Course 3: Multi-Environment IaC & GitOps Pipeline

1. Introduction

In the previous course, you:

- Set up a **GitHub repository** for Terraform code.
- Configured your **cloud provider CLI** and deployed a **VPC + subnet** using Terraform.
- Used a **remote backend** for Terraform state.

Now, you will extend your project to support **multiple environments (dev + prd)**, implement a **GitHub Actions pipeline**, and manage **IAM users/permissions** so the whole team (and me 😁) can access resources securely.

⚠️ Infrastructure must always be:

- **Reproducible** → `terraform apply`
- **Disposable** → `terraform destroy`
- **Safe** → credentials must be managed with care, never hardcoded or shared in plaintext

---

2. Prerequisites

- **Terraform CLI & cloud provider CLI** already installed.
- **One GitHub repository per group** (already created).
- **Two cloud projects per group**:
    - `teamX-dev` (development)
    - `teamX-prd` (production)

---

3. Multi-Environment Setup in Terraform

3.1 Separate `tfvars` Files

Create one variables file per environment:

`dev.tfvars`:

```
project_id = "teamX-dev"
region     = "us-central1"
vpc_name   = "teamX-dev-vpc"
cidr_block = "10.0.0.0/16"
```

`prd.tfvars`:

```
project_id = "teamX-prd"
region     = "europe-west1"
vpc_name   = "teamX-prd-vpc"
cidr_block = "10.1.0.0/16"
```

3.2 Separate Remote Backends

Each environment should have its own state bucket + key.
Example (GCP):

`dev.config`:

```
bucket  = "teamX-dev-tfstate"
prefix  = "vpc"
```

`prd.config`:

```
bucket  = "teamX-prd-tfstate"
prefix  = "vpc"
```

Initialize each environment with:

```
terraform init -backend-config=dev.config
terraform init -backend-config=prd.config
```

3.3 Testing Locally

- Dev deploy:

  ```
  terraform apply -var-file=dev.tfvars
  ```

- Prd deploy:

  ```
  terraform apply -var-file=prd.tfvars
  ```

- Destroy all:

  ```
  terraform destroy -var-file=dev.tfvars
  terraform destroy -var-file=prd.tfvars
  ```

✅ After destroy, **no infra should remain** (IAM, APIs, VPCs, etc.).
✅ After apply, **everything must redeploy automatically** without manual steps (except CI/CD credentials).

---

4. IAM & Permissions Management

4.1 Why IAM Matters

IAM = **Identity and Access Management**:

- Defines *who* can do *what* on *which* resources.
- Misconfigured IAM = **security breach**.
- Over-permissive IAM = **attack surface**.

4.3 Add members on Github

Give permissions to the members on your Github Repository using terraform:

- 4 Students
- Me (@Kloox)

⚠️ You may not want to remove these permissions when destroying your infra, do a separate terraform stack to manage the permissions

4.2 Add members on your Cloud provider

Give permissions to members to your Cloud provider using terraform:

- The 4 students
- Me:
    - **GCP**: add `jeremie@jjaouen.com` to your project then send your projects link via Teams.
    - **AWS**: create a user for me, export credentials, encrypt with my GPG key (see appendices), and send via Teams.

⚠️ You may not want to remove these permissions when destroying your infra, do a separate terraform stack to manage the permissions

⚠️ Handle credentials **securely**: never commit them to Git.

4.3 Terraform Example (GCP)

```
resource "google_project_iam_member" "jeremie" {
  project = var.project_id
  role    = "roles/editor" # I will never edit your resources, so
which role should I have ?
  member  = "user:jeremie@jjaouen.com"
}
```

---

5. Quick Tour: IAM & Billing UIs

Even though we automate with Terraform, you must learn to **read the console**.

- **GCP Console**:
    - IAM page: shows all members and their roles.
    - Billing page: shows costs per project and per service.
- **AWS Console**:
    - IAM dashboard: users, roles, policies.
    - Billing dashboard: cost breakdown and forecast.

👉 You need to be familiar with your cloud provider console and be able to read your IAM & Billing dashboards. You will also most likely need to check that you've deployed what you intended by checking the resources directly on the console.

