**Synopsys Bridge CLI Guide**

Suggested Workflow:

1. Raj edit/save Word doc
2. Raj return updated doc to Robert
3. Robert port changes to Dita
4. Robert / Raj review updates – live session
5. Robert port final changes to Dita
6. Deploy

# Contents

[Chapter 1. Overview](#_bookmark0) [4](#_bookmark0)

[Overview](#_bookmark1) [4](#_bookmark1)

[Support Matrix](#_bookmark2) [4](#_bookmark2)

[Files and Directories](#_bookmark3) [5](#_bookmark3)

[Download Synopsys Bridge](#_bookmark4) [6](#_bookmark4)

[Chapter 2. Synopsys Bridge CLI](#_bookmark5) [7](#_bookmark5)

[Using Synopsys Bridge CLI](#_bookmark6) [7](#_bookmark6)

[Passing Arguments using a JSON file](#_bookmark7) [7](#_bookmark7)

[Passing Arguments using the CLI](#_bookmark8) [8](#_bookmark8)

[Using Synopsys Bridge CLI with Polaris](#_bookmark9) [10](#_bookmark9)

[Using Synopsys Bridge CLI with Black Duck](#_bookmark13) [12](#_bookmark13)

[Using Synopsys Bridge CLI with Coverity Connect](#_bookmark14) [14](#_bookmark14)

[Chapter 3. Synopsys Bridge CLI Reference](#_bookmark15) [18](#_bookmark15)

[Complete List of Synopsys Bridge Arguments](#_bookmark16) [18](#_bookmark16)

[Exit Codes](#_bookmark17) [33](#_bookmark17)

[Logging and Diagnostics](#_bookmark18) [33](#_bookmark18)

[Chapter 4. GitHub - Synopsys Action](#_bookmark19) [35](#_bookmark19)

[GitHub Prerequisites](#_bookmark20) [35](#_bookmark20)

[Using Synopsys GitHub Action for Polaris](#_bookmark21) [36](#_bookmark21)

[Using Synopsys GitHub Action for Black Duck](#_bookmark22) [37](#_bookmark22)

[Using Synopsys GitHub Action for Coverity Cloud Deployment with Thin Client](#_bookmark23) [39](#_bookmark23)

[Additional GitHub Configuration](#_bookmark24) [41](#_bookmark24)

[Chapter 5. GitLab – Synopsys Template](#_bookmark25) [42](#_bookmark25)

[GitLab Prerequisites](#_bookmark26) [42](#_bookmark26)

[Using the Synopsys GitLab Template with Polaris](#_bookmark27) [43](#_bookmark27)

[Using the Synopsys GitLab Template with Black Duck](#_bookmark28) [44](#_bookmark28)

[Using the Synopsys GitLab Template for Coverity Cloud Deployment with Thin Client](#_bookmark29) [47](#_bookmark29)

Contents | iii

[Additional GitLab Configuration](#_bookmark30) [48](#_bookmark30)

[Chapter 6. Azure DevOps - Synopsys Security Scan](#_bookmark31) [50](#_bookmark31)

[Azure Prerequisites](#_bookmark32) [50](#_bookmark32)

[Using Azure DevOps Extension with Polaris](#_bookmark33) [51](#_bookmark33)

[Using Azure DevOps Extension with Black Duck](#_bookmark34) [52](#_bookmark34)

[Using Azure DevOps Extension with Coverity Connect with Thin Client](#_bookmark35) [54](#_bookmark35)

[Additional Azure Configuration](#_bookmark36) [55](#_bookmark36)

[Chapter 7. Glossary](#_bookmark37) [57](#_bookmark37)

# Chapter 1. Overview

## Overview

Use any of these to integrate Static Analysis Security Testing (SAST) and Software Composition Analysis (SCA) into your CI/CD pipelines:

* Synopsys Bridge CLI
* Synopsys Action for GitHub
* Synopsys Template for GitLab
* Synopsys Security Scan for Azure DevOps With any of the integrations above you can:
* Scan when you merge code or on a pull request.
* Optionally create pull request comments when new issues are found.
* Optionally create a new pull request to automatically update vulnerable components.

#### Further information

* [Support Matrix *(on*](#_bookmark2)[*page 4)*](#_bookmark2)
* [Files and Directories *(on*](#_bookmark3)[*page 5)*](#_bookmark3)
* [Download Synopsys Bridge *(on*](#_bookmark4)[*page 6)*](#_bookmark4)

## Support Matrix

The table below outlines the Synopsys security tools supported by Synopsys Bridge.

|  |  |  |
| --- | --- | --- |
| **Tool** | **Bridge Support?** | **Notes** |
| Polaris | Yes | Polaris users can use Synopsys Bridge CLI to automate SAST and/or SCA scans in their CI pipeline. Click here for SAST specific [system requirements](https://sig-product-docs.synopsys.com/bundle/coverity-docs/page/deploy-install-guide/topics/minimum_requirements.html). |
| Black Duck | Yes | Black Duck users can use Synopsys Bridge CLI to automate SCA scans in their CI pipeline. |

Synopsys Bridge CLI Guide | 1 - Overview | 5

|  |  |  |
| --- | --- | --- |
| **Tool** | **Bridge Support?** | **Notes** |
| Coverity Connect | Yes | Coverity users can use Synopsys Bridge CLI to automate SAST scans in their CI pipeline. Synopsys Bridge can be used with both an on-prem Coverity Connect as well as Coverity cloud deployment. Click here for [system requirements](https://sig-product-docs.synopsys.com/bundle/coverity-docs/page/deploy-install-guide/topics/minimum_requirements.html). |
| ~~Coverity on Polaris~~ | ~~No~~ | ~~Synopsys Bridge cannot run scans on Coverity on Polaris (al­ so known as "Classic Polaris")~~ |

Synopsys Bridge runs on the following operating systems:

##### OS System Requirements Notes

Linux 64-bit kernel, version 2.6.32+ with glibc 2.18 or later

Debian GNU is *not* supported

macOS OSX 11, 12, 13 macOS 11, 12 and 13 on Intel (M1 and M2 based Macs are not currently supported)

Windows x86\_64, Version 10 and 11 and Windows Server 2019 and 2022

Server Core not supported

## Files and Directories

By default, the Synopsys Bridge writes logs and temporary files to   <current\_working\_directory>/.bridge. You may change this default directory by using the --home <directory\_path> option.

The following files and directories are found under the Synopsys Bridge home directory:

* bridge.log
* diagnostics.json file with --diagnostics option. See [Logging and Diagnostics](#_bookmark18) *[(on](#_bookmark18) [page 33)](#_bookmark18)* for details.
* Adapter directories and the corresponding stdout and stderr log files.
* Additional temporary files

Synopsys Bridge CLI Guide | 1 - Overview | 6

## Download Synopsys Bridge

You can download the latest version of Synopsys Bridge from [Synopsys Artifactory](https://sig-repo.synopsys.com/artifactory/bds-integrations-release/com/synopsys/integration/synopsys-bridge/)

Polaris users can also download Synopsys Bridge directly from the Polaris user interface:

1. Click **username** at the top right.
2. Select **Accounts**.
3. Select **Downloads**.
4. Choose the appropriate package for your operating system.

