Cloudera AI

Cloudera Accelerators for Machine Learning Projects (AMPs)

Date published: 2020-07-16
Date modified: 2025-10-31

CLOUDƎRA

Legal Notice

Contents

Cloudera Accelerators for Machine Learning Projects

Cloudera Accelerators for Machine Learning Projects (AMPs) provide reference example projects in Cloudera AI. More than simplified quickstarts or tutorials, AMPs are fully-developed expert solutions created by Cloudera's research arm, Fast Forward Labs.

These solutions to common problems in this field demonstrate how to fully use the power of Cloudera AI. AMPs show you how to create Cloudera AI projects to solve your own use cases.

AMPs are available to install and run from the Cloudera AI user interface. As new AMPs are developed, they will become available to you for your study and use.

Note: In an airgapped installation, the default AMPs catalog included at installation and default AMPs may be inaccessible. See AMPs in airgapped environments.

Using AMPs

Get started with AMPs:

1. Log in to your Cloudera AI workbench, and in the left panel click AMPs.
2. Click on an AMP tile to read its description.
3. Click Configure Project and provide any configuration values required by the AMP. The Description field explains how to determine these configuration values. After you click Launch Project, the installation process may take several minutes.
4. When the installation is complete, click Overview to read the documentation for the AMP and explore the code and project structure.

Note: If nothing appears in the AMPs panel, an administrator may need to reconfigure and refresh the catalog. In Site Administration AMPs , click Refresh. The administrator can also refresh periodically to add newly developed AMPs to the panel.

Related Information
Custom AMP Catalog

Creating New AMPs

One great use for AMPs is to showcase reference examples specific to your business by creating your own AMPs in-house. Once a data science project has been built in Cloudera AI, you can package it and have the Cloudera AI Admin add it to the AMP Catalog.

Each individual AMP requires a project metadata file, which defines the environmental resources needed by the AMP, and the setup steps to install the AMP in a Cloudera AI Workbench. See AMP Project Specification for details.

> **Note:** You can store your AMPs in a git repo hosted on Github, Github Enterprise, or GitLab servers (not limited to github.com or gitlab.com.)
>
> Additionally, only simple authentication is supported, such as passing an API key, or including the username and password, as part of the URL. If additional authentication steps are required, then that git host is not supported.

You can also look at an example for Cloudera Accelerators for Machine Learning Projects, such as: .project-metadata.yaml.

Related Information
AMP Project Specification

AMP Project Specification

AMP projects include a project metadata file that provides configuration and setup details. These details may include environment variables and tasks to be run on startup.

YAML File Specification # Version 1.0

The project metadata file is a YAML file. It must be placed in your project's root directory, and must be named .project-metadata.yaml. The specifications for this file are listed below. You can also look at an example for one of the AMPs, such as:.project-metadata.yaml.

Fields

Fields for this YAML file are in snake_case. String fields are generally constrained by a fixed character size, for example string(64) is constrained to contain at most 64 characters. Click Show to see the list of fields.

Field Name	Type	Example	Description
name	string(200)	ML Demo	Required: The name of this project prototype. Prototype names do not need to be unique.
description	string(2048)	This demo shows off some cool applications of ML.	Required: A description for this project prototype.
author	string(64)	Cloudera Engineer	Required: The author of this prototype (can be the name of an individual, team, or organization).

Field Name	Type	Example	Description
date	date string	"2020-08-11"	The date this project prototype was last modified. It shall be in the format: "YYYY-MM-DD" (quotation marks are required).
specification_version	string(16)	0.1	Required: The version of the YAML file specification to use.
prototype_version	string(16)	1.0	Required: The version of this project prototype.
shared_memory_limit	number	0.0625	Additional shared memory in GB available to sessions running in this project. The default is 0.0625 GB (64MB).
environment_variables	environment variables object	See below	Global environment variables for this project prototype.
feature_dependencies	feature_dependencies	See below	A list of feature dependencies of this AMP. A missing dependency in workbench blocks the creation of the AMP.
engine_images	engine_images	See below	Engine images to be used with the AMP. These are recommendations and it does not prevent the user from launching an AMP with non recommended engine images.
runtimes	runtimes	See below	Runtimes to be used with the AMP. These are recommendations and it does not prevent the user from launching an AMP with non recommended runtimes.
tasks	task list	See below	A sequence of tasks, such as running Jobs or deploying Models, to be run after project import.

