
dapr
HELM
TiKV
KEDA
KubeCon
CloudNativeCon
China 2025

KubeCon CloudNativeCon China 2025

# Build a Large Model Inference Platform for Heterogeneous Chips Based on vLLM

**Haiwen Zhang**
ChinaMobile

Kante Yin
DaoCloud

# Large Model Inference Platform



Our large model inference platform, built on a cloud-native infrastructure, provides MaaS externally.

| | | | | | |
|---|---|---|---|---|---|
| MAAS Platform | API | Agent | Plugin | DataSet | ... |
| Models | deepseek | Qwen | 智谱清言 | 零一万物 01.AI | ... |
| Inference Engine | vLLM | | | | |
| CloudNative Infra | Ilmaz | Kubernetes | Istio | Prometheus | Harbor ... |
| Compute Infra | NVIDIA | Ascend | 天数智芯 Iluvatar CoreX | 壁仞科技 BIREN TECHNOLOGY | 昆仑芯 KUNLUNXIN ... |

# Challenges For Sailing LLM Inference



## Heterogeneous Cluster

- A variety of chips
- Hight-end GPU stockout frequently
- Cost & Performance tradeoffs

# Challenges For Sailing LLM Inference



## Heterogeneous Cluster

- A variety of chips
- Hight-end GPU stockout frequently
- Cost & Performance tradeoffs

## AI Gateway

- LLM-wise routing, e.g. KVCache, LoRA, load
- Token usage tracking to achieve model-specific demands, e.g. token-based rate limiting
- Failover across providers and hosted models

# Challenges For Sailing LLM Inference



## Heterogeneous Cluster

- A variety of chips
- Hight-end GPU stockout frequently
- Cost & Performance tradeoffs

## AI Gateway

- LLM-wise routing, e.g. KVCache, LoRA, load
- Token usage tracking to achieve model-specific demands, e.g. token-based rate limiting
- Failover across providers and hosted models

## Advanced Orchestration

- Multi-host inference for DeepSeek R1 etc.
- xPyD paradigm for disaggregated serving
- LLM-focused autoscaling, e.g. queue size, batch size

# Challenges For Sailing LLM Inference



## Heterogeneous Cluster

- A variety of chips
- Hight-end GPU stockout frequently
- Cost & Performance tradeoffs

## AI Gateway

- LLM-wise routing, e.g. KVCache, LoRA, load
- Token usage tracking to achieve model-specific demands, e.g. token-based rate limiting
- Failover across providers and hosted models

## Advanced Orchestration

- Multi-host inference for DeepSeek R1 etc.
- xPyD paradigm for disaggregated serving
- LLM-focused autoscaling, e.g. queue size, batch size

## Others

- Fairness in multi-tenant hierarchy
- Efficient model loading. Cache or GPU streaming
- Complexity along with steep learning curve
- More ...

# Run LLMs on Heterogeneous Clusters



## Heterogeneous Cluster

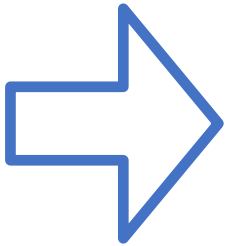
- A variety of chips
- Hight-end GPU stockout
- Cost & Performance tradeoffs

# Run LLMs on Heterogeneous Clusters

KubeCon CloudNativeCon China 2025

Heterogeneous Cluster

- A variety of chips
- Hight-end GPU stockout
- Cost & Performance tradeoffs



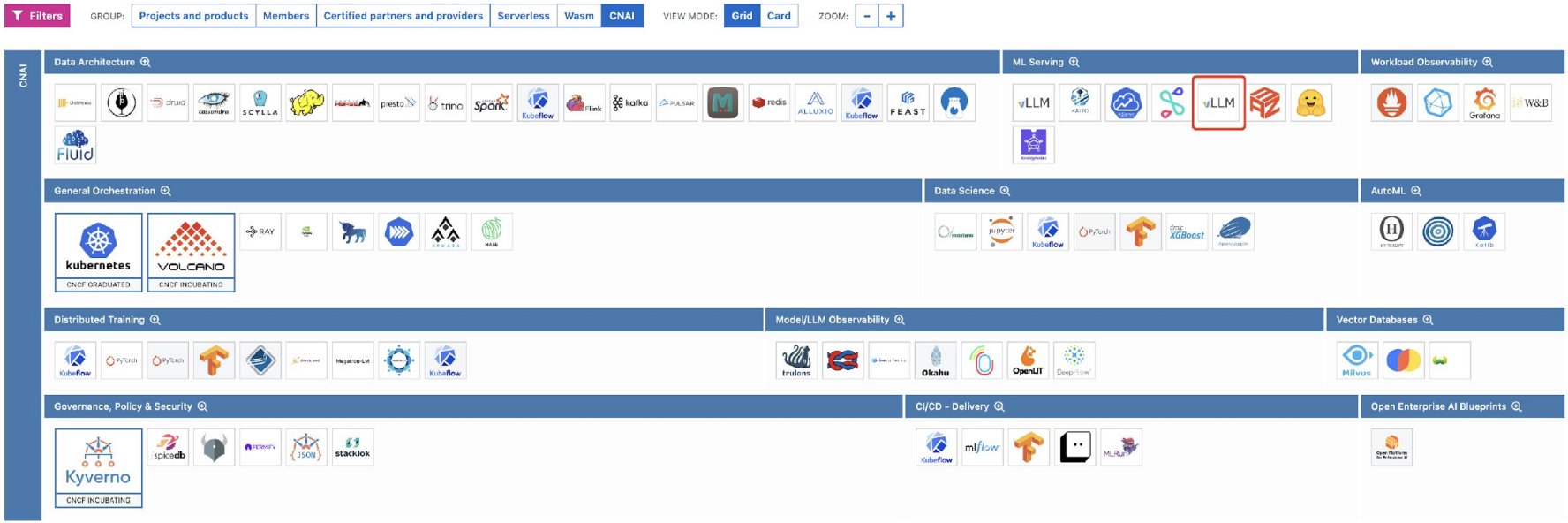
vLLM

+

llmaz

# Why vLLM: Introduction

vLLM, a fast and easy-to-use library for LLM inference and serving, is both an LF AI & Data incubation project and a PyTorch ecosystem project.




