Automated AWS Infrastructure with Terraform & GitLab CI/CD 🚀

🧩 Project Overview

This project demonstrates how I used Terraform to provision AWS infrastructure and integrated it with GitLab CI/CD for complete automation. The objective was to create a scalable, reusable, and automated workflow to manage cloud resources effectively.

🛠️ Technologies Used

- **Terraform** for Infrastructure as Code
- **GitLab CI/CD** for automation
- **AWS** (EC2, VPC, Subnet, Security Group)
- **GitLab Runner** for executing pipeline jobs
- **S3 & DynamoDB** for remote backend and state locking

⚙️ Infrastructure Components

Resource	Description
VPC	CIDR: `10.0.0.0/16`
Subnet	Public Subnet: `10.0.1.0/24`
Security Group	Inbound: Port 22 for SSH

EC2 Instance	AMI: `ami-06c8f2ec674c67112`, Type: `t2.micro`

. 📋 GitLab CI/CD Pipeline

Stages Defined:

```
stages:

  - validate

  - plan

  - apply

  - destroy
```

Breakdown:

- **Validate:** Runs `terraform validate` to ensure syntax is correct.
- **Plan:** Generates a plan file (`terraform plan -out=planfile`) and stores it as a pipeline artifact.
- **Apply:** Applies the plan manually with `terraform apply "planfile"`.
- **Destroy:** Manually destroys the infrastructure (`terraform destroy -auto-approve`).

🔐 Remote State Management

Terraform backend is configured with:

- **S3 Bucket** to store the `terraform.tfstate` file
- **DynamoDB Table (`backendterraform`)** to manage state locks
    - Partition key: `LockID` (String)

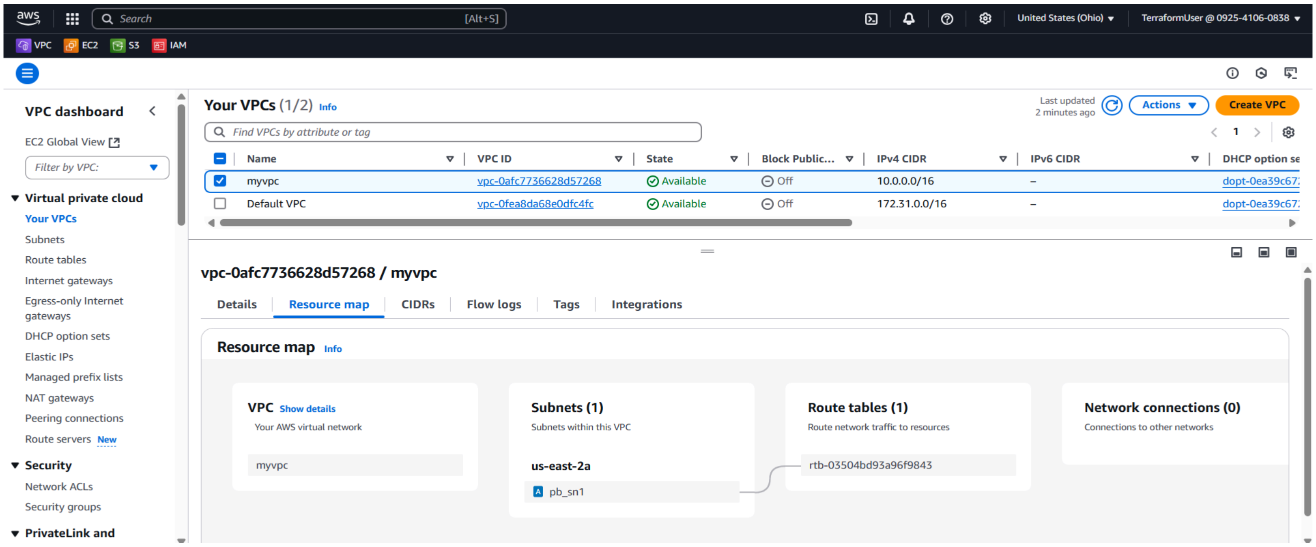
This prevents state corruption by ensuring only one process can modify the state at a time.

