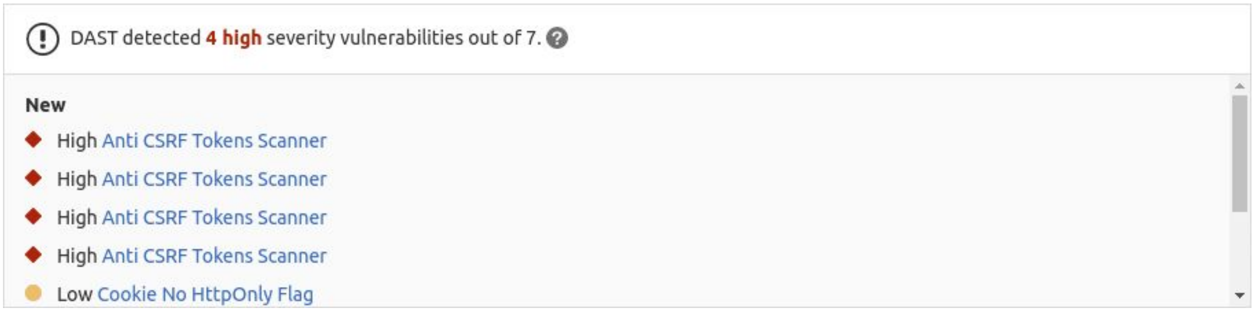
How to Work with gitlab and commands :

Overview

If you’re using GitLab CI/CD, you can analyze your running web applications for known vulnerabilities using Dynamic Application Security Testing (DAST). You can take advantage of DAST by either including the CI job in your existing `.gitlab-ci.yml`

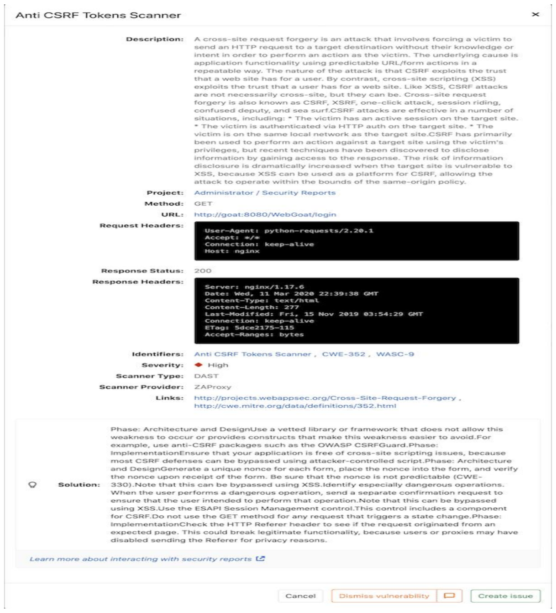
GitLab checks the DAST report, compares the found vulnerabilities between the source and target branches, and shows the information on the merge request.


DAST detected **4 high** severity vulnerabilities out of 7.

New

- High Anti CSRF Tokens Scanner
- High Anti CSRF Tokens Scanner
- High Anti CSRF Tokens Scanner
- High Anti CSRF Tokens Scanner
- Low Cookie No HttpOnly Flag

By clicking on one of the detected linked vulnerabilities, you can see the details and the URL(s) affected.


Anti CSRF Tokens Scanner

Description: A cross-site request forgery is an attack that involves forcing a victim to send an HTTP request to a target destination without their knowledge or intent in order to perform an action as the victim. The underlying cause is application functionality using predictable URL/form actions in a repeatable way. The nature of the attack is that CSRF exploits the trust that a web site has for a user. By contrast, cross-site scripting (XSS) exploits the trust that a user has for a web site. Like XSS, CSRF attacks are not necessarily cross-site, but they can be. Cross-site request forgery is also known as CSRF, XSRF, one-click attack, session riding, confused deputy, and sea surf.CSRF attacks are effective in a number of situations, including: * The victim has an active session on the target site. * The victim is authenticated via HTTP auth on the target site. * The victim is on the same local network as the target site.CSRF has primarily been used to perform an action against a target site using the victim's privileges, but recent techniques have been discovered to disclose information by gaining access to the response. The risk of information disclosure is dramatically increased when the target site is vulnerable to XSS, because XSS can be used as a platform for CSRF, allowing the attack to operate within the bounds of the same-origin policy.

