BentoML

Dec 11, 2019

Contents

BentoML is a flexible framework that accelerates the workflow of serving and deploying machine learning models in the cloud. It provides two set of high-level APIs:

- BentoService: Turn your trained ML model into versioned file bundle that can be deployed as containerize REST API server, PyPI package, CLI tool, or batch/streaming job
- YataiService: Manage and deploy your saved BentoML bundles into prediction services on Kubernetes cluster or cloud platforms such as AWS Lambda, SageMaker, Azure ML, and GCP Function etc

CHAPTER 1

Content

1.1 Quick Start

1.1.1 See it in action

Run the code in this guide here on Google's Colab:

Alternatively, download and run the notebook locally:

```
$ pip install jupyter
$ git clone http://github.com/bentoml/bentoml
$ jupyter notebook bentoml/guides/quick-start/bentoml-quick-start-guide.ipynb
```

1.1.2 Quick start walk through

Defining a prediction service with BentoML:

```
import bentoml
from bentoml.handlers import DataframeHandler
from bentoml.artifact import SklearnModelArtifact

@bentoml.env(pip_dependencies=["scikit-learn"])
@bentoml.artifacts([SklearnModelArtifact('model')])
class IrisClassifier(bentoml.BentoService):

    @bentoml.api(DataframeHandler)
    def predict(self, df):
        return self.artifacts.model.predict(df)
```

You can add multiple *bentoml.api* to a BentoService, and the *DataframeHandler* here tells BentoML the expected input format of this API.

The *bentoml.env* decorator allows user to specify the dependencies and environment settings for this prediction service and *bentoml.artifact* is used to describe the trained models to be bundled with this prediction service. In addition to *SklearnModelArtifact*, BentoML libraries also provides *PytorchModelArtifact*, *KerasModelArtifact*, *FastaiModelArtifact*, and *XgboostModelArtifact* etc.

Next, train a classifier model with Iris dataset and pack the trained model with the BentoService *IrisClassifier* defined above:

```
from sklearn import svm
from sklearn import datasets

clf = svm.SVC(gamma='scale')
iris = datasets.load_iris()
X, y = iris.data, iris.target
clf.fit(X, y)

# Create a iris classifier service with the newly trained model
iris_classifier_service = IrisClassifier.pack(model=clf)

# Save the entire prediction service to file bundle
saved_path = iris_classifier_service.save()
```

You've just created a BentoML bundle, it's a versioned file archive, containing the BentoService you defined, including the trained model artifacts, pre-processing code, dependencies and configurations.

1.1.2.1 Model serving via REST API

Now you can start a REST API server based off the saved BentoML bundle form command line:

```
bentoml serve {saved_path}
```

If you are doing this only local machine, visit http://127.0.0.1:5000 in your browser to play around with the API server's Web UI for debbugging and testing. You can also send prediction request with curl from command line:

```
curl -i \
  --header "Content-Type: application/json" \
  --request POST \
  --data '[[5.1, 3.5, 1.4, 0.2]]' \
  http://localhost:5000/predict
```

1.1.2.2 Model serving via Command Line Interface

Load the saved BentoML bundle directly from command line for inferencing:

```
bentoml predict {saved_path} --input='[[5.1, 3.5, 1.4, 0.2]]'

# alternatively:
bentoml predict {saved_path} --input='./iris_test_data.csv'
```

1.1.2.3 Distribute BentoML Bundle as PyPI package

BentoML bundle is pip-installable and can be directly distributed as a PyPI package:

```
pip install {saved_path}
```

```
# Your bentoML model class name will become packaged name
import IrisClassifier

installed_svc = IrisClassifier.load()
installed_svc.predict([[5.1, 3.5, 1.4, 0.2]])
```

This allow users to upload their BentoService to pypi.org as public python package or to their organization's private PyPi index to share with other developers.

```
!cd {saved_path} & python setup.py sdist upload
```

Note: You will have to configure ".pypirc" file before uploading to pypi index. You can find more information about distributing python package at: https://docs.python.org/3.7/distributing/index.html#distributing-index

1.1.2.4 Run REST API server with Docker

BentoML bundle is structured to work as a docker build context so you can easily build a docker image for this API server by using it as the build context directory:

```
docker build -t my_api_server {saved_path}

docker run -p 5000:5000 my_api_server
```

Note: You will need to install Docker before running this. Follow direction from this link: https://docs.docker.com/install

1.1.2.5 Learning More?

Interested in learning more about BentoML? Check out the Examples on BentoML github repository.

