Red Hat Summit

Connect

Engineering Guardrailed AI for Constrained Environments

Daniele Zonca

Chief Architect, AI Platform Engineering, Red Hat

In the era of agentic AI, speed must not come at the cost of **operational safety**.

This session explores how platform engineers can implement **guardrailed autonomy** to provide a framework where AI agents act freely while staying strictly within company policies and security principles.

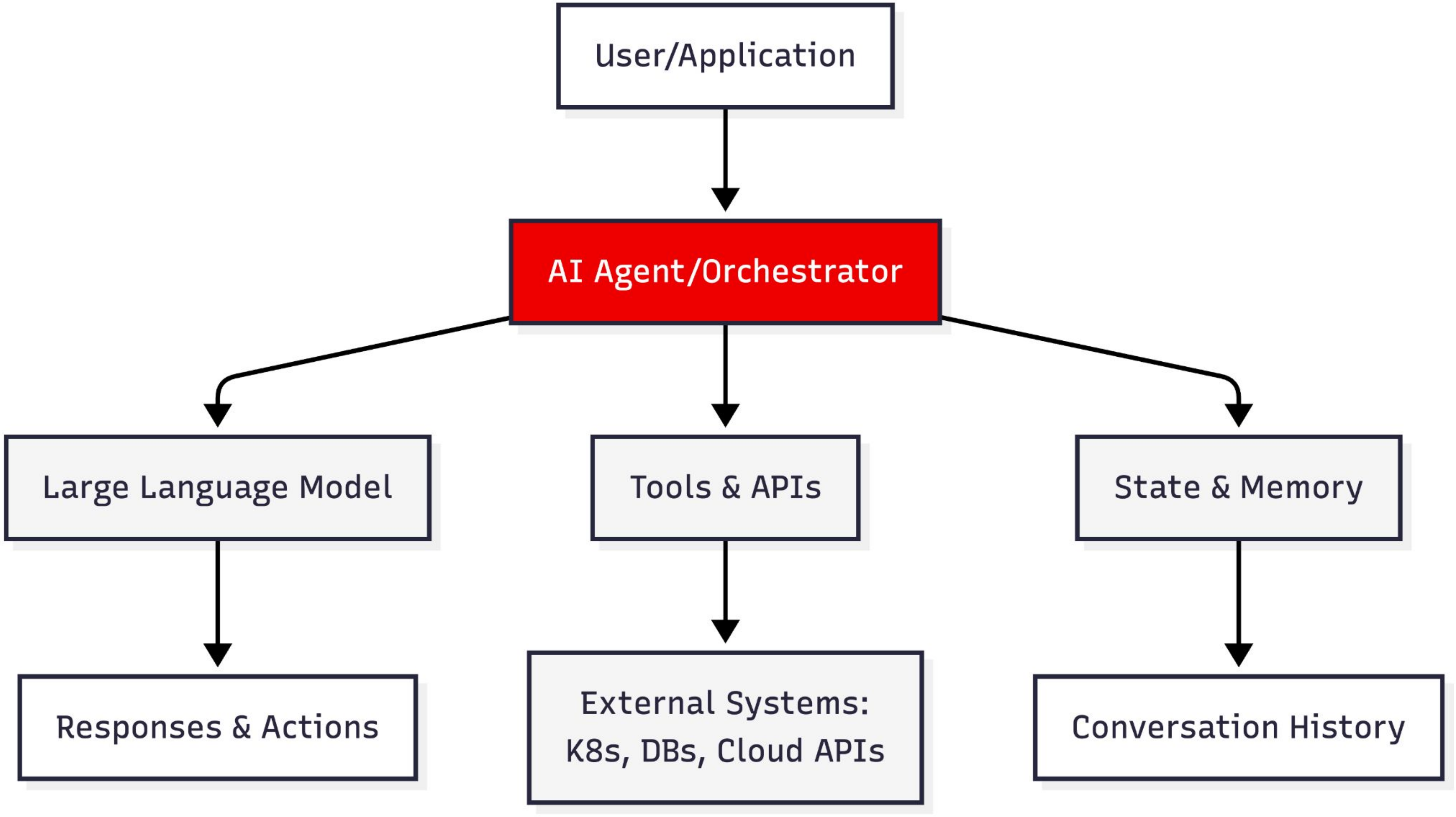
Through a practical demo, we will show how to build the **platform boundaries** that allow for constrained agency and ensure AI remains a productive and predictable asset in your environment.

In the era of agentic AI, speed must not come at the cost of **operational safety**.

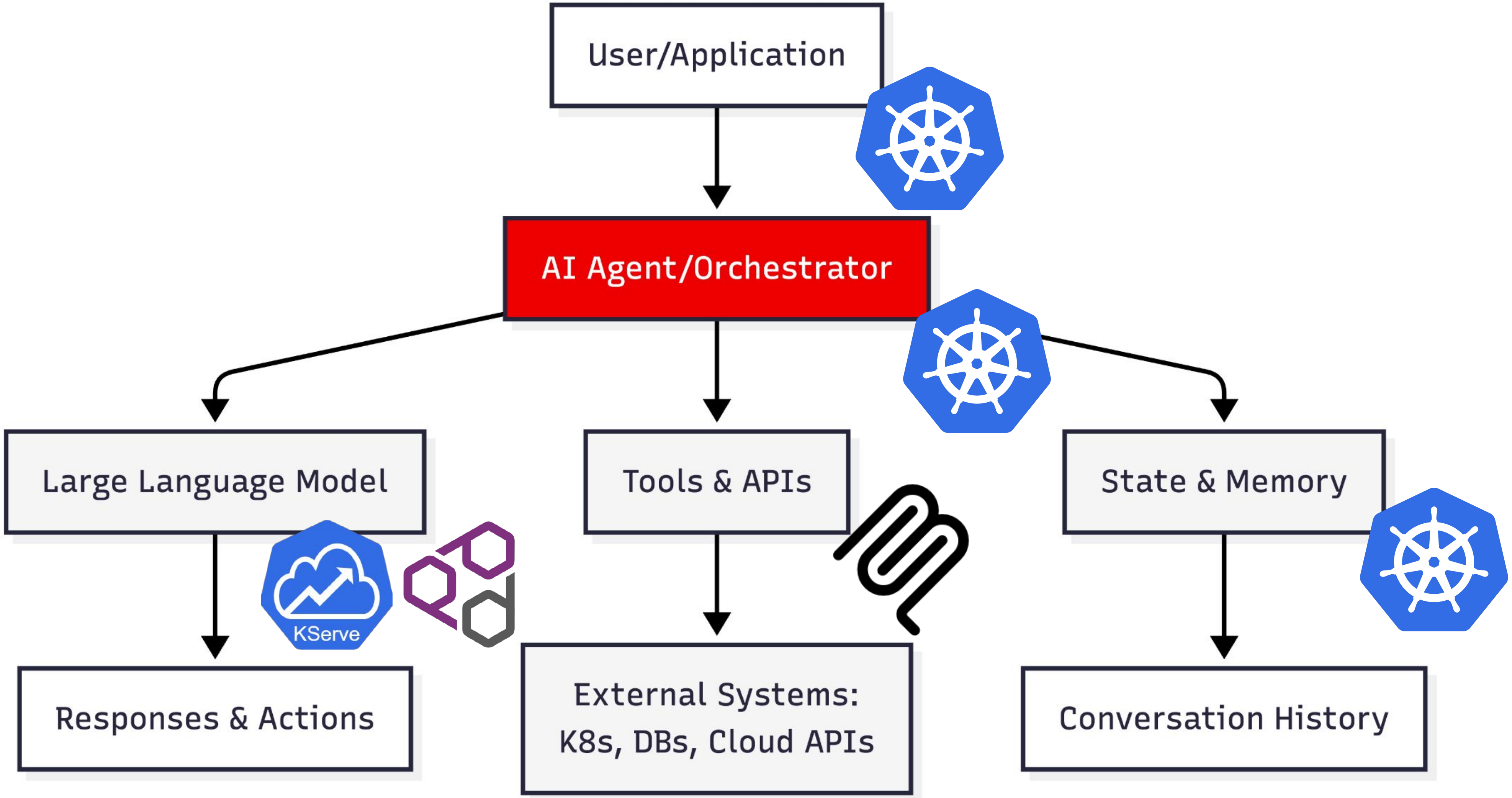
This session explores how platform engineers can implement **guardrailed autonomy** to provide a framework where AI agents act freely while staying strictly within company policies and security principles.

Through a practical demo, we will show how to build the **platform boundaries** that allow for constrained agency and ensure AI remains a productive and predictable asset in your environment.

Anatomy of an Agentic AI System



Anatomy of an Agentic AI System



User/Application

AI Agent/Orchestrator

Large Language Model

Tools & APIs

State & Memory

Responses & Actions

External Systems:
K8s, DBs, Cloud APIs

Conversation History

Constrained Environments: A Familiar Challenge

What We Learned from DevOps

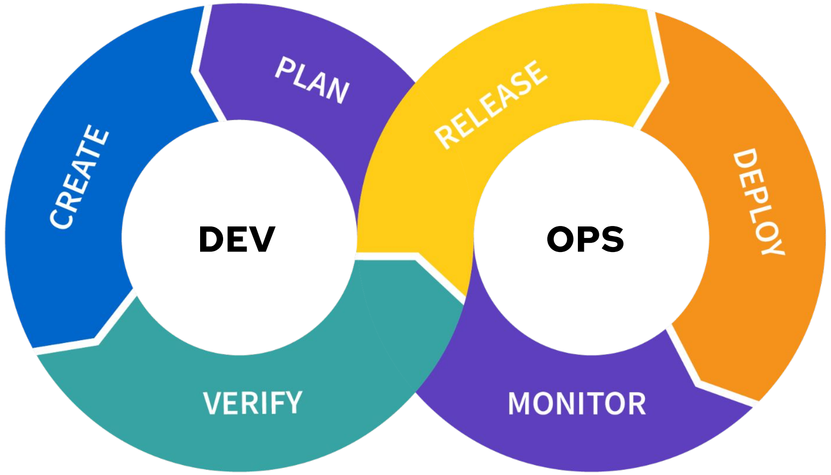
- **GitOps**: Declarative infrastructure
- **CI/CD**: Automated, governed pipelines
- **Internal Developer Platform**: Self-service with guardrails
- **Policy Enforcement**: Admission control (deploy) & network policies (runtime)
- **Result**: Flexibility + Governance

The AI Parallel

Same challenge, new domain:

- Enable innovation
- Preserve governance
- Meet compliance requirements

But with new complications...



Every Agentic AI System is Hybrid

The Hybrid Deployment Reality

Traditional Application via DevOps

- Single cluster deployment
- All components in single location
- Consistent network topology
- Uniform RBAC and policies

AI Platform Reality

- **LLMs are everywhere**: SaaS, local or dedicated GPU cluster
- **Multiple models**: Embeddings for RAG and Predictive AI models
- **Different Storage**: VectorDB, on-prem datalakes/data warehouse for data residency
- **Agent Runtime**: Different frameworks and tools (**MCP** as communication protocol)

Challenge: How to implement common governance?

