<table>
<thead>
<tr>
<th>Chapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Introduction to Using NGC with AWS</td>
</tr>
<tr>
<td>2. Preliminary Setup</td>
</tr>
<tr>
<td>2.1. Setting Up Your AWS Key Pair</td>
</tr>
<tr>
<td>2.2. Setting Up Security Groups for the EC2 Instance</td>
</tr>
<tr>
<td>2.3. Setting Up Security Groups for EFS</td>
</tr>
<tr>
<td>3. Launching a VM Instance from the AWS Console</td>
</tr>
<tr>
<td>3.1. Logging In and Selecting the AWS Zone</td>
</tr>
<tr>
<td>3.2. Selecting the NVIDIA Deep Learning AMI</td>
</tr>
<tr>
<td>3.3. Selecting an Amazon EC2 P3/G4 Instance Type and Configuring Instance Settings</td>
</tr>
<tr>
<td>3.4. Launching Your VM Instance</td>
</tr>
<tr>
<td>3.5. Connecting to Your VM Instance</td>
</tr>
<tr>
<td>3.6. Starting, Stopping, and Terminating Your VM Instance</td>
</tr>
<tr>
<td>4. Launching a VM Instance Using AWS CLI</td>
</tr>
<tr>
<td>4.1. Setting Up Environment Variables</td>
</tr>
<tr>
<td>4.2. Launching Your VM Instance</td>
</tr>
<tr>
<td>4.3. Connecting To Your VM Instance</td>
</tr>
<tr>
<td>4.4. Starting, Stopping, and Terminating Your VM Instance</td>
</tr>
<tr>
<td>5. Using the Amazon Elastic File System (EFS) for Persistent Data Storage</td>
</tr>
<tr>
<td>5.1. Creating an EFS</td>
</tr>
<tr>
<td>5.2. Mounting an EFS</td>
</tr>
<tr>
<td>5.3. Deleting an EFS</td>
</tr>
<tr>
<td>5.4. Managing an EFS Using AWS CLI</td>
</tr>
<tr>
<td>6. Using the Amazon Elastic Block Storage (EBS) for Persistent Data Storage</td>
</tr>
<tr>
<td>6.1. Creating an EBS</td>
</tr>
<tr>
<td>6.2. Attaching an EBS Volume to the EC2 Instance</td>
</tr>
<tr>
<td>6.3. Deleting an EBS</td>
</tr>
<tr>
<td>6.4. Copying Datasets to the EBS Volume</td>
</tr>
<tr>
<td>6.4.1. Uploading a Dataset to the EBS Volume</td>
</tr>
<tr>
<td>6.4.2. Copying a Dataset from Existing EFS Storage to the EBS Volume</td>
</tr>
<tr>
<td>6.5. Managing an EBS Volume Using AWS CLI</td>
</tr>
<tr>
<td>7. Examples of Running Containers</td>
</tr>
<tr>
<td>7.1. Logging Into the NGC Container Registry</td>
</tr>
<tr>
<td>7.2. Preparing to Run Containers</td>
</tr>
<tr>
<td>7.3. Running a Container</td>
</tr>
<tr>
<td>7.4. Example: MNIST Training Run Using PyTorch Container</td>
</tr>
<tr>
<td>7.5. Example: MNIST Training Run Using TensorFlow Container</td>
</tr>
<tr>
<td>8. Power User Method for Launching a VM Instance: Terraform</td>
</tr>
<tr>
<td>8.2. Launching Your VM Instance</td>
</tr>
<tr>
<td>8.3. Starting, Stopping, and Terminating Your VM Instance</td>
</tr>
</tbody>
</table>
NVIDIA makes available on the Amazon Web Service (AWS) platform a customized Amazon Machine Instance (AMI) optimized for the NVIDIA® Volta™ and Turing GPUs, called the Amazon EC2 P3 and G4 Instances respectively. Running NGC containers on this instance provides optimum performance for deep learning, machine learning, and HPC jobs.

Three flavors of the NVIDIA GPU Cloud image are available:

- Standard NVIDIA GPU Cloud Image
  - Includes Ubuntu Server, the NVIDIA driver, Docker CE, and the NVIDIA Container Runtime for Docker
- TensorFlow from NVIDIA image
  - The standard image plus a built-in, ready-to-use TensorFlow container
- PyTorch from NVIDIA image
  - The standard image plus a built-in, ready-to-use PyTorch container

For those familiar with the AWS platform, the process of launching the instance is as simple as logging into AWS, selecting the NVIDIA Deep Learning AMI and one of the Amazon EC2 P3 or G4 instance types, configuring settings as needed, then launching the instance. After launching the instance, you can SSH into the instance and start running deep learning jobs using framework containers from the NGC Container Registry.

