Cloudera Data Platform Machine Learning (CML) Analysis

Guillaume Moutier

TL;DR...

Positioning Statement

Red Hat OpenShift Data Science is not just “an other” data science platform.

Many offerings, like Cloudera Machine Learning, are a set of data science tools more or less integrated together.

Much more than that, RHODS is an add-on to an enterprise-grade application development platform, bringing actionable data science and data engineering capabilities to businesses and organizations.

		CML	RHODS
Deployment options	Cloud	AWS, Azure (GCP advertised for Cloudera Data Platform, not documented for CML)	OSD (AWS) & ROSA (fully managed) RHODS Self-managed supported on OCP on multiple public clouds
	On-Prem	~Yes, with Cloudera Data Science Workbench, requires a Cloudera Hadoop cluster	RHODS Self-managed
Platform Technology		Underlying managed Kubernetes (EKS or AKS), but no direct access to it.	Managed OpenShift, fully accessible, with all OpenShift features (CI/CD, Serverless, Monitoring,...).
Features	Notebooks	Workbench (Cloudera tool), JupyterLab (basic images). Ability to import your custom runtimes.	JupyterLab with different flavors of pre-installed packages. Ability to import your custom images.
	Model Serving	Basic s2i capabilities to publish code.	Developer view (s2i), Serverless, Model Serving (upcoming).
	App Serving	Basic s2i capabilities to publish code.	Developer view (s2i), Serverless, CI/CD,...
	Jobs, Pipelines	Basic features	OpenShift Jobs, Tekton Pipelines, CI/CD,...

		CML	RHODS
Features (cont'd)	Data Visualization	Fully integrated query builder and dashboard editor.	Visualization tools like Superset must be installed independently.
	Data Governance	Apache Atlas provided and fully integrated.	Governance tools must be installed independently. Pachyderm integrated with RHODS as an ISV
Pricing (comparison only)	Costs	584 USD/month for a standard worker node (m5.2xlarge)	588 USD/month for standard worker node (m5.2xlarge), plus infra costs (master nodes)@1448 USD/month.
	Scheme	Hourly costs for "up" instances. Paid in-advance "Cloud credit unitus" or Pay-as-you-go monthly bill.	Subscriptions, yearly commitment. Hourly consumption-based pricing coming in Q4'22/Q1'23 (RHODS on AWS Marketplace)

Overview, components and features

CML = Cloudera Data Platform (CDP) Machine Learning:

"CDP Machine Learning enables enterprise data science teams to collaborate across the full data lifecycle with immediate access to enterprise data pipelines, scalable compute resources, and access to preferred tools. Streamline the process of getting analytic workloads into production and intelligently manage machine learning use cases across the business at scale."

Deployment options:

- CDP Public Cloud/CDP Machine learning*: cloud-based offering, available on AWS and Azure.
- CDP Private Cloud: this is in fact Cloudera Data Science Workbench. It provides almost the same features as the Public Cloud Service, but requires a Cloudera Distributed Hadoop (CDH) cluster to host the workloads.

This analysis focuses on CDP Public Cloud, which is generally what people refer to as CML.

(*) the naming of the different platforms/offerings/solutions is clearly inconsistent between the marketing materials, the website, the documentation, the examples...

CML Components:

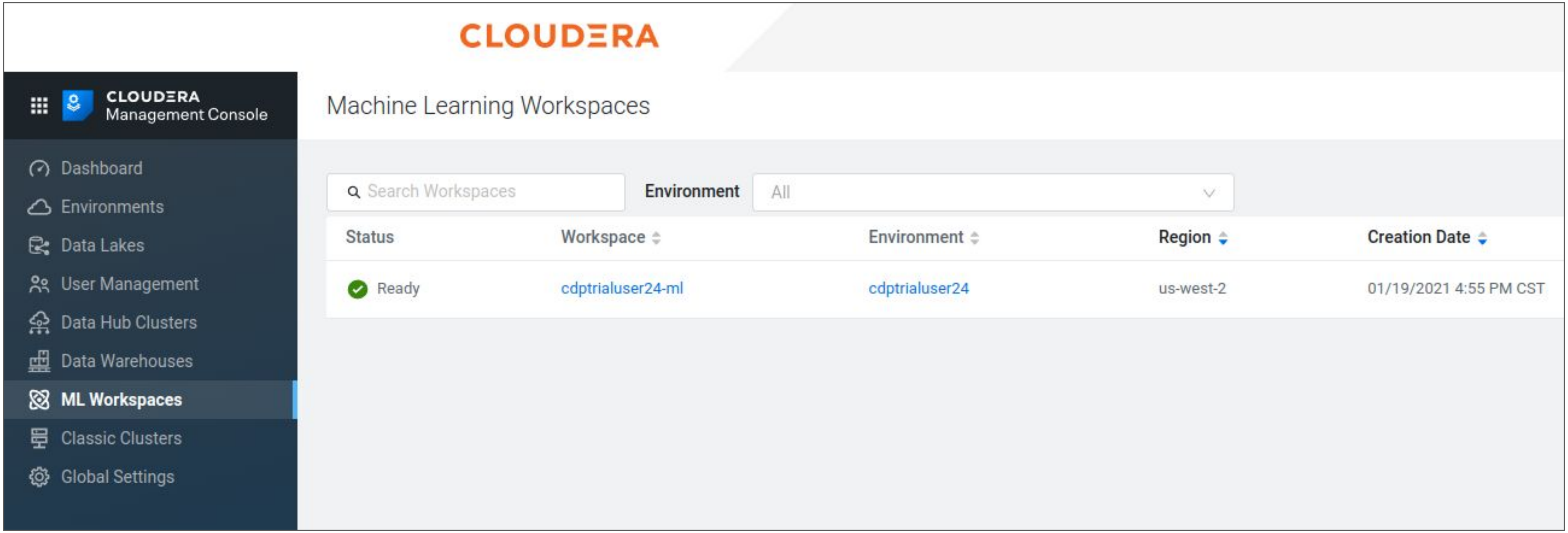
- CDP Control Plane:

  The main console (Cloudera Management Console) is used to create different “workspaces”. This is where you provide an AWS/Azure account, with the permission to create different elements: compute resources, DNS entries, storage,…

  This is also where you configure your Identity provider for SSO. When deployed, CML will create a FreeIPA server to “relay” the authentication requests.

  Given the requirements are met (example for AWS), CDP handles all the deployment process for a workspace.

