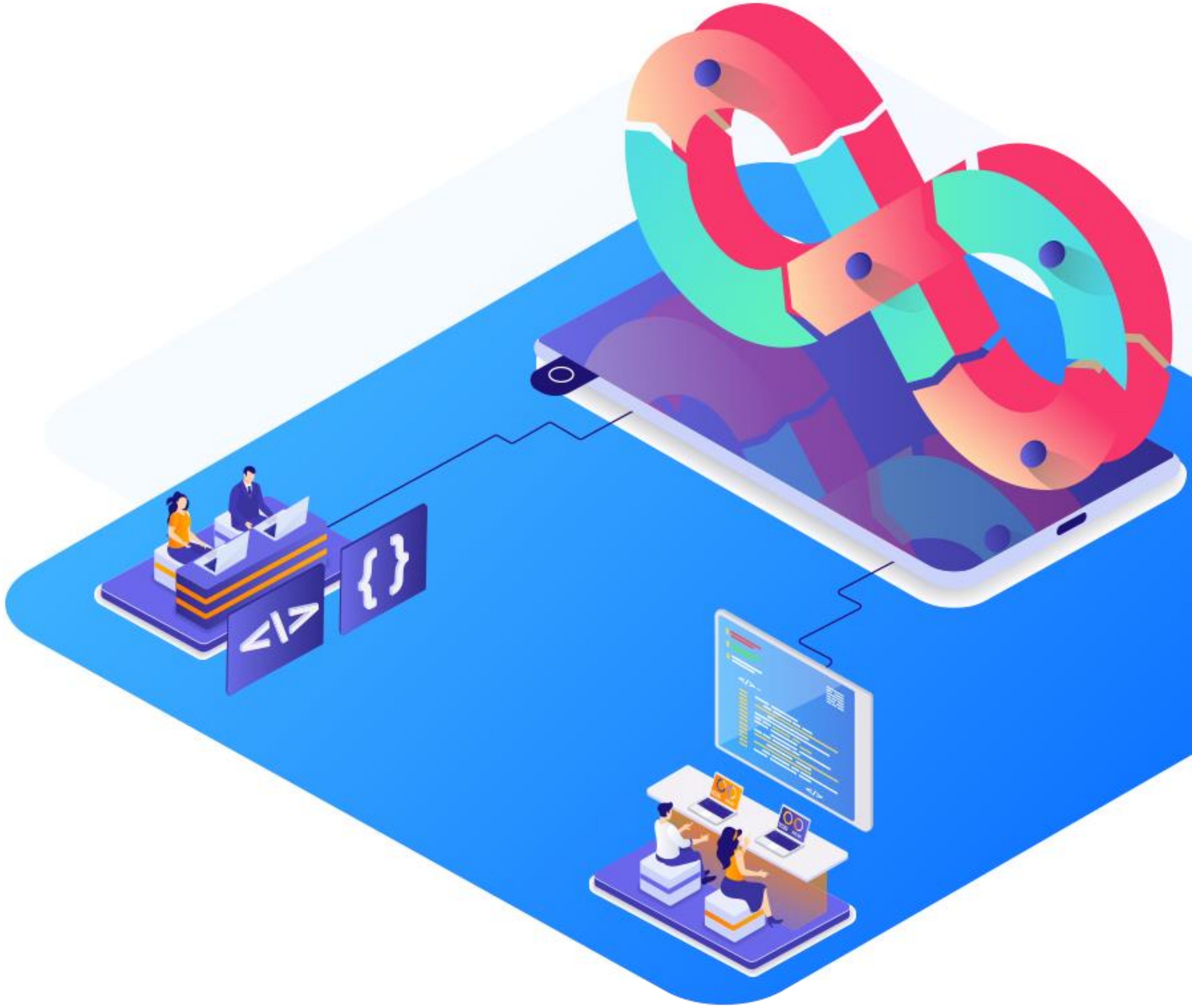
Configuration Management with Ansible and Terraform



Course-End Project

InfraPro

Objective

To automate the provisioning of infrastructure and configuring it with integration of Terraform and Ansible

Problem Statement and Motivation

Real-time scenario:

Royal Hotel is a globally leading chain of hotels. Recently, as part of scaling up operations, they aim to automate every operation in the hotel. For this, multiple applications are onboarded within all the hotel’s main server. To keep these applications up and running and to scale them appropriately, they need fully managed virtual machines on AWS.

They want to have an automated provisioned infrastructure through which they can create a new developer VM and manage some developer configurations on that server.

Industry Relevance

The following tools used in this project serve specific purposes within the industry:

1. **Ansible:** Ansible automates IT tasks, streamlining configuration management, application deployment, and orchestration. It uses simple, human-readable YAML files called playbooks.

