SHIP-HATS 2.0 WebApp Pipeline Webinar

Learning Events | Level 200 Tech

CLOUD | DEVOPS MACHINE LEARNING

Implementation and training services

CLOUD ENABLED PTE LTD

ABOUT US

ANIL BIDARI

CEO @ CLOUD ENABLED

SINCE 2013

INDIA | SINGAPORE | DUBAI

PHONE +65 -81320344

EMAIL

reach@thecloudenabled.com

WEBSITE

www.thecloudenabled.com

Understand e2e templates

Pipeline templates

SHIP-HATS 2.0 offers a library of CI/CD pipeline templates that simplifies configuration efforts.

E2E templates

- Main ci file to build your pipeline
- Build – Test – Deploy Yaml files with all variable keys defined (only values you need to add)

Modular templates

- Leverage these if Anything specific task in pipeline required
- example – check webapp is ready or not

Review 1 : e2e Templates

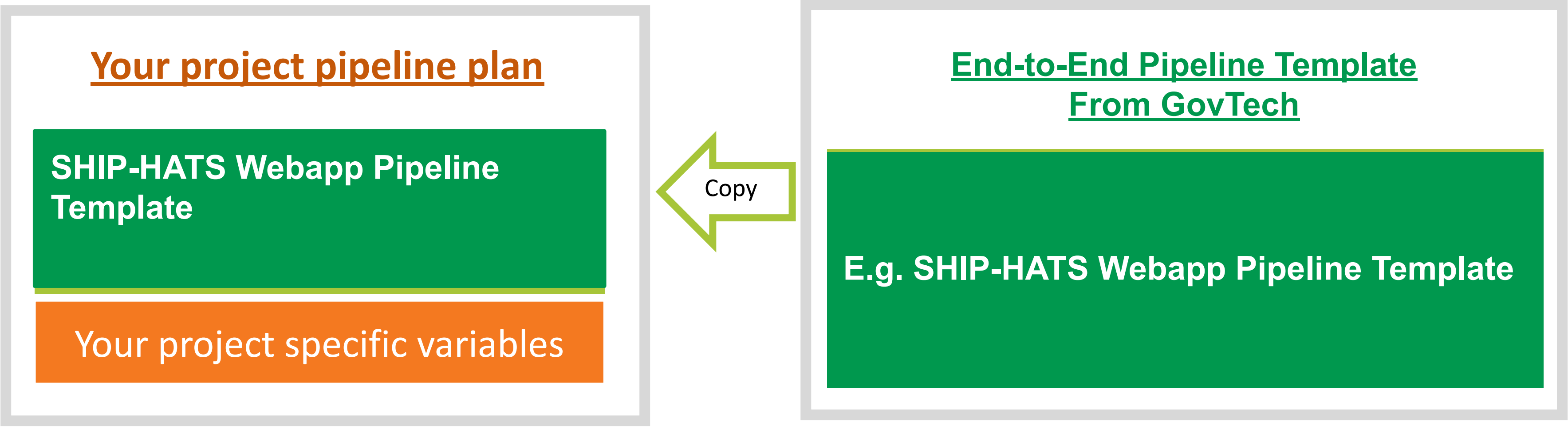
1. Go to Developer Portal: SHIP-HATS Pipeline Templates

2. Check the for E2E Templates

3. Let’s review SHIP-HATS Webapp E2E Template

Link to access to e2e template click here

End-to-end Pipeline Template



Your project pipeline plan

SHIP-HATS Webapp Pipeline Template

Your project specific variables

Copy

End-to-End Pipeline Template From GovTech

E.g. SHIP-HATS Webapp Pipeline Template

Demo

- Fork e2e template into your project
- Git clone e2e templates to your developer machine
- Add your program files and tests if any folder to local repo
- Push local repo back to your gitlab remote repository

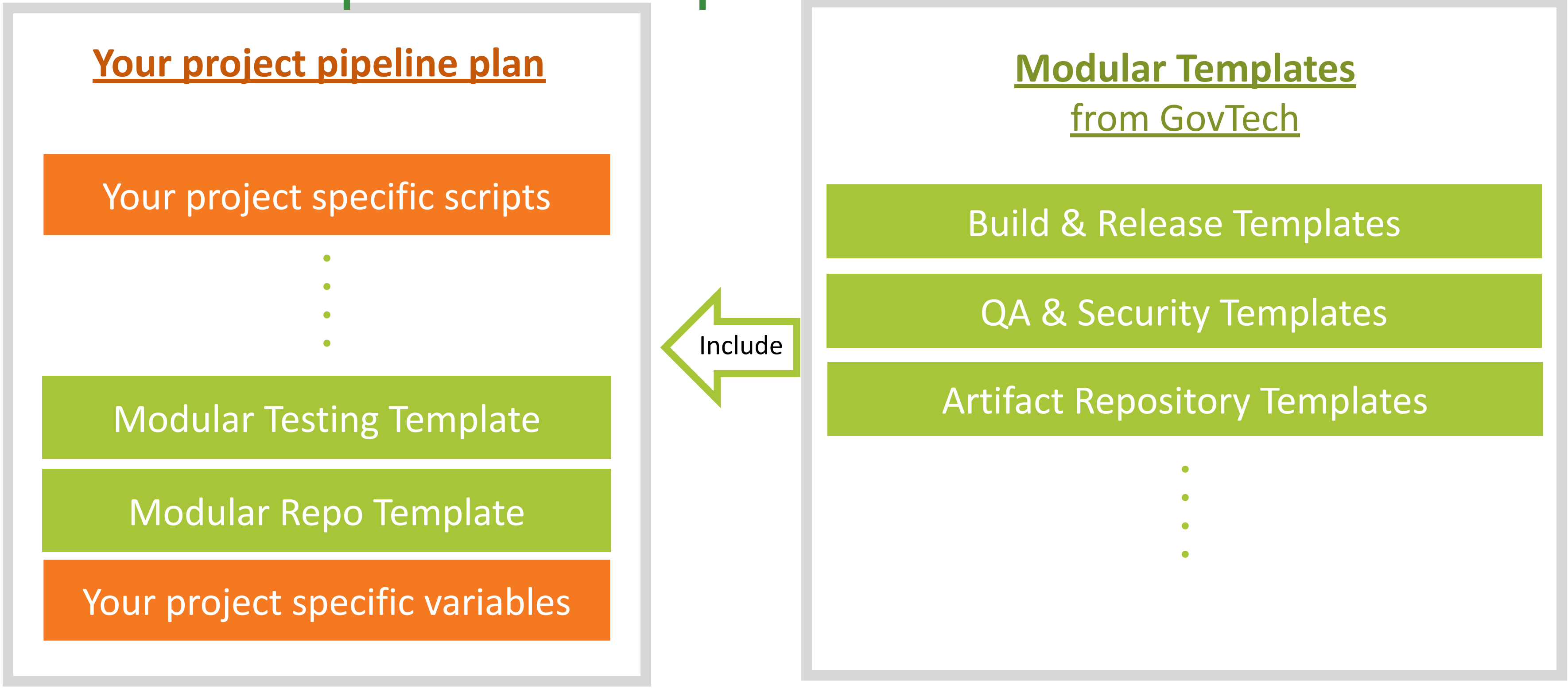
Review 2 : Modular Templates

1. Go to Developer Portal: SHIP-HATS Pipeline Templates

2. Check the Modular Templates: Common Template Testing & Scanning, Nexus Repo,

1. Let’s review AWS Secret Retrieval

Modular Pipeline Templates



Your project pipeline plan

Your project specific scripts

⋮

Modular Testing Template

Modular Repo Template

Your project specific variables

Include

Modular Templates

from GovTech

Build & Release Templates

QA & Security Templates

Artifact Repository Templates

⋮

Compliance Framework

✔ Automate adopting DevSecOps best practices based on industry pipeline security & IM8

- Examples:
  1. SCA including Dependency scanning
  2. SAST / DAST / Container scanning
  3. Gating before deployment to high stake environments
  4. Reports generation as part of provenance
  5. Signing and verification of signature on artefacts
  6. Checksum verification of artefacts
  7. Use of artifactory

✔ We will iteratively improve these to help you meet DevSecOps policy.

✔ Highly recommended

Compliance Framework

Benefits

- ✔ Setup CI/CD pipeline faster
- ✔ Leverage GitLab’s OTS security tools and reporting tools
- ✔ Flexibility to change to **non-GitLab Alternative Tools** in 2.0
- ✔ Better quality by achieving compliance to industry standards.

Know Your Resource Activity

- https://docs.developer.tech.gov.sg/docs/ship-hats-getting-started/learning-events bookmark this page in Developer portal

- Go to https://go.gov.sg/she and bookmark for feedback, feature requests - we priorities based on demand!