Shift-left: Build & Deploy Time

Platform Boundaries and Guardrails

What

- **Models**
  - Approved models and versions only
  - Performance and safety thresholds
- **Agent Behavior**
  - Allowed actions and tools
  - Topic and data restrictions
- **Deployment**
  - Environment isolation
  - Approval gates for production

How

- **Models**
  - Model Evaluation
  - Sign Artifacts with Secure Supply Chain
- **Agent Behavior**
  - (Automated) Red Team exercise
  - Tool validation
- **Deployment**
  - Admission control (like OPA/Kyverno)
  - RBAC

Runtime (Per-Request)

Platform Boundaries and Guardrails

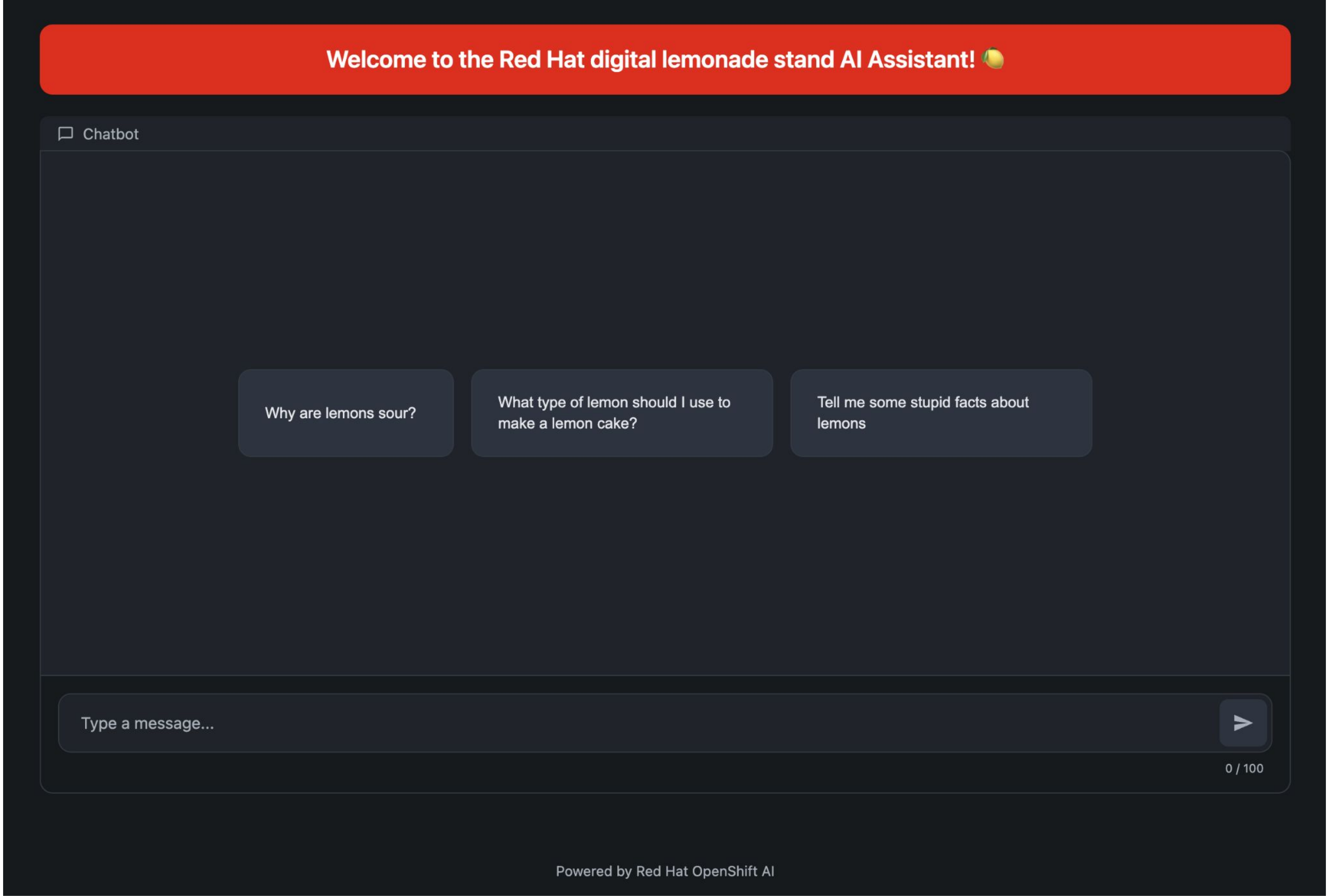
What

- **Input/Output**
  - Safety: allowed topics and patterns
  - Hallucination tolerance
- **Interaction**
  - Permitted operations
  - Data access scope
- **Resources**
  - Token/cost budgets
  - Execution/rate limits

How

- **Input/Output**
  - Configure detectors based on risks
  - Guardrails orchestration
- **Interaction**
  - Network policies
  - Encryption
- **Resources**
  - Unified (Standard) API
  - Quota management

Lemonade Stand



Welcome to the Red Hat digital lemonade stand AI Assistant! 🍋

Chatbot
Why are lemons sour?
What type of lemon should I use to make a lemon cake?
Tell me some stupid facts about lemons
Type a message...
0 / 100

Powered by Red Hat OpenShift AI

My lemonade is the best!

Customer service agent to learn more about my product

Two main risks to address

- All conversations with the agent are family friendly
  - **No toxic language**
- It does not promote our rival fruit juice vendors
  - **No rival mentions**

Build & Deploy Time

Shift-left: Build & Deploy Time

Platform Boundaries and Guardrails

What

- **Models**
  - Approved models and versions only
  - Performance and safety thresholds
- **Agent Behavior**
  - Allowed actions and tools
  - Topic and data restrictions
- **Deployment**
  - Environment isolation
  - Approval gates for production

How

- **Models**
  - Model Evaluation
  - Sign Artifacts with Secure Supply Chain
- **Agent Behavior**
  - (Automated) Red Team exercise
  - Tool validation
- **Deployment**
  - Admission control (like OPA/Kyverno)
  - RBAC

Shift-left: Build & Deploy Time

Platform Boundaries and Guardrails

What

- **Models**
  - Approved models and versions only
  - Performance and safety thresholds
- **Agent Behavior**
  - Allowed actions and tools
  - Topic and data restrictions
- **Deployment**
  - Environment isolation
  - Approval gates for production

