Machine Learning

Using Cloudera AI Inference service

Date published: 2020-07-16
Date modified: 2024-06-11

CLOUDERA

https://docs.cloudera.com/

Legal Notice

Contents

Deploying a Predictive Deep Learning Model..37

Accessing Cloudera AI Inference service Metrics...40

Known issues... 40

Using Cloudera AI Inference service

Cloudera AI Inference service provides a production-grade serving environment for hosting predictive and generative AI. It is designed to handle the challenges of production deployments, such as high availability, performance, fault tolerance, and scalability. The Cloudera AI Inference service allows data scientists and machine learning engineers to deploy their models quickly, without worrying about the infrastructure and maintenance.

Cloudera AI Inference service is built with tight integration of *NVIDIA NIM* and *NVIDIA Triton Inference Server*, providing industry-leading inference performance on NVIDIA GPUs. Model endpoint orchestration and management are built with the *KServe* model inference platform, a Cloud Native Computing Foundation (CNCF) open-source project. The platform provides standard-compliant model inference protocols, such as *Open Inference Protocol* for predictive models and *OpenAI API* for generative AI models.

Cloudera AI Inference service is seamlessly integrated with the *Cloudera Model Registry* enabling users to store, manage, and track their AI models throughout their lifecycle. This integration provides a central location to store model and application artifacts, metadata, and versions, making it easier to share and reuse AI artifacts across different teams and projects. It also simplifies the process of deploying models and applications to production, as users can select artifacts from the registry and deploy them with a single command.

Related Information

KServe

Open Inference Protocol

OpenAI API

NVIDIA NIM

NVIDIA Triton Inference Server

Cloudera Model Registry

Key Features

The key features of Cloudera AI Inference service includes:

- Easy to use interface: Streamlines the complexities of deployment and infrastructure, meaningfully reducing time to value for AI use cases.
- Real-time predictions: Allows users to serve AI models in real-time, providing low latency predictions for client requests.
- Monitoring and logging: Includes functionality for monitoring and logging, making it easier to troubleshoot issues and optimize workload performance.
- Advanced deployment patterns: Includes functionality for advanced deployment patterns, such as canary and blue-green deployments, and supports A/B testing, enabling users to deploy new versions of models gradually and compare their performance before deploying them to production.
- Optimized Performance: Integrates with NVIDIA NIM microservices and NVIDIA Triton Inference Server to accelerate inference performance on NVIDIA accelerated infrastructure.
- Model access: Offers access to NVIDIA foundation models, tailored for NVIDIA hardware to increase inference throughput and to reduce latency.
- REST API: Provides APIs for deploying, managing, and monitoring of model endpoints. These APIs enable integration with continuous integration and continuous deployment (CI/CD) pipelines and other tools used in the Machine Learning Operations (MLOps) and Large Language Model Operations (LLMOps) workflows.

Key Applications

Large Language Models deployed on Cloudera AI Inference service with NVIDIA NIM enable the following applications:

- Chatbots & Virtual Assistants: Empowers bots with human-like language understanding and responsiveness.
- Content Generation & Summarization: Generates high-quality content or distills lengthy articles into concise summaries with ease.
- Sentiment Analysis: Understands user sentiments in real-time, driving better business decisions.
- Language Translation: Breaks language barriers with efficient and accurate translation services.

Terminology

Lists the Cloudera AI Inference service terminology and usage.

- CML Serving App: This is the term used by the CDP CLI to refer to a specific instance of Cloudera AI Inference service.
- Model Endpoint: This refers to a deployed model that has a URL endpoint accessible over the network.
- Model Artifacts: Files stored in Model Registry that are necessary for deploying an instance of the model, such as model weights, metadata, and so on.
- API standard: The protocol that is exposed by a Model Endpoint. It can be either OpenAI (for NVIDIA NIM) or Open Inference Protocol for predictive models.
- CDP Workload Authentication Token: The bearer token used for authentication / authorization when accessing Cloudera AI Inference service API and model endpoints. Throughout this document this is referred to as "CDP_TOKEN".
- Model ID: This is the ID assigned to the model when it is registered to the Cloudera Model Registry.
- Model Version: This is the version of a registered model in the Cloudera Model Registry.

Limitations and Restrictions

Lists the limitations and restrictions when using Cloudera AI Inference service.

- API Stability: Both the Machine Learning Control Plane and Cloudera AI Inference service workload APIs and CLIs are under active development and are subject to change in a backward-incompatible way.
- Cloud Platforms: Cloudera AI Inference service is available on AWS and Azure.
- Supported Instance Types: Cloudera AI Inference service supports the same cloud instance types as those of Cloudera Machine Learning workspaces with a few exceptions. See *Known Issues* for information on unsupported instance types. The type or size of the model you want to deploy determines the cloud compute instance type. Some highly optimized versions of Large Language Models, for instance, work only on specific GPU architectures.
- CLI Only Interface for control plane operations: CDP CLI user interface is available for control plane operations to manage the life cycle of a Cloudera AI Inference service instance.
- No Non-Transparent Proxy Support: Cloudera AI Inference service has not been tested with a non-transparent proxy (NTP) setup in a private cluster. However, it works in a vanilla private cluster.
- No UDR Support in Azure: Cloudera AI Inference service has not been tested with User-Defined Route (UDR) setup of Azure clusters.
- Public Load Balancer: By default, Cloudera AI Inference service uses a private load balancer for cluster ingress. If you use a public load balancer instead, set the use PublicLoadBalancer parameter value to true in the creation payload.

  If you are on AWS and use a private load balancer for cluster ingress, you must have a VPN connection between your corporate network and the Virtual Private Cloud (VPC) in which the Cloudera AI Inference service is deployed. The Cloudera AI Inference service UI requires VPN connection.
- Logging: All Kubernetes pod logs, including pods that are running model servers, are scraped by the platform log aggregator service (fluentd). Model endpoint logs can be viewed from the Cloudera AI Inference service GUI. To view logs of other pods, you must first obtain the kubeconfig of the cluster and use the `kubectl` command. It is not possible to retrieve historical pod logs, and therefore there is no Diagnostic bundle feature at this time.
- Namespace: Model endpoints can only be deployed in the serving-default namespace.

Related Information
Managing Cloudera AI Inference service Endpoints using CDP CLI
Known Issues

Supported Model Artifact Formats

Lists Cloudera AI Inference service supported models:

- Text-generating and embedding Large Language Models (LLMs) packaged as *NVIDIA NIM*.
- Predictive models in the ONNX representation, registered to Cloudera Model Registry from a Cloudera Machine Learning Workspace. See *Register an ONNX model to Model Registry* as an example showing how to convert a sklearn model to ONNX and then register it to the Cloudera Model Registry. Refer to *Export a PyTorch model to ONNX* or *Getting Started Converting TensorFlow to ONNX* documentation regarding how models using these frameworks can be converted to the ONNX representation.

Related Information
Register an ONNX model to Model Registry
Getting Started Converting TensorFlow to ONNX, ONNX Runtime documentation
Export a PyTorch model to ONNX, PyTorch documentation
NVIDIA NIM Large Language Models

Prerequisites for setting up Cloudera AI Inference service

Consider the following prerequisites before setting up Cloudera AI Inference service.

CDP CLI

You must dowload CDP CLI and install it on your machine to create a Cloudera AI Inference service instance. For information on how to download CDP CLI, in the CDP console, click on the left navigation Help Download CLI . You must use CDP CLI version 0.9.123 or higher with Cloudera AI Inference service.

Compute cluster-enabled CDP environment

Cloudera AI Inference service requires a compute cluster-enabled CDP environment. You must either create the compute cluster-enabled environment, or convert your existing environment.

> **Note:** You cannot deploy Cloudera AI Inference service into the **default compute cluster** in the environment. You must create a separate compute cluster.
>
> Cloudera AI Inference service is compatible with either a Containerized Data Lake (CDL) or the VM based Data Lake. If you convert an existing environment to a compute cluster-enabled environment, the environment will retain its existing VM-based data lake.

Data Lake and Cloudera Manager supported versions

JSON Web Token based authentication from Cloudera Machine Learning workspaces to Cloudera AI Inference requires Data Lake version 7.2.18 or above, and Cloudera Manager version 7.12 or above.

Cloudera Model Registry

A Cloudera Model Registry must first be deployed in the same CDP environment in which you plan to deploy Cloudera AI Inference service. That is, Cloudera AI Inference service can only deploy models registered to a Cloudera Model Registry in the same CDP environment. For this release, Cloudera Model Registry version CML2 024.09-1 or higher version must be created before provisioning the Cloudera AI Inference service. If you have an

existing Cloudera Model Registry in the environment, you must first upgrade it to version CML2024.09-1 before provisioning Cloudera AI Inference service.

