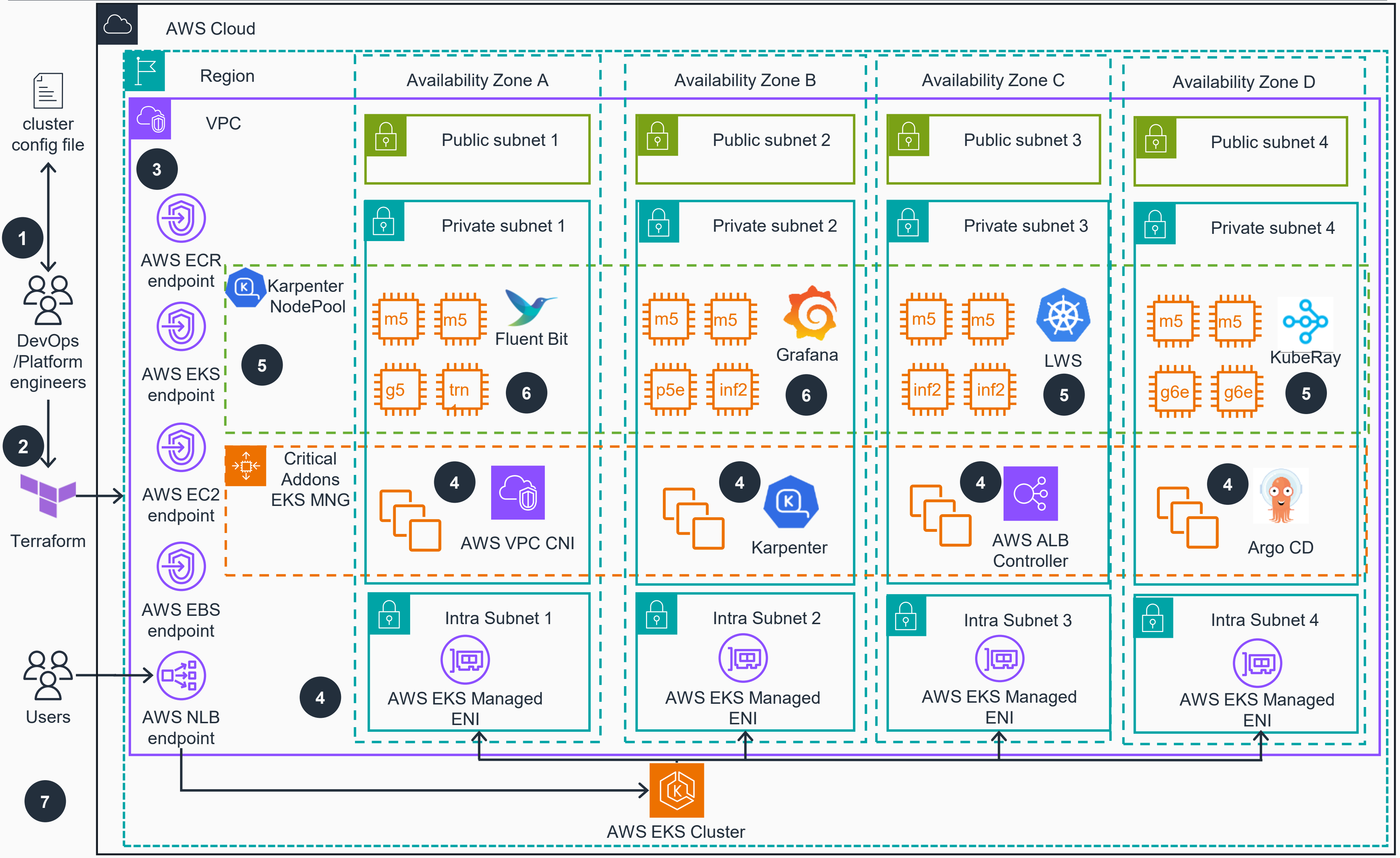
Guidance for Deploying Inference Ready Amazon EKS Clusters on AWS

Provision EKS

This architecture diagram shows how to provision an Amazon Elastic Kubernetes Service (EKS) Inference ready cluster with best practices configuration for AI workloads.



1. DevOps engineer defines a cluster Terraform variable file that contains the environment-specific configuration.
2. DevOps engineer applies the environment configuration using Terraform following the deployment process defined in the guidance to the target AWS account.
3. Provision and configure an Amazon Virtual Private Network (VPC). According to Reliability best practices, configure four Availability zones (AZs) to optimize node acquisition and high availability. Topology awareness defaults to keeping AI/ML workloads in the same AZ for performance/cost but is configurable for availability.
4. Provision Amazon Elastic Kubernetes Service (Amazon EKS) cluster with Managed Nodes Group (MNG) that run critical cluster add-ons (CoreDNS, AWS Load Balancer Controller - ALB and Karpenter) on its compute nodes. Karpenter will manage compute capacity for other add-ons, as well as deployed ML inference applications. Amazon Elastic Network Interface (ENIs) managed by **Amazon EKS** are provisioned in respective subnets
5. Deploy other important **Amazon EKS** add-ons (LWS, KubeRay etc.) based on the configurations defined in the cluster Terraform configuration file (see Step 1).
6. Deploy an observability stack including FluentBit, Prometheus and Grafana to collect metrics and logs from the environment. Deploy Service and Pod Monitors to watch for AI/ML workloads and collect metrics. Configure Grafana dashboards to automatically visualize related metrics and logs.

