

Show Submission Credentials

P0. Getting Started with Cloud Computing

Welcome to the Cloud

✓ Introduction
✓ AWS Benchmarking
✓ Azure Benchmarking
✓ GCP Benchmarking
✓ Introduction to Data Analytics using a Wikipedia Dataset
✓ Dataset Overview
🔒 Data Pre-processing
🔒 Data Analysis
🔒 Conclusion
🔒 Project Survey

Introduction

Introduction

Information

Learning Objectives

1. Experiment with the web portal to create and configure virtual machines on AWS, Azure and GCP.
2. Experiment with performance benchmarking and web service hosting on virtual machines.
3. Process a portion of a large dataset of text using sequential programs on cloud resources.
4. Practice test-driven development (TDD) with JUnit when developing solutions to complex problems.
5. Identify the potential limitations of using sequential programs to analyze large volumes of data.
6. Practice Linux tools (`grep` and `awk`) to search patterns in files and perform data analysis.
7. Adopt the Python Data Analysis Library (pandas) to solve data science problems progressively with interactive programming using Jupyter Notebook.
8. Explore the characteristics of real-world datasets and recognize arbitrary or corrupted records.
9. Develop a robust code that correctly executes in multiple environments, whether on your local computer or a remote virtual machine.
10. Experiment using Terraform to orchestrate and manage virtual machines with prepared worked examples.

Project Scenario

You have started a rotational internship program in a cloud computing consulting firm that provides consulting and solutions, leveraging multiple cloud service providers.

The internship program is for undergraduates and graduates with passion for expanding their skills in cloud computing, software engineering, and data science.

During the rotational program, you will gain exposure to a vast array of teams, cloud service providers, tools, processes and practices. By the end of the program, you will be considered whether to join the company long-term.

Your rotation occurs across two groups:

1. Cloud Migration team that provides solutions to the clients who need to move from on-premise infrastructure to the cloud and recommends cloud service providers and cloud products.
2. Data Science team that provides solutions to processing and analyzing data in order to gain business insight.

The firm has well-established DevOps/DataOps culture and processes. You do not only need to implement solutions that meet functional requirements. You are required to implement solutions that are correct, well-tested, readable and maintainable. In other words, the programs you develop must follow test-driven development (TDD), achieve required code coverage, and pass the code style check.

Information

General Details

The following table contains some general information about this project:

This project does not count to your course grade. Meanwhile, all the hands-on tasks in this project are auto-graded, and you will make submissions to validate the correctness of your solutions. The project-based learning with the submission-feedback cycle you will experience in this project will get you prepared to the next graded projects.

Prerequisites	Programming experience with Java 8 and Python 3
Primers	Git Best Practices; Intro to Maven and Checkstyle; Jupyter Notebook; Infrastructure as Code
Applicable Languages	In each task, we will specify the language(s) that you must use so that you can use provided starter code template. Note: If you use Maven, the <i>Maven central repository</i> is the only remote repository you are allowed to use.
Applicable Cloud Platform	AWS Benchmarking: AWS, Azure Benchmarking: Azure, GCP Benchmarking: GCP, Data Pre-processing and Data Analysis: AWS
Total Recommended Budget	\$5
Tags Required	Key: project , Value: getting-started-with-cloud-computing . Both the key and value should only include lowercase English letters and/or hyphens.

Danger

Tagging and Budget Requirements in the Course

It is important to note that this course treats the tagging and budget requirements equivalently to meeting a project's correctness and performance requirement.

Your future employers have asked us to train you to:

1. plan and monitor the spending of cloud resources
2. tag cloud resources to achieve billing and managerial goals
3. manage the lifecycle of cloud resources and terminate cloud resources in a timely manner

Tagging and budget management of cloud resources are most likely very new skills for you to learn, plan for and get used to.

In this project which does not count to your course grade, you will receive warning emails without any penalty if you fail to tag all the required resources and/or exceed the budget.

Starting from the next project, the penalty policies of tagging and budget will apply.

Danger

Project Grading Penalties

Violation	Penalty of the project grade
Failing to include all the required files in the final submission	Warning email
Submitting only executables (.jar , .pyc , etc.) without human-readable code (.py , .java , .sh , etc.) in the final submission	Warning email
Spending more than \$5 for this project on AWS	Warning email
Spending more than \$10 for this project on AWS	Warning email & Possible penalty to the course grade
Failing to tag all your resources for this project; Key: project and Value: getting-started-with-cloud-computing	Warning email
Provisioning AWS resources in regions other than us-east-1	Warning email
Submitting your AWS credentials, other secrets, submission username or any personally identifiable information (PII)	Warning email
Attempting to hack/tamper the grader	Warning email & Penalty to the course grade
Cheating, plagiarism or unauthorized assistance (please refer to the university policy on academic integrity and our syllabus)	Warning email & Penalty to the course grade

AWS Benchmarking

AWS Benchmarking

Your Task with the Cloud Migration Team

You first start with the cloud migration team. The team has a new client who would like to move from on-premise to the cloud for their web hosting. The client would like to be informed with a cloud hosting solution that has a good balance between performance and cost.

After analyzing the existing on-premise infrastructure of the client, your team decides that virtual machines are the best option to achieve a smooth cloud migration. The client is very cautious about the budget, since the budget was the major motivation to move from on-premise to the cloud in the first place.

The client also needs to decide the cloud service provider, based on factors such as pricing, performance, user-friendliness, etc.

Therefore, your supervisor asks you to experiment with different sizes/types of virtual machines across three popular cloud service providers: Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP).

Virtual Machines (VMs)

Cloud practitioners often perform computing on isolated resources called virtual machines (VMs). Cloud Service Providers (CSPs) provide different metered options of these VMs to users. You will need to choose the instance types/sizes with the OS and resources (memory, CPU, and network capabilities) that are best aligned with your need and budget.

As you can expect, virtual machines with more resources. Would provisioning an expensive instance always translate into an improvement of the performance given the application and/or computation demand? What is a good metric to measure performance? How often does the performance vary? These are all hard questions in the industry that require structured benchmarking and calculation, and this introductory project will give you some insight into these problems.

The objective of the following VM benchmarking tasks will introduce you to virtual machines. You will provision VMs, connect to the VMs in order to benchmark the VMs and also host web services.

This part of the project consists of into multiple sections. We will begin by asking you to launch 3 different types of virtual machines, assess their relative performance using a number of internal benchmarking tools, and finally assess their performance as public web servers. You will need to complete hands-on tasks and answer AssessMe questions before proceeding to the next section on this page.

Working on the cloud may be a new experience for you. To get better prepared for the hands-on activities in this project, please complete the released CSP primers before you start this project.

Prerequisites

Before getting started with this task, you need to complete the `AWS intro` primer so that you know how to create and configure an AWS instance from the console.

AWS EC2 instance types to benchmark

Instance	CPU	RAM
t4g.small	2	2

t4g.medium	2	4
t4g.large	2	8

Create an AWS EC2 VM

You will start with AWS EC2 instances and create 3 VMs of sizes `t4g.small` , `t4g.medium` , `t4g.large` .

Visit the AWS Portal (<https://aws.amazon.com>) and create a Virtual Machine running **Ubuntu 18.04 LTS**.

Remember to tag the instance as `project` and set the value as `getting-started-with-cloud-computing` .

1. Choose AMI2. Choose Instance Type3. Configure Instance4. Add Storage5. Add Tags6. Configure Security Group7. Review

Step 5: Add Tags

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver. A copy of a tag can be applied to volumes, instances or both. Tags will be applied to all instances and volumes. [Learn more](#) about tagging your Amazon EC2 resources.

Key (128 characters maximum)	Value (256 characters maximum)	Instances ⓘ	Volumes ⓘ	Network Interfaces ⓘ	Spot Instance Requests ⓘ
project	getting-started-with-cloud-computing	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Add another tag (Up to 50 tags maximum)

Figure: Add tags

Launch 3 instances of instance type `t4g.small` , `t4g.medium` , `t4g.large` . Each with 30 GB of General Purpose SSD (gp2) storage. Use the Amazon Machine Image Ubuntu Server 18.04 LTS (HVM), SSD Volume Type with the **64-bit (Arm)** architecture. You must select **64-bit (Arm)** in order to select ARM-based instances types such as `t4g` . You will see instance types such as `t2` `t3` instead if you selected the incorrect **64-bit (x86)** architecture.

To save on cost, we recommend using spot instances. If no spot instances are available, you can use on-demand instances.

Make sure that the security group has port `22` (for SSH) as well as `80` (for HTTP) open.

1. Choose AMI2. Choose Instance Type3. Configure Instance4. Add Storage5. Add Tags6. Configure Security Group7. Review

Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group: ☒ Create a new security group ☐ Select an existing security group

Security group name:

Description:

Type ⓘ	Protocol ⓘ	Port Range ⓘ	Source ⓘ	Description ⓘ
SSH	TCP	22	Custom 0.0.0.0/0	e.g. SSH for Admin Desktop
HTTP	TCP	80	Custom 0.0.0.0/0	e.g. SSH for Admin Desktop

Add Rule

Warning

Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

CancelPreviousReview and Launch

FeedbackEnglish (US)

© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy PolicyTerms of Use

Figure: Open the port 22 and 80 in the security group

When the instances are up and running, connect to the VMs with the command:

```
ssh -i <path-to-pem-file> ubuntu@<instance-dns>
```

CPU Benchmark

To benchmark CPU performance we will be using `sysbench`.

SysBench is a modular, cross-platform and multi-threaded benchmark tool for evaluating OS parameters that are important for a system running a database under intensive load.

The idea of this benchmark suite is to quickly get an impression about system performance without setting up complex database benchmarks or even without installing a database at all.

To install `sysbench` run:

```
sudo apt-get update
sudo apt-get install -y sysbench
```

The CPU benchmark finds prime numbers till 10,000 by default. You can change this by adding `--cpu-max-prime=<max limit>` and also specify the number of threads by using the `--threads` flag.

In this task, please use the following combinations for benchmarking:

Max Prime number	Number of CPU threads
20000	8
40000	1
50000	4

For example, the command below finds prime numbers till 20,000 with 8 threads:

```
sysbench --cpu-max-prime=20000 --threads=8 cpu run
```

Run the commands on the three VMs with different VM types (`t4g.small` , `t4g.medium` and `t4g.large`). Note down the results down for each of the instances. You might have to wait for the benchmarking to finish. An example output may look like as follows:

```
Running the test with following options:
Number of threads: 8
Initializing random number generator from current time
```

```
Prime numbers limit: 20000
```

```
Initializing worker threads...
```

```
Threads started!
```

```
CPU speed:
  events per second:   322.76
```

```
General statistics:
  total time:           10.0146s
  total number of events: 3233
```

```
Latency (ms):
  min:                  2.67
  avg:                  24.72
  max:                  90.19
  95th percentile:     34.95
  sum:                  79931.23
```

```
Threads fairness:
  events (avg/stddev):   404.1250/1.27
  execution time (avg/stddev): 9.9914/0.01
```

Here `events-per-second` means the number of prime numbers calculated in one second. `General Statistics` define the time for the test, and the number of prime numbers found(keeping in mind that the threshold for the highest prime number being defined in `--cpu-max-prime=<value>`). `Latency` refers to delay in processing and as such should be as low a value as possible. `Thread fairness` refers to event distribution by thread or the number of events that took place per thread (approximately).

File IO Benchmark

Next, we test the file IO performance of each instance. These values can vary widely depending upon the type of disk access pattern. For example, accessing the disk sequentially yields much higher IOPS than randomly accessing data on a disk. This gain in performance is due to factors such as caching, as well as the type of disk. Rotating (HDD) drives are impacted by the seek time and rotational latency, whereas Solid State Drives (SSDs) are impacted by the speed of the drive interface and the device's controller. To run a disk IO performance benchmark, we must create a large file and then read and write to it, either randomly or sequentially.

