🧪 Hands-on Lab: Configure MIG on A100

This lab will guide you through configuring **MIG (Multi-Instance GPU)** on an **NVIDIA A100 GPU**, exposing MIG instances to **Kubernetes**, and verifying correct scheduling and usage.

🛠️ Lab Objectives

By the end of this lab, you will:

- Enable MIG mode on an A100 GPU
- Create and view MIG instances
- Install NVIDIA device plugin with MIG support
- Expose MIG devices to Kubernetes
- Deploy a pod to a specific MIG profile
- Monitor and verify GPU resource assignment

⚙️ Lab Requirements

- A system with an **NVIDIA A100 GPU**
- Ubuntu 20.04 or later (bare metal or VM)
- Docker & Kubernetes (v1.20+)
- `nvidia-container-toolkit`, `nvidia-docker2`
- NVIDIA GPU Driver (465+)

- `nvidia-smi`, `kubectl`, `helm`

✅ Step 1: Enable MIG Mode on A100 GPU

```
sudo nvidia-smi -i 0 -mig 1
```

- This command enables MIG mode on the first GPU (GPU 0).
- If you have multiple A100s, repeat with `-i 1`, `-i 2`, etc.

🧠 *Note: A system reboot may be required if MIG was not previously enabled.*

✅ Step 2: Create MIG Instances (e.g., 3x 2g.10gb)

```
sudo nvidia-smi mig -cgi 0,1,2 -gi 1 -i 0
```

- `-cgi` specifies the Compute Instance profile ID (e.g., 0 = 2g.10gb)
- `-gi` specifies the number of instances to create
- `-i 0` targets GPU 0

📋 **Check available profiles:**

```
nvidia-smi mig -lgip
```

📋 **Check created instances:**

```
nvidia-smi -L
```

✅ Step 3: Verify MIG Devices

You should see output like:

```
GPU 0: MIG 2g.10gb Device 0 (UUID: MIG-GPU-...)
GPU 0: MIG 2g.10gb Device 1 (UUID: MIG-GPU-...)
GPU 0: MIG 2g.10gb Device 2 (UUID: MIG-GPU-...)
```

✅ Step 4: Deploy NVIDIA Device Plugin (MIG Mode)

Install with Helm (recommended):

```
helm repo add nvdp https://nvidia.github.io/k8s-device-plugin
helm repo update

helm install nvidia-device-plugin nvdp/nvidia-device-plugin \
  --set migStrategy=single \
  --set runtimeClassName=nvidia
```

Or use the YAML manifest:

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/ma
```

Ensure `migStrategy=single` is set to expose each MIG instance as a separate device.

✅ Step 5: Verify GPU Visibility in Kubernetes

Run the following to confirm:

```
kubectl get nodes "-o=custom-columns=NODE:.metadata.name,GPU:.status.allocatab
```

You should see fractional GPUs like:

```
NODE              GPU
gpu-node-1        3
```

✅ Step 6: Deploy a Pod that Uses One MIG Instance

Here's a sample pod definition (`mig-pod.yaml`):

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: mig-test
spec:
  containers:
  - name: cuda-container
    image: nvcr.io/nvidia/k8s/cuda-sample:vectoradd-cuda11.2
    resources:
      limits:
        nvidia.com/gpu: 1
    command: ["/bin/bash", "-c", "--"]
    args: ["sleep 3600"]
```

Apply it:

```
kubectl apply -f mig-pod.yaml
```

✅ Step 7: Verify MIG Pod Placement

Use:

```
kubectl describe pod mig-test
```

Then SSH into the node and run:

```
nvidia-smi
```

You should see the container mapped to one MIG instance.

✅ Step 8: Monitor MIG Usage with DCGM or Prometheus (Optional)

Install DCGM exporter:

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/dcgm-exporter/main/d
```

Integrate with Prometheus + Grafana dashboards to view:

- GPU utilization per MIG device
- Memory use
- Temperature
- Errors/failures

🧠 Bonus Tips

- MIG instances are not persistent across reboots by default. Use systemd or custom scripts to automate re-creation.
- Combine with **node selectors, labels, and taints** to schedule workloads more effectively in multi-user clusters.

🏁 Lab Complete – You've Configured MIG for Kubernetes!

You've now:

- Enabled and created MIG instances on A100
- Exposed them to Kubernetes via the device plugin
- Deployed and scheduled GPU-aware pods
- Verified correct GPU allocation and usage