  However both for AWS and Azure, administrators have many configuration steps to do to create subnets, permissions, storage,…



Machine Learning Workspaces

Environment: All

Status	Workspace	Environment	Region	Creation Date
✓ Ready	cdptrialuser24-ml	cdptrialuser24	us-west-2	01/19/2021 4:55 PM CST

The Cloudera Management Console

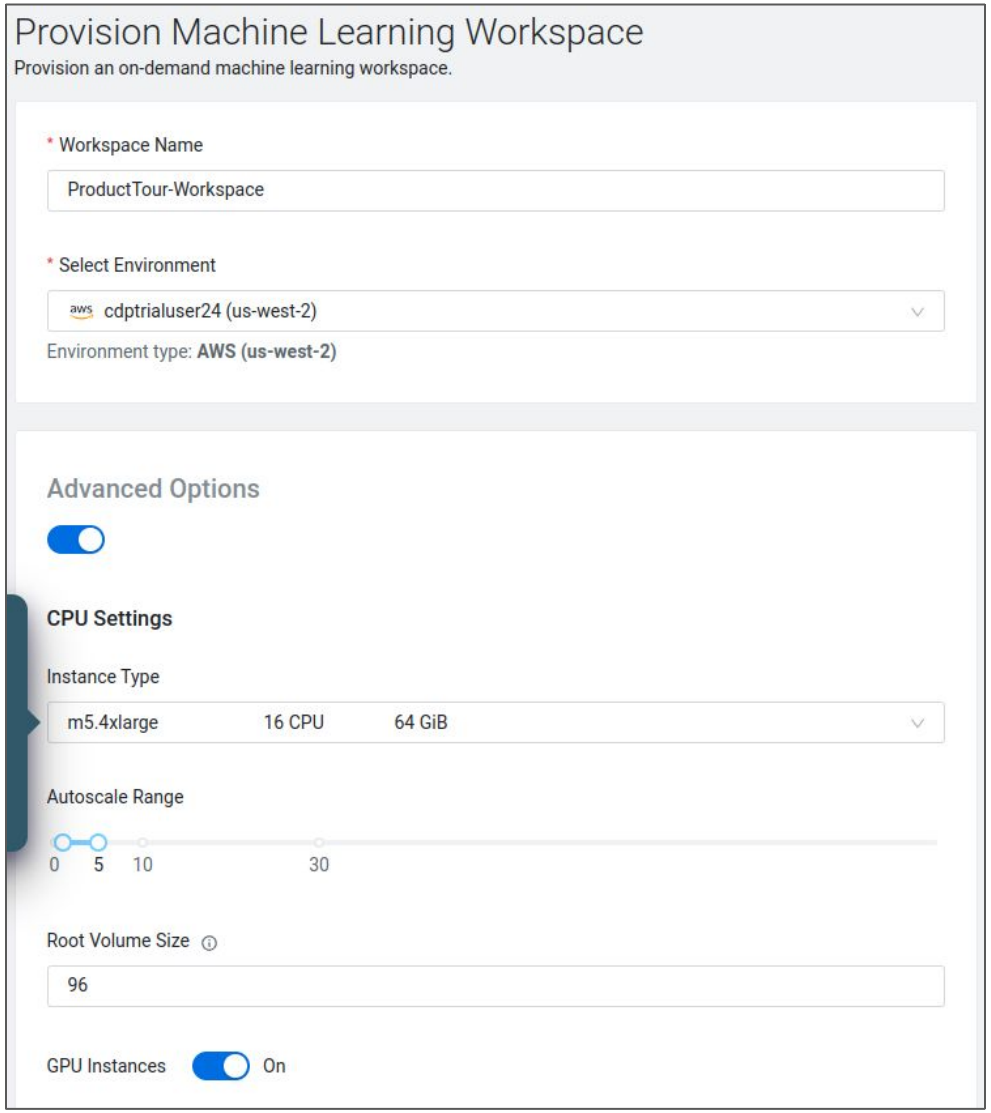
CML Components:

- Workspace:

  CDP creates a workspace by provisioning a new managed Kubernetes Cluster, installing CML into it, and provisioning the storage used to hold the projects and users' data (EFS on AWS, NFS Service on Azure).

  However, you have to provide many different information to properly create the workspace: instance types, autoscale range, Kubernetes configuration, Network settings (subnets, load balancers,...),....

From an architecture point of view, a workspace is then the equivalent of a RHODS deployment.



Provision Machine Learning Workspace

Provision an on-demand machine learning workspace.

* Workspace Name

ProductTour-Workspace

* Select Environment

aws cdptrialuser24 (us-west-2)

