

DS 5220: Advanced Cloud Computing

Lab 2: Creating and Managing EC2 Instances

Faizan Khan

January 30, 2026

Table of Contents

1. Part 1: Creating an EC2 Instance via the AWS Console
2. Part 2: Creating an EC2 Instance via the AWS CLI
3. Part 3: Creating an EC2 Instance with boto3
4. Submission Requirements
5. Summary
6. Appendix: Python Script

Part 1: Creating an EC2 Instance via the AWS Console

Instance Configuration

An EC2 instance was launched using the AWS Management Console with the following configuration:

Parameter	Value
Instance Name	my-first-instance
Instance ID	i-022a54888402a734a
AMI	Amazon Linux 2023 (ami-0532be01f26a3de55)
Instance Type	t3.micro (2 vCPUs, 1 GB RAM)
Public IPv4 Address	54.86.221.227
Availability Zone	us-east-1c
Key Pair	ds5220-lab2
Security Group	launch-wizard-1 (sg-03649244fb920dad3)

Storage Configuration

Two EBS volumes were attached to the instance:

Volume	Size	Type	Mount Point
Root (nvme0n1)	8 GB	gp3 (xfs)	/
Data (nvme1n1)	10 GB	gp3 (ext4)	/data

Volume Formatting and Mounting

The data volume was formatted with ext4 filesystem and mounted to /data. The following commands were executed:

```
$ sudo file -s /dev/nvme1n1
/dev/nvme1n1: data
$ sudo mkfs -t ext4 /dev/nvme1n1
$ sudo mkdir /data
$ sudo mount /dev/nvme1n1 /data
$ sudo chown ec2-user:ec2-user /data
```

System Information

Operating system and hardware details retrieved from the instance:

Property	Value
Operating System	Amazon Linux 2023.10.20260120
CPU	Intel Xeon Platinum 8175M @ 2.50GHz (2 vCPUs)
Memory	916 MB total, 611 MB available
Architecture	x86_64
Hypervisor	KVM

Checkpoint Questions

1. What AMI did you use, and what does it contain?

Amazon Linux 2023 (version 2023.10.20260120). It is a Fedora-like Linux distribution maintained by AWS, with support until 2029. It includes essential system utilities, package management (dnf), and is optimized for EC2.

2. What are the CPU and memory specifications of a t3.micro?

2 vCPUs (Intel Xeon Platinum 8175M @ 2.50GHz) and approximately 1 GB RAM (916 MB). The t3 family provides burstable performance instances.

3. How much storage is attached to your instance, and on how many volumes?

18 GB total across 2 volumes: an 8 GB root volume (nvme0n1) and a 10 GB data volume (nvme1n1).

4. What's the difference between the root volume and the data volume?

The root volume (nvme0n1) contains the OS, is mounted at /, uses xfs filesystem, and is deleted when the instance terminates. The data volume (nvme1n1) is additional storage, mounted at /data, uses ext4 filesystem, and can persist after instance termination.

5. Why do you need the private key file to connect?

AWS uses SSH key-pair authentication instead of passwords for security. The private key (.pem file) cryptographically proves your identity to the server, which has the corresponding public key.

6. What happens to your data volume if you terminate the instance?

By default, when created separately via the console, the data volume will persist and remain available. It must be deleted manually to avoid ongoing storage charges.

Part 2: Creating an EC2 Instance via the AWS CLI

Environment Setup

The AWS CLI was configured for the us-east-1 region. The following resources were identified and created:

Resource	Value
AMI ID	ami-0157af9aea2eef346
VPC ID	vpc-022aefb2869b1d62f
Security Group ID	sg-0ca338a4901703d3c (ds5220-ssh)

Instance Launch Command

The instance was launched using `aws ec2 run-instances` with block device mappings to create both volumes simultaneously:

```
aws ec2 run-instances \
  --image-id $AMI_ID \
  --instance-type t3.micro \
  --key-name ds5220-lab2 \
  --security-group-ids $SSH_SG_ID \
  --block-device-mappings '[...]'
```

Instance Details

Parameter	Value
Instance Name	cli-created-instance
Instance ID	i-00ab6a6760bf6a92d
Public IP	54.163.196.176
Availability Zone	us-east-1c

Volume Verification

The attached volumes were verified using `aws ec2 describe-volumes`:

Volume ID	Size	Type	State	Device
vol-02352bf9cd5486f56	10 GB	gp3	in-use	/dev/sdf
vol-0b1013775a64d2412	8 GB	gp3	in-use	/dev/xvda

SSH connection verified both volumes were visible via lsblk. Instance was terminated after verification.

