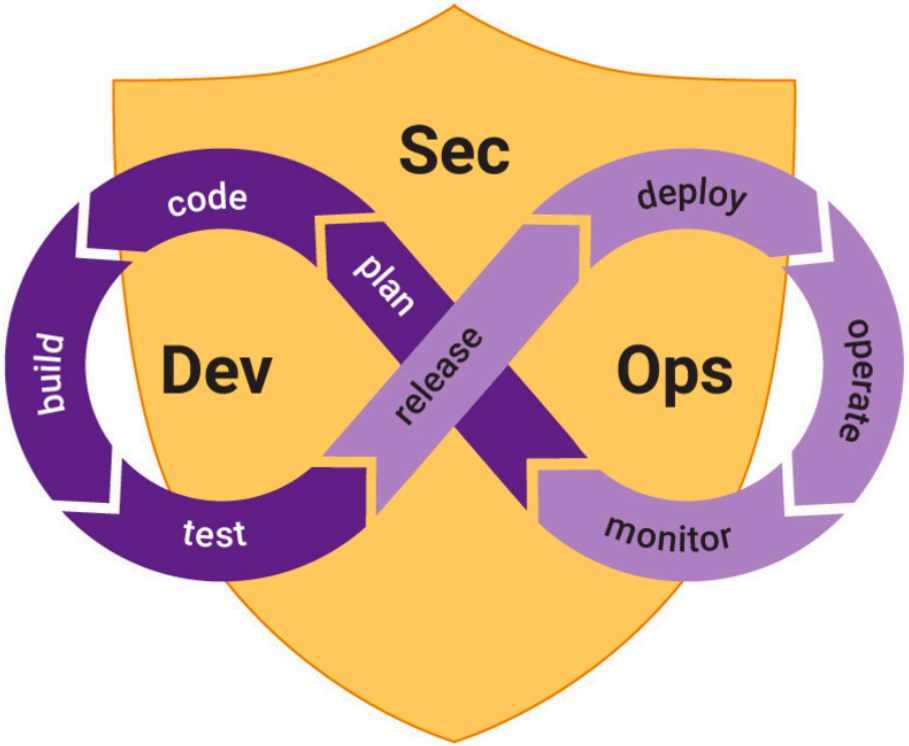
DevSecOps in Mobile Sec

DevSecOps

DevSecOps aims to implement Security as a part of the build process by integrating **Security tools as part of the Build process**, **Creating Security as a Culture** in the organization and **Automating as many processes** as possible



Source: https://www.synopsys.com/glossary/what-is-devsecops.html

Security in SDLC

Pre-Commit Hooks: Detect any hard coded passwords, API keys etc in the source code during the commit itself. For e.g *Talisman*, *git-secret*

IDE Plugins: Integrate Plugins to perform security analysis within the IDE itself, for e.g *Address Sanitizer in Xcode*, *SonarLint, FindBugs* etc

Secrets Management: Manage credentials such that can be integrated into the development toolset, for e.g *Ansible Vault*

Source Code Composition Analysis: Identify any vulnerable open-source libraries being use, for e.g Owasp dependency checker

Static Application Security Testing: Analyze source code or compiled version to find security flaws, for e.g *SonarQube*

Dynamic Application Security Testing: Scan the application in a running environment

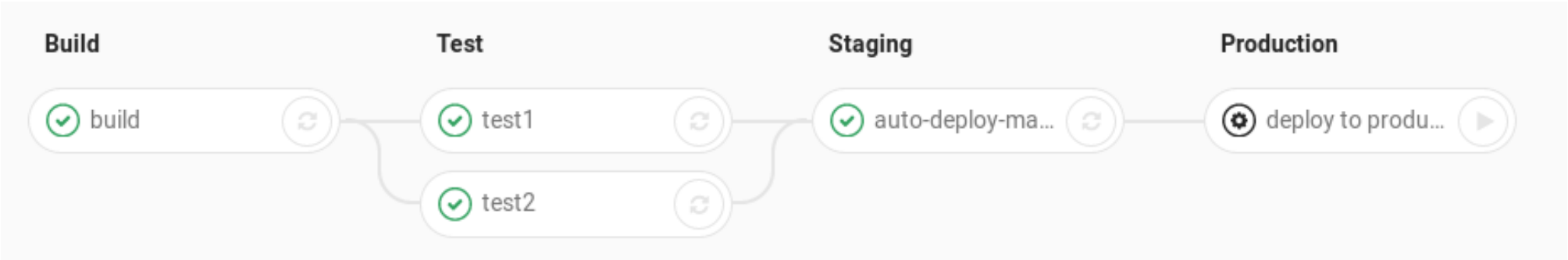
Container Scanning: Scan Docker images for vulnerabilities

