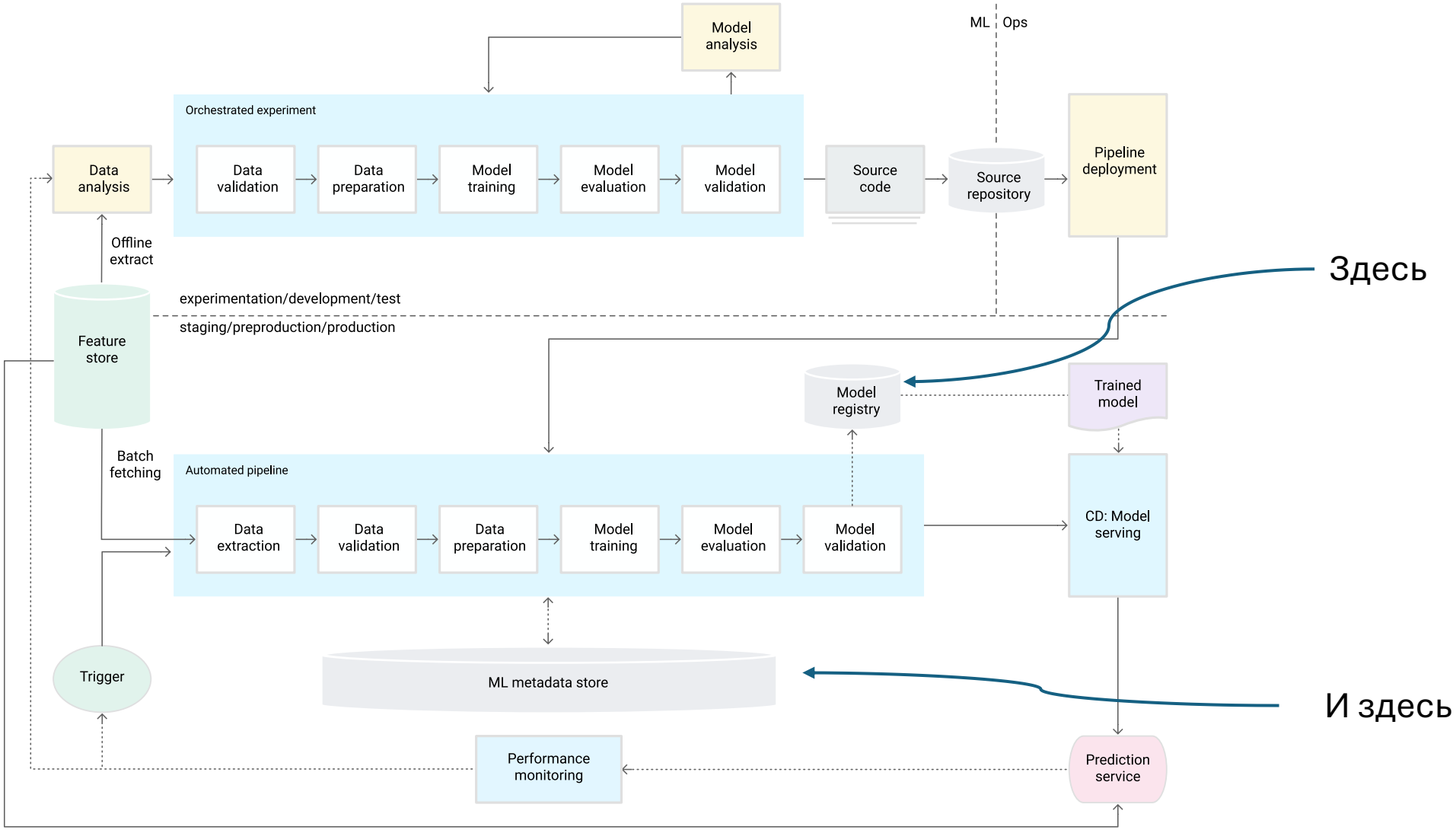
MLflow & ClearML

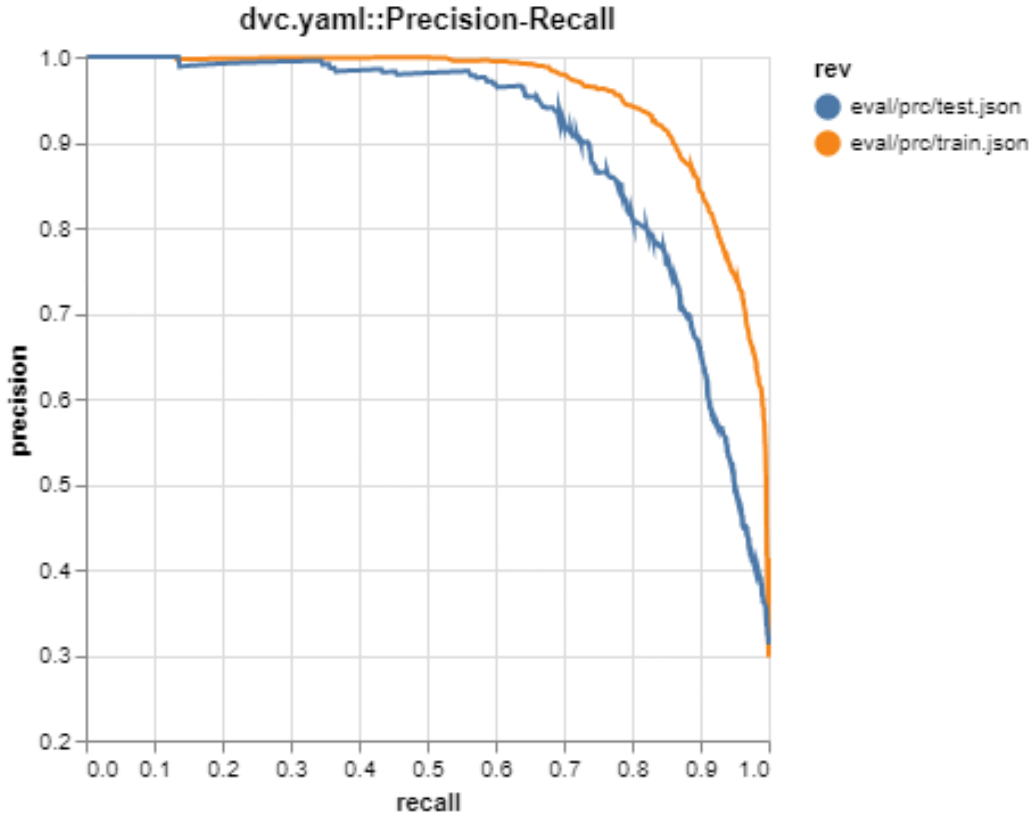
Лекция №6

Где находятся MfFlow & ClearML?



Как умеем смотреть метрики

```
$ dvc plots show
file:///Users/dvc/example-get-started/dvc_plots/index.html
```

