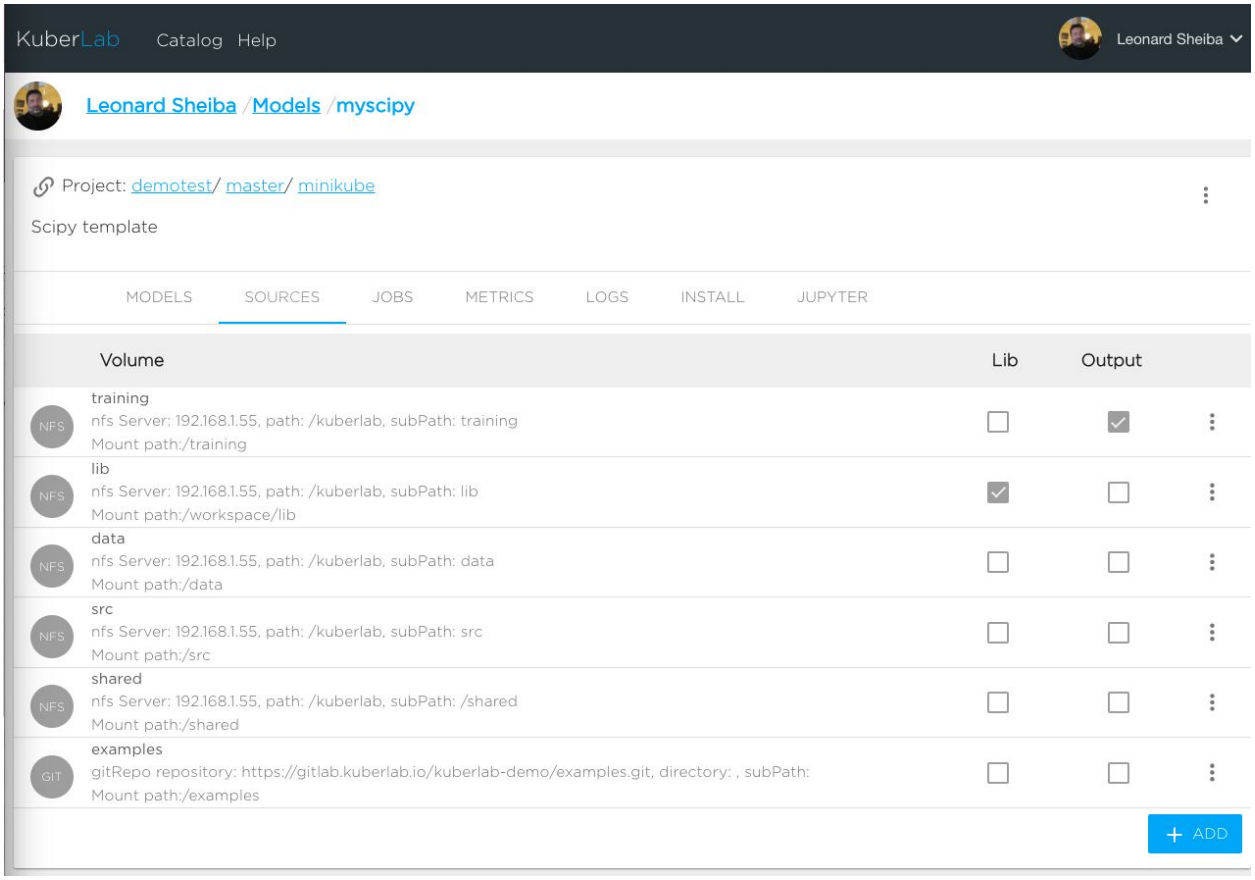
Storage and Data Sources Tutorial

When model created from generic template it will be configured with default set of data sources. Let's go to the SOURCES tab and examine them in details.



On the SOURCES page we can see the list of Volumes. Each volume represent a datasource configured to be accessed as folder in file system accessible by the model code or by the tools like Jupyter or Tensorboard integrated into environment.

When model is created from the template, those volumes, preconfigured in the template, will be initialised. Some volumes are mandatory, some optional and specific to the template and user can add and configure custom volumes based on the available datasource types.

