

WSL2 Üzerinde Minikube GPU Entegrasyonu ve HAMi Middleware Kurulum Denemeleri Raporu

Ysf

19 Ocak 2026

İçindekiler

1	Giriş ve Amaç	2
2	Sistem Özellikleri ve Ön Hazırlık	2
3	Karşılaşılan Problemler ve Çözüm Adımları	2
3.1	1. Docker Soket Erişim Hatası (Permission Denied)	2
3.2	2. HAMi Kurulumu ve "Segmentation Fault" Hatası	2
3.3	3. Docker Credential Helper Hatası	3
3.4	4. HAMi'nin Kaldırılması ve GPU Kapasite Sorunu	3
4	Nihai Çözüm ve Doğrulama	3
4.1	Uygulanan Kesin Çözüm Adımları	3
4.2	Sonuç Testi	4
5	Sonuç ve Öneriler	4

1 Giriş ve Amaç

Bu rapor, Windows Subsystem for Linux 2 (WSL2) altyapısı üzerinde çalışan Ubuntu 24.04 dağıtımında, Kubernetes ortamı (Minikube) oluşturulması ve GPU sanallaştırma aracı olan **HAMi (Heterogeneous AI Computing Middleware)** entegrasyonu sürecini belgelemektedir. Çalışmanın temel amacı, tek bir fiziksel GPU'nun (NVIDIA GTX 1650) sanallaştırılarak Kubernetes pod'ları arasında paylaşılmasıdır.

2 Sistem Özellikleri ve Ön Hazırlık

- **İşletim Sistemi:** Windows 11 (Host), Ubuntu 24.04 (WSL2 Guest)
- **Konteyner Motoru:** Docker Desktop (WSL2 Backend entegrasyonu aktif)
- **Kubernetes Dağıtımı:** Minikube v1.37.0
- **Donanım:** NVIDIA GeForce GTX 1650
- **Hedef Yazılım:** Project-HAMi (Eski adıyla vGPU Scheduler)

3 Karşılaşılan Problemler ve Çözüm Adımları

3.1 1. Docker Soket Erişim Hatası (Permission Denied)

Minikube, Docker sürücüsü ile başlatılmak istendiğinde `permission denied` hatası alındı.

```
1 Exiting due to PROVIDER_DOCKER_NEWGRP: "docker version ..." exit status
  1: permission denied while trying to connect to the docker API at
  unix:///var/run/docker.sock
```

Listing 1: Hata Çıktısı

Çözüm: Kullanıcı `docker` grubuna eklendi ve oturum yenilendi.

```
1 sudo usermod -aG docker $USER && newgrp docker
```

3.2 2. HAMi Kurulumu ve "Segmentation Fault" Hatası

Minikube, GPU desteği ile (`-gpu=all`) başlatıldı ve HAMi Helm chart'ları kullanılarak `kube-system` namespace'ine kuruldu. Node `gpu=on` etiketi ile işaretlendi.

Ancak, test pod'u oluşturulduğunda ve `nvidia-smi` komutu çalıştırıldığında konteyner çökmesi yaşandı.

```
1 kubectl exec -it gpu-pod -- nvidia-smi
2 #      kt      :
3 command terminated with exit code 139
```

Listing 2: Kritik Hata

Analiz: Exit code 139, bellek erişim hatası (Segmentation Fault) anlamına gelmektedir. HAMi'nin GPU belleğini bölmek için kullandığı `libvgpu.so` kütüphanesinin, WSL2'nin paravirtualized (yarı sanallaştırılmış) NVIDIA sürücülerini ile uyumsuz olduğu tespit edildi. HAMi, donanım seviyesinde bellek yönetimi yapmaya çalışırken WSL2 kernel'i ile çakışmaktaydı.

