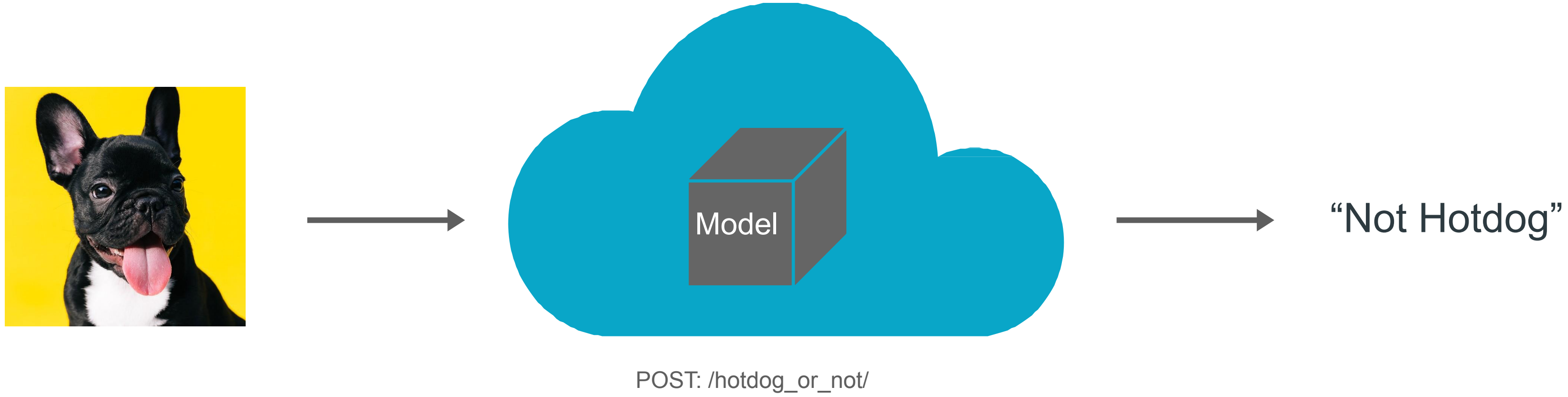
Model Serving systems

From the ground up

Chaoyu Yang
CEO @ BentoML.com

BENTOML

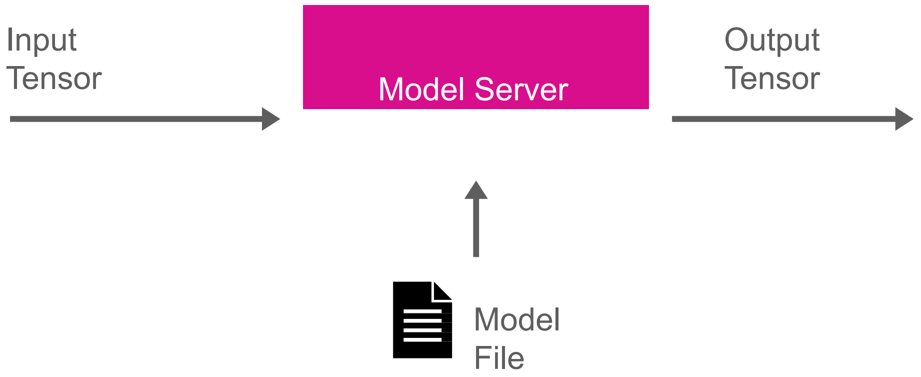
I have a trained model!