Environment type: **AWS (us-west-2)**

Advanced Options

CPU Settings

Instance Type

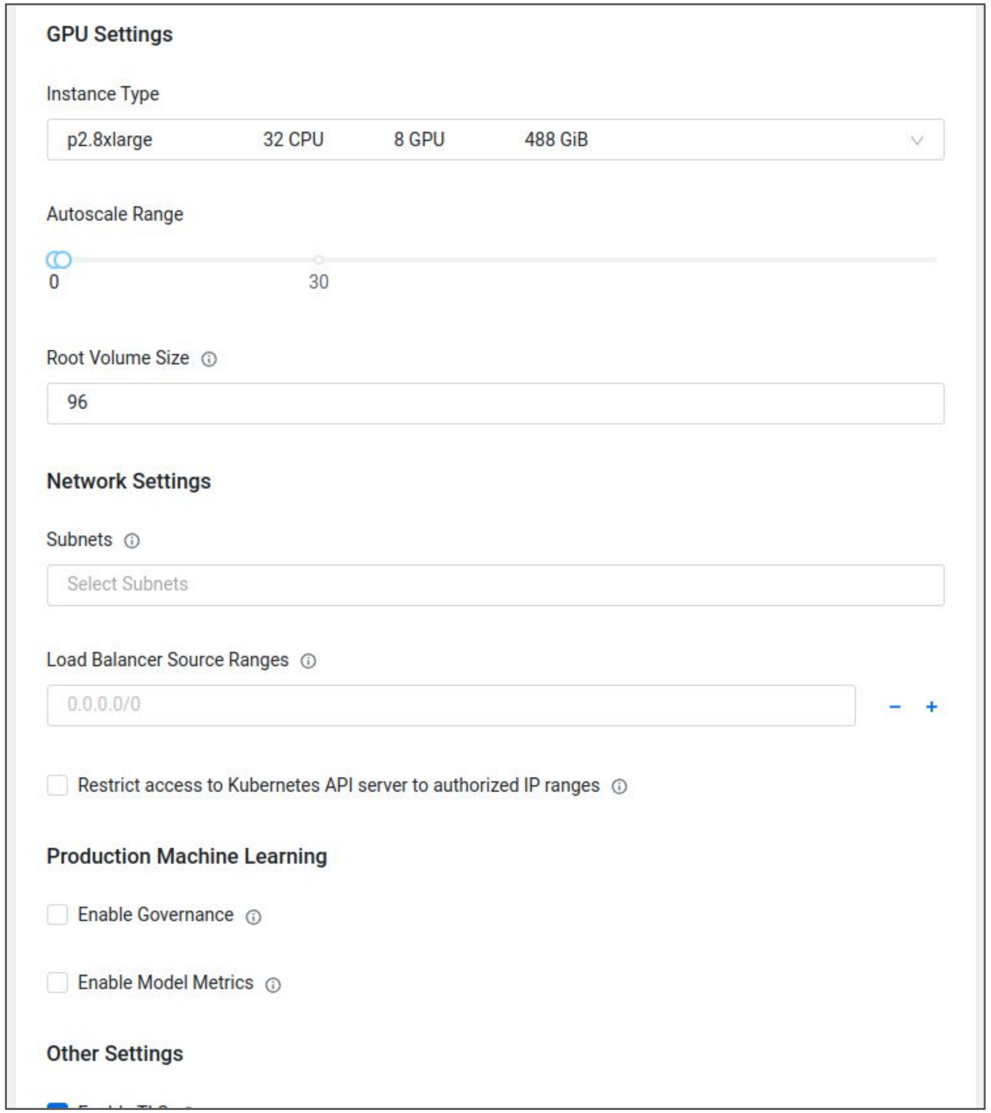
m5.4xlarge 16 CPU 64 GiB

Autoscale Range

Root Volume Size

96

GPU Instances On



GPU Settings

Instance Type

p2.8xlarge 32 CPU 8 GPU 488 GiB

Autoscale Range

Root Volume Size

96

Network Settings

Subnets

Select Subnets

Load Balancer Source Ranges

0.0.0.0/0

□ Restrict access to Kubernetes API server to authorized IP ranges

Production Machine Learning

□ Enable Governance

□ Enable Model Metrics

Other Settings

Creating a new Workspace

CML Components:

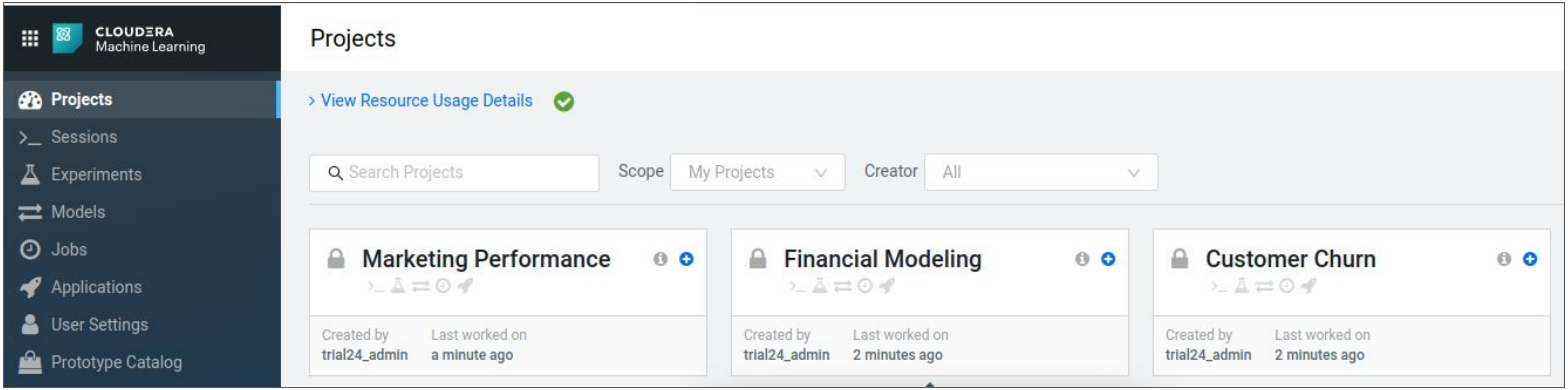
- Project:

  A workspace contains one or more projects. Each project can have its own visibility, access levels (viewer, operator, contributor administrator).

  A project provides access to different resources: sessions (notebook/editor), jobs, experiments, models, applications, files,...

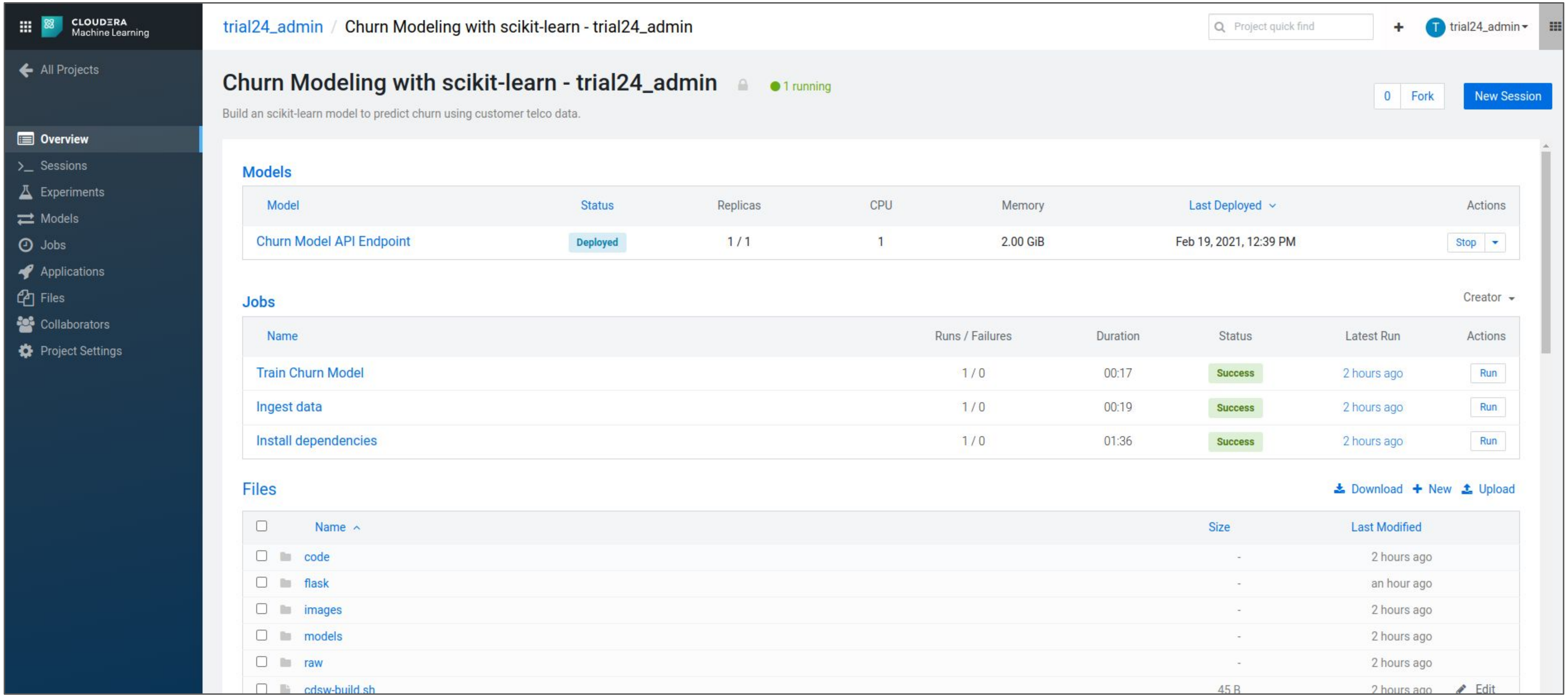
  Within a project, all the code and data is shared among users (with the right permissions). Meaning anyone in the project can access a running session (like Workbench, the internal Cloudera editor, or Jupyter notebook).