---

6. GitHub Actions & GitFlow

6.1 Why GitFlow Matters

Your **Git branching strategy (GitFlow)** is critical because it defines **how you collaborate** and how fast you can deliver changes.

You are free to implement your **own GitFlow**, but it must make sense with the DevOps principles we already discussed:

- **Fast-to-fail** → test early with short feedback loops.
- **Immutability** → avoid mutating infrastructure; redeploy instead.
- **Time-to-market** → keep merges and releases lightweight.
- **Collaborative work** → use feature branches, pull requests, and reviews.
- **DevOps cycle** → plan, code, build, test, release, deploy, operate, monitor.

Think twice about which jobs should be run in which branches, context, environment, etc. You should definitely look about existing GitFlow on the internet before definitively committing into a GitFlow.

👉 Your GitFlow is as important as your Terraform code — it defines **how you work as a team** on this project.

6.2 CI/CD Pipeline Requirements

Your pipeline must:

- Validate your terraform files: syntax must be valid and correctly indented (terraform fmt, terraform validate)

- Run `terraform plan` for both environments (dev + prd).

- Versioning & release (e.g., tagging: `v1.0.0`); then:
  - Apply changes in `dev`.
  - Apply changes in `prd`.

- Provide **manual destroy workflows** for dev and prd.

- Use the Github Actions environment feature & keyword.

- In your terraform, use variables when a value may be different depending on the environment.

- Don’t have any secrets/credentials hardcoded/pushed into the repository.

6.3 Example: GitHub Actions Workflow

`.github/workflows/terraform.yml`:

```yaml
name: Terraform CI/CD

on:
  push:
    branches:
      - main
    tags:
      - 'v*'
  pull_request:
    branches:
      - main

jobs:
  terraform:
    runs-on: ubuntu-latest

    strategy:
      matrix:
        env: [dev, prd]

    steps:
      - uses: actions/checkout@v3

      - name: Setup Terraform
        uses: hashicorp/setup-terraform@v2

      - name: Init Terraform
        run: terraform init -backend-config=backend-${{ matrix.env }}.hcl

      - name: Terraform Plan
        run: terraform plan -var-file=${{ matrix.env }}.tfvars

      - name: Terraform Apply (on tag only)
        if: startsWith(github.ref, 'refs/tags/')
        run: terraform apply -auto-approve -var-file=${{ matrix.env }}.tfvars
```

6.4 Example: Manual Destroy Workflow

`.github/workflows/terraform-destroy.yml`:

```
name: Terraform Destroy

on:
  workflow_dispatch:
    inputs:
      env:
        description: "Environment to destroy (dev or prd)"
        required: true
        default: "dev"

jobs:
  destroy:
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v3

      - name: Setup Terraform
        uses: hashicorp/setup-terraform@v2

      - name: Init Terraform
        run: terraform init -backend-config=backend-${{
github.event.inputs.env }}.hcl

      - name: Terraform Destroy
        run: terraform destroy -auto-approve -var-file=${{
github.event.inputs.env }}.tfvars
```

👉 These examples are purposely naive, you will need to extend/rework them. Refer to the CICD requirements and as always: RTFM.

6.5 About using Github Actions plugin/communautary Actions

You are allowed to use Github Actions plugin or Actions from the community to do your pipeline, however:

- It must be widely used in the community (No 2 stars repositories, 4 downloads/week, etc).
- It mustn’t cause security issues.
- You need to understand what it does and be able to justify the benefits in your pipelines.

---

7. Key Takeaways

- You now manage **two environments** (dev + prd)
- Your Terraform stack includes **VPC, subnet, IAM users/permissions** and all required configuration.
- Infrastructure is **fully disposable** with both local and CI/CD destroy options.
- You learned to read **IAM & Billing dashboards**.
- You've set up simple **GitHub Actions** + **GitFlow** = automated plans, tests, deploys and manual destroys for both environments.
- **Your GitFlow strategy matters** → it must reflect DevOps values like fast feedback, immutability, and collaboration.

