Creating and Mounting EBS Volumes on Linux

Many cloud providers offer virtual disks which can be attached to a cloud server similar to the way you might mount an HDD or SSD into your computer. These are basically virtualized partitions of a data center's storage unit which can be reallocated and re-sized as needed.

At Arcane, we separate project assets into different detachable disks (aka volumes). As an example, on a server with multiple websites, we will put each site on its own volume. We might even mount ancillary dependent assets like extensions or plugins into a subdirectory mountpoint. This has a lot of benefits:

- Makes it easier for us to backup and restore specific assets so we can keep the rest of our data intact.
- Allows for easier migration and scalability of these different assets.

We also separate filesystem data from application data to make it easier to migrate, scale, and **protect potential write failures (due to maxing out a volume) from impacting other assets.**

- In a traditional 1-volume setup, should you max our your disk due to file transfers, your kernel would also fail, thereby crashing your server.
- With your application/file storage separate from your kernel/file system data, if you were to max out your file storage, your kernel would remain operational and you could scale up your file storage volume in real-time.

Here's how you do this at AWS on a Linux kernel:

1. Within the AWS Console, visit the “Volumes” panel from the left navigation. Create a volume using the availability zone of an instance you’ve created.
2. Right-click on the volume and attach it to the instance you want it mounted to. When attaching, AWS will ask you for a device name such as /dev/xvdf which you can edit or leave by default. We recommend the default as AWS will ensure this is unique in order (ie. 1st = xvdf, 2nd = xvdg)
3. Now log into the server and decide where you want to mount your volume (this might be /var/www/html for example).
    a. To create a new fodler, run **$ mkdir PATH/TO/YOUR_DIRECTORY**
4. To see your newly created volume from step 2, run: **$ lsblk**
    a. You should see your volumes and their corresponding mount points including the one you set up in step 2.
5. Verify filesystem: This will make your volume from step 2 ready to receive data. Run the following with the correct name from step 3: **$file -s /dev/[xvdf]**
    a. If the response from step 5 indicates no filessytem, you can create it like this: $**mkfs -t xfs /dev/[xvdf]**
6. Now mount your volume to the mountpoint/directory: **$ mount /dev/[xvdf] /[directory]**

Creating and Mounting EBS Volumes on Linux

Now we will want to make sure that mount is preserved upon reboot. This is done via fstab.

1. Get the volume's UUID: **$ blkid**
    a. **You should be able to identify your volume from the steps above. The ID will look something like this 1a11c-1111-11d11-111-11111h1h1h11a**

2. Edit fstab: **$ vi /etc/fstab**
    a. Add/edit filemount with UUID to the bottom of the file. Make sure it contains "nofail", otherwise your server will not boot up in the event of a failed mount (which does happen).
    b. It should look something like this: UUID=1a11c-1111-11d11-111-11111h1h1h11a /PATH/TO/YOUR_DIRECTORY xfs defaults,nofail 0 2

3. Now test your work by unmounting (reverting your earlier work) and re-mounting with fstab's automated process:
    a. Unmount: **$ umount**
    b. Re-mount from fstab: **$ mount -a**

support@arcanestrategies.com
(541) 649-2508

Arcane Strategies
7 E. Congress St. #804
Savannah, GA 31401