

머신러닝 파이프라인

# Model Serving

## bentoML

송호연





# 목차



## Model Serving bentoML



1. bentoML 소개



2. bentoML 기초 실습

3. 적응형 마이크로 배치 실습

4. TF2 Fashion MNIST 실습



# 학습목표



## Model Serving bentoML



### 01. Model Serving bentoML에 대해 이해한다.

Model Serving bentoML의 개념에 이해한다.



### 02. Model Serving bentoML 실습을 통해 작동 방식을 이해한다.

실습을 진행하면서 **bentoML** 기본 사용법에 대해 공부한다.



### 03. 적응형 마이크로 배치 실습을 통해 작동 방식을 이해한다.

실습을 진행하면서 적응형 마이크로 배치 기본 사용법에 대해 공부한다.

### 04. TF2 Fashion MNIST 실습을 통해 작동 방식을 이해한다.

실습을 진행하면서 TF2 Fashion MNIST를 **bentoML**로 추론하는 법에 대해 공부한다.

bentoML 소개



01



# bentoML 소개



## 개요

bentoML은 머신러닝 모델을 서빙을 구현한 **open source** 프로젝트입니다.

bentoML은 mlflow와는 달리 모델 서빙에만 집중한 프로젝트입니다.

\*출처 : Google Cloud Tech Youtube([https://www.youtube.com/watch?v=\\_AY8mmbR1o4](https://www.youtube.com/watch?v=_AY8mmbR1o4))



# bentoML 소개



## 개요

**BentoML**의 주요 아이디어는 데이터 과학 팀이 테스트하기 쉽고 배포하기 쉬우며 통합하기 쉬운 방식으로 모델을 제공할 수 있어야 한다는 것입니다.

\*출처 : Google Cloud Tech Youtube([https://www.youtube.com/watch?v=\\_AY8mmbR1o4](https://www.youtube.com/watch?v=_AY8mmbR1o4))



# bentoML 소개



## 개요

1. 모델 훈련
2. BentoService 인스턴스 만들기
3. BentoService#pack 으로 학습된 모델 아티팩트 패키징
4. BentoService#save 로 Bento에 저장

\*출처 : <https://docs.bentoml.org/en/latest/concepts.html>

bentoML 실습



02





# bentoML 실습



## bentoml 설치

**pip install**로 bentoml을 설치한다.

```
pip install bentoml
```

\*출처 : [https://github.com/chris-chris/mlflow-tutorial/blob/master/lesson1\\_iris/sk\\_iris.py](https://github.com/chris-chris/mlflow-tutorial/blob/master/lesson1_iris/sk_iris.py)



# bentoML 실습



## iris 분류 모델 학습

### iris.py

시작하기 전에 BentoML을 사용하기 위한 학습된 모델을 준비하겠습니다.

```
from sklearn import svm
from sklearn import datasets

# Load training data
iris = datasets.load_iris()
X, y = iris.data, iris.target

# Model Training
clf = svm.SVC(gamma='scale')
clf.fit(X, y)
```

\* 출처 : [https://github.com/chris-chris/bentoml-tutorial/lesson1\\_helloworld/iris\\_service.py](https://github.com/chris-chris/bentoml-tutorial/lesson1_helloworld/iris_service.py)



# bentoML 실습



## iris 분류모델 서비스 정의 (1)

**iris\_classifier.py** - iris 모델을 제공하기 위해 생성된 최소 예측 서비스입니다.

```
import pandas as pd

from bentoml import env, artifacts, api, BentoService
from bentoml.adapters import DataframeInput
from bentoml.frameworks.sklearn import SklearnModelArtifact

@env(infer_pip_packages=True)
@artifacts([SklearnModelArtifact('model')])
class IrisClassifier(BentoService):

    @api(input=DataframeInput(), batch=True)
    def predict(self, df: pd.DataFrame):
        """
        An inference API named `predict` with Dataframe input adapter, which codifies
        how HTTP requests or CSV files are converted to a pandas Dataframe object as the
        inference API function input
        """
        return self.artifacts.model.predict(df)
```

\* 출처 : [https://github.com/chris-chris/bentoml-tutorial/lesson1\\_helloworld/iris\\_classifier.py](https://github.com/chris-chris/bentoml-tutorial/lesson1_helloworld/iris_classifier.py)



# bentoML 실습



## iris 분류모델 서비스 정의 (2)

**@artifact(...)** 는 예측 서비스에 필요한 모델을 정의합니다.

```
import pandas as pd

from bentoml import env, artifacts, api, BentoService
from bentoml.adapters import DataframeInput
from bentoml.frameworks.sklearn import SklearnModelArtifact

@env(infer_pip_packages=True)
@artifacts([SklearnModelArtifact('model')])
class IrisClassifier(BentoService):

    @api(input=DataframeInput(), batch=True)
    def predict(self, df: pd.DataFrame):
        """
        An inference API named `predict` with Dataframe input adapter, which codifies
        how HTTP requests or CSV files are converted to a pandas Dataframe object as the
        inference API function input
        """
        return self.artifacts.model.predict(df)
```