✅ Results & Benefits

- ⏱️ **80% reduction** in provisioning time
- 🔄 Fully automated, consistent deployments
- 💾 Version-controlled infrastructure with modular codebase
- 🔐 Secure state management and lock handling

📸 Screenshots

- AWS EC2 & VPC Dashboard



Your VPCs (1/2)

Find VPCs by attribute or tag

	Name	VPC ID	State	Block Public...	IPv4 CIDR	IPv6 CIDR	DHCP option se
☑	myvpc	vpc-0afc7736628d57268	✓ Available	⊖ Off	10.0.0.0/16	–	dopt-0ea39c67
□	Default VPC	vpc-0fea8da68e0dfc4fc	✓ Available	⊖ Off	172.31.0.0/16	–	dopt-0ea39c67

vpc-0afc7736628d57268 / myvpc

Resource map

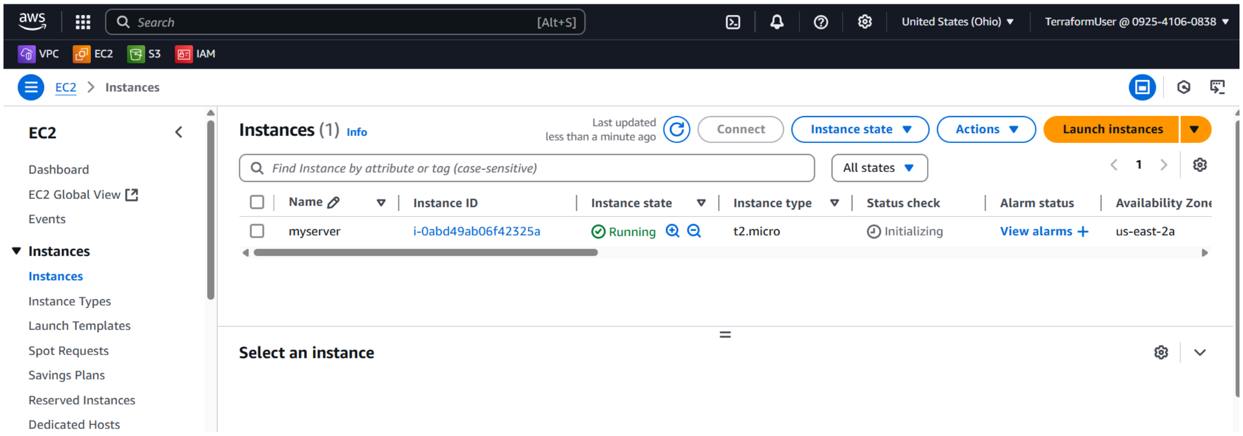
VPC — Your AWS virtual network: myvpc

Subnets (1) — Subnets within this VPC: us-east-2a, pb_sn1

Route tables (1) — Route network traffic to resources: rtb-03504bd93a96f9843

Network connections (0) — Connections to other networks

-

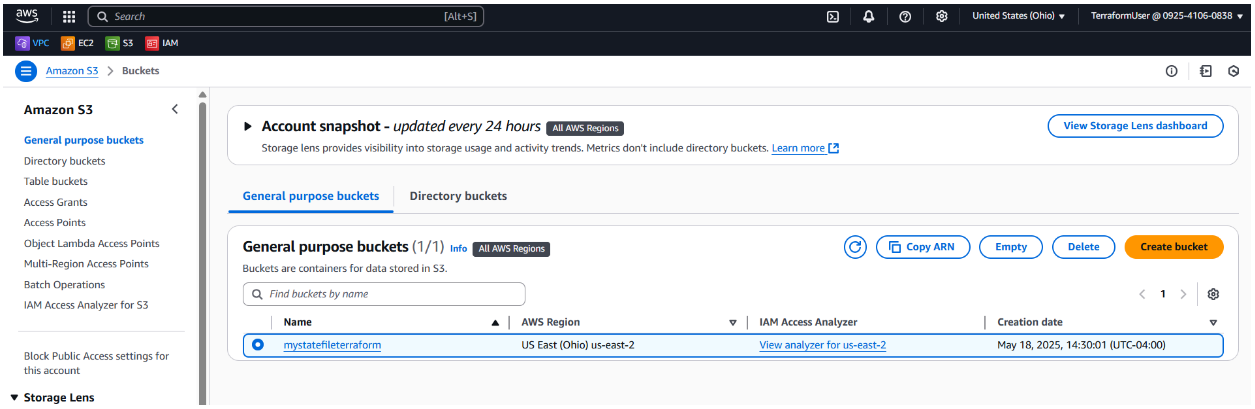


Instances (1)

Find Instance by attribute or tag (case-sensitive)

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
□	myserver	i-0abd49ab06f42325a	✓ Running	t2.micro	Initializing	View alarms +	us-east-2a