Mandatory Volumes - every volume has corresponding environment variable exposed throughout the model environment

1. TRAINING - $TRAINING_DIR
   TRAINING volume (or training folder) is used to save training logs and checkpoints. TensorBoard is looking at the folder of the training volume to look for data needed to

display analysis of the model training. Serving module will also use training folder as a source for the model to serve

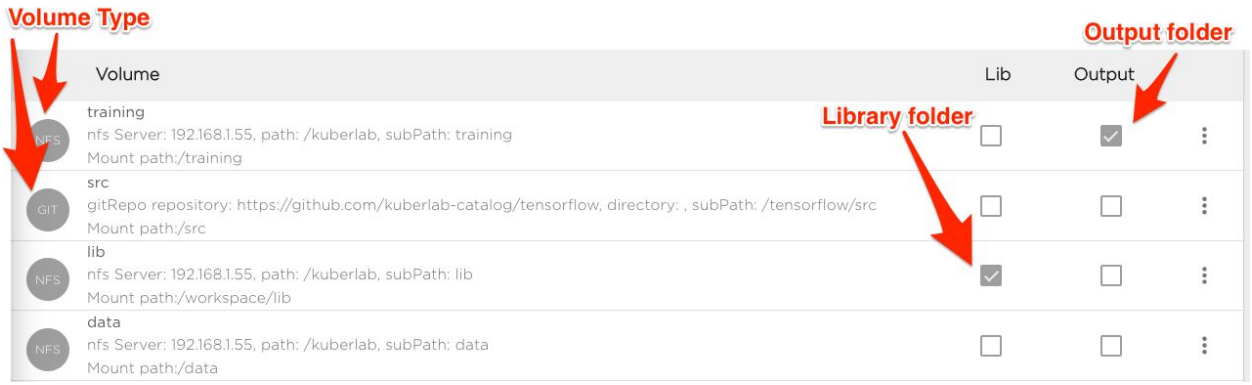
2. LIB - $LIB_DIR
   LIB volume (or lib folder) is used to install third party libraries and component during runtime. This functionality is exposed on INSTALL tab.
3. DATA - $DATA_DIR
   DATA volume (or data folder) is where model will look for data by default. There are several ways to save the data to the data folder which we will describe later.
4. SRC - $SRC_DATA
   SRC volume (or src folder) is the mapping of the model template git repository into the model storage space. The volume type is GIT and it is not persistent. In the future it will be possible to perform standard git operations on the data located in the GIT volume.
5. SHARED - TBD

Custom Volumes

1. EXAMPLES
   EXAMPLES volume (or examples folder) is specific to the model template. In this case It is referencing git repository with example project. This volume has type GIT and is not persistent.
2. USER volume is created by model developer. It can have any type of available storage resource.

Supported Volume Types:

1. GIT - not persistent, map GIT repository into model environment
2. NFS - persistent, mount NFS shares into model environment
3. S3 - persistent, map S3 buckets into model environment
4. CLUSTER - persistent storage predefined as part of the cluster, require minimum or no configuration