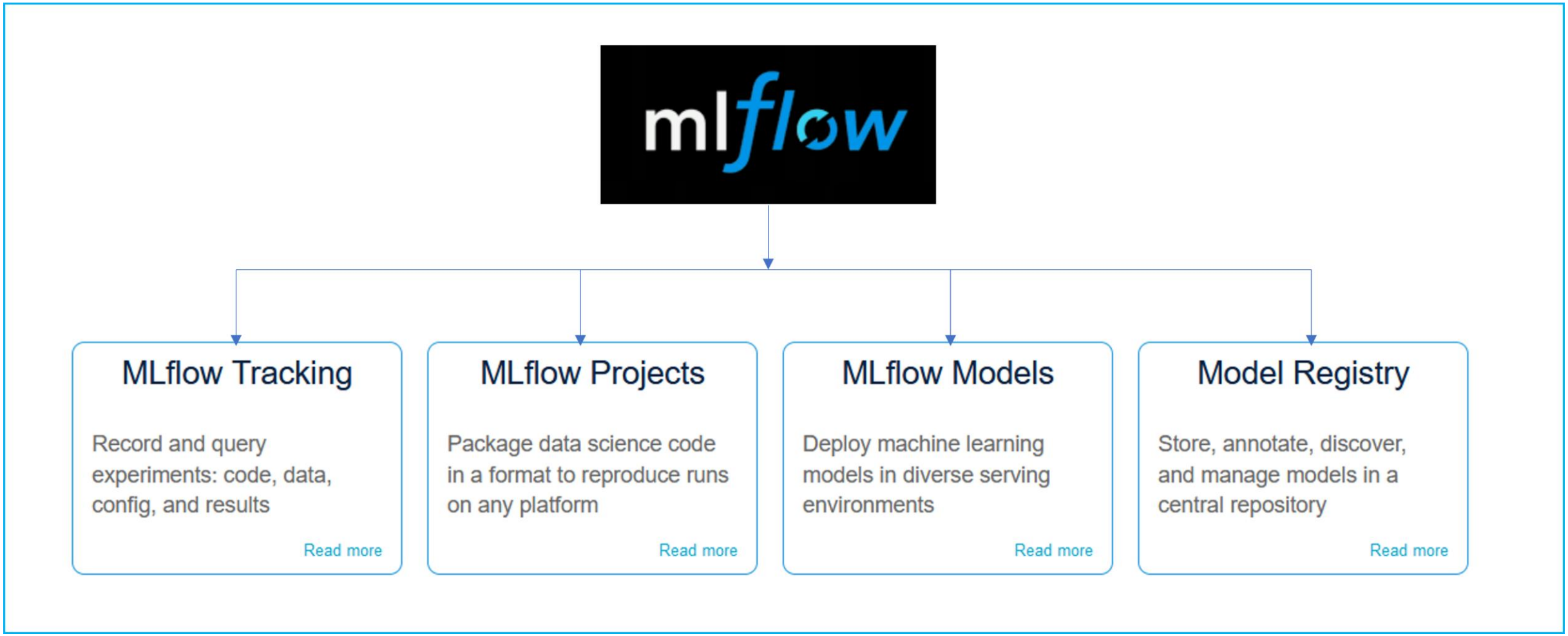


Как умеем смотреть метрики

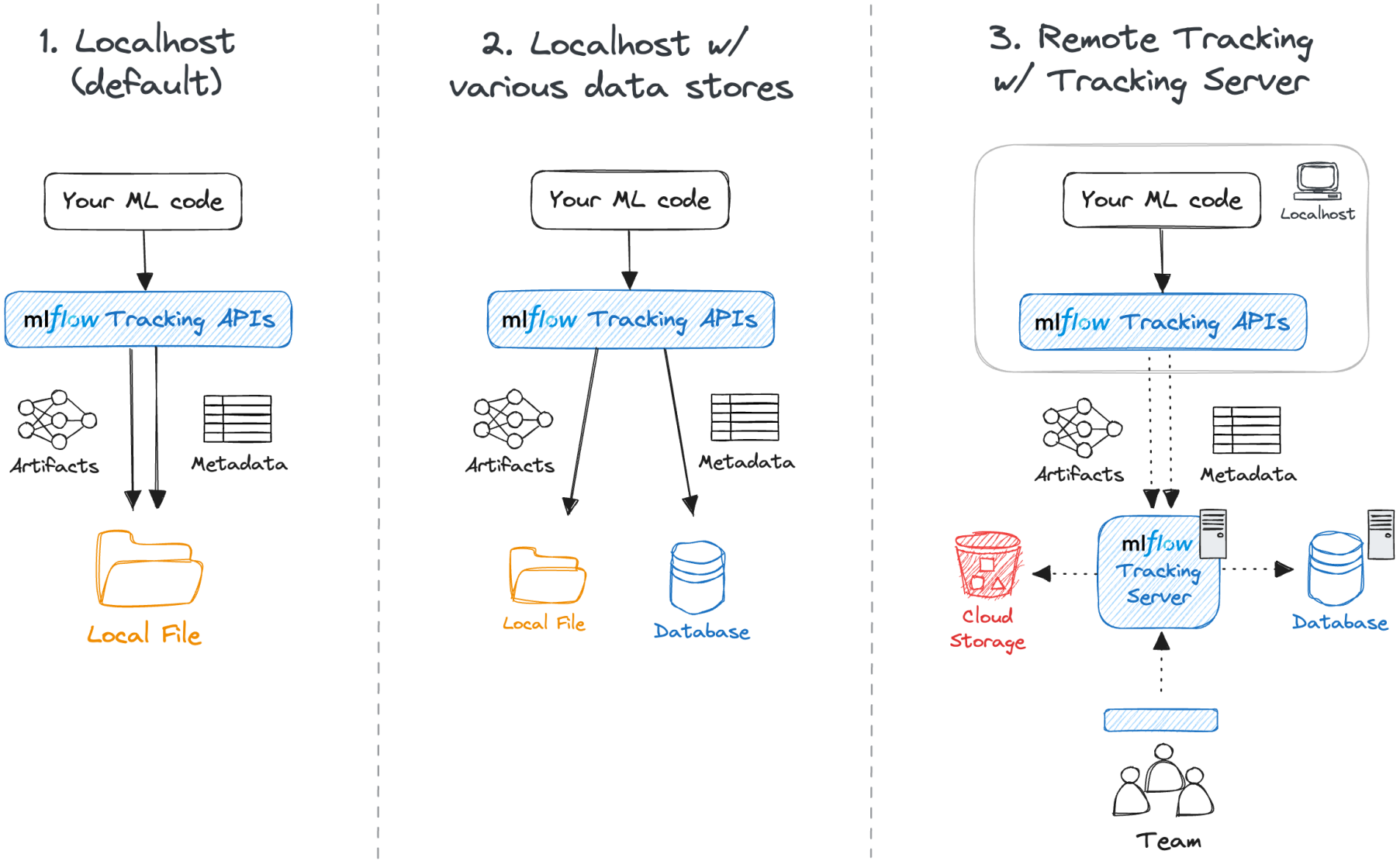
```
$ dvc params diff
Path          Param                    HEAD  workspace
params.yaml   featurize.max_features   100   200
params.yaml   featurize.ngrams         1     2
```

```
$ dvc metrics diff
Path                Metric          HEAD     workspace    Change
eval/metrics.json   avg_prec.test   0.9014   0.925        0.0236
eval/metrics.json   avg_prec.train  0.95704  0.97437      0.01733
eval/metrics.json   roc_auc.test    0.93196  0.94602      0.01406
eval/metrics.json   roc_auc.train   0.97743  0.98667      0.00924
```

MLflow


mlflow
MLflow Tracking
Record and query experiments: code, data, config, and results
Read more
MLflow Projects
Package data science code in a format to reproduce runs on any platform
Read more
MLflow Models
Deploy machine learning models in diverse serving environments
Read more
Model Registry
Store, annotate, discover, and manage models in a central repository
Read more

