Keeping it Small: Agentic Workflows with SLMs on K8S

Frank Fan - Senior Container Solution Architect , AWS

Part 01

Address challenges of Agentic workload

What is a Gen AI Agent

Intelligent, autonomous systems

Plan, reason, and act

Access to enterprise data

Ability to use tools

. . . it leads to challenges

Complexity

Coding gets complicated

- Complex prompts to limit hallucinations
- Fragile, hard to maintain

Accuracy

Agent gets confused

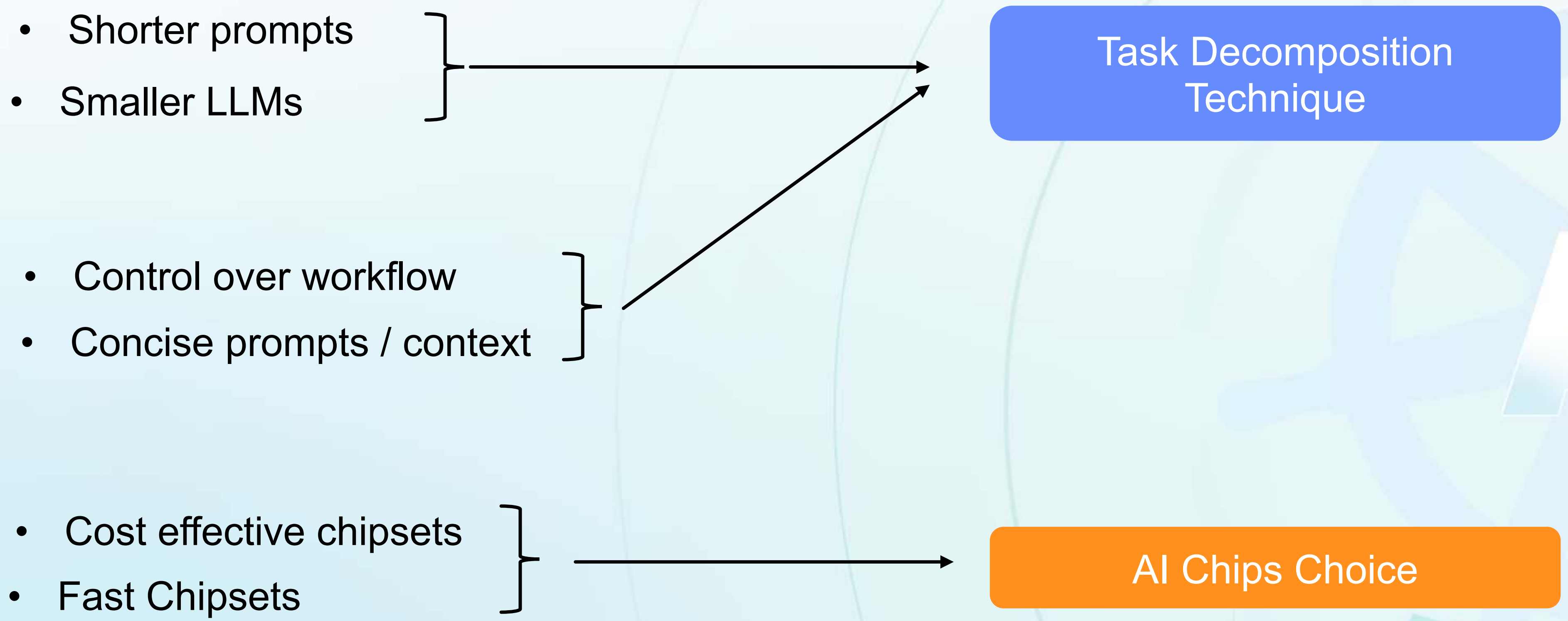
- Calling wrong tools
- Passing wrong arguments
- Inconsistent responses

Cost

Agent gets slower and more expensive

- Frontier models needed
- Prompt sizes grow
- Agents retry steps

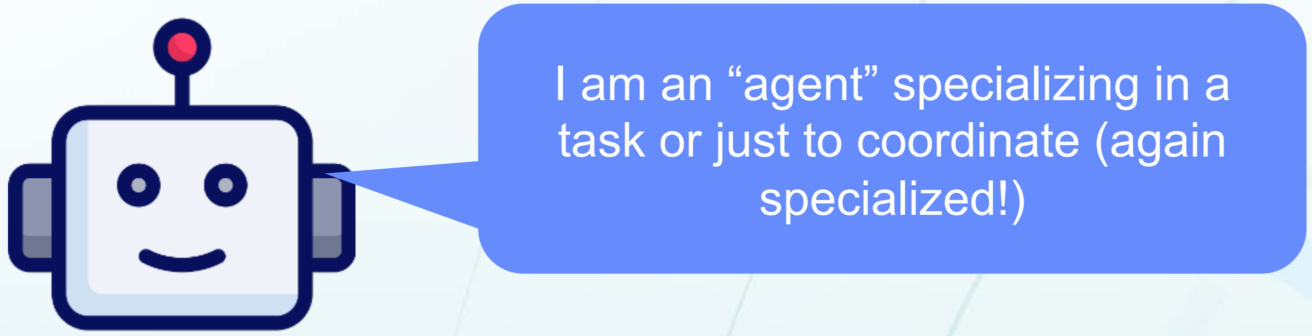
Hints to improve cost, performance and accuracy

