KubeCon | CloudNativeCon

Europe 2023

KubeCon | CloudNativeCon

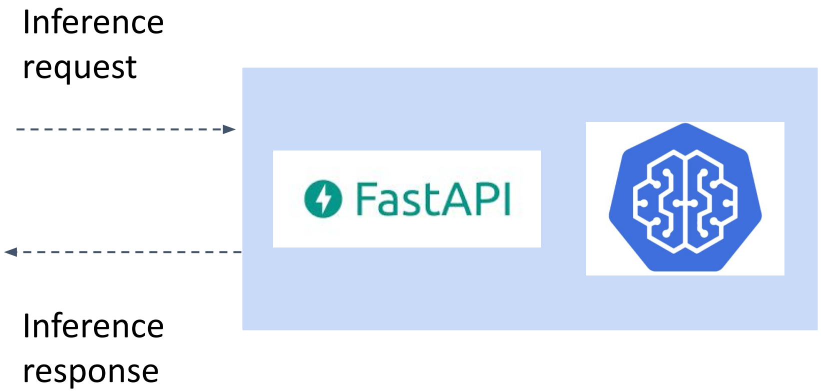
Europe 2023

The State and Future of Cloud Native ML Model Serving

Dan Sun, Bloomberg
Theofilos Papapanagiotou, AWS

Productionize AI Model is Challenging

"Launching AI application pilots is deceptively easy, but deploying them into production is notoriously challenging."

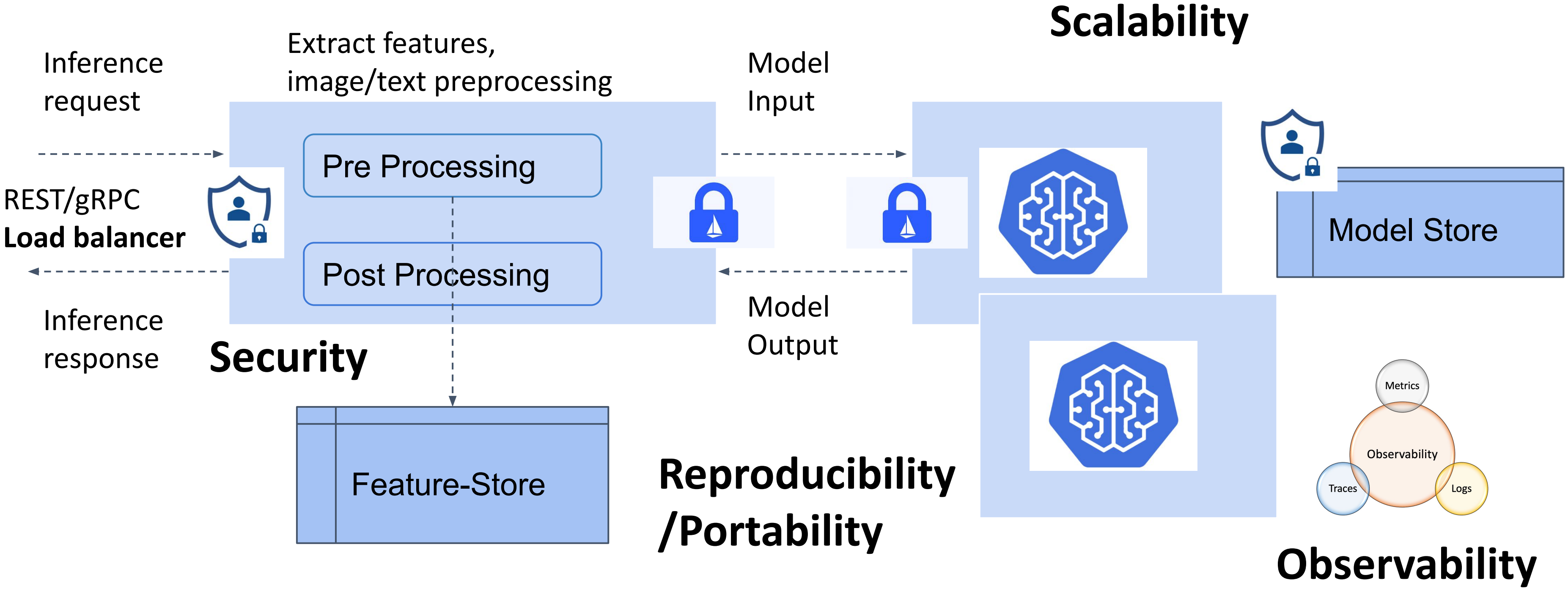


Inference request

Inference response

Productionize AI Model is Challenging

"Launching AI application pilots is deceptively easy, but deploying them into production is notoriously challenging."



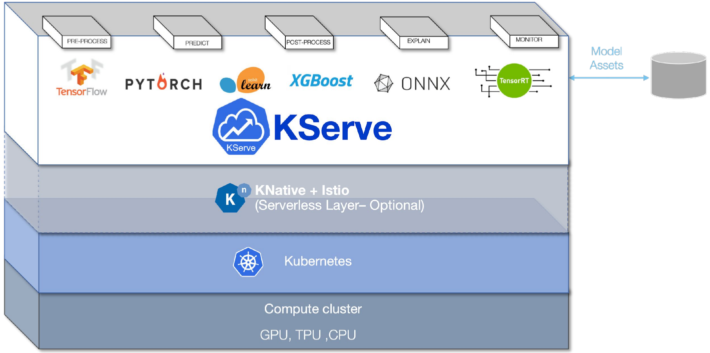
Why Kubernetes is a Great Platform for Serving Models?

- **Microservice:** Kubernetes handles container orchestration, facilitate deployment of microservices with resource management capabilities and load balancing.
- **Reproducibility/Portability**: ML model deployments can be written once, reproduced, and run with declarative yaml in a cloud agnostic way.
- **Scalability**: Kubernetes provides horizontal scaling features for both CPU and GPU workload.
- **Fault-tolerance**: Detect and recover from container failures, more resilient to outages and minimize downtime.

What’s KServe?

- **KServe** is a **highly scalable** and **standards**-based **cloud-native model inference platform** on **Kubernetes** for Trusted AI that encapsulates the complexity of deploying models to production.

- KServe can be deployed **standalone** or as an **add-on** component with **Kubeflow in the cloud or on-premises environment.**



KServe History

KubeCon CloudNativeCon North America 2019

Introducing KFServing: Serverless Model Serving on Kubernetes

Ellis Bigelow, Google & Dan Sun, Bloomberg

2019

2020

NVIDIA contributed Inference Protocol

RESILIENCE REALIZED

KubeCon CloudNativeCon North America 2021

Serving Machine Learning models Serverlessly at Scale Using KServe

Animesh Singh
IBM CTO and Distinguished Engineer
Watson AI and ML Open Technology

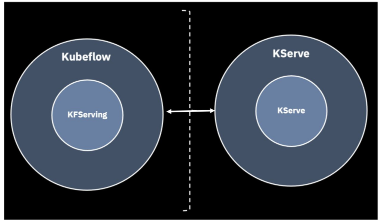
Twitter: @AnimeshSingh

IBM contributed ModelMesh

2021

2022

Bloomberg IBM NVIDIA. SELDON
Arrikto facebook inspur
cisco gojek Max Kelsen amazon
prosus Benevolent^AI CoreWeave vmware®



The Current State

- Features supported as of KServe 0.10 release

Core Inference

- Transformer/Predictor
- Serving Runtimes
- Custom Runtime SDK
- Open Inference Protocol
- Serverless Autoscaling
- Cloud/PVC Storage

Advanced Inference

- ModelMesh for Multi-Model Serving
- Inference Graph
- Payload Logging
- Request Batching
- Canary Rollout

Model Explanability & Monitoring

- Text, Image, Tabular Explainer
- Bias Detector
- Adversarial Detector
- Outlier Detector
- Drift Detector

Serving Runtimes

- **ModelSpec** specifies the model formats or version for trained model
- KServe automatically selects the serving runtime to instantiate the deployment that supports the given model format.