MLflow


1. Localhost (default)
Your ML code
mlflow Tracking APIs
Artifacts
Metadata
Local File
2. Localhost w/ various data stores
Your ML code
mlflow Tracking APIs
Artifacts
Metadata
Local File
Database
3. Remote Tracking w/ Tracking Server
Your ML code
Localhost
mlflow Tracking APIs
Artifacts
Metadata
mlflow Tracking Server
Cloud Storage
Database
Team

MLflow Tracking Runs

```python
import mlflow

with mlflow.start_run():
    mlflow.log_param("lr", 0.001)
    # Your ml code
    ...
    mlflow.log_metric("val_loss", val_loss)
```

```python
import mlflow

mlflow.autolog()

# Your training code...
```

MLflow Projects

```yaml
name: My Project

python_env: python_env.yaml
# or
# conda_env: my_env.yaml
# or
# docker_env:
#    image:  mlflow-docker-example

entry_points:
  main:
    parameters:
      data_file: path
      regularization: {type: float, default: 0.1}
    command: "python train.py -r {regularization} {data_file}"
  validate:
    parameters:
      data_file: path
    command: "python validate.py {data_file}"
```

MLflow Models

```
# Directory written by mlflow.sklearn.save_model(model, "my_model")
my_model/
├── MLmodel
├── model.pkl
├── conda.yaml
├── python_env.yaml
└── requirements.txt
```

- Python Function (`python_function`)
- R Function (`crate`)
- H_2O (`h2o`)
- Keras (`keras`)
- MLeap (`mleap`)
- PyTorch (`pytorch`)
- Scikit-learn (`sklearn`)
- Spark MLlib (`spark`)
- TensorFlow (`tensorflow`)
- ONNX (`onnx`)
- MXNet Gluon (`gluon`)
- XGBoost (`xgboost`)
- LightGBM (`lightgbm`)
- CatBoost (`catboost`)
- Spacy(`spaCy`)
- Fastai(`fastai`)
- Statsmodels (`statsmodels`)
- Prophet (`prophet`)
- Pmdarima (`pmdarima`)
- OpenAI (`openai`) (Experimental)
- LangChain (`langchain`) (Experimental)
- John Snow Labs (`johnsnowlabs`) (Experimental)
- Diviner (`diviner`)
- Transformers (`transformers`) (Experimental)
- SentenceTransformers (`sentence_transformers`) (Experimental)
- Promptflow (`promptflow`) (Experimental)

Mlflow Model Registry

mlflow 2.10.0 Experiments Models GitHub Docs

Registered Models

Create Model

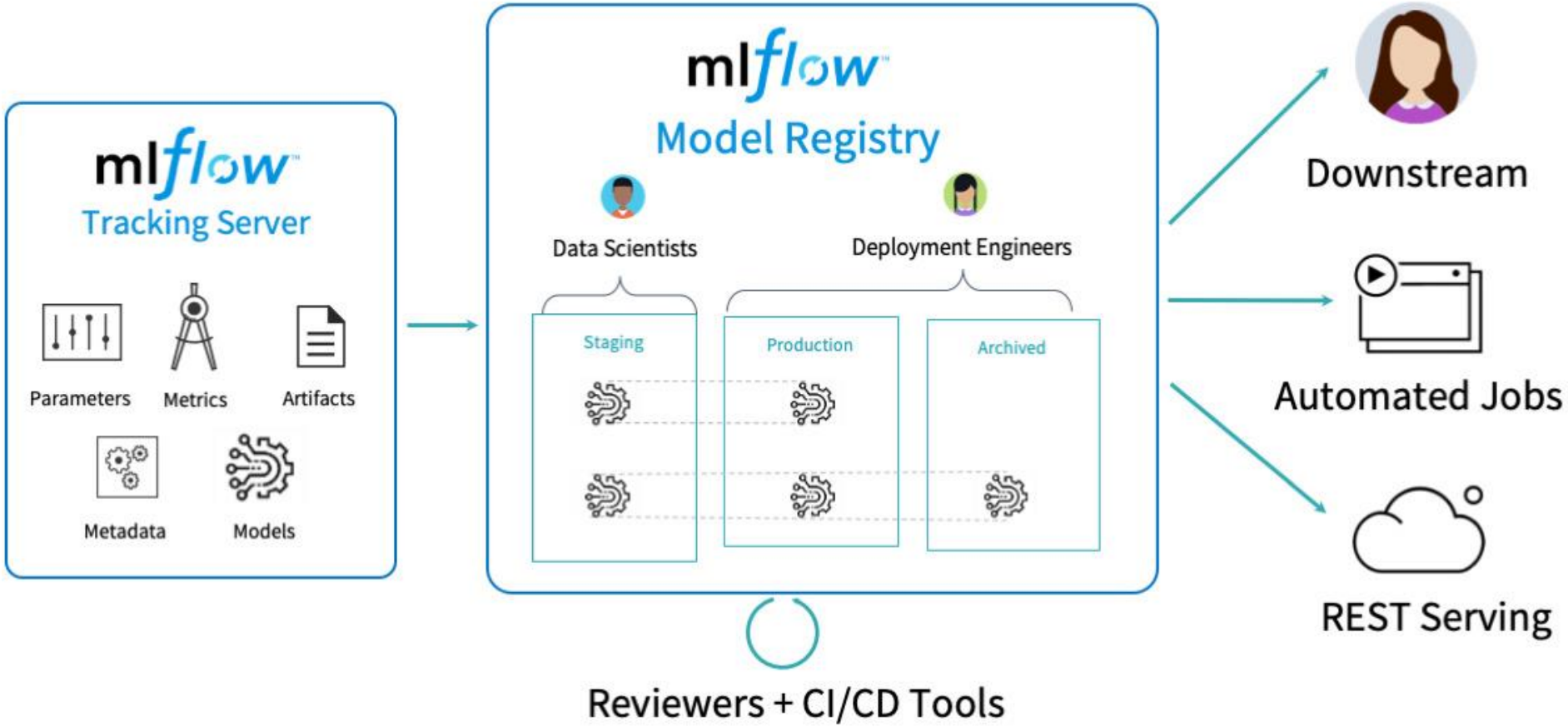
Filter registered models by name o...