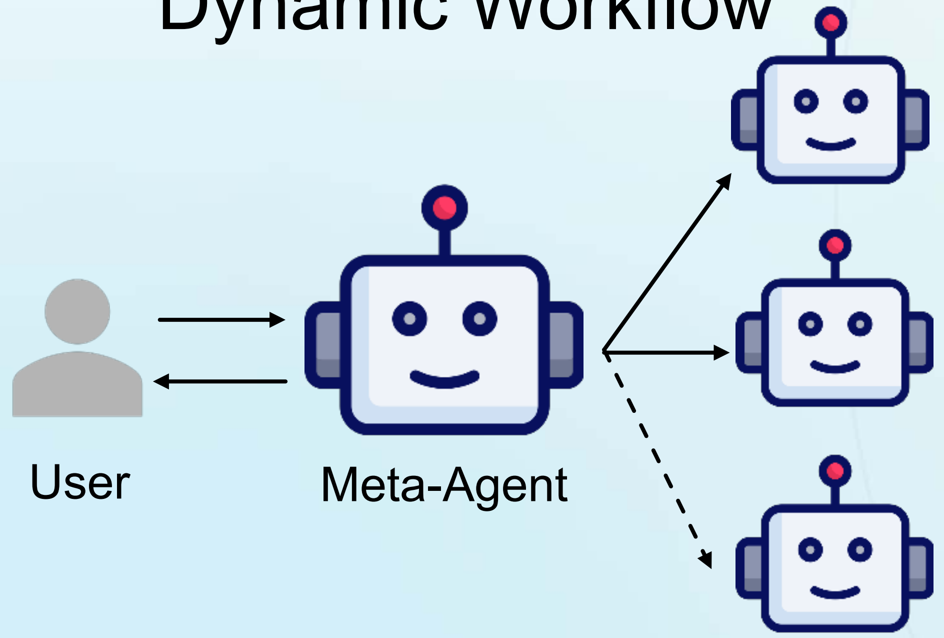


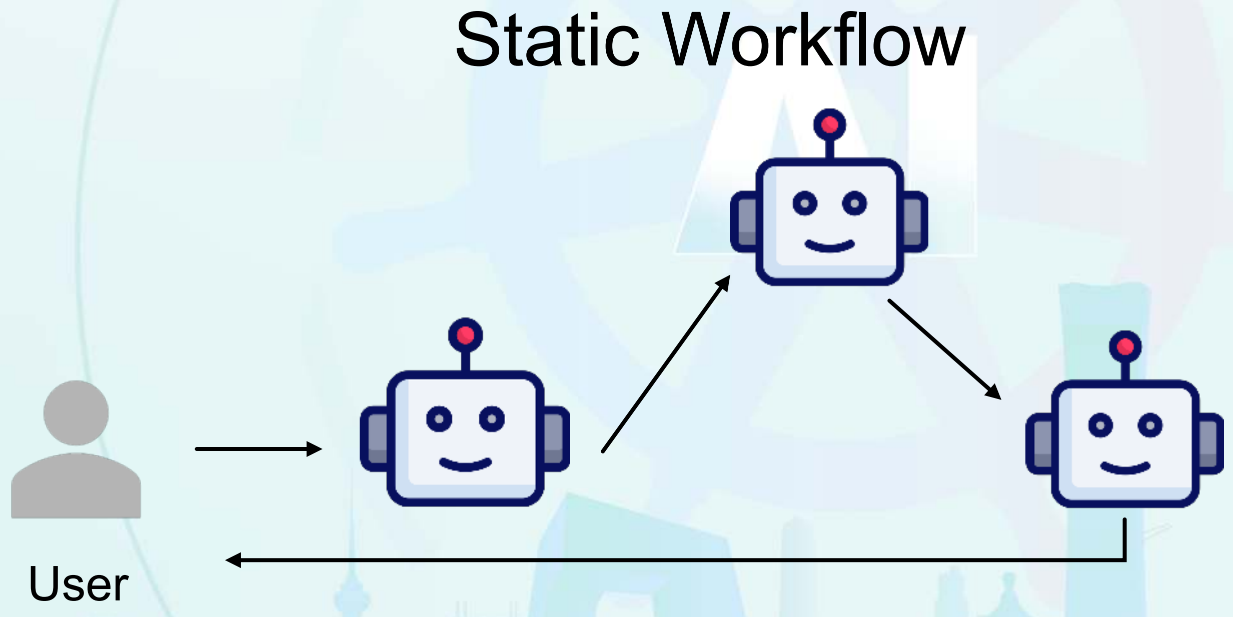
Part 02

Multi-agent workflows

Agent & workflows


I am an "agent" specializing in a task or just to coordinate (again specialized!)





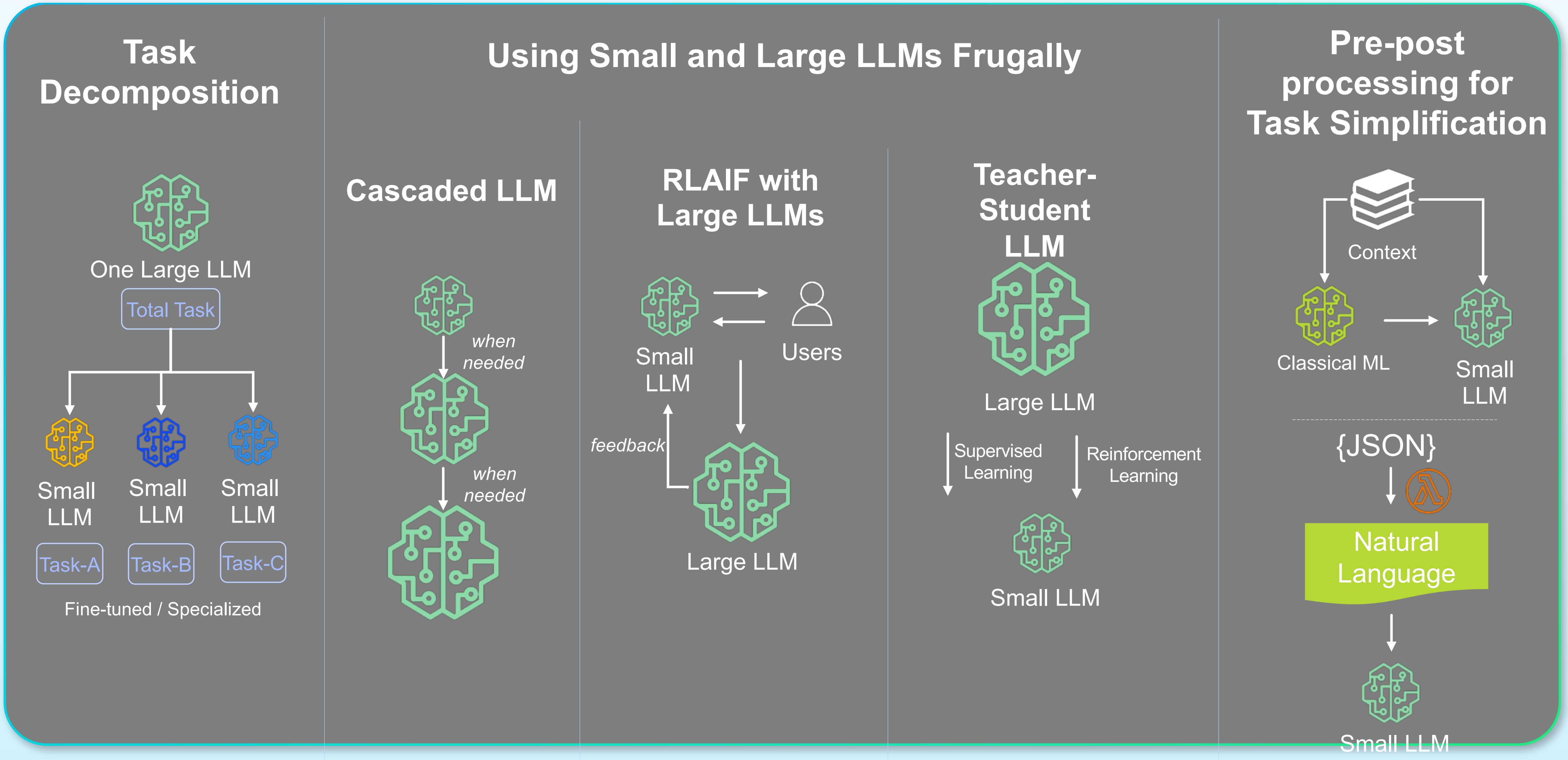
Why Small Language Model?