\* 출처 : [https://github.com/chris-chris/bentoml-tutorial/lesson1\\_helloworld/iris\\_classifier.py](https://github.com/chris-chris/bentoml-tutorial/lesson1_helloworld/iris_classifier.py)



# bentoML 실습



## iris 분류모델 서비스 정의 (3)

**@env** 는 예측 서비스에 필요한 종속성 및 환경 설정을 지정합니다.

```
import pandas as pd

from bentoml import env, artifacts, api, BentoService
from bentoml.adapters import DataframeInput
from bentoml.frameworks.sklearn import SklearnModelArtifact

@env(infer_pip_packages=True)
@artifacts([SklearnModelArtifact('model')])
class IrisClassifier(BentoService):

    @api(input=DataframeInput(), batch=True)
    def predict(self, df: pd.DataFrame):
        """
        An inference API named `predict` with Dataframe input adapter, which codifies
        how HTTP requests or CSV files are converted to a pandas Dataframe object as the
        inference API function input
        """
        return self.artifacts.model.predict(df)
```

\* 출처 : [https://github.com/chris-chris/bentoml-tutorial/lesson1\\_helloworld/iris\\_classifier.py](https://github.com/chris-chris/bentoml-tutorial/lesson1_helloworld/iris_classifier.py)



# bentoML 실습



## iris 분류모델 서비스 정의 (4)

**@api** 데코레이터는 예측 서비스에 액세스하기 위한 **endpoint**입니다. 추론 **API**를 정의합니다.

```
import pandas as pd

from bentoml import env, artifacts, api, BentoService
from bentoml.adapters import DataframeInput
from bentoml.frameworks.sklearn import SklearnModelArtifact

@env(infer_pip_packages=True)
@artifacts([SklearnModelArtifact('model')])
class IrisClassifier(BentoService):

    @api(input=DataframeInput(), batch=True)
    def predict(self, df: pd.DataFrame):
        """
        An inference API named `predict` with Dataframe input adapter, which codifies
        how HTTP requests or CSV files are converted to a pandas Dataframe object as the
        inference API function input
        """
        return self.artifacts.model.predict(df)
```

\* 출처 : [https://github.com/chris-chris/bentoml-tutorial/lesson1\\_helloworld/iris\\_classifier.py](https://github.com/chris-chris/bentoml-tutorial/lesson1_helloworld/iris_classifier.py)



# bentoML 실습



## 배포를 위해 예측 서비스 저장

### iris\_service.py

다음 코드는 `IrisClassifier` 위에서 정의한 예측 서비스 클래스로 학습된 모델을 패키징한 다음 배포 및 배포를 위해 `IrisClassifier` 인스턴스를 **BentoML** 형식으로 디스크에 저장합니다.

```
# import the IrisClassifier class defined above
from iris_classifier import IrisClassifier
from sklearn import svm
from sklearn import datasets

iris = datasets.load_iris()
X, y = iris.data, iris.target

clf = svm.SVC(gamma='scale')
clf.fit(X, y)

iris_classifier_service = IrisClassifier()

iris_classifier_service.pack('model', clf)

saved_path = iris_classifier_service.save()
```

\* 출처 : [https://github.com/chris-chris/bentoml-tutorial/lesson1\\_helloworld/iris\\_service.py](https://github.com/chris-chris/bentoml-tutorial/lesson1_helloworld/iris_service.py)



# bentoML 실습



## 배포를 위해 예측 서비스 저장

BentoML은 기본적으로 모든 패키지 모델 파일을

**`~/bentoml/repository/{service_name}/{service_version}`**

디렉토리에 저장합니다.

\*출처 : [https://github.com/chris-chris/bentoml-tutorial/lesson1\\_helloworld/iris\\_service.py](https://github.com/chris-chris/bentoml-tutorial/lesson1_helloworld/iris_service.py)