How

lm-evaluation-harness

- **Models**
  - Model Evaluation
  - Sign Artifacts with Secure Supply Chain

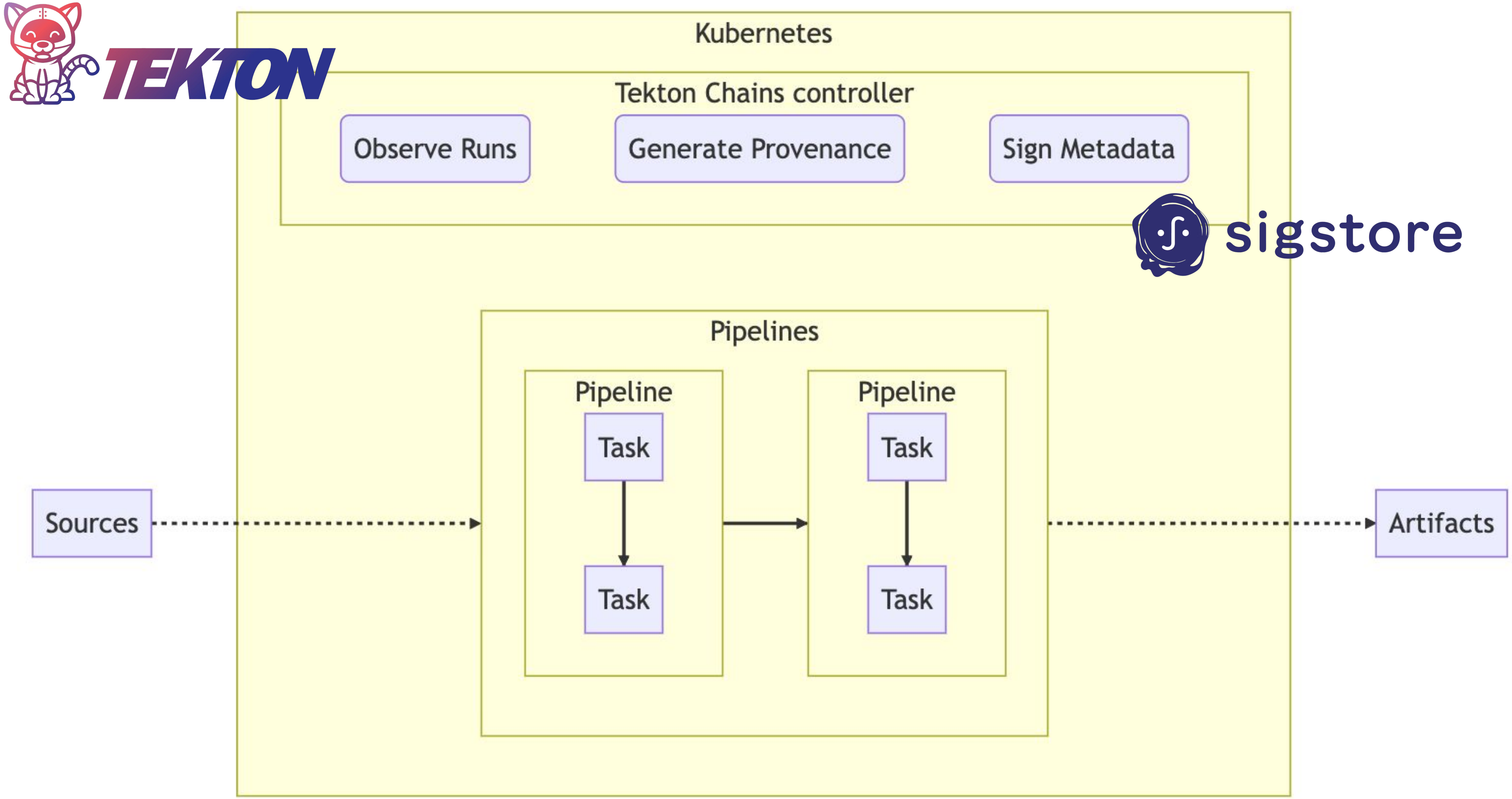
TrustyAI

- **Agent Behavior**
  - (Automated) Red Team exercise
  - Tool validation

TEKTON

NVIDIA garak

- **Deployment**
  - Admission control (like OPA/Kyverno)
  - RBAC



TEKTON

Kubernetes

Tekton Chains controller

Observe Runs

Generate Provenance

Sign Metadata

sigstore

Sources

Pipelines

Pipeline

Task

Task

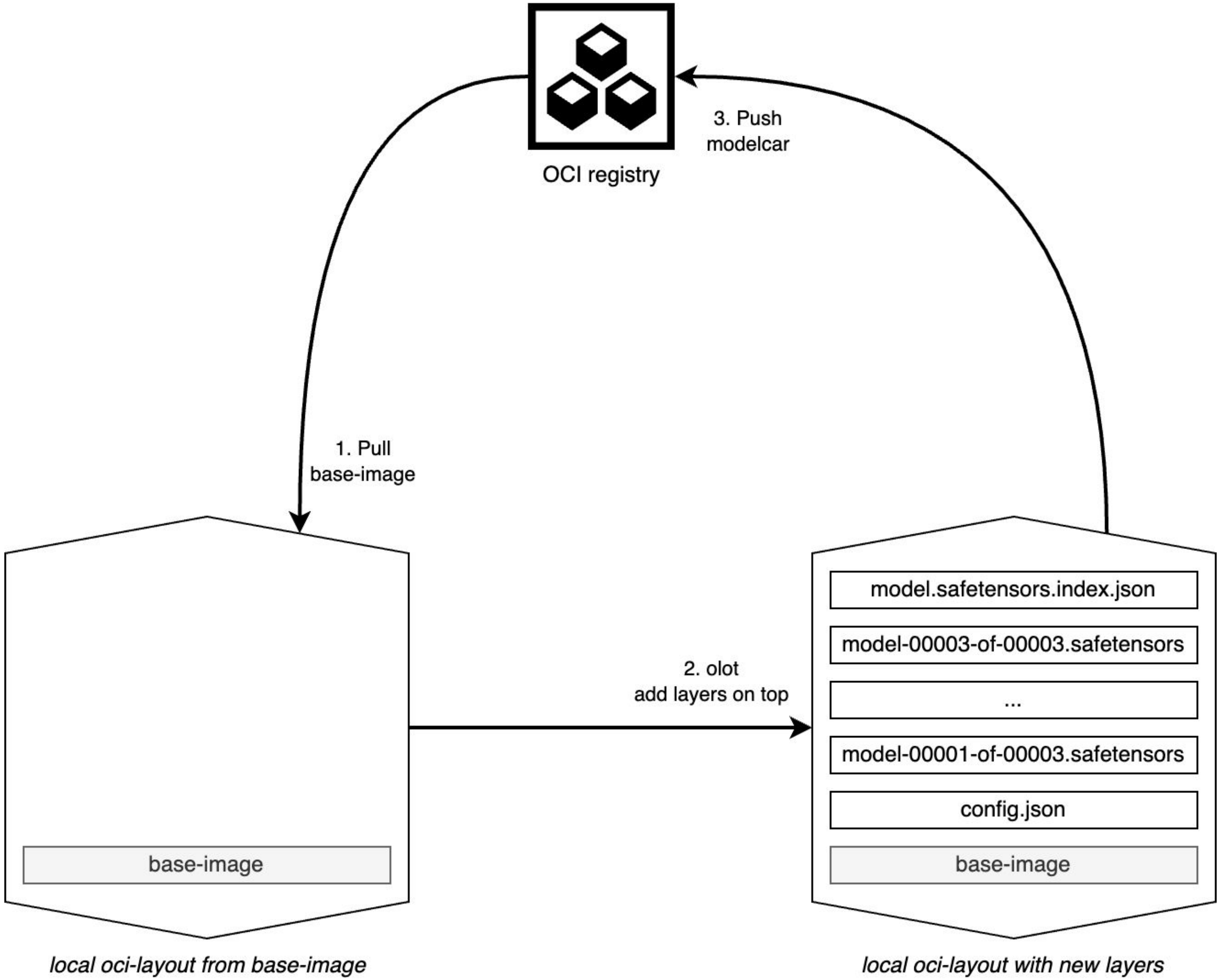
Pipeline

Task

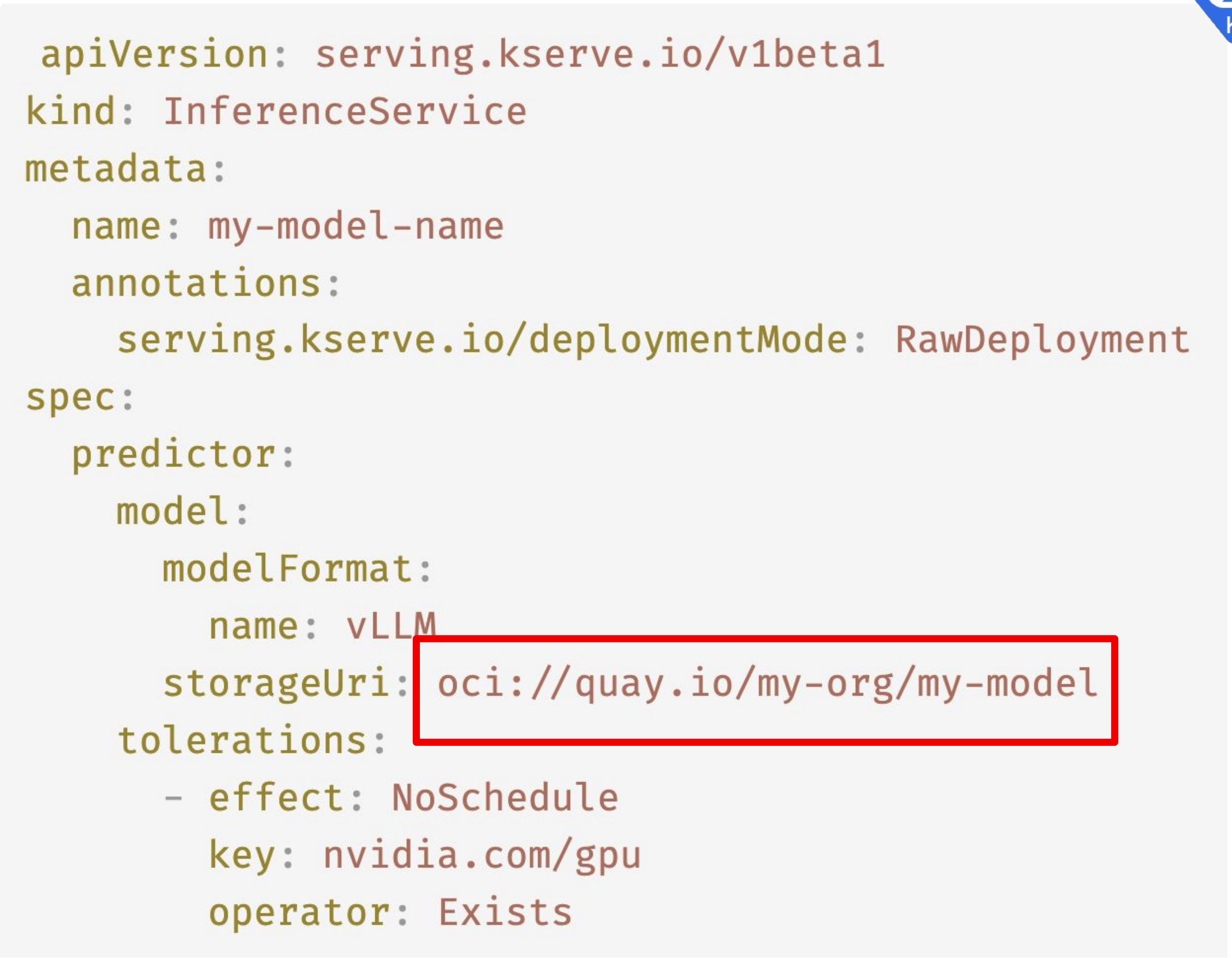
Task

Artifacts

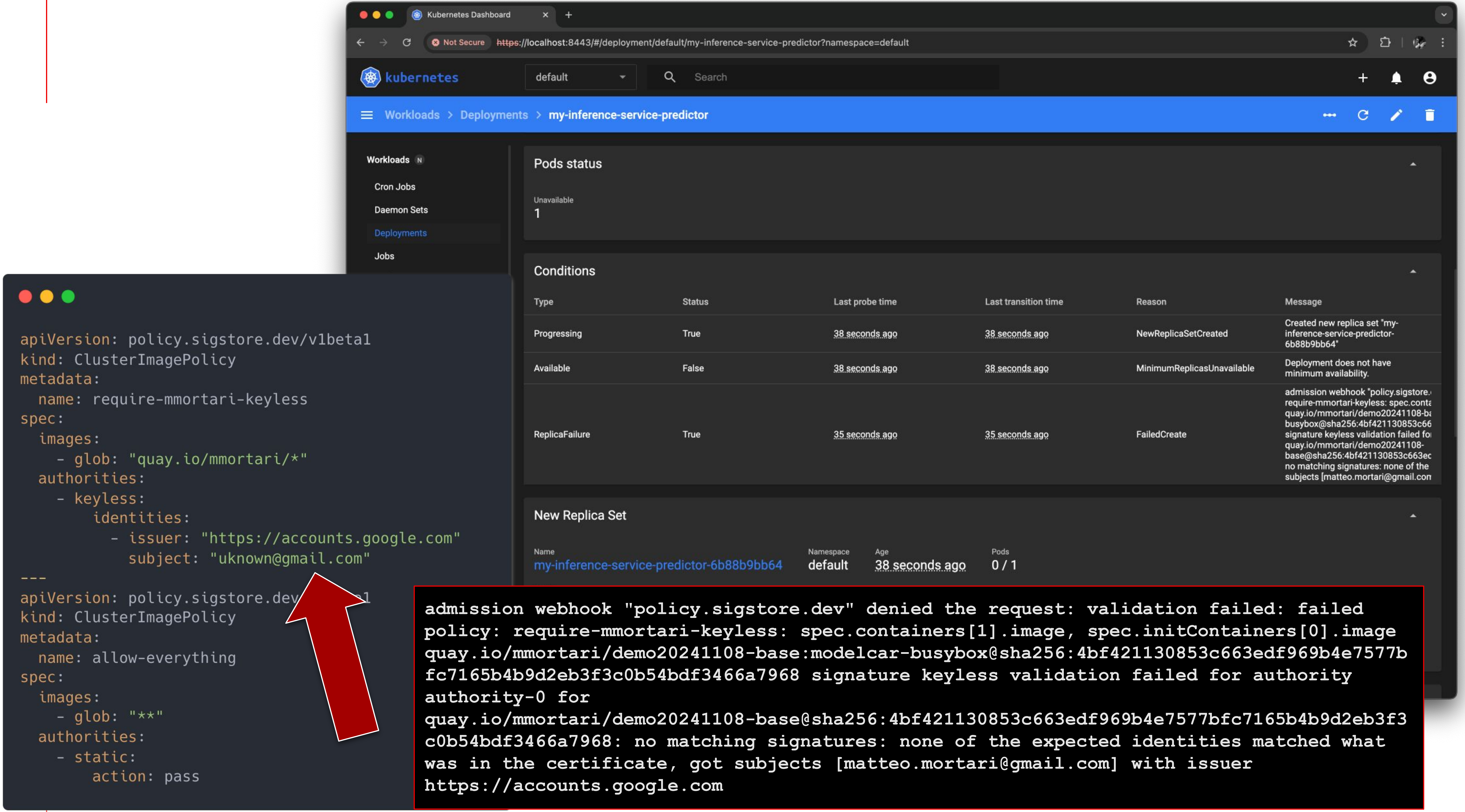
Package the model in OCI


OCI registry
3. Push
modelcar
1. Pull
base-image
2. olot
add layers on top
base-image
model.safetensors.index.json
model-00003-of-00003.safetensors
...
model-00001-of-00003.safetensors
config.json
base-image
local oci-layout from base-image
local oci-layout with new layers

Deploy the model



```yaml
apiVersion: serving.kserve.io/v1beta1
kind: InferenceService
metadata:
  name: my-model-name
  annotations:
    serving.kserve.io/deploymentMode: RawDeployment
spec:
  predictor:
    model:
      modelFormat:
        name: vLLM
      storageUri: oci://quay.io/my-org/my-model
    tolerations:
      - effect: NoSchedule
        key: nvidia.com/gpu
        operator: Exists
```



Kubernetes Dashboard
Not Secure
https://localhost:8443/#/deployment/default/my-inference-service-predictor?namespace=default
kubernetes
default
Search
Workloads > Deployments > my-inference-service-predictor
Workloads N
Cron Jobs
Daemon Sets
Deployments
Jobs

Pods status

Unavailable
1

Conditions

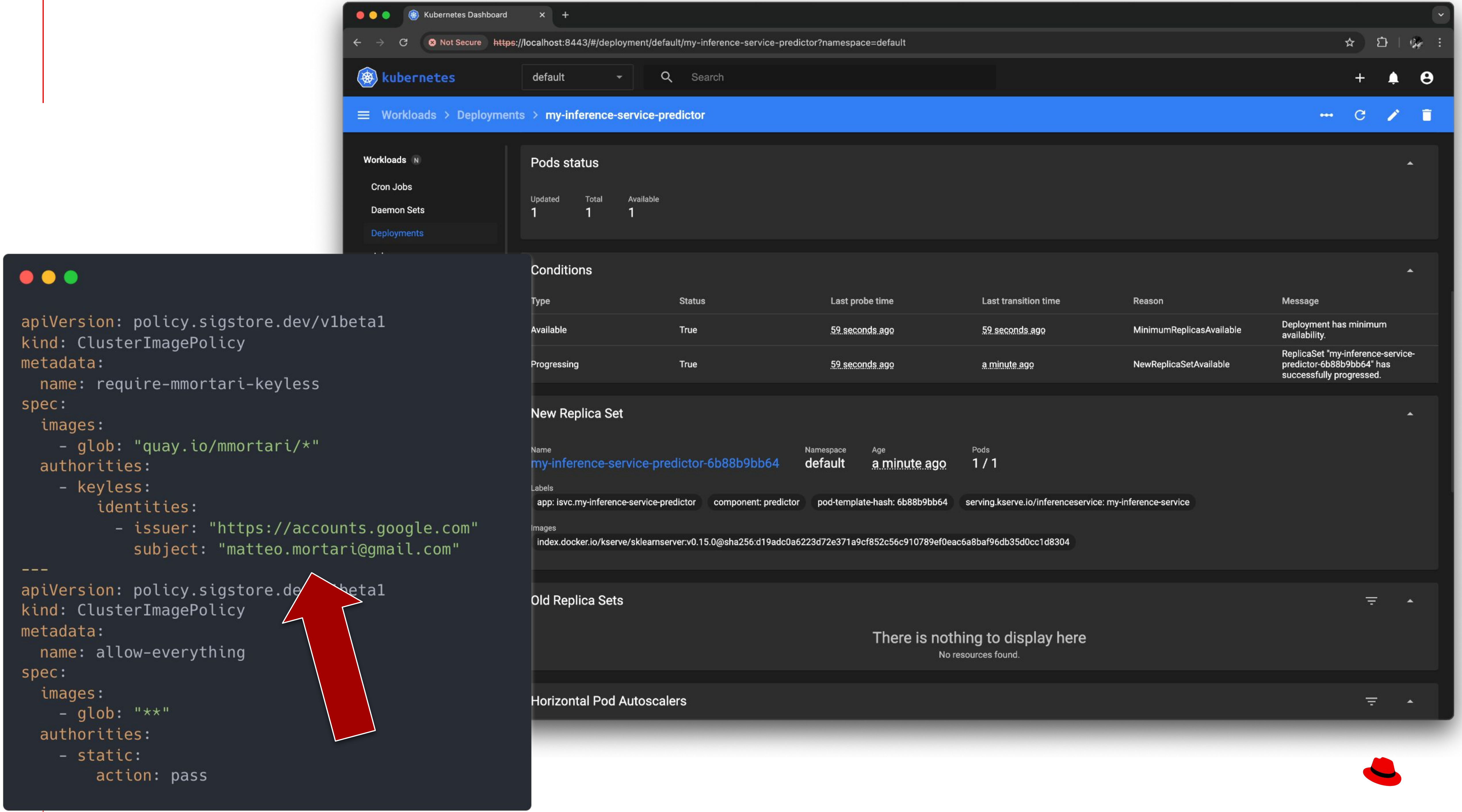
Type	Status	Last probe time	Last transition time	Reason	Message
Progressing	True	38 seconds ago	38 seconds ago	NewReplicaSetCreated	Created new replica set "my-inference-service-predictor-6b88b9bb64"
Available	False	38 seconds ago	38 seconds ago	MinimumReplicasUnavailable	Deployment does not have minimum availability.
ReplicaFailure	True	35 seconds ago	35 seconds ago	FailedCreate	admission webhook "policy.sigstore. require-mmortari-keyless: spec.conta quay.io/mmortari/demo20241108-ba busybox@sha256:4bf421130853c66 signature keyless validation failed fo quay.io/mmortari/demo20241108- base@sha256:4bf421130853c663ec no matching signatures: none of the subjects [matteo.mortari@gmail.con