Using `sysbench`, we can run the benchmark on all instances as follows:

Prepare a 5 GB file to check the disk IO performance. This can be done using the command below.

```
sysbench --file-total-size=5G fileio prepare
```


Make sure to create a new file for each `sysbench` run to prevent cache based speedup. Furthermore, set the size of the files to be larger than that of the VM's RAM to prevent the entire file from being cached

Now, run a benchmark that randomly reads and writes to this file. You should use the command:

```
sysbench --file-total-size=5G --file-test-mode=rndrw --rand-seed=0 --time=300 -  
-events=0 fileio run
```

Where `rndrw` stands for Random Read-Write. `rand-seed` initializes a Random number generator. `time` the time the test will run and `events` specify the maximum number of input and output requests (0 meaning unlimited) Therefore this test will run for 300 seconds where an unbounded number of random read and write operations will be made You should get a result as follows:

```
.....  
  
File operations:  
  reads/s:                447.00  
  writes/s:               298.00  
  fsyncs/s:               953.25  
  
Throughput:  
  read, MiB/s:            6.98  
  written, MiB/s:         4.66  
  
General statistics:  
  total time:              300.0003s  
  total number of events:  509477  
  
Latency (ms):  
  min:                    0.00  
  avg:                     0.59  
  max:                    1732.34  
  95th percentile:       2.91  
  sum:                    299488.17  
  
Threads fairness:  
  events (avg/stddev):    509477.0000/0.00  
  execution time (avg/stddev):  299.4882/0.00
```

`File operations` explain the number of operations that took place per second in the time span. `Throughput` explains the read and write speeds per second. `General statistics` explains the total time and number of events that took place. `Latency` refers to delay in file processing and as such should be as low as possible. `Thread fairness` refers to event distribution by thread or the number of events that took place per thread (approximately)

Host a website with the Apache Web Server

Select any VM among the three VMs you created in this task, and use the following commands to install the Apache Web Server:

```
sudo apt-get update
sudo apt-get install -y apache2
```

Apache web server has a default static webpage `/var/www/html/index.html` . Please use Linux editors/commands of your choice to replace the content of `/var/www/html/index.html` as an exact string `Cloud Computing is awesome!` .

Validate your work

Visit the public DNS of the VM, and you should expect to see a plain-text web page with the text `Cloud Computing is awesome!` .

Submit your task

Download and execute the submitter **exclusively** for this task. You will use different submitters in the next tasks.

```
cd ~

# download the submitter
wget cloudeveloper.blob.core.windows.net/getting-started-with-cloud-computing/sail/v1/aws/submitter -O submitter

# make the downloaded submitter executable
chmod +x submitter
```

You can run the following commands to set up your submission credentials and make a submission. Replace the "your_submission_username" with your username in the top right corner of this page, and replace "your_submission_password" with your submission password. You can find your submission password when you scroll up to the top and click **"Show Submission Password"**. Then run a `submitter` to submit this task and check your results on the Sail() platform.

```
export SUBMISSION_USERNAME="your_submission_username"
export SUBMISSION_PASSWORD="your_submission_password"
./submitter <Enter the public DNS of the VM>
```

When entering the public DNS, please avoid adding `http:\\` or `https:\\` or it would result in a failed submission.

Remember that your submission password can be found by clicking on the button at the top of this page. After running the `./submitter` command, you can check your submission result and feedback on the submission tab of the project on the Sail() platform. There is no limit on the number of submissions allowed before the project deadline. However, submissions must be separated by at least 60 seconds. Try not to exceed the recommended budget specified for this project module.

Danger

Terminate Resources in a Timely Manner

Please remember to **terminate all resources** once you are done in this task. Deleting each individual VM on the portal. You can navigate to "EC2 instances" and delete the instances via the portal.

Azure Benchmarking

Azure Benchmarking

Azure VM sizes to benchmark

Instance	CPU	RAM	Disk	Max IOPS
Standard_B1ms	1	2	2	640
Standard_B2s	2	4	4	1280
Standard_B2ms	2	8	4	1920

Create an Azure VM

In this task, you will benchmark three Azure VMs with different sizes: Standard_B1ms, Standard_B2s, and Standard_B2ms. Visit the Azure Portal (<https://portal.azure.com>) and create a Virtual Machine running **Ubuntu 18.04 LTS**. Use the Azure subscription for your task and keep the location **East US**.

First make sure you create a resource group. Navigate to the resource group section. Create a resource group.

[Home](#) > [Resource groups](#) >

Create a resource group

×

Basics

Tags

Review + create

Resource group - A container that holds related resources for an Azure solution. The resource group can include all the resources for the solution, or only those resources that you want to manage as a group. You decide how you want to allocate resources to resource groups based on what makes the most sense for your organization. [Learn more](#)

Project details

Subscription * ⓘ

Resource group * ⓘ

Resource details

Region * ⓘ (US) East US

[Review + create](#)

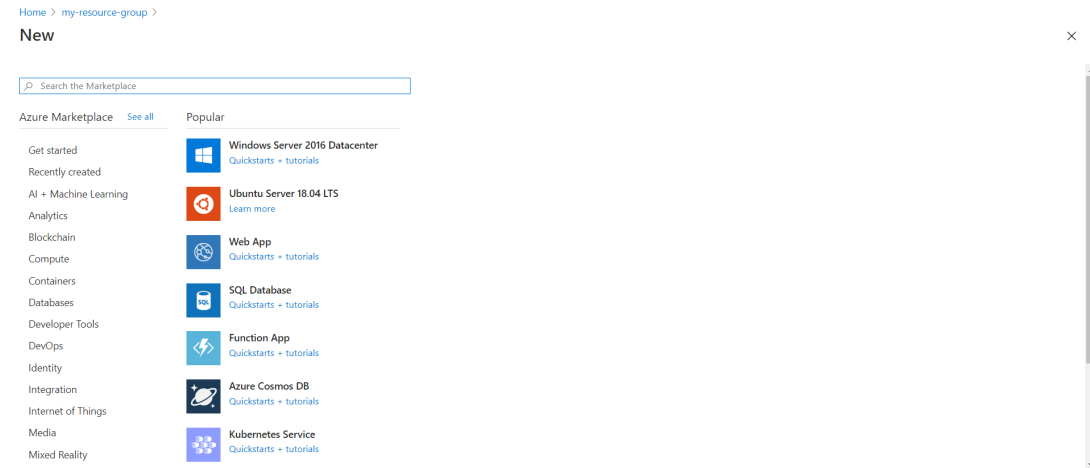
[< Previous](#)

[Next : Tags >](#)

In the Tags tab, add the tag `project : getting-started-with-cloud-computing`. Click **Review + Create** and create the resource group.

Now open the resource group and click on "+ Create".

Search and select Ubuntu Server 18.04 LTS.



On the VM setup page:

The screenshot shows the 'Create a virtual machine' page in the Microsoft Azure portal. The page is divided into several sections: Subscription, Resource group, Instance details, and Administrator account. The Subscription is set to 'my-resource-group'. The Resource group is set to 'my-resource-group'. The Instance details section includes: Virtual machine name (my-vm), Region (US East US), Availability options (No infrastructure redundancy required), Image (Ubuntu Server 18.04 LTS), Azure Spot instance (No), Size (Standard_B1ms - 1 vcpu, 2 GiB memory (US\$15.11/month)), and Administrator account (SSH public key or Password). The Administrator account section is currently set to Password.

- Set the name of the VM of your choose, the region as **(US) East US**
- Select the size as Standard_B1ms/Standard_B2s/Standard_B2ms.
- Select Password as the authentication type and set a username and password for the VM. Take note of the username and password since you will need it later.

Home > my-resource-group > New >

Create a virtual machine

✕

Select size

Administrator account

Authentication type

SSH public key

Password

Username *

✓

Password *

✓

Confirm password *

✓

Inbound port rules

Select which virtual machine network ports are accessible from the public internet. You can specify more limited or granular network access on the Networking tab.

Public inbound ports *

None

Allow selected ports

Select inbound ports *

HTTP (80), SSH (22)

⚠ This will allow all IP addresses to access your virtual machine. This is only recommended for testing. Use the Advanced controls in the Networking tab to create rules to limit inbound traffic to known IP addresses.

Review + create

< Previous

Next: Disks >

Set inbound port 22 open so that you can SSH into the instance as well as port 80 for HTTP access.

Home > my-resource-group > New >

Create a virtual machine

✕

Basics **Disks** Networking Management Advanced Tags Review + create

Azure VMs have one operating system disk and a temporary disk for short-term storage. You can attach additional data disks. The size of the VM determines the type of storage you can use and the number of data disks allowed. [Learn more](#)

Disk options

OS disk type *

Standard HDD

The selected VM size supports premium disks. We recommend Premium SSD for high IOPS workloads. Virtual machines with Premium SSD disks qualify for the 99.9% connectivity SLA.

Encryption type *

(Default) Encryption at-rest with a platform-managed key

Enable Ultra Disk compatibility

Yes

No

Ultra Disk compatibility is not available for this VM size and location.

Data disks

You can add and configure additional data disks for your virtual machine or attach existing disks. This VM also comes with a temporary disk.

LUN	Name	Size (GiB)	Disk type	Host caching
<div>Create and attach a new disk</div>				
<div>Attach an existing disk</div>				

Review + create

< Previous

Next: Networking >

Under the "Disks" tab, select **Standard HDD** as the Disk type which is most suitable for non-critical exploratory tasks.

Home > my-resource-group > New >

Create a virtual machine

✕

[Learn more](#)

Network interface

When creating a virtual machine, a network interface will be created for you.

Virtual network *

(new) my-resource-group-vnet

Create new

Subnet *

(new) default (10.0.0.0/24)

Public IP

(new) my-vm-ip

Create new

NIC network security group

None

Basic

Advanced

Public inbound ports *

None

Allow selected ports

Select inbound ports *

HTTP (80), SSH (22)

⚠ This will allow all IP addresses to access your virtual machine. This is only recommended for testing. Use the Advanced controls in the Networking tab to create rules to limit inbound traffic to known IP addresses.

Accelerated networking

On

Off

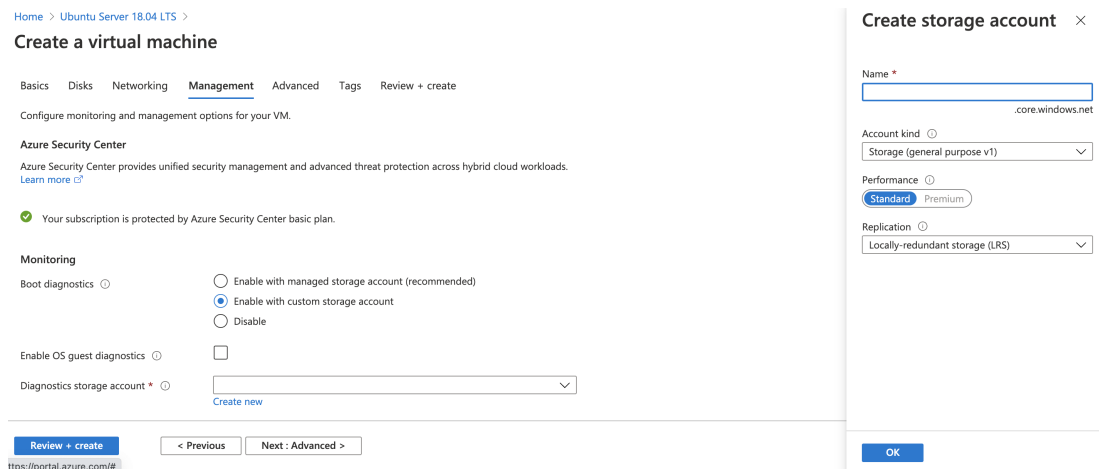
The selected VM size does not support accelerated networking.

Review + create

< Previous

Next: Management >

Under the "Networking" tab, make sure that both Port 80 and Port 22 are open.

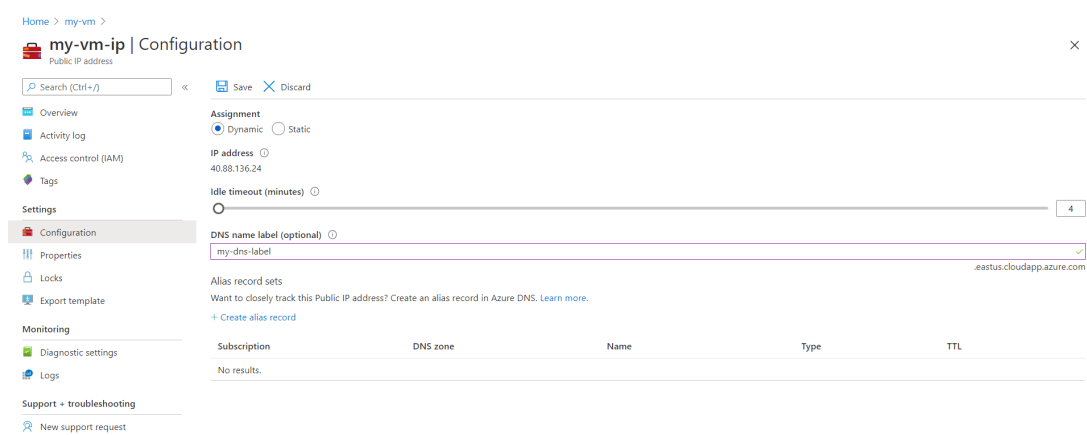


Under the "Management" tab, select Boot diagnostics as Disable .