This document provides step-by-step instructions for accomplishing this, including how to use the AWS CLI.

**Prerequisites**

These instructions assume the following:

- You have an AWS account - [https://aws.amazon.com](https://aws.amazon.com)
- Browsed the [NGC website](https://ngc.nvidia.com) and identified an available NGC container and tag to run on the VMI.
Windows Users: The CLI code snippets are for bash on Linux or Mac OS X. If you are using Windows and want to use the snippets as-is, you can use the Windows Subsystem for Linux and use the bash shell (you will be in Ubuntu Linux).

If you plan to use AWS CLI, then the CLI must be installed, updated to the latest version, and configured.

Some of the AWS CLI snippets in these instructions make use of jq, which should be installed on the machine from which you’ll run the AWS CLI. You may paste these snippets into your own bash scripts or type them at the command line.

Additionally, if you plan to access locked NGC containers, you will need to perform the following steps from the NGC website (see NGC Getting Started Guide)

- Signed up for an NGC account at https://ngc.nvidia.com/signup.
- Created an NGC API key for access to locked containers within the NGC container registry.
Chapter 2.
PRELIMINARY SETUP

Perform these preliminary setup tasks to simplify the process of launching the NVIDIA Deep Learning AMI.

2.1. Setting Up Your AWS Key Pair

If you do not already have Key Pairs defined, then you will need to setup your AWS Key Pair and have it on the machine on which you will use the AWS CLI, or from which you will SSH to the instance. In the examples, the key pair is named "my-key-pair".

Once you have your key pair downloaded, make sure they are only readable by you, and (Linux or OSX) move them to your ~/.ssh/ directory.

```
chmod 400 my-key-pair*
mv my-key-pair* ~/.ssh/
```

On Windows, the location will depend on the SSH client you use, so modify the NVAWS_KEYPATH variable in the instance launch snippets.

2.2. Setting Up Security Groups for the EC2 Instance

In order to reach your running instances you will need a Security Group allowing (at minimum) SSH access.

1. Log into the AWS Console (https://aws.amazon.com), then under the Compute section, click EC2.
2. Enter the Security Groups screen, located on the left under "Network & Security", "Security Groups".
3. Click **Create Security Group**.
4. Give the Security Group a name (for example, "my-sg"), description, and then click **Add Rule**
5. On the "Inbound" tab, add a rule for SSH, then click **Create**:
   - Type: **SSH**
   - Protocol: **TCP**
   - Port Range: **22**
   - Source: **My IP**
   
   You may need to widen the resulting IP filter if you're not on a fixed IP address, or want to access the instance from multiple locations such as work and home.

   The following shows the filled-out Create Security Group form using the example naming.

6. (Optional) Add additional rules.
   
   You may need to add additional rules for HTTP, HTTPS, or other Custom TCP ports depending on the deep learning frameworks you use.
   
   a) Return to the Security Group screen, select the group, then select **Edit inbound rules** from the Actions menu.
b) Click **Add Rule**, then create rules as needed and click **Save**.

Examples:

- For **DIGITS4**
  - Type: **Custom TCP Rule**
  - Protocol: TCP
  - Port Range: **3448**
  - Source: **My IP**

- For **HTTPS secure web frameworks**
  - Type: HTTPS
  - Protocol: TCP
  - Port Range: **443**
  - Source: **My IP**

Once created, the Group ID is listed in the Security Group table.

### 2.3. Setting Up Security Groups for EFS

If you will be using EFS storage, you need to create a second **Security Group** that specifies NFS access and then use that Security group when creating the EFS.

1. Click **Create Security Group** from the Security Groups page.
2. Give the Security Group a name (for example, "my-efs-sg"), description, and then click **Add Rule**.
3. On the "Inbound" tab, add a rule for NFS access, then click **Create**:
   - Type: NFS
   - Protocol: TCP
   - Port Range: **2049**
   - Source: **Anywhere**
3.1. Logging In and Selecting the AWS Zone

1. Log into the AWS Console (https://aws.amazon.com), then under the Compute section, click EC2.
2. Select the AWS Zone from the upper right of the top menu.