So the project is somewhat similar to a namespace, but with a UI providing direct access to all types of resources available.



Project	Created by	Last worked on
Marketing Performance	trial24_admin	a minute ago
Financial Modeling	trial24_admin	2 minutes ago
Customer Churn	trial24_admin	2 minutes ago

Projects view



trial24_admin / Churn Modeling with scikit-learn - trial24_admin

Churn Modeling with scikit-learn - trial24_admin ● 1 running

Build an scikit-learn model to predict churn using customer telco data.

Models

Model	Status	Replicas	CPU	Memory	Last Deployed	Actions
Churn Model API Endpoint	Deployed	1 / 1	1	2.00 GiB	Feb 19, 2021, 12:39 PM	Stop

Jobs

Name	Runs / Failures	Duration	Status	Latest Run	Actions
Train Churn Model	1 / 0	00:17	Success	2 hours ago	Run
Ingest data	1 / 0	00:19	Success	2 hours ago	Run
Install dependencies	1 / 0	01:36	Success	2 hours ago	Run

Files

□	Name	Size	Last Modified	
□	code	-	2 hours ago	
□	flask	-	an hour ago	
□	images	-	2 hours ago	
□	models	-	2 hours ago	
□	raw	-	2 hours ago	
□	cdsw-build.sh	45 B	2 hours ago	Edit

Overview of a project

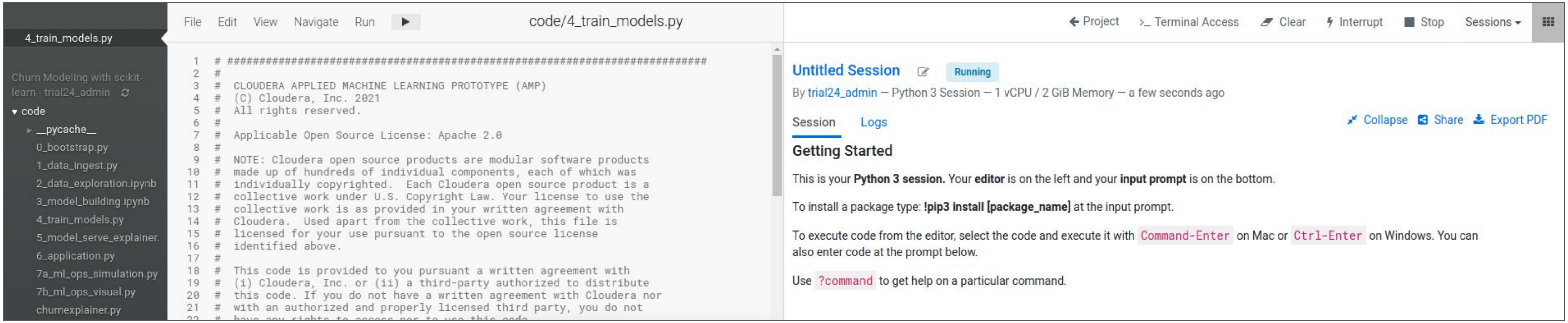
CML Components:

- Session:

  A session is an isolated Pod in Kubernetes in which you can execute code. The pod runs in a per-user namespace.

  Different “runtimes” are provided to launch a session, covering different languages, packages and capabilities (see later).

A session is similar to a Notebook launched from the RHODs Dashboard. The main differences are that for CML the project data is available directly to the session as it’s on a shared volume accessible by everything in the project, and that an opened session is accessible by any member of the project.



```
# ###########################################################################
#
#  CLOUDERA APPLIED MACHINE LEARNING PROTOTYPE (AMP)
#  (C) Cloudera, Inc. 2021
#  All rights reserved.
#
#  Applicable Open Source License: Apache 2.0
#
#  NOTE: Cloudera open source products are modular software products
#  made up of hundreds of individual components, each of which was
#  individually copyrighted.  Each Cloudera open source product is a
#  collective work under U.S. Copyright Law. Your license to use the
#  collective work is as provided in your written agreement with
#  Cloudera.  Used apart from the collective work, this file is
#  licensed for your use pursuant to the open source license
#  identified above.
#
#  This code is provided to you pursuant a written agreement with
#  (i) Cloudera, Inc. or (ii) a third-party authorized to distribute
#  this code. If you do not have a written agreement with Cloudera nor
#  with an authorized and properly licensed third party, you do not
#  have any rights to access nor to use this code.
```

Untitled Session

By trial24_admin — Python 3 Session — 1 vCPU / 2 GiB Memory — a few seconds ago

Getting Started

This is your **Python 3 session.** Your **editor** is on the left and your **input prompt** is on the bottom.

To install a package type: **!pip3 install [package_name]** at the input prompt.

To execute code from the editor, select the code and execute it with `Command-Enter` on Mac or `Ctrl-Enter` on Windows. You can also enter code at the prompt below.

Use `?command` to get help on a particular command.