Resource

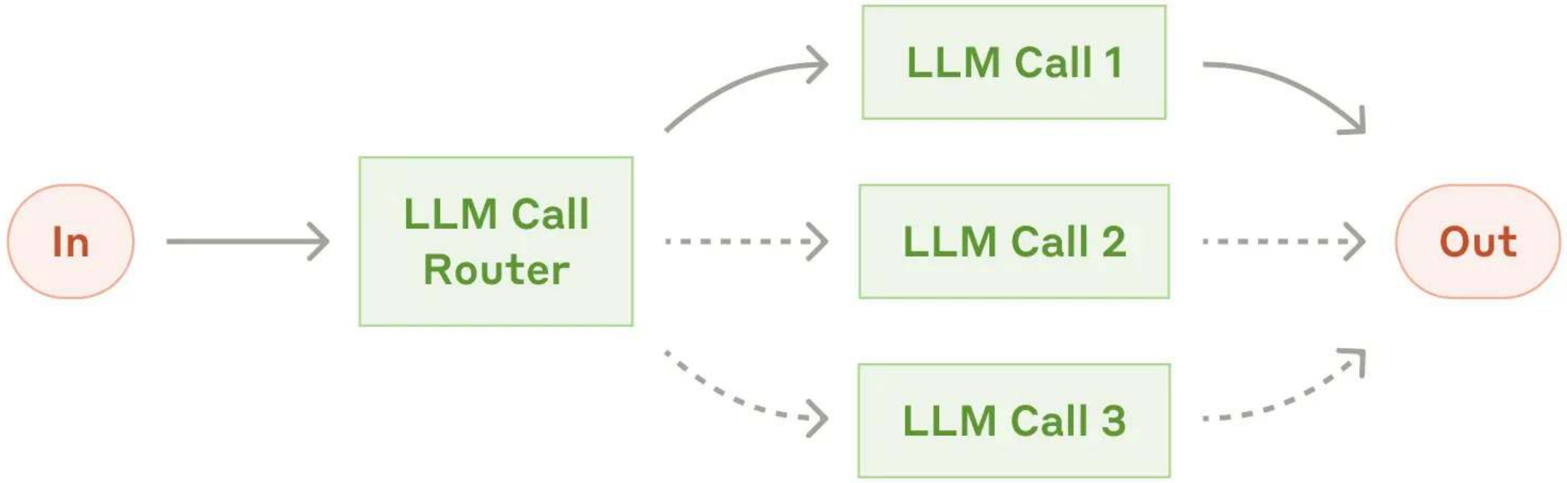
Speed

Customize

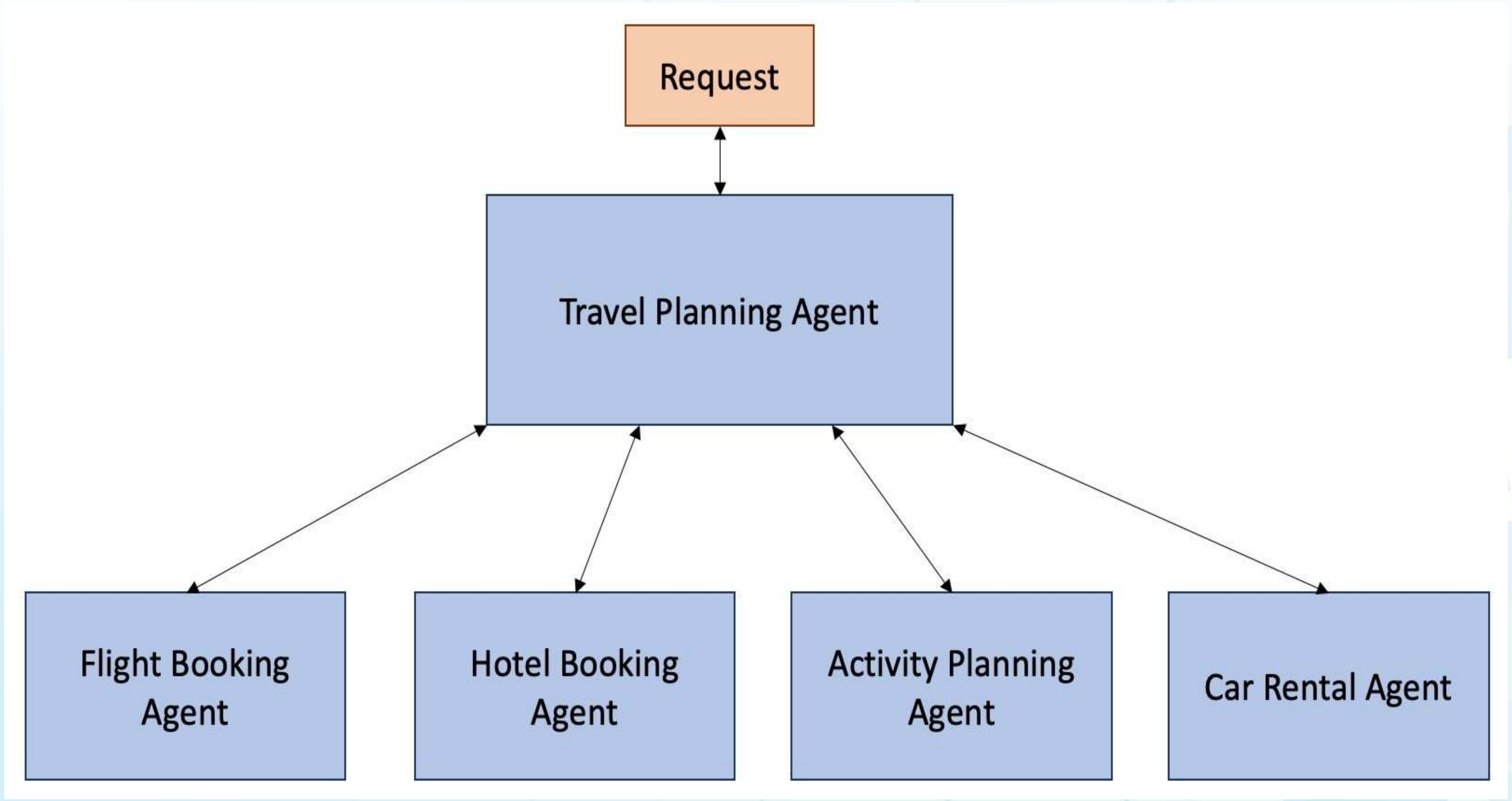
Frugal Architecture Design Patterns



Workflow : Routing



Example – Routing



Part 03

Implementation on K8S

Why self-host Language Model on K8S

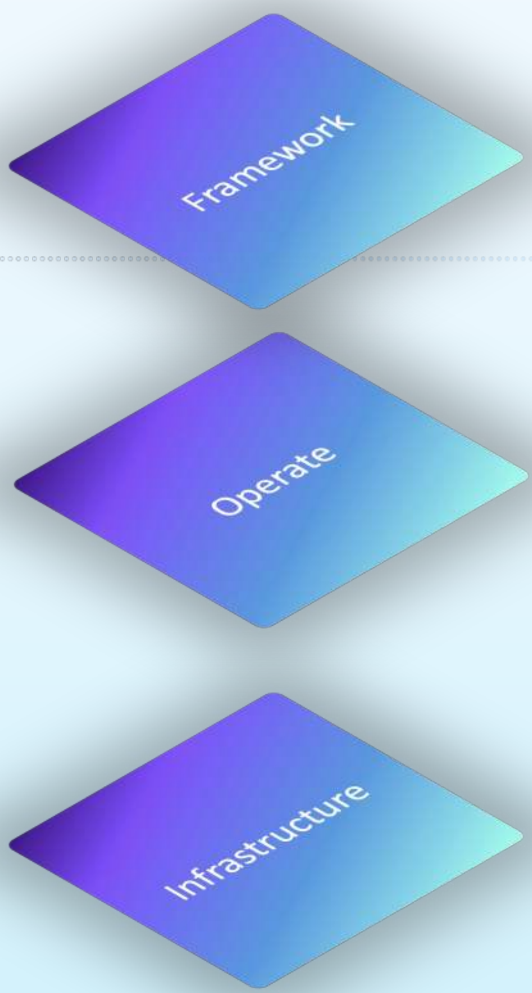
Data privacy and security

Connecting to data sources

Customizing

Accessing multiple models and newer versions

Running Agentic workload on K8S