Related Information
Installing CDP client
Using Compute Clusters in AWS environment
Using Compute Clusters in Azure environment
Setting up Model Registry

Authentication of Cloudera AI Inference service

Cloudera AI Inference service uses CDP Workload Authentication JSON Web Token (JWT) to authenticate users/clients that interact with all HTTP endpoints exposed by the service workload.

External Clients' Authentication

External clients that run outside of CDP data services and that interact with Cloudera AI Inference service API and with model endpoints must obtain a JWT from the CDP control plane, which must be passed as a bearer token in HTTP requests sent to the Cloudera AI Inference service API and model endpoints.

1. Obtain the JWT using CDP CLI.

```
$ CDP_TOKEN=$(cdp iam generate-workload-auth-token --workload-name [***W
orkload name***]
  | jq -r '.token')
```

The same workload name (DE) can be used for authentication to Cloudera AI Inference service as well.

2. Pass CDP_TOKEN in the HTTP request header.

```
$ curl -H "Authorization: Bearer ${CDP_TOKEN}" [***URL***]
```

The token obtained using this method expires in one hour by default. You must obtain a new token if your token is rejected by the server as expired. The server responds with a 401 HTTP response code if the token has expired.

3. Create tokens with longer lifetimes than the default 1 hour using the following CDP command:

```
$ cdp iam set-authentication-policy --workload-auth-token-expiration-sec
  [***EXPIRATION-TIME-IN-SECONDS***]
```

> ⚠️ **Important:** This setting is scoped for the CDP account, it is not specific to Cloudera AI Inference service. Increasing the token expiration has security implications. In contrast with API keys, JWT tokens cannot be revoked. Also note that the above command requires the AUTHENTICATION_POLICY CDP entitlement.

Internal Clients Authentication

Authenticated clients or users running inside another Cloudera Data Platform Data Service, like a Cloudera Machine Learning Workspace, or Hue UI in the same environment, can pass Data Lake issued JWTs for authentication to Cloudera AI Inference service running in the environment. For instance, inside a CML Workspace session, this JWT is available as the access_token field in the /tmp/jwt file.

Authorization of Cloudera AI Inference service

Cloudera AI Inference service implements role-based access control.

To create an instance of the service in a Cloudera Data Platform environment, and to perform modifications to the instance, such as the addition of a node group, the user must have the following roles:

- EnvironmentAdmin
- MLAdmin

Model endpoints can be viewed, created, deleted, and modified by users having EnvironmentUser role along with either one of the following roles:

- MLAdmin
- MLUser

To perform inference on deployed model endpoints, the user must have the MLUser role.

Related Information
Granting CDP Users Access to Cloudera Machine Learning Workspaces

Importing an NVIDIA Foundation Model NIM

Model Hub offers a curated list of top-performing models from the NVIDIA NGC Catalog.

The **Model Hub** page displays the list of different models along with their source type, tags, and description. You can import NVIDIA NGC Catalog models listed on the **Model Hub** page into the **Model Registry**. Imported models can be seen on the **Registered Models** page, and can be deployed using Cloudera AI Inference service.

Related Information
Importing models from NVIDIA NGC

Register an ONNX model to Model Registry

Register your ONNX model to Model Registry.

To deploy a predictive ONNX model, first you must register your model to the Model Registry running in the same environment as your Cloudera AI Inference service.

For information on how to register an ONNX model from a CML Workspace, see *Deploying a Predictive Deep Learning Model*. For more information, see Registering a model using the Model Registry user interface.

Related Information
Deploying a Predictive Deep Learning Model
Registering a model using the Model Registry user interface

Cloudera AI Inference service Concepts

Learn the following Cloudera AI Inference service concepts before setting up the Cloudera AI Inference service.

Platform

Cloudera AI Inference service is a Kubernetes-native model inference platform built using the Cloud Native Computing Foundation (CNCF)-hosted KServe model orchestration system. The Cloudera AI Inference service is built with enterprise-grade scalability, security and performance incorporated via integrations with NVIDIA NIM, NVIDIA Triton, and Cloudera Data Platform. It is designed to serve trained models for production-level use cases.

Runtime

Runtimes are the basic building blocks that are responsible for loading trained model artifacts into memory, and providing APIs that client applications can invoke to run inference requests. The Runtimes also provide metrics with which to monitor the performance of the models. The supported Runtimes in this release include various NVIDIA

NIM versions for text-generation and embedding tasks, and NVIDIA Triton for deep-learning models using the ONNX backend.

Autoscaling

Cloudera AI Inference service provides Cluster Node autoscaling and Model Endpoint autoscaling.

Cluster Node Autoscaling

> If a Cloudera AI Inference service runs on an autoscaling Kubernetes cluster, it can be configured with multiple auto-scaling worker node groups. There are two default node groups that are meant to run certain core services, and an arbitrary number of worker node groups where user workloads are scheduled. You can add or delete worker node groups from the cluster. The autoscaling range of an existing node group in the cluster can be changed.

Model Endpoint Autoscaling

> In addition to worker node autoscaling, Cloudera AI Inference service provides autoscaling at the model endpoint level by increasing or decreasing the number of replicas based on predefined, customizable scaling criteria. Scaling to or from zero replicas is also supported. The supported scaling criteria (or metrics) are Requests Per Second (RPS) and concurrency per replica of the model endpoint. For instance, you can configure your model endpoint to autoscale up if the number of concurrent RPS exceeds 100, so that latency requests are maintained at an acceptable level.

Canary Rollout and Rollback

Cloudera AI Inference service lets you roll out a new version of a model without bringing down the currently running version, using the canary rollout feature. When a new model endpoint is created using version A of a model, 100% of the traffic is routed by the system to the deployed model version. You can roll out another version, say B, under the same endpoint URL and give it a percentage of the traffic, say 10%. If the new version rolls out successfully, the system will send 90% of the traffic to version A, and 10% to version B. If version B does not successfully come up for some reason, the system will continue sending 100% of the traffic to the good version, which is A.

Note that each version of the model that is rolled out consumes the same amount of system resources allocated to each model endpoint replica. Therefore, if each replica is configured to use 2 CPUs and 5Gi of memory, the resource footprint will double when versions A and B are running. If version B turns out to be good, you can send 100% traffic to it and have version A be scaled to 0 replicas to reduce the resource consumption back to that of a single version. If, after sending 100% traffic to version B, you realize that version A is preferred, there is a way to rollback to the previous version. You can set the traffic for version B to 0% and the traffic is automatically sent back to version A. Setting the traffic to 0% rolls back to the last version that was ready with 100% traffic.

Example:

Suppose you begin with version A receiving 100% of the traffic. A new model, version B, is introduced and allocated 25% of the traffic. Upon evaluation, version B underperforms compared to version A, based on the collected metrics. While version B is still live, you proceed to deploy version C with 100% traffic. However, version C also underperforms relative to version A. Instead of redeploying version A manually, you can set version C's traffic to 0%, and all traffic reverts to version A. Note that version B, being an intermediary version that never reached 100% traffic, is not eligible for rollback.

Related Information
[KServer Documentation Website](#)

Cloudera AI Inference service Configuration and Sizing

Consider the following factors for the configuration and sizing of Cloudera AI Inference service.

Node Group Configuration

The configuration and size of Cloudera AI Inference service cluster is determined by the nature of the workloads you expect to deploy on the platform. Certain models might require GPUs, while other kinds of models might run only on CPUs. Similar considerations must be taken for the model endpoints. For example, you must determine the number of replicas of a model that are required to handle normal inference traffic, and there could be peak traffic for which additional replicas shall be spun up to keep user experience at acceptable levels.

> **Note:** Cloudera AI Inference service uses a rolling update strategy for all model endpoints. Consequently, during a rolling update, both the older and newer revisions of the endpoint are running at the same time until the update is completed. Therefore, your cluster must be able to accommodate this expanded resource footprint in order for the rolling update to succeed.

For example, when you deploy the following NVIDIA NIM for Llama 3.1:

In the above image, you can see that each replica of this model requires two A10G GPUs (due to the model being optimized for A10G with a tensor parallelism of two). You can assume that two replicas are required to handle normal traffic, but an additional replica would be required during peak traffic. Therefore, you must configure the endpoint to autoscale between two and three replicas. Consequently, for normal traffic 4 A10G GPUs must be allocated, and two more for peak traffic. To ensure seamless rolling update for this model endpoint, and assuming updates are made during off-peak traffic, your cluster must be able to add eight A10G GPUs on-demand. An optimal node group configuration for this scenario on AWS would be one that has 0-2 instances of the g5.12xlarge instance type. One instance runs two replicas during normal load, and the second instance is added using autoscaling during peak traffic, and during rolling update. Another, less cost-efficient, possibility is to use a single g5.48xlarge instance type in the node group, in which case all the resource headroom is available on a single node. The larger node also helps to ensure that autoscaling model replicas and rolling updates are quicker as you do not have to wait for a new node to be spun up, and the container images to be pulled.

