

# AWS Immersion - Hands-On-Labs

## Student Guide

Text Summarization with CML and Amazon Bedrock - Hands On Lab

### Objective:

In this exercise we will implement an LLM application using Cloudera Machine Learning and Amazon Bedrock, including:

<b>Lab 1: CML Walkthrough - Tour (CML Tour)</b>	<b>2</b>
<b>Lab 2: Text Summarization AMP Walkthrough</b>	<b>7</b>
<b>Lab 3: Manually Create Project via Git</b>	<b>9</b>
<b>Lab 4: Test Amazon Bedrock API using Jupyter Notebook</b>	<b>12</b>
<b>Lab 5: Deploy LLM application</b>	<b>17</b>
<b>Lab 6: Deploy AMP</b>	<b>23</b>

# Lab 1: CML Walkthrough - Tour ([CML Tour](#))

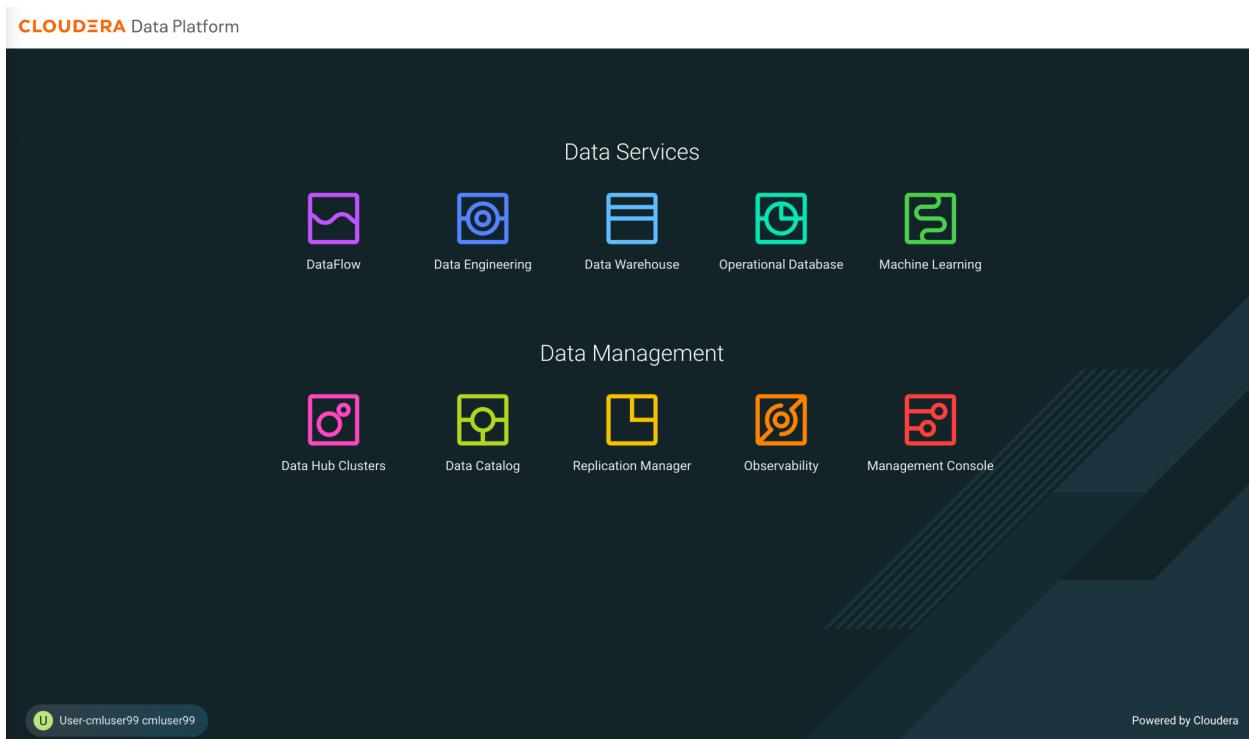
Cloudera Machine Learning (CML) makes it easy to bring Data Scientists to the data platform and makes it easier to overcome the challenges they face on a day to day basis.

With CML you can focus on writing the machine learning code and productionizing models instead of worrying about how to access the data.

**Workshop user login Keycloak SSO URL:**

<http://3.16.19.115/auth/realms/master/protocol/saml/clients/cdp-sso>

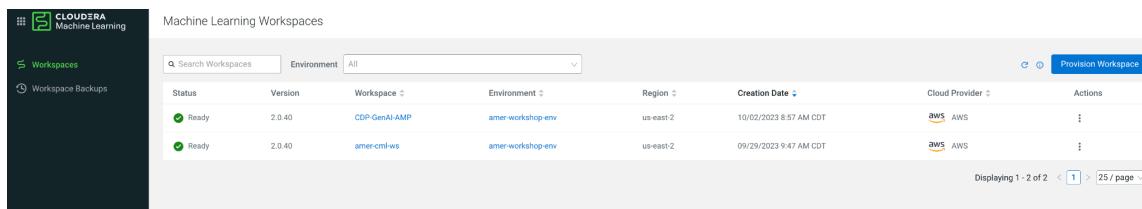
1. Click on the Machine Learning icon to continue.



## 2. Open this workspace by clicking on its name -amer-cml-ws

Workspaces are different areas where teams of data scientists can be separated based on factors such as the cloud provider they're working on and the environment they have access to.

Here we have the **amer-cml-ws** that is created on an environment called **amer-workshop-env** hosted on **AWS**. CML allows you to choose the particular compute instances you would like for a new Workspace, including using GPUs.

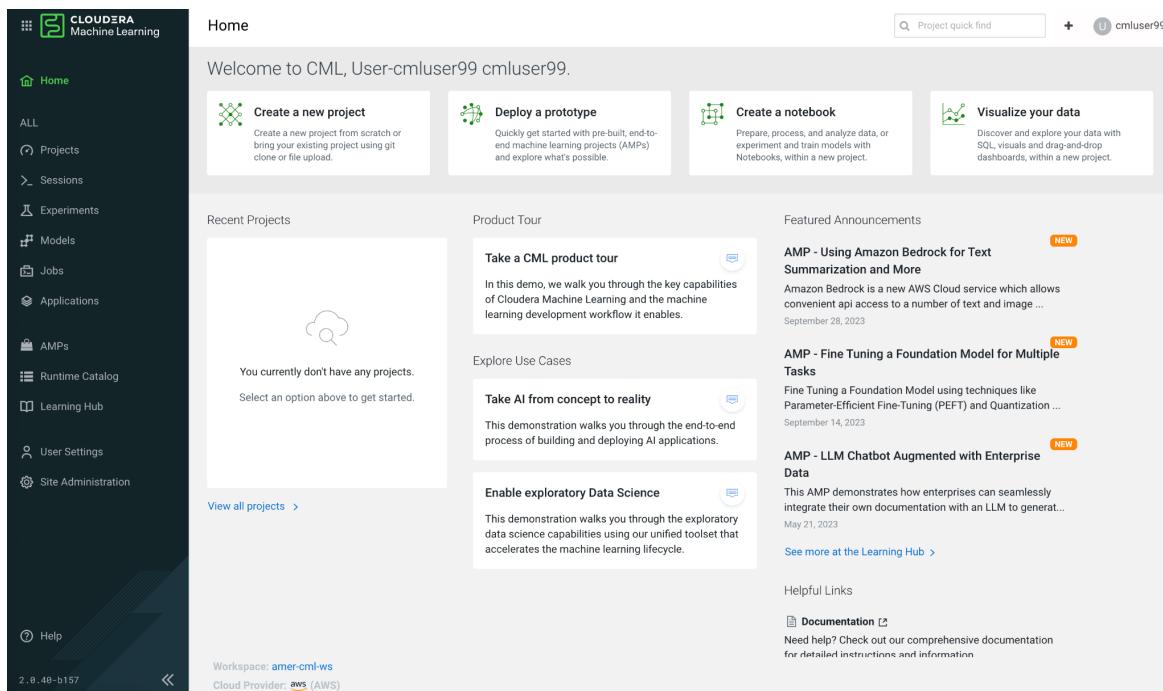


The screenshot shows the 'Machine Learning Workspaces' page. On the left, there's a sidebar with 'Workspaces' and 'Workspace Backups'. The main area has a table titled 'Machine Learning Workspaces' with columns: Status, Version, Workspace, Environment, Region, Creation Date, Cloud Provider, and Actions. There are two entries: one for 'CDP-GenAI-AMP' and one for 'amer-cml-ws'. Both are 'Ready' with version 2.0.40. The 'amer-cml-ws' entry is highlighted in blue. The table footer shows 'Displaying 1 - 2 of 2' and a '25 / page' dropdown.

On the main page you can see a list of projects that you are working on with your data science team members and action cards that take you through the process of creating a project, deploying prototypes, creating a notebook, and visualizing your data.

A project is a collaborative space for a team of Data Scientists to work on a single project together. Here you can view the code and files that belong to a project.

Also, all of the other assets that stem from this project (i.e. Models, Jobs) are visible here as well.



The screenshot shows the 'Home' page of Cloudera Machine Learning. The left sidebar includes links for Home, Projects, Sessions, Experiments, Models, Jobs, Applications, AMPS, Runtime Catalog, Learning Hub, User Settings, Site Administration, and Help. The main content area has a 'Welcome to CML, User-cmluser99 cmluser99.' message. It features four action cards: 'Create a new project', 'Deploy a prototype', 'Create a notebook', and 'Visualize your data'. Below these are sections for 'Recent Projects' (empty), 'Product Tour' (with 'Take a CML product tour' card), 'Explore Use Cases' (with 'Take AI from concept to reality' and 'Enable exploratory Data Science' cards), and 'Featured Announcements' (with cards for 'AMP - Using Amazon Bedrock for Text Summarization and More', 'AMP - Fine Tuning a Foundation Model for Multiple Tasks', and 'AMP - LLM Chatbot Augmented with Enterprise Data'). At the bottom, there's a 'Helpful Links' section with a 'Documentation' link.