Session view with the default Workbench editor (can also be Jupyter and other IDEs)

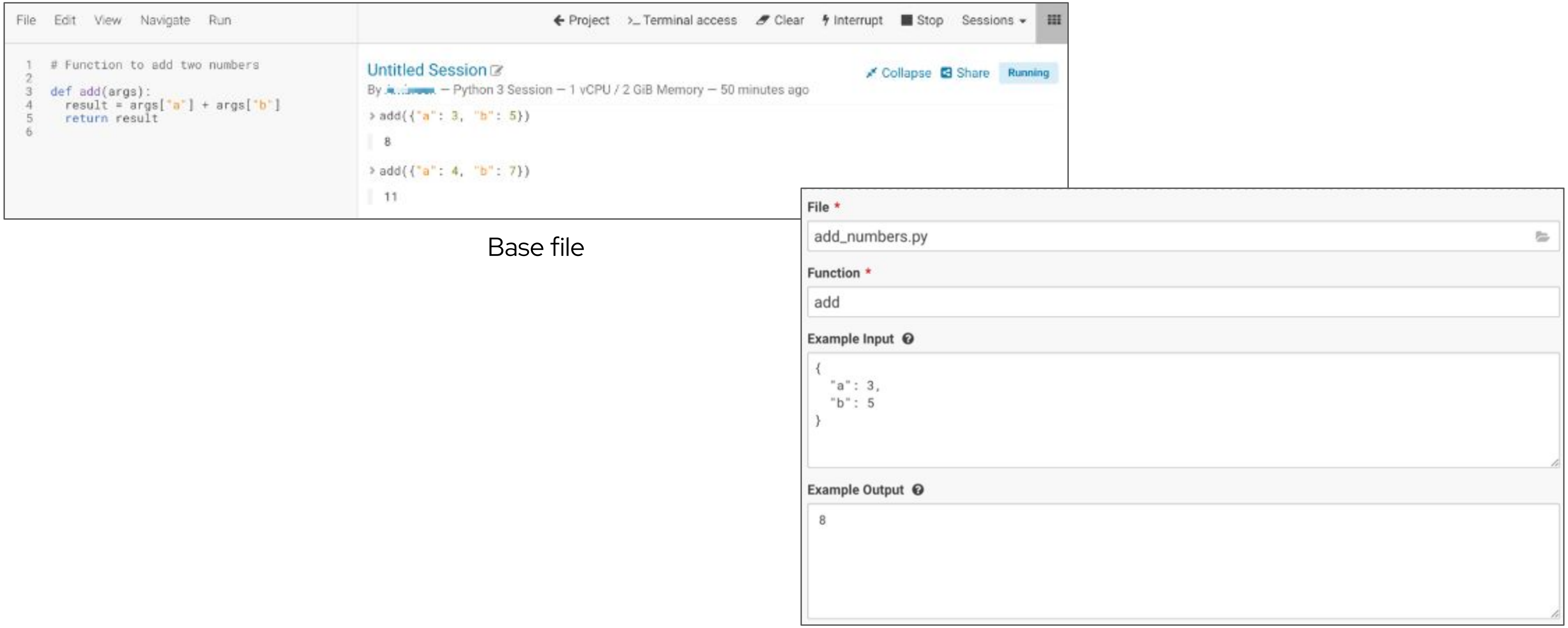
CML Components:

- Models:

  A model is a Python code or application that has been packaged API. The underlying packaging process is based on S2I.

  To create a model, you provide the source Python file in the project (therefore a model can only be created in the context of the project), the name of the entry function, and a JSON example input and output. The model is then automatically deployed.

The process is really similar to S2I or the Application creation process from the Developer view in OpenShift, but with really minimal configuration. It is applied directly from within a project, vs fetching source from a Git repository.



```
# Function to add two numbers

def add(args):
  result = args["a"] + args["b"]
  return result
```

Untitled Session

By [redacted] — Python 3 Session — 1 vCPU / 2 GiB Memory — 50 minutes ago

```
> add({"a": 3, "b": 5})
8
> add({"a": 4, "b": 7})
11
```