LangGraph

Amazon Bedrock Agents

AI Gateway

Observability

Runtime

CUDA Runtime

Ray

vLLM

Neuron Runtime

Compute

NVIDIA GPUs

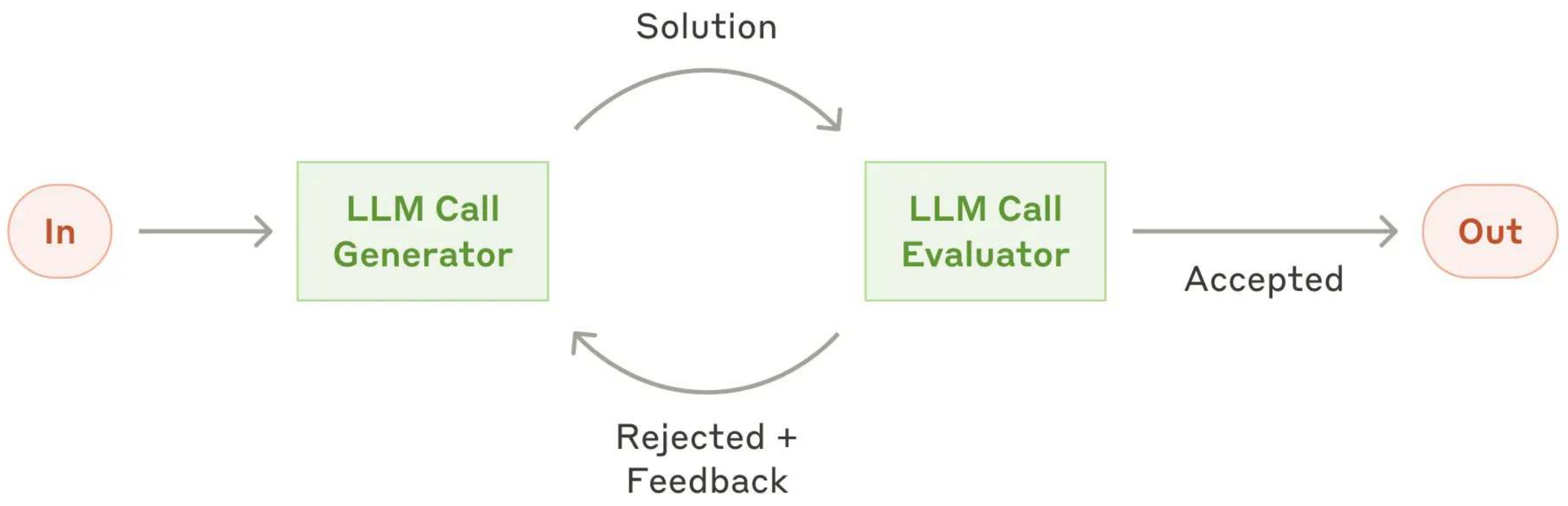
Inferentia

Trainium

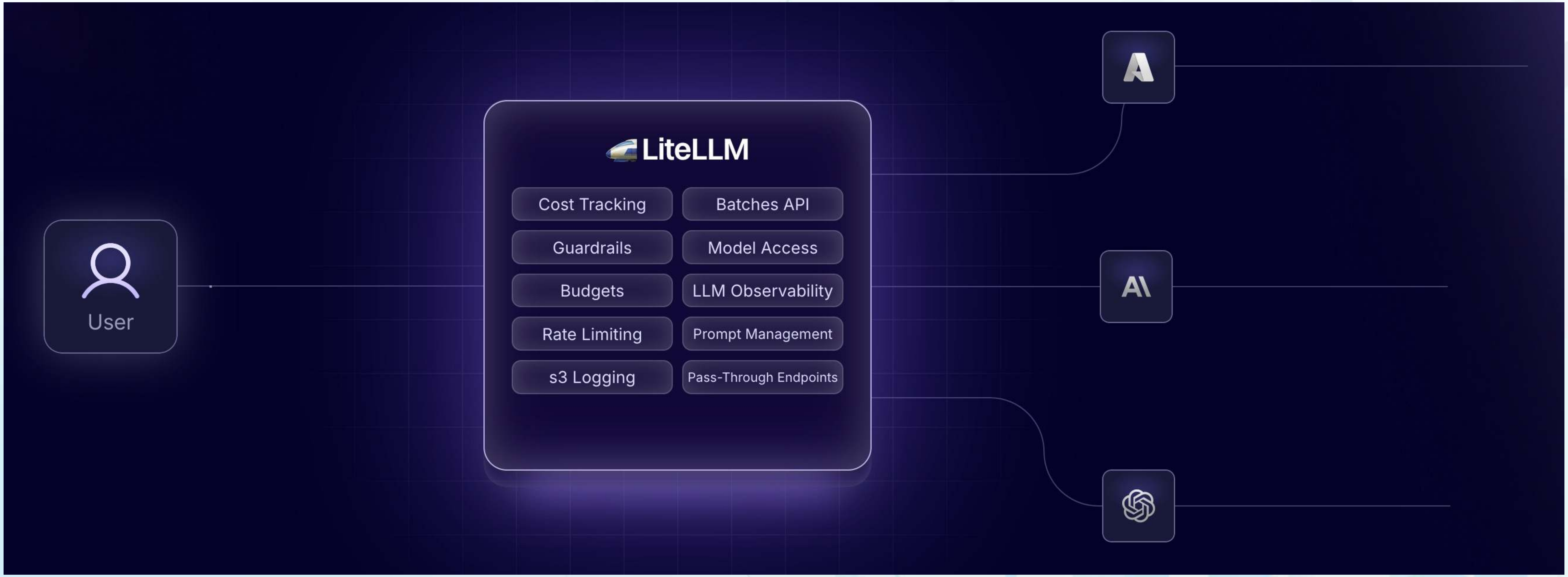
Graviton

Karpenter

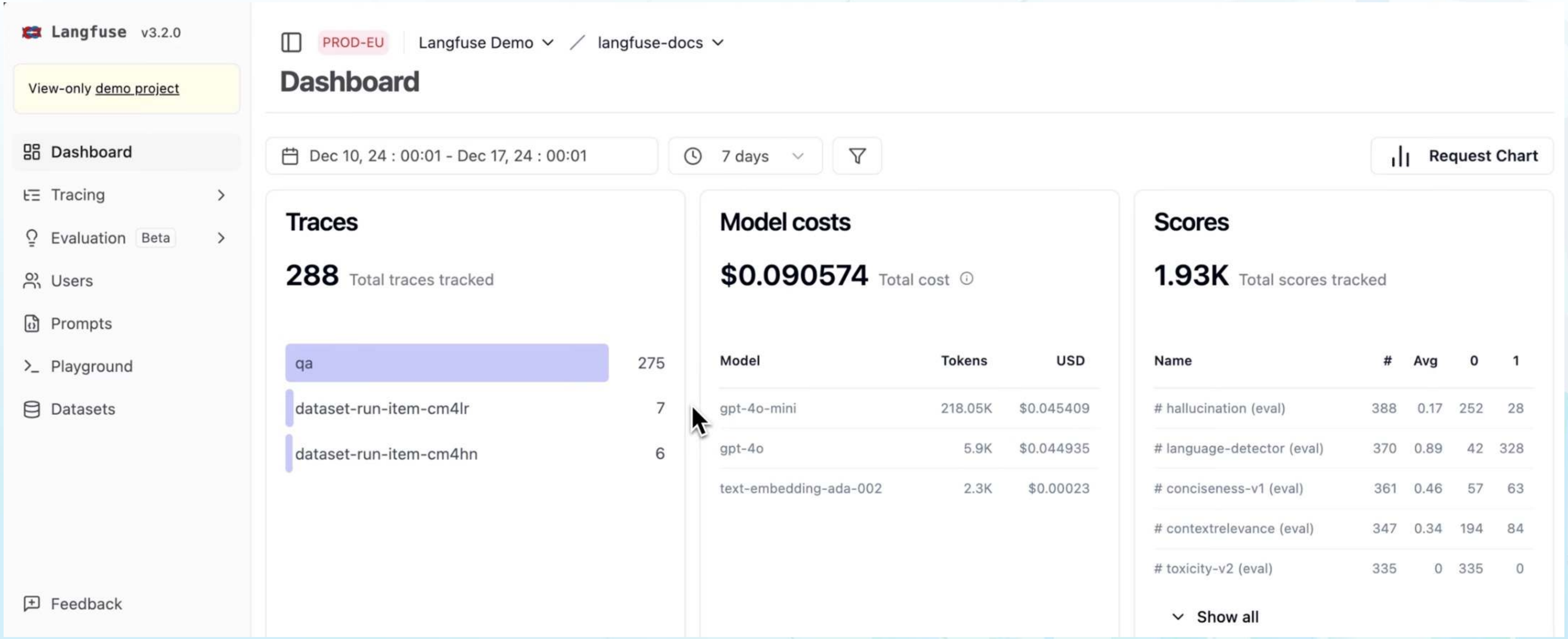
Workflow: Evaluator-optimizer



LiteLLM


User
LiteLLM
Cost Tracking
Batches API
Guardrails
Model Access
Budgets