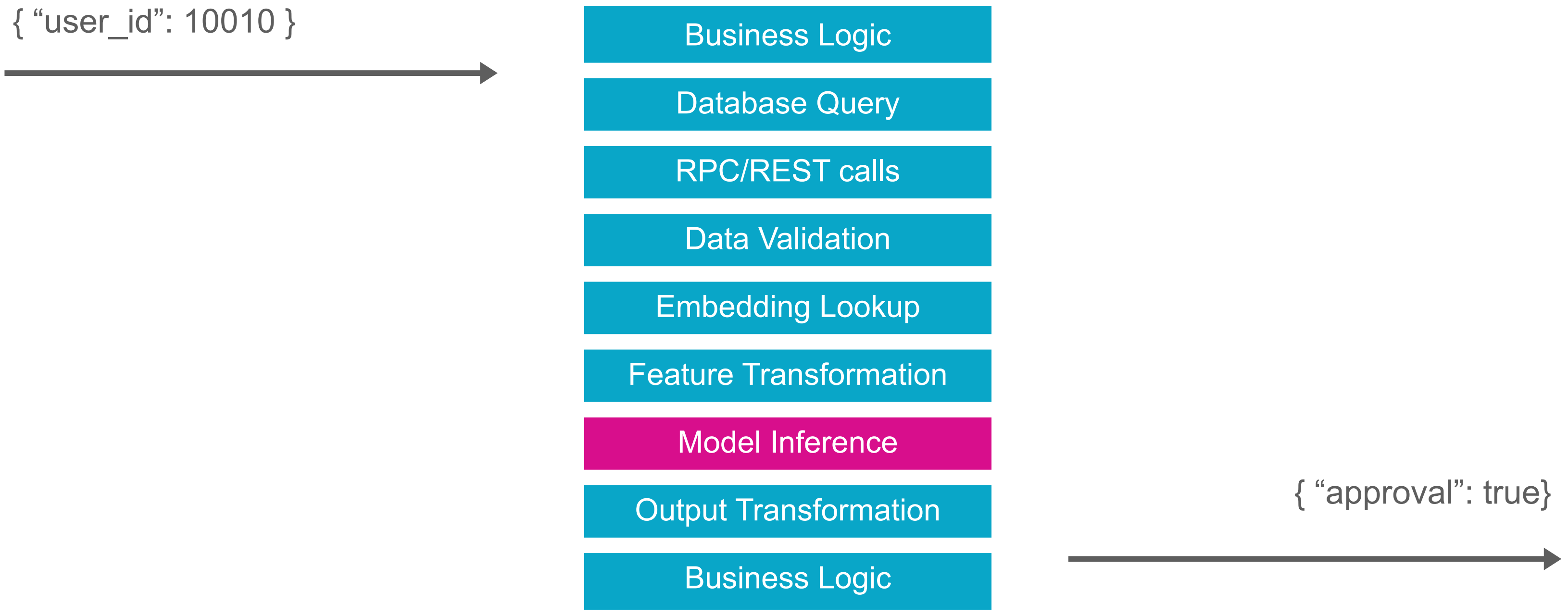
BENTOML

Using a Model Server

- TF-Serving
- Triton Inference Server
- TorchServe

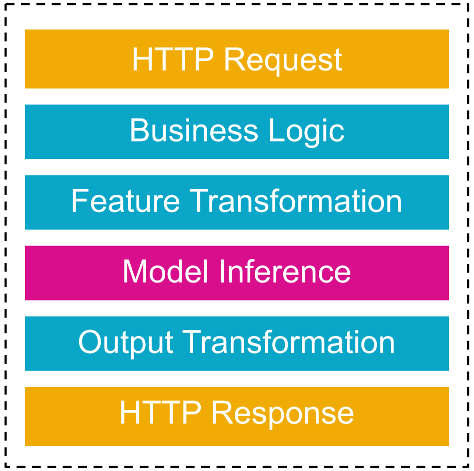


Real world ML application



BENTOML

Using a web serving framework

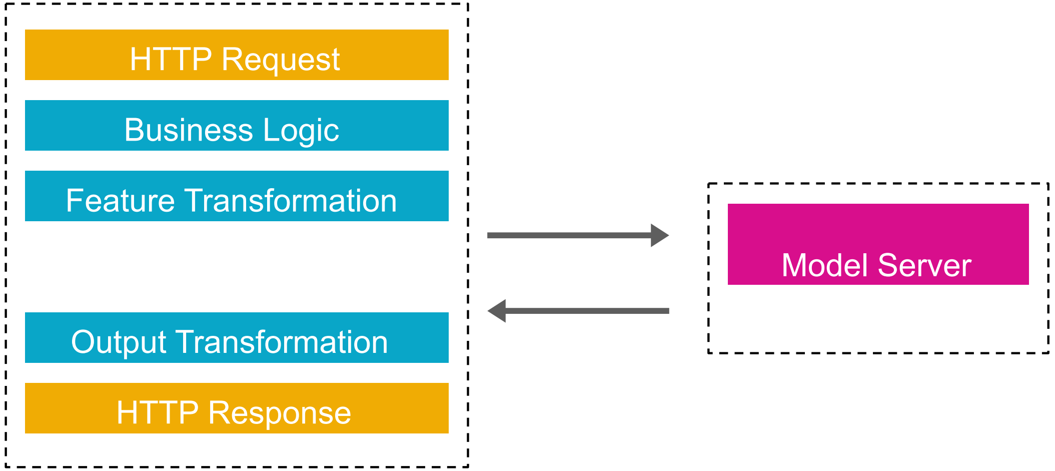


FastAPI
worker

- Scale by replicating process
- Low resource utilization, limited by GIL
- No batching