Base file

File *

add_numbers.py

Function *

add

Example Input

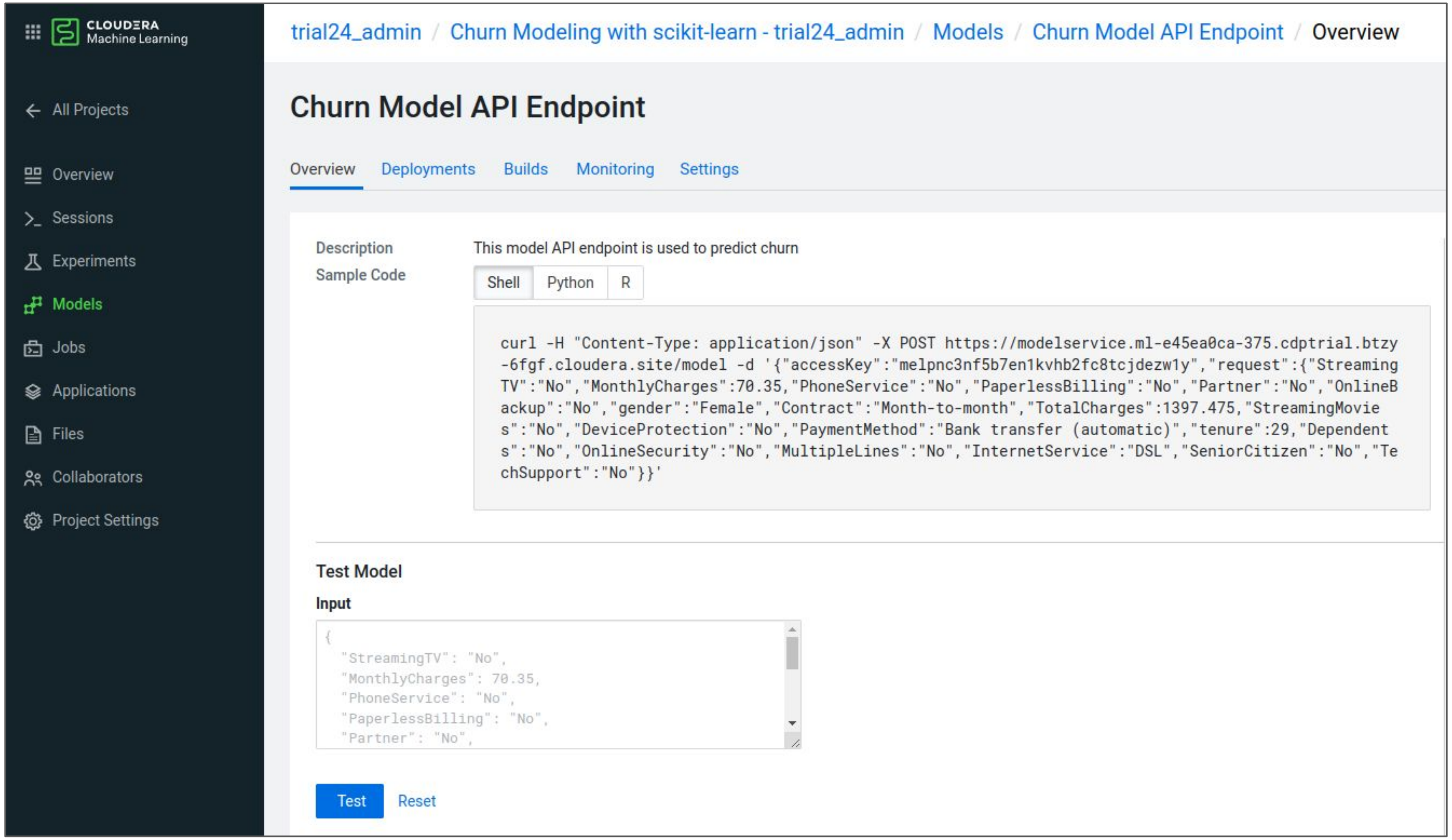
```
{
  "a": 3,
  "b": 5
}
```

Example Output

```
8
```

Model deployment

Model creation



Churn Model API Endpoint

Description: This model API endpoint is used to predict churn

Sample Code:

```
curl -H "Content-Type: application/json" -X POST https://modelservice.ml-e45ea0ca-375.cdptrial.btzy
-6fgf.cloudera.site/model -d '{"accessKey":"melpnc3nf5b7en1kvhb2fc8tcjdezw1y","request":{"Streaming
TV":"No","MonthlyCharges":70.35,"PhoneService":"No","PaperlessBilling":"No","Partner":"No","OnlineB
ackup":"No","gender":"Female","Contract":"Month-to-month","TotalCharges":1397.475,"StreamingMovie
s":"No","DeviceProtection":"No","PaymentMethod":"Bank transfer (automatic)","tenure":29,"Dependent
s":"No","OnlineSecurity":"No","MultipleLines":"No","InternetService":"DSL","SeniorCitizen":"No","Te
chSupport":"No"}}'
```

Test Model

Input

```
{
  "StreamingTV": "No",
  "MonthlyCharges": 70.35,
  "PhoneService": "No",
  "PaperlessBilling": "No",
  "Partner": "No",
```

Deployed model with REST API

CML Components:

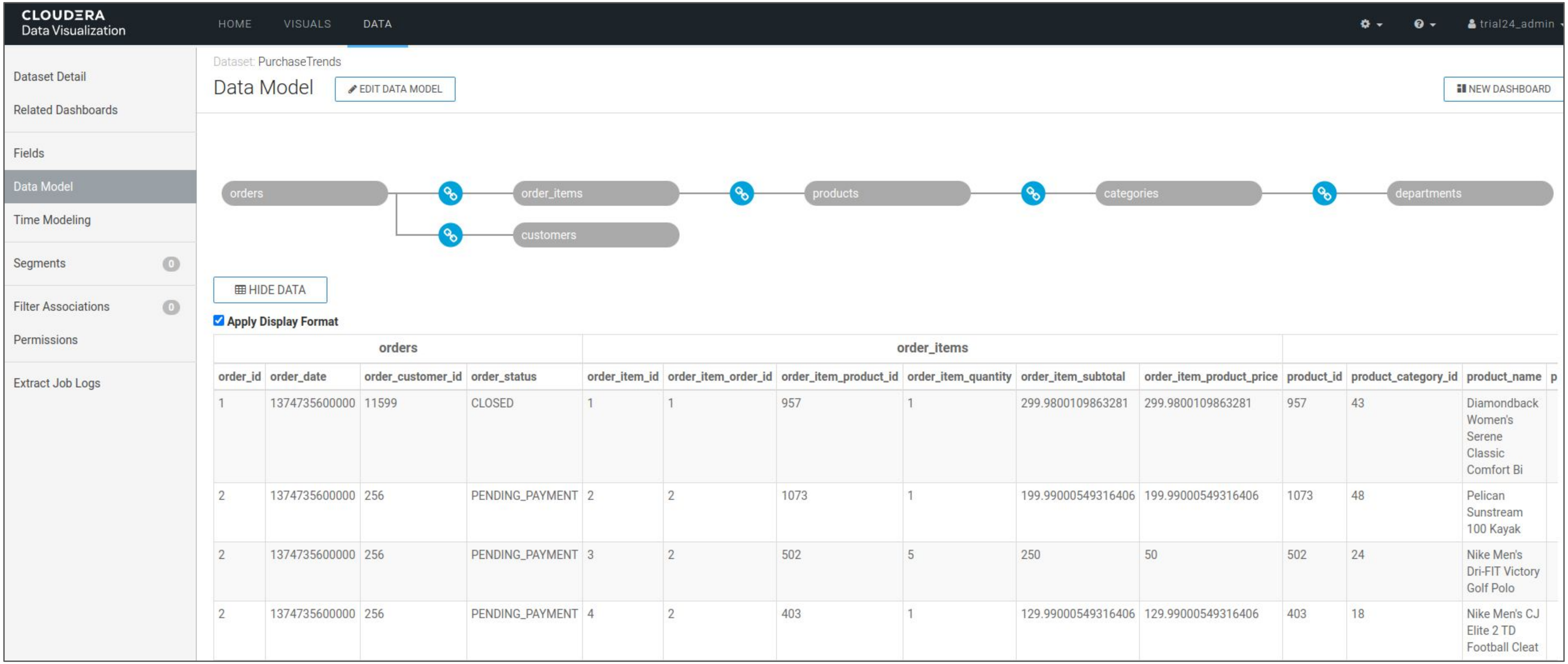
- Applications:

  A mechanism similar to the ones for the models can be used to deploy Applications. Again it’s an S2I process applied to a file to serve it as an application, using Flask or another lightweight framework.

