Introduction

This document provides a detailed explanation of the automation architecture implemented for our capstone project. The system was designed to ensure the web scraping pipelines run automatically without requiring constant manual intervention. All automation is currently hosted on an AWS EC2 instance, which was created under my personal AWS Free Tier account.

To enable continuity beyond my account and ensure that future students can continue this work, I have created a public Amazon Machine Image (AMI) of the configured environment. This allows anyone with an AWS account to launch an identical EC2 instance without needing to repeat the initial setup. The following sections outline the steps required to use the AMI, connect to the instance, understand the configuration, and manage the automated workflows.

AMI Access

To avoid loss of work when my Free Tier expires, I created a public AMI from the configured EC2 instance. Future students can use this AMI to replicate the environment.

- **AMI ID:** ami-07a95230fff01681f
- **Region:** ap-southeast-2 (Asia Pacific – Sydney)
- **Direct Console Link:** Launch this AMI

Instructions:

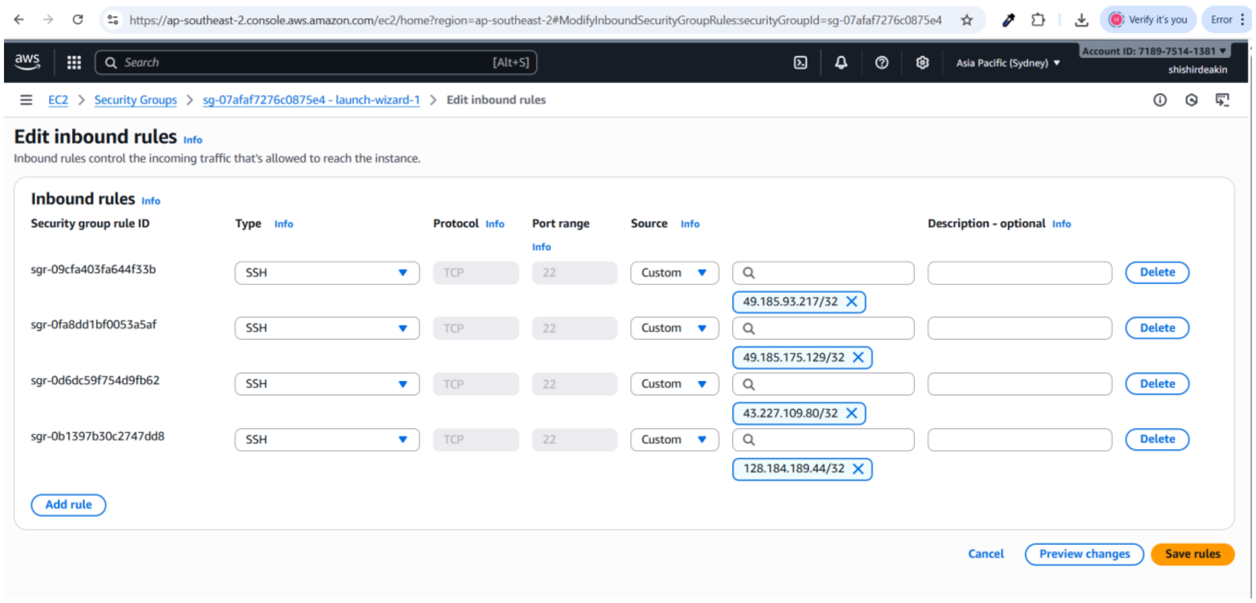
1. Log in to your AWS account.
2. Open the link above (or search for the AMI ID in EC2 → AMIs).
3. Click **Launch Instance**.
4. Choose an instance type (e.g., t2.micro for testing).
5. Configure networking and storage, then launch the instance.

Connecting to the Instance

Once the instance is launched, connect via SSH using the .pem key pair you generated during setup.

1. Allow SSH access to your IP by adjusting security group settings in the AWS console. (Check your current IP at whatismyip.com).
2. Note the **Public IPv4 address** of your running instance.
3. Open PowerShell on your local machine and navigate to the directory where your .pem file is stored.
4. Connect using the following command:
5. ssh -i your-key.pem ubuntu@<public-ip-address>
6. When prompted with *“Are you sure you want to continue connecting?”*, type yes and press Enter.

You will now be connected to the EC2 instance.



https://ap-southeast-2.console.aws.amazon.com/ec2/home?region=ap-southeast-2#ModifyInboundSecurityGroupRules:securityGroupId=sg-07afaf7276c0875e4

Account ID: 7189-7514-1381 · shishirdeakin · Asia Pacific (Sydney)

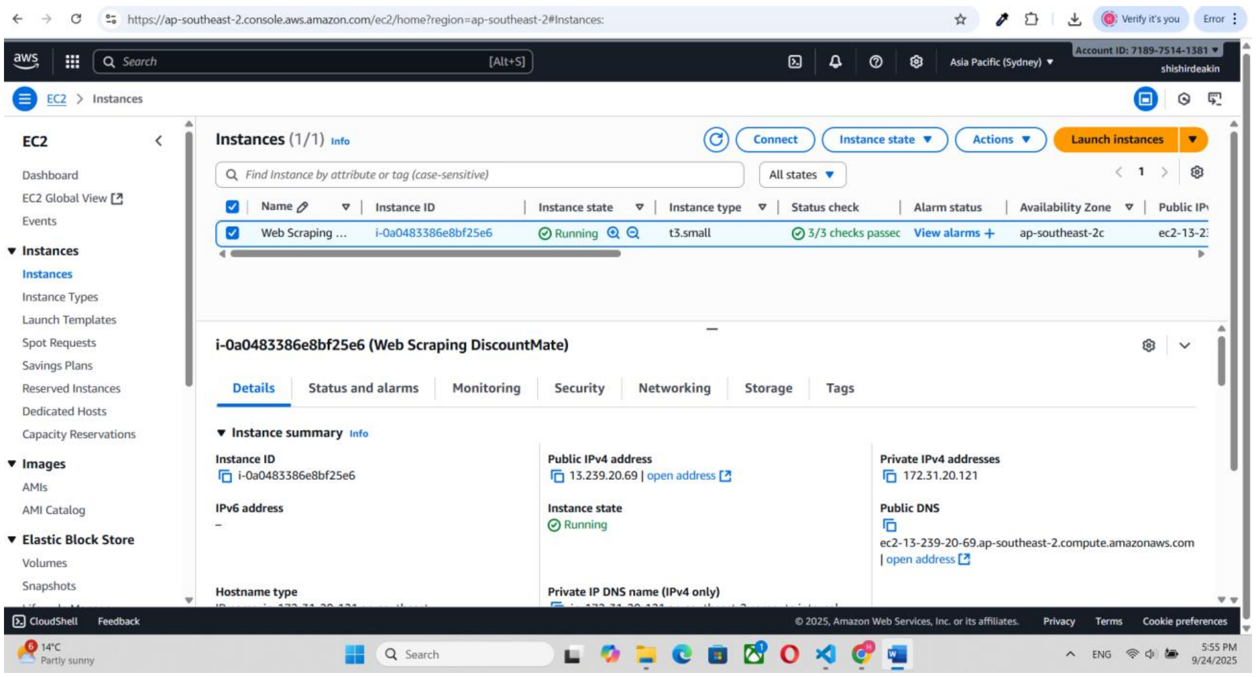
EC2 > Security Groups > sg-07afaf7276c0875e4 - launch-wizard-1 > Edit inbound rules

Edit inbound rules

Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules

Security group rule ID	Type	Protocol	Port range	Source		Description - optional
sgr-09cfa403fa644f33b	SSH	TCP	22	Custom	49.185.93.217/32	
sgr-0fa8dd1bf0053a5af	SSH	TCP	22	Custom	49.185.175.129/32	
sgr-0d6dc59f754d9fb62	SSH	TCP	22	Custom	43.227.109.80/32	
sgr-0b1397b30c2747dd8	SSH	TCP	22	Custom	128.184.189.44/32	



https://ap-southeast-2.console.aws.amazon.com/ec2/home?region=ap-southeast-2#Instances:

Asia Pacific (Sydney) — Account ID: 7189-7514-1381 — shishirdeakin

Instances (1/1)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv
Web Scraping ...	i-0a0483386e8bf25e6	Running	t3.small	3/3 checks passec	View alarms +	ap-southeast-2c	ec2-13-2

i-0a0483386e8bf25e6 (Web Scraping DiscountMate)

Instance summary

Instance ID: i-0a0483386e8bf25e6

Public IPv4 address: 13.239.20.69 | open address

Private IPv4 addresses: 172.31.20.121

IPv6 address: –

Instance state: Running

Public DNS: ec2-13-239-20-69.ap-southeast-2.compute.amazonaws.com | open address

Hostname type

Private IP DNS name (IPv4 only)