Under the "Tags" tab, add the tag project : getting-started-with-cloud-computing .

Click on **Review + Create** and create the VM.

Once the VM is ready, click on the VM to open the VM page and click on the link next to the **DNS name** attribute to set a public DNS for the VM.



Now enter a DNS name and click the save icon on top.

Repeat the above steps until you have 3 VM instances of sizes Standard_B1ms , Standard_B2s , and Standard_B2ms .

You can now connect to any of the VMs by running the command:

```
ssh <username>@<azure-vm-dns>
```

Steps to complete

1. Complete the CPU benchmarking and IO benchmarking for the 3 VMs.

2. Install Apache Web Server and host a static web page that contains the text Cloud Computing is awesome! on any VM.
3. Download and execute the submitter exclusively for this task.

How to Submit

Follow the same process as in the AWS Benchmarking task, with a different submitter **exclusively** for this task. Execute the submitter with **the public DNS (not public IP)** of the Azure VM.

```
wget clouddeveloper.blob.core.windows.net/getting-started-with-cloud-computing/sail/v1/azure/submitter -O submitter
chmod +x submitter
export SUBMISSION_USERNAME="your_submission_username"
export SUBMISSION_PASSWORD="your_submission_password"
./submitter <Enter the public DNS of the VM>
```

Danger

Terminate Resources in a Timely Manner

Please remember to **terminate all resources** in this task by deleting the Azure resource groups.

GCP Benchmarking

GCP Benchmarking

GCP VM sizes to benchmark

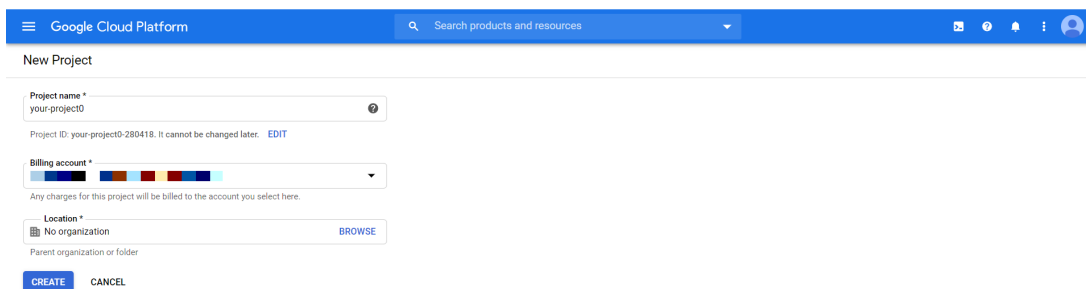
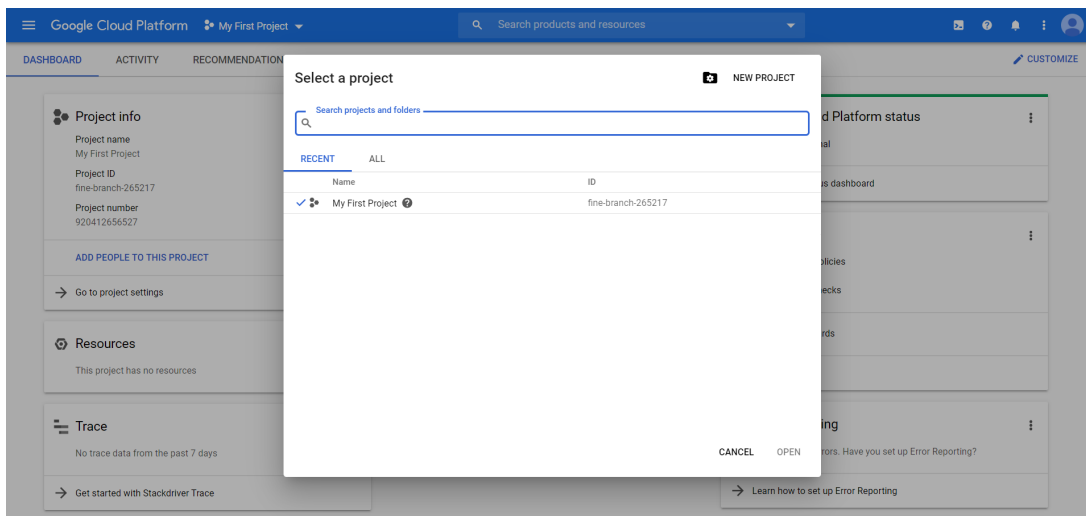
In this task, you will experiment with 3 GCP VMs of different sizes f1-micro , n1-standard-1 , n1-standard-8 .

Instance	CPU	RAM
f1-micro	1 (shared-core (https://cloud.google.com/compute/docs/general-purpose-machines#sharedcore))	0.60 GB
n1-standard-1	1	3.75 GB
n1-standard-8	8	30 GB

Create a GCP VM

Visit the GCP Portal (<https://console.cloud.google.com>) and create a Virtual Machine running **Ubuntu 18.04 LTS** Use the billing account that you set up in the Cloud Account Setup primer.

First, select or create a GCP project:



Search for "Compute engine" in the search bar or navigate via the app drawer (the three-line icon in the top left corner) to choose "Compute Engine". For a new GCP project, you may need to enable Compute Engine API first before provisioning VMs.

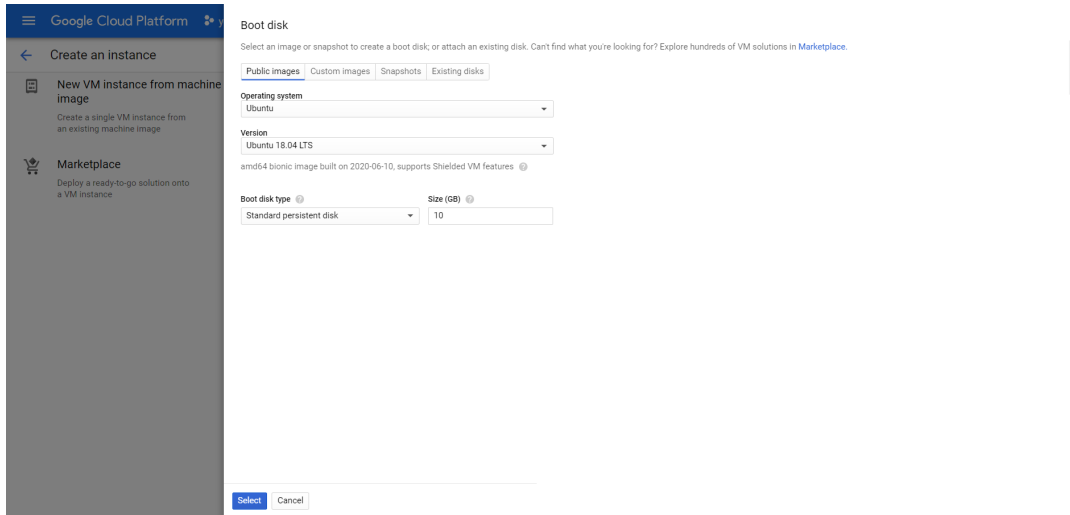
Click the "CREATE INSTANCE" button in the VM instances page.

Set the VM name of your choice.

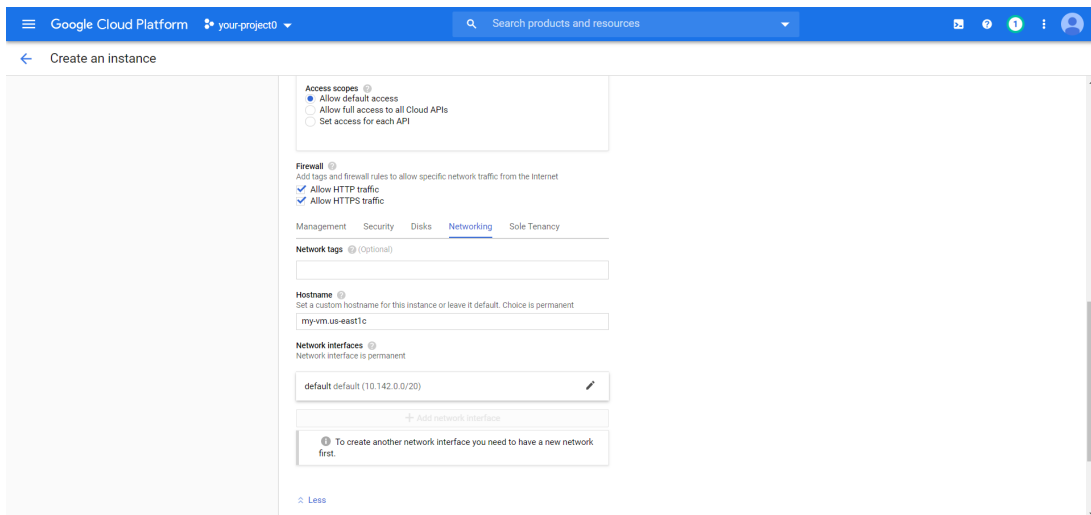
Select the VM size (known as "machine type" in GCP) as `f1-micro`, `n1-standard-1` or `n1-standard-8` accordingly for the three VMs.

Select the region as **us-east1** and the zone as **us-east1-c**.

Set the labels as `project : getting-started-with-cloud-computing`.

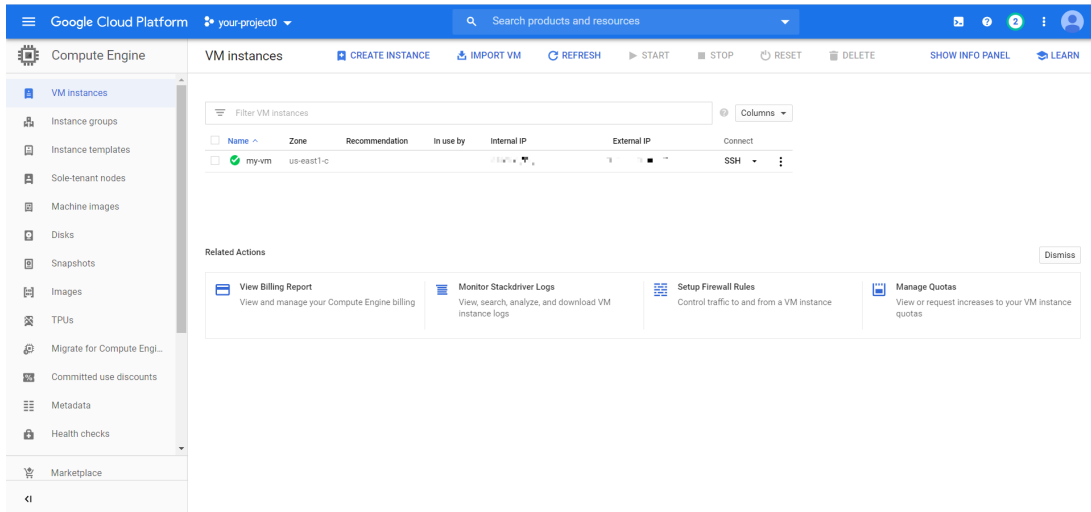


Select **Ubuntu 18.04** as the OS image.



Under the "Firewall" section, check both `Allow HTTP traffic` as well as `Allow HTTPS traffic`.

Click on **Create**. Once the VM is ready you should see the resource group page populated as:



Repeat the above steps until you have 3 VM instances that includes 1 f1-micro , 1 n1-standard-1 and 1 n1-standard-8).

You can now connect to any of the VMs by running the command in the GCP Cloud Shell.

```
gcloud compute ssh --zone "us-east1-c" <vm-name> --project <gcp-project-id>
```

Information

GCP does not provide managed public DNS for GCP VMs as AWS or Azure. If you need to add a public DNS to a GCP VM, you must configure the public DNS name with a domain name owned by you.

