Python and Script Learning

Python for Automation

Subprocess Module

The subprocess module in Python allows you to spawn new processes, connect to their input/output/error pipes, and obtain their return codes. This is essential for automation tasks that need to interact with system commands.

File: building_pipeline_without_shell.py

```python
#!/usr/bin/python3

import subprocess

#ps aux | grep python | wc -l

p1 = subprocess.Popen(["ps", "aux"], stdout=subprocess.PIPE)
p2 = subprocess.Popen(["grep", "python"] ,stdin=p1.stdout, stdout=subprocess.PIPE)
p3 = subprocess.Popen(["wc", "-l"], stdin=p2.stdout , stdout=subprocess.PIPE)

p1.stdout.close()
p2.stdout.close()

out, _ = p3.communicate()
print(int(out.decode().strip()))

```

File: error_handling.py

```python
#!/usr/bin/python3
import subprocess

try:
    result=subprocess.run(["ls", "wrong_argument"], check=True , capture_output=True , text=True)
    print(result.stdout)
except subprocess.CalledProcessError as e:
    print("Command failed")
    print("Return code : ", e.returncode)
    print("STDOUT : ", e.stdout)
    print("STDERR : ", e.stderr)
except FileNotFoundError as e:
    print("Executable not found : ", e)


```

Boto3 for AWS Integration

Boto3 is the Amazon Web Services (AWS) SDK for Python, enabling Python developers to write software that makes use of services like Amazon S3, Amazon EC2, and many others.

Real-world Automation

Log Parser Implementation

The log parser script demonstrates practical automation for system monitoring and maintenance tasks. It incorporates several key automation concepts:

Features Implemented:

- Automated log file processing

- Pattern matching and filtering
- Scheduled execution using cron jobs
- Error handling and logging

Cron Job Integration: Cron jobs enable automatic execution of the log parser at specified intervals, ensuring continuous monitoring without manual intervention.

File: log_parser.sh

```
#!/usr/bin/bash

#creating a csv file , to save the logs
echo "Date,Message" >> parsed_logs.csv

#getting last 10 logs of the /var/log/syslog file , (which is a system log file)
tail -10 /var/log/syslog | while read line; do
    date_section=$(echo "$line" | cut -d ' ' -f 1-3)
    message_section=$(echo "$line" | cut -d ' ' -f 4-)

    echo "$date_section,$message_section" >> parsed_logs.csv

done
```

```
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command

0 15 * * * /home/prashantm/prashant_m/log_backup.sh
```

Backup Script Implementation

The backup script showcases comprehensive file management automation with the following capabilities:

Core Functionality:

- Automated file archiving
- Compression and storage optimization

Shell Scripting Features:

- Loop constructs for batch processing
- Function definitions for reusable code
- Variable management and parameter handling

File: log_backup.sh

```bash
#!/usr/bin/bash

#running log_parser.sh to collect the system logs
./log_parser.sh

#making the backup with todays date
current_date=$(date +%y-%m-%d)

#creating a tarball of each day
tar -cvf backup/backup_$current_date.tar.gz parsed_logs.csv

echo "backup is stored as backup_$current_date.tar.gz"