Project: Administrator / Security Reports

Method: GET

URL: http://goat:8080/WebGoat/login

Request Headers:

```
User-Agent: python-requests/2.20.1
Accept: */*
Connection: keep-alive
Host: nginx
```

Response Status: 200

Response Headers:

```
Server: nginx/1.17.6
Date: Wed, 11 Mar 2020 22:39:38 GMT
Content-Type: text/html
Content-Length: 277
Last-Modified: Fri, 15 Nov 2019 03:54:29 GMT
Connection: keep-alive
ETag: 5dce2175-115
Accept-Ranges: bytes
```

Identifiers: Anti CSRF Tokens Scanner , CWE-352 , WASC-9

Severity: ● High

Scanner Type: DAST

Scanner Provider: ZAProxy

Links: http://projects.webappsec.org/Cross-Site-Request-Forgery , http://cwe.mitre.org/data/definitions/352.html

Solution: Phase: Architecture and DesignUse a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.For example, use anti-CSRF packages such as the OWASP CSRFGuard.Phase: ImplementationEnsure that your application is free of cross-site scripting issues, because most CSRF defenses can be bypassed using attacker-controlled script.Phase: Architecture and DesignGenerate a unique nonce for each form, place the nonce into the form, and verify the nonce upon receipt of the form. Be sure that the nonce is not predictable (CWE-330).Note that this can be bypassed using XSS.Identify especially dangerous operations. When the user performs a dangerous operation, send a separate confirmation request to ensure that the user intended to perform that operation.Note that this can be bypassed using XSS.Use the ESAPI Session Management control.This control includes a component for CSRF.Do not use the GET method for any request that triggers a state change.Phase: ImplementationCheck the HTTP Referer header to see if the request originated from an expected page. This could break legitimate functionality, because users or proxies may have disabled sending the Referer for privacy reasons.

Dynamic Application Security Testing (DAST) uses the popular open source tool OWASP Zed Attack Proxy to perform an analysis on your running web application.

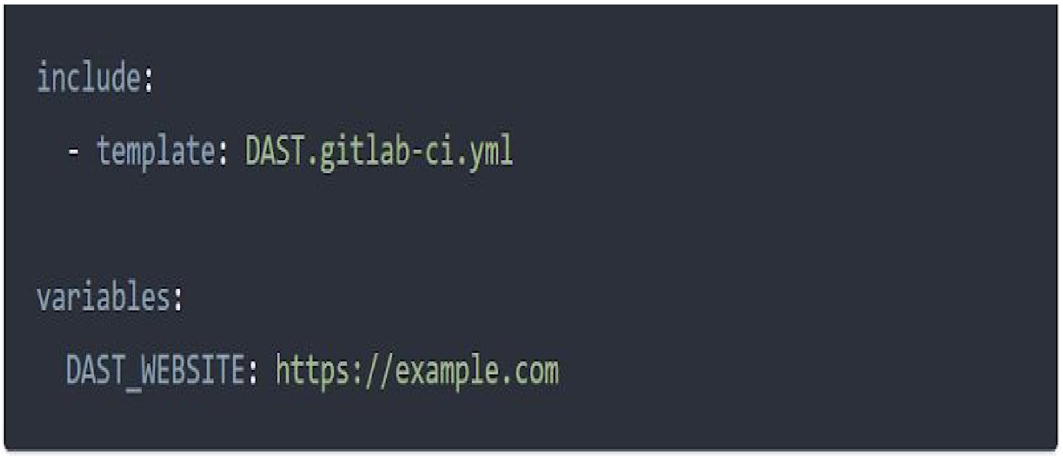
By default, DAST executes ZAP Baseline Scan and performs passive scanning only. It doesn't actively attack your application. However, DAST can be configured to also perform an *active scan*: attack your application and produce a more extensive security report.

Requirements

To run a DAST job, you need GitLab Runner with the `docker` executor.

Configuration

include the `DAST.gitlab-ci.yml` template that's provided as a part of your GitLab installation.