Example

```
name: ML Demo
description: >-
```

```
This demo shows off some cool applications of ML.
author: Cloudera Engineer
date: '2020-08-11T17:40:00.839Z'
specification_version: 1.0
environment_variables:
...
tasks:
...
```

Environment variables object

The YAML file can optionally define any number of global environment variables for the project under the environment field. This field is an object, containing keys representing the names of the environment variables, and values representing details about those environment variables. Click Show to see the list of fields in the Environment variables object.

Field Name	Type	Example	Description
default	string	"3"	The default value for this environment variable. Users may override this value when importing this project prototype.
description	string	The number of Model replicas, 3 is standard for redundancy.	A short description explaining this environment variable.
required	boolean	true	Whether the environment variable is required to have a non-empty value, the default is false.

Example: This example creates four environment variables.

```
environment_variables:
  AWS_ACCESS_KEY:
    default: ""
    description: "Access Key ID for accessing S3 bucket"
  AWS_SECRET_KEY:
    default: ""
    description: "Secret Access Key for accessing S3 bucket"
    required: true
  HADOOP_DATA_SOURCE:
    default: ""
    description: "S3 URL to large data set"
    required: false
  MODEL_REPLICAS:
    default: "3"
    description: "Number of model replicas, 3 is standard for redundancy"
    required: true
```

Feature Dependencies

AMPs might depend on some optional features of a workbench. The feature_dependencies field accepts a list of such features. Unsatisfied feature dependencies prevent the AMP from being launched in a workbench, and display an appropriate error message. The supported feature dependencies are as follows:

- model_metrics

Runtimes Specification

The runtimes field accepts a list of runtimes objects defined as follows. This Runtimes specification can be added per task or per project.

```
- editor: the_name_of_the_editor # case-sensitive string required. e.g. Work
bench, Jupyter, etc. (how it appears in the UI)
  kernel: the_kernel # case-sensitive string required. e.g. Python 3.6, Pyth
on 3.8, R 3.6, etc. (how it appears in the UI)
  edition: the_edition # case-sensitive string required. e.g. Standard, N
vidia GPU, etc. (how it appears in the UI)
  version: the_short_version # case-sensitive string optional. e.g. 2021.
03, 2021.05, etc. (how it appears in the UI)
  addons: the_list_addons_needed # list of case-sensitive strings optional.
 e.g Spark 2.4.7 - CDP 7.2.11 - CDE 1.13, etc. (how it appears in the UI)
```

This example specifies the Runtimes the Workbench version for Python 3.8.

```
runtimes:
  - editor: Workbench
    kernel: Python 3.8
    edition: Standard
    addons: ['Spark 2.4.7 - CDP 7.2.11 - CDE 1.13']
```

Engine Images Specification

The engine_images field accepts a list of engine_image objects defined as follows:

```
- image_name: the_name_of_the_engine_image # string (required)
  tags: # list of strings (optional)
    - the_tag_of_engine_image
    - ...
```

This example specifies the official engine image with version 11 or 12:

```
engine_images:
  - image_name: engine
    tags:
      - 12
      - 11
```

This example specifies the most recent version of the dataviz engine image in the workbench:

```
engine_images:
  - image_name: cmldataviz
  - image_name: cdswdataviz
```

Note that when specifying CDV images, both cmldataviz and cdswdataviz must be specified. When tags are not specified, the most recent version of the engine image with the matching name is recommended. The following rule is used to determine the most recent engine_image with the matching name:

Official Engine (engine) and CDV (cmldataviz and cdswdataviz) images

Since the officially released engine images follow semantic versioning (where a newer version is always larger than any older version, when compared with >), the most recent engine image is the one with the largest tag. For example, engine:14 will be recommended over engine:13 and cmldataviz:6.3.4-b13 is recommended over cmldataviz:6.2.1-b12.

Custom engine images

There is no way for Cloudera AI to determine the rules for customer custom engine image tags, and therefore there is no reliable way to determine the most recent custom engine image. Use the engine image that has the correct matching name and has the newest ID. The newest ID means that the engine image is the most recently added engine image.