2. **Terraform:** Terraform automates the provisioning and management of infrastructure using declarative configuration files. It supports multiple cloud providers and services, enabling consistent infrastructure deployment and scaling.

3. **Jenkins:** It automates the building, testing, and deployment of software projects, enhancing continuous integration and continuous delivery (CI/CD) pipelines.

4. **AWS:** It is a comprehensive cloud computing platform providing on-demand compute power, database storage, content delivery, and other functionalities to help businesses scale and grow.

Tasks

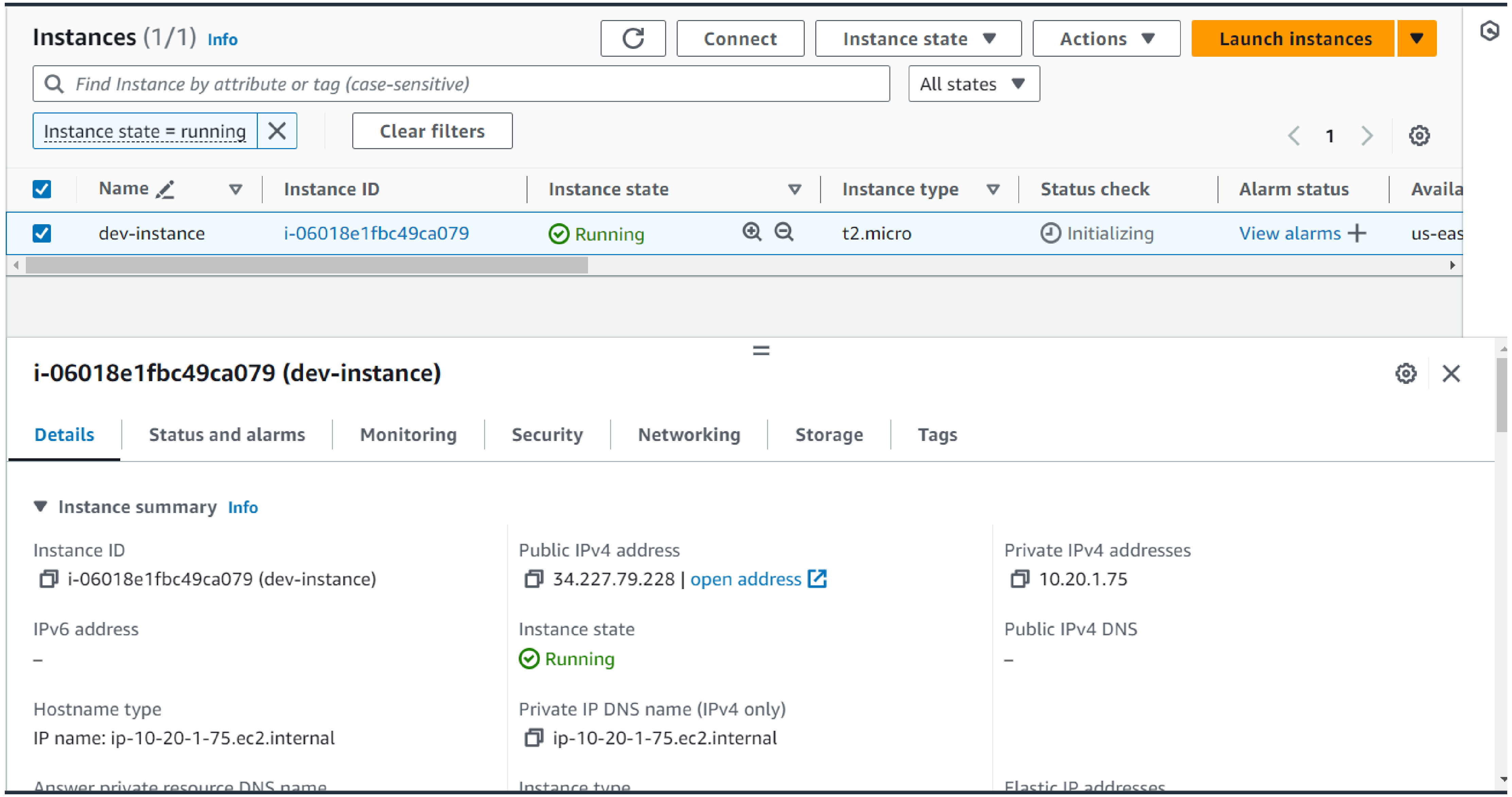
The following tasks outline the process of provisioning and infrastructure using Terraform and parameterised Jenkins job:

1. Configure Terraform with new ssh key which will be used as key pair for launching VMs
2. Configure AWS CLI with access key and secret key to establish connection remotely
3. Write Terraform script to provision and empty sandbox
4. Add various setting to the sandbox like VPC, security group, route table, subnets, and key pair
5. Create Ansible playbook which will be invoked by Terraform for configuration management operations
6. Executing Terraform script with the created keys to establish connection and configure the provisioned VM.