In order to use NVIDIA Volta and Turing GPUs in AWS, you must select a region that has Amazon EC2 P3 or G4 instances available. The examples in this guide use instances in US West (Oregon) - us-west-2. Check with AWS for Amazon EC2 P3 or G4 instance availability in other regions.

3.2. Selecting the NVIDIA Deep Learning AMI
NVIDIA publishes and maintains an AMI with all the software needed to pull and run the NGC deep learning containers. This AMI should be used as the basis for your Volta instance types.

1. Click Launch Instance.

2. Select the NVIDIA Deep Learning AMI.
   a) Select AWS Marketplace on the left, and search for "NVIDIA Deep Learning AMI", and select the NVIDIA AMI you want to use - either the basic, the PyTorch AMI, or the TensorFlow AMI.
   b) Click Continue on the details page.

3.3. Selecting an Amazon EC2 P3/G4 Instance Type and Configuring Instance Settings

1. Select one of the Amazon EC2 P3 or G4 instance types according to your GPU, CPU, and memory requirements.

2. Click Review and Launch to review the default configuration settings, or continue with the instructions in the next section to configure each setting step-by-step.

3. After choosing an instance type, click Next: Configure Instance Details.

   There are no instance details that need to be configured, so you can proceed to the next step.

4. Add storage.

   Click Next: Add Storage.

   While the default 32 GiB for the root volume can be changed, users should not use the root volume for storing datasets since the root volume is destroyed when the instance is terminated. NVIDIA provides directions in this guide for using EFS volumes for datasets and deep learning job results. See the chapter Using the Amazon Elastic File System (EFS) for Persistent Data Storage.

5. Add tags.

   Naming your instances helps to keep multiple instances organized.
   a) Click Next: Add Tag.
   b) Click Add Tag and then fill in the following information:
      Key: "Name"
Value: <instance name, such as "My GPU”>

6. Configure a Security Group
   a) Click Next: Configure Security Group.
   b) Click Select an existing security group and select the Security Group you created during Preliminary Setup.

3.4. Launching Your VM Instance

1. Click Review and Launch.

   A window pops up and asks which key pair to use.

2. Select Choose an existing key pair, select your key pair, then check the acknowledgement checkbox.

3. Click Launch Instances.

3.5. Connecting to Your VM Instance

1. After launching your instance, click View Instances, locate your instance from the list, then wait until it is in the ‘running’ state.

2. When it is in the running state, select it from the list and then click Connect.

3. Follow the instructions in the pop-up window to establish an SSH connection to the instance.
   
   Be sure to use 'ubuntu' for the username.
If you plan to access locked NGC containers, you will need to log in to the NGC container registry. See Logging in to the NGC Container Registry for instructions.

If the instructions for SSH login do not work, see the AWS Connect to Your Linux Instance documentation for additional information.

### 3.6. Starting, Stopping, and Terminating Your VM Instance

Once you are done with your instance you can stop (to be started again later) or terminate (delete) it. Refer to the Instance Lifecycle in the AWS documentation for more information.

Instances can be controlled from the Instances page, using the "Actions"->"Instance State" menu to stop, start, or terminate Instances.
Chapter 4.
LAUNCHING A VM INSTANCE USING AWS CLI

This flow and the code snippets in this section are for Linux or Mac OS X. If you are using Windows, you can use the Windows Subsystem for Linux and use the bash shell (where you will be in Ubuntu Linux).

A comprehensive set of example bash scripts for automating the AWS CLI, and a Terraform configuration, are provided at https://github.com/nvidia/ngc-examples. You can download the scripts and modify them to meet your requirements. The code examples that follow use similar environment variables and structure as the scripts.