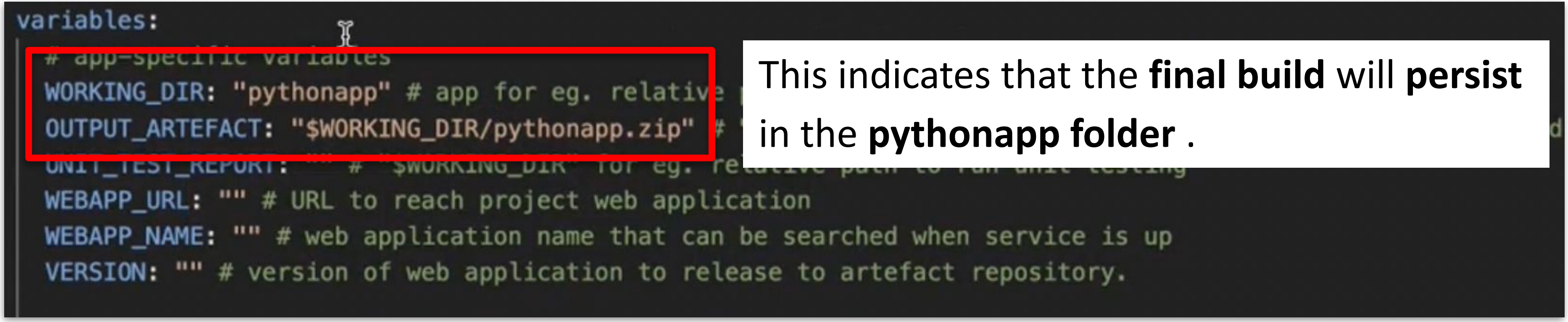
Configure Build

Define the **app-specific details** such as **how to build the webapp** in **BUILD.gitlab-ci.yml** file

Variables Defined

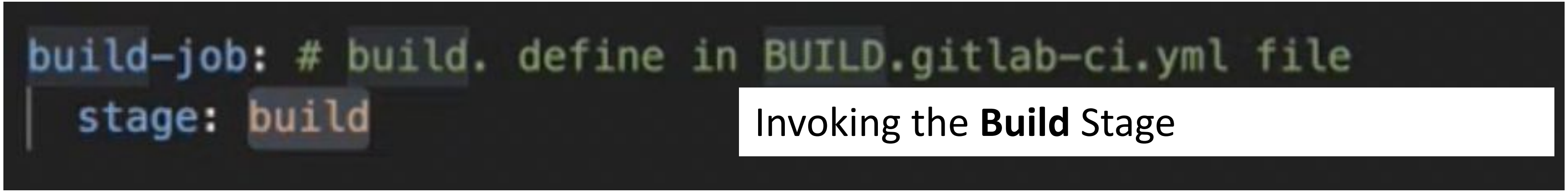
- **WORKING_DIR**
- **OUTPUT_ARTEFACT**

Observe .gitlab-ci.yml



```
variables:
  # app-specific variables
  WORKING_DIR: "pythonapp" # app for eg. relative
  OUTPUT_ARTEFACT: "$WORKING_DIR/pythonapp.zip" #
  UNIT_TEST_REPORT: "" # "$WORKING_DIR" for eg. relative path to run unit testing
  WEBAPP_URL: "" # URL to reach project web application
  WEBAPP_NAME: "" # web application name that can be searched when service is up
  VERSION: "" # version of web application to release to artefact repository.
```

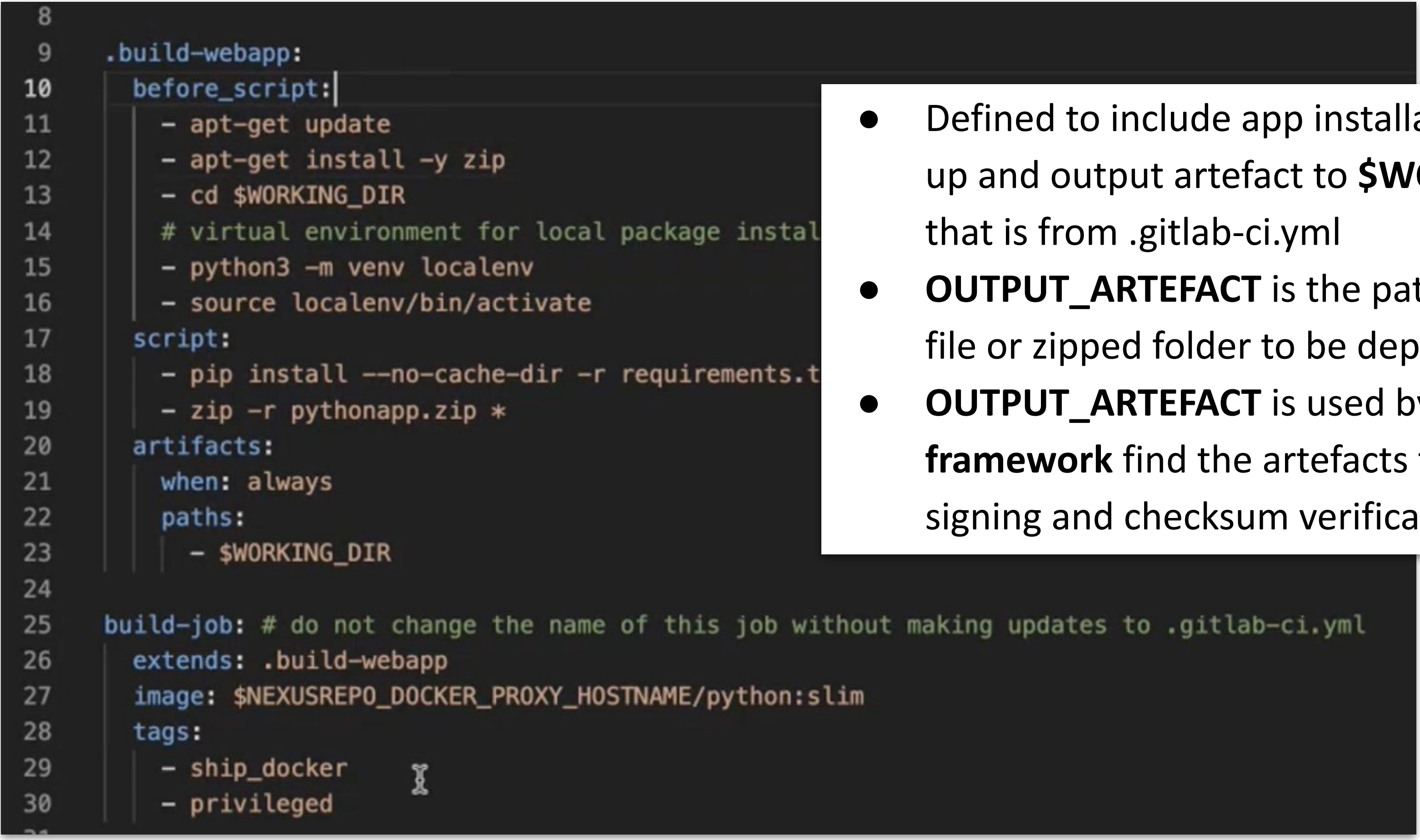
This indicates that the **final build** will **persist** in the **pythonapp folder** .



```
build-job: # build. define in BUILD.gitlab-ci.yml file
  stage: build
```

Invoking the **Build** Stage

Observe BUILD.gitlab-ci.yml



```
.build-webapp:
  before_script:
    - apt-get update
    - apt-get install -y zip
    - cd $WORKING_DIR
    # virtual environment for local package instal
    - python3 -m venv localenv
    - source localenv/bin/activate
  script:
    - pip install --no-cache-dir -r requirements.t
    - zip -r pythonapp.zip *
  artifacts:
    when: always
    paths:
      - $WORKING_DIR

build-job: # do not change the name of this job without making updates to .gitlab-ci.yml
  extends: .build-webapp
  image: $NEXUSREPO_DOCKER_PROXY_HOSTNAME/python:slim
  tags:
    - ship_docker
    - privileged
```

- Defined to include app installation, zipping up and output artefact to **$WORKINGDIR** that is from .gitlab-ci.yml
- **OUTPUT_ARTEFACT** is the path of the binary file or zipped folder to be deployed.
- **OUTPUT_ARTEFACT** is used by **compliance framework** find the artefacts for scanning, signing and checksum verification

