
KubeCon
CloudNativeCon
North America 2024

KubeCon | CloudNativeCon

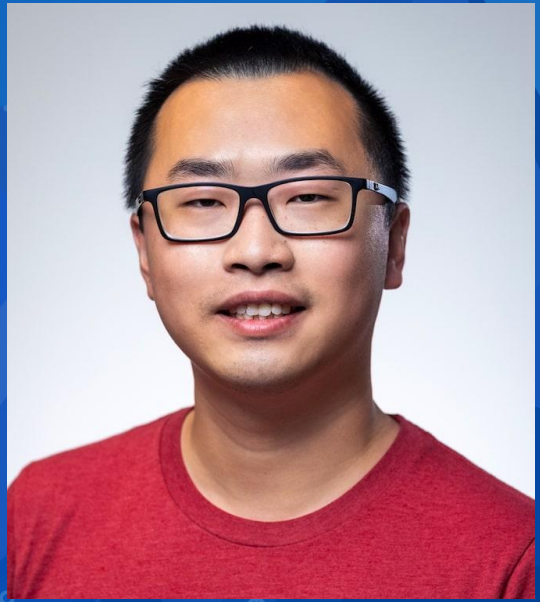
North America 2024

WG Serving: Accelerating AI/ML Inference Workloads on Kubernetes

Yuan Tang, Red Hat
Eduardo Arango, NVIDIA

KubeCon CloudNativeCon North America 2024

Who am I?



Project Lead: Argo and Kubeflow

Co-Chair: K8s WG Serving

Maintainer: KServe, XGBoost, TensorFlow, etc.

Author: *Distributed Machine Learning Patterns, etc.*

Social handle: @TerryTangYuan

Principal Software Engineer @ Red Hat

KubeCon CloudNativeCon North America 2024

Who am I?

- Sylabs - Singularity Software engineer
- Red Hat - OpenShift Senior Software engineer
- NVIDIA - Cloud Native Senior Systems Software Engineer