Therefore, in this task, you will use the public IP to access the GCP VM. When you submit your solution in this task, you should also provide the public IP of the GCP VM.

Steps to complete

1. Complete the CPU benchmarking and IO benchmarking for the 3 VMs.
2. Install Apache Web Server and host a static web page that contains the text Cloud Computing is awesome! on any VM. Validate your task by accessing `http://<public-ip>` NOT `https://<public-ip>`.
3. Download and execute the submitter exclusively for this task.

How to Submit

Follow the same process as in the AWS Benchmarking task, with a different submitter exclusively for this task.

```
wget clouddveloper.blob.core.windows.net/getting-started-with-cloud-computing/sail/v1/gcp/submitter -O submitter
chmod +x submitter
export SUBMISSION_USERNAME="your_submission_username"
export SUBMISSION_PASSWORD="your_submission_password"
./submitter <Enter the public IP of the VM>
```

Danger

Terminate Resources in a Timely Manner

Please remember to **terminate all resources** in this task by shutting down the GCP project.

Introduction to Data Analytics using a Wikipedia Dataset

Introduction to Data Analytics using a Wikipedia Dataset

Your Task with the Data Science team

The Data Science team has a client who would like to detect public trending topics in order to maximize advertisement effectiveness. You are asked to pre-process and analyze real-world dataset in order to understand the trends.

Wikipedia is often a great reflector of current events. Pages that are being accessed with increasing frequency often indicate an event related to the topic of the page. Births, deaths, Google Doodles, wardrobe malfunctions, or even something mundane as a technical conference can often trigger a spurt of interest in a topic.

Wikipedia serves as a fairly unbiased source of reporting the news, and sometimes even reveals hidden patterns.

You will analyze data with a single instance running small scripts on an hour's worth of Wikipedia traffic log. We will focus exclusively on English Wikipedia and ignore the rest of the Wikimedia project, to cater to the interest of students.

To be able to do this, we will write a basic filter that is capable to process our hourly dataset. This form of sanity testing your code on a subset of the data will be useful in all your future data wrangling tasks.

Danger

Accessing 15-319/15-619 Resources

In this activity, we require you to SSH into Ubuntu VMs running on the AWS cloud.

Before you provision any resources, make sure that you have updated your 12-digit AWS account ID in your course profile. Failing to do so will cause you to be unable to use the custom image for this project we provide.

In this task, you can only launch a `t3.micro` instance.

Danger

Warnings

Please go through the necessary primers and set up your workstation to support easy and secure access to your cloud resources. Make sure that you are using the VM image provided by us when you are working on the programming tasks.

We will explore the usage of tools and libraries to solve common tasks efficiently. Therefore, we support Maven and `pip` for you to manage Java or Python 3 packages and dependencies using industry standards. We have also installed the GNU tools you may need. However, as you do not have `root` access, you cannot `sudo apt-get install` other Linux tools on the instance.

There are some guidelines for asking questions on Piazza to address:

- Feel free to create public posts to discuss general ideas so everyone can learn.
- Create a private post if you find anything unclear or broken.
- Teaching staff are always willing to share ideas to improve your learning experience. However, It is **NOT** the duty of teaching staff to help you debug. When creating a post, provide all efforts you already invested to look for an answer first!

Dataset Overview

Dataset Overview

Wikimedia maintains hourly pageview statistics (https://wikitech.wikimedia.org/wiki/Analytics/Data_Lake/Traffic/Pageviews) for all objects stored on Wikimedia servers as publicly accessible datasets. We will use these statistics to analyze pageview trends and derive the trending topics on Wikipedia for a particular time range.

Logs are stored with public access every hour in flat files of text. Each line of a file corresponds to pageviews upon one single wiki page on the Wikimedia servers.

Each line in the pageview files has 4 space-separated fields:

```
domain_code page_title count_views total_response_size
```

Field name	Description
domain_code	Domain name of the request (abbreviated).

page_title	Page title holds the title of the unnormalized (https://en.wikipedia.org/wiki/URL_normalization) (we will discuss this later) section after <code>/wiki/</code> in the request URL, e.g. <code>Main_Page</code> , <code>Carnegie_Mellon_University</code> , <code>User_talk:K6ka</code> .
count_views	The number of times this page has been viewed in the respective hour.
total_response_size	The total response size caused by the requests for this page in the respective hour. This column was defined in the former <code>pagecounts</code> (https://wikitech.wikimedia.org/wiki/Analytics/Data/Pagecounts-raw) dataset which has been deprecated and replaced by the <code>pageview</code> dataset. The <code>total_response_size</code> column was retained for backward compatibility but the new <code>Pageview API</code> (https://wikitech.wikimedia.org/wiki/Analytics/PageviewAPI) no longer uses it, and you will find that this column has a 0 value in most entries.

`domain_code` has two parts, a language identifier and the abbreviation of the sub-project suffix (https://wikitech.wikimedia.org/wiki/Analytics/Data/Pagecounts-all-sites#Contained_data) of the domain name.

The subproject suffix of the domain name (domain trailing part) is coded as an abbreviation, for example, `.wikibooks.org` is coded as `.b` and `.mediawiki.org` is coded as `.w`. The only exception is that project `.wikipedia.org` is indicated by the absence of any abbreviation, for example, `en.wikipedia.org` will be coded as `en`.

In this project we will focus on data from `.wikipedia.org` in English, which has the exact domain code: `en`.

For example:

```
en Carnegie_Mellon_University 34 0
```

This line means that 34 requests were made to `"en.wikipedia.org/wiki/Carnegie_Mellon_University"`, the Wikipedia **English desktop** site for Carnegie Mellon University.

Disambiguating abbreviations ending in ".m"

An abbreviation could end in `.m` in 2 ways. The first case is that the domain is a project on `wikimedia.org` (like `commons.wikimedia.org` with abbreviation `commons.m`). Another case is when the domain is the mobile site of Wikipedia (like `en.m.wikipedia.org` with abbreviation `en.m`).

A domain code in `<language>.m` format indicates a mobile site and a domain code in `<project>.m` format belongs to `wikimedia.org`, and there is no collision between these two conventions. In this project, we'll also include data from English mobile sites with the exact domain code `en.m`.

To summarize, here are valid examples (from either English desktop site or mobile site):

```
en Carnegie_Mellon_University 34 0
en.m Carnegie_Mellon_University_in_Qatar 1 0
```

Invalid examples:

```
commons.m 1848 1 0
en.m.b A+_Certification 2 0
en.q Donald_Trump 11 0
de Hillary_Clinton 870 0
```

In this project, you will only use the first hour of March 8, 2018, as we are beginning our process of Wikipedia information exploration, during which certain special events occurred.

The data to process and analyze has been uploaded to the following Azure Blob Storage location: <https://clouddeveloper.blob.core.windows.net/datasets/sequential-analysis/wikipediatraf/2018-march-madness/pageviews-20180310-000000.gz> .

You can `wget` the file via:

```
wget https://clouddeveloper.blob.core.windows.net/datasets/sequential-analysis/wikipediatraf/2018-march-madness/pageviews-20180310-000000.gz
```

Do not uncompress the file. Instead, you are required to read the compressed archive directly as the input stream, e.g.

```
zcat < pageviews-20180310-000000.gz | less
zcat < pageviews-20180310-000000.gz | ./your_executable
```

Data Pre-processing

Data Pre-processing

Pre-processing the initial data so that it is in the proper form is crucial before you begin analyzing it to gain any insights. **Never assume that the dataset is perfectly clean and well-formed.** The phrase "garbage in, garbage out," points out that if your input data is malformed, you should not expect to gain useful insight. The phrase is particularly applicable to data mining, machine learning, or any data analysis project. Thus, the representation and quality of data must be ascertained and ensured before analyzing the data.

Your task is to transform and filter out rows from the logs based on the following rules. You **must use Java** in this task. We provide starter code to develop your Java solution using test-driven development. You will need to handle a set of rules below, which can seem overwhelming at first glance. However, we will provide you with scaffolding and examples.

1. **URL normalization and percent-encoding:** According to RFC 3986 (https://en.wikipedia.org/wiki/Uniform_Resource_Identifier#Syntax), URLs should be normalized and special characters will be converted by percent-encoding (https://en.wikipedia.org/wiki/Percent-encoding#Percent-encoding_reserved_characters).

For example, user talk created by K6ka with the title

```
User: K6ka
```

matches this URL:

```
https://en.wikipedia.org/wiki/User%3AK6ka
```

Although % percent-encoded titles are expected in URLs, decoded titles are expected in pageview logs. Due to arbitrary user behavior and third-party usage, various titles pointing to the same page still exist for some pages, such as `Special%3ASearch` and `Special:Search`. As a result, we need to map all these titles to the same original title `Special:Search`. We will accomplish this by making use of percent decoding.

However, the definition and solution of percent-encoding for Wikipedia is slightly different from the widely used definition, and you may find that most percent decoders will break with our dataset. The reason for this is that there are many % symbols which are not followed by hex chars, such as `1%_rule_(Internet_culture)`. Most percent decoders can only be used in **perfectly encoded character sequences** where % has been encoded into `%25`.

Once again, please do not assume that the dataset is perfectly formatted or encoded.

Developers of The Wikimedia Foundation are using their decoder with the following two differences in contrast with the common URL decoder.

- They keep percent signs that are not followed by hexadecimal digits.
- They do not convert plus-signs to spaces.

We don't want you to reinvent the wheel, hence, **we offer you code snippets of a percent decoder in Java in the starter code**. You are allowed to use the percent-decoder code snippet without citing it in the references file.

Note that URL normalization usually requires other steps besides percent-encoding. In this project, we will ask you to only complete percent-encoding. Additionally, bear in mind that you can reorder the rules if you wish in your code, as long as percent decoding and handling dirty data are considered as the prerequisites for other rules.

Note that the URL Encoded Characters

(<http://www.degraeve.com/reference/urlencoding.php>) include whitespace characters such as tabs and spaces, which may change the number of columns in the records after percent-decoding.

2. **Handle dirty data:** Removing malformed data is a necessary step in data pre-processing. After applying percent decoding, you should split the record into columns and filter out the lines that don't have the four columns as expected.

When splitting the record into columns, you should use `String.split("\\s+")` in Java. The splitting algorithm will be applied as follows: occurrences of consecutive whitespace are regarded as one single separator, and the result will contain no empty strings at the start or end if the string has leading or trailing whitespace. For example, you should filter out the following records:

```
en 506 0      # 3 columns, separated by single spaces
en   506 0    # 3 columns, separated by single or consecutive spaces
```

In data processing, you need to make sure that your code must not crash when given malformed data.

3. **Desktop/mobile Wikipedia sites:** We will focus on English Desktop/Mobile Wikipedia pages, keep the rows only if the first column is exactly `en` or `en.m` (**case sensitive**) and exclude all others.
4. **Wikipedia namespaces:** There are many special pages in Wikipedia that are not actual Wikipedia articles. Remove these pages. A Wikipedia namespace is a set of Wikipedia pages whose names begin with a particular reserved word recognized by the MediaWiki software (followed by a colon).

For example, in the user namespace, all titles begin with "User:". In the case of the article (or main) namespace, in which encyclopedia articles appear, the reserved word and colon are **absent**. The details for namespaces are as follows for your reference, and in this project, we will only focus on namespace 0 (Main/Article):

Wikipedia namespaces			
Subject namespaces		Talk namespaces	
0	(Main/Article)	Talk	1
2	User	User talk	3
4	Wikipedia	Wikipedia talk	5
6	File	File talk	7
8	MediaWiki	MediaWiki talk	9
10	Template	Template talk	11
12	Help	Help talk	13
14	Category	Category talk	15
100	Portal	Portal talk	101
108	Book	Book talk	109
118	Draft	Draft talk	119
446	Education Program	Education Program talk	447
710	TimedText	TimedText talk	711
828	Module	Module talk	829
2300	Gadget	Gadget talk	2301
2302	Gadget definition	Gadget definition talk	2303
Virtual namespaces			
-1	Special		
-2	Media		

Figure 2: List of Wikimedia namespaces (Source (<https://en.wikipedia.org/wiki/Wikipedia:Namespace>))

Disambiguation pages, templates, navboxes, user pages, discussion pages, file pages, category pages, help pages, and Wikipedia policy pages are not articles. Keep pages only if they belong to namespace 0 (Main/Article) and those grouped into other namespaces should be excluded.

