

Laboratório Integrador –

Data Integration + CI/CD + IaC + Configuration Management

Visão Geral do Objetivo

Construir um pipeline completo que:

1. Provisiona a infraestrutura (S3 e EC2) com Terraform;
2. Configura a instância EC2 com Ansible;
3. Automatiza a execução via GitHub Actions;
4. Processa e carrega dados nas camadas Raw → Trusted → Client.

Etapa 0 – Criação do Ambiente Base (Control Node)

Objetivo

Padronizar o ambiente de execução do laboratório, criando uma **EC2 central** (Control Node) que conterà todas as ferramentas necessárias:

- **Terraform** (IaC)
- **Ansible** (Configuração)
- **Git** (Versionamento e CI/CD)
- **AWS CLI + Python + pip** (integração com AWS e pipeline de dados)

Essa instância será o **ponto de orquestração** do laboratório integrador.

0.1 Criar a instância EC2 de controle

1. Acesse o console AWS e vá em **EC2 → Launch Instance**
2. Configure:
 - **Name:** control-node-sprint5
 - **AMI:** Amazon Linux 2 (x86_64)
 - **Tipo:** t2.micro
 - **Key pair:** selecione uma existente ou crie uma nova
 - **Network:** default VPC
 - **Security Group:** libere portas 22 (SSH) e 80 (HTTP)
 - **Storage:** 20 GB (padrão é suficiente)
3. Clique em **Launch Instance**

Após iniciar, conecte-se via terminal:

```
ssh -i "sua-chave.pem" ec2-user@<IP_PUBLICO_DA_INSTANCIA>
```

0.2 Instalar ferramentas no Control Node

Tudo será instalado diretamente dentro dessa EC2.

Atualizar pacotes

```
sudo yum update -y
```

Instalar o Terraform

```
sudo yum install -y yum-utils  
sudo yum-config-manager --add-repo  
https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo  
sudo yum install terraform -y  
terraform -v
```

Instalar o Ansible

```
sudo amazon-linux-extras install ansible2 -y  
ansible --version
```

Instalar Git

Mesmo que o repositório esteja hospedado no **github.com**, o Git é necessário localmente para:

- Clonar o repositório da turma;
- Fazer commits e pushes para o GitHub (se permitido);
- Versionar código localmente.

```
sudo yum install git -y  
git --version
```

Instalar AWS CLI e Python

```
sudo yum install awscli python3 python3-pip -y  
aws --version  
python3 --version  
pip3 install boto3 pandas
```

0.3 Configurar credenciais AWS

O ambiente geralmente já fornece permissões temporárias via perfil **voclabs** ou **default**. Basta validar a sessão:

```
aws sts get-caller-identity
```

Se o retorno incluir seu AccountId e UserId, o ambiente está autenticado.

0.4 Estrutura inicial do projeto dentro da EC2

```
mkdir sprint5-lab && cd sprint5-lab
mkdir -p infra ansible data scripts .github/workflows
echo "Estrutura criada com sucesso!"
tree .
```

Resultado esperado:

```
sprint5-lab/
├─ infra/
├─ ansible/
├─ data/
├─ scripts/
└─ .github/workflows/
```

0.5 Configurar o GitHub no Control Node

Mesmo com o GitHub na nuvem, será necessário **autenticar localmente** para permitir o versionamento do repositório do projeto.

O PASSO 1 DEVE SER EXECUTADO APENAS SE VOCÊ NÃO TIVER NENHUM REPOSITÓRIO CONFIGURADO NO GITHUB. CASO JÁ TENHA ALGUM REPOSITÓRIO QUE QUEIRA UTILIZAR, IGNORE ESTE PASSO. BASTA CONFIRMAR QUE ELE CONTÉM A ESTRUTURA DE PASTAS E WORKFLOWS.

Passo 1 – Criar (ou confirmar) o repositório GitHub

1. Acesse <https://github.com>
2. Clique em **New Repository**
3. Configure:
 - **Repository name:** sprint5-lab-arqnuvem
 - **Visibility:** Public (ou Private, se preferir)
 - **Não** marque “Add README” (para evitar conflitos com o clone posterior)

4. Clique em **Create Repository**

Passo 2 – Configurar o Git no Control Node

Mesmo que o repositório esteja no GitHub.com, o **Git local** é necessário para:

- Clonar o repositório remoto;
- Criar commits das alterações locais;
- Enviar (push) o código para o GitHub Actions.

```
# Configure suas credenciais
git config --global user.name "Seu Nome"
git config --global user.email "seuemail@exemplo.com"
```

Verifique se o Git está configurado:

```
git config --list
```

Passo 3 – Clonar o repositório no Control Node

Dentro do seu Control Node:

```
cd ~
git clone https://github.com/<seu_usuario>/sprint5-lab-arqnuvem.git
cd sprint5-lab-arqnuvem
```

Passo 4 – Confirmar estrutura

Execute:

```
ls -R
```

Deve aparecer algo como:

```
.
├─ infra/
├─ ansible/
├─ data/
├─ scripts/
└─ .github/
    └─ workflows/
```

Etapa 2 – Estrutura de diretórios do projeto

Crie uma pasta de projeto e organize os arquivos da seguinte forma:

```
sprint5-lab/
├── infra/                                # Terraform (Infraestrutura)
│   ├── main.tf
│   ├── variables.tf
│   └── terraform.tfvars
├── ansible/                             # Ansible (Configuração)
│   ├── inventory
│   └── setup.yml
├── data/                                # Arquivos de dados
│   └── sales.csv
├── scripts/                             # Python scripts de tratamento
│   └── data_pipeline.py
└── .github/workflows/                   # CI/CD
    └── pipeline.yml
```

Etapa 3 – Terraform (Infraestrutura)

Objetivo: Criar um bucket S3 e uma instância EC2.