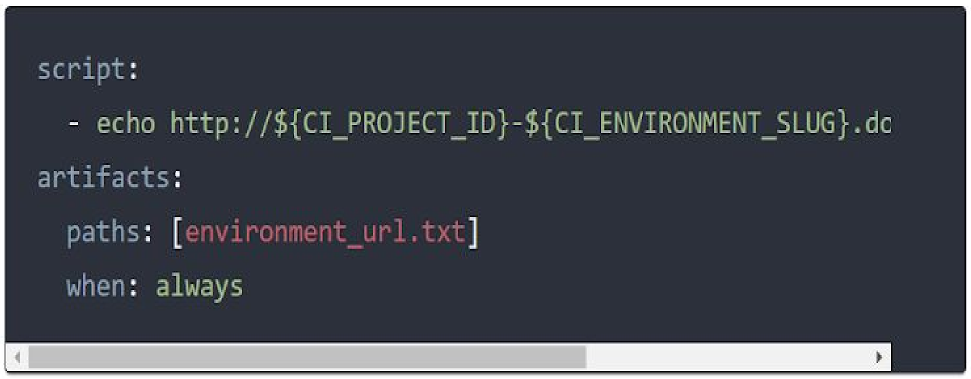
```yaml
include:
  - template: DAST.gitlab-ci.yml

variables:
  DAST_WEBSITE: https://example.com
```

Add the following to your `.gitlab-ci.yml` file:

There are two ways to define the URL to be scanned by DAST:

1. Set the `DAST_WEBSITE` variable.
2. Add it in an `environment_url.txt` file at the root of your project. This is useful for testing in dynamic environments. To run DAST against an application dynamically created during a GitLab CI/CD pipeline, a job that runs prior to the DAST scan must persist the application's domain in an `environment_url.txt` file. DAST automatically parses the `environment_url.txt` file to find its scan target.
   For example, in a job that runs prior to DAST, you could include code that looks similar to:



```
script:
  - echo http://${CI_PROJECT_ID}-${CI_ENVIRONMENT_SLUG}.dc
artifacts:
  paths: [environment_url.txt]
  when: always
```

1. You can see an example of this in our Auto DevOps CI YAML file.

If both values are set, the `DAST_WEBSITE` value takes precedence.

The included template creates a `dast` job in your CI/CD pipeline and scans your project's source code for possible vulnerabilities.

The results are saved as a DAST report artifact that you can later download and analyze. Due to implementation limitations we always take the latest DAST artifact available. Behind the scenes, the GitLab DAST Docker image is used to run the tests on the specified URL and scan it for possible vulnerabilities.

By default, the DAST template uses the latest major version of the DAST Docker image. Using the `DAST_VERSION` variable, you can choose how DAST updates:

- Automatically update DAST with new features and fixes by pinning to a major version (such as `1`).
- Only update fixes by pinning to a minor version (such as `1.6`).
- Prevent all updates by pinning to a specific version (such as `1.6.4`).

Find the latest DAST versions on the Releases page.

When DAST scans run

When using `DAST.gitlab-ci.yml` template, the `dast` job is run last as shown in the ex. Below:

```
stages:
  - build
  - test
  - deploy
  - dast
```

Be aware that if your pipeline is configured to deploy to the same webserver in each run, running a pipeline while another is still running could cause a race condition where one pipeline overwrites the code from another pipeline. The site to be scanned should be excluded from changes for the duration of a DAST scan. The only changes to the site should be from the DAST scanner. Be aware that any changes that users, scheduled tasks, database changes, code changes, other pipelines, or other scanners make to the site during a scan could lead to inaccurate results.