To install, simply unzip and add synopsys-bridge executable to your PATH or use absolute PATH to synopsys-bridge executable.

# Chapter 2. Synopsys Bridge CLI

## Using Synopsys Bridge CLI

Once you have synopsys-bridge executable installed, you are ready to use Synopsys Bridge to integrate SAST and/or SCA scans into your CI/CD pipeline.

You can run Synopsys Bridge in one of the following two ways:

1. By passing arguments thru a JSON file
2. By passing arguments on the command line

For a complete list of exit codes returned by Synopsys Bridge, see the [Exit Code](https://synopsys-theme-dev.zoominsoftware.io/bundle/bridge/page/documentation/c_exit-codes.html) table.

### Passing Arguments using a JSON file

Passing arguments using a JSON file greatly simplifies the command line and promotes reuse. Here are the steps:

1. Create an access token in the web interface of the Synopsys security product you are integrating with.
2. Use environment variable(s) to pass sensitive information such as password or access token to Synopsys Bridge (recommended for security purposes). Synopsys Bridge automatically picks up values passed thru these variables.
   * Example:  export BRIDGE\_POLARIS\_ACCESSTOKEN=<*POLARIS\_ACCESSTOKEN>*.
3. Pass the JSON file to Synopsys Bridge using the --input command line option.
4. Pass the Synopsys security product you are integrating with using the --stage option.

Here are the example commands:

export BRIDGE\_POLARIS\_ACCESSTOKEN=<*POLARIS\_ACCESSTOKEN*>

synopsys-bridge --stage polaris --input input.json



**Note:**

Depending on your OS, you will need to use appropriate mechanism to set environment variables.

Here is the input.json file:

{

"data": {

"polaris": {

"application": {

"name": "<APPLICATION\_NAME>"

},

"project": {

"name": "<PROJECT\_NAME>"

},

"assessment": {

"types": ["SAST", "SCA"]

},

"serverUrl": "<SERVER\_URL>"

}

}

}



**Note:**

It is recommended that you save the JSON file at the root of the project directory being scanned. The JSON file can have any name as long as it has a .json extension.



**Tip:**

You can use different .json files for different use cases.

For a complete list of environment variables and command line arguments, see [Complete List of Synopsys Bridge Arguments](#_bookmark16) *[(on](#_bookmark16) [page 18)](#_bookmark16)*.

For tool specific information and examples, see:

* [Using Synopsys Bridge with Polaris *(on*](#_bookmark9)[*page 10)*](#_bookmark9)
* [Using Synopsys Bridge with Black Duck *(on*](#_bookmark13)[*page 12)*](#_bookmark13)
* [Using Synopsys Bridge with Coverity Connect *(on*](#_bookmark14)[*page 14)*](#_bookmark14)

### 

### Passing Arguments using the CLI

You can also pass arguments on the command line as an alternative to passing arguments using a JSON file.

Here are the steps:

1. Create an access token in the web interface of the Synopsys security product you are integrating with.
2. Use environment variable(s) to pass sensitive information such as password or access token to Synopsys Bridge (recommended for security purposes). Synopsys Bridge automatically picks up values passed thru these variables.
   * Example:  export BRIDGE\_POLARIS\_ACCESSTOKEN=<*POLARIS\_ACCESSTOKEN>*.
3. Pass the necessary command line arguments as shown in the example below.

export BRIDGE\_POLARIS\_ACCESSTOKEN="<POLARIS\_ACCESSTOKEN>"

synopsys-bridge --stage polaris polaris.project.name="<PROJECT\_NAME>" \

polaris.application.name="<APPLICATION\_NAME>" \

polaris.assessment.types=SAST,SCA \

polaris.serverUrl="<POLARIS\_SERVERURL>"

For a complete list of environment variables and command line arguments, see [Complete List of Synopsys Bridge Arguments](#_bookmark16) *[(on](#_bookmark16) [page 18)](#_bookmark16)*.

See Schema Resources And Extensions *(on page )* for Synopsys Bridge resources.

For tool specific information and examples, see:

* [Using Synopsys Bridge with Polaris *(on*](#_bookmark9)[*page 10)*](#_bookmark9)
* [Using Synopsys Bridge with Black Duck *(on*](#_bookmark13)[*page 12)*](#_bookmark13)
* [Using Synopsys Bridge with Coverity Connect *(on*](#_bookmark14)[*page 14)*](#_bookmark14)

Synopsys Bridge CLI Guide | 2 - Synopsys Bridge CLI | 10

## Using Synopsys Bridge CLI with Polaris

As a Polaris customer, you can use Synopsys Bridge to automate SAST and SCA scanning in your CI/CD pipeline.

You can use Synopsys Bridge to run Polaris scans in the following two ways:

* [Running Polaris scans with a JSON file *(on*](#_bookmark10)[*page 10)*](#_bookmark10)
* [Running Polaris scans on the command line *(on*](#_bookmark11)[*page 11)*](#_bookmark11)

In addition to running scans, you can also optionally configure Synopsys Bridge to create fix pull requests for SCA issues. Currently, only NPM is supported. For more information, see [Complete List of Synopsys Bridge Arguments](#_bookmark16) *[(on](#_bookmark16) [page 18)](#_bookmark16)*.



**Note:**

As an alternative to Synopsys Bridge, you can also use Synopsys Action for GitHub or Synopsys Template for GitLab or Synopsys Security Scan for Azure DevOps.

#### Running Polaris scans with a JSON file

Synopsys Bridge for Polaris uses Coverity for SAST scans and Black Duck for SCA scans under the hood. Depending on the task, you may need to pass additional SAST and SCA configurations.

After passing sensitive access token and password information using the BRIDGE\_POLARIS\_ACCESSTOKEN environmental variable, run Synopsys Bridge and pass the JSON file using the --input command line option.

Here is a command line example for Polaris:

export BRIDGE\_POLARIS\_ACCESSTOKEN=<POLARIS\_ACCESSTOKEN>

synopsys-bridge --stage connect --input input.json

The above example uses the following:

* BRIDGE\_POLARIS\_ACCESSTOKEN environment variable to pass sensitive information such as password or access token to Synopsys Bridge (recommended for security purposes). Note that Synopsys Bridge automatically picks up values passed thru these environment variables.