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Project Files and Scripts

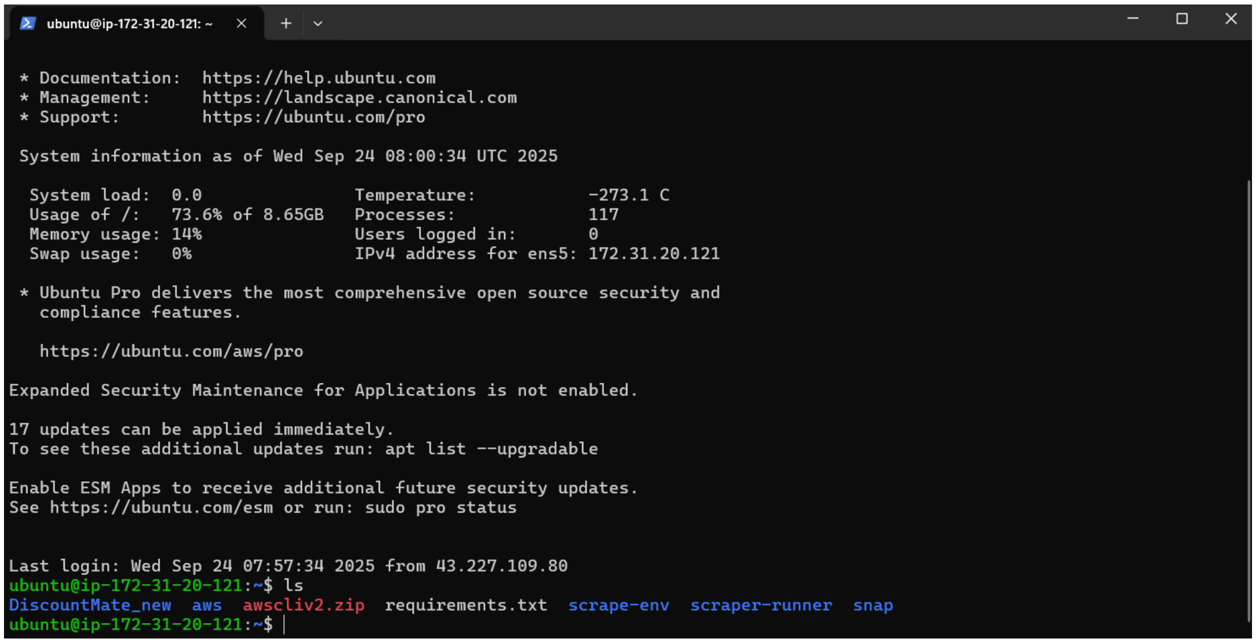
Inside the EC2 instance:

- Running ls will display the available project directories.
- The main scraping logic is stored inside the **scraper-runner** directory.

Within this directory, the script **run_all_scrapers.sh** controls the automation. This script handles:

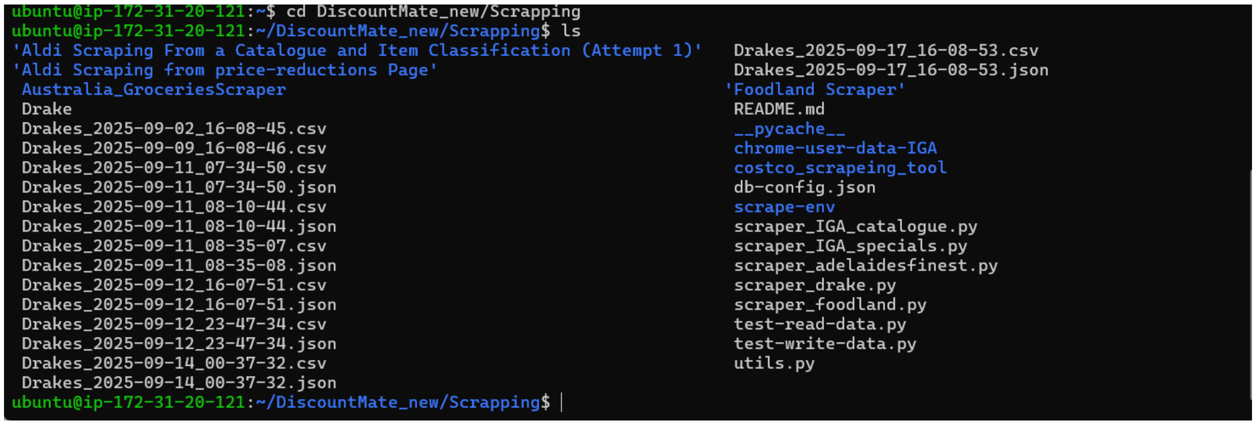
- Logging
- Exporting environment paths
- Activating the Python virtual environment
- Running the scraping jobs
- Automatically shutting down the instance after completion to save costs

If modifications are needed (e.g., adding scrapers, changing paths), open this script with:



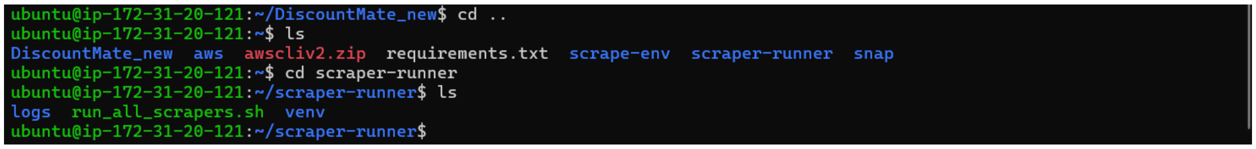
```
 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Wed Sep 24 08:00:34 UTC 2025

  System load:  0.0               Temperature:           -273.1 C
  Usage of /:   73.6% of 8.65GB   Processes:             117
  Memory usage: 14%               Users logged in:       0
  Swap usage:   0%                IPv4 address for ens5: 172.31.20.121

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.

   https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

17 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


Last login: Wed Sep 24 07:57:34 2025 from 43.227.109.80
ubuntu@ip-172-31-20-121:~$ ls
DiscountMate_new  aws  awscliv2.zip  requirements.txt  scrape-env  scraper-runner  snap
ubuntu@ip-172-31-20-121:~$ |
```

Our scraping code lives here which we pulled from remote GitHub repo:



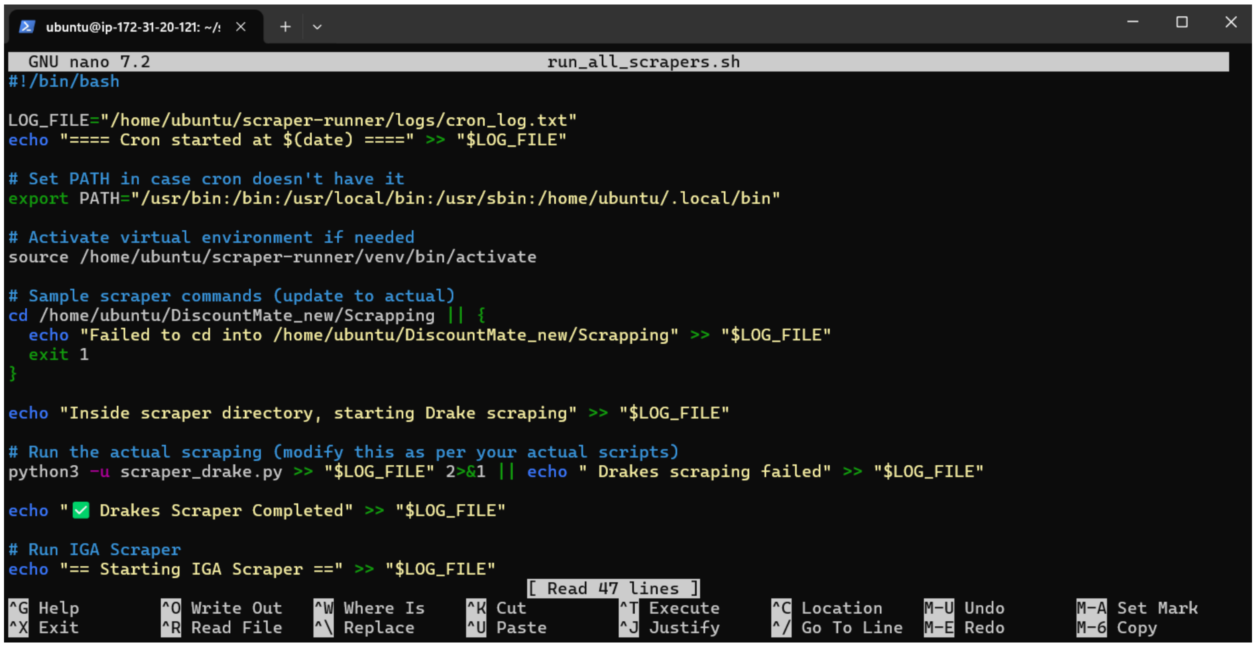
```
ubuntu@ip-172-31-20-121:~$ cd DiscountMate_new/Scrapping
ubuntu@ip-172-31-20-121:~/DiscountMate_new/Scrapping$ ls
'Aldi Scraping From a Catalogue and Item Classification (Attempt 1)'   Drakes_2025-09-17_16-08-53.csv
'Aldi Scraping from price-reductions Page'                             Drakes_2025-09-17_16-08-53.json
 Australia_GroceriesScraper                                           'Foodland Scraper'
 Drake                                                                 README.md
 Drakes_2025-09-02_16-08-45.csv                                        __pycache__
 Drakes_2025-09-09_16-08-46.csv                                        chrome-user-data-IGA
 Drakes_2025-09-11_07-34-50.csv                                        costco_scrapeing_tool
 Drakes_2025-09-11_07-34-50.json                                       db-config.json
 Drakes_2025-09-11_08-10-44.csv                                        scrape-env
 Drakes_2025-09-11_08-10-44.json                                       scraper_IGA_catalogue.py
 Drakes_2025-09-11_08-35-07.csv                                        scraper_IGA_specials.py
 Drakes_2025-09-11_08-35-08.json                                       scraper_adelaidesfinest.py
 Drakes_2025-09-12_16-07-51.csv                                        scraper_drake.py
 Drakes_2025-09-12_16-07-51.json                                       scraper_foodland.py
 Drakes_2025-09-12_23-47-34.csv                                        test-read-data.py
 Drakes_2025-09-12_23-47-34.json                                       test-write-data.py
 Drakes_2025-09-14_00-37-32.csv                                        utils.py
 Drakes_2025-09-14_00-37-32.json
ubuntu@ip-172-31-20-121:~/DiscountMate_new/Scrapping$ |
```

