

Aim:

Implementing Python program to parse the given timelog data files to get how many hours and minutes the author spent in each file on cloud platform using python.

Objective:

Design a cloud web page to select the time log data file from your local machine and then read the time log data file and report the result.

Introduction:

AWS (Amazon Web Services) is a comprehensive, evolving cloud computing platform provided by Amazon that includes a mixture of infrastructure as a service (IaaS), platform as a service (PaaS) and packaged software as a service (SaaS) offering.

Flask is an API of Python that allows us to build up web-applications. It was developed by Armin Ronacher. **Flask's** framework is more explicit than Django's framework and is also easier to learn because it has less base code to implement a simple web-Application.

Implementation:

Using Flask build a web-application for python program(tlparser.py). The below is the code for tlparser_app.py

tlparser_app.py:

```
#!/usr/bin/env python
# coding: utf-8
import os
from flask import Flask, render_template, request, redirect, url_for
#from werkzeug.utils import secure_filename
from tlparser import *

app = Flask(__name__)
app.config['UPLOAD_PATH'] = 'uploads'
@app.route('/')
def home():
```

```

    return render_template('index.html')

@app.route('/', methods = ['POST'])
def upload_tl_file():
    uploaded_tl_file = request.files['file']
    if uploaded_tl_file.filename != '':
        tl_file = os.path.join(app.config['UPLOAD_PATH'], uploaded_tl_file.filename)
        #filename = secure_filename(f.filename)
        uploaded_tl_file.save(tl_file)
        total_work_time_hours, total_work_time_mins = tlparsed(tl_file)
        return render_template("index.html", total="Total Work Time: " +
str(int(total_work_time_hours)) + " Hours and " + str(int(total_work_time_mins)) + " Minutes",
                                filename=uploaded_tl_file.filename)
    else:
        #render_template("index.html")
        redirect(url_for('home'))
#b = open(os.path.join(app.config['UPLOAD_PATH'], filename), 'r')
#output = tlparsed(b)
if __name__ == '__main__':
    app.run(host='0.0.0.0', debug=True)

```

tlparser.py:

```

import argparse
import datetime
import re

```

```

def tlparsed(tl_file):
    with open(tl_file, 'r') as f:

```

```

#all_lines = f.readlines()

all_lines = f.read().splitlines()

time_log_flag = False

total_work_time_seconds = 0

time_log_pattern = 'Time Log'

work_date_pattern = '[0-9]{1,2}/[0-9]{1,2}/[0-9]{2,4}'

work_times_pattern = '\d{1,2}:\d{1,2}[ap]m\s+-\s+\d{1,2}:\d{1,2}[ap]m'

for index, line in enumerate(all_lines):

    print("Line #%d: %s" %(index+1, line))

    if (not time_log_flag):

        time_log_found = re.match(time_log_pattern, line, re.IGNORECASE)

        if time_log_found:

            time_log_flag = True

            print(f"Time Log Found: {time_log_found}\n")

        else:

            print(f"Ignoring Line {index+1}: {line}\n")

    else:

        work_date_found = re.search(work_date_pattern, line, re.IGNORECASE)

        work_times_found = re.search(work_times_pattern, line, re.IGNORECASE)

        #print("Work Date Found: ", work_date_found)

        #print("Work Times Found: ", work_times_found)

        if work_date_found and work_times_found:

            work_start_date_str = work_date_found.group(0)

            previous_work_date_str = work_start_date_str

        elif work_times_found and (not work_date_found):

            work_start_date_str = previous_work_date_str

        print("work start date: ", work_start_date_str)

