

Senior Design Project

V-etch

Virtualized AI Inference in the Cloud and at the Edge

Ali Sao

Adam Le

Kevin Ly Chhim

Dalen Smith

Tentative Faculty Advisor: Prof. QV Dang

University of California, Irvine

Department of Electrical Engineering and Computer Science

2026–2027

1 Introduction

For our Senior Design Project, we are building **V-etch**, a platform for provisioning and managing AI inference across cloud and edge computing environments. Inference means running an already-trained model to produce an output, such as generating an answer to a prompt. Not every model requires specialized AI hardware, but CPU and memory demands still affect whether a workload can run efficiently.

We want to study how inference resources can be allocated, isolated, and measured. We will first deploy real models inside cloud virtual machines, then extend the same resource-aware approach to an edge device with tighter hardware limits. Our goal is to manage *where* complete inference requests run, not to split one model computation across devices.

2 Inference and Resource Constraints

A model needs enough memory to load its weights and enough compute to answer requests within a useful time. The same model can behave differently when given two versus four CPUs, or when other workloads compete for the same host. We will measure these tradeoffs instead of assuming that general-purpose cloud CPUs are always cheaper or faster than dedicated accelerators.

Each cloud inference environment will run in a KubeVirt virtual machine with an assigned CPU and memory budget. Separate VMs give us independent guest environments, failure boundaries, and repeatable allocations, even when they share one physical server. Containers already provide resource limits, so we will also consider VM startup and memory overhead and use a container baseline if time permits.

This resource-constrained design naturally extends to edge inference. Edge devices have more limited memory and compute, and may require a compatible smaller or quantized model. Comparing the same workload on cloud and edge will help us decide when each backend is suitable.

3 Software Infrastructure

We will develop locally on a Linux machine with KVM support, then deploy the cloud system using **AWS EC2 and EKS**. Kubernetes manages the cluster, while **KubeVirt** provisions Linux VMs on compatible worker nodes. A Go-based V-etch operator coordinates VM creation and resource requests. Inside a VM, a guest agent manages an inference worker, tracks readiness, and reports capacity and metrics.

Our planned cloud path is: **EC2 worker node** → **Kubernetes/EKS** → **KubeVirt VM** → **guest agent** → **inference runtime**. The edge device, a Qualcomm-based Arduino VENTUNO Q, will run its own compatible inference runtime rather than a KubeVirt VM. Both backends will share a request and response format so we can route whole requests between them.

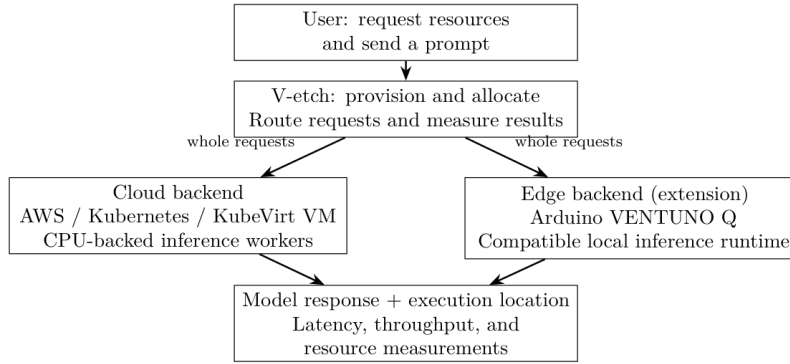


Figure 1: V-etch cloud and edge inference workflow.

4 User Interface and Workflow

Users will interact with V-etch through a Go command-line interface. For example, a user can request an open-source model with four virtual CPUs and 8 GB of memory. V-etch validates the request, creates a VM, starts the runtime, and provides an endpoint to submit prompts. The response includes the model output, execution location, latency, and resource measurements.

An **automatic mode** will route each complete request using available capacity and performance measurements. V-etch builds on existing inference runtimes rather than training models or emulating GPUs.

5 Major Tasks and Team Responsibilities

The following is our proposed division of work; everyone will help with integration and testing.

Member	Primary responsibility and deliverable
Ali Sao	Kubernetes/KubeVirt, local and AWS VM provisioning, resource allocation.
Adam Le	Inference runtime, model compatibility, benchmarks and performance metrics.
Kevin Ly Chhim	VENTUNO Q edge setup, edge inference and hardware evaluation.
Dalen Smith	Go CLI, shared inference interface, user workflow and integration tests.

6 Milestones and Deliverables

Stage	What we can demonstrate
Local foundation	Boot and manage a Linux KubeVirt VM; enforce CPU/memory allocations.
First inference	Deploy a model inside a VM and return a response to a prompt.
AWS deployment	Repeat VM provisioning and inference on compatible EC2/EKS infrastructure.
Resource testing	Measure latency, throughput, memory use, startup time and isolation tradeoffs.
Edge backend	Run a compatible model on VENTUNO Q and compare performance.
Final demo	Use automatic mode to route full requests to cloud or edge; show results and metrics.

7 Evaluation and Final Demonstration

In the final demo, a user will request a model and compute budget, observe VM provisioning, and submit inference requests. V-etch will report outputs, selected backend, latency, throughput, and CPU/memory measurements. We will vary available resources or worker readiness to demonstrate that allocation and request placement respond to constraints. The target is **cloud plus edge routing of complete requests**; split-model inference is outside our scope.

8 Scope, Feedback, and Advisor Status

Feedback raised two main concerns: project cost and an overly broad scope. In response, we narrowed the first deliverable to CPU-backed inference in local and AWS-hosted VMs, followed by edge execution and whole-request routing. We will use small open-source models and mostly local development to limit AWS spending. We will not build a custom accelerator, inference kernel, or model-training pipeline.

Advisor: Prof. QV Dang (tentative). **Faculty contacted:** Prof. Sheu (recommended a better-aligned advisor but offered to help if needed), Prof. Shoukry (unable to advise), and Prof. Al Faruque (not responded). We will continue refining resource limits, model compatibility, and benchmarking methods with faculty or TA feedback.