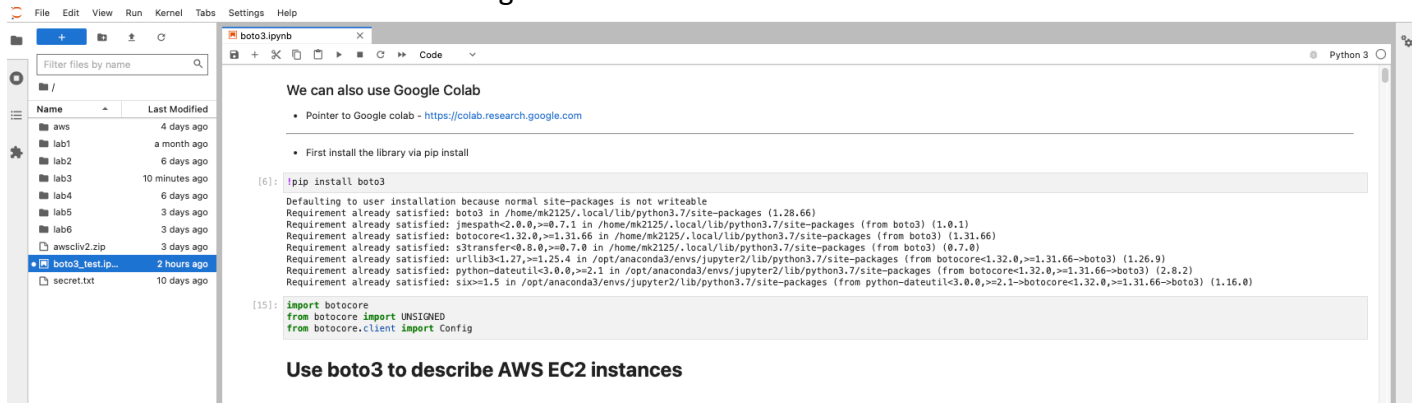


Mushran Khan

Lab3

First the boto3 was installed and configured.



Then we need to create an key ID and use the and access key to connect to my console. This was used to check all the AWS EC2 instances. The following code was used.

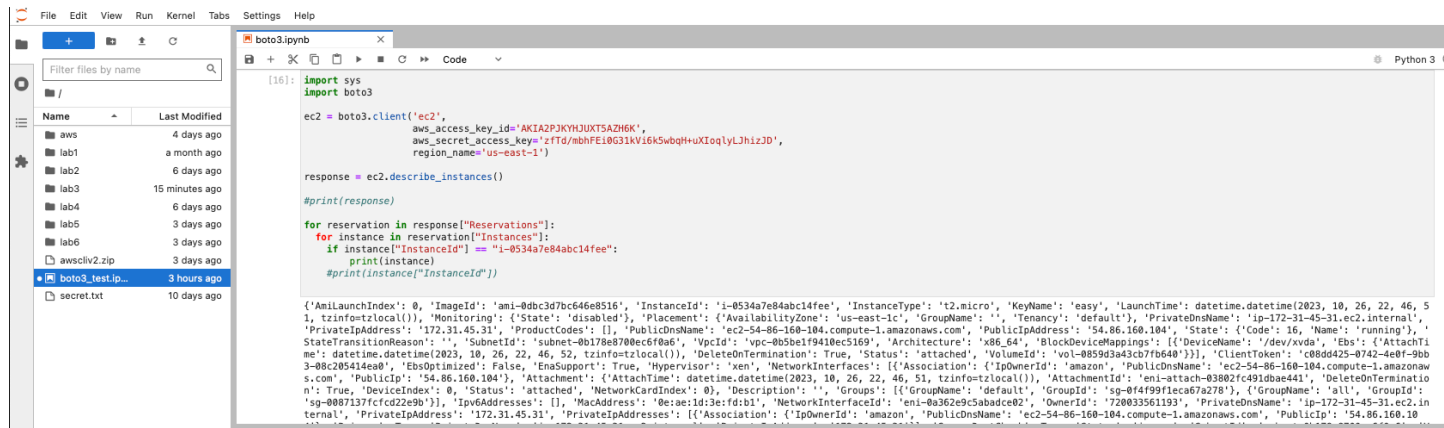
```
import sys
import boto3
```

```
ec2 = boto3.client('ec2',
                   aws_access_key_id='AKIA2PJKYHJUXT5AZH6K',
                   aws_secret_access_key='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',
                   region_name='us-east-1')
```

```
response = ec2.describe_instances()
```

```
#print(response)
```

```
for reservation in response["Reservations"]:
    for instance in reservation["Instances"]:
        if instance["InstanceId"] == "i-0534a7e84abc14fee":
            print(instance)
        #print(instance["InstanceId"])
```