```

```

print("work start date length: ", len(work_start_date_str))

if work_times_found:
    work_times_str = work_times_found.group(0)
    print(work_times_str)
    work_times = work_times_str.upper().split("-")
    work_start_time_str = work_times[0].strip()
    work_end_time_str = work_times[1].strip()
    work_start_time_of_day = work_start_time_str[-2:].upper()
    work_end_time_of_day = work_end_time_str[-2:].upper()
    if (work_start_time_of_day == "PM") and (work_end_time_of_day == "AM"):
        if (len(work_start_date_str.split("/"))[-1]) == 2):
            work_end_date = datetime.datetime.strptime(work_start_date_str, '%m/%d/%y')
+ datetime.timedelta(days=1)
            work_end_date_str = work_end_date.strftime('%m/%d/%y')
        elif (len(work_start_date_str.split("/"))[-1]) == 4):
            work_end_date = datetime.datetime.strptime(work_start_date_str, '%m/%d/%Y')
+ datetime.timedelta(days=1)
            work_end_date_str = work_end_date.strftime('%m/%d/%Y')
        else:
            work_end_date_str = work_start_date_str
    work_start_date_time_str = work_start_date_str + " " + work_start_time_str
    work_end_date_time_str = work_end_date_str + " " + work_end_time_str
    print(f"Start Date Time: {work_start_date_time_str} End Date Time:
{work_end_date_time_str}")
    if (len(work_start_date_str.split("/"))[-1]) == 2):
        work_start_datetime_obj = datetime.datetime.strptime(work_start_date_time_str,
'%m/%d/%y %l:%M%p')
        work_end_datetime_obj = datetime.datetime.strptime(work_end_date_time_str,
'%m/%d/%y %l:%M%p')

```

```

        elif (len(work_start_date_str.split("/"))[-1]) == 4):

            work_start_datetime_obj = datetime.datetime.strptime(work_start_date_time_str,
'%m/%d/%Y %l:%M%p')

            work_end_datetime_obj = datetime.datetime.strptime(work_end_date_time_str,
'%m/%d/%Y %l:%M%p')

            work_time = work_end_datetime_obj - work_start_datetime_obj

            work_time_seconds = work_time.total_seconds()

            total_work_time_seconds += work_time_seconds

            print(f"Work Time Minutes: {work_time_seconds/60}")

            print(f"Total Work Time Minutes: {total_work_time_seconds/60} \n")

        else:

            print(f"Ignoring Line #{index+1}: {line}\n")

    print(f"Total Work Time: {int(total_work_time_seconds)} Seconds")

    total_work_time = datetime.timedelta(seconds=total_work_time_seconds)

    print("Total Work Time: ", total_work_time)

    total_work_time_mins, total_work_time_secs = divmod(total_work_time_seconds, 60)

    total_work_time_hours, total_work_time_mins = divmod(total_work_time_mins, 60)

    return (total_work_time_hours, total_work_time_mins)

if __name__ == '__main__':

    # Get command line arguments

    parser = argparse.ArgumentParser()

    parser.add_argument("time_log_file", help="Time Log File")

    args = parser.parse_args()

    time_log_file = args.time_log_file

    total_work_time_hours, total_work_time_mins = tlparsed(time_log_file)

    print(f"Total Work Time: {int(total_work_time_hours)} Hours and
{int(total_work_time_mins)} Minutes")

```

index.html:

```
<!DOCTYPE html>
<html>
  <head>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
  </head>
  <body>
    <div id="div1" >
      <center><h2>Comparative Programing Languages</h2></center>
    </div>
    <div id="div2" >
      <center><h2>Kalyan Veluri</h2>
      <h2>2820437</h2></center>
    </div>
    <div id="div3" >
      <center><h1>Time Log Parser</h1></center>
    </div>
    <div id="div4">
      <center>
        <form action = "" method = "POST"
          enctype = "multipart/form-data">
          <input type = "file" name = "file" accept=".txt" class = "inp" id = "inp"/>
          <br></br>
        </br></br>
          <input type = "submit"/>
        </form>
      </center>
    </div>
    <div id='div5'><center>
      <h3>{{output}}</h3></br>{{tot}}</center>
    </div>
    <footer id='div6'></footer>
  </body>
</html>
```