Select an instance



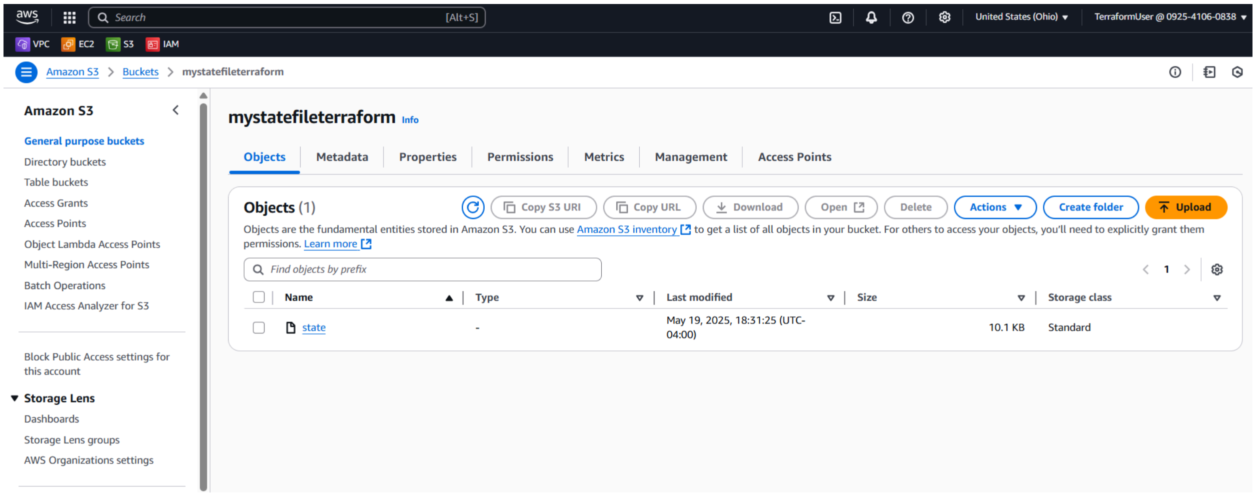
▶ **Account snapshot -** *updated every 24 hours* All AWS Regions

Storage lens provides visibility into storage usage and activity trends. Metrics don't include directory buckets. Learn more

General purpose buckets (1/1) Info All AWS Regions

Buckets are containers for data stored in S3.

	Name	AWS Region	IAM Access Analyzer	Creation date
●	mystatefileterraform	US East (Ohio) us-east-2	View analyzer for us-east-2	May 18, 2025, 14:30:01 (UTC-04:00)