Using the boto3, I create a new VM in Amazon EC2 (“micro” type) and named it “baseline VM”.
The follow code was used.

```
tags = [
    {
        "ResourceType": "instance",
        "Tags": [
            {
                "Key": "Name",
                "Value": "baseline VM"
            }
        ]
    }
]
```

```
new_instances = ec2.run_instances(
    ImageId="ami-0df435f331839b2d6",
    MinCount=1,
    MaxCount=1,
    InstanceType="t2.micro",
    KeyName="easy",
    TagSpecifications = tags,
)
```

```
print(new_instances["Instances"][0]["InstanceId"])
```

boto3.ipynb

```

4}), {'Primary': True, 'PrivateDnsName': 'ip-172-31-45-31.ec2.internal', 'PrivateIpAddress': '172.31.45.31'}], 'SourceDestCheck': True, 'Status': 'in-use', 'SubnetId': 'subnet-0b178e8700ec6f0a6', 'V
pcId': 'vpc-0b5be1f9410ec5169', 'InterfaceType': 'interface'}, {'RootDeviceName': '/dev/xvda', 'RootDeviceType': 'ebs', 'SecurityGroups': [{'GroupName': 'default', 'GroupId': 'sg-8f4f99f1eca67a27
8'}], {'GroupName': 'all', 'GroupId': 'sg-0087137fcfd422e9b'}], 'SourceDestCheck': True, 'Tags': [{'Key': 'Name', 'Value': 'baseLine VM'}], 'VirtualizationType': 'hvm', 'CpuOptions': {'CoreCount':
1, 'ThreadsPerCore': 1}, 'CapacityReservationSpecification': {'CapacityReservationPreference': 'open'}, 'HibernationOptions': {'Configured': False}, 'MetadataOptions': {'State': 'applied', 'HttpTok
ens': 'required', 'HttpPutResponseHopLimit': 2, 'HttpEndpoint': 'enabled', 'HttpProtocolIpv6': 'disabled', 'InstanceMetadataTags': 'disabled'}, 'EnclaveOptions': {'Enabled': False}, 'BootMode': 'ue
fi-preferred', 'PlatformDetails': 'Linux/UNIX', 'UsageOperation': 'RunInstances', 'UsageOperationUpdateTime': datetime.datetime(2023, 10, 26, 22, 46, 51, tzinfo=tzlocal()), 'PrivateDnsNameOptions':
{'HostnameType': 'ip-name', 'EnableResourceNameDnsARecord': True, 'EnableResourceNameDnsAAAARecord': False}, 'MaintenanceOptions': {'AutoRecovery': 'default'}, 'CurrentInstanceBootMode': 'legacy-bi
os'})

```

Use boto3 to create new AWS EC2 instance

```

[13]: tags = [
        {
            "ResourceType": "instance",
            "Tags": [
                {
                    "Key": "Name",
                    "Value": "baseLine VM"
                }
            ]
        }
    ]

    new_instances = ec2.run_instances(
        ImageId="ami-0d7435f331839b2d6",
        MinCount=1,
        MaxCount=1,
        InstanceType="t2.micro",
        KeyName="easy",
        TagSpecifications = tags,
    )

    print(new_instances["Instances"][0]["InstanceId"])

```

i-0af4eb4522035d802

S3 operations

- Reference: <https://boto3.amazonaws.com/v1/documentation/api/latest/guide/s3-examples.html>

aws

Services Search [Option+S]

EC2 Dashboard

EC2 Global View

Events

Instances

Instances

Instance Types

Launch Templates

Spot Requests

Savings Plans

Reserved Instances

Dedicated Hosts

Capacity Reservations

Images

AMIs

AMI Catalog

Elastic Block Store

Volumes

CloudShell Feedback

Instances (1/11) Info

Find Instance by attribute or tag (case-sensitive)