### 3. Click on **AMPs** on the left menu bar

Applied ML Prototypes (AMPs) provide reference example machine learning projects in Cloudera Machine Learning. More than simplified quickstarts or tutorials, AMPs are fully-developed expert solutions created by Cloudera's research arm, Fast Forward Labs.

The screenshot shows the Cloudera Machine Learning interface with the 'AMPs' option selected in the left sidebar. The main area displays a grid of 12 pre-built machine learning prototypes:

- Amazon Bedrock**: Text Summarization and more with Amazon Bedrock. Tags: BEDROCK, LLM.
- Cloudera Machine Learning**: Fine-Tuning a Foundation Model for Multiple Tasks (with QLoRA). Tags: HUGGINGFACE, QLORA.
- LLM Chatbot Augmented with Enterprise Data**: LLM Chatbot Augmented with Enterprise Data. Tags: CHATBOT, LLM.
- Churn Modeling with scikit-learn**: Churn Prediction using Logistic Regression. Tags: CHURN PREDICTION, LOGISTIC REGRESSION.
- Deep Learning for Image Analysis**: Computer Vision and Image Analysis. Tags: COMPUTER VISION, IMAGE ANALYSIS.
- Deep Learning for Anomaly Detection**: Anomaly Detection using TensorFlow. Tags: ANOMALY DETECTION, TENSORFLOW.
- NeuralQA**: Question Answering using BERT. Tags: QUESTION ANSWERING, BERT.
- Structural Time Series**: Time Series Forecasting using Prophet. Tags: TIME SERIES, PROPHET.
- Analyzing News Headlines with SpaCy**: Analyzing News Headlines with SpaCy. Tags: SPACY, NLP.
- Deep Learning for Question Answering**: Automated Question Answering and Extractive Ques. Tags: AUTOMATED QUESTION ANSWERING, EXTRACTIVE QUES.
- Explaining Models with LIME and SHAP**: Interpreting and Explaining Models. Tags: INTERPRETABILITY, EXPLAINABILITY.
- Active Learning**: Active Learning and Learning with Limited Labeled Data. Tags: ACTIVE LEARNING, LEARNING WITH LIMITED LABELED D.

Each prototype card includes a thumbnail image, a brief description, and tags. The interface also features a search bar, a project quick find, and a user profile at the top right.

4. Click on **Runtime Catalog** on the left menu bar

Managing ML Runtimes provides overview, installation, set up, configuration, and customization information for Machine Learning Runtimes. ML Runtimes are responsible for running the code written by users and intermediating access to the Data Hub.

You can think of an ML Runtime as a virtual machine, customized to have all the necessary dependencies to access the computing cluster while keeping each project's environment entirely isolated.

The screenshot shows the Cloudera Machine Learning interface with the 'Runtime Catalog' selected in the sidebar. The main area displays a table of runtime configurations. The columns are: Status, Editor, Kernel, Edition, Versions, and a right-pointing arrow. The data includes various Workbench and JupyterLab entries across different kernels (CDV, Conda, Python 3.10, 3.10, 3.7, 3.7, 3.8, 3.8, 3.9, 3.9, PBJ Workbench) and editions (Standard, Nvidia GPU). A search bar at the top and a 'Add Runtime' button are also visible.

Status	Editor	Kernel	Edition	Versions	
Green checkmark	Workbench	Cloudera Data Visualization	CDV 7.1.3	1	>
Green checkmark	JupyterLab	Conda	Tech Preview	2	>
Green checkmark	JupyterLab	Python 3.10	Nvidia GPU	2	>
Green checkmark	JupyterLab	Python 3.10	Standard	2	>
Green checkmark	JupyterLab	Python 3.7	Nvidia GPU	2	>
Green checkmark	JupyterLab	Python 3.7	Standard	2	>
Green checkmark	JupyterLab	Python 3.8	Nvidia GPU	2	>
Green checkmark	JupyterLab	Python 3.8	Standard	2	>
Green checkmark	JupyterLab	Python 3.9	Nvidia GPU	2	>
Green checkmark	JupyterLab	Python 3.9	Standard	2	>
Green checkmark	PBJ Workbench	Python 3.10	Nvidia GPU	2	>
Green checkmark	PBJ Workbench	Python 3.10	Standard	2	>
Green checkmark	PBJ Workbench	Python 3.7	Nvidia GPU	2	>
Green checkmark	PBJ Workbench	Python 3.7	Standard	2	>
Green checkmark	PBJ Workbench	Python 3.8	Nvidia GPU	2	>
Green checkmark	PBJ Workbench	Python 3.8	Standard	2	>
Green checkmark	PBJ Workbench	Python 3.9	Nvidia GPU	2	>

## 5. Click on **Learning Hub** on the left menu bar

Learning Hub highlights Featured Announcements, Research and Resources, including Blog Posts, Research Reports and Documentation. It also contains a Feed associated with Cloudera Machine Learning that contains features, blog posts, documentation updates, etc...

## 6. Click on **Site Administration** on the left menu bar

Only site administrators have access to a Site Administration dashboard that can be used to manage the workspace. When you're in Site Administration, you will see an array of tabs for all the tasks you can perform as a site administrator.

# Lab 2: Text Summarization AMP Walkthrough

## Text Summarization and more with Amazon Bedrock

Amazon Bedrock is a fully managed service that makes foundation models from leading AI startups and Amazon available via an API, so you can choose from a wide range of foundation models to find the one that is best suited for your use case.

This repository demonstrates using an instruction to summarize a text using the Amazon Bedrock API.

## AMP Overview

In this AMP we will show you how to set up your environment and use the Amazon Bedrock API and its models. There are two models that you can choose from.

The models we will be highlighting are `amazon.titan-tg1-large` and `anthropic.claude-v2` (supports 100K tokens in the prompt). When calling each of these models using the API, it is important to note the format of the prompt it expects and the schema of the request API.

The screenshot shows the Cloudera Machine Learning (CML) interface. On the left, there's a sidebar with navigation links: Home, ALL, Projects, Sessions, Experiments, Models, Jobs, Applications, AMPs (highlighted), Runtime Catalog, Learning Hub, User Settings, Site Administration, and Help. The main area is titled "Applied ML Prototypes" and contains several cards representing different projects:

- Text Summarization and more with Amazon Bedrock**: Tags: BEDROCK, LLM. Description: "AMPs are pre-built, end-to-end ML projects specifically designed to kickstart use cases." Includes a thumbnail of a guitar and a smartphone.
- Fine-Tuning a Foundation Model for Multiple Tasks (with QLoRA)**: Tags: HUGGINGFACE, QLORA. Description: "Fine-tune a foundation model for multiple tasks using QLoRA." Includes a thumbnail of a person working on a computer.
- LLM Chatbot Augmented with Enterprise Data**: Tags: CHATBOT, LLM. Description: "Augment an LLM chatbot with enterprise data." Includes a thumbnail of a person holding a smartphone.
- Churn Modeling with scikit-learn**: Tags: CHURN PREDICTION, LOGISTIC REGRESSION. Description: "Build a churn prediction model using scikit-learn." Includes a thumbnail of a bar chart.
- Deep Learning for Image Analysis**: Tags: COMPUTER VISION, IMAGE ANALYSIS. Description: "Perform deep learning for image analysis tasks." Includes a thumbnail of a car image.
- Deep Learning for Anomaly Detection**: Tags: ANOMALY DETECTION, TENSORFLOW. Description: "Perform deep learning for anomaly detection." Includes a thumbnail of a graph.
- NeuralQA**: Tags: QUESTION ANSWERING, BERT. Description: "Build a neural question answering system." Includes a thumbnail of a brain diagram.
- Structural Time Series**: Tags: TIME SERIES, PROPHET. Description: "Analyze time series data using Prophet." Includes a thumbnail of a line graph.
- Analyzing News Headlines with SpaCy**: Tags: SPACY, NLP. Description: "Analyze news headlines using SpaCy." Includes a thumbnail of a news article snippet.
- Explaining Models with LIME and SHAP**: Tags: INTERPRETABILITY, EXPLAINABILITY. Description: "Explain machine learning models using LIME and SHAP." Includes a thumbnail of a scatter plot.
- Active Learning**: Tags: ACTIVE LEARNING, LEARNING WITH LIMITED LABELED D. Description: "Perform active learning on limited labeled data." Includes a thumbnail of a hand writing a letter.

The instruction prompt is also customizable to instruct the called model to perform a different text generation task. For example, an engineered prompt to provide a very simple and understandable summarization might look like this:

Please explain CML at a 5 year old level.

*Machine learning is a big helper for companies to do their work and make their products better. It helps automate things and make decisions faster. Cloudera Machine Learning makes it easier for scientists to work together and create machine learning models. They can use different languages and computers to do this in one place. It also helps keep track of everything and make sure it works well.*

**CML\_AMP\_AI\_Text\_Summarization\_with\_Amazon\_Bedrock / README.md**

**Text Summarization and more with Amazon Bedrock**

Amazon Bedrock is a fully managed service that makes foundation models from leading AI startups and Amazon available via an API, so you can choose from a wide range of foundation models to find the one that is best suited for your use case.