--stage argument to specify the Synopsys security product in use

Here is the input.json file:

{

"data": {

"polaris": {

"application": {

"name": "*<APPLICATION\_NAME>*"

},

"project": {

"name": "*<PROJECT\_NAME>*"

},

"assessment": {

"types": ["SCA", "SAST"]

},

Synopsys Bridge CLI Guide | 2 - Synopsys Bridge CLI | 11

"serverUrl": "<*POLARIS\_URL>*"

}

}

}

The above example uses the following schema resources:

* polaris.serverUrl for Polaris URL
* polaris.application.name for Polaris Application to use. Note that the specified application must exist on Polaris with appropriate entitlements.
* polaris.project.name for Polaris Project to use. Note that the specified project must exist on Polaris. You can set polaris.onboarding to true if you want Synopsys Bridge to automatically create the project in case it doesn’t yet exist on Polaris.
* polaris.assessment.types specifies the type of scan to be run: SAST or SCA or SAST,SCA

For the required minimum set of arguments that you need to pass to integrate Synopsys Bridge with Polaris, refer to Polaris specific resources page under [Schema Resources And Extensions](https://synopsys-theme-dev.zoominsoftware.io/bundle/bridge/page/documentation/c_schema.html)

For a complete list of environment variables and command line arguments, see [Complete List of Synopsys Bridge Arguments](#_bookmark16) *[(on](#_bookmark16) [page 18)](#_bookmark16)*.

#### 

#### Running Polaris scans on the command line

Instead of using a JSON file, you can pass all arguments via the command line.

Here is a command line example for Polaris:

export BRIDGE\_POLARIS\_ACCESSTOKEN=*<POLARIS\_ACCESSTOKEN>*

synopsys-bridge --stage polaris polaris.project.name="*<PROJECT\_NAME>*" \ polaris.application.name="*<APPLICATION\_NAME>*" \ polaris.assessment.types=SAST,SCA \

polaris.serverUrl="*<SERVERURL>"*

The above example uses the following:

* BRIDGE\_POLARIS\_ACCESSTOKEN environment variable to pass sensitive information such as password or access token to Synopsys Bridge (recommended for security purposes). Note that Synopsys Bridge automatically picks up values passed thru these environment variables.
* --stage argument to specify the Synopsys security product in use
* polaris.serverUrl for Polaris URL
* polaris.application.name for Polaris Application to use. Note that the specified application  must exist on Polaris with appropriate entitlements.
* polaris.project.name for Polaris Project to use. Note that the specified project must exist on Polaris. You can set polaris.onboarding to true if you want Synopsys Bridge to automatically create the project in case it doesn’t yet exist on Polaris.
* polaris.assessment.types specifies the type of scan to be run: SAST or SCA or SAST,SCA

Synopsys Bridge CLI Guide | 2 - Synopsys Bridge CLI | 12

For the required minimum set of arguments that you need to pass to integrate Synopsys Bridge with Polaris, refer to Polaris specific resources page under [Schema Resources And Extensions](https://synopsys-theme-dev.zoominsoftware.io/bundle/bridge/page/documentation/c_schema.html)

For a complete list of environment variables and command line arguments, see [Complete List of Synopsys Bridge Arguments](#_bookmark16) *[(on](#_bookmark16) [page 18)](#_bookmark16)*.

For additional SAST-specific details, see [Additional SAST configuration requirements](#_bookmark12) *[(on](#_bookmark12) [page 12)](#_bookmark12)*.

#### Additional SAST configuration requirements

A coverity.yml configuration file is required for

* Static analysis of compiled languages like C/C++, C# and Java.
* Optimizing static analysis when results are unsatisfactory.

Certain Coverity Connect scans on Polaris require configuration of additional capture settings using a coverity.yaml file. See [Configuring Coverity Thin Client for use with Synopsys Bridge and Polaris](https://polaris.synopsys.com/developer/default/documentation/t_cov-thin-client) in the *Polaris Developer Portal* for more information.

## 

## Using Synopsys Bridge CLI with Black Duck

As a Black Duck customer, you can use Synopsys Bridge to automate SCA scanning in your CI/CD pipeline.

You can use Synopsys Bridge with Black Duck in the following two ways to run scans:

* Running Black Duck scans with a JSON file
* Running Black Duck scans on the command line

In addition to running scans, you can also optionally configure Synopsys Bridge to perform the following. For more information, see [Complete List of Synopsys Bridge Arguments](#_bookmark16) *[(on](#_bookmark16) [page 18)](#_bookmark16)*.

1. Scan pull requests
2. Add comments to pull requests
3. Create fix pull requests (NPM only)



**Note:**

As an alternative to Synopsys Bridge, you can also use Synopsys Action for GitHub or Synopsys Template for GitLab or Synopsys Security Scan for Azure DevOps.

#### Pass Sensitive Data as Environmental Variables

Before passing arguments with Synopsys Bridge, it is recommended that you pass user name and password arguments using environmental variables for security reasons:

export BRIDGE\_BLACKDUCK\_TOKEN=*<BLACKDUCK\_TOKEN>*

#### Running Black Duck scans with a JSON file

Here is an example command:

synopsys-bridge --stage blackduck --input input.json

The above example uses the following:

* BRIDGE\_BLACKDUCK\_TOKEN environment variable to pass sensitive information such as password or access token to Synopsys Bridge (recommended for security purposes). Note that Synopsys Bridge automatically picks up values passed thru these environment variables.
* --stage argument to specify the Synopsys security product in use

Synopsys Bridge CLI Guide | 2 - Synopsys Bridge CLI | 13

Here is the input.json file:

{

"data": {

"blackduck": {

"url": *<BLACKDUCK\_URL>*,

"scan": {

"full": true,

"failure": {

"severities": ["CRITICAL"]

}

}

}

}

}

#### 

The above example uses the following schema resources:

* blackduck.url for Black Duck URL
* blackduck.scan.full should be set to true so that Intelligent scans are run by Synopsys Bridge
* blackduck.scan.failure.severities is a list of severities that is used by Synopsys Bridge to decide if the CI pipeline should be failed or not

For the required minimum set of arguments that you need to pass to integrate Synopsys Bridge with Black Duck, refer to Black Duck specific resources page under [Schema Resources And Extensions](https://synopsys-theme-dev.zoominsoftware.io/bundle/bridge/page/documentation/c_schema.html)

For a complete list of environment variables and command line arguments, see [Complete List of Synopsys Bridge Arguments](#_bookmark16) *[(on](#_bookmark16) [page 18)](#_bookmark16)*.

#### Running Black Duck scans on the command line

Instead of using a JSON file, you can pass all arguments via the command line.

Here is a command line example for Black Duck:

export BRIDGE\_BLACKDUCK\_TOKEN=*<BLACKDUCK\_TOKEN>*

synopsys-bridge --stage blackduck \ blackduck.url=*<BLACKDUCK\_URL>* \ blackduck.scan.failure.severities=CRITICAL,HIGH \

blackduck.scan.full=true

:

The above example uses the following:

* BRIDGE\_BLACKDUCK\_TOKEN  environment variable to pass sensitive information such as password or access token to Synopsys Bridge (recommended for security purposes). Note that Synopsys Bridge automatically picks up values passed thru these environment variables.
* --stage argument to specify the Synopsys security product in use
* blackduck.url for Black Duck URL
* blackduck.scan.full should be set to true so that Intelligent scans are run by Synopsys Bridge
* blackduck.scan.failure.severities is a comma separated list of severities that is used by Synopsys Bridge to decide if the CI pipeline should be failed or not

For the required minimum set of arguments that you need to pass to integrate Synopsys Bridge with Black Duck, refer to Black Duck specific resources page under [Schema Resources And Extensions](https://synopsys-theme-dev.zoominsoftware.io/bundle/bridge/page/documentation/c_schema.html)

For a complete list of environment variables and command line arguments, see [Complete List of Synopsys Bridge Arguments](#_bookmark16) *[(on](#_bookmark16) [page 18)](#_bookmark16)*.