4.1. Setting Up Environment Variables

Set up the following environment variables which can be used in the commands for launching the VM instance:

**Security Group ID (NVAWS_SG_ID)**

The Security Group ID is used as part of the instance creation process. Once created the Group ID can be looked up in the AWS Console, or retrieved by name with the following snippet, and stored in the $NVAWS_SG_ID environment variable.

```
NVAWS_SG_NAME='my-sg'
NVAWS_SG_ID=$(aws ec2 describe-security-groups --group-name "$NVAWS_SG_NAME" | jq .SecurityGroups[0].GroupId | sed 's/"//g') && echo NVAWS_SG_ID=$NVAWS_SG_ID
```

**Image ID (NVAWS_IMAGE_ID)**

The following snippet will list the current "NVIDIA Deep Learning AMI" Image ID, and stored in the $NVAWS_IMAGE_ID environment variable.

```
NVAWS_IMAGE_NAME='NVIDIA Deep Learning AMI'
```
Other Environment Variables

Set up other env variables as follows, using your information:

```bash
NVAWS_KEYNAME=my-key-pair
NVAWS_KEYPATH=~/.ssh/
NVAWS_REGION=us-west-2
NVAWS_INSTANCE_TYPE=p3.2xlarge
NVAWS_EBS_GB=32
NVAWS_NAME_TAG='My GPU'
```

Be sure to set a unique `NVAWS_NAME_TAG` for each instance you launch.

### 4.2. Launching Your VM Instance

Launch the instance and capture the resulting JSON:

```bash
NVAWS_LAUNCH_JSON=$(aws ec2 run-instances --image-id $NVAWS_IMAGE_ID \ 
--instance-type $NVAWS_INSTANCE_TYPE \ 
--region $NVAWS_REGION \ 
--key-name $NVAWS_KEYNAME \ 
--security-group-ids $NVAWS_SG_ID \ 
--block-device-mapping \ 
"{"DeviceName":"/dev/sda1","Ebs":{"VolumeSize":$NVAWS_EBS_GB}}"]" \ 
--tag-specifications \ 
"ResourceType=instance,Tags=[{"Key=Name,Value=$NVAWS_NAME_TAG}]"
NVAWS_INSTANCE_ID=$(echo $NVAWS_LAUNCH_JSON | jq .Instances[0].InstanceId | sed 's/"//g') && echo NVAWS_INSTANCE_ID=$NVAWS_INSTANCE_ID
```

The resulting Instance ID is stored in the `NVAWS_INSTANCE_ID` environment variable.

The launch process can take several minutes once a machine is available, and can be watched in the AWS Console Instances page or with the CLI using:

```bash
aws ec2 describe-instance-status --instance-id $NVAWS_INSTANCE_ID \ 
| jq '.InstanceStatuses[0].InstanceState.Name + " " + .InstanceStatuses[0].SystemStatus.Status'
```

Once the instance is "running initializing", you will be able to get the Public DNS name with:

```bash
NVAWS_DNS=$(aws ec2 describe-instances --instance-id $NVAWS_INSTANCE_ID | jq \ 
'.Reservations[0].Instances[0].PublicDnsName' | sed 's/"//g') && \ echo NVAWS_DNS=$NVAWS_DNS
```

### 4.3. Connecting To Your VM Instance

SSH should work shortly after the instance reaches "running ok".
If started with CLI snippets and environment variables above, the command to SSH to your instance is:

```bash
ssh -i $NVAWS_KEYPATH/$NVAWS_KEYNAME.pem ubuntu@$NVAWS_DNS
```

Otherwise use your .pem key filename and the Public DNS name from the AWS Console to connect:

```bash
ssh -i my-key-pair.pem ubuntu@public-dns-name
```

If these instructions for SSH login do not work, see the AWS Connect to Your Linux Instance documentation for additional information.

If you plan to access locked NGC containers, you will need to log in to the NGC container registry. See Logging in to the NGC Container Registry for instructions.

### 4.4. Starting, Stopping, and Terminating Your VM Instance

Once you are done with your instance you can stop (to be started again later) or terminate (delete) it. Refer to the Instance Lifecycle in the AWS documentation for more information.

**Stop:**

```bash
aws ec2 stop-instances --instance-ids $NVAWS_INSTANCE_ID
```

**Start:**

```bash
aws ec2 start-instances --instance-ids $NVAWS_INSTANCE_ID
```

**Terminate:**

```bash
aws ec2 terminate-instances --instance-ids $NVAWS_INSTANCE_ID
```
You can create an elastic file system (EFS) from the AWS Console. EFS is suggested since a single EFS volume can be simultaneously accessed by multiple concurrent instances, and can be expanded beyond the initial size.

For working with a large number of small files, such as in a dataset, elastic block storage (EBS) offers better performance. For instructions on setting up and using EBS, see the section Using the Amazon Elastic Block Storage (EBS) for Persistent Data Storage.