This repository demonstrates using an instruction to summarize a text using the Amazon Bedrock API.

**AMP Overview**

In this AMP we will show you how to setup your environment and use the Amazon Bedrock API and its models. There are two models that you can choose from.

The models we will be highlighting are `amazon.titan-tg1-large` and `anthropic.claude-v2` (supports 100K tokens in the prompt). When calling each of these models using the API, it is important to note the format of the prompt it expects and the schema of the request API.

**Amazon Bedrock Text Summarization**

Model: `amazon.titan-tg1-large`

Instruction: Please provide a summary of the following text. Do not add any information that is not mentioned in the text below.

Input Text: Machine learning has become one of the most critical capabilities for modern businesses to grow and stay competitive today. From automating internal processes to optimizing the design, creation, and marketing processes behind virtually every product consumed, ML models have permeated almost every aspect of our work and personal lives.

ML development is iterative and complex, made even harder because most ML tools aren't built for the entire machine learning lifecycle. Cloudera Machine Learning on Cloudera Data Platform accelerates time-to-value by enabling data scientists to work in a single, unified platform that is all inclusive for powering any AI use case. Purpose-built for agile experimentation and production ML workflows, Cloudera Machine Learning manages everything from data preparation to MLops, to prediction serving and deployment. ML changes the machine learning lifecycle with greater speed and agility to discover opportunities which can mean the difference for your business.

Each ML workspace enables teams of data scientists to develop, test, train, and ultimately deploy machine learning models for building predictive applications all on the data under management within the enterprise data cloud. ML workspaces support fully containerized execution of Python, R, Scala, and Spark workloads through flexible and enterprise engines.

Advanced Generation Options

Bedrock API Request Details

```

{
  "instruction": "[instruction]\n[:]",
  "maxTokenCount": "[max_tokens]",
  "stopSequences": "[",
  "temperature": "[temperature]",
  "topP": "[top_p]"
}
  
```

Output

Modern businesses depend on machine learning to grow and stay competitive, but most ML tools are not built for the entire lifecycle. Cloudera Machine Learning on Cloudera Data Platform simplifies ML development by enabling data scientists to collaborate in a single, unified platform that supports agile experimentation and production ML workflows. It manages everything from data preparation to MLops and predictive reporting, allowing businesses to solve critical ML challenges quickly and efficiently.

Input Format

```

{
  "inputText": "[input_text]",
  "textGenerationConfig": {
    "maxTokenCount": "[max_tokens]",
    "stopSequences": "[",
    "temperature": "[temperature]",
    "topP": "[top_p]"
  }
}
  
```

AWS Credentials

Summarize

Reset

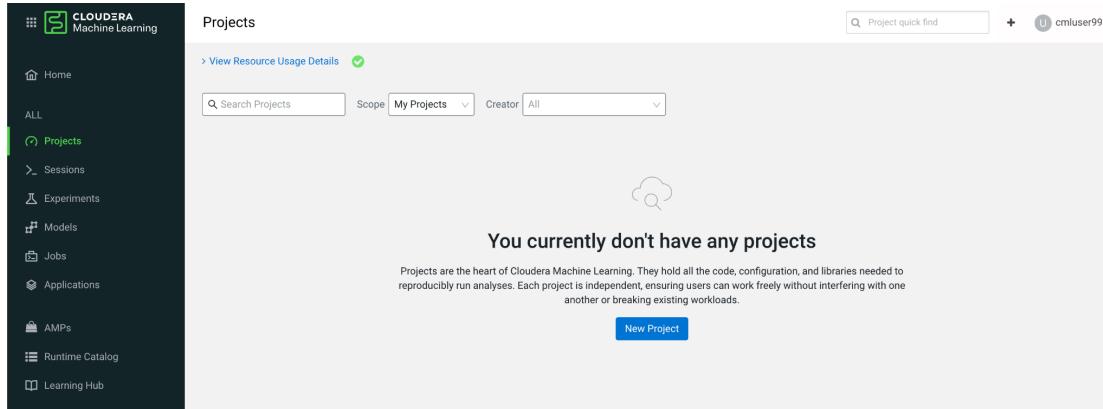
The instruction prompt is also customizable to instruct the called model to perform a different text generation task. For example, an engineered prompt to provide a very simple and understandable summarization might look like this:

Please explain CML at a 5 year old level.

# Lab 3: Manually Create Project via Git

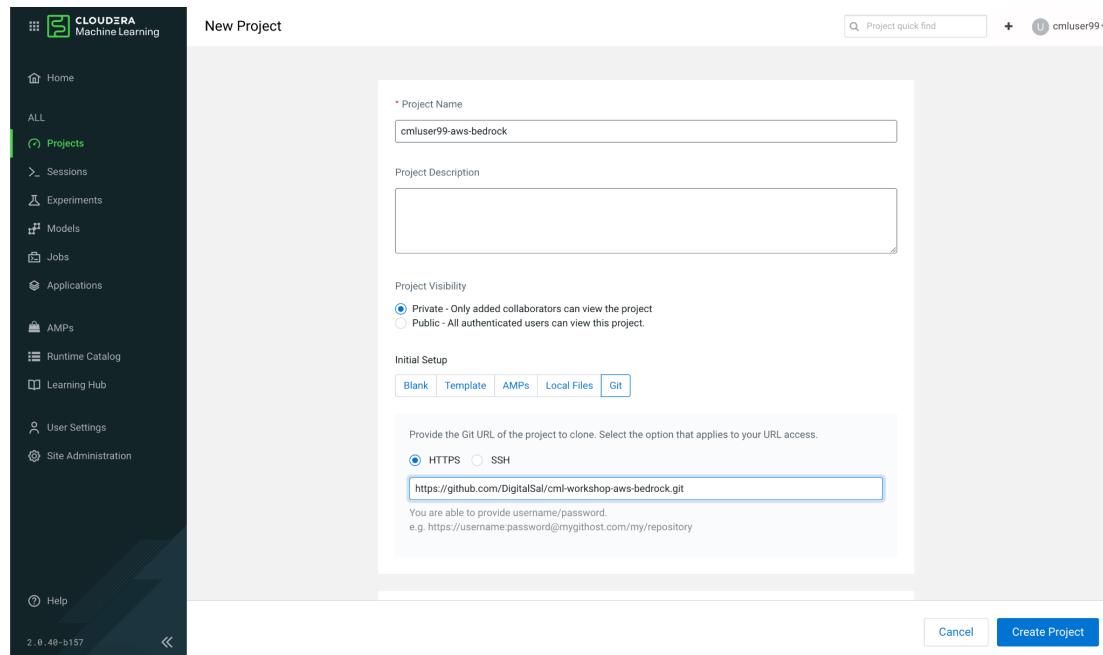
In Lab 3 we're going to manually create a project from github and configure it to run a Text Summarization app that uses Amazon Bedrock APIs.

1. Click on **Projects** on the left menu bar
2. Click on the **New Project** button on the main screen



The screenshot shows the Cloudera Machine Learning interface. On the left, a dark sidebar lists various options: Home, ALL, Projects (which is selected and highlighted in green), Sessions, Experiments, Models, Jobs, Applications, AMPs, Runtime Catalog, and Learning Hub. The main content area is titled 'Projects' and displays a message: 'You currently don't have any projects'. Below this message is a small icon of a cloud with a magnifying glass. A brief description follows: 'Projects are the heart of Cloudera Machine Learning. They hold all the code, configuration, and libraries needed to reproducibly run analyses. Each project is independent, ensuring users can work freely without interfering with one another or breaking existing workloads.' At the bottom right of the main area is a blue 'New Project' button.

3. Complete the **New Project** form.
  - Provide a **Project Name**.
  - Project Visibility: **Private**
  - Initial Setup: **Git**
    - Git URL: <https://github.com/DigitalSal/cml-workshop-aws-bedrock.git>
  - Click on the **Create Project** button



The screenshot shows the 'New Project' dialog box. The sidebar on the left is identical to the previous screenshot. The main dialog box is titled 'New Project'. It contains the following fields:

- \* Project Name: 'cmiuser99-aws-bedrock'
- Project Description: An empty text area.
- Project Visibility: A radio button group where 'Private - Only added collaborators can view the project' is selected, and 'Public - All authenticated users can view this project' is unselected.
- Initial Setup: A tabbed section with tabs for 'Blank', 'Template', 'AMPS', 'Local Files', and 'Git'. The 'Git' tab is selected.
- Provide the Git URL of the project to clone. Select the option that applies to your URL access:
  - HTTPS (radio button selected)
  - SSH (radio button unselected)
- Git URL input field: 'https://github.com/DigitalSal/cml-workshop-aws-bedrock.git'

At the bottom right of the dialog box are two buttons: 'Cancel' and 'Create Project'.

Your new project should look something like this.

4. Click on the **New Session** button.

The screenshot shows the Cloudera Machine Learning interface. On the left, a sidebar menu includes options like Home, All Projects, PROJECT Overview, Sessions, Data, Experiments, Models, Jobs, Applications, Files, Collaborators, Project Settings, AMPs, Runtime Catalog, Learning Hub, User Settings, and Help. The version is listed as 2.0.40-b157. The main area displays a project named "cmluser99 / cmluser99-aws-bedrock". The "Models" section indicates no models yet. The "Jobs" section indicates no jobs yet. The "Files" section lists several files: "amp\_1\_session-install-deps", "amp\_2\_app", "images", "utils", "README.md", "requirements.txt", and "summarization\_example.ipynb". A "Text Summarization and more with Amazon Bedrock" section describes the service. At the bottom, it shows the workspace as "amer-cml-ws" and the cloud provider as "aws (AWS)".