## Using Synopsys Bridge CLI with Coverity

As a Coverity customer, you can use Synopsys Bridge to automate SAST scanning in your CI/CD pipeline.



**Note:**

You can use Synopsys Bridge with both on-prem Coverity Connect as well as Coverity cloud deployment. Details below.

You can integrate Synopsys Bridge with Coverity in the following ways:

1. Running scans using a JSON file
2. Running scans using the command line

In addition to running scans, you can also optionally configure Synopsys Bridge to add comments to pull requests. For more information, see [Complete List of Synopsys Bridge Arguments](#_bookmark16) *[(on](#_bookmark16) [page 18)](#_bookmark16)*.



**Note:**

As an alternative to Synopsys Bridge, you can also use Synopsys Action for GitHub or Synopsys Template for GitLab or Synopsys Security Scan for Azure DevOps.

#### 

#### Running Coverity scans using a JSON file

Before running Synopsys Bridge, it is recommended that you pass user name and password arguments using environmental variables for security reasons:

Here are the example commands:

export BRIDGE\_COVERITY\_CONNECT\_USER\_NAME=*<COV\_USER>*

export BRIDGE\_COVERITY\_CONNECT\_USER\_PASSWORD=*<COVERITY\_PASSPHRASE>*

synopsys-bridge --stage connect --input input.json

Here is an example input.json file that you can use with on-prem Coverity Connect:

Synopsys Bridge CLI Guide | 2 - Synopsys Bridge CLI | 15

{

"data":

{

"coverity":

{

"connect": {

"url": "*<Connect URL>*", "project":{

"name": "*<PROJECT\_NAME>*"

},

"stream": {

"name": "*<STREAM\_NAME>*"

},

"policy": {

"view": "*<View Name / Id>*"

}

},

"local": true

}

}

}

Here is an example input.json file that you can use with Coverity cloud deployment:

Synopsys Bridge CLI Guide | 2 - Synopsys Bridge CLI | 15

{

"data":

{

"coverity":

{

"connect": {

"url": "*<Connect URL>*", "project":{

"name": "*<PROJECT\_NAME>*"

},

"stream": {

"name": "*<STREAM\_NAME>*"

},

"policy": {

"view": "*<View Name / Id>*"

}

}

}

}

}

The above examples use the following:

* BRIDGE\_COVERITY\_CONNECT\_USER\_NAME and BRIDGE\_COVERITY\_CONNECT\_USER\_PASSWORD environment variables to pass sensitive information such as password or access token to Synopsys Bridge (recommended for security purposes). Note that Synopsys Bridge automatically picks up values passed thru these environment variables.
* --stage argument to specify the Synopsys security product in use
* coverity.connect.url for Coverity Connect URL
* coverity.connect.project.name for project on Coverity Connect to be used
* coverity.connect.stream.name for stream on Coverity Connect to be used
* coverity.connect.policy.view for Coverity policy view used by Synopsys bridge to decide if the CI pipeline should be failed or not
* Coverity.local to let Synopsys bridge know if this is an on-prem Coverity Connect or a Coverity cloud deployment

#### Running Coverity Connect scans using the command line

Instead of using a JSON file, you can pass arguments on the command line.

Here are the example commands that can be used with on-prem Coverity Connect:

export BRIDGE\_COVERITY\_CONNECT\_USER\_NAME=<COV\_USER>

export BRIDGE\_COVERITY\_CONNECT\_USER\_PASSWORD=<COVERITY\_PASSPHRASE>

synopsys-bridge --stage bridge \ coverity.connect url=<*COVERITY\_URL>* \

coverity.connect.project.name=*<COVERITY\_PROJECT>* \ coverity.connect.stream.name=*<COVERITY\_STREAM>* \ coverity.connect.policy.view=*<COVERITY\_VIEW\_NAME>* \

coveriy.local=true

Synopsys Bridge CLI Guide | 2 - Synopsys Bridge CLI | 16

Here are the example commands that can be used with Coverity cloud deployment:

export BRIDGE\_COVERITY\_CONNECT\_USER\_NAME=<COV\_USER>

export BRIDGE\_COVERITY\_CONNECT\_USER\_PASSWORD=<COVERITY\_PASSPHRASE>

synopsys-bridge --stage bridge \ coverity.connect url=<*COVERITY\_URL>* \

coverity.connect.project.name=*<COVERITY\_PROJECT>* \ coverity.connect.stream.name=*<COVERITY\_STREAM>* \ coverity.connect.policy.view=*<COVERITY\_VIEW\_NAME>*

The above examples use the following:

* BRIDGE\_COVERITY\_CONNECT\_USER\_NAME and BRIDGE\_COVERITY\_CONNECT\_USER\_PASSWORD environment variables to pass sensitive information such as password or access token to Synopsys Bridge (recommended for security purposes). Note that Synopsys Bridge automatically picks up values passed thru these environment variables.
* --stage argument to specify the Synopsys security product in use
* coverity.connect.url for Coverity Connect URL
* coverity.connect.project.name for project on Coverity Connect to be used
* coverity.connect.stream.name for stream on Coverity Connect to be used
* coverity.connect.policy.view for policy view to be used to decide if the CI pipeline should be failed or not
* Coverity.local to let Synopsys bridge know if this is an on-prem Coverity Connect or a Coverity cloud deployment

For the required minimum set of arguments that you need to pass to integrate Synopsys Bridge with Coverity Cloud Deployment, refer to Coverity Connect specific resources page under [Schema Resources And Extensions](https://synopsys-theme-dev.zoominsoftware.io/bundle/bridge/page/documentation/c_schema.html)

For a complete list of environment variables and command line arguments, see [Complete List of Synopsys Bridge Arguments](#_bookmark16) *[(on](#_bookmark16) [page 18)](#_bookmark16)*.

For more details, see the [Complete List of Synopsys Bridge Arguments *(on*](#_bookmark16)[*page 18)*](#_bookmark16).

# 

# Chapter 3. Synopsys Bridge CLI Reference

## Complete List of Synopsys Bridge Arguments

This page lists all the arguments that Synopsys Bridge supports. Arguments can be passed thru environment variables, command line or a JSON file.



**Tip:**

We recommend that you pass sensitive information such as access tokens using environment variables.

For a list of arguments that are common to all Synopsys security products, refer to Universal Bridge Arguments below.