Now, if you go inside scraper-runner and type ls, you will see a shell script file called **run_all_scrapers.sh**



```
ubuntu@ip-172-31-20-121:~/DiscountMate_new$ cd ..
ubuntu@ip-172-31-20-121:~$ ls
DiscountMate_new  aws  awscliv2.zip  requirements.txt  scrape-env  scraper-runner  snap
ubuntu@ip-172-31-20-121:~$ cd scraper-runner
ubuntu@ip-172-31-20-121:~/scraper-runner$ ls
logs  run_all_scrapers.sh  venv
ubuntu@ip-172-31-20-121:~/scraper-runner$
```

Type nano **run_all_scrapers.sh** and it will open the script that we use to run our all scrapers.



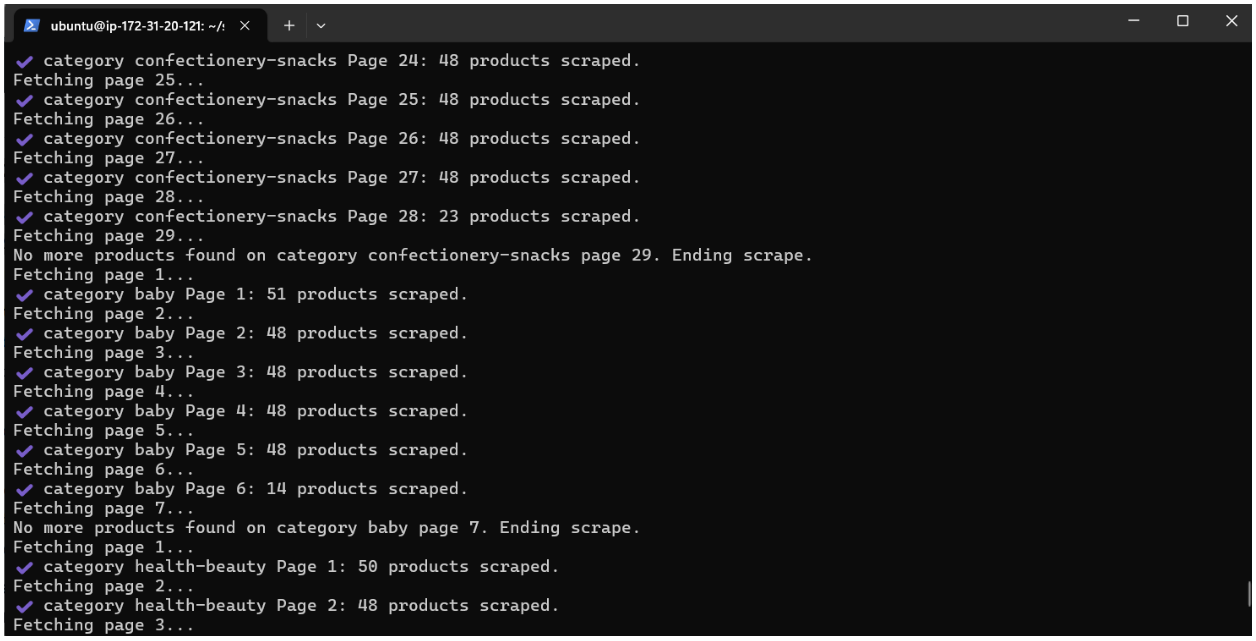
```
ubuntu@ip-172-31-20-121: ~/:
  GNU nano 7.2                              run_all_scrapers.sh
#!/bin/bash

LOG_FILE="/home/ubuntu/scraper-runner/logs/cron_log.txt"
echo "==== Cron started at $(date) ====" >> "$LOG_FILE"

# Set PATH in case cron doesn't have it
export PATH="/usr/bin:/bin:/usr/local/bin:/usr/sbin:/home/ubuntu/.local/bin"

# Activate virtual environment if needed
source /home/ubuntu/scraper-runner/venv/bin/activate

# Sample scraper commands (update to actual)
cd /home/ubuntu/DiscountMate_new/Scrapping || {
  echo "Failed to cd into /home/ubuntu/DiscountMate_new/Scrapping" >> "$LOG_FILE"
  exit 1
}

echo "Inside scraper directory, starting Drake scraping" >> "$LOG_FILE"

# Run the actual scraping (modify this as per your actual scripts)
python3 -u scraper_drake.py >> "$LOG_FILE" 2>&1 || echo " Drakes scraping failed" >> "$LOG_FILE"

echo "✅ Drakes Scraper Completed" >> "$LOG_FILE"

# Run IGA Scraper
echo "== Starting IGA Scraper ==" >> "$LOG_FILE"
                                   [ Read 47 lines ]
^G Help         ^O Write Out    ^W Where Is     ^K Cut          ^T Execute      ^C Location     M-U Undo        M-A Set Mark
^X Exit         ^R Read File    ^\ Replace      ^U Paste        ^J Justify      ^/ Go To Line   M-E Redo        M-6 Copy
```

If you need to add other scrapers or export path or anything, this is the place.

In this file, we configure logs, export path, set up the scraping virtual environment, give path and command to run scraping, and include command at last to auto stop the EC2 instance after the scraping is completed so that we can save costs.

If you go inside logs directory inside scraper_runner, you will see log files.

Our logs are in the cron_log.txt file. Type cat cron_log.txt and you will see the logs.



```
ubuntu@ip-172-31-20-121: ~/:
✔ category confectionery-snacks Page 24: 48 products scraped.
Fetching page 25...
✔ category confectionery-snacks Page 25: 48 products scraped.
Fetching page 26...
✔ category confectionery-snacks Page 26: 48 products scraped.
Fetching page 27...
✔ category confectionery-snacks Page 27: 48 products scraped.
Fetching page 28...
✔ category confectionery-snacks Page 28: 23 products scraped.
Fetching page 29...
No more products found on category confectionery-snacks page 29. Ending scrape.
Fetching page 1...
✔ category baby Page 1: 51 products scraped.
Fetching page 2...
✔ category baby Page 2: 48 products scraped.
Fetching page 3...
✔ category baby Page 3: 48 products scraped.
Fetching page 4...
✔ category baby Page 4: 48 products scraped.
Fetching page 5...
✔ category baby Page 5: 48 products scraped.
Fetching page 6...
✔ category baby Page 6: 14 products scraped.
Fetching page 7...
No more products found on category baby page 7. Ending scrape.
Fetching page 1...
✔ category health-beauty Page 1: 50 products scraped.
Fetching page 2...
✔ category health-beauty Page 2: 48 products scraped.
Fetching page 3...
```

This will help you debug whether the scraper ran successfully and if they encountered any error.