From [CNAI landscape]

# Why vLLM: Features



## Wide range of model support

- 35+ model architectures including vision language models
- Collaborating with model vendors

## Diverse hardware support

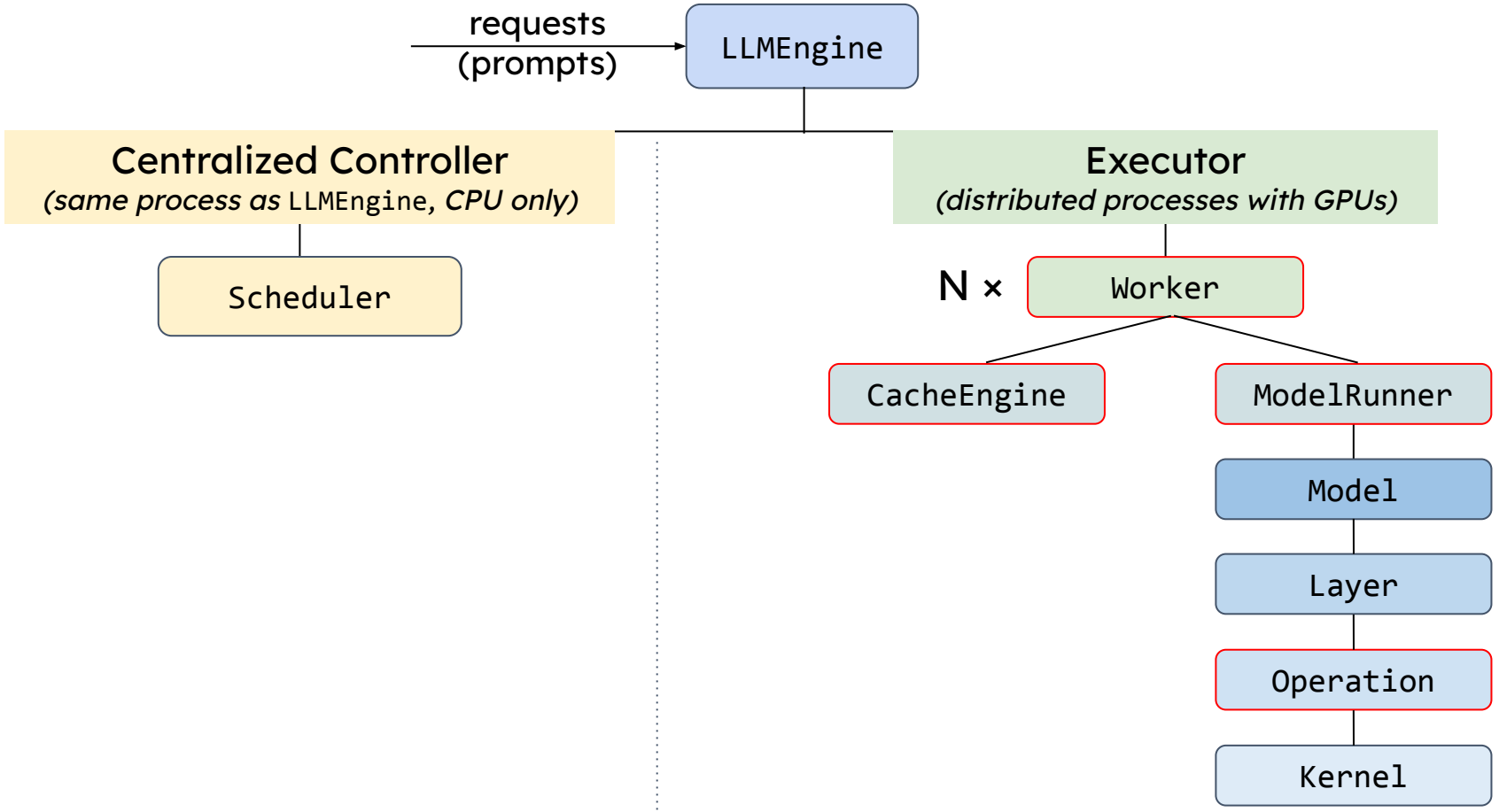
- NVIDIA, AMD, Intel GPUs
- Intel/AMD CPU
- Inferentia, TPU, Gaudi

## End-to-end inference optimizations

- CUDA graph
- Speculative decoding
- Quantization (GPTQ, AWQ, FP8)
- Automatic prefix caching
- Chunked prefills (a.k.a. Dynamic SplitFuse)
- Multi-LoRA serving
- Constrained decoding
- FlashAttention & FlashDecoding