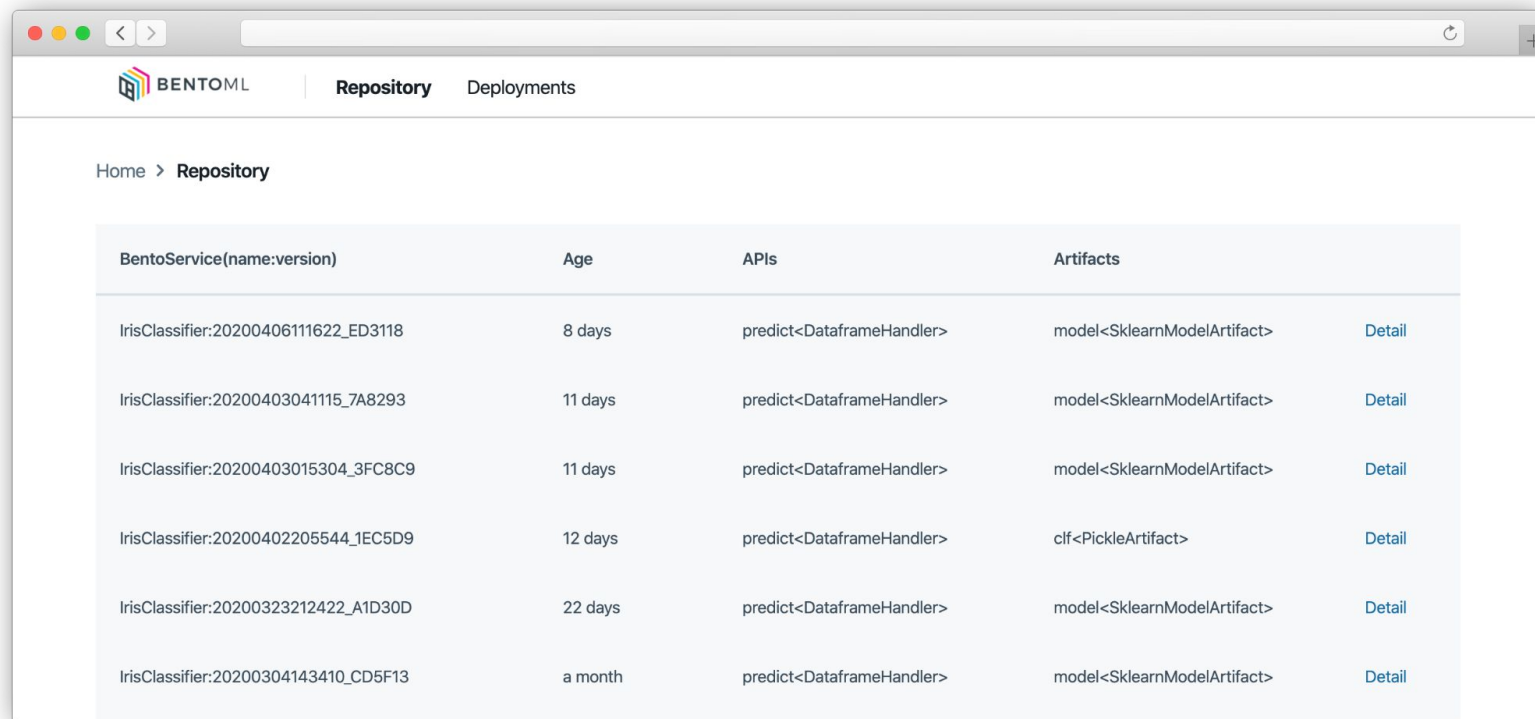


# bentoML 실습



## 배포를 위해 예측 서비스 저장

BentoML은 또한 **YataiService**라는 모델 관리 구성 요소와 함께 제공되며, 이는 팀이 웹 **UI** 및 **API**를 통해 패키지 모델을 관리하고 액세스 할 수있는 중앙 허브를 제공합니다.



The screenshot shows the BentoML web interface with the 'Repository' tab selected. It displays a table of deployed services with columns for service name, age, APIs, and artifacts. Each row includes a 'Detail' link for more information.

BentoService(name:version)	Age	APIs	Artifacts	
IrisClassifier:20200406111622_ED3118	8 days	predict<DataframeHandler>	model<SklearnModelArtifact>	<a href="#">Detail</a>
IrisClassifier:20200403041115_7A8293	11 days	predict<DataframeHandler>	model<SklearnModelArtifact>	<a href="#">Detail</a>
IrisClassifier:20200403015304_3FC8C9	11 days	predict<DataframeHandler>	model<SklearnModelArtifact>	<a href="#">Detail</a>
IrisClassifier:20200402205544_1EC5D9	12 days	predict<DataframeHandler>	clf<PickleArtifact>	<a href="#">Detail</a>
IrisClassifier:20200323212422_A1D30D	22 days	predict<DataframeHandler>	model<SklearnModelArtifact>	<a href="#">Detail</a>
IrisClassifier:20200304143410_CD5F13	a month	predict<DataframeHandler>	model<SklearnModelArtifact>	<a href="#">Detail</a>