Instance Volume Sizing

Cloudera AI Inference service downloads model artifacts to the instance volume (also known as root volume) of the node where the model replica pod is scheduled. Large Language Model artifacts can be tens to hundreds of gigabytes large. To ensure adequate storage space for these artifacts, Cloudera recommends provisioning at least 1 terabyte of instance volume for those node groups that will be serving LLMs.

> **Note:** The instance volume of an existing node group cannot be altered. You must first delete the node group and then add it back to the cluster with the new instance volume size.

Choosing an NVIDIA NIM Profile

The following guidance here is in the context of Cloudera AI Inference service. See *Nvidia documentation* for information about NVIDIA NIM profiles. NVIDIA NIM comes in three kinds of optimization profiles:

- Latency: This profile minimizes Time to First Token (TTFT) and Inter-Token Latency (ITIL) by using higher tensor parallelism, that is, using more GPUs.
- Throughput: This maximizes the token throughput per GPU by utilizing the minimum number of GPUs to host the model.
- Generic: Unlike the first two profiles, this profile uses the vLLM backend to load the model and run inference against it. This profile provides the most flexibility in terms of choosing GPU models at the cost of lower performance. Note that not all NVIDIA NIM models provide the generic profiles.

For a given precision, the latency profile provides the highest performance by utilizing the maximum number of GPUs while the generic profile offers the most flexibility by sacrificing performance and precision. The throughout profile strikes a good balance between the other two.

Cloudera AI Inference service lets you choose which NVIDIA NIM profile to deploy, so that only artifacts specific to the chosen profile are downloaded to your Registry to save on storage costs.

The choice of profile is determined by the followings:

- Hardware budget - which cloud instance types you have access to.
- Model performance requirement in terms of latency and throughput.

As an example, let us look at some of the available profiles for the instruction-tuned Llama 3.1 8b model. Each entry in the Optimization picker specifies the model name, GPU architecture, floating point precision, and profile type:

Let us compare the difference between A100 BF16 Throughput and A100 BF16 Latency profiles:

* Select Optimization

Llama 3.1 8B Instruct A100 BF16 Latency ⌄

PROFILE	PRECISION	GPU	COUNT
Latency	BF16	A100	2
GPU DEVICE	**NIM VERSION**	**FEAT_LORA**	
20b2:10de	1.0.0	false	

* Select Optimization

Llama 3.1 8B Instruct A100 BF16 Throughput ⌄

PROFILE	PRECISION	GPU	COUNT
Throughput	BF16	A100	1
GPU DEVICE	**NIM VERSION**	**FEAT_LORA**	
20b2:10de	1.0.0	false	

As shown in the figures, the GPU count per model replica of the latency profile is double the size of the throughput profile.

Quantization, for example, FP8 vs BF16. A quantized model, if available, will have lower resource footprint and higher performance (latency and throughput) than a non-quantized one.

The actual latency and throughput as seen by a client application is affected by the end-end performance of the network between the client and the model server, which will include authentication and authorization checks, plus the performance of the chosen NVIDIA NIM profile, as well as the number of concurrent connections. For any chosen profile, end-to-end latency and throughput can be optimized by increasing the number of endpoint replicas for the model.

Related Information
Nvidia Documentation Website

Managing Cloudera AI Inference service Endpoints using CDP CLI

Cloudera AI Inference service provides a CLI interface to manage the life cycle of the service and associated infrastructure.

Prerequisites for Cloudera AI Inference service using CDP CLI

Consider the prerequisites for Cloudera AI Inference service using CDP CLI

Cloudera AI Inference service requires a compute cluster enabled Cloudera Data Platform environment. You must either create the compute cluster enabled environment or convert your existing environment. You must collect the Cloudera Resource Name (CRN) of the cluster and Cloudera Data Platform environment to create Cloudera AI Inference service.

1. Create a compute cluster-enabled CDP environment to prepare the environment for Cloudera AI Inference service.
2. Once the default compute cluster is Running, proceed to the next step.

3. On the environment UI click on the **Compute Clusters** tab and click on the **Add Compute Cluster** button to create a cluster for Cloudera AI Inference.

4. When the cluster diplays Running status, copy the CRN of the cluster.

5. Copy the CRN of your CDP environment, which can be found on the environment UI, as shown in the figure.

Creating a Cloudera AI Inference service instance

The recommended way to create a Cloudera AI Inference service is to first generate the CLI input skeleton, customize the JSON file, and then pass the file to the creation command.

The following example present how to create a Cloudera AI Inference service by generating the CLI input skeleton, customizing the JSON file, and then passing the file to the creation command:

1. Generate the JSON skeleton payload and save it to a file:

```
$ cdp ml create-ml-serving-app --generate-cli-skeleton > /tmp/create-ser
ving-app-input.json
```

2. Customize the JSON file to use it when creating a Cloudera AI Inference service instance. The following are the sample JSON files of AWS and Azure:

AWS JSON file example

```
{
    "appName": "my-aws-caii-cluster",
    "environmentCrn": "[***CDP-ENVIRONMENT-CRN***]",
    "clusterCrn": "[***COMPUTE-CLUSTER-CRN***]",
    "provisionK8sRequest": {
        "instanceGroups": [
            {
                "instanceType": "m5.4xlarge",
                "instanceTier": "ON-DEMAND",
                "instanceCount": 1,
                "name": "[***OPTIONAL-LEAVE BLANK***]",
                "rootVolume": {
                    "size": 256
                },
                "autoscaling": {
                    "minInstances": 0,
                    "maxInstances": 5,
                    "enabled": true
                }
            },
            {
                "instanceType": "p4de.24xlarge",
                "instanceCount": 1,
                "rootVolume": {
                    "size": 1024
                },
                "autoscaling": {
                    "minInstances": 0,
                    "maxInstances": 5,
                    "enabled": true
                }
            }
        ],
        "environmentCrn": "[***CDP-ENVIRONMENT-CRN***]",
        "tags": [
            {
                "key": "experience",
                "value": "cml-serving"
            }
        ]
    },
    "usePublicLoadBalancer": true,
    "skipValidation": false,
    "loadBalancerIPWhitelists": [
        ""
    ],
    "subnetsForLoadBalancers": [
        ""
    ],
    "staticSubdomain": "mydomain"
```

```
    }
```

Azure JSON file example

```
{
    "appName": "my-azure-caii-cluster",
    "environmentCrn": "[***CDP-ENVIRONMENT-CRN***]",
    "clusterCrn": "[***COMPUTE-CLUSTER-CRN***]",
    "provisionK8sRequest": {
        "instanceGroups": [
            {
                "instanceType": "Standard_D4s_v3",
                "instanceCount": 1,
                "rootVolume": {
                    "size": 256
                },
                "autoscaling": {
                    "minInstances": 0,
                    "maxInstances": 5,
                    "enabled": true
                }
            },
            {
                "instanceType": "Standard_ND96asr_A100_v4",
                "instanceCount": 1,
                "rootVolume": {
                    "size": 1024
                },
                "autoscaling": {
                    "minInstances": 0,
                    "maxInstances": 5,
                    "enabled": true
                }
            }
        ],
        "environmentCrn": "[***CDP-ENVIRONMENT-CRN***]",
        "tags": [
            {
                "key": "experience",
                "value": "cml-serving"
            }
        ]
    },
    "usePublicLoadBalancer": true,
    "skipValidation": false,
    "loadBalancerIPWhitelists": [
        ""
    ],
    "subnetsForLoadBalancers": [
        ""
    ],
    "staticSubdomain": "mydomain"
}
```

3. Use the JSON file created in the previous step to create the Cloudera AI Inference service instance:

```
$ cdp ml create-ml-serving-app --cli-input-json file:///tmp/create-servi
ng-app-input.json
```

After a successful invocation of the create command, the CRN of the Cloudera AI Inference service instance that is created is displayed. The command adds the requested compute worker node groups to the existing Kubernetes cluster specified by the clusterCrn field in the request body, and installs the necessary software components.

A typical configuration with two worker node groups would take about 15-20 minutes to complete creation.

Note: In the above example, you set the staticSubdomain parameter to mydomain. Set this parameter, if necessary, to have a custom ingress subdomain for your Cloudera AI Inference service. If you leave this value blank, a randomly generated subdomain name, such as ml-92373b63-5f1, will be created.

Listing Cloudera AI Inference services instances

You can list the Cloudera AI Inference service instances in your Cloudera Data Platform tenant with the help of a command.