Task list

This defines a list of tasks that can be automatically run on project import. Each task will be run sequentially in the order they are specified in this YAML file. Click Show to see the list of fields.

Field Name	Type	Example	Description
type	string	create_job	Required: The type of task to be executed. See below for a list of allowed types.
short_summary	string	Creating a Job that will do a task.	A short summary of what this task is doing.
long_summary	string	Creating a Job that will do this specific task. This is important because it leads up to this next task.	A long summary of what this task is doing.

Jobs

Create Job

Example

```
- type: create_job
  name: howdy
  entity_label: howdy
  script: greeting.py
  arguments: Ofek 21
  short_summary: Creating a job that will greet you.
  environment_variables:
  SAMPLE_ENVIRONMENT_VARIABLE: CREATE/RUN_JOB
  kernel: python3
```

Click Show to see the list of fields.

Field Name	Type	Example	Description
type	string	create_job	Required: Must be create_job.
name	string	howdy	Required: Job name.
entity_label	string	howdy	Required: Uniquely identifies this job for future tasks, i.e. run_job tasks. Entity labels must be lowercase alphanumeric,

Field Name	Type	Example	Description
			and may contain hyphens or underscores.
script	string	greeting.py	Required: Script for this Job to run.
kernel	string	python3	Required: What kernel this Job shall use. Acceptable values are python2, pyth on3, r, and scala. Note that scala might not be supported for every cluster.
arguments	string	Ofek 21	Command line arguments to be given to this Job when running.
environment_variables	environment variables object	See above	See above
cpu	number	1.0	The amount of CPU virtual cores to allocate for this Job, the default is 1.0.
memory	number	1.0	The amount of memory in GB to allocate for this Job, the default is 1.0.
gpu	integer	0	The amount of GPU to allocate for this Job, the default is 0.
timeout	integer	10	The amount of time in minutes to wait before timing out this Job, the default is 10.
timeout_kil	boolean	false	Whether or not to stop this Job when it times out, the default is false.

Run Job

Example run job task:

```
- type: run_job
  entity_label: howdy
  short_summary: Running the job that will greet you.
  long_summary: >-
    Running the job that will greet you. It will greet you by the name
    which is the first and only command line argument.
```

Most Job run tasks shall just contain the type and entity_label fields. Click Show to see the list of fields.

Field Name	Type	Example	Description
type	string	run_job	Required: Must be run_job.
entity_label	string	howdy	Required: Must match an entity_label of a previous create_job task.

However, they can optionally override previously defined fields. Click Show to see the list of fields.

Field Name	Type	Example	Description
script	string	greeting.py	Required: Script for this Job to run.
kernel	string	python3	Required: What kernel this Job shall use. Acceptable values are python2, python3, r, and scala. Note that scala might not be supported for every cluster.
arguments	string	Ofek 21	Command line arguments to be given to this Job when running.
environment_variables	environment variables object	See above	See above
cpu	number	1.0	The amount of CPU virtual cores to allocate for this Job, the default is 1.0
memory	number	1.0	The amount of memory in GB to allocate for this Job, the default is 1.0.
gpu	integer	0	The amount of GPU to allocate for this Job, the default is 0.
shared_memory_limit	number	0.0625	Limits the additional shared memory in GB that can be used by this Job, the default is 0.0625 GB (64MB).

Models

Note: All models have authentication disabled, so their access key alone is enough to interact with them.

Resources object

Models may define a resources object which overrides the amount of resources to allocate per Model deployment.

Click Show to see the list of fields.

Field Name	Type	Example	Description
cpu	number	1.0	The number of CPU virtual cores to allocate per Model deployment.
memory	number	2.0	The amount of memory in GB to allocate per Model deployment.
gpu	integer	0	The amount of GPU to allocate per Model deployment.

For example:

```
resources:
  cpu: 1
  memory: 2
```

Replication policy object

Models may define a replication policy object which overrides the default replication policy for Model deployments.

Click Show to see the list of fields.

Field Name	Type	Example	Description
type	string	fixed	Must be fixed if present.
num_replicas	integer	1	The number of replicas to create per Model deployment.