Instance state (client) != terminated X Instance state (client) != terminated X Instance state (client) != terminated X Show more (+1) Clear filters

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...
<input checked="" type="checkbox"/>	secondary VM	i-0a87ddb105b9e8a4f	Terminated	t2.micro	-	No alarms	us-east-1b	-	-
<input type="checkbox"/>	baseline VM	i-0534a7e84abc14fee	Running	t2.micro	2/2 checks passed	No alarms	us-east-1c	ec2-54-86-160-104.co...	54.86.160.104
<input type="checkbox"/>	secondary VM	i-0573811d281f4f0dd	Terminated	t2.micro	-	No alarms	us-east-1b	-	-
<input type="checkbox"/>	secondary VM	i-09ede5e9160ca7920	Terminated	t2.micro	-	No alarms	us-east-1b	-	-
<input type="checkbox"/>	docker	i-0bd640df7e3f0e365	Running	t2.micro	2/2 checks passed	No alarms	us-east-1b	ec2-54-175-91-198.co...	54.175.91.198
<input type="checkbox"/>	secondary VM	i-062ec5a279ca32b2e	Terminated	t2.micro	-	No alarms	us-east-1b	-	-
<input type="checkbox"/>	secondary VM	i-0ff320b2687fd9a2c	Terminated	t2.micro	-	No alarms	us-east-1b	-	-

Instance: i-0a87ddb105b9e8a4f (secondary VM)

Details Security Networking Storage Status checks Monitoring Tags

Instance summary Info

© 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preference

Using the Instance ID and the follow code, to monitor the CPUUtilization of the baseline VM in an endless loop.

```

now = datetime.now()
client = boto3.client("cloudwatch", region_name="us-east-1")
response = client.get_metric_statistics(
    Namespace="AWS/EC2",
    MetricName="CPUUtilization",

```

```

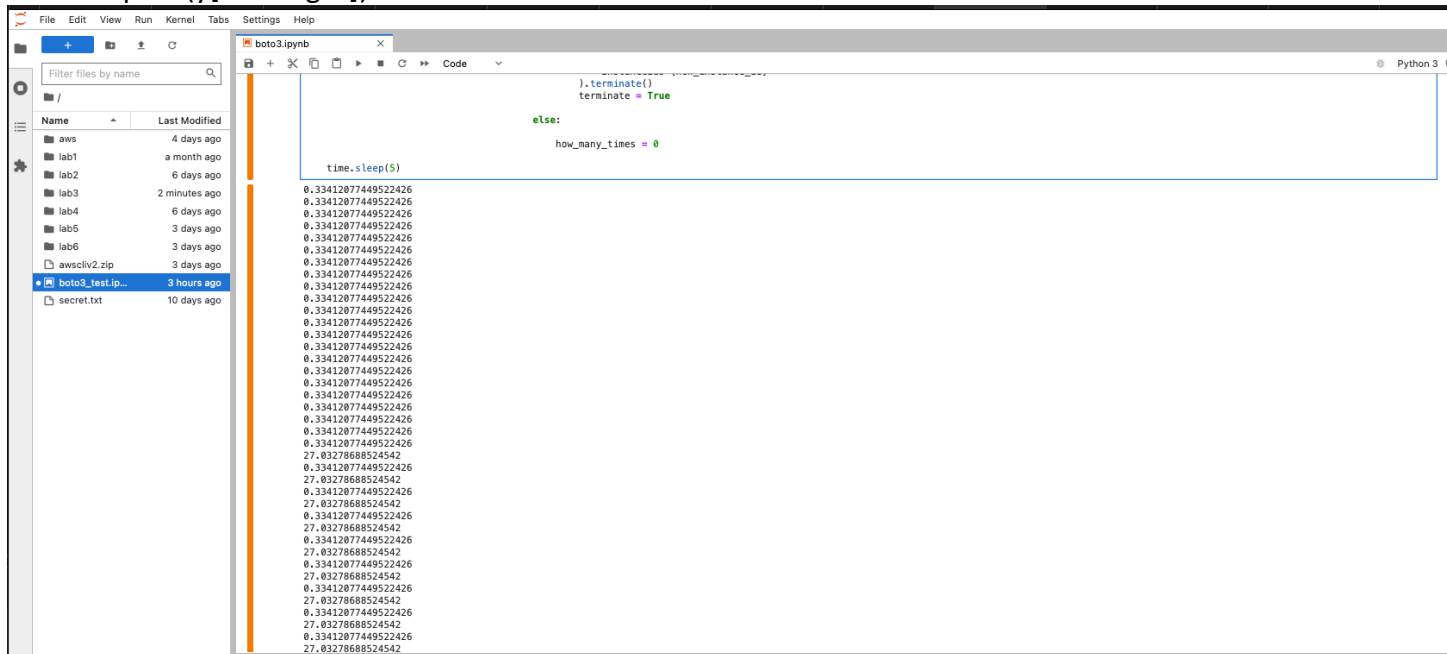
Dimensions=[
    {"Name": "InstanceId", "Value": "i-0534a7e84abc14fee"},
],
StartTime=now - timedelta(seconds=600),
EndTime=now,
Period=60,
Statistics=[
    "Average",
],
Unit="Percent",
)

