Security in IaC

DevSecOps Workshop

João Oliveira - 2022184283 - uc2022184283@student.uc.pt
Cybersecurity Laboratory - Master in Informatics Engineering
Faculty of Sciences and Technologies - University of Coimbra

December 2025

1. Introduction

This workshop extends the DevSecOps foundations from Laboratory #3 by focusing on Infrastructure as Code (IaC) security. While Lab #3 covered application security in CI/CD pipelines, this workshop addresses the critical challenge of securing infrastructure definitions before deployment.

Participants will implement specialized security scanning tools (Checkov, Trivy, Gitleaks) in GitHub Actions pipelines to detect misconfigurations, exposed credentials, and policy violations in Terraform code.

2. Objectives

This workshop has the following core objectives:

1. Understand the DevOps philosophy, secure pipeline and threat modeling concepts
2. Identify risks in a CI/CD pipeline on infrastructure managed with IaC tools
3. Integrate, configure and maintain multiple security tools and procedures.

3. Learning Outcomes

The learning outcomes are:

- Shift-left security
- Security as Code
- Continuous Security
- Defense in depth

4. Requirements

The following tools are required:

1. Git
2. GitHub account
3. Code Editor

Additionally, if you want to deploy the project or validate locally, you also need:

1. A Google Cloud Platform (GCP) account with a free tier or an open billing account

2. GCP command line interface (CLI)
3. Terraform, as well as Tfsec, Checkov, Trivy and Gitleaks

5. Activities

This workshop will be performed in phases:

- Phase 1 - Environment Setup
- Phase 2 - Static IaC Code Analysis Setup
- Phase 3 - Issue Lifecycle and Resolution
- Phase 4 - Additional Tooling and Security Gates

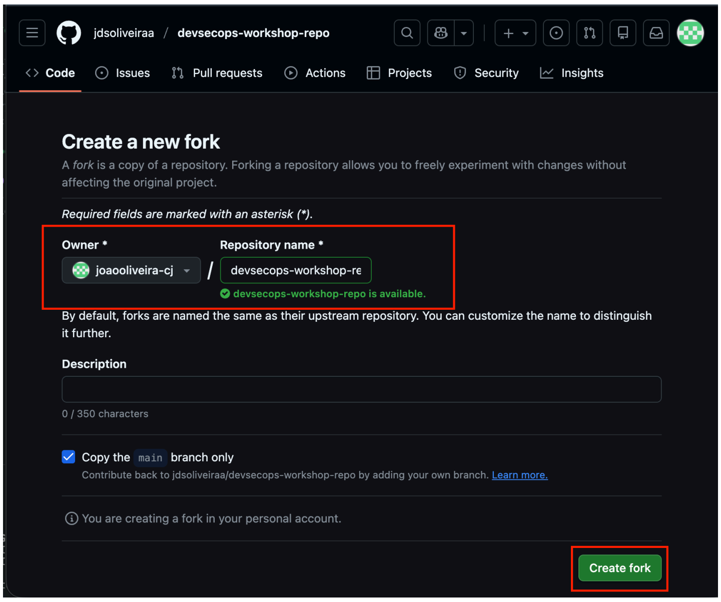
5.1. Phase 1 - Activities

This phase has the following goals:

- Setup your local environment - tools and repo
- Understand the starting repository purpose

Steps:

1. Make sure you have git, VSCode and a GitHub free account
   a. Install tools locally (macOS)**:** brew install terraform checkov trivy gitleaks
2. Create a fork of the starting repo - https://github.com/jdsoliveiraa/devsecops-workshop-repo/fork



Create a new fork

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.

Required fields are marked with an asterisk (*).

Owner * joaooliveira-cj / Repository name * devsecops-workshop-re

devsecops-workshop-repo is available.

By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

Description

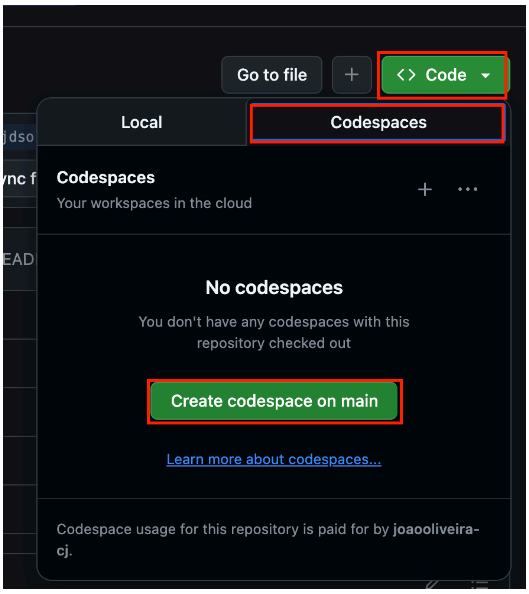
0 / 350 characters

☑ Copy the `main` branch only
Contribute back to jdsoliveiraa/devsecops-workshop-repo by adding your own branch. Learn more.

You are creating a fork in your personal account.

Fig. 1 - Creating a fork

3. Clone your repository locally or use GitHub Codespaces by clicking in "Code" -> "Codespaces" tab -> "Create codespace on main"



No codespaces

You don't have any codespaces with this repository checked out

Codespace usage for this repository is paid for by **joaooliveira-cj**.