Security in SDLC

1. Tools should support **CLI**
2. Preferably should have a docker image
3. Should finish execution in 10 minutes
4. Much more customizable (remove false positives, exclude files etc)
5. Should support Parsable output (XML, JSON etc)

Gitlab Pipelines

A pipeline is a group of **jobs** that get executed in stages. All of the **jobs** in a stage are executed in parallel (if there are enough concurrent Runners), and if they all succeed, the pipeline moves on to the next stage. If one of the **jobs** fails, the next stage is not (usually) executed. https://forge.etsi.org/rep/help/ci/pipelines.md



In Gitlab, the pipeline is put inside a file named .**gitlab-ci.yml** in the main repository

Sample CI/CD pipelines: https://docs.gitlab.com/ee/ci/examples/

.gitlab-ci.yml keywords

image: Use a specific docker image e.g image: python:3.6

before_script: Override a set of commands that are executed before job

services: Use Docker service images

allow_failure: Allow a job to fail. A failed job does not cause the pipeline to fail

artifacts: List of files and directories to attach to a job on success

when: when to run job (for e.g manual, delayed)

only: Run only on that specific branch

Some keywords can be set globally as the default for all jobs with the `default:` keyword. Default keywords can then be overridden by job-specific configuration.

Source: https://docs.gitlab.com/ee/ci/yaml/

Sample .gitlab-ci.yml (C++)

C++.gitlab-ci.yml 702 Bytes Edit Web IDE Lock

```yaml
# use the official gcc image, based on debian
# can use verions as well, like gcc:5.2
# see https://hub.docker.com/_/gcc/
image: gcc

build:
  stage: build
  # instead of calling g++ directly you can also use some build toolkit like make
  # install the necessary build tools when needed
  # before_script:
  #   - apt update && apt -y install make autoconf
  script:
    - g++ helloworld.cpp -o mybinary
  artifacts:
    paths:
      - mybinary
      # depending on your build setup it's most likely a good idea to cache outputs to reduce the build time
      # cache:
      #   paths:
      #     - "*.o"

# run tests using the binary built before
test:
  stage: test
  script:
    - ./runmytests.sh
```

