project name is - AI-Powered Minimalist Container Runtime

Project Description: AI-Powered Minimalist Container Runtime

This project is a lightweight container runtime built in Python, designed to provide process isolation, resource management, and networking similar to Docker.

It integrates AI-powered anomaly detection, auto-healing, and monitoring to enhance container performance and security.

---

Key Features:

1.Minimalist Container Runtime: Uses Linux namespaces & cgroups for process isolation and resource control.

2.AI-Based Anomaly Detection: Monitors system metrics and detects unusual behavior in containers.

3.Auto-Healing Containers: Automatically restarts failed or unhealthy containers.

4.Live Monitoring Dashboard: Web-based UI (HTML, TailwindCSS, Chart.js) for real-time CPU, memory, and container status tracking.

5.Networking & Security: Supports container-to-container communication with security hardening.

6.AWS & Kubernetes Integration: Can be deployed on AWS with Kubernetes for scalability.

---

System Architecture:

1. Components

Container Runtime

Uses Linux namespaces & cgroups for process isolation.

Provides basic start, stop, and restart functionalities.

2.AI Anomaly Detection

Monitors CPU, memory, and network usage.

Detects unusual behavior using machine learning.

3.Auto-Healing & Load Balancing

Automatically restarts failed containers.

Distributes workload across multiple instances.

4.Networking & Security

Container-to-container communication support.

Security hardening for restricted access.

5.Monitoring Dashboard

Displays real-time CPU & memory usage.

Shows running containers & alerts.

---

Tech Stack:

1.Backend: Python (Flask, psutil, AI models for anomaly detection)

2.Frontend: HTML, Tailwind CSS, JavaScript, Chart.js

3.AI Integration: Machine learning for detecting container failures

4.Deployment: AWS EC2, Kubernetes

---

Implementation:

1 Backend (Python & Flask)

Manages container lifecycle (start, stop, restart).

Monitors system resources.

AI model detects anomalies.

API endpoints provide container & monitoring data.

2 Frontend (Web UI)

Fetches real-time data via Flask API.

Displays CPU, Memory, Container List, and Alerts.

3 AI-Based Anomaly Detection

Uses machine learning to identify unusual container behavior.

Triggers alerts for abnormal resource usage.

4 Auto-Healing & Load Balancing

Automatically restarts unhealthy containers.

Balances workloads across multiple instances.

5 AWS & Kubernetes Integration

Deploys runtime on AWS EC2 with auto-scaling.

Uses Kubernetes for orchestration and high availability.

6 Security Measures

Container Isolation via namespaces.

Resource Limiting via cgroups.

Firewall & Access Control policies.

7. Future Enhancements

Container Image Support for portable deployments.

More Advanced AI Models for real-time threat detection.

Full Kubernetes Compatibility for enterprise scalability.

---

Conclusion:

This project provides an AI-driven, lightweight, and secure container runtime with built-in auto-healing and anomaly detection, making it ideal for cloud and edge computing environments.