Fig. 2 - Creating a codespace

4. Explore the initial repo files and structure:
   a. In the ‘terraform’ folder, you have all the resource definitions using HCL.
   b. In ‘patches’ folder, you have a patch file that edits the terraform code with secure definitions

5.2. Phase 2 - Activities

This phase has the following goals:

- Double check GH Actions CICD Pipeline configuration
- Verify Trivy configuration and validate its execution

Steps:

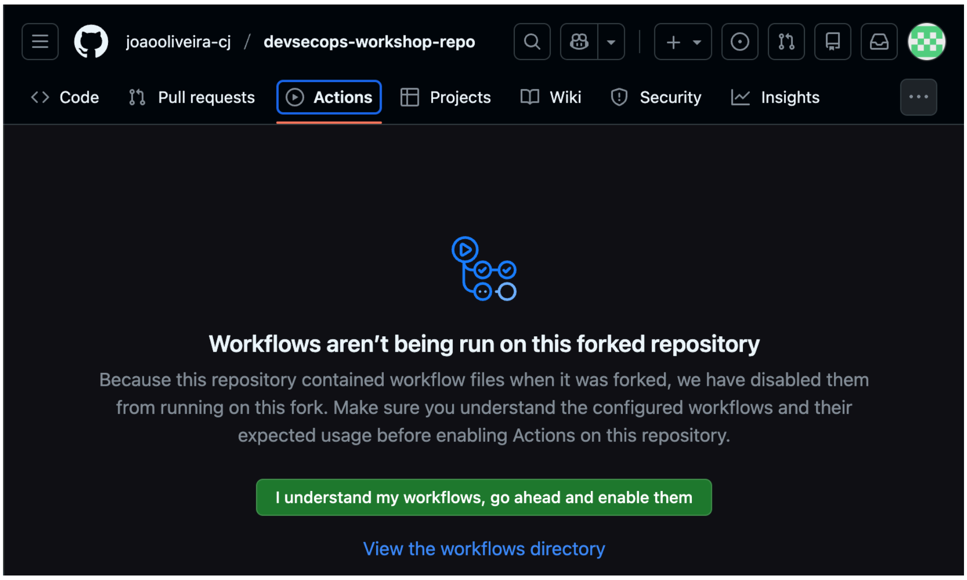
1. Trivy now includes Terraform code analysis, as the tool “tfsec” is now merging into Trivy[1]. For Trivy to analyze infrastructure code, the configuration mode must be specified in GH Actions pipeline code:

```
# Vulnerability scanning using Trivy but for config files
- name: Run Trivy
  uses: aquasecurity/trivy-action@master
  with:
    scan-type: 'config'
```

[1] Tfsec used to be the primary static security analysis for IaC until it was merged with Trivy, after “Aqua Security”’s acquisition of Tfsec.

```
    scan-ref: 'terraform'
    severity: 'CRITICAL,HIGH,MEDIUM,LOW'
    exit-code: '0'
    format: 'sarif'
    output: 'trivy-results.sarif'

- name: Upload Trivy results to GitHub Security
  uses: github/codeql-action/upload-sarif@v4
  if: always()
  with:
    sarif_file: 'trivy-results.sarif'
```

2. This is enabled in the pipeline code. After creating the fork you must explicitly enable GH Actions pipeline executions:


joaooliveira-cj / devsecops-workshop-repo
Code
Pull requests
Actions
Projects
Wiki
Security
Insights
I understand my workflows, go ahead and enable them
View the workflows directory

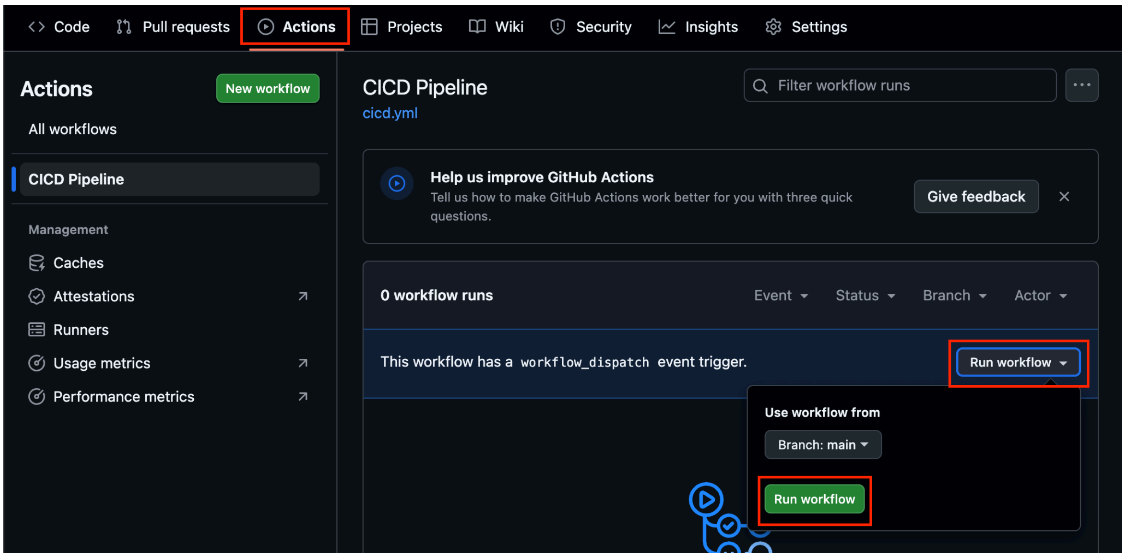
Workflows aren’t being run on this forked repository

Because this repository contained workflow files when it was forked, we have disabled them from running on this fork. Make sure you understand the configured workflows and their expected usage before enabling Actions on this repository.

Fig. 3 - Enabling Workflows

This warning is expected, as the GH actions code might perform undesired actions if not properly reviewed.

3. After accepting and waiting for the pipeline execution, you should now see the Trivy findings in the Security tab.
   You will need to manually trigger the pipeline execution by clicking in “Run workflow” in the “Actions” tab:



CICD Pipeline
cicd.yml

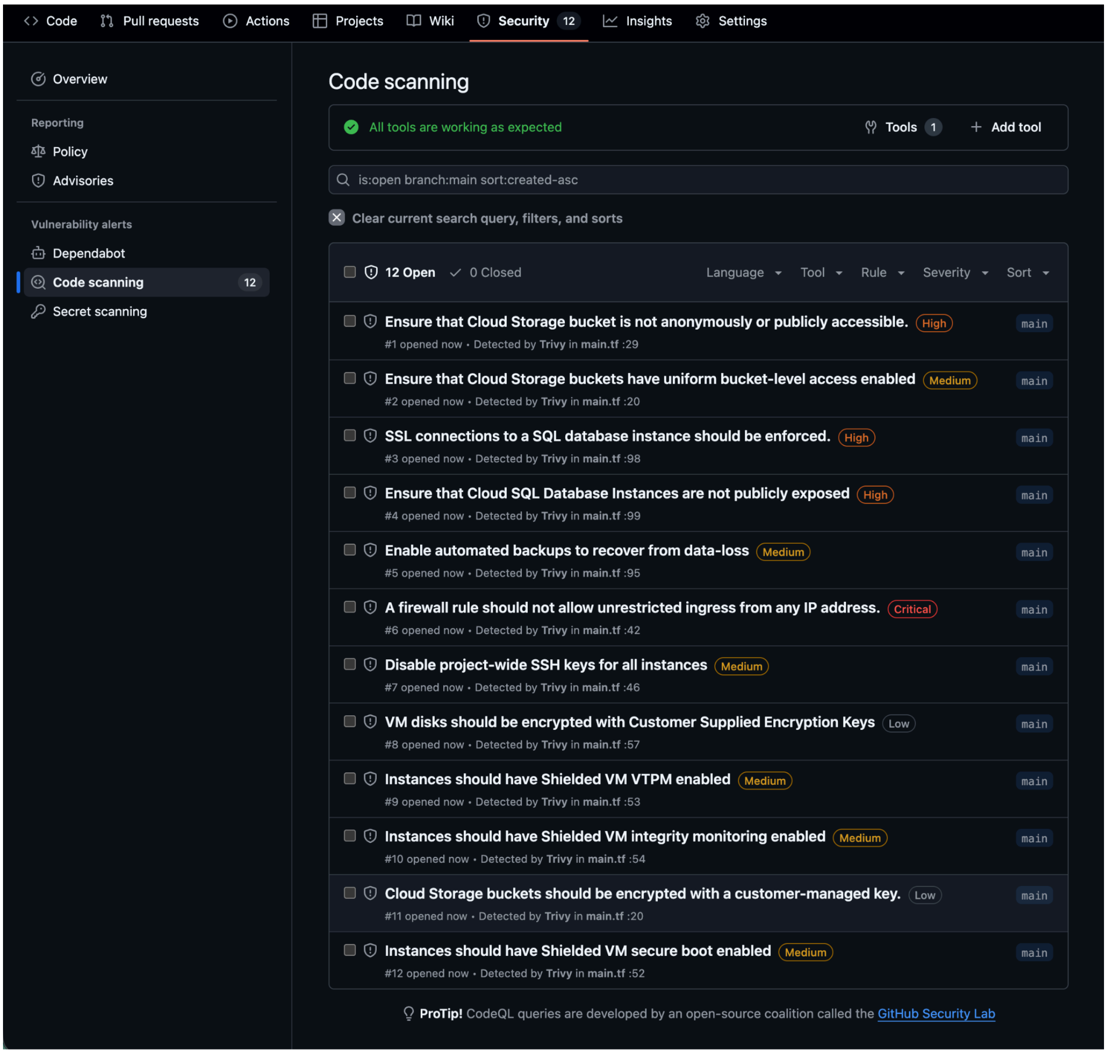
Help us improve GitHub Actions
Tell us how to make GitHub Actions work better for you with three quick questions.

0 workflow runs

This workflow has a `workflow_dispatch` event trigger.

Fig. 4 - Triggering workflow execution

4. After the pipeline concludes its execution, you can now see Trivy's findings in the "Security" tab - see below:



Code scanning

All tools are working as expected

is:open branch:main sort:created-asc

Clear current search query, filters, and sorts

12 Open ✓ 0 Closed

□ Ensure that Cloud Storage bucket is not anonymously or publicly accessible. High — main
#1 opened now • Detected by Trivy in main.tf :29

□ Ensure that Cloud Storage buckets have uniform bucket-level access enabled Medium — main
#2 opened now • Detected by Trivy in main.tf :20

□ SSL connections to a SQL database instance should be enforced. High — main
#3 opened now • Detected by Trivy in main.tf :98

□ Ensure that Cloud SQL Database Instances are not publicly exposed High — main
#4 opened now • Detected by Trivy in main.tf :99

□ Enable automated backups to recover from data-loss Medium — main
#5 opened now • Detected by Trivy in main.tf :95

□ A firewall rule should not allow unrestricted ingress from any IP address. Critical — main
#6 opened now • Detected by Trivy in main.tf :42

□ Disable project-wide SSH keys for all instances Medium — main
#7 opened now • Detected by Trivy in main.tf :46

□ VM disks should be encrypted with Customer Supplied Encryption Keys Low — main
#8 opened now • Detected by Trivy in main.tf :57

□ Instances should have Shielded VM VTPM enabled Medium — main
#9 opened now • Detected by Trivy in main.tf :53

□ Instances should have Shielded VM integrity monitoring enabled Medium — main
#10 opened now • Detected by Trivy in main.tf :54

□ Cloud Storage buckets should be encrypted with a customer-managed key. Low — main
#11 opened now • Detected by Trivy in main.tf :20

□ Instances should have Shielded VM secure boot enabled Medium — main
#12 opened now • Detected by Trivy in main.tf :52

ProTip! CodeQL queries are developed by an open-source coalition called the GitHub Security Lab