```

```

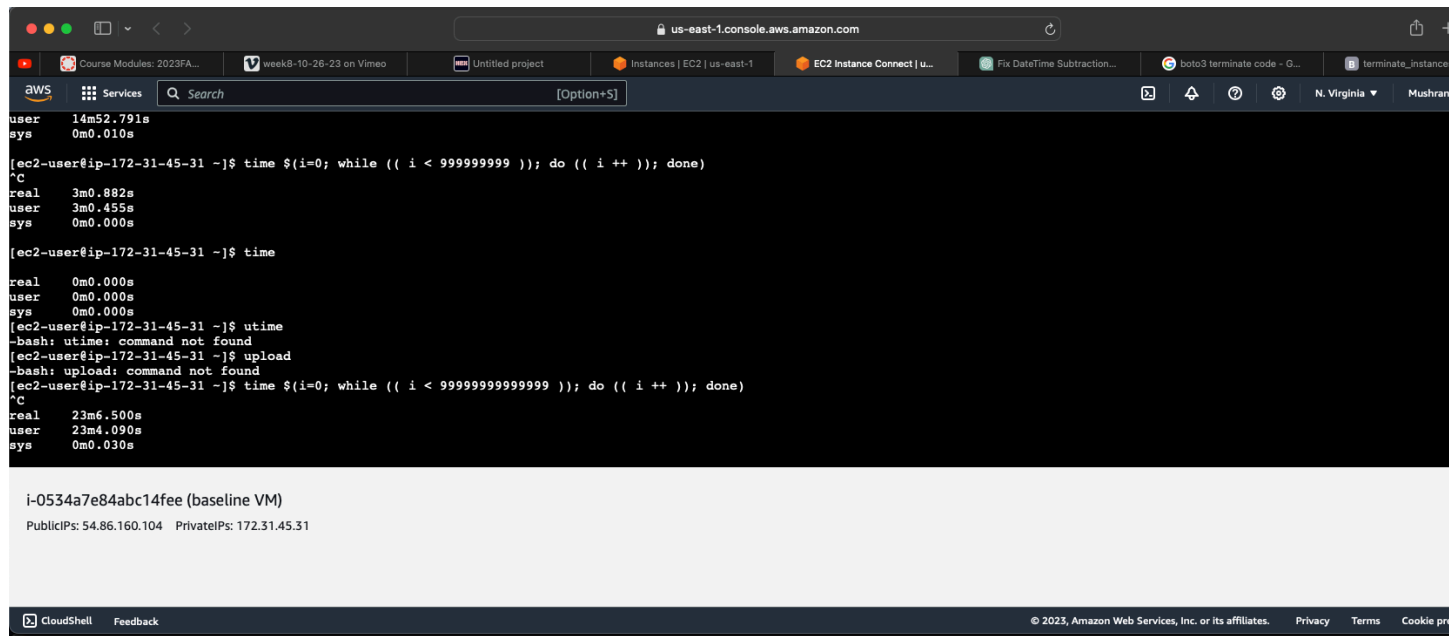
for k, v in response.items():
    if k == "Datapoints":
        for y in v:
            print(y["Average"])

```



Load was generated by connecting to the baseline VM and using the benchmark code.

```
time $(i=0; while (( i < 999999999 )); do (( i ++ )); done)
```



