A Guide to Using The Cryptosoft SBOM Creation Utility

The OWASP ® Dependency-Track offering requires a CycloneDX formatted SBOM (Software Bill-Of-Materials) as input to its capabilities. You can create this SBOM yourself or Cryptosoft provides a utility to help create one for you.

We provide GitHub Actions to create the SBOM or services to create one for your custom toolchain (we have provided an example for Jenkins).

1. Using Cryptosoft Github Actions in your Github Repository

We simplify that by making available GitHub Actions for inclusion into your Github workflow. Here is the link to our Actions in the Github Marketplace (https://github.com/marketplace?type=&verification=&query=cryptosoft+).

We have Four GitHub Actions: Two for uploading a GitHub repository's SBOM to the Dependency-Track and two actions for aggregating a GitHub repository's SBOM with an existing project in Dependency-Track

- `CryptosoftInc/Dependency-Track-Javascript@1.0.0`
- `CryptosoftInc/Dependency-Track@1.0.0`
- `CryptosoftInc/Aggregate-Sbom-Javascript@1.0.0`
- `CryptosoftInc/Aggregate-Sbom@1.0.0`

Languages Supported for - `CryptosoftInc/Dependency-Track-Javascript@1.0.0`

- All Javascript projects

Languages Supported for - `CryptosoftInc/Dependency-Track@1.0.0 are given below:`

Language/Platform	Package format	Transitive dependencies
java	maven (pom.xml [1]), gradle (build.gradle, .kts), scala (sbt), bazel	Yes unless pom.xml is manually parsed due to unavailability of maven or errors
php	composer.lock	Yes
python	pyproject.toml, setup.py, requirements.txt [2], Pipfile.lock, poetry.lock, bdist_wheel, .whl, .egg-info (Tool is under enhancement for poetry.lock file , for poetry.lock click here to generate SBOM and send to dependency-track)	Yes using the automatic pip install/freeze. When disabled, only with Pipfile.lock and poetry.lock
go	binary, go.mod, go.sum, Gopkg.lock	Yes except binary
ruby	Gemfile.lock, gemspec	Only for Gemfile.lock
rust	binary, Cargo.toml, Cargo.lock	Only for Cargo.lock

.Net	.csproj, packages.config, project.assets.json [3], packages.lock.json, .nupkg	Only for project.assets.json, packages.lock.json
dart	pubspec.lock, pubspec.yaml	Only for pubspec.lock
haskell	cabal.project.freeze	Yes
c/c++	conan.lock, conanfile.txt	Yes only for conan.lock
clojure	Clojure CLI (deps.edn), Leiningen (project.clj)	Yes unless the files are parsed manually due to lack of clojure cli or leiningen command

For more information about the supported languages, visit this link https://github.com/CycloneDX/cdxgen

How to implement our Action in your Repository

Pre-requisites:

1. Make sure the dependency file exists in the root directory of your GitHub repository.

   A dependency file is the file that lists the external libraries, packages, or modules that a project depends on to function properly. It specifies the specific versions or ranges of versions that are compatible with the project.
   Example:
   JavaScript: package.json (used by npm, Yarn)
   Python: requirements.txt (used by pip)
   Ruby: Gemfile (used by Bundler)
   Java: pom.xml (used by Apache Maven)
   PHP: composer.json (used by Composer)
   .NET: packages.config (used by NuGet)
   Go: go.mod (used by Go Modules)

2. Create an Administrator API Key from the Dependency-Track UI.

   The Administrator API key is used to authenticate the user to access the endpoint. It serves as a unique identifier or token that allows applications to authenticate themselves and gain access to specific resources.

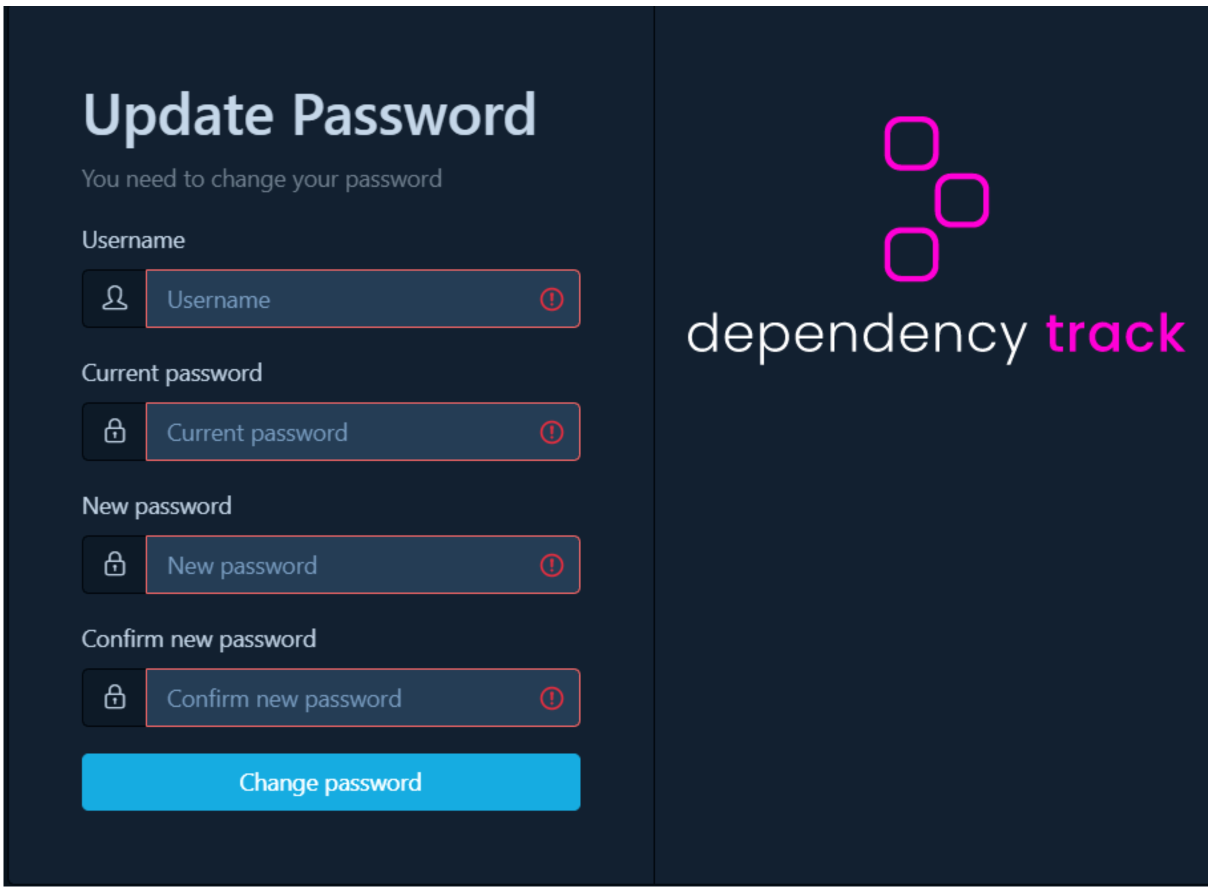
Steps to Create the Administrator API key:

1. Once you have successfully subscribed to the Cryptosoft OWASP Dependency-Track service you will receive two URLs, one is for your Dependency-Track instance and the second is a ‘backend url’ for use in creating your SBOM with this utility.
2. Enter the Dependency-Track instance URL in your browser.
3. Login using the default credentials.