Fig. 5 - Trivy results in Security tab

5.3. Phase 3 - Activities

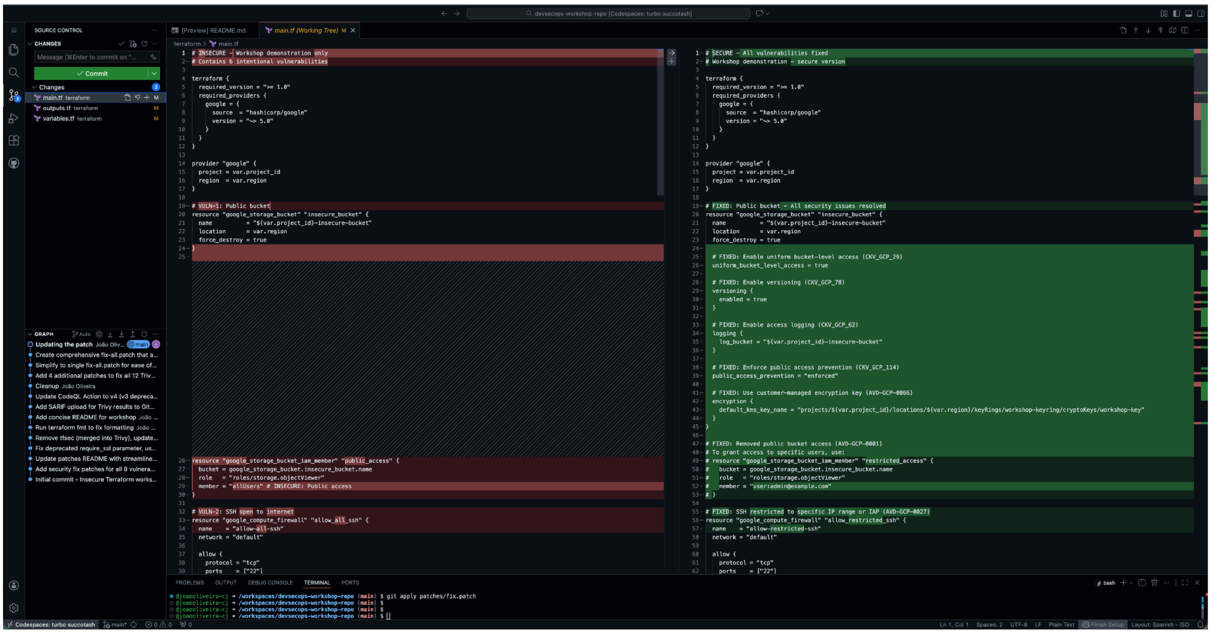
This phase has the following goals:

- See Trivy in action by identifying the lifecycle of a code vulnerability
- Apply code fixes to vulnerable resource definitions

Steps:

1. Go back to our Codespace or your local IDE
2. Fix our bad, vulnerable and insecure code. To accomplish this, you can either:
    a. Research each issue carefully by using Trivy's findings, Terraform documentation and Google Cloud Terraform Backend module's documentation. Apply the fixes manually.
    b. Use an LLM/AI code assistant like Github Copilot, Claude Code or by just copy-pasting to ChatGPT.
    c. Apply the code patch available in the repo - this will save time and the environment.
3. To run the code patch: `git apply patch/fix.patch`
4. Inspecting the code changes the patch applied, you can see the many fixes on the resource definitions such as:
    - Storage Bucket (insecure_bucket):
        - Enabled uniform bucket-level access
        - Enabled versioning
        - Added access logging
        - Enforced public access prevention
        - Added KMS encryption
        - Removed public IAM access (allUsers)
    - Firewall (allow_restricted_ssh):
        - Restricted SSH to IAP range only (35.235.240.0/20)
    - Compute Instance (insecure_instance):
        - Enabled Shielded VM (secure boot, vTPM, integrity monitoring)
        - Added disk KMS encryption
        - Enabled OS Login
        - Blocked project-wide SSH keys
        - Reduced service account scopes
    - IAM (insecure_sa):
        - Replaced Editor role with specific roles (compute.instanceAdmin, storage.objectViewer)
    - Database (insecure_db):
        - Enabled backups
        - Disabled public IP
        - Enforced SSL/TLS (TRUSTED_CLIENT_CERTIFICATE_REQUIRED)
    - Variables:
        - Removed hardcoded secrets
        - Marked sensitive variables as sensitive = true
    - Outputs:

- Replaced database_public_ip with connection_name
- Marked service_account_email as sensitive



Updating the patch João Oliv... main
Create comprehensive fix-all.patch that a...
Simplify to single fix-all.patch for ease of...
Add 4 additional patches to fix all 12 Triv...
Cleanup João Oliveira
Update CodeQL Action to v4 (v3 depreca...
Add SARIF upload for Trivy results to Git...
Add concise README for workshop João ...
Run terraform fmt to fix formatting João ...
Remove tfsec (merged into Trivy), update...
Fix deprecated require_ssl parameter, us...
Update patches README with streamline...
Add security fix patches for all 8 vulnera...
Initial commit - Insecure Terraform works...

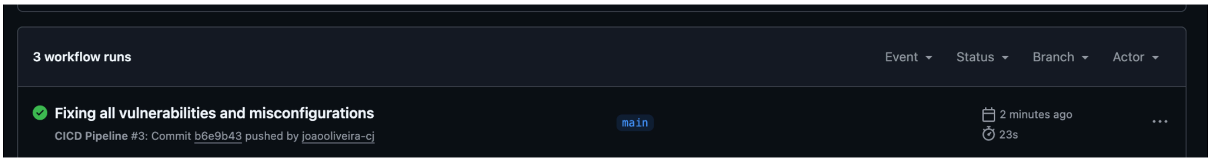
```
# INSECURE - Workshop demonstration only
# Contains 6 intentional vulnerabilities

terraform {
  required_version = ">= 1.0"
  required_providers {
    google = {
      source  = "hashicorp/google"
      version = "~> 5.0"
    }
  }
}

provider "google" {
  project = var.project_id
  region  = var.region
}

# VULN-1: Public bucket
resource "google_storage_bucket" "insecure_bucket" {
  name          = "${var.project_id}-insecure-bucket"
  location      = var.region
  force_destroy = true
}

resource "google_storage_bucket_iam_member" "public_access" {
  bucket = google_storage_bucket.insecure_bucket.name
  role   = "roles/storage.objectViewer"
  member = "allUsers" # INSECURE: Public access
}

# VULN-2: SSH open to internet
resource "google_compute_firewall" "allow_all_ssh" {
  name    = "allow-all-ssh"
  network = "default"

  allow {
    protocol = "tcp"
    ports    = ["22"]
```

```
# SECURE - All vulnerabilities fixed
# Workshop demonstration - secure version

terraform {
  required_version = ">= 1.0"
  required_providers {
    google = {
      source  = "hashicorp/google"
      version = "~> 5.0"
    }
  }
}

provider "google" {
  project = var.project_id
  region  = var.region
}

# FIXED: Public bucket - All security issues resolved
resource "google_storage_bucket" "insecure_bucket" {
  name          = "${var.project_id}-insecure-bucket"
  location      = var.region
  force_destroy = true

  # FIXED: Enable uniform bucket-level access (CKV_GCP_29)
  uniform_bucket_level_access = true

  # FIXED: Enable versioning (CKV_GCP_78)
  versioning {
    enabled = true
  }

  # FIXED: Enable access logging (CKV_GCP_62)
  logging {
    log_bucket = "${var.project_id}-insecure-bucket"
  }

  # FIXED: Enforce public access prevention (CKV_GCP_114)
  public_access_prevention = "enforced"

  # FIXED: Use customer-managed encryption key (AVD-GCP-0066)
  encryption {
    default_kms_key_name = "projects/${var.project_id}/locations/${var.region}/keyRings/workshop-keyring/cryptoKeys/workshop-key"
  }
}

# FIXED: Removed public bucket access (AVD-GCP-0001)
# To grant access to specific users, use:
# resource "google_storage_bucket_iam_member" "restricted_access" {
#   bucket = google_storage_bucket.insecure_bucket.name
#   role   = "roles/storage.objectViewer"
#   member = "user:admin@example.com"
# }

# FIXED: SSH restricted to specific IP range or IAP (AVD-GCP-0027)
resource "google_compute_firewall" "allow_restricted_ssh" {
  name    = "allow-restricted-ssh"
  network = "default"

  allow {
    protocol = "tcp"
    ports    = ["22"]
```

```
@joaooliveira-cj → /workspaces/devsecops-workshop-repo (main) $ git apply patches/fix.patch
@joaooliveira-cj → /workspaces/devsecops-workshop-repo (main) $
@joaooliveira-cj → /workspaces/devsecops-workshop-repo (main) $
@joaooliveira-cj → /workspaces/devsecops-workshop-repo (main) $ 
```