For product specific arguments, refer to the product specific section below:

* Polaris
* Black Duck
* Coverity

<Robert: Move “Universal Bridge Arguments” table from the bottom of the page to here above the “Polaris” section”>

#### Polaris

##### Arguments to Pass

**Argument Input Mode Required Notes**

**Command Line Argu­ ment**

**Environment Variable JSON field**

Access token polaris.ac•

cesstoken

Server URL po•

BRIDGE\_POLARIS\_ACCESS• TOKEN

BRIDGE\_POLARIS\_•

polaris.ac• cesstoken

po•

Yes For security reasons, pass this as an en­ vironmental variable.

Yes For security

laris.serverurlSERVERURL

laris.serverurl

reasons, pass this as an en­ vironmental variable.

Application Name

polaris.ap• plication•

.name

BRIDGE\_POLARIS\_APPLI• CATION\_NAME

polaris.ap• plication•

.name

Yes Application must exist on Polaris, and have right entitlements.

Synopsys Bridge CLI Guide | 3 - Synopsys Bridge CLI Reference | 19

##### Argument Input Mode Required Notes

Project Name po•

laris.project•

.name

BRIDGE\_POLARIS\_•

PROJECT\_NAME

po• laris.project•

.name

Yes. if polaris.on• boarding is set to true, Bridge will create the project as necessary.

Assessment Type

Tool Install Di­ rectory

Auto Create Projects

polaris.as• sessment•

.types

tool.install•

.directory

polaris.on• boarding

BRIDGE\_POLARIS\_ASSESS• MENT\_TYPES

BRIDGE\_TOOL\_INSTALL\_• DIRECTORY

BRIDGE\_POLARIS\_ON• BOARDING

polaris.as• sessment•

.types

tool.install•

.directory

polaris.on• boarding

Yes Comma sep­ arated values. Accept ed values SAST or SCA or SAST,SCA

No Directory

to which Bridge

downloads the underlying scan tools. Defaults to

*<User>*/.bridge

No If set to true, Bridge will attempt to create project on Polaris if it does not exist. Default is false.



**Note:** The appli­ cation with proper entmust al­

Synopsys Bridge CLI Guide | 3 - Synopsys Bridge CLI Reference | 20

##### Argument Input Mode Required Notes



ready exist with prop­ er en­ title­ ments. Bridge will error out

if the appli­ cation does not exist.

Polaris Triage po•

laris.triage

BRIDGE\_POLARIS\_TRIAGE po•

laris.triage

No If you are entitled to Auto-Triage feature on Polaris, you can use this option to enable the feature.

Possible val­ ues are RE•

NOT\_• REQUIRED and NOT\_ENTITLED.

QUIRED,

##### JSON Input

Here is a sample input.json file that can be used with Polaris:

{

"data": {

"polaris": {

"application": {

Synopsys Bridge CLI Guide | 3 - Synopsys Bridge CLI Reference | 21

"name": "*<Application Name>*"

},

"project": {

"name": "*<Project Name>*"

},

"assessment": {

"types": ["SCA", "SAST"]

},

"serverUrl": "<Polaris URL>"

}

}

}

Here are the commands to run:

export BRIDGE\_POLARIS\_ACCESSTOKEN=<POLARIS\_ACCESSTOKEN>

synopsys-bridge --stage Polaris --input input.json

:

#### Black Duck

##### Arguments to Pass

**Argument Input Mode**

**Re­ quired**

**Notes**

Black Duck URL

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Command Line Argu­**  **ment** | **Environment Variable** | **JSON field** |  |
| URL | blackduck.url | BRIDGE\_BLACKDUCK\_URL | blackduck.url | Yes |
| Token | blackduck.to• ken | BRIDGE\_BLACKDUCK\_TOKEN | blackduck.to• ken | Yes |
| Full scan | blackduck.s• can.full | BRIDGE\_BLACKDUCK\_SCAN\_• FULL | blackduck.s• can.full | No |

Black Duck Access token

Performs a full/intelligent scan when set to true. Required and used for scanning based on SCM push events.

Performs a rapid scan when set to false . Required for SCM pull request events.

true or false. (Default: false).

Synopsys Bridge CLI Guide | 3 - Synopsys Bridge CLI Reference | 22

##### Argument Input Mode

**Re­ quired**

**Notes**

Install Di­ rectory

blackduck.in• stall.direc• tory

BRIDGE\_BLACKDUCK\_INSTAL• L\_DIRECTORY

blackduck.in• No

stall.direc• tory

Path to directory where detect.jar resides.

Default: <

$HOME>/.bridge/ blackduck

Failure severities

blackduck.s• can.failure•

.severities

BRIDGE\_BLACKDUCK\_SCAN\_• FAILURE\_SEVERITIES

blackduck.s• No

can.failure•

.severities

Used by Bridge to determine whether to break the build or not.

If provided, Bridge will break the build and returns exit code 8.

Create fix pull requests

blackduck.au• tomation.fix• pr

BRIDGE\_BLACKDUCK\_AU• TOMATION\_FIXPR

blackduck.au• No

tomation.fix• pr

If set to true, Bridge creates fix pull requests for vulnerable direct dependencies. Default: false)

Note: Currently only NPM is supports.

##### Note: Requires SCM information including token as documented in section SCM Information needed for “Creating Fix Pull Requests” feature

##### below.

Synopsys Bridge CLI Guide | 3 - Synopsys Bridge CLI Reference | 23

##### Argument Input Mode

**Re­ quired**

**Notes**

Add comments to pull requests

blackduck.au• tomation•

.prcomments

BRIDGE\_BLACKDUCK\_AU• TOMATION\_PRCOMMENTS

blackduck.au• No

tomation•

.prcomments

If set to true , Bridge adds comments to pull requests for new issues introduced in the pull request.

Requires Rapid Scan to be run on pull require events. This flag is ignored if full scan is run.

##### Note: Requires SCM information including token as documented in section SCM Information needed for “Adding Comments to Pull Requests” feature

##### below

Synopsys Bridge CLI Guide | 3 - Synopsys Bridge CLI Reference | 24

##### Argument Input Mode

**Re­ quired**

**Notes**

##### SCM Information needed for “Creating Fix Pull Requests” feature

To use this feature, you must pass the following SCM arguments.

##### SCM

**Argu­ ment**

**Input Mode**

**Re­ quired**

**Notes**

GitHub

User Token

Repos­ itory Name

##### Command Line Argu­ ment

github.user•

.token

##### Environment Variable

BRIDGE\_• GITHUB\_• USER\_TOKEN

##### JSON Field

github.user•

.token

github.repos• itory.name

Yes GitHub token with access to the project being scanned, and per­ mission to create PR

Yes Name of the GitHub Repository

Branch Name

github.repos­ itory.branch­

|  |  |  |  |
| --- | --- | --- | --- |
| github.repos• itory.name | BRIDGE\_• | |  |
| GITHUB\_RE• | | |
| POSITORY\_• | | |
| NAME |  | |
|  | | |
| github.repos• | BRIDGE\_• | |  |
| itory.branch• | GITHUB\_RE• | | |
| .name  github.repos• | POSITORY\_• | | |
| BRANCH\_NAME | | |
| BRIDGE\_• | |  |

.name

Yes Name of Branch (typically main /

master)

