1.5 Hands-on Lab: Set up a GPU-powered VM on AWS with NVIDIA drivers

What you'll do

Launch an EC2 GPU instance, SSH in, verify the GPU/driver, install Docker + NVIDIA Container Toolkit, and run a quick GPU test in a container.

Prerequisites (5–10 min)

- An AWS account with permissions to launch EC2.
- A key pair (PEM) and a default VPC/subnet.
- Basic terminal/SSH access.

> **Tip:** If this is your first time using GPU instances, you may need a service quota increase before you can launch p4/p5/g6. AWS support can raise it.
> **Instance/GPU mapping (for choosing later):**
> • **g6** → NVIDIA **L4** (great for inference/video) (Amazon Web Services, Inc.)
> • **g5** → NVIDIA **A10G** (graphics/inference) (Amazon Web Services, Inc., NVIDIA Developer)
> • **p4d** → NVIDIA **A100** (training) (Amazon Web Services, Inc., NVIDIA Blog)
> • **p5/p5e/p5en** → NVIDIA **H100/H200** (frontier training) (Amazon Web Services, Inc.)

Part A — Launch the EC2 GPU instance (Console) (8–12 min)

1. **Pick a Region**
   Choose an AWS region where your target instance family is available (e.g., us-east-1).

2. **AMI (the easy path):**
   In EC2 → *Launch instance* → **Application and OS Images (Amazon Machine Image)**, search for:

- **"Deep Learning Base GPU AMI (Amazon Linux 2/2023)"** or a framework AMI (e.g., "Deep Learning AMI GPU PyTorch"). These AMIs include NVIDIA drivers/CUDA and are maintained by AWS. (AWS Documentation)

> Prefer this AMI for a smoother first run. (You can always use vanilla Ubuntu later and install drivers manually.)

3. **Instance type**
   Pick one that matches your workload and budget, e.g., **g6.xlarge** (L4) for lightweight GPU work, **p4d.24xlarge** (A100) or **p5.48xlarge** (H100) for heavy training. (Vantage Instances, Amazon Web Services, Inc.)

4. **Key pair & network**
   Select your key pair. Use the default VPC/subnet. Create a **Security Group** that allows **SSH (TCP 22)** from your IP.

5. **Storage**
   Set **50–200 GB** gp3 EBS (containers + datasets need space).

6. **Launch**
   Click **Launch instance** and wait until the state is **Running**.

Part B — Connect & verify the GPU (3–5 min)

1. **SSH in**

```
ssh -i /path/to/your-key.pem ec2-user@<EC2_PUBLIC_IP>   # Amazon Linux
# or
ssh -i /path/to/your-key.pem ubuntu@<EC2_PUBLIC_IP>     # Ubuntu-based AMIs
```

2. **Check the GPU & driver**

```
nvidia-smi
```

You should see the GPU model (L4/A10G/A100/H100), driver version, and CUDA version. If you used a Deep Learning Base GPU AMI, drivers are already present. (AWS Documentation)

> If `nvidia-smi` is missing on a non-DLAMI image, install the correct driver/CUDA using NVIDIA's Linux guide (see "Optional: manual driver install" below). (NVIDIA Docs)

Part C — Install Docker & NVIDIA Container Toolkit (8–12 min)

> Skip Docker if your AMI already includes it—otherwise:

Amazon Linux 2023 / 2

```
sudo yum update -y
sudo yum install -y docker
sudo systemctl enable --now docker
sudo usermod -aG docker $USER
newgrp docker
```

Ubuntu

```
sudo apt-get update
sudo apt-get install -y docker.io
sudo usermod -aG docker $USER
newgrp docker
```

Add the NVIDIA Container Toolkit (official method)

```
# 1) Add repo + key (Ubuntu/Debian example)
curl -fsSL https://nvidia.github.io/libnvidia-container/gpgkey | \
  sudo gpg --dearmor -o /usr/share/keyrings/nvidia-container-toolkit-keyring.gpg
curl -s -L https://nvidia.github.io/libnvidia-container/stable/deb/nvidia-container-toolkit.list | \
  sed 's#deb https://#deb [signed-by=/usr/share/keyrings/nvidia-container-toolkit-keyring.gpg] https://#g' | \
  sudo tee /etc/apt/sources.list.d/nvidia-container-toolkit.list

sudo apt-get update
sudo apt-get install -y nvidia-container-toolkit

# 2) Configure Docker to use NVIDIA runtime
sudo nvidia-ctk runtime configure --runtime=docker
sudo systemctl restart docker
```

(Use the distro-specific variant from NVIDIA's guide if not on Ubuntu.) (NVIDIA Docs)

Part D — Quick GPU test inside a container (2–4 min)

Run a CUDA container and check `nvidia-smi` **from inside** the container:

```
docker run --rm --gpus all nvidia/cuda:12.5.0-runtime-ubuntu22.04 nvidia-smi
```

You should see the same GPU table (driver/library versions) printed by the container—confirming Docker + GPU pass-through works. (If you prefer NGC images, use `nvcr.io/...` after logging in with an NGC API key.) (NVIDIA Docs)

Optional: Manual driver/CUDA install (if you didn't use a DLAMI)

If you launched a plain Ubuntu/AL2023 image and need to install drivers/CUDA yourself, follow NVIDIA's **CUDA Installation Guide for Linux** to match your driver to the CUDA/toolkit version, then reboot and re-run `nvidia-smi`. (NVIDIA Docs)

Troubleshooting (quick hits)

- **`nvidia-smi` not found** → Driver isn't installed or path isn't set. Install via DLAMI or NVIDIA's Linux guide, then reboot. (AWS Documentation, NVIDIA Docs)
- **Container can't see GPUs** → Make sure **NVIDIA Container Toolkit** is installed and `nvidia-ctk runtime configure --runtime=docker` ran; restart Docker. (NVIDIA Docs)
- **Wrong instance for your workload** → Use **g6 (L4)** for inference/video, **p4d (A100)** or **p5/p5e (H100/H200)** for training. (Amazon Web Services, Inc.)

Clean up (important)

When you're done, **Stop** the instance to pause charges (EBS still billed) or **Terminate** it to delete compute + storage and stop all charges.

You're done 🎉

You now have an AWS GPU VM with drivers and containerized GPU access—ready to pull Triton, PyTorch/TensorFlow, DeepStream, or your own images and start building!