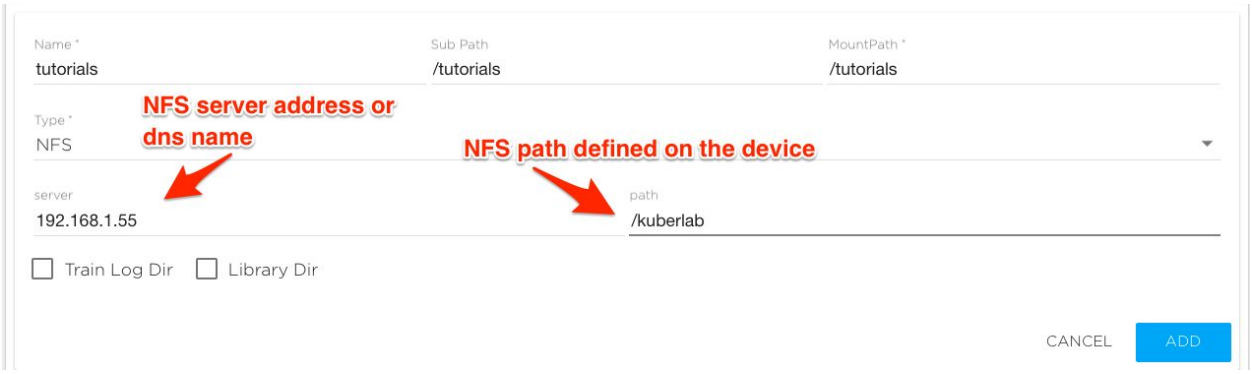
Volume Definition

Volume defined by the following attributes:

1. Name of the volume
2. SubPath defines the folder position in the filesystem layout of the storage device. If SubPath is started with leading “/” it is shared and will be defined relative to the shared folder of the storage device. Without leading “/” folder is private to the model and will be defined relative to the private model folder
3. MountPath defines the folder as it is seen by the model code and other components.
4. Volume type can be:

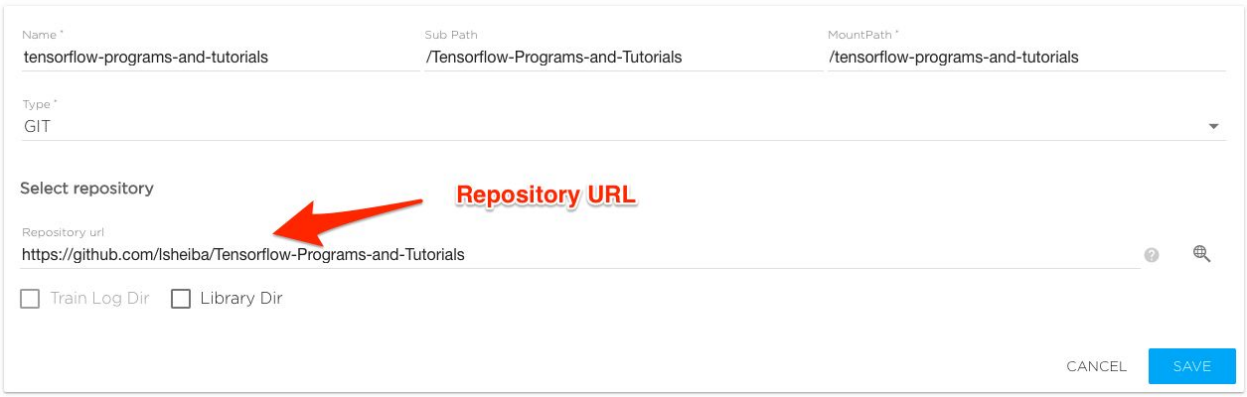
Cluster Storage volume:
This is the volume defined by the storage configured as part of the cluster. The configuration parameters are used from cluster configuration automatically.

