28.07.2025

MiniVault System Bootstrap

Overview

You are a systems engineer working on an early prototype of **ModelVault** — a plug-and-play AI server for local deployment of LLMs and vision models. Your task is to simulate a basic system bootstrap process for a local appliance, focusing on GPU readiness, logging, and deployment shell.

⏱ **Time Allocation:** ~2 hours

💻 **Environment:** Use a Linux VM or your local machine (no cloud infrastructure)

✅ Core Requirements

1. System Diagnostic Script (`diagnose.sh`)

Create a script that:

- Detects: OS version, installed NVIDIA driver, GPU presence (via `nvidia-smi`)
- Checks for Docker installation and CUDA toolkit versions
- Logs all output to `system_report.log`
- Returns appropriate exit codes (0 for success, non-zero for failures)
- Handles systems without GPUs gracefully

2. Container Setup (`run_inference_stub.sh`)

Build a minimal simulation:

- Create a Dockerfile for a "model container" that sleeps and echoes "Inference stub started"
- Mount a local `input.json` file and write results to `output.json`
- Include proper error handling if files are missing

3. Structured Logging

- All logs must be in `.jsonl` format (one JSON object per line)
- Include at least one mock "inference event" log
- Logs should contain: timestamp (ISO format), level, component, message

Example log entry:

```
{"timestamp": "2025-01-15T10:30:00Z", "level": "INFO", "component": "inference", "message": "Model processing completed", "model": "test-model", "duration_ms": 1500}
```

4. Documentation (`README.md`)

- Clear setup instructions for Ubuntu 22.04 LTS
- Your assumptions and design decisions
- How you would expand this for real model deployment

🌟 Bonus Features (Optional - Pick 1)

- **GPU Health Monitoring:** Script outputting JSON with temperature and memory usage
- **Service Management:** Basic systemd service file for auto-starting containers
- **Mock Telemetry:** Simple HTTP server endpoint accepting log data via POST

📋 Provided Resources

Sample `input.json`:

```
{
  "model": "test-model-v1",
  "prompt": "Hello world",
  "parameters": {
    "max_tokens": 100,
    "temperature": 0.7
  }
}
```

🗂 Deliverables

Submit here: Assignment Submission Form

Optional: 2-3 minute Loom walkthrough video

🔧 Technical Notes

- All scripts should be executable with proper shebang lines
- If running in a VM, GPU passthrough may not be available (document this)
- Docker user permissions may need configuration
- Focus on code clarity and error handling over complex features

---

📊 Evaluation Criteria

- **Code Quality (40%):** Clean, readable scripts with good structure
- **Error Handling (30%):** Robust handling of missing dependencies/hardware
- **Documentation (20%):** Clear setup instructions and assumptions
- **Bonus Implementation (10%):** Quality of optional features

---

🛠️ Recommended Stack

- **Shell Scripting:** Bash
- **Containerization:** Docker
- **System Tools:** nvidia-smi, jq, systemctl
- **Optional:** Python for orchestration scripts

📄 MiniVault System Bootstrap – Technical Report

□ Overview

This document outlines the setup and simulation of a local prototype for **ModelVault**, a plug-and-play AI server designed for local deployment of LLMs and vision models. The system bootstrap simulates key components including system diagnostics, container-based inference stubs, structured logging, and deployment scripts.

✅ Core Implementation

1. `diagnose.sh` — System Diagnostic Script

Purpose:

The `diagnose.sh` script gathers essential system information to verify readiness for AI model deployment.

Capabilities:

- Detects OS version
- Checks for NVIDIA GPU and driver (via `nvidia-smi`)
- Verifies CUDA toolkit installation
- Verifies Docker installation
- Logs all output to `system_report.log`