Secrets scanning example - Trufflehog

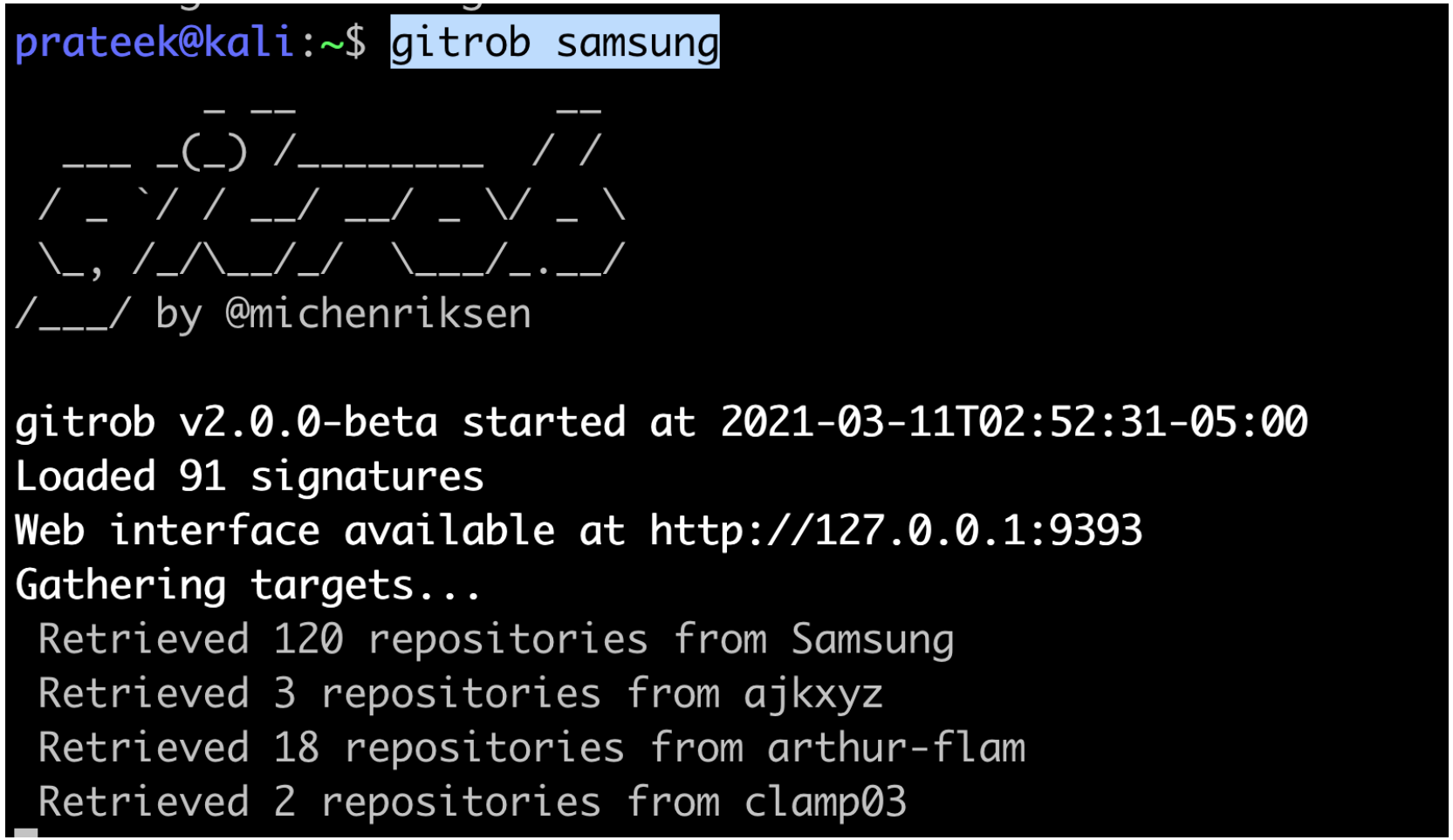
Scan through git repositories for secrets, looking into commit history and branches. For e.g **trufflehog**

```
prateek:: trufflehog --regex --entropy=False https://github.com/dxa4481/truffl
ehog.git
~~~~~~~~~~~~~~~~~~~~~
Reason: Generic Secret
Date: 2021-01-31 00:20:09
Hash: 288f35ed2e73145e3578207e896a4cf1600bea45
Filepath: test_all.py
Branch: origin/dev
Commit: Fixed since_commit and tests (#219)