InferenceService (for user)

```
apiVersion: serving.kserve.io/v1beta1
kind: InferenceService
metadata:
  name: example-sklearn-isvc
spec:
  predictor:
    model:
      modelFormat:
        name: sklearn
        version: 1.0
      storageUri: s3://bucket/sklearn/mnist.joblib
      runtime: kserve-sklearnserver # optional
```

Serving Runtime (for KServe admin)

```
apiVersion: serving.kserve.io/v1alpha1
kind: ClusterServingRuntime
metadata:
  name: kserve-sklearnserver
spec:
  supportedModelFormats:
    - name: sklearn
      version: "1"
      autoSelect: true
  containers:
    - name: kserve-container
      image: kserve/sklearnserver:latest
      args:
        - --model_name={{.Name}}
        - --model_dir=/mnt/models
        - --http_port=8080
      resources:
        requests:
          cpu: "1"
          memory: 2Gi
```

Serving Runtime Support Matrix

Serving Runtime/ Model Format	scikit-learn	xgboost	lightgbm	TensorFlow	PyTorch	TorchScript	ONNX	MLFlow	Custom
MLServer (open)	✓	✓	✓					✓	✓
Triton (open)				✓		✓	✓		
TorchServe (v1, open)					✓	✓			
KServe Runtime (v1, open)	✓	✓	✓						✓
TFServing (v1)				✓					

Open Inference Protocol

- Open inference protocol enables a standardized high performance inference data plane.
- Allows interoperability between serving runtimes.
- Allows building client or benchmarking tools that can work with a wide range of serving runtimes
- It is implemented by KServe, MLServer, Triton, TorchServe, OpenVino, AMD Inference Server.

https://github.com/kserve/open-inference-protocol

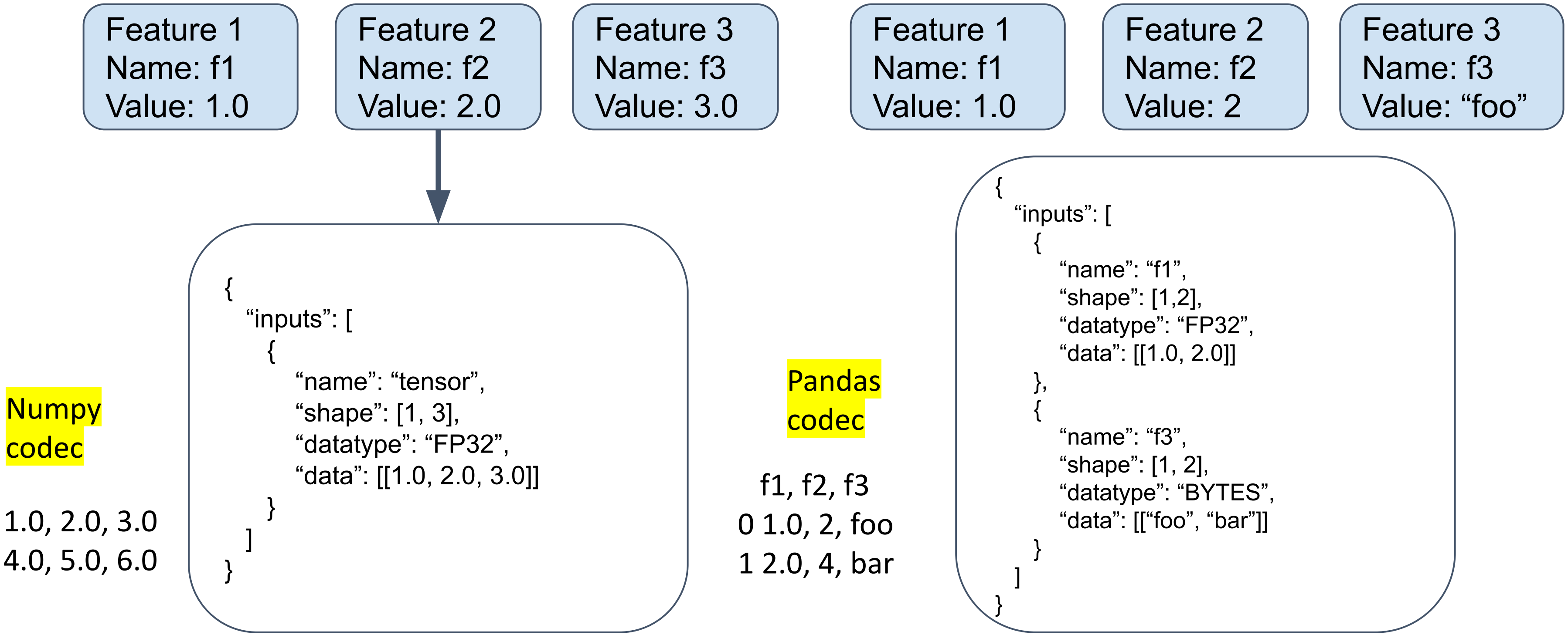
Open Inference Protocol

- REST vs. gRPC: Ease of use vs high performance

REST	gRPC
GET v2/health/live	rpc ServerLive(ServerLiveRequest) returns (ServerLiveResponse)
GET v2/health/ready	rpc ServerReady(ServerReadyRequest) returns (ServerReadyResponse)
GET v2/models/{model_name}/ready	rpc ModelReady(ModelReadyRequest) returns (ModelReadyResponse)
GET v2/models/{model_name}	rpc ModelMetadata(ModelMetadataRequest) returns (ModelMetadataResponse)
POST v2/models/{model_name}/infer	rpc ModelInfer(ModelInferRequest) returns (ModelInferResponse)

Open Inference Protocol Codec

- Single Input vs. Multiple Inputs with numpy/pandas codec



Feature 1
Name: f1
Value: 1.0

Feature 2
Name: f2
Value: 2.0

Feature 3
Name: f3
Value: 3.0

Numpy codec

```
1.0, 2.0, 3.0
4.0, 5.0, 6.0
```

```
{
  “inputs”: [
    {
      “name”: “tensor”,
      “shape”: [1, 3],
      “datatype”: “FP32”,
      “data”: [[1.0, 2.0, 3.0]]
    }
  ]
}
```

Feature 1
Name: f1
Value: 1.0

Feature 2
Name: f2
Value: 2

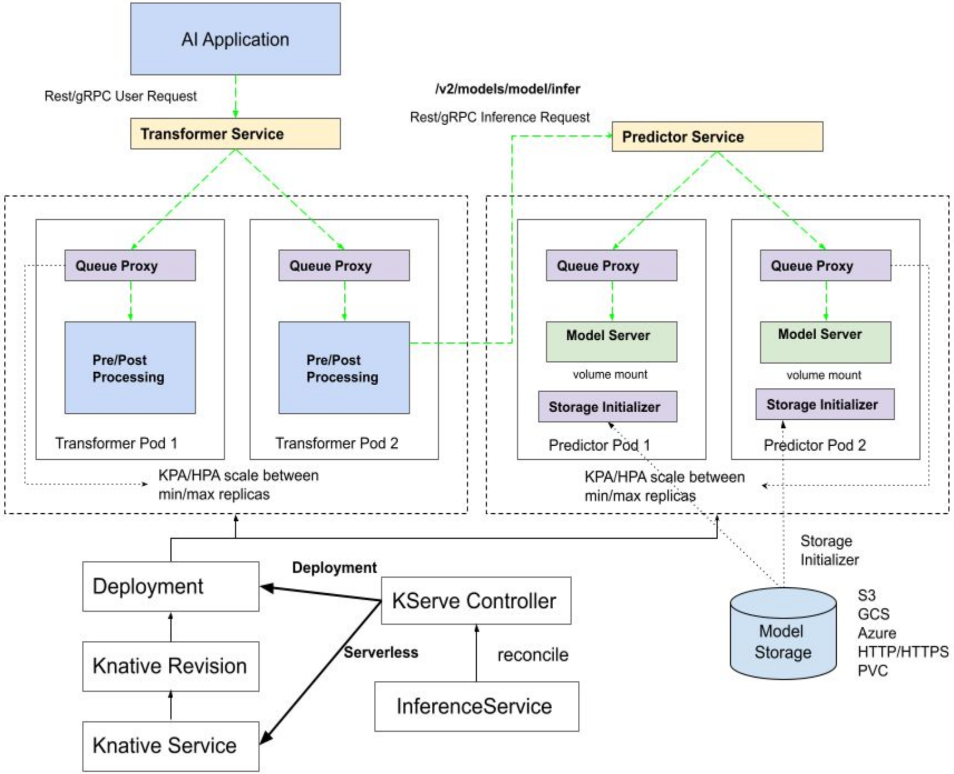
Feature 3
Name: f3
Value: “foo”