   Username – admin

   Password – admin

4. You will be asked to reset the password, please reset the password and login again to the Dependency-Track website.



Update Password

You need to change your password

Username

Username

Current password

Current password

New password

New password

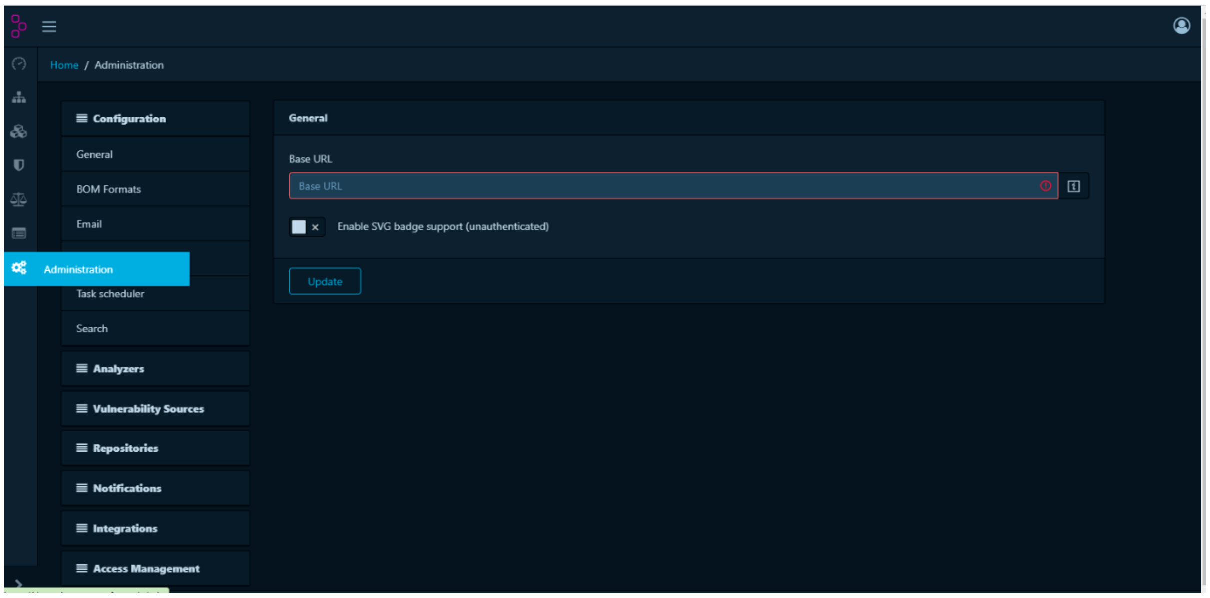
Confirm new password

Confirm new password

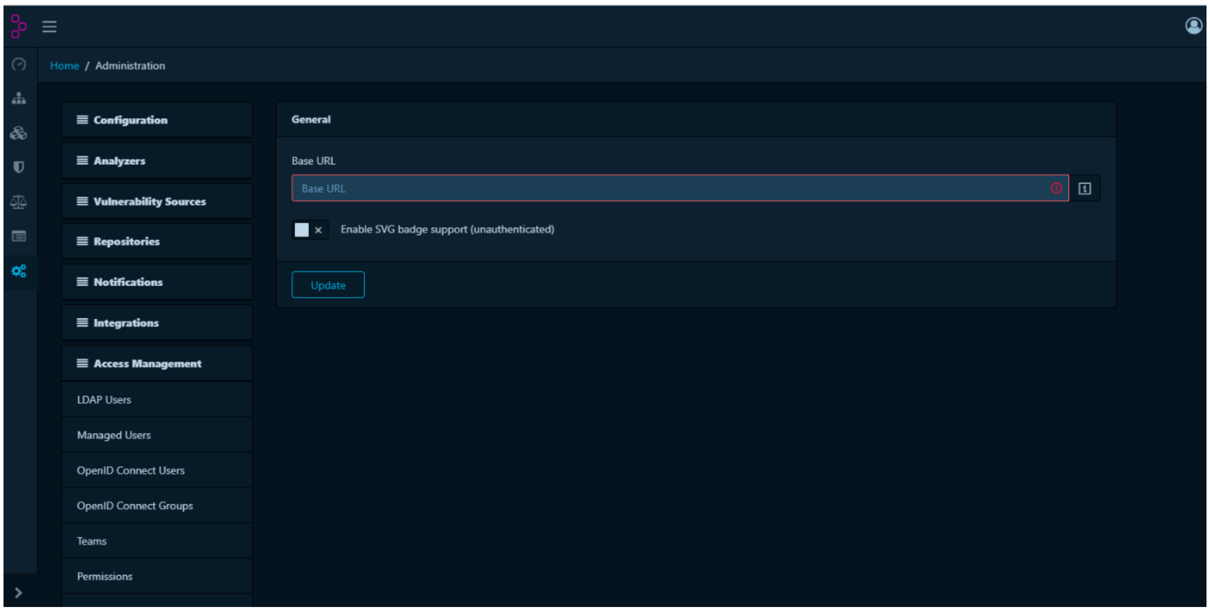
Change password

dependency track

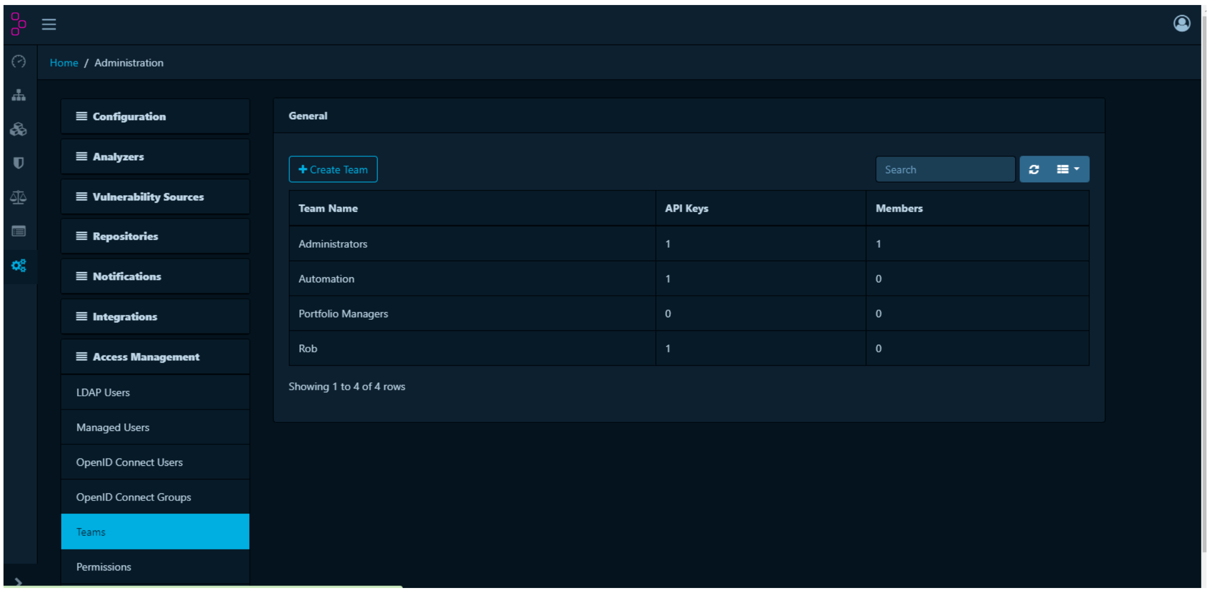
5. After login, In the left sidebar, click the “Administration” link



6. In the Administration page, click “Access Management”



7. Click on “Teams” under Access Management

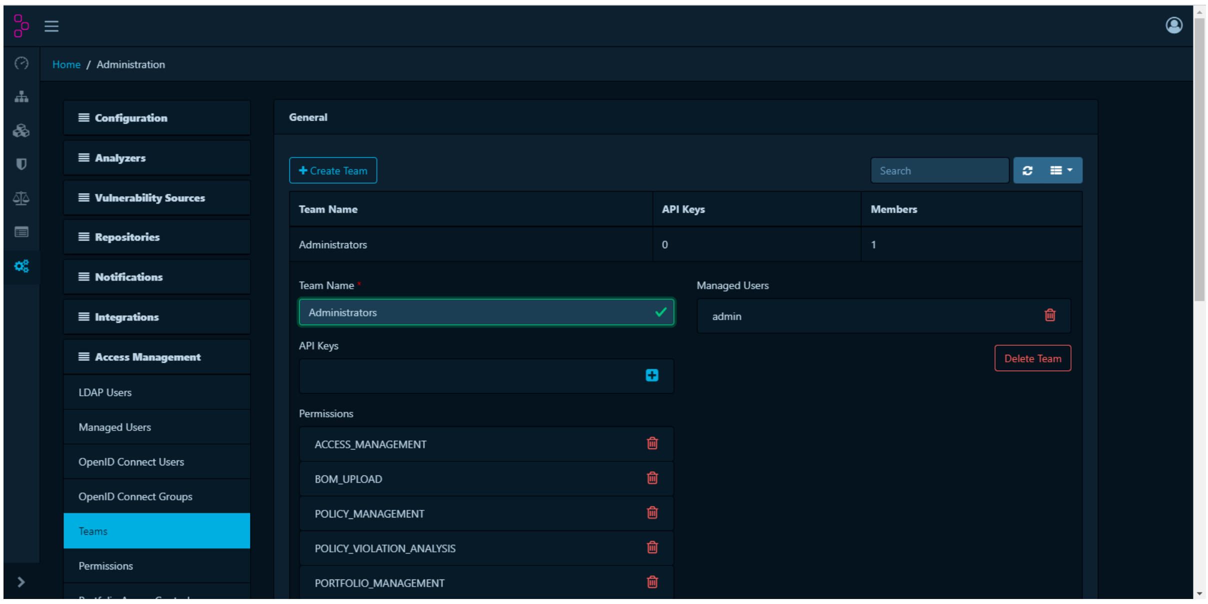

Home / Administration

General

Team Name	API Keys	Members
Administrators	1	1
Automation	1	0
Portfolio Managers	0	0
Rob	1	0