New Replica Set

Name	Namespace	Age	Pods
my-inference-service-predictor-6b88b9bb64	default	38 seconds ago	0 / 1

```yaml
apiVersion: policy.sigstore.dev/v1beta1
kind: ClusterImagePolicy
metadata:
  name: require-mmortari-keyless
spec:
  images:
    - glob: "quay.io/mmortari/*"
  authorities:
    - keyless:
        identities:
          - issuer: "https://accounts.google.com"
            subject: "uknown@gmail.com"
---
apiVersion: policy.sigstore.dev/v1beta1
kind: ClusterImagePolicy
metadata:
  name: allow-everything
spec:
  images:
    - glob: "**"
  authorities:
    - static:
        action: pass
```

```
admission webhook "policy.sigstore.dev" denied the request: validation failed: failed
policy: require-mmortari-keyless: spec.containers[1].image, spec.initContainers[0].image
quay.io/mmortari/demo20241108-base:modelcar-busybox@sha256:4bf421130853c663edf969b4e7577b
fc7165b4b9d2eb3f3c0b54bdf3466a7968 signature keyless validation failed for authority
authority-0 for
quay.io/mmortari/demo20241108-base@sha256:4bf421130853c663edf969b4e7577bfc7165b4b9d2eb3f3
c0b54bdf3466a7968: no matching signatures: none of the expected identities matched what
was in the certificate, got subjects [matteo.mortari@gmail.com] with issuer
https://accounts.google.com
```



Kubernetes Dashboard
Not Secure
https://localhost:8443/#/deployment/default/my-inference-service-predictor?namespace=default
kubernetes
default
Search
Workloads > Deployments > my-inference-service-predictor
Workloads N
Cron Jobs
Daemon Sets
Deployments

Pods status

Updated	Total	Available
1	1	1

Conditions

Type	Status	Last probe time	Last transition time	Reason	Message
Available	True	59 seconds ago	59 seconds ago	MinimumReplicasAvailable	Deployment has minimum availability.
Progressing	True	59 seconds ago	a minute ago	NewReplicaSetAvailable	ReplicaSet "my-inference-service-predictor-6b88b9bb64" has successfully progressed.

New Replica Set

Name	Namespace	Age	Pods
my-inference-service-predictor-6b88b9bb64	default	a minute ago	1 / 1

Labels

app: isvc.my-inference-service-predictor
component: predictor
pod-template-hash: 6b88b9bb64
serving.kserve.io/inferenceservice: my-inference-service

Images

index.docker.io/kserve/sklearnserver:v0.15.0@sha256:d19adc0a6223d72e371a9cf852c56c910789ef0eac6a8baf96db35d0cc1d8304

Old Replica Sets

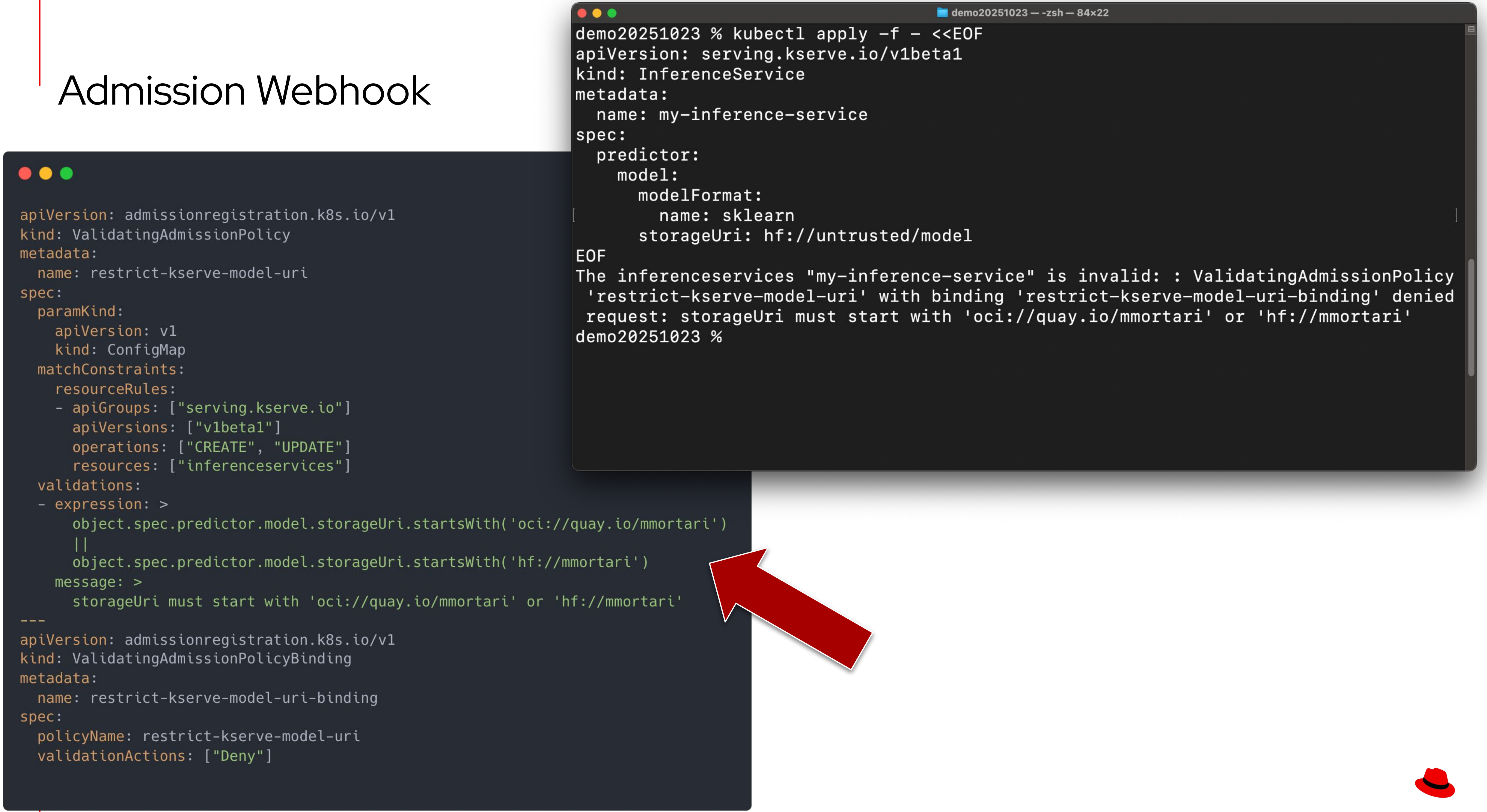
There is nothing to display here

No resources found.

Horizontal Pod Autoscalers

```yaml
apiVersion: policy.sigstore.dev/v1beta1
kind: ClusterImagePolicy
metadata:
  name: require-mmortari-keyless
spec:
  images:
    - glob: "quay.io/mmortari/*"
  authorities:
    - keyless:
        identities:
          - issuer: "https://accounts.google.com"
            subject: "matteo.mortari@gmail.com"
---
apiVersion: policy.sigstore.de      beta1
kind: ClusterImagePolicy
metadata:
  name: allow-everything
spec:
  images:
    - glob: "**"
  authorities:
    - static:
        action: pass
```

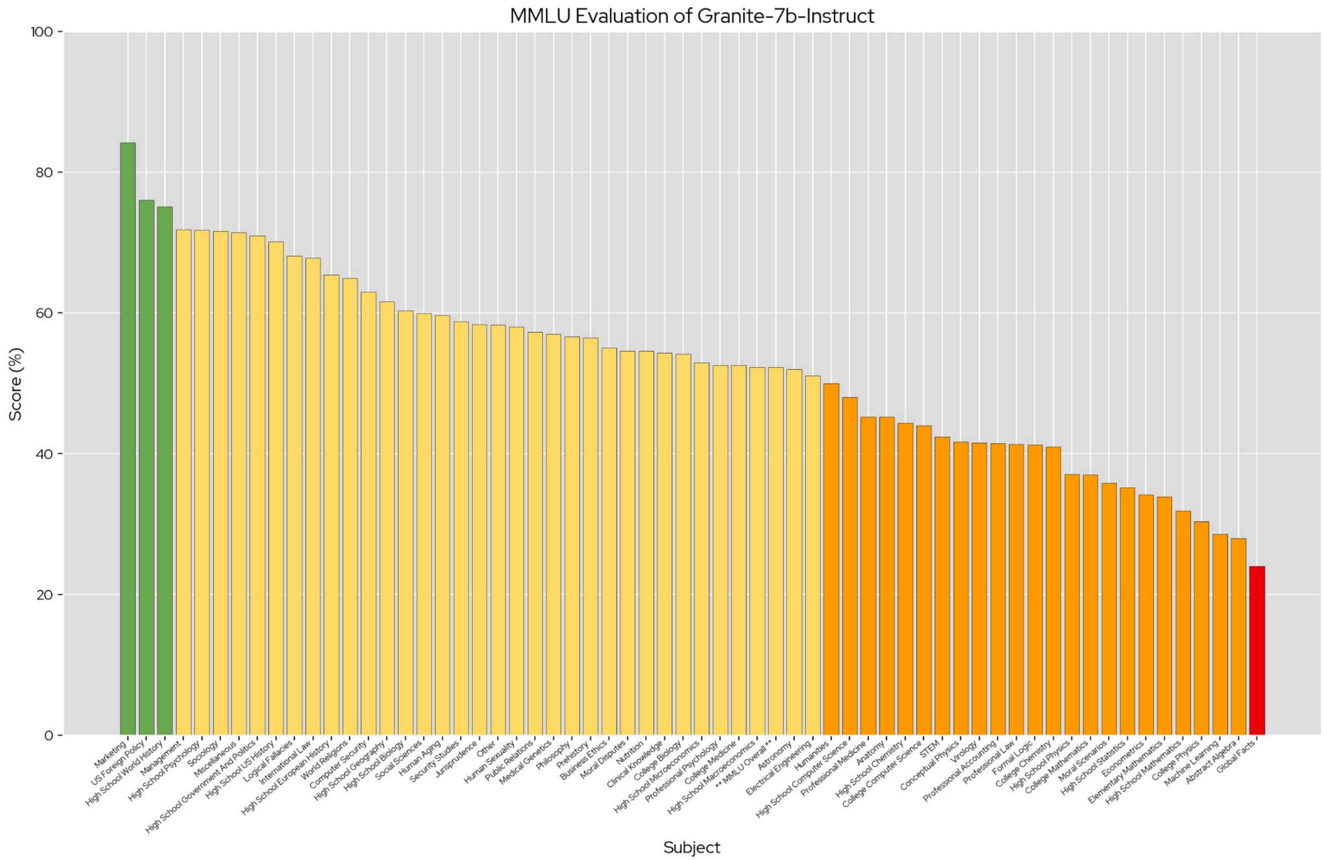
Admission Webhook



```yaml
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingAdmissionPolicy
metadata:
  name: restrict-kserve-model-uri
spec:
  paramKind:
    apiVersion: v1
    kind: ConfigMap
  matchConstraints:
    resourceRules:
    - apiGroups: ["serving.kserve.io"]
      apiVersions: ["v1beta1"]
      operations: ["CREATE", "UPDATE"]
      resources: ["inferenceservices"]
  validations:
  - expression: >
      object.spec.predictor.model.storageUri.startsWith('oci://quay.io/mmortari')
      ||
      object.spec.predictor.model.storageUri.startsWith('hf://mmortari')
    message: >
      storageUri must start with 'oci://quay.io/mmortari' or 'hf://mmortari'
---
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingAdmissionPolicyBinding
metadata:
  name: restrict-kserve-model-uri-binding
spec:
  policyName: restrict-kserve-model-uri
  validationActions: ["Deny"]
```

```
demo20251023 % kubectl apply -f - <<EOF
apiVersion: serving.kserve.io/v1beta1
kind: InferenceService
metadata:
  name: my-inference-service
spec:
  predictor:
    model:
      modelFormat:
        name: sklearn
      storageUri: hf://untrusted/model
EOF
The inferenceservices "my-inference-service" is invalid: : ValidatingAdmissionPolicy
 'restrict-kserve-model-uri' with binding 'restrict-kserve-model-uri-binding' denied
 request: storageUri must start with 'oci://quay.io/mmortari' or 'hf://mmortari'
demo20251023 %
```

Model Evaluation (LM-Eval)



Perform a huge variety of evaluation tasks over LLMs to understand and quantify their knowledge, capabilities, and behaviors.