Como o ambiente Academy tem restrições de IAM, usaremos apenas recursos básicos (S3 + EC2) e as permissões já atribuídas ao usuário.

3.1 Arquivo /infra/main.tf

```
provider "aws" {
    region = var.aws_region
}

resource "aws_s3_bucket" "lab" {
    bucket = var.bucket_name
}

resource "aws_s3_bucket_versioning" "versioning" {
    bucket = aws_s3_bucket.lab.id
    versioning_configuration {
        status = "Enabled"
    }
}

resource "aws_instance" "app_server" {
    ami          = "ami-0c02fb55956c7d316"    # Amazon Linux 2
    region       = var.aws_region
    availability_zone = var.aws_availability_zone
    instance_type = "t2.micro"
    tags = {
        Name = "Sprint5-EC2"
    }
}

output "bucket_name" {
    value = aws_s3_bucket.lab.bucket
}

output "ec2_public_ip" {
    value = aws_instance.app_server.public_ip
}
```

3.2 Arquivo /infra/variables.tf

```
variable "aws_region" {  
    default = "us-east-1"  
}  
  
variable "bucket_name" {  
    default = "lab-sprint5-arqnuvem"  
}
```

3.3 Executar os comandos

```
cd infra  
terraform init  
terraform plan  
terraform apply -auto-approve
```

Ao final, anote o **nome do bucket** e o **IP público da EC2**.

Etapa 4 – Ansible (Configuração da EC2)

Objetivo: Instalar dependências básicas (Python, boto3, pandas) e preparar diretórios.

4.1 Arquivo /ansible/inventory

```
[ec2]  
  
<IP-PUBLICO-DA-EC2> ansible_user=ec2-user  
ansible_ssh_private_key_file=~/.ssh/<sua-chave>.pem
```

4.2 Arquivo /ansible/setup.yml

```
- hosts: ec2  
  become: yes  
  tasks:  
    - name: Instalar pacotes básicos  
      yum:  
        name: [python3, python3-pip]  
        state: present  
  
    - name: Instalar bibliotecas Python  
      pip:  
        name: [boto3, pandas]
```

```
- name: Criar diretórios de dados
  file:
    path: "/data/{{ item }}"
    state: directory
  loop:
    - raw
    - trusted
    - client
```

4.3 Executar o Ansible

```
cd ansible
ansible-playbook -i inventory setup.yml
```

Verifique se a EC2 possui os diretórios /data/raw, /data/trusted, /data/client.

Etapa 5 – Script de Integração de Dados

Objetivo: Realizar o fluxo **Raw** → **Trusted** → **Client** com Python e S3.

Arquivo /scripts/data_pipeline.py

```
import boto3, pandas as pd, os

bucket = "lab-sprint5-arqnuvem"
s3 = boto3.client("s3")
local = "data/sales.csv"

# RAW
s3.upload_file(local, bucket, "raw/sales.csv")

# TRUSTED
df = pd.read_csv(local).dropna()
df.columns = [c.strip().lower().replace(" ", "_") for c in df.columns]
df.to_csv("trusted_sales.csv", index=False)
s3.upload_file("trusted_sales.csv", bucket, "trusted/sales.csv")
```



```
# CLIENT
client = df[['order_id','amount']]
client.to_csv("client_sales.csv", index=False)
s3.upload_file("client_sales.csv", bucket, "client/sales.csv")

print("✅ Pipeline executado com sucesso!")
```

Etapas 6 – GitHub Actions (Automação CI/CD)

Objetivo: Orquestrar tudo via pipeline.

6.1 Secrets necessários

No repositório → **Settings** → **Secrets** → **Actions**

Adicione:

- AWS_ACCESS_KEY_ID
- AWS_SECRET_ACCESS_KEY
- AWS_REGION → us-east-1

6.2 Arquivo .github/workflows/pipeline.yml

```
name: Sprint5 Full Automation

on:
  push:
    branches: [ "main" ]

jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout
        uses: actions/checkout@v3

      - name: Configurar credenciais AWS
```

```
    uses: aws-actions/configure-aws-credentials@v4
    with:
      aws-access-key-id: ${ secrets.AWS_ACCESS_KEY_ID }
      aws-secret-access-key: ${ secrets.AWS_SECRET_ACCESS_KEY }
      aws-region: ${ secrets.AWS_REGION }

- name: Terraform Apply
  run: |
    cd infra
    terraform init
    terraform apply -auto-approve

- name: Configurar EC2 com Ansible
  run: |
    cd ansible
    ansible-playbook -i inventory setup.yml

- name: Executar pipeline de dados
  run: |
    python scripts/data_pipeline.py
```

Etapas 7 – Teste e Validação

1. Faça um **commit/push** no branch main.
2. Vá até o menu **Actions → Sprint5 Full Automation → Run workflow**.
3. Acompanhe as etapas no log:
 - Terraform cria recursos
 - Ansible configura EC2
 - Python executa o pipeline
4. Valide no console do **S3**:
 - raw/sales.csv
 - trusted/trusted_sales.csv
 - client/client_sales.csv