Secure System Development - Lab 1 Report

Overview

This report documents the steps taken to set up a self-managed GitLab server and Runner. The tasks include deploying GitLab Server, setting up GitLab Runner, and integrating SAST using Semgrep.

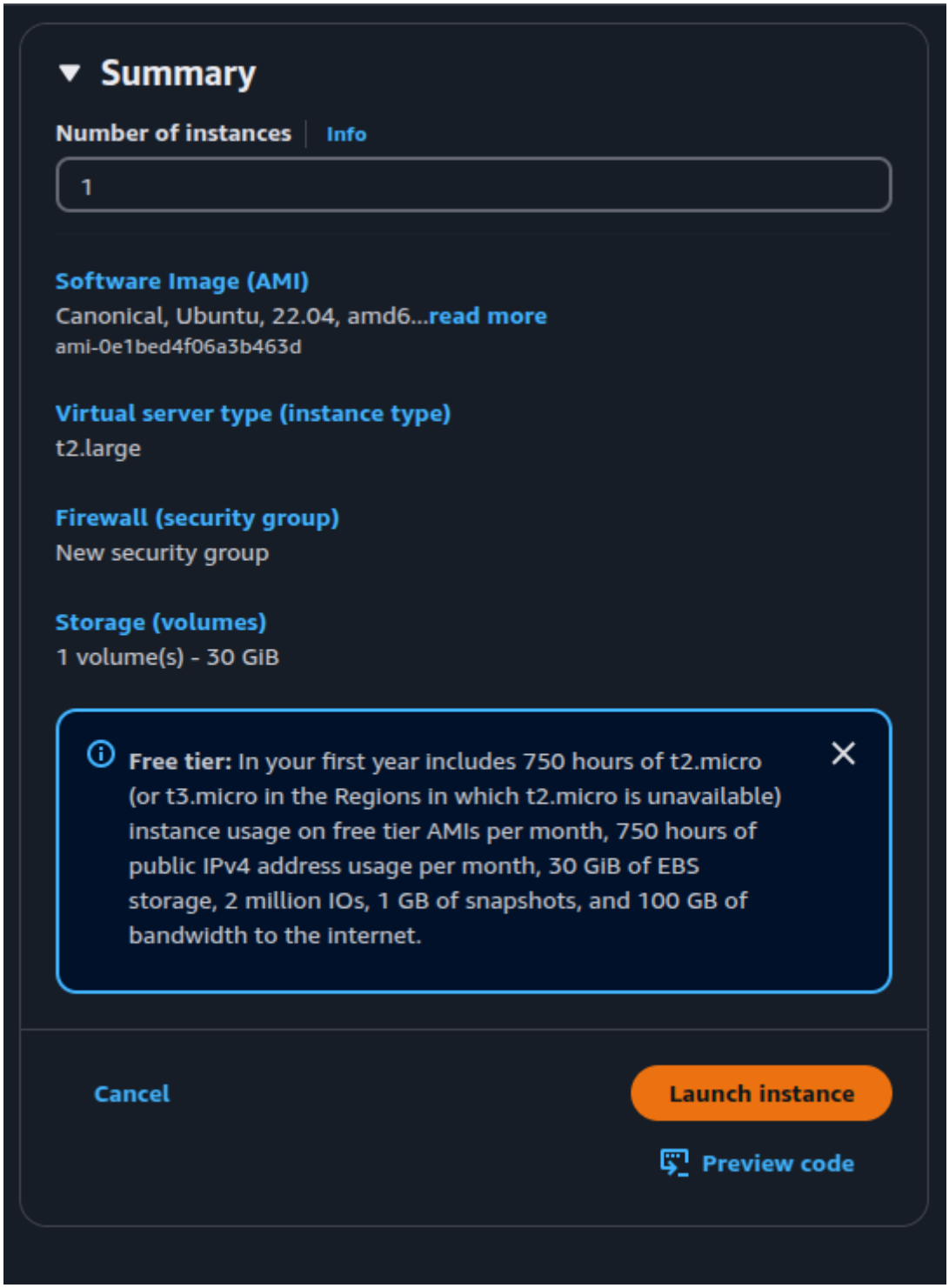
Task 1: Setting Up GitLab Server

Step 1: Provisioning EC2 Instances

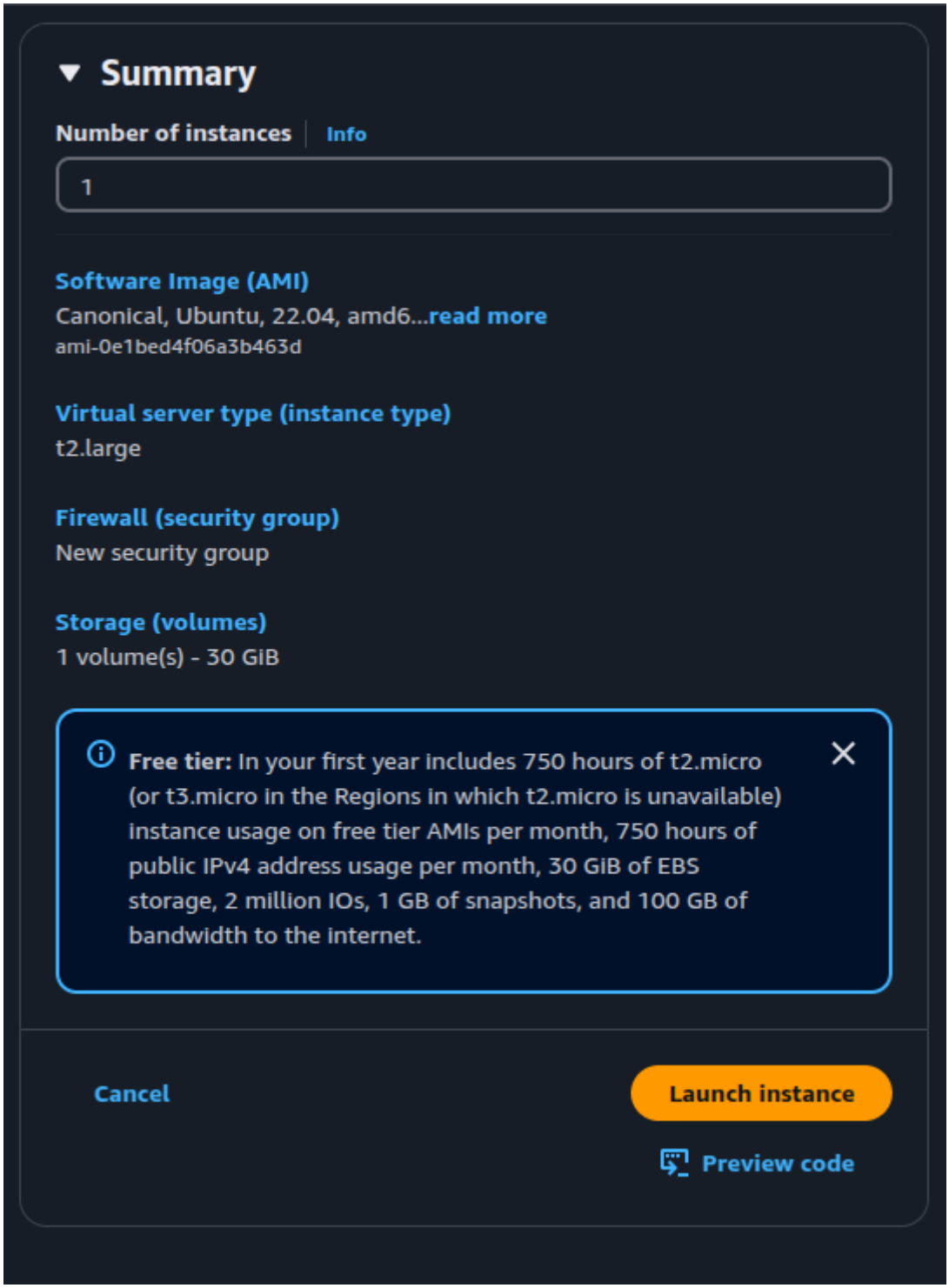
- Created two Amazon EC2 instances:
  - **GitLab Server**: Hosts the GitLab application.
  - **GitLab Runner**: Executes CI/CD jobs.

Evidence:

- GitLab Server Instance:



- GitLab Runner Instance:



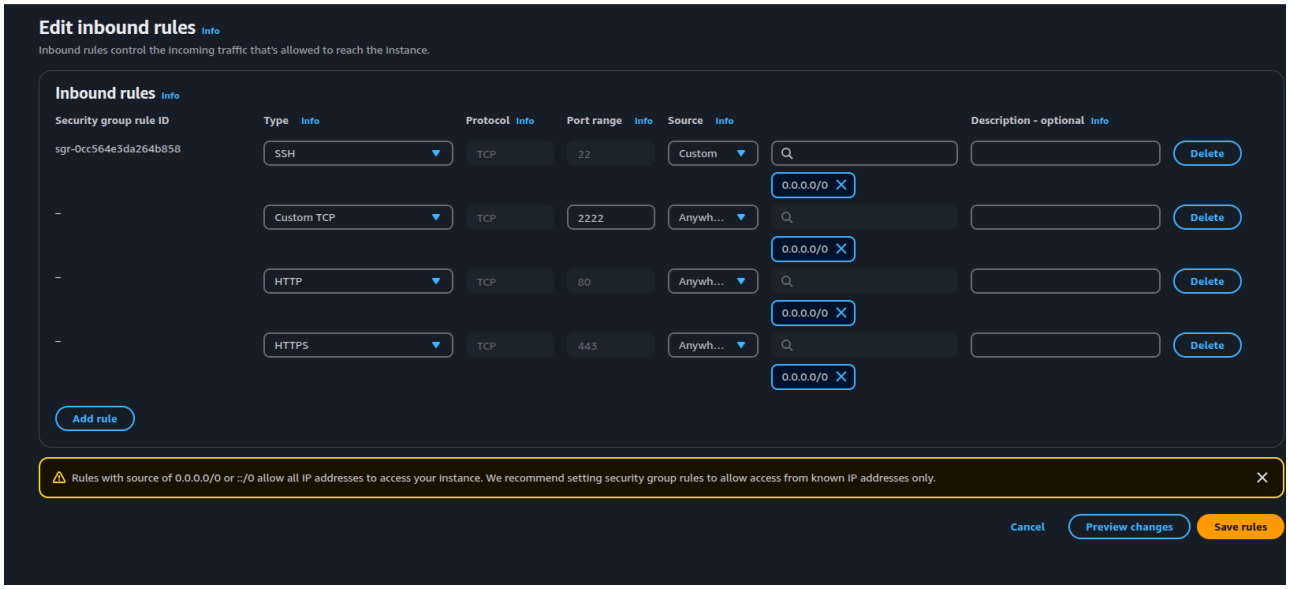
Step 2: Configuring Security Groups

- Updated security groups to allow inbound traffic:
  - **GitLab Server**:
    - SSH on port 2222
    - HTTP on port 80
    - HTTPS on port 443
  - **GitLab Runner**:
    - SSH on port 22
    - HTTP on port 80

