What Is an Instance Store?

Instance stores provide temporary storage that is physically attached to the host machine. Unlike EBS volumes—which reside on separate machines and are accessed via protocols like iSCSI—instance stores enable your EC2 instance to access a local physical drive directly on the host. Their temporary nature means that the stored data is lost if the instance is moved to a different host.

How Instance Stores Work

Consider a scenario where multiple EC2 instances run on the same host machine. When an EC2 instance utilizes an instance store:

- The instance directly accesses a physical drive on that host.
- Provided the instance remains on the same host, even after a reboot, it retains access to its instance store data.

However, when an instance is shut down and later restarted on a different host, it loses access to the original instance store and all associated data because the new host has a different storage configuration.

```
Warning
Moving an EC2 instance to a different host results in the
loss of any data stored on the original instance store. Use
instance stores only for data that can be safely discarded.
```

Use Cases and Limitations

Instance stores are best suited for applications where temporary data storage is acceptable and data persistence is not required. Common use cases include:

- Caching
- Buffering
- Temporary file storage

When designing systems, keep in mind that instance stores are not reliable for storing persistent data, as data loss will occur if the instance is migrated.

To summarize:

- Instance stores provide block-level, temporary storage for EC2 instances.
- Data on an instance store is lost if the instance migrates to a different host.
- They are ideal for transient data such as caches or scratch files, but not for persistent storage.

Launching an Instance with an Instance Store

To begin, navigate to the EC2 console and launch a new instance. For this demonstration, name your instance "instance store demo" and select the Amazon Linux 64-bit AMI. Note that not all EC2 instance types support instance stores; for example, t2.micro and other free tier instances lack instance store volumes. Select an instance type that includes instance store support—even if it comes at a small cost for this short demo.

After choosing an appropriate instance type, select any key pair of your preference.

Review the configuration details. You will notice that the default root volume is 8 GB, and further down, you will see an instance store volume (approximately 75 GB, reported as 69.8 GB). The device name for the instance store is typically something like /dev/nvme0n1 or /dev/nvme1n1. Also, ensure that "Auto-assign Public IP" is enabled so you can easily connect to your instance.

Once your instance is deployed, return to the EC2 console's instance tab. You should now see the instance with its assigned public IP.

Connecting and Configuring the Instance Store

After noting the public IP, SSH into your instance. Once connected, run the following command to verify the block devices:

```
lsblk
```

You'll see output similar to:

```
NAME         MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
nvme1n1     259:0    0  69.8G  0 disk
nvme0n1     259:1    0    8G  0 disk
├─nvme0n1p1 259:2    0    8G  0 part /
└─nvme0n1p128 259:4    0   10M  0 part
```

Here, nvme0n1 is your root volume (with partition nvme0n1p1 mounted as /), and nvme1n1 is the instance store volume.

Verify if a file system exists on the instance store volume by running:

```
sudo file -s /dev/nvme1n1
```

If the output shows "data," no file system exists. Create an XFS file system on the volume as follows:

```
sudo mkfs -t xfs /dev/nvme1n1
```

Then, validate the file system creation by running:

```
sudo file -s /dev/nvme1n1
```

You should now see that an SGI XFS filesystem is present on the device.

Next, create a directory to serve as the mount point and mount the instance store with these commands:

```
sudo mkdir /instance-demo
sudo mount /dev/nvme1n1 /instance-demo/
```

Confirm that the volume is mounted by executing:

```
df -h
```

Now, navigate into the mount directory and create a test file to verify that data is being written to the instance store:

```
cd /instance-demo
echo "test" | sudo tee test
```

The file "test" is now stored on your instance store volume.

```
Tip
Keep in mind that the instance store is optimized for fast,
temporary storage. It is ideal for cache, buffers, or
temporary files, but not for data that requires
```

```
persistence.
```

Demonstrating the Ephemeral Nature of Instance Store Data

Data on an instance store is ephemeral. A simple reboot will not change the physical host, so your instance store volume and its data remain intact. You can verify this by rebooting the instance from the EC2 console (right-click the instance and select "Reboot"). After the reboot, the public IP address will remain the same.

However, when you stop and then start your instance, it is relocated to a different physical host. This process replaces the instance store with a new, empty volume. To observe this behavior:

1. Record the current public IP address.
2. Stop the instance using the EC2 console.
3. Wait a few minutes, then start the instance again.

After the instance restarts, note the change in the public IP address, which confirms it now runs on a different physical host. SSH into the instance again and run:

```
lsblk
```

You will notice the instance store volume (nvme1n1) is still present. However, if you check the file system with:

```
sudo file -s /dev/nvme1n1
```

and list the contents of /instance-demo, you will find that the test file created earlier is missing. This confirms that data from the previous instance store has been lost due to the instance migration.

```
Warning
Do not use instance stores for storing critical data.
Always rely on persistent storage solutions like Amazon EBS
for data that must be retained.
```

Quick Reference Table

Command/Action	Description
lsblk	List block devices
sudo file -s /dev/nvme1n1	Check for a file system on the instance store volume
sudo mkfs -t xfs /dev/nvme1n1	Create an XFS file system on the instance store
sudo mkdir /instance-demo	Create a mount directory
sudo mount /dev/nvme1n1 /instance-demo/	Mount the instance store

df -k	Confirm the mount status
`echo "test"	sudo tee test`