Be sure to *join BentoML slack channel <http://bit.ly/2N5IpbB>* to hear about the latest development updates.

1.2 API Reference

1.2.1 BentoML

1.2.1.1 BentoService

class bentoml.**BentoService**

BentoService packs a list of artifacts and exposes service APIs for BentoAPIServer and BentoCLI to execute. By subclassing BentoService, users can customize the artifacts and environments required for a ML service.

```
>>>  from bentoml import BentoService, env, api, artifacts, ver
>>>  from bentoml.handlers import DataframeHandler
>>>  from bentoml.artifact import SklearnModelArtifact
```

(continues on next page)

(continued from previous page)

```
>>>
>>>  @ver(major=1, minor=4)
>>>  @artifacts([SklearnModelArtifact('clf')])
>>>  @env(pip_dependencies=["scikit-learn"])
>>>  class MyMLService(BentoService):
>>>
>>>      @api(DataframeHandler)
>>>      def predict(self, df):
>>>          return self.artifacts.clf.predict(df)
>>>
>>>  bento_service = MyMLService()
>>>  bento_service.pack('clf', trained_classifier_model)
>>>  bento_service.save_to_dir('/bentoml_bundles')
```

classmethod name()
return BentoService name

versioneer()
Function used to generate a new version string when saving a new BentoService bundle. User can also override this function to get a customized version format

set_version(*version_str=None*)
Manually override the version of this BentoService instance

get_service_apis()
Return a list of user defined API functions

Returns List of user defined API functions

Return type list(BentoServiceAPI)

1.2.1.2 api

bentoml.**api**(*handler_cls*, **args*, ***kwargs*)
Decorator for adding api to a BentoService

Parameters

- **handler_cls** (*bentoml.handlers.BentoHandler*) – The handler class for the API function.
- **api_name** (str, optional) – API name to replace function name
- **api_doc** (str, optional) – Docstring for API function
- ****kwargs** – Additional keyword arguments for handler class. Please reference to what arguments are available for the particular handler

Raises ValueError – API name must contains only letters

```
>>> from bentoml import BentoService, api
>>> from bentoml.handlers import JsonHandler, DataframeHandler
>>>
>>> class FraudDetectionAndIdentityService(BentoService):
>>>
>>>     @api(JsonHandler)
>>>     def fraud_detect(self, parsed_json):
>>>         # do something
>>>
```

(continues on next page)

(continued from previous page)

```
>>>     @api(DataframeHandler, input_json_orient='records')
>>>     def identity(self, df):
>>>         # do something
```

1.2.1.3 env

bentoml.**env**(*setup_sh=None*, *pip_dependencies=None*, *conda_channels=None*, *conda_dependencies=None*)

Define environment and dependencies required for the BentoService being created

Parameters

- **setup_sh** (*str*) – bash script for initializing docker environment before loading the BentoService for inferencing
- **pip_dependencies** (*list(str)*) – List of python PyPI package dependencies
- **conda_channels** (*list(str)*) – List of conda channels required for conda dependencies
- **conda_dependencies** (*list(str)*) – List of conda dependencies required

1.2.1.4 artifacts

bentoml.**artifacts**(*artifacts*)

Define artifacts required to be bundled with a BentoService

Parameters **artifacts** (*list(bentoml.artifact.BentoServiceArtifact)*) – A list of desired artifacts required by this BentoService

1.2.1.5 ver

bentoml.**ver**(*major*, *minor*)

Decorator for specifying the version of a custom BentoService.

Parameters

- **major** (*int*) – Major version number for Bento Service
- **minor** (*int*) – Minor version number for Bento Service

BentoML uses semantic versioning for BentoService distribution:

- MAJOR is incremented when you make breaking API changes
- MINOR is incremented when you add new functionality without breaking the existing API or functionality
- PATCH is incremented when you make backwards-compatible bug fixes

'Patch' is provided(or auto generated) when calling BentoService#save, while 'Major' and 'Minor' can be defined with '@ver' decorator

```
>>>  @ver(major=1, minor=4)
>>>  @artifacts([PickleArtifact('model')])
>>>  class MyMLService(BentoService):
>>>     pass
>>>
```

(continues on next page)