For example:

```
replication_policy:
  type: fixed
  num_replicas: 1
```

Model examples list

Models may include examples, which is a list of objects containing a request and response field, each containing a valid object inside, as shown in the example:

```
examples:
  - request:
      name: Ofek
      age: 21
    response:
      greeting: Hello Ofek (21)
  - request:
      name: Jimothy
      age: 43
    response:
      greeting: Hello Coy (43)
```

Click Show to see the list of fields.

Field Name	Type	Example	Description
request	string	See above	Required: An example request object.
response	string	See above	Required: The response to the above example request object.

Create Model

Example:

```
- type: create_model
  name: Say hello to me
  entity_label: says-hello
  description: This model says hello to you
  short_summary: Deploying a sample model that you can use to greet you
  access_key_environment_variable: SHTM_ACCESS_KEY
  default_resources:
  cpu: 1
  memory: 2
```

Click Show to see the list of fields.

Field Name	Type	Example	Description
type	string	create_model	Required: Must be create_model.
name	string	Say hello to me	Required: Model name
entity_label	string	says-hello	Required: Uniquely identifies this model for future tasks, i.e. build_model and deploy_model tasks. Entity labels must be lowercase alphanumeric, and may contain hyphens or underscores.
access_key_environment_variable	string	SHTM_ACCESS_KEY	Saves the model's access key to an environment variable with the specified name.
default_resources	resources object	See above	The default amount of resources to allocate per Model deployment.

Field Name	Type	Example	Description
default_replication_policy	replication policy object	See above	The default replication policy for Model deployments.
description	string	This model says hello to you	Model description.
visibility	string	private	The default visibility for this Model.

Build Model

Example

```
 - type: build_model
   entity_label: says-hello
   comment: Some comment about the model
   examples:
     - request:
         name: Ofek
         age: 21
     response:
       greeting: Hello Ofek (21)
   target_file_path: greeting.py
   target_function_name: greet_me
   kernel: python3
   environment_variables:
     SAMPLE_ENVIRONMENT_VARIABLE: CREATE/BUILD/DEPLOY_MODEL
```

Field Name	Type	Example	Description
type	string	build_model	Required: Must be buil d_model.
entity_label	string	says-hello	Required: Must match an entity_label of a previous create_model task.
target_file_path	string	greeting.py	Required: Path to file that will be run by Model.
target_function_name	string	greet_me	Required: Name of function to be called by Model.
kernel	string	python3	What kernel this Model shall use. Acceptable values are python2, pyth on3, r, and scala. Note that scala might not be supported for every cluster.

Field Name	Type	Example	Description
comment	string	Some comment about the model	A comment about the Model.
examples	model examples list	See above	A list of request/response example objects.
environment_variables	environment variables object	See above	See above

Deploy Model

Example:

```
- type: deploy_model
  entity_label: says-hello
  environment_variables:
  SAMPLE_ENVIRONMENT_VARIABLE: CREATE/BUILD/DEPLOY_MODEL
```

Most deploy model tasks shall only contain the type and entity_label fields. Click Show to see the list of fields.

Field Name	Type	Example	Description
type	string	deploy_model	Required: Must be deploy_model.
entity_label	string	says-hello	Required: Must match an entity_label of a previous deploy_model task.

However, they can optionally override previously defined fields. Click Show to see the list of fields.

Field Name	Type	Example	Description
cpu	number	1.0	The number of CPU virutal cores to allocate for this Model deployment.
memory	number	2.0	The amount of memory in GB to allocate for this Model deployment.
gpu	integer	0	The amount of GPU to allocate for this Model deployment.
replication_policy	replication policy object	See above	The replication policy for this Model deployment.
environment_variables	environment variables object	See above	Overrides environment variables for this Model deployment.

Applications

Start Application

Example:

```
- type: start_application
  subdomain: greet
  script: greeting.py
  environment_variables:
    SAMPLE_ENVIRONMENT_VARIABLE: START_APPLICATION
  kernel: python3
```

Click Show to see the list of fields.