Configure Testing

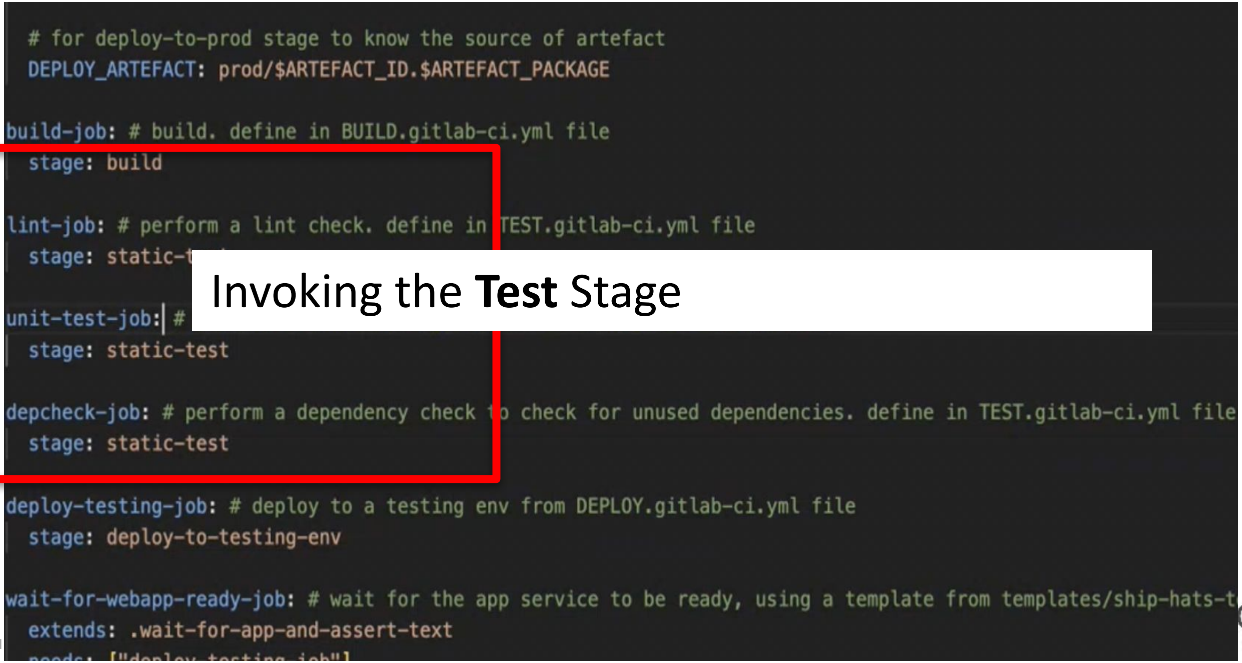
Define the **app-specific tests** for the WebApp, in **TEST.gitlab-ci.yml** file.

Types of tests included here are:

- Unit/integration testing
- Language-based linting
- Framework-based dependency checks
- E2E testing

Observe 1) .gitlab-ci.yml

vim .gitlab-ci.yml

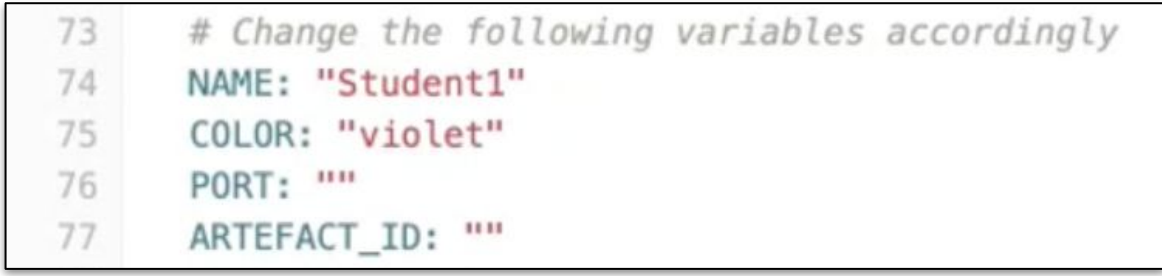


```
  # for deploy-to-prod stage to know the source of artefact
  DEPLOY_ARTEFACT: prod/$ARTEFACT_ID.$ARTEFACT_PACKAGE

build-job: # build. define in BUILD.gitlab-ci.yml file
  stage: build

lint-job: # perform a lint check. define in TEST.gitlab-ci.yml file
  stage: static-t

unit-test-job: #
  stage: static-test

depcheck-job: # perform a dependency check to check for unused dependencies. define in TEST.gitlab-ci.yml file
  stage: static-test

deploy-testing-job: # deploy to a testing env from DEPLOY.gitlab-ci.yml file
  stage: deploy-to-testing-env

wait-for-webapp-ready-job: # wait for the app service to be ready, using a template from templates/ship-hats-t
  extends: .wait-for-app-and-assert-text
  needs: ["deploy-testing-job"]
```

Invoking the **Test** Stage

Let’s Personalise

vim **.gitlab-ci.yml**



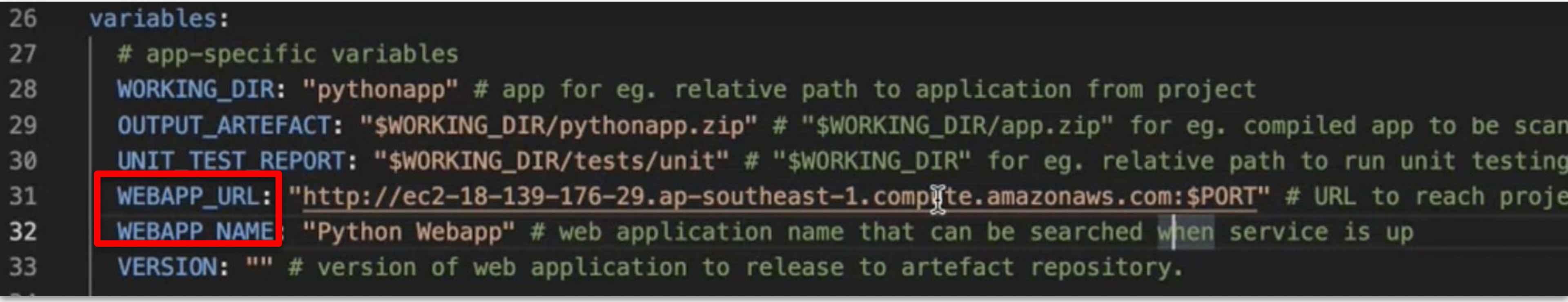
```
73        # Change the following variables accordingly
74        NAME: "Student1"
75        COLOR: "violet"
76        PORT: ""
77        ARTEFACT_ID: ""
```

Variable	Description
NAME	Name to be displayed in the WebApp and for tests, spaces to be separated/delimited by an additional backslash E.g. If the name is John Smith, NAME='John\ Smith’ (line 75)
COLOR	Color of header in the webapp (CSS color scheme) Example violet (Line 76)

Configure Deploy

Define the **infrastructure specifics** in deploy the webapp in **DEPLOY.gitlab-ci.yml file**. This depends on how the application is hosted.

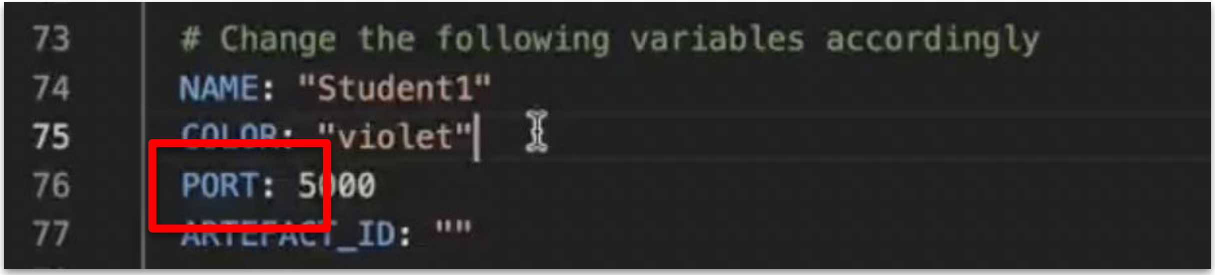
Observe 1) .gitlab-ci.yml



```
variables:
  # app-specific variables
  WORKING_DIR: "pythonapp" # app for eg. relative path to application from project
  OUTPUT_ARTEFACT: "$WORKING_DIR/pythonapp.zip" # "$WORKING_DIR/app.zip" for eg. compiled app to be scan
  UNIT_TEST_REPORT: "$WORKING_DIR/tests/unit" # "$WORKING_DIR" for eg. relative path to run unit testing
  WEBAPP_URL: "http://ec2-18-139-176-29.ap-southeast-1.compute.amazonaws.com:$PORT" # URL to reach proje
  WEBAPP_NAME: "Python Webapp" # web application name that can be searched when service is up
  VERSION: "" # version of web application to release to artefact repository.
```

