Tutorial 3 – Best Practices for Working
with Virtual Machines on Amazon EC2

Disclaimer: Subject to updates as corrections are found
Version 0.10
Scoring: 20 pts maximum
** *This tutorial can optionally be completed in two-person teams.* **

The purpose of this tutorial is to introduce the Amazon Elastic Compute Cloud (EC2) service, an Infrastructure-as-a-Service platform that provides virtual machines (i.e. VMs) on demand, and discuss best practices for cost saving for using VMs on Amazon EC2.

This tutorial will focus on the use of spot instances, VMs which are available at a reduced price by trading-off availability guarantees to better leverage excess capacity of particular AWS cloud regions. We will also discuss best practices for data storage when working with EC2 VMs including the use of Elastic Block Store (EBS) persistent disk volumes, as well as local ephemeral disks. We will discuss ways to create, suspend (pause), and terminate VMs to eliminate unnecessary VM uptime so that charges are for when VMs are actually used to save costs. To complete this tutorial, the use of Amazon Cloud Credits is required as creating EC2 spot instances is not free.

Instance Types

Amazon EC2 instances (also called virtual machines or VMs) have a letter designating the family, and a number identifying the generation. Presently there are 7 generations (1-7). Most families do not have VM types spanning all 7 generations of instances.

CPU Instances combine a family letter and generation number followed by a period and a size (small, large, xlarge, 2xlarge, 4xlarge, 8xlarge, etc.) The sizes correspond to the same configuration, but an increasing quantity of available resources (e.g. CPU cores, memory, network capacity, storage capacity). The most common instance families include:

Commonly used EC2 instance families:
c	Compute Optimized Instances	Typically fast CPUs, but less memory
m	General Purpose Instances	More memory than C-family, slower CPUs
r	High Memory Instances	More memory than C or M
t	Burstable instances (cpu-time limited)	Lower-cost instances with CPU quotas

Less common families include:
|		
f	FPGA Instances	Instances with an on-board programmable FPGA
h/hs	Storage-optimized instances	Instances with enhanced disk capacity (HDDs)
i	Storage-optimized instances	Instances with enhanced disk capacity (SSDs)
p/g	GPU Instances	Instances with on-board GPU(s)
x	Extra-high memory	Instances with extreme memory

Note on how the CPUs supporting EC2 VMs are evolving:
Beginning with 5th generation instances both Intel, AMD, and ARM64-based processors became available on AWS. Starting with the 6th generation of instances AWS adopted a new naming convention where the first letter after the number indicates the CPU architecture type. For 6th and 7th generation instances "a" is for AMD, "i" is for intel, and "g" is for Amazon's proprietary ARM64 Graviton processors which are now in their 3rd release (for example a1, c6g, and c7g). The Amazon Graviton3 processor featured on c7g, hpc7g, m7g, and r7g instances is an ARM processor which stands for Advanced RISC Machine. RISC stands for "Reduced Instruction Set Computing". This CPU architecture is different than that used by classic Intel Complex Instruction Set Computing (CISC) processors known as x86. **A key detail is that software for Intel (x86) may not run on ARM directly without recompilation or emulation.** Compilers (e.g. C/C++, etc.) create machine specific code for the target processor. Linux applications generally are quite portable from x86 to ARM through recompilation. Windows 10/11 is not supported on ARM processors, but applications that haven't been compiled may only run through 32-bit emulation.

The website **https://instances.vantage.sh/** (formerly **https://www.ec2instances.info/**) is an excellent tool to query/search the various types of VM instances available from EC2. On this page, check out the "Columns" drop-down list which is used to customize meta-data returned about instance types. It is possible to examine the **Physical Processor** and **Clock Speed (GHz)** of instances. The processor type and clock speed are key parameters that impact a VM's performance. The website summarizes Linux/Windows on-demand, reserved, and spot pricing. On demand pricing is the standard price offered for ad-hoc usage. Reserved pricing requires a long-term usage contract where they buyer commits to a minimum run-time. Here, the customer is billed 24/7 for reserved instances even if they are not active. There is a minimum contract commitment time. Recently AWS began offering a Reserved Instance Marketplace where users can resell unused reserved instance time.

It is recommended to complete the tutorial using a web browser from the same operating system as your Putty or SSH client (e.g. Ubuntu terminal). This will make it easy to copy-and-paste between the browser and terminal.

Starting Out

Log into the AWS Management console, and select the **EC2** service:
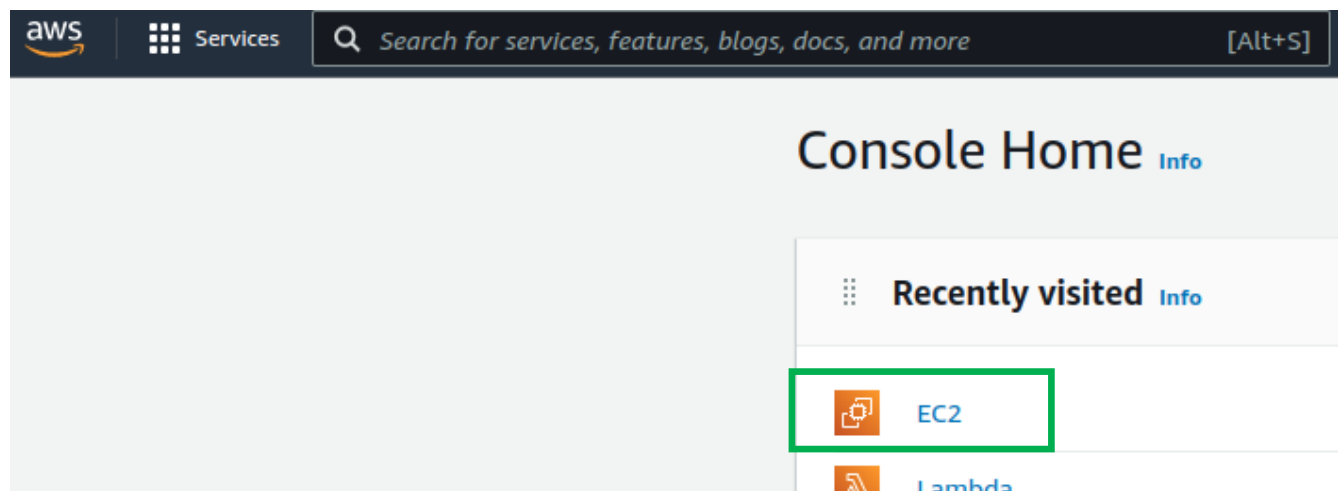
Regions and Availability Zones

The United States has four distinct AWS regions. Each region consists of at least three distinct availability zones. Think of an availability zone as a separate physical data center. Think of a physically separate facility that requires travel (e.g. by car) to reach. Availability Zones are regionally local, but it may take from ~ 10 to 60 minutes or more to travel between the separate facilities. Availability Zones have separate power sources, and are ideally designed to minimize the likelihood that all zones within a Region can fail at the same time. Data centers may also be located in areas with low cost and highly reliable electricity.