Pandas codec

```
   f1, f2, f3
0 1.0, 2, foo
1 2.0, 4, bar
```

```
{
  “inputs”: [
    {
      “name”: “f1”,
      “shape”: [1,2],
      “datatype”: “FP32”,
      “data”: [[1.0, 2.0]]
    },
    {
      “name”: “f3”,
      “shape”: [1, 2],
      “datatype”: “BYTES”,
      “data”: [[“foo”, “bar”]]
    }
  ]
}
```

Inference Service Deployment

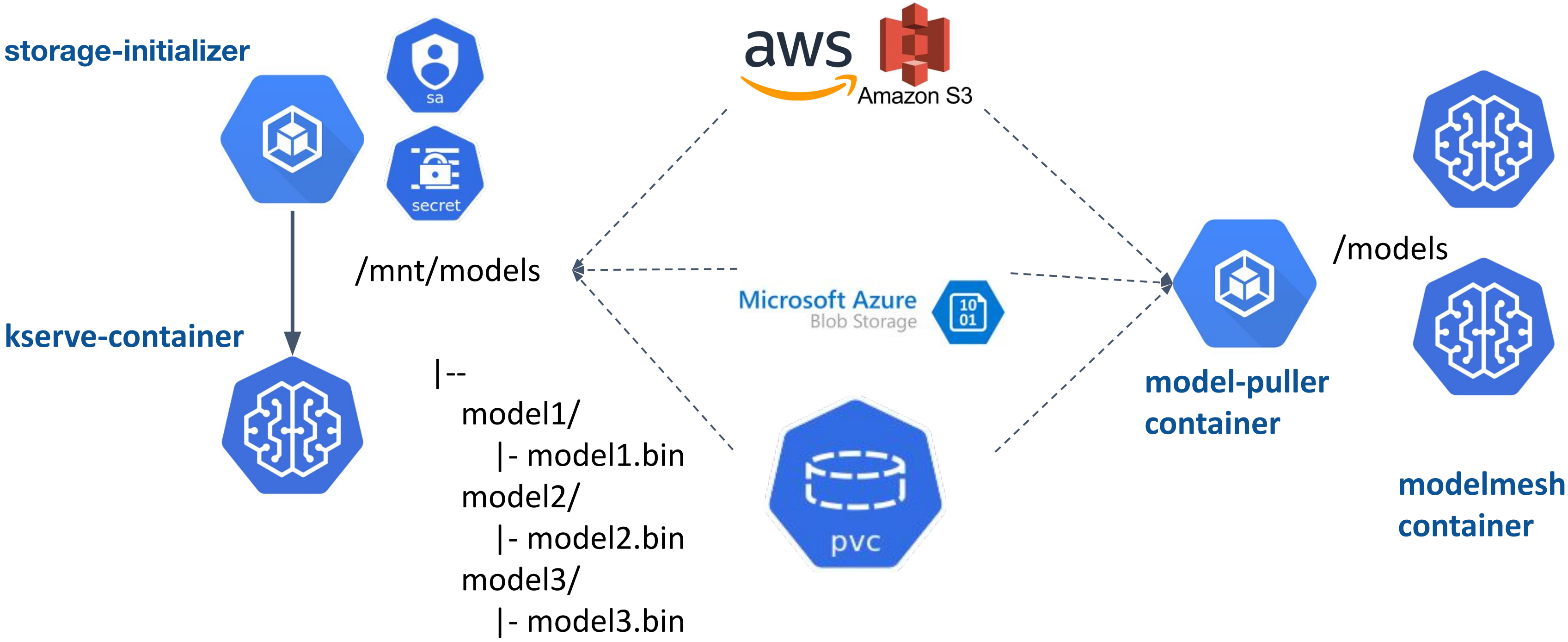


https://kserve.github.io/website/0.10/modelserving/v1beta1/transformer/feast/

```
apiVersion: "serving.kserve.io/v1beta1"
kind: "InferenceService"
metadata:
 name: "sklearn-feast-transformer"
spec:
 transformer:
  containers:
  - image:
kserve/driver-transformer:latest
    name: kserve-container
    command:
    - "python -m driver_transformer"
    args:
    - --entity_ids
    - driver_id
    - --feature_refs
    - driver_hourly_stats:acc_rate
    - driver_hourly_stats:avg_daily_trips
    - driver_hourly_stats:conv_rate
 Predictor:
  model:
    modelFormat:
     name: sklearn
    storageUri: "gs://pv/driver"
```

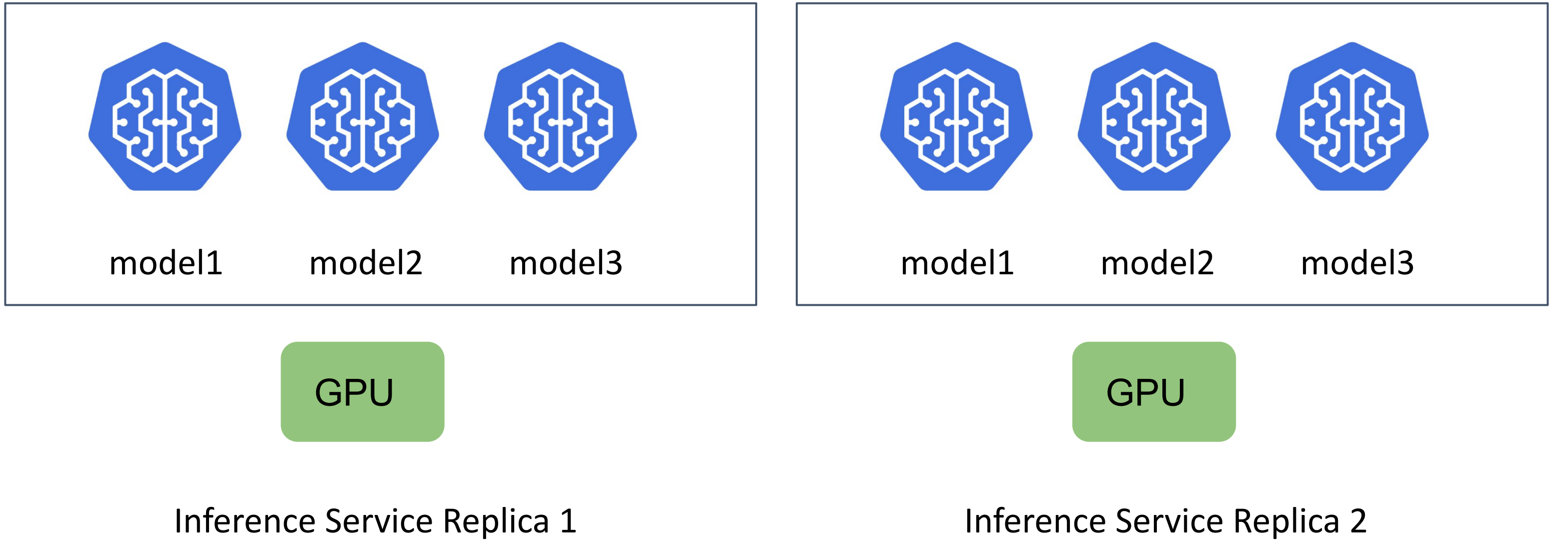
Model Storage Patterns

- Runs as init container to download the models before starting KServe container.
- Runs as sidecar to pull models as needed in ModelMesh.