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<h1><b>Result:</b></h1>
{{filen}}
</br>
</br>
```

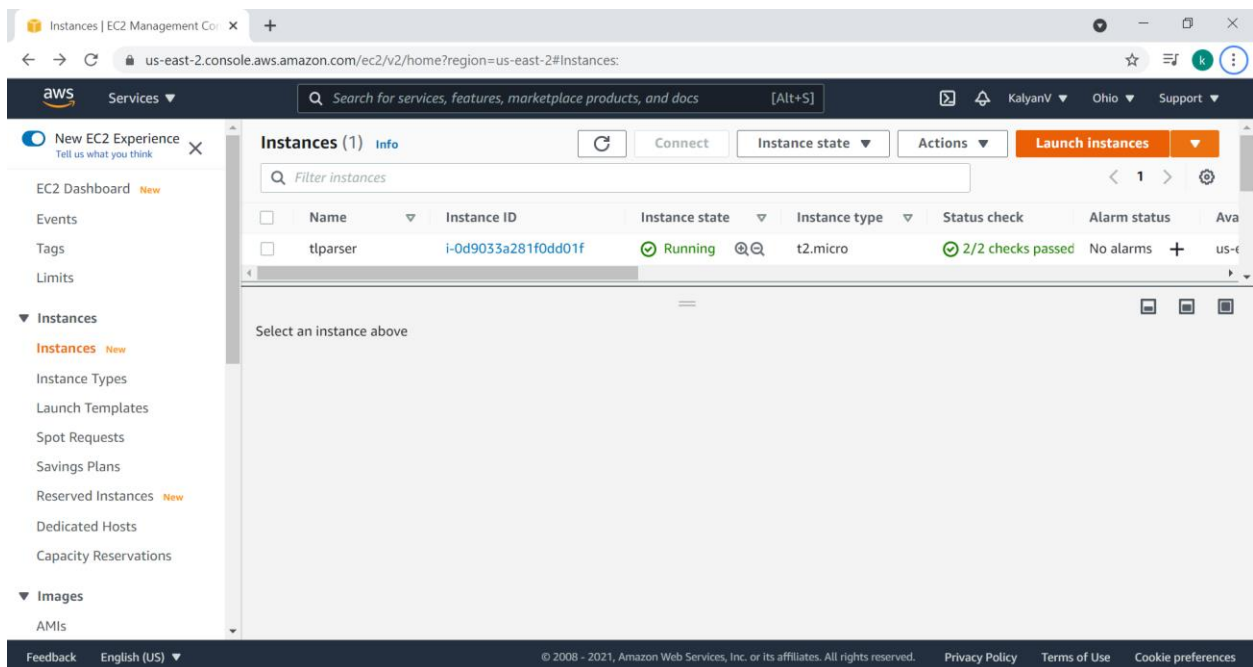
</br>
</br>
{{total}}

Implementation of Cloud Webpage using AWS:

An **EC2 instance** is a virtual server in Amazon's Elastic Compute Cloud (**EC2**) for running applications on the Amazon Web Services (**AWS**) infrastructure. ... Users can select an AMI provided by **AWS**, the user community, or through the **AWS Marketplace**. Users can also create their own AMIs and share them.

Created an Instance using EC2 instance in AWS,

Launch instances using Launch instances on right top of screen.



Step 1: Choose an Amazon Machine Image (AMI)

Select **Ubuntu Server 16.04 LTS (HVM), SSD Volume Type**

Launch instance wizard | EC2 Ma x +

us-east-2.console.aws.amazon.com/ec2/v2/home?region=us-east-2#LaunchInstanceWizard:

aws Services Search for services, features, marketplace products, and docs [Alt+S] KalyanV Ohio Support

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 1: Choose an Amazon Machine Image (AMI)

Cancel and Exit

64-bit (x86)

Built with NVIDIA CUDA, cuDNN, NCCL, GPU Drivers, Intel MKL-DNN, Docker, NVIDIA-Docker and EFA support. For a fully managed experience, check: <https://aws.amazon.com/sagemaker>

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Ubuntu Server 16.04 LTS (HVM), SSD Volume Type - ami-0d563aeddd4be7fff (64-bit x86) / ami-0bff25b43a4479334 (64-bit Arm) **Select**

Free tier eligible

Ubuntu Server 16.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Deep Learning Base AMI (Amazon Linux 2) Version 41.0 - ami-0ffd676bf2e1668c8 **Select**

Built with NVIDIA CUDA, cuDNN, NCCL, GPU Drivers, Intel MKL-DNN, Docker, NVIDIA-Docker and EFA support. For a fully managed experience, check: <https://aws.amazon.com/sagemaker>

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Deep Learning AMI (Amazon Linux) Version 47.0 - ami-09767d59cbb98388 **Select**

Feedback English (US) © 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use Cookie preferences

Step 2: Choose an Instance Type

Launch instance wizard | EC2 Ma x +

us-east-2.console.aws.amazon.com/ec2/v2/home?region=us-east-2#LaunchInstanceWizard:

aws Services Search for services, features, marketplace products, and docs [Alt+S] KalyanV Ohio Support

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they can meet your computing needs.

Filter by: All instance families Current generation Show/Hide Columns

Currently selected: t2.micro (- ECUs, 1 vCPUs, 2.5 GHz, -, 1 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
<input type="checkbox"/>	t2	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
<input checked="" type="checkbox"/>	t2	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	t2	t2.small	1	2	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	t2	t2.medium	2	4	EBS only	-	Low to Moderate	Yes