```

File: parsed_logs.csv

```
Date,Message
Aug 29 12:21:43,wrl698prashantm systemd[1]: quickupdate.service: Scheduled restart job, restart counter is at 304.
Aug 29 12:21:43,wrl698prashantm systemd[1]: Stopped Seqrite Update Service..
Aug 29 12:21:43,wrl698prashantm systemd[1]: Starting Seqrite Update Service....
Aug 29 12:21:43,wrl698prashantm systemd[1]: Started Seqrite Update Service..
Aug 29 12:21:43,wrl698prashantm brltty[3433]: /lib/brltty/libbrlttyxa2.so: cannot open shared object file: No such file or directory
Aug 29 12:21:43,wrl698prashantm brltty[3433]: cannot load screen driver: /lib/brltty/libbrlttyxa2.so
Aug 29 12:21:43,wrl698prashantm brltty[3433]: screen driver not loadable: a2
Aug 29 12:21:48,wrl698prashantm brltty[3433]: /lib/brltty/libbrlttyxa2.so: cannot open shared object file: No such file or directory
Aug 29 12:21:48,wrl698prashantm brltty[3433]: cannot load screen driver: /lib/brltty/libbrlttyxa2.so
Aug 29 12:21:48,wrl698prashantm brltty[3433]: screen driver not loadable: a2
Date,Message
Aug 29 12:35:48,wrl698prashantm google-chrome.desktop[4322]: [4367:4373:0829/123548.916912:ERROR:services/network/restricted_cookie_manager.cc:1156]
top_frame_origin from renderer='null [internally: (39D7230B0C7C597A25D18F0D74150623) anonymous]' from browser='https://www.perplexity.ai';
Aug 29 12:35:53,wrl698prashantm brltty[3433]: /lib/brltty/libbrlttyxa2.so: cannot open shared object file: No such file or directory
Aug 29 12:35:53,wrl698prashantm brltty[3433]: cannot load screen driver: /lib/brltty/libbrlttyxa2.so
Aug 29 12:35:53,wrl698prashantm brltty[3433]: screen driver not loadable: a2
Aug 29 12:35:54,wrl698prashantm systemd[1]: quickupdate.service: Main process exited, code=exited, status=255/EXCEPTION
Aug 29 12:35:54,wrl698prashantm systemd[1]: quickupdate.service: Failed with result 'exit-code'.
Aug 29 12:35:54,wrl698prashantm systemd[1]: quickupdate.service: Scheduled restart job, restart counter is at 387.
Aug 29 12:35:54,wrl698prashantm systemd[1]: Stopped Seqrite Update Service..
Aug 29 12:35:54,wrl698prashantm systemd[1]: Starting Seqrite Update Service....
Aug 29 12:35:54,wrl698prashantm systemd[1]: Started Seqrite Update Service..
Date,Message
Aug 29 12:36:38,wrl698prashantm brltty[3433]: screen driver not loadable: a2
Aug 29 12:36:43,wrl698prashantm brltty[3433]: /lib/brltty/libbrlttyxa2.so: cannot open shared object file: No such file or directory
Aug 29 12:36:43,wrl698prashantm brltty[3433]: cannot load screen driver: /lib/brltty/libbrlttyxa2.so
Aug 29 12:36:43,wrl698prashantm brltty[3433]: screen driver not loadable: a2
Aug 29 12:36:45,wrl698prashantm systemd[1]: quickupdate.service: Main process exited, code=exited, status=255/EXCEPTION
Aug 29 12:36:45,wrl698prashantm systemd[1]: quickupdate.service: Failed with result 'exit-code'.
Aug 29 12:36:45,wrl698prashantm systemd[1]: quickupdate.service: Scheduled restart job, restart counter is at 392.
Aug 29 12:36:45,wrl698prashantm systemd[1]: Stopped Seqrite Update Service..
Aug 29 12:36:45,wrl698prashantm systemd[1]: Starting Seqrite Update Service....
Aug 29 12:36:45,wrl698prashantm systemd[1]: Started Seqrite Update Service..
```

Project Deliverables

1. Python Script: EC2 Lifecycle Management via Boto3

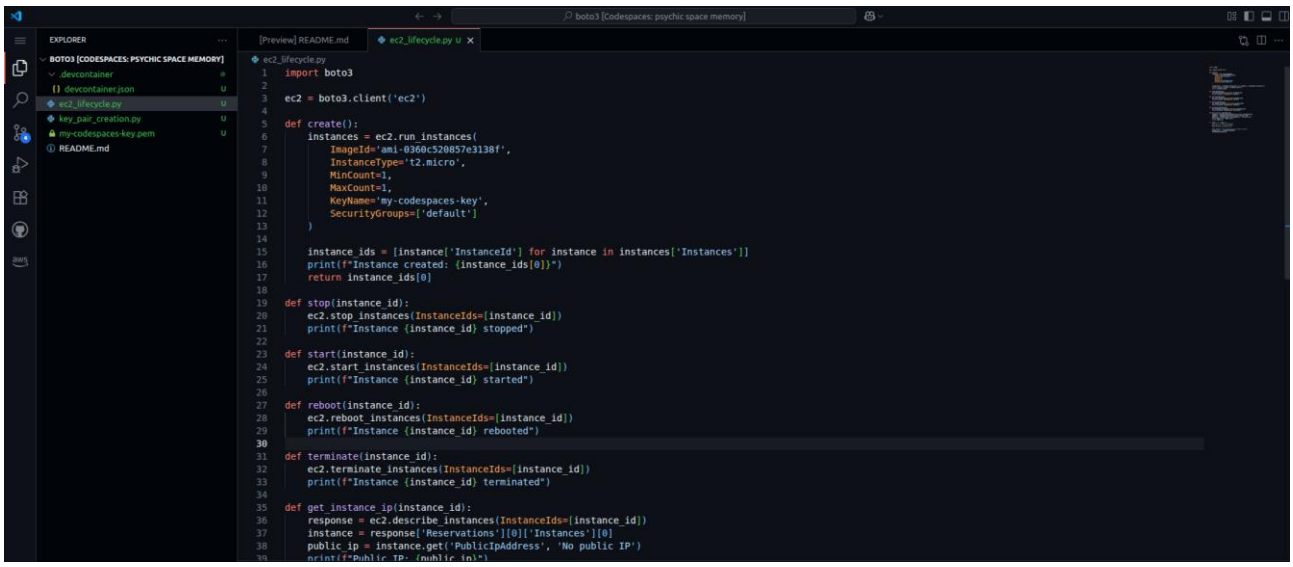
This Python script demonstrates essential EC2 instance management operations using boto3, providing fundamental control over AWS EC2 instances.