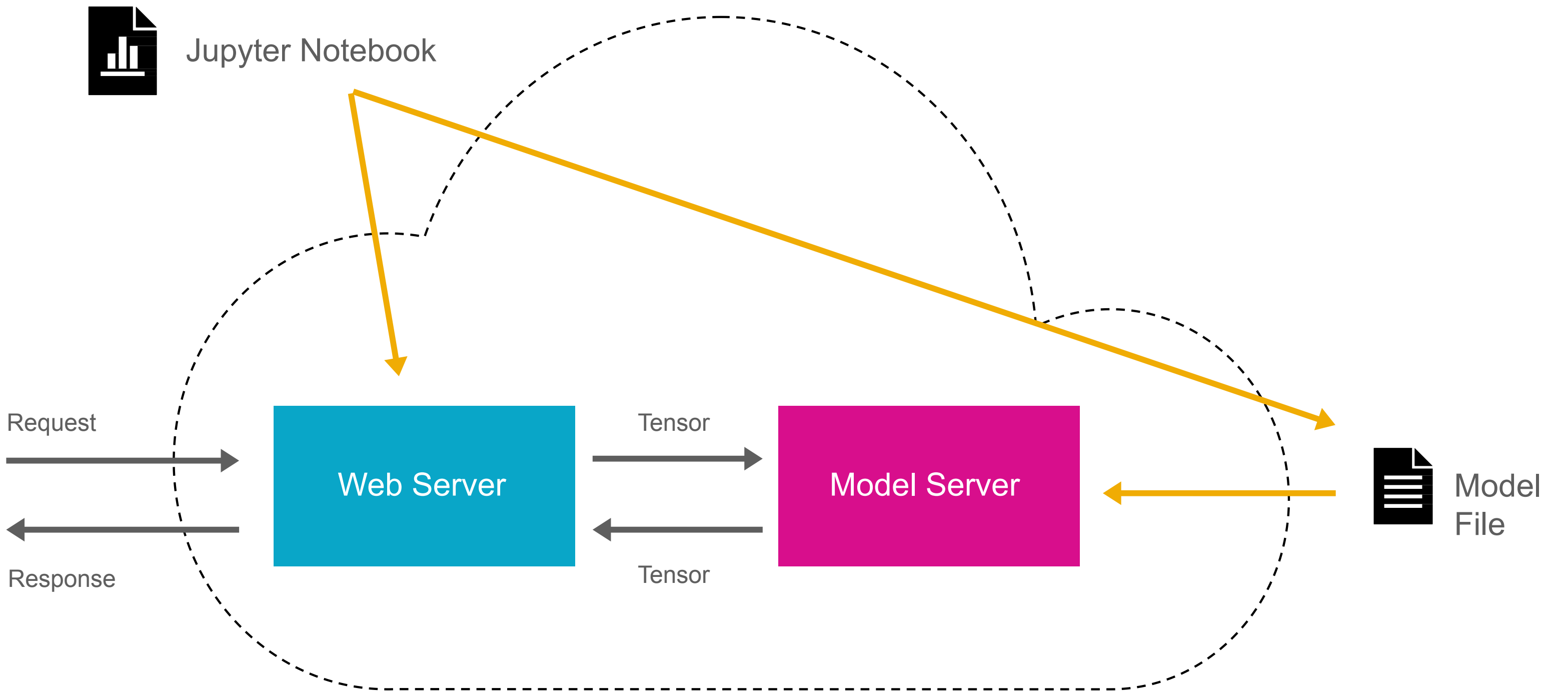
BENTOML

Web Server + Model Server, best from both worlds?



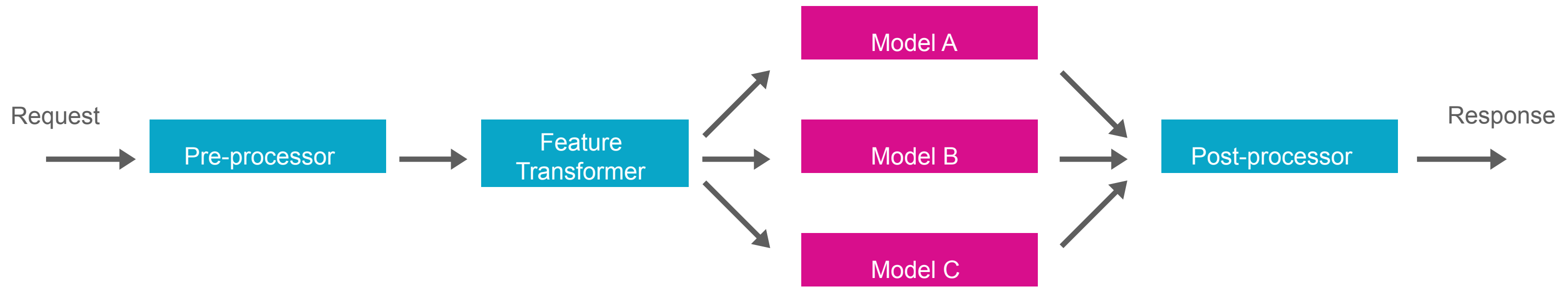
BENTOML

Model and code are tightly coupled



BENTOML

Serving Complex Pipelines



- Complex micro-services or task-queue based architecture
- Hard to optimize performance and eliminate bottleneck
- Hard to evaluate model performance end-to-end

BENTOML

Day 1 and Day 2 problems 👉 even higher infrastructure cost

- New ML Framework support?
- Fine-tune batching behavior for each model
- Retrain model and CI/CD pipeline
- Model performance monitoring
- A/B Testing, Canary rollout, Multi-armed bandit
- …

Introducing BentoML

The Unified Model Serving Framework

BENTOML

Save and version all your models in one place

```
model = train_model()

import bentoml
bentoml.pytorch.save('recommender', model)
```

- One model store that works for all ML frameworks
- Model registry backed by cloud blob storage
- Preserve model dependency versions, metadata, and labels