NFS volume

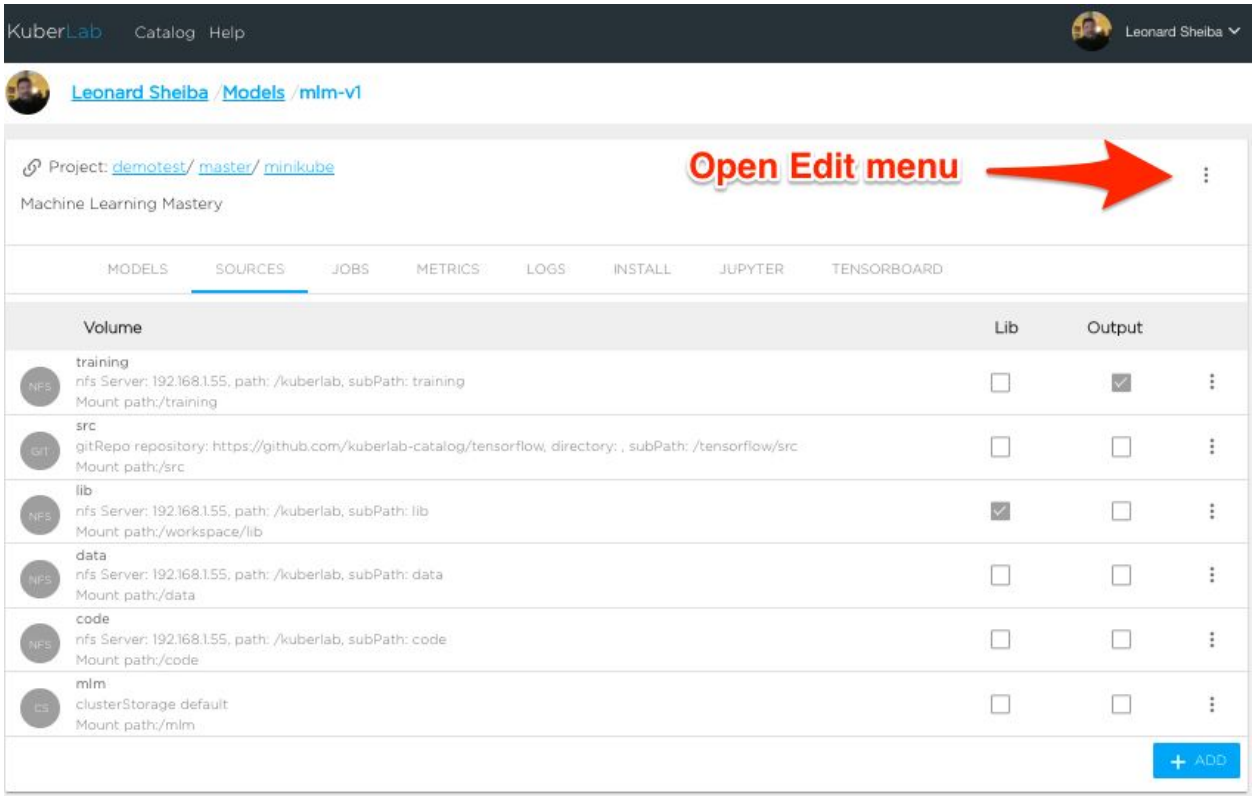


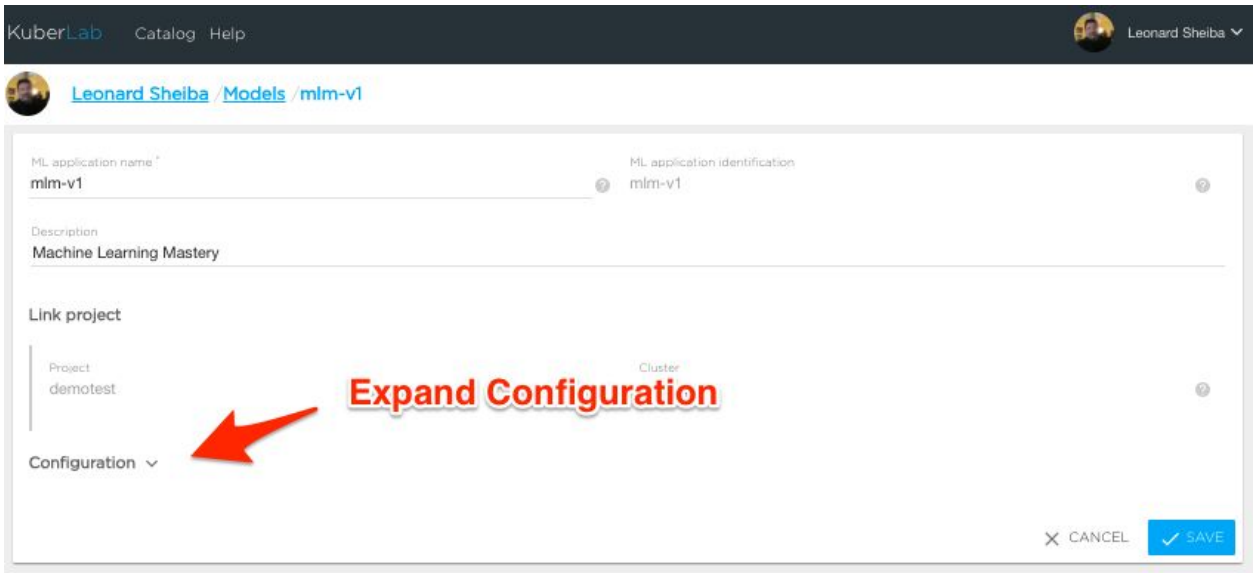
S3 volume