\* 출처 : <https://docs.bentoml.org/en/latest/quickstart.html>



# bentoML 실습



## bentoML IrisClassifier 인퍼런스 서버 런치

**IrisClassifier** 인퍼런스 서버를 실행한다.

```
$ bentoml serve IrisClassifier:latest
```

\* 출처 : <https://docs.bentoml.org/en/latest/quickstart.html>



# bentoML 실습



## bentoML IrisClassifier 추론 요청

**curl로 IrisClassifier** 인퍼런스 작업을 요청한다.

```
$ curl -i \  
  --header "Content-Type: application/json" \  
  --request POST \  
  --data '[[5.1, 3.5, 1.4, 0.2]]' \  
  http://localhost:5000/predict  
  
HTTP/1.1 200 OK  
Content-Type: application/json  
X-Request-Id: 25f1ea56-2365-4d38-a9ab-8d0a67dd0505  
Content-Length: 3  
Server: Werkzeug/0.15.5 Python/3.8.3  
Date: Sun, 25 Apr 2021 11:22:03 GMT  
  
[0]
```

\* 출처 : <https://docs.bentoml.org/en/latest/quickstart.html>



# bentoML 실습



## bentoML IrisClassifier 추론 요청

**curl로 IrisClassifier** 인퍼런스 작업을 요청한다.

```
$ bentoml run IrisClassifier:latest predict --input '[[5.1, 3.5, 1.4, 0.2]]'
```

```
[2021-04-25 20:23:39,540] INFO - Getting latest version IrisClassifier:20210419162756_11E81F
[2021-04-25 20:23:46,762] INFO - {'service_name': 'IrisClassifier', 'service_version':
'20210419162756_11E81F', 'api': 'predict', 'task': {'data': '[[5.1, 3.5, 1.4, 0.2]]', 'task_id':
'13738c6a-8c80-461b-87fa-cc30e987bb19', 'batch': 1, 'cli_args': ('--input', '[[5.1, 3.5, 1.4, 0.2]]'),
'inference_job_args': {}, 'result': {'data': '[0]', 'http_status': 200, 'http_headers': (('Content-Type',
'application/json'),)}, 'request_id': '13738c6a-8c80-461b-87fa-cc30e987bb19'}
[0]
```

\* 출처 : <https://docs.bentoml.org/en/latest/quickstart.html>



# bentoML 실습



## bentoML IrisClassifier 도커 컨테이너화

**bentoML IrisClassifier**을 컨테이너화 합니다.

```
$ bentoml containerize IrisClassifier:latest -t iris-classifier
```

\* 출처 : <https://docs.bentoml.org/en/latest/quickstart.html>



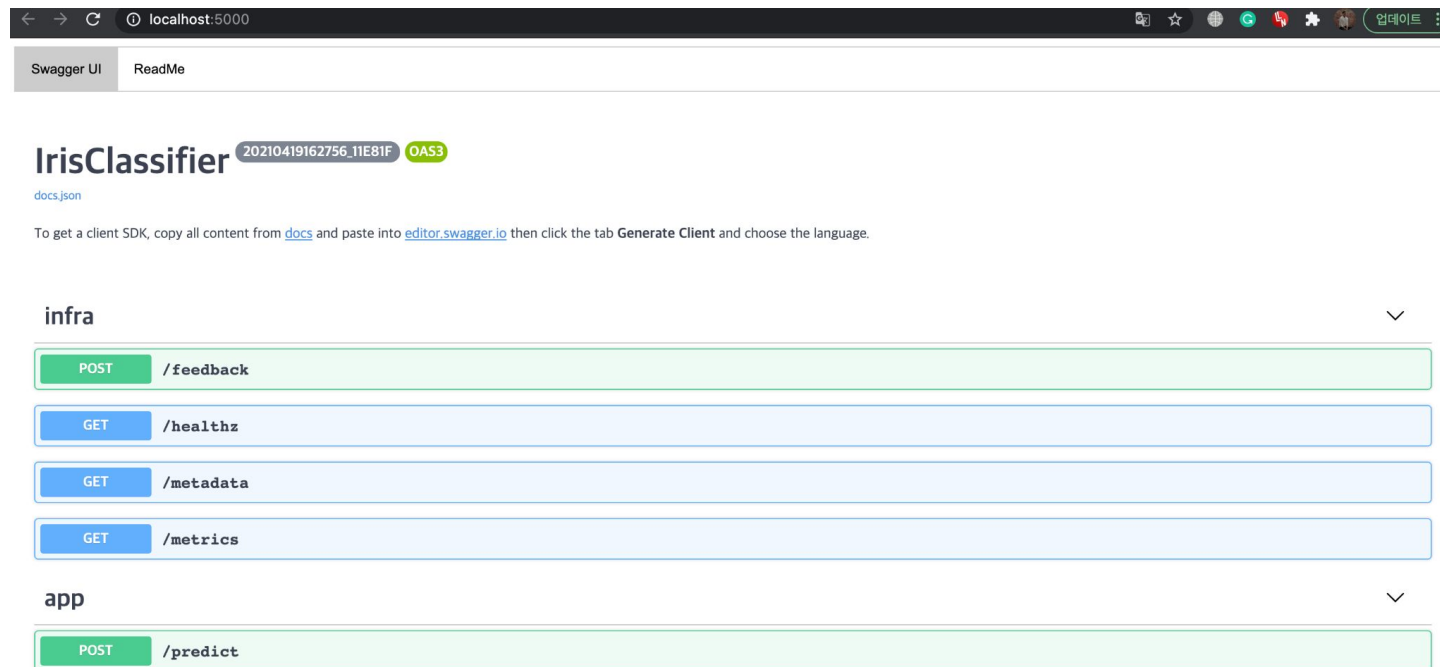
# bentoML 실습



## Swagger를 활용한 반응형 API 문서

<http://localhost:5000>

으로 접속하면, Swagger로 추론 API의 Spec을 확인할 수 있습니다.