Name	Latest version	Aliased versions	Created by	Last modified	Tags
iris_model_dev	Version 17			2023-09-25 12:50:...	—
iris_model_prod	Version 11	@ champion: Version 11 +3		2023-10-26 17:10:...	—
iris_model_staging	Version 11			2023-09-25 12:46:...	—
iris_model_testing	Version 1			2023-09-27 13:17:...	—
mnist_model_dev	Version 12			2023-09-25 12:39:...	—
mnist_model_prod	Version 8	@ challenger: Version 8 +1		2024-01-19 10:35:...	—
mnist_model_staging	Version 8			2023-09-25 12:51:...	—

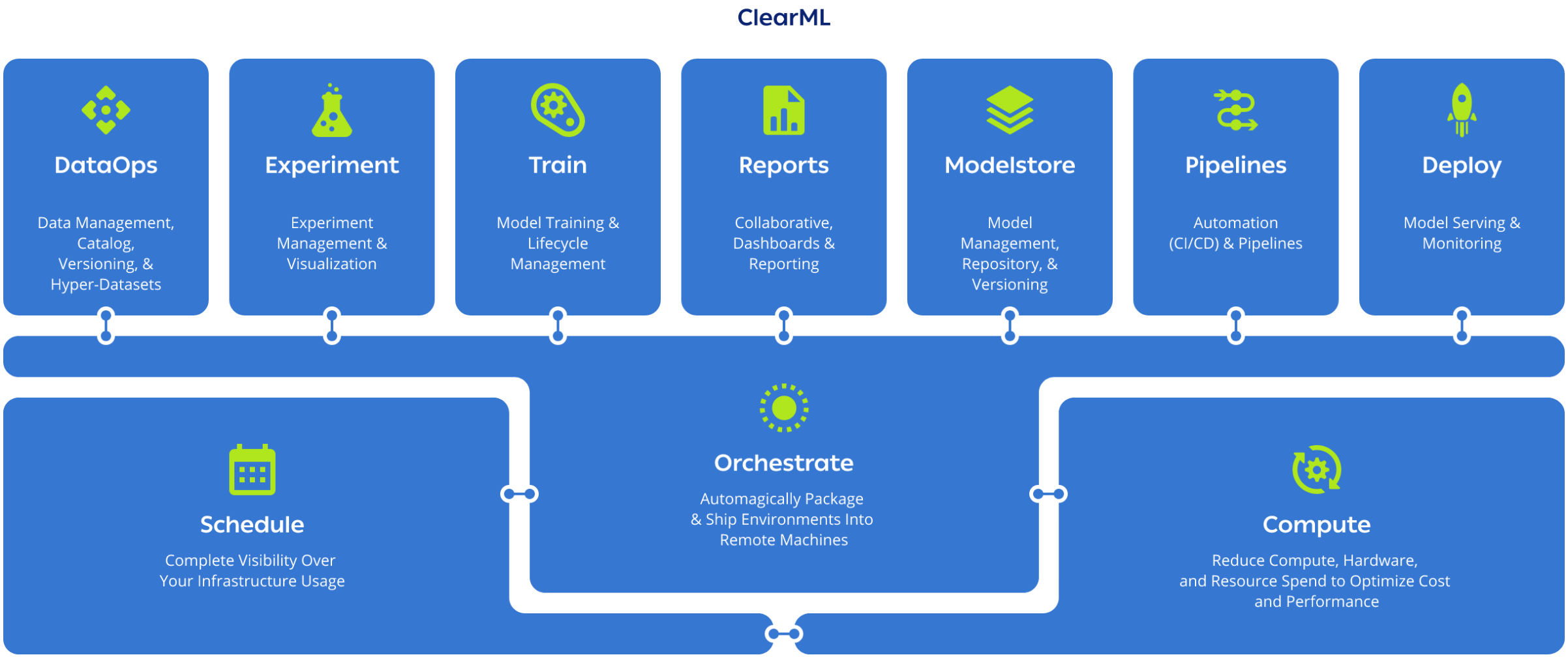
New model registry UI

Previous Next 25 / page

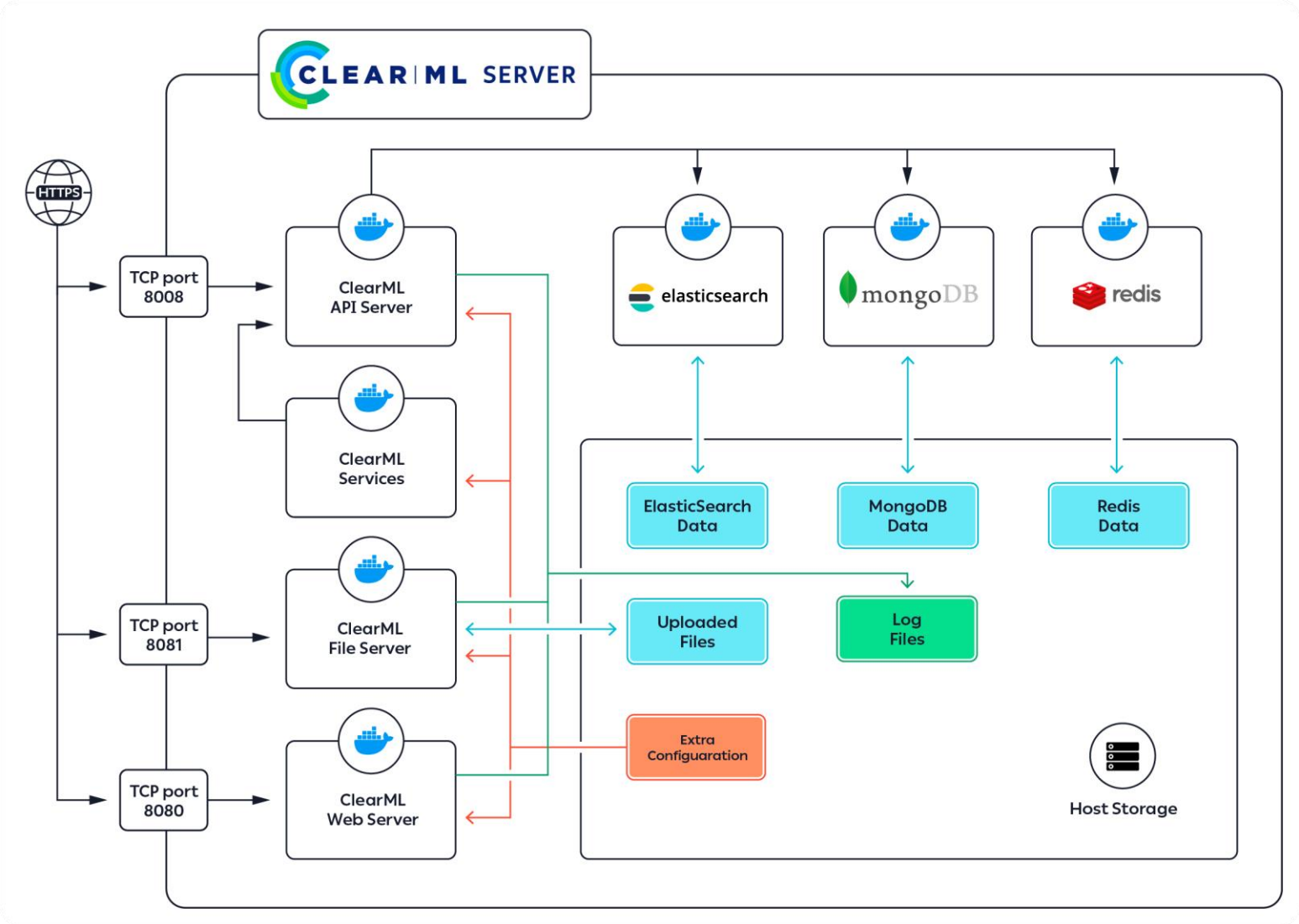
MLFlow


mlflow
Tracking Server
Parameters
Metrics
Artifacts
Metadata
Models
mlflow
Model Registry
Data Scientists
Deployment Engineers
Staging
Production
Archived
Downstream
Automated Jobs
REST Serving
Reviewers + CI/CD Tools