Multi-Model Serving

- Single model single container usually under-utilize the resources.
- You can share the container resources by deploying multiple models, especially for the GPU sharing cases.
- When scaling up the same set of models need to be scaled out together.



model1 model2 model3

GPU

Inference Service Replica 1

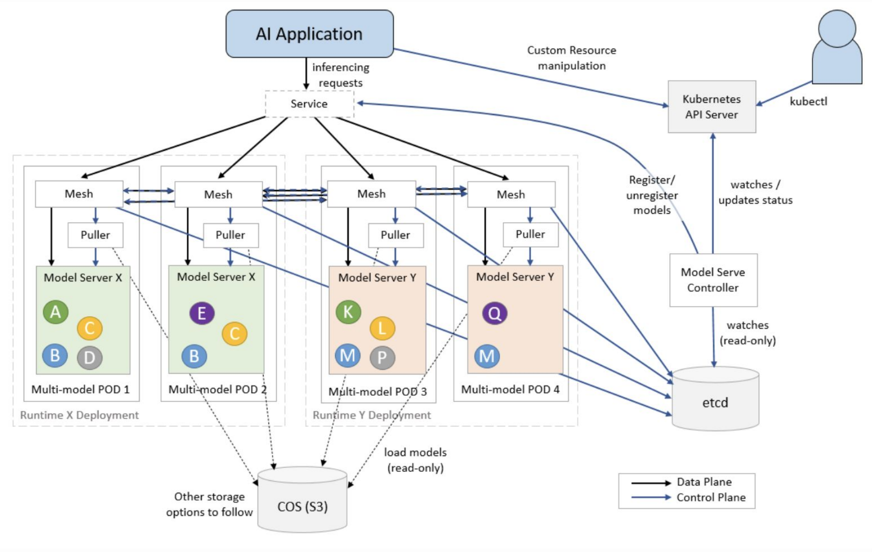
model1 model2 model3

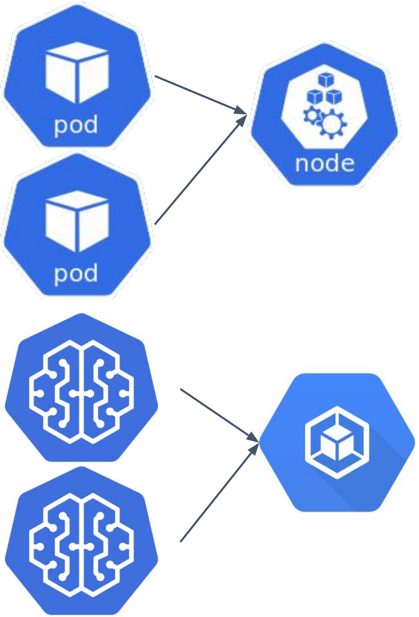
GPU

Inference Service Replica 2

ModelMesh

- Models can be scaled independently while sharing the container resource
- ModelMesh is a scale-out layer for model servers that can load and serve multiple models concurrently which uses the same KServe InferenceService API.
- Designed for high volume, high density, high churn production serving, it has powered IBM production AI services for a number of years and manages thousands of models.



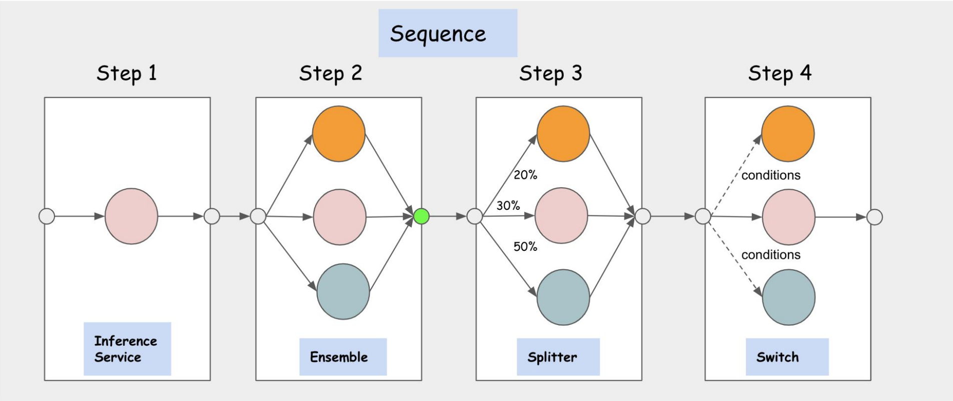


Kubernetes schedules pod onto nodes

ModelMesh schedules models onto container

Inference Graph

- Inference Graph is built for more complex **multi-stage inference pipelines**.
- Inference Graph is deployed in a **declarative way and highly scalable.**
- Inference Graph supports **Sequence, Switch, Ensemble and Splitter** nodes.
- Inference Graph is **highly composable**. It is made up with a list of routing nodes and each node consists of a set of routing steps which can be either route to an InferecenService or another node.



```
apiVersion: "serving.kserve.io/v1alpha1"
kind: "InferenceGraph"
metadata:
  name: "dog-breed-pipeline"
spec:
  nodes:
    root:
      routerType: Sequence
      steps:
      - serviceName: cat-dog-classifier
        name: cat_dog_classifier # step name
      - serviceName: dog-breed-classifier
        name: dog_breed_classifier
        data: $request
        condition: "[@this].#(predictions.0==\"dog\")"
```

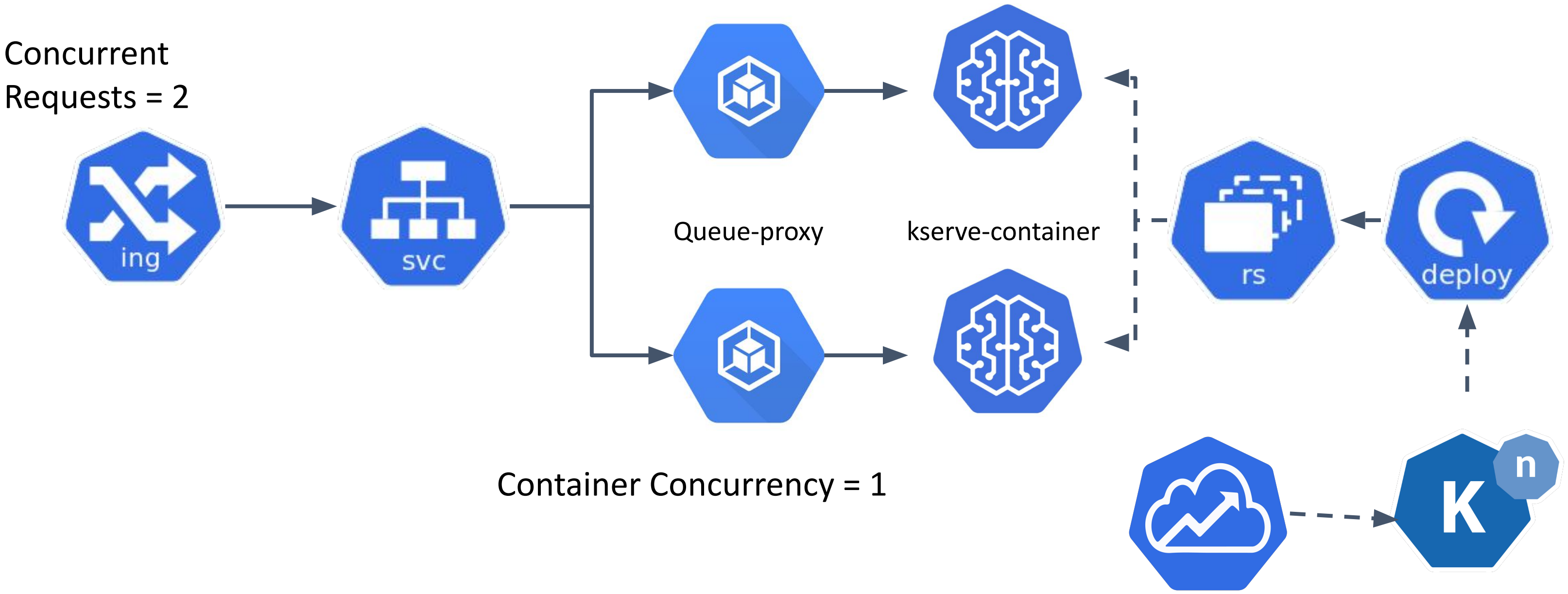
KServe Cloud Native Ecosystem

- **KServe** offers seamless integration with many CNCF projects to empower the production model deployments for security, observability, and serverless capabilities.