---

8. Go further

- Look on the internet for Terraform, Gcloud, Aws CLI **cheat sheets** if you are not familiar with them yet.
- Setup git hooks to automatically format and validate your terraform code when pushing.
- Add an **infracost** job to your pipeline to evaluate the cost impact of your changes.

👉 **Next Course Preview**:
We'll design the **final project architecture**:

- Cloud-managed **Kubernetes cluster**.
- **Custom GitHub Actions runners** inside K8s.
- **Workload Identity** (short-lived credentials for CI/CD).
- A **managed SQL database**.
- A **scalable app** deployed in Kubernetes with full CRUD operations on the database.

9. Appendices

- My GPG public key:

-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGjVjBYBEACrUUP5XMVE6TqLPKjSWChtYpETzDT+wAojujnFG2LMPggJC+dL
TJrUnu2JOV6uejSbcPbQXCCXcPKOIPi0LAtAW3NBCjRlrlZ81YpsiIo9lhJekjP7
V++LNcwduoy35sl1icFnpyd90imWEJ0Wk+AxxALyxTBiaMvoiO36vuZ8M5QXQkB4
6xBJ7xChwvT2ysruwBL1iw6UUCTtVRTFb3OLVIyK7Ll4HAF2s88w2MC1vXKHrPpR
qlYQInPpW+qXCICLhsqIIVjRAprbqfqtsOrEMQAQ8gUrjvDHXRbQILNRhe9ghG+m
UADfRj7gPmep+m19WQDCPGFNjwUH7KdU+PiJfG8KnFzgZ7mfmV4iTSw3TiCHyh2h
XXka3wEGlo49wR3gWZskMANg/Lpbslw4zF50fl4uGu2t/ERuyf6y6ZWxKY5INngw
foPrbCzPOHvSCB3TlzCQeLKJIuqtVgvo3nh6Ce4u4XeOFAEaPfW01KXl+K/uYFgF
RUovPyzqAefMvdeifrOr+A7OJYvrw9L1MoZaFGHFHwBHsKLVrxhviTvu1BYEw9Qo
JXshXLT7ziks/UIUQSInJWEhUCcxR+YKdNj0oqshDTL8fzoSThRRwZAPY9ciq8Y9
SXs97YNGB1f/DUVNmQs3Uw/vtB5PuchLHpzLDGYGN4FcAl5QtClM8KjyIwARAQAB
tEdKw6lyw6ltaWUgSkFPVUVOIChHUEcga2V5IGZvciBFcGl0ZWNoIElBQyBtb2R1
bGUpIDxqZXJlbWllQGpqYW91ZW4uY29tPokCTgQTAQoAOBYhBJhLBQgHtKya5aVJ
FEKylOmKdzW4BQJo1YwWAhsDBQsJCAcCBhUKCQgLAgQWAgMBAh4BAheAAAoJEEKy
lOmKdzW4eUkP/00kJjBDLlNK++VU1tLg9K+COpTg4yeAKHgjJDo252GdACTvCC3K
xZAh8uFXQLO4WefawgXydfqXSm+T2s4i6GD3CAq5+P2AtxnPGALU+upjz2Xrxvda
csX2VIe7LVlyO+inVO1Xw0X6joMXHO5wj+k9Lj/+r+jDvXin8itmPZu9bQj6CUyb
ztmW58L/vUzxg2YuoW92/q9yF365Av9R32VspGn1e9oF7VL+W3j4lMuMXOKTHQK4