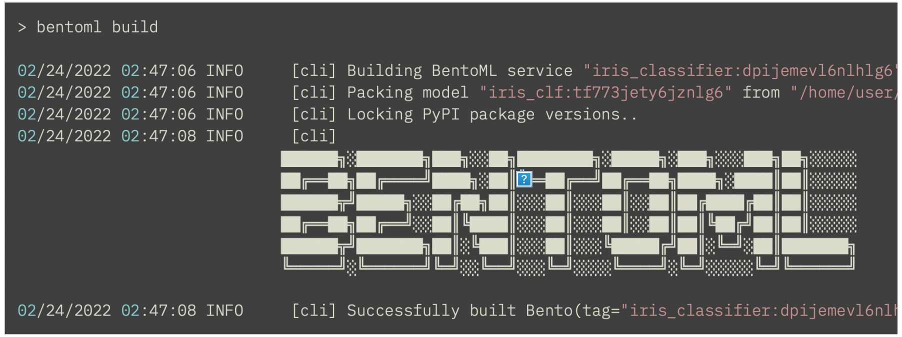
BENTOML

Describe entire serving pipeline in Python

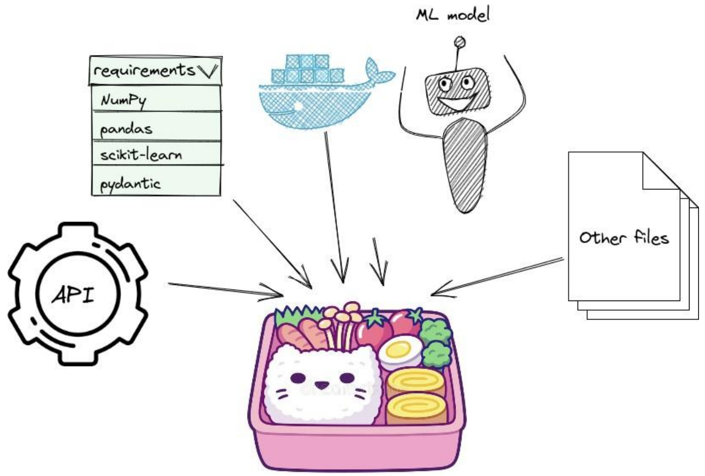
```python
import asyncio
import bentoml
from bentoml.io import Image, Text
from mylib import pre_process, post_process

svc = bentoml.Service("my_ml_service")
model_a_runner = bentoml.xgboost.ModelRunner('model_a:latest')
model_b_runner = bentoml.pytorch.ModelRunner('model_b:latest')

@svc.api(input=Image(), output=Text())
async def predict(input_image):
    model_input = pre_process(input_image)

    results = asyncio.gather(
        model_a_runner.async_run(model_input),
        model_b_runner.async_run(model_input),
    )

    return post_process(results)
```

BENTOML

Build Bento for deployment



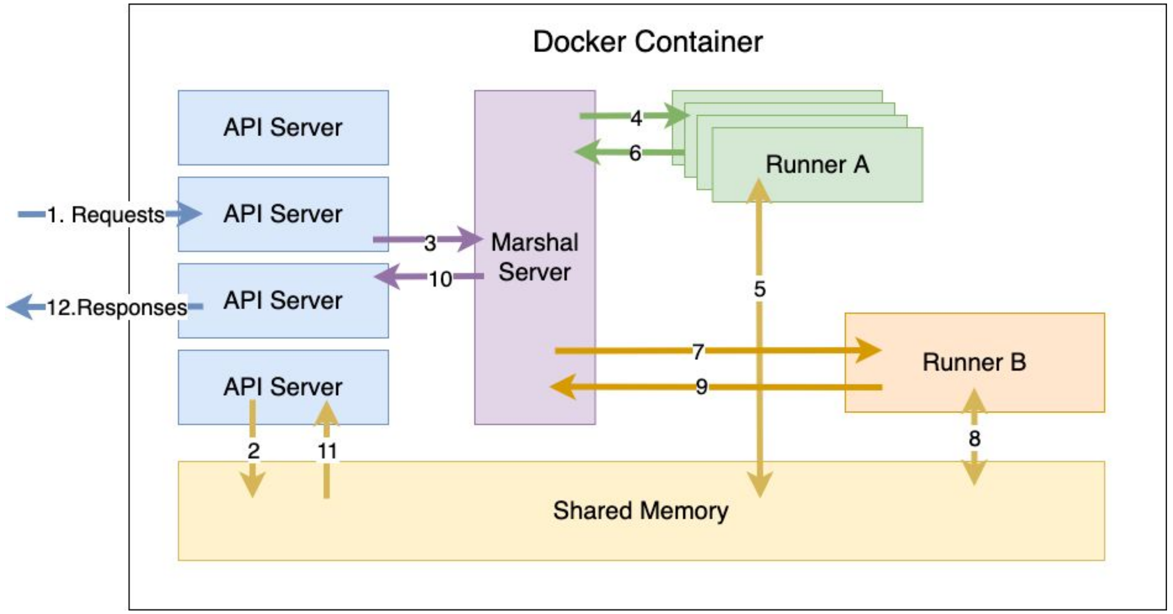
```
> bentoml build

02/24/2022 02:47:06 INFO     [cli] Building BentoML service "iris_classifier:dpijemevl6nlhlg6"
02/24/2022 02:47:06 INFO     [cli] Packing model "iris_clf:tf773jety6jznlg6" from "/home/user/
02/24/2022 02:47:06 INFO     [cli] Locking PyPI package versions..
02/24/2022 02:47:08 INFO     [cli]

02/24/2022 02:47:08 INFO     [cli] Successfully built Bento(tag="iris_classifier:dpijemevl6nl
```