Open Policy Agent
envoy
Knative
Istio
OpenTelemetry
Buildpacks.io
spiffe
SPIRE
gRPC
cloudevents

Serverless Inference

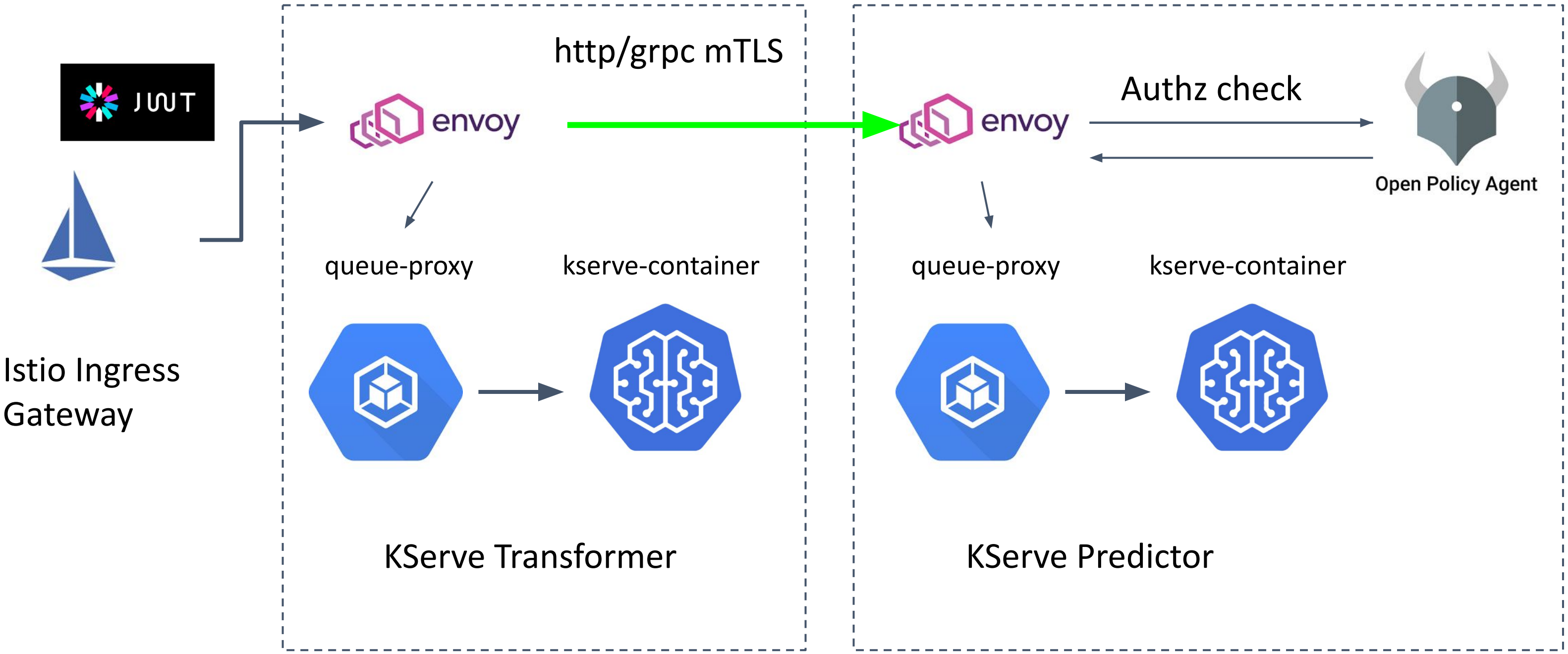
- Autoscaling based on incoming request RPS or Concurrency.
- Unified autoscaler for both CPUs and GPUs.



KnativeCon talk 2022

ServiceMesh: Secure Inference Service

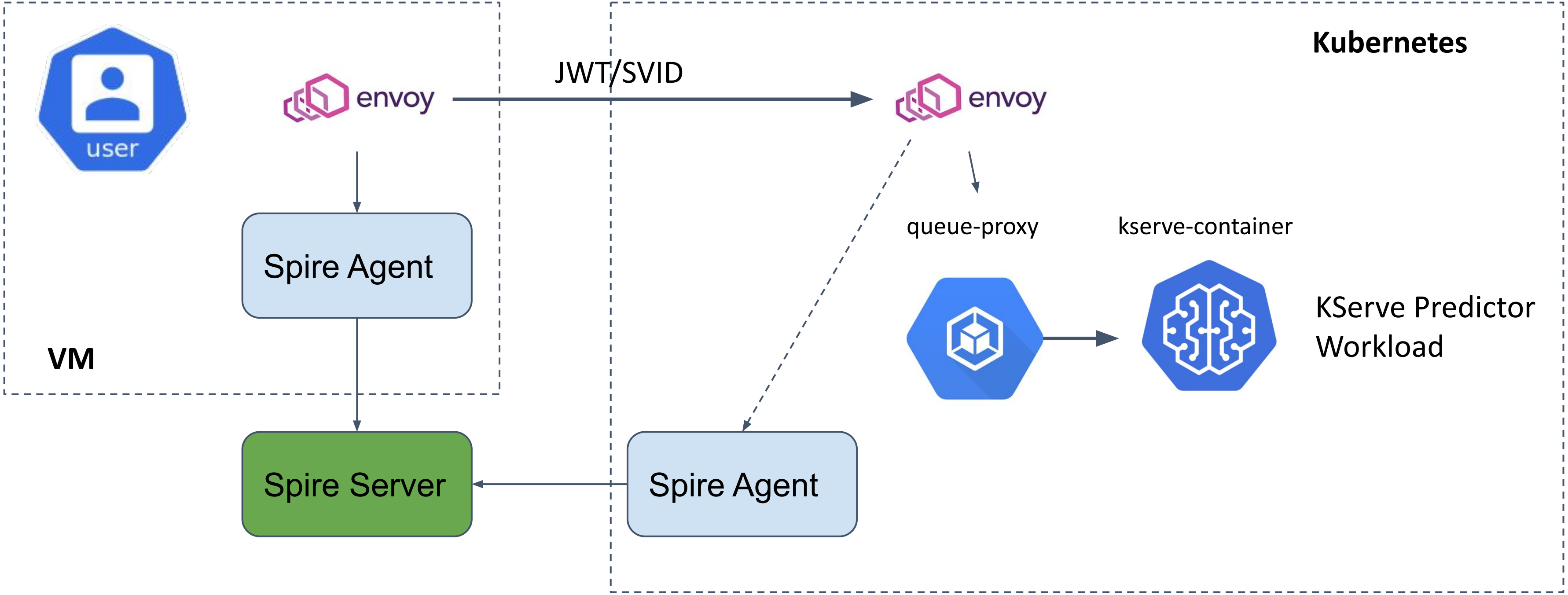
- Istio control plane mounts the client certificates to sidecar proxy to allow automatic authentication between transformer and predictor and encrypt service traffic.
- Authorization can be implemented with Istio Authorization Policy or Open Policy Agent.