(continued from previous page)

```
>>>  svc = MyMLService()
>>>  svc.pack("model", trained_classifier)
>>>  svc.set_version("2019-08.iteration20")
>>>  svc.save()
>>>  # The final produced BentoService bundle will have version:
>>>  # "1.4.2019-08.iteration20"
```

1.2.1.6 save

bentoml.**save**(*bento_service*, *base_path=None*, *version=None*)

Save given bento_service via BentoML's default Yatai service, which manages all saved Bento files and their deployments in cloud platforms. If remote yatai service has not been configured, this will default to saving new Bento to local file system under BentoML home directory

Parameters

- **bento_service** (*bentoml.service.BentoService*) – a Bento Service instance
- **base_path** (*str*) – optional, base path of the bento repository
- **version** (*str*) – optional,

Returns URI to where the BentoService is being saved to

1.2.1.7 save_to_dir

bentoml.**save_to_dir**(*bento_service*, *path*, *version=None*)

Save given BentoService along with all its artifacts, source code and dependencies to target file path, assuming path exist and empty. If target path is not empty, this call may override existing files in the given path.

Parameters

- **bento_service** (*bentoml.service.BentoService*) – a Bento Service instance
- **path** (*str*) – Destination of where the bento service will be saved

1.2.1.8 load

bentoml.**load**(*bundle_path*)

Load bento service from local file path or s3 path

Parameters **bundle_path** (*str*) – The path that contains saved BentoService bundle, supporting both local file path and s3 path

Returns a loaded BentoService instance

Return type bentoml.service.BentoService

1.2.2 Artifacts

1.2.2.1 SklearnModelArtifact

class bentoml.artifact.**SklearnModelArtifact**(*name*, *pickle_extension='.pkl'*)

Abstraction for saving/loading scikit learn models using sklearn.externals.joblib

Parameters

- **name** (*str*) – Name for the artifact
- **pickle_extension** (*str*) – The extension format for pickled file

Raises ImportError – sklean package is required for SklearnModelArtifact

1.2.2.2 PytorchModelArtifact

class bentoml.artifact.**PytorchModelArtifact**(*name*, *file_extension='.pt'*)

Abstraction for saving/loading objects with torch.save and torch.load

Parameters **name** (*string*) – name of the artifact

Raises

- ImportError – torch package is required for PytorchModelArtifact
- TypeError – invalid argument type, model being packed must be instance of torch.nn.Module

1.2.2.3 KerasModelArtifact

class bentoml.artifact.**KerasModelArtifact**(*name*, *custom_objects=None*, *model_extension='.h5'*, *store_as_json_and_weights=False*)

Abstraction for saving/loading Keras model

Parameters

- **name** (*string*) – name of the artifact
- **custom_objects** (*dict*) – dictionary of Keras custom objects for model
- **store_as_json_and_weights** (*bool*) – flag allowing storage of the Keras model as JSON and weights

Raises

- ImportError – keras or tensorflow.keras package is required for KerasModelArtifact
- TypeError – invalid argument type, model being packed must be instance of keras.engine.network.Network, tf.keras.models.Model, or their aliases

1.2.2.4 FastaiModelArtifact

class bentoml.artifact.**FastaiModelArtifact**(*name*)

Saving and Loading FastAI Model

Parameters **name** (*str*) – Name for the fastai model

Raises

- ImportError – Require fastai package to use Fast ai model artifact
- TypeError – invalid argument type, model being packed must be instance of fastai.basic_train.Learner

1.2.2.5 TensorflowSavedModelArtifact

class bentoml.artifact.**TensorflowSavedModelArtifact**(*name*)

Abstraction for saving/loading Tensorflow model in tf.saved_model format

Parameters **name** (*string*) – name of the artifact

Raises ImportError – tensorflow package is required for TensorflowSavedModelArtifact