Field Name	Type	Example	Description
bypass_authentication	boolean	True	When enabled, allows unauthenticated access to an application
type	string	start_application	Required: Must be start_application.
subdomain	string	greet	Required: Application subdomain, which must be unique per Application, and must be alphanumeric and hyphen-delimited. Application subdomains are also converted to lowercase.
kernel	string	python3	Required: What kernel this Application shall use. Acceptable values are python2, python3, r, and scala. Note that scala might not be supported for every cluster.
entity_label	string	greeter	Uniquely identifies this application for future tasks. Entity labels must be lowercase alphanumeric, and may contain hyphens or underscores.
script	string	greeting.py	Script for this Application to run.
name	string	Greeter	Application name, defaults to 'Untitled application'.

Field Name	Type	Example	Description
description	string	Some description about the Application	Application description, defaults to 'No description for the app'.
cpu	number	1.0	The number of CPU virutal cores to allocate for this Application.
memory	number	1.0	The amount of memory in GB to allocate for this Application.
gpu	integer	0	The amount of GPU to allocate for this Application.
shared_memory_limit	number	0.0625	Limits the additional shared memory in GB that can be used by this application, the default is 0.0625 GB (64MB).
environment_variables	environment variables object	See above	See above
static_subdomain	boolean	True	When enabled, subdomain will not get randomized.

Experiments

Run Experiment

Example:

```
 - type: run_experiment
   script: greeting.py
   arguments: Ofek 21
   kernel: python3
```

Click Show to see the list of fields.

Field Name	Type	Example	Description
type	string	run_experiment	Required: Must be run_experiment.
script	string	greeting.py	Required: Script for this Experiment to run.
entity_label	string	test-greeter	Uniquely identifies this experiment for future tasks. Entity labels must be lowercase alphanumeric,

Field Name	Type	Example	Description
			and may contain hyphens or underscores.
arguments	string	Ofek 21	Command line arguments to be given to this Experiment when running.
kernel	string	python3	What kernel this Experiment shall use. Acceptable values are python2, python3, r, and scala. Note that scala might not be supported for every cluster.
comment	string	Comment about the experiment	A comment about the Experiment.
cpu	number	1.0	The amount of CPU virtual cores to allocate for this Experiment.
memory	number	1.0	The amount of memory in GB to allocate for this Experiment.
gpu	number	0	The amount of GPU to allocate for this Experiment.

Sessions

Run Sessions

Example:

```
- type: run_session
  name: How to be greeted interactively
  code: |
    import os
    os.environ['SAMPLE_ENVIRONMENT_VARIABLE'] = 'SESSION'

    !python3 greeting.py Ofek 21

    import greeting
    greeting.greet_me({'name': 'Ofek', 'age': 21})
  kernel: python3
  memory: 1
  cpu: 1
  gpu: 0
```

Click Show to see the list of fields.

Field Name	Type	Example	Description
type	string	run_session	Required: Must be run_session.
	string	See above for code, greeting.py for script	Required: Either the code or script field is required to exist for the run Session task, not both. code is a direct block of code that will be run by the Session, while script is a script file that will be executed by the Session.
kernel	string	python3	Required: What kernel this Session shall use. Acceptable values are python2, python3, r, and scala. Note that scala might not be supported for every cluster.
cpu	number	1.0	Required: The amount of CPU virtual cores to allocate for this Session.
memory	number	1.0	Required: The amount of memory in GB to allocate for this Session.
entity_label	string	greeter	Uniquely identifies this session for future tasks. Entity labels must be lowercase alphanumeric, and may contain hyphens or underscores.
name	string	How to be greeted interactively	Session name.
gpu	integer	0	The amount of GPU to allocate for this Session.

Restarting a failed AMP setup

If any task fails after initiating an AMP, you can resume it from the last unsuccessful step using the Resume option. You can use the Redeploy option to re-import and execute tasks based on the modified .project-metadata.yaml file.

Resuming an AMP

The resume action lets you resume the tasks from the last failed step. This eliminates the need to create a new project to launch the AMP again. You can resume an AMP setup process only from the failure state.

Redeploying an AMP

When you redeploy, it stops all the long-running workloads (Model Deployment or Application) linked to the current prototype, deletes the existing tasks, and re-imports tasks from the .project-metadata.yaml file.

- In the Accelerated ML Projects page, after launching a project, in the AMPs Setup Steps window, you can use the Redeploy or Resume option to restart the AMP setup process.