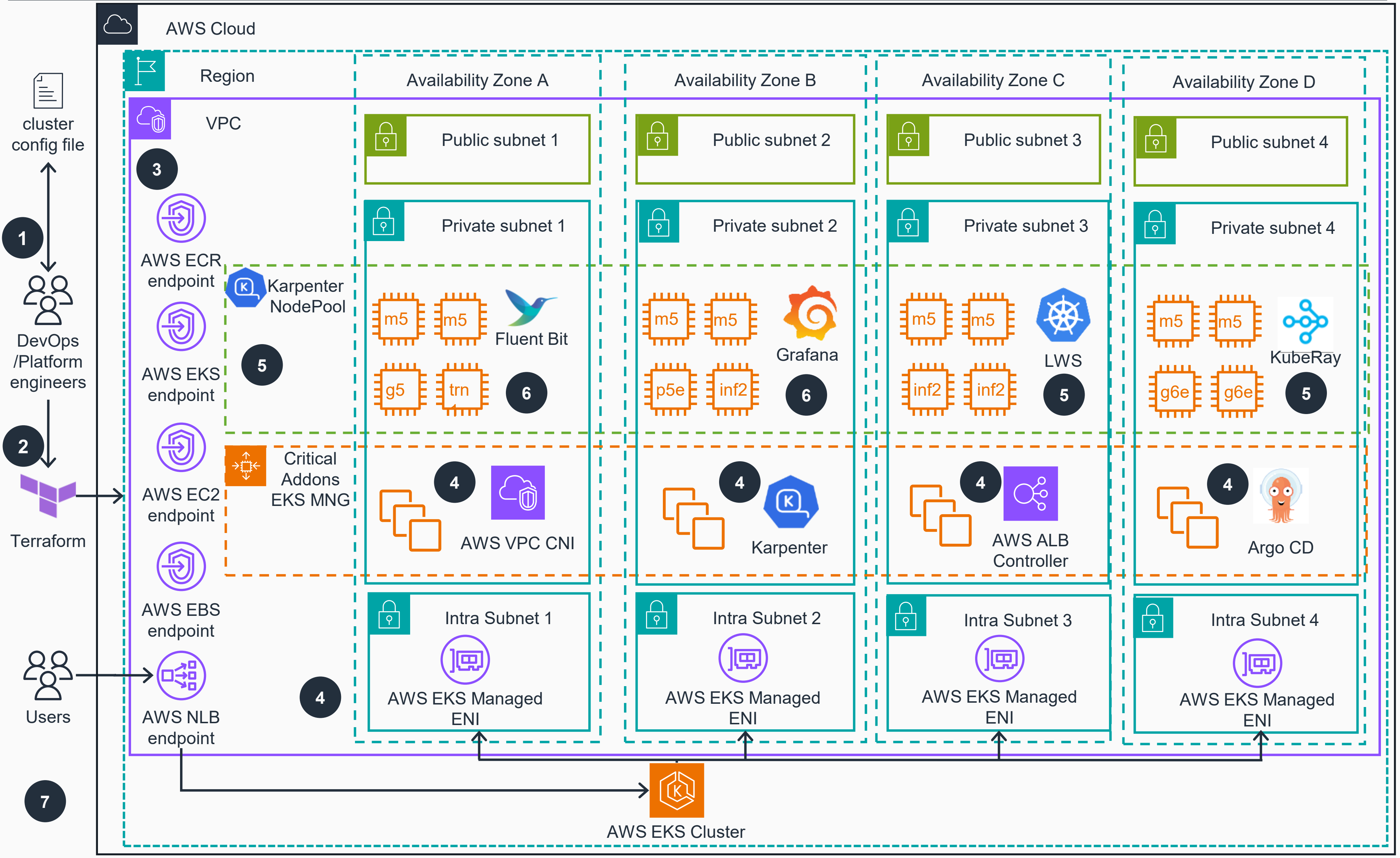
Reviewed for technical accuracy October,29 2025

AWS Reference Architecture

Guidance for Deploying Inference Ready Amazon EKS Clusters on AWS

Provision EKS

This architecture diagram shows how to provision an Amazon Elastic Kubernetes Service (EKS) Inference ready cluster with best practices configuration for AI workloads.



7 Users access Amazon EKS K8s API with AWS Network Load Balancer (NLB) endpoint and may deploy containerized AI/ML inference workloads via Kubernetes CLI using the AI on **Amazon EKS** project inference Helm charts or other repositories.