EFS is available in most regions with Amazon EC2 P3 or G4 instances.

5.1. Creating an EFS

1. Open the EFS Console.
   Go to the main AWS console and select EFS under the "Storage" section.
2. Click Create file system.
3. At the Configure file system access page, make sure the Security group that you created with NFS access is selected, then click Next Step.
4. Configure optional settings:
   a) Enter a value for the Name key
      This is the name of your EFS.
   b) Under Performance mode, select General purpose
      The default setting will work for many workloads. You can experiment with other settings that are better suited to your workload.
   c) Encryption is not needed if you are working with public datasets. For private datasets it's optional and involves additional setup steps. Read the AWS documentation if you need encryption.
   d) Click Next step.
5. Review the options and then click **Create File System**.

### 5.2. Mounting an EFS

1. **Open the EFS Console**
   
   Go to the main AWS console and select **EFS** under the "Storage" section.

2. **Select your EFS to expand its details.**

3. **Click **Amazon EC2 mount instructions**.**

   The console displays instructions specific to the selected EFS file system. The nfs-common package is already installed in the NVIDIA AMI, so only the mount step is required. The snippet that AWS shows assumes mounting to an "efs" directory, but the other values including DNS are correct.

```
Mounting your file system

1. Open an SSH client and connect to your EC2 instance. (find out how to connect)
2. Create a new directory on your EC2 instance, such as "efs":
   - `sudo mkdir efs`
3. Mount your file system using the DNS name. (Mounting considerations)
   - `sudo mount -t nfs4 -o nfsvers=4.1,hard,timeo=600,retrans=2,fs
   - f5d47a5c.efs.us-west-2.amazonaws.com:/efs`

If you are unable to connect, please see our troubleshooting documentation.
```

There are several documents linked from the mounting instructions explaining advanced mounting options and troubleshooting.

To mount your EFS automatically every time you stop and then start the instance, you need to edit the `/etc/fstab` file that is provided in the instance. Refer to the Amazon documentation **Mounting Automatically**

### 5.3. Deleting an EFS

Be aware that once you delete an EFS, you cannot undelete it.

1. **Open the EFS Console.**

   Go to the main AWS console and select **EFS** under the "Storage" section.

2. **Select your EFS.**

3. **Using the "Actions" dropdown, select Delete file system.**
4. In the confirmation dialog, enter the file system ID, and then click **Delete File System**.

### 5.4. Managing an EFS Using AWS CLI

It is recommended that you use the AWS Console for EFS management. If you need to manage EFS file systems with the CLI, NVIDIA has created scripts available on GitHub at [https://github.com/nvidia/ngc-examples](https://github.com/nvidia/ngc-examples).

These scripts will let you perform basic EFS management and can serve as the basis for further automation.
Chapter 6.
USING THE AMAZON ELASTIC BLOCK STORAGE (EBS) FOR PERSISTENT DATA STORAGE

You can create elastic block storage (EBS) from the AWS Console. Like EFS, EBS is used for persistent data storage, but offers better performance when using a large number of small files. Unlike EFS, EBS cannot be shared across multiple VMs. To share persistent data storage, you need to use EFS.

The instructions set up a general purpose SSD volume type. However, you can specify a provisioned IOPS SSD for higher throughput, or set up software RAID, using mdadm, to create a volume with multiple EBS volumes. See the Amazon documentation RAID Configuration on Linux for instructions on how to set up software RAID on local disks.

EBS is available in most regions with Amazon EC2 P3 or G4 instances.

6.1. Creating an EBS

1. Open the EBS Volumes Console.
   Go to the main AWS console, click EC2, then expand Elastic Block Store from the side menu, if necessary, and click Volumes.

2. Click Create Volume.

3. Make selections at the Create Volume page.
   - Select General Purpose SSD (GP2) for the Volume Type.
     If higher throughput is needed, select Provisioned IOPS SSD (IO1).
   - Specify the volume size and Availability Zone.
   - (Optional) Add Tags.
   - Encryption is not needed if you are working with public datasets.
   - Snapshot ID is not needed.
4. Review the options and then click **Create Volume**.

6.2. Attaching an EBS Volume to the EC2 Instance

1. Once you have created the EBS volume, select the volume and then select Actions->**Attach Volume**.
2. Specify your EC2-instance ID as well as a drive letter for the device name (for example, sdf), then click **Attach**.