- 100+ out-of-the box evaluations or *tasks*
- Create custom tasks via Unitxt

Wraps EleutherAI's lm-evaluation-harness into a k8s environment

Language Model Evaluation (LM-Eval)

A few sample tasks from the 100+ default tasks

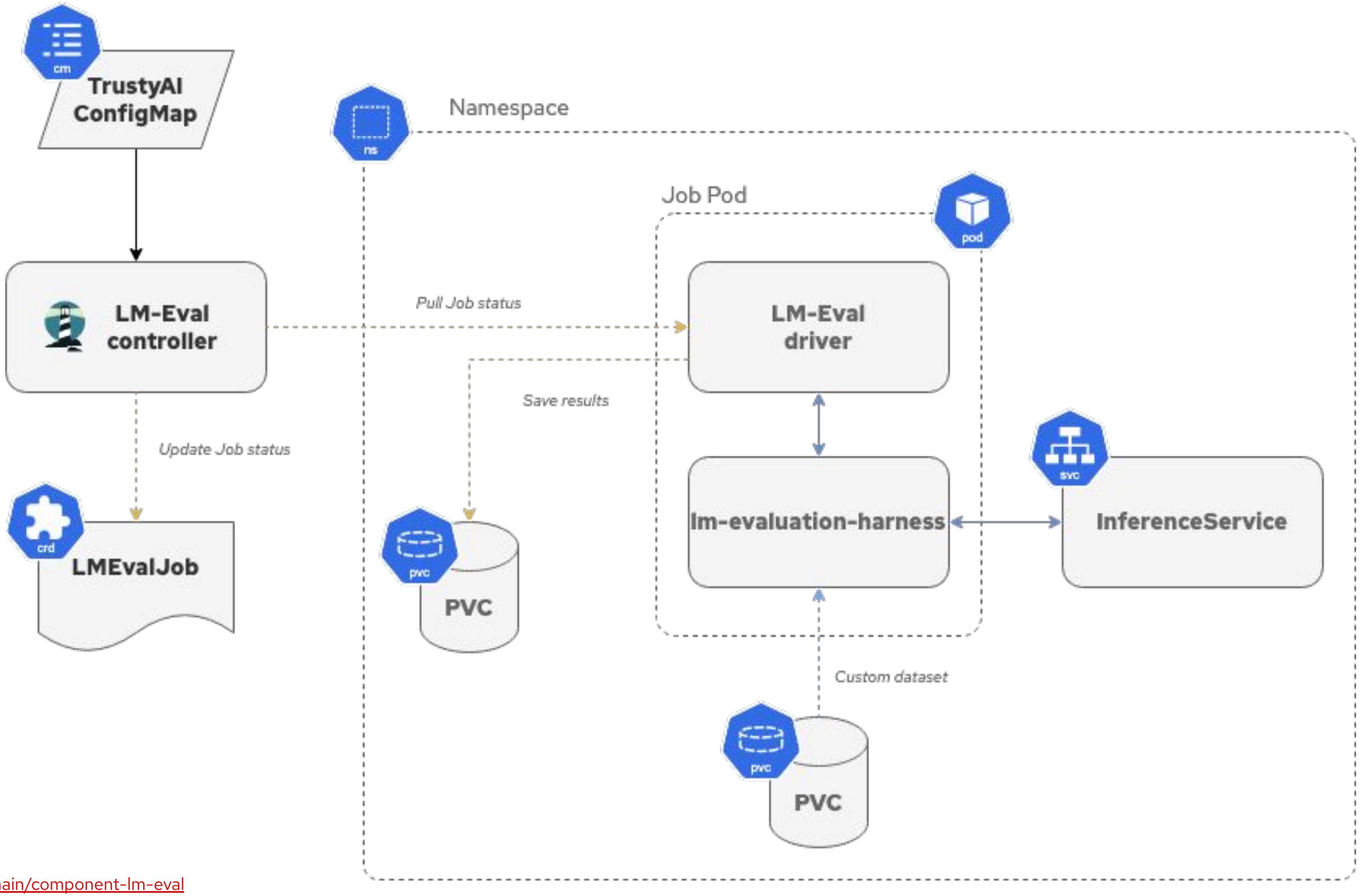
logiqa	Logical reasoning tasks requiring advanced inference and deduction.
anli	Adversarial natural language inference tasks designed to test model robustness.
asdiv	Tasks involving arithmetic and mathematical reasoning challenges.
realtoxicityprompts	Tasks to evaluate language models for generating text with potential toxicity.
medqa	Multiple choice question answering based on the United States Medical License Exams
eq_bench	Tasks focused on equality and ethics in question answering and decision-making.
crows_pairs	Tasks designed to test model biases in various sociodemographic groups.

Language Model Evaluation (LM-Eval)

A few sample tasks from the 100+ default tasks

logiqa	Logical reasoning tasks requiring advanced inference and deduction.
anli	Adversarial natural language inference tasks designed to test model robustness.
asdiv	Tasks involving arithmetic and mathematical reasoning challenges.
realtoxicityprompts	Tasks to evaluate language models for generating text with potential toxicity.
medqa	Multiple choice question answering based on the United States Medical License Exams
eq_bench	Tasks focused on equality and ethics in question answering and decision-making.
crows_pairs	Tasks designed to test model biases in various sociodemographic groups.

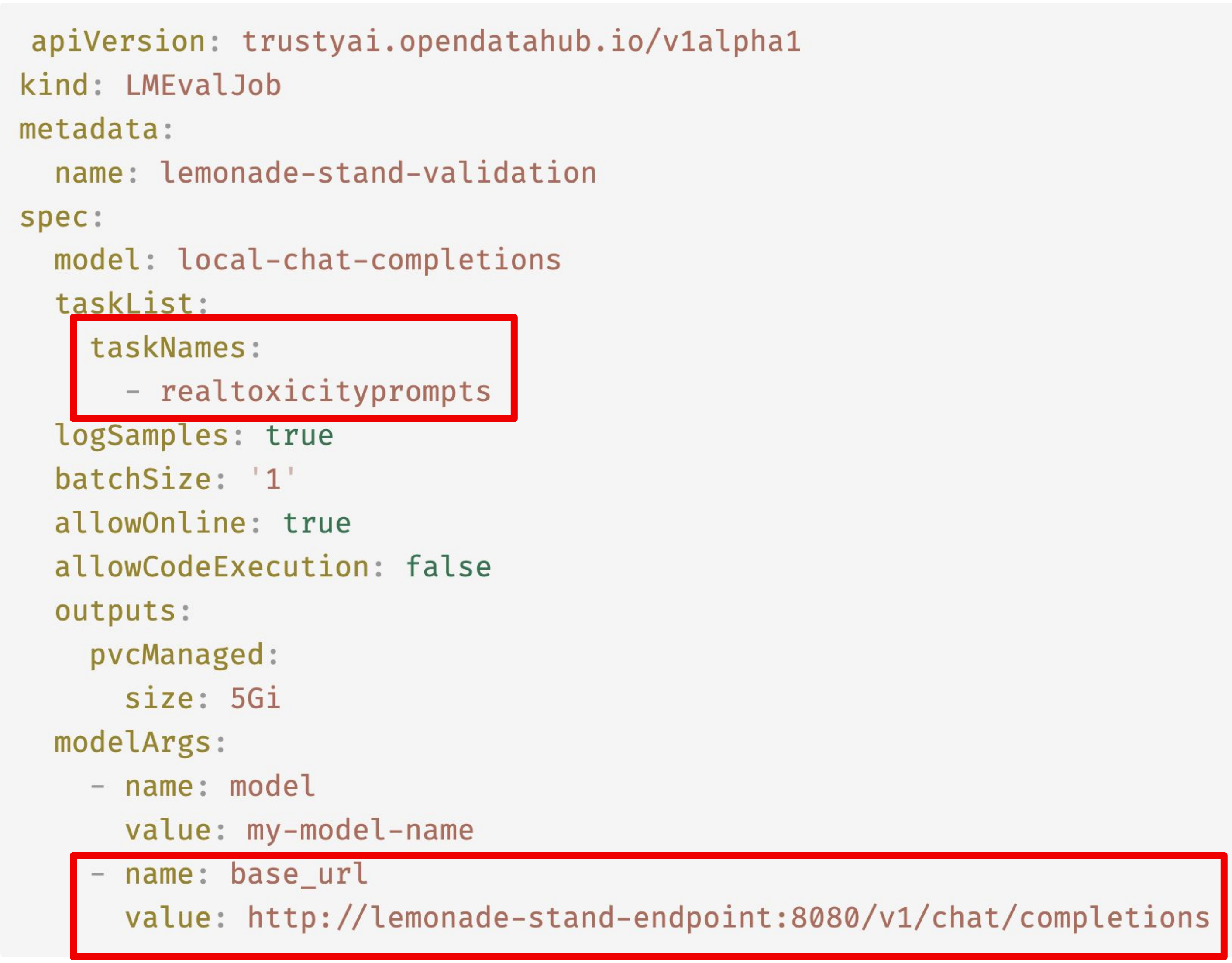
TrustyAI LMEvalJob



https://trustyai.org/docs/main/component-lm-eval

Evaluate the model

TrustyAI



```
apiVersion: trustyai.opendatahub.io/v1alpha1
kind: LMEvalJob
metadata:
  name: lemonade-stand-validation
spec:
  model: local-chat-completions
  taskList:
    taskNames:
      - realtoxicityprompts
  logSamples: true
  batchSize: '1'
  allowOnline: true
  allowCodeExecution: false
  outputs:
    pvcManaged:
      size: 5Gi
  modelArgs:
    - name: model
      value: my-model-name
    - name: base_url
      value: http://lemonade-stand-endpoint:8080/v1/chat/completions
```

Runtime
(Per-Request)

Runtime (Per-Request)

Platform Boundaries and Guardrails

What

- **Input/Output**
  - Safety: allowed topics and patterns
  - Hallucination tolerance
- **Interaction**
  - Permitted operations
  - Data access scope
- **Resources**
  - Token/cost budgets
  - Execution/rate limits

How

- **Input/Output**
  - Configure detectors based on risks
  - Guardrails orchestration
- **Interaction**
  - Network policies
  - Encryption
- **Resources**
  - Unified (Standard) API
  - Quota management

Runtime (Per-Request)

Platform Boundaries and Guardrails

What

- **Input/Output**
  - Safety: allowed topics and patterns
  - Hallucination tolerance
- **Interaction**
  - Permitted operations
  - Data access scope
- **Resources**
  - Token/cost budgets
  - Execution/rate limits