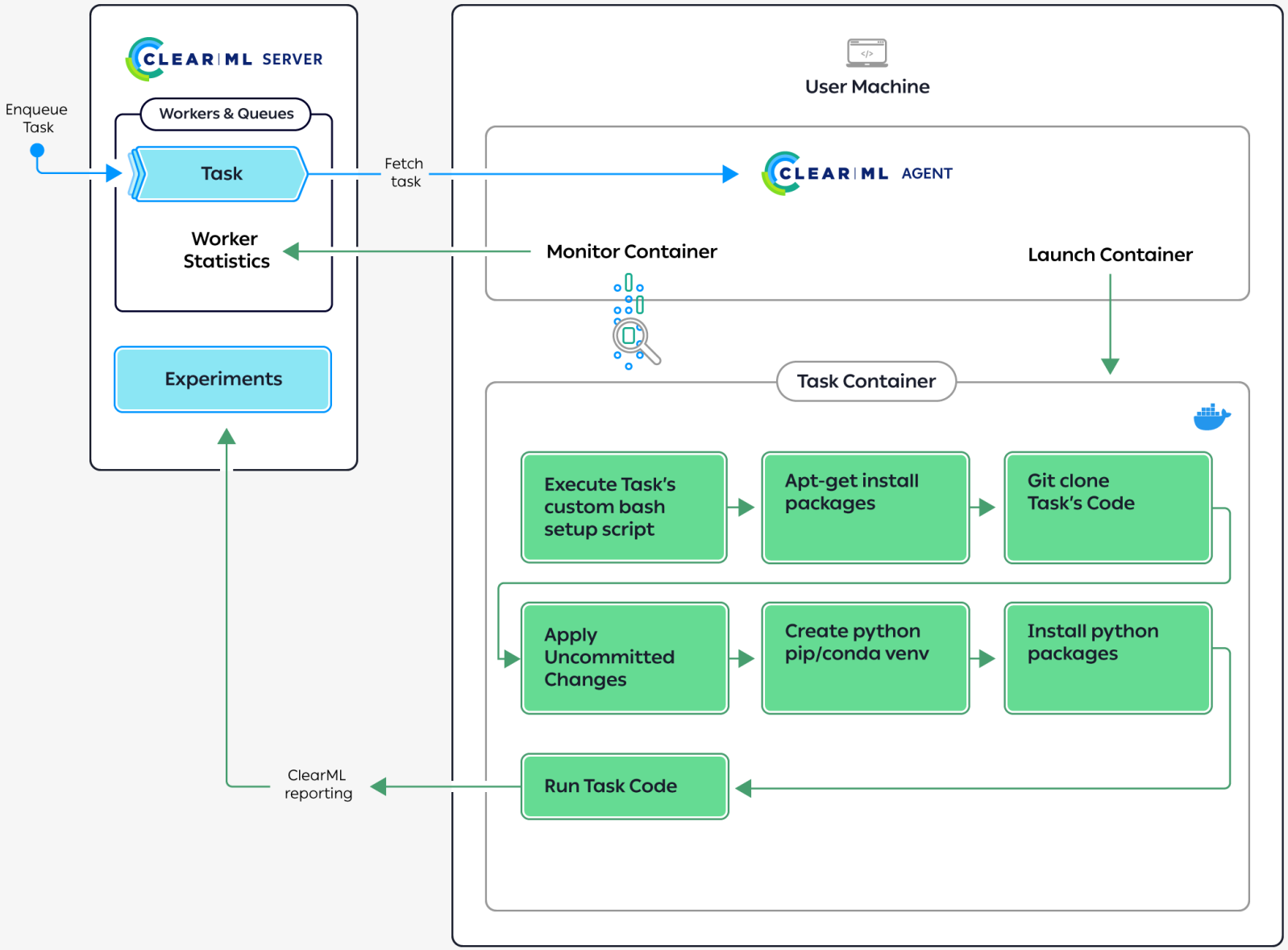
ClearML


ClearML
DataOps
Data Management, Catalog, Versioning, & Hyper-Datasets
Experiment
Experiment Management & Visualization
Train
Model Training & Lifecycle Management
Reports
Collaborative, Dashboards & Reporting
Modelstore
Model Management, Repository, & Versioning
Pipelines
Automation (CI/CD) & Pipelines
Deploy
Model Serving & Monitoring
Schedule
Complete Visibility Over Your Infrastructure Usage
Orchestrate
Automagically Package & Ship Environments Into Remote Machines
Compute
Reduce Compute, Hardware, and Resource Spend to Optimize Cost and Performance

ClearML


CLEAR|ML SERVER
HTTPS
TCP port 8008
TCP port 8081
TCP port 8080
ClearML API Server
ClearML Services
ClearML File Server
ClearML Web Server
elasticsearch
mongoDB
redis
ElasticSearch Data
MongoDB Data
Redis Data
Uploaded Files
Log Files
Extra Configuaration
Host Storage