From the fourth vLLM meetup slides

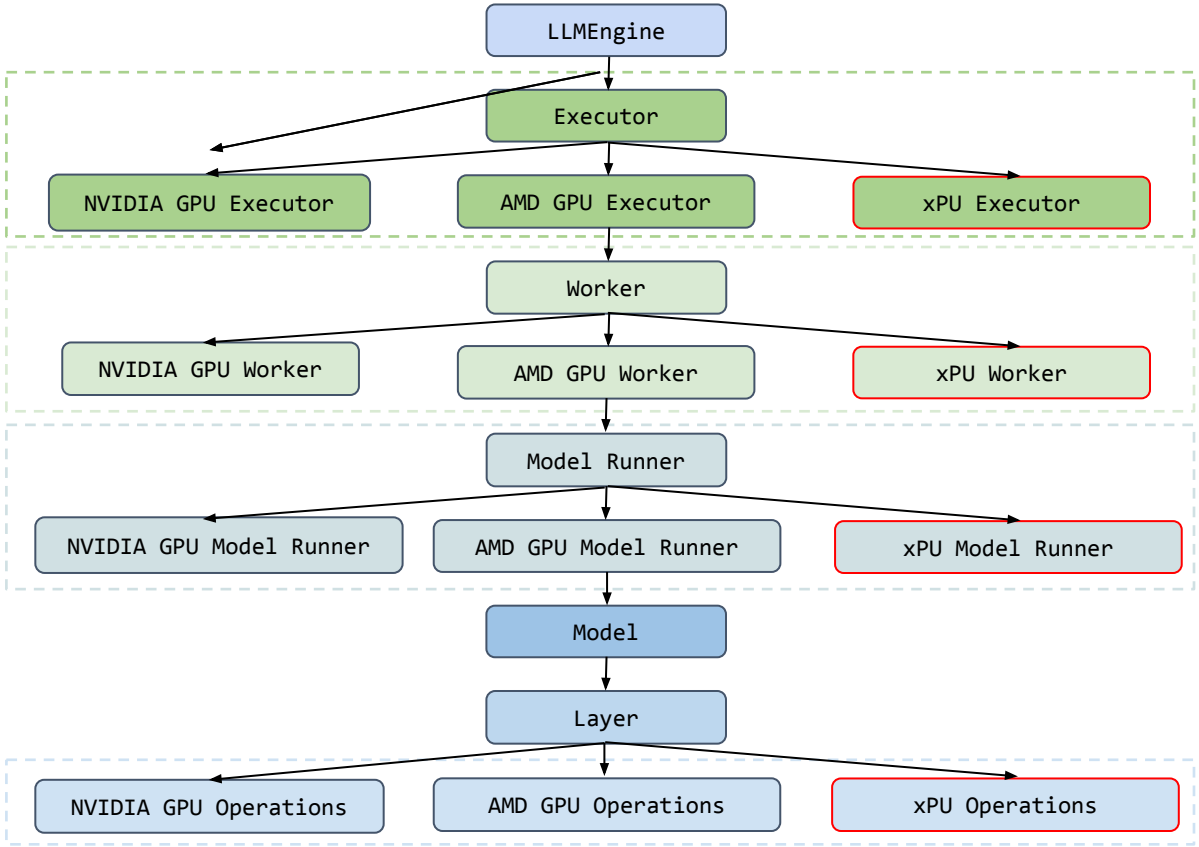
# Why vLLM: Architecture



# How vLLM Supports Heterogeneous Chips



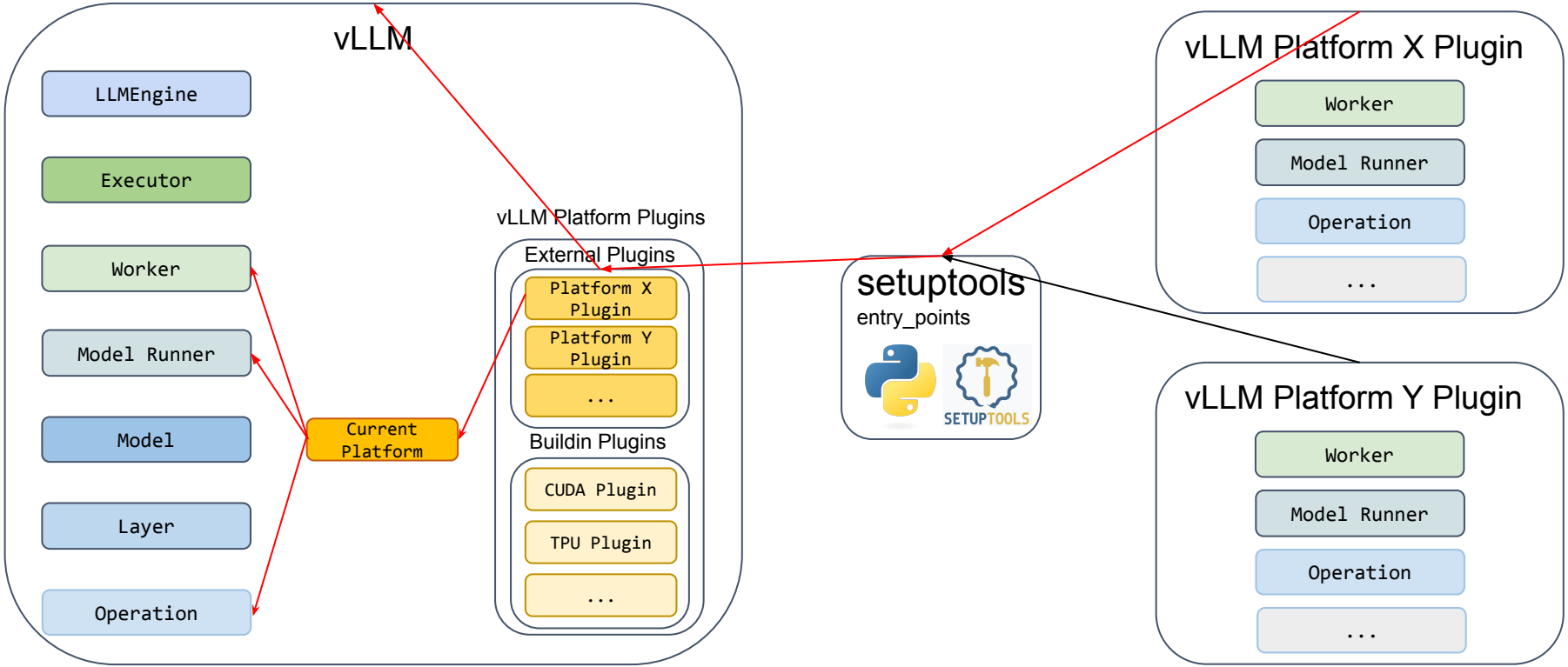
Before vLLM 0.7.0, Without Platform Plugins System



# How vLLM Supports Heterogeneous Chips



After vLLM 0.7.0, With Platform Plugins System

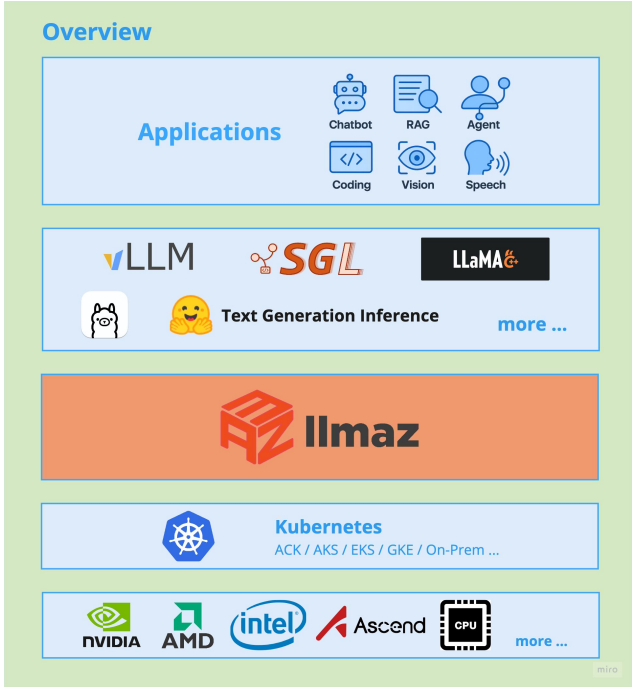



Based on Solution 2, Huawei has open-sourced the [vLLM-Ascend] project for Ascend chips, and we are collaborating with Huawei on its development.

# What is llmaz

KubeCon CloudNativeCon China 2025

Easy, advanced inference platform for large language models on Kubernetes.