5. Provide a session name and click the **Start Session** button to start a Workbench session.

The screenshot shows the "Start A New Session" dialog. The sidebar menu is visible on the left. The main dialog has a "Session Name" field containing "Data Science". Under "Runtime", "Editor" is set to "Workbench", "Kernel" is "Python 3.9", "Edition" is "Standard", and "Version" is "2023.08". A note says "Configure additional runtime options in Project Settings". Below this, "Enable Spark" is turned off, and "Spark 3.2.3 - CDE 1.19.2 - HOTFIX-2" is selected. Under "Runtime Image", it shows "docker.repository.cloudera.com/cloudera/cdsw/ml-runtime-workbench-python3.9-standard:2023.08.2-b8". In the "Resource Profile" section, "2 vCPU / 4 GiB Memory" is selected. At the bottom right are "Cancel" and "Start Session" buttons.

6. Click the **Close** button to close the Connection code snippet pop up.

The screenshot shows the AWS Bedrock Data Science interface. On the left, there's a sidebar with project files: cmluser99-aws-bedrock, project-metadata.yaml, amp\_1\_session-install-deps (selected), setup.py, amp\_2\_app, images, README.md, requirements.txt, summarization\_example.ipynb, and utils. The main area has a terminal window with the command: !pip install --no-cache-dir -r requirements.txt. Below it is a "Connection Code Snippet" pop-up window. The snippet lists two connections: "amer-workshop" (Spark Data Lake) and "amer-hive" (Hive Virtual Warehouse). It includes sample Python code for connecting to the Spark Data Lake:

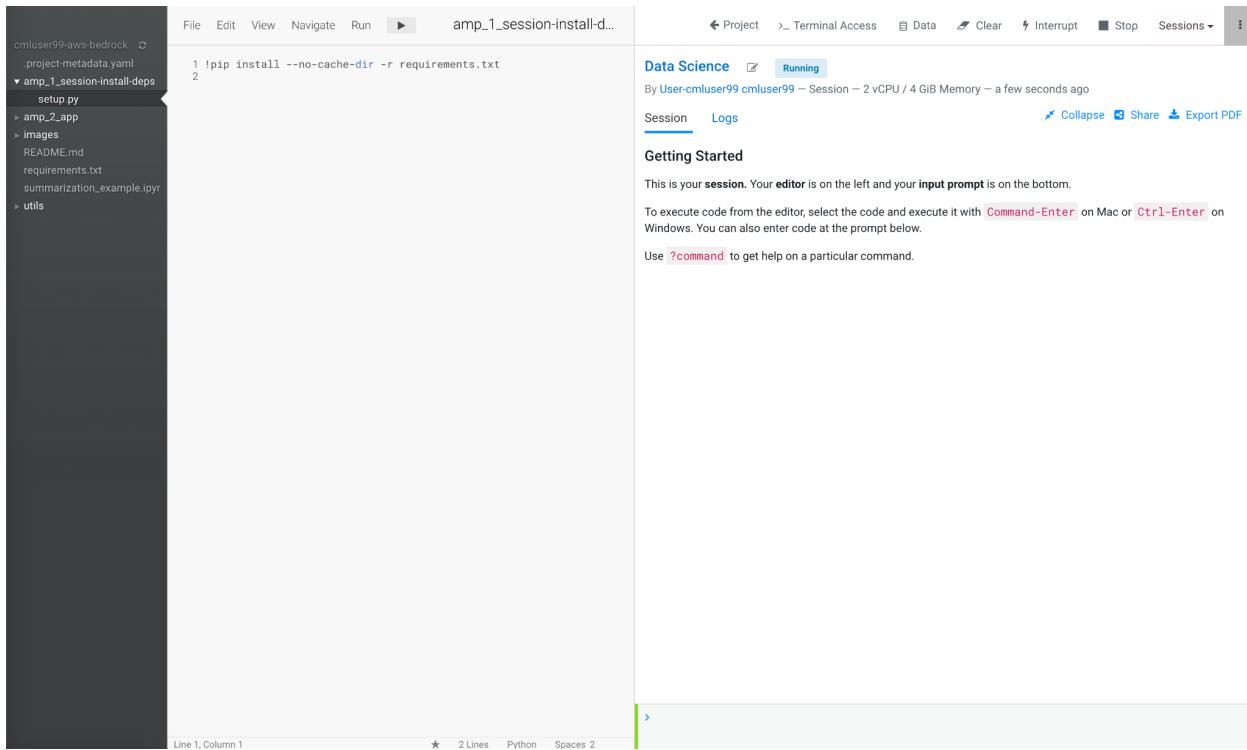
```
import cmldata.v1 as cmldata
CONNECTION_NAME = "amer-workshop"
conn = cmldata.get_connection(CONNECTION_NAME)
spark = conn.get_spark_session()

# Sample usage to run query through spark
EXAMPLE_SQL_QUERY = "show databases"
spark.sql(EXAMPLE_SQL_QUERY).show()
```

At the bottom of the pop-up, there are checkboxes for "Don't show me this again for:" (with options "This Project", "All Projects") and a "Close" button.

# Lab 4: Test Amazon Bedrock API using Jupyter Notebook

1. Select the **setup.py** file in the **amp\_1\_session-install-deps** folder. This file installs requirements listed in the requirements.txt file.



The screenshot shows a Jupyter Notebook interface with a code editor on the left and a terminal output on the right. The code editor displays the command:

```
1 !pip install --no-cache-dir -r requirements.txt
```

The terminal output shows the command is running:

Data Science    Running  
By User-cmluser99 cmluser99 — Session — 2 vCPU / 4 GiB Memory — a few seconds ago

Session    Logs    [Collapse](#) [Share](#) [Export PDF](#)

**Getting Started**

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

Line 1, Column 1    ★ 2 Lines    Python    Spaces 2

This code will take a few minutes to run. When code has finished you will see the green indicator that shows the cursor is ready.

Now we're going to run a jupyter notebook to test our code.

2. Click on **Sessions** menu and then click on **+New Session**

File Edit View Navigate Run ➔ amp\_1\_session-install-d...

cmcluser99-aws-bedrock

- project-metadata.yaml
- amp\_1\_session-install-deps
- setup.py
- > amp\_2\_app
- > images
- README.md
- requirements.txt
- summarization\_example.ipynb
- > utils

```
1 !pip install --no-cache-dir -r requirements.txt
2
```

Data Science ➔ Running  
By User-cmcluser99 cmcluser99 — Session — 2 vCPU / 4 GiB Memory — a few seconds ago