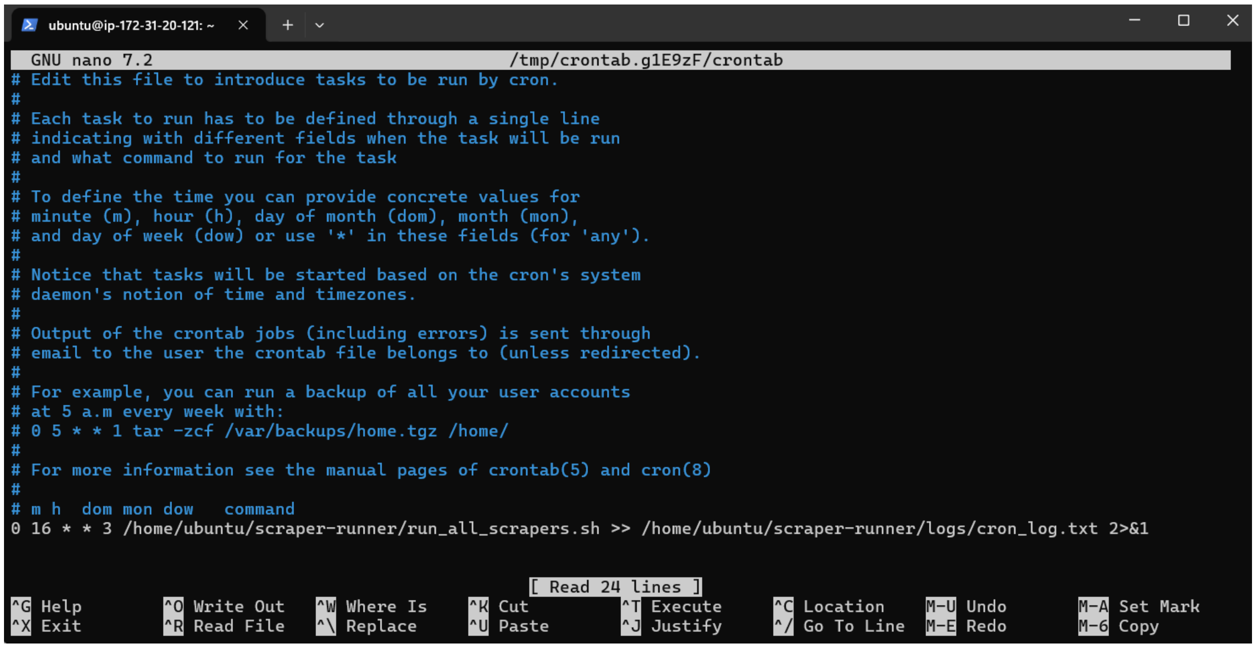
Cron Job Automation

The EC2 instance is configured with a cron job that automatically runs run_all_scrapers.sh at the scheduled time and writes logs to cron_log.txt.

To view or edit the cron job:

crontab -e

You will see the cron job set up to automatically run the run_all_scrapers.sh file and send logs to cron_log.txt



```
  GNU nano 7.2                                  /tmp/crontab.g1E9zF/crontab
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# 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 16 * * 3 /home/ubuntu/scraper-runner/run_all_scrapers.sh >> /home/ubuntu/scraper-runner/logs/cron_log.txt 2>&1


                                            [ Read 24 lines ]
^G Help         ^O Write Out    ^W Where Is     ^K Cut          ^T Execute      ^C Location     M-U Undo        M-A Set Mark
^X Exit         ^R Read File    ^\ Replace      ^U Paste        ^J Justify      ^/ Go To Line   M-E Redo        M-6 Copy
```

This is the part on the EC2 instance. But, to make sure this cron runs successfully on this instance, we need to make sure that the EC2 instance is up and running at least 15 minutes before the above scheduled cron job.

EventBridge Integration

Since the cron job requires the EC2 instance to be running beforehand, Amazon EventBridge is used to automatically **start the EC2 instance 15 minutes before the scheduled cron job**.

Setup Summary:

1. Create an IAM role with permissions for EC2 start/describe actions.
2. Attach a policy with the following JSON:

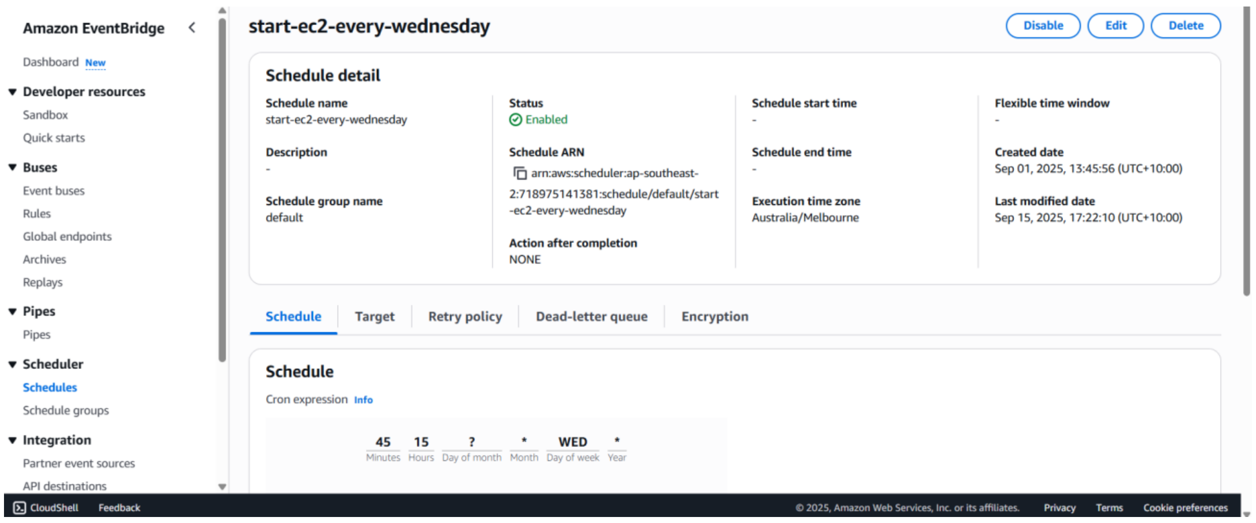
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:StartInstances",
        "ec2:DescribeInstances"
      ],
      "Resource": "arn:aws:ec2:ap-southeast-2:718975141381:instance/i-0a0483386e8bf25e6"
    }
  ]
}
```

3. Assign this IAM role to the EventBridge schedule.
4. Create a schedule that triggers 15 minutes before the cron job.
5. Configure EventBridge to target the EC2 instance and execute **StartInstances**.

6. In addition, a separate IAM role with **AmazonEC2FullAccess** policy is attached to the EC2 instance, enabling it to stop itself once the scrapers finish execution.

The detailed steps with screenshot are provided below:

You can see I have this EventBridge rule currently on AWS that aligns with our cron job above.



start-ec2-every-wednesday

Schedule detail

Schedule name start-ec2-every-wednesday	**Status** Enabled	**Schedule start time** -	**Flexible time window** -
Description -	**Schedule ARN** arn:aws:scheduler:ap-southeast-2:718975141381:schedule/default/start-ec2-every-wednesday	**Schedule end time** -	**Created date** Sep 01, 2025, 13:45:56 (UTC+10:00)
Schedule group name default	**Action after completion** NONE	**Execution time zone** Australia/Melbourne	**Last modified date** Sep 15, 2025, 17:22:10 (UTC+10:00)

Schedule

Cron expression

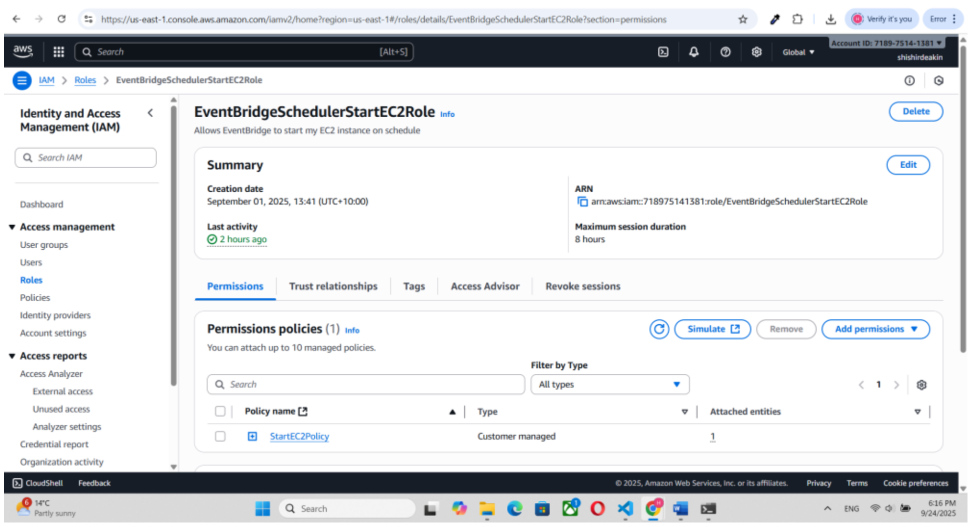
Minutes	Hours	Day of month	Month	Day of week	Year
45	15	?	*	WED	*

So, you need to create a similar EventBridge rule on your AWS account.

But note that the EventBridge should have permission to auto start the EC2 instance. So, you should first create the required IAM roles and attach this so that it has permission to start instance.

So, go to your IAM console. And create role.

You need to create role like below



https://us-east-1.console.aws.amazon.com/iamv2/home?region=us-east-1#/roles/details/EventBridgeSchedulerStartEC2Role?section=permissions

IAM > Roles > EventBridgeSchedulerStartEC2Role

EventBridgeSchedulerStartEC2Role

Allows EventBridge to start my EC2 instance on schedule

Summary

Creation date September 01, 2025, 13:41 (UTC+10:00)

ARN arn:aws:iam::718975141381:role/EventBridgeSchedulerStartEC2Role

Last activity 2 hours ago

Maximum session duration 8 hours

Permissions policies (1)

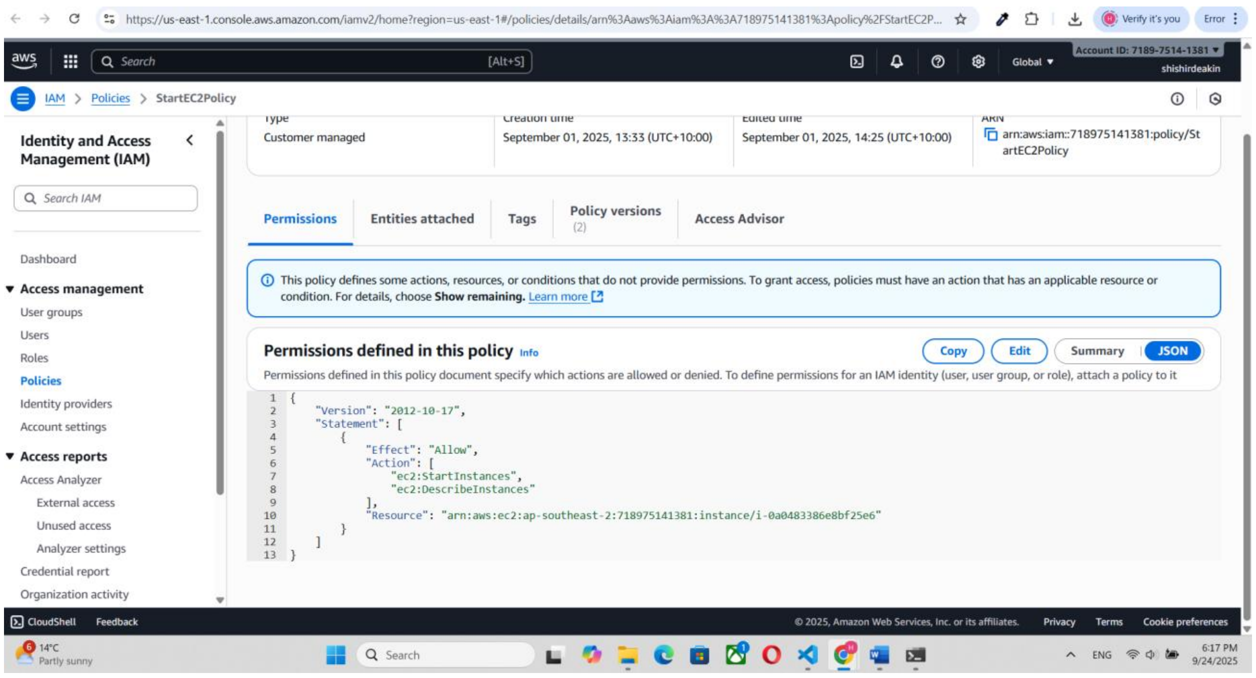
You can attach up to 10 managed policies.

Policy name	Type	Attached entities
StartEC2Policy	Customer managed	1

And attach Policy to it

Go to Policy and create policy with the following JSON:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:StartInstances",
        "ec2:DescribeInstances"
      ],
      "Resource": "arn:aws:ec2:ap-southeast-2:718975141381:instance/i-0a0483386e8bf25e6"
    }
  ]
}
```