CML Components:

- Data Visualization:

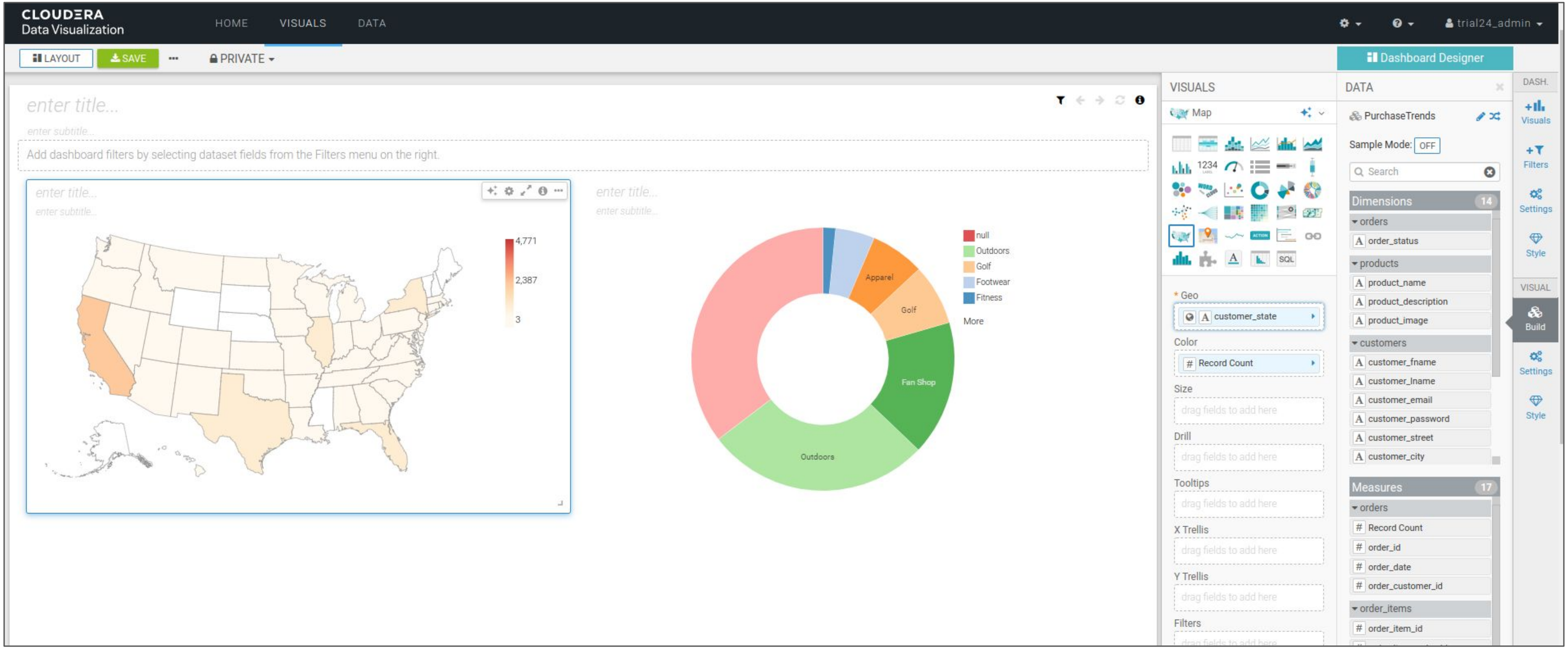
  Tool to connect to data, create datasets through queries, and create visualizations and dashboards.



☑ Apply Display Format

orders				order_items										
order_id	order_date	order_customer_id	order_status	order_item_id	order_item_order_id	order_item_product_id	order_item_quantity	order_item_subtotal	order_item_product_price	product_id	product_category_id	product_name	p
1	1374735600000	11599	CLOSED	1	1	957	1	299.9800109863281	299.9800109863281	957	43	Diamondback Women's Serene Classic Comfort Bi	
2	1374735600000	256	PENDING_PAYMENT	2	2	1073	1	199.99000549316406	199.99000549316406	1073	48	Pelican Sunstream 100 Kayak	
2	1374735600000	256	PENDING_PAYMENT	3	2	502	5	250	50	502	24	Nike Men's Dri-FIT Victory Golf Polo	
2	1374735600000	256	PENDING_PAYMENT	4	2	403	1	129.99000549316406	129.99000549316406	403	18	Nike Men's CJ Elite 2 TD Football Cleat	

Data Visualization – Creating datasets



Data Visualization – Creating dashboards

Analysis and comparison with RHODS

- CML is a fully integrated environment for data exploration, model training, model and application serving, data visualization…
- The infrastructure layer (Public Cloud), is identical to RHODS. However the platform implementation is different:
    - CML deploys one or many different Kubernetes clusters (one per workspace).
    - CML does not provide a direct access to the underlying Kubernetes cluster. It can of course be done because the Kubernetes cluster runs under the customer's account, but it's still under the control of CDP. When a workspace is destroyed, the Kubernetes cluster is also removed.
    - So CML is not an "extension" to a Kubernetes platform, as RHODS is an add-on to Managed OpenShift. CML hides all the benefits of the integration with an Enterprise-Level solution. Although you can "publish" applications and models, the mechanism is clearly meant for development/PoCs, not for production workloads that you would have to implement by yourself elsewhere.

- The exploration and model training part is similar to RHODS, with on-demand environments, with the following differences:
  - Sessions in CML are not exclusive to a specific user, nor do they have access to a personal data store. Everything is handled at the project level, based on a shared environment (a shared drive accessible through the project). This can be useful from a sharing perspective (same can be achieved in RHODS with RWX volumes), but dangerous or hard to maintain from a security perspective (necessity to create groups, teams, access rights,...).
  - Model serving is rudimentary, it’s just publishing a Python file you write yourself.

…/…

…/…

- ○ The provided runtimes in CML are pretty poor. JupyterLab runtimes come with a set of libraries equivalent to the minimal image in ODH/RHODS. The bare minimum to run a notebook, but not even common ones like Scipy, Numpy,… Users are expected to install packages themselves, on a per project basis. Installed packages are persisted in the project, and therefore available for all runtime. The approach is interesting, but also means that in the same project you cannot use different versions of a library.
- ○ You can create your own custom images, but it's a manual process (there is no automation like the one Red Hat's Thoth's team is preparing).

- The data visualization part is completely integrated within the tool, similar to Superset for some aspects, but with enhanced capabilities for dataset creations and manipulations.
- The is a data governance/lineage module based on Apache Atlas that is integrated in CML. All metadata related to new data being brought in or modified, models being published,… is fed to Atlas to provide this governance overview.
- Spark is available through the standard Kubernetes support built into Spark. You simply submit your Spark job to Kubernetes and the required containers are created automatically. Only Spark 2.4.7 is supported, and Spark 3.1.1 is “available” (not marked as supported) through an add-on. All integration, access,… has to be done manually, similar as launching a Spark job from your laptop with a Kubernetes cluster as a target.
- Jobs and Pipelines are available. Jobs are similar to the basic Kubernetes ones, with a UI and and an API to create them. Pipelines are rudimentary chained jobs by indicating dependencies.

Pricing

Preliminary notes:

- Pricing comparison made on AWS infrastructure, with ROSA.
- Direct pricing comparison is somewhat difficult and biased as RHODS includes/requires ROSA subscriptions whereas CML runs on barebone Kubernetes (without even giving access).
- CML pricing calculator is not really clear, with nothing said about master nodes, does not include storage,…
- CML includes autoscaling, allowing infra going down to zero worker nodes.
- CML is based on an hourly price, monthly-billed, whereas RHODS is subscription based for one year (although you can scale the worker nodes and not pay for the pure infra part).