Project References

- **Task 1:** Lesson 02
- **Task 2:** Lesson 05
- **Task 3 and 6:** Lesson 08
- **Task 4:** Lesson 09
- **Task 5:** Lesson 09

Output Screenshots



Instances (1/1) Info

Find Instance by attribute or tag (case-sensitive)

All states ▼

Instance state = running ✕

☑	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availa
☑	dev-instance	i-06018e1fbc49ca079	✓ Running	t2.micro	Initializing	View alarms +	us-eas

i-06018e1fbc49ca079 (dev-instance)

▼ Instance summary Info

Instance ID
i-06018e1fbc49ca079 (dev-instance)

Public IPv4 address
34.227.79.228 | open address

Private IPv4 addresses
10.20.1.75

IPv6 address
–

Instance state
✓ Running

Public IPv4 DNS
–

Hostname type
IP name: ip-10-20-1-75.ec2.internal

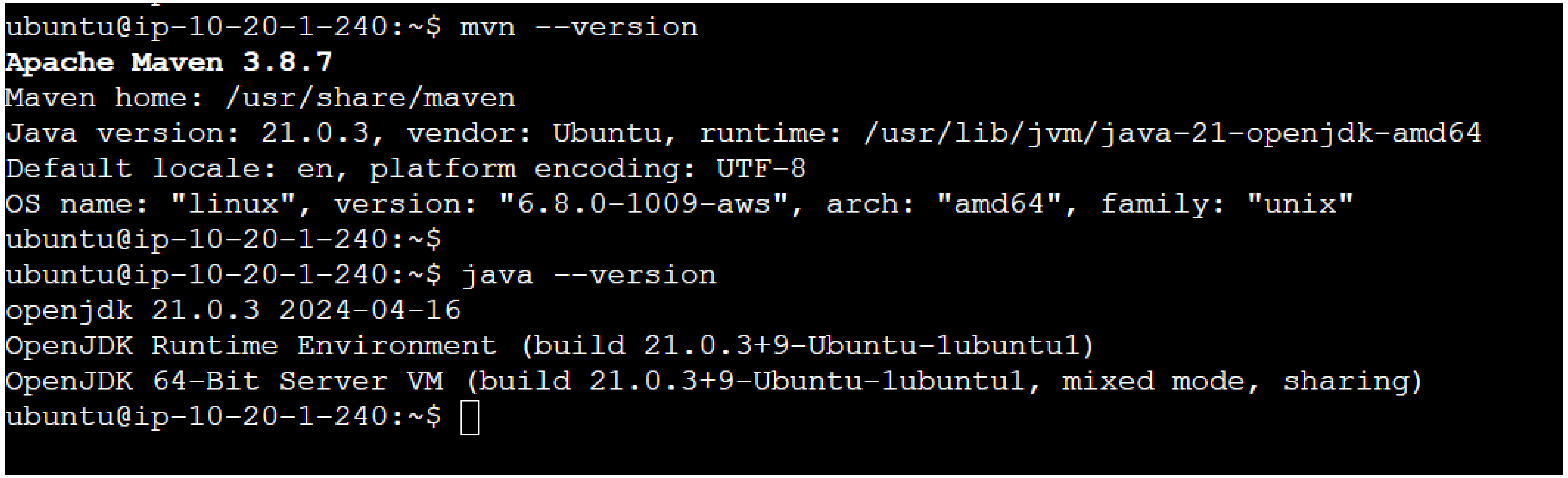
Private IP DNS name (IPv4 only)
ip-10-20-1-75.ec2.internal

Answer private resource DNS name

Instance type

Elastic IP addresses

Output Screenshots



```
ubuntu@ip-10-20-1-240:~$ mvn --version
Apache Maven 3.8.7
Maven home: /usr/share/maven
Java version: 21.0.3, vendor: Ubuntu, runtime: /usr/lib/jvm/java-21-openjdk-amd64
Default locale: en, platform encoding: UTF-8
OS name: "linux", version: "6.8.0-1009-aws", arch: "amd64", family: "unix"
ubuntu@ip-10-20-1-240:~$
ubuntu@ip-10-20-1-240:~$ java --version
openjdk 21.0.3 2024-04-16
OpenJDK Runtime Environment (build 21.0.3+9-Ubuntu-1ubuntu1)
OpenJDK 64-Bit Server VM (build 21.0.3+9-Ubuntu-1ubuntu1, mixed mode, sharing)
ubuntu@ip-10-20-1-240:~$
```

Thank you