https://us-east-1.console.aws.amazon.com/iamv2/home?region=us-east-1#/policies/details/arn%3Aaws%3Aiam%3A%3A718975141381%3Apolicy%2FStartEC2P...

Account ID: 7189-7514-1381 shishirdeakin

IAM > Policies > StartEC2Policy

Type	Creation time	Edited time	ARN
Customer managed	September 01, 2025, 13:33 (UTC+10:00)	September 01, 2025, 14:25 (UTC+10:00)	arn:aws:iam::718975141381:policy/StartEC2Policy

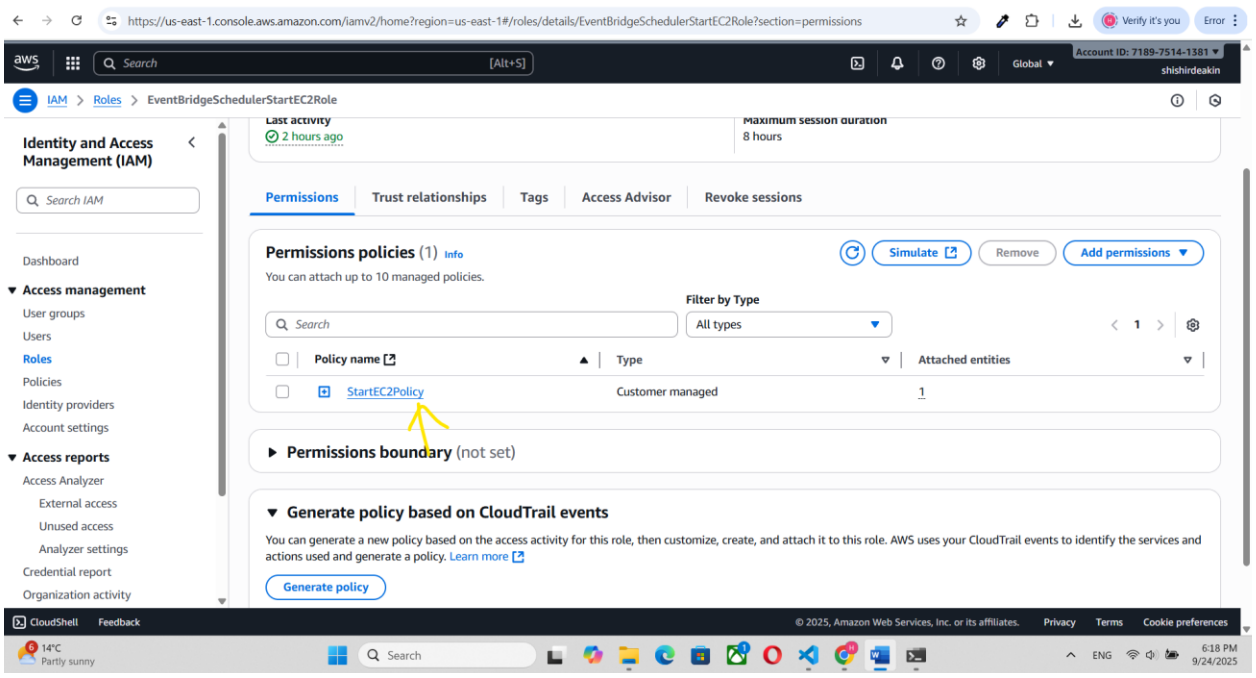
This policy defines some actions, resources, or conditions that do not provide permissions. To grant access, policies must have an action that has an applicable resource or condition. For details, choose **Show remaining.** Learn more

Permissions defined in this policy Info

Permissions defined in this policy document specify which actions are allowed or denied. To define permissions for an IAM identity (user, user group, or role), attach a policy to it