Showing 1 to 4 of 4 rows

8. Click on “Administrators” under the Team Name


Home / Administration

General

Team Name	API Keys	Members
Administrators	0	1

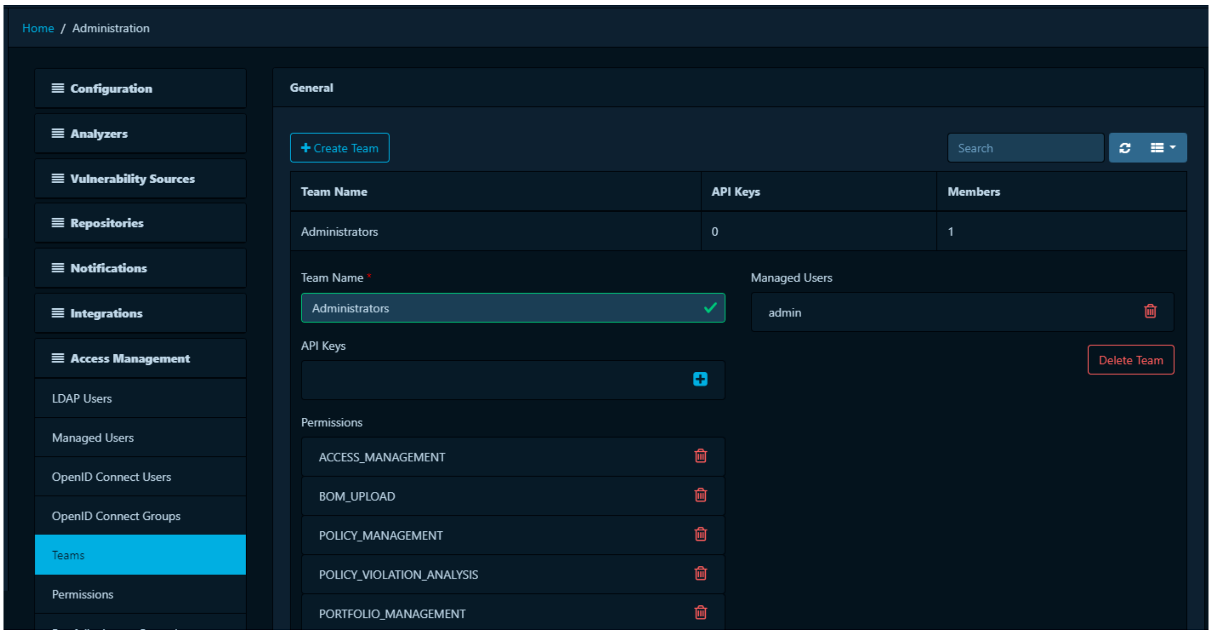
Team Name *: Administrators

Managed Users: admin

API Keys

Permissions
- ACCESS_MANAGEMENT
- BOM_UPLOAD
- POLICY_MANAGEMENT
- POLICY_VIOLATION_ANALYSIS
- PORTFOLIO_MANAGEMENT

9. Click on the + icon under API Keys


Home / Administration

General

Create Team

Team Name	API Keys	Members
Administrators	0	1

Team Name *
Administrators

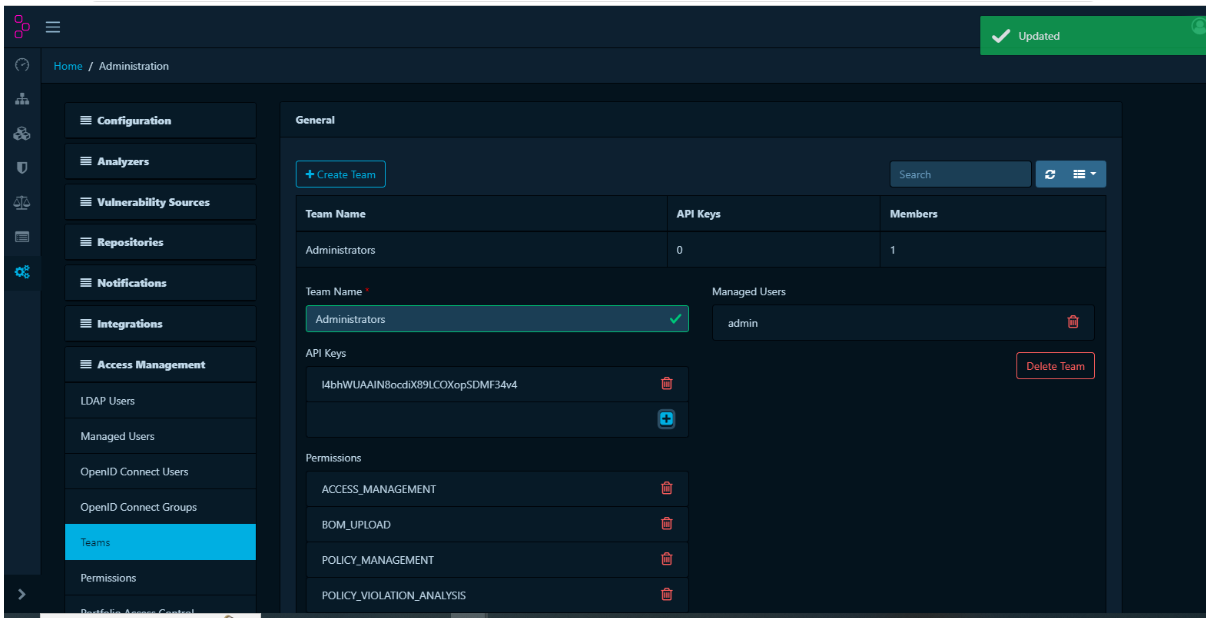
API Keys

Managed Users
admin

Delete Team

Permissions
ACCESS_MANAGEMENT
BOM_UPLOAD
POLICY_MANAGEMENT
POLICY_VIOLATION_ANALYSIS
PORTFOLIO_MANAGEMENT

10. A new API key will now be generated


Updated

Home / Administration

General

Create Team

Team Name	API Keys	Members
Administrators	0	1

Team Name *
Administrators

API Keys
l4bhWUAAIN8ocdiX89LCOXopSDMF34v4

Managed Users
admin

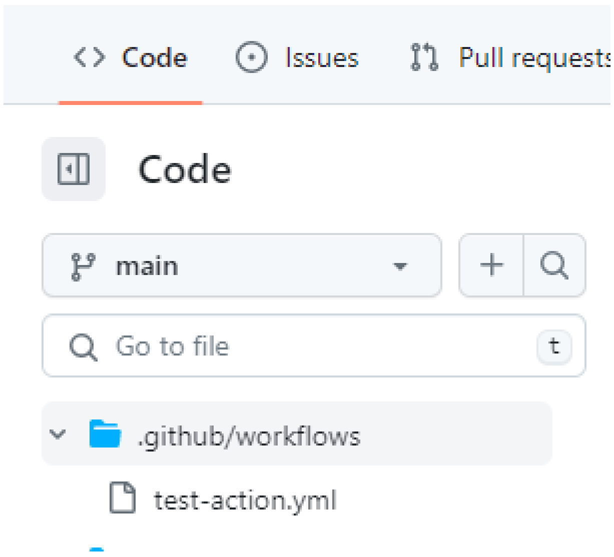
Delete Team

Permissions
ACCESS_MANAGEMENT
BOM_UPLOAD
POLICY_MANAGEMENT
POLICY_VIOLATION_ANALYSIS

Steps to run our Action:

1. Create a workflow in the root of your GitHub repository as shown.

   .github/workflows/<filename>.yml



2. Copy/paste our GitHub Action code snippet in the created workflow yml file.

```
name: Your-Workflow-Name
on: push
jobs:
  myJob:
    runs-on: ubuntu-latest
    steps:
    - name:Cryptosoft-SBOM-generator
      id: Cryptosoft-SBOM-generator
      uses: CryptosoftInc/Dependency-Track-Javascript@1.0.0
      with:
        dt-url: <your dt url>
        api-key: ${{ secrets.apiKey }}
        project-name: <your project name>
        project-version: <your project version >
```

Provide the inputs as mentioned:

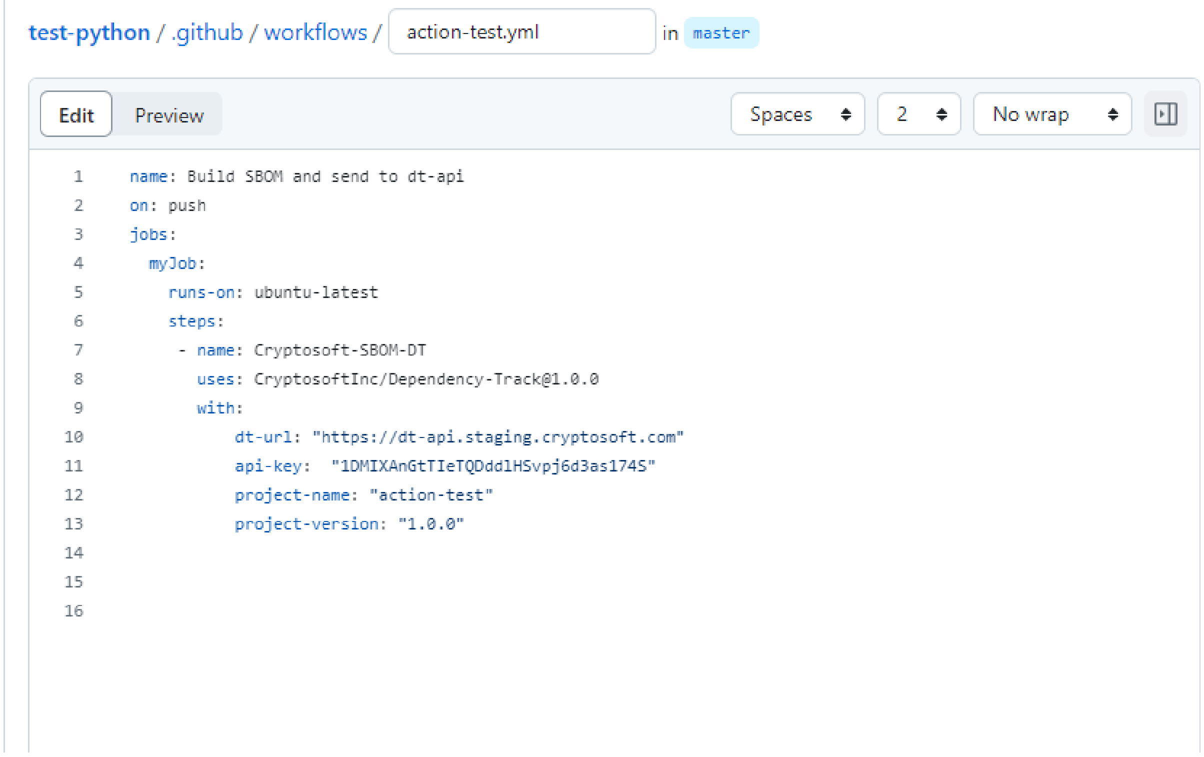
project-name: Enter your project name

project-version: Enter the project version

dt-url – Enter your Dependency-Track “backend URL”

api-key – Place your API Key in the GitHub secrets and pass the secrets ref or directly pass the API Key in inputs

Example:

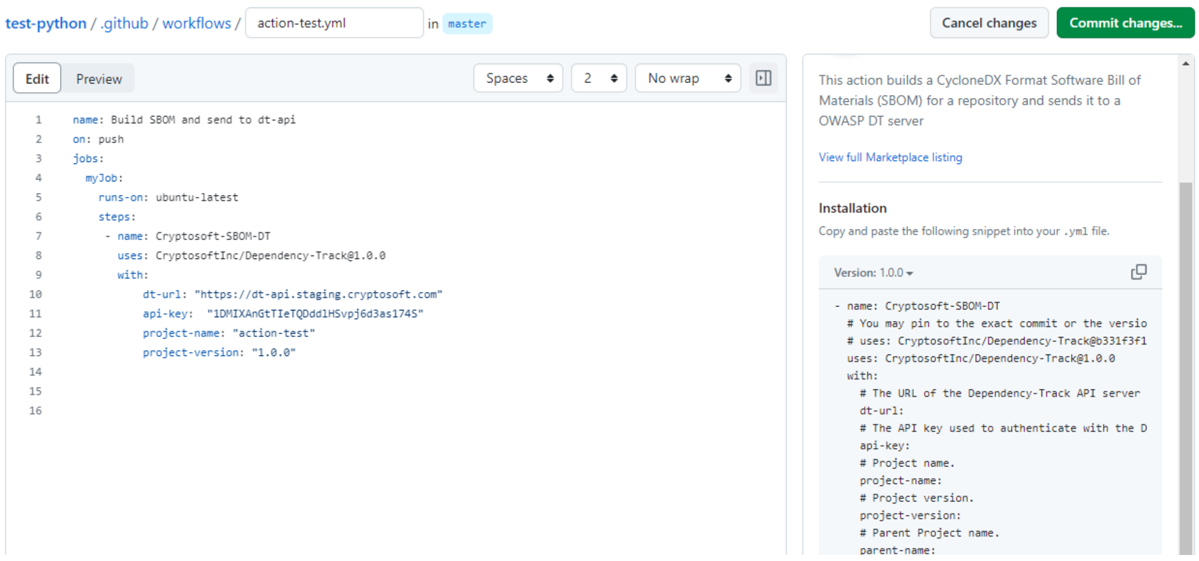

test-python / .github / workflows / action-test.yml in master
Edit
Preview
Spaces
2
No wrap

```yaml
name: Build SBOM and send to dt-api
on: push
jobs:
  myJob:
    runs-on: ubuntu-latest
    steps:
     - name: Cryptosoft-SBOM-DT
       uses: CryptosoftInc/Dependency-Track@1.0.0
       with:
           dt-url: "https://dt-api.staging.cryptosoft.com"
           api-key:  "1DMIXAnGtTIeTQDdd1HSvpj6d3as174S"
           project-name: "action-test"
           project-version: "1.0.0"



```

3. Click on the “Commit changes” button.


test-python / .github / workflows / action-test.yml in master
Cancel changes
Commit changes...
Edit
Preview
Spaces
2
No wrap
View full Marketplace listing
Version: 1.0.0

```yaml
name: Build SBOM and send to dt-api
on: push
jobs:
  myJob:
    runs-on: ubuntu-latest
    steps:
     - name: Cryptosoft-SBOM-DT
       uses: CryptosoftInc/Dependency-Track@1.0.0
       with:
           dt-url: "https://dt-api.staging.cryptosoft.com"
           api-key:  "1DMIXAnGtTIeTQDdd1HSvpj6d3as174S"
           project-name: "action-test"
           project-version: "1.0.0"