\* 출처 : <https://docs.bentoml.org/en/latest/quickstart.html>

적응형 마이크로 배칭



03



## 적응형 마이크로 배칭



### 마이크로배치가 중요한 이유

**TensorFlow** 모델을 서빙하는 동안 개별 모델 추론 요청을 모아서 한 번에 처리하는 것이 성능에 좋습니다. 특히 **GPU**와 같은 하드웨어 가속기가 제공하는 높은 처리량을 활용하려면 일괄 처리가 필요합니다.

\*출처 : <https://docs.bentoml.org/en/latest/quickstart.html>

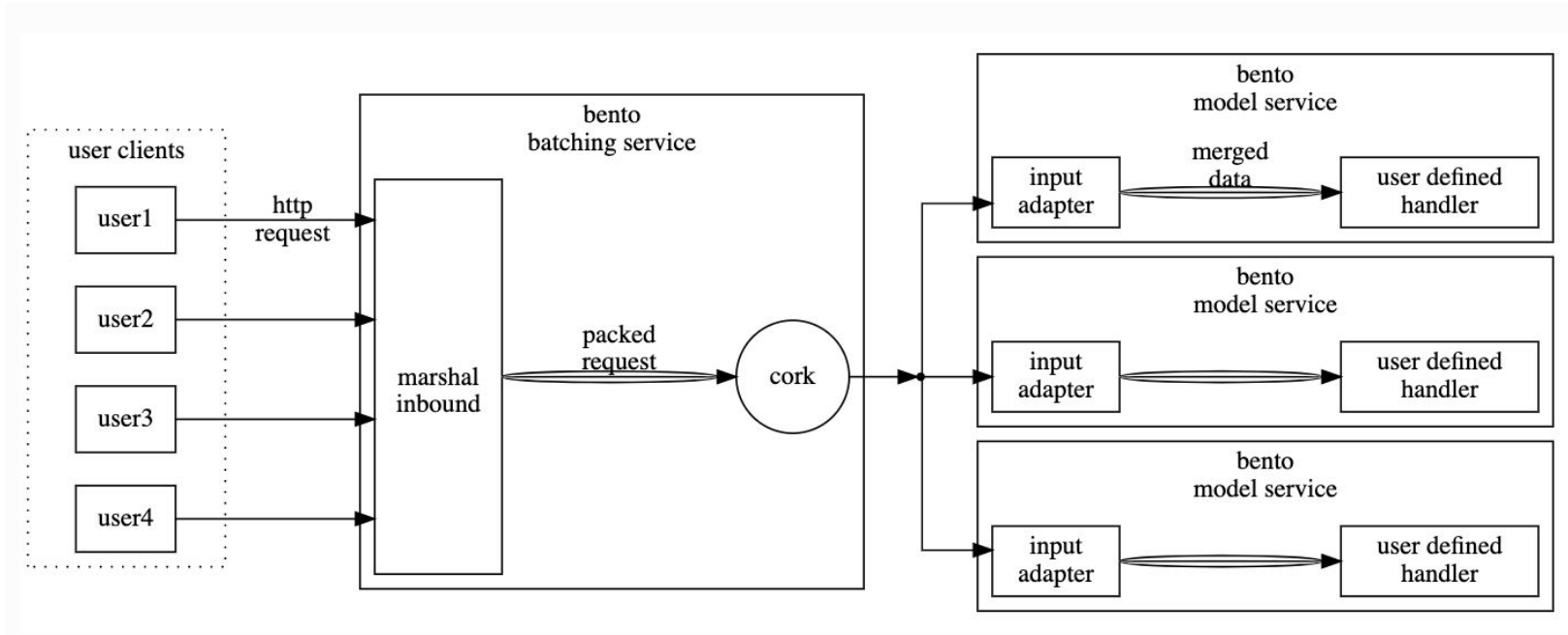




# 적응형 마이크로 배칭



## 아키텍처 및 데이터 흐름



\*출처 : <https://docs.bentoml.org/en/latest/quickstart.html>



## 적응형 마이크로 배칭



### 마이크로배치의 매개 변수 및 개념

- **인바운드 요청** : 사용자 클라이언트의 요청