Authentication

It's also possible to authenticate the user before performing the DAST checks

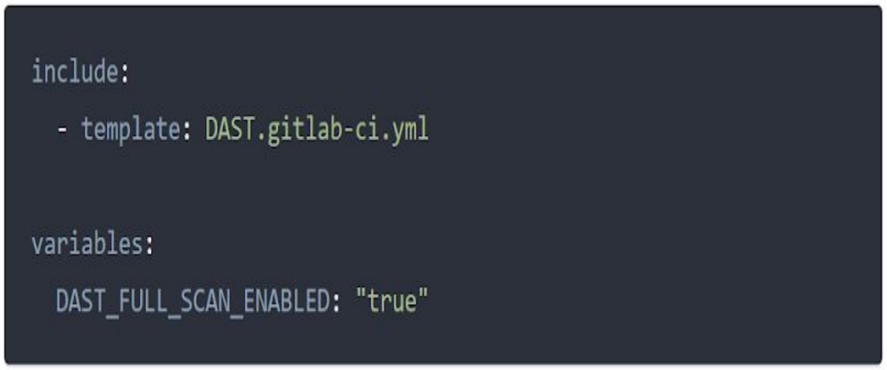
```
include:
  - template: DAST.gitlab-ci.yml

variables:
  DAST_WEBSITE: https://example.com
  DAST_AUTH_URL: https://example.com/sign-in
  DAST_USERNAME_FIELD: session[user]  # the name of username fi
  DAST_PASSWORD_FIELD: session[password]  # the name of passwor
  DAST_SUBMIT_FIELD: login # the `id` or `name` of the element
  DAST_FIRST_SUBMIT_FIELD: next # the `id` or `name` of the ele
  DAST_AUTH_EXCLUDE_URLS: http://example.com/sign-out,http://ex
```

The results are saved as a DAST report artifact that you can later download and analyze. Due to implementation limitations, we always take the latest DAST artifact available.

Full scan

DAST can be configured to perform ZAP Full Scan, which includes both passive and active scanning against the same target website:



```
include:
  - template: DAST.gitlab-ci.yml

variables:
  DAST_FULL_SCAN_ENABLED: "true"
```

Domain validation

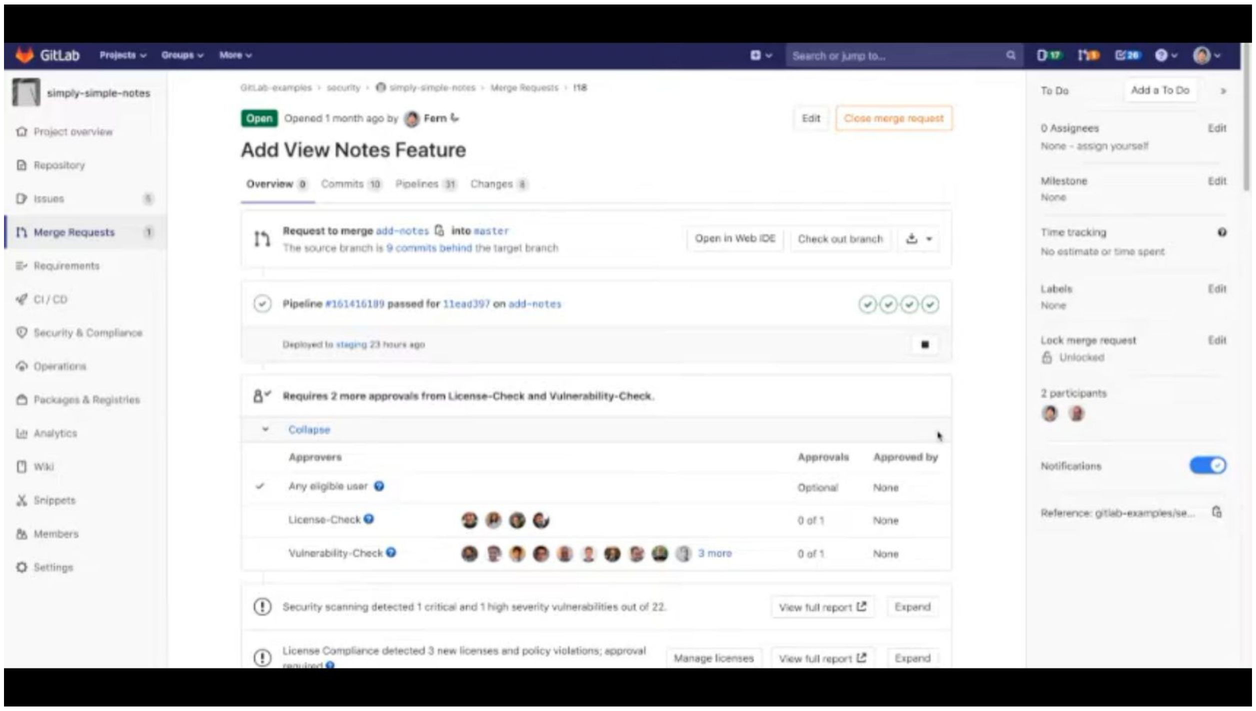
The DAST job can be run anywhere, which means you can accidentally hit live web servers and potentially damage them. You could even take down your production environment. For that reason, you should use domain validation.

```yaml
include:
  - template: DAST.gitlab-ci.yml

variables:
  DAST_FULL_SCAN_ENABLED: "true"
  DAST_FULL_SCAN_DOMAIN_VALIDATION_REQUIRED: "true"
```

Execution of this :

1.