   This creates a `/dev/xvd{f,d}` (or the driver letter that you picked) virtual disk on your EC2 instance.

   You can view the volume by running the `lsblk` command.

   ```bash
   ~$ lsblk
   NAME     MAJ:MIN RM  SIZE   RO TYPE MOUNTPOINT
   xvda     202:0    0 128G   0 disk
   └─xvda1  202:1    0 128G   0 part /
   xvdf     202:16   0 250G   0 disk
   ``

3. Create a filesystem on the EBS volume.

   ```bash
   ~# mkfs.ext4 /dev/xvdf
   ```

   mke2fs 1.42.13 (17-May-2015)
   Creating filesystem with 65536000 4k blocks and 16384000 inodes
   Filesystem UUID: b0e3dee3-bf86-4e69-9488-cf4d4b57b367
   Superblock backups stored on blocks:
   32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
   4096000, 7962624, 11239424, 20480000, 23887872
   Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done

4. Mount the volume to a mount directory.

```
~# mount /dev/xvdf /data
```

To mount the volume automatically every time the instance is stopped and restarted, add an entry to `/etc/fstab`. Refer to Amazon Documentation [Making a Volume Available for Use](#).

### 6.3. Deleting an EBS

Be aware that once you delete an EBS, you cannot undelete it.

1. Open the EBS Volumes Console.
   
   Go to the main AWS console, click **EC2**, then expand **Elastic Block Store** from the side menu, if necessary, and click **Volumes**.

2. Select your EBS.

3. Detach the volume from the EC2 instance.
   
   Select Actions->**Detach Volume**, then click **Yes, Detach** from the confirmation dialog.

4. Delete the storage volume.
   
   Select Actions->**Delete Volume** and then click **Yes, Delete** from the confirmation dialog.

### 6.4. Copying Datasets to the EBS Volume

Once you have created the EBS volume, you can upload datasets to the volume or copy the dataset from an existing EFS.

#### 6.4.1. Uploading a Dataset to the EBS Volume

1. Mount the EBS volume to `/data`.

   Issue the following to perform the one-time mount.

   ```
sudo mkdir /data
sudo mount /dev/xvdf /data
sudo chmod 777 /data
```

2. Copy the dataset onto the EBS volume in `/data`.

   ```
   scp -i <.pem> -r local_dataset_dir/ ubuntu@<ec2-instance>:/data
   ```
6.4.2. Copying a Dataset from Existing EFS Storage to the EBS Volume

1. Mount the EFS storage to /data, using the EFS storage DNS name.
   Issue the following to perform the one-time mount.

   ```bash
   sudo mkdir /efs
   sudo mount -t nfs4 -o nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2 \
   EFS-DNS-NAME:/ /efs
   sudo chmod 777 /efs
   sudo cp -r /efs/<dataset> to /data
   ```

2. Copy the dataset from the EFS to the EBS volume.

   ```bash
   sudo cp -r /efs/<dataset> to /data
   ```

6.5. Managing an EBS Volume Using AWS CLI

It is recommended that you use the AWS Console for EBS management. If you need to manage EBS file systems with the CLI, NVIDIA has created scripts available on GitHub at https://github.com/nvidia/ngc-examples.

These scripts will let you perform basic EBS management and can serve as the basis for further automation.
Chapter 7.
EXAMPLES OF RUNNING CONTAINERS

This chapter walks you through the process of logging in to the NGC container registry, pulling and running a container, and using file storage and data disks for storage.

7.1. Logging Into the NGC Container Registry

You need to log in to the NGC container registry only if you want to access locked containers from the registry. Most of the NGC containers are freely available (unlocked) and do not require an NGC account or NGC API key.

You do not need to log into the NGC container registry if you are using either the NVIDIA Deep Learning with PyTorch or the NVIDIA Deep Learning with TensorFlow and intend to use the containers already built into the image.

If necessary, log in to the NGC container registry manually by running the following script from the VMI.