LLM Observability
Rate Limiting
Prompt Management
s3 Logging
Pass-Through Endpoints

Observability - Langfuse



Langfuse v3.2.0

View-only demo project

Dashboard
Tracing
Evaluation Beta
Users
Prompts
Playground
Datasets

Feedback

PROD-EU Langfuse Demo / langfuse-docs

Dashboard

Dec 10, 24 : 00:01 - Dec 17, 24 : 00:01 | 7 days | Request Chart

Traces

288 Total traces tracked

qa	275
dataset-run-item-cm4lr	7
dataset-run-item-cm4hn	6

Model costs

$0.090574 Total cost

Model	Tokens	USD
gpt-4o-mini	218.05K	$0.045409
gpt-4o	5.9K	$0.044935
text-embedding-ada-002	2.3K	$0.00023

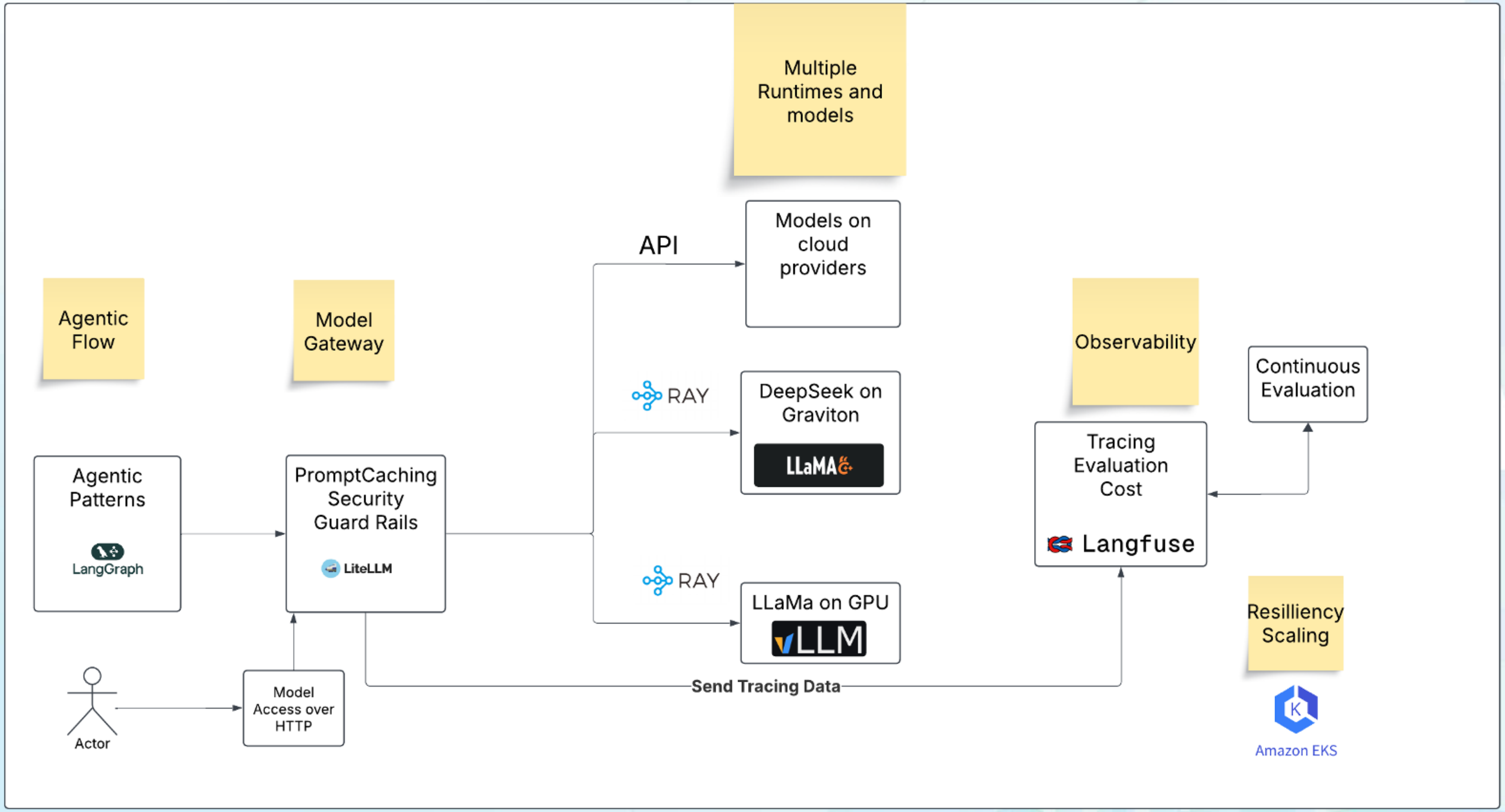
Scores

1.93K Total scores tracked

Name	#	Avg	0	1
# hallucination (eval)	388	0.17	252	28
# language-detector (eval)	370	0.89	42	328
# conciseness-v1 (eval)	361	0.46	57	63
# contextrelevance (eval)	347	0.34	194	84
# toxicity-v2 (eval)	335	0	335	0