Open Opened 1 month ago by Fern

Add View Notes Feature

Request to merge add-notes into master
The source branch is 9 commits behind the target branch

Pipeline #161416189 passed for 11ead397 on add-notes
Deployed to staging 23 hours ago

Requires 2 more approvals from License-Check and Vulnerability-Check.

Approvers		Approvals	Approved by
Any eligible user		Optional	None
License-Check		0 of 1	None
Vulnerability-Check	3 more	0 of 1	None

Security scanning detected 1 critical and 1 high severity vulnerabilities out of 22.

License Compliance detected 3 new licenses and policy violations; approval required

0 Assignees
None - assign yourself

Milestone
None

Time tracking
No estimate or time spent

Labels
None

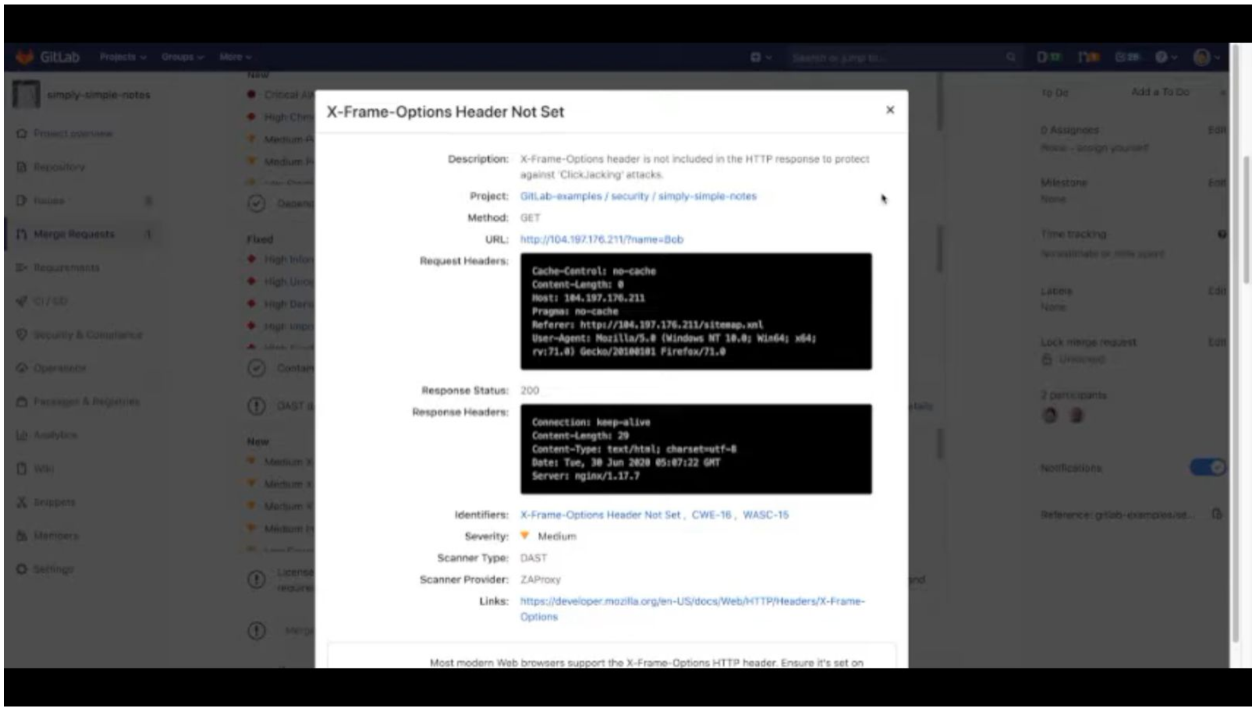
Lock merge request
Unlocked

2 participants

Notifications

Reference: gitlab-examples/se...

2.



X-Frame-Options Header Not Set

Description: X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.