- **WEBAPP_URL = “ec2-18-139-176-29.ap-southeast-1.compute.amazonaws.com:$PORT"**
- **WEBAPP_NAME**="Python Webapp"

Change port number in .gitlab-ci.yml

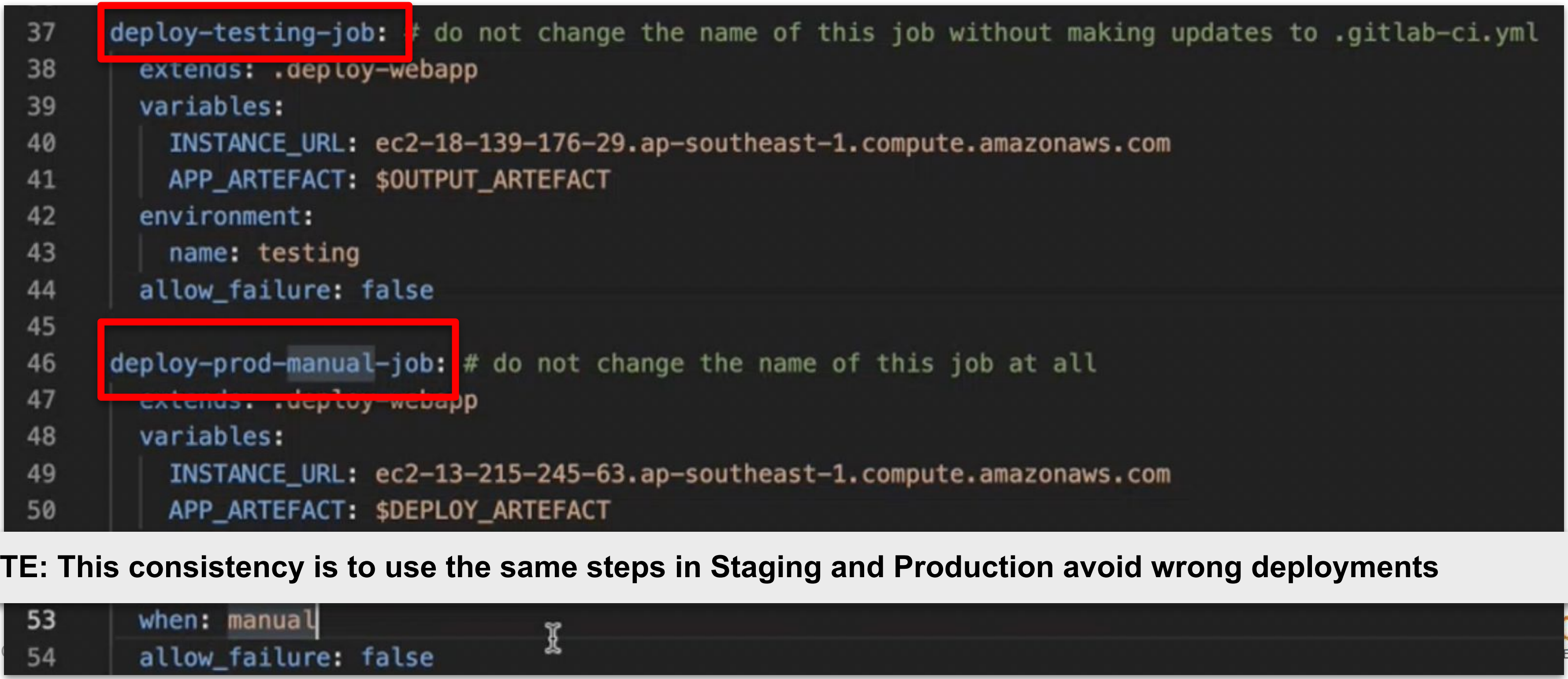


```
# Change the following variables accordingly
NAME: "Student1"
COLOR: "violet"
PORT: 5000
ARTEFACT_ID: ""
```

- **PORT: 5000 + <Class Index Number > Example: 5001, .. 5025**

Observe 2) DEPLOY.gitlab-ci.yml

Deployments defined - one for staging, another for production

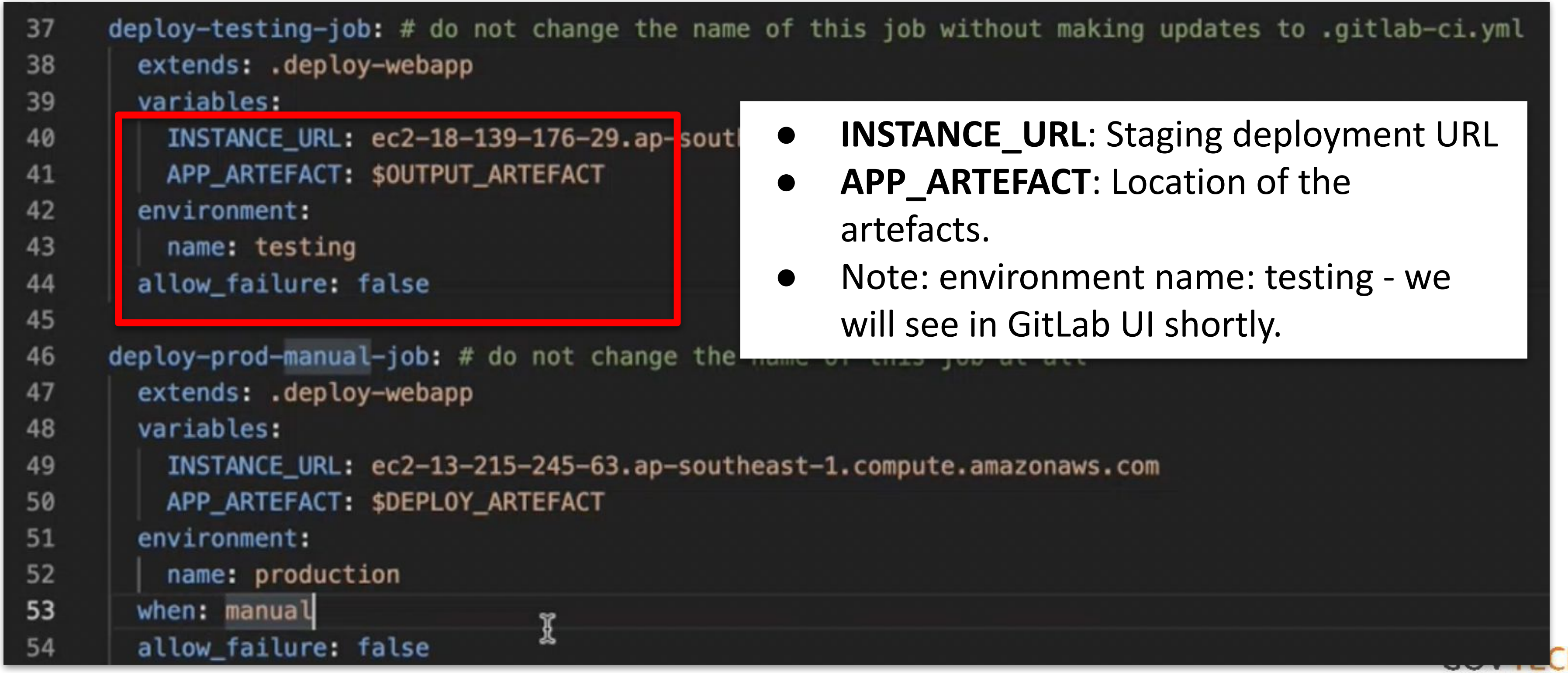


```
deploy-testing-job: # do not change the name of this job without making updates to .gitlab-ci.yml
  extends: .deploy-webapp
  variables:
    INSTANCE_URL: ec2-18-139-176-29.ap-southeast-1.compute.amazonaws.com
    APP_ARTEFACT: $OUTPUT_ARTEFACT
  environment:
    name: testing
  allow_failure: false

deploy-prod-manual-job: # do not change the name of this job at all
  extends: .deploy-webapp
  variables:
    INSTANCE_URL: ec2-13-215-245-63.ap-southeast-1.compute.amazonaws.com
    APP_ARTEFACT: $DEPLOY_ARTEFACT
```

```
  when: manual
  allow_failure: false
```

NOTE: This consistency is to use the same steps in Staging and Production avoid wrong deployments