```
ngc-login.sh <your-NGC-API-key>
```

From this point you can run Docker commands and access locked NGC containers from the VM instance.

7.2. Preparing to Run Containers

The VMI includes a mechanism for supporting GPUs within Docker containers to obtain the best performance. Depending on the NVIDIA VMI version, the mechanisms are as follows.

- Native GPU support with Docker-CE
  Requires Docker-CE 19.03 or later (Included in NVIDIA VMIs 19.10 and later)
Examples of Running Containers

- NVIDIA Container Runtime with Docker-CE
  Included in NVIDIA VMIs prior to 19.10

Using Native GPU Support with Docker-CE
Use this method with NVIDIA VMIs version 19.10 and later.
Use `docker run --gpus` to run GPU-enabled containers.

- Example using all GPUs
  
  ```
  $ docker run --gpus all ...
  ```

- Example using two GPUs
  
  ```
  $ docker run --gpus 2 ...
  ```

- Examples using specific GPUs
  
  ```
  $ docker run --gpus "device=1,2" ...
  $ docker run --gpus "device=UUID-ABCDEF,1" ...
  ```

Using the NVIDIA Container Runtime with Docker-CE
Use this method with NVIDIA VMIs prior to version 19.10
Use `docker run` and specify `runtime=nvidia`.

  ```
  $ docker run --runtime=nvidia ...
  ```

### 7.3. Running a Container

This section explains the basic process for running a container on the NVIDIA Deep Learning for TensorFlow, the NVIDIA Deep Learning for PyTorch, and the basic NVIDIA Deep Learning.

**Running the Built-in TensorFlow Container**
To run the TensorFlow container in the VM created from the NVIDIA Deep Learning for TensorFlow, refer to the release notes for the correct tag to use, then enter the following command.

**On NVIDIA VMIs version 19.10 and later**

```
docker run --gpus all --rm -it nvcr.io/nvidia/tensorflow:<tag>
```

**On NVIDIA VMIs prior to version 19.10**

```
docker run --runtime=nvidia --rm -it nvcr.io/nvidia/tensorflow:<tag>
```

**Running the Built-in PyTorch Container**
To run the PyTorch container in the VM created from the NVIDIA Deep Learning, refer to the release notes for the correct tag to use, then enter the following command.
On NVIDIA VMIs version 19.10 and later

docker run --gpus all --rm -it nvcr.io/nvidia/pytorch:<tag>

On NVIDIA VMIs prior to version 19.10

docker run --runtime=nvidia --rm -it nvcr.io/nvidia/pytorch:<tag>

Running a Container from the NGC Container Registry

To run containers from the NGC container registry,

1. If necessary, log in to the NGC container registry as explained in the previous section.
2. Enter the following commands.

docker pull nvcr.io/nvidia/<container-image>:<tag>

On NVIDIA VMIs version 19.10 and later

docker run --gpus all --rm -it nvcr.io/nvidia/<container-image>:<tag>

On NVIDIA VMIs prior to version 19.10

docker run --runtime=nvidia --rm -it nvcr.io/nvidia/<container-image>:<tag>

7.4. Example: MNIST Training Run Using PyTorch Container

Once logged in to the NVIDIA GPU Cloud Image instance, you can run the MNIST example under PyTorch.

Note that the PyTorch example will download the MNIST dataset from the web.

1. Pull and run the PyTorch container:

    docker pull nvcr.io/nvidia/pytorch:18.02-py3

    On NVIDIA VMIs version 19.10 and later

    docker run --gpus all --rm -it nvcr.io/nvidia/pytorch:18.02-py3.10

    On NVIDIA VMIs prior to version 19.10

    docker run --runtime=nvidia --rm -it nvcr.io/nvidia/pytorch:18.02-py3.10

2. Run the MNIST example:

    cd /opt/pytorch/examples/mnist

    python main.py
7.5. Example: MNIST Training Run Using TensorFlow Container

Once logged in to the NVIDIA GPU Cloud image, you can run the MNIST example under TensorFlow.

Note that the TensorFlow built-in example will pull the MNIST dataset from the web.

1. Pull and run the TensorFlow container.

```bash
docker pull nvcr.io/nvidia/tensorflow:18.08-py3
```
On NVIDIA VMIs version 19.10 and later

```bash
docker run --gpus all --rm -it nvcr.io/nvidia/tensorflow:18.08-py3
```
On NVIDIA VMIs prior to version 19.10

```bash
docker run --runtime=nvidia --rm -it nvcr.io/nvidia/tensorflow:18.08-py3
```

2. Following this tutorial: https://www.tensorflow.org/get_started/mnist/beginners, run the `MNIST_with_summaries` example.