Project: GitLab-examples / security / simply-simple-notes

Method: GET

URL: http://104.197.176.211/?name=Bob

Request Headers:

```
Cache-Control: no-cache
Content-Length: 0
Host: 104.197.176.211
Pragma: no-cache
Referer: http://104.197.176.211/sitemap.xml
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64;
rv:71.0) Gecko/20100101 Firefox/71.0
```

Response Status: 200

Response Headers:

```
Connection: keep-alive
Content-Length: 29
Content-Type: text/html; charset=utf-8
Date: Tue, 30 Jun 2020 05:07:22 GMT
Server: nginx/1.17.7
```

Identifiers: X-Frame-Options Header Not Set, CWE-16, WASC-15

Severity: Medium

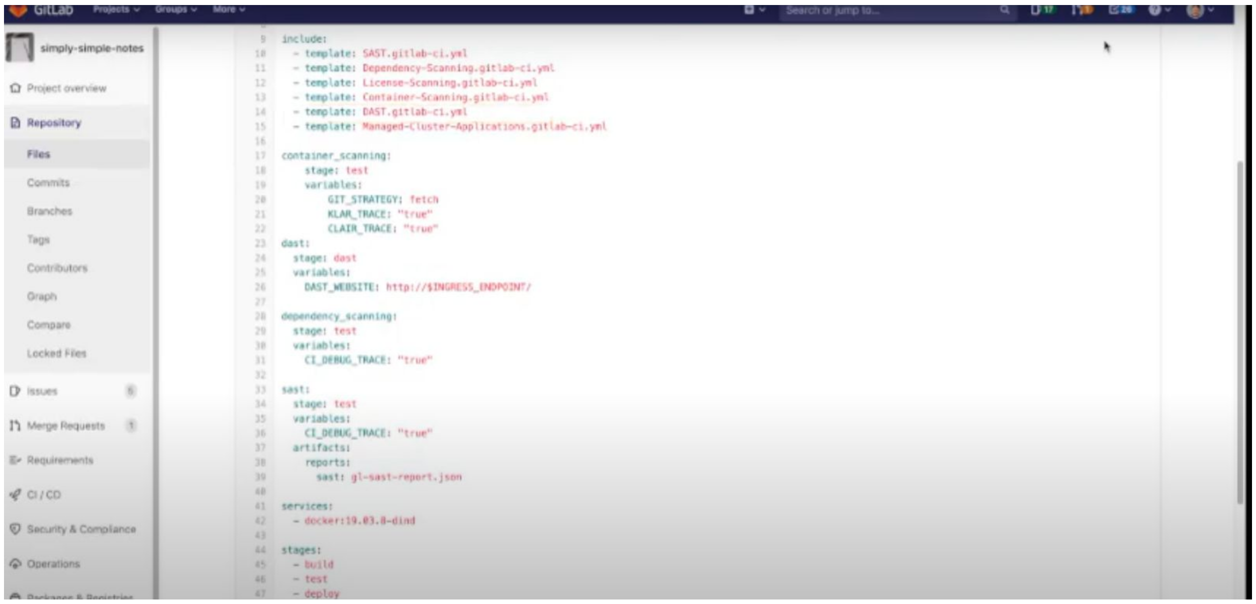
Scanner Type: DAST

Scanner Provider: ZAProxy

Links: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options

Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on

3.



```
include:
  - template: SAST.gitlab-ci.yml
  - template: Dependency-Scanning.gitlab-ci.yml
  - template: License-Scanning.gitlab-ci.yml
  - template: Container-Scanning.gitlab-ci.yml
  - template: DAST.gitlab-ci.yml
  - template: Managed-Cluster-Applications.gitlab-ci.yml

container_scanning:
    stage: test
    variables:
        GIT_STRATEGY: fetch
        KLAR_TRACE: "true"
        CLAIR_TRACE: "true"
dast:
  stage: dast
  variables:
    DAST_WEBSITE: http://$INGRESS_ENDPOINT/

dependency_scanning:
  stage: test
  variables:
    CI_DEBUG_TRACE: "true"

sast:
  stage: test
  variables:
    CI_DEBUG_TRACE: "true"
  artifacts:
    reports:
      sast: gl-sast-report.json

services:
  - docker:19.03.8-dind

stages:
  - build
  - test
  - deploy
```