Example usage:

```
>>> import tensorflow as tf
>>>
>>> # Option 1: custom model with specific method call
>>> class Adder(tf.Module):
>>>     @tf.function(input_signature=[tf.TensorSpec(shape=None, dtype=tf.
↪float32)])
>>>     def add(self, x):
>>>         return x + x + 1.
>>> model_to_save = Adder()
>>> # ... compiling, training, etc
>>>
>>> # Option 2: Sequential model (direct call only)
>>> model_to_save = tf.keras.Sequential([
>>>     tf.keras.layers.Flatten(input_shape=(28, 28)),
>>>     tf.keras.layers.Dense(128, activation='relu'),
>>>     tf.keras.layers.Dense(10, activation='softmax')
>>> ])
>>> # ... compiling, training, etc
>>>
>>> import bentoml
>>> from bentoml.handlers import JsonHandler
>>> from bentoml.artifact import TensorflowSavedModelArtifact
>>>
>>> @bentoml.env(pip_dependencies=["tensorflow"])
>>> @bentoml.artifacts([TensorflowSavedModelArtifact('model')])
>>> class TfModelService(bentoml.BentoService):
>>>
>>>     @bentoml.api(JsonHandler)
>>>     def predict(self, json):
>>>         input_data = json['input']
>>>         prediction = self.artifacts.model.add(input_data)
>>>         # prediction = self.artifacts.model(input_data)  # if Sequential mode
>>>         return prediction.numpy()
>>>
>>> svc = TfModelService()
>>>
>>> # Option 1: pack directly with Tensorflow trackable object
>>> svc.pack('model', model_to_save)
>>>
>>> # Option 2: save to file path then pack
>>> tf.saved_model.save(model_to_save, '/tmp/adder/1')
>>> svc.pack('model', '/tmp/adder/1')
```

1.2.2.6 XgboostModelArtifact

class bentoml.artifact.**XgboostModelArtifact**(*name*, *model_extension='.model'*)

Abstraction for save/load object with Xgboost.

Parameters

- **name** (*string*) – name of the artifact
- **model_extension** (*string*) – Extension name for saved xgboost model

Raises

- ImportError – xgboost package is required for using XgboostModelArtifact
- TypeError – invalid argument type, model being packed must be instance of xgboost.core.Booster

1.2.2.7 H2oModelArtifact

class bentoml.artifact.**H2oModelArtifact**(*name*)

Abstraction for saving/loading objects with h2o.save_model and h2o.load_model

Parameters **name** (*str*) – Name for this h2o artifact..

Raises ImportError – h2o package is required to use H2o model artifact

1.2.2.8 PickleArtifact

class bentoml.artifact.**PickleArtifact**(*name*, *pickle_module=<module 'bentoml.utils.cloudpickle' from '/home/docs/checkouts/readthedocs.org/user_builds/bentoml/checkouts/latest/b*, *pickle_extension='.pkl'*)

Abstraction for saving/loading python objects with pickle serialization

Parameters

- **name** (*str*) – Name for the artifact
- **pickle_module** (*module | str*) – The python module will be used for pickle and unpickle artifact, default pickle module in BentoML's fork of *cloudpickle*, which is identical to the Apache Spark fork
- **pickle_extension** (*str*) – The extension format for pickled file.

1.2.2.9 TextFileArtifact

class bentoml.artifact.**TextFileArtifact**(*name*, *file_extension='.txt'*, *encoding='utf8'*)

Abstraction for saving/loading string to/from text files

Parameters

- **name** (*str*) – Name of the artifact
- **file_extension** (str, optional) – The file extention used for the saved text file. Defaults to ".txt"
- **encoding** (*str*) – The encoding will be used for saving/loading text. Defaults to "utf8"

1.2.3 Handlers

1.2.3.1 DataframeHandler

class bentoml.handlers.**DataframeHandler**(*orient='records'*, *output_orient='records'*, *typ='frame'*, *input_dtypes=None*)

Dataframe handler expects inputs from rest request or cli options that can be converted into a pandas Dataframe, and pass down the dataframe to user defined API function. It also returns response for REST API call or print result for CLI call

Parameters

- **orient** (*str*) – Incoming json orient format for reading json data. Default is records.
- **output_orient** (*str*) – Prefer json orient format for output result. Default is records.
- **typ** (*str*) – Type of object to recover for read json with pandas. Default is frame
- **({str** (*input_dtypes*) – str}): A dict of column name and data type.

Raises

- ValueError – Incoming data is missing required columns in input_dtypes
- ValueError – Incoming data format is not handled. Only json and csv

1.2.3.2 ImageHandler

class bentoml.handlers.**ImageHandler**(*input_names=('image',)*, *accept_image_formats=None*, *pilmode='RGB'*)

Transform incoming image data from http request, cli or lambda event into numpy array.