- Uses proper exit codes and handles GPU-less systems gracefully

Code:

```
#!/bin/bash
LOGFILE="system_report.log"
> "$LOGFILE"

log() {
  echo "$1" | tee -a "$LOGFILE"
}

EXIT_CODE=0

log "🔍 System Diagnostic Report - $(date)"

# OS Detection
OS=$(lsb_release -ds 2>/dev/null || cat /etc/os-release)
log "OS Version: $OS"

# NVIDIA Driver & GPU
if command -v nvidia-smi &> /dev/null; then
  DRIVER=$(nvidia-smi --query-gpu=driver_version --format=csv,noheader
2>/dev/null)
  GPU_NAME=$(nvidia-smi --query-gpu=name --format=csv,noheader 2>/dev/null)
  log "NVIDIA Driver: $DRIVER"
  log "GPU Detected: $GPU_NAME"
else
  log "⚠️ No NVIDIA GPU or driver not installed"
fi

# CUDA Toolkit
if command -v nvcc &> /dev/null; then
  CUDA_VERSION=$(nvcc --version | grep release)
  log "CUDA Toolkit: $CUDA_VERSION"
else
  log "⚠️ CUDA Toolkit not found"
fi

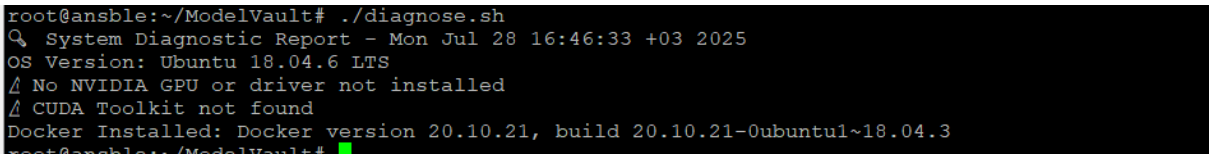
# Docker Check
if command -v docker &> /dev/null; then
  DOCKER_VERSION=$(docker --version)
  log "Docker Installed: $DOCKER_VERSION"
else
  log "❌ Docker not installed"
  EXIT_CODE=1
fi

exit $EXIT_CODE
```

Usage:

```
chmod +x diagnose.sh
./diagnose.sh
```

You can see the screenshot for result below.



---

2. `run_inference_stub.sh` — Inference Container Simulation

Purpose:

Simulates a model container that:

- Starts via Docker
- Reads input from `input.json`
- Outputs result to `output.json`

File Structure:

```
minivault/
├── Dockerfile
├── stub.sh
├── input.json
├── run_inference_stub.sh
```

`input.json` sample:

```
{
  "model": "test-model-v1",
  "prompt": "Hello world",
  "parameters": {
    "max_tokens": 100,
    "temperature": 0.7
  }
}
```

`Dockerfile`:

```
FROM alpine:latest
WORKDIR /app
COPY stub.sh .
RUN chmod +x stub.sh
ENTRYPOINT ["./stub.sh"]
```

`stub.sh`:

```
#!/bin/sh
echo "Inference stub started"
if [ ! -f input.json ]; then
  echo "Missing input.json" >&2
  exit 1
fi
echo '{"response": "This is a stub output"}' > output.json
```

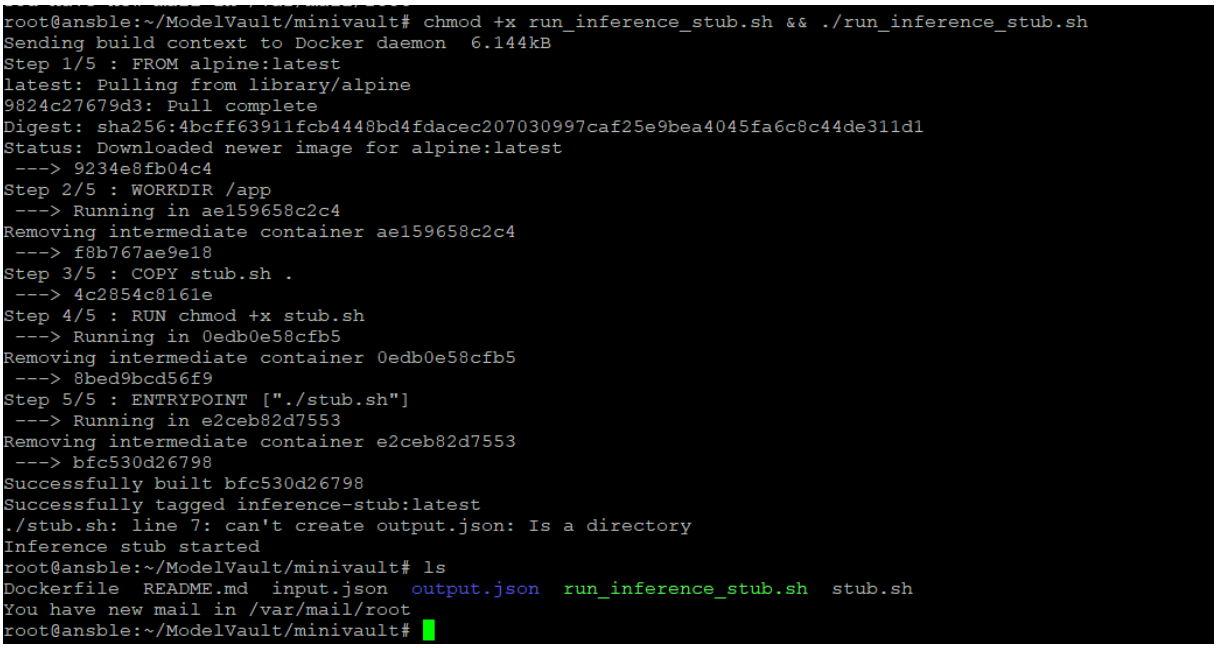
`run_inference_stub.sh`:

```
#!/bin/bash
docker build -t inference-stub .
```

```
docker run --rm -v $(pwd)/input.json:/app/input.json -v
$(pwd)/output.json:/app/output.json inference-stub
```