The screenshot shows an AWS CloudShell terminal window with the following content:

```
user      14m52.791s
sys       0m0.010s

[ec2-user@ip-172-31-45-31 ~]$ time $(i=0; while (( i < 999999999 )); do (( i ++ )); done)
^C
real      3m0.882s
user      3m0.455s
sys       0m0.000s

[ec2-user@ip-172-31-45-31 ~]$ time
real      0m0.000s
user      0m0.000s
sys       0m0.000s
[ec2-user@ip-172-31-45-31 ~]$ utime
-bash: utime: command not found
[ec2-user@ip-172-31-45-31 ~]$ upload
-bash: upload: command not found
[ec2-user@ip-172-31-45-31 ~]$ time $(i=0; while (( i < 9999999999999 )); do (( i ++ )); done)
^C
real      23m6.500s
user      23m4.090s
sys       0m0.030s
```

i-0534a7e84abc14fee (baseline VM)
PublicIPs: 54.86.160.104 PrivateIPs: 172.31.45.31

CloudShell Feedback

© 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

We used the following code to automatically create a new secondary VM when the CPU utilization crossed the 50% usage.

if y["Average"] > 50 and not secondary:

 if how_many_times < 5:
 how_many_times += 1

else:

 how_many_times = 0

Create secondary VM

```
tags = [  
    {  
        "ResourceType": "instance",  
        "Tags": [{"Key": "Name", "Value": "secondary VM"}],  
    }  
]
```

```
new_instances = ec2.create_instances(  
    ImageId="ami-0df435f331839b2d6",  
    MinCount=1,  
    MaxCount=1,  
    InstanceType="t2.micro",  
    KeyName="easy",  
    TagSpecifications=tags,  
)
```

```
new_instance_id = new_instances[0].id
print(f"New Instance ID: {new_instance_id}")
```

The screenshot displays a Jupyter Notebook interface at the top, with a file explorer on the left showing a directory with files like 'aws', 'lab1', 'lab2', 'lab3', 'lab4', 'lab5', 'lab6', 'awscliV2.zip', 'boto3_test.ip...', and 'secret.txt'. The notebook code shows a list of instance IDs, with the last line indicating a new instance ID: 'New Instance ID: i-0e9b35f3140b8fb6a'.

Below the notebook is the AWS Management Console. The left sidebar shows the navigation menu with 'Instances' selected. The main panel shows a list of EC2 instances. The table below represents the data shown in the console:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
docker	i-0bd640df7e3f0e365	Running	t2.micro	2/2 checks passed	No alarms	us-east-1b	ec2-54-175-91-198.co...
secondary VM	i-062ec5a279ca32b2e	Terminated	t2.micro	-	No alarms	us-east-1b	-
secondary VM	i-0ff320b2687fd9a2c	Terminated	t2.micro	-	No alarms	us-east-1b	-
secondary VM	i-069cce145cbd97c32	Terminated	t2.micro	-	No alarms	us-east-1b	-
secondary VM	i-06d734b472afaafc3	Terminated	t2.micro	-	No alarms	us-east-1b	-
secondary VM	i-09d9e32db2465467b	Terminated	t2.micro	-	No alarms	us-east-1b	-
secondary VM	i-0e9b35f3140b8fb6a	Running	t2.micro	2/2 checks passed	No alarms	us-east-1b	ec2-54-208-39-190.co...

The console also shows details for the selected instance 'i-0e9b35f3140b8fb6a (secondary VM)', including tabs for Details, Security, Networking, Storage, Status checks, Monitoring, and Tags.

Once the CPU utilization came down below 50%, the code automatically notices it and terminate the secondary VM machine.

while not terminate:

while not terminate:

```
now = datetime.now()
client = boto3.client("cloudwatch", region_name="us-east-1")
response = client.get_metric_statistics(
```

```

Namespace="AWS/EC2",
MetricName="CPUUtilization",
Dimensions=[
    {
        "Name": "InstanceId",
        "Value": "i-0534a7e84abc14fee",
    },
],
StartTime=now - timedelta(seconds=600),
EndTime=now,
Period=60,
Statistics=[
    "Average",
],
Unit="Percent",
)
for k, v in response.items():
    if k == "Datapoints":
        for y in v:
            print(y["Average"])

            if y["Average"] > 50 and not secondary:
                if how_many_times < 5:
                    how_many_times += 1

                else:

                    # Terminate secondary VM
                    ec2.instances.filter(InstanceIds=[new_instance_id]).terminate()
                    terminate = True

            else:

                how_many_times = 0