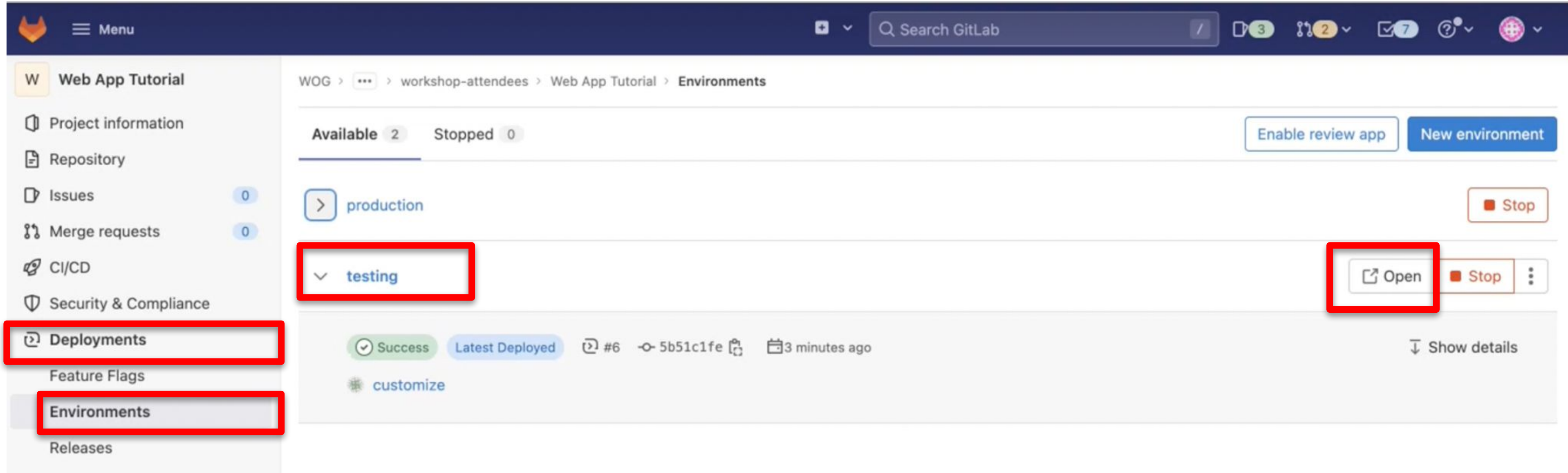
Observe 2) DEPLOY.gitlab-ci.yml



```
deploy-testing-job: # do not change the name of this job without making updates to .gitlab-ci.yml
  extends: .deploy-webapp
  variables:
    INSTANCE_URL: ec2-18-139-176-29.ap-sout
    APP_ARTEFACT: $OUTPUT_ARTEFACT
  environment:
    name: testing
  allow_failure: false

deploy-prod-manual-job: # do not change the name of this job at all
  extends: .deploy-webapp
  variables:
    INSTANCE_URL: ec2-13-215-245-63.ap-southeast-1.compute.amazonaws.com
    APP_ARTEFACT: $DEPLOY_ARTEFACT
  environment:
    name: production
  when: manual
  allow_failure: false
```

- **INSTANCE_URL**: Staging deployment URL
- **APP_ARTEFACT**: Location of the artefacts.
- Note: environment name: testing - we will see in GitLab UI shortly.

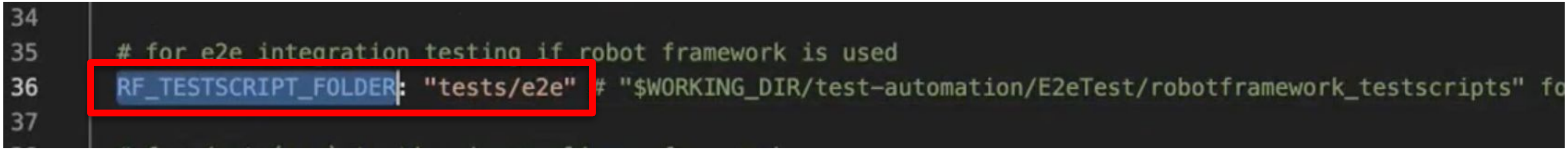
Observe Open the App



Configure Runtime Test

Defining Robot Framework in this example.

Observe 1) .gitlab-ci.yml



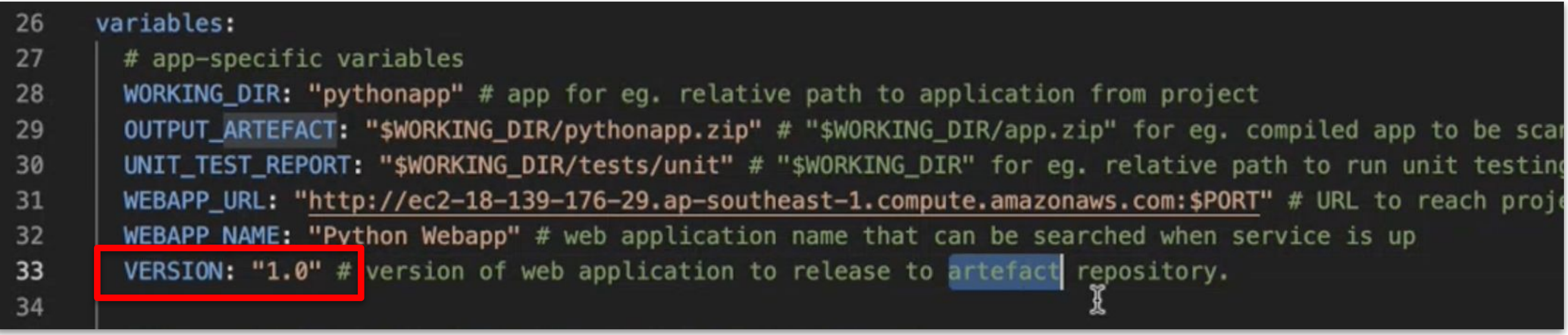
```
  # for e2e integration testing if robot framework is used
  RF_TESTSCRIPT_FOLDER: "tests/e2e" # "$WORKING_DIR/test-automation/E2eTest/robotframework_testscripts" fo
```

- Set **RF_TESTSCRIPT_FOLDER** as “**tests/e2e**”

Configure Publish

Defining how to publish artefacts to Nexus Repo using Maven in this example.

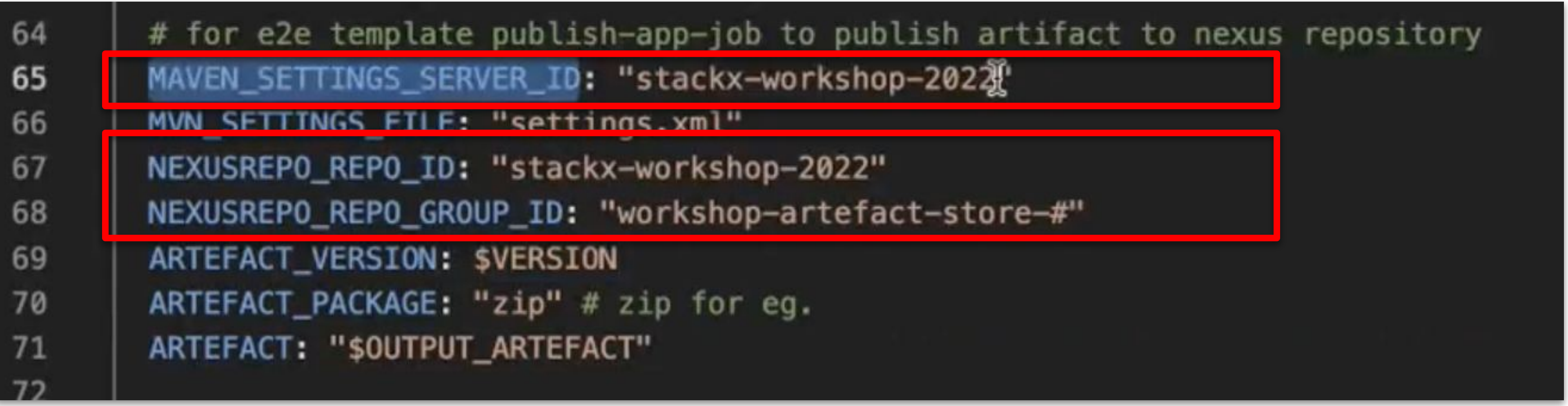
Observe 1) Variables .gitlab-ci.yml



```
variables:
  # app-specific variables
  WORKING_DIR: "pythonapp" # app for eg. relative path to application from project
  OUTPUT_ARTEFACT: "$WORKING_DIR/pythonapp.zip" # "$WORKING_DIR/app.zip" for eg. compiled app to be sca
  UNIT_TEST_REPORT: "$WORKING_DIR/tests/unit" # "$WORKING_DIR" for eg. relative path to run unit testin
  WEBAPP_URL: "http://ec2-18-139-176-29.ap-southeast-1.compute.amazonaws.com:$PORT" # URL to reach proje
  WEBAPP_NAME: "Python Webapp" # web application name that can be searched when service is up
  VERSION: "1.0" # version of web application to release to artefact repository.