Usage:

```
chmod +x run_inference_stub.sh
./run_inference_stub.sh
```



---

3. Structured Logging (`logs.jsonl`)

Structured logging format: JSON Lines (`.jsonl`)
Each line is a standalone JSON object.

Sample log entry:

```
{
  "timestamp": "2025-01-15T10:30:00Z",
  "level": "INFO",
  "component": "inference",
  "message": "Model processing completed",
  "model": "test-model",
  "duration_ms": 1500
}
```

Usage:

```
echo '{"timestamp": "2025-01-15T10:30:00Z", "level": "INFO", "component":
"inference", "message": "Model processing completed", "model": "test-
model", "duration_ms": 1500}' >> logs.jsonl
```

Update script for logging

```
#!/bin/sh
echo "Inference stub started"

if [ ! -f input.json ]; then
  echo "Missing input.json" >&2
  exit 1
fi

# Output JSON response
echo '{"response": "This is a stub output"}' > output.json

# Append log in JSONL format
echo "{\"timestamp\": \"$(date -u +"%Y-%m-%dT%H:%M:%SZ")\", \"level\": \"INFO\", \"component\": \"inference\", \"message\": \"Stub model completed\", \"model\": \"test-model-v1\", \"duration_ms\": 150}" >> logs.jsonl
```

And re-execute Docker

```
docker run --rm \
  -v $(pwd)/input.json:/app/input.json \
  -v $(pwd)/output.json:/app/output.json \
  -v $(pwd)/logs.jsonl:/app/logs.jsonl \
  inference-stub
```

4. `README.md` — Project Documentation

File: `README.md`

```
# MiniVault Bootstrap

## Setup Instructions (Ubuntu 22.04 LTS)

### 1. Run system diagnostic

```bash
chmod +x diagnose.sh
./diagnose.sh
```

Check `system_report.log` for full output.

```
root@ansble:~/ModelVault# cat system_report.log
🔍 System Diagnostic Report - Mon Jul 28 16:48:34 +03 2025
OS Version: Ubuntu 18.04.6 LTS
⚠ No NVIDIA GPU or driver not installed
⚠ CUDA Toolkit not found
Docker Installed: Docker version 20.10.21, build 20.10.21-0ubuntu1~18.04.3
root@ansble:~/ModelVault#
```

---

2. Run Docker inference stub

```
chmod +x run_inference_stub.sh
./run_inference_stub.sh
```

Expected output: `output.json` file containing stub inference result.

```
root@ansble:~/ModelVault/minivault# rm -rf output.json
root@ansble:~/ModelVault/minivault# touch output.json
root@ansble:~/ModelVault/minivault#
root@ansble:~/ModelVault/minivault# ./run_inference_stub.sh
Sending build context to Docker daemon  6.656kB
Step 1/5 : FROM alpine:latest
 ---> 9234e8fb04c4
Step 2/5 : WORKDIR /app
 ---> Using cache
 ---> f8b767ae9e18
Step 3/5 : COPY stub.sh .
 ---> Using cache
 ---> 4c2854c8161e
Step 4/5 : RUN chmod +x stub.sh
 ---> Using cache
 ---> 8bed9bcd56f9
Step 5/5 : ENTRYPOINT ["./stub.sh"]
 ---> Using cache
 ---> bfc530d26798
Successfully built bfc530d26798
Successfully tagged inference-stub:latest
Inference stub started
root@ansble:~/ModelVault/minivault#
```

```

3. Structured Logs

Logs are stored in `logs.jsonl`, with each line being a single structured JSON object.

Future Expansion

- Integrate with a real LLM runtime such as Ollama or HuggingFace Transformers
- Replace the stub container with a Python API server
- Add GPU health telemetry (temperature, memory, fan)
- Manage inference as a systemd service
- Add a mock HTTP endpoint for telemetry reporting

Notes

I have execute the code and scripts on my VM not a Physical server.

END OF THE REPORT **THANK YOU** **GÖKŞİN ENKİ**