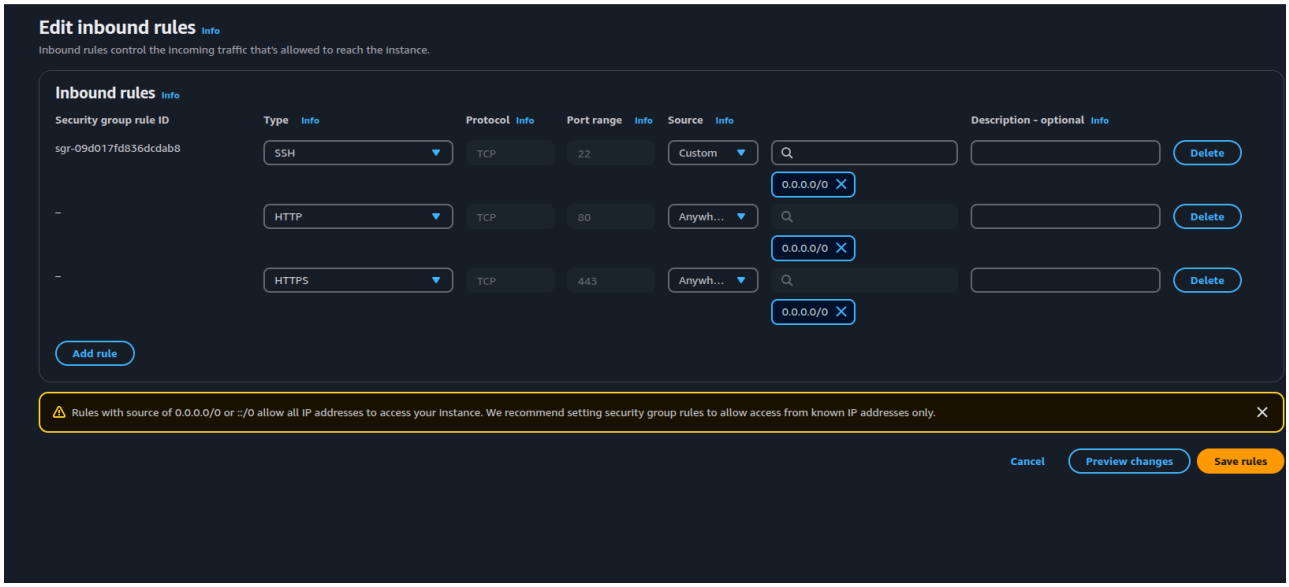
- HTTPS on port 443

Evidence:

- GitLab Server Security Group:



- GitLab Runner Security Group:



Step 3: Installing Docker on GitLab Server

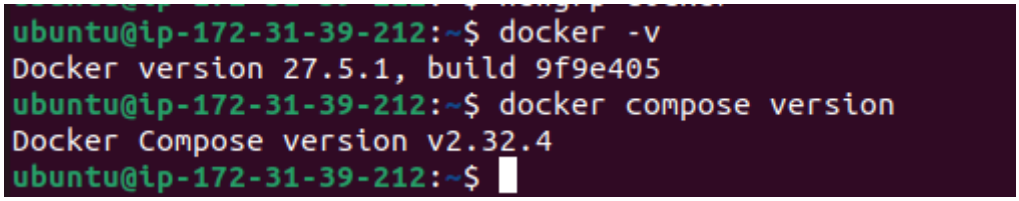
- Connected to the GitLab Server instance via SSH and installed Docker.

Commands Executed:

```
ssh -i "gitlab1.pem" ubuntu@ec2-54-157-39-84.compute-1.amazonaws.com

# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o
/etc/apt/keyrings/docker.asc
```

```
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-
by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}")
stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update

sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-
plugin docker-compose-plugin

sudo usermod -aG docker $USER
newgrp docker
docker -v
docker compose version
```

Evidence:

- Docker Installation: 

Step 4: Creating `docker-compose.yml` for GitLab Server

- Created a `docker-compose.yml` file to configure and run the GitLab container.

File Content:

```
version: '3.8'
services:
  gitlab:
    image: gitlab/gitlab-ce:latest
    container_name: 22BS283-gitlab
    restart: always
    hostname: 'gitlab.test.local'
    environment:
      GITLAB_OMNIBUS_CONFIG: |
        external_url 'https://gitlab.test.local'
        gitlab_rails['gitlab_shell_ssh_port'] = 2222
        nginx['http2_enabled'] = true
        nginx['redirect_http_to_https'] = true
        nginx['ssl_certificate'] = "/etc/gitlab/ssl/gitlab.test.local.crt"
        nginx['ssl_certificate_key'] =
"/etc/gitlab/ssl/gitlab.test.local.key"
        gitlab_rails['registry_enabled'] = false
        mattermost['enable'] = false
```

```
          gitlab_pages['enable'] = false
          gitlab_kas['enable'] = false
          letsencrypt['enable'] = false
      ports:
        - '80:80'
        - '443:443'
        - '2222:22'
      volumes:
        - '/srv/gitlab/config:/etc/gitlab'
        - '/srv/gitlab/logs:/var/log/gitlab'
        - '/srv/gitlab/data:/var/opt/gitlab'
        - '/etc/gitlab/ssl:/etc/gitlab/ssl'
      shm_size: '256m'
```

Evidence:

- `docker-compose.yml` File:

```
ubuntu@ip-172-31-39-212:~$ cd gitlab-server/
ubuntu@ip-172-31-39-212:~/gitlab-server$ nano docker-compose.yml
ubuntu@ip-172-31-39-212:~/gitlab-server$ cat docker-compose.yml
version: '3.8'
services:
  gitlab:
    image: gitlab/gitlab-ce:latest
    container_name: 22BS283-gitlab
    restart: always
    hostname: 'gitlab.local.test'
    environment:
      GITLAB_OMNIBUS_CONFIG: |
        external_url 'https://gitlab.local.test'
        gitlab_rails['gitlab_shell_ssh_port'] = 2222
        nginx['http2_enabled'] = true
        nginx['redirect_http_to_https'] = true
        nginx['ssl_certificate'] = "/etc/gitlab/ssl/_wildcard.local.test.crt"
        nginx['ssl_certificate_key'] = "/etc/gitlab/ssl/_wildcard.local.test.key"
        gitlab_rails['registry_enabled'] = false
        mattermost['enable'] = false
        gitlab_pages['enable'] = false
        gitlab_kas['enable'] = false
        letsencrypt['enable'] = false
    ports:
      - '80:80'
      - '443:443'
      - '2222:22'
    volumes:
      - '/srv/gitlab/config:/etc/gitlab'
      - '/srv/gitlab/logs:/var/log/gitlab'
      - '/srv/gitlab/data:/var/opt/gitlab'
      - '/etc/gitlab/ssl:/etc/gitlab/ssl'
    shm_size: '256m'
ubuntu@ip-172-31-39-212:~/gitlab-server$
```

Step 5: Generating Self-Signed Certificates with `mkcert`

- Installed `mkcert` and generated self-signed certificates for HTTPS.

Evidence:

- Certificate Generation:

```
        $ mkcert -install
        Install the local CA in the system trust store.

        $ mkcert example.org
        Generate "example.org.pem" and "example.org-key.pem".

        $ mkcert example.com myapp.dev localhost 127.0.0.1 ::1
        Generate "example.com+4.pem" and "example.com+4-key.pem".

        $ mkcert "*.example.it"
        Generate "_wildcard.example.it.pem" and "_wildcard.example.it-key.pem".

        $ mkcert -uninstall
        Uninstall the local CA (but do not delete it).

For more options, run "mkcert -help".
ubuntu@ip-172-31-39-212:~/gitlab-server$ mkcert -install
Created a new local CA 💥
The local CA is now installed in the system trust store! ⚡

ubuntu@ip-172-31-39-212:~/gitlab-server$ sudo mk
mk_modmap         mkdosfs           mkfs              mkfs.cramfs       mkfs.ext4         mkfs.msdos        mkfs.xfs          mklost+found      mksquashfs
mkcert            mke2fs            mkfs.bfs          mkfs.ext2         mkfs.fat          mkfs.ntfs         mkhomedir_helper  mknod             mkswap
mkdir             mkfifo            mkfs.btrfs        mkfs.ext3         mkfs.minix        mkfs.vfat         mkinitramfs       mkntfs            mktemp
ubuntu@ip-172-31-39-212:~/gitlab-server$ sudo mkdir -p /etc/gitlab/ssl
ubuntu@ip-172-31-39-212:~/gitlab-server$ sudo mkcert -key-file /etc/gitlab/ssl/gitlab.local.test.key -cert-file /etc/gitlab/ssl/gitlab.local.test.crt gitlab.local.test
Created a new local CA 💥
Note: the local CA is not installed in the system trust store.
Run "mkcert -install" for certificates to be trusted automatically ⚠️

Created a new certificate valid for the following names 📜
 - "gitlab.local.test"

The certificate is at "/etc/gitlab/ssl/gitlab.local.test.crt" and the key at "/etc/gitlab/ssl/gitlab.local.test.key" ✅

It will expire on 10 May 2027 🗓

ubuntu@ip-172-31-39-212:~/gitlab-server$
```

Step 6: Updating `/etc/hosts`

- Updated the `/etc/hosts` file to resolve `gitlab.test.local` to the server's public IP.

Evidence:

- Updated `/etc/hosts`:

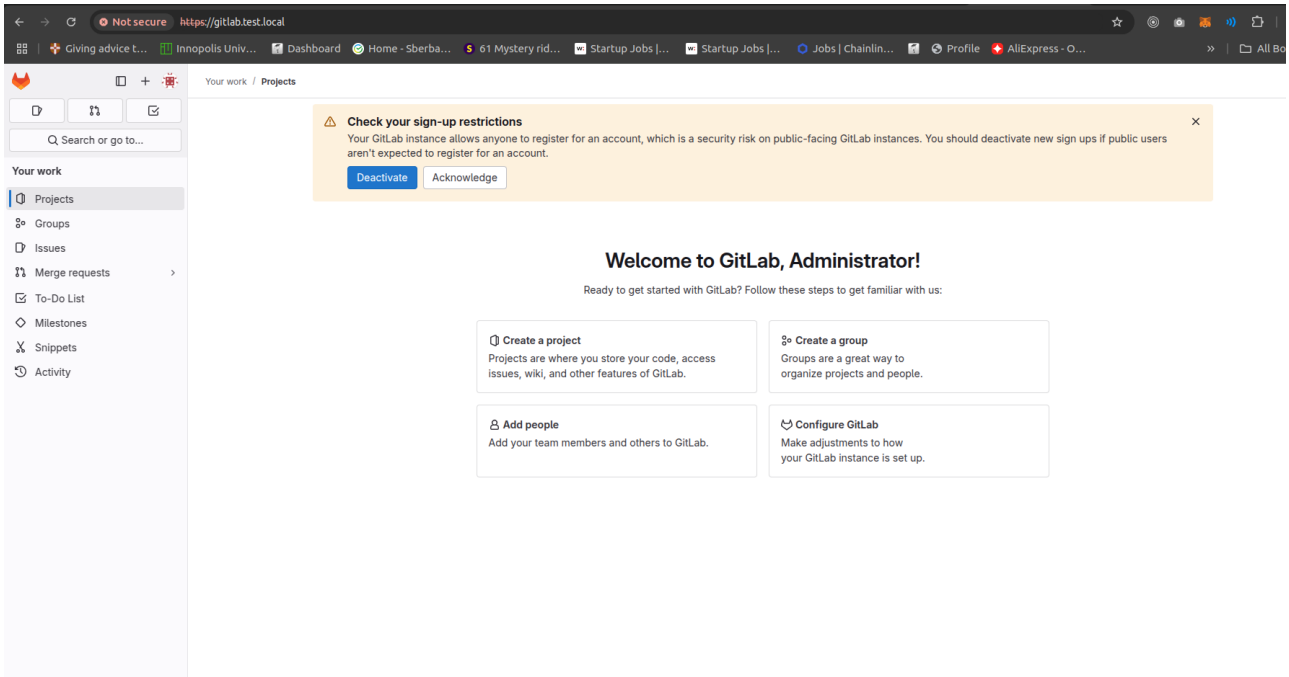
```
127.0.0.1 localhost
54.157.39.84 gitlab.test.local
# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
```

