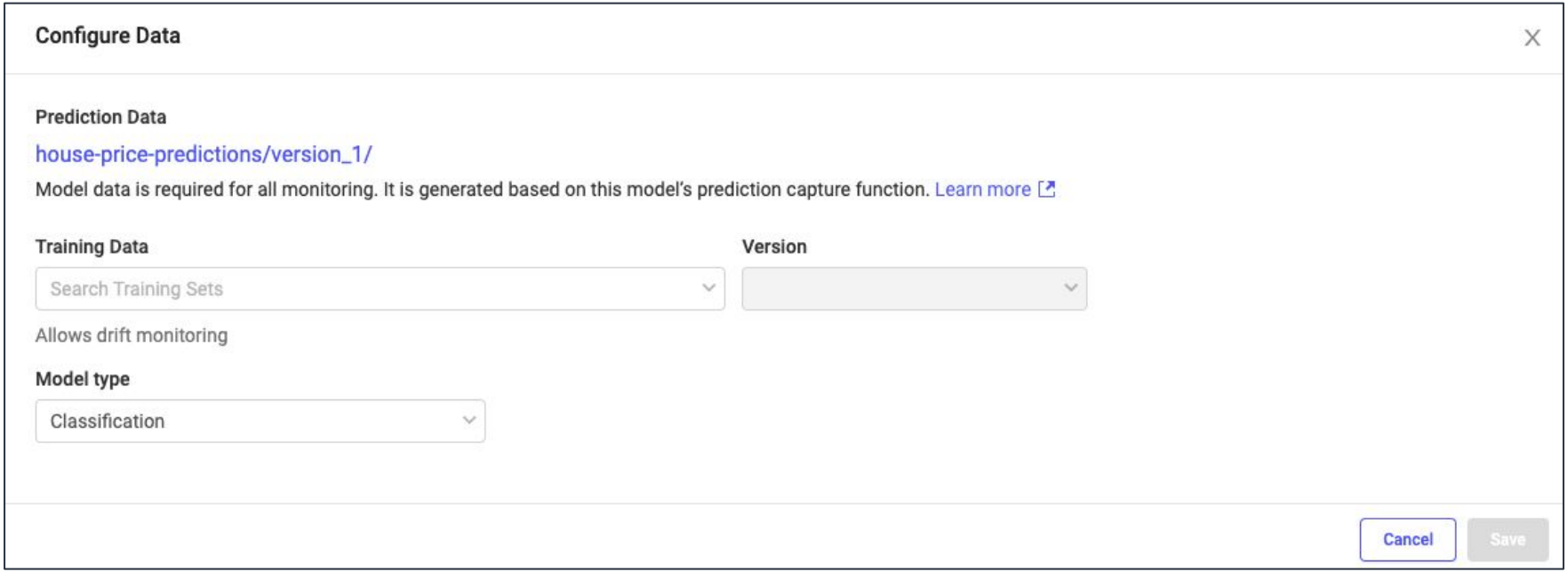
Take-home Activity: Explore Data Drift

1. Navigate to the Model Monitor page
2. Open any model
3. Open the data drift page
4. Change the test type and recalculate

Activate Monitoring



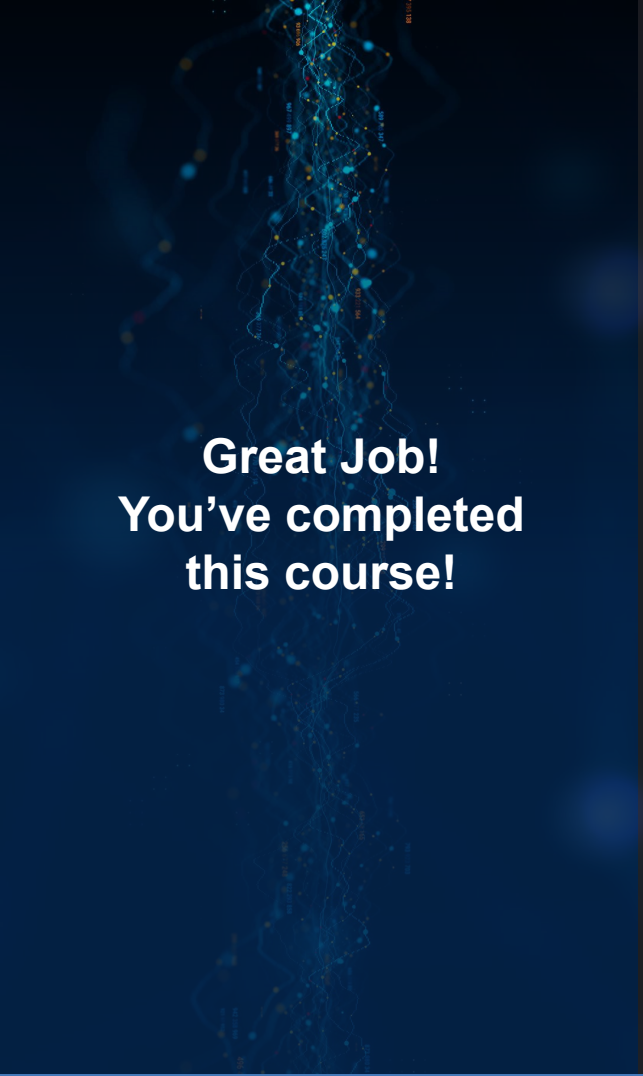
Activate Monitoring



Activity: Configure Monitoring

For the Model API, enable monitoring by selecting the training set to track drift against

1. Go to your newly published Model API overview page and navigate to the ‘Monitoring’ tab
2. Open ‘Configure Monitoring’ > ‘Data’ to select the right training set and version and set the model type to ‘Regression’
3. Call the model multiple times by pasting your model URL and token into the Model_caller.ipynb notebook and running
4. Refresh and view drift, set thresholds, etc



Great Job!
You've completed this course!

Project Setup	Settings and Collaboration
	Git in Domino
	Data in Domino
Running Code	Workspaces and Jobs
	Environments
	On-demand Compute Clusters
Publishing	Model APIs
	Web Applications
	Jobs and Scheduled Reports
	Launchers
Operationalizing	Use Domino programmatically
	Integrated Model Monitoring

DOMINO

Need Help!?

The Docs:
docs.dominodatalab.com

Community:
community.dominodatalab.com

Submit a Ticket:
tickets.dominodatalab.com

Self-service Training:
learn.dominodatalab.com

Support

DOMINO