Use the following command to list Cloudera AI Inference service instances in your Cloudera Data Platform tenant, across all environments:

```
$ cdp ml list-ml-serving-apps
```

A Cloudera AI Inference service instance that has been created successfully shall show the status as installation:finished similar to the following:

```
{
            "cloudPlatform": "AZURE",
            "appName": "serving-multi-gpu-az",
            "appCrn": "[***APPLICATION-CRN***]",
            "environmentCrn": "[***ENVIRONMENT-CRN***]",
            "environmentName": "eng-ml-env-azure-v2",
            "ownerEmail": "admin@example.com",
            "mlServingVersion": "1.2.0-b72",
            "isPrivateCluster": false,
            "creationDate": "2024-08-01T16:36:32.811000+00:00",
            "cluster": {
                "clusterName": "ml-92373b63-5f1",
                "domainName": "ml-92373b63-5f1.eng-ml-e.xcu2-8y8x.dev.cldr.
work",
                "liftieID": "liftie-wqq856vz",
                "isPublic": false,
                "ipAllowlist": "0.0.0.0/0",
                "authorizedIpRangesAllowList": false,
                "tags": [],
                "instanceGroups": [],
                "clusterCrn": "[***CLUSTER-CRN***]"
            },
            "status": "installation:finished",
            "usePublicLoadBalancer": false,
            "httpsEnabled": true,
            "subnetsForLoadBalancers": []
}
```

Describing Cloudera AI Inference service instance

The describe command shows you all the detailed configuration information about a specific Cloudera AI Inference service, including which node groups are currently configured.

You can describe a Cloudera AI Inference service instance using the following command:

```
$ cdp ml describe-ml-serving-app --app-crn [***APPLICATION-CRN***]
```

The following is a sample output from the above command, edited for brevity:

```
{
    "app": {
        "cloudPlatform": "AZURE",
        "appName": "serving-multi-gpu-az",
        "appCrn": "[***APPLICATION-CRN***]",
        "environmentCrn": "[***ENVIRONMENT_CRN***]",
        "environmentName": "eng-ml-int-env-azure-v2",
        "ownerEmail": "admin@example.com",
        "mlServingVersion": "1.2.0-b69",
        "isPrivateCluster": false,
        "creationDate": "2024-09-03T18:57:16.288000+00:00",
        "cluster": {
            "clusterName": "ml-b0504a3f-bbc",
            "domainName": "ml-b0504a3f-bbc.eng-ml-i.svbr-nqvp.int.cldr.wor
k",
            "liftieID": "liftie-vjyrtsfn",
            "isPublic": false,
            "ipAllowlist": "0.0.0.0/0",
            "authorizedIpRangesAllowList": false,
            "tags": [],
            "instanceGroups": [
             ...
            ],
            "clusterCrn": "crn:[***CLUSTER-CRN***]"
        },
        "status": "installation:finished",
        "usePublicLoadBalancer": false,
        "httpsEnabled": true,
        "subnetsForLoadBalancers": []
    }
}
```

Managing Node Groups

You can add or delete node groups to your cluster.

Adding a Node Group to an existing instance

You can add one or more node groups to your cluster if you do not have the right worker node hardware (for example, incorrect GPU models for a new workload) in your existing cluster configuration.

Procedure

1. Generate the JSON skeleton to create the instance:

```
$ cdp ml add-instance-groups-ml-serving-app --generate-cli-skeleton > /t
mp/add-instance-group.json
```

2. Edit the file to add the node group details:

```
{
    "appCrn": "[***APPLICATION-CRN***]",
    "instanceGroups": [
        {
            "instanceType": "g5.8xlarge",
            "instanceCount": 0,
            "rootVolume": {
                "size": 1024
            },
            "autoscaling": {
                "minInstances": 0,
                "maxInstances": 4,
                "enabled": true
            }
        },
        {
            "instanceType": "p3.8xlarge",
            "instanceCount": 0,
            "rootVolume": {
                "size": 1024
            },
            "autoscaling": {
                "minInstances": 0,
                "maxInstances": 4,
                "enabled": true
            }
        }
    ]
}
```

3. Use the JSON file created in the previous step to add the node group:

```
$ cdp ml add-instance-groups-ml-serving-app --cli-input-json file:///tmp/
add-instance-group.json
```

Modifying a node group in an existing instance

Follow the instructions to reconfigure an existing node group.

You can reconfigure an existing node group to change its autoscaling range, you can use the following command:

```
$ cdp ml modify-ml-serving-app --app-crn [***APPLICATION-CRN***] --instance-
group-name [***GROUP-NAME***] --min [***MINIMUM-NODE***] --max [***MAXIMUM-N
ODE***] --instance-type [***INSTANCE-TYPE***]
```

Consider the following example:

```
$ cdp ml modify-ml-serving-app --app-crn <your-application-crn> --instance-g
roup-name mlgpu1 --min 2 --max 4 --instance-type g5.24xlarge
```

The instance group name can be obtained using the `describe-ml-serving-app` command.

Related Information

Describing Cloudera AI Inference service instance

Deleting a node group from an existing Cloudera AI Inference service instance

Consider the guidelines here to delete a node group from an existing Cloudera AI Inference service instance.

Use the following command to delete a worker node group:

```
$ cdp ml delete-instance-group-ml-serving-app
```

Consider the following example:

```
$ cdp ml delete-instance-group-ml-serving-app --app-crn <YOUR-APP-CRN> --ins
tance-group-name mlgpu2
```

Related Information
Describing Cloudera AI Inference service instance

Deleting Cloudera AI Inference service instance

Consider the following instructions for deleting Cloudera AI Inference service instance.

You can delete a Cloudera AI Inference service instance if it is no longer needed:

```
$ cdp ml delete-ml-serving-app --app-crn <app-crn>
```

Note: The above command only deletes the Cloudera AI Inference service and its associated cloud resources, such as cluster node groups and load balancers. The underlying compute cluster will not be deleted. If you want to delete the compute cluster itself once Cloudera AI Inference service has been deleted, you can do it from the **Compute Clusters** tab in the environment UI.

Caution: Always delete Cloudera AI Inference service first before deleting the underlying compute cluster.

Obtaining Control Plane Audit Logs for Cloudera AI Inference service

Consider the following instructions for obtaining Control plane audit logs for Cloudera AI Inference.

Use the following commands to obtain audit log entries and the logs therein for troubleshooting purposes:

```
cdp ml get-audit-events --resource-crn [***APPLICATION-CRN***]
cdp ml get-logs --request-id [***REQUEST-ID***] --resource-crn [***APPLIC
ATION-CRN***]
```

Obtaining the kubeconfig for the Cloudera AI Inference service Cluster

Troubleshoot Kubernetes-level issues in the cluster.

To troubleshoot Kubernetes-level issues in the cluster, such as installation or upgrade failure, or any other cluster level issue, you must first obtain the kubeconfig of the cluster.

Related Information
Granting Remote Access to ML Workspaces

Obtaining kubeconfig on AWS

Follow the guidelines for obtaining kubeconfig on AWS.

About this task

You need the user's Amazon Resource Name (ARN) to obtain the kubeconfig file. You can get the ARN from the user or look up a user's ARN in your AWS account. To obtain a user's ARN from the AWS account, complete the following steps:

Procedure

1. Obtain the Amazon Resource Name (ARN) .

   a) Using AWS UI, go to your organization's AWS Account Identity and Access Management (IAM) Users and look up the user. The ARN is available on the **Summary** page.

   b) Using AWS CLI, run the following command to get the ARN:

   ```
   $ aws sts get-caller-identity
   ```

   Consider the following example:

   ```
   # Sample output
   {
   "UserId": "ABCDE12345FGHIJKLMNO6789",
   "Account": "888888888888",
   "Arn": "arn:aws:iam::888888888888:user/joesmith"
   }
   ```

2. Obtain your Cloudera AI Inference service CRN, run the following command:

   ```
   $ cdp ml list-ml-serving-apps
   ```

3. Grant access to the Cloudera AI Inference service AWS cluster, run the following command:

   ```
   $ cdp ml grant-ml-serving-app-access \
      --resource-crn [***APPLICATION-CRN***] \
      --identifier [***AWS_ARN***]
   ```

4. Obtain the kubeconfig used to access the above cluster by running:

   ```
   $ cdp ml get-ml-serving-app-kubeconfig \
      --app-crn [***APPLICATION-CRN***] \
      | jq '.kubeconfig' | yq > myconfig.yaml
   ```

5. Send the downloaded Kubernetes configuration file to the user who has been granted access. To connect to the Amazon EKS cluster, you must have aws-iam-authenticator installed.

6. Get the logs of the Cloudera AI Inference service API server:

   ```
   $ KUBECONFIG=myconfig.yaml kubectl get logs deployment/api -n cml-serving
   ```

Obtaining kubeconfig on Azure

Consider the following guidelines for obtaining kubeconfig on Azure.

