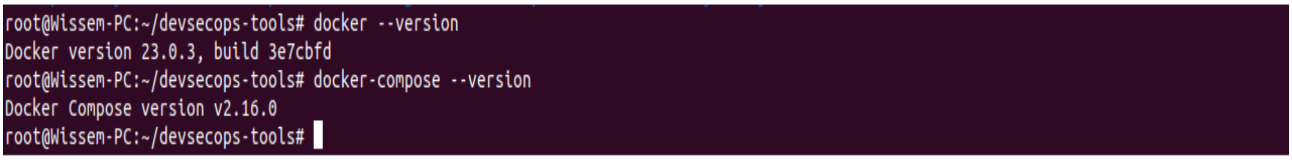
Steps for Using Docker Compose

1. **Install Docker and Docker Compose**

   Make sure Docker and Docker Compose are installed on your machine. You can check the installations using the following commands:

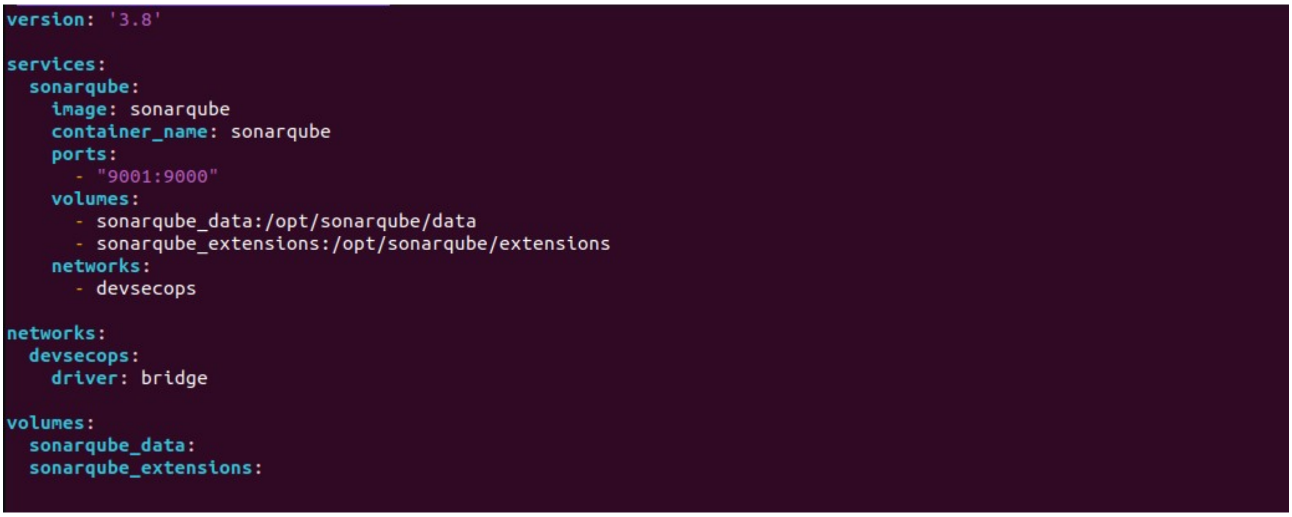
docker –version
docker-compose –version



```
root@Wissem-PC:~/devsecops-tools# docker --version
Docker version 23.0.3, build 3e7cbfd
root@Wissem-PC:~/devsecops-tools# docker-compose --version
Docker Compose version v2.16.0
root@Wissem-PC:~/devsecops-tools# 
```

Create the `docker-compose.yml` File

Create a file named `docker-compose.yml` in your desired directory. Paste the following content into this file:



```
version: '3.8'

services:
  sonarqube:
    image: sonarqube
    container_name: sonarqube
    ports:
      - "9001:9000"
    volumes:
      - sonarqube_data:/opt/sonarqube/data
      - sonarqube_extensions:/opt/sonarqube/extensions
    networks:
      - devsecops

networks:
  devsecops:
    driver: bridge

volumes:
  sonarqube_data:
  sonarqube_extensions:
```

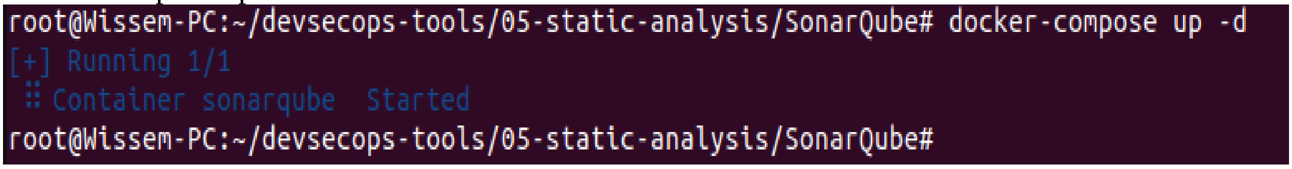
Explanations:

- `version: '3.8'`: Specifies the version of the Docker Compose file format.
- `services`: Defines the services to be deployed. Here, the `sonarqube` service is defined with:
  - `image`: Uses the official SonarQube Docker image.
  - `container_name`: Names the container for this service.
  - `ports`: Maps port 9000 of the container to port 9001 on the host.
  - `volumes`: Mounts volumes to persist data and extensions.
  - `networks`: Associates the service with the `devsecops` network.
- `networks`: Defines a network named `devsecops` using the `bridge` driver.
- `volumes`: Defines volumes `sonarqube_data` and `sonarqube_extensions` used to persist data.

Start the Services

Open a command line in the directory containing the `docker-compose.yml` file and run the following command to start the services:

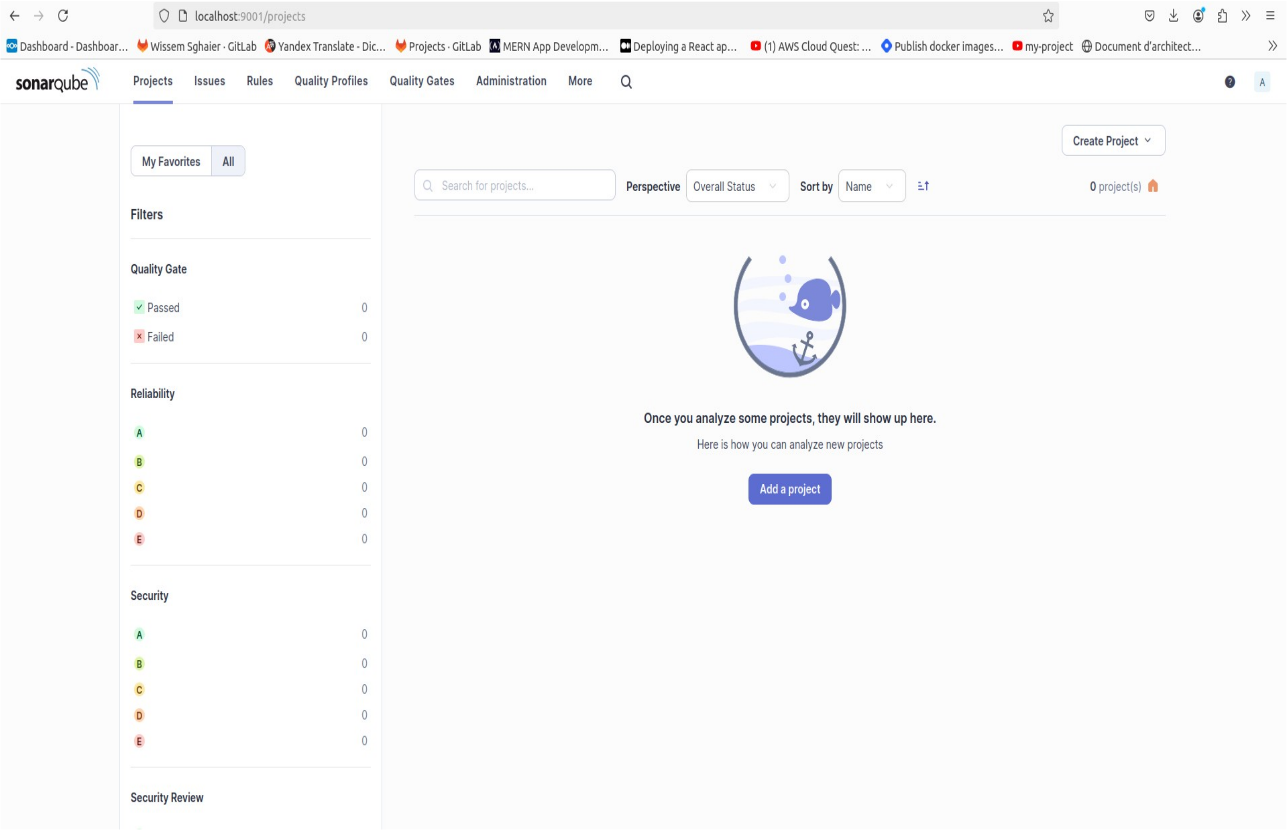
docker-compose up -d



```
root@Wissem-PC:~/devsecops-tools/05-static-analysis/SonarQube# docker-compose up -d
[+] Running 1/1
 ⠿ Container sonarqube  Started
root@Wissem-PC:~/devsecops-tools/05-static-analysis/SonarQube#
```

- This command lists the running containers along with their status.
- **Access the Application**

You can access SonarQube by opening a web browser and visiting http://localhost:9001. The port 9001 on your host is mapped to port 9000 on the SonarQube container.