```

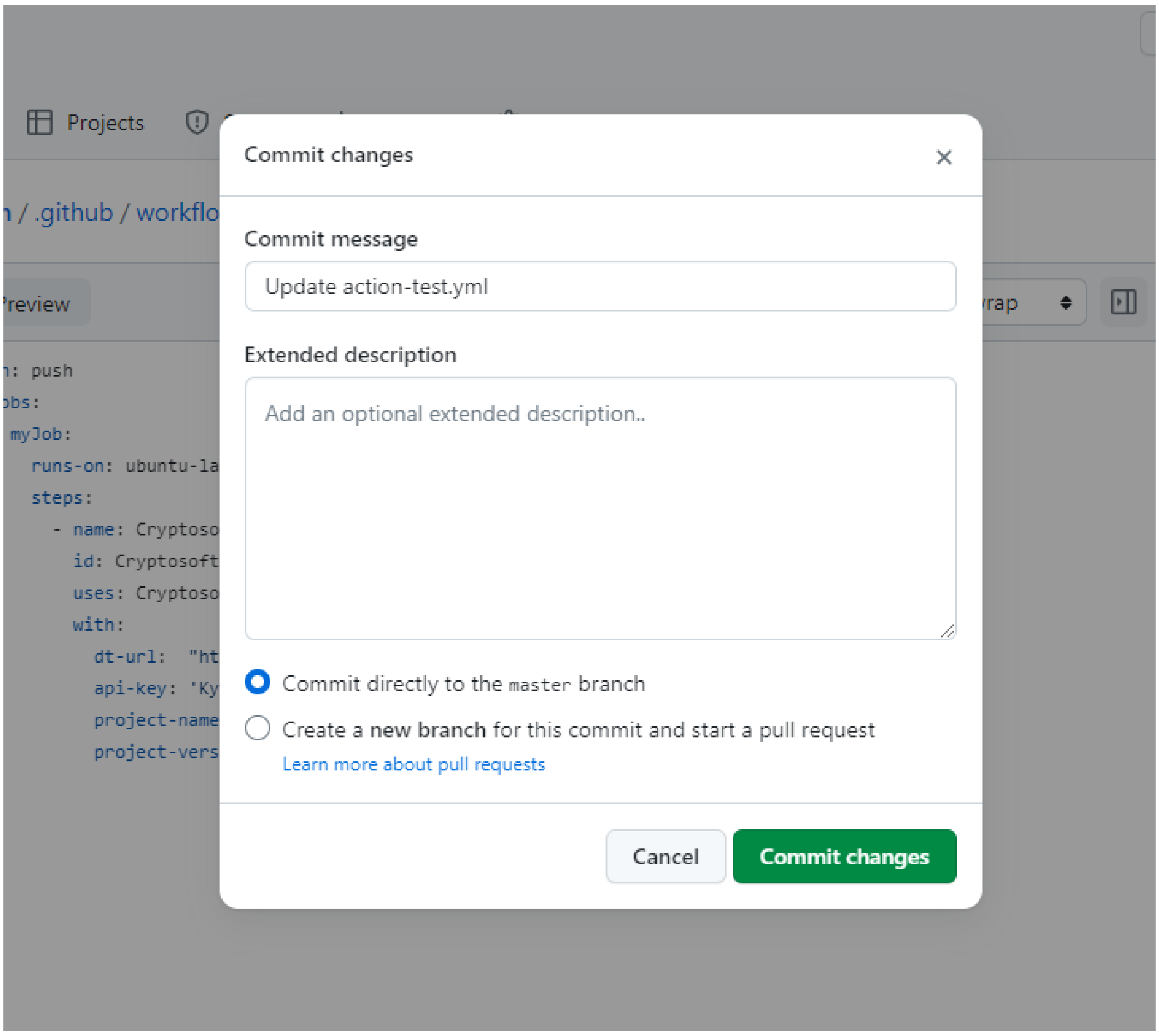
This action builds a CycloneDX Format Software Bill of Materials (SBOM) for a repository and sends it to a OWASP DT server

Installation

Copy and paste the following snippet into your `.yml` file.

```yaml
- name: Cryptosoft-SBOM-DT
  # You may pin to the exact commit or the versio
  # uses: CryptosoftInc/Dependency-Track@b331f3f1
  uses: CryptosoftInc/Dependency-Track@1.0.0
  with:
    # The URL of the Dependency-Track API server
    dt-url:
    # The API key used to authenticate with the D
    api-key:
    # Project name.
    project-name:
    # Project version.
    project-version:
    # Parent Project name.
    parent-name:
```

4. Click on the “Commit changes” button in the dialog box


Commit changes

Commit message

Update action-test.yml

Extended description

Add an optional extended description..

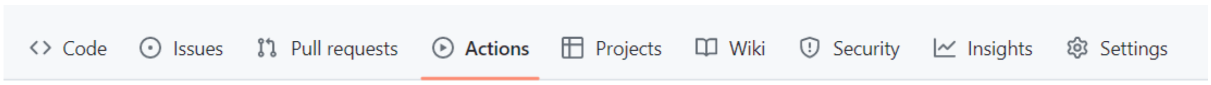
[✓ Commit directly to the `master` branch]

Create a **new branch** for this commit and start a pull request
Learn more about pull requests

```
: push
obs:
 myJob:
   runs-on: ubuntu-la
   steps:
     - name: Cryptoso
       id: Cryptosoft
       uses: Cryptoso
       with:
         dt-url:  "ht
         api-key: 'Ky
         project-name
         project-vers
```

5. After clicking the “Commit changes” button, the Action will be run, the SBOM will be generated for the GitHub repository and it will be uploaded to the provided Dependency-Track backend url.

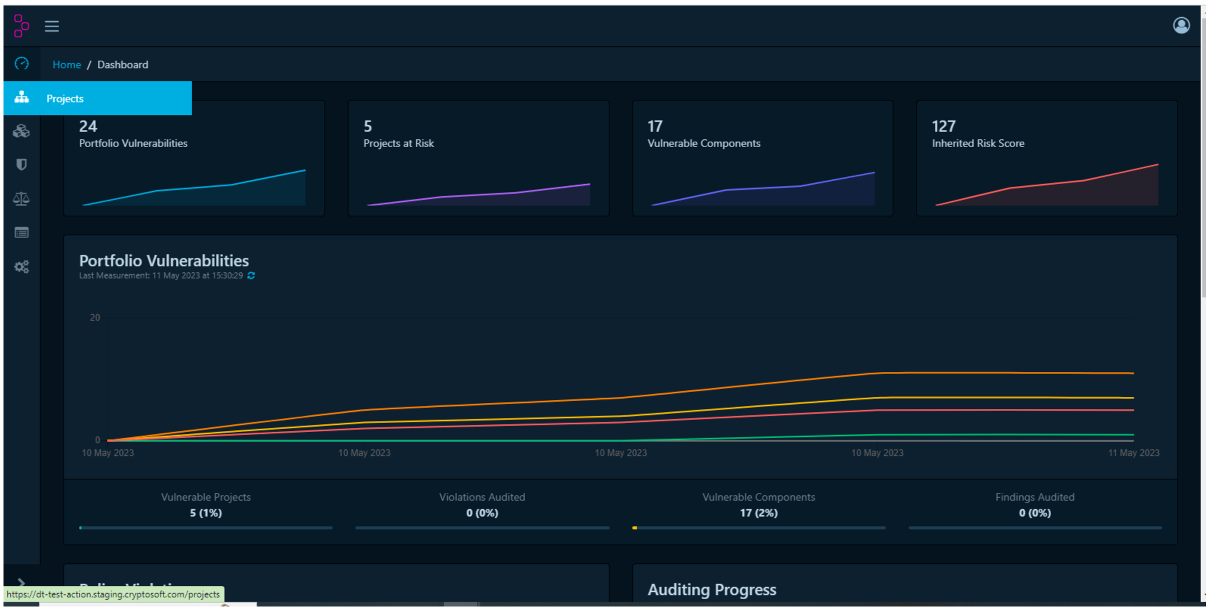
6. You can see your GitHub Action status by clicking on “Actions” tab as shown below:



7. Enter the Dependency-Track instance URL in the browser.

8. Login to the website

9. Go to the Projects



Home / Dashboard

Projects

24	5	17	127
Portfolio Vulnerabilities	Projects at Risk	Vulnerable Components	Inherited Risk Score

Portfolio Vulnerabilities

Last Measurement: 11 May 2023 at 15:30:29

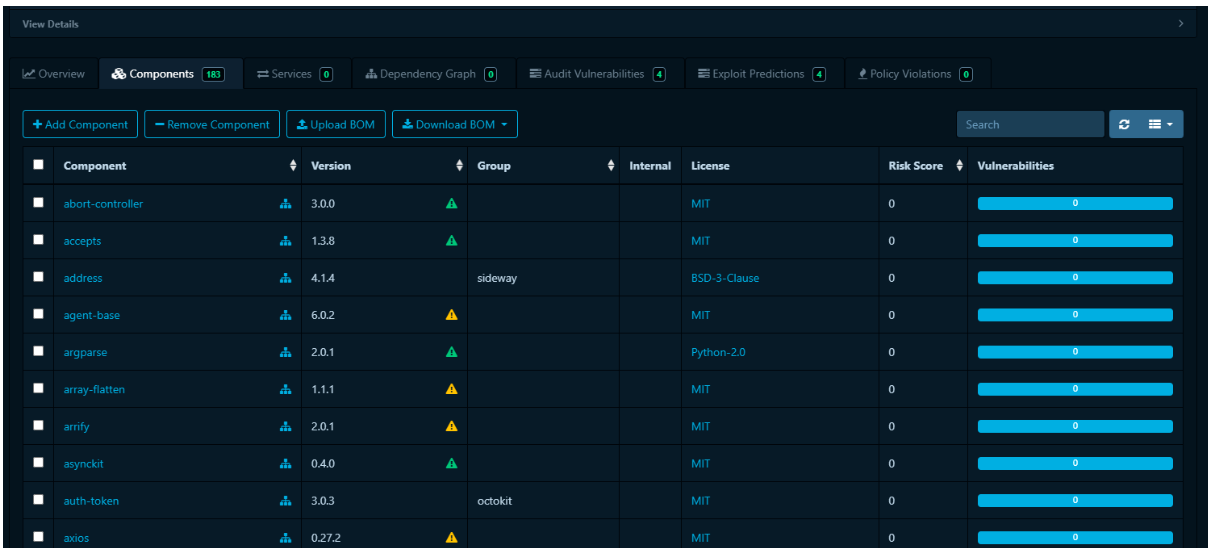
Vulnerable Projects	Violations Audited	Vulnerable Components	Findings Audited
5 (1%)	0 (0%)	17 (2%)	0 (0%)

Auditing Progress

https://dt-test-action.staging.cryptosoft.com/projects

10. Click on the created project name.

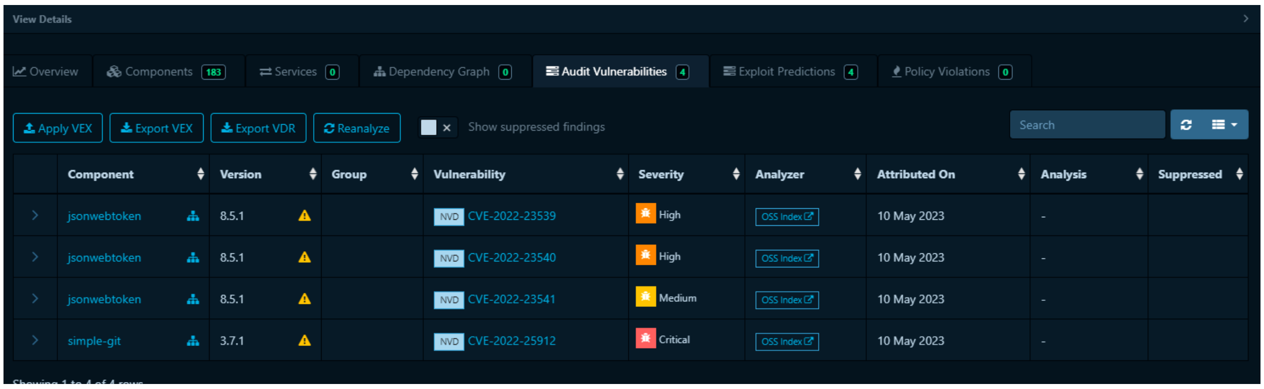
11. Click on components in the navigation bar to view components of the project.