Cancel Previous **Review and Launch** Next: Configure Instance Details

Feedback English (US) © 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use Cookie preferences

Step 3: Configure Instance Details

Launch instance wizard | EC2 Ma x +

us-east-2.console.aws.amazon.com/ec2/v2/home?region=us-east-2#LaunchInstanceWizard:

aws Services Search for services, features, marketplace products, and docs [Alt+S] KalyanV Ohio Support

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of instances 1 Launch into Auto Scaling Group

Purchasing option ☐ Request Spot instances

Network vpc-fe009b95 (default) Create new VPC

Subnet No preference (default subnet in any Availability Zone) Create new subnet

Auto-assign Public IP Use subnet setting (Enable)

Placement group ☐ Add instance to placement group

Capacity Reservation Open

Domain join directory No directory Create new directory

Cancel Previous Review and Launch Next: Add Storage

Feedback English (US) © 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use Cookie preferences

Step 4: Add Storage

Launch instance wizard | EC2 Ma x +

us-east-2.console.aws.amazon.com/ec2/v2/home?region=us-east-2#LaunchInstanceWizard:

aws Services Search for services, features, marketplace products, and docs [Alt+S] KalyanV Ohio Support

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. [Learn more](#) about storage options in Amazon EC2.

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encryption
Root	/dev/sda1	snap-01075a9bb921226f2	8	General Purpose SSD (gp2)	100 / 3000	N/A	<input checked="" type="checkbox"/>	Not Encrypt

Add New Volume

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. [Learn more](#) about free usage tier eligibility and usage restrictions.

Cancel Previous Review and Launch Next: Add Tags

Feedback English (US) © 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use Cookie preferences

Step 5: Add Tags

Create a key and download it.

Covert the downloaded .pem file to .ppk using Puttygen

Launch instance wizard | EC2 Ma x +

us-east-2.console.aws.amazon.com/ec2/v2/home?region=us-east-2#LaunchInstanceWizard:

aws Services Search for services, features, marketplace products, and docs [Alt+S] KalyanV Ohio Support

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. 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 ⁱ	Volumes ⁱ	Network Interfaces ⁱ
CSU		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Add another tag (Up to 50 tags maximum)

Cancel Previous **Review and Launch** Next: Configure Security Group

Feedback English (US) © 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use Cookie preferences

Step 6: Configure Security Group

Launch instance wizard | EC2 Ma x +

us-east-2.console.aws.amazon.com/ec2/v2/home?region=us-east-2#LaunchInstanceWizard:

aws Services Search for services, features, marketplace products, and docs [Alt+S] KalyanV Ohio Support