- Bento 🍱 is just like docker for ML deployment
- Package all your models, serving pipeline code, and dependencies into a bento
- Easy to test and ready for deployment

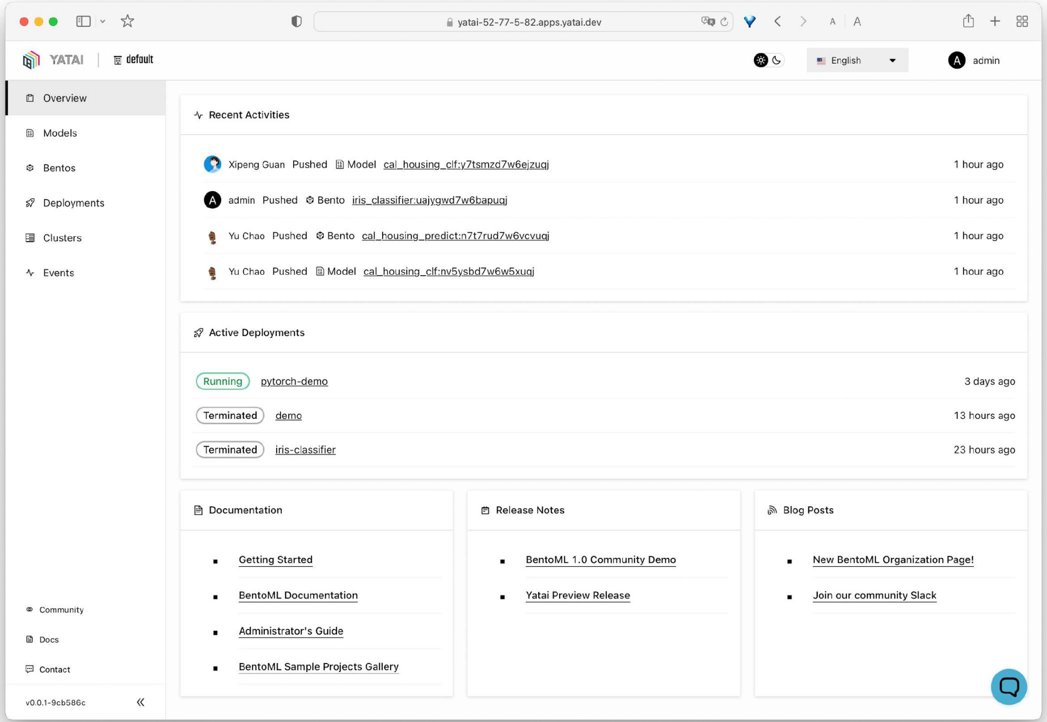
BENTOML

High-performance serving out-of-the-bento



Single-node deployment architecture

Yatai: BentoML at scale on Kubernetes


yatai-52-77-5-82.apps.yatai.dev

YATAI | default

English

admin

- Overview
- Models
- Bentos
- Deployments
- Clusters
- Events

Recent Activities

Xipeng Guan Pushed Model cal_housing_clf:y7tsmzd7w6ejzuqj	1 hour ago
admin Pushed Bento iris_classifier:uajygwd7w6bapuqj	1 hour ago
Yu Chao Pushed Bento cal_housing_predict:n7t7rud7w6vcvuqj	1 hour ago
Yu Chao Pushed Model cal_housing_clf:nv5ysbd7w6w5xuqj	1 hour ago