Repository Owner

GitLab GitLab

API URL

itory.owner•

.name

git• lab.api.url

GITHUB\_RE• POSITORY\_• OWNER\_NAME

BRIDGE\_GIT• LAB\_URL

github.repos• itory.owner•

.name

git• lab.api.url

Yes Repository Owner

Yes API URL for GitLab

Synopsys Bridge CLI Guide | 3 - Synopsys Bridge CLI Reference | 25

##### SCM

**Argu­ ment**

**Input Mode**

**Re­ quired**

**Notes**

User Token

Repos­ itory Name

Branch Name

gitlab.user•

.token

gitlab.repos• itory.name

gitlab.repos•

itory.branch•

.name

BRIDGE\_GIT• LAB\_USER\_TO• KEN

BRIDGE\_GIT•

LAB\_REPOSI•

TORY\_NAME

BRIDGE\_•

GITLAB\_RE•

POSITORY\_•

BRANCH\_NAME

gitlab.user•

.token

gitlab.repos• itory.name

gitlab.repos• itory.branch•

.name

Yes User token with access to the project being scanned, and per­ mission to create PR comments.

Yes Name of repo

Yes Name of Branch (typically main /

master)

Azure

Azure API URL

azure.api.url

BRIDGE\_•

AZURE\_API\_•

URL

azure.api.url Yes Azure API URL

User Token

Orga­ niza­ tion Name

azure.user•

.token

azure.organi• zation.name

BRIDGE\_•

AZURE\_USER\_•

TOKEN

BRIDGE\_•

AZURE\_OR• GANIZATION\_• NAME

azure.user•

.token

azure.organi• zation.name

Yes User token with access to the project being scanned and per­ mission to create PRs.

Yes Azure organization name

Project

azure.project• BRIDGE\_•

azure.project• Yes Azure project name

Name

Repos­ itory Name

Branch Name

.name

azure.reposi• tory.name

|  |  |
| --- | --- |
| BRIDGE\_• | |
| AZURE\_RE• | |
| POSITORY\_• | |
| NAME |  |
|  | |
| BRIDGE\_• | |
| AZURE\_RE• | |

azure.repos• itory.branch•

.name

AZURE\_• PROJECT\_NAME

POSITORY\_• BRANCH\_NAME

.name

azure.reposi• tory.name

azure.repos• itory.branch•

.name

Yes Azure repo name

Yes Azure repo branch name (typical­ ly main / master)

Synopsys Bridge CLI Guide | 3 - Synopsys Bridge CLI Reference | 26

##### SCM

**Argu­ ment**

**Input Mode**

**Re­ quired**

**Notes**

Pull Re­ quest Number

azure.repos• itory.pull•

.number

BRIDGE\_• AZURE\_RE• POSITORY\_• PULL\_NUMBER

azure.repos• itory.pull•

.number

Yes Pull request to add comments to

##### SCM Information needed for “Adding Comments to Pull Requests” feature

**SCM**

**Argu­ ment**

**Command Line Argu­ ment**

**Input Mode**

**Environment Variable**

**JSON Field**

**Re­ quired**

**Notes**

GitHub

User Token

Repos­ itory Name

github.user•

.token

BRIDGE\_• GITHUB\_• USER\_TOKEN

github.user•

.token

Yes User token with access to the project being scanned, and per­ mission to create PR

Yes Name of Repo

Branch Name

Repository Owner

github.repos• itory.branch•

.name

|  |  |  |  |
| --- | --- | --- | --- |
| github.repos• itory.name | BRIDGE\_• | |  |
| GITHUB\_RE• | | |
| POSITORY\_• | | |
| NAME |  | |
|  | | |
| github.repos• | BRIDGE\_• | |  |
| itory.branch• | GITHUB\_RE• | | |
| .name  github.repos• itory.owner•  .name  github.repos• | POSITORY\_• | | |
| BRANCH\_NAME | | |
| BRIDGE\_• | |  |
| GITHUB\_RE• | | |
| POSITORY\_• | | |
| OWNER\_NAME | | |
|  | | |
| BRIDGE\_• | |  |

github.repos• itory.owner•

.name

Yes Name of Branch (typically main / master)

Yes Repo Owner Name

Pull Re­ quest Num­ ber

itory.pull•

.number

GITHUB\_RE• POSITORY\_• PULL\_NUMBER

github.repos• itory.pull•

.number

Yes Pull request on which to add comments.

Synopsys Bridge CLI Guide | 3 - Synopsys Bridge CLI Reference | 27

##### SCM

**Argu­ ment**

**Input Mode**

**Re­ quired**

**Notes**

GitLab

GitLab API URL

User Token

Repos­ itory Name

Branch Name

Pull Re­ quest Num­ ber

git• lab.api.url

gitlab.user•

.token

gitlab.repos• itory.name

gitlab.repos•

itory.branch•

.name

gitlab.repos• itory.pull•

.number

BRIDGE\_GIT• LAB\_URL

BRIDGE\_GIT• LAB\_USER\_TO• KEN

BRIDGE\_GIT•

LAB\_REPOSI•

TORY\_NAME

BRIDGE\_•

GITLAB\_RE•

POSITORY\_•

BRANCH\_NAME

BRIDGE\_GIT•

LAB\_REPOSI•

TORY\_PULL\_•

NUMBER

git• lab.api.url

gitlab.user•

.token

gitlab.repos• itory.name

gitlab.repos• itory.branch•

.name

gitlab.repos• itory.pull•

.number

Yes API URL for GitLab

Yes User token with access to the project being scanned, and per­ mission to create PR comments.

Yes Name of repo

Yes Name of Branch (typically main /

master)

Yes Pull request to add comments to

Azure

Azure API URL

azure.api.url

BRIDGE\_•

AZURE\_API\_•

URL

azure.api.url Yes Azure API URL

User Token

Orga­ niza­ tion Name

azure.user•

.token

azure.organi• zation.name

BRIDGE\_•

AZURE\_USER\_•

TOKEN

BRIDGE\_•

AZURE\_OR• GANIZATION\_• NAME

azure.user•

.token

azure.organi• zation.name

Yes User token with access to the project being scanned and per­ mission to create PRs.