Procedure

Locate the liftie cluster identifier for your Cloudera AI Inference service by issuing a describe command using Cloudera Data Platform CLI:

```
$ cdp ml describe-ml-serving-app --app-crn [***APPLICATION-NAME***]
```

```
{
    "app": {
        "appName": "[***APPLICATION-NAME***]",
        "appCrn": "[***APPLICATION-CRN***]",
        "environmentCrn": "[***ENVIRONMENT-CRN***]",
        "environmentName": "[***ENVIRONMENT-NAME***]",
        "ownerEmail": "user@org.com",
        "mlServingVersion": "1.2.0-b72",
        "isPrivateCluster": false,
```

```
            "creationDate": "2024-05-07T14:27:16.817000+00:00",
            "cluster": {
                "clusterName": "ml-1fcaa8cf-a94",
                "domainName": "[***DOMAIN-NAME***]"
                "liftieID": "liftie-3544nrdr",
                "isPublic": false,
                "ipAllowlist": "0.0.0.0/0",
                "authorizedIpRangesAllowList": false
            },
            "status": "installation:finished",
            "usePublicLoadBalancer": false,
            "httpsEnabled": true
        }
    }
```

The cluster identifier is in app.cluster.liftieID.

Locate this entry in the Azure Portal's Kubernetes services page and do the following:

a) Open the entry.

b) Click on **Connect**.

c) Copy the command under **Download cluster credentials** to your terminal and execute.

   You are now authenticated and your kubectl context is set up to interact with the AKS cluster.

Managing Cloudera AI Inference service Endpoints using UI

Cloudera AI Inference service lets you deploy models saved in the Cloudera Model Registry.

Cloudera AI Inference service provides a graphical user interface accessible from the CML Control Plane for managing model endpoints. HTTP/REST APIs are also available for programmatic model endpoint management.

Cloudera AI Inference service UI provides new features to view, create, and edit model endpoints from the control plane UI. Each Cloudera AI Inference service instance can have multiple deployed model endpoints. The Model Endpoints page provides a comprehensive list view of all the deployed model endpoints in a given Cloudera Data Platform account.

Related Information

Supported Model Artifact Formats

Creating a Model Endpoint using UI

The **Create Model Endpoint** page allows you to select a specific Cloudera AI Inference service instance and a model version from Cloudera Model Registry to create a new model endpoint.

About this task

The following steps illustrate how to create a Llama 3.1 model endpoint.

Procedure

1. In the **Cloudera Data Platform** (CDP) console, click the Machine Learning tile.

   The **Machine Learning Workspaces** page displays.

2. Click Model Endpoints under Deployments on the left navigation menu.

   The **Model Endpoints** landing page is displayed.

3. Click the Create Endpoint button.

The **Create Endpoint** page displays the details to create an endpoint.

4. In the Select Environment & Cluster dropdown list, select your Cloudera Data Platform environment and the Cloudera AI Inference service instance within which you want to create the endpoint.

5. In the Name textbox, enter a name for the endpoint.

6. In the Description textbox add a short description of the endpoint. Ensure that the description text you enter is below 5000 characters.

7. Under Served Model Builder, select the model (Model), and modelversion (Version) from the dropdown you want to deploy. Using the Traffic slider bar, specify the traffic split between different model versions that you deploy. It is always set to 100% for the first model version which you cannot change.

8. Under Resource Profile, select the type of the instance from the Instance Type dropdown list. For *NVIDIA NIM* models this field is mandatory. Specify which compute node instance type you wish to run on your model replicas. The instance type you choose depends on the capabilities of the instance type and on what is required by the NVIDIA NIM. The field is optional for normal predictive model endpoints.

9. In the GPU field, specify the number of GPUs each model endpoint replica requires. This field gets populated automatically for NVIDIA NIM models.

10. In the CPU field, specify the number of CPUs each model endpoint replica requires.

11. In the Memory field, specify the amount of CPU memory each model endpoint replica requires.

12. Using the Autoscale Range slider bar, specify the minimum and maximum number of replicas for the model endpoint. Based on which autoscaling parameter you choose, the system scales the number of replicas to meet the incoming load.

13. Under Advanced Options, in the Autoscale Metric Type select the autoscale metric type as Requests per second (RPS) or Concurrency per replica of the endpoint.
   If you choose to scale as per RPS and the Target Metric Value is set to 200, then the system automatically adds a new replica when it sees that a single replica is handling 200 or more requests per second. If the RPS falls below 200, then the system scales down the endpoint by terminating a replica.

14. In Tags, add any custom key and value pairs for your own use.

15. Click Create Endpoint to create the endpoint.

   It can take tens of minutes for the endpoint to be ready. The time taken is determined by the following points:

   • whether a new node has to be scaled-in to the cluster.
   • the time taken to pull the necessary container images.
   • the time taken to download model artifacts to the cluster nodes from the Model Registry.

   In the above example, we are creating an endpoint for the instruct variant of Llama 3.1 8B, which has been optimized to run on two NVIDIA A10G GPUs per replica. The cluster we are deploying into is in AWS, and it has a node group consisting of g5.12xlarge instance types. In this example there are no other node groups that have instance types containing A10G GPUs, so the g5.12xlarge instance type is the only choice. For NVIDIA NIM models that specify GPU models and count, the GPU field of the resource configuration page will be filled in automatically by the UI.

   ⚠️ **Important:** We set the autoscale range as 1-2, with the scaling metric set to concurrency. The scale metric target is set to 100, meaning that if the number of concurrent requests per replica exceeds 100, the system will automatically add a new replica, up to a maximum of two replicas. You must ensure that your cluster has sufficient capacity to expand the requisite node group to run the desired number of replicas.

Listing Model Endpoints using UI

The **Model Endpoints** landing page displays a table view that represents a comprehensive list of all the model endpoints across the running Cloudera AI Inference instances in your Cloudera Data Platform account.
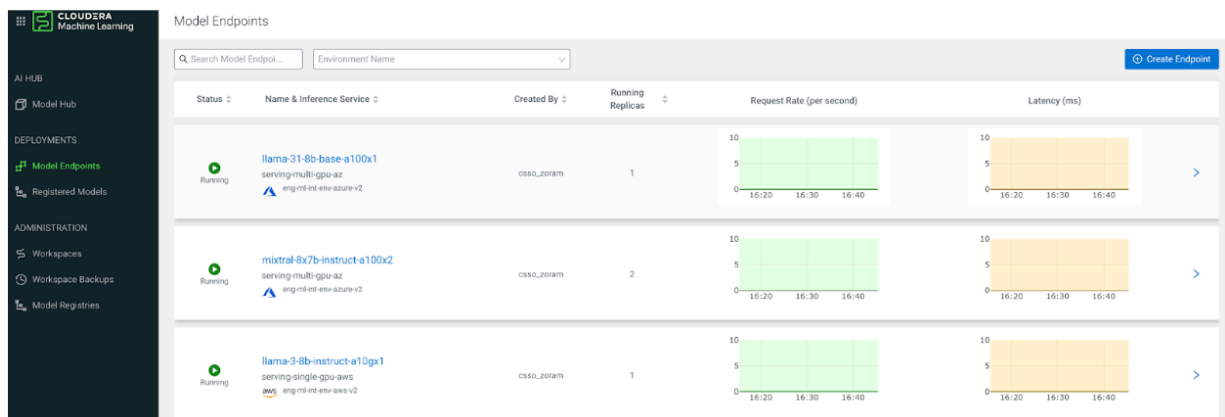
Procedure

1. In the **Cloudera Data Platform** (CDP) console, click the Machine Learning tile.

   The **Machine Learning Workspaces** page displays.

2. Click Model Endpoints under Deployments on the left navigation menu.

The **Model Endpoints** landing page is displayed.

Viewing details of a Model Endpoint using UI

You can view the information of a Model Endpoint, edit its configuration, view metric charts on resource utilization, and so on.

Procedure

1. In the **Cloudera Data Platform** (CDP) console, click the Machine Learning tile.

The **Machine Learning Workspaces** page displays.

2. Click Model Endpoints under Deployments on the left navigation menu.

The **Model Endpoints** landing page is displayed.