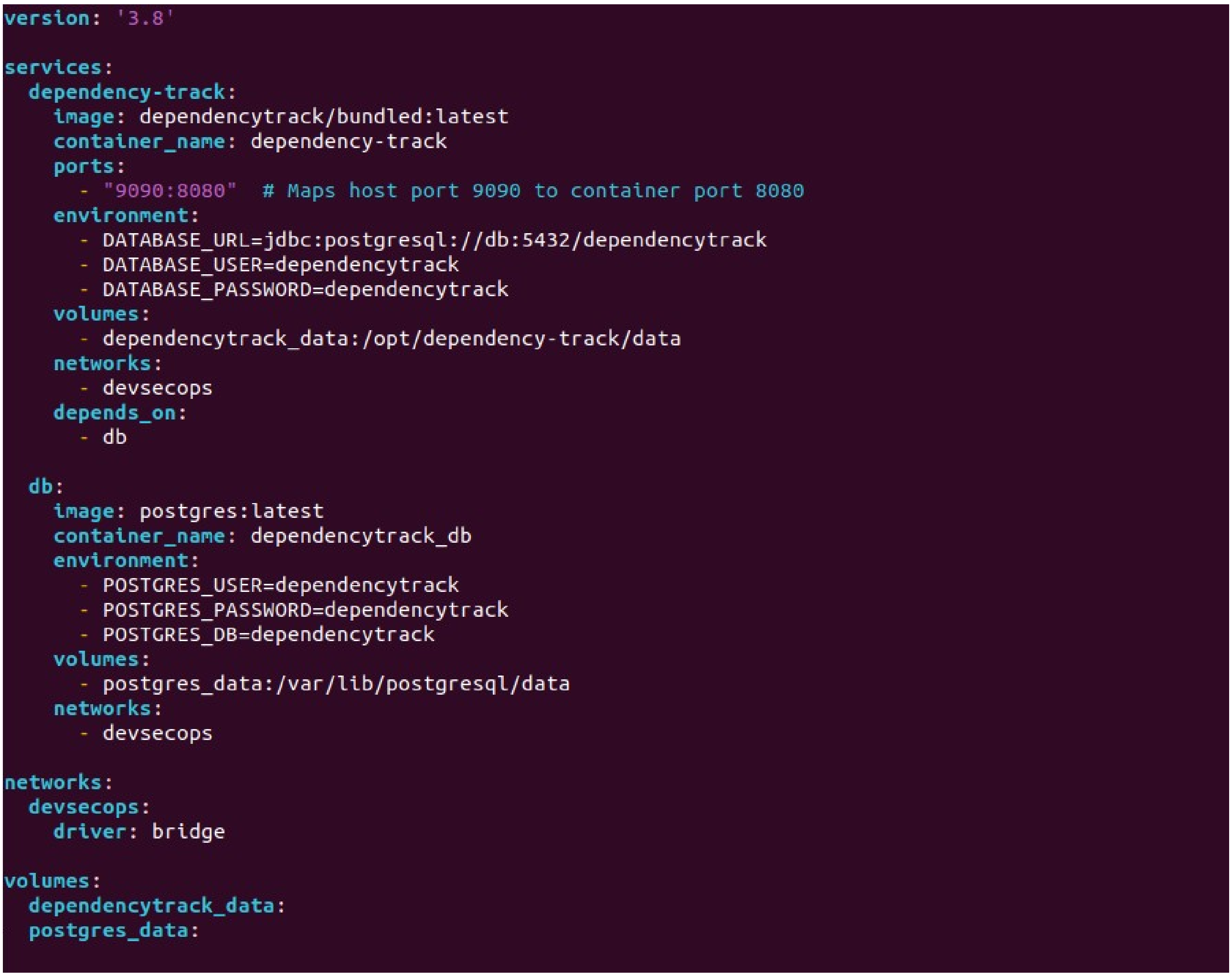
localhost:9001/projects

Once you analyze some projects, they will show up here.

Here is how you can analyze new projects

Create the `docker-compose.yml` File

Create a file named `docker-compose.yml` in your desired directory. Paste the following content into this file:



```yaml
version: '3.8'

services:
  dependency-track:
    image: dependencytrack/bundled:latest
    container_name: dependency-track
    ports:
      - "9090:8080"  # Maps host port 9090 to container port 8080
    environment:
      - DATABASE_URL=jdbc:postgresql://db:5432/dependencytrack
      - DATABASE_USER=dependencytrack
      - DATABASE_PASSWORD=dependencytrack
    volumes:
      - dependencytrack_data:/opt/dependency-track/data
    networks:
      - devsecops
    depends_on:
      - db

  db:
    image: postgres:latest
    container_name: dependencytrack_db
    environment:
      - POSTGRES_USER=dependencytrack
      - POSTGRES_PASSWORD=dependencytrack
      - POSTGRES_DB=dependencytrack
    volumes:
      - postgres_data:/var/lib/postgresql/data
    networks:
      - devsecops

networks:
  devsecops:
    driver: bridge

volumes:
  dependencytrack_data:
  postgres_data:
```

- **Explanations**:

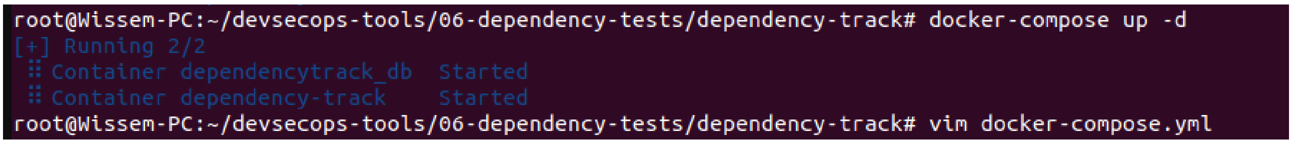
- `version: '3.8'`: Specifies the Docker Compose file format version.
- `services`: Defines the services to be deployed. Here, two services are defined:
    - `dependency-track`:
        - `image`: Uses the `dependencytrack/bundled:latest` Docker image.
        - `container_name`: Names the container `dependency-track`.
        - `ports`: Maps port 8080 of the container to port 9090 on the host.
        - `environment`: Sets environment variables for the service.
        - `volumes`: Mounts a volume to persist data.
        - `networks`: Associates the service with the `devsecops` network.
        - `depends_on`: Specifies that `dependency-track` depends on the `db` service to start first.
    - `db`:

    - `image`: Uses the `postgres:latest` Docker image.
    - `container_name`: Names the container `dependencytrack_db`.
    - `environment`: Sets environment variables for the PostgreSQL database.
    - `volumes`: Mounts a volume to persist database data.
    - `networks`: Associates the service with the `devsecops` network.
- `networks`: Defines a network named `devsecops` with the `bridge` driver.
- `volumes`: Defines volumes `dependencytrack_data` and `postgres_data` to persist data.

Start the Services

Open a command line in the directory containing the `docker-compose.yml` file and run the following command to start the services:

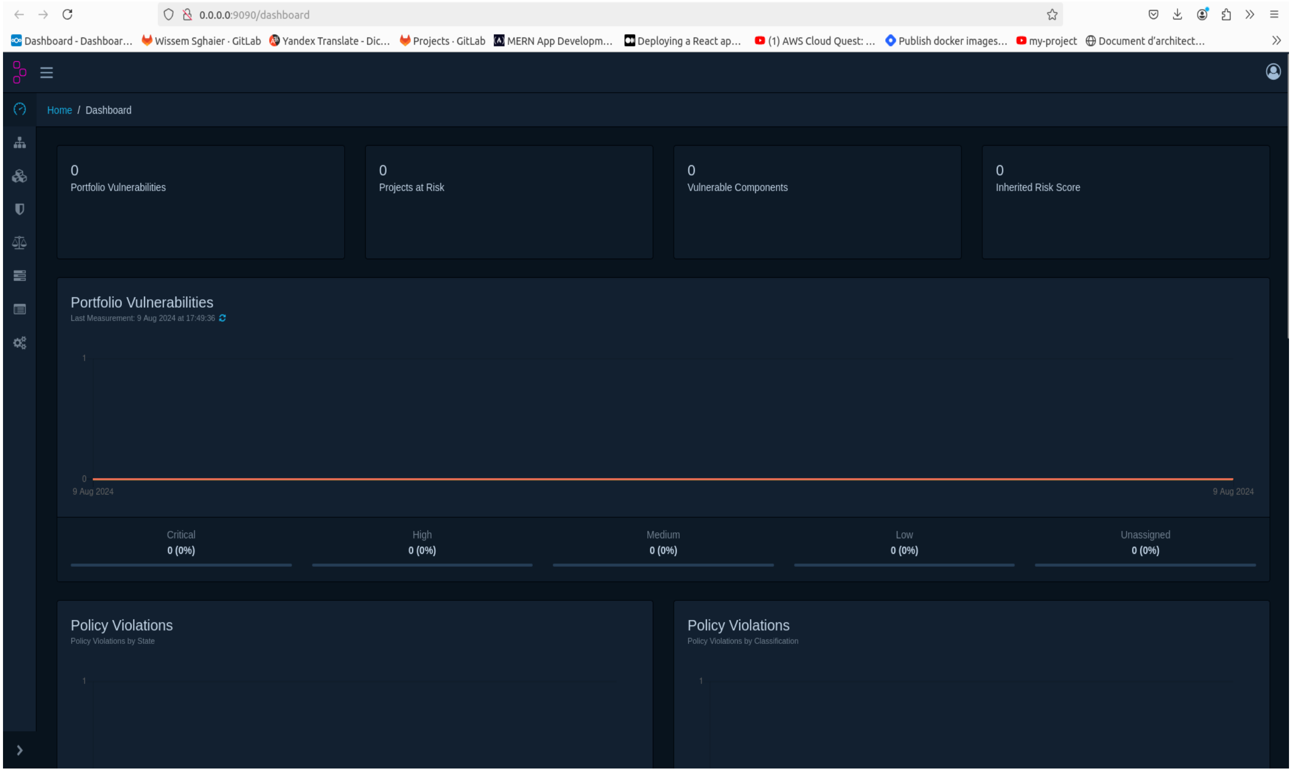
docker-compose up -d



```
root@Wissem-PC:~/devsecops-tools/06-dependency-tests/dependency-track# docker-compose up -d
[+] Running 2/2
 ⠿ Container dependencytrack_db  Started
 ⠿ Container dependency-track    Started
root@Wissem-PC:~/devsecops-tools/06-dependency-tests/dependency-track# vim docker-compose.yml
```

Access the Application

You can access SonarQube by opening a web browser and visiting http://localhost:9090/dashboard. The port 9090 on your host is mapped to port 9090 on the Dependency-track container.