Yes Azure organization name

Project

azure.project• BRIDGE\_•

azure.project• Yes Azure project name

Name

.name

AZURE\_• PROJECT\_NAME

.name

Synopsys Bridge CLI Guide | 3 - Synopsys Bridge CLI Reference | 28

##### SCM

**Argu­ ment**

**Input Mode**

**Re­ quired**

**Notes**

Repos­ itory Name

azure.reposi• tory.name

|  |  |
| --- | --- |
| BRIDGE\_• | |
| AZURE\_RE• | |
| POSITORY\_• | |
| NAME |  |
|  | |
| BRIDGE\_• | |
| AZURE\_RE• | |

azure.reposi• tory.name

Yes Azure repo name

Branch Name

azure.repos• itory.branch•

.name

POSITORY\_• BRANCH\_NAME

azure.repos• itory.branch•

.name

Yes Azure repo branch name (typical­ ly main / master)

##### JSON Input

Here is a sample input.json file that can be used with Black Duck:

{

"data": {

"blackduck": {

"url": *<BlackDuck url>*, "scan": {

"full": true,

"failure": {

"severities": ["CRITICAL"]

}

}

}

}

}

I

Here are the commands to run:

export BRIDGE\_BLACKDUCK\_TOKEN=<BLACKDUCK\_TOKEN>

synopsys-bridge --stage blackduck --input input.json

#### Coverity Connect

Synopsys Bridge CLI Guide | 3 - Synopsys Bridge CLI Reference | 29

##### Argu­ ment

Coverity URL

##### Com­ mand Line Ar­ gument

coveri•

coveri•

**Input Mode**

**Environ­ ment Variable**

BRIDGE\_•

##### JSON

**field**

**Re­ quired**

Yes

##### Notes

ty.con•

COVERI•

ty.con•

User Name

Pass­ word

Project Name

nect.url

coveri•

ty.con•

nect•

.user•

.name

coveri•

ty.con•

nect•

.user•

.pass• word

coveri•

ty.con•

nect•

TY\_CON•

NECT\_URL BRIDGE\_•

COVERI•

TY\_CON•

NECT\_• USER\_NAME

|  |  |  |
| --- | --- | --- |
| .name | |  |
|  | | |
| coveri• | | |
| ty.con• | | |
| nect• | | |
| .user• | | |
| .pass• | | |
| word |  | |
|  | | |
| coveri• | | |
| ty.con• | | |
| nect• | |  |
| .project | | |
| .name | |  |

BRIDGE\_•

COVERI•

TY\_CON•

NECT\_• USER\_• PASSWORD

BRIDGE\_•

nect.url

coveri•

ty.con•

nect•

.user•

Yes For security reasons, it is recom pass this as an environ­ mental variable.

Yes For security reasons it is best to pass this as an environ­ mental variable.

Yes Project must exist on Coverity Instance

.project• •

|  |
| --- |
| COVERI• |
| TY\_CON• |
| NECT\_• |

Stream Name

View

coveri•

.name

coveri­ ty.con­ nect­

.stream­

.name

PROJECT\_• NAME

BRIDGE\_•

COVERI•

TY\_CON•

NECT\_• STREAM\_• NAME

BRIDGE\_•

coveri•

ty.con•

nect•

.stream•

.name

coveri•

Yes Stream must exist on Coverity Instance.

No

Coverity platform's view name/ID.

ty.con•

ty.con•

COVERI•

Synopsys Bridge CLI Guide | 3 - Synopsys Bridge CLI Reference | 30

##### Argu­ ment

**Input Mode**

**Re­ quired**

**Notes**

Bridge will break the build if issues are found in the view provided by user and returns [exit code 8](#_bookmark17) *[(on](#_bookmark17) [page 33)](#_bookmark17)*

Install direc­ tory

tion•

.prcom• ment

|  |  |  |  |
| --- | --- | --- | --- |
|  | nect• | TY\_CON• | nect• |
| .policy• | NECT\_• | .policy• |
| .view | POLICY\_• | .view |
|  | VIEW |  |
|  | coveri• | BRIDGE\_• | coveri• No |
| Add comments to pull requests | ty.con• | COVERI• | ty.con• |
|  | nect.au• | TY\_CON• | nect.au• |
|  | toma• | NECT\_AU• | toma• |

cover• ity.in• stall•

.direc• tory

TOMATION\_• tion•

.prcom•

PRCOMMENT

ment

BRIDGE\_• No

cover•

ity.in•

COVERI•

TY\_INS•

stall•

.direc•

TALL\_•

DIRECTORY

tory

If set to true , Bridge adds comments to pull requests for new issues introduced in the pull request.

Requires Rapid Scan to be run on pull require events. This flag is ignored if full scan is run.

##### Note: Requires SCM information including token as documented in section SCM Information needed for “Adding Comments to Pull Requests” feature

Path to directory where coverity resides. Default: <

$HOME>/.bridge/ coverity

Local analysis

coveri• ty.local

BRIDGE\_•

COVERI•

TY\_LOCAL

coveri• No

ty.local

To use Synopsys Bridge with on-prem Coverity Connect, set this to true. When set to true, Bridge will download full analysis kit and will perform capture and analysis locally.

With Coverity cloud deployments, Synopsys uses Thin Client and this option should be set to false

Default: false.



**Note:**

To use Synopsys Bridge with on-prem Coverity Connect, you must set the “Coverity.local” to true as described above.

Synopsys Bridge CLI Guide | 3 - Synopsys Bridge CLI Reference | 31

Here is a sample input.json file that can be used with Coverity Cloud:

{

"data":

{

"coverity":

{

"connect": {

"url": "*<Connect URL>*",

"project":{

"name": "*<PROJECT\_NAME>*"

},

"stream": {

"name": "*<STREAM\_NAME>*"

},

"policy": {

"view": "*<View Name / Id>*"

},

"automation": { "prcomment" : false

}

}

}

}

}

Here is a sample input.json file that can be used with on-prem Coverity Connect:

{

"data":

{

"coverity":

{

"connect": {

"url": "*<Connect URL>*",

"project":{

"name": "*<PROJECT\_NAME>*"

},

"stream": {

"name": "*<STREAM\_NAME>*"

},

"policy": {

"view": "*<View Name / Id>*"

},

"automation": { "prcomment" : false

}

},

“local” : true

}

}

}

Here are the commands to run:

export BRIDGE\_COVERITY\_CONNECT\_USER\_NAME=<COV\_USER>

export BRIDGE\_COVERITY\_CONNECT\_USER\_PASSWORD=<COVERITY\_PASSPHRASE>

synopsys-bridge --stage blackduck --input input.json

#### Universal Synopsys Bridge Arguments

These arguments can be passed on the command line, but not as part of a JSON file.

**Command**

**Description**

**Required?**

synopsys-bridge

Command to invoke Synopsys Bridge.

Yes

Synopsys Bridge CLI Guide | 3 - Synopsys Bridge CLI Reference | 32

Enables debug logs under the Synopsys No Bridge home directory. Creates a diagnos• tics.json file containing the final state

data inside the Synopsys Bridge home di­ rectory, but masking sensitive information like tokens and passwords. See [Logging](#_bookmark18)

[and Diagnostics *(on*](#_bookmark18)[*page 33)*](#_bookmark18).

--diagnostics

No

Turns on verbose logging.

-- verbose

No

Specifies a schema to load

-- schema

Specifies a specific version of Synopsys No

Bridge to run.

-- version

No

Sets a home directory.

-- home

No

Outputs JSON format logs in the bridge.log file in the Synopsys Bridge home directory. See [Logging and Diag­](#_bookmark18)

[nostics *(on*](#_bookmark18)[*page 33)*](#_bookmark18).

--json-log-file

Outputs JSON format logs. See [Logging](#_bookmark18) No

[and Diagnostics *(on*](#_bookmark18)[*page 33)*](#_bookmark18).