- PhD in Computer Science focused on DevOps for HPC

Carlos Eduardo Arango Gutierrez, PhD

Senior Systems Software Engineer @ NVIDIA

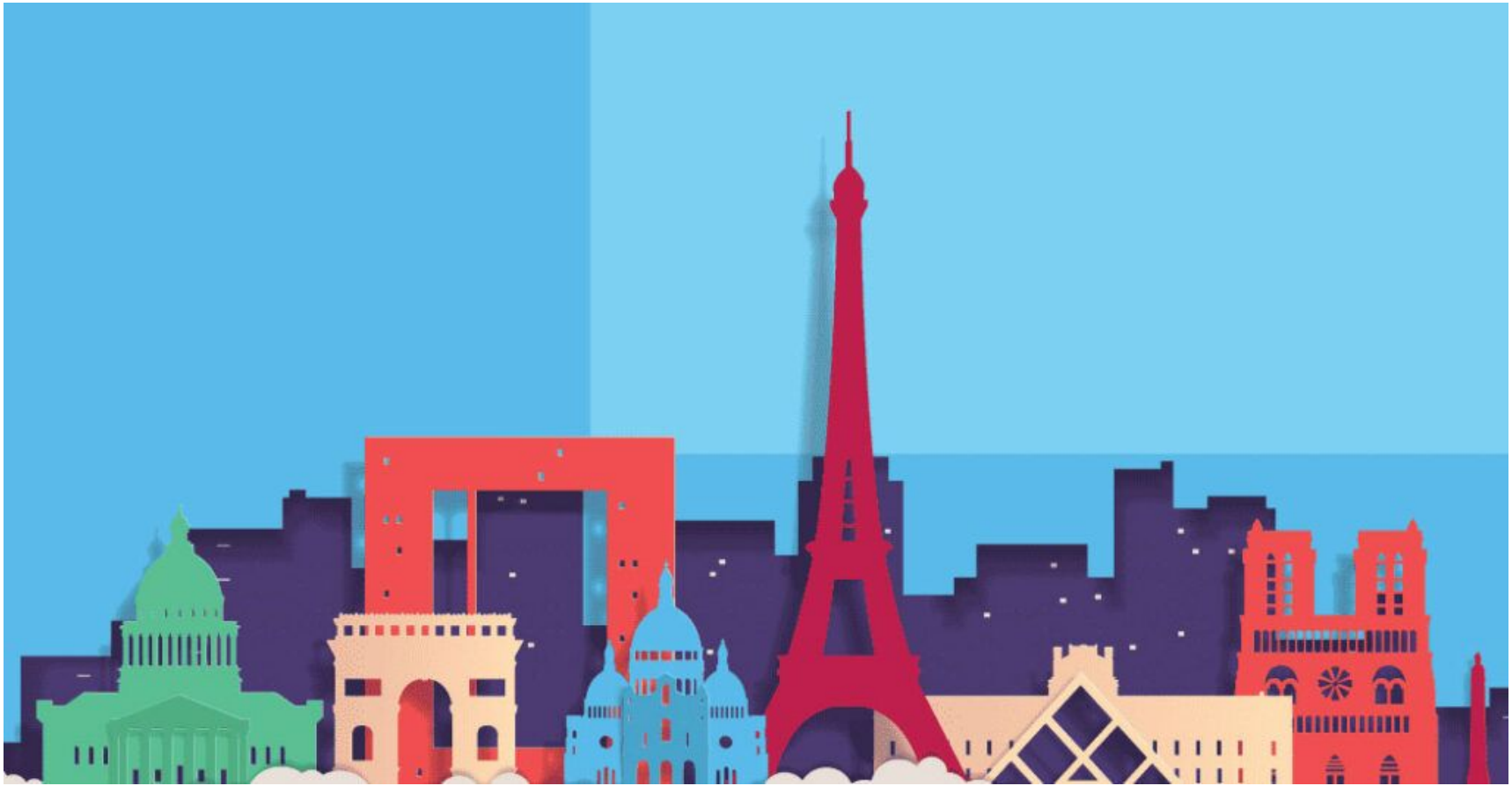
KubeCon | CloudNativeCon

North America 2024

But...How did it get started?

Once upon a time in Paris…

How did it get started?



How did it get started?

So why not WG-Inference?

Yeah we get that question a lot…

KubeCon | CloudNativeCon

North America 2024

WG's all the way

Why we needed a specific WG

WG’s all the way

Pre-Kubecon Paris 2024 we had

- K8s WG-Batch
- CNCF Batch
- CNCF AI WG

After we added:

- K8s WG-Serving
- K8s WG-Device-Management

WG Leadership

with 250+ community members!

KubeCon

CloudNativeCon

North America 2024

Why is it important?

What are the missions/goals of this WG?

Our goals

- **Enhance Kubernetes Workload Controllers** Improve Kubernetes workload controllers for seamless integration with hardware accelerators, and optimize their performance with popular inference serving frameworks and model servers.

- **Investigate Orchestration and Scalability Solutions** Explore and contribute to new projects that enhance orchestration, scaling, and load balancing for inference workloads, ensuring compatibility with other Kubernetes-based workloads.

- **Optimize Resource Sharing for Serving Workloads** Enable safe execution of serving workloads on Kubernetes while dynamically allocating unused capacity to batch processing frameworks, optimizing resource utilization.

KubeCon | CloudNativeCon

North America 2024

How is this WG operated?

Divide and conquer!

WG-Serving: Orchestration

- Identify challenges in implementing high-level abstractions for orchestrating serving workloads
- Work closely with ecosystem projects, e.g. KServe, Ray, etc.
- Collect solved challenges, pain points, and use cases from these projects
- Relevant subprojects:
  - Serving Catalog
  - LLM Instance Gateway

WG-Serving: Multi-node/Multi-host

- Extracting patterns and solving challenges for multi-host or multi-node inference
- Discussing multi-host/multi-node inference implementations, cost-effectiveness, capacity optimization
- Increased demand for serving large models on multiple nodes
- LeaderWorkerSet (LWS): common multi-host deployment patterns where large models are sharded
- Multi-node support in orchestration tools, e.g. KServe (ready for testing)