Step 7: Running GitLab Server

- Started the GitLab container using `docker-compose`.

Evidence:

- GitLab Server Running:

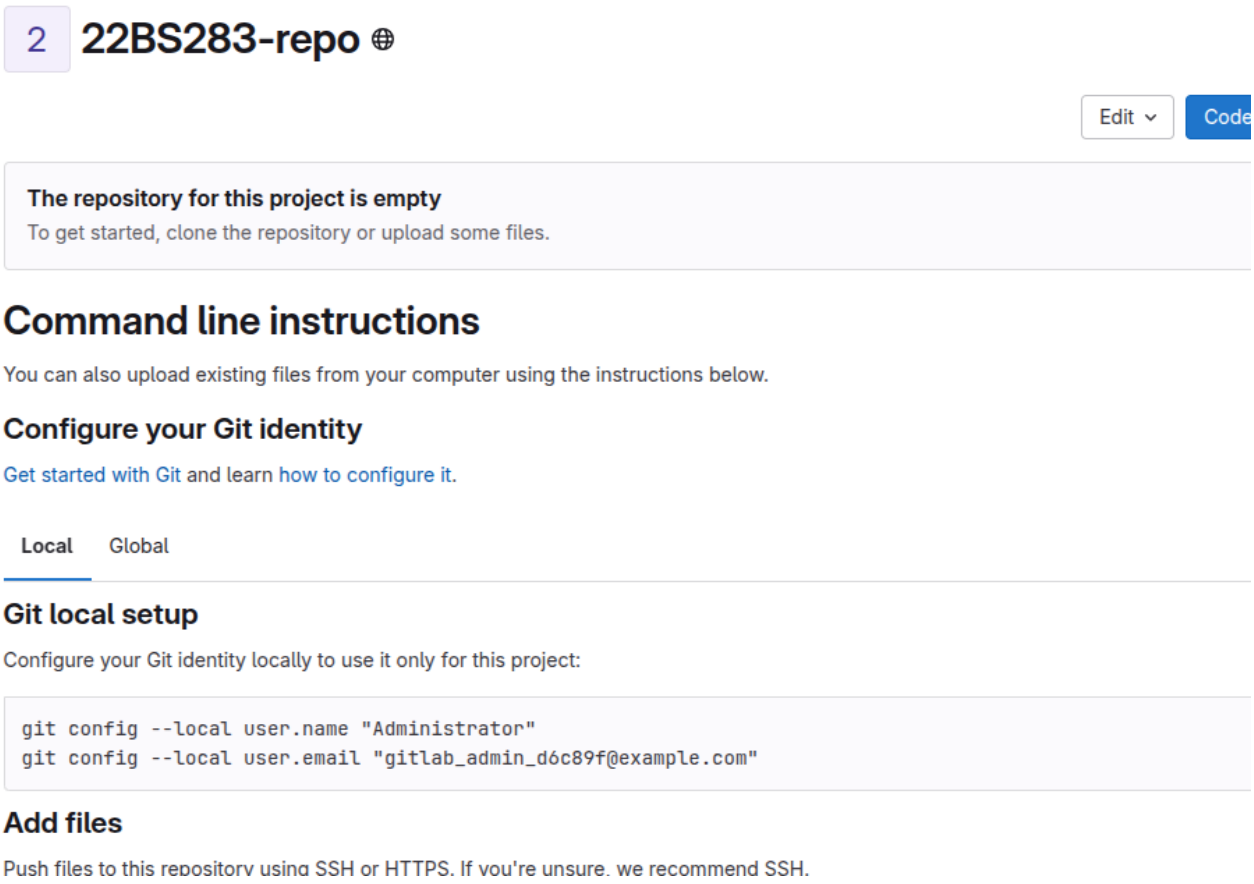


Step 8: Creating a Repository

- Created an empty repository named `22BS283-repo`.

Evidence:

- Repository Creation:



Task 2: Setting Up GitLab Runner

Step 1: Installing GitLab Runner

- Installed GitLab Runner on the second EC2 instance.

Evidence:

- GitLab Runner Installation:

```
ubuntu@ip-172-31-47-104:~$ gitlab-runner -version
Version:      17.8.3
Git revision: 690ce25c
Git branch:   17-8-stable
GO version:   go1.23.2 X:cacheprog
Built:        unknown
OS/Arch:      linux/amd64
```

Step 2: Updating `/etc/hosts` on Runner

- Updated the `/etc/hosts` file on the Runner to resolve `gitlab.test.local`.