Part 3: Creating an EC2 Instance with boto3

Python Environment Setup

A Python virtual environment was created and boto3 was installed:

```
$ python3 -m venv venv
$ source venv/bin/activate
$ pip install boto3
```

Script Execution Output

The create_ec2.py script was executed, producing the following output:

```
Creating EC2 instance in us-east-1...
Creating instance: boto3-created-instance
Instance created: i-0061ea5ca72e3cae4
Waiting for instance to be running...
```

Instance Details

Parameter	Value
Instance ID	i-0061ea5ca72e3cae4
Instance Type	t3.micro
AMI ID	ami-0157af9aea2eef346
State	running
Public IP	3.80.229.28
Private IP	172.31.27.133
Availability Zone	us-east-1c

Attached Storage

Volume	Volume ID	Size	Type	Delete
/dev/xvda	vol-0302d5c3f81bc07dc	8 GB	gp3	True
/dev/sdf	vol-0fa3f0e17abd3aacf	10 GB	gp3	False

Instance was terminated upon user confirmation. The data volume persisted as configured with DeleteOnTermination: False.

Submission Requirements

1. Instance Details (aws ec2 describe-instances)

Field	Value
InstanceId	i-022a54888402a734a
PublicIpAddress	54.86.221.227
SecurityGroups	sg-03649244fb920dad3 (launch-wizard-1)

2. lsblk Output (after mounting data volume)

```
NAME                MAJ:MIN RM  SIZE RO  TYPE MOUNTPOINTS
nvme1n1             259:0    0   10G  0  disk /data
nvme0n1             259:1    0    8G  0  disk
└─nvme0n1p1         259:2    0    8G  0  part /
└─nvme0n1p127       259:3    0    1M  0  part
└─nvme0n1p128       259:4    0   10M  0  part /boot/efi
```

3. df -h /data Output

```
Filesystem      Size  Used Avail Use% Mounted on
/dev/nvme1n1    9.8G   24K   9.3G   1% /data
```

Summary

This lab demonstrated three approaches to EC2 instance creation, each with its own advantages:

- **AWS Console:** Point-and-click interface, ideal for learning and one-off tasks. Provides visual feedback and easy navigation.
- **AWS CLI:** Command-line automation, suitable for scripting and repeatable deployments. Enables infrastructure-as-code practices.
- **boto3 (Python):** Full programmatic control, best for complex workflows and integration with applications. Enables dynamic, conditional infrastructure management.

All three methods achieved the same result: a t3.micro instance running Amazon Linux 2023 with an 8 GB root volume and a 10 GB data volume in the us-east-1c availability zone.

Key Concepts Demonstrated

- **AMIs:** Templates defining instance software and configuration
- **Instance Types:** Hardware specifications (CPU, RAM, network)
- **EBS Storage:** Persistent block storage with configurable lifecycle
- **Volume Management:** Creating, attaching, formatting, and mounting volumes
- **SSH Key Pairs:** Secure authentication for instance access
- **Availability Zones:** Regional resource placement and compatibility requirements

Cleanup Confirmation

All instances have been terminated and orphaned volumes have been deleted to avoid ongoing charges.

Appendix: Python Script (create_ec2.py)

```
import boto3
import time
from pprint import pprint

# Create EC2 client and resource for us-east-1
ec2_client = boto3.client('ec2', region_name='us-east-1')
ec2_resource = boto3.resource('ec2', region_name='us-east-1')

AWS_AMI_ID = "ami-0157af9aea2eef346"
SSH_SG_ID = "sg-0ca338a4901703d3c"

def create_instance(ami_id, instance_name="boto3-created-instance"):
    """Create an EC2 instance with additional EBS volume"""
    print(f"\nCreating instance: {instance_name}")

    block_device_mappings = [
        {'DeviceName': '/dev/xvda',
         'Ebs': {'VolumeSize': 8, 'VolumeType': 'gp3',
                  'DeleteOnTermination': True}},
        {'DeviceName': '/dev/sdf',
         'Ebs': {'VolumeSize': 10, 'VolumeType': 'gp3',
                  'DeleteOnTermination': False}}]

    instances = ec2_resource.create_instances(...)
    instance.wait_until_running()
    return instance

# Full script available in ~/ds5220-lab2/create_ec2.py
```