View Details

Component	Version	Group	Internal	License	Risk Score	Vulnerabilities
abort-controller	3.0.0			MIT	0	0
accepts	1.3.8			MIT	0	0
address	4.1.4	sideway		BSD-3-Clause	0	0
agent-base	6.0.2			MIT	0	0
argparse	2.0.1			Python-2.0	0	0
array-flatten	1.1.1			MIT	0	0
arrify	2.0.1			MIT	0	0
asynckit	0.4.0			MIT	0	0
auth-token	3.0.3	octokit		MIT	0	0
axios	0.27.2			MIT	0	0

12. Click on “Audit Vulnerabilities” in the navigation bar to view the vulnerabilities.



	Component	Version	Group	Vulnerability	Severity	Analyzer	Attributed On	Analysis	Suppressed
›	jsonwebtoken	8.5.1 ⚠		NVD CVE-2022-23539	High	OSS Index	10 May 2023	-	
›	jsonwebtoken	8.5.1 ⚠		NVD CVE-2022-23540	High	OSS Index	10 May 2023	-	
›	jsonwebtoken	8.5.1 ⚠		NVD CVE-2022-23541	Medium	OSS Index	10 May 2023	-	
›	simple-git	3.7.1 ⚠		NVD CVE-2022-25912	Critical	OSS Index	10 May 2023	-	

2. Creating multiple repositories within a single parent project

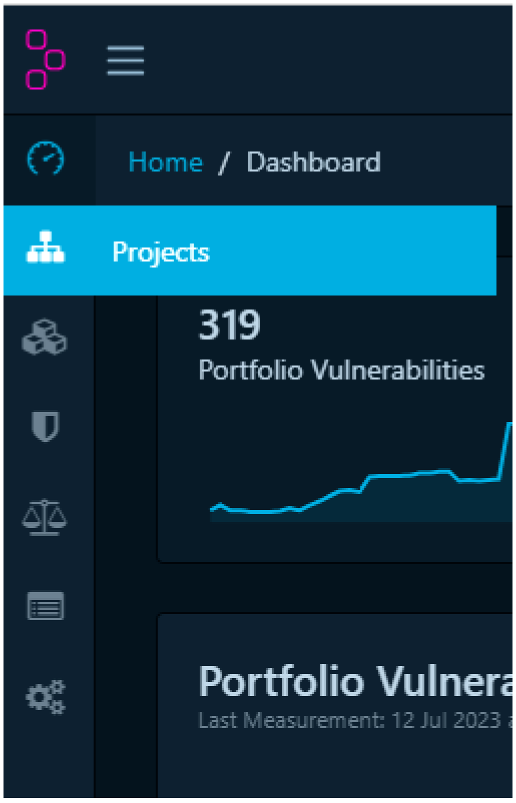
In the previous steps we learned how to create an SBOM for a project using our GitHub Actions. The following steps will guide you on how to create SBOMs for multiple repositories within a single Dependency-Track parent project.

The first step is to create the parent project for the repositories.

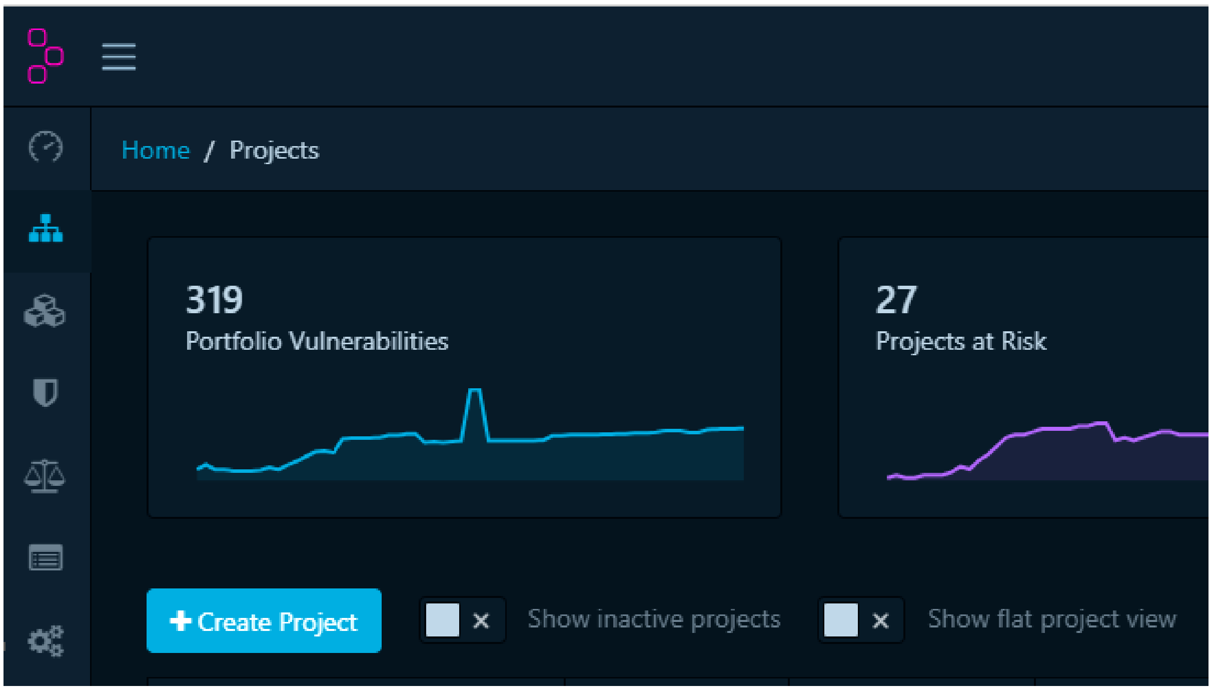
2.1 Creating the parent project:

i) Login to the Dependency-Track using your credentials. If you haven't set up the Administrator API key yet, Please follow the steps in the Steps to Create the Administrator API key section.

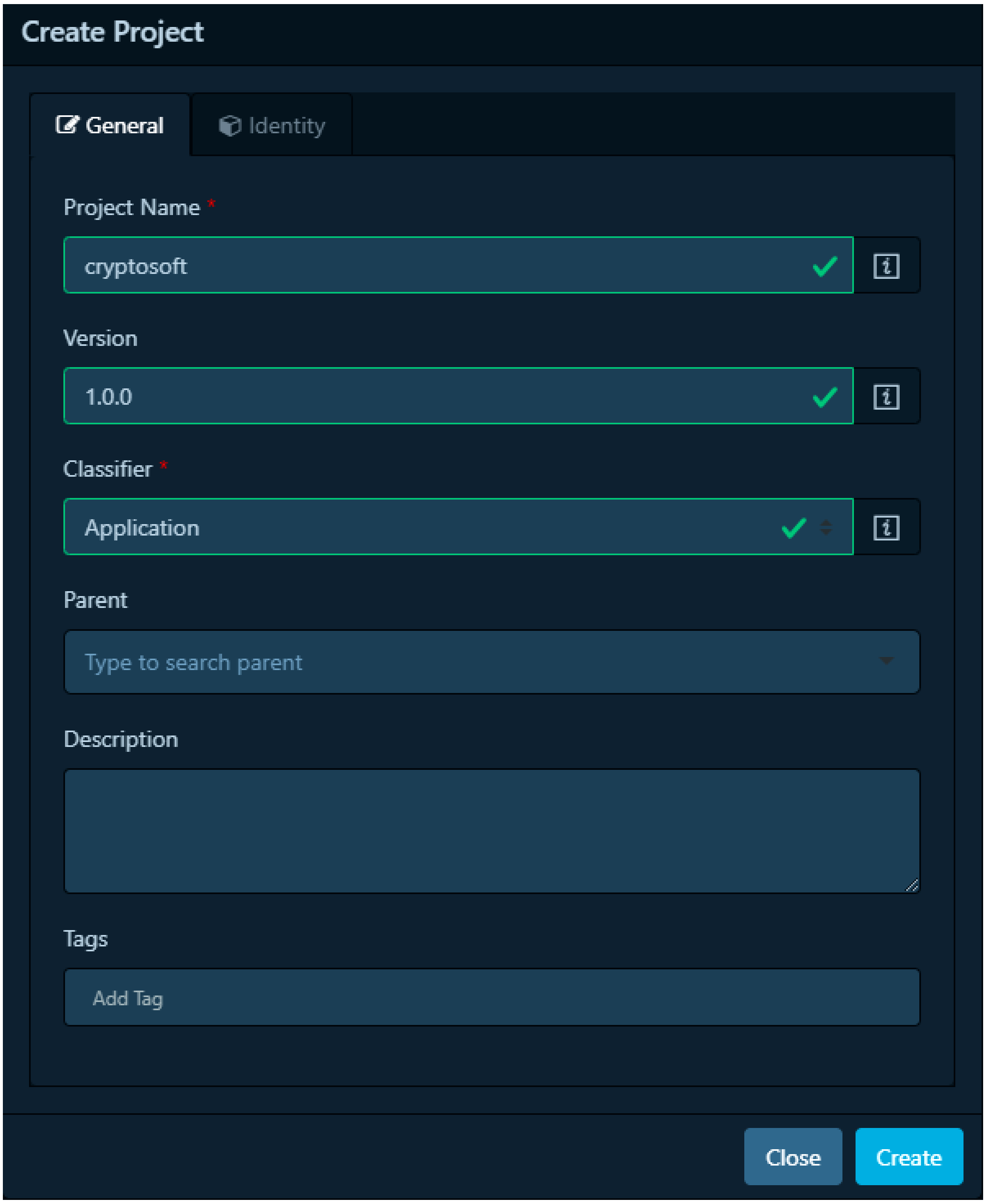
ii) Click the "projects" link in the sidebar



iii) Now the Projects page will be displayed. On the Projects page, Click the "Create Project" button



iv) A “Create Project” popup will be displayed. Fill in the project details - project name, project version and classifier. Then click the “Create” button to create a new project.


Create Project

General | Identity

Project Name *: cryptosoft

Version: 1.0.0

Classifier *: Application

Parent: Type to search parent

Description:

Tags: Add Tag

Close | Create

v) Now that we have created a parent project, we can proceed to create multiple repositories below it.