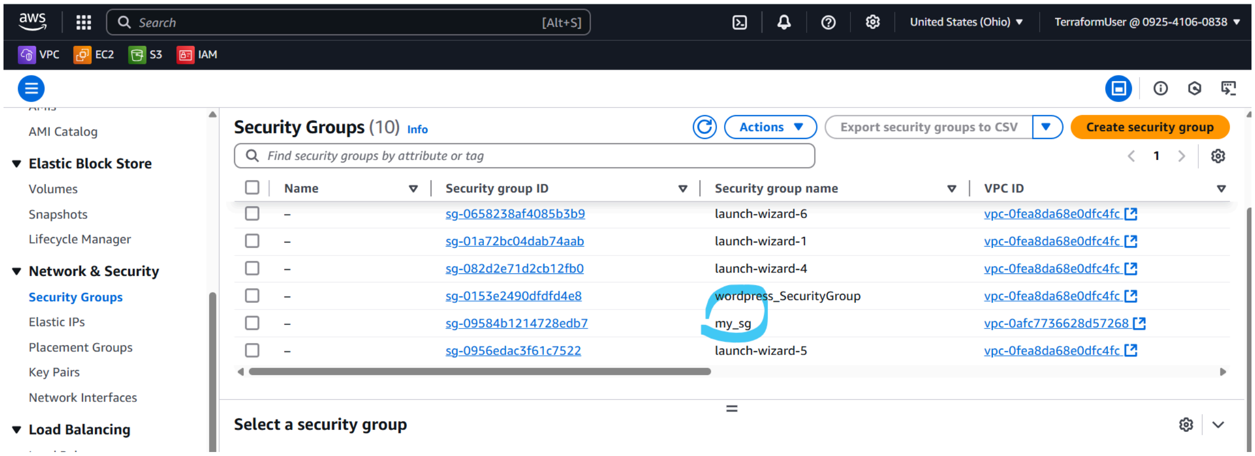


mystatefileterraform Info

Objects (1)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

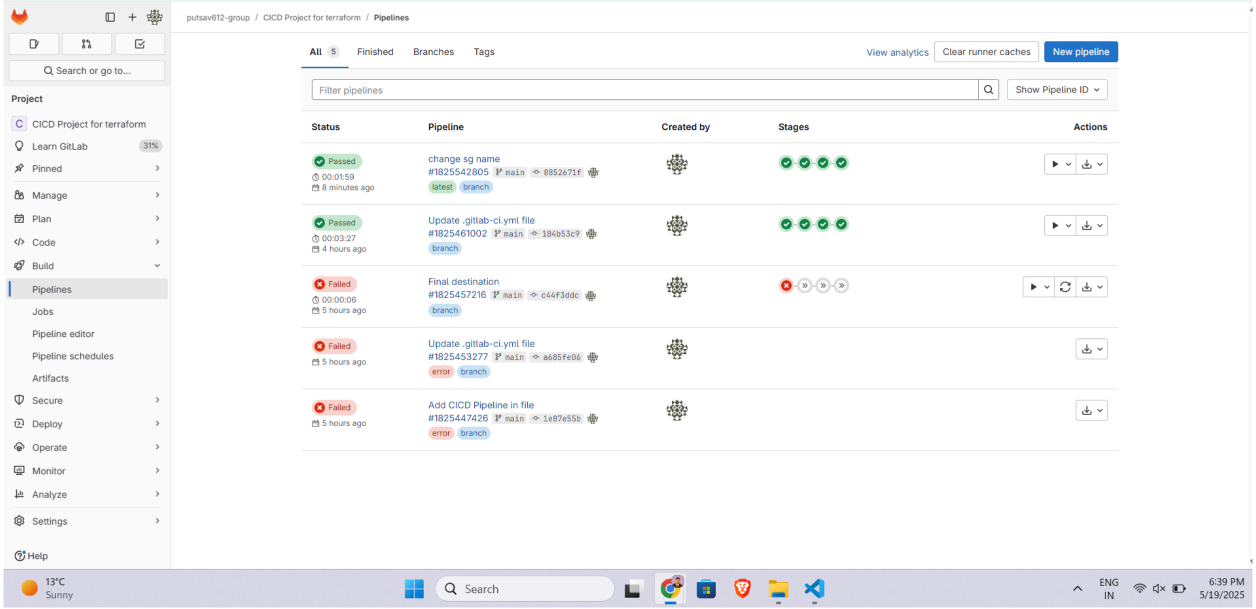
□	Name	Type	Last modified	Size	Storage class
□	state	-	May 19, 2025, 18:31:25 (UTC-04:00)	10.1 KB	Standard



Security Groups (10) Info

□	Name	Security group ID	Security group name	VPC ID
□	–	sg-0658238af4085b3b9	launch-wizard-6	vpc-0fea8da68e0dfc4fc
□	–	sg-01a72bc04dab74aab	launch-wizard-1	vpc-0fea8da68e0dfc4fc
□	–	sg-082d2e71d2cb12fb0	launch-wizard-4	vpc-0fea8da68e0dfc4fc
□	–	sg-0153e2490dfdfd4e8	wordpress_SecurityGroup	vpc-0fea8da68e0dfc4fc
□	–	sg-09584b1214728edb7	my_sg	vpc-0afc7736628d57268
□	–	sg-0956edac3f61c7522	launch-wizard-5	vpc-0fea8da68e0dfc4fc

Select a security group



putsav612-group / CICD Project for terraform / Pipelines

Status	Pipeline	Created by	Stages	Actions
Passed 00:01:59 8 minutes ago	change sg name #1825542805 main 8852671f latest branch		✓ ✓ ✓ ✓	
Passed 00:03:27 4 hours ago	Update .gitlab-ci.yml file #1825461002 main 184b53c9 branch		✓ ✓ ✓ ✓	
Failed 00:00:06 5 hours ago	Final destination #1825457216 main c44f3ddc branch		✕ » » »	
Failed 5 hours ago	Update .gitlab-ci.yml file #1825453277 main a685fe06 error branch			
Failed 5 hours ago	Add CICD Pipeline in file #1825447426 main 1e87e55b error branch			

- ## 🧠 Lessons Lear
- Modular Terraform design for reusability
- Best practices in CI/CD automation using GitLab
- Remote state handling with S3 and DynamoDB
- Artifact handling and manual approval stages

📍 Conclusion

This project showcases my practical knowledge in DevOps, infrastructure provisioning, and automation pipelines. It is scalable, production-ready, and built using real-world DevOps principles.