MIG (Multi-Instance GPU)

MIG allows you to partition a single GPU into multiple independent GPU instances with dedicated memory, compute, and scheduling. These slices act like isolated GPUs to the OS and containers.

E.g.:

> An A100 40GB can be split into 7 instances of 5GB GPU compute slices.

Step 1: Enable MIG Mode on GPU

```
#!/bin/bash
echo "Enabling MIG mode..."
sudo nvidia-smi -i 0 -mig 1
echo "MIG mode enabled on GPU 0. Please reboot the system now."
```

After applying this script reboot the system

Step 2: Create MIG Instances :

Each instance is defined by a **Compute GPU Instance (CGI) profile.**

Example Profiles for A100:

```
| Profile ID | Name     | Description         |
| ---------- | -------- | ------------------- |
| 0          | 1g.5gb   | Smallest            |
| 9          | 3g.20gb  | Medium              |
| 19         | 7g.40gb  | Full GPU (no MIG)   |
```

Now lets create 3 x 1g.10gb slices:

```
#!/bin/bash
echo "Creating MIG instances..."
sudo nvidia-smi mig -cgi 14,14,14 -C
echo "Created 3 instances of 1g.10gb profile."
```

-cgi = compute gpu instance profile ID
-c = commit to apply the configuration

Step 3: Assign MIG Devices to Containers or Users:

- Docker run –gpus “‘device=UUID”’ …
- Use nvidia-smi -L to list MIG devices UUIDS

Step 4: Delete All MIG Instances:

```
#!/bin/bash
echo "Deleting all MIG instances..."
sudo nvidia-smi mig -dci
echo "Deleted all Compute Instances."
```

Supported MIG Profiles for A100 80 GB (SXM4):

```
| Profile Name | Profile ID | GPU Slice | Memory Slice | Max Instances
|
| ------------ | ---------- | --------- | ------------ | -------------
|
| **1g.10gb**  | 14         | 1/7       | 10 GB        | 7
|
| **2g.20gb**  | 9          | 2/7       | 20 GB        | 3
|
| **3g.40gb**  | 5          | 3/7       | 40 GB        | 2
|
| **4g.40gb**  | 2          | 4/7       | 40 GB        | 1
|
| **7g.80gb**  | 0          | Full GPU  | 80 GB        | 1
|
```

Key Terms(About the infra):

1. GPU

- Refers to the **physical GPU** ID.
- You have 8 physical **NVIDIA A100 80GB SXM4** GPUs. This example output is from **GPU 0**.
- MIG enables slicing this GPU into multiple logical instances.

2. GI (GPU Instance ID)

- A **GPU Instance** (GI) is a logical partition of the physical GPU.
- It defines **memory partitions and compute engine isolation**.
- Each GI gets its own memory chunk (e.g., 10 GB in `1g.10gb`) and connects to one or more compute instances (CI).
- Think of GI as a "container" of resources carved from the full GPU.

3. CI (Compute Instance ID)

- A **Compute Instance** is the actual execution unit, tied to a GPU instance.
- It defines how many **Streaming Multiprocessors (SMs)** and **compute resources** (Tensor Cores, CUDA cores, etc.) are allocated.
- Each CI is associated with a specific GI.

4. MIG Dev (MIG Device ID)

- This is the logical **MIG device number** exposed to the system.
- You use this ID when running workloads via `nvidia-smi`, Docker (`--gpus`), Kubernetes, or other orchestration platforms.
- Example: `docker run --gpus '"device=0:3"' ...` uses MIG Dev 3 on GPU 0.

5. Memory-Usage

- Shows **current memory usage** vs **total memory allocated** for that MIG slice.
- In your config: `9728MiB` (~9.5 GB) per instance (matching `1g.10gb` profile).
- `BAR1-Usage`: Reserved memory used for host-device communication (used in virtualization contexts).

6. SM (Streaming Multiprocessors)

- The number of **SMs assigned** to the instance. Each SM contains:
    - CUDA cores (for general-purpose computation)
    - Tensor cores (for deep learning)
    - Shared memory and control logic
- In your case, each instance has `14 SMs`, which is 1/7 of an A100's 108 SMs.

7. Volatile ECC (Error Correction Code)

- Monitors for **uncorrected GPU memory errors**.
- In your case: `0` → no errors. Very important for data center reliability.

8. Shared Engines

These represent **special-purpose hardware accelerators** inside the GPU:

Engine	Description	Use Case
`CE`	Compute Engine	Core CUDA/Tensor ops
`ENC`	Video Encoder	Video encoding (e.g., H.264/H.265)
`DEC`	Video Decoder	Video decoding
`OFA`	Optical Flow Accelerator	Optical flow estimation
`JPG`	JPEG Engine	Hardware-accelerated image encoding/decoding

All your instances show `1 CE`, meaning each has a compute engine available. Others are `0` because video/image features are not enabled in `1g.10gb`.

Visual Analogy

Imagine your A100-80GB GPU is a **pizza with 8 slices (SMs)** and 80GB of cheese (memory):

- GI = the **box** holding a slice (defines memory)
- CI = the **actual slice of pizza** inside that box (defines compute)
- MIG Dev = the **number tag** on the box so people know which to pick
- Docker/K8s use MIG Dev IDs to hand out those boxes to apps