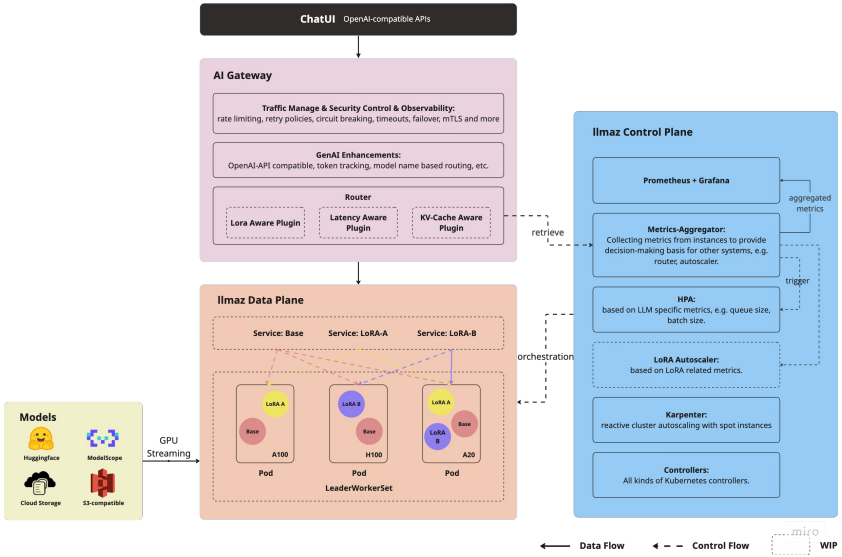



## Key Features:

- **Easy to Use:** Support to set recommended templates for quick start. Separation of concerns.
- **Heterogeneous Cluster Support**: GPU unawareness for users when serving LLMs.
- **Advanced Orchestration**: Multi-Host & Homogeneous xPyD support with LeaderWorkerSet.
- **Various Model Providers**: Handle model loading with HuggingFace, ModelScope, ObjectStores.
- **AI Gateway:** Plugin system with Latency Aware & LoRA Aware & KVCache Aware routing strategies (until end of June). We're working together with Envoy AI Gateway team.
- **Scaling Efficiency**: LLM-focused scaling with HPA and re-active spot instance autoscaling with Karpenter.
- **Build-in ChatUI**: Open WebUI integration to offer capacities like function call, RAG, web search and more.

# What is llmaz

Easy, advanced inference platform for large language models on Kubernetes.




## Key Features:

- **Easy to Use:** Support to set recommended templates for quick start. Separation of concerns.
- **Heterogeneous Cluster Support**: GPU unawareness for users when serving LLMs.
- **Advanced Orchestration**: Multi-Host & Homogeneous xPyD support with LeaderWorkerSet.
- **Various Model Providers**: Handle model loading with HuggingFace, ModelScope, ObjectStores.
- **AI Gateway:** Plugin system with Latency Aware & LoRA Aware & KVCache Aware routing strategies (until end of June). We're working together with Envoy AI Gateway team.
- **Scaling Efficiency**: LLM-focused scaling with HPA and re-active spot instance autoscaling with Karpenter.
- **Build-in ChatUI**: Open WebUI integration to offer capacities like function call, RAG, web search and more.

# Example:



**How to run Qwen2.5-7B in a heterogeneous cluster?**

# Example:



## OpenModel:

```yaml
apiVersion: llmaz.io/v1alpha1
kind: OpenModel
metadata:
  name: qwen
spec:
  familyName: qwen2.5
  source: # ModelScope or ObjectStores
    modelHub:
      name: Huggingface
      modelID: Qwen/Qwen2.5-7B
  inferenceConfig:
    flavors:
      - name: a10 # first option
        limits:
          nvidia.com/gpu: 1
        nodeSelector:
          gpu.nvidia.com/model: a10
      - name: a100-40gb # second option
        limits:
          nvidia.com/gpu: 1
        nodeSelector:
          gpu.nvidia.com/model: a100
      - name: “4090" # third option
        limits:
          nvidia.com/gpu: 1
        nodeSelector:
          gpu.nvidia.com/model: "4090"
```

## Playground:

```yaml
apiVersion: inference.llmaz.io/v1alpha1
kind: Playground
metadata:
  name: qwen
spec:
  replicas: 6
  modelClaim:
    modelName: qwen
```

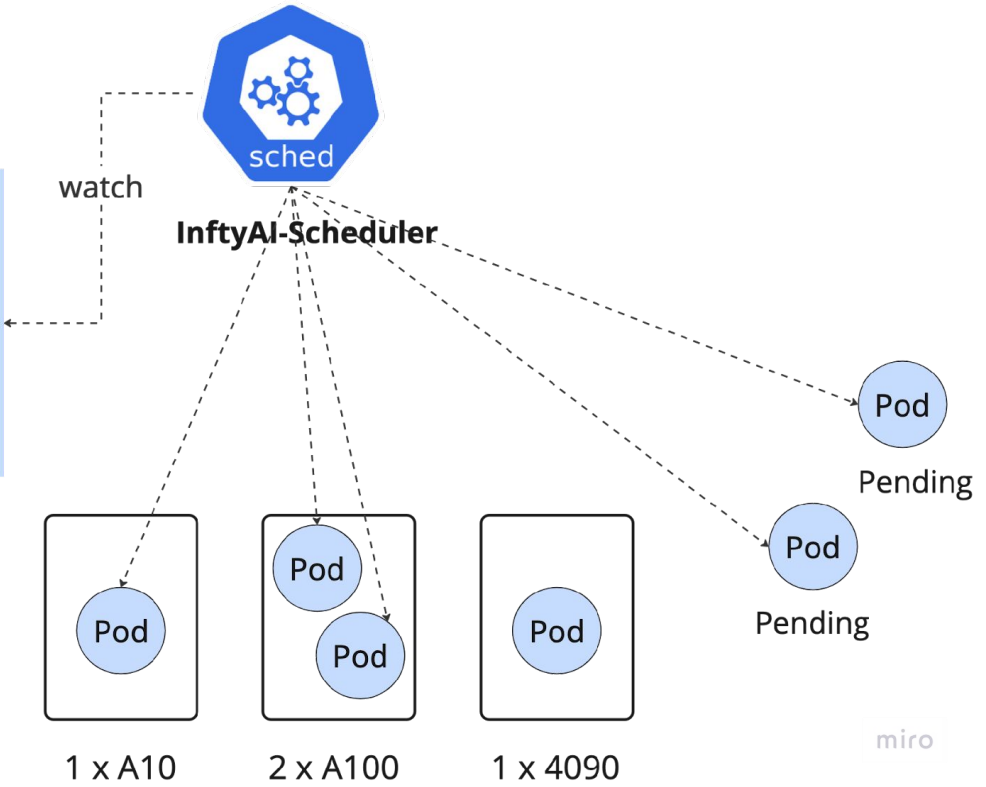
# Example:



**Kubectl apply -f**

```
apiVersion: inference.llmaz.io/v1alpha1
kind: Playground
metadata:
  name: qwen
spec:
  replicas: 6
  modelClaim:
    modelName: qwen
```

*Flavors: A10 -> A100 -> 4090*



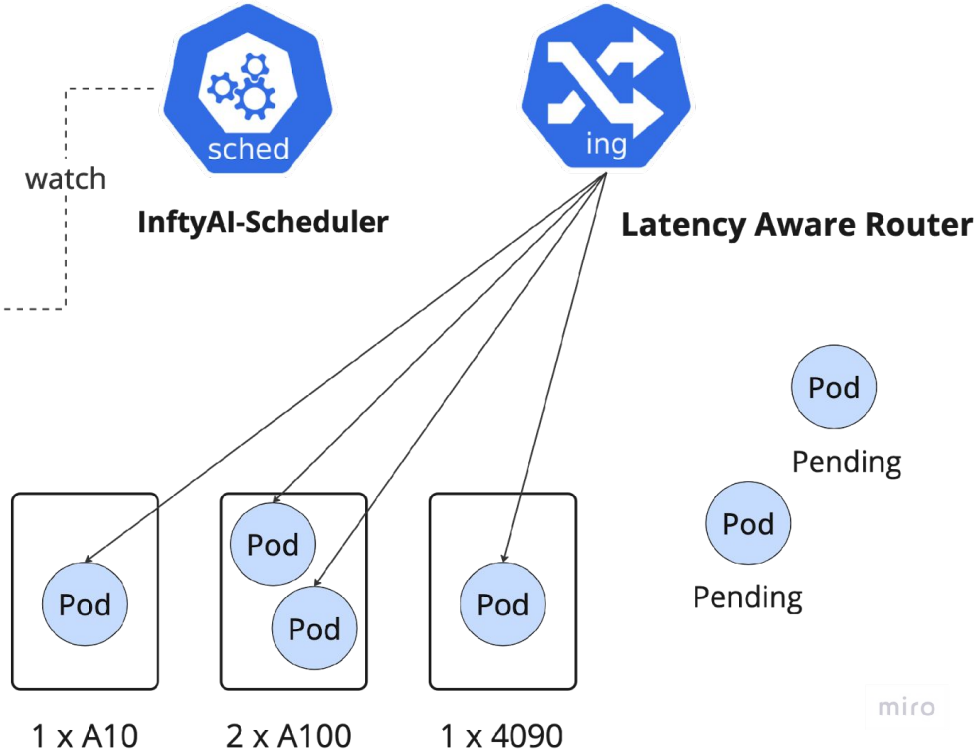
# Example:



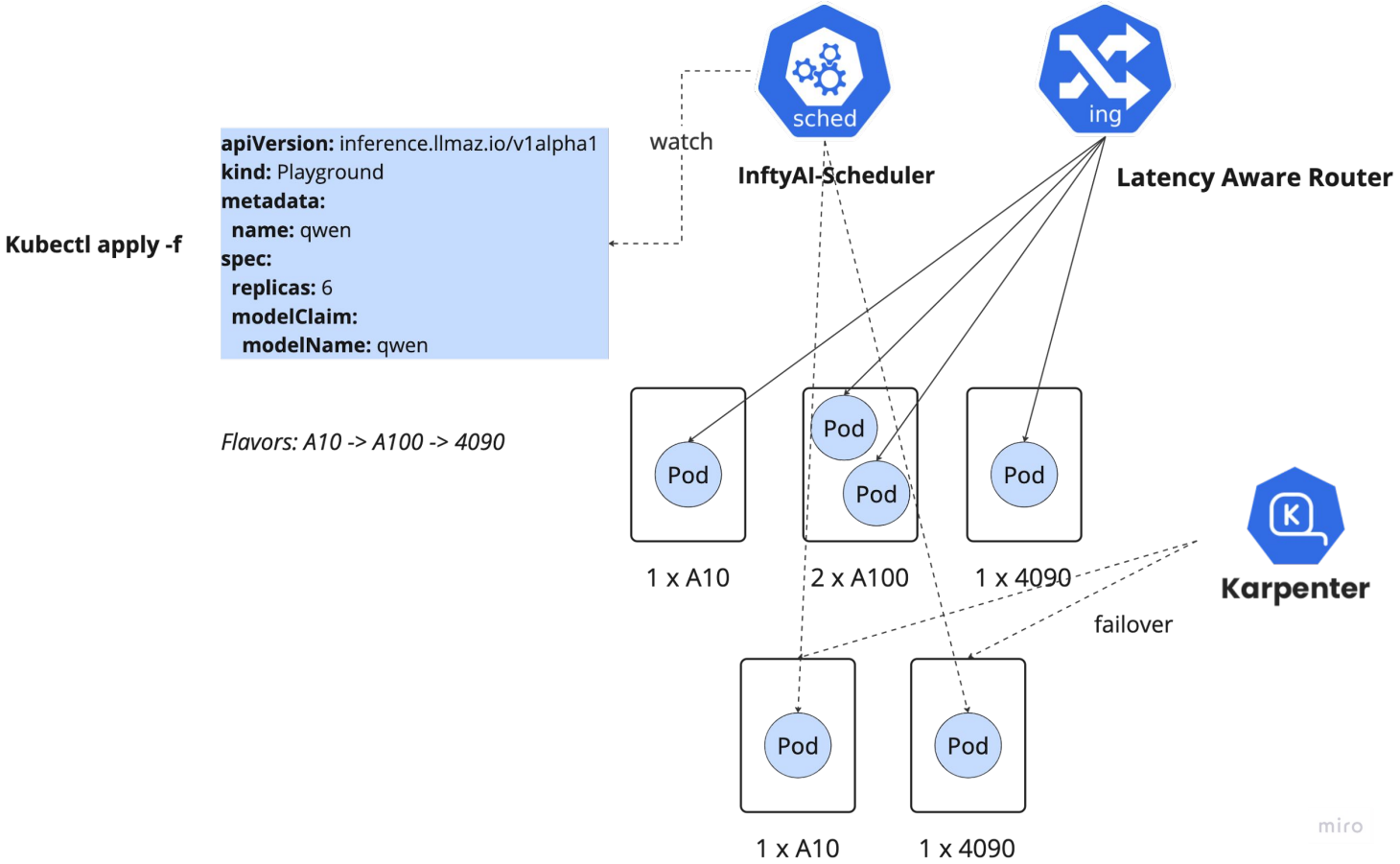
**Kubectl apply -f**

```
apiVersion: inference.llmaz.io/v1alpha1
kind: Playground
metadata:
  name: qwen
spec:
  replicas: 6
  modelClaim:
    modelName: qwen
```