```

- **VERSION = “1.0"** Version of web app published to artefact repository

Observe 1) Variables .gitlab-ci.yml



```yaml
# for e2e template publish-app-job to publish artifact to nexus repository
MAVEN_SETTINGS_SERVER_ID: "stackx-workshop-2022"
MVN_SETTINGS_FILE: "settings.xml"
NEXUSREPO_REPO_ID: "stackx-workshop-2022"
NEXUSREPO_REPO_GROUP_ID: "workshop-artefact-store-#"
ARTEFACT_VERSION: $VERSION
ARTEFACT_PACKAGE: "zip" # zip for eg.
ARTEFACT: "$OUTPUT_ARTEFACT"
```

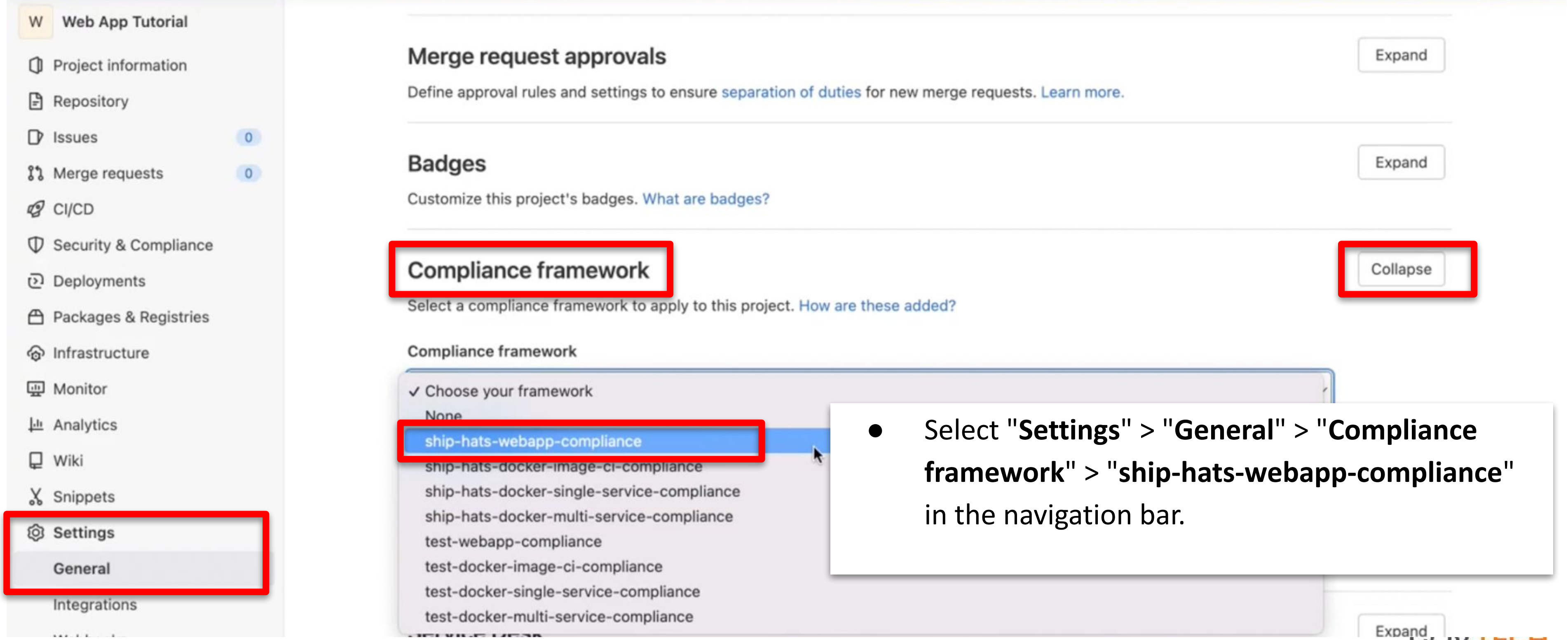
- **VERSION = “1.0"** Version of web app published to artefact repository
- **NEXUSREPO_REPO_GROUP_ID = “workshop-artefact-store-CLASS INDEX NUMBER”** Variable for publish-maven-artefact template.

Apply Compliance Framework

Compliance Framework applies of the Security component in DevSecOps on top of any pipeline.

We will apply **Webapp-compliance** in this example

Apply Compliance



W Web App Tutorial

- Project information
- Repository
- Issues 0
- Merge requests 0
- CI/CD
- Security & Compliance
- Deployments
- Packages & Registries
- Infrastructure
- Monitor
- Analytics
- Wiki
- Snippets
- Settings
  - General
  - Integrations

Merge request approvals Expand

Define approval rules and settings to ensure separation of duties for new merge requests. Learn more.

Badges Expand

Customize this project's badges. What are badges?

Compliance framework Collapse

Select a compliance framework to apply to this project. How are these added?

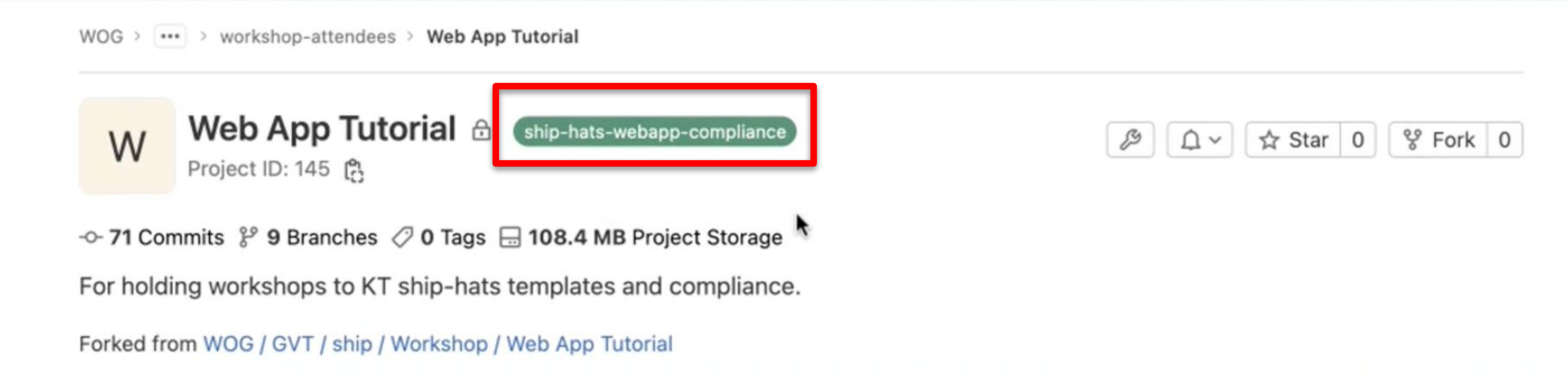
Compliance framework

- ✓ Choose your framework
- None
- ship-hats-webapp-compliance
- ship-hats-docker-image-ci-compliance
- ship-hats-docker-single-service-compliance
- ship-hats-docker-multi-service-compliance
- test-webapp-compliance
- test-docker-image-ci-compliance
- test-docker-single-service-compliance
- test-docker-multi-service-compliance

Expand

- Select "**Settings**" > "**General**" > "**Compliance framework**" > "**ship-hats-webapp-compliance**" in the navigation bar.