```

The entire code is shown below.

```

import boto3
import sys
import time

```

```

from datetime import datetime, timedelta

ec2 = boto3.resource("ec2", region_name="us-east-1")

secondary = False
how_many_times = 0

while True:

    now = datetime.now()
    client = boto3.client("cloudwatch", region_name="us-east-1")
    response = client.get_metric_statistics(
        Namespace="AWS/EC2",
        MetricName="CPUUtilization",
        Dimensions=[
            {"Name": "InstanceId", "Value": "i-0534a7e84abc14fee"},
        ],
        StartTime=now - timedelta(seconds=600),
        EndTime=now,
        Period=60,
        Statistics=[
            "Average",
        ],
        Unit="Percent",
    )

    for k, v in response.items():
        if k == "Datapoints":
            for y in v:
                print(y["Average"])

                new_instance_id = ""

                if y["Average"] > 50 and not secondary:
                    if how_many_times < 5:
                        how_many_times += 1

                else:
                    how_many_times = 0

                # Create secondary VM
                tags = [

```



```

    {
        "ResourceType": "instance",
        "Tags": [{"Key": "Name", "Value": "secondary VM"}],
    }
]

```

```

new_instances = ec2.create_instances(
    ImageId="ami-0df435f331839b2d6",
    MinCount=1,
    MaxCount=1,
    InstanceType="t2.micro",
    KeyName="easy",
    TagSpecifications=tags,
)

```

```

new_instance_id = new_instances[0].id
print(f"New Instance ID: {new_instance_id}")

```

```

secondary = True
terminate = False

```

```

while not terminate:
    now = datetime.now()
    client = boto3.client("cloudwatch", region_name="us-east-1")
    response = client.get_metric_statistics(
        Namespace="AWS/EC2",
        MetricName="CPUUtilization",
        Dimensions=[
            {
                "Name": "InstanceId",
                "Value": "i-0534a7e84abc14fee",
            },
        ],
        StartTime=now - timedelta(seconds=600),
        EndTime=now,
        Period=60,
        Statistics=[
            "Average",
        ],
        Unit="Percent",
    )
    for k, v in response.items():
        if k == "Datapoints":
            for y in v:

```

```
print(y["Average"])
```

```
if y["Average"] > 50 and not secondary:
```

```
    if how_many_times < 5:
```

```
        how_many_times += 1
```

```
    else:
```

```
        # Terminate secondary VM
```

```
        ec2.instances.filter(InstanceIds=[new_instance_id]).terminate()
```

```
        terminate = True
```

```
    else:
```

```
        how_many_times = 0
```

```
time.sleep(5)
```

The screenshot shows the AWS Management Console interface. The main content area displays the 'Instances (4)' page. A search bar at the top allows filtering by attribute or tag. Below the search bar, there are three filter buttons: 'Instance state (client) != terminated', 'Instance state (client) != terminated', and 'Instance state (client) != terminated'. A 'Show more (+1)' button and a 'Clear filters' button are also present. The table below lists the instances:

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
<input type="checkbox"/>	baseline VM	i-0534a7e84abc14fee	Running	t2.micro	2/2 checks passed	No alarms	us-east-1c	ec2-54-86...
<input type="checkbox"/>	docker	i-0bd640df7e3f0e365	Running	t2.micro	2/2 checks passed	No alarms	us-east-1b	ec2-54-17...
<input type="checkbox"/>	secondary VM	i-0ca499d391438efd1	Running	t2.micro	2/2 checks passed	No alarms	us-east-1b	ec2-23-22...
<input type="checkbox"/>	secondary VM	i-0e9b35f3140b8fb6a	Terminated	t2.micro	-	No alarms	us-east-1b	-

Below the table, there is a 'Select an instance' section with a search bar and a list of instances. The bottom of the console shows the 'CloudShell' and 'Feedback' buttons, along with the copyright notice: '© 2023, Amazon Web Services, Inc. or its affiliates.'