Script Functionality:

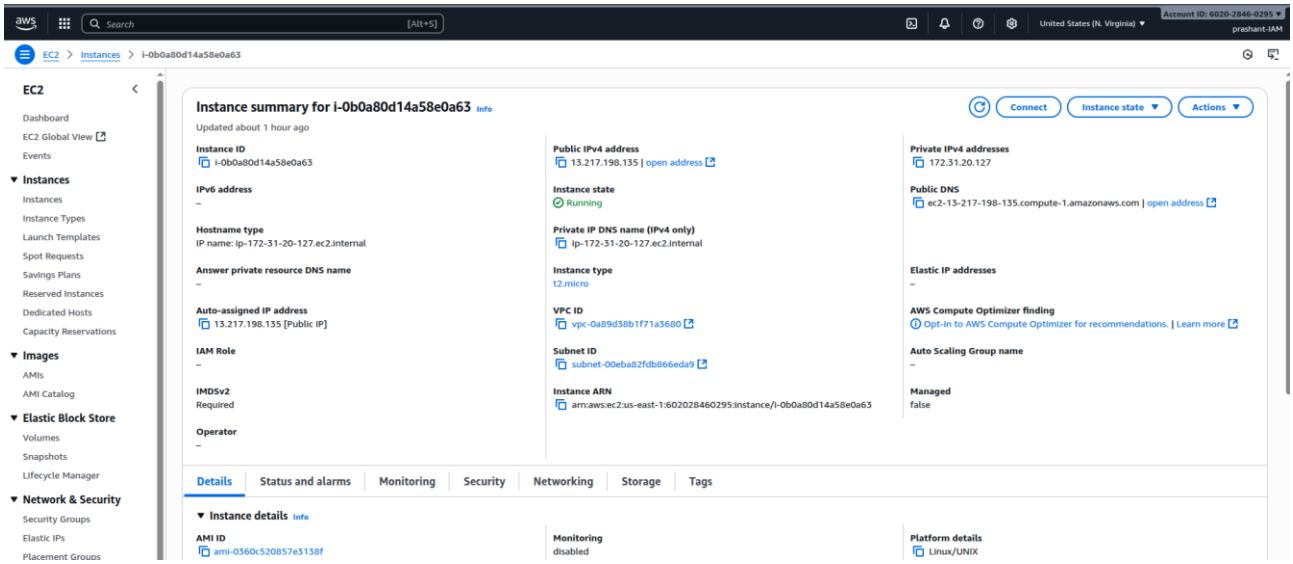
- **Create**: Launch new EC2 instances with specified configurations
- **Start**: Start stopped EC2 instances
- **Stop**: Stop running EC2 instances
- **Reboot**: Restart EC2 instances
- **Terminate**: Permanently delete EC2 instances
- **Get Instance IP**: Retrieve public/private IP addresses of instances

Key Implementation Details:

- AWS credential configuration and authentication
- Instance ID management and validation
- Instance state verification before operations



```python
import boto3

ec2 = boto3.client('ec2')

def create():
    instances = ec2.run_instances(
        ImageId='ami-0360c520857e3138f',
        InstanceType='t2.micro',
        MinCount=1,
        MaxCount=1,
        KeyName='my-codespaces-key',
        SecurityGroups=['default']
    )

    instance_ids = [instance['InstanceId'] for instance in instances['Instances']]
    print(f"Instance created: {instance_ids[0]}")
    return instance_ids[0]

def stop(instance_id):
    ec2.stop_instances(InstanceIds=[instance_id])
    print(f"Instance {instance_id} stopped")

def start(instance_id):
    ec2.start_instances(InstanceIds=[instance_id])
    print(f"Instance {instance_id} started")

def reboot(instance_id):
    ec2.reboot_instances(InstanceIds=[instance_id])
    print(f"Instance {instance_id} rebooted")

def terminate(instance_id):
    ec2.terminate_instances(InstanceIds=[instance_id])
    print(f"Instance {instance_id} terminated")

def get_instance_ip(instance_id):
    response = ec2.describe_instances(InstanceIds=[instance_id])
    instance = response['Reservations'][0]['Instances'][0]
    public_ip = instance.get('PublicIpAddress', 'No public IP')
```



---

2. Shell Script: Log Archiving with Email Alerts

This shell script provides automated log management with notification capabilities, demonstrating practical system administration automation.