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:StartInstances",
                "ec2:DescribeInstances"
            ],
            "Resource": "arn:aws:ec2:ap-southeast-2:718975141381:instance/i-0a0483386e8bf25e6"
        }
    ]
}
```

Then you have to attach this policy to the EventBridge role you created like below:



https://us-east-1.console.aws.amazon.com/iamv2/home?region=us-east-1#/roles/details/EventBridgeSchedulerStartEC2Role?section=permissions

IAM > Roles > EventBridgeSchedulerStartEC2Role

Last activity: 2 hours ago
Maximum session duration: 8 hours

Permissions | Trust relationships | Tags | Access Advisor | Revoke sessions

Permissions policies (1)

You can attach up to 10 managed policies.

Filter by Type: All types

	Policy name	Type	Attached entities
☐	StartEC2Policy	Customer managed	1

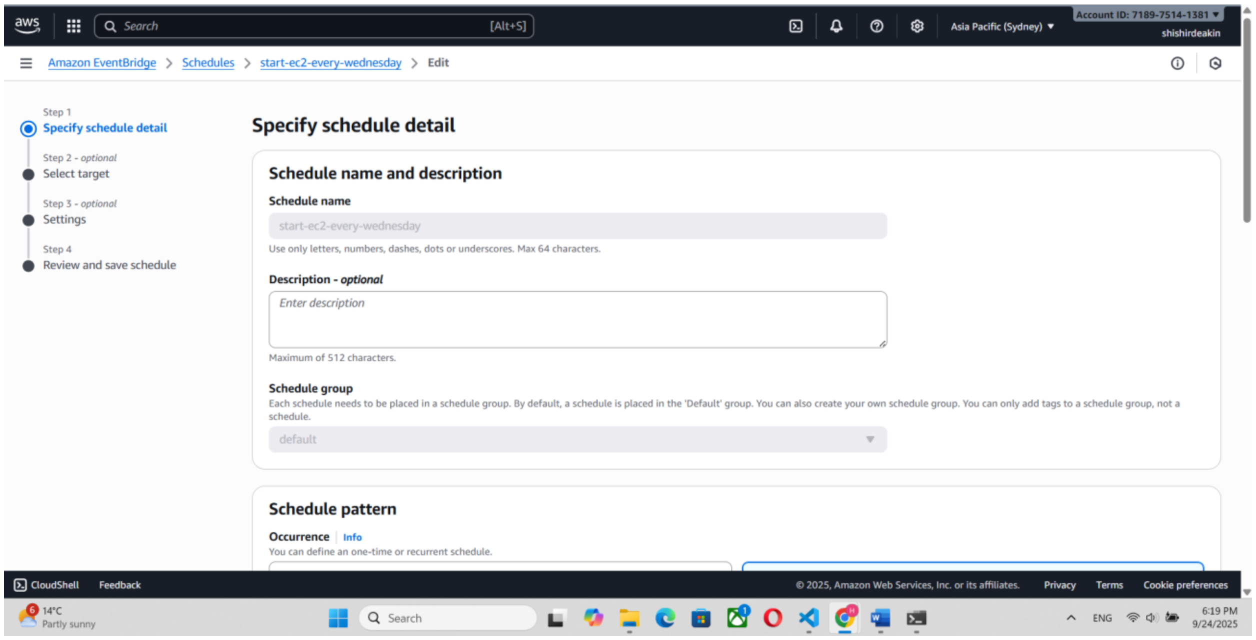
Permissions boundary (not set)

Generate policy based on CloudTrail events

You can generate a new policy based on the access activity for this role, then customize, create, and attach it to this role. AWS uses your CloudTrail events to identify the services and actions used and generate a policy. Learn more

Once that is done, come to Amazon EventBridge Schedule and create Schedule

Specify schedule name and details



Amazon EventBridge > Schedules > start-ec2-every-wednesday > Edit

Step 1 Specify schedule detail
Step 2 - optional Select target
Step 3 - optional Settings
Step 4 Review and save schedule

Specify schedule detail

Schedule name and description

Schedule name: start-ec2-every-wednesday

Use only letters, numbers, dashes, dots or underscores. Max 64 characters.

Description - optional: Enter description

Maximum of 512 characters.

Schedule group

Each schedule needs to be placed in a schedule group. By default, a schedule is placed in the 'Default' group. You can also create your own schedule group. You can only add tags to a schedule group, not a schedule.

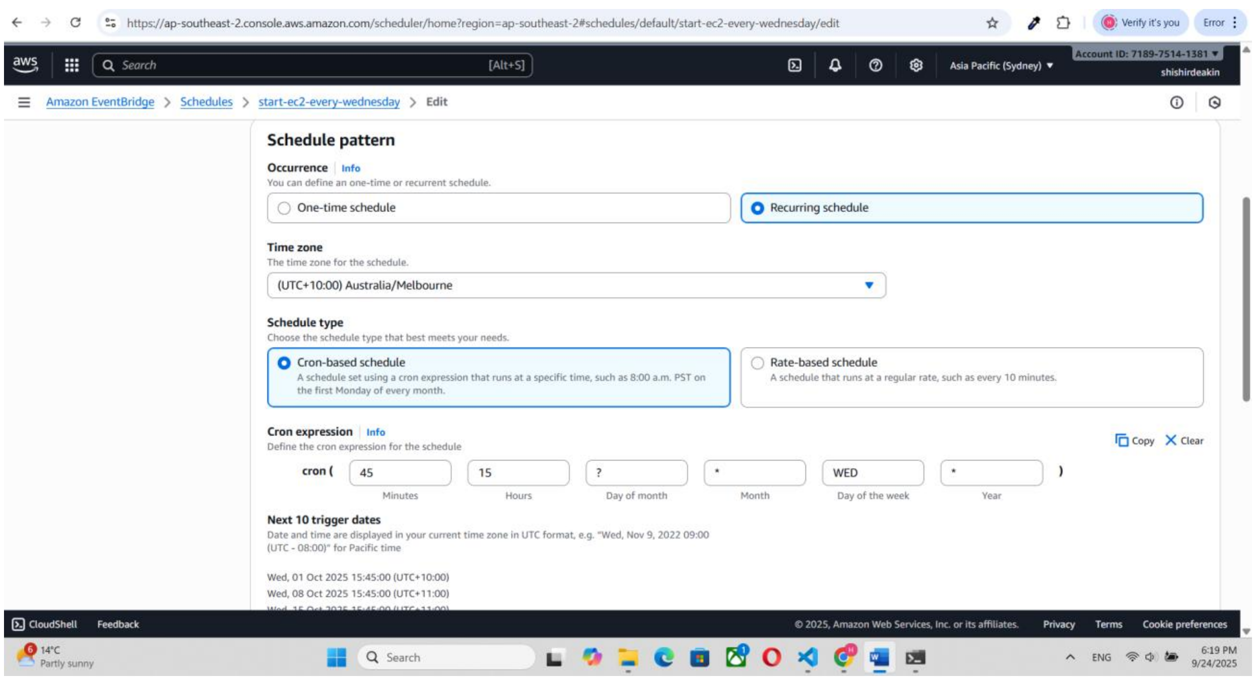
default

Schedule pattern

Occurrence

You can define an one-time or recurrent schedule.

Create cron pattern like below:



Schedule pattern