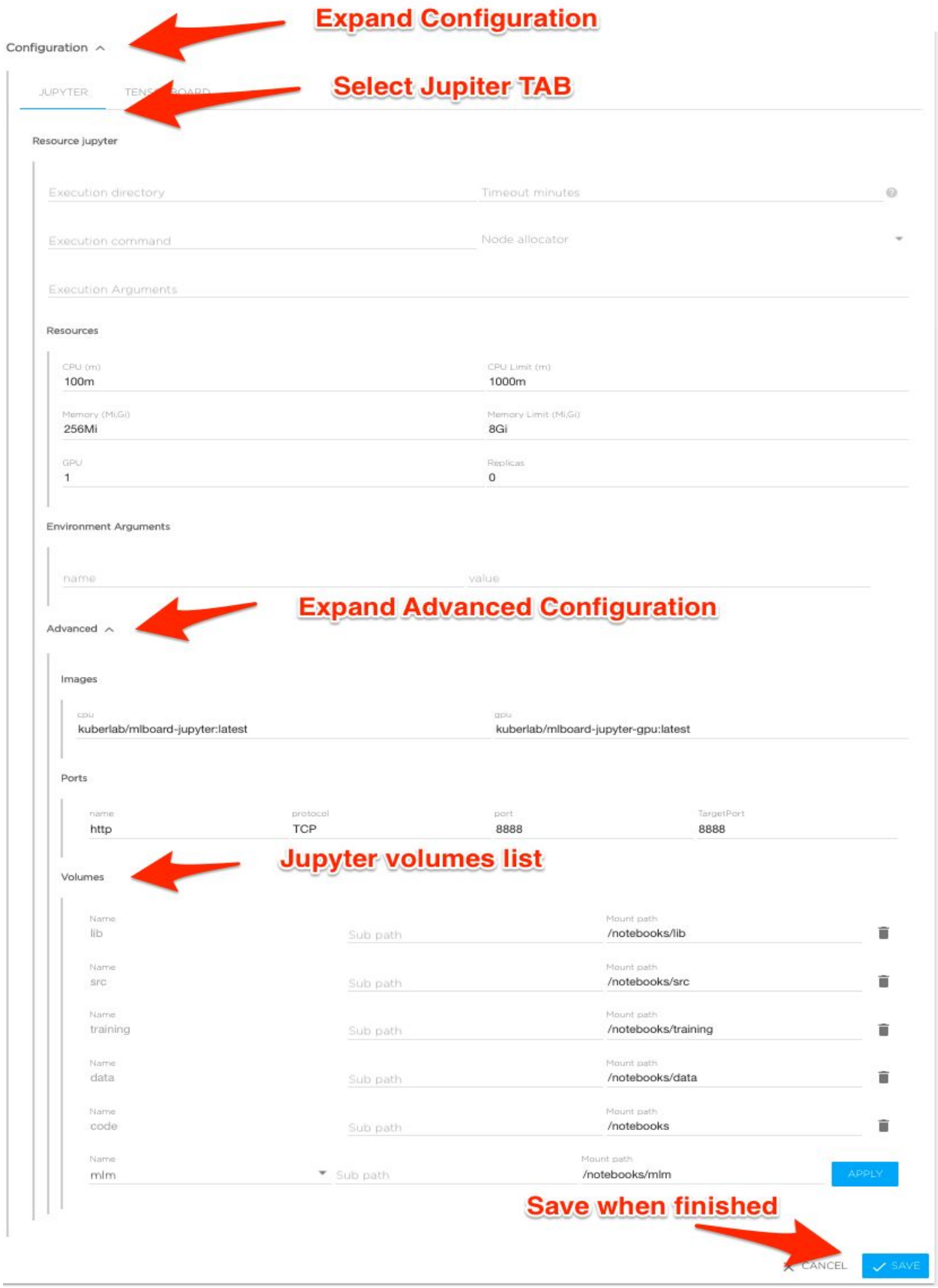
GIT Volume



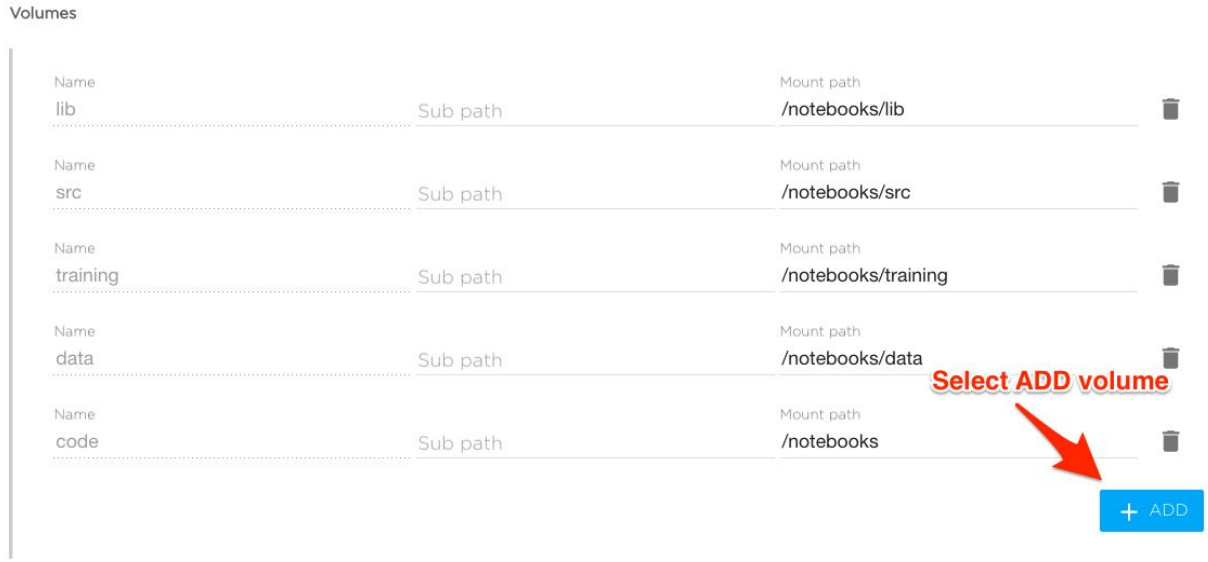
Volume mapping for Jupyter.
Jupyter require additional configuration which will map Volumes into Jupyter folder. To do that several steps need to be done.