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. 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: launch-wizard-3
Description: launch-wizard-3 created 2021-07-04T14:54:56.983-04:00

Type ⁱ	Protocol ⁱ	Port Range ⁱ	Source ⁱ	Description ⁱ
SSH	TCP	22	Custom 0.0.0.0/0	e.g. SSH for Admin Desktop
HTTPS	TCP	443	Custom 0.0.0.0/0, ::0	e.g. SSH for Admin Desktop
HTTP	TCP	80	Custom 0.0.0.0/0, ::0	e.g. SSH for Admin Desktop
Custom TCP f	TCP	5000	Custom 0.0.0.0/0	e.g. SSH for Admin Desktop

Cancel Previous **Review and Launch**

Feedback English (US) © 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use Cookie preferences

Step 7: Review Instance Launch

The screenshot shows the 'Launch Instance Wizard' in the AWS Management Console, specifically Step 7: Review Instance Launch. The wizard is for launching a t2.micro instance in the us-east-2 region. The instance configuration is as follows:

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
t2.micro	-	1	1	EBS only	-	Low to Moderate

Below the configuration table, the 'Security Groups' section shows a group named 'launch-wizard-3' with the description 'launch-wizard-3 created 2021-07-04T14:54:56.983-04:00'. The group's rules are listed in the following table:

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	0.0.0.0/0	
HTTPS	TCP	443	0.0.0.0/0	
HTTPS	TCP	443	:::0	
HTTP	TCP	80	0.0.0.0/0	
HTTP	TCP	80	:::0	

At the bottom of the wizard, there are buttons for 'Cancel', 'Previous', and 'Launch'. The footer of the console shows the date '© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.' and links to 'Privacy Policy', 'Terms of Use', and 'Cookie preferences'.

The screenshot shows the 'Instances' page in the AWS Management Console. A single instance named 't1parser' with ID 'i-0d9033a281f0dd01f' is shown in a 'Running' state. The instance is a t2.micro type. The 'Details' tab is selected, showing the following information:

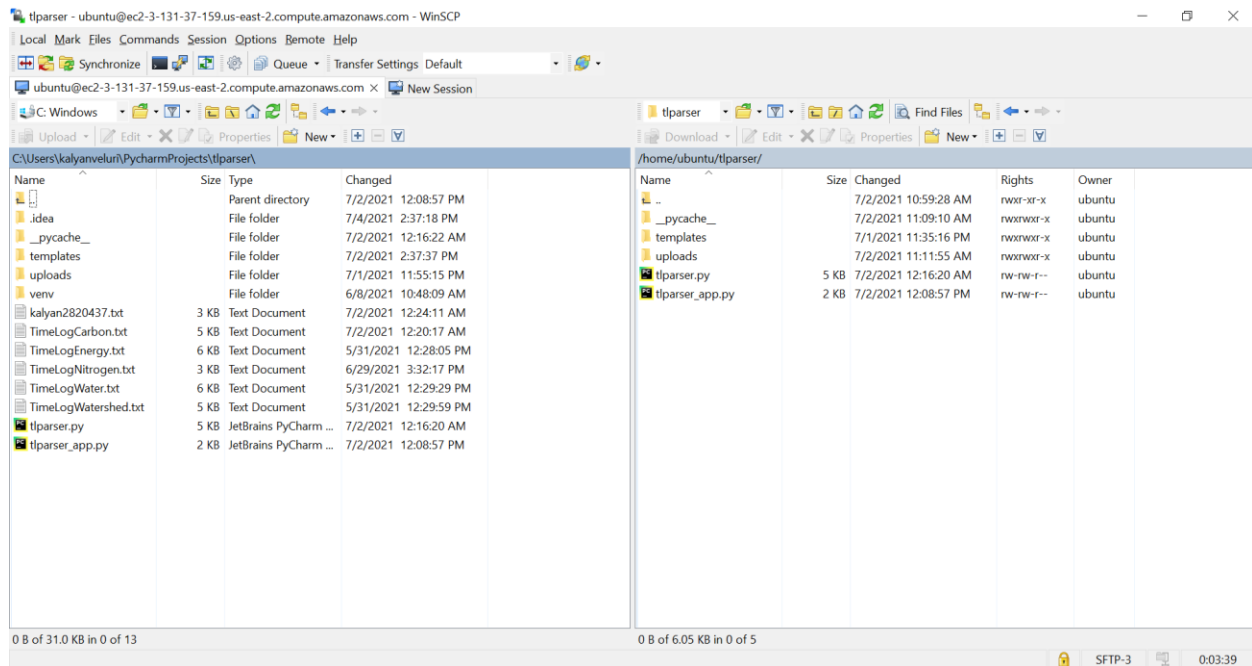
Instance ID	Public IPv4 address	Private IPv4 addresses
i-0d9033a281f0dd01f (t1parser)	3.131.37.159 open address	172.31.1.104

Other details include:

- Instance state: Running
- Public IPv4 DNS: ec2-3-131-37-159.us-east-2.compute.amazonaws.com | [open address](#)
- Private IPv4 DNS: ip-172-31-1-104.us-east-2.compute.internal
- Elastic IP addresses: -
- VPC ID: vpc-fa00b95

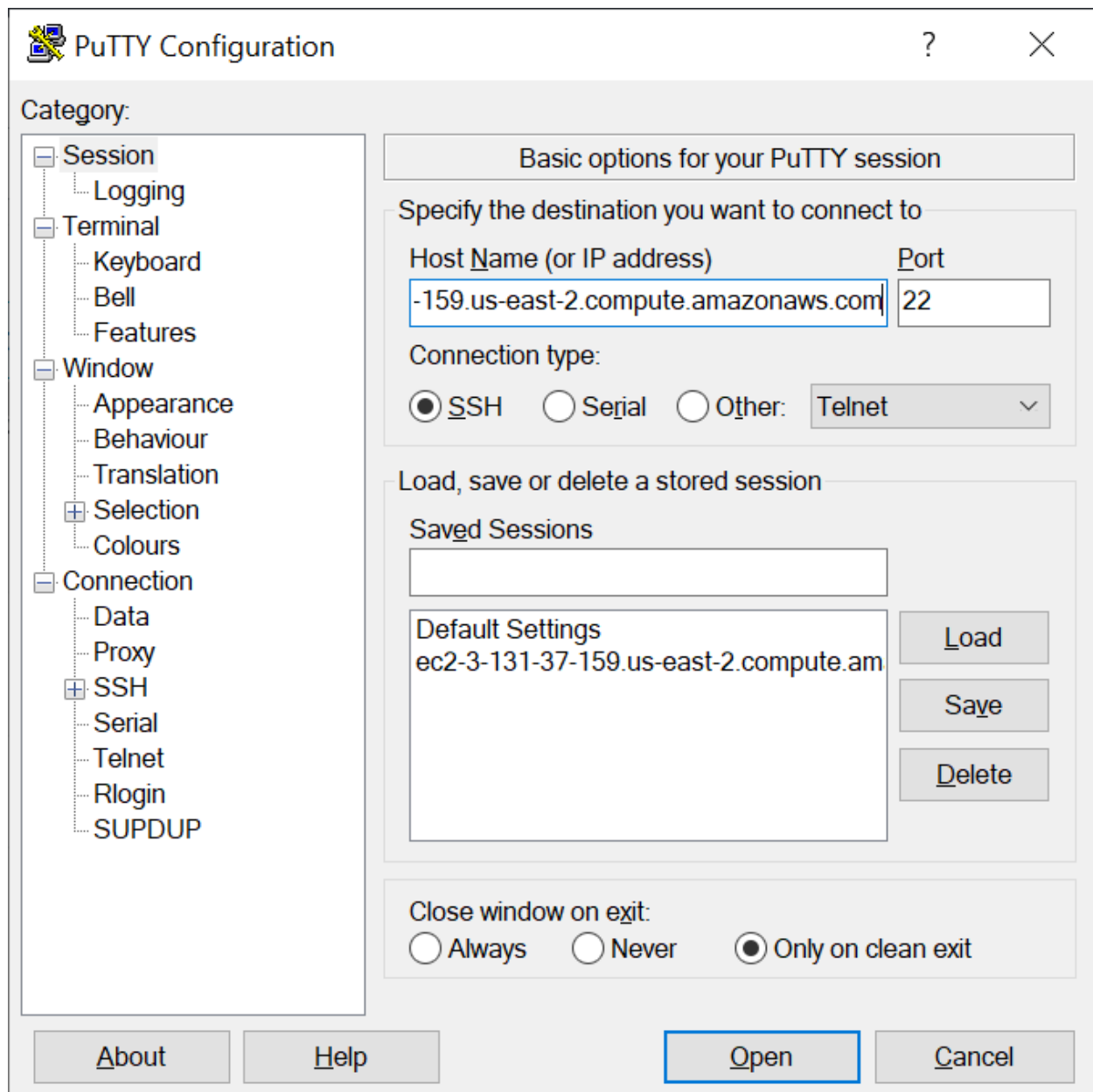
The footer of the console shows the date '© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.' and links to 'Privacy Policy', 'Terms of Use', and 'Cookie preferences'.

Using WinSCP connect to instance ID and copy the files to the directory.



Putty:

Use putty and the add the Public IPv4 DNS which show for your instance created in AWS in the Host Name (or IP address).



Open the Session and create a directory. In my case I created **"kven"**

Use the following commands and install Python3, Virtualenv, flask etc;