The Wikimedia Foundation offers users a JSON file listing all the namespaces (<https://en.wikipedia.org/w/api.php?action=query&meta=siteinfo&siprop=namespaces&format=json>) which need to be excluded, as well as a readable version (<https://en.wikipedia.org/w/api.php?action=query&meta=siteinfo&siprop=namespaces>).

Page titles may contain spaces visually, but their URLs will replace spaces with underscores and this rule also applies to our pageview data. **In our starter template, we will offer you a case-insensitive prefix blacklist similar to this:**

```
media:
special:
talk:
user:
user_talk: # instead of "User talk:"
wikipedia:
wikipedia_talk:
... # some title prefixes omitted
gadget_definition_talk:
```

As we mentioned, some titles use %3A or %3a instead of : due to percent-encoding. Before you apply this blacklist, such titles should have already been percent decoded into : .

Blacklist filtering can take less effort if you make use of helper utility libraries in Java. For example:

```
StringUtils.startsWithAny(title, prefixes);           # java with Apache Co
mons Lang library
```

Note that the standard namespace blacklist is **case insensitive**. You may need to change the code above to make it case insensitive.

5. **Wikipedia article title limitation:** Wikipedia policy states that if any article starts with an English letter, the letter must be capitalized. For example, the English Wikipedia article for iPad actually matches this URL:

```
https://en.wikipedia.org/wiki/IPad
```

Filter out all page titles that start with lowercase English characters. You may notice that some page titles don't start with English letters but digits, symbols, lowercase characters in other languages, etc., **DO NOT** filter them.

Be cautious and read the documentation whenever you want to use any built-in utility, for example, read the Javadoc carefully if you want to use `Character#isLowerCase(char ch)` (**Hint:** Check if they will match English lowercase letters only, or any Unicode lowercase letters.)

6. **Miscellaneous filename extensions:** Despite having already filtered pages in `File:` or `Media:` namespace, you may still get files instead of articles.
 - Media file names are case-sensitive, for example, `picture.jpg` and `picture.JPG` are not identical files. Nevertheless, we should use **case insensitive** matching to filter all the media files, ending with any of these suffixes: `png`, `gif`, `jpg`, `jpeg`, `tiff`, `tif`, `xcf`, `mid`, `ogg`, `ogv`, `svg`, `djvu`, `oga`, `flac`, `opus`, `wav`, `webm`
 - A favicon (short for favorite icon), is a file containing one or more small icons. Browsers that provide favicon support typically display a page's favicon in the browser's address bar (sometimes in the history as well) and next to the page's name in a list of bookmarks. Originally, the favicon was a file called `favicon.ico` placed in the root directory (e.g., `http://en.wikipedia.org/favicon.ico`) of a web site. Therefore, we should exclude any files with `.ico` suffix.

- The robots exclusion standard, also known as the robots exclusion protocol or simply `robots.txt`, is a standard used by websites to communicate with web crawlers and other web robots. The standard specifies how to inform the web robot about which areas of the website should not be processed or scanned. Robots are often used by search engines to categorize websites. Exclude any files with the `.txt` suffix.

In our starter template, we will offer you a case-insensitive suffix blacklist including the following filename extensions:

```
.png
.gif
.jpg
.jpeg
.tiff
.tif
.xcf
.mid
.ogg
.ogv
.svg
.djvu
.oga
.flac
.opus
.wav
.webm
.ico
.txt
```

7. Wikipedia Disambiguation: Disambiguation

(<https://en.wikipedia.org/wiki/Wikipedia:Disambiguation>) in Wikipedia is the process of resolving conflicts when one article title can have different meanings in different fields or scenarios.

For example, the word "CPU ([https://en.wikipedia.org/wiki/CPU_\(disambiguation\)](https://en.wikipedia.org/wiki/CPU_(disambiguation)))" can refer to a computer's central processing unit, a human enzyme, software updates in Oracle products such as the Oracle Database and Java and others.

Filter all disambiguation pages with the suffix `_(disambiguation)` (**case insensitive**), suffixes like `_%28disambiguation%29` should have already been decoded at this point. Don't filter any page with a specific topic such as `Numb_(Linkin_Park_song)`.

The `_(disambiguation)` is included in the suffix blacklist in starter template

8. Special pages: Finally, there are some special pages to exclude as well. Page titles that are exactly (**case sensitive**) in the following list should be excluded:

```
404.php
Main_Page
-
```

- `Main_Page` is the main entrance of the Wikipedia site.
- `404.php` is caused when a user attempts to follow a broken or dead link.
- `-` is a single hyphen-minus character, why does this get many accesses every hour? If you read the Java source code of Pageview

(<https://github.com/wikimedia/analytics-refinery-source/blob/master/refinery-core/src/main/java/org/wikimedia/analytics/refinery/core/PageviewDefinition.java>) and you will find the clue: - is used whenever any unknown project or article is encountered.

Output format: Output the remaining articles in the following format:

```
[page_title]\t[count_views]
```

Where `\t` is a *tab*.

For example:

```
Carnegie_Mellon_University 34
```

Name the output file exactly `output` , and follow these rules strictly:

- **If there are records from both desktop and mobile sites for the same page title, sum the accesses into one record.**
- Sort the output in **descending numerical order** of the number of accesses
- Break ties by ascending **lexicographical** order (based on the Unicode value of each character in the strings) of page titles. You can just use `String.compareTo(String anotherString)` in Java.

To get a full understanding of the big picture, **please continue reading the writeup before you start writing code.**

Test-driven development

When working with big data, it is often challenging to validate complicated results and analysis. Some beginners put off debugging until the full program is implemented, and when things go wrong, it can be hard to debug the monolithic program. Iterative divide and conquer problem-solving techniques using test-driven development (TDD) will help you reduce the defects and increase efficiency when solving complex problems.

Test-driven development is a software development process in which a developer starts by writing an initially failing test case, then refactors the code to pass the test, and repeat. Examples will be provided in the starter code later. Writing meaningful tests before implementing the code helps to clarify the boundaries of the problem and enables developers to solve complicated problems through a series of small steps.

The data pre-processing task above is complicated, consists of multiple rules, and should be solved with test-driven development. To help you get started, we provide you with the Java TDD starter template. You will use JUnit 5 library to solve the problem with test-driven development.

JUnit 5 is a unit testing framework for the Java programming language. JUnit has been playing an important role in Java development with test-driven development.

Let us take an example of how to work with JUnit 5.

Suppose you want to implement a complicated data filtering and transformation program; and you divide the problem into multiple methods. Let us assume the file hierarchy of the Maven project is as follows:

```
|-- pom.xml
|-- src
|   |-- main
|   |   |-- java
|   |       |-- Filter.java
|   |-- test
|       |-- java
|           |-- FilterTest.java
```

You can start by solving the first method. The first step is to define the signature of the first method.

```
// file: src/main/java/Filter.java
public class Filter {

    public static boolean containsCloud(final String record) {
        // it is okay to start with an incorrect solution
        // the method signature is what matters
        return false;
    }
}
```

The next step is to create test cases for this method. Make sure the test cases cover both positive and negative scenarios.

```
// file: src/test/java/FilterTest.java
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.*;

class FilterTest {
    @Test
    void testContainsCloud() {
        // positive
        assertTrue(Filter.containsCloud("cloud computing"));
        // negative
        assertFalse(Filter.containsCloud("on-premise"));
    }
}
```

With the test cases created, you can now run `mvn test` to confirm the tests will run and fail. The failure report will go like this:

```
[INFO] -----
[INFO] Running FilterTest
[ERROR] Tests run: 1, Failures: 1, Errors: 0, Skipped: 0, Time elapsed: 0.021 s
<<< FAILURE! - in FilterTest
[ERROR] testContainsCloud Time elapsed: 0.017 s <<< FAILURE!
org.opentest4j.AssertionFailedError: expected: <true> but was: <false>
    at FilterTest.testContainsCloud(FilterTest.java:9)

[INFO]
[INFO] Results:
[INFO]
[ERROR] Failures:
[ERROR]   FilterTest.testContainsCloud:9 expected: <true> but was: <false>
[INFO]
[ERROR] Tests run: 1, Failures: 1, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD FAILURE
[INFO] -----
```

JUnit reports what test cases have been run. Upon failures, JUnit's reports the expected result and the actual result per failed test case.

After creating the test cases, implement the unfinished method, for example:

```
public class Filter {
    public static boolean containsCloud(final String record) {
        return record.contains("cloud");
    }
}
```

Run `mvn test` again. If your program can pass the tests, the JUnit report will be as follows:

```
[INFO] T E S T S
[INFO] -----
[INFO] Running FilterTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.012 s -
in FilterTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
```

With these passed tests, you can be confident with this method and can move on to the TDD iteration for the next method. You can eventually implement a well-structured program that solves a complicated problem.

Now you have learned the basics of JUnit 5 and are ready to start working on your task. You need to first launch an AWS EC2 machine using a specified base Amazon Machine Image (or AMI) provided by us.

JaCoCo

Code coverage is the measure that describes how much source code is executed when the unit test finished. High code coverage indicates that more source code is executed during testing, which suggests that there is a smaller chance of the code containing uncaught bugs.

Java Code Coverage tools (a.k.a. JaCoCo) is an open-source toolkit for measuring and reporting Java code coverage.

Create a Code Coverage Report

To run unit tests and create a code coverage report, run the following command.

```
mvn clean test && mvn jacoco:report
```

With the JaCoCo plugin configured in `pom.xml` , running unit tests with JUnit (e.g., with the command `mvn test`) will create a coverage report in a binary format at the path `target/jacoco.exec` .

The command `mvn jacoco:report` will then convert the binary code coverage report into multiple formats (HTML, XML, and CSV).

You can open the navigable and readable HTML index page in the browser if you are developing the code on your local laptop.

```
./target/site/jacoco/index.html
```

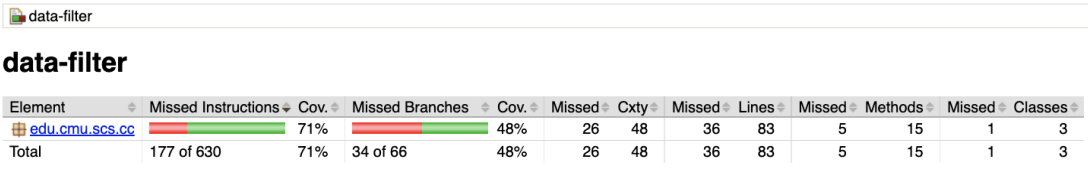


Figure: the index page of a sample report.

Interpret a Code Coverage Report

Jacoco analyzes Java byte code and maps the collected information back to source code to visualize the code coverage at line level granularity.

JaCoCo uses a set of different counters to calculate coverage metrics. In this project, you will focus on two basic coverage counters: Instructions and Branches.

Instructions (C0 Coverage)

The most fine-grained unit is per single Java byte code instruction. Instruction coverage provides information about the amount of code that has been executed or missed. This metric is completely independent of the formatting of the source code.

Branches (C1 Coverage)

JaCoCo also calculates branch coverage for all `if` and `switch` statements. This metric counts the total number of such branches in a method and determines the number of executed or missed branches. Exception handling is not considered as a branch in the context of this

counter definition.