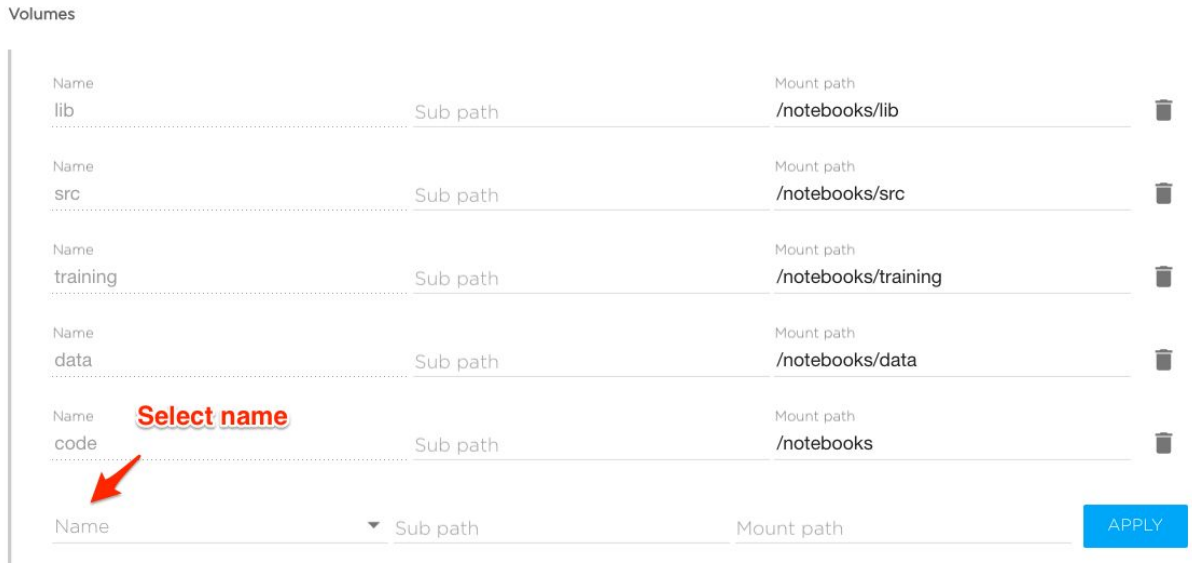




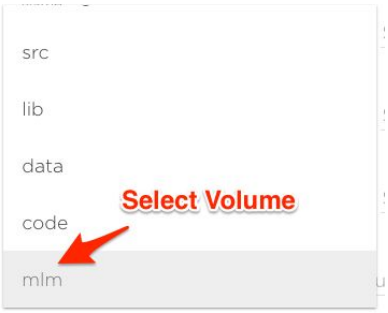
In the volumes section of Jupyter configuration form select ADD volume



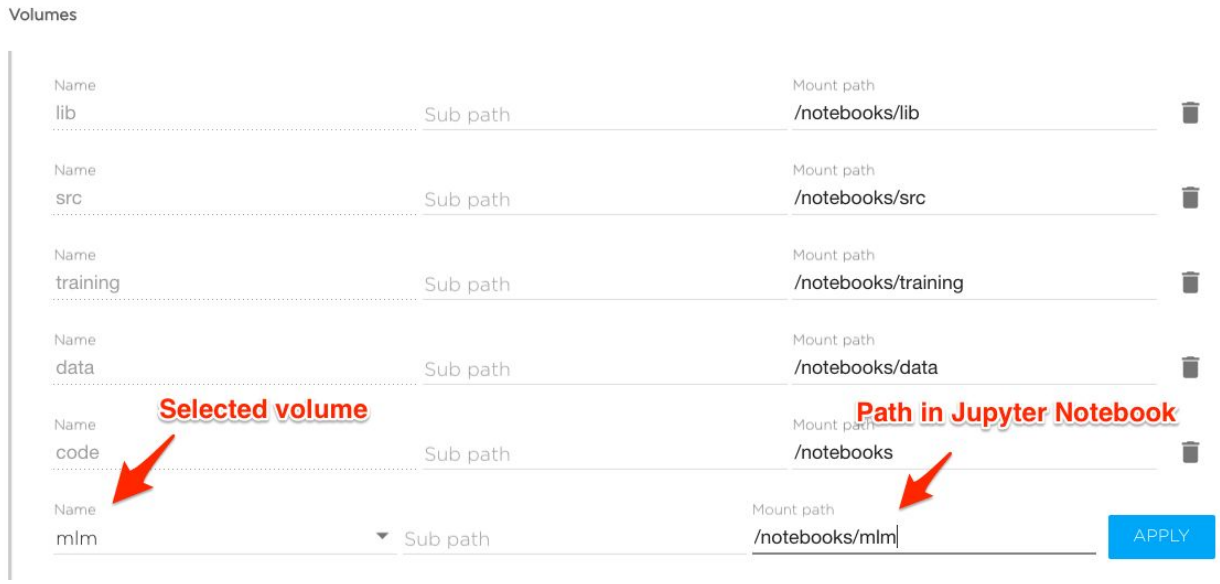
Select volume Name to use



When you select Name, drop down menu will show available volume names to attach to



Type the MountPath where to map new volume relative to the root folder “/notebooks”. If you type path “/notebooks/mlm”, when you open Jupyter notebook you will see “/mlm”



Afterwards select APPLY and SAVE configuration. Environment will be rebuilt and reloaded, which may take a little extra time.