SPIFFE and SPIRE: Strong Identity

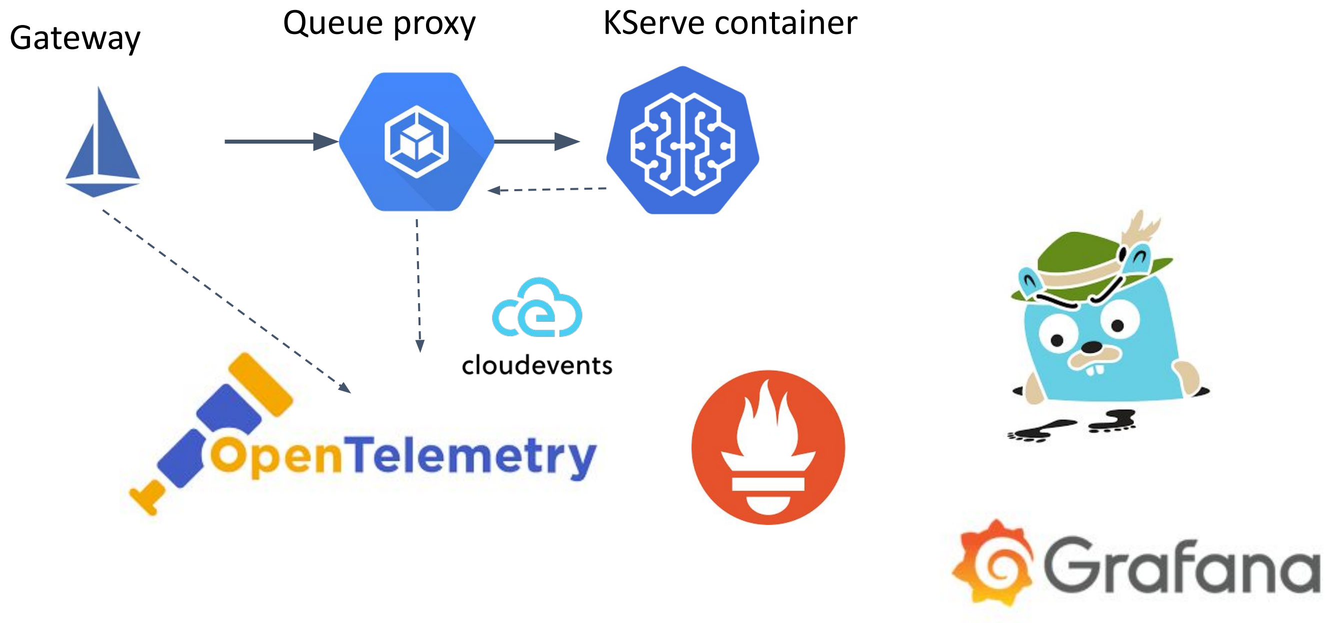
- **SPIFFE** is a secure identity framework that can be federated across heterogeneous environments.

- **SPIRE** distinguishes from other identity providers, such as API keys or secrets, with an attestation process before issuing the credential.



KServe Observability

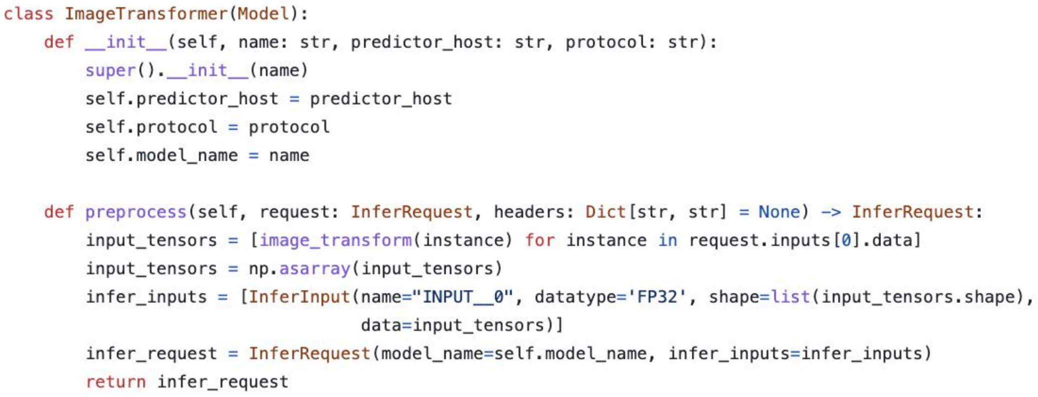
- **OpenTelemetry** can be used to instructment, collect and export metrics, log and tracing data that you can analyze to understand the production Inference Service performance.



Cloud Native Buildpacks

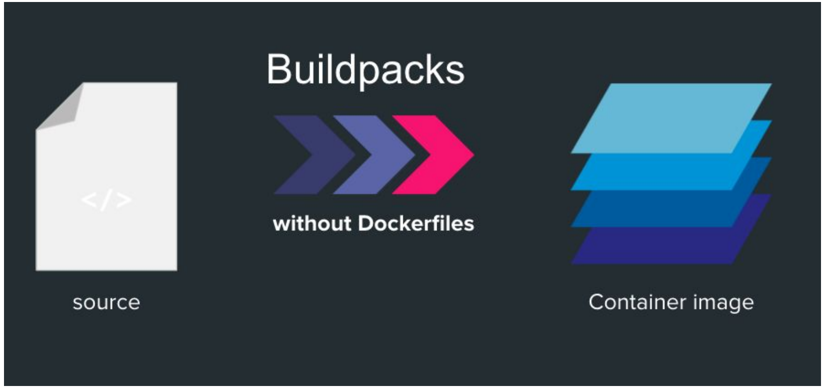
- Cloud Native Buildpacks can turn your custom transformer or predictor into a container image without Dockerfile and run anywhere.

- Ensures meeting the security and compliance requirements.



```
class ImageTransformer(Model):
    def __init__(self, name: str, predictor_host: str, protocol: str):
        super().__init__(name)
        self.predictor_host = predictor_host
        self.protocol = protocol
        self.model_name = name

    def preprocess(self, request: InferRequest, headers: Dict[str, str] = None) -> InferRequest:
        input_tensors = [image_transform(instance) for instance in request.inputs[0].data]
        input_tensors = np.asarray(input_tensors)
        infer_inputs = [InferInput(name="INPUT__0", datatype='FP32', shape=list(input_tensors.shape),
                                   data=input_tensors)]
        infer_request = InferRequest(model_name=self.model_name, infer_inputs=infer_inputs)
        return infer_request
```