* Fixed since_commit and tests

* Updated testing comment

Co-authored-by: Dylan Ayrey <dxa4481@rit.edu>
secret = '9ed54617547cfca783e0f81f8dc5c927e3d1e345'
```

Secrets scanning example - gitrob

Can be used to scan an organization, and its members public repos

https://github.com/michenriksen/gitrob



Task 5 mins: Create your project for DevSecOps

We will be using the vulnerable python application Django.nv for our labs today https://github.com/nVisium/django.nV

1. On Gitlab, **Create a New Project -> Import Project -> Repo by Url -> Paste https://github.com/nVisium/django.nV ,** Make sure it's a private project, Rename the project **Djangonv-devsecops** and click **Create**

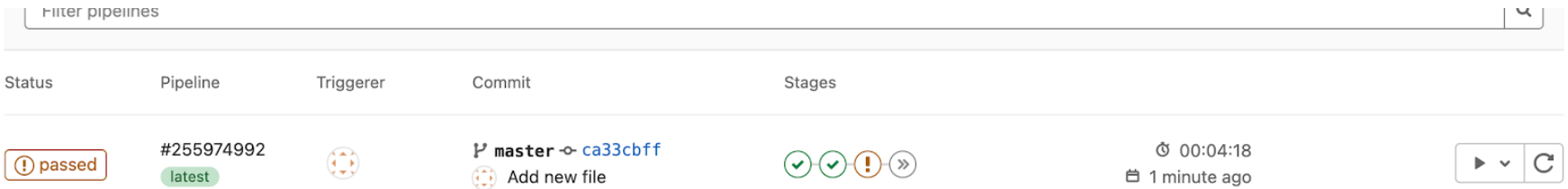
2. In the repo, add a **new File** by Clicking on **+** -> **New File**

3. Name it **.gitlab-ci.yml**

4. Paste the code from the file **1.txt** under **Day 2 -> DevSecOps**

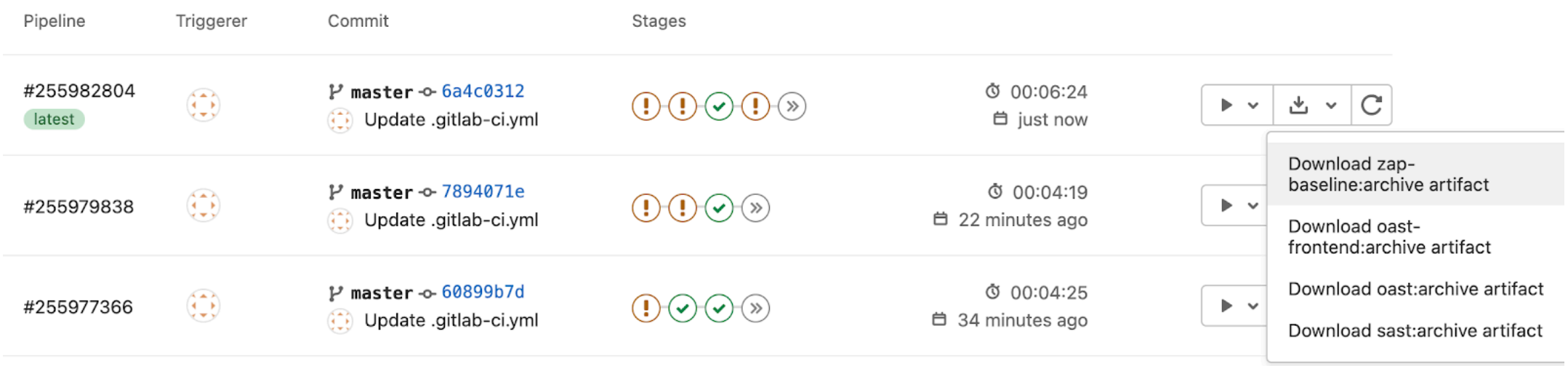
5. Click **Commit Changes**

6. On the Left Section go to **CI/CD -> Pipelines** and watch as all the stages complete execution



Task 15 mins: Adding SAST, SCA and DAST

1. Edit **.gitlab-ci.yml**
2. Paste the code from the following URL **Day 2 -> DevSecOps -> 2.txt** and **Commit Changes**
3. Notice the addition of **dast** job
4. Go to **CI/CD -> Pipelines** and check the results once pipeline finishes execution
5. Download the artifacts file and analyze the output of the tool



Task 10 mins

1. What is a ZAP baseline scan in DevSecOps ?
2. How do you ensure some jobs run only when changes are committed to master and not any other branch ?