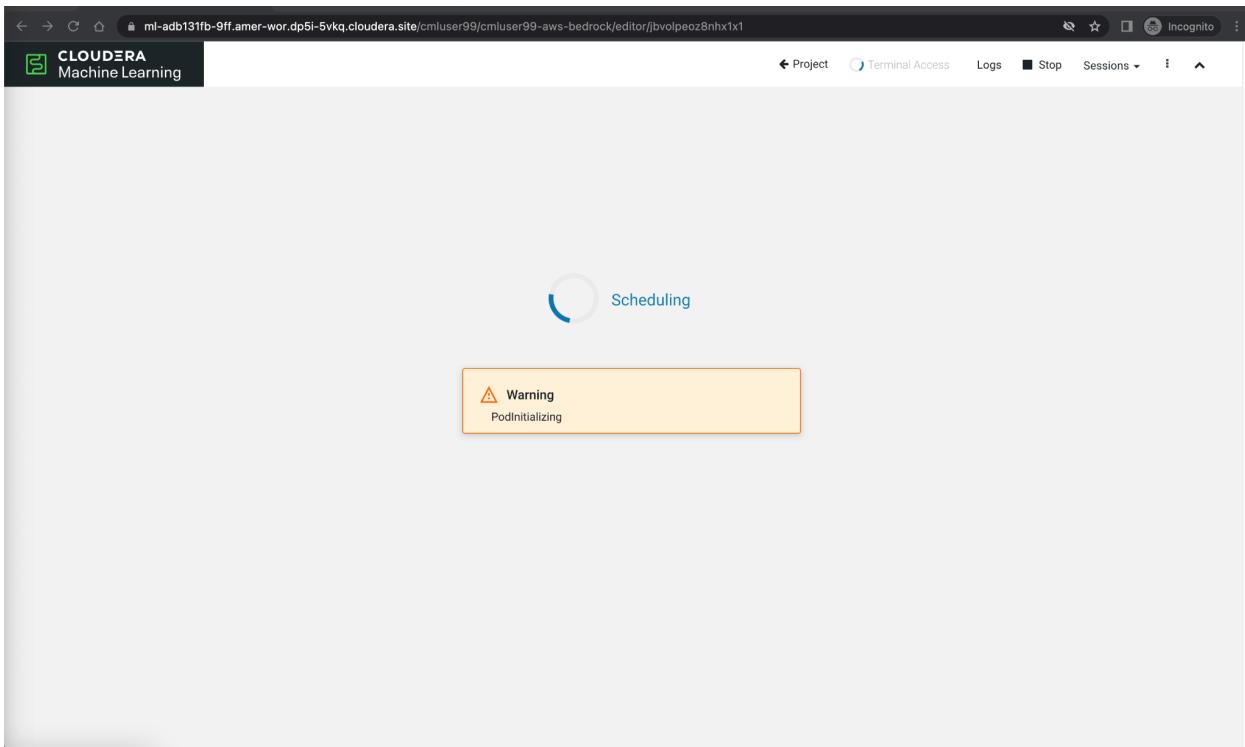
Session Logs Collapse Share Export PDF

```
Downloading tqdm-4.62.0-py3-none-any.whl (/8 KB) 78.3/78.3 kB 270.5 MB/s eta 0:00:00
    Downloading filelock-3.12.4-py3-none-any.whl (11 kB)
    Downloading fsspec-2023.9.2-py3-none-any.whl (173 kB) 173.4/173.4 kB 300.9 MB/s eta 0:00:00
    Downloading jsonschema-specifications-2023.7.1-py3-none-any.whl (17 kB)
    Downloading referencing-0.38.2-py3-none-any.whl (25 kB)
    Downloading rpds_py-0.10.3-cp39-cp39-manylinux_2_17_x86_64_manylinux2014_x86_64.whl (1.2 MB) 1.2/1.2 MB 312.7 MB/s eta 0:00:00
    Downloading exceptiongroup-1.1.3-py3-none-any.whl (14 kB)
Building wheels for collected packages: ffmpeg
    Building wheel for ffmpeg (setup.py) ... done
        Created wheel for ffmpeg: filename=ffmpeg-0.3.1-py3-none-any.whl size=5595 sha256=f9d044dd1009ec023339fc0800e4444e3717b9f37bucc08706f60e1a76c605
        Stored in directory: /tmp/pip-ephem-wheel-cache-yrflv1mj/wheels/ff1/f1/8d/36792b023b526b7c2ed5db30932def7b18cf39d7acbe0572
Successfully built ffmpeg
Installing collected packages: pydub, ffmpeg, websockets, tzdata, typing-extensions, tqdm, toolz, sniffio, semantic-version, rpds-py, python-multipart, pyasn1, orjson, numpy, markupsafe, jmespath, h11, fsspec, filelock, exceptiongroup, docutils, colorama, attrs, annotated-types, aiofiles, uvicorn, rsa, referencing, pydantic-core, pandas, jinja2, huggingface-hub, botocore, anyio, starlette, s3transfer, pydantic, jsonschema-specifications, httpcore, jsonschema, httpx, fastapi, botos3, awscli, gradio-client, altair, gradio
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following dependency conflicts.
mflow-cml-plugin 0.0.1 requires typing-extensions==4.1.1, but you have typing-extensions 4.8.0 which is incompatible.
Successfully installed aiofiles-23.2.1 altair-5.1.2 annotated-types-0.5.0 anyio-3.7.1 attrs-23.1.0 awscli-1.29.57 botocore-1.28.57 botocore-1.31.57 colorama-0.4.4 docutils-0.16 exceptiongroup-1.1.3 fastapi-1.29.2 ffmpy-0.3.1 filelock-3.12.4 httpcore-0.23.2 grad-3.44.4 gradio-client-0.5.1 h11-0.14.0 httpcore-0.18.0 httpx-0.25.8 huggingface-hub-0.17.3 jinja2-3.1.2 jmespath-1.0.1 jsonschema-4.19.1 jsonschema-specifications-2023.7.1 markupsafe-2.1.3 numpy-1.26.0 orjson-3.9.7 pandas-2.1.1 pyasn1-0.8.5 pydantic-2.4.2 pydantic-core-2.10.1 pydub-0.2.5.1 python-multipart-0.0.6 referencing-0.30.2 rpds_py-0.10.3 rsa-4.7.2 s3transfer-0.7.0 semantic-version-2.10.0 sniffio-1.3.0 starlette-0.19.0 toolz-0.12.0 tqdm-4.66.1 typing-extensions-4.8.0 tzdata-2023.3 uvicorn-0.23.2 websockets-11.0.3
```

3. Make sure to select the **JupyterLab** editor and click **Start Session** button

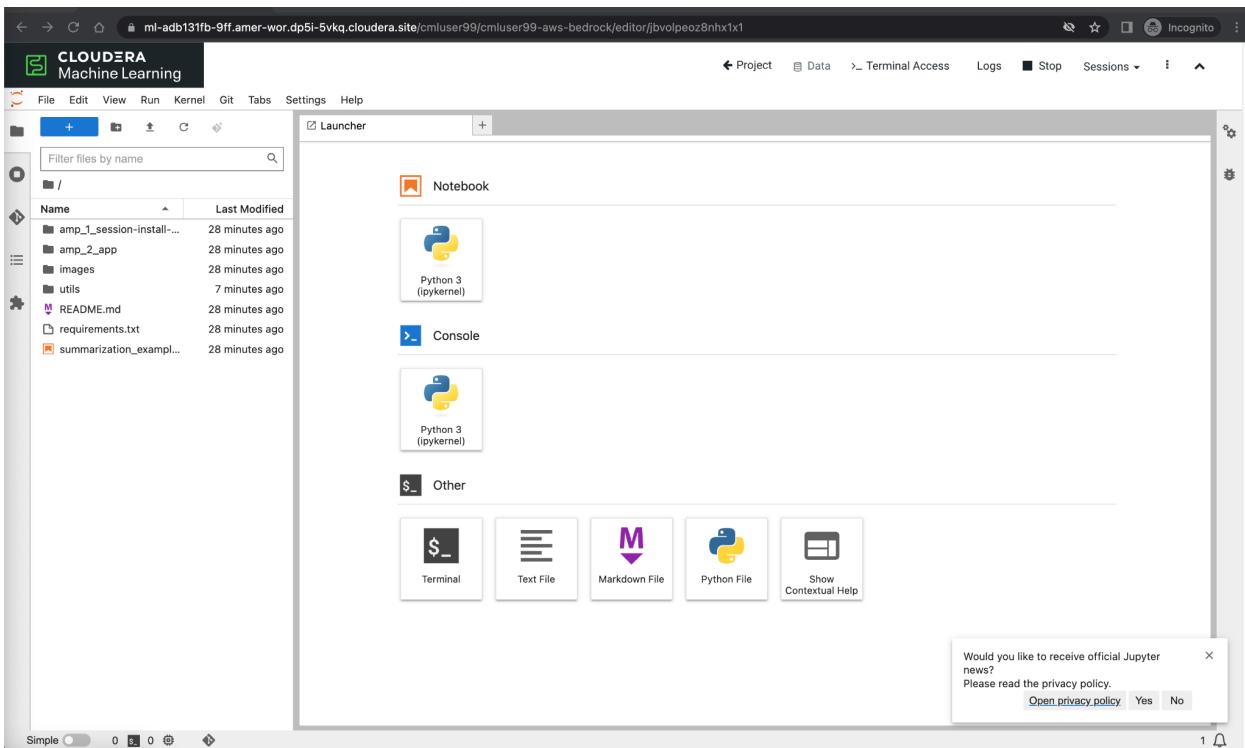
The screenshot shows the Cloudera Machine Learning web interface. The left sidebar contains navigation links such as Home, All Projects, Overview, Sessions (which is selected), Data, Experiments, Models, Jobs, Applications, Files, Collaborators, Project Settings, AMPS, Runtime Catalog, Learning Hub, and Help. The main content area is titled "Start A New Session". It displays "Running Sessions" with two entries: "Data Science" started 23 minutes ago and "Untitled Session" started 24 minutes ago. Below this, a "Session Name" field is populated with "Data Science". The "Runtime" section includes fields for "Editor" (set to JupyterLab), "Kernel" (Python 3.9), "Edition" (Standard), and "Version" (2023.08). A "Configure additional runtime options in Project Settings" link is present. A "Enable Spark" toggle switch is turned on, with the dropdown showing "Spark 3.2.3 - CDE 1.19.2 - HOTFIX-2". The "Runtime Image" is listed as "docker.repository.cloudera.com/cloudera/cdsweb/ml-runtime-jupyterlab:python3.9-standard:2023.08.2-b8". The "Resource Profile" dropdown shows "2 vCPU / 4 GiB Memory". At the bottom right are "Cancel" and "Start Session" buttons.

This will take a few seconds to load.



Once the JupyterLab editor is open your screen should look like this.

#### 4. Click on the **summarization\_example.ipynb** file



Feel free to look over the code.

5. In the first block, update the code to the following:

- `os.environ["AWS_DEFAULT_REGION"] = "us-east-1"`
- `os.environ["AWS_ACCESS_KEY_ID"] = "AKIA6I6SNFMLB7UQIFWL"`
- `os.environ["AWS_SECRET_ACCESS_KEY"] = "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"`

6. Click on **Run -> Run All Cells** or run them line by line

The screenshot shows a Jupyter Notebook interface with the following content:

- File Explorer:** Shows a directory structure with files like `amp_1_session-install...`, `amp_2_app`, `images`, `utils`, `README.md`, and `requirements.txt`.
- Code Cells:**
  - [1]: 

```
import os
os.environ["AWS_DEFAULT_REGION"] = ""
os.environ["AWS_ACCESS_KEY_ID"] = ""
os.environ["AWS_SECRET_ACCESS_KEY"] = ""
```
  - [2]: 

```
!pip install -q --no-cache-dir -r requirements.txt
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following dependency conflicts.
mlflow-cml-plugin 0.0.1 requires typing-extensions==4.1.1, but you have typing-extensions 4.8.0 which is incompatible.
```
  - [3]: 

```
import json
import os
from typing import Optional
import boto3
from botocore.config import Config
```
  - [4]: 

```
def get_bedrock_client(
    assumed_role: Optional[str] = None,
    endpoint_url: Optional[str] = None,
    region: Optional[str] = None,
```
- Output:** A message box asking if the user wants to receive official Jupyter news, with options to "Open privacy policy", "Yes", and "No".
- Bottom Status Bar:** Shows "Initializing..." and "Python 3 (ipykernel) | Idle".