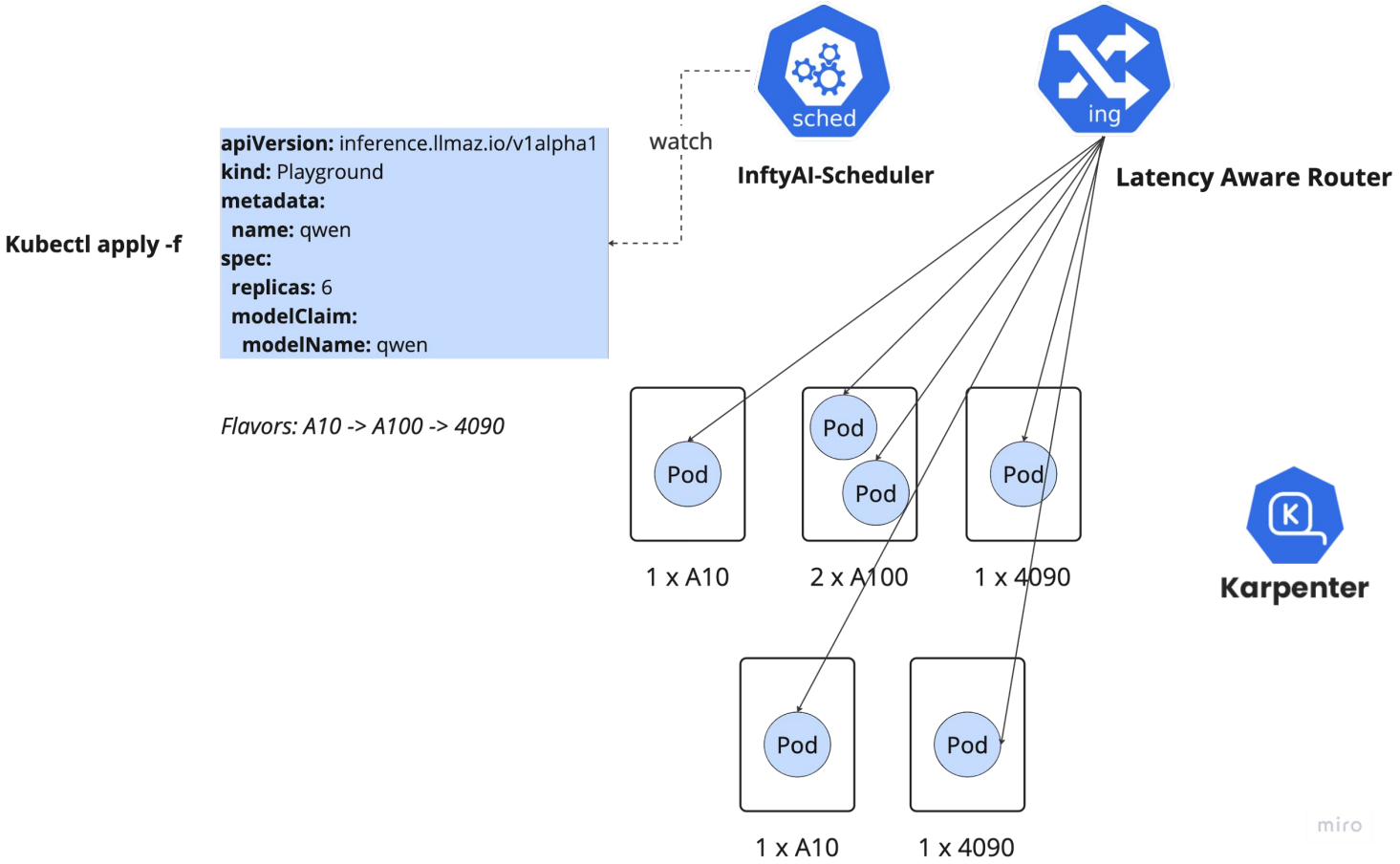
*Flavors: A10 -> A100 -> 4090*



# Example:

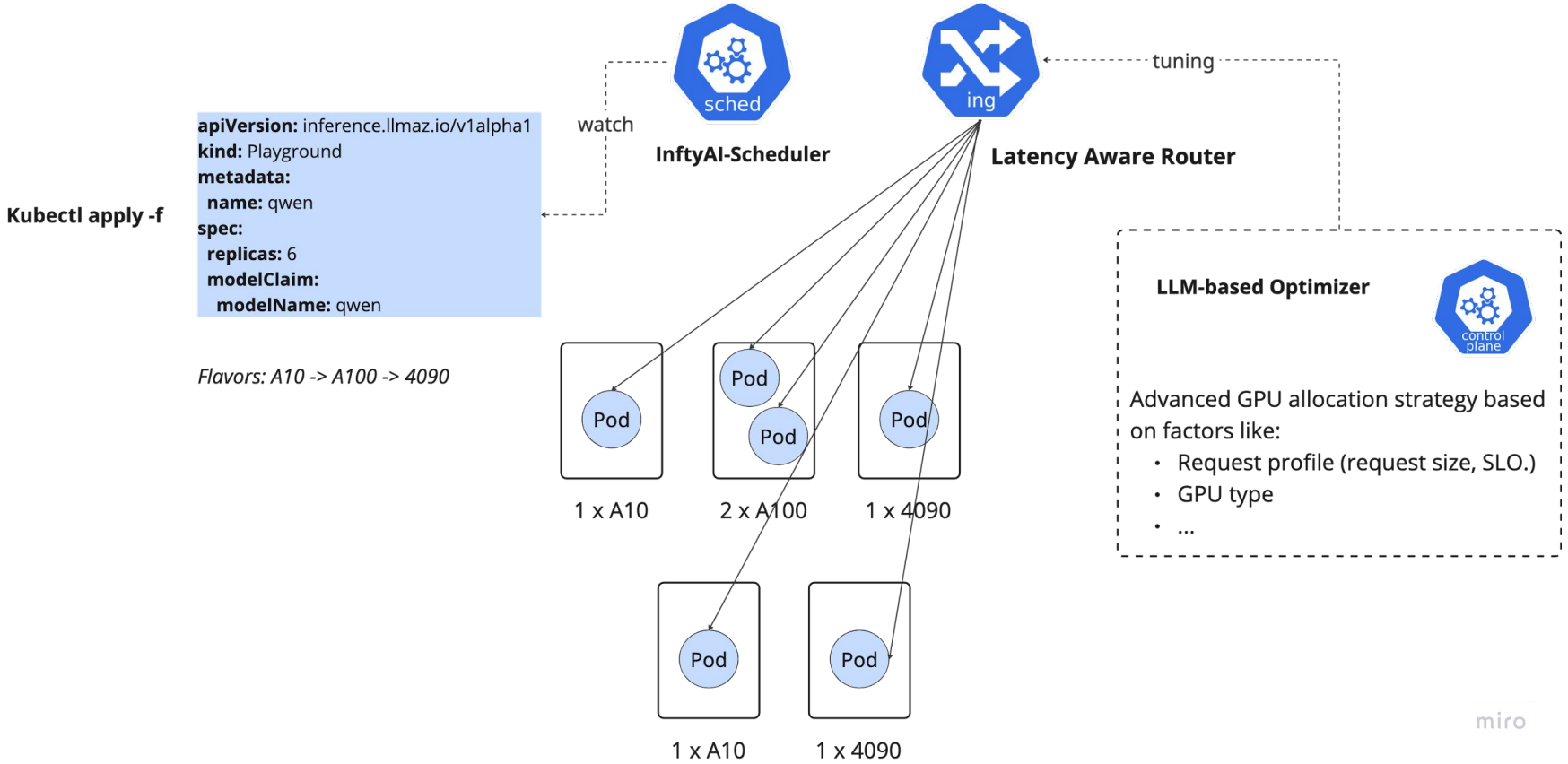

Kubectl apply -f
apiVersion: inference.llmaz.io/v1alpha1
kind: Playground
metadata:
name: qwen
spec:
replicas: 6
modelClaim:
modelName: qwen
Flavors: A10 -> A100 -> 4090
watch
sched
InftyAI-Scheduler
ing
Latency Aware Router
Pod
Pod
Pod
Pod
1 x A10
2 x A100
1 x 4090
Karpenter
failover
Pod
Pod
1 x A10
1 x 4090
miro

# Example:


Kubectl apply -f
apiVersion: inference.llmaz.io/v1alpha1
kind: Playground
metadata:
name: qwen
spec:
replicas: 6
modelClaim:
modelName: qwen
watch
sched
InftyAI-Scheduler
ing
Latency Aware Router
Flavors: A10 -> A100 -> 4090
Pod
Pod
Pod
Pod
Pod
Pod
Pod
1 x A10
2 x A100
1 x 4090
1 x A10
1 x 4090
Karpenter
miro

# Example:




Kubectl apply -f
apiVersion: inference.llmaz.io/v1alpha1
kind: Playground
metadata:
name: qwen
spec:
replicas: 6
modelClaim:
modelName: qwen
Flavors: A10 -> A100 -> 4090
watch
sched
InftyAI-Scheduler
ing
Latency Aware Router
tuning
LLM-based Optimizer
control plane
Advanced GPU allocation strategy based on factors like:
Request profile (request size, SLO.)
GPU type
...
Pod
1 x A10
Pod
Pod
2 x A100
Pod
1 x 4090
Pod
1 x A10
Pod
1 x 4090

# Example:



## OpenModel:

```yaml
apiVersion: llmaz.io/v1alpha1
kind: OpenModel
metadata:
  name: qwen
spec:
  familyName: qwen2.5
  source: # ModelScope or ObjectStores
    modelHub:
      name: Huggingface