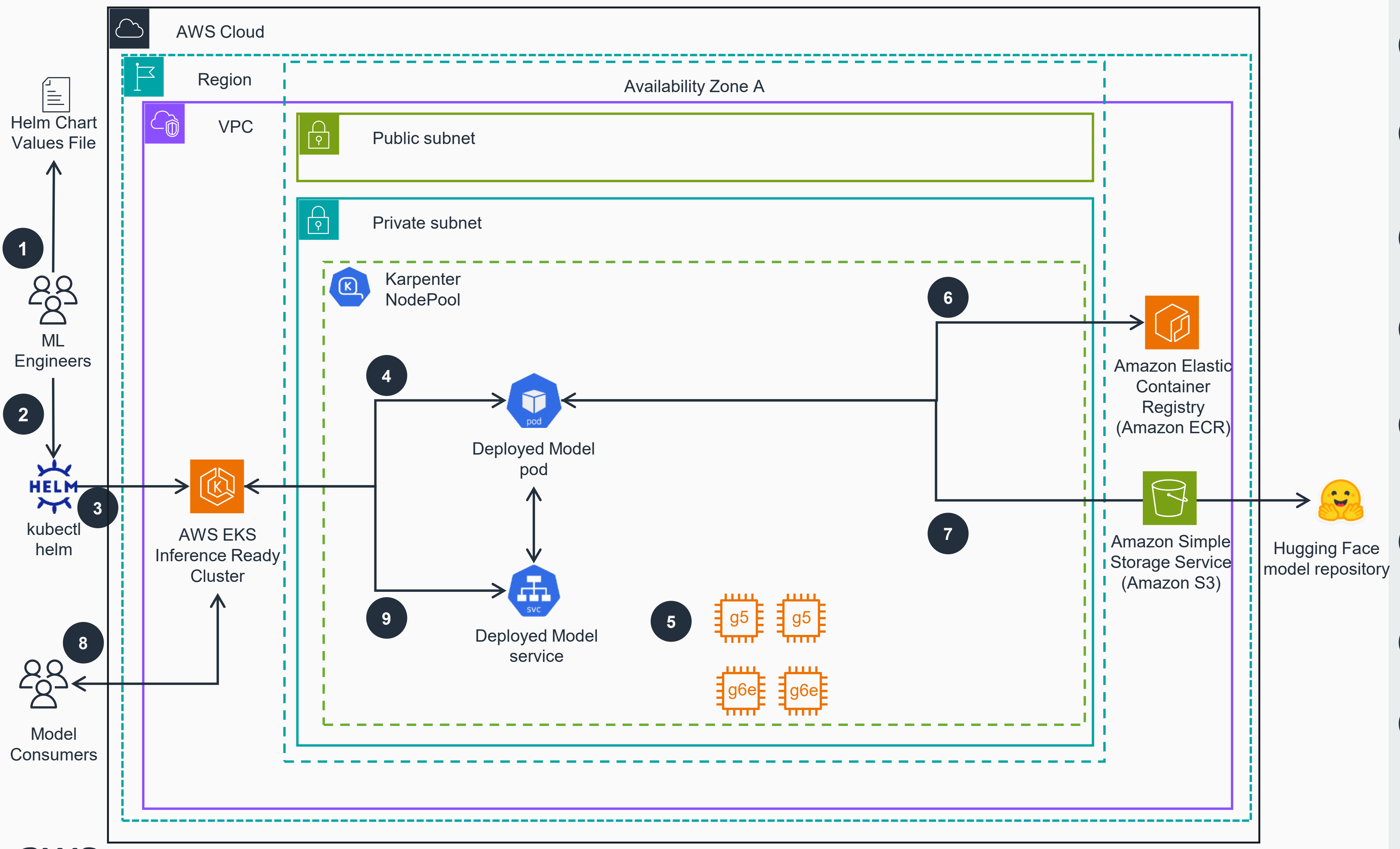
Reviewed for technical accuracy October,29 2025

AWS Reference Architecture

Guidance for Deploying Inference Ready Amazon EKS Clusters on AWS

Deploying AI Models

This architecture diagram shows highlights of deploying AI Models on the Inference-Ready Amazon EKS Cluster using Helm templates.



1. ML Engineers use a Helm chart `values` template file to set values for Helm chart for model deployment.
2. ML Engineers use `kubectl/helm` CLI tools to deploy the templated Helm chart to the Amazon EKS environment.
3. Model Helm Templates are applied to Amazon Elastic Kubernetes Service (EKS) API of Inference Ready Amazon EKS cluster to start deployment of desired model.
4. Amazon EKS API creates a Kubernetes pod and a service for the single model container deployment.
5. Karpenter auto-scaler provisions Amazon Elastic Compute Cloud (EC2) instances to fulfill the compute node resource request to schedule model pods.
6. Docker container image requested by the deployed model for the container is pulled from Amazon Elastic Container Registry (ECR).
7. The weights for the deployed model are pulled from Hugging Face repository into Amazon S3 for faster loading and loaded into the model server.
8. External Model consumer users can now port-forward the Kubernetes service port to their local machines for accessing the model.
9. User/application requests to deployed models in the Inference Ready AWS EKS cluster are routed through the Kubernetes service to the model pods, responses are returned the same way.

Reviewed for technical accuracy October,29 2025

AWS Reference Architecture