*Note: Titan model will error out.*

The screen should look like this when completed:

A screenshot of the Cloudera Machine Learning JupyterLab interface. The top navigation bar includes Project, Data, Terminal Access, Logs, Stop, Sessions, and Help. The left sidebar shows a file tree with files like README.md, requirements.txt, and summarization\_example.ipynb. The main area displays a code cell with Python 3 (ipykernel) content:

```
[14]: modelId = 'anthropic.claude-v2'
response = boto3_bedrock.invoke_model(body=body, modelId=modelId, accept='application/json', contentType='application/json')
response_body = json.loads(response.get('body').read())

The response body is specific to the Claude Model API, see AWS Bedrock documentation for more details.

[15]: result = response_body.get('completion')
print(result)

Here is a summary of the key points from the text:
- Machine learning has become critical for modern businesses to stay competitive by automating processes and optimizing product s.
- ML development is complex and iterative. Most ML tools aren't built for the entire lifecycle.
- Cloudera Machine Learning accelerates time-to-value by enabling collaboration on a unified platform for any AI use case, manag ing everything from data prep to MLops and reporting.
- ML workspaces in Cloudera let data science teams develop, test, train and deploy models to build predictive apps using Python, R, Scala and Spark.
```

At the bottom right, a modal dialog box asks "Would you like to receive official Jupyter news? Please read the privacy policy." with "Open privacy policy" and "Yes" / "No" buttons.

7. Stop the JupyterLab session, by clicking on **Stop** in the top menu.

# Lab 5: Deploy LLM application

With CML can deploy web applications. In this lab we will manually deploy a UI that interacts with the LLM models.

The application file is located in the **amp\_2\_app** folder called **bedrock-app.py**.

1. Stop session and get back to the Project.

The screenshot shows a Jupyter Notebook interface with two panes. The left pane displays Python code for generating prompts for summarization models. The right pane shows the terminal output of the code execution.

```
File Edit View Navigate Run ▶ amp_2_app/bedrock-ap...◀ Project ▶ Terminal Access ⚡ Data ⚡ Clear ⚡ Interrupt █ Stop Sessions...
```

**cmluser99-aws-bedrock**

- project-metadata.yaml
- ▼ **amp\_1\_session-install-deps**
- setup.py
- amp\_2\_app
- bedrock-app.py
- example.txt
- images
- README.md
- requirements.txt
- summarization\_example.ipynb
- utils

```
1 import gradio as gr
2 import os
3 import json
4 from utils import bedrock
5
6 accept = 'application/json'
7 contentType = 'application/json'
8
9 with open('amp_2_app/example.txt', 'r') as file:
10     example_text = file.read()
11 examples = {'CML Documentation': example_text}
12 def example_lookup(text):
13     if text:
14         return examples[text]
15     return ''
16
17 example_instruction = "Please provide a summary of the following text"
18
19 def clear_out():
20     cleared_tuple = (gr.Textbox.update(value=""), gr.Textbox.update(value=""))
21     return cleared_tuple
22
23 # List of LLM models to use for text summarization
24 models = ['amazon.titan-tg1-large', 'anthropic.claude-v2']
25
26 # Setting up the prompt syntax for the corresponding model
27 def prompt_construction(modelId, instruction="instruction", prompt=""):
28     if modelId == "amazon.titan-tg1-large":
29         full_prompt = instruction + "\n<text>" + prompt + "</text>"
30     elif modelId == "anthropic.claude-v2":
31         full_prompt = "Human: " + instruction + "\n<text>" + prompt + "\n</text> Assistant:"
32     else:
33         return full_prompt
34
35
36 # Setting up the API call in the correct format for the corresponding model
37 def json_format(modelId, tokens, temperature, top_p, full_prompt=""):
38     if modelId == "amazon.titan-tg1-large":
39         body = json.dumps({"inputText": full_prompt,
40                            "textGenerationConfig": {
41                                "maxTokenCount": tokens,
42                                "stopSequences": [],
43                                "temperature": temperature,
44                                "topP": top_p}})
45     elif modelId == "anthropic.claude-v2":
46         body = json.dumps({"prompt": full_prompt,
47                            "max_tokens_to_sample": tokens,
48                            "temperature": temperature,
49                            "top_k": 250,
50                            "top_p": top_p,
51                            "stop_sequences": []})
52
53
54     return body
55
56
```

**Data Science** Running

By User-cmluser99 cmluser99 - Session - 2 vCPU / 4 GiB Memory — a few seconds ago

**Sessions** **Logs**

Downloading tqdm-4.66.1-py3-none-any.whl (8 KB) 78.3/78.3 kB 270.5 MB/s eta 0:00:00

Downloading filelock-3.12.4-py3-none-any.whl (11 kB)

Downloading fsspec-2023.9.2-py3-none-any.whl (173 kB) 173.4/173.4 kB 300.9 MB/s eta 0:00:00

Downloading jsonschema\_specsifications-2023.7.1-py3-none-any.whl (17 kB)

Downloading referencing-0.30.2-py3-none-any.whl (25 kB)

Downloading rpds-py-0.10.3-cp39-manylinux\_2\_17\_x86\_64.manylinux2014\_x86\_64.whl (1.2 MB) 1.2/1.2 MB 312.7 MB/s eta 0:00:00

Downloading exceptiongroup-1.1.3-py3-none-any.whl (14 kB)

Building wheels for collected packages: ffmpy

Building wheel for ffmpy (setup.py) ... done

Created wheel for ffmpy: filename=ffmpy-0.3.1-py3-none-any.whl size=5595 sha256=fd9d044dd1009ec92333fc008e4444e3717b937fbbcc88706f60e1a76c685

Stored in directory: /tmp/pip-ephem-wheel-cache-yr1fvmj/wheels/1f/f1/b/367922b023b526b7c2ced5d38932deff7b18cf397da66e5872

Successfully built ffmpy

Installing collected packages: pydub, ffmpy, websockets, tzdata, typing-extensions, tqdm, tolz, sniffio, semver, rpds-py, python-multipart, pyasn1, orjson, numpy, markupsafe, jmespath, h11, fsspec, filelock, exceptiongroup, docutils, colorama, attrs, annotated-types, aiofiles, uvicorn, rsa, referencing, pydantic-core, pandas, jinja2, huggingface-client, botocore, anyio, starlette, s3transfer, pydantic, jsonschema-specifications, httpcore, json schema, httpx, fastapi, boto3, awscli, gradio-client, altair, gradisq

ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following dependency conflicts.

mfiflow-cml-plugin 0.8.1 requires typing-extensions==4.1.1, but you have typing-extensions 4.8.0 which is incompatible.

Successfully installed aiofiles-23.2.1 altair-5.1.2 annotated-types-0.5.0 anyio-3.7.1 attrs-23.1.0 awslimits-1.29.57 boto3-1.28.57 botocore-1.31.57 colorama-0.4.1 docutils-1.16 exceptiongroup-1.1.3 fastapi-0.103.2 ffmpy-0.3.1 filelock-3.12.4 fsspec-2023.9.2 gradio-3.44.4 gradio-0.5.1 h11-0.14.1 httpcore-1.18.0 httpx-0.25.5 huggingface-hub-0.17.3 jinja2-3.1.2.1 jmespath-1.0.1 jsonschema-4.19.1 jsonschema\_specsifications-2023.7.1 markupsafe-2.1.3 numpy-1.26.0 orjson-3.9.7 pandas-1.2.1 pyasn1-0.5.0 pydantic-2.4.2 pydantic-core-2.10.0 pydub-0.25.1 python-multipart-0.0.6 referencing-0.30.2 rpds-py-0.10.3 rsa-4.7.2 s3transfer-0.7.0 semver-2.18.0 sniffio-1.3.0 starlette-0.27.0 toolz-0.12.0 tqdm-4.66.1 typing-extensions-4.8.0 tzdata-2023.3 uvicorn-0.23.2 websockets-11.0.3

2. Click on the **Project Settings** menu item on the left.

3. Click on the **Advanced** tab.

- Enter 3 Environment Variables and their values that the application will reference:
  - AWS\_DEFAULT\_REGION** = us-east-1
  - AWS\_ACCESS\_KEY\_ID** = AKIA6I6SNFMLB7UQIFWL
  - AWS\_SECRET\_ACCESS\_KEY** = 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
- Click the **Submit** button when complete.

The screenshot shows the 'Project Settings' page for a project named 'cmluser99'. The 'Advanced' tab is selected. Under 'Environment Variables', there are five fields with placeholder values and checkboxes for visibility:

- CDSW\_APP\_POLLING\_ENDPOINT
- PROJECT\_OWNER
- AWS\_ACCESS\_KEY\_ID
- AWS\_SECRET\_ACCESS\_KEY
- AWS\_DEFAULT\_REGION

At the bottom right is a blue 'Submit' button.

- Click on the **Application** menu item on the left.
- Click on the **New Application** button

The screenshot shows the 'Applications' page. The left sidebar has 'Applications' selected. The main area displays a message: 'You currently don't have any applications'. Below the message is a small icon of a cloud with a magnifying glass. A blue 'New Application' button is located at the bottom right of the main area.