      modelID: Qwen/Qwen2.5-7B
  inferenceConfig:
    flavors:
      - name: a10 # first option
        limits:
          nvidia.com/gpu: 1
        nodeSelector:
          gpu.nvidia.com/model: a10
      - name: a100-40gb # second option
        limits:
          nvidia.com/gpu: 1
        nodeSelector:
          gpu.nvidia.com/model: a100
      - name: “4090" # third option
        limits:
          nvidia.com/gpu: 1
        nodeSelector:
          gpu.nvidia.com/model: "4090"
```

## Playground:

```yaml
apiVersion: inference.llmaz.io/v1alpha1
kind: Playground
metadata:
  name: qwen
spec:
  replicas: 4
  modelClaim:
    modelName: qwen
  backendRuntimeConfig:
    backendName: sglang
  elasticConfig:
    minReplicas: 1
    maxReplicas: 100
```

Easy Setup

# Example:



## OpenModel:

```yaml
apiVersion: llmaz.io/v1alpha1
kind: OpenModel
metadata:
  name: qwen
spec:
  familyName: qwen2.5
  source: # ModelScope or ObjectStores
    modelHub:
      name: Huggingface
      modelID: Qwen/Qwen2.5-7B
  inferenceConfig:
    flavors:
      - name: a10 # first option
        limits:
          nvidia.com/gpu: 1
        nodeSelector:
          gpu.nvidia.com/model: a10
      - name: a100-40gb # second option
        limits:
          nvidia.com/gpu: 1
        nodeSelector:
          gpu.nvidia.com/model: a100
      - name: “4090" # third option
        limits:
          nvidia.com/gpu: 1
        nodeSelector:
          gpu.nvidia.com/model: "4090"
```