Script Features:

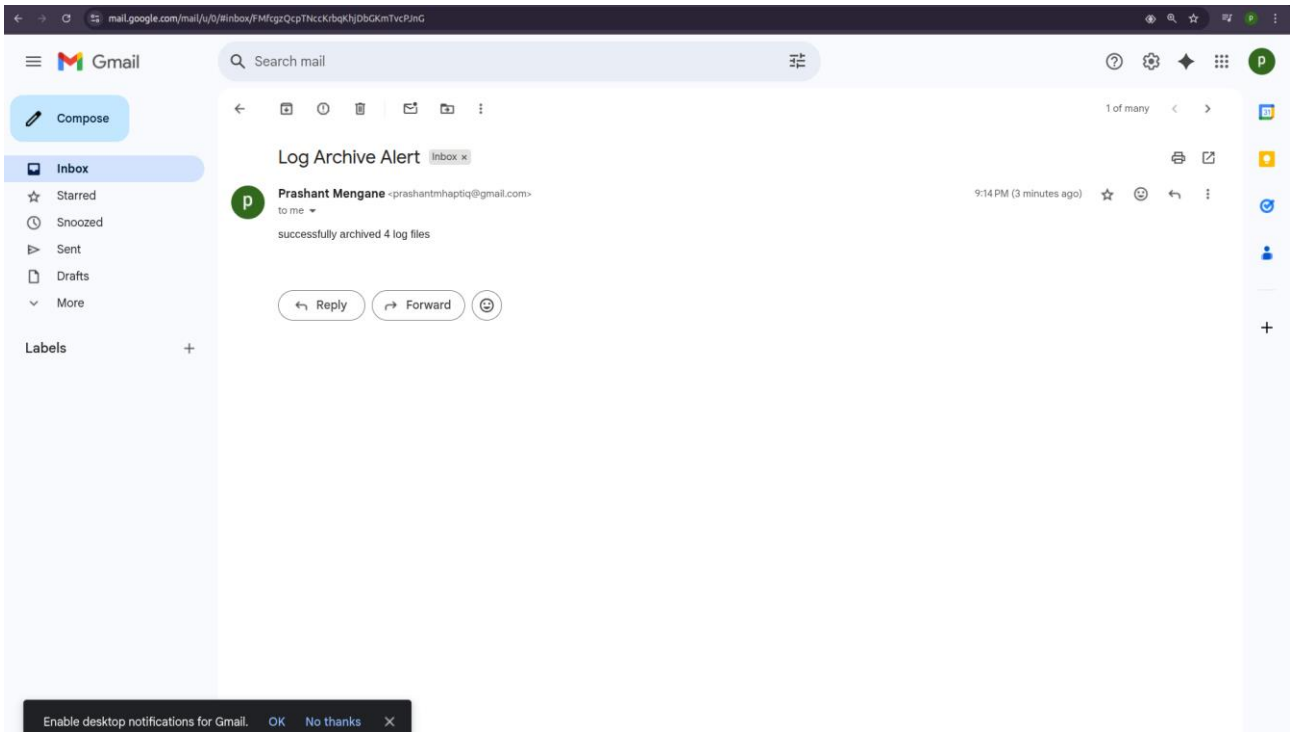
- Automated log file identification and archiving
- Compression for storage efficiency
- Email notification system
- Error handling and logging

Technical Implementation:

- Conditional logic for file processing
- Loop structures for batch operations
- Email integration using system mail tools

File: archive_log.sh

```bash
#!/usr/bin/bash

LOG_DIR="/var/log"
ARCHIVE_DIR="/tmp/archived_logs"

#create the archive_dir is not located
mkdir -p $ARCHIVE_DIR


#this is the email on which we will send alerts
EMAIL="prashantmhaptiq@gmail.com"


#the logs which we want to archive
all_logs="syslog dpkg.log mail.log auth.log"
file_count=0

#looping over the all_logs
for log in $all_logs; do
  if [ -f "$LOG_DIR/$log" ]; then
    cp $LOG_DIR/$log $ARCHIVE_DIR/
    file_count=$((file_count+1))
  fi
done


#compressing the logs
if [ $file_count -gt 0 ]; then
  gzip $ARCHIVE_DIR/*

  #sending the email alert
  echo "successfully archived $file_count log files" | mail -s "Log Archive Alert" $EMAIL
else
  echo "Didn't found and log files" | mail -s "No Logs Found" $EMAIL

fi
```




```
/var/log🔒
➜cd /tmp/archived_logs/
auth.log.gz  dpkg.log.gz  mail.log.gz  syslog.gz

/tmp/archived_logs
➜
```

Testing and Validation

Test Cases Covered:

1. **EC2 Management Script:**
    - Instance creation with different AMI types
    - Starting and stopping instances in various states
    - Rebooting running instances
    - Terminating instances safely
    - Retrieving IP addresses for active instances
    - Error handling for invalid instance IDs
    - AWS authentication and permission validation
2. **Log Archiving Script:**
    - Large file handling
    - Disk space validation
    - Email delivery confirmation
    - Archive integrity verification