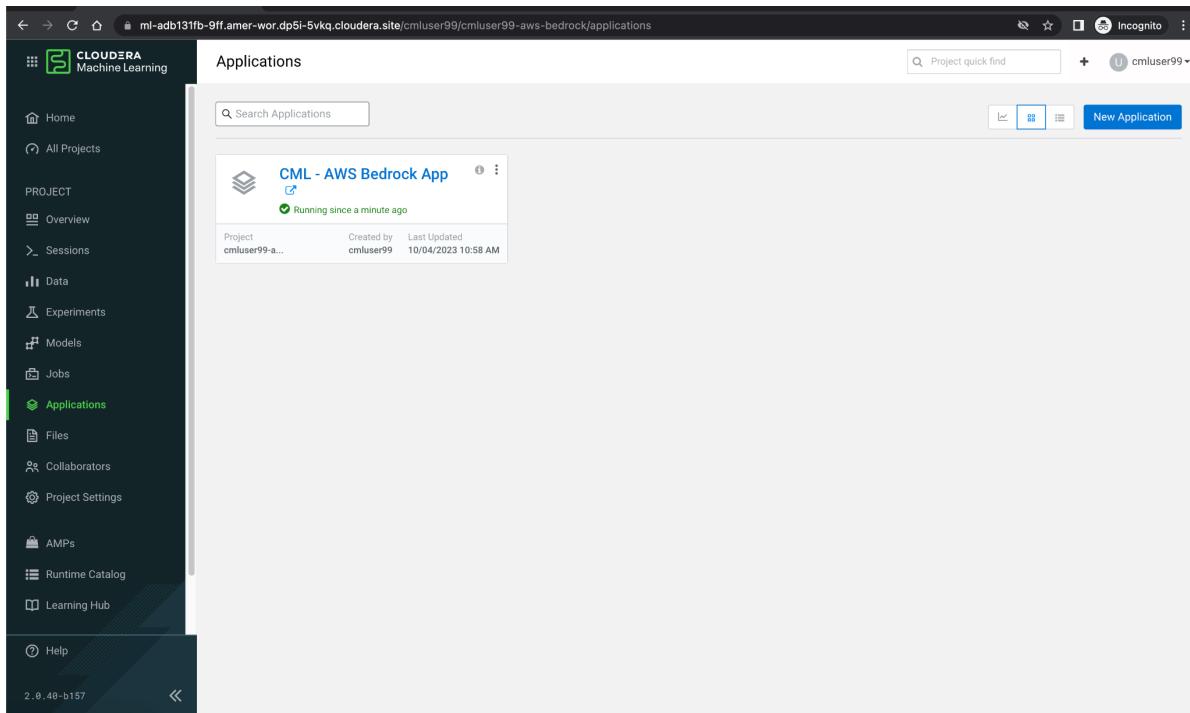
## 8. Create an Application - enter the following details

- Name: Provide your app a name
- Subdomain: make it unique by adding your username
- Script: Select the bedrock-app.py file in the amp\_2\_app folder
- Runtime Editor: Workbench
- Click the Create Application button at the bottom of the page.

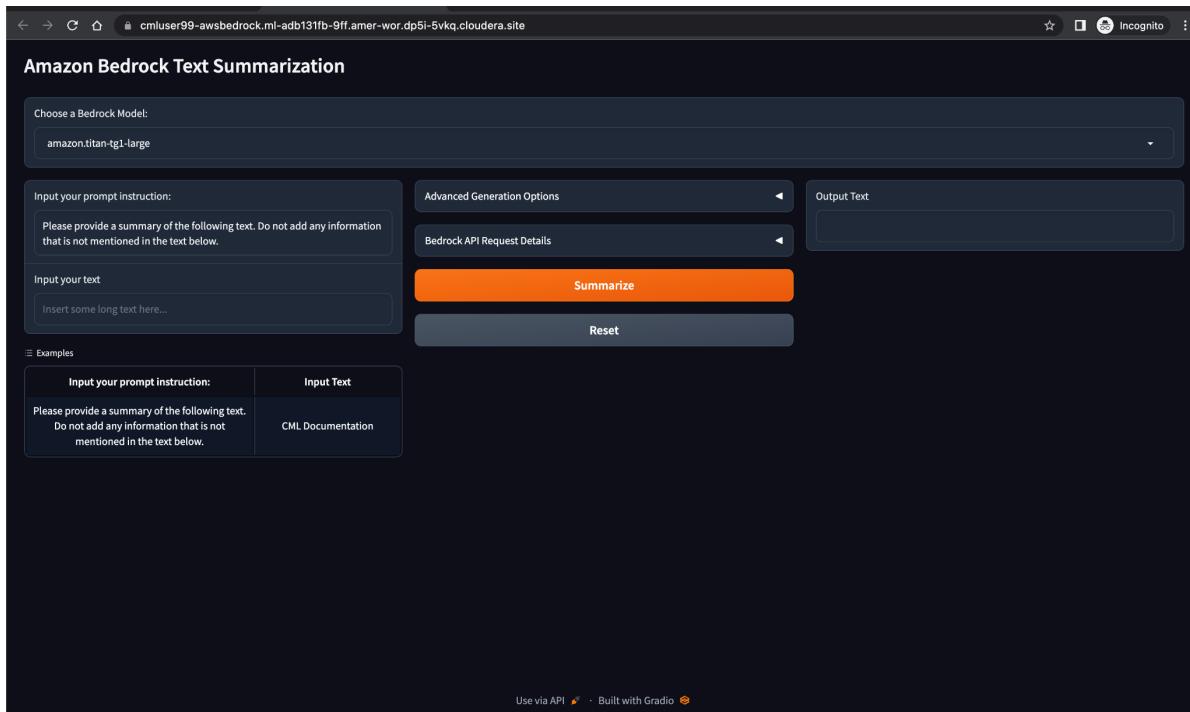
The screenshot shows the 'Create an Application' form in the Cloudera Machine Learning interface. The 'General' tab is selected. The 'Name' field contains 'CML-AWS Bedrock App'. The 'Subdomain' field contains 'cmluser99-awsbedrock'. The 'Description' field is empty. The 'Script' field contains 'amp\_2\_app/bedrock-app.py'. Under the 'Runtime' section, the 'Editor' is set to 'Workbench' and the 'Kernel' is 'Python 3.9'. The 'Edition' is 'Standard' and the 'Version' is '2023.08'. A note says 'Configure additional runtime options in Project Settings.' At the bottom, it shows 'Workspace: amer-cml-ws' and 'Cloud Provider: AWS (AWS)'. The URL in the address bar is [ml-adb131fb-9ff.amer-wor.dp5i-5vkq.cloudera.site/cmluser99/cmluser99-aws-bedrock/applications/new](https://ml-adb131fb-9ff.amer-wor.dp5i-5vkq.cloudera.site/cmluser99/cmluser99-aws-bedrock/applications/new).

The screenshot shows the 'Create an Application' form in the Cloudera Machine Learning interface. The 'Runtime' tab is selected. It shows the same configuration as the previous screenshot: 'Editor' set to 'Workbench' with 'Kernel' 'Python 3.9', 'Edition' 'Standard', and 'Version' '2023.08'. Below this, there is a 'Enable Spark' toggle switch which is turned off, and a dropdown menu showing 'Spark 3.2.3 - CDE 1.19.2 - HOTFIX.2'. Under the 'Runtime Image' section, it lists the Docker image: '- docker.repository.cloudera.com/cloudera/cdsw/ml-runtime-workbench-python3.9-standard:2023.08.2-b8'. The 'Resource Profile' dropdown is set to '2 vCPU / 4 GiB Memory'. In the 'Environment Variables' section, there is a text input field containing 'CDSW\_APP\_POLLING\_ENDPOINT /'. A note says 'Environmental variables will override the project environment.' At the bottom, there is a 'Create Application' button. At the very bottom, it shows 'Workspace: amer-cml-ws' and 'Cloud Provider: AWS (AWS)'. The URL in the address bar is [ml-adb131fb-9ff.amer-wor.dp5i-5vkq.cloudera.site/cmluser99/cmluser99-aws-bedrock/applications/new](https://ml-adb131fb-9ff.amer-wor.dp5i-5vkq.cloudera.site/cmluser99/cmluser99-aws-bedrock/applications/new).

9. Now that you have an application, click on it to open the app.



10. For the Titan Model, click on the **Examples** table. This will populate the “**Input your text**” box
11. Click on the **Summarize** button. This should generate an error message, like when we were testing.