How

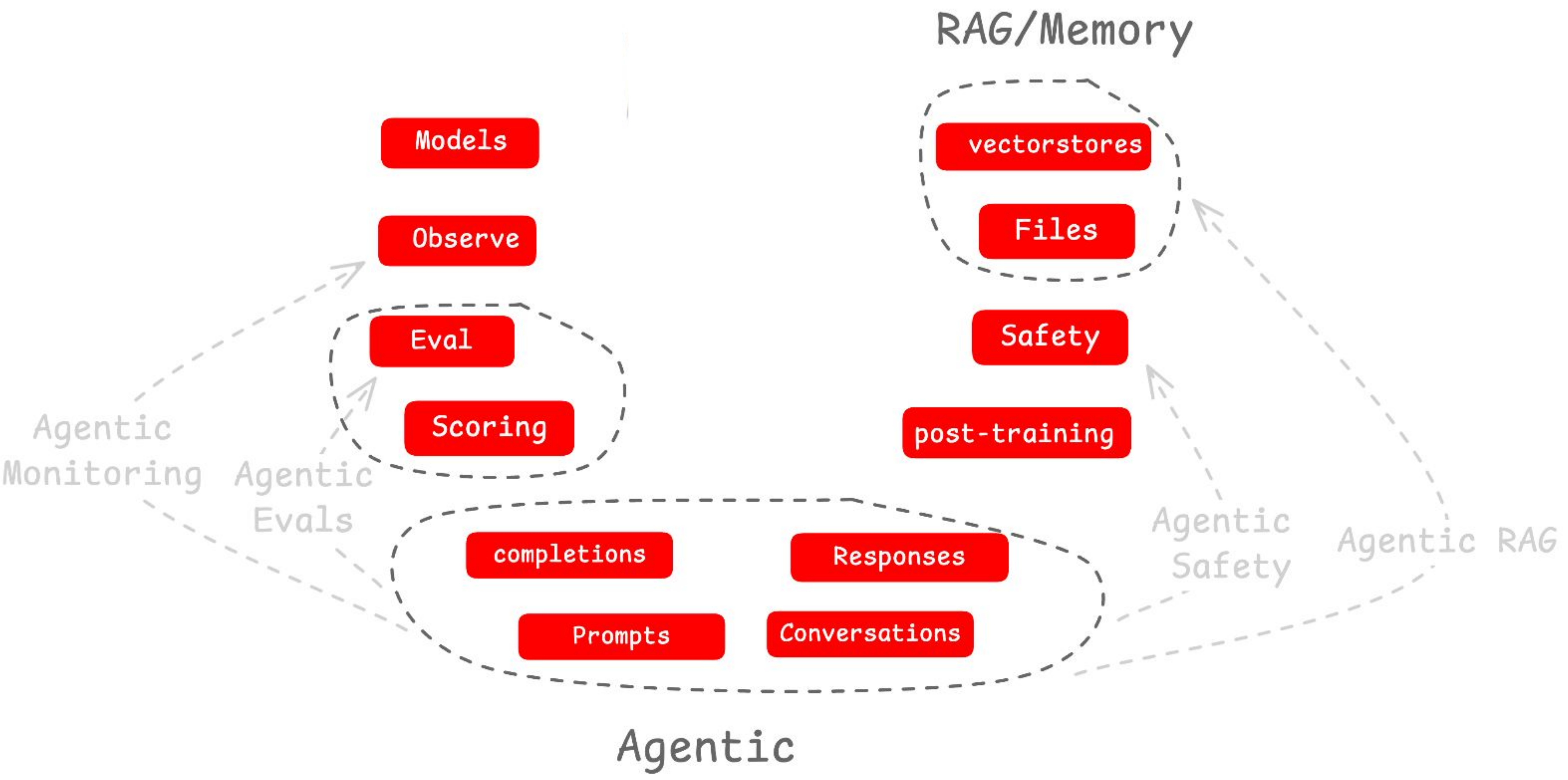
- **Input/Output**
  - Configure detectors based on risks
  - Guardrails orchestration
- **Interaction**
  - Network policies
  - Encryption
- **Resources**
  - Unified (Standard) API
  - Quota management

TrustyAI

Llama Stack

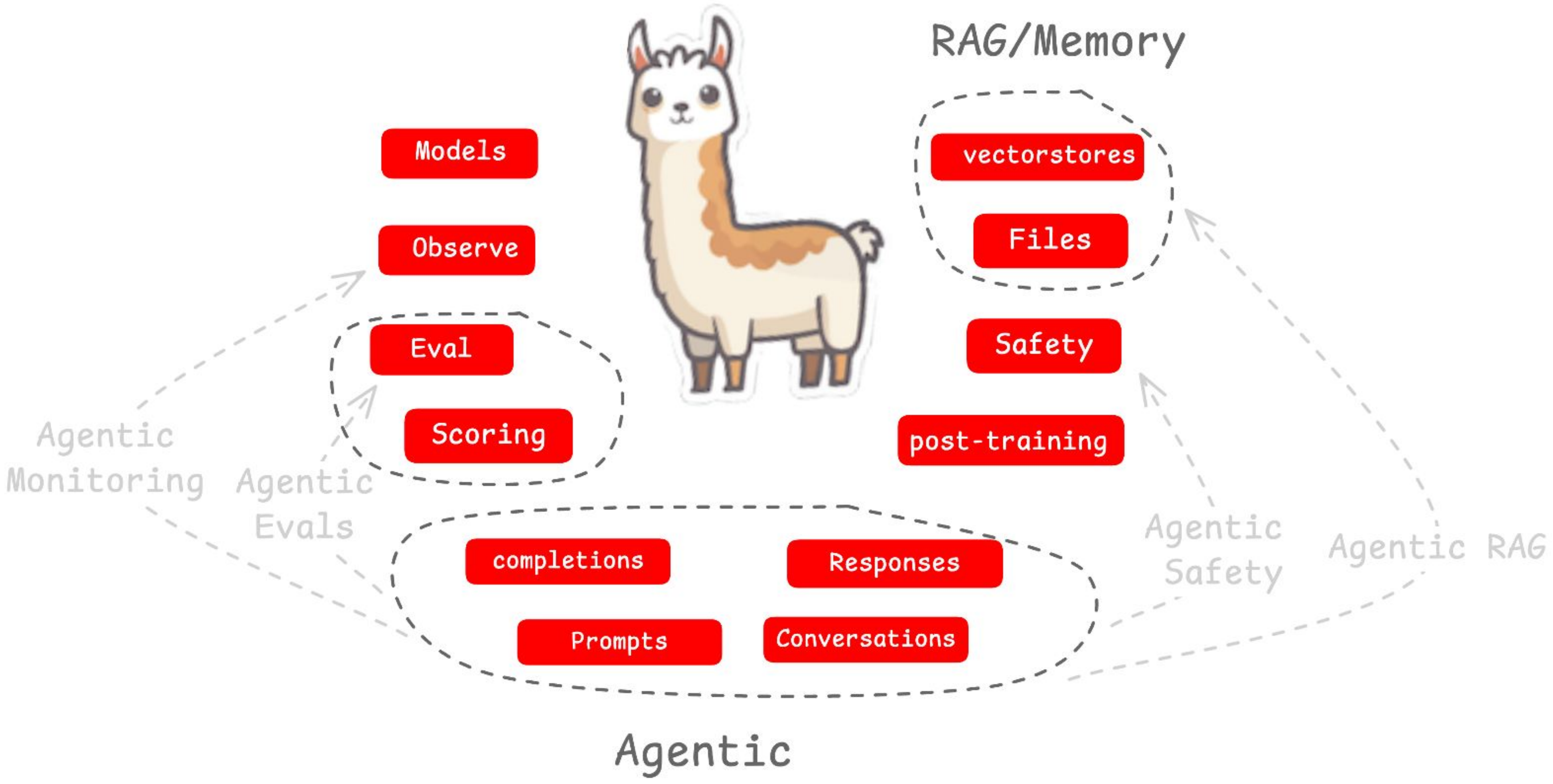
Unified API: Why

Multiple Entities (API) that are local to the cluster or remote

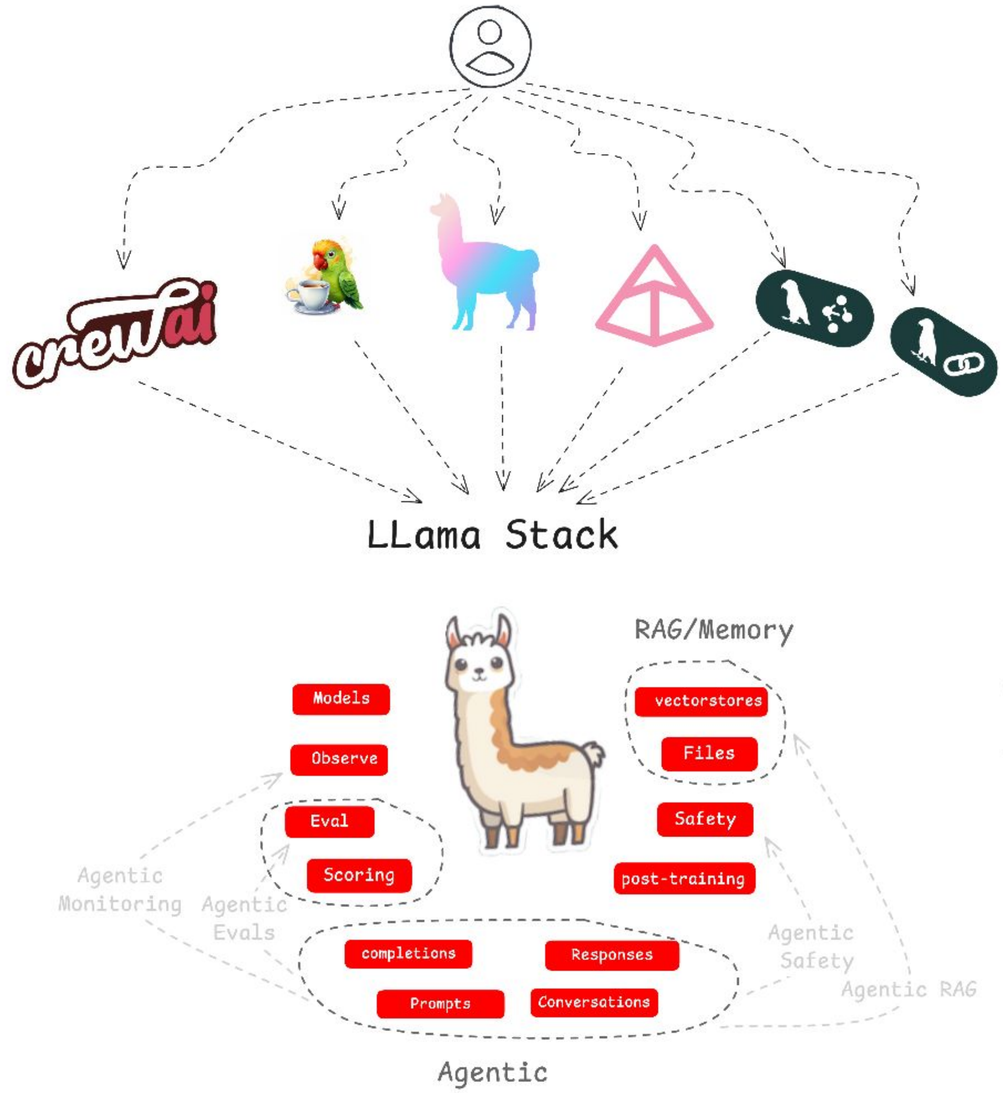


Llama Stack: a single API layer

OpenAI API Compatible

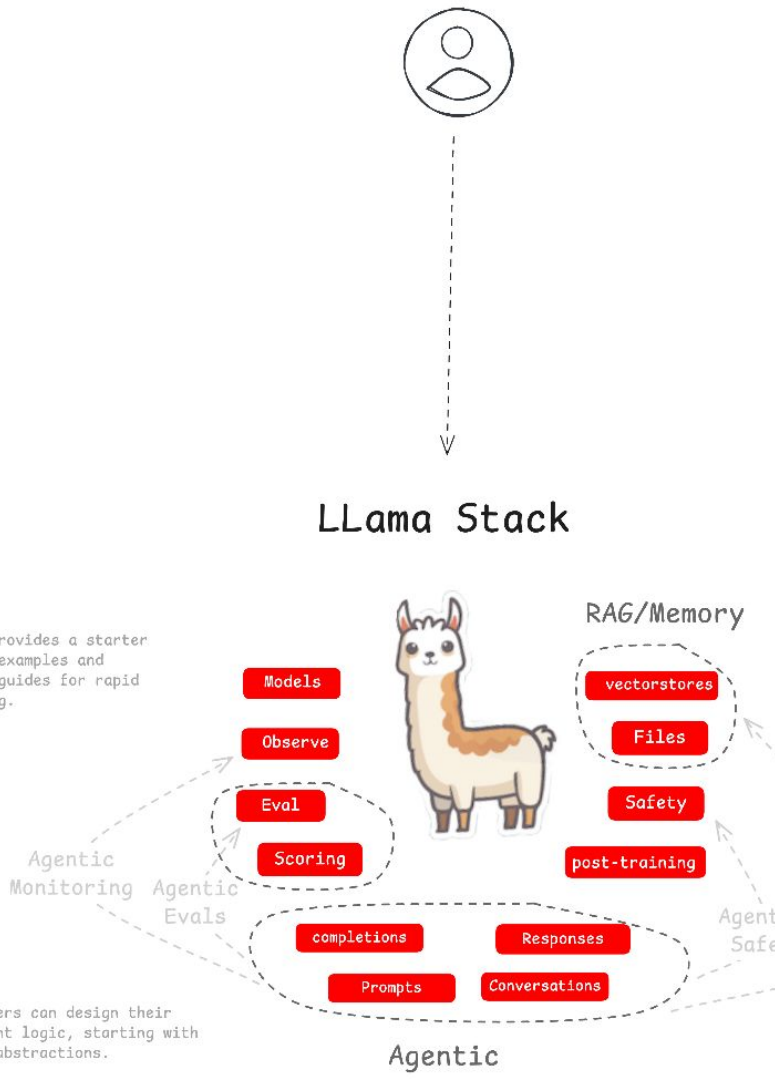


Build with your favorite Framework



Build Directly with Llama Stack

Red Hat provides a starter kit with examples and "how-to" guides for rapid onboarding.