Satellite view of one of the availability zones in the us-east-2 region near Boardman, Oregon:

US Regions:

Name	Location	Notes
us-east-1	Northern Virginia	The first AWS region
us-east-2	Ohio	Currently the least expensive
us-west-1	Northern California	Most expensive US region
us-west-2	Oregon	Boardman and Umatilla Oregon

It is recommended to use the **us-east-2 Ohio** region for cloud resources throughout TCSS 462/562. Currently (Fall 2023) the Ohio region has some of the lowest prices for on-demand and spot VM instances. Consolidating resources in one region reduces duplication of data and also the likelihood of accidentally creating resources (e.g. VMs) in another region and forgetting to terminate them resulting in accidental charges. While network latency, the time it takes for network traffic to travel to us-east-2 (Ohio) and back to Washington state will be slightly longer, but the costs for using resources will be lower. One way to mitigate this latency is by launching VMs to serve as service clients in the availability zone.

The AWS region can be changed using the dropdown list in the upper right-hand corner of the GUI. Select "**Ohio**".

Spot Instances

Spot instances provide savings vs. on-demand VM instances. Amazon sells VMs at reduced "spot" market prices to sell unused cloud capacity. Pricing is generally only 1/4 to 1/2 of full price. The caveat is that when demand spikes for a particular VM type in a specific region or availability zone, instances will be automatically terminated when the market price (e.g. going rate) exceeds the customer's bid price (called a spot price). There is a 2-minute warning provided to users when this occurs which can be checked for.
(see article: https://aws.amazon.com/blogs/aws/new-ec2-spot-instance-termination-notices/)

In nearly all cases, **spot instances** should be used for course work/research.

Cost Saving Measure #1

ALWAYS USE SPOT INSTANCES FOR COURSE/RESEARCH RELATED PROJECTS

VM Disk Storage

Every VM needs a disk. The disk volume where the operating system is installed is called the ROOT volume. There are two primary types of disk volumes on EC2: **Elastic Block Store (EBS)** and **Instance Store**. EBS disk volumes are hosted by remote servers that are attached to Amazon EC2 virtual machines using a very high-speed network connection. EBS volumes are replicated automatically behind the scenes to provide data redundancy and to improve Input/Output (I/O) performance. Instance-store volumes are hosted using local disks (HDD, SSD) directly attached to the physical server hosting your VM. This was the predominant type of disk on EC2 for 1st and 2nd generation instances. Starting with 3rd generation instances, EBS volumes became more prolific/common. Starting with 4th generation instances, *only* EBS volumes are allowed to serve as the ROOT volume on a VM. This has important cost implications. With an instance store volume, there is no additional storage cost for the VM. Because this feature is now only available with legacy instances (3rd generation and before) it is no longer a best practice / cost savings measures.

With EBS volumes, in addition to standard VM costs, every VM incurs a 24/7 storage charge for the EBS disk volume. This results in a hidden-cost of cloud computing. For every VM you pay twice. Once for the VM, and again for the disk. The extra cost is "justified" because EBS volumes are "persistent" in that they can be RETAINED/KEPT to preserve all data when the VM is shutdown intentionally or accidentally. The additional cost is 8c/GB/month (general purpose 3 type) or 10c/GB/month (general purpose 2 type). The standard EBS ROOT volume size is currently 8GB for Ubuntu Linux. This results in an additional charge of 64 cents per month. With larger ROOT volumes, the additional cost can become substantial. For example a 800 GB ROOT volume will add an additional ~9 cents per hour to the cost for the VM. Depending on the instance type, this can be more than the actual VM cost !!

You can attach multiple EBS volumes to a VM. C6i supports up to 28 volumes, c7i now supports 128 volumes. That's a lot of disks!

WARNING ABOUT SAVING EBS VOLUMES

A common accident that results in unexpected cloud computing charges, is to create a VM, and mark the VM to have a persistent EBS volume that does not get deleted on termination. When the volume is large (e.g. 800 GB), this results in a charge of $64/month. Think of an EBS VOLUME as a LIVE PIECE of HARDWARE. When you have one in your account it can be attached immediately to _any VM_ with no preparation/initialization. This "LIVE" behavior justifies the high price. Servers in the cloud are actively preserving your data in a LIVE-STATE so that it is available in a split-second for connection to a new VM. The key message here to save cloud computing costs is:

Cost Saving Measure #2:

NEVER LEAVE AN EBS VOLUME IN YOUR ACCOUNT
THAT IS NOT ATTACHED TO A RUNNING VM

If you want to retain the data on an EBS volume once a VM is shutdown, **make a snapshot**. Standard snapshots storage is 5c/GB/month, but snapshots are compressed and stored using S3 to save even more. For more information regarding EBS snapshots see:
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSSnapshots.html

Starting with 5th generation instances, EC2 offered "d" versions. The "d" indicates a local ephemeral (which means temporary) disk is available to the VM. These ephemeral local disks are not backed up when the VM terminates. All data is lost unless the user writes code/scripts to back up the data. A huge advantage of the "d" instances though is that this local storage is very inexpensive. For spot instances in the Viriginia region, in some cases it is FREE! The spot prices for c6id.2xlarge instances were cheaper in October 2023 (us-east-1d) than c6i.2xlarge. This means, the VM with a local 474 GB ephemeral solid state disk (SSD) (c6id.2xlarge) was going for less than the VM without one ! (19.39c/hour vs. 23.05c/hour). Having a local ephemeral disk is good for applications that need access to large amounts of local, fast disk space. Also this ephemeral storage can be free in some cases. You don't need to pay the EBS costs of 8c/GB/month for it.