Evidence:

- Updated `/etc/hosts` on Runner:

```
  GNU nano 6.2
127.0.0.1 localhost
54.157.39.84 gitlab.test.local
# The following lines are desirable for IPv6 capable ho
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
```

Step 3: Adding GitLab Server Certificate to Runner

- Added the GitLab server's self-signed certificate to the Runner to establish trust.

Evidence:

- Certificate Addition:

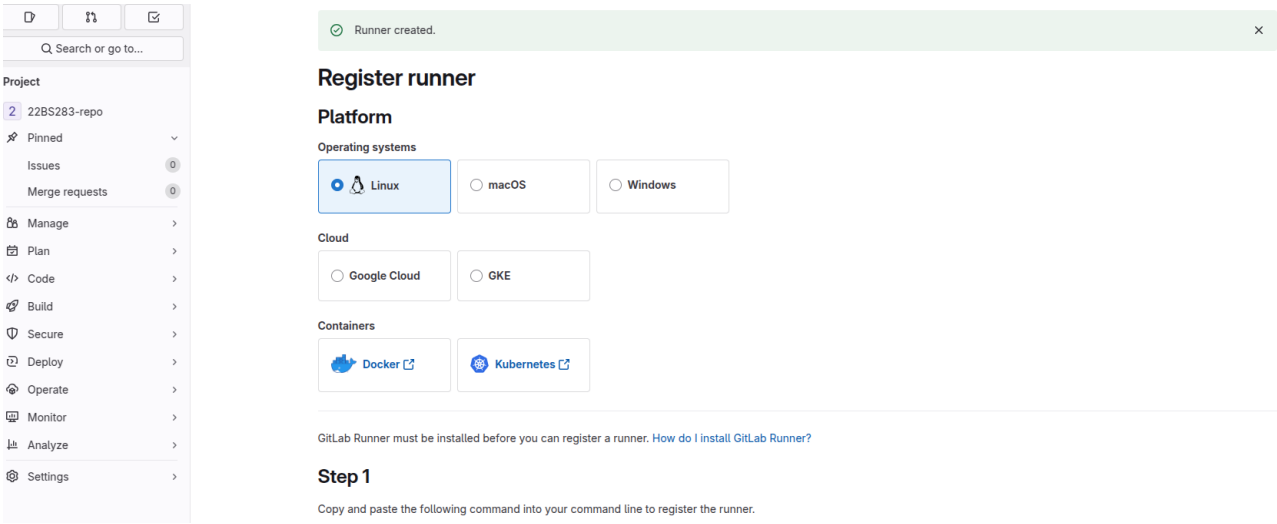
```
ubuntu@ip-172-31-47-104:~$ sudo nano gitlab.test.local.crt
ubuntu@ip-172-31-47-104:~$ sudo cp gitlab.test.local.crt /usr/local/share/ca-certificates/
sudo update-ca-certificates
Updating certificates in /etc/ssl/certs...
rehash: warning: skipping ca-certificates.crt,it does not contain exactly one certificate or CRL
1 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d...
done.
ubuntu@ip-172-31-47-104:~$
```

Step 4: Creating and Registering Runner

- Created a GitLab Runner with the tag `22BS283-runner` and registered it with the GitLab server.

Evidence:

- Runner Creation:



- Runner Registration:

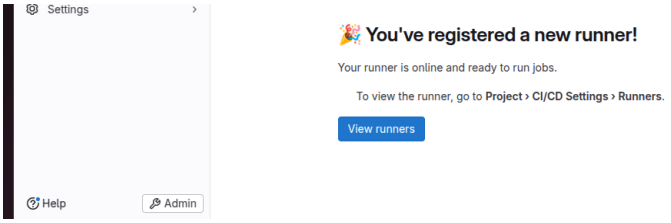
```
done.
ubuntu@ip-172-31-47-104:~$ gitlab-runner register  --url https://gitlab.test.local  --token glrt-t3_q4_sZy29fZznT9mj1RBJ
Runtime platform                                    arch=amd64 os=linux pid=3712 revision=690ce25c version=17.8.3
WARNING: Running in user-mode.
WARNING: The user-mode requires you to manually start builds processing:
WARNING: $ gitlab-runner run
WARNING: Use sudo for system-mode:
WARNING: $ sudo gitlab-runner...

Created missing unique system ID                    system_id=s_5fb68c104fd4
Enter the GitLab instance URL (for example, https://gitlab.com/):
[https://gitlab.test.local]: https://gitlab.test.local
Verifying runner... is valid                        runner=t3_q4_sZy
Enter a name for the runner. This is stored only in the local config.toml file:
[ip-172-31-47-104]: 22BS283-runner
Enter an executor: virtualbox, docker-windows, docker+machine, kubernetes, instance, custom, parallels, docker, docker-autoscaler, shell, ssh:
shell
Runner registered successfully. Feel free to start it, but if it's running already the config should be automatically reloaded!

Configuration (with the authentication token) was saved in "/home/ubuntu/.gitlab-runner/config.toml"
ubuntu@ip-172-31-47-104:~$ gitlab-runner run
Runtime platform                                    arch=amd64 os=linux pid=3721 revision=690ce25c version=17.8.3
Starting multi-runner from /home/ubuntu/.gitlab-runner/config.toml...  builds=0 max_builds=0
WARNING: Running in user-mode.
WARNING: Use sudo for system-mode:
WARNING: $ sudo gitlab-runner...

Configuration loaded                                builds=0 max_builds=1
listen_address not defined, metrics & debug endpoints disabled  builds=0 max_builds=1
[session_server].listen_address not defined, session endpoints disabled  builds=0 max_builds=1
Initializing executor providers                     builds=0 max_builds=1
```

Step 5: Validating Runner Connection

- Verified that the Runner is connected to the GitLab server.