3lttLnmaVsu+V5swU3FVCLyppAAtO/q4KD4vvvoEd0BmDZu1dyiYH9/SXnzpqfMy
m+FVH7ZyvpvUSpljX9AR+8roZ+IQlai66OBFMp40EIVgKzAoSyyvpKiyJIPm0UuK
Ey0R7ZYgUS1a0kq1WD8eZLKuardbedushbekinE7jCh/sQ2AlbXwQ6rR4Q6IgoKo
+FALD8vRB/tuDikPlgifJSGJ6eGO7twm8j0MbyPfBFxUwCIyTynN/bhl7QphOsAJ
4H+lOH9PsdYwD73i1QT2+0HPvIMNgZXuG6bipolxMBH5/A2V8tjpztoIw+jMdjnj
rQQnt0eB5KUItHBwrOHETAuJQboxlQ7CFPP1IY29/OjyJTye5O5Hf4llk+hHwNGI
FDkBF8HjhmDDAst182rJoYiT+HTx6x/2WeUmVGiAN2mFT7yFl1vEXonRuQINBGjV
jBYBEACv1rNYDmtbD7EKmq/0SoSyGkf/MoDBD48BvnvCEuIbwEJA89dKqlfUps+c
GR9un4UMh9eD3Ey3IGBuppoFH08e9iXgT96bFmhO7WqgHLedQt3RcDVlM1HOEkoA
hHWNFNgwUXU5sTyOgg/UAQhFkH0yq1VLCKiaf98uCzGRYwXLrge2a9+Elq321HQq
mqxNbGnIFymyW6T3cYEhcqmGH0pdAWNLKliviwa1VaHyzl1vitK+qN25DIjP2JEe
4D1KvU6E/8cTojsWCs3gUxGofNgpE3FVDZcXKELSfuPSbDa8gslNJC0WC189okZ3
d+M0/wu348H3N7fAV69Gryi/pHaCWVKwm0bRrpDzf888wPkhWw6U+Z26enB/Zic1
3uTd7kmxVQDiNbc9mTGAZnW88xvZ6hmWnR/HbU6LQ3JnJwcQbe08aaNcDzFUhX5O
Amejz86DOTX4NVdnO+jrTcKWs14CtdihpE93GGz/cOEDjqMVKDY9m+W7Ag6sS569
cY2vPdVAF94ArfwFJxsexGFn3WOpqdBZWXT3vx38ItdbviEzKl1FJPEafLqpcMPc
MCONJFqxcg27LST4lLZH4sxyr+z0tn7TfaImFrVds82GM0FHHvgickL9q3LYVs5+
MJLiJLdKuvTjpq1dEQTM78xhCESESDqKUqbyAzgO/bSQB2C6/wARAQABiQI2BBgB
CgAgFiEEmEsFCAe0rJrlpUkUQrKU6Yp3NbgFAmjVjBYCGwwACgkQQrKU6Yp3Nbij
8Q/+OySSeaC/qlVz/p6KNaZXW1iCsZ1CLFtGunoIGoZLkbZx809bkZoUxE4B16mt
5YrQXc5LW92fejPupkNohWETpt8Kx5bnCKDW8RyYOOs0KmH+kZt3cVp02kod3ckN
lGLuB4qdVJ4JH30Pabb4qwpcfMr+/s/lHt4R7XBADo7pt8CYfXcg5vztTUgcEYOh
01T7sXtAYv9Wl+XWlhoK68bRbh5BFKMmifc+4P0ZQzi+JHdybVmQNMo58oWIZEb5
hf82cW5Aitb9vtd/LQyM3NyLVebbuNP2Sj/5My/xYIviyhHfiNnOCUOUWAmNTpNV
AYcl0LbmYEURJYEopBzLRQl1XQ3cozpBzmHKdjvEUXXGAztshCypmZ5zVGg1Y0pD
ReUoUr1YNZpF2mmmo9OhXw5fLjiuAyqH6h7FqepmRa/oKreFkFtPaGPQjRXpnpeJ
R/0tdc4jThrZOH1fl6xlQzOF+7Uv4YTAmkdkdzCHWrXTWNAYflCEMfOsIfjpaGZx
5uGfYFEWtjM4SVw6AW/2phYmLqNHSkNvQ1m35gCCxBZ7SLyGGXWV0ajGWutzlaqZ
0mEiebhzs2QEVWaPqUzthOEi3/+nCPD3cTcAunT3aRacDOiRlRR8Bwxp4mb3cxtj
h4VIu6KuOR3Eg1UROfwbedSkTqFlnOsNU4lNrdO8hSdr1oA=
=dciJ
-----END PGP PUBLIC KEY BLOCK-----