- **아웃 바운드 요청** : 업스트림 모델 서버에 대한 요청
- **mb\_max\_batch\_size**: 배치의 최대 크기입니다. 이 매개 변수는 처리량 / 대기 시간 상충 관계를 제어하고, 일부 리소스 제약을 초과하는 너무 큰 배치 (예 : 배치의 데이터를 보관하기 위한 **GPU** 메모리)를 방지합니다. 기본값 : **1000**.
- **mb\_max\_latency**: 서비스의 지연 시간 목표 (밀리 초)입니다. 기본값 : **10000**.
- **아웃 바운드 세마포어** : 세마포어는 병렬 처리 정도, 즉 동시에 처리되는 최대 배치 수를 나타냅니다. 동일한 수의 모델 서버 작업자로 도시락 서비스를 시작할 때 자동으로 설정됩니다 .
- **예상 시간** : 모델 서버가 배치를 실행하는 데 걸리는 예상 시간입니다. 대기열의 기록 데이터 및 현재 배치 크기에서 유추됩니다.

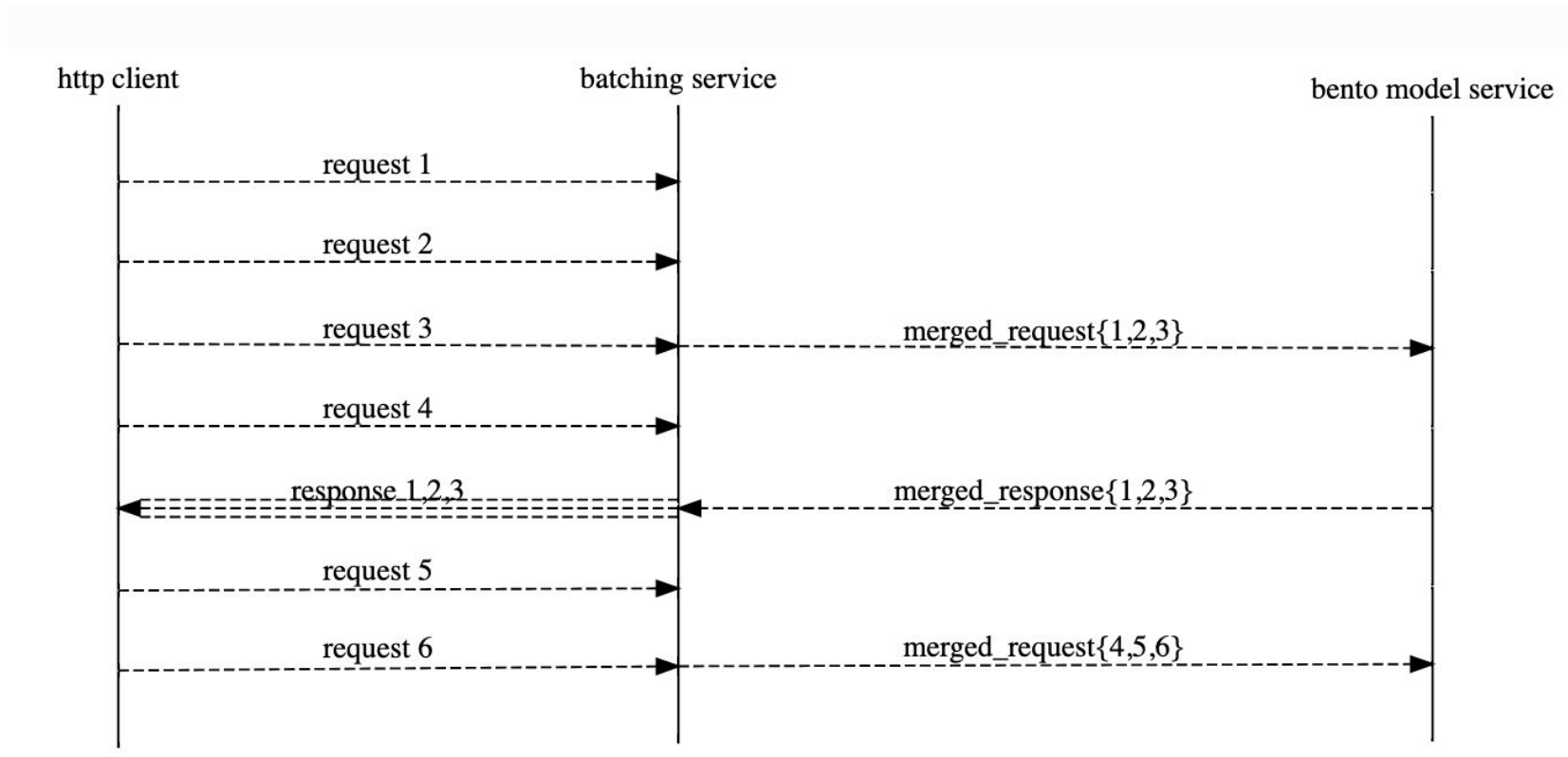
\*출처 : <https://docs.bentoml.org/en/latest/quickstart.html>