3. Select a model endpoint to see its details.

Editing a Model Endpoint Configuration using UI

You can edit the configuration of a specific model endpoint for resources or select model versions.
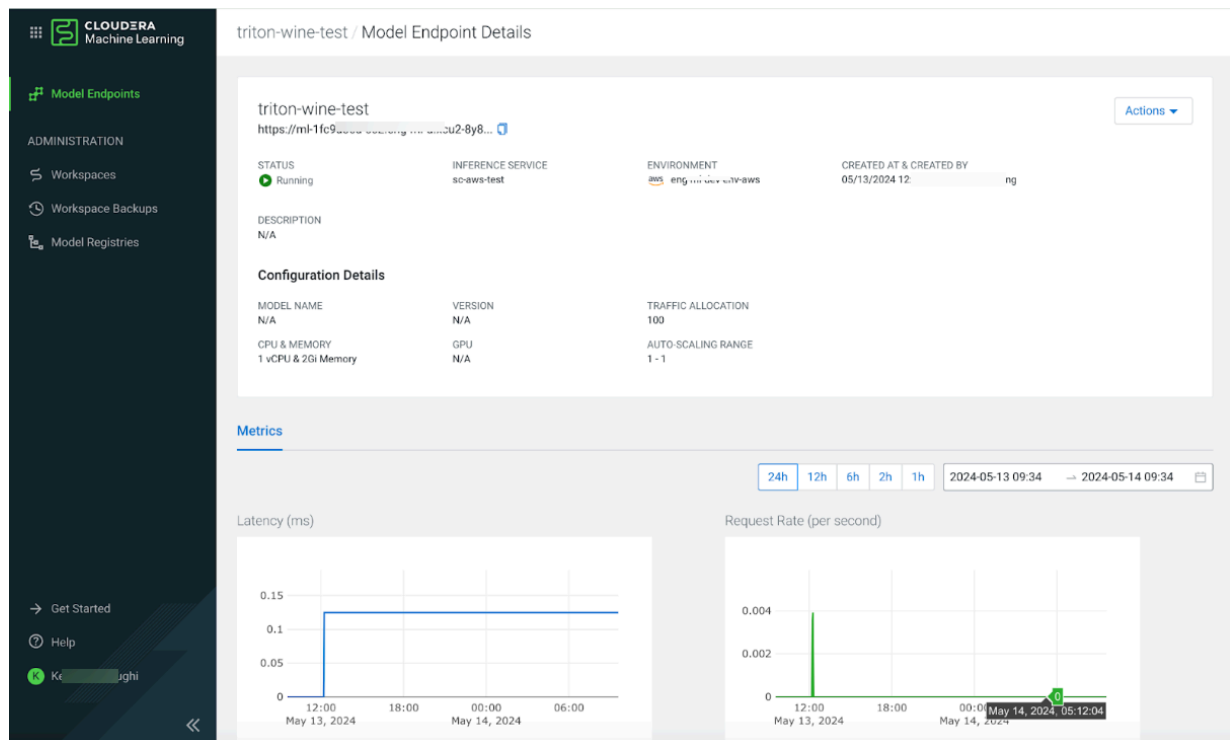
Procedure

1. In the **Cloudera Data Platform** (CDP) console, click the Machine Learning tile.

   The **Machine Learning Workspaces** page displays.

2. Click Model Endpoints under Deployments on the left navigation menu.

   The **Model Endpoints** landing page is displayed.

3. Select a model endpoint that you want to edit.

4. Select Edit Configuration from the Actions dropdown list.

5. Edit the resource profile or add model versions.

   a. In the Resource Profile tab, specify the CPU, Memory, GPU, and Autoscale range.
   b. In the Served Models tab, add new model versions and specify the traffic percentage for the model versions.

6. Click Update.

Managing Cloudera AI Inference service Endpoints using API

You can interact with the Cloudera AI Inference service API using an HTTP/REST client, such as cURL.

Related Information
Curl documentation

Preparing to interact with the Cloudera AI Inference service API

To interact with Cloudera AI Inference service API, you need to obtain the domain name of the Cloudera AI Inference service and your Cloudera Data Platform (CDP) JSON Web Token (JWT) and save it as environment variables.

About this task

⚠️ **Important:** The below commands use the `jq` command, which must be installed on your system.

Procedure

1. Obtain the domain name of the Cloudera AI Inference service and CDP JSON Web Token (JWT) from the output of `list-ml-serving-apps` or `describe-ml-serving-app` CDP commands.

```
export DOMAIN=$(cdp ml describe-ml-serving-app --app-crn [***APP CRN***] |
  jq -r '.app.cluster.domainName')
```

2. Store your CDP_TOKEN as an environment variable:

```
export CDP_TOKEN=$(cdp iam generate-workload-auth-token --workload-name DE
  | jq -r '.token')
```

Creating a Model Endpoint using API

You can select a specific Cloudera AI Inference service instance and a model version from Cloudera Model Registry to create a new model endpoint.

Procedure

1. Retrieve a registered model's model ID and model version.

   This information is available in the Registered Models page in the CML control plane UI.

2. Create the model specification for the selected model.

```
# cat ./examples/mlflow/model-spec-cml-registry.json
```

```
{
  "namespace": "serving-default",
  "name": "mlflow-wine-test-from-registry-onnx",
  "source": {
    "registry_source": {
      "version": 2,
      "model_id": "3azn-tmqe-wsze-5u4s"
    }
  }
}
```

```
export DOMAIN=$(cdp ml describe-ml-serving-app --app-crn [***app-crn***] |
 jq -r '.app.cluster.domainName')
```

3. Create the model endpoint by using the following Cloudera Machine Learning serving `deployEndpoint` API:

```
curl -v  -H "Content-Type: application/json" -H "Authorization: Bearer $
{CDP_TOKEN}" "https://${DOMAIN}/api/v1alpha1/deployEndpoint" -d @./examp
les/mlflow/model-spec-cml-registry.json
```

The DOMAIN looks like ml-67814ad5-b79.eng-ml-d.xcu2-8y8x.dev.cldr.work.

You can retrieve the CDP_TOKEN by performing the steps from Preparing to interact with the Cloudera AI.

Note:

You can only specify serving-default as the namespace into which the Model Endpoint can be deployed.

Related Information

Register an ONNX model to Model Registry

Preparing to interact with the Cloudera AI Inference service API

Listing Model Endpoints using API

Consider the following details for listing Model Endpoints using API.

Procedure

List model endpoints with the help of the following command:

```
curl -H "Content-Type: application/json" -H "Authorization: Bearer ${CDP_TOK
EN}" "https://${DOMAIN}/api/v1alpha1/listEndpoints" -d '{
  "namespace": "serving-default"
}'

{"endpoints":[
  {
    "namespace":"serving-default",
    "name":"mlflow-wine-test-from-registry-onnx",
    "url":"https://[***domain***]/namespaces/serving-default/endpoints/[*
**endpoint_name***]/v2/models/[***model_name***]/infer",
    "state":"Loaded"
  }
]}%
```

Describing a Model Endpoint using API

Consider the instructions for describing a Model Endpoint using API.

Procedure

Use the following command to describe a given model endpoint. The `describeEndpoint` API shows you all the detailed configuration and runtime snapshot of the endpoint. The output below has been elided for brevity.

```
curl -s -H "Authorization: Bearer ${CDP_TOKEN}" -H "Content-Type: applicatio
n/json" https://${DOMAIN}/api/v1alpha1/describeEndpoint -d '{"namespace": "s
erving-default", "name": "[***ENDPOINT_NAME***]"}'|jq
{
  "namespace": "serving-default",
  "name": "[***ENDPOINT_NAME***]",
  "url": "https://[***DOMAIN***]/namespaces/serving-default/endpoints/[***E
NDPOINT_NAME***]/v2/models/[***MODEL_ID***]/infer",
  "conditions": [
    {
      "type": "IngressReady",
      "status": "True",
      "severity": "",
      "last_transition_time": "1726713090",
      "reason": "",
      "message": ""
    },

  "observed_generation": 1,
  "replica_count": 1,

  ...

  "autoscaling": {
    "min_replicas": "1",
    "max_replicas": "80",
    "autoscalingconfig": {
      "metric": "concurrency",
      "target": "20",
    }
  },

 ...

  "api_standard": "oip",
  "has_chat_template": false,
  "metricFormat": "triton",
  "task": "inference"
}
```

Related Information

Nvidia documentation: Inputs and outputs

Preparing to interact with the Cloudera AI Inference service API

Making an inference call to a Model Endpoint with an OpenAI API

Language models for text generation are deployed using NVIDIA's NIM microservices. These endpoints are compliant with the *OpenAI Protocol*. See NVIDIA NIM documentation for supported OpenAI APIs and NVIDIA NIM specific extensions such as Function Calling and Structured Generation.

Related Information

OpenAI Platform protocol

Nvidia documentation: API Reference

Inference using OpenAI Python SDK client in a CML Workspace Session

Consider the following instructions on interacting with an instruction-tuned large language model endpoint hosted on Cloudera AI Inference service.

Procedure

You can use the following template to interact with an instruction-tuned large language model endpoint hosted on Cloudera AI Inference service:

```
from openai import OpenAI
import json

API_KEY = json.load(open("/tmp/jwt"))["access_token"]
MODEL_ID = "[***MODEL_ID***]"

client = OpenAI(
  base_url = "[***BASE_URL***]",
  api_key = API_KEY,
  )

completion = client.chat.completions.create(
  model=MODEL_ID,
  messages=[{"role":"user","content":"Write a one-sentence definition of
GenAI."}],
  temperature=0.2,
  top_p=0.7,
  max_tokens=1024,
  stream=True
)

for chunk in completion:
  if chunk.choices[0].delta.content is not None:
    print(chunk.choices[0].delta.content, end="")
```

Where base_url is the endpoint URL up to the API version v1. To get the base_url, copy the endpoint URL and delete the last two path components.

Copy the endpoint URL from the **Model Endpoint Details** UI and modify it to

```
https://[***DOMAIN***]/namepaces/serving-default/endpo
ints/[***ENDPOINT_NAME***]/v1
```

MODEL_ID is the ID assigned to the model when it is registered to the Cloudera Model Registry. You can find this in the **Model Endpoint Details** UI.

