Elastic Block Storage
Jem Camba - 2019-03-06 - in Storage

Elastic Block Storage (available in AMS1, EWR1, NRT1, & SJC1) is a multi-tenant storage service.

Product Basics
Packet’s Block Storage service looks and smells a lot like Amazon’s EBS. You can create volumes of different performance profiles, leverage snapshot policies and more - all via our API or portal. The service is highly redundant.

Comparing our block storage service to similar offerings at other clouds, an important difference is that at Packet you don’t have the benefit of a platform-managed hypervisor layer (we only offer bare metal compute and never leave an agent or any other tools/software on your server after it is deployed).

As such, there is a bit more you should know about how to troubleshoot or deal with any issues that may arise. Check out the tips below.

Performance Tiers
When you need persistent storage, with built-in replication and snapshots, our block storage product is a very useful tool. We offer two performance tiers:

- **Standard Tier** ($0.000104/GB per hour) - With 500 IOPS per volume this is good for backups, dev/test, and medium use datasets.
- **Premium Tier** ($0.000223/GB per hour) - With 15,000 IOPS per volume this is targeted at higher I/O heavy workloads.
**Use Cases**
While our Block Storage product is the perfect match for a wide variety of use cases, the first and most important step is to understand when *not* to use block storage at Packet.

As a general rule of thumb, databases are not a good fit for our Block Storage, especially those that are not resilient due to minor hiccups. While uncommon, even a small loss of connectivity to some databases can cause major issues.

If block storage doesn't meet your needs, we offer two other options:

1. Various local disk options (SSD's and NVMe Flash) on each server config.
2. The s1.large storage-focused server includes 24 TB of SATA w/ SSD's for cache.

**Deploying & Managing Block Volumes (Linux)**

**Creating a storage volume**
If you have not yet created a volume and click on the Storage view within your Project, you would see the following message:

![Oops, You Don't Have Any Volumes Yet!](image)

Packet's multi-tenant block storage service is mounted over our network via iSCSI, and is available in EWR, SJC, AMS, and NRT. You can choose from two performance tiers, and set schedules for snapshots. Learn more here.

To create a new volume, click on the 'Deploy Storage Volume' button. On the following screen, give the volume a description, choose the size of the volume in GB, and select the location, then performance tier. If you'd like to set up automated snapshots, you can do so via the optional snapshot settings button.
What are snapshots and how do they work?

A snapshot is a differential copy of your volume made at a specific moment in time. If you have a 500GB volume, but only 250 GB of data on it, your snapshot is only 250 GB. With snapshots, you are only billed for the changes to the original block device from the time the snapshot is made — as such, a policy of 5 snapshots may end up being only a small amount of storage if your rate of change on the device is low. Billing for all snapshots is at our lower $0.07/GB per month tier.

Attach / Detach the Volume to the Server

After your volume is created, you need to route it to the server you would like to mount it on. To do so, go to the server detail page that you’d like to connect the block device to and click on the Storage submenu on the left. Select your volume from the drop-down box and click ‘Attach’. This routes the volume to your block device and allows you to mount it from your local server via iSCSI.
Once attached to your server, you will see the volume name, size & where it is located:

![Attached Storage Volumes]

**Attach / Detach Scripts**

As Packet does not have an agent running on any client-server or remote access to client machines, you need to run the final attach of the iSCSI volume to your server on your own. Packet provides a block storage management script to make this process easy. You can access the attach script via Github [here](#).

In order to complete the attachment process, you will need to run the following on the device's CLI:

⚠️ **ARM users**: Please note, the scripts are built with x86 in mind, and do not currently include JQ for aarch64. As such, you will need to first manually install jq as follows:

```bash
apt-get install jq
```

All other users:

```bash
root@block:~# packet-block-storage-attach -m queue


create: volume-32829074  (36001405d589377db1a14b08913f04a14)  undef
DATERA,IBLOCK

size=1000G  features='1 queue_if_no_path'  hwhandler='1 alua'  wp=undef

`-+- policy='round-robin 0'  prio=1  status=undef

|- 10:0:0:0  sdc 8:32  undef  ready  running

`- 11:0:0:0  sdd 8:48  undef  ready  running

Block device /dev/mapper/volume-32829074 is available for use
The option `-m queue` is very important, if block storage becomes inaccessible, this directive will keep the IO in-memory buffer until it becomes accessible.

**Creating the filesystem with gdisk**

After mounting the block device, you need to partition the block device. You can do so using `gdisk`, as follows (replace the `volume_name_here` with the volume label outputted by the attach script above):

```
root@block:~# gdisk /dev/mapper/{volume_name_here}
```

You can use the default settings or adjust according to your particular use-case. After partitioning the disk you can verify the volume partition was successfully created as follows:

```
root@block:~# ls -al /dev/mapper/

```

If the new partition is not showing, please run:

```
kpartx -u /dev.mapper/volume-xxxxxx and verify again with ls -al /dev.mapper/*
```

**Create a filesystem on the volume**

Now that you’ve set up the partition, you must format the partition with a filesystem. For this example, we will make use of `ext4` although you are welcome to choose whatever filesystem you’d like:

```
root@block:~# mkfs.ext4 /dev/mapper/volume-32829074-part1
```

```
mke2fs 1.42.13 (17-May-2015)

Creating filesystem with 262143739 4k blocks and 65536000 inodes

Filesystem UUID: 20e0931a-255b-4bf7-8e22-5da91198279f

Superblock backups stored on blocks:
```
Allocating group tables: done

Writing inode tables: done

Creating journal (32768 blocks): done

Writing superblocks and filesystem accounting information: done

Mount the partition

Finally, we're ready to mount the formatted partition. To do so, issue:

```
root@block:~# mount -t ext4 /dev/mapper/volume-6a18e249-part1 /mnt/
```

If you run `fdisk -l` the new storage will be displayed

```
Disk /dev/mapper/volume-6a18e249: 161.1 GB, 161061273600 bytes
45 heads, 3 sectors/track, 2330168 cylinders, total 314572800 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 1048576 bytes
Disk identifier: 0x45d81206

Device Boot Start End Blocks Id System
/dev/mapper/volume-6a18e249-part1 2048 314572799 157285376 83 Linux
```

Remount storage should your device be rebooted

In order for the device to mount on boot, we need to add an entry to fstab. Instead of using the volume name (e.g. `/dev/mapper/volume-xxx name`), we need to grab the UUID of the volume device, as follows:
As you can see from the output above, the UUID in this case is 20e0931a-255b-4bf7-8e22-5da91198279f. You would then proceed to add an entry to the fstab in /etc/fstab, as follows:

```
echo 'UUID=20e0931a-255b-4bf7-8e22-5da91198279f /mnt ext4 _netdev 0 0' | tee -a /etc/fstab
```

⚠️ **Debian Users**: A known issue when utilizing the UUID & rebooting the device will cause the multipath connection to fail. Instead of UUID please use the volume name in fstab:

```
/dev/mapper/volume-13a7e577-part1 /storage ext4 _netdev 0 0
```

### Detach / Delete Storage Volume

If you want to detach and delete a storage volume, or simply move it to a new server you should:

- Detach it via cli
  ```
  root@Ubuntu-block-storage:~# umount /dev/mapper/volume-xxxxx-part1
  ```
- Once the unmount is confirmed, run the block storage detach script
  ```
  packet-block-storage-detach
  ```
- Then, detach the device from the server in the portal: Device View > Storage > Manage > Detach
- Optionally, delete the block device from portal: Project View > Storage > Options > Delete

### Tips & Tricks

Our Block Storage service looks and smells a lot like Amazon’s EBS. One important difference is that when you connect a block storage volume on Packet to a server on Packet, you don’t have the benefit of a hypervisor layer. As such, there is a bit more you should know about how to troubleshoot or deal with any issues that may arise.

While our Block Storage product is the perfect match for a wide variety of use cases, the first and most important step is to understand when *not* to use block storage at Packet.

As a general rule of thumb, databases are not a good fit for our Block Storage, especially
those that are not resilient due to minor hiccups. While uncommon, even a small loss of connectivity to some databases can cause major issues.

**Redundancy**

Each installation of our Block Storage product is made up of three physical servers, each with redundant network and power. In order to support data redundancy, each volume you create is replicated twice on the storage cluster.

The following commands will show more information about these paths:

```
root@block-storage:~# iscsiadm -m session


```

From this output, you can see the two iSCSI target IPs. If you are able to ping them fine, that means there is good network connectivity to the volume. Next, we recommend that you check DM-multipath with the following command:

```
root@block-storage:~# multipath -ll

volume-9674e518 (360014057ee4d5efb13a4c33a24ba765f) dm-0 DATERA,IBLOCK
size=200G features='1 queue_if_no_path' hwhandler='1 alua' wp=rw

`-+- policy='round-robin 0' prio=1 status=active

|- 9:0:0:0 sdc 8:32 active ready running

`- 8:0:0:0 sdd 8:48 active ready running
```

**Resiliency**

Using the `-m queue` option when attaching a volume is recommended since, during instances of very brief network blips, the server will keep IO in a memory buffer until the volume is reachable again. This greatly enhances resiliency and will prevent the volume from going into read-only mode.

Although we do our best to prevent downtime, it would be a lie to say that network attached anything will have a 100% uptime, forever. There will be cases when a volume will
go into read-only mode, such as if there is a longer network interruption, or there are issues with the storage cluster itself (volumes go read-only in order to protect any information that is there against possible corruption).

The best way to bring a volume back in this scenario is to detach it with the following command (you can specify which volume to detach by optionally adding the volume's ID):

```
packet-block-storage-detach {volume-id}
```

After a few seconds, reattach the volume with:

```
packet-block-storage-attach -v -m queue {volume-id}
```

The Verbose option will show any possible issues during the attach process. If the attach command is not successful, or if it shows any problems, please contact us.

**Tags**

- block storage