# 적응형 마이크로 배칭



## 시퀀스 & 동작 원리



\*출처 : <https://docs.bentoml.org/en/latest/quickstart.html>



## 적응형 마이크로 배칭



### 마이크로배치의 매개 변수 및 개념

최적의 효율성을 달성하기 위해 **CORK** 디스패처는 인바운드 요청 을 **cork/release** 하는 적응형 제어를 수행 합니다. 릴리스는 다음과 같은 경우에 발생합니다.

- 다음 조건 중 하나를 충족합니다.
  - **waited time + estimated time**이 **mb\_max\_latency**를 초과할 때
  - 다음 인바운드 요청을 기다릴 필요가 없을 때
- 그리고 아웃 바운드 세마포가 잠겨 있지 않을 때

\*출처 : <https://docs.bentoml.org/en/latest/quickstart.html>



## 적응형 마이크로 배칭



### 주요 디자인 결정사항과 트레이드 오프들

처리량 및 지연 시간은 **API** 서버에 가장 큰 영향을 미칩니다.

**BentoML**은 다음과 같이 자동으로 배치를 미세 조정합니다 (우선 순위에 따라 정렬).

1순위. 사용자 정의 제약 조건 `mb_max_batch_size` 및 `mb_max_latency` 을 보장.

2순위. **Throughput** 최대화

3순위. 평균 **Latency Time** 최소화

\*출처 : <https://docs.bentoml.org/en/latest/quickstart.html>



## 적응형 마이크로 배칭



### 주요 디자인 결정사항과 트레이드 오프들

Tensorflow Serving과는 달리,

BentoML은 자동적으로 **wait timeout**, **batch size**을 조정하여  
최대의 처리량과 최소의 응답시간을 찾습니다.

```
class MovieReviewService(bentoml.BentoService):  
    @bentoml.api(  
        input=DataframeInput(),  
        mb_max_latency=10000,  
        mb_max_batch_size=1000,  
        batch=True)  
    def predict(self, inputs):  
        pass
```

\*출처 : <https://docs.bentoml.org/en/latest/quickstart.html>



## 적응형 마이크로 배칭



### 파라미터에 대한 조언

만약 RAM, GPU가 처리할 수 있는 배치사이즈가 100이라면, **mb\_max\_batch\_size**는

100으로 설정

만약 당신의 API를 사용하는 client의 timeout이 200ms라면, **mb\_max\_latency**는 200을 설정

만약 당신이 모델이 소드가 100ms 정도 걸린다면, **mb\_max\_latency**를 10\*100ms 정도

class MovieReviewService(bentoml.BentoService):

```
@bentoml.api(
    input=DataframeInput(),
    mb_max_latency=10000,
    mb_max_batch_size=1000,
    batch=True)
```

```
def predict(self, inputs):
    pass
```

\*출처 : <https://docs.bentoml.org/en/latest/quickstart.html>

TF2 이미지 분류 서빙 실습

A decorative graphic in the bottom right corner consisting of a grid of dots. The dots are arranged in a roughly rectangular shape, with some dots missing, creating a pattern that resembles a stylized '04'. The dots are white and light blue, set against a background of blue and teal gradients.

04





# TF2 이미지 분류 서빙 실습



## Fashion MNIST 모델 서빙 실습

```
import io

# TensorFlow
import tensorflow as tf

# Helper libraries
import numpy as np
import matplotlib.pyplot as plt
print(tf.__version__)
```

\* 출처 : <https://docs.bentoml.org/en/latest/quickstart.html>

# TF2 이미지 분류 서빙 실습

## Fashion MNIST 모델 서빙 실습

```
fashion_mnist = tf.keras.datasets.fashion_mnist
(_train_images, train_labels), (_test_images, test_labels) = fashion_mnist.load_data()
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
train_images = _train_images / 255.0
test_images = _test_images / 255.0
```

\* 출처 : <https://docs.bentoml.org/en/latest/quickstart.html>



# TF2 이미지 분류 서빙 실습



## Fashion MNIST 모델 서빙 실습

```
class FashionMnist(tf.keras.Model):
    def __init__(self):
        super(FashionMnist, self).__init__()
        self.cnn = 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')
        ])

    @staticmethod
    def image_bytes2tensor(inputs):
        with tf.device("cpu:0"):
            inputs = tf.map_fn(lambda i: tf.io.decode_png(i, channels=1), inputs, dtype=tf.uint8)
            inputs = tf.cast(inputs, tf.float32)
            inputs = (255.0 - inputs) / 255.0
            inputs = tf.reshape(inputs, [-1, 28, 28])
            return inputs
```