## Playground:

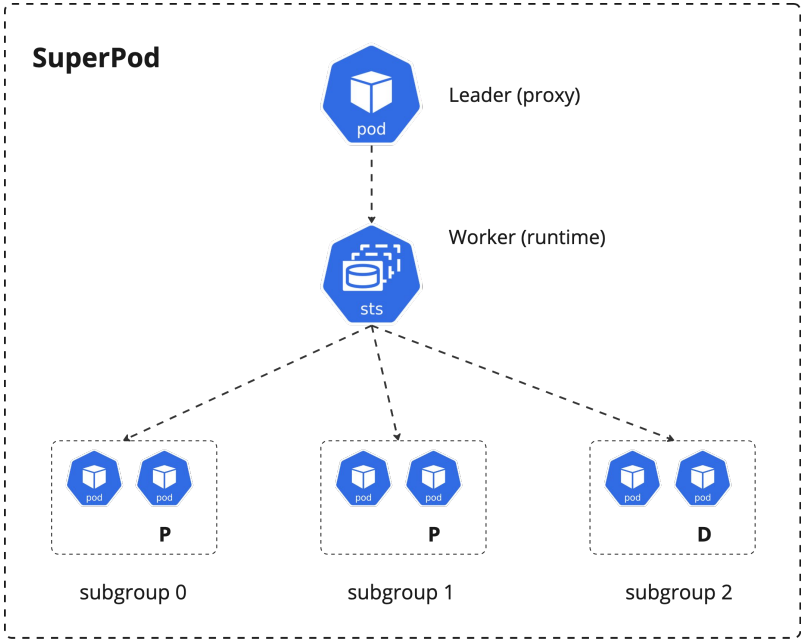
```yaml
apiVersion: inference.llmaz.io/v1alpha1
kind: Playground
metadata:
  name: qwen
spec:
  replicas: 4
  modelClaim:
    modelName: qwen
  backendRuntimeConfig:
    backendName: sglang
  elasticConfig:
    minReplicas: 1
    maxReplicas: 100
```

Easy Setup

## **Inference Service** for more complicated cases:

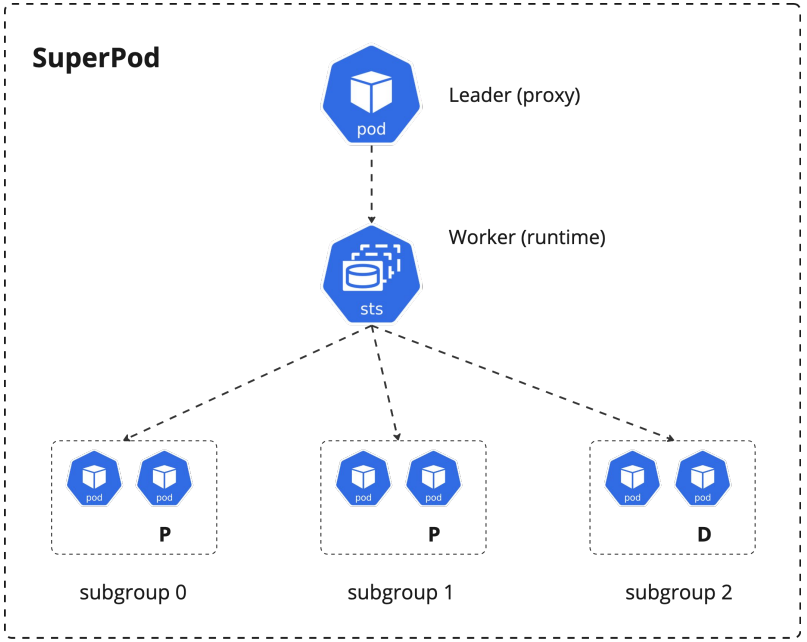
```yaml
apiVersion: inference.llmaz.io/v1alpha1
kind: Service
metadata:
  name: qwen
spec:
  replicas: 4
  modelClaims:
    models:
    - name: qwen
  workloadTemplate: ...
```

# What's Next (xPyD)




One Replica of LWS (P:D=2:1)

# What's Next (xPyD)




One Replica of LWS (P:D=2:1)

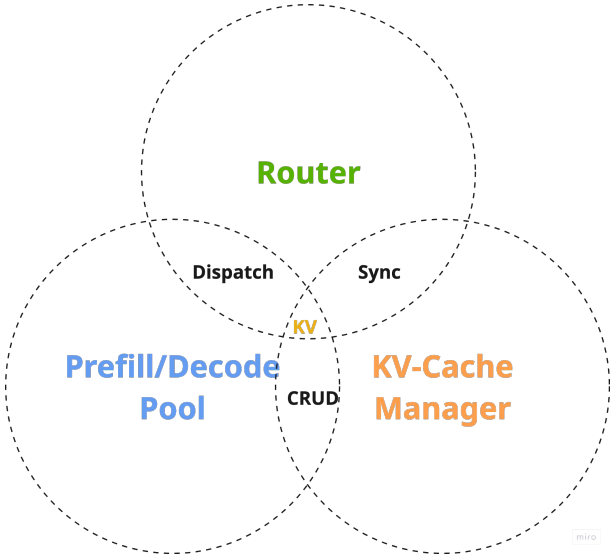
**Homogeneous Disaggregated Serving with LWS:**

- Proxy can't scale independently
- Static PD ratio, scaling PD as a whole
- With identical Pod template, no separate resource pools for P & D

# What's Next (xPyD)

**Heterogeneous Disaggregated Serving:**

- Multiple role templates, e.g. Proxy, Prefill, Decode
- Rolling update strategy based on P-D ratio
- Independent scaling capacity

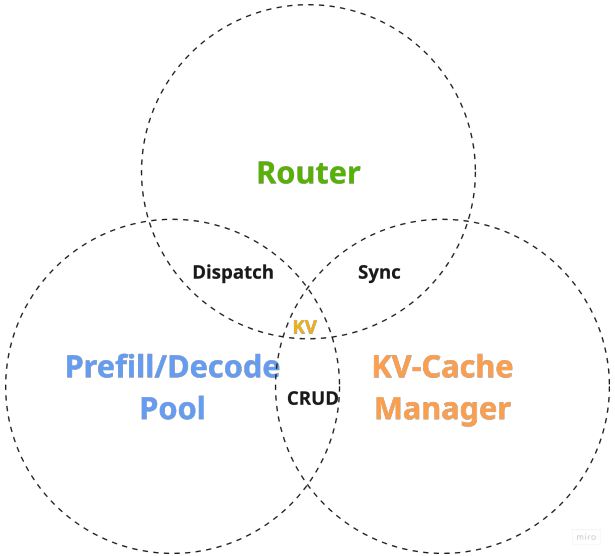


# What's Next (xPyD)



**Heterogeneous Disaggregated Serving:**

- Multiple role templates, e.g. Proxy, Prefill, Decode
- Rolling update strategy based on P-D ratio
- Independent scaling capacity




**We're looking to build a new orchestration on top of LWS.**

# Join Us




💙 Thanks to all the contributors!

😚 We'll donate to CNCF soon.

🏃 Join us together!

*https://github.com/InftyAI/llmaz*

# Join Us



InftyAI

Slack

WeChat

*https://github.com/InftyAI/llmaz*


dapr
HELM
TiKV
KEDA
KubeCon
CloudNativeCon
China 2025
Thanks!