```bash
cd /opt/tensorflow/tensorflow/examples/tutorials/mnist
python mnist_with_summaries.py
```
Chapter 8.
POWER USER METHOD FOR LAUNCHING A VM INSTANCE: TERRAFORM

Terraform enables you to safely and predictably create, change, and improve production infrastructure - and in this case easily get an AWS instance running. Instructions for installing Terraform on multiple platforms are in the Terraform Getting Started along with an overview of Terraform concepts.

8.1. Setting Up a Security Group

Use the AWS Console to set up a security group as explained in the Preliminary Setup section.

8.2. Launching Your VM Instance

Get the Terraform configuration files (a Terraform config is a directory with several .tf files) from https://github.com/nvidia/ngc-examples. Once you clone that git repo to an ngc-examples directory on your system, you will find the Terraform files in ngc-examples/aws/terraform.

You will also need your AWS credentials by either:

- Installing and configuring the AWS CLI (see Prerequisites). Terraform will use those AWS credentials.

Or

- Add your access and secret keys to the configuration file in the provider section of variables.tf.

Initialize Terraform in that directory:

```
terraform init
```
Look at variables.tf and check that the variables agree with your desired setup and file locations. It’s best practice to run `terraform plan` before any change in order to test and check for errors:

```
terraform plan
```

Then apply the configuration:

```
terraform apply
```

Your Amazon EC2 P3 or G4 instance will now be initializing, and Terraform will output the SSH command to connect to the instance once it’s running. This may take several minutes.

At any time you can get the instance parameters and SSH line shown when you started the instance. Enter the configuration directory and type:

```
terraform output
```

### 8.3. Starting, Stopping, and Terminating Your VM Instance

Stop: (Requires that AWS CLI is installed)

```
aws ec2 stop-instances --instance-ids <instance_id>
```

Start: (Requires that AWS CLI is installed)

```
aws ec2 start-instances --instance-ids <instance_id>
```

Terminate:

```
terraform destroy
```
Notice

THE INFORMATION IN THIS GUIDE AND ALL OTHER INFORMATION CONTAINED IN NVIDIA DOCUMENTATION REFERENCED IN THIS GUIDE IS PROVIDED “AS IS.” NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE INFORMATION FOR THE PRODUCT, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA’s aggregate and cumulative liability towards customer for the product described in this guide shall be limited in accordance with the NVIDIA terms and conditions of sale for the product.

THE NVIDIA PRODUCT DESCRIBED IN THIS GUIDE IS NOT FAULT TOLERANT AND IS NOT DESIGNED, MANUFACTURED OR INTENDED FOR USE IN CONNECTION WITH THE DESIGN, CONSTRUCTION, MAINTENANCE, AND/OR OPERATION OF ANY SYSTEM WHERE THE USE OR A FAILURE OF SUCH SYSTEM COULD RESULT IN A SITUATION THAT THREATENS THE SAFETY OF HUMAN LIFE OR SEVERE PHYSICAL HARM OR PROPERTY DAMAGE (INCLUDING, FOR EXAMPLE, USE IN CONNECTION WITH ANY NUCLEAR, AVIONICS, LIFE SUPPORT OR OTHER LIFE CRITICAL APPLICATION). NVIDIA EXPRESSLY DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS FOR SUCH HIGH RISK USES. NVIDIA SHALL NOT BE LIABLE TO CUSTOMER OR ANY THIRD PARTY, IN WHOLE OR IN PART, FOR ANY CLAIMS OR DAMAGES ARISING FROM SUCH HIGH RISK USES.

NVIDIA makes no representation or warranty that the product described in this guide will be suitable for any specified use without further testing or modification. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer’s sole responsibility to ensure the product is suitable and fit for the application planned by customer and to do the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer’s product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this guide. NVIDIA does not accept any liability related to any default, damage, costs or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this guide, or (ii) customer product designs.

Other than the right for customer to use the information in this guide with the product, no other license, either expressed or implied, is hereby granted by NVIDIA under this guide. Reproduction of information in this guide is permissible only if reproduction is approved by NVIDIA in writing, is reproduced without alteration, and is accompanied by all associated conditions, limitations, and notices.

Trademarks

NVIDIA, the NVIDIA logo, and Volta are trademarks and/or registered trademarks of NVIDIA Corporation in the United States and other countries.

Docker and the Docker logo are trademarks or registered trademarks of Docker, Inc. in the United States and/or other countries.

Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright

© 2019 NVIDIA Corporation. All rights reserved.