\* 출처 : <https://docs.bentoml.org/en/latest/quickstart.html>

# TF2 이미지 분류 서빙 실습

## Fashion MNIST 모델 서빙 실습

```
@tf.function(input_signature=[tf.TensorSpec(shape=(None,), dtype=tf.string)])  
def predict_image(self, inputs):  
    inputs = self.image_bytes2tensor(inputs)  
    return self(inputs)  
  
def call(self, inputs):  
    return self.cnn(inputs)
```

\* 출처 : <https://docs.bentoml.org/en/latest/quickstart.html>



# TF2 이미지 분류 서빙 실습



## 테스트용 이미지 생성

```
# pick up a test image
d_test_img = _test_images[0]
print(class_names[test_labels[0]])

plt.imshow(255.0 - d_test_img, cmap='gray')
plt.imsave("test.png", 255.0 - d_test_img, cmap='gray')

# read bytes
with open("test.png", "rb") as f:
    img_bytes = f.read()

# verify saved image
assert tf.reduce_mean(FashionMnist.image_bytes2tensor(tf.constant([img_bytes])) - d_test_img) < 0.01
```

\* 출처 : <https://docs.bentoml.org/en/latest/quickstart.html>



# TF2 이미지 분류 서빙 실습



## 모델 학습

```
model = FashionMnist()
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.fit(train_images, train_labels, epochs=50)
```

\*출처 : <https://docs.bentoml.org/en/latest/quickstart.html>



# TF2 이미지 분류 서빙 실습



## 모델 추론 동작 체크

```
predict = model.predict_image(tf.constant([img_bytes]))  
klass = tf.argmax(predict, axis=1)  
[class_names[c] for c in klass]
```

\*출처 : <https://docs.bentoml.org/en/latest/quickstart.html>



# TF2 이미지 분류 서빙 실습



## BentoService 클래스 생성

### **tensorflow\_fashion\_mnist.py**

```
import bentoml
import tensorflow as tf
from bentoml.artifact import TensorflowSavedModelArtifact
from bentoml.adapters import TfTensorInput

FASHION_MNIST_CLASSES = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
                          'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']

@bentoml.env(pip_dependencies=['tensorflow', 'numpy', 'pillow'])
@bentoml.artifacts([TensorflowSavedModelArtifact('model')])
class FashionMnistTensorflow(bentoml.BentoService):

    @bentoml.api(input=TfTensorInput(), batch=True)
    def predict(self, inputs):
        outputs = self.artifacts.model.predict_image(inputs)
        output_classes = tf.math.argmax(outputs, axis=1)
        return [FASHION_MNIST_CLASSES[c] for c in output_classes]
```

\* 출처 : <https://docs.bentoml.org/en/latest/quickstart.html>



# TF2 이미지 분류 서빙 실습

## BentoService 클래스 생성

```
from tensorflow_fashion_mnist import FashionMnistTensorflow

bento_svc = FashionMnistTensorflow()
bento_svc.pack("model", model)
saved_path = bento_svc.save()
```

\* 출처 : <https://docs.bentoml.org/en/latest/quickstart.html>



## bentoML 실습



### bentoML FashionMnistTensorflow 인퍼런스 서버 런치

**FashionMnistTensorflow** 인퍼런스 서버를 실행한다.

```
$ bentoml serve FashionMnistTensorflow:latest
```

\* 출처 : <https://docs.bentoml.org/en/latest/quickstart.html>



# bentoML 실습



## bentoML FashionMnistTensorflow 추론 요청

**curl로 FashionMnistTensorflow 인퍼런스 작업을 요청한다.**

```
$ echo '{"instances":[{"b64":"$(base64 test.png)"}]}' > test.json  
$ curl -X POST http://localhost:5000/predict -d @test.json --header "Content-Type: application/json"
```

```
["Ankle boot"]
```

\*출처 : <https://docs.bentoml.org/en/latest/quickstart.html>



## 짚어보기



### Model Serving bentoML



#### 01. Model Serving bentoML에 대해 이해한다.

Model Serving bentoML의 개념에 이해한다.



#### 02. Model Serving bentoML 실습을 통해 작동 방식을 이해한다.

실습을 진행하면서 bentoML 기본 사용법에 대해 공부한다.



#### 03. 적응형 마이크로 배치 실습을 통해 작동 방식을 이해한다.

실습을 진행하면서 적응형 마이크로 배치 기본 사용법에 대해 공부한다.

#### 04. TF2 Fashion MNIST 실습을 통해 작동 방식을 이해한다.

실습을 진행하면서 TF2 Fashion MNIST를 bentoML로 추론하는 법에 대해 공부한다.

머신러닝 파이프라인

# Model Serving

## bentoML

송호연



감사합니다.

THANKS FOR WATCHING