```
sudo apt update
```

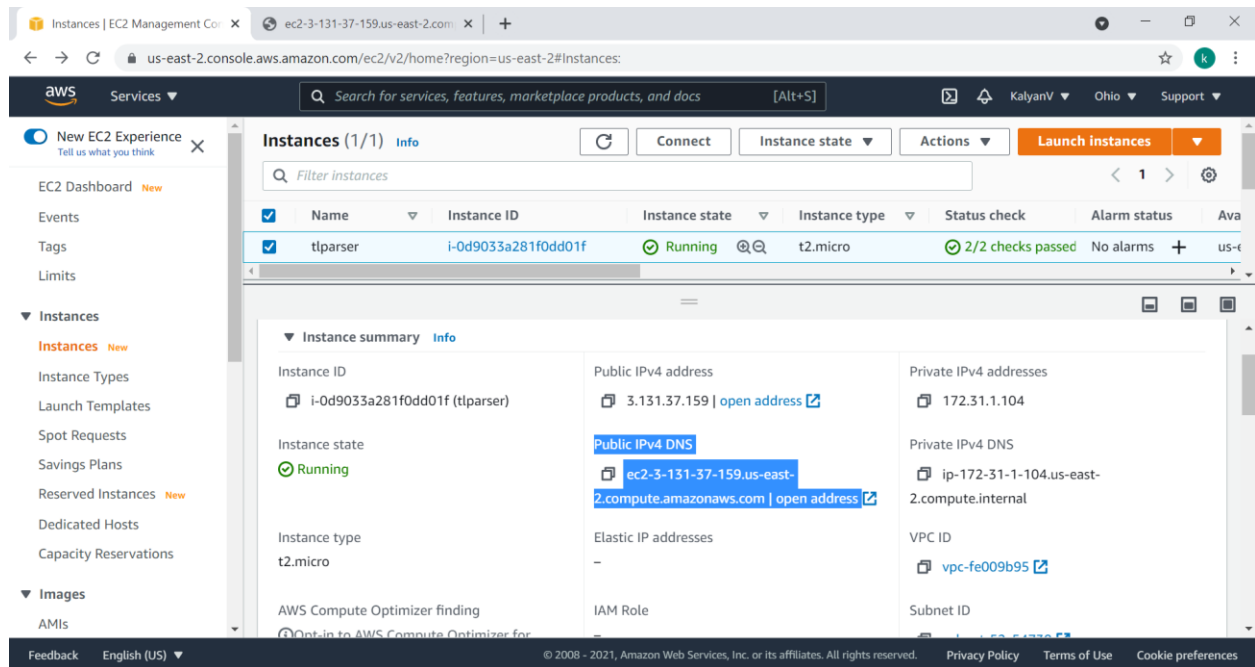
```
sudo apt install python3-pip
```