Handle incoming image data from different sources, transform them into numpy array and pass down to user defined API functions

Parameters

- **input_names** (*string[]*) – A tuple of acceptable input name for HTTP request. Default value is (image,)
- **accept_image_formats** (*string[]*) – A list of acceptable image formats. Default value is loaded from bentoml config 'apiserver/default_image_handler_accept_file_extensions', which is set to ['.jpg', '.png', '.jpeg', '.tiff', '.webp', '.bmp'] by default. List of all supported format can be found here: https://imageio.readthedocs.io/en/stable/formats.html
- **pilmode** (*string*) – The pilmode to be used for reading image file into numpy array. Default value is 'RGB'. Find more information at: https://imageio.readthedocs.io/en/stable/format_png-pil.html

Raises ImportError – imageio package is required to use ImageHandler

1.2.3.3 FastaiImageHandler

class bentoml.handlers.**FastaiImageHandler**(*input_names=('image',)*, *accept_image_formats=None*, *convert_mode='RGB'*, *div=True*, *cls=None*, *after_open=None*)

BentoHandler specified for handling image input following fastai conventions by passing type fas-

tai.vision.Image to user API function and providing options such as div, cls, and after_open

Parameters

- **input_names** (*[str]*) – A tuple of acceptable input name for HTTP request. Default value is (image,)
- **accept_image_formats** (*[str]*) – A list of acceptable image formats. Default value is loaded from bentoml config 'apiserver/default_image_handler_accept_file_extensions', which is set to ['.jpg', '.png', '.jpeg', '.tiff', '.webp', '.bmp'] by default. List of all supported format can be found here: https://imageio.readthedocs.io/en/stable/formats.html
- **convert_mode** (*str*) – The pilmode to be used for reading image file into numpy array. Default value is 'RGB'. Find more information at https://imageio.readthedocs.io/en/stable/format_png-pil.html
- **div** (*bool*) – If True, pixel values are divided by 255 to become floats between 0. and 1.
- **cls** (*Class*) – Parameter from fastai.vision `open_image`, default is `fastai.vision.Image`
- **after_open** (*func*) – Parameter from fastai.vision `open_image`, default is None

Raises

- `ImportError` – imageio package is required to use FastaiImageHandler
- `ImportError` – fastai package is required to use FastaiImageHandler

1.2.3.4 JsonHandler

class bentoml.handlers.**JsonHandler**

JsonHandler parses REST API request or CLI command into parsed_json(a dict in python) and pass down to user defined API function

1.3 CLI

1.3.1 bentoml

BentoML CLI tool

```
bentoml [OPTIONS] COMMAND [ARGS]...
```

Options

-q, --quiet

Hide process logs and only print command results

--verbose, --debug

Print verbose debugging information for BentoML developer

--version

Show the version and exit.

1.3.1.1 <API_NAME>

Run a API defined in saved BentoService bundle from command line

```
bentoml <API_NAME> [OPTIONS] API_NAME BUNDLE_PATH
```

Options

--with-conda
Run API server in a BentoML managed Conda environment

Arguments

API_NAME
Required argument

BUNDLE_PATH
Required argument

1.3.1.2 config

Configure BentoML configurations and settings

```
bentoml config [OPTIONS] COMMAND [ARGS]...
```

reset

Reset all local BentoML configs to default

```
bentoml config reset [OPTIONS]
```

set

Set config value in local BentoML configuration file

```
bentoml config set [OPTIONS] [UPDATES]...
```

Arguments

UPDATES
Optional argument(s)

unset

Unset config in local BentoML configuration file

```
bentoml config unset [OPTIONS] [UPDATES]...
```

Arguments

UPDATES
Optional argument(s)

view

View local BentoML configurations

```
bentoml config view [OPTIONS]
```

view-effective

View effective BentoML configs, including default config values and local config overrides

```
bentoml config view-effective [OPTIONS]
```

1.3.1.3 deployment

Commands for creating and managing BentoService deployments on cloudcomputing platforms or kubernetes cluster

```
bentoml deployment [OPTIONS] COMMAND [ARGS]...
```

apply

Apply model service deployment from yaml file

```
bentoml deployment apply [OPTIONS]
```

Options

-f, --file <deployment_yaml>
[required]

-o, --output <output>

Options json|yaml

--wait, --no-wait
Wait for apply action to complete or encounter an error.If set to no-wait, CLI will return immediately. The default value is wait

create

Create a BentoService model serving deployment

```
bentoml deployment create [OPTIONS] NAME
```

Options

-b, --bento <bento>
Target BentoService to be deployed, referenced by its name and version in format of name:version. For example: "iris_classifier:v1.2.0" [required]

-p, --platform <platform>
Which cloud platform to deploy this BentoService to [required]

Options aws-lambda|gcp-function|aws-sagemaker|kubernetes|custom

-n, --namespace <namespace>
Deployment namespace managed by BentoML, default value is "default" whichcan be changed in BentoML configuration file

-l, --labels <labels>
Key:value pairs that are attached to deployments and intended to be usedto specify identifying attributes of the deployments that are meaningful to users

--annotations <annotations>
Used to attach arbitary metadata to BentoService deployments, BentoML library and other plugins can then retrieve this metadata.