Inference using OpenAI Python SDK client on a local machine

Consider the following guidelines for Inference using OpenAI Python SDK client.

Procedure

Specify the CDP_TOKEN as the `API_KEY`. A common way is to export CDP_TOKEN as an environment variable and access that from your Python code:

```python
from openai import OpenAI
import os
API_KEY = os.environ['CDP_TOKEN']
MODEL_ID = [***MODEL_ID***]

client = OpenAI(
  base_url = "[***BASE_URL***]",
  api_key = API_KEY,
  )

completion = client.chat.completions.create(
  model=MODEL_ID,
  messages=[{"role":"user","content":"Write a one-sentence definition of
GenAI."}],
  temperature=0.2,
  top_p=0.7,
  max_tokens=1024,
  stream=True
)

for chunk in completion:
  if chunk.choices[0].delta.content is not None:
    print(chunk.choices[0].delta.content, end="")
```

Related Information
Preparing to interact with the Cloudera AI Inference service API
Nvidia documentation: API Reference
Authentication of Cloudera AI Inference service - External Clients Authentication

OpenAI Inference Protocol Using Curl

Consider this example for OpenAI Inference Protocol Using Curl.

An example inference payload for the OpenAI Protocol:

```
# cat ./llama-input.json

{
    "messages": [
        {
            "content": "You are a polite and respectful chatbot helping peop
le plan a vacation.",
            "role": "system"
        },
        {
```

```
            "content": "What should I do for a 4 day vacation in Spain?",
            "role": "user"
        }
    ],
    "model": "6dwc-5h3v-aza0-l823",
    "max_tokens": 200,
    "top_p": 1,
    "n": 1,
    "stream": false,
    "stop": "\n",
    "frequency_penalty": 0.0
}
```

```
curl -H "Content-Type: application/json" -H "Authorization: Bearer ${CDP_TOK
EN}" "https://${DOMAIN}/namespaces/serving-default/endpoints/llama-3-1/v1/ch
at/completions" -d @./llama-input.json

Spain offers a diverse range of experiences, making it perfect for a 4-day v
acation…
```

Making an inference call to a Model Endpoint with Open Inference Protocol

Cloudera AI Inference service serves predictive ONNX models using the Nvidia Triton Server. The deployed endpoints are compliant with Open Interface Protocol version 2.

Note: The model inference request and response payloads are different from models deployed on a CML workspace.

Related Information

Open Inference Protocol Specification

Nvidia Triton Inference Server

Open Inference Protocol Using Python SDK

Use the following code sample to interact with a model endpoint using the Open Inference Protocol.

Procedure

Run the following command to customize the ENDPOINT_NAME, DOMAIN, and MODEL_ID placeholders. The inference request payload shape must match what your model is expecting, which you can determine with the read_model_metadata() function.

```
#!pip install open-inference-openapi

from open_inference.openapi.client import OpenInferenceClient, InferenceRe
quest
import httpx
import json
import os

#
# inspired by https://pypi.org/project/open-inference-openapi/
#

CDP_TOKEN = os.environ['CDP_TOKEN']
BASE_URL = 'https://[***DOMAIN***]/namespaces/serving-default/endpoints/[*
**ENDPOINT_NAME***]'
MODEL_NAME = '[***MODEL_ID***]'
headers = {'Authorization': 'Bearer ' + CDP_TOKEN,
```

```
            'Content-Type': 'application/json'}

httpx_client = httpx.Client(headers=headers)
client = OpenInferenceClient(base_url=BASE_URL, httpx_client=httpx_client)
# Check that the server is live, and it has the model loaded
client.check_server_readiness()
metadata = client.read_model_metadata(MODEL_NAME)
metadata_str = json.dumps(json.loads(metadata.json()), indent=2)
print(metadata_str)

# Make an inference request
pred = client.model_infer(
    MODEL_NAME,
    request=InferenceRequest(
        inputs=[
            {
            "name": "float_input",
            "shape": [1, 11],
            "datatype": "FP32",
            "data": [7.4] * 11  # This is a shorter way to write the same
 data
            }
        ]
    ),
)

json_resp_str = json.dumps(json.loads(pred.json()), indent=2)
print(json_resp_str)
```

Open Inference Protocol Using Curl

Consider the following instructions for using Open Interference Protocol Using Curl.

Procedure

You can construct the input of the inference call based on the Open Inference Protocol V2:

```
# cat ./examples/wine-input.json

{
  "parameters": {
    "content_type": "pd"
  },
  "inputs": [
    {
      "name": "float_input",
      "shape": [
        1,
        11
      ],
      "datatype": "FP32",
      "data": [
        7.4,
        7.4,
        7.4,
        7.4,
        7.4,
        7.4,
        7.4,
        7.4,
        7.4,
        7.4,
        7.4
```

```
        ]
      }
    ]
}
```

```
curl -H "Content-Type: application/json" -H "Authorization: Bearer ${CDP_TOK
EN}" "https://${DOMAIN}/namespaces/serving-default/endpoints/[***ENDPOINT_NA
ME***]/v2/models/model/infer" -d @./examples/wine-input.json

{
   "model_name":"model",
   "model_version":"1",
   "outputs":[
     {
       "name":"variable",
       "datatype":"FP32",
       "shape":[1,1],
       "data":[5.535987377166748]
     }
   ]
}
```

Deleting a Model Endpoint using API

Consider the following instruction to delete a Model Endpoint using API.

Procedure

Delete the Model Endpoint deployed above, using the API.

```
curl -v  -H "Content-Type: application/json" -H "Authorization:
Bearer ${CDP_TOKEN}"
"https://${DOMAIN}/api/v1alpha1/deleteEndpoint" -d
'{"namespace": "serving-default", "name": "[***ENDPOINT_NAME***]"}'
```

Autoscaling Model Endpoints using API

You can configure the Model Endpoints deployed on Cloudera AI Inference service to auto-scale to zero instances when there is no load.

Procedure

The following deployment specification shows an example endpoint that auto-scales between zero and four replicas:

```
# cat ./examples/mlflow/model-spec-cml-registry.json
{
   "namespace": "serving-default",
   "name": "mlflow-wine-test-from-registry-onnx",
   "source": {
     "registry_source": {
       "version": 1,
       "model_id": "yf0o-hrxq-l0xj-8tk9"
     }
   },
   "autoscaling": {
     "min_replicas": "0",
     "max_replicas": "4"
   }
```

```
}
```

Tuning auto-scaling sensitivity using the API

To customize the autos-caling sensitivity and requisites, set the target and metric fields in the autoscaling.autoscalingconfig parameter.

About this task

The metric field is the data you are watching to make auto-scaling decisions, which you can set to either concurrency or to RPS (requests per second):

- Concurrency is the number of requests that each replica of the model shall aim to handle at once.
- RPS is the calculated requests per second handled over the polling period.
- Target is the target value of the metric that you aim to maintain. If the Target value is exceeded when calculating the metric value, new replicas will be spun up or down to maintain the Target value as the upper bound.

Procedure

Set these values as follows:

```
# cat ./examples/mlflow/model-spec-cml-registry.json
{
  "namespace": "serving-default",
  "name": "mlflow-wine-test-from-registry-onnx",
  "source": {
    "registry_source": {
      "version": 1,
      "model_id": "yf0o-hrxq-l0xj-8tk9"
    }
  },
  "autoscaling": {
    "min_replicas": "0",
    "max_replicas": "4",
    "autoscalingconfig": {
        "metric": "concurrency",
        "target": "25"
    }
  }
}
```

The above example scales between zero and four replicas aiming to maintain at most 25 concurrent requests per replica, scaling the number of replicas deployed to maintain this target.

Running Models on GPU

Follow the guidelines for running Models on GPU.

Procedure

To run a Model Endpoint on GPU, specify the number of GPUs to use per model replica, as follows. You must specify the GPU count as a *string*.

```
# cat ./examples/mlflow/model-spec-cml-registry.json
{
  "namespace": "serving-default",
  "name": "mlflow-wine-test-from-registry-onnx",
  "source": {
```

```
      "registry_source": {
        "version": 1,
        "model_id": "yf0o-hrxq-l0xj-8tk9"
      }
    },
    "resources": {
      "num_gpus": "1"
    }
  }
}
```

The resources field supports specifying the following properties:

- req_cpu - the requested number of CPU cores, which is a string as per Kubernetes standards.
- req_memory - the requested amount of memory, which again follows Kubernetes conventions.
- num_gpus - the requested number of GPUs.

Related Information

Resource Units in Kubernetes - Kubernetes documentation

Memory Resource Units - Kubernetes documentation

Deploying models with Canary deployment using API

Cloudera AI Inference service allows users to control traffic percentage to specific model deployments.