Fig. 6 - Patch fixes

5. Push the changes! See the pipeline in action

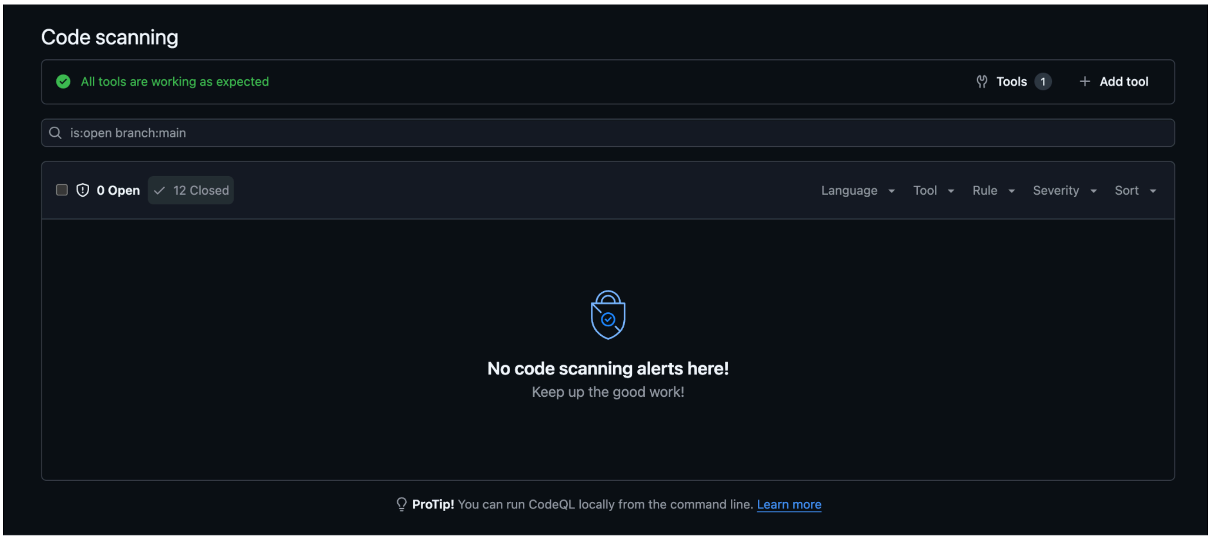


3 workflow runs

✓ Fixing all vulnerabilities and misconfigurations
CICD Pipeline #3: Commit b6e9b43 pushed by joaooliveira-cj
main
2 minutes ago
23s

Fig. 7 - Pipeline execution

6. Checking the Security tab in the Code scanning results tab, you should now see all the vulnerabilities Trivy identified being marked as secure.



Code scanning

✓ All tools are working as expected

is:open branch:main

□ 0 Open ✓ 12 Closed

No code scanning alerts here!