3. In the same pipeline, add the following tools in the integration stage (target: **demo.testfire.net**)
    1. NIkto for Web Server Scanning: https://github.com/sullo/nikto
    2. SSLyze for SSL/certificate related issues https://github.com/nabla-c0d3/sslyze
    3. Nmap for port Scanning https://nmap.org/book/port-scanning-tutorial.html

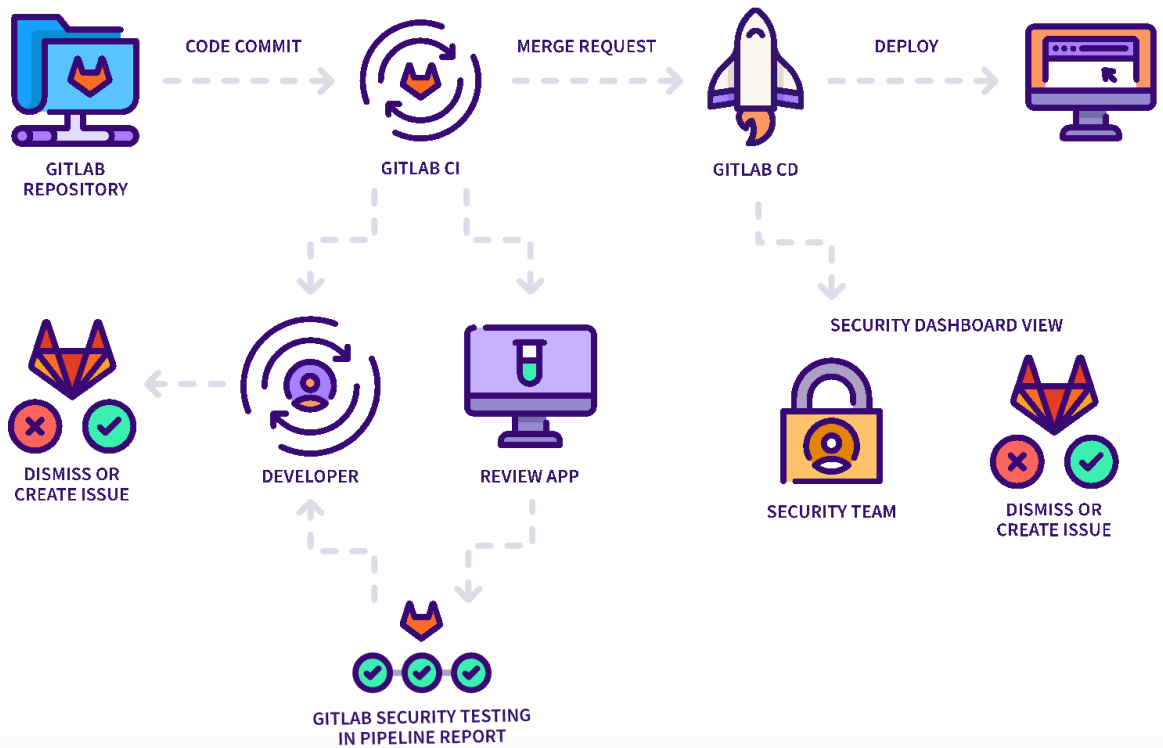
Solution: **Day 2 -> DevSecOps -> 3.txt**

-

Gitlab DevSecOps

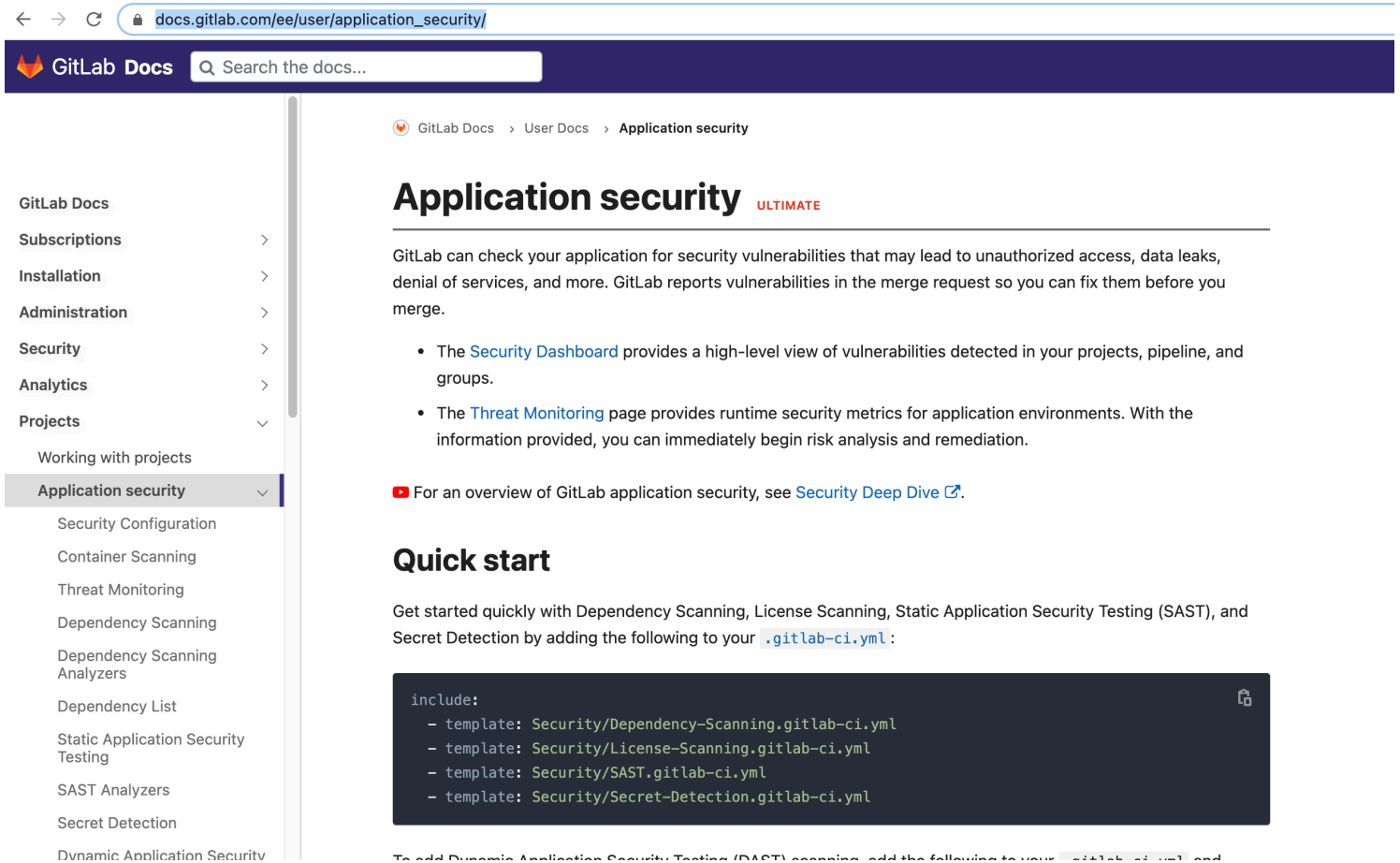
How GitLab enables DevSecOps

- **Every piece of code** is tested upon commit for security threats, without incremental cost.
- **The developer** can remediate now, while they are still working in that code, or create an issue with one click.
- **The security pro** can see and manage unresolved vulnerabilities captured as a by-product of software development.
- **Single source of truth** can focus collaboration on remediation, eliminating translation and finger pointing.
- **A single tool** reduces cost to buy, integrate and maintain point solutions throughout the DevOps pipeline.



https://about.gitlab.com/solutions/dev-sec-ops/

Gitlab DevSecOps



https://docs.gitlab.com/ee/user/application_security/

Task 10 mins: Adding Gitlab's DevSecOps to CI/CD

1. On Gitlab, **Create a New Project -> Import Project -> Repo by Url -> https://github.com/OWASP/Vulnerable-Web-Application** Rename the project **Owasp-VWA,** make sure it's a **Public Project** and click **Create**
2. In the repo, add a new File
3. Name it **.gitlab-ci.yml**