Branch coverage is highlighted in the HTML report as follows:

- No coverage: No branches in the line has been executed (red diamond)
- Partial coverage: Only a part of the branches in the line have been executed (yellow diamond)
- Full coverage: All branches in the line have been executed (green diamond)

data-filter > edu.cmu.scs.cc > DataFilter

DataFilter											
Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	
checkSuffix(String[])	<div><div></div></div>	0%	<div><div></div></div>	0%	3	3	4	4	1	1	
checkFirstLetter(String[])	<div><div></div></div>	0%	<div><div></div></div>	0%	3	3	6	6	1	1	
checkAllRules(String[])	<div><div></div></div>	0%	<div><div></div></div>	0%	7	7	6	6	1	1	
static {...}	<div><div></div></div>	100%		n/a	0	1	0	3	0	1	
checkPrefix(String[])	<div><div></div></div>	100%	<div><div></div></div>	100%	0	3	0	6	0	1	
checkSpecialPage(String[])	<div><div></div></div>	100%	<div><div></div></div>	100%	0	3	0	4	0	1	
lambda\$sortRecords\$0(Map.Entry, Map.Entry)	<div><div></div></div>	100%	<div><div></div></div>	100%	0	2	0	4	0	1	
checkDomain(String[])	<div><div></div></div>	100%	<div><div></div></div>	100%	0	3	0	1	0	1	
sortRecords(TreeMap)	<div><div></div></div>	100%		n/a	0	1	0	4	0	1	
checkDataLength(String[])	<div><div></div></div>	100%	<div><div></div></div>	100%	0	2	0	1	0	1	
getColumns(String)	<div><div></div></div>	100%		n/a	0	1	0	1	0	1	
Total	73 of 422	82%	20 of 36	44%	13	29	16	40	3	11	

Figure: the code coverage page of a DataFilter class.

Information

How to achieve 100% branch coverage

```
14.     public static String trueOrFalse(boolean condition) {
15.         if (condition) {
16.             return "true";
17.         } else {
18.             return "false";
19.         }
20.     }
```

The coverage report above is generated based on the unit test below:

```
@Test
public void trueOrFalse() {
    App.trueOrFalse(true);
}
```

The `if (condition)` is always true with the test case we provide. Hence, the `return "false"` statement is never executed. To cover the other branch, a test case where `condition` is `false` needs to be included, as shown below.


```
@Test
public void trueOrFalse() {
    App.trueOrFalse(true);
    App.trueOrFalse(false);
}
```

Let's take a look at a more complex example.

```
22.     public static String and(boolean cond1, boolean cond2) {
23.         if (cond1 && cond2) {
24.             return "true";
25.         } else {
26.             return "false";
27.         }
28.     }
```

The coverage report above is generated based on the unit test below:

```
@Test
public void and() {
    App.and(true, true);
    App.and(false, true);
}
```

Even though both branches are covered by the unit test, the `if (cond1 && cond2)` is only partially covered. To fully cover this `if` statement, both `cond1` and `cond2` should be evaluated at least one time `true` and one time `false` during unit test. Therefore, a test case where `cond2` is `false` needs to be included, as shown below.

```
@Test
public void and() {
    App.and(true, true);
    App.and(false, true);
    App.and(true, false);
}
```

Solving Projects on AWS EC2

Many of the projects in this course will involve creating EC2 Virtual Machines and writing specific code on them.

Most of your work will rely on using a specified base Amazon Machine Image (or AMI) which has been pre-configured by the course staff to have all the tools necessary to complete each project.

Before getting started with the programming tasks, start with the following steps:

1. Provision a `t3.micro` EC2 instance in the `us-east-1` region using the Terraform templates we provided. The Terraform templates use our student AMI named "Cloud Computing Project Image (version)" with the ID `ami-04537cfe22bace769`. You should not launch it from the web console.

Information

Provision your EC2 instance with Terraform

During the course, we will use Terraform to automate the provisioning and orchestration of infrastructure. The most practical benefits are:

1. Skill training: In future projects, you will be required to use and submit Terraform to orchestrate more complicated infrastructures. Hence, we recommend that you start using Terraform.
2. Resource Management: You can run `terraform destroy` to terminate all the resources once you have finished the project, without forgetting any resources.
3. Tagging: Terraform provides a simple interface by which you can apply tags to all resources you will need to provision in this course.

Step-by-Step Guide

1. Please finish the *Infrastructure as Code* primer first.
2. Please install terraform (<https://www.terraform.io/intro/getting-started/install.html#installing-terraform>) on your personal computer.
3. Download and extract the Terraform configuration files using the following commands.

```
wget https://clouddeveloper.blob.core.windows.net/getting-started-with-cloud-computing/terraform/aws-ec2-fleet.tgz
```

```
tar -xvzf aws-ec2-fleet.tgz
```

4. Make sure `awscli` is installed and run the following command to configure `awscli` with your AWS credentials. You can find your credentials by navigating to the "My Security Credentials -> Access keys" tab within the AWS console.

```
aws configure
```

5. Create a file named `terraform.tfvars` and set the values of the variables defined at `variables.tf`. A sample of `terraform.tfvars` for this project:

```
key_name="<put your pem key name here>" # NOTE: DO NOT include the ".pem" file extension.  
project_tag="getting-started-with-cloud-computing"  
ami_id="ami-04537cfe22bace769"
```

6. Then run the following commands.

```
terraform init  
  
terraform apply
```

Note

1. If you cannot provision Spot instances, please contact AWS and request to increase your Spot instance limits ASAP. Before your limit increase request gets approved, you may use on-demand instances instead with the following Terraform template:

```
wget https://clouddeveloper.blob.core.windows.net/getting-started-with-cloud-computing/terraform/aws-ec2-on-demand.tgz
```

2. The templates will allow incoming traffic to TCP port 22 for SSH and port 80 for HTTP.
3. Always destroy the resources launched by terraform through terraform, instead of terminating the instances manually through the web console or CLI. **Note:** The core concept of AWS EC2 fleet is "target capacity", and an AWS EC2 fleet will continuously attempt to meet and maintain the target capacity. The EC2 Fleet Terraform template we provided has a target capacity as 1. If you terminate any instances (e.g., from the web console) without removing the fleet, the fleet will launch new instance(s) which may lead to over-budget.
4. When you want to take a break from working on this project, you should terminate your resources. The Terraform templates we provide you for this project do not allow you to "Stop" your instance and so you must "Terminate" your instances (using terraform) in order to save your budget. When doing so, we highly recommend you diligently back up your code before terminating your instances.

Hints: IDE plugins such as HashiCorp Terraform / HCL language support (<https://plugins.jetbrains.com/plugin/7808-hashicorp-terraform--hcl-language-support>) can be helpful to validate and format your Terraform code.

Danger

Troubleshooting Terraform

We provide you with a checklist of the most common pitfalls when you provision EC2 instances with Terraform.

Please read this section so that you build an index in your memory of the possible Terraform pitfalls. If you run into such issues later, you can revisit this section and search the solutions. To build an index means you do not have to memorize every detail, instead, please focus on the problem description and symptoms.

Use the correct credentials

Problem

Using incorrect credentials is the most common pitfall for beginners. When you run `aws configure` to set up the AWS credentials, you want to make sure that you are using the root credentials. You want to make sure that you are not using any

credentials that belong to an IAM role or an IAM user. Pay attention to this since you already created a pair of IAM credentials for Online Mob Programming to access Cloud 9 that you should not use in this project.

Symptoms

If you are using the wrong credentials, here are the examples of the errors that you will get.

The error message contains an IAM user or IAM role

```
* An error occurred (AccessDenied) when calling the GetRole operation:
User: arn:aws:iam::<account>:user/<username> is not authorized to perform: ...
```

Solution

Use the root credentials of your account which you can find or create from the web console. Visit the security credential page (https://console.aws.amazon.com/iam/home?region=us-east-1#/security_credential), click "the Access keys (access key ID and secret access key)" to find or create the credentials.

Run "terraform init" before you run "terraform apply"

Symptom

No suitable provider version installed

```
* provider.aws: no suitable version installed
version requirements: ...
versions installed: none
```

Solution

You want to ensure that you run `terraform init` to initialize a Terraform working directory before you run `terraform apply`.

Information

Troubleshooting EC2 fleets

The Terraform script provided to you creates an EC2 fleet. A fleet is a group of instance(s). You can define separate configurations for Spot and on Demand instances using EC2 fleet. To check the EC2 fleets you have active run the command:

```
# Fill in your credentials
$ aws configure
# Print all the fleets
aws ec2 describe-fleets
{
  "Fleets": [
    {
      "ActivityStatus": "...",
      "CreateTime": "...",
      "FleetId": "<FLEET_ID>",
      ...
    }
  ]
}

# Inspect the details of instance(s) of an EC2 fleet by fleet ID
aws ec2 describe-fleet-instances --fleet-id <FLEET_ID>
{
  "ActiveInstances": [
    {
      "InstanceId": "<instance_id>",
      "InstanceType": "...",
      "SpotInstanceRequestId": "<spot_request_id>",
      "InstanceHealth": "healthy"
    }
  ],
  "FleetId": "<FLEET_ID>"
}
```

If the fleet `ActivityStatus` shows `error`, you will need to troubleshoot the error. If you do not see an error section at the end of `aws ec2 describe-fleets` command run:

```
aws ec2 describe-fleet-history --fleet-id "<FLEET_ID>" --start-time <y
yyy-mm-ddThh:mm:ss.000Z>
{
  "HistoryRecords": [
    {
      "EventInformation": {
        ...
      },
      {
        "EventInformation": {
          "EventDescription": <ERROR DETAILS>
        }
      }
    }
  ]
}
```

You should take a look at the `EventInformation.EventDescription` value to get information on the error that resulted in the EC2 instantiation to fail.

Below is an example error message if you use an invalid key pair (e.g., by mistakenly appending `.pem` when specifying the key pair name):

```
"Repeated errors have occurred processing the launch specification \"t3.micro, ami-04537cfe22bace769, Linux/UNIX, us-east-1a while launching spot instance\". It will not be retried for at least 13 minutes.
Error message: com.amazonaws.services.ec2.model.AmazonEC2Exception: The key pair <Invalid key pair> does not exist
(Service: AmazonEC2; Status Code: 400; Error Code: InvalidKeyPair.NotFound; Proxy: null)"
```

You might also need to delete the launch template if it was created when running terraform apply. You can do so by running:

```
aws ec2 delete-launch-template --launch-template-name student_image_launch_template
```

You can learn more about EC2 fleets here
(<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-fleet.html>)

Danger

T-series burst performance

AWS has a credit configuration called `unlimited` mode for the T family. This option allows you to burst the CPU performance on an instance unlimitedly, regardless of the CPU credits. If you are not careful and provision the `t3.micro` instance with a credit-configuration of `unlimited` you are prone to run out of budget very quickly. You may decide whether you want to disable the unlimited mode, or keep it enabled while exercising caution to the CPU credits.

To check if your instance has burst performance please run:

```
aws ec2 describe-instance-credit-specifications --instance-id <INSTANCE_ID>
{
  "InstanceCreditSpecifications": [
    {
      "InstanceId": "i-1234567890abcdef0",
      "CpuCredits": "unlimited"
    }
  ]
}
```

You can find the `INSTANCE_ID` on the AWS portal under the `Instance summary` tab.

As you can see the `CpuCredits` have been configured to be `unlimited` meaning the instance is burstable. To convert a burstable instance to a `standard` instance run:

```
aws ec2 modify-instance-credit-specification --region us-east-1 --i
nstance-credit-specification "InstanceId=<INSTANCE_ID>,CpuCredits=stan
dard"
```

If you wish to learn more about Burstable instances please visit here (<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/burstable-performance-instances-unlimited-mode-concepts.html>).

2. Unlike the previous tasks in this project, you can no longer use the `ubuntu` or `ec2-user` to SSH into an instance. Every week, you will have to follow these steps.