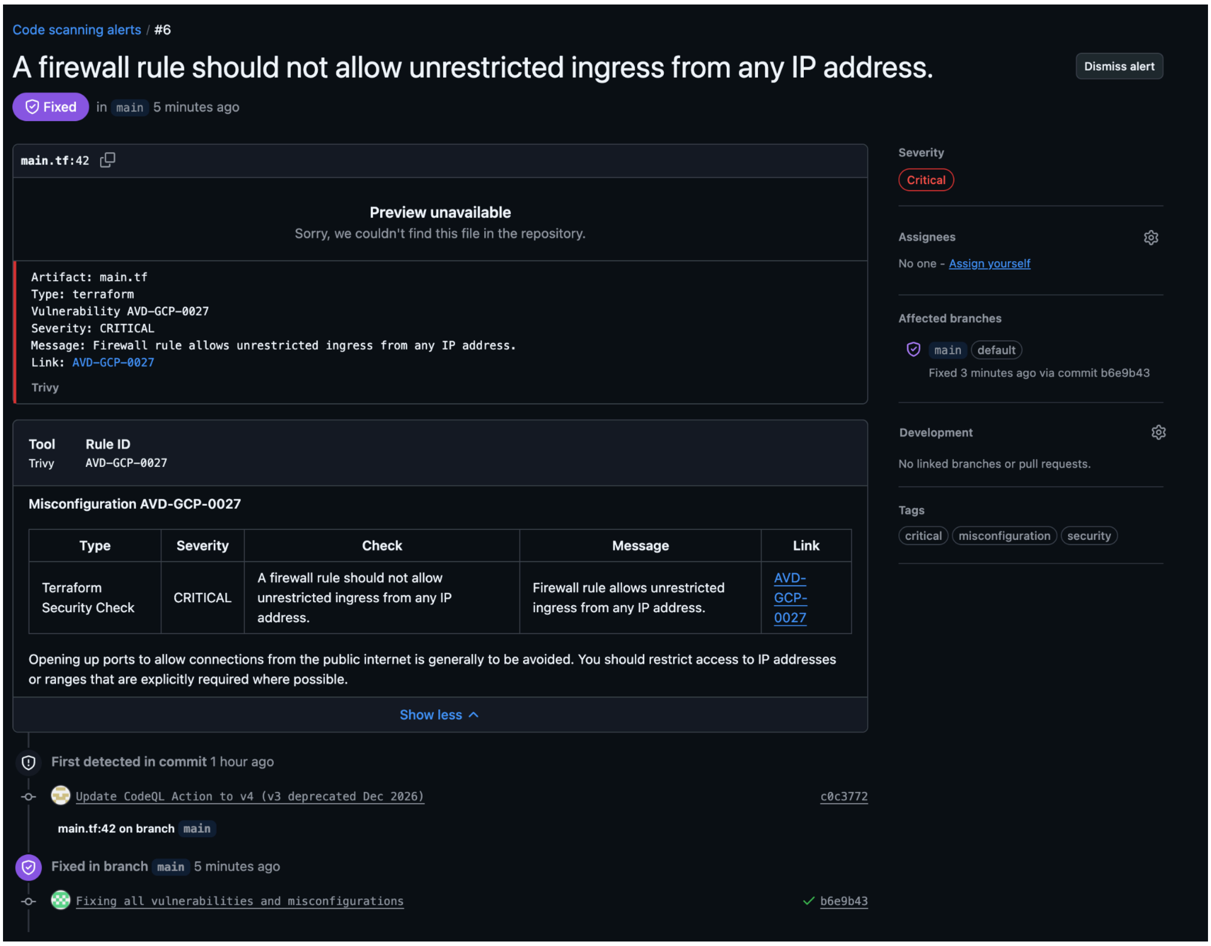
Keep up the good work!

ProTip! You can run CodeQL locally from the command line. Learn more

Fig. 8 - Security Issues marked as closed

7. You can also validate by going on each individual issue, and see that the code fixes

pushed did solve the issue.



A firewall rule should not allow unrestricted ingress from any IP address.

Preview unavailable
Sorry, we couldn't find this file in the repository.

```
Artifact: main.tf
Type: terraform
Vulnerability AVD-GCP-0027
Severity: CRITICAL
Message: Firewall rule allows unrestricted ingress from any IP address.
Link: AVD-GCP-0027
```

Trivy

Tool	Rule ID
Trivy	AVD-GCP-0027

Misconfiguration AVD-GCP-0027

Type	Severity	Check	Message	Link
Terraform Security Check	CRITICAL	A firewall rule should not allow unrestricted ingress from any IP address.	Firewall rule allows unrestricted ingress from any IP address.	AVD-GCP-0027

Opening up ports to allow connections from the public internet is generally to be avoided. You should restrict access to IP addresses or ranges that are explicitly required where possible.

First detected in commit 1 hour ago

Update CodeQL Action to v4 (v3 deprecated Dec 2026) c0c3772

main.tf:42 on branch main

Fixed in branch main 5 minutes ago

Fixing all vulnerabilities and misconfigurations ✓ b6e9b43

Severity: Critical

Assignees: No one - Assign yourself

Affected branches: main default — Fixed 3 minutes ago via commit b6e9b43

Development: No linked branches or pull requests.

Tags: critical misconfiguration security

Fig. 9 - Critical issue marked as fixed

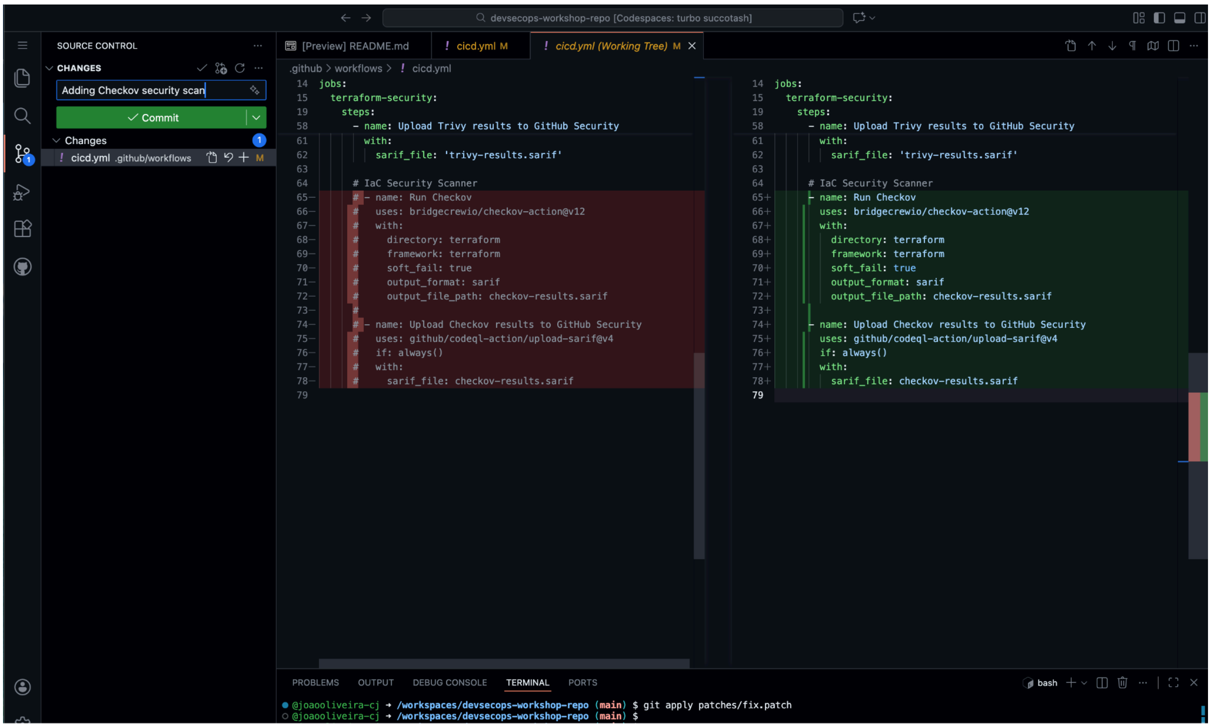
5.4. Phase 4 - Activities

This phase has the following goals:

- Enable additional security scanning tools
- Implement security gates to our CI pipeline

Steps:

1. Go back to your working directory.
2. Open the file ./.github/workflows/cicd.yml
3. Go to the end of the file and uncomment the last two steps - see below. Commit and push, then wait for the pipeline execution.