Here is a feature comparison of instance store vs. EBS volumes:

Instance Store Volumes	Elastic Block Store (EBS) Volumes
- Disk hosted local to VM (on same physical server) - Provides cheap temporary data storage to VM - Non-persistent volume (no backup) - No automatic replication of data - Supports both VMs run in paravirtual mode (pv) as well as with full virtualization (hvm) mode. - 3 types of disks: HDD (old), SSD, and NVMe SSD - I/O operations per second (IOPS) based on HW capabilities and sharing	- Data is persistent and saved when VM terminates - 99.999% availability - Automatic replication within availability zone - Snapshot support - Can modify volume size as needed (vertical scaling) - SSD or HDD - Auto recovery - common types: gp3, gp2, io2, io1 - scalable IOPS with i01/io2 type (I/O operations per second) - default ROOT volume size is based on the OS snapshot used – typically 8 GB for Linux - when used for ROOT volume, virtualization type is full virtualization (hvm) mode

1. **Launch an EBS-backed Amazon EC2 Ubuntu 22.04 instance**
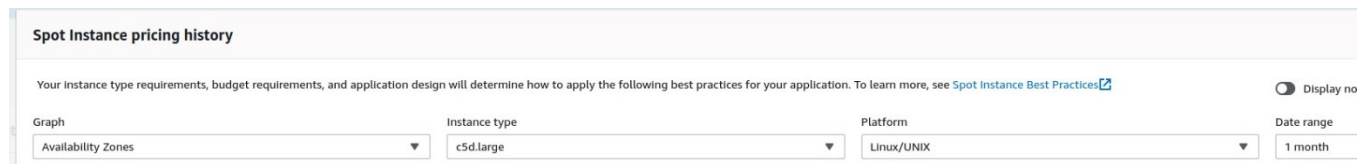
From the EC2 Dashboard.
Click on the "Spot Requests" option on the left-hand menu:
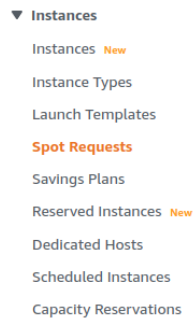
Before launching an auction-based "spot instance", check the going rate for VMs in your region. Select the "Pricing History" button:

Filter the graph as follows:

Write down the least expensive "availability zone" within the Ohio region (they may all be the same).
The options include: us-east-2a, us-east-2b, us-east-2c, and us-east-2d. The price shown is the price per hour to rent a c5d.large VM.

Next, compare the pricing with "c5.large" VMs. The c5.large VMs do not offer a local instance store nVME SSD disk. Also, try looking at "c6i.large" and "c6id.large". These are the new Intel-based VMs that are using 10 nanometer based Intel Xeon CPUs.

You can further check the capabilities of the c5d.large VM using this website:
https://www.ec2instances.info/?filter=c5d

Adjust or remove the filter to inspect other types of VMs.

Next, let's launch a c5d.large spot instance.
Escape out of the price graph by pressing the 'X' in the window's upper-right hand corner.
On the menu on the far-left click "**Instances**" that is under Instances:

▼ Instances

Instances

Try launching a spot instance using the normal ec2 instance launch wizard.
Press the "Launch Instances" button:

Launch instances ▼

Name and tags
At the top of the user interface under **Name and tags**, a VM name and tags can be specified.
Tags allow key/value assignments to help identify VMs.
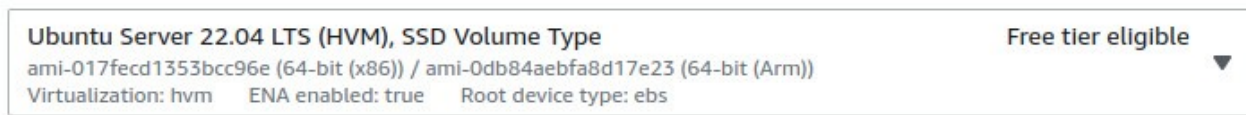These key/value pairs can be queried using the EC2 API to programmatically identify specific VMs that have a designated purpose for your application.
We will not use this feature right now.

Application and OS Images (Amazon Machine Images)
For the **Application and OS images (Amazon Machine Image)** select the **Ubuntu** button.
Then from the **Amazon Machine Image (AMI)** dropdown list,
Select Ubuntu 22.04 LTS (HVM) as the Amazon Machine Image.

Ubuntu Server 22.04 LTS (HVM), SSD Volume Type   Free tier eligible
ami-017fecd1353bcc96e (64-bit (x86)) / ami-0db84aebfa8d17e23 (64-bit (Arm))   ▼
Virtualization: hvm   ENA enabled: true   Root device type: ebs
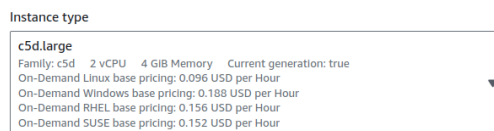
Amazon Machine Images are snapshots of pre-installed machines. Choosing an AMI will cause its snapshot to be replicated onto your live EBS volume. Choosing an AMI is how you select the operating system for the VM. It is possible to create your own AMI with any operating system you'd like, but in most cases preexisting AMIs are available for most OSes.

Instance Type
Next, specify the **Instance Type**.
In the dropdown, search for and select "c5d.large". 'c' is from the "Compute optimized" family of ec2 instances. Note how VM hosting using commercial operating systems such as Windows and RedHat Linux is more expensive due to licensing costs.

Instance type
c5d.large
Family: c5d   2 vCPU   4 GiB Memory   Current generation: true
On-Demand Linux base pricing: 0.096 USD per Hour
On-Demand Windows base pricing: 0.188 USD per Hour   ▼
On-Demand RHEL base pricing: 0.156 USD per Hour
On-Demand SUSE base pricing: 0.152 USD per Hour

Key pair (login)
If you already have a key pair, then select it from the dropdown list. If not, select "Create new key pair". When creating a new key pair, using the defaults is ok. Download the file to a safe location.

Network Settings
Next we will make changes to the network settings for your VM launch request.
To make changes, click on the **Edit** button on the right-hand side of the Network Settings frame.

Here we will specify the Virtual Private Cloud (VPC) that the VM is created on. Every VM has a VPC. **Most of the time we use the default VPC.** Make sure the (default) is selected for the VPC. Creating a custom VPC allows for specialized network configurations.

Subnet:
This is how you select a specific availability zone. Each subnet has an availability zone. This is the physical data center in the region where the VM will be created. On the right-side of each subnet, see where it says, for example: "Availability Zone: us-east-2b". Each availability zone has different spot pricing. This is what we inspected on page 6. If you don't know the least expensive availability zone for the "c5d.large" instance type in us-east-2 (Ohio), go back to page 6, and check again. Write down the least expensive availability zone for "c5d.large". It is important to know how to find the lowest spot instance price, and then how to create the spot instance. Earlier, AWS showed spot pricing in the **Launch Instance** GUI, but this functionality has been removed. It now requires two steps. Keep in mind it is possible to implement this cost optimization in code by checking pricing, and creating the cheapest instance using the AWS SDKs/APIs.

Cost Savings Measure #3:

**SET THE SPOT INSTANCE SUBNET TO MATCH
THE LEAST EXPENSIVE AVAILABILITY ZONE**

For example, if the lowest spot instance price is for the us-east-2b availability zone, then select the "2b" subnet:

Auto-assign Public IP:
Subnets have a default setting whether a VM should receive a public IP address.
The property is called "Auto-assign public IPv4 address". Be sure this is set to Enable:

Next under **Firewall (security group):**
Select the radio button: "Select existing security group".