1. Open the URL `http://<your-ec2-dns>` in your web browser. Please note that you must use HTTP instead of HTTPS. Note that the "open address" link in the Public IPv4 DNS section of the EC2 instance from the AWS console uses HTTPS instead of HTTP.
 2. Enter your submission username and submission password (from the top of this page, click the "Show Submission Password" button to view your submission password). Then click the "Launch Project" button of **Getting Started with Cloud Computing** section.
 3. After you click the submit button, it may take several minutes before the page automatically jumps to `http://<your-ec2-dns>/logs`, please be patient. You will see logs indicating files are being transferred and your environment is being prepared. Do not log into the instance until the log tells you to do so. If an error is reported, please create a private Piazza post to inform the staff.
 4. Once the installation finishes, the log will likely tell you the username and the SSH private key that you can use to log in to the instance. Alternatively, it may tell you that you have not submitted or validated your AWS accounts and will not let you proceed.
3. If Step 2 was successful, you should be able to log into the instance using the PEM/PPK file as the user `clouduser` with the following command: `ssh -i {path_to_your_pem_file} clouduser@<your-ec2-dns>`
4. It is a good strategy in our course to stop your instance (http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/Stop_Start.html) to save the budget and restart it to continue your work. Keep in mind that:
1. Stopping an instance is different from terminating it. **Remember to terminate all resources by the project deadline** to avoid getting a tagging penalty when the next project starts.
 2. You will not get charged by EC2 usage for a stopped instance or data transfer fees, but you will still pay for the storage of the EBS volumes.
 3. When you restart a stopped instance, note that the DNS name will most likely change. You need to update your commands or configuration when you use SSH or SFTP. There is no need to visit `http://<your-new-ec2-dns>` in your web browser again. **Warning: Re-login from `http://<your-new-ec2-dns>` will reset the project folder.**
5. In case you corrupt project files and need to recover them, you can revisit `http://<your-ec2-dns>` in your web browser and click the submit button to reinitialize the project folder. The previous `runner.sh` will be backed up, named as

`runner.sh.timestamp`, and a clean `runner.sh` will be created. Other project files will be overwritten, such as `submitter`. **Also, remember your source code files (java files) will be overwritten, so please backup your code.**

Your task

Information

Suggestions on working with Remote Servers

We recommend that you do not edit remote code directly, **unless you have the expertise, and are confident about what you are doing**. Instead, we recommend that you edit code locally with an IDE on your laptop or personal computer. You can then share your code between your personal laptop and remote instance by using a **private** GitHub repository or a tool like `scp`. Once you are confident, you can then upload your code to a remote VM for testing and deployment. *Note: the teaching staff does not provide support for local environment issues on your personal computer. If you attempt to run your project code locally and run into issues, please use the remote VM you provisioned with the course VM image.*

The possible approaches include but are not limited to:

1. Use a **private** GitHub repository.
2. Use the `scp` command to copy your code between your local machine and the remote server.
3. Use `scp` and make sure to copy the `output` file from your local machine to the remote server.
4. IDEs (Integrated Development Environments), such as IntelliJ (<https://www.jetbrains.com/help/idea/tutorial-deployment-in-product.html>), may provide features and/or plugins to easily work with remote servers.

If you have trouble working with remote servers, feel free to create a post on Piazza.

In this task, we provide you with the code template

`src/main/java/edu/cmu/scs/cc/DataFilter.java` and
`src/main/java/edu/cmu/scs/cc/Main.java`.

You need to modify `Main.java` and make the try-with-resources statement encoding aware.

`DataFilter.java` defines a set of filter methods that you need to implement. The code template has already implemented the other requirements:

- Merge both desktop and mobile sites for the same page title if any.
- Sort the output in descending numerical order of the number of accesses and break ties by ascending lexicographical order.
- Output the results into a file named as exactly `output`.

Hence, you should focus on implementing the filter methods by following a test-driven development approach. `src/test/java/edu/cmu/scs/cc/DataFilterTest.java` contains the test cases of the `DataFilter` class. We provide you with a set of test cases for the first several filter methods, and you need to add the tests for the rest of the methods. Note that the test cases should cover both positive and negative cases.


```
@Test
void checkSuffix()
@Test
void checkFirstLetter()
@Test
void checkAllRules()
```

After you create the test cases in `DataFilterTest.java` . Change directory to `/home/clouduser/workspace/` and run `mvn test` to run the test cases and validate your programs. Of course, you will not pass all the tests now.

```
# change directory to workspace
cd /home/clouduser/workspace

# run the test
mvn test

[ERROR] ...

[INFO]
[INFO] Results:
[INFO]
[ERROR] Errors:
[ERROR]   DataFilterTest.checkAllRules:88 Runtime add test cases on your own
[ERROR]   DataFilterTest.checkDataLength:32 » Runtime To be implemented
[ERROR]   DataFilterTest.checkDomain:44 » Runtime To be implemented
[ERROR]   DataFilterTest.checkFirstLetter:83 Runtime add test cases on your own
[ERROR]   DataFilterTest.checkPrefix:66 » Runtime To be implemented
[ERROR]   DataFilterTest.checkSpecialPage:54 » Runtime To be implemented
[ERROR]   DataFilterTest.checkSuffix:78 Runtime add test cases on your own
[INFO]
[ERROR] Tests run: 9, Failures: 0, Errors: 7, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD FAILURE
[INFO] -----
```

You will get `BUILD FAILURE` until your program can pass all the tests. Your task is to implement the unfinished methods in `DataFilter.java` and pass all test cases.

Keep iterating the development of `DataFilter.java` until you get `BUILD SUCCESS` .

```
...
[INFO] Running edu.cmu.scs.cc.DataFilterTest
[INFO] Tests run: 9, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.02 s -
in edu.cmu.scs.cc.DataFilterTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 9, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
```

TDD emphasizes writing unit tests ahead of writing the code. Writing unit tests before writing your code helps you to structure your code in a way that easily facilitates testing. For example, in this task you divided your code into shorter methods that you might write as one long complicate method otherwise. TDD helps you separate the concerns and make your code clean, easy-to-read, and robust.

Once you get `BUILD SUCCESS`, your next task is to achieve 100% code coverage for the `DataFilter` class. You don't need to get 100% code coverage for other classes, such as `PercentDecoder` and `Main`. You can generate a code coverage report with JaCoCo using the following command.

```
mvn clean test && mvn jacoco:report
```

You can then open the navigable and readable HTML index page in the browser if you are developing the code on your local laptop.

Once you get 100% code coverage on the `DataFilter` class, you will be ready to submit your task now.

How to submit

In the `filter()` method of `./runner.sh`, we provide you with the command to execute the data filter program. You are **NOT** allowed to change this method.

1. You can run a `submitter` to submit this task and check your results on the Sail() platform.

```
export SUBMISSION_USERNAME="your_submission_username"
export SUBMISSION_PASSWORD="your_submission_password"
./submitter
```

Type `./submitter -h` to get the usage example.

2. Remember that your submission password can be found by clicking on the button at the top of this page.
3. After running the `./submitter` command, you can check your submission result and feedback on the submission tab of the project on the Sail() platform.

There is no limit on the number of submissions allowed before the project deadline. However, submissions must be separated by at least 60 seconds.

Try not to exceed the recommended budget specified for this project module.

Danger

Be cautious about implicit reliance on your environment

Your code should work well and its behaviors should be consistent on different systems. You are allowed to write and test the code on your laptop first. **To help you learn best practices, we will test your code in a different environment.**

If your code seems to run well locally but behaves differently upon submission, read this panel carefully before you create posts on Piazza.

Failing to realize the potential difference among development, testing, and production environments can lead to pitfalls. If your code behaves well in your development environment, it does not guarantee that your code will work perfectly in other environments.

You should make your code independent from unpredictable/uncontrollable external environments. Please pay attention to **encoding-aware I/O operations, newlines, and locale** in your code and always **handle them explicitly instead of relying on the system default setting**. Besides, watch out for **versions and absolute/relative paths**. All the following topics can be pitfalls of this project.

Locale

On POSIX platforms such as Linux, locale identifiers are defined in this format:

```
[language[_territory][.codeset][@modifier]].
```

You can get the locale setting on your machine with command `locale`.

Locale on a Linux system can determine the default encoding in locale-aware programs. Locale is also an important topic in Internationalization/Localization, which will make a difference in date, time, number, currency, and so on. In this project, we will explore how the default encoding may change the behavior of programs and the practice to make the programs independent from the system default encoding.

Encoding-aware I/O

Let us start with File I/O.

```
BufferedWriter bw = new BufferedWriter(new FileWriter(OUTPUT));
```

This seems good, and it will work correctly on your laptop in most cases.

If you run Findbugs (<http://findbugs.sourceforge.net/>) upon this snippet, you will get this SEVERE warning:

Reliance on default encoding

Found a call to a method which will perform a byte to String (or String to byte) conversion, and will assume that the default platform encoding is suitable. This will cause the application behavior to vary between platforms. Use an alternative API and specify a charset name or Charset object explicitly.

Similarly, the following snippet in python 2/3 relies on the value returned by `locale.getpreferredencoding(False)`:

```
with open(fname, "r") as f:
```

This bug can be easily ignored when you only run and test your code in your development environment, in which the default encoding will be set to UTF-8. However, if the production environment has another default encoding, your code will produce

unexpected output. If you see weird output like `????`, replacement characters ([https://en.wikipedia.org/wiki/Specials_\(Unicode_block\)#Replacement_character](https://en.wikipedia.org/wiki/Specials_(Unicode_block)#Replacement_character)) or empty boxes in the future, check your encoding! Many traditional encodings will change unsupported code points into question marks, such as ISO-8859-1, a.k.a. Latin-1. Failing to deal with encoding when handling input can cause more severe failure, as the program may crash. For example, processing UTF-8 encoded input as ASCII in Python 3 may cause `UnicodeEncodeError: 'ascii' codec can't encode character: ordinal not in range(128)`.

You must set encoding explicitly when your program converts an input stream of bytes to strings, and when your program converts strings to an output stream of bytes. **You CANNOT pass all the test cases if you fail to make your program encoding-aware.**

Here are examples to handle encoding in an explicit fashion:

For example, in Java 8:

```
BufferedReader br = new BufferedReader(  
    new InputStreamReader(new FileInputStream(INPUT), StandardCharsets.UTF_8));  
PrintWriter printWriter = new PrintWriter(new File(OUTPUT), "UTF-8");  
Stream<String> stream = Files.lines(  
    Paths.get(OUTPUT), StandardCharsets.UTF_8)
```

In Python 3:

```
# for more information, read https://docs.python.org/3/howto/unicode.html  
with open(fname, "rt", encoding='utf-8') as f:
```

In Python 2.7:

```
# for more information, read https://docs.python.org/2/howto/unicode.html  
with io.open(fname, "wt", encoding='utf-8') as f:
```

You can set the encoding of Standard I/O in Python with the environment variable `PYTHONIOENCODING`, note that it only applies to Standard I/O (stdin/stdout/stderr) but not File I/O:

```
$ PYTHONIOENCODING=UTF-8 python3 script.py
```

Okay, let's try another example! Will the following code produce the same output in multiple environments?

```
System.out.println(str);
```

No! Even `System.out`, as a `PrintStream`, is system dependent! To help you overcome this hurdle and pass all the tasks in this project, instead, please follow the best practice and we provide you with an example:

```
Scanner in = new Scanner(  
    new BufferedInputStream(System.in), "UTF-8");  
PrintWriter out = new PrintWriter(  
    new OutputStreamWriter(System.out, "UTF-8"), true);
```

If you cannot get a full score upon submission and you wonder if it is caused by encoding, you may test your program on your instance by running your program with different locales:

```
LC_ALL=en_US.UTF-8 ./your_program # `locale charmap` will return `UTF-8`  
LC_ALL=C ./your_program           # `locale charmap` will return `ANSI_X  
3.4-1968` (ascii)
```

Encoding-awareness covers not only I/O but also the source code. If there are UTF-8 characters in the source code, including the comments, the Java compiler can break if the system default encoding does not support UTF-8. You should set the source code encoding using `javac -encoding utf8 *.java` or use the Maven approach (<https://stackoverflow.com/questions/3017695/how-to-configure-encoding-in-maven>). Although the source code encoding in Python is independent of system default encoding and Python 3 supports UTF-8 by default, Python 2 will default to ASCII and you must follow this Python Enhancement Proposal (<https://www.python.org/dev/peps/pep-0263/#defining-the-encoding>) to define a source code encoding in Python 2.

If you are passionate about mining insight from data, keep in mind that encoding-unaware data processing may fail to produce the expected output **silently**. If you are enthusiastic about API design, please make your library encoding-aware. Even in one of the most widely-used utility projects, Apache Commons (<http://commons.apache.org/>), the contributors feel the pain caused by encoding reliance. Apache Commons IO (<https://commons.apache.org/proper/commons-io/index.html>), after 12 years since the initial release and more than 4 years since the encoding reliance was reported, the library finally deprecates all the encoding-unaware methods in the 2.5 (https://commons.apache.org/proper/commons-io/upgradeto2_5.html) release. Encodings can be more vital when working with web applications and databases, bear this in mind when you work on them in the future.

"It does not make sense to have a string without knowing what encoding it uses." -- Joel Spolsky

Newline (EOL)

Believe it or not, `System.out.println(str)` has yet another problem. Try your code on both Linux and Windows platforms, and compare the md5sum of the standard output -- and they will be different.