3.3 3. Docker Credential Helper Hatası

Sorunun Minikube'den mi yoksa altyapıdan mı kaynaklandığını anlamak için doğrudan Docker üzerinde test yapılmak istendi. Ancak imaj çekilirken aşağıdaki hata alındı:

```
1 docker: error getting credentials - err: fork/exec /usr/bin/docker-credential-desktop.exe: exec format error
```

Çözüm: WSL2 Linux ortamının, Windows executable (.exe) dosyalarını credential helper olarak kullanamaması nedeniyle Docker konfigürasyon dosyası geçici olarak devre dışı bırakıldı.

```
1 mv ~/.docker/config.json ~/.docker/config.json.bak
```

Bu işlem sonrası `docker run -gpus all ...` komutu başarılı oldu ve altyapının sağlam olduğu doğrulandı.

3.4 4. HAMi'nin Kaldırılması ve GPU Kapasite Sorunu

HAMi'nin WSL2 ile uyumsuz olduğu anlaşıldıktan sonra, stabilite için standart NVIDIA Device Plugin yapısına dönülmesine karar verildi. HAMi kaldırıldı, ancak Minikube node'u üzerindeki GPU kapasitesi 0 olarak görünmeye başladı.

```
1 kubectl describe node minikube | grep "nvidia.com/gpu"
2 #      kt      :
3 nvidia.com/gpu: 0
```

Sebebi: HAMi kaldırıldığında, yerine geçmesi gereken standart plugin otomatik olarak devreye girmedi. Manuel olarak kurulan eklentiler ise Minikube ile tam uyum sağlamadı.

4 Nihai Çözüm ve Doğrulama

Sistemi kararlı bir hale getirmek için "Temiz Kurulum" (Clean Slate) yaklaşımı uygulandı.

4.1 Uygulanan Kesin Çözüm Adımları

1. **Ortam Temizliği:** Mevcut Minikube kümesi tamamen silindi.

```
1 minikube delete
2
```

2. **Yeniden Başlatma:** Minikube, GPU flag'i ile sıfırdan başlatıldı.

```
1 minikube start --driver=docker --gpus=all
2
```

3. **Eklenti Aktivasyonu:** NVIDIA device plugin, Minikube addon sistemi üzerinden aktif edildi.

```
1 minikube addons enable nvidia-device-plugin
2
```

4. **Kapasite Kontrolü:** GPU kaynağının başarıyla tanımlandığı görüldü.

```
1 kubectl describe node minikube | grep "nvidia.com/gpu"
2 #      kt      : nvidia.com/gpu: 1
3
```

4.2 Sonuç Testi

NVIDIA CUDA imajı içeren bir test pod'u (`gpu-final-test`) oluşturuldu ve içerisine girilerek GPU durumu sorgulandı.

```
1 +-----+
2 | NVIDIA-SMI 590.48.01                Driver Version: 591.59          CUDA Version: 13.1
3 |-----+-----+-----+
4 | GPU   Name                               Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC
5 | Fan   Temp   Perf           Pwr:Usage/Cap |      Memory-Usage | GPU-Util  Compute M.
6 |=====+=====+=====+
7 |    0   NVIDIA GeForce GTX 1650            On      | 00000000:01:00.0 Off  |
8 +-----+-----+-----+
```

Şekil 1: Başarılı nvidia-smi çıktısı

5 Sonuç ve Öneriler

Yapılan testler sonucunda, **HAMi (Heterogeneous AI Computing Middleware)** aracının bellek sanallaştırma özelliklerinin şu an için WSL2 GPU passthrough sürücülerine uyumsuz olduğu ve sistem çökmelerine (Segmentation Fault) neden olduğu belirlenmiştir.

WSL2 üzerinde Kubernetes ile GPU tabanlı yapay zeka geliştirmeleri (Bitirme Projesi kapsamında) yapmak için en kararlı yöntemin, **Standart NVIDIA Device Plugin** kullanımı olduğu doğrulanmıştır. Bu yapılandırma ile Docker konteynerleri GPU'ya doğrudan ve sorunsuz erişebilmektedir.