- 12. Choose a Bedrock Model:** Select the Claude Model
  - 13. Click the Summarize button.** You should see the **Output Text**
- Feel free to test with other text*

The screenshot shows a web browser window titled "Amazon Bedrock Text Summarization". The URL is "cmluser99-awsbedrock.ml-adb131fb-9ff.amer-wor.dp5i-5vkq.cloudera.site". The interface has a dark theme with orange and grey accents. On the left, there's a sidebar with "Examples" and two input fields: "Input your prompt instruction:" and "Input Text". The main area starts with "Choose a Bedrock Model:" dropdown set to "anthropic.claude-v2". Below it is a text box with instructions: "Please provide a summary of the following text. Do not add any information that is not mentioned in the text below." A large text area contains the input text: "Machine learning has become one of the most critical capabilities for modern businesses to grow and stay competitive today. From automating internal processes to optimizing the design, creation, and marketing processes behind virtually every product consumed, ML models have permeated almost every aspect of our work and personal lives. ML development is iterative and complex, made even harder because most ML tools aren't built for the entire machine learning lifecycle. Cloudera Machine Learning on Cloudera Data Platform accelerates time-to-value by enabling data scientists to collaborate in a single unified platform that is all inclusive for powering any AI use case. Purpose-built for agile experimentation and production ML workflows, Cloudera Machine Learning manages everything from data preparation to MLOps, to predictive reporting. Solve mission critical ML challenges along the entire lifecycle with greater speed and agility to discover opportunities which can mean the difference for your business. Each ML workspace enables teams of data scientists to develop, test, train, and ultimately deploy machine learning models for building predictive applications all on the data under management within the enterprise data cloud. ML workspaces support fully-containerized execution of Python, R, Scala, and Spark workloads through flexible and extensible engines." To the right, there are three buttons: "Advanced Generation Options", "Bedrock API Request Details", and a prominent orange "Summarize" button. Below these are "Reset" and "Output Text" sections. The "Output Text" section displays the summarized text: "Here is a summary of the key points from the text: - Machine learning has become critical for modern businesses to stay competitive by automating processes and optimizing products. - ML development is complex and iterative. Most ML tools aren't built for the entire lifecycle. - Cloudera Machine Learning accelerates time-to-value by enabling collaboration on a unified platform for any AI use case. It manages everything from data prep to MLOps and reporting. - ML workspaces in Cloudera let teams of data scientists develop, test, train, and deploy models to build predictive apps using Python, R, Scala, Spark, etc."

# Lab 6: Deploy AMP

Now that we've manually created a project, tested code and manually deployed an app, let's deploy a CML AMP from the AMP catalog.

1. Click on **AMPs** on the menu item on the left
2. Click on the **Text Summarization and more with Amazon Bedrock AMP**

The screenshot shows the CML interface with the 'AMPS' menu item highlighted. The main content area displays a grid of pre-built ML projects. The 'Text Summarization and more with Amazon Bedrock' AMP is selected, showing its details: it uses the Amazon Bedrock service and is based on HUGGINGFACE/QLORA. It includes a preview image of a smartphone displaying text summarization results and a detailed description of the project's purpose and components.

3. Click on the **Configure Project** button

The screenshot shows the 'Text Summarization and more with Amazon Bedrock' AMP configuration dialog. The 'Configure Project' button is highlighted in blue at the bottom right of the dialog window. The dialog contains descriptive text about the AMP's purpose and usage, along with tabs for 'View on GitHub' and 'Cancel'.

#### 4. Populate the Environment Variables:

- **AWS\_DEFAULT\_REGION** = us-east-1
- **AWS\_ACCESS\_KEY\_ID** = AKIA6I6SNFMLB7UQIFWL
- **AWS\_SECRET\_ACCESS\_KEY** = 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

#### 5. Click on the **Launch Project** button

Configure Project: Text Summarization and more with Amazon Bedrock - cmluser99

AMP Name: AI Text Summarization with Amazon Bedrock (v1)

This AMP demonstrates an application which uses Amazon Bedrock Models for LLM inference with a customizable summarization usecase.

**Environment Variables**

The settings below were defined by the AMP:

Name	Value	Description
AWS_ACCESS_KEY_ID	aws_key	AWS Key ID. Check the Amazon Bedrock documentation for information about role access
AWS_SECRET_ACCESS_KEY	aws_secret_key	AWS Secret Key
AWS_DEFAULT_REGION	us-west-2	AWS region

**Runtime**

Editor: PBJ Workbench   Kernel: Python 3.9   Edition: Standard   Version: 2023.05

Enable Spark:

**Cancel** **Launch Project**

This will go through a couple of steps to complete. This may take a few minutes.

cmluser99 / Text Summarization and more with Amazon Bedrock - cmluser99 / AMP Status

AMP Name: AI Text Summarization with Amazon Bedrock (v1)

This AMP demonstrates an application which uses Amazon Bedrock Models for LLM inference with a customizable summarization usecase.

Completed 0 of 2 steps

Step 1 Install Dependencies <a href="#">View details</a>	started 10/4/2023 11:02 AM
!pip install --no-cache-dir -r requirements.txt	
Step 2 Start Application	not yet started
Launching Application and UI. Bedrock model invocation will use the configured AWS role.	

6. When complete, go to the Applications section by clicking on the menu item on the left.

The screenshot shows the Cloudera Machine Learning web interface. On the left, there is a sidebar with various project management and machine learning components. The 'Applications' section is highlighted with a green bar at the top. The main content area is titled 'Applications' and contains a search bar and a 'New Application' button. A single application card is displayed, titled 'CML Text Summarization with Amazon Bedrock', which is 'Running since 4 minutes ago'. Below the card, there are columns for Project (Text Summa...), Created by (cmluser99), and Last Updated (10/04/2023 11:06 AM).

7. Feel free to skip, or test as in the previous lab.

The screenshot shows the 'Amazon Bedrock Text Summarization' interface. At the top, it says 'Choose a Bedrock Model:' with a dropdown menu showing 'amazon.titan-tg1-large'. Below that is a section for 'Input your prompt instruction:' containing placeholder text: 'Please provide a summary of the following text. Do not add any information that is not mentioned in the text below.' To the right of this is an 'Advanced Generation Options' dropdown and an 'Output Text' field. In the center, there is a large orange 'Summarize' button. Below the 'Summarize' button is a 'Reset' button. At the bottom left, there is an 'Examples' section with another input field and a 'Input Text' button. At the very bottom of the page, there are links for 'Use via API' and 'Built with Gradio'.

The screenshot shows the Amazon Bedrock Text Summarization interface. At the top, it says "Choose a Bedrock Model:" with "anthropic.claude-v2" selected. Below that, there's a section for "Input your prompt instruction:" containing the text: "Please provide a summary of the following text. Do not add any information that is not mentioned in the text below." To the right of this is an "Advanced Generation Options" dropdown and a "Bedrock API Request Details" dropdown. In the center, there are two buttons: "Summarize" (orange) and "Reset" (grey). On the far left, under "Input your text", there are three sections of text: 1) A general statement about machine learning becoming critical for businesses. 2) A detailed explanation of Cloudera Machine Learning's purpose-built nature for ML development, mentioning its integration with Cloudera Data Platform and support for various AI use cases. 3) A description of ML workspace capabilities, including data management, experimentation, and deployment across Python, R, Scala, and Spark. On the right, under "Output Text", is a summary of the key points from the input text, listing: - Machine learning has become critical for modern businesses to grow and stay competitive by automating processes and optimizing products. - ML development is complex and iterative. Most ML tools aren't built for the entire machine learning lifecycle. - Cloudera Machine Learning accelerates time-to-value by enabling collaboration on a unified platform for any AI use case, managing everything from data preparation to MLOps and reporting. - Cloudera ML workspaces allow teams of data scientists to develop, test, train, and deploy ML models to build predictive applications using Python, R, Scala, Spark, and other tools.

# All Done!

# THANK YOU!

## APPENDIX

### AMP Setup

You must have an AWS account with access to Bedrock and the following environment variables defined with your corresponding account details.

AWS\_DEFAULT\_REGION

AWS\_ACCESS\_KEY\_ID

AWS\_SECRET\_ACCESS\_KEY

If you would like to use a federated role, you can set up BEDROCK\_ASSUME\_ROLE in the Project Environment variables after AMP launch. (Remember to restart the configured CML Application as well.)

### Enable AWS IAM permissions for Bedrock

The AWS identity you set here must have sufficient AWS IAM permissions to call the Amazon Bedrock service.

For example, To grant full bedrock access to your identity, you can:

Open the AWS IAM Console

Find your Role

Select Add Permissions > Create Inline Policy to attach new inline permissions, open the JSON editor and paste in the below example policy:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "BedrockFullAccess",  
            "Effect": "Allow",  
            "Action": ["bedrock:*"],  
            "Resource": "*"  
        }  
    ]  
}
```

### AMP Requirements

#### CPU

- CML CPU workloads with resource profiles up to (2 vCPU / 8 GiB Memory) will be provisioned

#### CML Runtime

- PBJ Workbench - Python 3.9 - 2023.08

main

Preview Code Blame 73 lines (53 loc) · 3.53 KB

Go to file

AMP Setup

You must have an AWS account with access to Bedrock and the following environment variables defined with your corresponding account details.

- AWS\_DEFAULT\_REGION
- AWS\_ACCESS\_KEY\_ID
- AWS\_SECRET\_ACCESS\_KEY

If you would like to use a federated role, you can set up BEDROCK\_ASSUME\_ROLE in the Project Environment variables after AMP launch. (Remember to restart the configured CML Application as well.)

Enable AWS IAM permissions for Bedrock

The AWS identity you set here must have sufficient [AWS IAM permissions](#) to call the Amazon Bedrock service.

For example, To grant full bedrock access to your identity, you can:

- Open the [AWS IAM Console](#)
- Find your [Role](#)
- Select *Add Permissions* > *Create Inline Policy* to attach new inline permissions, open the JSON editor and paste in the below example policy:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "BedrockFullAccess",  
            "Effect": "Allow",  
            "Action": ["bedrock:*"],  
            "Resource": "*"  
        }  
    ]  
}
```

AMP Requirements

CPU

- CML CPU workloads with resource profiles up to (2 vCPU / 8 GiB Memory) will be provisioned

CML Runtime

PBJ Workbench - Python 3.9 - 2023.08

Jupyter Notebook Examples

We have included a notebook example that demonstrate the code in the rest of the AMP project in a more digestible manner.

Text summarization example

A [notebook example](#) is provided to demonstrate what text summarization using the Amazon Bedrock API looks like in a single script.

Documentation · Share feedback