2.2. Creating multiple repositories within a parent project

1. After creating a parent project, follow the steps in Steps to run our Action, to create a workflow.
2. Copy/paste our GitHub Action code snippet in the created workflow yml file.

```
name: Your-Workflow-Name
on: push
jobs:
  myJob:
    runs-on: ubuntu-latest
    steps:
    - name:Cryptosoft-SBOM-generator
      id: Cryptosoft-SBOM-generator
      uses: CryptosoftInc/Dependency-Track-Javascript@1.0.0
      with:
        dt-url: <your dt url>
        api-key: ${{ secrets.apiKey }}
        project-name: <your project name>
        project-version: <your project version >
        parent-name: <your parent project name>
        parent-version: <your parent project version >
```

Provide the inputs as mentioned:

parent-name: Enter your parent project name

parent-version: Enter the parent project version

project-name: Enter your project name

project-version: Enter the project version

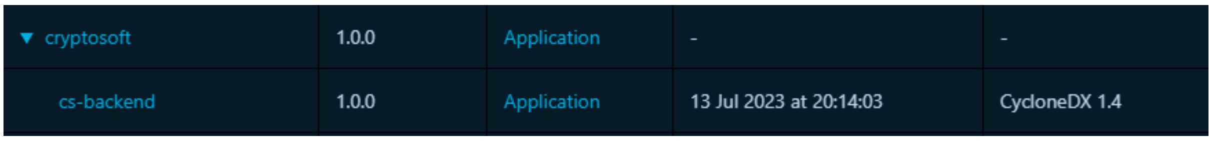
dt-url: Enter your Dependency-Track “backend URL”

api-key: Place your API Key in the GitHub secrets and pass the secrets ref or directly pass the API Key in inputs

Example

```yaml
name: sbom-test
on: push
jobs:
  myJob:
    runs-on: ubuntu-latest
    steps:
      - name: Cryptosoft-SBOM-generator
        id: Cryptosoft-SBOM-generator
        uses: CryptosoftInc/Dependency-Track-Javascript@1.0.0
        with:
          dt-url: "https://dt-api.staging.cryptosoft.com/api/v1/bom"
          api-key: "1DMIXAnGtTIeTQDdd1HSvpj6d3as174S "
          project-name: "cs-backend"
          project-version: "1.0.0"
          parent-name: "cryptosoft"
          parent-version: "1.0.0"
```

3. Follow the steps in Steps to run our Action to execute the action and view the project in Dependency-Track
4. Login to the Dependency-Track UI and navigate to the “Projects” page, and you will see the new project has been created under the designated parent project.



▼ cryptosoft	1.0.0	Application	-	-
cs-backend	1.0.0	Application	13 Jul 2023 at 20:14:03	CycloneDX 1.4

3. Creating an SBOM from multiple repositories under a single project

The upcoming steps will guide you in combining a GitHub repository's SBOM with an existing Dependency-Track project to aggregate the SBOMs.

By using this method, we can upload the SBOMs of multiple repositories to a single project, which results in obtaining the aggregated SBOM within the single project.

For Javascript projects, use **`CryptosoftInc/Aggregate-Sbom-Javascript@1.0.0`**

For non-Javascript projects, use **`CryptosoftInc/Aggregate-Sbom@1.0.0`**

These actions will enable you to aggregate the SBOMs accordingly.

1. Follow the steps in Steps to run our Action, to create a workflow.
2. Copy/paste our GitHub Action code snippet in the created workflow yml file.

```
name: Your-Workflow-Name
on: push
jobs:
  myJob:
    runs-on: ubuntu-latest
    steps:
    - name:Cryptosoft-SBOM-Dependency-Track
      id: Cryptosoft-SBOM-Dependency-Track
      uses: CryptosoftInc/Aggregate-Sbom@1.0.0
      with:
        dt-url: <your dt url>
        api-key: ${{ secrets.apiKey }}
        project-name: <your project name>
        project-version: <your project version >
```

Provide the inputs as mentioned:

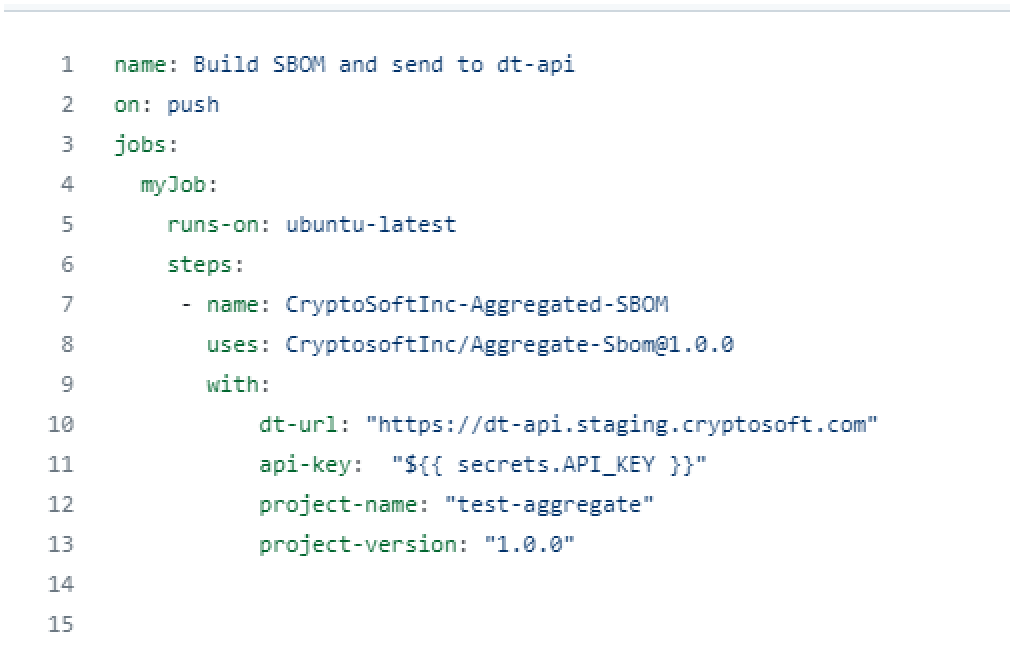
project-name: Enter your project name

project-version: Enter the project version

dt-url: Enter your Dependency-Track "backend URL"

api-key: Place your API Key in the GitHub secrets and pass the secrets ref or directly pass the API Key in inputs

Example



```yaml
name: Build SBOM and send to dt-api
on: push
jobs:
  myJob:
    runs-on: ubuntu-latest
    steps:
     - name: CryptoSoftInc-Aggregated-SBOM
       uses: CryptosoftInc/Aggregate-Sbom@1.0.0
       with:
           dt-url: "https://dt-api.staging.cryptosoft.com"
           api-key:  "${{ secrets.API_KEY }}"
           project-name: "test-aggregate"
           project-version: "1.0.0"


```

5. Follow the steps in Steps to run our Action to execute the action and view the project in Dependency-Track.
6. After logging into the Dependency-Track UI, navigate to the “Projects” page. You will be able to see the new project if it has been successfully created. In case the project already exists, you will find the aggregated SBOMs associated with that project.

4. Using a custom toolchain

The process consists of two steps: SBOM generation and uploading it to Dependency-Track.

For SBOM generation, you have various options available depending on your project requirements. You can utilize tools like CycloneDX Tool Center, Syft Tool, or Microsoft SBOM Tool. These tools enable you to generate an SBOM tailored to your specific project. We have provided later a sample for Jenkins which was created using the cdxgen tool.

Moving on to the second step, you need to invoke the Dependency-Track API with the necessary inputs to successfully upload the SBOM. To accomplish this, you can employ a sample curl command as follows:

```
curl -k -X POST "{Dependency-Track-backend-url}/api/v1/bom" \
        -H "Content-Type:multipart/form-data" \
        -H "X-Api-Key:${<DT-Api-key>}" \
        -F "autoCreate=true" \
        -F "projectName=<your-project-name>" \
        -F "projectVersion=<your-project-version>" \
        -F "bom=@<Generated-SBOM-file>"
```

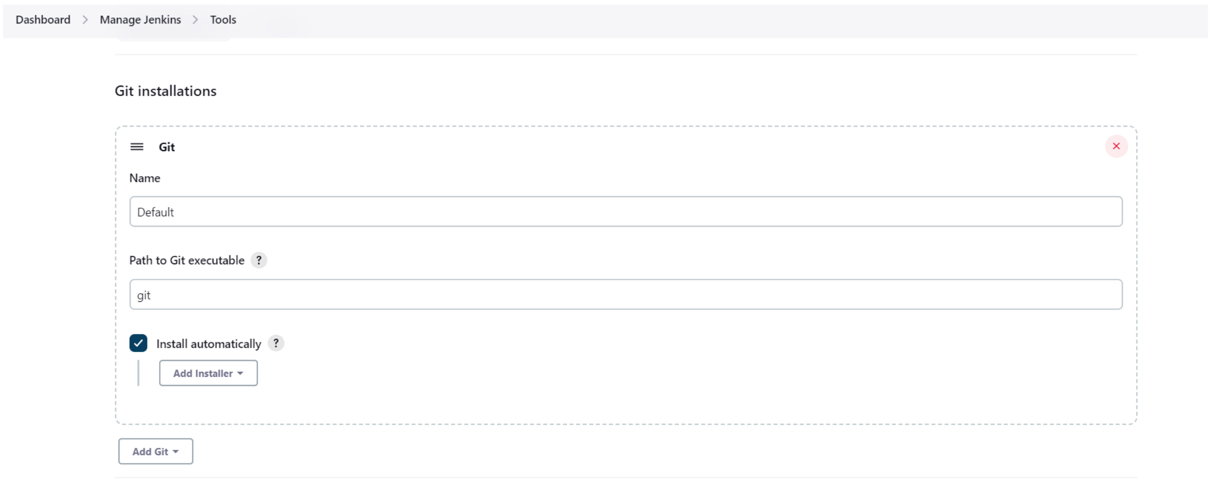
*****Generated SBOM file supported formats .xml and .json***

GENERAL TEMPLATE FOR USING THE SBOM UTILITY FOR JENKINS:

In the explanation below we are demonstrating how to generate the SBOM using Jenkins and upload it to Dependency-Track. We are using the cdxgen tool for the SBOM generation.

Jenkins Setup:

- Install **Git** plugin
- Go to the Jenkins Dashboard→Manage Jenkins→Tools where we can see the Git plugin.