Active Deployments

Running	pytorch-demo	3 days ago
Terminated	demo	13 hours ago
Terminated	iris-classifier	23 hours ago

Documentation

- Getting Started
- BentoML Documentation
- Administrator's Guide
- BentoML Sample Projects Gallery

Release Notes

- BentoML 1.0 Community Demo
- Yatai Preview Release

Blog Posts

- New BentoML Organization Page!
- Join our community Slack

Community

Docs

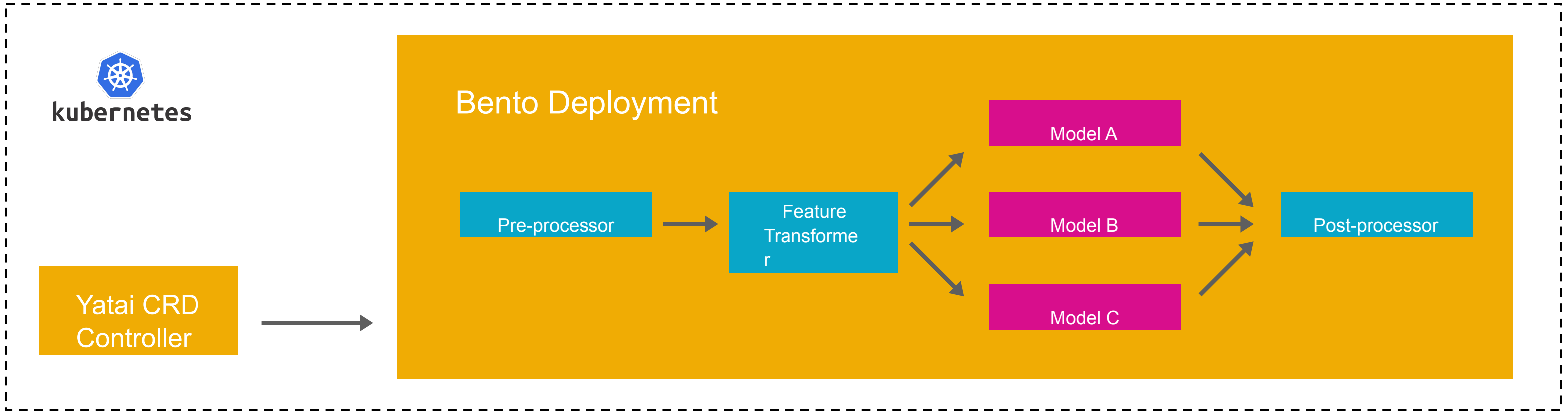
Contact

v0.0.1-9cb586c

```yaml
apiVersion: yatai.bentoml.org/v1beta1
kind: BentoDeployment
metadata:
  ...
spec:
  bento_tag: 'fraud_detector:dpijemevl6nlhlg6'
  autoscaling:
    minReplicas: 3
    maxReplicas: 20
    metrics:
    - type: Resource
      resource:
        name: cpu
      ...
  resources:
    limits:
      cpu: 500m
    requests:
      cpu: 200m
  runners:
    model_runner_a:
      autoscaling:
        minReplicas: 1
        maxReplicas: 5
        metrics:
        - type: Requests
          resource:
            name: backlog
```

BENTOML

High-performance model serving at scale



- Auto-scale at individual runner level to eliminate bottleneck
- Automatically adjust batching parameter based on traffic
- Kubernetes native, advanced CI/CD made easy

BENTOML

Thank you!

github.com/bentoml

chaoyu@bentoml.com