A newline, also known as 'end of line' (EOL), is a special character or sequence of characters signifying the end of a line of text and the start of a new line. The actual codes representing a newline vary across operating systems, which can be a problem when exchanging text files between systems with different newline representations.

Systems based on ASCII or a compatible character set use either LF (Line feed, `'\n'`), CR (Carriage return, `'\r'`), or CR followed by LF (CR+LF, `'\r\n'`). Unix and Unix-like systems, e.g. Linux, Mac OS X, etc., use `'\n'` while Windows use `'\r\n'`.

You may edit your code on your laptop during your development and testing phases, however, keep in mind if you are a Windows user, the code may seem "visually" correct but will behave differently on a Unix or Unix-like System. **Make sure you set the EOL to UNIX format in your editor, especially when you write bash scripts locally, or your scripts might break when you upload the script or you "copy & paste" the code to the remote instance.** You may use `cat -e filename` to make sure there is no CR (Carriage return, '\r') in your code (there will be ^M at the end of each line if CR exists).

It is also recommended to always handle newlines explicitly in your code.

For example, in Java 8:

```
//OS-dependent code
printWriter.println(entry.getKey() + "\t" + entry.getValue());
printWriter.printf("%s\t%s\n", entry.getKey(), entry.getValue());
printWriter.print(entry.getKey() + "\t" + entry.getValue() + System.lineSeparator());
//System.lineSeparator() can be replaced with System.getProperty("line.separator");

//OS-independent code with identical output in various operating systems
printWriter.print(entry.getKey() + "\t" + entry.getValue() + "\n");
```

In Python 3:

```
# OS-dependent code
output = open('output', 'wt', encoding='utf-8')
output.write(line + '\n')
# Python automatically translates "\n" to the proper newline of the current OS.
# it will be converted to "\r\n" in Windows

# To make it OS-independent, we can specify the newline character explicitly.
# OS-independent code
output = open('output', 'wt', encoding='utf-8', newline='\n')
output.write(line + '\n')
```

Versions & Compatibility

Bash

If you want to use `awk` to do regex/string operations ignoring case, `IGNORECASE` seems to be a good idea. It can be set to a non-zero value.

```
awk 'BEGIN{IGNORECASE = 1;}...' FILENAME
```

The command will work well on our student AMI, but it will break on Mac OS X. Because `IGNORECASE` is implemented in GNU Awk (a.k.a. `gawk`) but not in Mac's `awk`. Alternatively, `tolower()` is supported by both versions.

Similarly, the BSD `grep` on Mac is different from the GNU `grep` on our student AMI. There is also no `wget` installed on Mac OS X and an "imperfect substitute" is `curl -O`. Consider using Homebrew (<https://brew.sh/>) if you need to install GNU tools on a Mac

platform.

Whenever using tools among different systems, remember that there can always be various versions. For example, a lot of popular Unix-like tools have their GNU versions (<http://directory.fsf.org/wiki/GNU>). Remember to check the versions first before you start using tools and choose more compatible solutions if possible.

Python

If you are a Python user, specify your Python version explicitly.

Use either `python2` or `python3` to run python, do not use `python`, and rely on the default one in the system environment. This also applies to the usage of `pip2` and `pip3` over `pip`. Alternatively, set the proper shebang ([https://en.wikipedia.org/wiki/Shebang_\(Unix\)](https://en.wikipedia.org/wiki/Shebang_(Unix))) line to specify the interpreter to use and run your script as an executable, such as `./script.py`. If you decide to use shebang, be extremely careful because some shebang lines can still break in different system environments, such as `#!/usr/bin/python3` because the python installation path may vary across platforms. `#!/usr/bin/env python3` is the best way to define the shebang for portability.

Java

When you are trying to run a class compiled with Java 8 into a lower JRE, you will get an "Unsupported major.minor version Error". We are using Java 8, GNU Awk, and GNU grep in our course.

If you are in doubt about any versioning issue of tools/languages/libraries we support, please create a private post on Piazza.

Absolute/Relative Paths

DO NOT write code like the example below because your code won't be able to execute anywhere other than the absolute path.

```
python3 /home/clouduser/workspace/script.py
```

Replace any absolute path with a relative one!

Conclusion

To sum up, you should always pay attention to the potential causes of implicit reliance in your implementation, and it will help you a lot during **this project, this semester, and hopefully in your career.**

Danger

Performance matters when processing big data, hence, the data filter program should not have a large overhead because the overhead will get amplified as the data size scales. You are required to make your program efficient and get rid of any overhead, otherwise, it will cause a "Grading timeout" with a 0 score. For example, do not compile java code in `filter()`.

Besides, we block most internet connections during the grading.

1. Do not use any Maven remote repositories other than the *Maven central repository*.
2. Do not send any HTTP/HTTPS requests in your code.

If you get a "Grading timeout", do not simply resubmit without changing your code as the result will be the same.

Data Analysis

Data Analysis

After filtering the data, you are expected to analyze your results and answer a set of questions, which are present in the file `/home/clouduser/workspace/runner.sh` and `/home/clouduser/workspace/data_analysis.ipynb` on your instance. To complete this module, do the following:

1. Go to the project folder located at `/home/clouduser/workspace`
2. The project folder consists of the following files: the runner script `runner.sh`, `submitter` to submit your solutions and `reference` to cite the links and students you get help from, and other data files for you to work on. You have permissions to edit the `runner.sh`, `data_analysis.ipynb` and `references` files.
3. Edit the script `runner.sh` and replace the `:` in each question with commands/code used to answer the questions (`q1-q5`). We recommend using bash scripting, but it is not required.

Do not move any of the provided files.

If you are using any external scripts, ensure you are calling the correct scripts from `runner.sh`. Please ensure you are placing all your code in the same folder and also assume that the dataset is present in the current folder.

When you need to access the dataset in your code assume that it is present in the working directory (i.e., do not use any absolute or relative paths for accessing the dataset -- we will auto-grade it later in a single jailed environment).

e.g. Instead of `head /home/clouduser/workspace/output`, use `head output` OR `head ./output`

You can, and should, test the execution of the `runner.sh` file after you finish these questions. You can do this by typing `./runner.sh` into your terminal window when you are in the `/home/clouduser/workspace/` directory. When you do this testing, ensure the data file `pageviews-20180310-000000.gz` is in the same directory, otherwise you may not see any output as you would have no data to analyze locally!

4. Edit the `data_analysis.ipynb` to implement your solution for `q6 - q9` using Jupyter Notebook. You may need to edit the `q6 - q9` in `runner.sh` so that they are encoding-aware. A Python 3 virtual environment has been provided under `/home/clouduser/virtualenv`. You can activate the virtual environment and start the Jupyter server with the following commands.


```
source /home/clouduser/virtualenv/bin/activate
jupyter notebook --no-browser
```

You will get prompted with the following message:

```
Copy/paste this URL into your browser when you connect for the first time,
to login with a token:
http://localhost:8888/?token=<token>
```

5. Then you need to create a secure connection between your local computer and the EC2 instance with SSH tunneling.

```
ssh -i <your pem key> -L 8888:localhost:8888 clouduser@<ec2-dns>
```

You can now access the Jupyter server by visiting `http://localhost:8888/?token=<token>` in your browser.

6. Edit the text file `references` to include all the links that you referred to for completing this project. Also, include the submission usernames of all students who you might have discussed general ideas with when working on this project in the same file. This is extremely important.

We analyze many aspects of your AWS and the Sail() platform usage, as well as automated code similarity detection tools.

NOTE: Citing resources and having big picture discussions with other students does not excuse cheating. **You are not allowed to look at or discuss code with another person. Similarly, you are not allowed to use any code snippet from anyone or anywhere on the Internet.**

When in doubt, please post privately on Piazza.

7. You can run and check your answers by typing `./runner.sh` from the workspace folder. **Note: Make sure that the Python 3 virtual environment is activated.** In case you get a **permission denied** exception please run `chmod +x runner.sh`. The `setup()` function in `runner.sh` will convert your `data_analysis.ipynb` to a python script `data_analysis.py`. `q5 - q9` in `runner.sh` will then use the python script to generate the output.

Running this script should print out the answers to all the questions you have answered. Please ensure the answers are printing correctly before you submit your answers.

If you want to focus on one question and get a readable unescaped answer, we offer you `./runner.sh -r <question_id>` from the workspace folder to run one single question. Type `./runner.sh -h` to get the usage example.

8. Once you have completed all the questions, you can submit the answers to the evaluation system using the `submitter` executable. Run the executable using the command `./submitter` from the auto-grader folder. **Note: Make sure that you have activated the Python 3 virtual environment when submitting.**

```
export SUBMISSION_USERNAME="your_submission_username"
export SUBMISSION_PASSWORD="your_submission_password"
./submitter
```

Remember that your submission password can be found by clicking on the button at the top of this page. **Make sure you open port 80 for incoming HTTP traffic before you proceed.**

Type `./submitter -h` to get the usage example.

What to Submit

When submitting, please make sure the following source code is in your workspace folder:

- Code to convert `pageviews-20180310-000000.gz` to output used in `filter()` in `runner.sh`.
- The `output` file generated and used to get the answers to each of the data analysis question.
- Make sure that the source code is present (`.sh` , `.java` , `.py` , etc.).
- Your code should demonstrate good style and discipline. Every time you write code, remember that someone else is going to read it. Make your code readable, self-explanatory, and standardized, which will avoid many potential bugs. **Hard to read code of poor quality will lead to a loss of points during manual grading.**
- **Lack of comments, especially for complicated code, will lead to a loss of points during manual grading.**

Performance matters when processing big data, hence, the data filter program should not have a large overhead because the overhead will get amplified as the data size scales.

Danger

You are required to make your program efficient and get rid of any overhead, otherwise, it will cause a "Grading timeout" with no score.

If you get a "Grading timeout", do not simply resubmit without changing your code as the result will be the same.

If your submitted code does not produce the same results like the ones on your local computer, consider the topics mentioned in "Be cautious about implicit reliance on your environment".

Conclusion

Danger

Terminate Resources in a Timely Manner

Run `terraform destroy` to conveniently terminate all the resources managed by Terraform in the data pre-processing and analysis task.

Information

In this project, here is what you have accomplished:

1. You experimented with cloud resource provisioning with multiple cloud service providers.
2. You experiment with the cloud-based development and deployment workflow.
3. You practiced how to complete hands-on activities with solution-feedback cycle.
4. You quickly studied diverse topics within a short timeframe (i.e., a week), and transferred your learning to complete hands-on tasks with real-world scenarios:
 1. tools (e.g., cloud platforms, Maven, Terraform, JUnit, JaCoCo, Jupyter Notebook, Pandas, Linux tools such as awk and grep, etc.)
 2. practices (e.g., test-driven development, code coverage, encoding-aware I/O, etc.)
 3. processes (e.g., budget, tagging and lifecycle management for cloud resources, etc.)

In this project, we provided you with code skeletons (in Java, Bash, and Python) with sufficient comments, self-explanatory code, consistent indentation, good code style, etc., so that you can learn from good examples. By starting with a very structured project, it helps you to pace yourself to get familiar with how to work with cloud and solve logistic issues if any. It also helps you build the momentum to solve projects with increasing difficulty.

As you may notice, as the scaffolding in this project is well-structured by design, there may not be too much space left for you to improve the code quality. However, the cloud computing course is all about hands-on experience solving real-world problems, and there is not as much scaffolding in the real world. Starting from the next projects, you may notice that the scaffolding will keep decreasing. You will need to rely on yourself to write code in high quality, or you may lose points during manual grading.

You will get increasingly unstructured projects through the course, and your skills to write code with high quality will be challenged fully through the learning of the cloud computing course.

The future projects will share a similar workflow and learning experience, with increasing challenges and decreasing scaffolding. You may use this project to evaluate your readiness to complete this course.

Project Survey

Project Survey

Please leave us feedback for this project here. This will help us strengthen this project for future offerings.

End of Project Survey

 jingyiwu098@gmail.com (未分享) [切换帐号](#) 

*必填

Course Identifier *

This field is pre-filled, please do not modify it.

s22-15619

Module Identifier *

This field is pre-filled, please do not modify it.


getting-started-with-cloud-computing

Sail() Username *

您的回答

Learning *

	Very little	Less than expected	Typical	More than expected	Quite a lot
How much did you learn from this project?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

 How many hours did you spend on this project? *

Please enter a positive integer