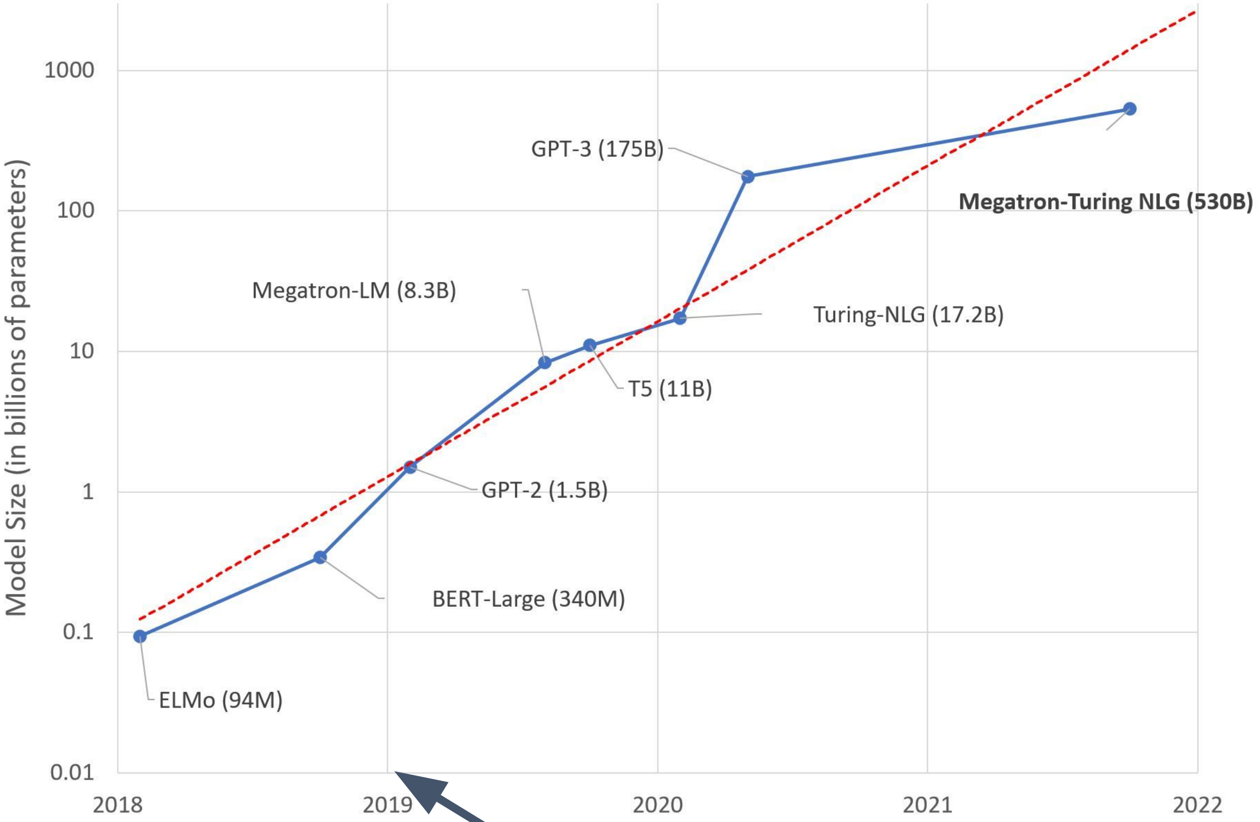
Looking Forward

KServe 1.0 Roadmap

- Graduate KServe core inference capabilities to stable/GA
- Open Inference Protocol coverage for all supported serving runtimes
- Graduate KServe SDK to 1.0
- Graduate ModelMesh and Inference Graph
- Large Language Model (LLM) Support
- KServe Observability and Security improvements
- Update KServe 1.0 documentation

https://github.com/kserve/kserve/blob/master/ROADMAP.md

Large Language Models



KServe released

Source: https://huggingface.co/blog/large-language-models

Distributed Inference for LLM

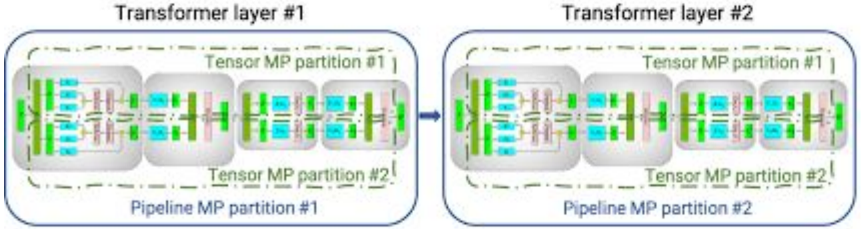
- Large transformer based models with hundreds of billions of parameters are computationally expensive.

- NVIDIA FasterTransformer implements an accelerated engine for inference of transformer based model spanning many GPUs and nodes in distributed manner.

- Huggingface Accelerate allows distributed inference with sharded checkpoints using less memory.

- Tensor parallelism allows running inference on multiple GPUs.

- Pipeline parallelism allows on multi-GPU and multi-node environment.



Large Language Model Challenges

- **Inference hardware requirements**
    - Cost: Requires significant computational resources with high end GPUs and large amount of memory
    - Latency: long response time up to tens of seconds

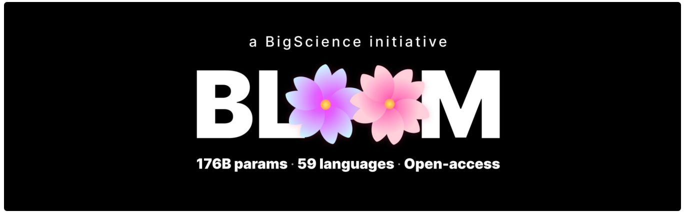
- **Model blob/file sizes in GBs(BLOOM):**
    - 176bln params = 360GB
    - 72 splits of 5GB which needs to mapped to multiple GPU devices

- **Model loading time**
    - From network (S3, minio) to instance disk
    - From instance disk to CPU RAM
    - From CPU RAM to GPU RAM

- **Model Serving Runtime:** FasterTransformer-Triton (32GB)

- **Model data can be sensitive and private for inference**

LLM(BLOOM 176bln) Inference Demo



```
curl https://fastertransformer.default.kubecon.theofpa.people.aws.dev/v1/models/fastertransformer:predict  -d '{
  "inputs":
  [
    {
      "input":"Kubernetes is the best platform to serve your models because",
      "output_len":"40"
    }
  ]
}'
```

[["Kubernetes is the best platform to serve your models because it is a cloud-based service that is built on top of the Kubernetes API. It is a great platform for developers who want to build their own Kubernetes application"]]

https://github.com/kserve/kserve/tree/master/docs/samples/v1beta1/triton/fastertransformer

LLM Download and Loading Time

```
$ git lfs clone https://huggingface.co/bigscience/bloom-560m

$ ls -lh bloom-560m
total 3.2G
-rw-r--r-- 1 theofpa domain^users  693 Apr 13 11:16 config.json
-rw-r--r-- 1 theofpa domain^users 1.1G Apr 13 11:22 flax_model.msgpack
-rw-r--r-- 1 theofpa domain^users  16K Apr 13 11:16 LICENSE
-rw-r--r-- 1 theofpa domain^users 1.1G Apr 13 11:22 model.safetensors
-rw-r--r-- 1 theofpa domain^users 1.1G Apr 13 11:22 pytorch_model.bin
-rw-r--r-- 1 theofpa domain^users  21K Apr 13 11:16 README.md
-rw-r--r-- 1 theofpa domain^users   85 Apr 13 11:16 special_tokens_map.json
-rw-r--r-- 1 theofpa domain^users  222 Apr 13 11:16 tokenizer_config.json
-rw-r--r-- 1 theofpa domain^users  14M Apr 13 11:16 tokenizer.json
```

Huggingface transformer to FasterTransformer conversion based on tensor parallelisms and target precision.

```
python3 FasterTransformer/examples/pytorch/gpt/utils/huggingface_bloom_convert.py\
 -o fastertransformer/1 -i ./bloom-560m/

$ ls -lhS fastertransformer/1 | head
total 2.1G
-rw-r--r-- 1 theofpa domain^users 980M Apr 13 11:28 model.wte.bin
-rw-r--r-- 1 theofpa domain^users  16M Apr 13 11:28 model.layers.0.mlp.dense_4h_to_h.weight.0.bin
-rw-r--r-- 1 theofpa domain^users  16M Apr 13 11:28 model.layers.0.mlp.dense_h_to_4h.weight.0.bin
-rw-r--r-- 1 theofpa domain^users  16M Apr 13 11:28 model.layers.10.mlp.dense_4h_to_h.weight.0.bin
-rw-r--r-- 1 theofpa domain^users  16M Apr 13 11:28 model.layers.10.mlp.dense_h_to_4h.weight.0.bin
-rw-r--r-- 1 theofpa domain^users  16M Apr 13 11:28 model.layers.11.mlp.dense_4h_to_h.weight.0.bin
-rw-r--r-- 1 theofpa domain^users  16M Apr 13 11:28 model.layers.11.mlp.dense_h_to_4h.weight.0.bin
-rw-r--r-- 1 theofpa domain^users  16M Apr 13 11:28 model.layers.12.mlp.dense_4h_to_h.weight.0.bin
-rw-r--r-- 1 theofpa domain^users  16M Apr 13 11:28 model.layers.12.mlp.dense_h_to_4h.weight.0.bin
```

```
storage-initializer:

2023-04-13 19:28:34.151 1 root INFO [downlo
Copying contents of s3://kubecon-models/mo
2023-04-13 19:29:28.922 1 root INFO [downlo
Successfully copied s3://kubecon-models/mo
/mnt/models

predictor:

I0413 19:29:34.205155 1 libfastertransforme
TRITONBACKEND_ModelInitialize: fastertrans
1)
I0413 19:29:45.448307 1 model_lifecycle.cc:
successfully loaded 'fastertransformer' ve
```