And select the **default** security group from the list of choices.

In your default security group, you can add common firewall access rules to enable rapid access to newly created VMs. When selecting the security group, firewall rules are displayed. They can not be changed here. They can be modified later using the Security Group GUI.

Next, configure Storage options.

By default under "Configure storage", c5d.large will by default set the ROOT (EBS) volume to have 8GB, using the gp2 EBS volume type.

We can identify an EBS volume based on the Volume Type. Some available types for EBS include general purpose (gp2 and gp3), provisioned SSD (io1), provisioned SSD (io2), and magnetic (which is fact no longer "standard" by the way). These can be seen in the drop down.

>>> FOR THIS TUTORIAL, PLEASE USE the default, gp2 – General Purpose 2 as the EBS volume type.

>>> For this tutorial, we will need to increase the size of the ROOT volume from 8 GB to 15 GB. *
This additional disk space, will be needed later on in the tutorial.

Note for shrinking a volume, it may not be possible to reduce the volume below 8GB. This is because the originating Ubuntu snapshot (e.g. snap-0b440a54993a44c36) expects at least 8 GB.

Next, inspect the ephemeral disks. Ephemeral (temporary local) disks can be viewed by clicking "show details" for "Instance store volumes". Then expand "volume 2":

9
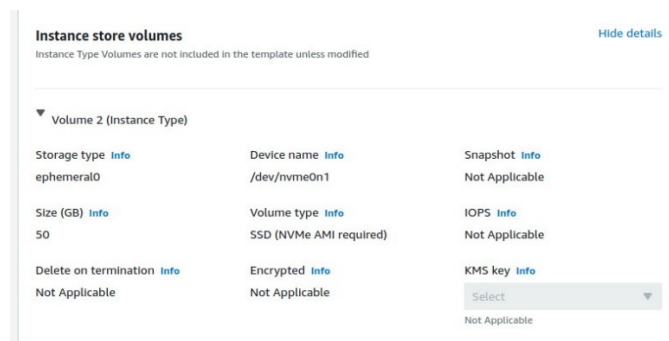
Ephemeral literally means "lasting for a very short time". The data on ephemeral disks is not persisted. In computing, ephemeral refers to temporary, non-persisted resources (i.e. with no backup).

How to request a spot instance:
Next, to specify creation of a spot instance, expand "**Advanced Details**". Then check to box for "**Request Spot Instances**" and click the "**Customize**" link to the right to obtain the following GUI:

Specify a "Maximum price" of "0.15" for 15 cents.

For "**Request type**", specify "One-time", as opposed to a persistent request.

When the request type is a "persistent request", when the spot VM is terminated, either voluntary, or accidentally, the VM will always automatically RECREATE itself!
This leads to the next cost savings measure:

Cost Saving Measure #4:

<div align="center">

BE CAREFUL USING PERSISTENT REQUESTS FOR SPOT INSTANCES

</div>

When using persistent spot requests, when you intentionally terminate the VM, the VM will automatically be recreated within 1-2 minutes resulting in ongoing charges !!! Imagine, you shut your computer down for the weekend, only to learn that the VM came back to life, and charged for multiple days of inactivity !! When using Persistent Spot Requests, it is necessary to physically <mark>DELETE THE SPOT REQUEST</mark> in addition to terminating the instance. Failure to do so, will result in ADDITIONAL CHARGES ! **BE WARNED !**

For the "**Interruption behavior**" it is typical to select "Terminate". The "Stop" option for a spot instance requires a "Persistent" request type.

Placement group

Next, scroll down in the Advanced Details to "**Placement group**".

Placement groups allow for strategic placement of multiple VMs across physical servers. This feature only pertains when you're launching more than 1 VM. This is important for distributed computing tasks where 5 or more VMs are needed.

To use this feature use must first "Create new placement group".

The table below summarizes the options, and their corresponding implications for distributed computing:

Placement group strategy	Implications for distributed computing
Cluster	Groups VMs as close together on the same HW as possible. Useful when low network latency between VMs is important.
Spread	Spreads VMs across physically separate racks/HW. Limited to 7 VMs per availability zone. Important when running many instances of the same kind of work at the same time. This is ideal for MapReduce clusters.
Partition	Launches VMs into distinct partitions, where each partition consists of servers very close together having low latency. This is helpful if an application has distinct components that need to be scaled and hosted separately where individual nodes of an application tier need low network latency.

For this VM, do not configure a placement group.

Next, Review the details of the configured instance in the Summary window pane to the right.
This allows a check to see that all settings are acceptable prior to creating the VM.
If everything looks okay, press the "Launch Instance" button:

Launch instance

If this is the first time you've created a VM, you should have created a new key pair which is used to establish a secure shell (ssh) connection to your new VM.

2. **Log into your Amazon EC2 Spot Instance**

Select "Instances" on the left-hand side of the screen.

Locate your newly launched VM. It should be the only one!
Select your VM on the left:

☑	–	i-0c200e3fbfb06305c	⊘ Running ⊕⊖	c5d.large	⊘ 2/2 checks …	No alarms +	us-east-2a	ec2-3-129-217-163.us…	3.129.217.163

Then, below look for "Public IPv4 address" and select the copy-icon just to the LEFT of the IP address:

Public IPv4 address

3.129.217.163 |

This copies the IP address to the clipboard.
Before connecting to the VM, configure the security group to allow SSH access.

Click on the "Security" tab of the instance.

Click on the security group ID:

Security groups

sg-████████ (default)

On the lower screen, with the "**Inbound rules**" tab selected, on the far right press the button:

Edit inbound rules

At the bottom of the list, press the button:

Add rule

Configure a new rule as follows:

Type/select: "SSH", and for the drop-down list, select: "My IP"

| SSH ▼ | TCP | 22 | My IP ▼ |

Another recommended Security Group permission to add is the "All ICMP – IPv4" permission. This enables you to "ping" your VM. Configure as follows:

| All ICMP - IPv4 ▼ | ICMP | All | My IP ▼ |

For every new VM that is launched, when you associate the default security group, these security group settings will be applied. Over time you can accumulate rules in your security group for popular places where you use your laptop. This could be places like your home, the local Starbucks, and the UW-Tacoma campus.

As an alternative to selecting "MyIP" you can specify a wider subnetwork address range. This way if your computer receives different IP addresses within the same CIDR (address) block, you don't have to update the security group as often. (e.g. every day) CIDR block stands for classless-inter-domain routing. A CIDR block is a group of IP addresses the form a "sub-network". Within your house, all of the devices sharing your WiFi

connection usually receive private IP addresses sharing the same CIDR-block, where all addresses are private (not public IPs).