--region <region>
Directly mapping to cloud provider region. Option applicable to platform:AWS Lambda, AWS SageMaker, GCP Function

--instance-type <instance_type>
Type of instance will be used for inference. Option applicable to platform: AWS SageMaker, AWS Lambda, GCP Function

--instance-count <instance_count>
Number of instance will be used. Option applicable to platform: AWS SageMaker

--api-name <api_name>
User defined API function will be used for inference. Option applicableto platform: AWS SageMaker

--kube-namespace <kube_namespace>
Namespace for kubernetes deployment. Option applicable to platform: Kubernetes

--replicas <replicas>
Number of replicas. Option applicable to platform: Kubernetes

--memory-size <memory_size>
Maximum Memory Capacity for AWS Lambda function, you can set the memory size in 64MB increments from 128MB to 3008MB. The default value is 1024 MB.

--timeout <timeout>
The amount of time that AWS Lambda allows a function to run before stopping it. The default is 3 seconds. The maximum allowed value is 900 seconds

--service-name <service_name>
Name for service. Option applicable to platform: Kubernetes

--service-type <service_type>
Service Type. Option applicable to platform: Kubernetes

-o, --output <output>

Options json|yaml

--wait, --no-wait
Wait for cloud resources to complete creation or until an error is encountered. When set to no-wait, CLI will return immediately after sendingrequest to cloud platform.

Arguments

NAME
Required argument

delete

Delete deployment

```
bentoml deployment delete [OPTIONS] NAME
```

Options

-n, --namespace <namespace>
Deployment namespace managed by BentoML, default value is "default" whichcan be changed in BentoML configuration file

--force
force delete the deployment record in database and ignore errors when deleting cloud resources

Arguments

NAME
Required argument

describe

View the detailed state of the deployment

```
bentoml deployment describe [OPTIONS] NAME
```

Options

-n, --namespace <namespace>
Deployment namespace managed by BentoML, default value is "default" whichcan be changed in BentoML configuration file

-o, --output <output>

Options jsonlyaml

Arguments

NAME
Required argument

get

Get deployment current state

```
bentoml deployment get [OPTIONS] NAME
```

Options

-n, --namespace <namespace>
Deployment namespace

-o, --output <output>

Options json|yaml

Arguments

NAME
Required argument

list

List active deployments

```
bentoml deployment list [OPTIONS]
```

Options

-n, --namespace <namespace>
Deployment namespace managed by BentoML, default value is "default" whichcan be changed in BentoML configuration file

--all-namespaces

--limit <limit>
Limit how many deployments will be retrieved

--filters <filters>
List deployments containing the filter string in name or version

-l, --labels <labels>
List deployments matching the giving labels

-o, --output <output>

Options json|yaml|table

1.3.1.4 info

List all APIs defined in the BentoService loaded from saved bundle

```
bentoml info [OPTIONS] BUNDLE_PATH
```

Arguments

BUNDLE_PATH
Required argument

1.3.1.5 open-api-spec

Display API specification JSON in Open-API format

```
bentoml open-api-spec [OPTIONS] BUNDLE_PATH
```

Arguments

BUNDLE_PATH
Required argument

1.3.1.6 serve

Start REST API server hosting BentoService loaded from saved bundle

```
bentoml serve [OPTIONS] BUNDLE_PATH
```

Options

--port <port>
The port to listen on for the REST api server, default is 5000.

--with-conda
Run API server in a BentoML managed Conda environment

Arguments

BUNDLE_PATH
Required argument

1.3.1.7 serve-gunicorn

Start REST API server from saved BentoService bundle with gunicorn

```
bentoml serve-gunicorn [OPTIONS] BUNDLE_PATH
```

Options

-p, --port <port>

-w, --workers <workers>
Number of workers will start for the gunicorn server

--timeout <timeout>

--with-conda
Run API server in a BentoML managed Conda environment

Arguments

BUNDLE_PATH
Required argument

Index

Symbols

A

B