SOURCE CONTROL
CHANGES
Commit
Changes
cicd.yml .github/workflows
[Preview] README.md
cicd.yml M
cicd.yml (Working Tree) M
.github > workflows > cicd.yml
devsecops-workshop-repo [Codespaces: turbo succotash]
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
bash

Adding Checkov security scan

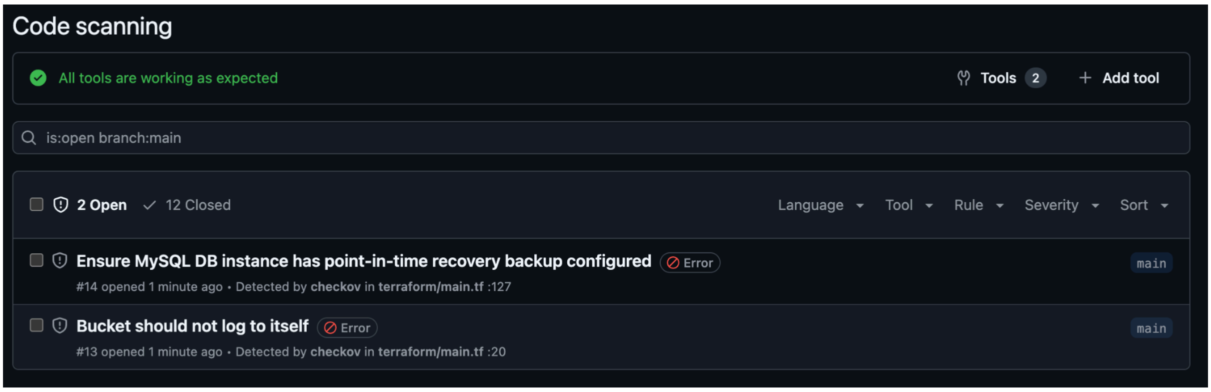
```yaml
jobs:
  terraform-security:
    steps:
      - name: Upload Trivy results to GitHub Security
        with:
          sarif_file: 'trivy-results.sarif'

      # IaC Security Scanner
      #- name: Run Checkov
      #  uses: bridgecrewio/checkov-action@v12
      #  with:
      #    directory: terraform
      #    framework: terraform
      #    soft_fail: true
      #    output_format: sarif
      #    output_file_path: checkov-results.sarif
      #
      #- name: Upload Checkov results to GitHub Security
      #  uses: github/codeql-action/upload-sarif@v4
      #  if: always()
      #  with:
      #    sarif_file: checkov-results.sarif
```

```yaml
jobs:
  terraform-security:
    steps:
      - name: Upload Trivy results to GitHub Security
        with:
          sarif_file: 'trivy-results.sarif'

      # IaC Security Scanner
      - name: Run Checkov
        uses: bridgecrewio/checkov-action@v12
        with:
          directory: terraform
          framework: terraform
          soft_fail: true
          output_format: sarif
          output_file_path: checkov-results.sarif

      - name: Upload Checkov results to GitHub Security
        uses: github/codeql-action/upload-sarif@v4
        if: always()
        with:
          sarif_file: checkov-results.sarif
```

```
● @joaooliveira-cj → /workspaces/devsecops-workshop-repo (main) $ git apply patches/fix.patch
○ @joaooliveira-cj → /workspaces/devsecops-workshop-repo (main) $
```

Fig. 10 - Enabling Checkov scan

4. Checkov scan should have executed successfully and uploaded its additional findings to the Security tab:



All tools are working as expected

is:open branch:main

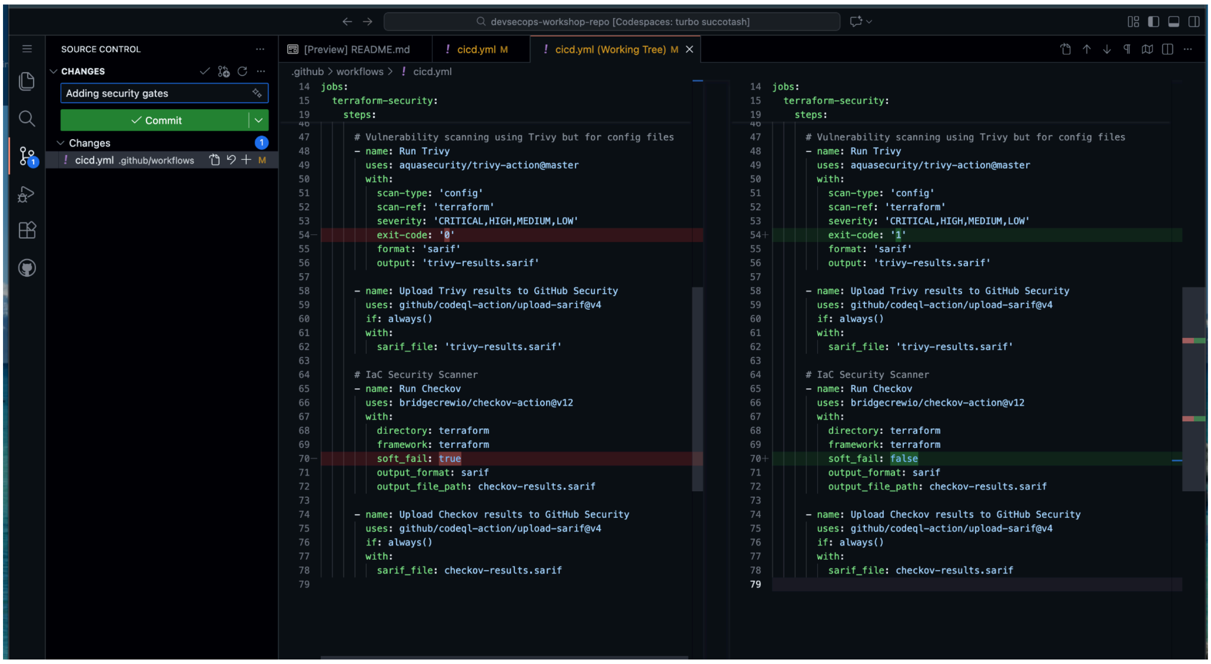
□ 2 Open ✓ 12 Closed

□ **Ensure MySQL DB instance has point-in-time recovery backup configured** Error — main
#14 opened 1 minute ago • Detected by **checkov** in **terraform/main.tf** :127