WG-Serving: Autoscaling

- Understanding the autoscaling needs, techniques and nuances of running inference workloads.
- Breaking down the various user objectives/intents and how each influences how one could set up autoscaling.
- Coming together and sharing knowledge around optimizing the quickly changing landscape of inference workloads.
- Closely collaborating with the various initiatives:
  - Benchmarking was created out of this workstream to establish a common way of measuring baseline performance with the model scale patterns and to quantify future improvements.
  - Contributing best-practices around configuring autoscaling in the Serving Catalog.
  - Leveraging the Inference Gateway to provide autoscaling that works hand-in-hand with the load balancer.
  - Standardizing metrics with OpenTelemetry
- Standardizing on model server metrics
- Distributing models via OCI VolumeSource

WG-Serving: DRA

- Helped push for BETA on 1.32!
- Working on formulating requirements for DRA
  - Partitioned devices support
  - Multihost serving
- Device failures handling use cases for DRA, and to k8s in general

Current Initiatives

Current Initiatives: LLM Instance Gateway

Repo: [kubernetes-sigs/llm-instance-gateway]

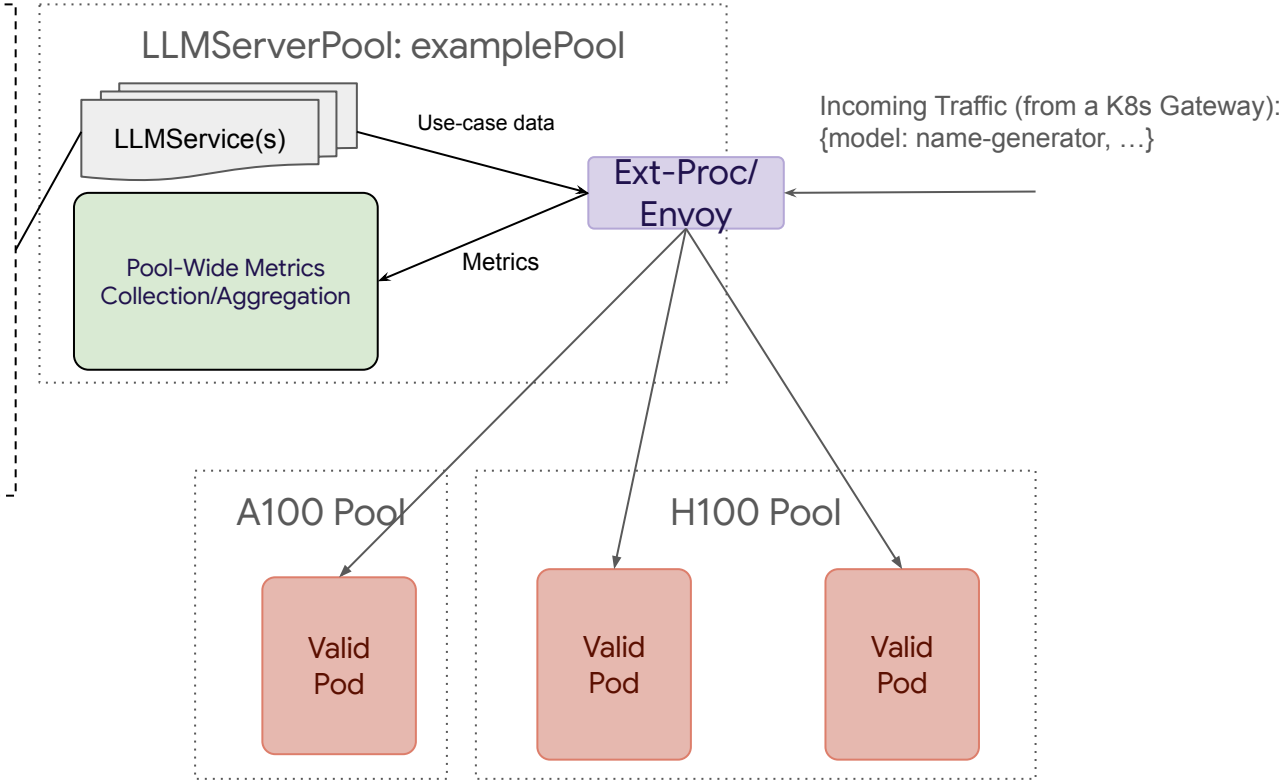
Envoy-based tooling to support:

- Efficient accelerator sharing
  - Many LoRA adapters on shared hardware, with intelligent routing to maximize throughput
- Operational resilience
  - Fairness across LLM Services while respecting:
    - distinct priorities & distinct latency objectives
- Fast reconfiguration
  - Simplify rollout of new adapters for LLM Services; allowing for gradual, safe rollout of new adapters

Current Initiatives: LLM Instance Gateway

Example LLMService

```
{
modelName: 'name-generator',
poolRefs: ['examplePool'],
TargetModels: [
  {
    name:'name-generator-v3',
    weight: 20
  },
  {
    name: 'name-generator-v2',
    weight: 80,
  }]
}
```



DRA in 1.32

- Merged in 1.32
  - Structured parameters (DRA MVP) graduated to **Beta**
  - Faster scheduling (up to 16x)
  - Removal of classic DRA
  - Driver-owned resource claim status (for multi-networking use cases, primarily)
  - Significant progress on autoscaler integration
- Drivers for 1.32 (out-of-tree)
  - Example driver (ready)
  - NVIDIA DRA Driver for GPUs (ready)
  - CNI DRA Driver (in progress)
  - Google TPU Driver (in progress)
- Missed but on track for alpha in 1.33:
  - Prioritized alternatives in device requests
  - Support for partitionable devices
  - DRA resource health status in Pod Status
- Many more to come…

OCI Volume Source

- Allow to use an OCI image as a Volume Source
- Standardizing on model weights distribution
  - Versioning
  - Distribution format
  - Optimizations

[VolumeSource: OCI Artifact and/or Image · Issue #4639 · kubernetes/enhancements · GitHub]