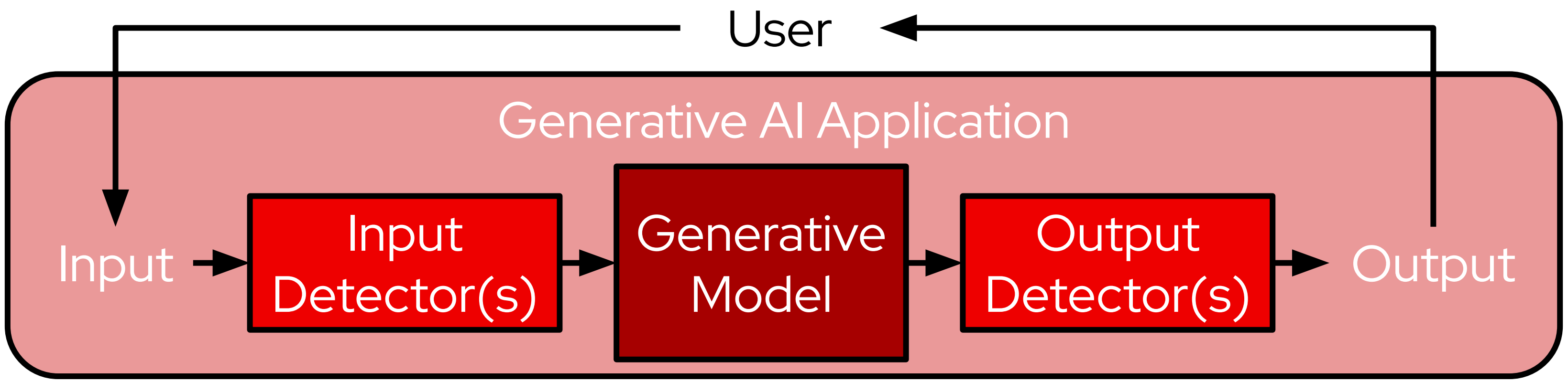
You build agentic applications directly with Llama Stack's native APIs (e.g., OpenAI-compatible Responses API).

Aevelopers can design their own agent logic, starting with Simple abstractions.

Highly customizable

fully integrated, enterprise-supported stack.

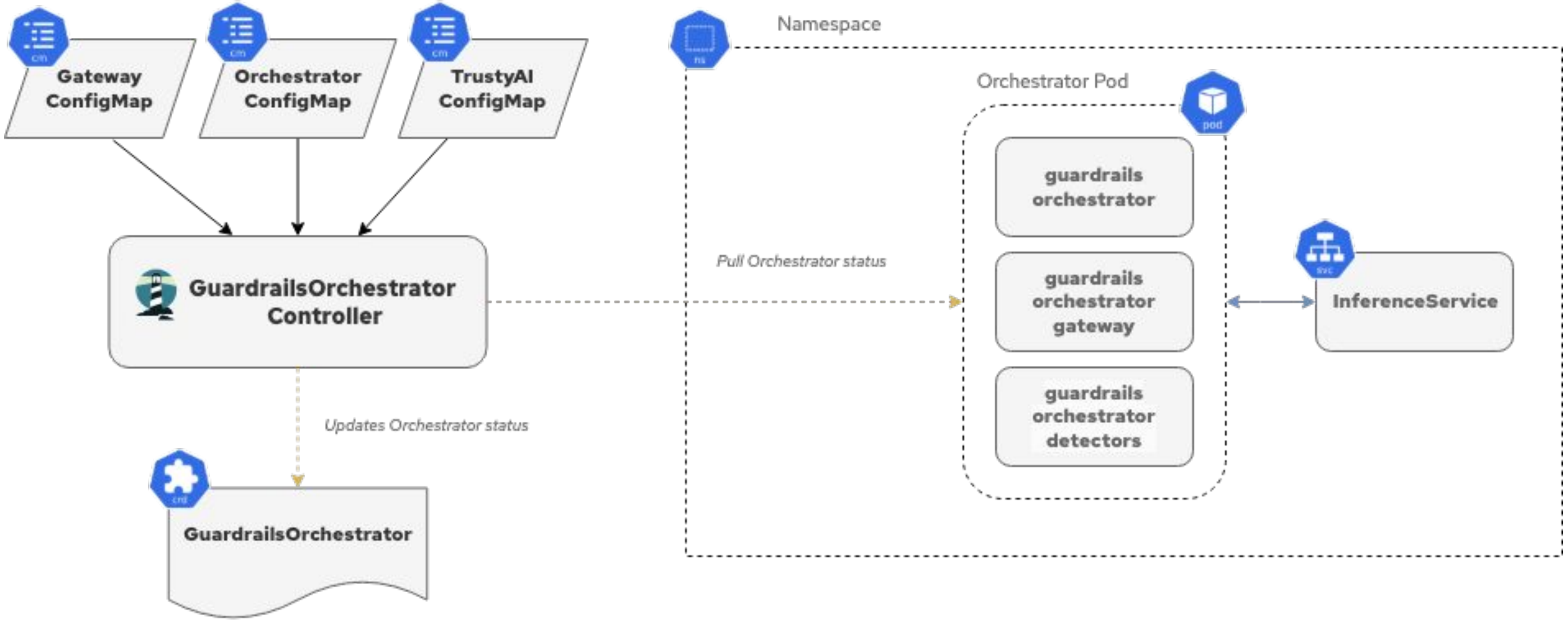
Guardrails



Moderates the **interaction pathways** between users and generative models, with

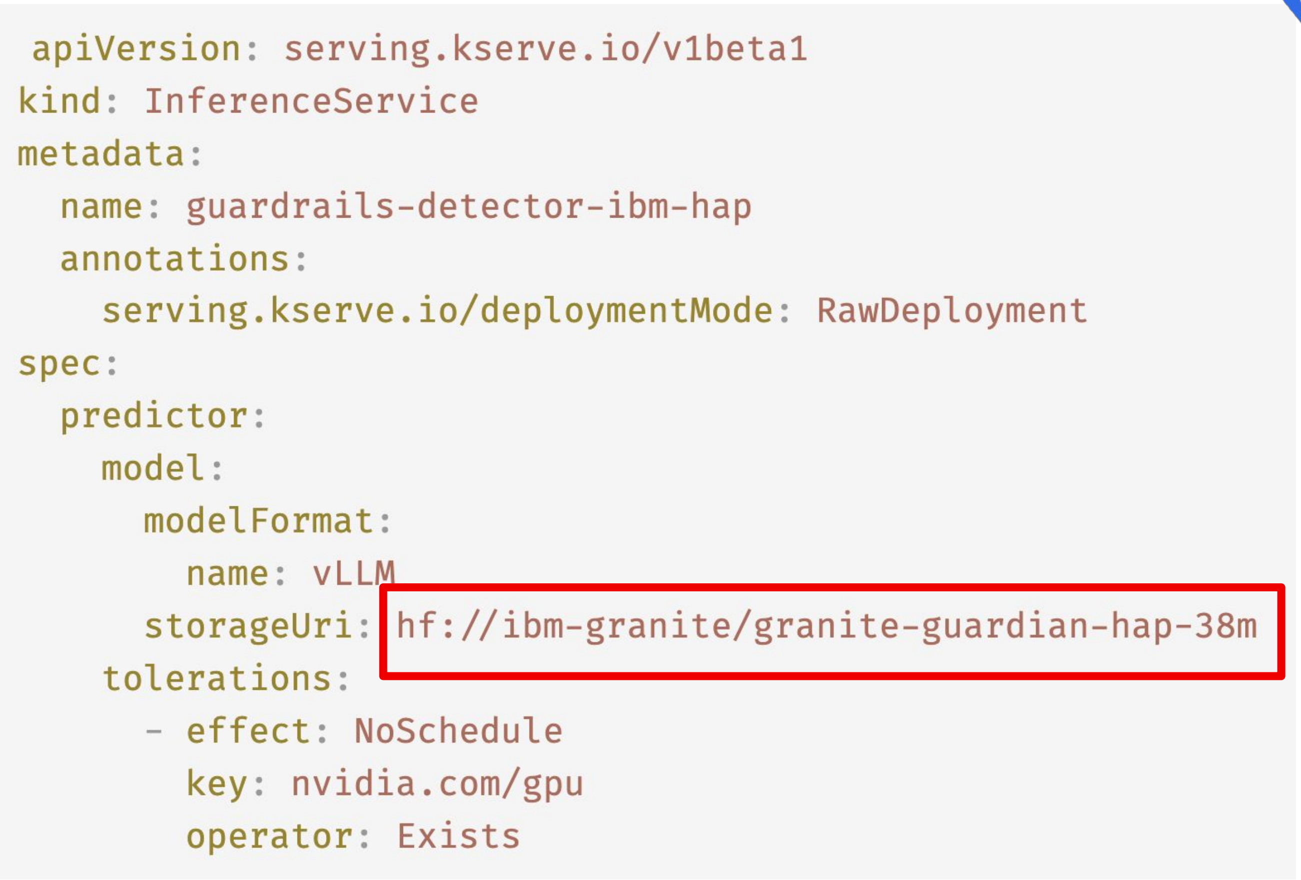
- Customizable input and output content validators
- Request-time configuration allowing **dynamic, per-request guardrailing**

TrustyAI Guardrails Orchestrator



Deploy the detectors

IBM Granite Guardian HAP 38M

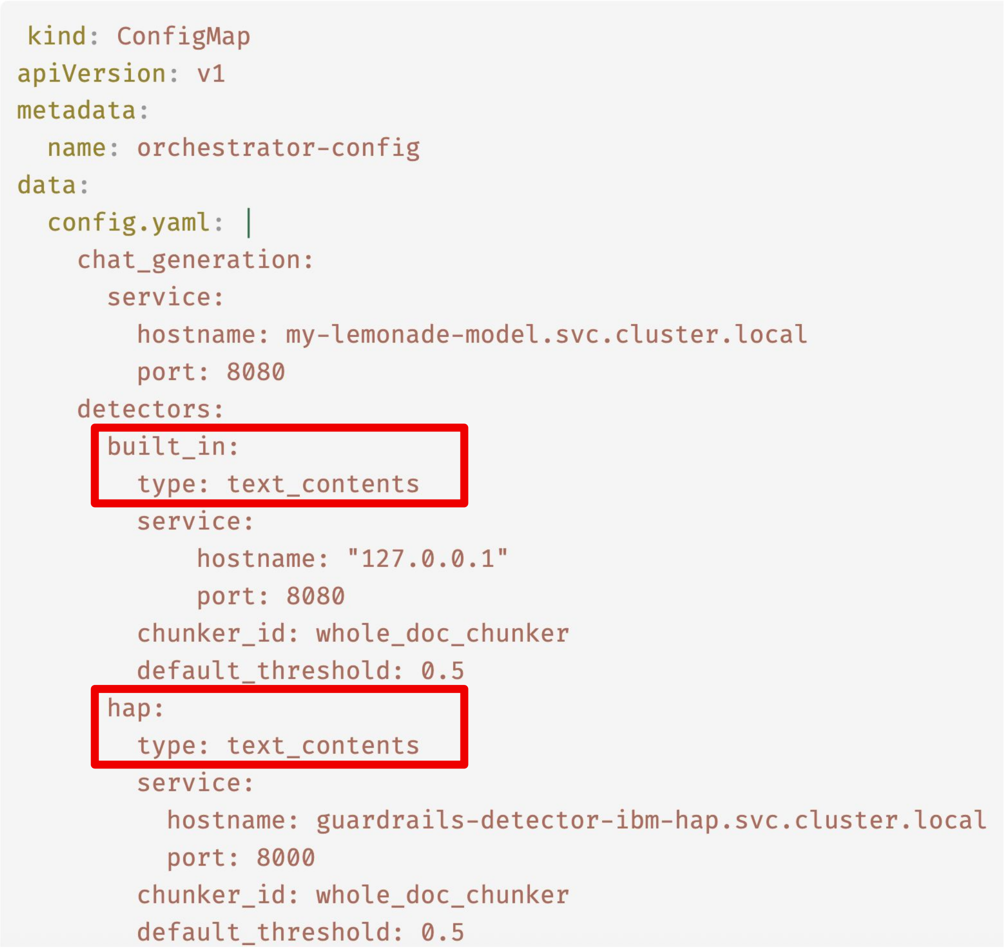


```
apiVersion: serving.kserve.io/v1beta1
kind: InferenceService
metadata:
  name: guardrails-detector-ibm-hap
  annotations:
    serving.kserve.io/deploymentMode: RawDeployment
spec:
  predictor:
    model:
      modelFormat:
        name: vLLM
      storageUri: hf://ibm-granite/granite-guardian-hap-38m
    tolerations:
      - effect: NoSchedule
        key: nvidia.com/gpu
        operator: Exists
```