Find out your IP address.
In your web browser, open Google search, and type in "What is my IP?".

Your IP address should appear.
Note the first 3 numbers.
Let's add SSH permission for your CIDR network block.

If your IP is for example **120.118.53.108**, then your 24-bit CIDR block which would include all 255 public internet addresses on the subnet would be 120.118.53.0/24. At UW some addresses may be public internet addresses, others will be private. Check if the reported IP from your browser matches the configred IP address in your Linux environment using the "**ifconfig**" command. If this command is not installed in your Ubuntu environment, it can be installed with "**sudo apt install net-tools**". You will need to recognize the network device name. The wireless device name could be "wlp0s2f3". It won't be "lo". Look for the "inet". Does it match what the browser says? If no, then your IP address is not public on the internet, but your traffic is routed. Check the routing with "**route -n**". The internet gateway will be the Gateway listed for the "0.0.0.0" destination. "0.0.0.0" is a wildcard meaning every address.

As an alternative to MyIP, in the security group, your entire CIDR block, e.g. 120.118.53.0/24 can be added, but if your gateway is not the address reported when checking "what is my IP" in a google search, then this may not be helpful. Instead, inspect "what is my IP" from day to day to see if it ever changes, and create a CIDR block to capture the range of observed addresses.

The motivation for adding a CIDR block is that if you commonly use a network where traffic appears from a range of public gateways then you may receive various addresses in the same block from day-to-day. If you add a range vs. an individual IPs this may allow VM connectivity without reconfiguring the security group as often.

An even better way to set the network address range is to ask your network administrator what the wireless address range is. The range of dynamically assigned IPs from a given wireless network may exceed 255. In this case you may want to specify '/21' which would allow up to 4096 unique addresses and not the 256 associated with '/24'.

Once complete, save changes:

Save rules

Return to the list of Instances, by clicking the "Instances" option on the left-hand menu.

Now navigate to a Linux terminal.
If using Windows without a Linux environment, a 3rd-party program like PuTTY is required.
To learn more, see this tutorial:
https://www.ssh.com/ssh/putty/windows/install
Alternatively the Windows Subsystem for Linux can be used to use Linux commands like "ssh".

On Ubuntu, open a terminal, and navigate to the subdirectory using "cd" where you have stored your key using the absolute path. For example: "/home/username/awskeys/"

First try pinging your VM

```
ping 3.129.217.163
```

Use CTRL-C to stop the ping:

```
PING 3.129.217.163 (3.129.217.163) 56(84) bytes of data.
64 bytes from 3.129.217.163: icmp_seq=1 ttl=33 time=70.3 ms
64 bytes from 3.129.217.163: icmp_seq=2 ttl=33 time=70.7 ms
64 bytes from 3.129.217.163: icmp_seq=3 ttl=33 time=72.3 ms
64 bytes from 3.129.217.163: icmp_seq=4 ttl=33 time=70.5 ms
```

Pinging provides a rough estimate of the network latency between your computer, and the VM. Here the ping sends a 64-byte packet to the cloud VM. The cloud responds and the client measures the round-trip response time.

Ping-time to the VM is a good way to test your network's ability to access cloud resources. This also measures the round-trip **network latency**. Network latency is the time it takes for your network packet to travel to the cloud server and back. Using cloud resources that are too far away will result in higher network latency. Resources in the south Pacific may have several times the network latency of those in the continental U.S. A VM in Oregon should have a lower ping time than Ohio, Virginia, or a VM in a different continent.

Now launch an ssh session as follows:

<replace with your VM's IP>
ssh –i mykey.pem <u>ubuntu@3.129.217.163</u>

or if not in the same directory as the key:

<replace with your VM's IP>
ssh –i /home/username/awskeys/mykey.pem <u>ubuntu@3.129.217.163</u>

You will then be prompted, because your machine doesn't recognize the MAC address of the host, say 'YES'; to save the HW address and proceed with the connection:

```
The authenticity of host '3.129.217.163 (3.129.217.163)' can't be established.
ECDSA key fingerprint is SHA256:dhfCG+k/Zz7p13d39cAIiGfTCKW0zTHwtLTdoJQspp4.
Are you sure you want to continue connecting (yes/no/[fingerprint])? Yes
```

You may receive the following error:

```
Warning: Permanently added '3.129.217.163' (ECDSA) to the list of known hosts.
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@           WARNING: UNPROTECTED PRIVATE KEY FILE!          @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Permissions 0664 for 'mykey.pem' are too open.
```

```
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
Load key "mykey.pem": bad permissions
Permission denied (publickey).
```

This indicates that the ssh key permissions are too open.
Adjust with the following command:

```
chmod 0600 mykey.pem
```

or

```
chmod 0600 /home/username/awskeys/mykey.pm
```

After adjusting the key permissions, log into the VM again.

Inspect various aspects of your newly launched VM with the following commands:

Operation	Command
Inspect CPU	`lscpu`
Inspect memory	`free -h`
Inspect the disks	`df -Th`
Inspect the OS kernel	`uname -a`
Inspect the OS version	`cat /etc/os-release`

3. Preparing ephemeral disk volumes

By default, ephemeral disk volumes for c5d instances are not pre-initialized on launch.

The "lsblk" command describes block devices on Linux.
Check your VM's block devices:

```
$ lsblk
NAME            MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
loop0             7:0     0 25.1M  1 loop /snap/amazon-ssm-agent/5656
loop1             7:1     0 55.6M  1 loop /snap/core18/2560
loop2             7:2     0 63.2M  1 loop /snap/core20/1623
loop3             7:3     0   47M  1 loop /snap/snapd/16292
loop4             7:4     0  103M  1 loop /snap/lxd/23541
nvme1n1         259:0     0 46.6G  0 disk
nvme0n1         259:1     0    8G  0 disk
├─nvme0n1p1     259:2     0  7.9G  0 part /
├─nvme0n1p14    259:3     0    4M  0 part
└─nvme0n1p15    259:4     0  106M  0 part /boot/efi
```

In Linux, disks are assigned block device names. VMs have an odd convention where on EC2 block names for disks share the same letters, but have different numbers. For the c5d.large, the EBS volume block device name is therefore "nvme0n1". This is the Linux physical device name. Disks are organized into partitions, where each partition can be formatted (i.e. organized) using a different file system. The file system provides a catalog

allowing the operating system to quickly store and retrieve files. The most common for Ubuntu is currently "ext4".

Filesystem types are seen using the "df" command with the "T" option.
Filesystems use inodes, which are file records, to store and retrieve individual files. If a disk runs out of inodes it will no longer be able to store additional files even if there is space remaining on the disk. Inodes can be displayed with the "df" command using the "i" option.