□ **Bucket should not log to itself** Error — main
#13 opened 1 minute ago • Detected by **checkov** in **terraform/main.tf** :20

Fig. 11 - Viewing Checkov findings in security tab

5. These additional findings provide additional value to our setup. For instance, we can see that even with our code vulnerabilities fixed, a bucket logging to itself is not useful in case of disaster recovery, or that our Cloud SQL instance might not have backups properly configured.
6. To ensure that subsequent steps (such as deployment steps) do not execute on instances where our tools detect security issues, add security gates to the tests. We will make Trivy and Checkov fail if the tools detect issues on the codebase.

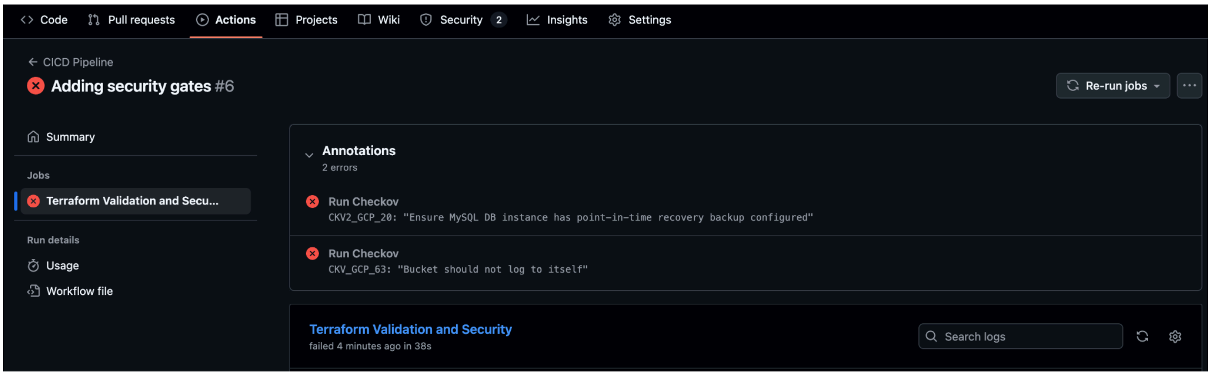

devsecops-workshop-repo [Codespaces: turbo succotash]
SOURCE CONTROL
CHANGES
Adding security gates
✓ Commit
Changes
cicd.yml .github/workflows M
[Preview] README.md
cicd.yml M
cicd.yml (Working Tree) M
.github > workflows > cicd.yml

```yaml
jobs:
  terraform-security:
    steps:
      # Vulnerability scanning using Trivy but for config files
      - name: Run Trivy
        uses: aquasecurity/trivy-action@master
        with:
          scan-type: 'config'
          scan-ref: 'terraform'
          severity: 'CRITICAL,HIGH,MEDIUM,LOW'
          exit-code: '0'
          format: 'sarif'
          output: 'trivy-results.sarif'

      - name: Upload Trivy results to GitHub Security
        uses: github/codeql-action/upload-sarif@v4
        if: always()
        with:
          sarif_file: 'trivy-results.sarif'

      # IaC Security Scanner
      - name: Run Checkov
        uses: bridgecrewio/checkov-action@v12
        with:
          directory: terraform
          framework: terraform
          soft_fail: true
          output_format: sarif
          output_file_path: checkov-results.sarif

      - name: Upload Checkov results to GitHub Security
        uses: github/codeql-action/upload-sarif@v4
        if: always()
        with:
          sarif_file: checkov-results.sarif
```

```yaml
jobs:
  terraform-security:
    steps:
      # Vulnerability scanning using Trivy but for config files
      - name: Run Trivy
        uses: aquasecurity/trivy-action@master
        with:
          scan-type: 'config'
          scan-ref: 'terraform'
          severity: 'CRITICAL,HIGH,MEDIUM,LOW'
          exit-code: '1'
          format: 'sarif'
          output: 'trivy-results.sarif'

      - name: Upload Trivy results to GitHub Security
        uses: github/codeql-action/upload-sarif@v4
        if: always()
        with:
          sarif_file: 'trivy-results.sarif'

      # IaC Security Scanner
      - name: Run Checkov
        uses: bridgecrewio/checkov-action@v12
        with:
          directory: terraform
          framework: terraform
          soft_fail: false
          output_format: sarif
          output_file_path: checkov-results.sarif

      - name: Upload Checkov results to GitHub Security
        uses: github/codeql-action/upload-sarif@v4
        if: always()
        with:
          sarif_file: checkov-results.sarif
```

Fig. 12 - Enabling security scans

7. Commit and push the code changes. View the pipeline execution. You can now validate that the pipeline failed upon Checkov's security scan tool results.



Adding security gates #6

Terraform Validation and Secu...

Annotations

2 errors

Run Checkov

```
CKV2_GCP_20: "Ensure MySQL DB instance has point-in-time recovery backup configured"
```

Run Checkov

```
CKV_GCP_63: "Bucket should not log to itself"
```

Terraform Validation and Security

failed 4 minutes ago in 38s

Fig. 13 - Viewing Checkov findings in CICD pipeline results

You now have a secure environment that you could create using Terraform! You can use these tools to further analyze new additions to the infrastructure resource declaration and ensure all activities