Occurrence Info
You can define an one-time or recurrent schedule.

One-time schedule [✓ Recurring schedule]

Time zone
The time zone for the schedule.
(UTC+10:00) Australia/Melbourne

Schedule type
Choose the schedule type that best meets your needs.

[✓ Cron-based schedule] A schedule set using a cron expression that runs at a specific time, such as 8:00 a.m. PST on the first Monday of every month.

Rate-based schedule A schedule that runs at a regular rate, such as every 10 minutes.

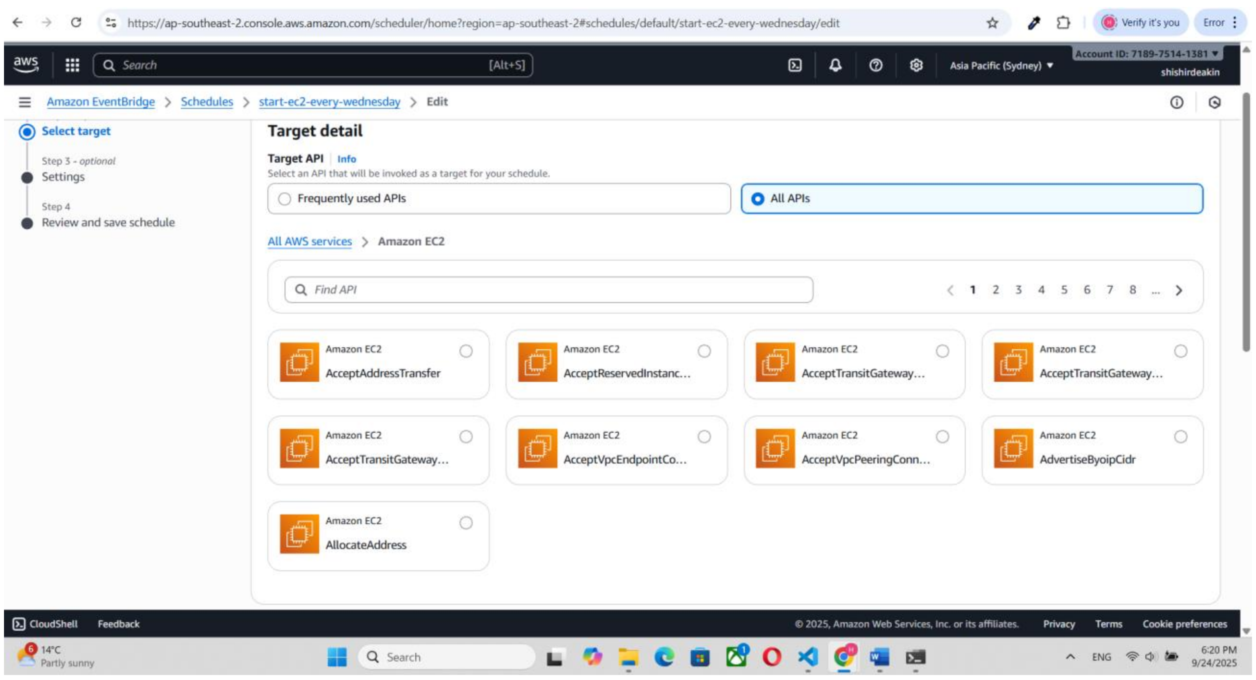
Cron expression Info
Define the cron expression for the schedule

	Minutes	Hours	Day of month	Month	Day of the week	Year	
cron (	45	15	?	*	WED	*	)

Next 10 trigger dates
Date and time are displayed in your current time zone in UTC format, e.g. "Wed, Nov 9, 2022 09:00 (UTC - 08:00)" for Pacific time

Wed, 01 Oct 2025 15:45:00 (UTC+10:00)
Wed, 08 Oct 2025 15:45:00 (UTC+11:00)

In Target, find Amazon EC2 and find **StartInstances**



Target detail

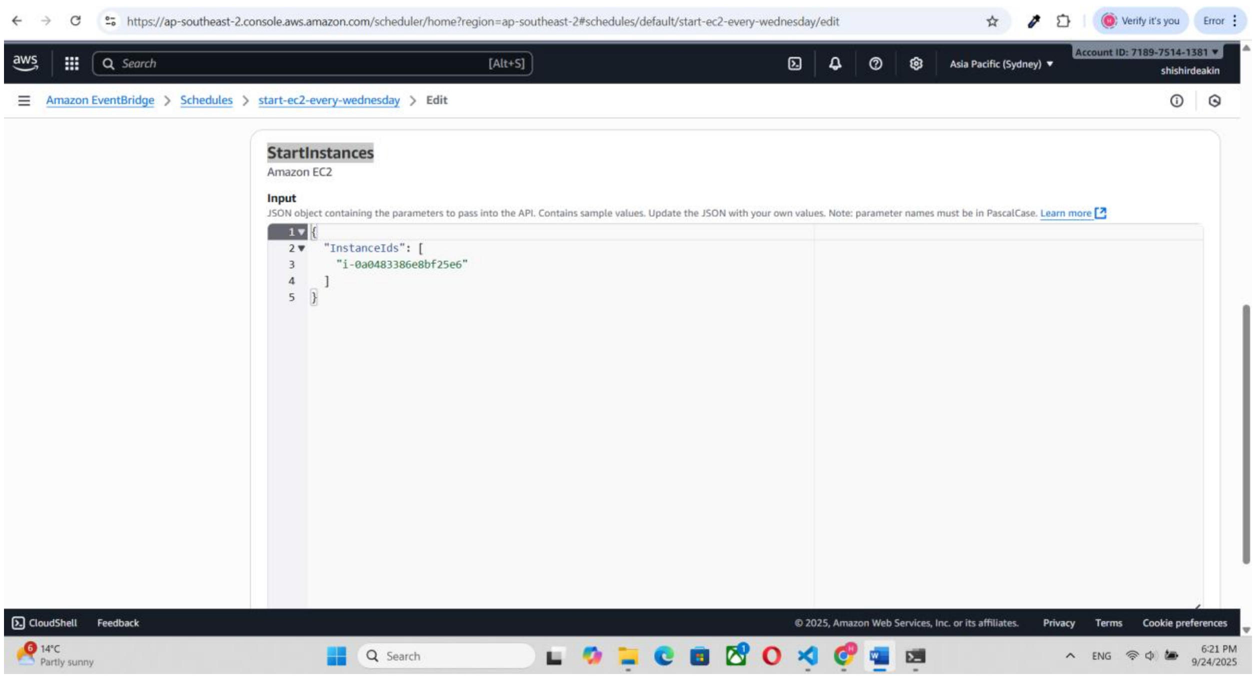
Target API Info
Select an API that will be invoked as a target for your schedule.

Frequently used APIs [✓ All APIs]

All AWS services > Amazon EC2

Amazon EC2 AcceptAddressTransfer
Amazon EC2 AcceptReservedInstanc…
Amazon EC2 AcceptTransitGateway…
Amazon EC2 AcceptTransitGateway…
Amazon EC2 AcceptTransitGateway…
Amazon EC2 AcceptVpcEndpointCo…
Amazon EC2 AcceptVpcPeeringConn…
Amazon EC2 AdvertiseByoipCidr
Amazon EC2 AllocateAddress

Then provide your instance-id like below:

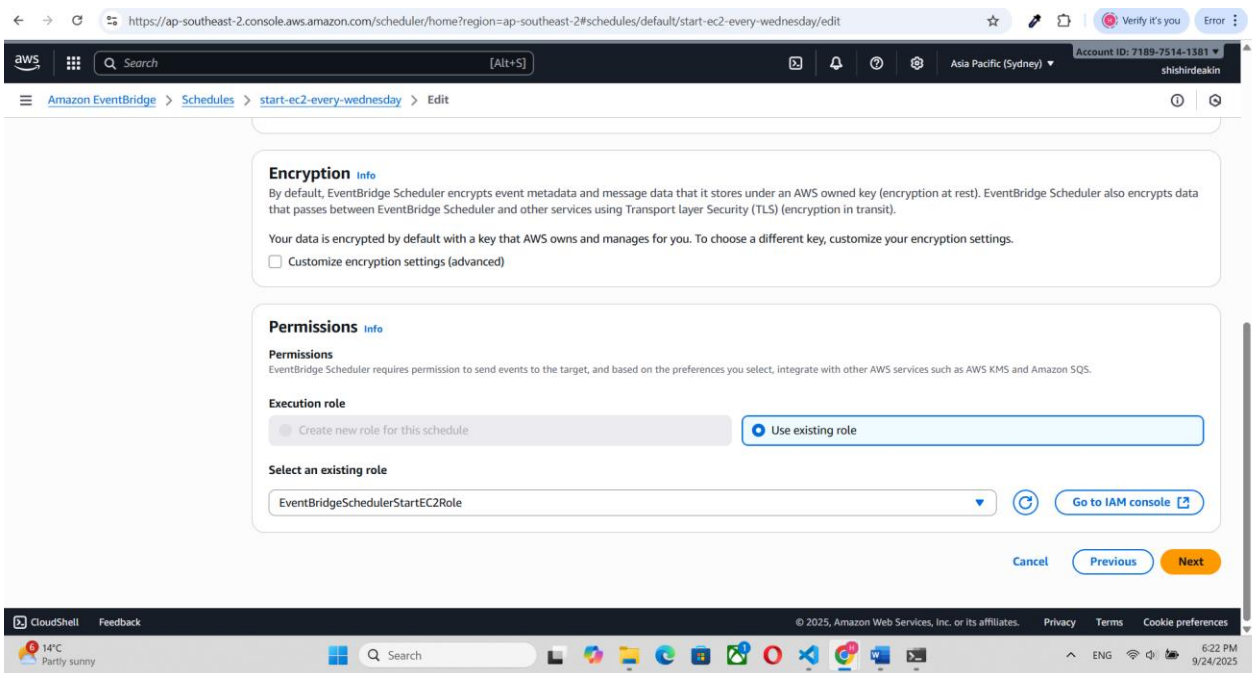

https://ap-southeast-2.console.aws.amazon.com/scheduler/home?region=ap-southeast-2#schedules/default/start-ec2-every-wednesday/edit
Verify it's you
Error
Search [Alt+S]
Asia Pacific (Sydney)
Account ID: 7189-7514-1381
shishirdeakin
Amazon EventBridge > Schedules > start-ec2-every-wednesday > Edit
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
14°C Partly sunny
6:21 PM 9/24/2025

StartInstances
Amazon EC2

Input
JSON object containing the parameters to pass into the API. Contains sample values. Update the JSON with your own values. Note: parameter names must be in PascalCase. Learn more