Show all

Implementation on K8S



Part 04

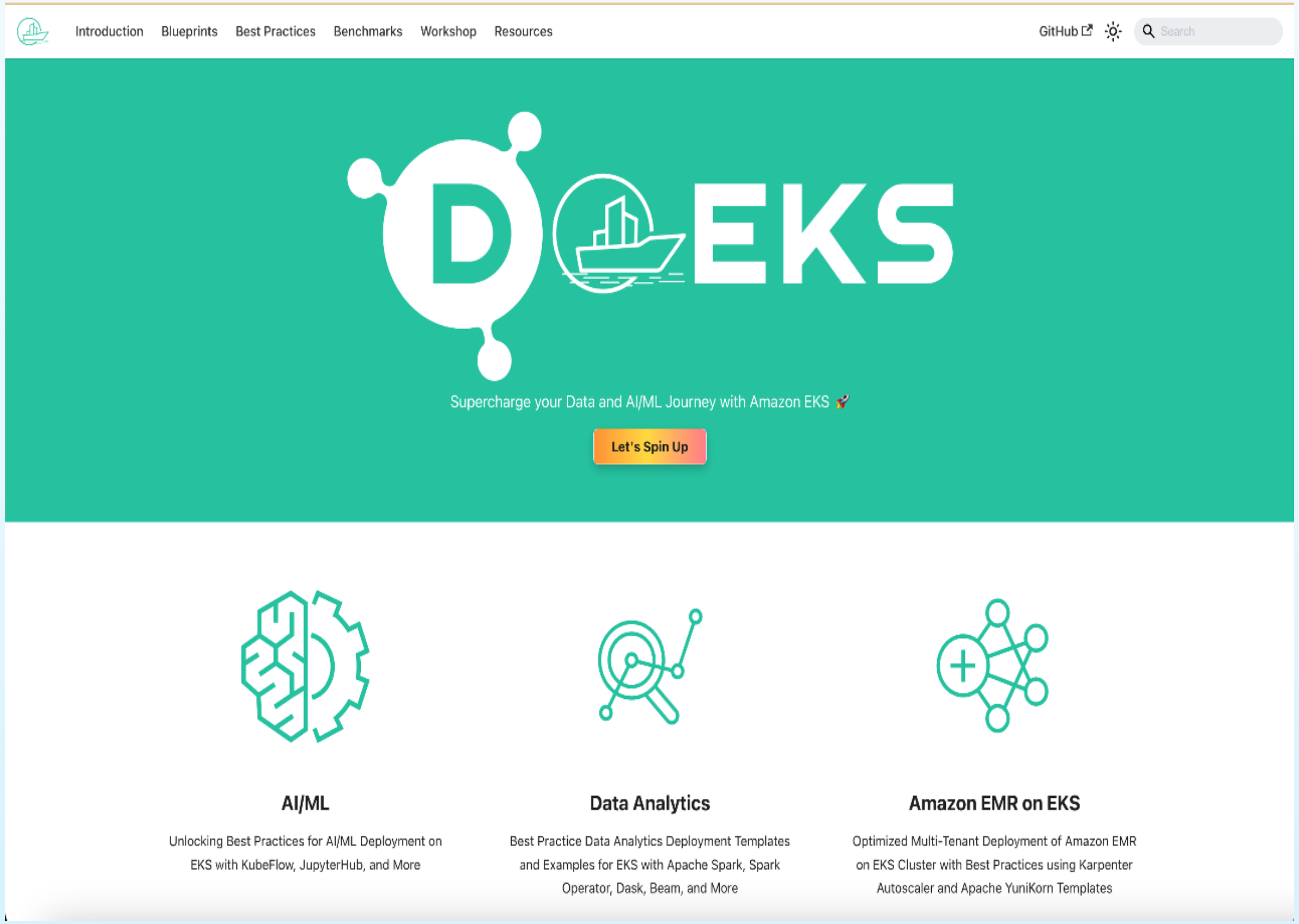
Key takeaways

Key Takeaways

- Choice of Language Models
- Deploy models on AWS Trainium , Inferentia and Graviton
- Multi-agent workflow patterns
- Improve user experience – Centralized Security and observability

Data on EKS project

- Gen AI patterns
- IaC templates
- Best practices
- Engage with us on GitHub



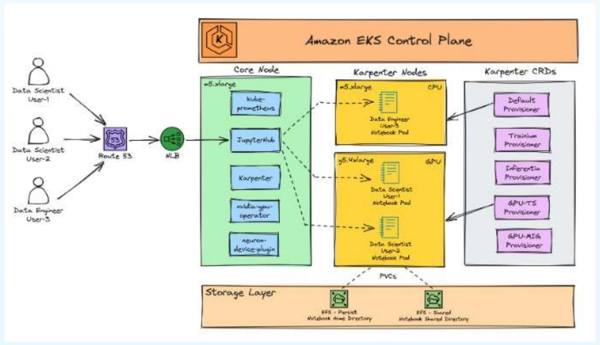
Supercharge your Data and AI/ML Journey with Amazon EKS 🚀

Unlocking Best Practices for AI/ML Deployment on EKS with KubeFlow, JupyterHub, and More

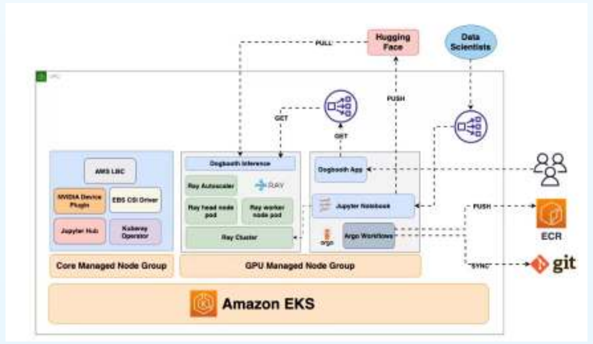
Best Practice Data Analytics Deployment Templates and Examples for EKS with Apache Spark, Spark Operator, Dask, Beam, and More