ClearML Agent flow diagram

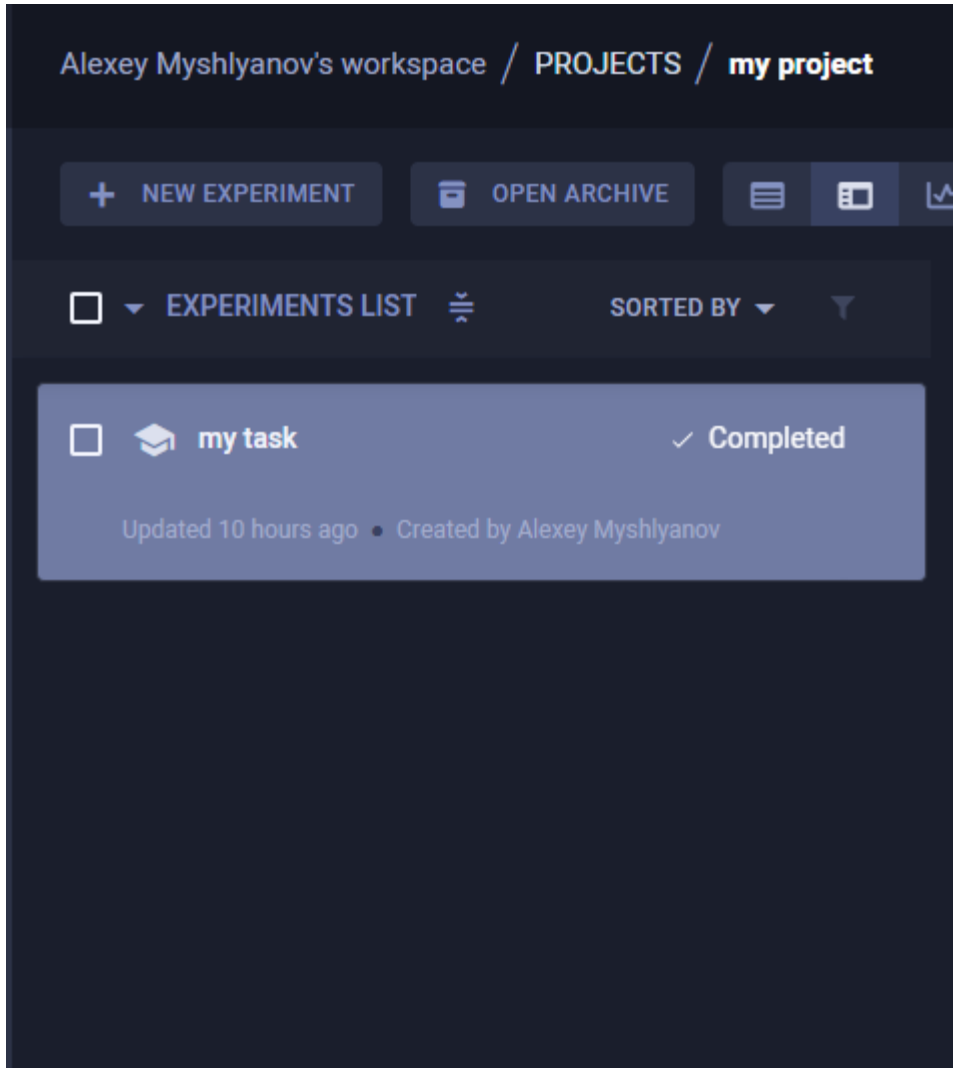


ClearML Tracking

```python
from clearml import Task


task = Task.init(
    project_name='example',    # project name of at least 3 characters
    task_name='task template', # task name of at least 3 characters
    task_type=None,
    tags=None,
    reuse_last_task_id=True,
    continue_last_task=False,
    output_uri=None,
    auto_connect_arg_parser=True,
    auto_connect_frameworks=True,
    auto_resource_monitoring=True,
    auto_connect_streams=True,
)
```

```python
Task.init(project_name='main_project/sub_project', task_name='test')
```



ClearML Tracking

- **Hyperparameters** - ClearML logs the following types of hyperparameters:
  - Command Line Parsing - ClearML captures any command line parameters passed when invoking code that uses standard python packages, including:
    - click
    - argparse
    - Python Fire
    - LightningCLI
  - TensorFlow Definitions (`absl-py`)
  - Hydra - ClearML logs the OmegaConf which holds all the configuration files, as well as values overridden during runtime.
- **Metrics, scalars, plots, debug images** reported through supported frameworks, including:
  - Matplotlib
  - Tensorboard
  - TensorboardX
- **Execution details** including:
  - Git information
  - Uncommitted code modifications - In cases where no git repository is detected (e.g. when a single python script is executed outside a git repository, or when running from a Jupyter Notebook), ClearML logs the contents of the executed script
  - Python environment
  - Execution configuration
- **Models** - ClearML automatically logs and updates the frameworks:
  - TensorFlow
  - Keras
  - PyTorch
  - AutoKeras
  - CatBoost
  - Fast.ai
  - LightGBM
  - MegEngine
  - MONAI
  - scikit-learn (only using joblib)
  - XGBoost (only using joblib)
  - YOLOv8
  - YOLOv5