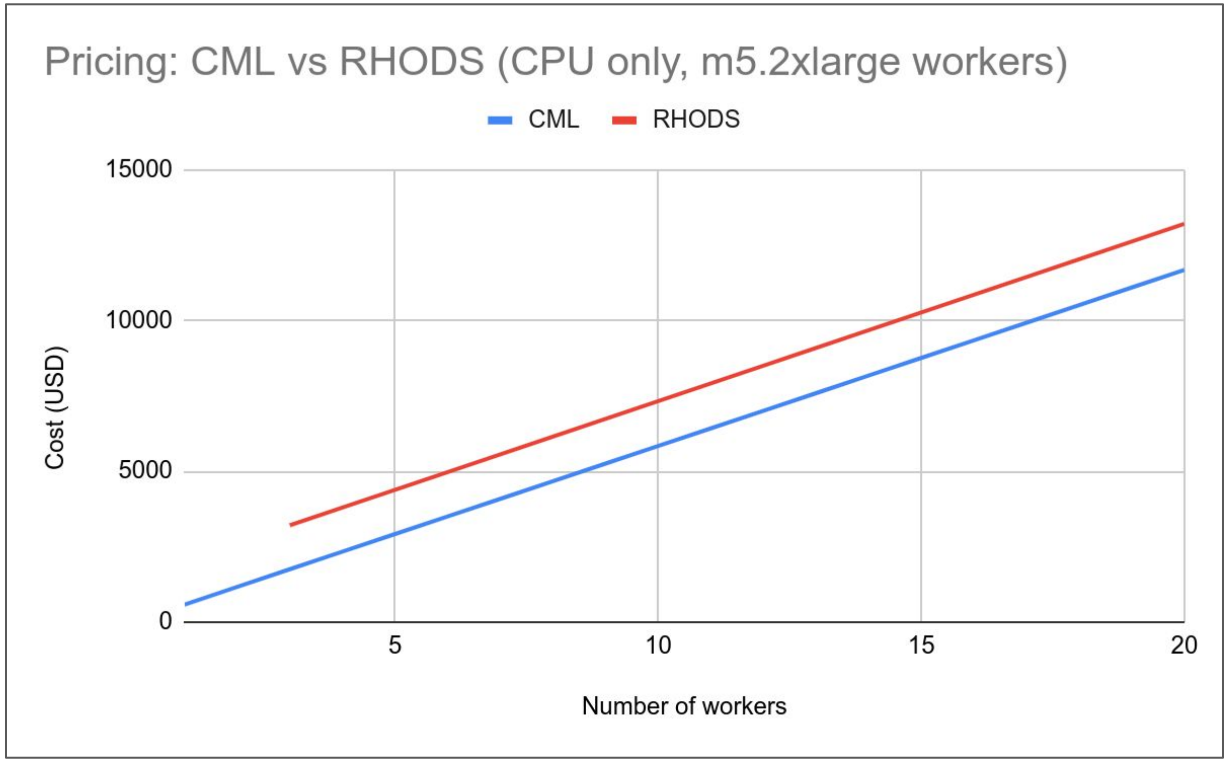
Basic pricing comparison:

- Based on "additional" worker node pricing, to remove masters/infra considerations.
- Based on one full month (730 hours) consumption, to remove auto scaling considerations.
- m5.2xlarge instances are used in both cases (base instances in RHODS pricing calculator).
- Storage costs are removed from RHODS price calculator as CML does not list them. They should be equivalent though.

	CML	RHODS
CPU - Per one m5.2xlarge instance, monthly	584 USD	588 USD
GPU - Per one m5.2xlarge instance, monthly		TBD

Pricing comparison depending on cluster size:

- Same assumptions as before, but with the total cost of infrastructure for RHODS.
- As in both cases the prices are per instance and licence costs are based on total vCPU, the evolution is linear and parallel.
- The only difference is in the “entry costs” for RHODS for the initial master nodes (1448 USD/month) that are required for OpenShift.



Conclusion

- Apart from the data governance with Atlas, and the Data visualization part, CML is really similar to RHODS in its intent: providing an easy to use environment for data scientists.
- There is nothing you can do in CML that you could not do on RHODS (you can still install and use Atlas alongside RHODS).
- The underlying components (infrastructure, platform, packages,...) are similar, but implemented in different ways:
  - RHODS is an extension to OpenShift, building on its enterprise-level components and implementations to bring data science tools into the mix. Therefore it benefits not only from those tools, but also from the whole OpenShift ecosystem for the whole lifecycle of intelligent applications.
  - CML is a set of well integrated data science tools that happens to use Kubernetes as an engine. Which is not a surprise as it's mostly a reimplementation of Cloudera Data Science Workbench, using Kubernetes as the processing platform instead of Cloudera hadoop cluster.

- Many features of CML (Models, Applications, Jobs, Pipelines,...) are simple scripts with a UI on top. All the underlying mechanisms are already part of OpenShift, and easy to use with the Developer UI.
- Providing equivalent features in RHODS through the Dashboard is minimal work, basically creating the UI and calling the OpenShift API to create the necessary objects (Deployments, Services, Routes,...).
- But moreover, Model Serving, Applications and Pipelines in CML should not even be called that. It's light years away from things we already have like Tekton Pipelines or Application Deployment/CI/CD, or what we are coming up with like data science model serving (aka Model Mesh Serving).
- RHODS provides ready-to-use curated and supported images for the most common packages (Scikit, Tensorflow, PyTorch,...), whereas CML does not provide anything further than the base IDE (their Workbench or JupyterLab). This is a clear differentiation for easy onboarding.

- CML does not provide any examples, walkthrough, tutorials on how to use the platform, neither in its documentation nor directly from the environment as we do.
- CML has an advantage over RHODS in the workspace/projects aspect to clearly separate workloads and share elements. But work is already in progress to bring similar features into RHODS.

To sum it up, CML and RHODS are really similar in terms of “raw” features, with CML being more integrated, but RHODS catching up on that.

However, CML is a cloud evolution/offering of a tool initially made for data scientists focusing on the data analysis/modeling part, whereas RHODS is an addition to an enterprise-grade cloud-native application platform, bringing actionable data science capabilities to businesses and organizations.

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.

- linkedin.com/company/red-hat
- youtube.com/user/RedHatVideos
- facebook.com/redhatinc
- twitter.com/RedHat