Verify Compliance Framework is Applied



WOG › ••• › workshop-attendees › Web App Tutorial

W

Web App Tutorial ship-hats-webapp-compliance

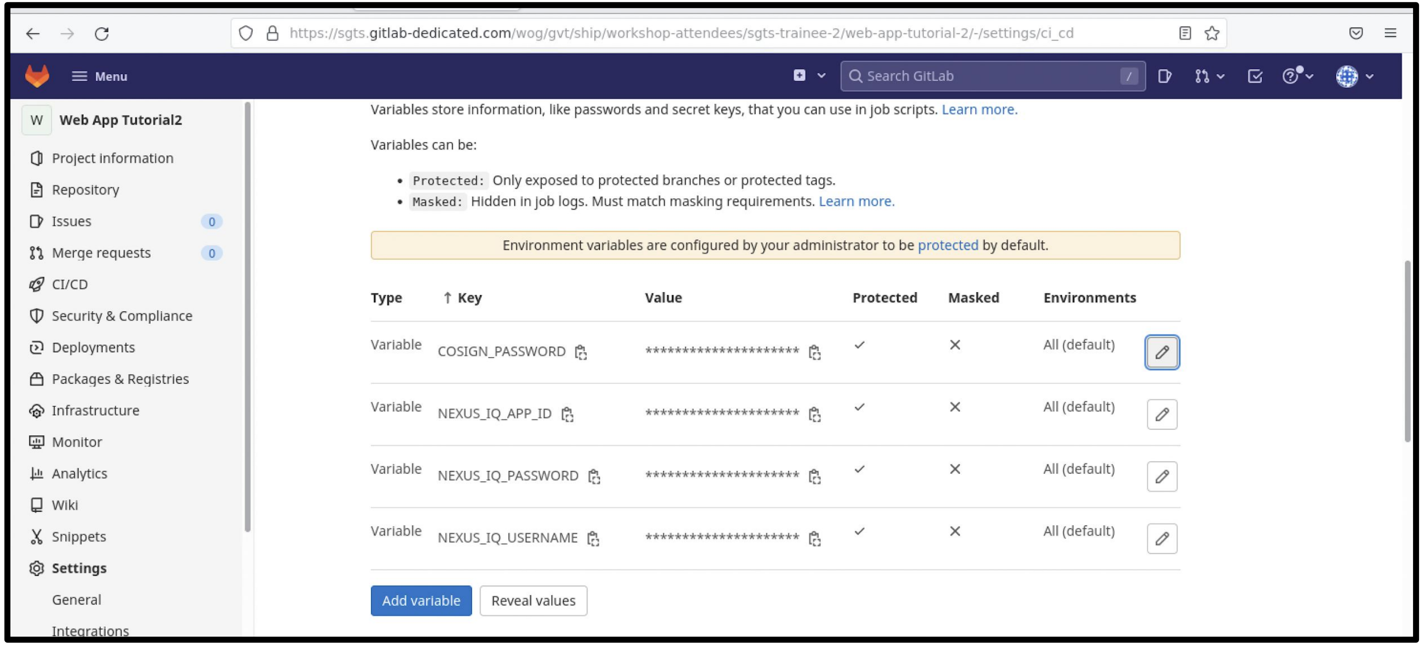
Project ID: 145

71 Commits 9 Branches 0 Tags 108.4 MB Project Storage

For holding workshops to KT ship-hats templates and compliance.

Forked from WOG / GVT / ship / Workshop / Web App Tutorial

verify after adding variables



Variables store information, like passwords and secret keys, that you can use in job scripts. Learn more.

Variables can be:

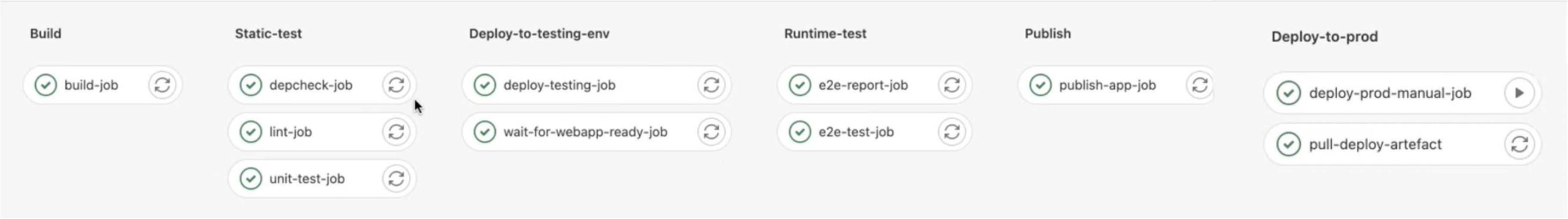
- `Protected:` Only exposed to protected branches or protected tags.
- `Masked:` Hidden in job logs. Must match masking requirements. Learn more.

Environment variables are configured by your administrator to be protected by default.

Type	↑ Key	Value	Protected	Masked	Environments
Variable	COSIGN_PASSWORD	*********************	✓	✕	All (default)
Variable	NEXUS_IQ_APP_ID	*********************	✓	✕	All (default)
Variable	NEXUS_IQ_PASSWORD	*********************	✓	✕	All (default)
Variable	NEXUS_IQ_USERNAME	*********************	✓	✕	All (default)

Observe Additional Jobs Included

Before applying compliance



Build
- ✓ build-job

Static-test
- ✓ depcheck-job
- ✓ lint-job
- ✓ unit-test-job

Deploy-to-testing-env
- ✓ deploy-testing-job
- ✓ wait-for-webapp-ready-job

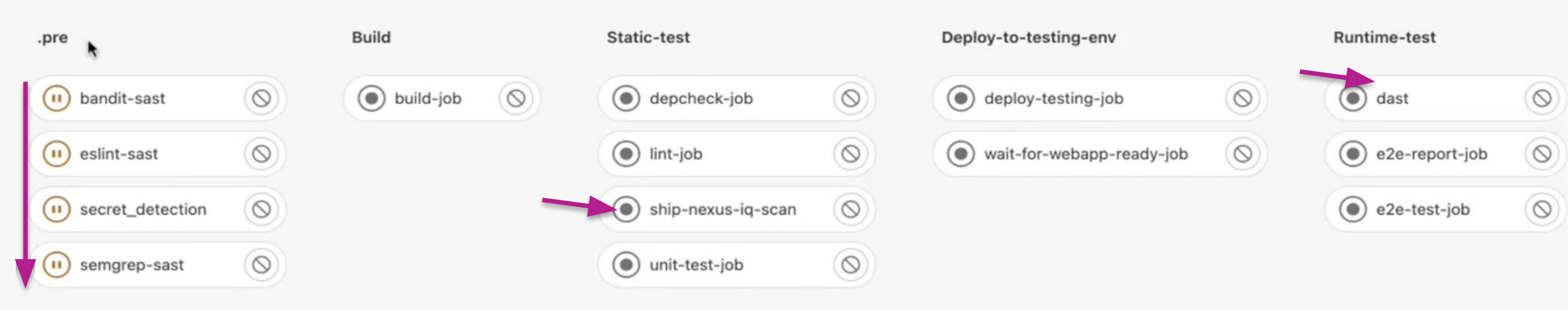
Runtime-test
- ✓ e2e-report-job
- ✓ e2e-test-job

Publish
- ✓ publish-app-job

Deploy-to-prod
- ✓ deploy-prod-manual-job
- ✓ pull-deploy-artefact

Observe Additional Jobs Included

After applying compliance



.pre
- bandit-sast
- eslint-sast
- secret_detection
- semgrep-sast

Build
- build-job

Static-test
- depcheck-job
- lint-job
- ship-nexus-iq-scan
- unit-test-job

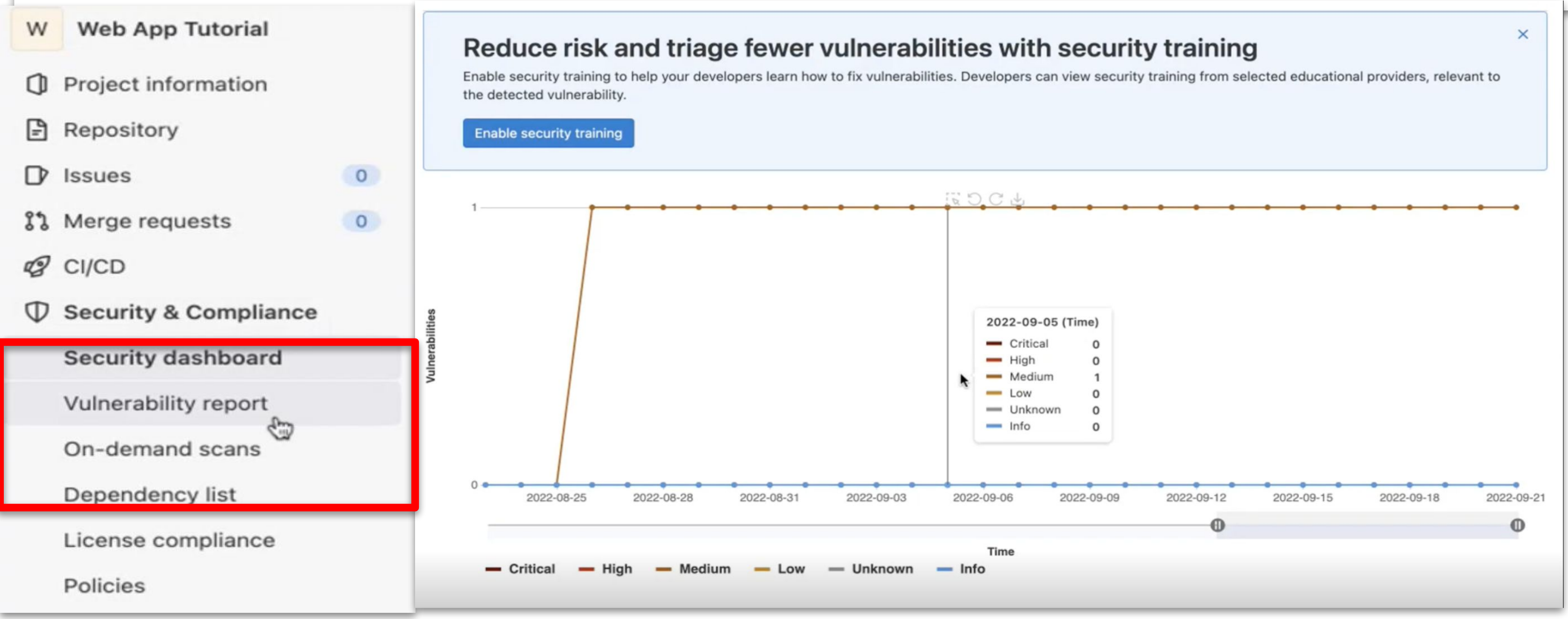
Deploy-to-testing-env
- deploy-testing-job
- wait-for-webapp-ready-job