Evidence:

- Runner Validation:



Task 3: Integrating SAST with GitLab CI

Step 1: Cloning a Vulnerable Application

- Cloned the Damn Vulnerable Java Application (DVJA) and removed its `.git` directory.

Evidence:

- Cloning DVJA:

```
ubuntu@ip-172-31-39-212:~/22bs283-repo$ git clone https://github.com/appsecco/dvja.git
Cloning into 'dvja'...
remote: Enumerating objects: 256, done.
remote: Total 256 (delta 0), reused 0 (delta 0), pack-reused 256 (from 1)
Receiving objects: 100% (256/256), 1.17 MiB | 17.85 MiB/s, done.
Resolving deltas: 100% (54/54), done.
ubuntu@ip-172-31-39-212:~/22bs283-repo$ cd dvja/
ubuntu@ip-172-31-39-212:~/22bs283-repo/dvja$ rm -r .
./             ../            .dockerignore  .git/          .gitignore     .idea/
ubuntu@ip-172-31-39-212:~/22bs283-repo/dvja$ rm -r .git
rm: remove write-protected regular file '.git/objects/pack/pack-bb135212baf9b4e2ff0119116e061287014851b5.idx'? yes
rm: remove write-protected regular file '.git/objects/pack/pack-bb135212baf9b4e2ff0119116e061287014851b5.pack'? yes
ubuntu@ip-172-31-39-212:~/22bs283-repo/dvja$ cp -R . ~/22bs283-repo/
ubuntu@ip-172-31-39-212:~/22bs283-repo/dvja$ rm -r d
db/                docker-compose.yml  docs/
ubuntu@ip-172-31-39-212:~/22bs283-repo/dvja$ cd ..
ubuntu@ip-172-31-39-212:~/22bs283-repo$ ls
Dockerfile  LICENSE  README.md  db  docker-compose.yml  docs  dvja  pom.xml  scripts  src
ubuntu@ip-172-31-39-212:~/22bs283-repo$ rm -r dvja/
ubuntu@ip-172-31-39-212:~/22bs283-repo$ ls
Dockerfile  LICENSE  README.md  db  docker-compose.yml  docs  pom.xml  scripts  src
ubuntu@ip-172-31-39-212:~/22bs283-repo$
```

Step 2: Creating `.gitlab-ci.yml`

- Added a `.gitlab-ci.yml` file to run Semgrep scans on the codebase.

File Content:

```yaml
stages:
  - test

semgrep-sast:
  stage: test
  script:
    - semgrep --config=auto --json > semgrep-report.json
  artifacts:
    paths:
      - semgrep-report.json
  rules:
    - if: $CI_COMMIT_REF_NAME == "main"
```

Step 3: Pushing Code and Testing CI

- Pushed the code to the repository and verified that the CI pipeline runs on every push to the `main` branch.

Evidence:

- CI Pipeline Execution:

semgrep-sast

Passed Started just now by Administrator

```
Search visible log output
1  Running with gitlab-runner 17.8.3 (690ce25c)
2    on 22BS283-runner t3_q4_sZy, system ID: s_5fb68c104fd4
3  Preparing the "shell" executor                                   00:00
4  Using Shell (bash) executor...
5  Preparing environment                                            00:00
6  Running on ip-172-31-47-104...
7  Getting source from Git repository                               00:00
8  Fetching changes with git depth set to 20...
9  Reinitialized existing Git repository in /home/ubuntu/builds/t3_q4_sZy/0/root/22bs283-repo/.git/
10 Checking out b36d40ad as detached HEAD (ref is main)...
11 Removing semgrep-report.json
12 Skipping Git submodules setup
13 Executing "step_script" stage of the job script                  00:11
14 $ semgrep --config=auto --json > semgrep-report.json
15 METRICS: Using configs from the Registry (like --config=p/ci) reports pseudonymous rule metrics to semgrep.dev.
16 To disable Registry rule metrics, use "--metrics=off".
17 Using configs only from local files (like --config=xyz.yml) does not enable metrics.
18 More information: https://semgrep.dev/docs/metrics
19
20
21
22 Scan Status
23
24   Scanning 189 files tracked by git with 1060 Code rules:
25
26   Language      Rules   Files          Origin      Rules
27   ──────────────────────────         ──────────────────────
28   <multilang>      63     348          Community    1060
29   java            118      21
30   yaml             31       2
31   dockerfile        5       1
32   json              4       1
33   bash              4       1
34   html              1       1
35
36
37
38
39 Scan Summary
40
41 Some files were skipped or only partially analyzed.
42   Partially scanned: 1 files only partially analyzed due to parsing or internal Semgrep errors
43   Scan skipped: 2 files matching .semgrepignore patterns
44   For a full list of skipped files, run semgrep with the --verbose flag.
45 Ran 222 rules on 174 files: 4 findings.
46 A new version of Semgrep is available. See https://semgrep.dev/docs/upgrading
47 Versions prior to 1.76.0 will cease to work soon, please upgrade.
48 Uploading artifacts for successful job                           00:00
49 Uploading artifacts...
50 Runtime platform                                    arch=amd64 os=linux pid=5442 revision=690ce25c version=17.8.3
51 semgrep-report.json: found 1 matching artifact files and directories
52 Uploading artifacts as "archive" to coordinator... 201 Created  id=9 responseStatus=201 Created token=glcbt-64
53 Cleaning up project directory and file based variables            00:00
54 Job succeeded
```