Dashboard > Manage Jenkins > Tools

Git installations

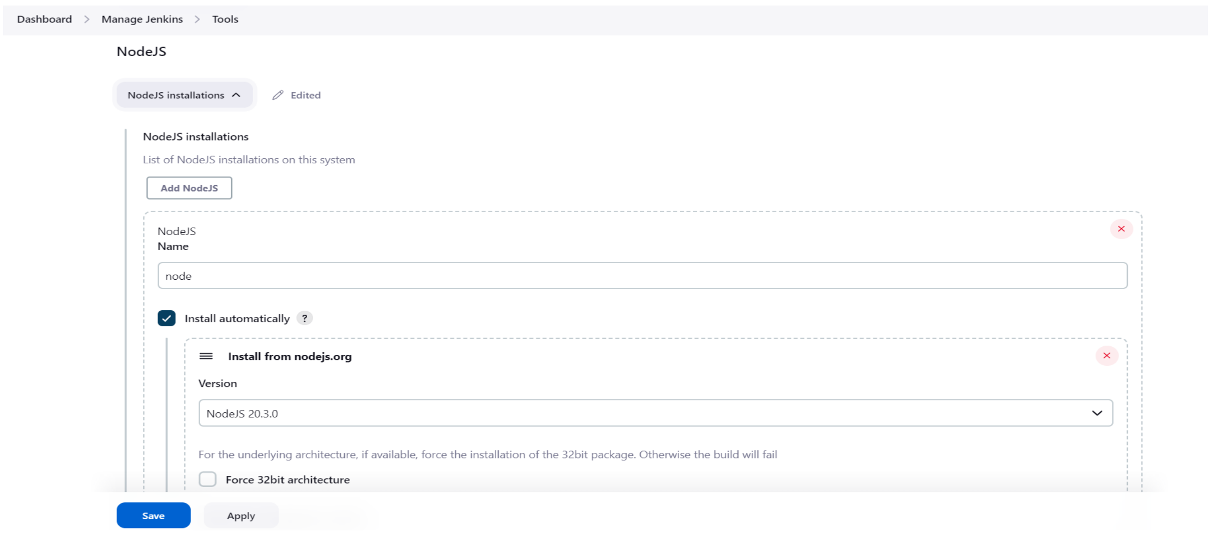
Git

Name: Default

Path to Git executable: git

☑ Install automatically

- Click on Add NodeJs and provide the name as **"git"** and check the Install automatically option with the latest version.
- Click on Apply with the above setting and then Click on Save.
- Install NodeJS plugin
- Go to the Jenkins Dashboard→Manage Jenkins→Tools where we can see the NodeJs plugin.
- Click on "Add NodeJs" and provide the name as **"node"** and check the Install automatically option with the latest version.
- Click on "Apply" with the above setting and then Click on Save.


Dashboard > Manage Jenkins > Tools

NodeJS

NodeJS installations — Edited

NodeJS installations

List of NodeJS installations on this system

NodeJS

Name: node

☑ Install automatically

Install from nodejs.org

Version: NodeJS 20.3.0

For the underlying architecture, if available, force the installation of the 32bit package. Otherwise the build will fail

☐ Force 32bit architecture

- The Jenkinsfile pipeline below will support NodeJS, ReactJs, Angular.
- Make sure the Jenkinsfile exists in the root directory of your Github repository.

```
pipeline {
    agent any
    tools {
        nodejs "node"                                                STEP 1
    }
    stages {
        stage('Checkout') {
            steps {
                checkout scm
            }
        }
        stage('Install Dependencies') {
            steps {
                sh 'npm install'                                     STEP 2
            }
        }
        stage('Install SBOM tool') {
            steps {
                sh 'npm install -g @cyclonedx/cdxgen@8.6.0'          STEP 3'
            }
        }
        stage('Generate SBOM') {
            steps {
                sh 'export FETCH_LICENSE=true && cdxgen -r -o bom.json'
                script {
                    def sbom = readFile('bom.json')
                    echo "Generated SBOM:\n$sbom"                    STEP 4
                }
            }
        }
        stage('Upload SBOM to Dependency-Track') {
            steps {
   withCredentials([string(credentialsId: '<Your apikey name>', variable:
'X_API_KEY')]) {
                    sh """
                    curl -k -X POST "https://<Your dependency-track backend
url>/api/v1/bom" \
                    -H "Content-Type:multipart/form-data" \
                    -H "X-Api-Key:${X_API_KEY}" \
                    -F "autoCreate=true" \
                    -F "projectName=<your project name>"\            STEP 5
                    -F "projectVersion=<Your project version>" \
                    -F "bom=@bom.json"
                    """ }
            }
        }
    }
}
```

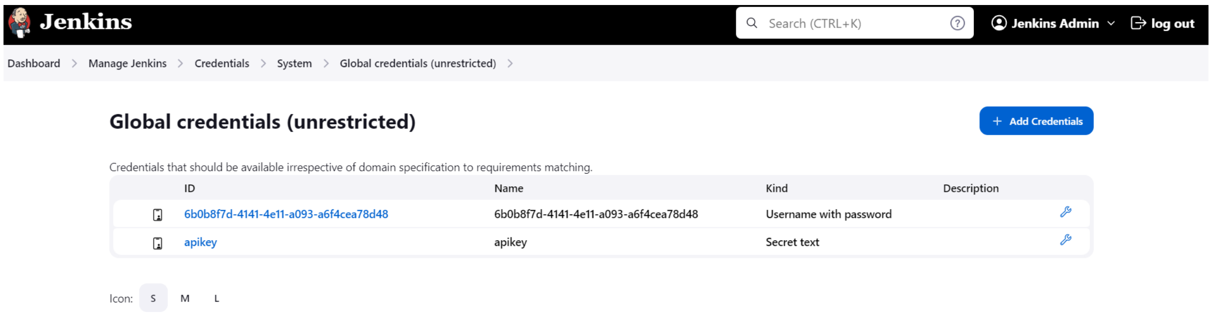
For other programming languages use project specific commands in **STEP 2** to install project dependencies.
Ex: Golang- “***go mod why***” , Python - “***pip install***” etc. as shown below:

```
stage('Install Dependencies') {
            steps {
                sh 'go mod why'                          STEP 2
            }
        }
```

- Once the changes are committed by adding the Jenkinsfile to your GitHub repository, we need to connect the GitHub repository to Jenkins and create a secret for storing the Dependency-Track APIKEY.

Generation of the Dependency-Track APIKEY as a Secret in Jenkins

- In the Jenkins pipeline we are passing the APIKEY of Dependency-Track as secret which needs to be added in the Credentials.
- Go to the Jenkins Dashboard→Manage Jenkins→Credentials→System→Global credentials (unrestricted) and click on **Add Credentials.**

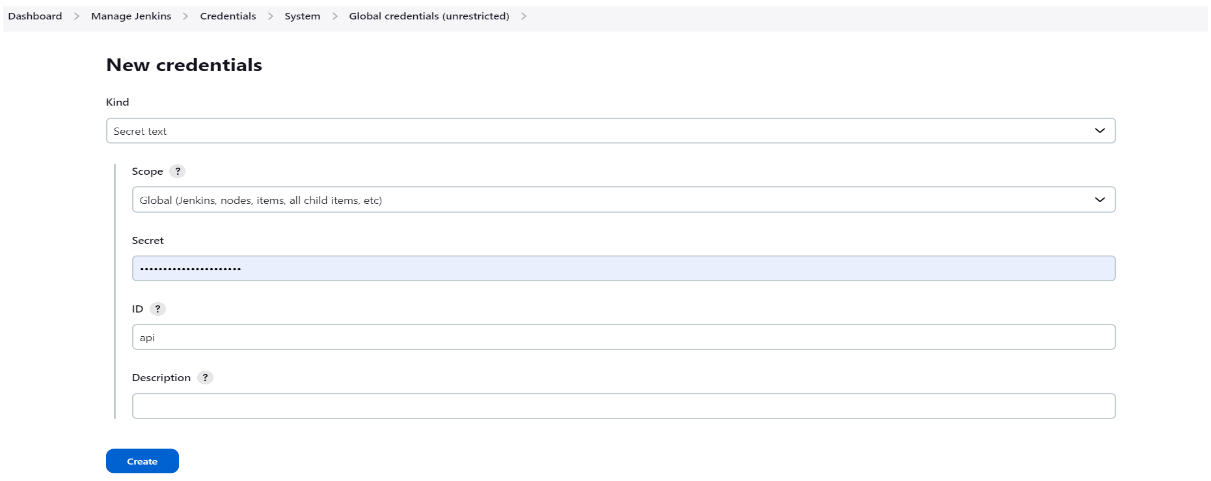

Jenkins
Search (CTRL+K)
Jenkins Admin
log out
Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >
+ Add Credentials
Icon: S M L

Global credentials (unrestricted)

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
6b0b8f7d-4141-4e11-a093-a6f4cea78d48	6b0b8f7d-4141-4e11-a093-a6f4cea78d48	Username with password	
apikey	apikey	Secret text	

- Now select the kinds as **Secret Text** and paste the APIKEY in the Secret option and provide a name in the ID


Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >
Create

New credentials

Kind: Secret text

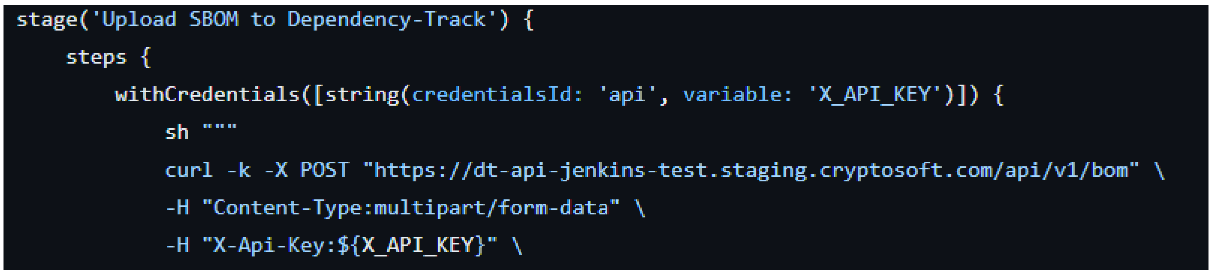
Scope: Global (Jenkins, nodes, items, all child items, etc)

Secret: ························

ID: api

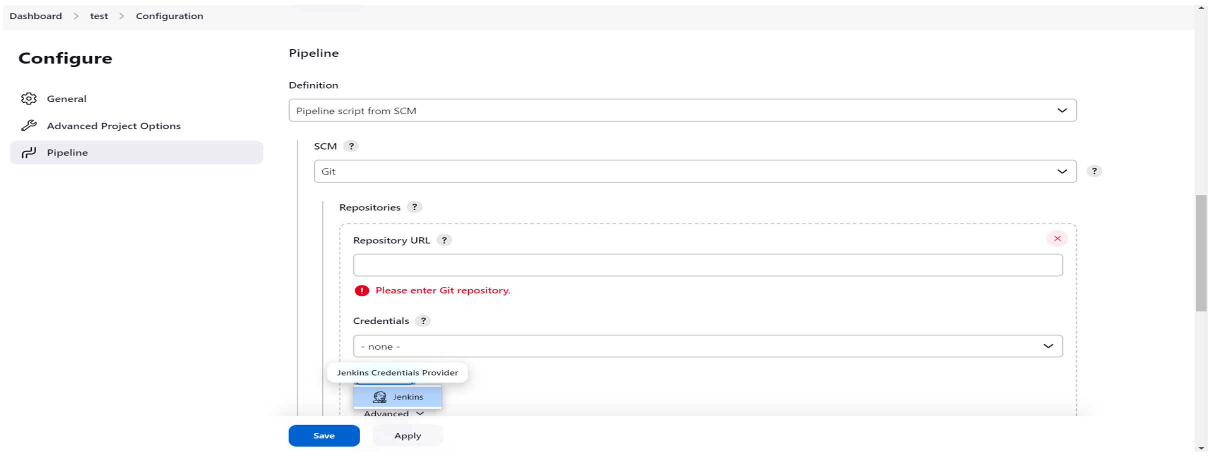
Description:

- Click on “Create”.
- This Secret will be used in the Jenkins pipeline for pushing the SBOM to the Dependency-Track backend.



```
stage('Upload SBOM to Dependency-Track') {
    steps {
        withCredentials([string(credentialsId: 'api', variable: 'X_API_KEY')]) {
            sh """
            curl -k -X POST "https://dt-api-jenkins-test.staging.cryptosoft.com/api/v1/bom" \
            -H "Content-Type:multipart/form-data" \
            -H "X-Api-Key:${X_API_KEY}" \
```

- Create the Jenkins pipeline project and then select the
  - Definition — Pipeline script from SCM
  - SCM — Git
  - Repository URL — GH repository for SBOM


Dashboard > test > Configuration

Configure

General
Advanced Project Options
Pipeline

Pipeline

Definition: Pipeline script from SCM

SCM: Git

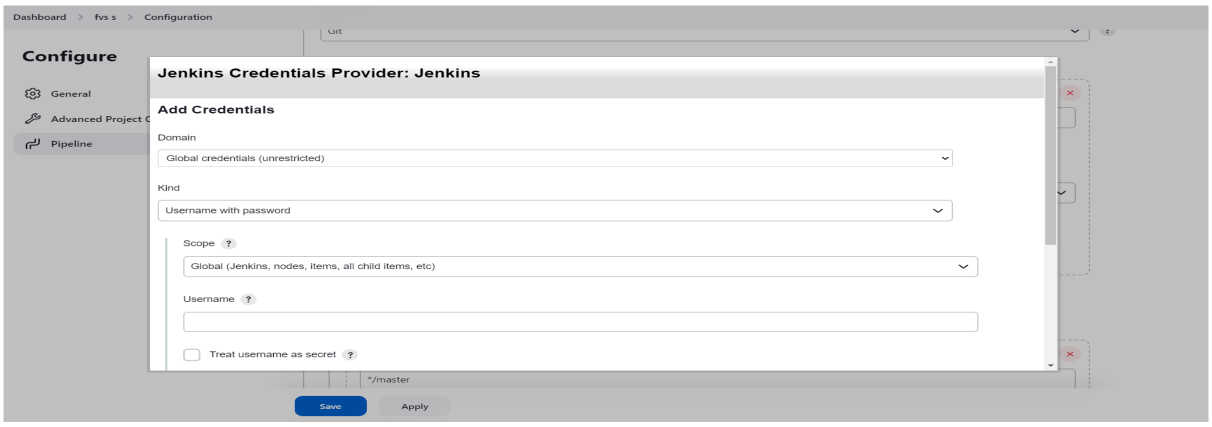
Repositories

Repository URL:

Please enter Git repository.

Credentials: - none -

  - Credentials — Add GitHub username and password as secret


Dashboard > fvs s > Configuration

Configure

General
Advanced Project O
Pipeline

Jenkins Credentials Provider: Jenkins

Add Credentials

Domain: Global credentials (unrestricted)

Kind: Username with password

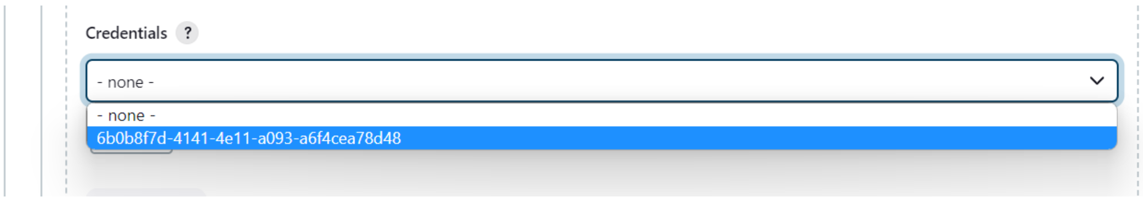
Scope: Global (Jenkins, nodes, items, all child items, etc)

Username:

☐ Treat username as secret

*/master

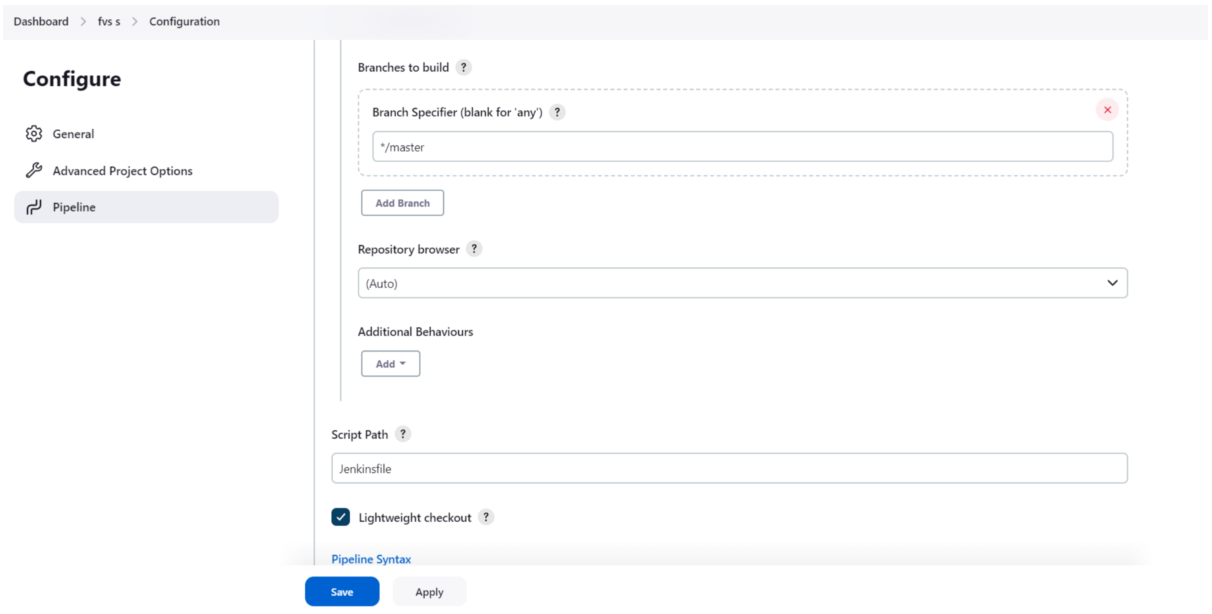
- Once the GitHub username and password are set as secret then we can use the name of the secret or ID which has been generated during the secret creation for connecting the GitHub repository.


Credentials

- none -

- none -
6b0b8f7d-4141-4e11-a093-a6f4cea78d48

- Mention the branch name and the Jenkinsfile for creating the SBOM



Dashboard > fvs s > Configuration

Configure

- General
- Advanced Project Options
- Pipeline

Branches to build ?

Branch Specifier (blank for 'any') ?: */master

Add Branch

Repository browser ?: (Auto)

Additional Behaviours

Add

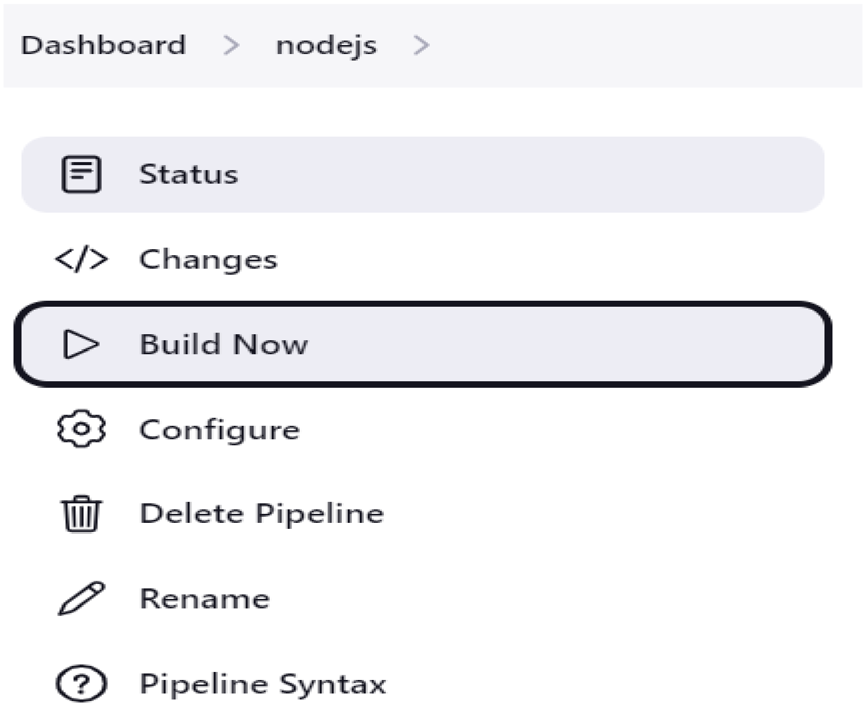
Script Path ?: Jenkinsfile

✓ Lightweight checkout ?

Pipeline Syntax

Save Apply

- Click on Apply
- Once the Jenkins Project is created then Click on the project and then click on the **Build Now** option on the left sidebar.



Dashboard > nodejs >

- Status
- Changes
- Build Now
- Configure
- Delete Pipeline
- Rename
- Pipeline Syntax

- Then we will see the build has been started which will run the Steps from the Jenkinsfile.
- Once the build process is completed successfully we can check the SBOM project name has been pushed to Dependency-Track.
- Go to the Dependency-Track frontend url and you will see the Project has been created with the version number.

- **Failure/Error Case**
- If the Jenkins server is running in kubernetes and the build steps have been killed or terminated with any memory error then we need to add the snippet below to the initiating pipeline in your Jenkinsfile:

```
pipeline {
    agent {
        kubernetes {
            cloud 'kubernetes'
            yaml """
                apiVersion: v1
                kind: Pod
                spec:
                  containers:
                  - name: jnlp
                    image: jenkins/inbound-agent:3107.v665000b_51092-15
                    resources:
                      limits:
                        cpu: '1'
                        memory: '2Gi'
                      requests:
                        cpu: '900m'
                        memory: '1Gi'
                    tty: true
            """
        }
          // Your Rest pipeline steps
    }

}
```

GITHUB WORKFLOW TO GENERATE SBOM FOR PYTHON WITH POETRY.LOCK

For Python projects with a poetry.lock dependency file, create a workflow in the root of your GitHub repository as shown in this step. Use the code snippet below to generate the SBOM and upload it to Dependency-Track.

```
name: test-actions
on:
  push:
jobs:
  myJob:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - name: Setup Python
        uses: actions/setup-python@v3.1.4
      - name: Install tool
        run: pip install cyclonedx-bom
      - name: Generate BOM
        run: |
             cyclonedx-py --poetry --format xml -o sbom.xml

      - name: Submit SBOM
        run: |
          curl -k -X POST "<Your DT URL>/api/v1/bom" \
            -H "Content-Type: multipart/form-data" \
            -H "X-Api-Key:<Your API_KEY>" \
            -F "autoCreate=true" \
            -F "projectName=<Your project Name>" \
            -F "projectVersion=<Your project version>" \
            -F "bom=@sbom.xml"
```

* Substitute your Dependency-Track url , api-key , project name and project version at the highlighted place.

END