0
Portfolio Vulnerabilities

0
Projects at Risk

0
Vulnerable Components

0
Inherited Risk Score

Portfolio Vulnerabilities
Last Measurement: 9 Aug 2024 at 17:49:36

Critical
0 (0%)

High
0 (0%)

Medium
0 (0%)

Low
0 (0%)

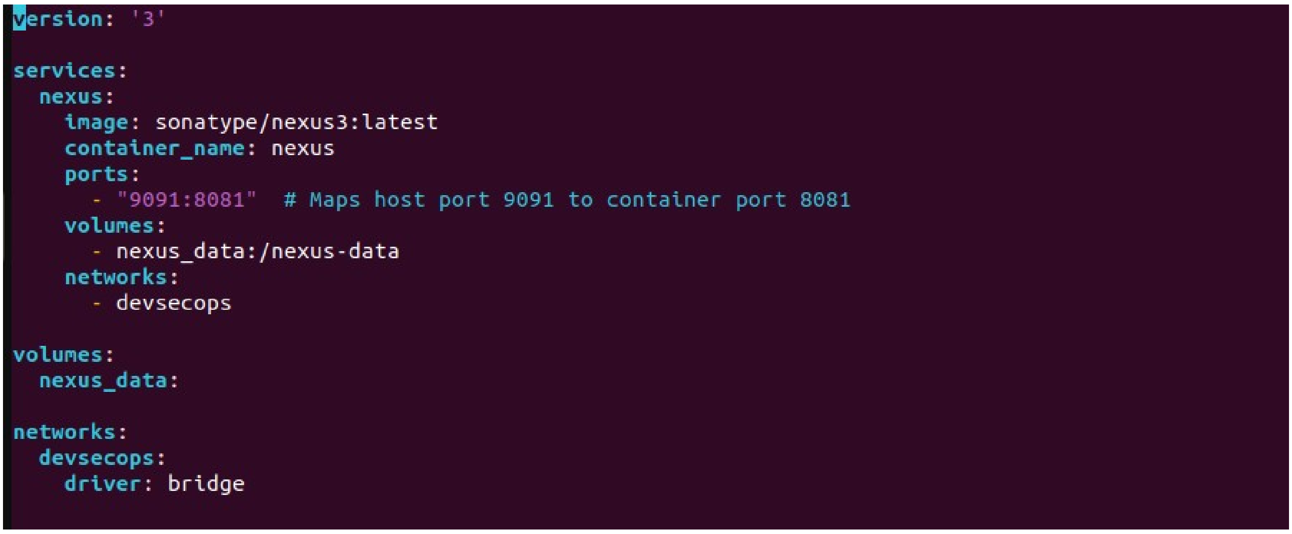
Unassigned
0 (0%)

Policy Violations
Policy Violations by State

Policy Violations
Policy Violations by Classification

Create the `docker-compose.yml` File

Create a file named `docker-compose.yml` in your desired directory. Paste the following content into this file:



```yaml
version: '3'

services:
  nexus:
    image: sonatype/nexus3:latest
    container_name: nexus
    ports:
      - "9091:8081"  # Maps host port 9091 to container port 8081
    volumes:
      - nexus_data:/nexus-data
    networks:
      - devsecops

volumes:
  nexus_data:

networks:
  devsecops:
    driver: bridge
```

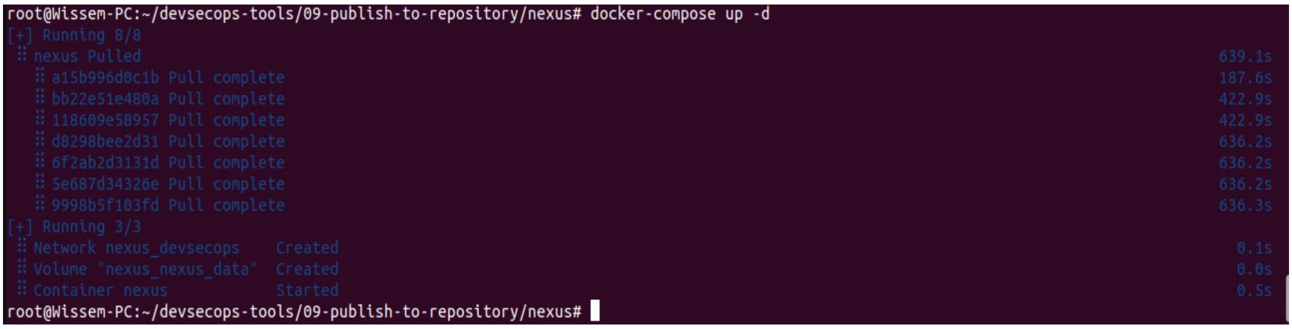
Explanations:

- `version: '3'`: Specifies the Docker Compose file format version.
- `services`: Defines the services to be deployed. Here, the `nexus` service is defined with:
  - `image`: Uses the `sonatype/nexus3:latest` Docker image for Nexus Repository.
  - `container_name`: Names the container `nexus`.
  - `ports`: Maps port 8081 of the container to port 9091 on the host. This allows you to access Nexus on port 9091 on your local machine.
  - `volumes`: Mounts a volume `nexus_data` to persist Nexus data.
  - `networks`: Associates the service with the `devsecops` network.
- `volumes`: Defines a volume named `nexus_data` for persisting Nexus data.
- `networks`: Defines a network named `devsecops` using the `bridge` driver.

- **Start the Services**

Open a command line in the directory containing the `docker-compose.yml` file and run the following command to start the services:

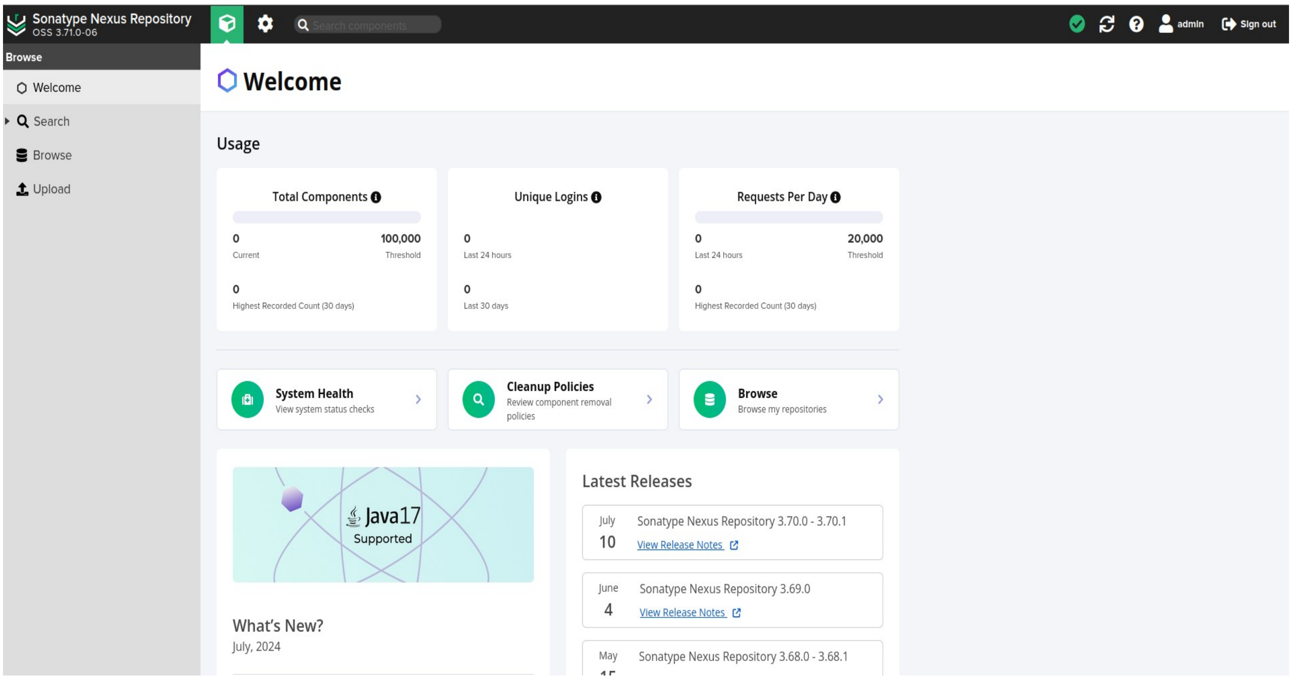
docker-compose up -d



```
root@Wissem-PC:~/devsecops-tools/09-publish-to-repository/nexus# docker-compose up -d
[+] Running 8/8
 ⠿ nexus Pulled                                                                639.1s
   ⠿ a15b996d0c1b Pull complete                                                187.6s
   ⠿ bb22e51e480a Pull complete                                                422.9s
   ⠿ 118609e58957 Pull complete                                                422.9s
   ⠿ d8298bee2d31 Pull complete                                                636.2s
   ⠿ 6f2ab2d3131d Pull complete                                                636.2s
   ⠿ 5e687d34326e Pull complete                                                636.2s
   ⠿ 9998b5f103fd Pull complete                                                636.3s
[+] Running 3/3
 ⠿ Network nexus_devsecops    Created                                            0.1s
 ⠿ Volume "nexus_nexus_data"  Created                                            0.0s
 ⠿ Container nexus            Started                                            0.5s
root@Wissem-PC:~/devsecops-tools/09-publish-to-repository/nexus#
```

Access the Application

You can access the Nexus Repository Manager by opening a web browser and visiting http://localhost:9091. Port 9091 on your host is mapped to port 8081 on the Nexus container.



Welcome

Usage

Total Components
0 Current
100,000 Threshold
0 Highest Recorded Count (30 days)

Unique Logins
0 Last 24 hours
0 Last 30 days

Requests Per Day
0 Last 24 hours
20,000 Threshold
0 Highest Recorded Count (30 days)

System Health
View system status checks

Cleanup Policies
Review component removal policies

Browse
Browse my repositories

Java17
Supported

What's New?
July, 2024

Latest Releases

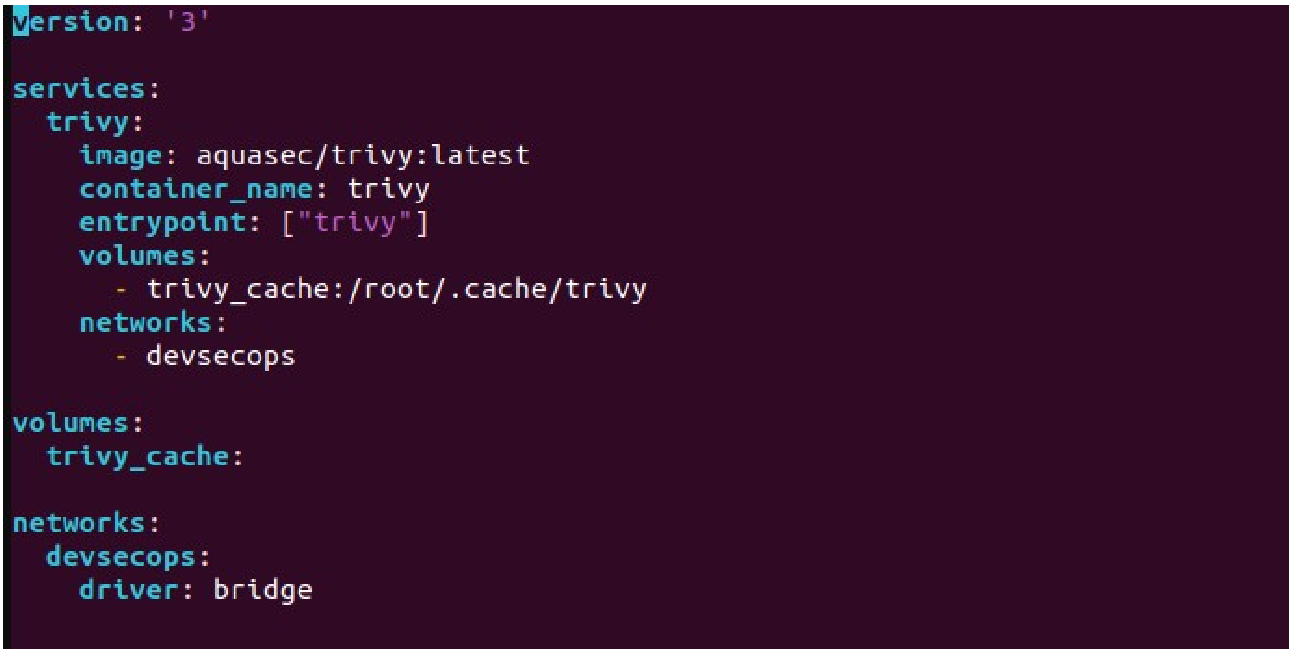
July 10 — Sonatype Nexus Repository 3.70.0 - 3.70.1
View Release Notes