Runtime-test
- dast
- e2e-report-job
- e2e-test-job

Bandit - tool used (depends on what ur programme language)
Eslint : tool used (depends on what ur programme language)
secret_detection – gitlab native
Semgrep-sast : gitlab native

Observe Security Issues

Option 2 **Security & Compliance > Security Dashboard or Vulnerability Report**



Reduce risk and triage fewer vulnerabilities with security training

Enable security training to help your developers learn how to fix vulnerabilities. Developers can view security training from selected educational providers, relevant to the detected vulnerability.

2022-09-05 (Time)

Critical	0
High	0
Medium	1
Low	0
Unknown	0
Info	0

Alternative tools

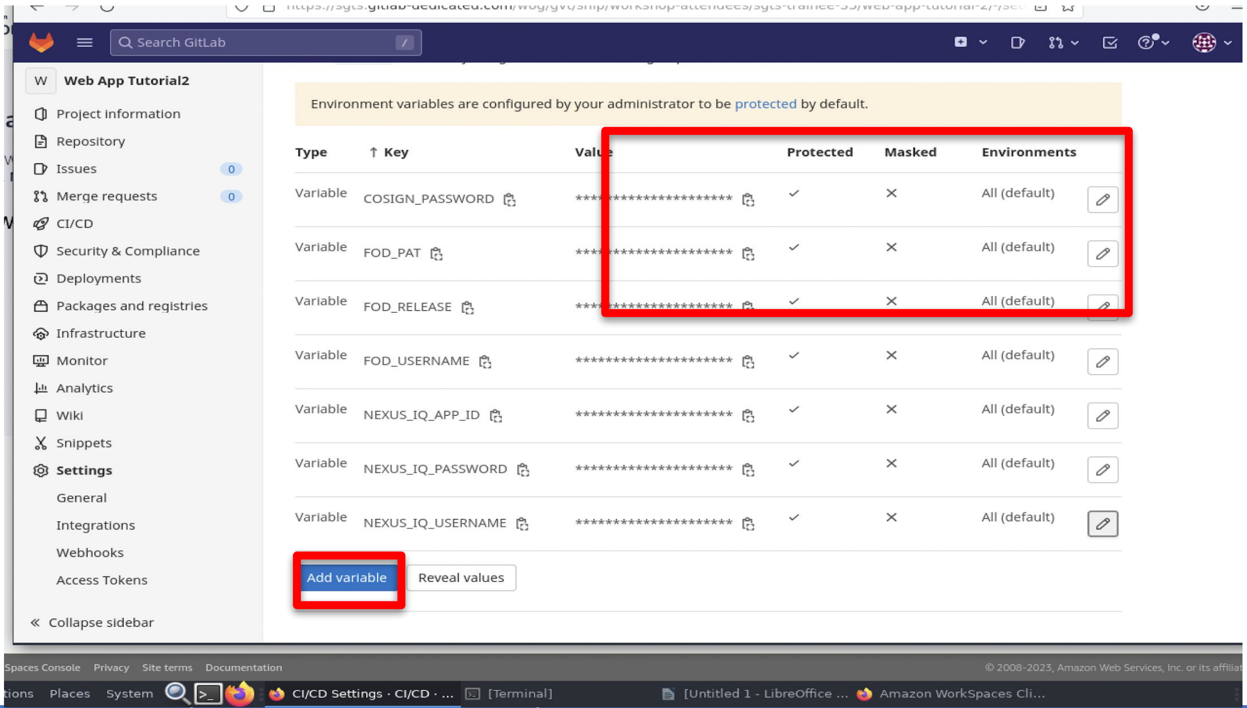
Demo how to integrate Fortify On Demand

Add FOD Variables

Add fod variables

Notes:
https://sgts.gitlab-dedicated.com/wog/ship-hats-compliance/-/blob/main/scans/static.gitlab-ci.yml

Set Variables



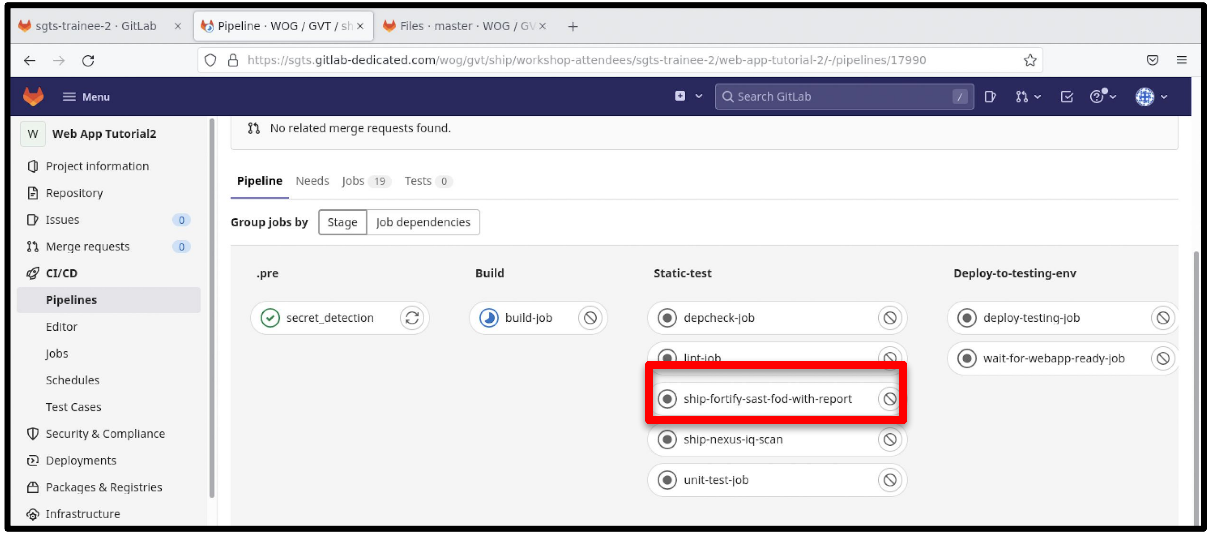
Web App Tutorial2

Environment variables are configured by your administrator to be protected by default.

Type	↑ Key	Value	Protected	Masked	Environments
Variable	COSIGN_PASSWORD	*********************	✓	×	All (default)
Variable	FOD_PAT	*********************	✓	×	All (default)
Variable	FOD_RELEASE	*********************	✓	×	All (default)
Variable	FOD_USERNAME	*********************	✓	×	All (default)
Variable	NEXUS_IQ_APP_ID	*********************	✓	×	All (default)
Variable	NEXUS_IQ_PASSWORD	*********************	✓	×	All (default)
Variable	NEXUS_IQ_USERNAME	*********************	✓	×	All (default)

Variable	Value	Description
FOD_RELEASE	xxx	Release number
FOD_USERNAME	Username	Username of FOD
FOD_PAT	Personal access	Your generated PAT

Adds one more job



sgts-trainee-2 · GitLab | Pipeline · WOG / GVT / sh | Files · master · WOG / GV

https://sgts.gitlab-dedicated.com/wog/gvt/ship/workshop-attendees/sgts-trainee-2/web-app-tutorial-2/-/pipelines/17990

No related merge requests found.

Group jobs by: Stage

.pre	Build	Static-test	Deploy-to-testing-env
secret_detection	build-job	depcheck-job	deploy-testing-job
		lint-job	wait-for-webapp-ready-job
		ship-fortify-sast-fod-with-report	
		ship-nexus-iq-scan	
		unit-test-job	

Thank You