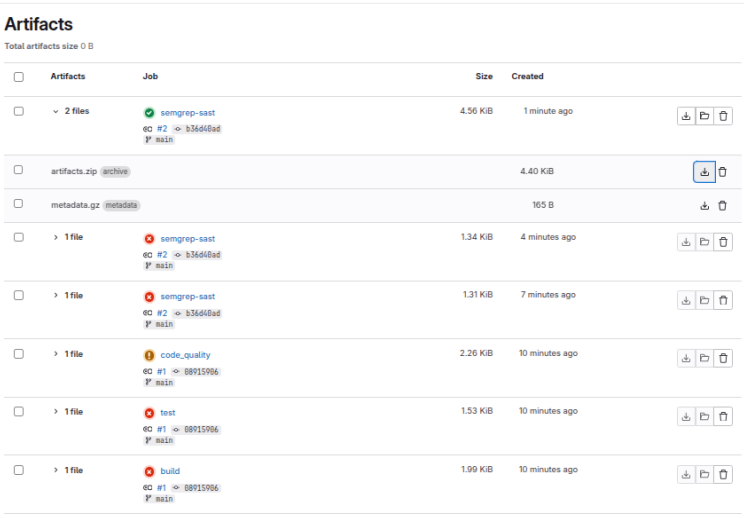
- CI Pipeline:

All 9 Finished

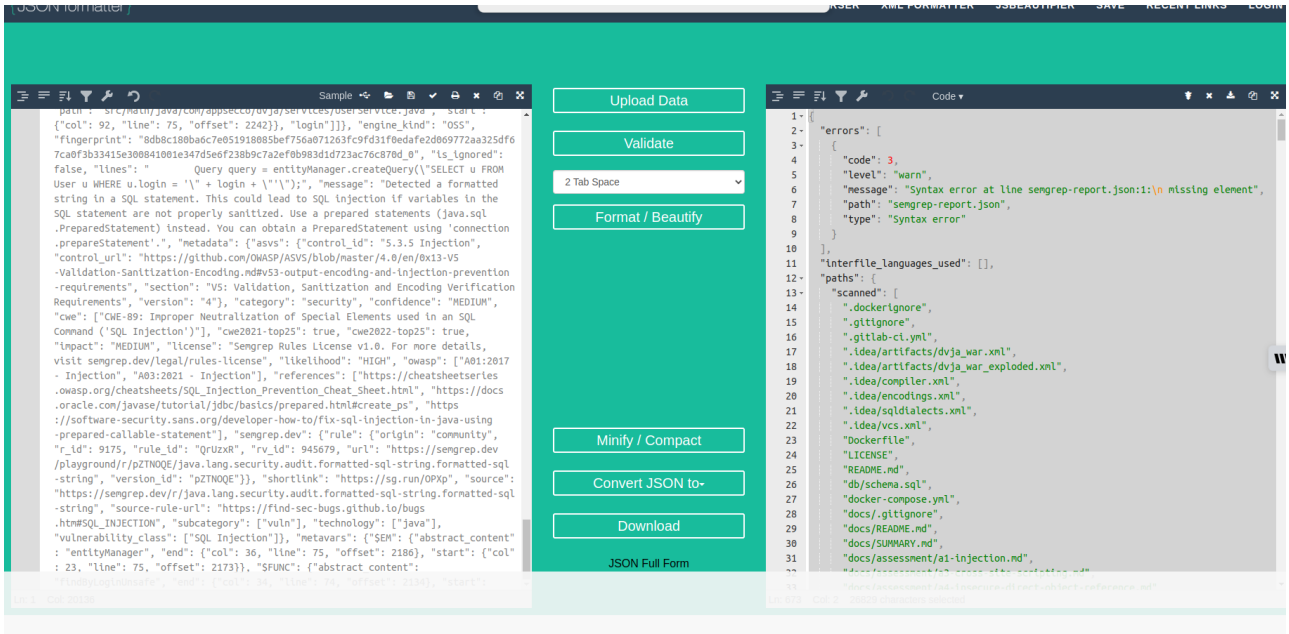
Filter jobs

Status	Job	Pipeline	Coverage
Passed 00:00:14 just now	#9: semgrep-sast main b36d40ad	#2 created by Stage: test	
Failed 3 minutes ago	#8: semgrep-sast main b36d40ad	#2 created by Stage: test	
Failed 6 minutes ago	#7: semgrep-sast main b36d40ad	#2 created by Stage: test	
Skipped	#6: secret_detection main 08915986 allowed to fail	#1 created by Stage: test	
Skipped	#5: semgrep-sast main 08915986 allowed to fail	#1 created by Stage: test	
Skipped	#4: container_scanning main 08915986 allowed to fail	#1 created by Stage: test	
Failed 9 minutes ago	#3: code_quality main 08915986 allowed to fail	#1 created by Stage: test	
Failed 9 minutes ago	#2: test main 08915986	#1 created by Stage: test	
Failed 00:00:01 9 minutes ago	#1: build main 08915986	#1 created by Stage: build	

- Artifacts:



- Formated artifact file:



Step 4: Analyzing the SAST Report

- Identified a SQL Injection vulnerability in `ProductService.java`.

Vulnerable Code:

```
Query query = entityManager.createQuery("SELECT p FROM Product p WHERE
p.name LIKE '%" + name + "%'");
```

Mitigation:

- Use parameterized queries to prevent SQL Injection.

Fixed Code:

```
String queryString = "SELECT p FROM Product p WHERE p.name LIKE :name";
Query query = entityManager.createQuery(queryString)
                           .setParameter("name", "%" + name + "%");
```