Inspect your two devices with the "sudo fdisk -l" command:

sudo fdisk -l /dev/nvme0n1

sudo fdisk -l /dev/nvme1n1

Make a note one which device is the EBS disk and which is the instance store volume. It should be possible to identify the ephemeral disk as the one that is closer to 50 GB in size.

Using the device name for the ephemeral instance store volume, execute the following command sequence.

Note these commands could be used in a script to auto-initialize the VM at boot time:

IMPORTANT: BEFORE USING THESE COMMANDS IT IS ESSENTIAL TO VERIFY THAT YOU ARE PERFORMING THEM ON THE NVME INSTANCE STORAGE DISK. SWAPPING THEM WILL DELETE ALL DATA ON YOUR ROOT DISK AND WILL RENDER YOUR VM INOPERABLE

Verify the proper disk.
Check the "Disk model"

```
$ sudo fdisk -l /dev/nvme0n1
Disk /dev/nvme0n1: 8 GiB, 8589934592 bytes, 16777216 sectors
Disk model: Amazon Elastic Block Store
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes
Disklabel type: gpt
Disk identifier: AB2BB3CE-C07A-4946-9DF4-AA54E872DA24

Device              Start        End   Sectors   Size Type
/dev/nvme0n1p1    227328  16777182  16549855   7.9G Linux filesystem
/dev/nvme0n1p14     2048     10239      8192     4M BIOS boot
/dev/nvme0n1p15    10240    227327    217088   106M EFI System

Partition table entries are not in disk order.
ubuntu@ip-172-31-3-121:~$ sudo fdisk -l /dev/nvme1n1
Disk /dev/nvme1n1: 46.57 GiB, 50000000000 bytes, 97656250 sectors
Disk model: Amazon EC2 NVMe Instance Storage
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

Once verifying that you have the correct disk:

```
#assign a label to the disk
sudo parted -s /dev/nvme1n1 mklabel GPT

#create the disk partition, use all the space
sudo parted -s /dev/nvme1n1 mkpart primary ext4 2048s 100%

#format the new partition with the ext4 file system
sudo mkfs.ext4 -q /dev/nvme1n1p1

#mount the newly formatted filesystem
sudo mount /dev/nvme1n1p1 /mnt
```

Check that the new file system has been created and mounted with the **df -hT** command.

At this stage, the EBS volume is mounted at "/", and the local NVMe SSD disk is mounted at "/mnt". We can profile performance of both types of cloud disks using the same VM.

4. **Make a Snapshot of a live ROOT filesystem to backup data**

As you install new software on the VM and perform custom configuration, it becomes desirable to SAVE the modified ROOT file system containing the Ubuntu operating system and any new software packages and configuration installed.

To do this, return to the EC2 management console.
Select "Instances" on the left-side menu, and find and select your instance.

In the upper right-hand corner select the "Actions" drop-down menu:
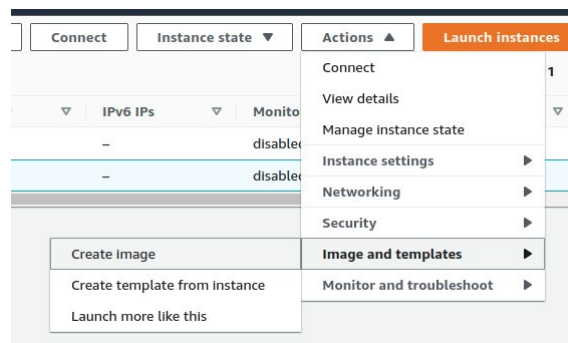
Scroll down and select "Image and templates" and "Create Image".

This will launch a wizard to configure saving a new Amazon Machine Image (AMI) that will store all changes made to the ROOT filesystem. To prevent the VM from shutting down and rebooting during the snapshot process, you can select "No reboot" Enable. In some cases it may be preferable to shutdown the VM when making a backup to ensure that server applications have stopped writing to the disk, a condition which could

result in the snapshot having corrupted partially written data. If you know your VM is dormant/quiet it is generally safe to not reboot.

Create image Info
An image (also referred to as an AMI) defines the programs and settings

Instance ID
  i-0c200e3fbfb06305c

Image name

test-vol-tcss562

Maximum 127 characters. Can't be modified after creation.

Image description - *optional*

Image description

Maximum 255 characters

No reboot
☑ Enable

To create an image, select the "**Create Image**" button on the bottom-right.
While the snapshot is being created, check its status by selecting "Snapshots" on the left-hand menus:

Snapshots (1)

Owned by me ▼   Q *Search*

	Name		Snapshot ID		Size		Description		Storage...		Snapshot status		Started		Progress		Encryption		KMS
☐	–		snap-0c2ff7ef3bf7fd13e		8 GiB		Created by CreateImage(i-...		Standard		⏱ Pending		2022/10/12 23:30 GMT-7		☹ Unavailable (1%)		Not encrypted		–

Cost Savings Measure #5:

**TO SAVE/PERSIST DATA, USE EBS SNAPSHOTS AND THEN
DELETE EBS VOLUMES FOR TERMINATED EC2 INSTANCES.**

EBS snapshot storage is only 5 cents/GB/month. For subsequent snapshots of the same volume, you are only charged for the deltas (differences) in storage size. In additional snapshots are stored using compression, and charges only for the actual data stored. For example, if you're disk volume is 16GB, but only 8GB is filled with data, then you're only charged for the 8GB.

There are some hidden things going on here.

AWS does not disclose the actual storage used per snapshot in a way that is easy to obtain. Total monthly snapshot storage charges are shown in the monthly bill. This can be found be selecting the drop-down menu in the upper right hand corner for your name. Then select "Billing Dashboard" from the list, "Bills" from the left. Scroll down, and expand "Elastic Compute Cloud", select and expand the region like "US East (Ohio), expand "EBS", and you should see snapshot charges if any. This provides the aggregate monthly charge for all snapshots which reports total GB storage. The GUI and CLI, however, do not expose detailed snapshot storage consumption information on a per-snapshot basis. I have two snapshots of a 16 GB volume. Snapshots record incremental differences. I mounted the two snapshots by creating 16 GB volumes for each snapshot and found I was using a total of 8.9 GB storage on both volumes. The remaining space was free space. In the monthly bill I I was charged for "9.837 GB-Mo".

The discrepancy must from overhead of incremental snapshots. To read more on this issue search the internet for "ec2 snapshot compression".

This leads to another cost savings measure:

Cost Savings Measure #6:

UNUSED SNAPSHOTS AND UNUSED EBS VOLUMES
SHOULD BE PROMPTLY DELETED !!

They will incur charges at high rates over time if left unused in the account draining cloud credits leading to incurring credit card charges. Deleting a snapshot is a two-step process. First, the AMI must be deregistered. Second, the snapshot must be deleted. These are separate steps in the GUI. Deregistering the snapshot does not prevent storage charges.

5. **Pausing an instance to save cost**

When working on a project, it may be desirable to PAUSE the virtual machine to save/retain its state to prevent having to recreate it. This is acceptable if changing working locations, for example going from school to home. This can also be acceptable if stopping work for the day, and looking to resume from the same point tomorrow.

While paused, a VM only incurs EBS storage charges at a rate of 8c/GB/month (gp3) or 10c/GB/month (gp2).

Try pausing your running VM.
In the ec2 console, select your instance, and from the "Instance state" menu, select "Stop instance".
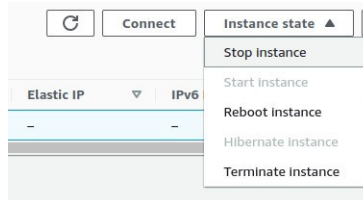
Unfortunately, unless you specified the "persistent" spot request earlier, the instance can't be stopped. It must be terminated, and recreated.

Because you've created an snapshot (AMI) from step 4, try recreating the instance using your AMI as persistent spot request. Only the left-hand side menu select "AMIs". The select the drop down and specify "Owned by me" as the AMI type:
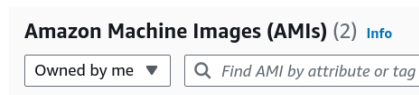
Now, select your recently created AMI, and press the orange launch button in the upper-right hand corner:

Follow the wizard steps as before, except this time it is not necessary to select the operating system.

A spot request can be made persistent when creating the VM. Under "**Advanced Details**", specify "**Request Spot Instances**" and click on the "**Customize**" link. For the "**Request type**", specify "**Persistent**", and for the "**Interruption behavior**" specify "**Stop**".

Once the instance is created and running, you can now try the "Stop instance" feature to pause the VM. This will allow you to Stop and Start the VM at-will pausing the hourly VM cost. While stopped the EBS charges remain: 10c/GB/month for EBS volumes (for gp2).

The caveat is that when you "Terminate instance" you also must ==**DELETE THE SPOT REQUEST**==. Failure to do so will result in the VM resurrecting itself indefinitely in your account resulting in ongoing unwanted charges.

Cost Savings Measure #7:

**USE PERSISTENT SPOT REQUESTS AND THE "STOP" FEATURE
TO PAUSE VMS DURING SHORT BREAKS**

Stopping a VM will persist the data using the EBS volume, preventing ongoing charges for the instance.

Again is it vital to ==**DELETE THE SPOT REQUEST**== when entirely done with the instance.

6. **Compare EBS and Instance Store Disk Performance using Bonnie++**

The last step of this tutorial is to compare disk performance using Bonnie++.
It is easy to install bonnie++.
From your VM's command line simply type:

```
$sudo apt install bonnie++
```

We've mounted your instance store volume under "/mnt". The instance store volume represents space on the local hardware that hosts the VM. For the c5d.large instance type, this is a local NVMe SSD drive.

Your root partition is an EBS volume, mounted under "/".
By default, the EBS volume is created as a GP2 – General Purpose 2 SSD EBS volume.
This volume is granted a baseline of 100 IOPS (I/O operations per second), and is burstable to 3,000 using a credit-based allocation approach.

For a detailed description of how Amazon manages IOPS for EBS volumes see this article:
http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSVolumeTypes.html

Amazon allows a certain number of IOPS relative to the overall disk size. This is approximated based on the typical number of IOPS for a hard disk based on different usage scenarios.

7. **Run the Bonnie++ disk benchmark utility to test EBS and Instance Store disk performance**

To run our disk benchmark, we will use ~ 7,520 MB of free disk space.

First run bonnie on the EBS volume:

The /tmp directory is on the "/" root partition.
Using /tmp as the bonnie directory will test the EBS volume performance.

Bonnie++ wants to create files that are twice the size of the RAM of the VM.
Here you should have previously increased the root volume size to 15 GB or else the EBS volume won't have enough space on a c5d.large. Available disk space can be checked with: "df -h"

```
$sudo chmod a+rwx /tmp
$bonnie++ -d /tmp -s 7520M -n 0 -m TEST-EBS -f -b -q  > bonnie.csv
```

This will capture the output to a text file.

Next, run bonnie on the instance store volume.
First, we need to create a Linux disk partition on the unformatted local SSD.
To determine the device name of your local SSD, use the '**lsblk**' command.
The device name is probably 'nvme1n1'. It will appear in the output of lsblk as the disk without any "part" items, which means partitions. Below, if your SSD device has a name other than 'nvme1n1', replace the new name with 'nvme1n1' below.
Now create a Linux ext4 partition. Often the 'fdisk' command is used, which is menu driven.
But for simplicity, specific commands are provided to create the partition using 'parted'.

```
# create the partition

$sudo parted /dev/nvme1n1 mklabel msdos

# match the size (46.6GB) with the size displayed by 'lsblk':

$sudo parted -a optimal /dev/nvme1n1 mkpart primary ext4 0% 46.6GB

# create the file system – this is formatting the disk

$sudo mkfs.ext4 /dev/nvme1n1p1

# mount the disk

$ sudo mkdir /mnt

$ sudo mount /dev/nvme1n1p1 /mnt -t ext4

# create a temp directory on the disk

$ sudo mkdir /mnt/tmpi

$ sudo chmod a+rwx /mnt/tmpi
```

The /mnt/tmpi directory is on the "/mnt" partition which is on the local SSD instance store disk.
Using /mnt/tmpi will allow us to test the local SSD instance store disk performance.
We use the same file size on the instance store volume for comparison purposes.