June 4 — Sonatype Nexus Repository 3.69.0
View Release Notes

May — Sonatype Nexus Repository 3.68.0 - 3.68.1

Create the `docker-compose.yml` File

Create a file named `docker-compose.yml` in your desired directory. Paste the following content into this file:



```
version: '3'

services:
  trivy:
    image: aquasec/trivy:latest
    container_name: trivy
    entrypoint: ["trivy"]
    volumes:
      - trivy_cache:/root/.cache/trivy
    networks:
      - devsecops

volumes:
  trivy_cache:

networks:
  devsecops:
    driver: bridge
```

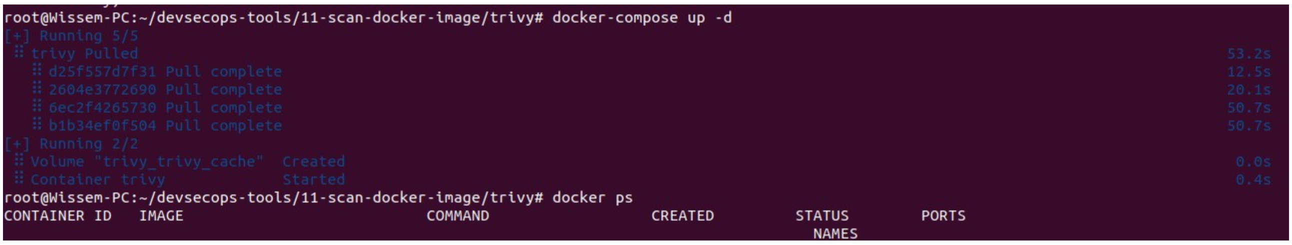
Explanations:

- `version: '3'`: Specifies the version of the Docker Compose file format.
- `services`: Defines the services to be deployed. Here, the `trivy` service is defined with:
  - `image`: Uses the `aquasec/trivy:latest` Docker image for Trivy, a vulnerability scanner.
  - `container_name`: Names the container `trivy`.
  - `entrypoint`: Sets the entrypoint for the container to the `trivy` command, allowing you to specify the command to run when the container starts.
  - `volumes`: Mounts a volume `trivy_cache` to persist Trivy cache data, which can speed up subsequent scans.
  - `networks`: Associates the service with the `devsecops` network.
- `volumes`: Defines a volume named `trivy_cache` for persisting cache data.
- `networks`: Defines a network named `devsecops` using the `bridge` driver.

Start the Services

Open a command line in the directory containing the `docker-compose.yml` file and run the following command to start the services:

docker-compose up -d

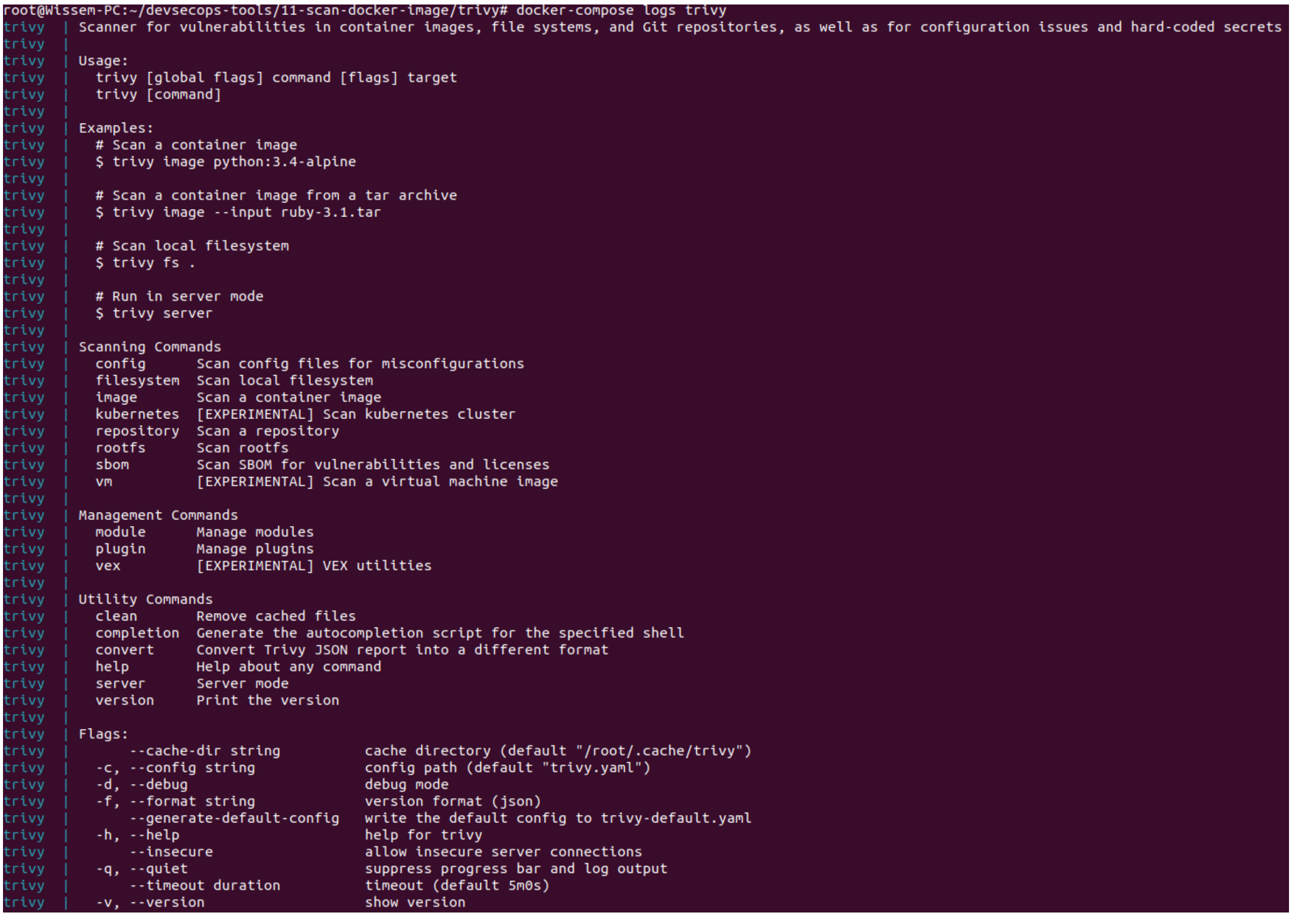


```
root@Wissem-PC:~/devsecops-tools/11-scan-docker-image/trivy# docker-compose up -d
[+] Running 5/5
 ⠿ trivy Pulled                                                                                   53.2s
   ⠿ d25f557d7f31 Pull complete                                                                   12.5s
   ⠿ 2604e3772690 Pull complete                                                                   20.1s
   ⠿ 6ec2f4265730 Pull complete                                                                   50.7s
   ⠿ b1b34ef0f504 Pull complete                                                                   50.7s
[+] Running 2/2
 ⠿ Volume "trivy_trivy_cache"  Created                                                             0.0s
 ⠿ Container trivy             Started                                                             0.4s
root@Wissem-PC:~/devsecops-tools/11-scan-docker-image/trivy# docker ps
CONTAINER ID   IMAGE                              COMMAND                  CREATED         STATUS         PORTS
                                                                                                 NAMES
```

Using Trivy

Once the Trivy container is running, you can execute commands in it using `docker exec`. For example, to scan a Docker image for vulnerabilities, use:

docker-compose logs trivy

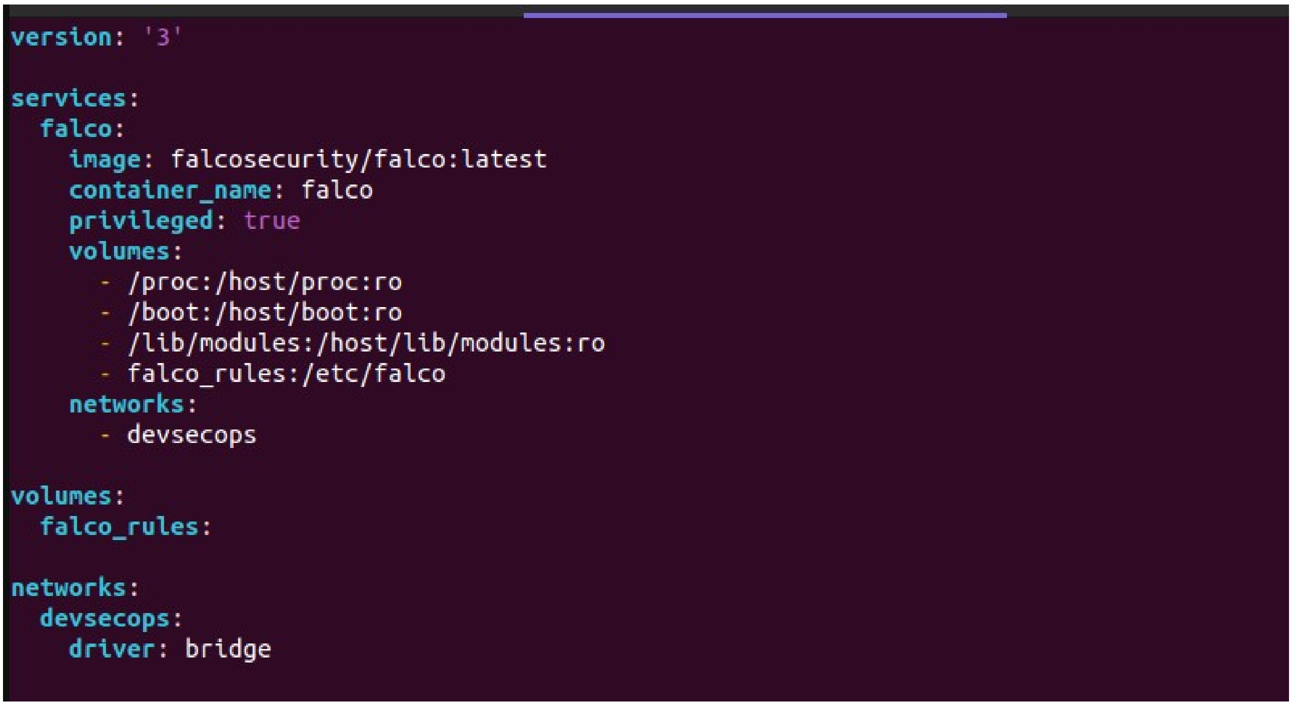


```
root@Wissem-PC:~/devsecops-tools/11-scan-docker-image/trivy# docker-compose logs trivy
trivy  | Scanner for vulnerabilities in container images, file systems, and Git repositories, as well as for configuration issues and hard-coded secrets
trivy  |
trivy  | Usage:
trivy  |   trivy [global flags] command [flags] target
trivy  |   trivy [command]
trivy  |
trivy  | Examples:
trivy  |   # Scan a container image
trivy  |   $ trivy image python:3.4-alpine
trivy  |
trivy  |   # Scan a container image from a tar archive
trivy  |   $ trivy image --input ruby-3.1.tar
trivy  |
trivy  |   # Scan local filesystem
trivy  |   $ trivy fs .
trivy  |
trivy  |   # Run in server mode
trivy  |   $ trivy server
trivy  |
trivy  | Scanning Commands
trivy  |   config      Scan config files for misconfigurations
trivy  |   filesystem  Scan local filesystem
trivy  |   image       Scan a container image
trivy  |   kubernetes  [EXPERIMENTAL] Scan kubernetes cluster
trivy  |   repository  Scan a repository
trivy  |   rootfs      Scan rootfs
trivy  |   sbom        Scan SBOM for vulnerabilities and licenses
trivy  |   vm          [EXPERIMENTAL] Scan a virtual machine image
trivy  |
trivy  | Management Commands
trivy  |   module      Manage modules
trivy  |   plugin      Manage plugins
trivy  |   vex         [EXPERIMENTAL] VEX utilities
trivy  |
trivy  | Utility Commands
trivy  |   clean       Remove cached files
trivy  |   completion  Generate the autocompletion script for the specified shell
trivy  |   convert     Convert Trivy JSON report into a different format
trivy  |   help        Help about any command
trivy  |   server      Server mode
trivy  |   version     Print the version
trivy  |
trivy  | Flags:
trivy  |       --cache-dir string          cache directory (default "/root/.cache/trivy")
trivy  |   -c, --config string             config path (default "trivy.yaml")
trivy  |   -d, --debug                     debug mode
trivy  |   -f, --format string             version format (json)
trivy  |       --generate-default-config   write the default config to trivy-default.yaml
trivy  |   -h, --help                      help for trivy
trivy  |       --insecure                  allow insecure server connections
trivy  |   -q, --quiet                     suppress progress bar and log output
trivy  |       --timeout duration          timeout (default 5m0s)
trivy  |   -v, --version                   show version
```

docker exec trivy trivy image your-image:tag

Create the `docker-compose.yml` File

Create a file named `docker-compose.yml` in your desired directory. Paste the following content into this file:



```
version: '3'

services:
  falco:
    image: falcosecurity/falco:latest
    container_name: falco
    privileged: true
    volumes:
      - /proc:/host/proc:ro
      - /boot:/host/boot:ro
      - /lib/modules:/host/lib/modules:ro
      - falco_rules:/etc/falco
    networks:
      - devsecops

volumes:
  falco_rules:

networks:
  devsecops:
    driver: bridge
```

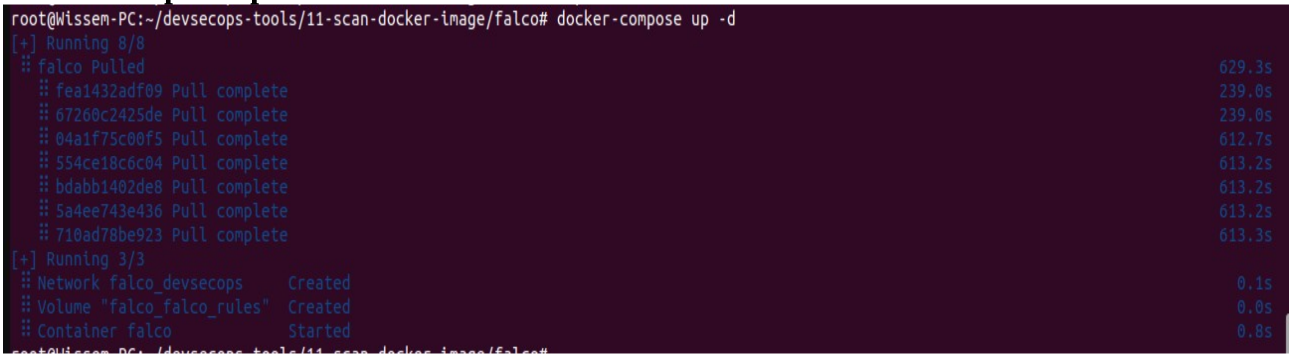
Explanations:

- `version: '3'`: Specifies the Docker Compose file format version.
- `services`: Defines the services to be deployed. Here, the `falco` service is defined with:
  - `image`: Uses the `falcosecurity/falco:latest` Docker image for Falco.
  - `container_name`: Names the container `falco`.
  - `privileged: true`: Grants the container elevated privileges, which is necessary for Falco to access kernel and system data.
  - `volumes`: Mounts directories and volumes into the container:
    - `/proc:/host/proc:ro`: Mounts the host's `/proc` directory as read-only inside the container.
    - `/boot:/host/boot:ro`: Mounts the host's `/boot` directory as read-only inside the container.
    - `/lib/modules:/host/lib/modules:ro`: Mounts the host's `/lib/modules` directory as read-only inside the container.
    - `falco_rules:/etc/falco`: Mounts a named volume for Falco rules.
  - `networks`: Associates the service with the `devsecops` network.
- `volumes`: Defines a volume named `falco_rules` for Falco rule configurations.
- `networks`: Defines a network named `devsecops` using the `bridge` driver.

Start the Services

Open a command line in the directory containing the `docker-compose.yml` file and run the following command to start the services:

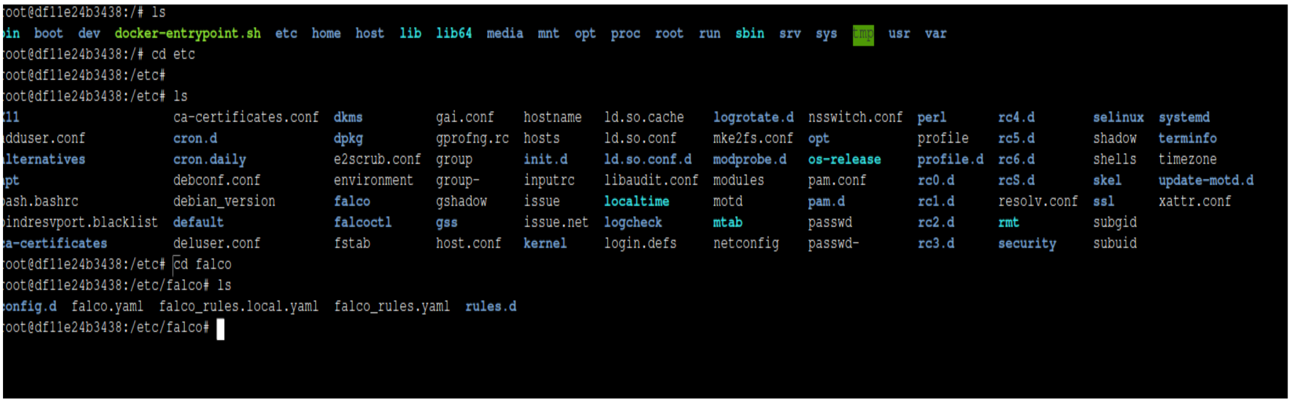
docker-compose up -d



```
root@Wissem-PC:~/devsecops-tools/11-scan-docker-image/falco# docker-compose up -d
[+] Running 8/8
 ⠿ falco Pulled                                                                          629.3s
   ⠿ fea1432adf09 Pull complete                                                          239.0s
   ⠿ 67260c2425de Pull complete                                                          239.0s
   ⠿ 04a1f75c00f5 Pull complete                                                          612.7s
   ⠿ 554ce18c6c04 Pull complete                                                          613.2s
   ⠿ bdabb1402de8 Pull complete                                                          613.2s
   ⠿ 5a4ee743e436 Pull complete                                                          613.2s
   ⠿ 710ad78be923 Pull complete                                                          613.3s
[+] Running 3/3
 ⠿ Network falco_devsecops      Created                                                    0.1s
 ⠿ Volume "falco_falco_rules"   Created                                                    0.0s
 ⠿ Container falco              Started                                                    0.8s
root@Wissem-PC:~/devsecops-tools/11-scan-docker-image/falco#
```

Configure Falco Rules

Falco uses rules to detect suspicious activity. The rules are typically placed in a configuration file. Since you are mounting a volume `falco_rules` to `/etc/falco`, you should ensure that the Falco rules are available in this volume. You can modify or add rules to the `falco_rules` volume based on your requirements.