```
sudo apt install virtualenv
```

```
pip install flask
```

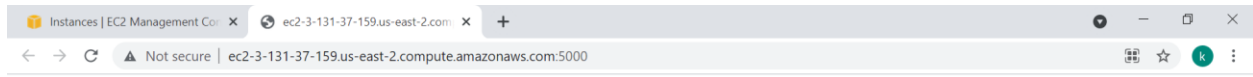
Open the directory where you stored the python files and Run the python code using python filename.py (I used python tlparger_app.py)

Open the instance which created in AWS and open the public IPv4 DNS.



Here is my instance address (<http://ec2-3-131-37-159.us-east-2.compute.amazonaws.com:5000/>)

Result:



Comparative Programing Languages

Kalyan Veluri

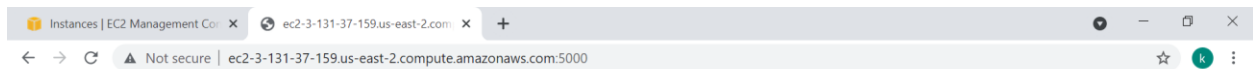
2820437

Time Log Parser

Choose File No file chosen

Submit

Result:



Comparative Programing Languages

Kalyan Veluri

2820437

Time Log Parser

Choose File No file chosen

Submit

Result:

kalyan2820437.txt

Total Work Time: 110 Hours and 50 Minutes

Reference:

- <https://www.geeksforgeeks.org/python-program-convert-time-12-hour-24-hour-format/>
- <https://www.tutorialspoint.com/adding-time-in-python>

- <https://flask.palletsprojects.com/en/2.0.x/patterns/fileuploads/>
- <https://towardsdatascience.com/creating-a-website-to-host-your-python-web-application-f06f694a87e8>

My Webpage Link:

<http://ec2-3-131-37-159.us-east-2.compute.amazonaws.com:5000/>