```
$bonnie++ -d /mnt/tmpi -s 7520M -n 0 -m TEST-SSD -f -b -q  >> bonnie.csv
```
Bonnie provides formatting utilities which take the CSV output and convert the output to either text (txt) or html.

These utilities are:
bon_csv2html
bon_csv2text

To use the utilities to generate formatted output, simply redirect your bonnie.csv output file into the utility:

```
$bon_csv2txt < bonnie.csv
```
AND
```
$bon_csv2html < bonnie.csv
```

Now, using the clipboard, copy the contents of the bon_csv2html output, and save this as a local .html file on your laptop. Try opening the file in a web browser by navigating from "file://"

Tutorial Questions:

Submit written answers as a PDF file on canvas. If submitting with a partner, include both names at the top of the PDF as they appear in CANVAS.

Only one person in the team should submit the assignment!

Include the HTML table output (or txt output if HTML is unavailable) from all Bonnie tests at the bottom of the PDF. Failure to include Bonnie output will result in a 0 for Bonnie questions.

Bonnie Questions

1. (2 points) What EC2 instance type did you run Bonnie++ on? This really should be c5d.large. If this is not c5d.large, please repeat with c5d.large. If for some reason there is an issue using c5d.large, it is best to consult with the instructor than submit the assignment with another VM type.

2. (2 points) What was the sequential output block read throughput (in K/sec) for the EBS and Instance Store volumes?

3. (2 points) How much CPU capacity was required for sequential output block reads for the EBS and Instance Store volumes?

4. (2 points) What was the sequential output block rewrites and sequential input block read throughput for the EBS and Instance Store volumes?

5. (2 points) How much CPU capacity was required to perform sequential output block rewrites and sequential input block reads for the EBS and Instance Store volumes?

6. (2 points) What was the random seek disk performance for the EBS and Instance Store volumes?

Cost Savings Questions

7. (2 points) If using a spot instance with a persistent spot request, what must be done when terminating the instance to prevent accidental charges?

8. (2 points) After a course/research project ends, what must be done to delete EBS snapshots used as AMIs with ROOT filesystems?

9. (2 points) When finished with a VM, how much money can be saved using an EBS snapshot to backup the data for a 10 GB EBS volume versus retaining the original EBS volume?

10. (2 points) When creating spot instances, what approach can be used to minimize costs within an AWS Region?

11. (2 points) What are persistent spot requests good for when working with spot instances?   What cost savings feature do they enable?

12. (2 points) What best practice should be used when creating ec2 instances for course projects and research to save cloud computing credits and money?

8.   Bonus Activity: Benchmark a "Provisioned IOPS" EBS Volume

(Optional / Non-graded) Provisioned IOPS EBS volumes forgo the credit based approach to provide consistent, guaranteed performance of EBS volumes within +/- 10%. The catch, this performance costs more, and is not offered as a FREE tier resource. The free tier offers up to 30GBs of general purpose EBS storage a month in the first year after you have created your AWS account.   To learn about **FREE resources**, visit https://aws.amazon.com/free.

Create a new IOPS EBS volume, and attach it to our currently running spot instance:

Go to EC2 | Elastic Block Store | Volumes

And then click "Create Volume".

Specify as follows:

Volume Type: Provisioned IOPS SSD (io1)
Size (Gib): 10
IOPS: 100
Availability Zone: us-east-2c (**the zone must match where your VM is running in order to attach it !!!**)

According to Amazon, an IOPS EBS volume with these settings will cost 12.5 cents per GB compared to 10 cents per GB for a general-purpose (GP2) EBS volume. Additionally, Amazon charges for the guaranteed IOPS. The charge for 1-month is $.065 x 100 IOPS or $6.50/month. The total monthly cost of a 10GB volume with a 100 IOPS guarantee is $7.75/month.

** There would be no practical reason to ever create this EBS volume since the guaranteed IOPS is no better than what GP2 provides. **

Once the volume has been created (try clicking refresh a few times), select "Action"
and then "Attach Volume"

Please note the disclaimer in yellow.

As of this writing the disclaimer is out of date for 5th gen SSD enabled resources. When attaching to your c5d instance the device name will be mapped to /dev/nvme2n1. This can be seen with the '**lsblk**' command.

Next you'll need to partition, format, and mount the disk:

```
$sudo parted /dev/nvme2n1 mklabel msdos

$sudo parted -a optimal /dev/nvme2n1 mkpart primary ext4 0% 10.0GB

$sudo mkfs.ext4 /dev/nvme2n1p1

$ sudo mkdir /mnt2

$ sudo mount /dev/nvme2n1p1 /mnt2 -t ext4

Then check that the disk is available at /mnt2:

$ df -h
```

Create a local tmp directory and grant world read write execute permission:

```
$sudo mkdir /mnt2/tmp_ebs

$sudo chmod a+rwx /mnt2/tmp_ebs
```

Now, let's test this volume and append our results to the bonnie.csv output file:

```
$bonnie++ -d /mnt2/tmp_ebs -s 7520M -n 0 -m TEST-EBS -f -b -q >> bonnie.csv
```

Note, the test will be *much* slower because of the very low IOPS of the EBS volume.

Once the test completes, capture your bon_csv2html output, and copy and paste the HTML to a html file on your laptop, save it, and view in a browser.

Or, alternatively, use bon_csv2text and view using the command line.
How does the provisioned IOPS EBS volume perform?

9. **Cleanup**

At the end of the tutorial:

Be sure to:

#1 - **<u>TERMINATE</u>** all EC2 instances

#2 - **<u>DELETE</u>** all EBS volumes

Failing to clean up could result in loss of AWS credits and/or AWS charges to a credit card.

10. **SAVING STORAGE COSTS WITH S3**

The Simple Storage Service (S3) is a key-value object storage facility on AWS capable of storing any type of data. If you are working to deploy an application that requires large datasets, it is recommended to create virtual machines with ephemeral disks. VMs with ephemeral disks often have a lower-case d such as c5d.large as used in this tutorial. There is also m5d, r5d, and z1d VM types among others. The idea is that any data stored on the ephemeral disk will be lost if the VM is stopped or shutdown. The simple storage service offers a low cost option for storing this data. S3 is a good alternative to EBS volumes and snapshots. S3 storage is just 2.3 cents/GB/month in contrast to 5 cents/GB/month for EBS Snapshots and 10 cents/GB/month for EBS volumes. Using the AWS CLI, a BASH script can be written to create a tar zip archive file which can then be pushed to an S3 bucket to store any data from your ephemeral disk(s).

To backup data on a mounted disk to S3, first create a tar.gz file. You'll need sufficient storage to store this file:

```
# all content under /mnt is compiled into a single tar gzipped file:

sudo bash
tar czf mydisk.tar.gz /mnt

# to access S3 from the VM, install the 'awscli' package as in tutorial 0, and run
# aws configure or alternatively, transfer the data to your Ubuntu machine using sftp,
# and upload to S3

# copy data to S3
# first create an S3 bucket – S3 buckets require globally unique names
# because the bucket name becomes a unique identifier

# replace "initials" with your initials or a unique id

aws s3 mb s3://462-562.f23.initials.vm-data

# upload the data
aws s3 cp mydisk.tar.gz s3://462-562.f23.initials.vm-data/
```

KEY POINT: be cognizant when working with large volumes of data. Creating a large 500 or 1000GB EBS volume can be VERY EXPENSIVE. Instead use a c5d/m5d instance with ephemeral disks, and then use S3 and tar/gzip to shuttle data to and from the ephemeral drive(s) at less than ¼ the monthly cost.

26