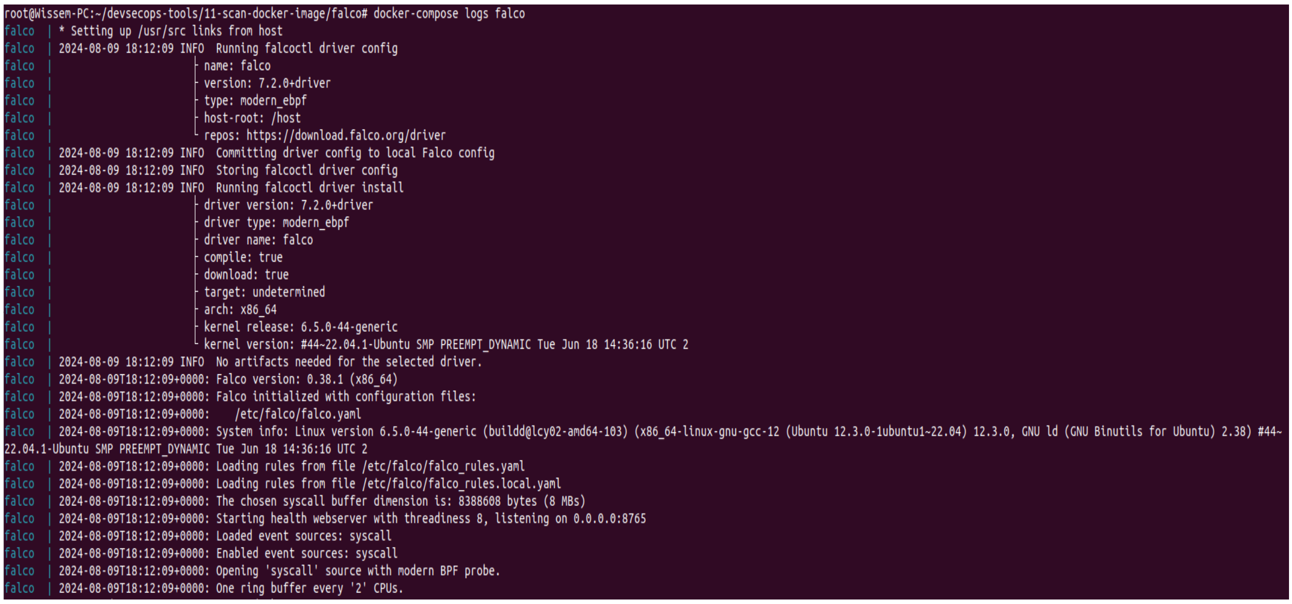


```
oot@df11e24b3438:/# ls
in  boot  dev  docker-entrypoint.sh  etc  home  host  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
oot@df11e24b3438:/# cd etc
oot@df11e24b3438:/etc#
oot@df11e24b3438:/etc# ls
11                     ca-certificates.conf  dkms         gai.conf     hostname   ld.so.cache     logrotate.d  nsswitch.conf  perl       rc4.d        selinux  systemd
dduser.conf            cron.d                dpkg         gprofng.rc   hosts      ld.so.conf      mke2fs.conf  opt            profile    rc5.d        shadow   terminfo
lternatives            cron.daily            e2scrub.conf group        init.d     ld.so.conf.d    modprobe.d   os-release     profile.d  rc6.d        shells   timezone
pt                     debconf.conf          environment  group-       inputrc    libaudit.conf   modules      pam.conf       rc0.d      rcS.d        skel     update-motd.d
ash.bashrc             debian_version        falco        gshadow      issue      localtime       motd         pam.d          rc1.d      resolv.conf  ssl      xattr.conf
indresvport.blacklist  default               falcoctl     gss          issue.net  logcheck        mtab         passwd         rc2.d      rmt          subgid
a-certificates         deluser.conf          fstab        host.conf    kernel     login.defs      netconfig    passwd-        rc3.d      security     subuid
oot@df11e24b3438:/etc# cd falco
oot@df11e24b3438:/etc/falco# ls
onfig.d  falco.yaml  falco_rules.local.yaml  falco_rules.yaml  rules.d
oot@df11e24b3438:/etc/falco# 
```

- **View Falco Logs**

You can view the logs of the Falco container to ensure it is running correctly and to see any potential errors or output:

docker-compose logs falco



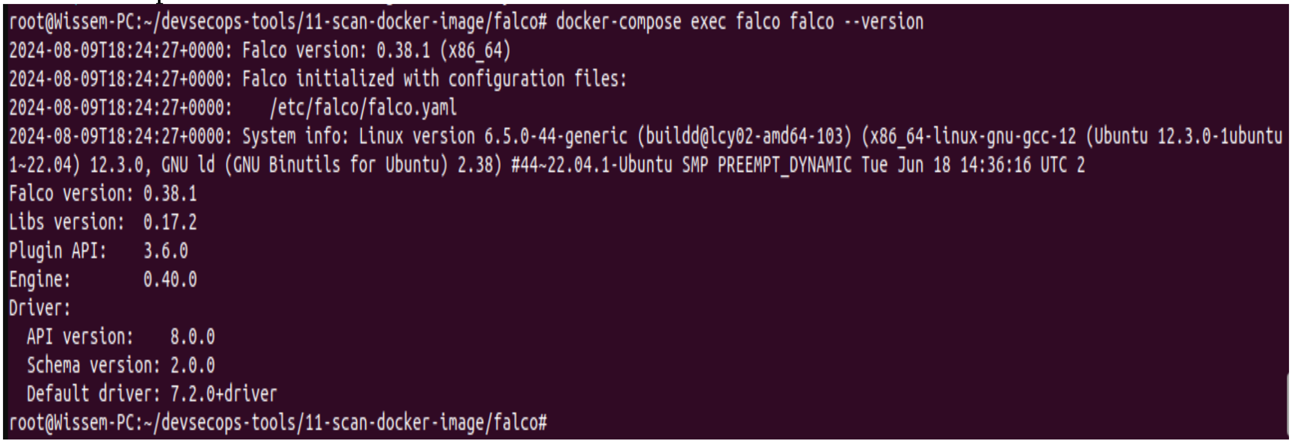
```
root@Wissem-PC:~/devsecops-tools/11-scan-docker-image/falco# docker-compose logs falco
falco  | * Setting up /usr/src links from host
falco  | 2024-08-09 18:12:09 INFO  Running falcoctl driver config
falco  |                          ├ name: falco
falco  |                          ├ version: 7.2.0+driver
falco  |                          ├ type: modern_ebpf
falco  |                          ├ host-root: /host
falco  |                          └ repos: https://download.falco.org/driver
falco  | 2024-08-09 18:12:09 INFO  Committing driver config to local Falco config
falco  | 2024-08-09 18:12:09 INFO  Storing falcoctl driver config
falco  | 2024-08-09 18:12:09 INFO  Running falcoctl driver install
falco  |                          ├ driver version: 7.2.0+driver
falco  |                          ├ driver type: modern_ebpf
falco  |                          ├ driver name: falco
falco  |                          ├ compile: true
falco  |                          ├ download: true
falco  |                          ├ target: undetermined
falco  |                          ├ arch: x86_64
falco  |                          ├ kernel release: 6.5.0-44-generic
falco  |                          └ kernel version: #44~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Tue Jun 18 14:36:16 UTC 2
falco  | 2024-08-09 18:12:09 INFO  No artifacts needed for the selected driver.
falco  | 2024-08-09T18:12:09+0000: Falco version: 0.38.1 (x86_64)
falco  | 2024-08-09T18:12:09+0000: Falco initialized with configuration files:
falco  | 2024-08-09T18:12:09+0000:    /etc/falco/falco.yaml
falco  | 2024-08-09T18:12:09+0000: System info: Linux version 6.5.0-44-generic (buildd@lcy02-amd64-103) (x86_64-linux-gnu-gcc-12 (Ubuntu 12.3.0-1ubuntu1~22.04) 12.3.0, GNU ld (GNU Binutils for Ubuntu) 2.38) #44~
22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Tue Jun 18 14:36:16 UTC 2
falco  | 2024-08-09T18:12:09+0000: Loading rules from file /etc/falco/falco_rules.yaml
falco  | 2024-08-09T18:12:09+0000: Loading rules from file /etc/falco/falco_rules.local.yaml
falco  | 2024-08-09T18:12:09+0000: The chosen syscall buffer dimension is: 8388608 bytes (8 MBs)
falco  | 2024-08-09T18:12:09+0000: Starting health webserver with threadiness 8, listening on 0.0.0.0:8765
falco  | 2024-08-09T18:12:09+0000: Loaded event sources: syscall
falco  | 2024-08-09T18:12:09+0000: Enabled event sources: syscall
falco  | 2024-08-09T18:12:09+0000: Opening 'syscall' source with modern BPF probe.
falco  | 2024-08-09T18:12:09+0000: One ring buffer every '2' CPUs.
```

This will display the logs for the `falco` service. Look for any startup messages or errors that might indicate issues.

Check Container Health

If you want to perform a more automated health check, you can create a custom script that tests basic functionality of Falco. For example:

docker-compose exec falco falco –version

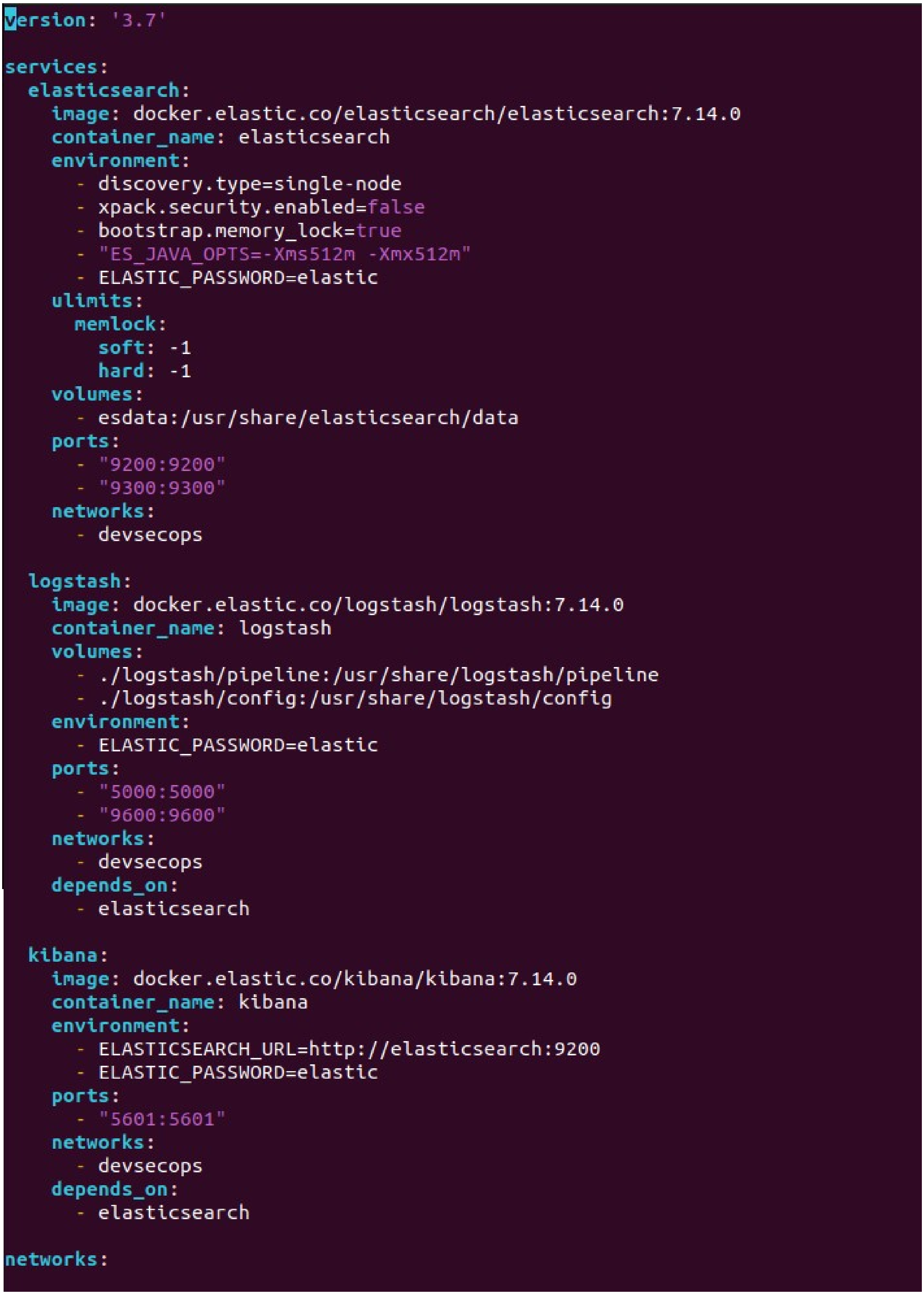


```
root@Wissem-PC:~/devsecops-tools/11-scan-docker-image/falco# docker-compose exec falco falco --version
2024-08-09T18:24:27+0000: Falco version: 0.38.1 (x86_64)
2024-08-09T18:24:27+0000: Falco initialized with configuration files:
2024-08-09T18:24:27+0000:    /etc/falco/falco.yaml
2024-08-09T18:24:27+0000: System info: Linux version 6.5.0-44-generic (buildd@lcy02-amd64-103) (x86_64-linux-gnu-gcc-12 (Ubuntu 12.3.0-1ubuntu
1~22.04) 12.3.0, GNU ld (GNU Binutils for Ubuntu) 2.38) #44~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Tue Jun 18 14:36:16 UTC 2
Falco version: 0.38.1
Libs version:  0.17.2
Plugin API:    3.6.0
Engine:        0.40.0
Driver:
  API version:    8.0.0
  Schema version: 2.0.0
  Default driver: 7.2.0+driver
root@Wissem-PC:~/devsecops-tools/11-scan-docker-image/falco#
```

This command will return the version of Falco, confirming that Falco is installed and accessible within the container.

Create the `docker-compose.yml` File

Create a file named `docker-compose.yml` in your desired directory. Paste the following content into this file:



```yaml
version: '3.7'

services:
  elasticsearch:
    image: docker.elastic.co/elasticsearch/elasticsearch:7.14.0
    container_name: elasticsearch
    environment:
      - discovery.type=single-node
      - xpack.security.enabled=false
      - bootstrap.memory_lock=true
      - "ES_JAVA_OPTS=-Xms512m -Xmx512m"
      - ELASTIC_PASSWORD=elastic
    ulimits:
      memlock:
        soft: -1
        hard: -1
    volumes:
      - esdata:/usr/share/elasticsearch/data
    ports:
      - "9200:9200"
      - "9300:9300"
    networks:
      - devsecops

  logstash:
    image: docker.elastic.co/logstash/logstash:7.14.0
    container_name: logstash
    volumes:
      - ./logstash/pipeline:/usr/share/logstash/pipeline
      - ./logstash/config:/usr/share/logstash/config
    environment:
      - ELASTIC_PASSWORD=elastic
    ports:
      - "5000:5000"
      - "9600:9600"
    networks:
      - devsecops
    depends_on:
      - elasticsearch

  kibana:
    image: docker.elastic.co/kibana/kibana:7.14.0
    container_name: kibana
    environment:
      - ELASTICSEARCH_URL=http://elasticsearch:9200
      - ELASTIC_PASSWORD=elastic
    ports:
      - "5601:5601"
    networks:
      - devsecops
    depends_on:
      - elasticsearch

networks:
```

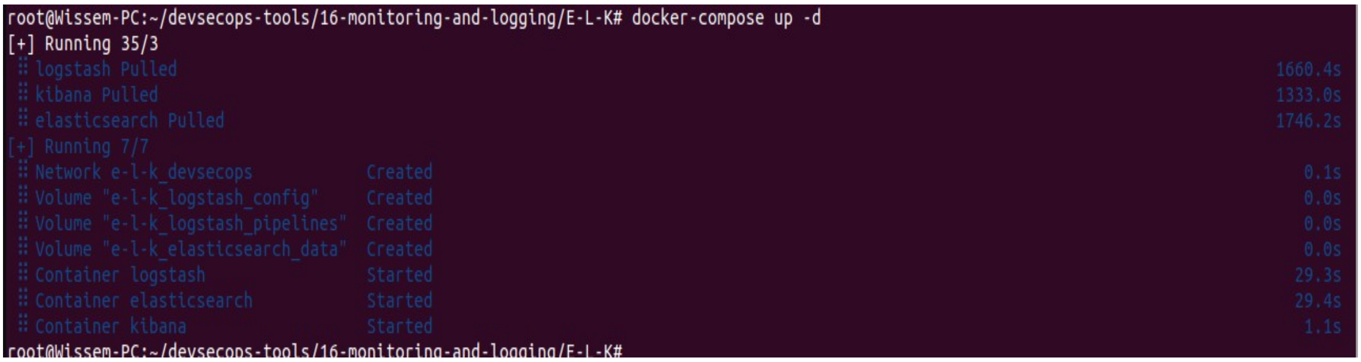
2. Explanation of the Docker Compose File

- **`version: '3.7'`**: Specifies the version of the Docker Compose file format.
- **`services`**: Defines the services that will be run. Each service corresponds to a container.
    - **`elasticsearch`**:
        - **`image`**: Specifies the Docker image to use.
        - **`container_name`**: The name of the container.
        - **`environment`**: Sets environment variables inside the container.
        - **`ulimits`**: Configures resource limits for the container.
        - **`volumes`**: Mounts volumes to persist data.
        - **`ports`**: Maps container ports to host ports.
        - **`networks`**: Connects the container to a specific network.
    - **`logstash`**:
        - **`image`**: Specifies the Docker image to use.
        - **`container_name`**: The name of the container.
        - **`volumes`**: Mounts configuration files and pipelines from the host to the container.
        - **`environment`**: Sets environment variables inside the container.
        - **`ports`**: Maps container ports to host ports.
        - **`depends_on`**: Ensures that `elasticsearch` starts before `logstash`.
    - **`kibana`**:
        - **`image`**: Specifies the Docker image to use.
        - **`container_name`**: The name of the container.
        - **`environment`**: Sets environment variables inside the container.
        - **`ports`**: Maps container ports to host ports.
        - **`depends_on`**: Ensures that `elasticsearch` starts before `kibana`.
- **`networks`**: Defines the network configuration.
    - **`devsecops`**: Creates a bridge network named `devsecops`.
- **`volumes`**: Defines named volumes to persist data.
    - **`esdata`**: A volume used by Elasticsearch to store data.

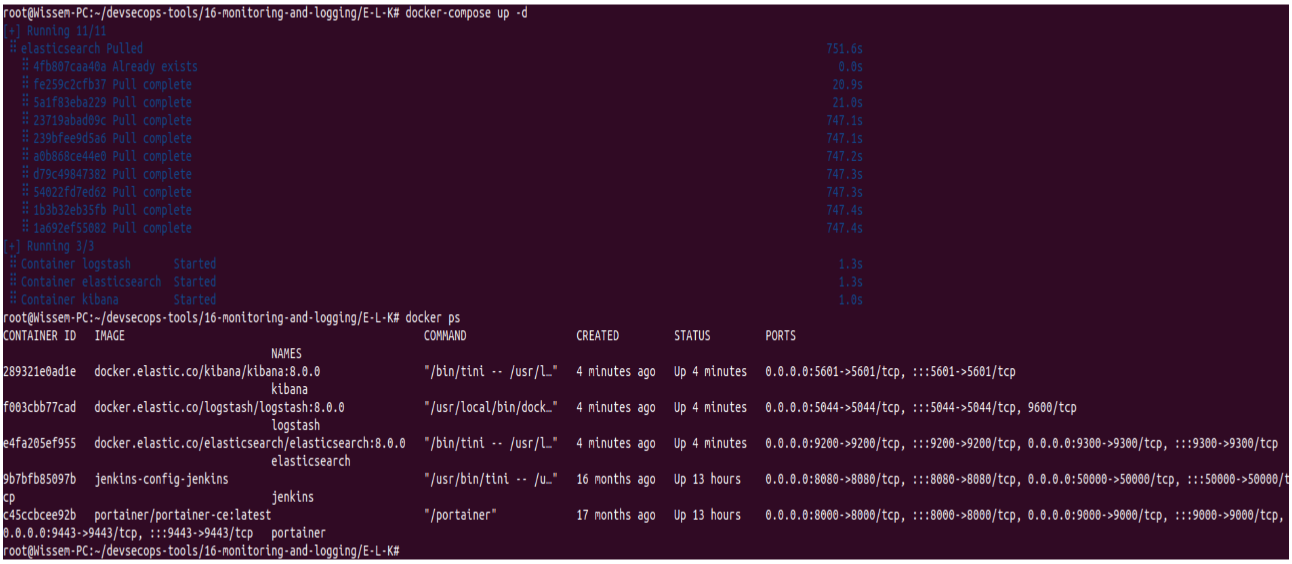
Start the Services

Open a command line in the directory containing the `docker-compose.yml` file and run the following command to start the services:

docker-compose up -d



```
root@Wissem-PC:~/devsecops-tools/16-monitoring-and-logging/E-L-K# docker-compose up -d
[+] Running 35/3
 ⠿ logstash Pulled                                                                   1660.4s
 ⠿ kibana Pulled                                                                     1333.0s
 ⠿ elasticsearch Pulled                                                              1746.2s
[+] Running 7/7
 ⠿ Network e-l-k_devsecops           Created                                            0.1s
 ⠿ Volume "e-l-k_logstash_config"    Created                                            0.0s
 ⠿ Volume "e-l-k_logstash_pipelines" Created                                            0.0s
 ⠿ Volume "e-l-k_elasticsearch_data" Created                                            0.0s
 ⠿ Container logstash                Started                                           29.3s
 ⠿ Container elasticsearch           Started                                           29.4s
 ⠿ Container kibana                  Started                                            1.1s
root@Wissem-PC:~/devsecops-tools/16-monitoring-and-logging/E-L-K#
```



```
root@Wissem-PC:~/devsecops-tools/16-monitoring-and-logging/E-L-K# docker-compose up -d
[+] Running 11/11
 ⠿ elasticsearch Pulled                                                          751.6s
   ⠿ 4fb807caa40a Already exists                                                   0.0s
   ⠿ fe259c2cfb37 Pull complete                                                   20.9s
   ⠿ 5a1f83eba229 Pull complete                                                   21.0s
   ⠿ 23719abad09c Pull complete                                                  747.1s
   ⠿ 239bfee9d5a6 Pull complete                                                  747.1s
   ⠿ a0b868ce44e0 Pull complete                                                  747.2s
   ⠿ d79c49847382 Pull complete                                                  747.3s
   ⠿ 54022fd7ed62 Pull complete                                                  747.3s
   ⠿ 1b3b32eb35fb Pull complete                                                  747.4s
   ⠿ 1a692ef55082 Pull complete                                                  747.4s
[+] Running 3/3
 ⠿ Container logstash       Started                                                1.3s
 ⠿ Container elasticsearch  Started                                                1.3s
 ⠿ Container kibana         Started                                                1.0s
root@Wissem-PC:~/devsecops-tools/16-monitoring-and-logging/E-L-K# docker ps
CONTAINER ID   IMAGE                                                  COMMAND                  CREATED          STATUS          PORTS
                                        NAMES
289321e0ad1e   docker.elastic.co/kibana/kibana:8.0.0                  "/bin/tini -- /usr/l…"   4 minutes ago    Up 4 minutes    0.0.0.0:5601->5601/tcp, :::5601->5601/tcp
                                        kibana
f003cbb77cad   docker.elastic.co/logstash/logstash:8.0.0              "/usr/local/bin/dock…"   4 minutes ago    Up 4 minutes    0.0.0.0:5044->5044/tcp, :::5044->5044/tcp, 9600/tcp
                                        logstash
e4fa205ef955   docker.elastic.co/elasticsearch/elasticsearch:8.0.0    "/bin/tini -- /usr/l…"   4 minutes ago    Up 4 minutes    0.0.0.0:9200->9200/tcp, :::9200->9200/tcp, 0.0.0.0:9300->9300/tcp, :::9300->9300/tcp
                                        elasticsearch
9b7bfb85097b   jenkins-config-jenkins                                 "/usr/bin/tini -- /u…"   16 months ago    Up 13 hours     0.0.0.0:8080->8080/tcp, :::8080->8080/tcp, 0.0.0.0:50000->50000/tcp, :::50000->50000/t
cp                                      jenkins
c45ccbcee92b   portainer/portainer-ce:latest                          "/portainer"             17 months ago    Up 13 hours     0.0.0.0:8000->8000/tcp, :::8000->8000/tcp, 0.0.0.0:9000->9000/tcp, :::9000->9000/tcp,
0.0.0.0:9443->9443/tcp, :::9443->9443/tcp   portainer
root@Wissem-PC:~/devsecops-tools/16-monitoring-and-logging/E-L-K#
```

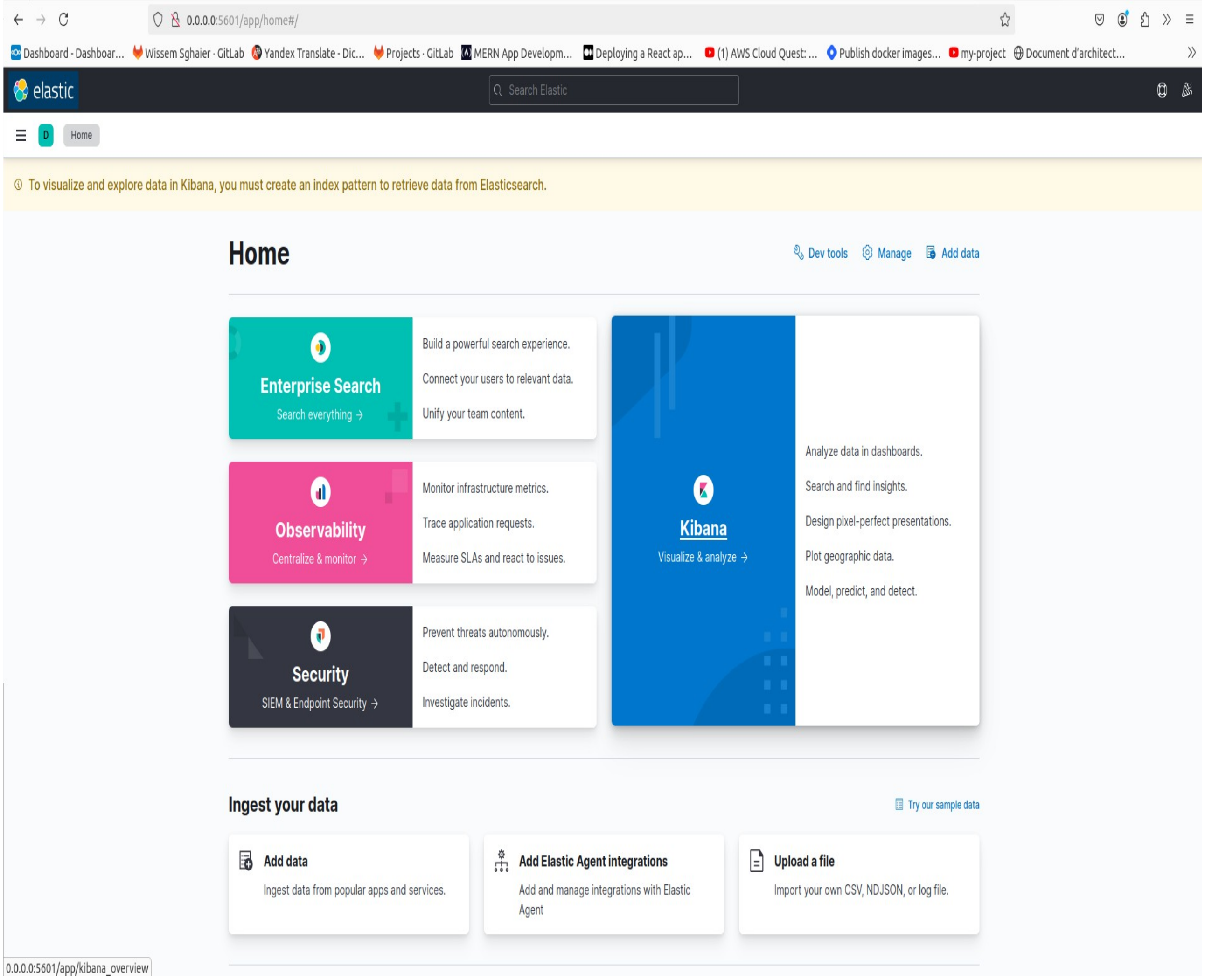
Access the ELK Stack

- **Elasticsearch**: Access Elasticsearch at http://localhost:9200.
- **Logstash**: Access Logstash at http://localhost:5044 (typically used for HTTP input and output).
- **Kibana**: Access Kibana at http://localhost:5601 to visualize and interact with data stored in Elasticsearch.

docker exec -it elasticsearch bin/elasticsearch-setup-passwords interactive

Access the Services:

- **Elasticsearch**: Open a web browser and navigate to `http://localhost:9200` to access Elasticsearch.
- **Logstash**: Open a web browser and navigate to `http://localhost:9600` to access Logstash's monitoring interface.
- **Kibana**: Open a web browser and navigate to `http://localhost:5601` to access Kibana.



0.0.0.0:5601/app/home#/

To visualize and explore data in Kibana, you must create an index pattern to retrieve data from Elasticsearch.

Build a powerful search experience.
Connect your users to relevant data.
Unify your team content.

Monitor infrastructure metrics.
Trace application requests.
Measure SLAs and react to issues.

Prevent threats autonomously.
Detect and respond.
Investigate incidents.

Analyze data in dashboards.
Search and find insights.
Design pixel-perfect presentations.
Plot geographic data.
Model, predict, and detect.

Ingest data from popular apps and services.

Add and manage integrations with Elastic Agent

Import your own CSV, NDJSON, or log file.

0.0.0.0:5601/app/kibana_overview

1. Create `docker-compose.yml` File

Save the following configuration to a file named `docker-compose.yml`:

profilpicture.png

```
version: '3'

services:
  elasticsearch:
    image: docker.elastic.co/elasticsearch/elasticsearch:8.0.0
    container_name: elasticsearch
    environment:
      - discovery.type=single-node
      - ES_JAVA_OPTS=-Xms2g -Xmx2g  # Set heap size to 2GB
    ports:
      - "9200:9200"
      - "9300:9300"
    volumes:
      - elasticsearch_data:/usr/share/elasticsearch/data
    networks:
      - devsecops

  logstash:
    image: docker.elastic.co/logstash/logstash:8.0.0
    container_name: logstash
    ports:
      - "5044:5044"
    volumes:
      - logstash_config:/usr/share/logstash/config
      - logstash_pipelines:/usr/share/logstash/pipeline
    networks:
      - devsecops

  kibana:
    image: docker.elastic.co/kibana/kibana:8.0.0
    container_name: kibana
    ports:
      - "5601:5601"
    depends_on:
      - elasticsearch
    environment:
     - ELASTICSEARCH_HOSTS=http://elasticsearch:9200
     - ELASTICSEARCH_USERNAME=kibana
     - ELASTICSEARCH_PASSWORD=kibana
    networks:
      - devsecops

volumes:
  elasticsearch_data:
  logstash_config:
  logstash_pipelines:
  kibana_data:

networks:
  devsecops:
    driver: bridge
```