Procedure

1. Deploy a model. The traffic value defaults to 100. The following is an example payload:

```
# cat ./examples/mlflow/model-spec-cml-registry.json
{
  "namespace": "serving-default",
  "name": "mlflow-wine-test-from-registry-onnx",
  "source": {
    "registry_source": {
      "version": 1,
      "model_id": "yf0o-hrxq-l0xj-8tk9"
    }
  }
}
```

2. After deploying a model to the endpoint, direct the traffic to the *Canary* model by updating the model with the desired traffic value set:

```
# cat ./examples/mlflow/model-spec-cml-registry-canary.json
{
  "namespace": "serving-default",
  "name": "mlflow-wine-test-from-registry-onnx",
  "source": {
    "registry_source": {
      "version": 2,
      "model_id": "yf0o-hrxq-l0xj-8tk9"
    }
  }
  "traffic": "35",
}
```

The percentage of the traffic set in the traffic field diverts to the newly deployed model version and the remaining traffic is directed to the previously deployed model version.

3. Update the inference endpoint as follows, when you want your canary model to serve all of your incoming requests:

```
# cat ./examples/mlflow/model-spec-cml-registry-promote.json
{
   "namespace": "serving-default",
   "name": "mlflow-wine-test-from-registry-onnx",
   "source": {
     "registry_source": {
       "version": 2,
       "model_id": "yf0o-hrxq-l0xj-8tk9"
     }
   }
   "traffic": "100",
}
```

4. Use the following payload to content to the model:

```
curl -H "Content-Type: application/json" -H
"Authorization: Bearer ${CDP_TOKEN}"
https://${DOMAIN}/namespaces/serving-default/endpoints/mlflow-wine-test-f
rom-registry-onnx/v2/models/yf0o-hrxq-l0xj-8tk9/infer
-d @./examples/url/wine-input.json
{"model_name":"yf0o-hrxq-l0xj-8tk9","model_version":"2","outputs":[{"name"
:"variable","datatype":"FP32","shape":[1,1],"data":[0.0]}]}
```

Deploying a Predictive Deep Learning Model

The following example illustrates how to deploy the *resnet-18* image classification model from *ONNX Model Zoo*.

About this task

You must first import the model into a session in CML Workspace, which must be in the same CDP environment as the AI Inference service and Cloudera Model Registry.

Procedure

Upload the model artifact to your project's file system.

a) From the **Project** overview page, complete the upload from your local computer using the upload button. In your session, load the artifact to the Cloudera Machine Learning experiments page by using the following script:

```
!pip install onnx mlflow onnxruntime

import onnx
import mlflow
import onnxruntime

resnet18 = onnx.load("resnet18-v1-7.onnx")
mlflow.set_experiment("resnet18")
with mlflow.start_run() as run:
    mlflow.onnx.log_model(resnet18,
                          "resnet-demo",
                          registered_model_name="model1")
```

b) Navigate to the **Experiments** page and locate your experiment.
c) Select the desired experiment and select the artifact folder.
d) Select Register model to upload to model registry.

e) Navigate to the AI Inference Service to deploy the model, using the following payload to deploy the *resnet18* model into our cluster. Replace the *model_id* and *version* with the matching values from the workspace model registry page:

```
curl -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${CDP_TOKEN}" \
  "https://${DOMAIN}/api/v1alpha1/deployEndpoint" -d \
'{
    "namespace": "serving-default",
    "name": "resnet-18-onnx",
    "source": {
      "registry_source": {
        "version": 1,
        "model_id": "pr5z-mc4s-hrxq-5zg4"
      }
    }
}
```

f) Follow the status of this model through the describe model endpoint:

```
curl -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${CDP_TOKEN}" \
  "https://${DOMAIN}/api/v1alpha1/describeEndpoint" -d \
'{
    "namespace": "serving-default",
    "name": "resnet18-onnx"
}'
```

See the following output when you describe the model endpoint:

```
{
  "namespace": "serving-default",
  "name": "resnet18-onnx",
  "url": "",
  "conditions": [...]
  "status": {
    "failed_copies": 0,
    "total_copies": 0,
    "active_model_state": "",
    "target_model_state": "Loading",
    "transition_status": "InProgress"
  },
  "observed_generation": 2,
  "replica_count": 0,
  "created_by": "csso_user",
  "description": "",
  "created_at": "2024-05-10T19:22:21Z",
  "resources": {
    "req_cpu": "1",
    "req_memory": "2Gi"
    "num_gpus": "N/A"
  },
  "source": {
    "registry_source": {
      "model_id": "pr5z-mc4s-hrxq-5zg4",
      "version": 1
    }
  },
  "autoscaling": {...},
  "endpointmetadata": {
    "current_model": {
      "registry_source": {
        "model_id": "pr5z-mc4s-hrxq-5zg4",
```

```
      "version": 1
    }
  },
  "previous_model": null
},
"traffic": {
  "current_revision_traffic": "100",
  "previous_revision_traffic": "0"
},
"api_standard": "oip",
"chat": false
}
```

From this output you can infer the following information:

- The model comes from the Cloudera Model Registry, its ID is pr5z-mc4s-hrxq-5zg4, its version is 1.
- The Model Endpoint is conforming to the Open Inference Protocol (OIP) API standard.
- The chat attribute is false, therefore the model is only able to respond on the /v2/models/[***model_name***]/infer endpoint.

Loading is in progress, The URL is not yet present for this endpoint.

Eventually, the model is ready and the described payload shall appear as follows:

```
{
  "namespace": "serving-default",
  "name": "resnet18-onnx",
  "url": "https://ml-1fcaa8cf-a94.eng-ml-i.svbr-nqvp.int.cldr.work/namesp
aces/serving-default/endpoints/resnet18-onnx/v2/models/pr5z-mc4s-hrxq-5z
g4/infer",
  "conditions": [ ...
  ],
  "status": {
    "failed_copies": 0,
    "total_copies": 1,
    "active_model_state": "Loaded",
    "target_model_state": "Loaded",
    "transition_status": "UpToDate"
  },
  "observed_generation": 2,
  "replica_count": 1,
  "created_by": "csso_user",
  "description": "",
  "created_at": "2024-05-10T19:22:21Z",
  "resources": {
    "req_cpu": "1",
    "req_memory": "2Gi",
    "num_gpus": "N/A"
  },
  "source": {
    "registry_source": {
      "model_id": "pr5z-mc4s-hrxq-5zg4",
      "version": 1
    }
  },
  "autoscaling": {
    "min_replicas": "1",
    "max_replicas": "1",
    "autoscalingconfig": {
      "metric": "",
      "target": "",
      "target_utilization": "",
      "scale_to_zero_retention": "",
```

```
          "activation_scale": "",
          "scale_down_delay": "",
          "panic_window_percentage": "",
          "panic_threshold": "",
          "stable_window": ""
      }
    },
    "endpointmetadata": {
      "current_model": {
        "registry_source": {
          "model_id": "pr5z-mc4s-hrxq-5zg4",
          "version": 1
        }
      },
      "previous_model": null
    },
    "traffic": {
      "current_revision_traffic": "100",
      "previous_revision_traffic": "0"
    },
    "api_standard": "oip",
    "chat": false
}
```

Results

Now you have loaded the *resnet18* model into your Cloudera AI Inference service cluster.

Related Information
ONNX models

Accessing Cloudera AI Inference service Metrics

Cloudera AI Inference service exposes Prometheus metrics for the deployed Model Endpoints. The UI displays plots of a few chosen metrics for each model endpoint.

For additional metrics, the Prometheus server can be accessed directly at https://${DOMAIN_NAME}/prometheus by passing the authorization bearer token.

Predictive models deployed with the NVIDIA Triton Inference Server and NVIDIA NIM embedding models export metrics prefixed with nv_. Refer to the *NVIDIA NIM documentation* for metrics exported by the NVIDIA NIM Large Language Model (LLM) runtimes for text-generation.

Related Information
Authentication of Cloudera AI Inference service
Nvidia NIM Large Language Models, Observability

Known issues

There are some known issues you might run into while using Cloudera AI Inference service.

- The following compute instance types are not supported by Cloudera AI Inference service:
  - Azure: NVadsA10_v5 series.
  - AWS: p4d.24xlarge

- Updating the description after a model has been added to an endpoint will lead to a UI mismatch in the model builder for models listed by the model builder and the models deployed.
- When you create a model endpoint from the **Create Endpoint** page, even though the instance type selection is not mandatory, the instance creation fails if the instance type is not selected.
- DSE-39626: If no worker node can be found within 10 minutes to schedule a model endpoint that is either newly created or is scaling up from 0, the system will give up trying to create and schedule the replica. A common reason for this behavior is insufficient cloud quota, or capacity constraints on the cloud service provider's side. You could either ask for increased quota, or try to use an instance type that is more readily available.

Related Information

Obtaining the kubeconfig for the Cloudera AI Inference service Cluster

NVadsA10_v5 Microsoft specification

p4d.24xlarge NAWS documentation