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: example-pod
spec:
  volumes:
  - name: oci-volume
    image:
      reference: "example.com/my-image:latest"
      pullPolicy: IfNotPresent
  containers:
  - name: my-container
    image: busybox
    volumeMounts:
    - mountPath: /data
      name: oci-volume
```

Current Initiatives: Serving Catalog

Repo: https://github.com/kubernetes-sigs/wg-serving/tree/main/serving-catalog

- Provides working examples for:
  - Popular model servers: vLLM, TGI, Jetstream, Triton/TensorRT, etc.
  - Popular open models: Llama, Mistral, etc.
  - Different deployment patterns: single or multi-host inference.
  - Different primitives or orchestration frameworks: Deployment, LWS, KServe InferenceService
- Explores recommended configurations and patterns for inference workloads
  - Common parameters that users frequently need to tweak and vary
  - Differences of cloud provider specific configurations and accelerators
- Stores templated working examples of popular frameworks, model servers, and models.
  - [Public] Kustomized Blueprints - Serving Catalog
  - [Public] K8s LLM Serving Catalog
- Supports multiple runtimes like vLLM and JetStream and several models, with more model server / model support in progress.
- Contributions from autoscaling for HPA configurations.
- Upcoming support for multi-host inference templates.

Current Initiatives: Benchmarking Tool

Current state:

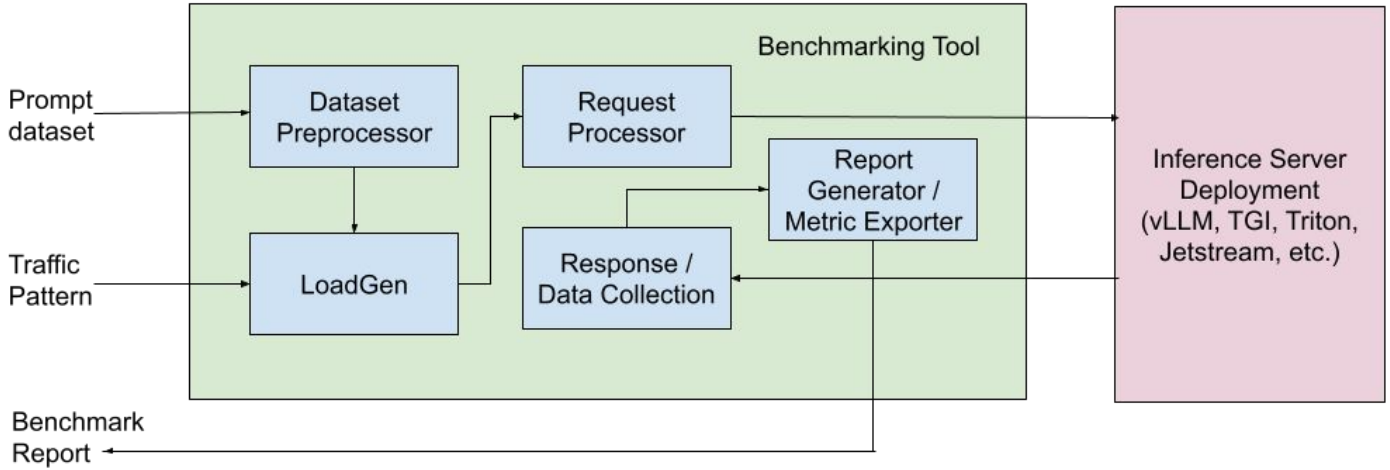
- IBM benchmarking tool: fmperf (foundation model performance)
  - https://github.com/fmperf-project/fmperf
  - Launch inference servers on k8s from developer machine and measure performance based on load generation
  - Distributed load testing
- GKE benchmarking tool
  - https://github.com/GoogleCloudPlatform/ai-on-gke/tree/main/benchmarks
  - Distributed load testing to support higher concurrency load gen
  - Run automated benchmarks on GKE for AI workloads via Terraform automation
- Red Hat benchmarking tool
  - https://github.com/openshift/ols-load-generator/
  - OpenShift HTTP Benchmarking of RAG applications

Current Initiatives: Benchmarking Tool

Future state: one standard benchmarking tool built collaboratively

Goals:

- Benchmark-as-code tool that can be used as a library
- Model server and hardware agnostic
- Simple to deploy on Kubernetes or run independently
- Solve different benchmarking uses like autoscaling, LoRA use cases with instance gateway, etc.



KubeCon | CloudNativeCon

North America 2024

Involvement in Ecosystem Projects

All together building community!

Ecosystem Projects

KServe

- LLM Instance Gateway integration
- Serving Catalog contribution
- Leverage benchmarking tool
- Identify feature gaps through WG discussions
- Roadmap: https://github.com/kserve/kserve/blob/master/ROADMAP.md

Blueprint Feature	Similar KServe Feature (if any)	Missing Parts from KServe (if any)	Implementation Plan in KServe (if applicable)
Serving Class	ServingRuntime		
Single-host Serving	KServe Raw deployment mode	Ability to define various policies	Make the additional functionalities configurable
Multi-host Serving	N/A	Not supported yet	Multi-host support implementation ready for testing: https://github.com/kserve/kserve/pull/3972
Model Policy (supported model spec, single/multi-model support)	supportedModelFormat, ModelMesh	N/A	Plan to add direct support for multi model for KServe InferenceService
ModelSpec.DataSource	storageUrl, supportedUriFormats	N/A	N/A
ModelSpec.Adapters	N/A	LoRA is not supported yet	Plan to support multiple StorageUris: https://github.com/kserve/kserve/issues/3413
Server Policy (mandatory and optional flags, resource spec)	Args and resource spec in ServingRuntime	N/A	N/A
Autoscaling Policy	scaleTarget, scaleMetric	More granular and metric-based auto scaling	Plan to integrate with KEDA: KServe Integration …
Resource Policy (accelerator type and resource requirements)	Resource spec in ServingRuntime	N/A	N/A

WG-Serving & KServe Talks

AI Day, Sponsored Keynote: Advancing Cloud Native AI Innovation Through Open Collaboration

Optimizing Load Balancing and Autoscaling for Large Language Model (LLM) Inference on Kubernetes

Unlocking Potential of Large Models in Production

Engaging the KServe Community, The Impact of Integrating a Solutions with Standardized CNCF Projects

Best Practices for Deploying LLM Inference, RAG and Fine Tuning Pipelines on K8s

Getting Involved

- WG information (calendar, workstreams, organizers, subprojects, etc.)
  - https://github.com/kubernetes/community/tree/master/wg-serving
- #wg-serving on K8s Slack Workspace
- Reach out to co-chairs and subproject leads

KubeCon | CloudNativeCon

North America 2024