4. Paste the code from the file **4.txt** under **Day 2 -> DevSecOps**
5. Click **Commit Changes**
6. On the Left Section go to **CI/CD -> Pipelines** and watch as all the stages complete execution
7. Once finished, on the left, go to **Security & Compliance -> Vulnerability Report**

Task 10 mins: Adding Gitlab's DevSecOps to CI/CD

Critical	High	Medium	Low	Info	Unknown
0	19	203	453	18	0

Status Detected +1 more | **Severity** All severities | **Scanner** All scanners | **Activity** All

Detected	Status	↓ Severity	Description	Identifier	Scanner	Activity
2021-03-11	Detected	High	Easy XSS detected because of direct user input with $_SERVER on echo XSS/XSS_level5.php:14	PHPCS_SecurityAudit.BadFunctions.EasyXSS.EasyXSSerr	SAST	
2021-03-11	Detected	High	Easy XSS detected because of direct user input with $_GET on echo XSS/XSS_level1.php:22	PHPCS_SecurityAudit.BadFunctions.EasyXSS.EasyXSSerr	SAST	
2021-03-11	Detected	High	Filesystem function getimagesize() detected with dynamic parameter directly from user input FileUpload/fileupload3.php:28	PHPCS_SecurityAudit.BadFunctions.FilesystemFunctions.ErrFilesystem	SAST	
2021-03-11	Detected	High	Filesystem function move_uploaded_file() detected with dynamic parameter directly from user input FileUpload/fileupload3.php:38	PHPCS_SecurityAudit.BadFunctions.FilesystemFunctions.ErrFilesystem	SAST	

References

https://www.practical-devsecops.com/

https://medium.com/faun/gitlab-and-devsecops-pipeline-implementation-part-i-3e03e2e2a919

https://www.youtube.com/watch?v=NScuPB5uwg4

https://about.gitlab.com/blog/2019/06/20/announcing-gitlab-devsecops/

-