Optimized Multi-Tenant Deployment of Amazon EMR on EKS Cluster with Best Practices using Karpenter Autoscaler and Apache YuniKorn Templates

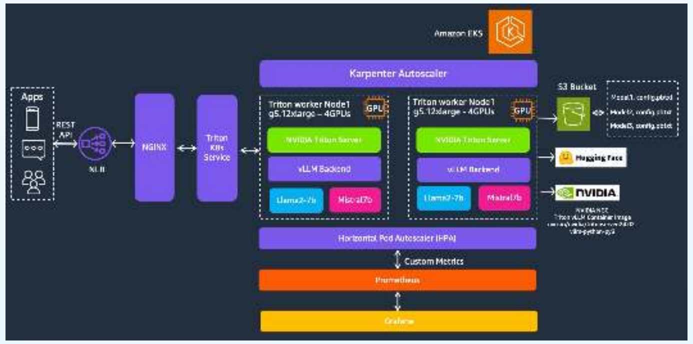
Reference patterns for gen AI on EKS



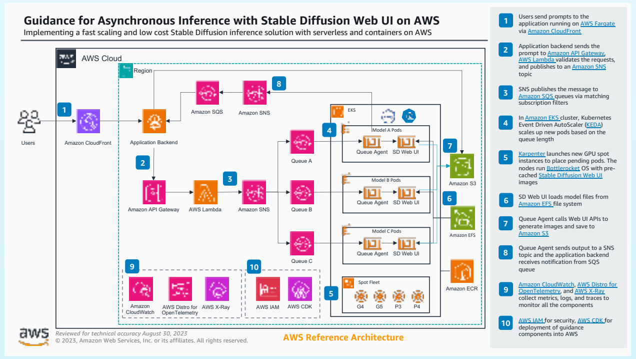
Multi-tenant JupyterHub platform



End-end generative AI orchestration platform



High performance inference platform on EKS (NVIDIA Triton with vLLM)



Guidance for Asynchronous Inference with Stable Diffusion Web UI on AWS
Implementing a fast scaling and low cost Stable Diffusion inference solution with serverless and containers on AWS
AWS Cloud
Region
Users
Amazon CloudFront
Application Backend
Amazon SQS
Amazon SNS
Amazon API Gateway
AWS Lambda
Queue A
Queue B
Queue C
EKS
Model A Pods
Model B Pods
Model C Pods
Queue Agent
SD Web UI
Spot Fleet
G4
G5
P3
P4
Amazon S3
Amazon EFS
Amazon ECR
Amazon CloudWatch
AWS Distro for OpenTelemetry
AWS X-Ray
AWS IAM
AWS CDK
Reviewed for technical accuracy August 30, 2023
© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.
AWS Reference Architecture

1 Users send prompts to the application running on AWS Fargate via Amazon CloudFront

2 Application backend sends the prompt to Amazon API Gateway, AWS Lambda validates the requests, and publishes to an Amazon SNS topic

3 SNS publishes the message to Amazon SQS queues via matching subscription filters

4 In Amazon EKS cluster, Kubernetes Event Driven AutoScaler (KEDA) scales up new pods based on the queue length

5 Karpenter launches new GPU spot instances to place pending pods. The nodes run Bottlerocket OS with pre-cached Stable Diffusion Web UI images

6 SD Web UI loads model files from Amazon EFS file system

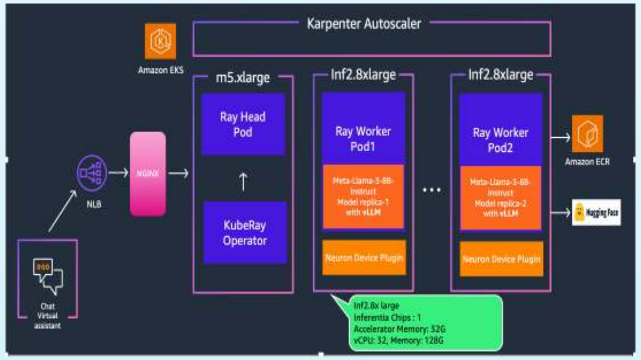
7 Queue Agent calls Web UI APIs to generate images and save to Amazon S3

8 Queue Agent sends output to a SNS topic and the application backend receives notification from SQS queue

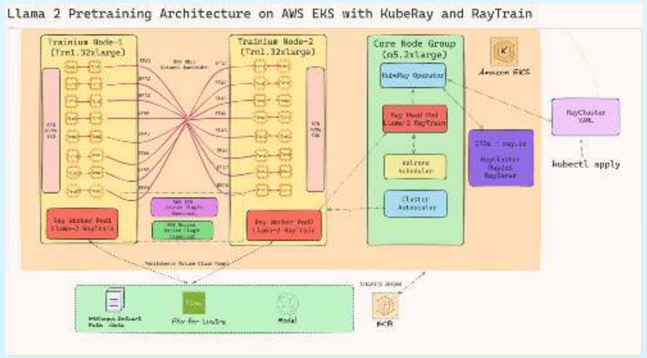
9 Amazon CloudWatch, AWS Distro for OpenTelemetry, and AWS X-Ray collect metrics, logs, and traces to monitor all the components

10 AWS IAM for security, AWS CDK for deployment of guidance components into AWS

Async inference with Stable Diffusion on EKS



Inference of Llama-3-8B with RayServe/vLLM on EKS



Llama 2 Pretraining Architecture on AWS EKS with KubeRay and RayTrain
Trainium Node-1 (Trn1.32xlarge)
Trainium Node-2 (Trn1.32xlarge)
Core Node Group (m5.2xlarge)
Amazon EKS
KubeRay Operator
RayCluster YAML
kubectl apply
ECR

Llama2 distributed pretraining on Trn1 with RayTrain

Thanks.