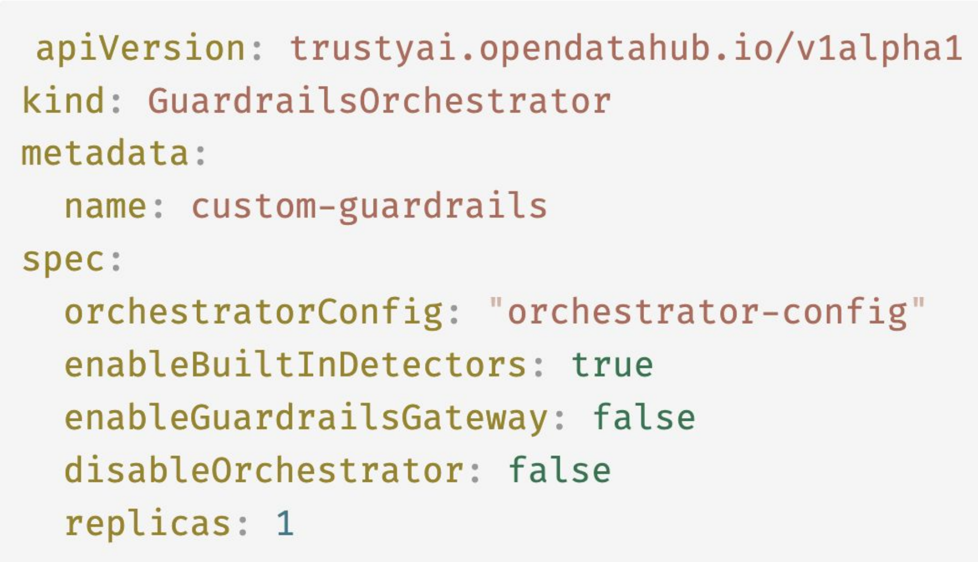
https://github.com/trustyai-explainability/trustyai-llm-demo/tree/LLSPlayground/lemonade-stand-demo
https://huggingface.co/ibm-granite/granite-guardian-hap-38m

Deploy the detectors (cont.)

TrustyAI Guardrails Orchestrator



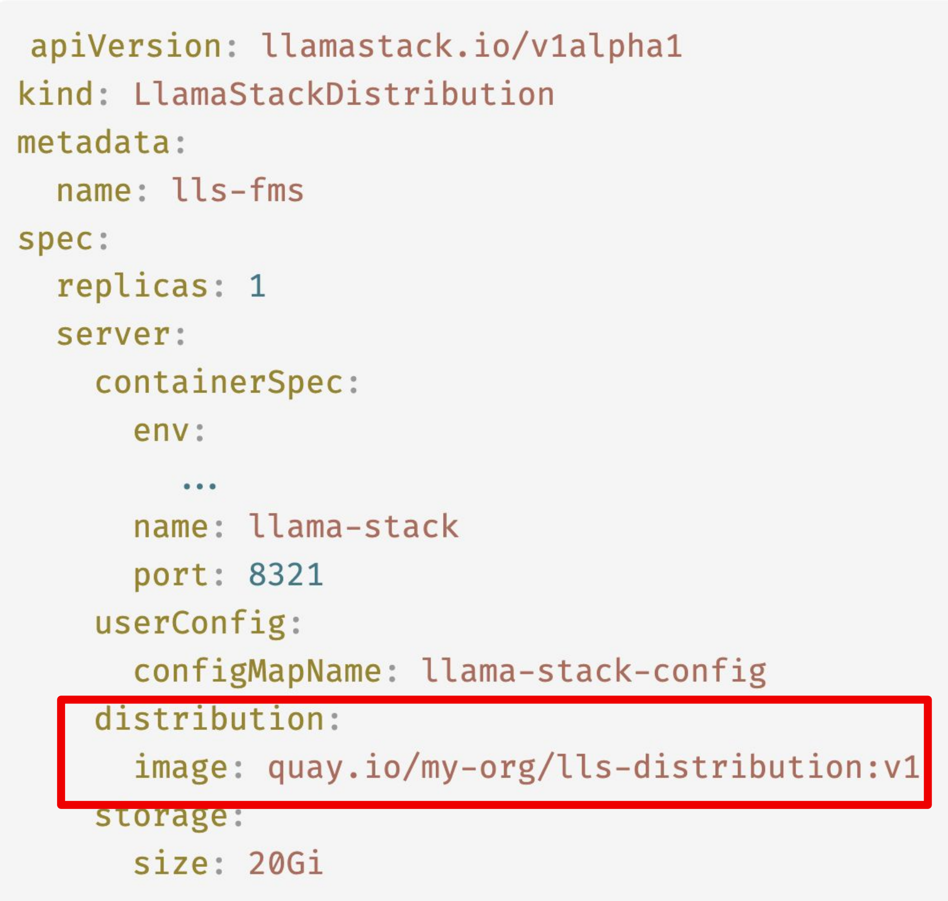
```
kind: ConfigMap
apiVersion: v1
metadata:
 name: orchestrator-config
data:
 config.yaml: |
  chat_generation:
   service:
    hostname: my-lemonade-model.svc.cluster.local
    port: 8080
  detectors:
   built_in:
    type: text_contents
    service:
      hostname: "127.0.0.1"
      port: 8080
    chunker_id: whole_doc_chunker
    default_threshold: 0.5
   hap:
    type: text_contents
    service:
     hostname: guardrails-detector-ibm-hap.svc.cluster.local
     port: 8000
    chunker_id: whole_doc_chunker
    default_threshold: 0.5
```



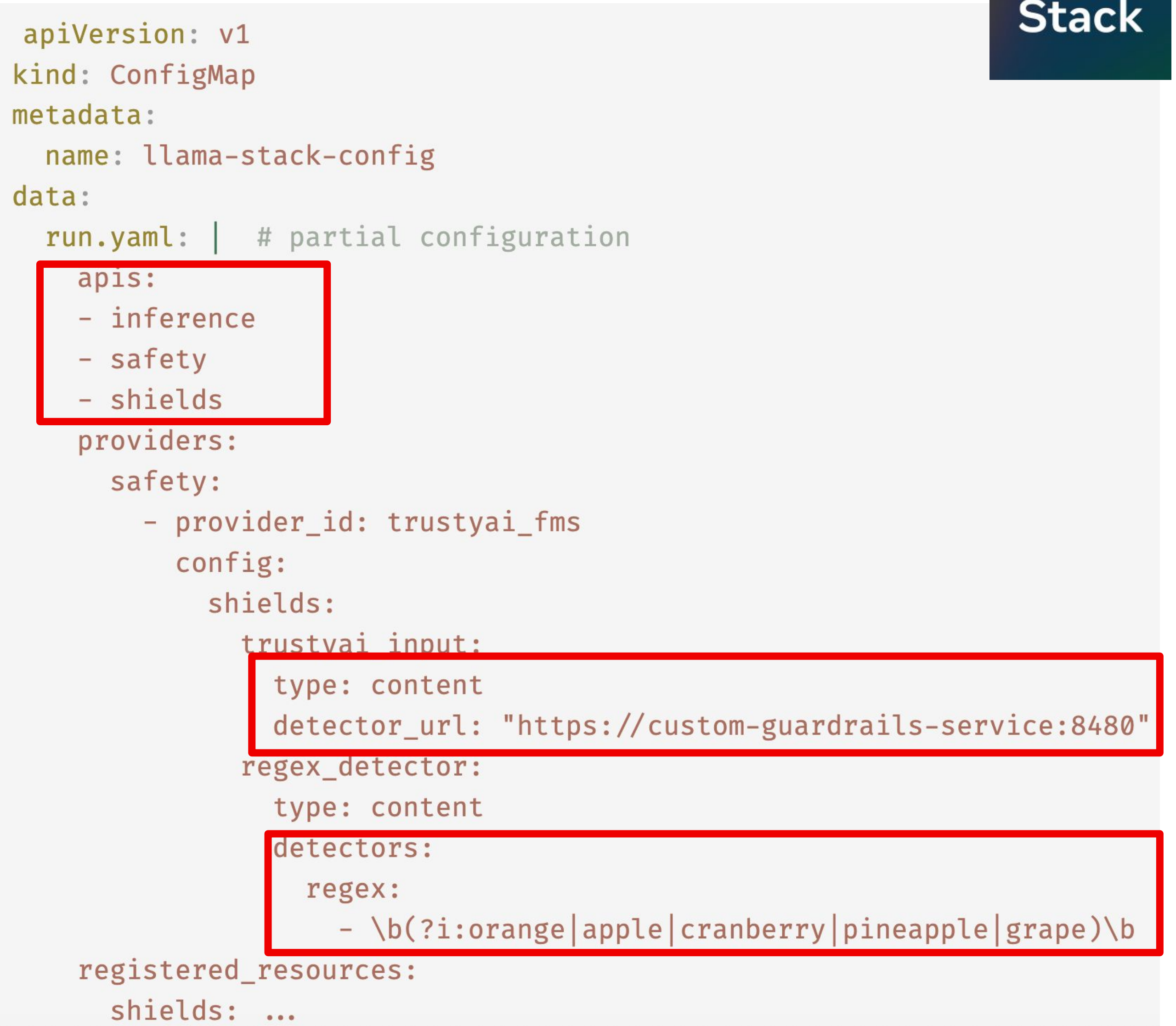
```
apiVersion: trustyai.opendatahub.io/v1alpha1
kind: GuardrailsOrchestrator
metadata:
  name: custom-guardrails
spec:
  orchestratorConfig: "orchestrator-config"
  enableBuiltInDetectors: true
  enableGuardrailsGateway: false
  disableOrchestrator: false
  replicas: 1
```

https://github.com/trustyai-explainability/trustyai-llm-demo/tree/LLSPlayground/lemonade-stand-demo

Deploy Llama Stack



```yaml
apiVersion: llamastack.io/v1alpha1
kind: LlamaStackDistribution
metadata:
  name: lls-fms
spec:
  replicas: 1
  server:
    containerSpec:
      env:
        ...
      name: llama-stack
      port: 8321
    userConfig:
      configMapName: llama-stack-config
    distribution:
      image: quay.io/my-org/lls-distribution:v1
    storage:
      size: 20Gi
```



```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: llama-stack-config
data:
  run.yaml: |  # partial configuration
    apis:
    - inference
    - safety
    - shields
    providers:
      safety:
        - provider_id: trustyai_fms
          config:
            shields:
              trustyai_input:
                type: content
                detector_url: "https://custom-guardrails-service:8480"
              regex_detector:
                type: content
                detectors:
                  regex:
                    - \b(?i:orange|apple|cranberry|pineapple|grape)\b
    registered_resources:
      shields: ...
```

https://github.com/trustyai-explainability/trustyai-llm-demo/tree/LLSPlayground/llama-stack-playground

Takeaways

Implementation Best Practices

1. Treat AI Like Infrastructure

- GitOps for AI policies and guardrails (version control)
- CI/CD for safety and guardrail testing
- Approval process
- Staged rollouts (dev -> test -> prod)
- Integrate best practices in existing development tools (Internal Developer Platform)

2. Layer Your Defense (Like K8s)

- **Network policies** to isolate workload and accepted flows
- Centralized runtime governance using unified API layer (**AI Gateway**)
  - Both for security and cost control (**Model-as-a-Service**)
- RBAC and Admission controllers
- Runtime security to mitigate risks (guardrails)

Red Hat Summit

Connect

Thank you

in linkedin.com/company/red-hat

f facebook.com/redhatinc

youtube.com/user/RedHatVideos

twitter.com/RedHat