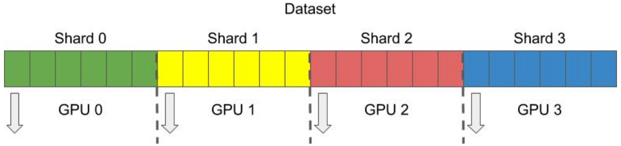


Image source: https://docs.nvidia.com/deeplearning/dali/user-guide/docs/examples/general/multigpu.html#Sharding

BloombergGPT - Powered by KServe

Introducing BloombergGPT, Bloomberg's 50-billion parameter large language model, purpose-built from scratch for finance

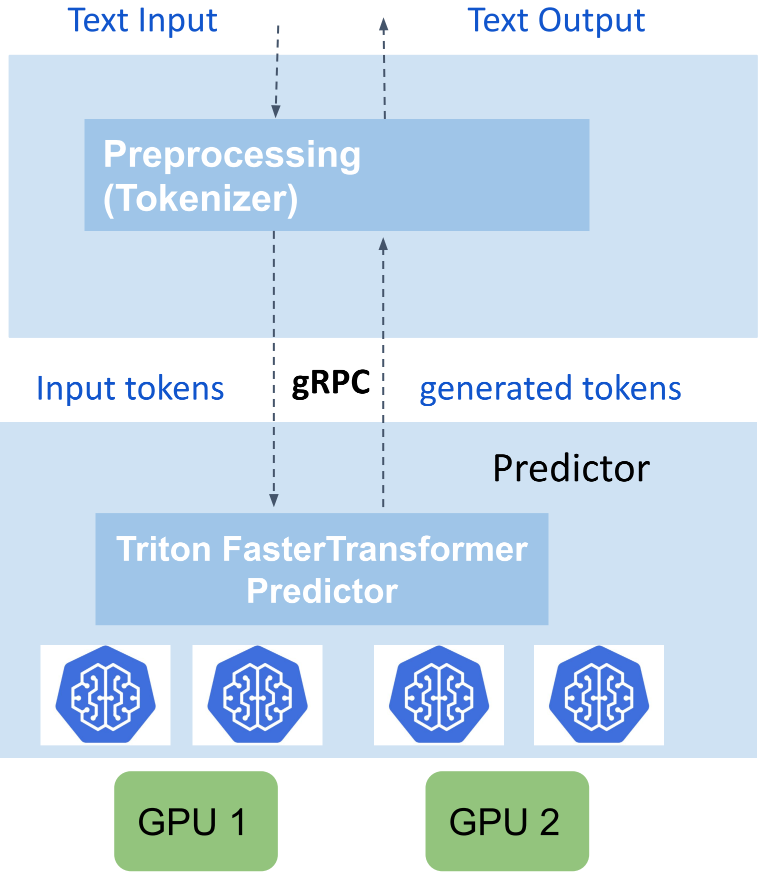
- Trained approximately 53 days on 64 servers, each with 8 A100 GPUs on AWS Sagemaker.

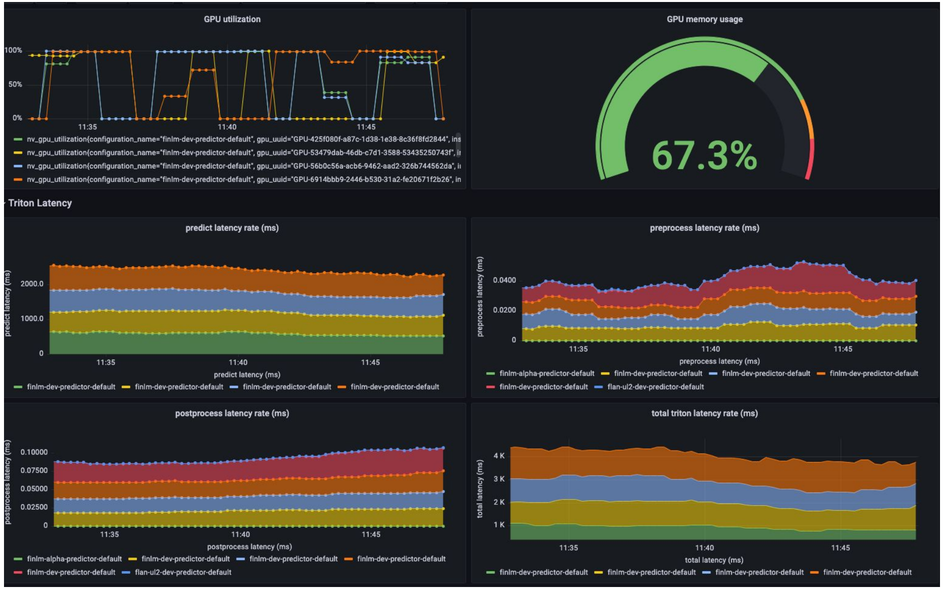
- HF checkpoints are converted into FasterTransformer format setting target precision to BF16, tensor parallelism 2 and pipeline parallelism 1.

- Deployed on KServe on-prem for internal product development with NVIDIA Triton FasterTransformer Serving Runtime on 2 A100 GPU (80G memory) for each replica.

https://www.bloomberg.com/company/press/bloomberggpt-50-billion-parameter-llm-tuned-finance/
BloombergGPT Paper: https://arxiv.org/abs/2303.17564

BloombergGPT - KServe Deployment




GPU utilization
100%
50%
0%
11:35
11:40
11:45
nv_gpu_utilization{configuration_name="finlm-dev-predictor-default", gpu_uuid="GPU-425f080f-a87c-1d38-1e38-8c36f8fd2844", ins
nv_gpu_utilization{configuration_name="finlm-dev-predictor-default", gpu_uuid="GPU-53479dab-46db-c7d1-3588-53435250743f", in
nv_gpu_utilization{configuration_name="finlm-dev-predictor-default", gpu_uuid="GPU-56b0c56a-acb6-9462-aad2-326b744562da", in
nv_gpu_utilization{configuration_name="finlm-dev-predictor-default", gpu_uuid="GPU-6914bbb9-2446-b530-31a2-fe20671f2b26", in
GPU memory usage
67.3%
Triton Latency
predict latency rate (ms)
predict latency (ms)
2000.0
1000.0
0
11:35
11:40
11:45
predict latency (ms)
finlm-dev-predictor-default
finlm-dev-predictor-default
finlm-dev-predictor-default
finlm-dev-predictor-default
preprocess latency rate (ms)
preprocess latency (ms)
0.0400
0.0200
0
11:35
11:40
11:45
preprocess latency (ms)
finlm-alpha-predictor-default
finlm-dev-predictor-default
finlm-dev-predictor-default
finlm-dev-predictor-default
finlm-dev-predictor-default
flan-ul2-dev-predictor-default
postprocess latency rate (ms)
postprocess latency (ms)
0.10000
0.07500
0.05000
0.02500
0
11:35
11:40
11:45
postprocess latency (ms)
finlm-alpha-predictor-default
finlm-dev-predictor-default
finlm-dev-predictor-default
finlm-dev-predictor-default
finlm-dev-predictor-default
flan-ul2-dev-predictor-default
total triton latency rate (ms)
total latency (ms)
4 K
3 K
2 K
1 K
11:35
11:40
11:45
total latency (ms)
finlm-dev-predictor-default
finlm-dev-predictor-default
finlm-dev-predictor-default
finlm-dev-predictor-default

KServe Community

- KServe Website
- KServe on GitHub
- KServe Community

Please scan the QR Code above
to leave feedback on this session