```
{
  "InstanceIds": [
    "i-0a0483386e8bf25e6"
  ]
}
```

Next, in the Permissions, select Use existing role and select the permission you created above.


https://ap-southeast-2.console.aws.amazon.com/scheduler/home?region=ap-southeast-2#schedules/default/start-ec2-every-wednesday/edit
Verify it's you
Error
Search [Alt+S]
Asia Pacific (Sydney)
Account ID: 7189-7514-1381
shishirdeakin
Amazon EventBridge > Schedules > start-ec2-every-wednesday > Edit
Go to IAM console
Cancel
Previous
Next
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
14°C Partly sunny
6:22 PM 9/24/2025

Encryption Info
By default, EventBridge Scheduler encrypts event metadata and message data that it stores under an AWS owned key (encryption at rest). EventBridge Scheduler also encrypts data that passes between EventBridge Scheduler and other services using Transport layer Security (TLS) (encryption in transit).

Your data is encrypted by default with a key that AWS owns and manages for you. To choose a different key, customize your encryption settings.

☐ Customize encryption settings (advanced)

Permissions Info
Permissions
EventBridge Scheduler requires permission to send events to the target, and based on the preferences you select, integrate with other AWS services such as AWS KMS and Amazon SQS.

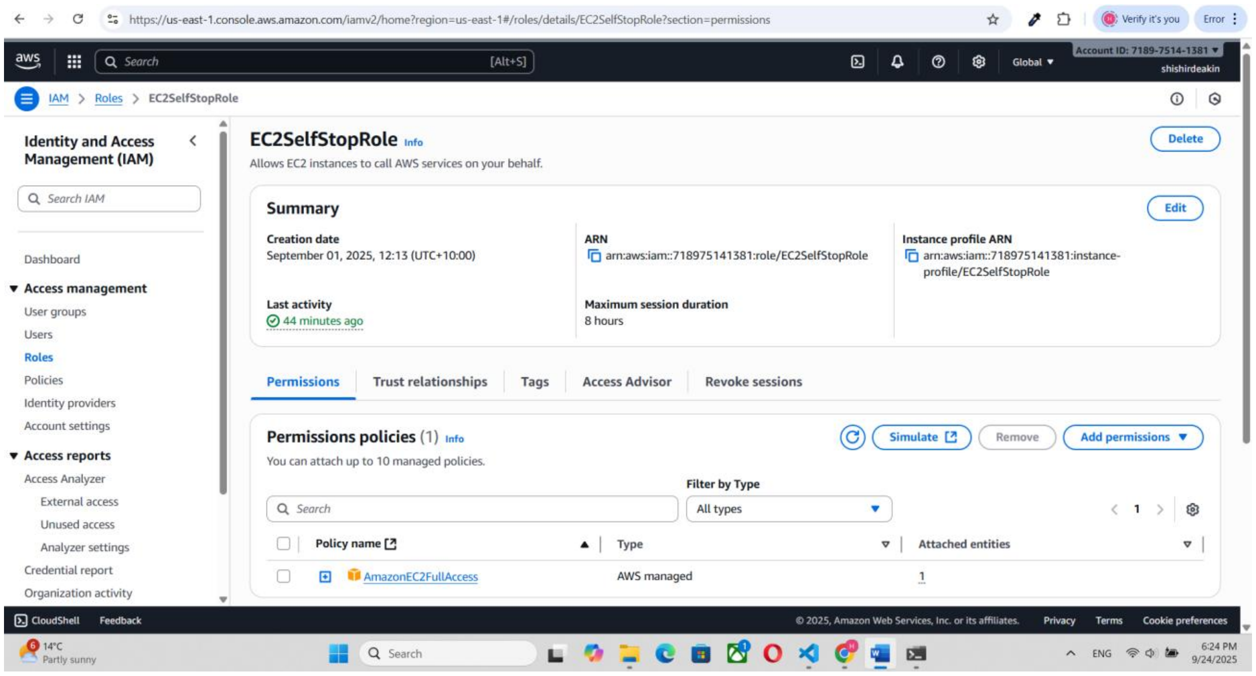
Execution role
Create new role for this schedule [✓ Use existing role]

Select an existing role
EventBridgeSchedulerStartEC2Role

Finally, save the schedule.

Now, you have to create another IAM role that allows EC2 instance to stop it by itself once the scraping is complete.

In this role, attach AmazonEC2FullAccess policy.


https://us-east-1.console.aws.amazon.com/iamv2/home?region=us-east-1#/roles/details/EC2SelfStopRole?section=permissions

IAM > Roles > EC2SelfStopRole

EC2SelfStopRole

Allows EC2 instances to call AWS services on your behalf.

Summary

Creation date	ARN	Instance profile ARN
September 01, 2025, 12:13 (UTC+10:00)	arn:aws:iam::718975141381:role/EC2SelfStopRole	arn:aws:iam::718975141381:instance-profile/EC2SelfStopRole
Last activity: 44 minutes ago	Maximum session duration: 8 hours	

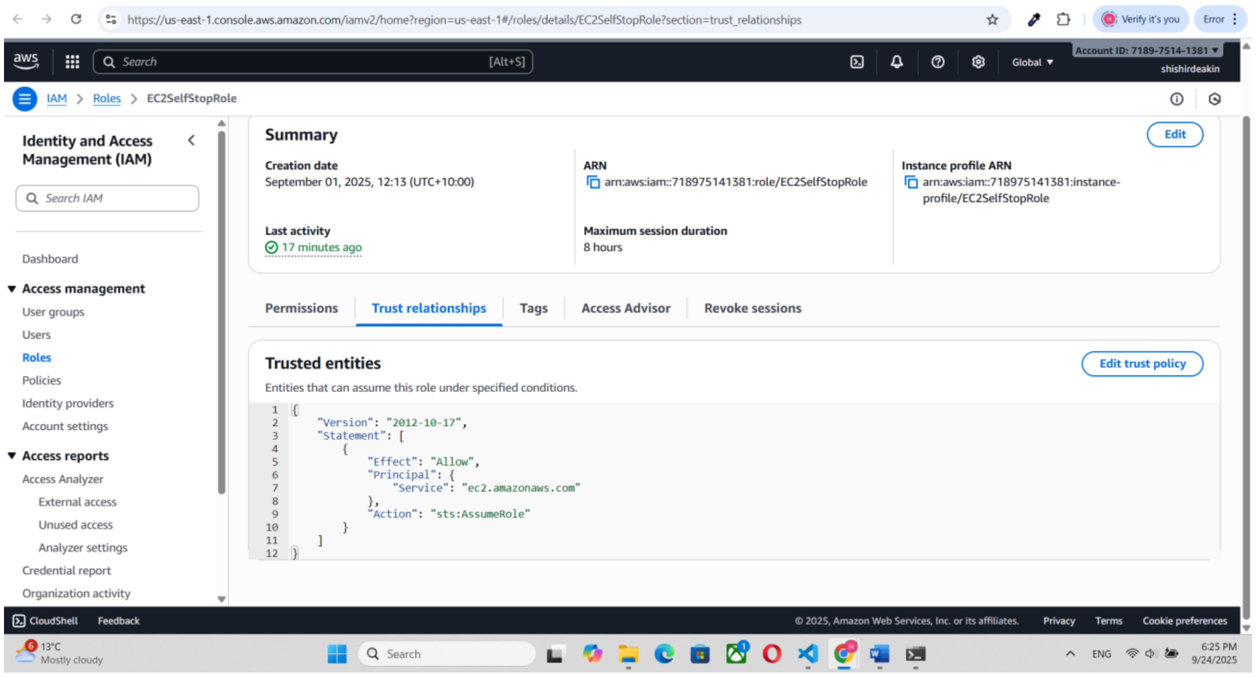
Permissions | Trust relationships | Tags | Access Advisor | Revoke sessions

Permissions policies (1)

You can attach up to 10 managed policies.

Filter by Type: All types

Policy name	Type	Attached entities
AmazonEC2FullAccess	AWS managed	1


https://us-east-1.console.aws.amazon.com/iamv2/home?region=us-east-1#/roles/details/EC2SelfStopRole?section=trust_relationships

IAM > Roles > EC2SelfStopRole

Summary

Creation date	ARN	Instance profile ARN
September 01, 2025, 12:13 (UTC+10:00)	arn:aws:iam::718975141381:role/EC2SelfStopRole	arn:aws:iam::718975141381:instance-profile/EC2SelfStopRole
Last activity: 17 minutes ago	Maximum session duration: 8 hours	

Permissions | Trust relationships | Tags | Access Advisor | Revoke sessions

Trusted entities

Entities that can assume this role under specified conditions.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ec2.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Conclusion

With the AMI, cron job, and EventBridge integration in place, this automation framework ensures the scrapers run reliably on a weekly basis with minimal manual intervention. All results are stored in MongoDB collections for further processing. Future students can continue building on this foundation by extending the scraping logic or improving automation while reusing the existing infrastructure.