--json-log

Shows the help file for Synopsys Bridge. No

--help

The --input command loads a JSON file Required for inputting a JSON containing common arguments to run files.

scans

--input

The --stage command specifies a group Yes of adapters to run (such as --stage po•

laris).

--stage

**Required?**

**Description**

**Command**

Synopsys Bridge CLI Guide | 3 - Synopsys Bridge CLI Reference | 33

## Exit Codes

After running a Synopsys Bridge command, you will receive a response code (see below) while full response details appear in the console. If Synopsys Bridge runs into problems, it outputs colored ERROR and WARN lines in the console response.

Synopsys Bridge replies with different exit codes depending upon execution results. Any exit code other than 0 should be seen as a build-breaking condition in your CI/CD platform.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Code** | **Code Name** | **Description** |
| 0 |  | Normal | Synopsys Bridge exited without any errors. |
| 1 |  | UndefinedError | Undefined error. Review the log for details. |
| 2 |  | AdapterError | Synopsys Bridge received a non-0 exit code from an internal adapter. Review the log for details. |
| 3 |  | ShutdownFailed | Synopsys Bridge failed to shut itself down. Review the log for details. |
| 8 |  | BridgeBuildBreak | The config option bridge.break  is set to true but Synopsys Bridge is unable to enforce this. As a workaround, create a simple script to call Synopsys Bridge and implement build break logic in your script. |
| 9 |  | StartupFailed | Failed to initiate Synopsys Bridge. Review the log for details. |

## Logging and Diagnostics

Synopsys Bridge offers multiple logging and diagnostic options. By default, logs are written to <current\_working\_directory>/.bridge directory. User can change this default location by passing the --home <directory\_path> option.

#### Logging

Synopsys Bridge offers multiple logging options.

Synopsys Bridge CLI Guide | 3 - Synopsys Bridge CLI Reference | 34

* Pass --json-log to output JSON format logs.
* Pass --json-log-file to enable JSON format logs in the bridge.log file in the Synopsys Bridge home directory .

#### Diagnostics

To enable Synopsys Bridge diagnostics mode, pass a --diagnostics command line option. With this option set, Synopsys Bridge:

. writes additional diagnostics information to bridge.log

* passes diagnostics related options to underlying tools so that they create logs under the Synopsys Bridge home directory.
* writes execution state date to diagnostics.json file under the Synopsys Bridge home directory

# 

# Chapter 4. GitHub - Synopsys Action

The Synopsys GitHub Action can be used to integrate Synopsys security testing into your CI pipeline.

You can download Synopsys GitHub Action directly from the GitHub Marketplace: [https://github.com/marketplace/actions/synopsys-action](https://github.com/marketplace/actions/synopsys-action" \t "_blank).

By including and configuring the Synopsys Action in your workflow.yml file, you can quickly integrate Synopsys security products into your CI pipeline.

We recommend using GitHub secrets for sensitive data like access tokens.

For more information, see:

* [GitHub Prerequisites *(on*](#_bookmark20)[*page 35)*](#_bookmark20)
* [Using Synopsys GitHub Action for Polaris *(on*](#_bookmark21)[*page 36)*](#_bookmark21)
* [Using Synopsys GitHub Action for Black Duck *(on*](#_bookmark22)[*page 37)*](#_bookmark22)
* [Using Synopsys GitHub Action for Coverity Cloud Deployment with Thin Client *(on*](#_bookmark23)[*page 39)*](#_bookmark23)
* [Additional GitHub Configuration *(on*](#_bookmark30)[*page 48)*](#_bookmark30)

## GitHub Prerequisites

Before configuring Synopsys Action into your workflow, you must meet the following prerequisites:

#### GitHub Runner Setup

* Runners are the machines that execute jobs in a GitHub Actions workflow. To use GitHub runners in your project, GitHub Actions must be enabled for a repository/organization settings in order for required workflows to run (**Repository Settings** → **SelectActions** → **General** → **Actions permissions**).
* GitHub runner can be Self-hosted or GitHub-hosted. For installing Self-hosted runners, see [Self- hosted runners](https://docs.github.com/en/actions/hosting-your-own-runners). For installing GitHub-hosted runners, see [GitHub-hosted runners](https://docs.github.com/en/actions/using-github-hosted-runners/about-github-hosted-runners).

#### Configure GitHub Secrets

Sensitive data such as access tokens, user names, passwords and even URLs must be configured using GitHub secrets (**GitHub** → **Project** → **Settings** → **Secrets and Variables** → **Actions**).

Synopsys Bridge CLI Guide | 4 - GitHub - Synopsys Action | 36

#### Configure GitHub Token

github\_token is required as input when running Black Duck Fix PR, Black Duck/Coverity PR Comment. There are two different types of tokens that can be passed to github\_token:

* Token can be GitHub specified secrets.GITHUB\_TOKEN with required workflow read and write permissions (**GitHub** → **Project** → **Settings** → **Actions** → **General** → **Workflow Permissions**). It will be created by GitHub at start of each workflow run.
* If you need a token that requires permissions that aren't available in the secrets.GITHUB\_TOKEN, create a Personal Access Token (PAT) with required scopes (**Select Profile Photo** → **Settings** → **Developer Settings** → **Personal access tokens**). For more information, see [Granting Additional Permissions](https://docs.github.com/en/actions/security-guides/automatic-token-authentication#granting-additional-permissions). PAT must have repo and api scope to perform Black Duck Fix PR or Black Duck/ Coverity PR Comment.

#### Create workflow

Create a new workflow (**GitHub** → **Project** → **Actions** → **New Workflow** → **Setup a workflow yourself**) and configure he required fields. Push those changes and GitHub runner will initiate the workflow which can be seen on the **Actions** tab on main page of the repository.

## Using Synopsys GitHub Action for Polaris

Before running a pipeline using the Synopsys GitHub Action with Polaris, you must set the appropriate applications and entitlements in your Polaris environment.

Using Synopsys Action, you can perform scans on push events to main branches. Pull request scanning is currently not supported for Polaris.

Add the following code block to your existing *workflow*.yml file in your .github/workflows directory. (If you need to create a workflow, go to the repository you're integrating with Polaris on the GitHub UI, click the **Actions** tab at the top, then click **New Workflow**.)

Below is an example of a *workflow*.yml file configured for Polaris.

name: polaris-sig-action on:

push:

branches: [ main, master, develop, stage, release ] workflow\_dispatch:

jobs:

build:

Synopsys Bridge CLI Guide | 4 - GitHub - Synopsys Action | 37

runs-on: [ ubuntu-latest ] steps:

* name: Checkout Source uses: actions/checkout@v3
* name: Polaris Scan

uses: [synopsys-sig/synopsys-action@v1.2.0](mailto:synopsys-sig/synopsys-action@v1.2.0) with:

polaris\_serverUrl: *${{ secrets.POLARIS\_SERVERURL }}* polaris\_accessToken: *${{ secrets.POLARIS\_ACCESSTOKEN }}* polaris\_application\_name: *${{ github.event.repository.name }}* polaris\_project\_name: *${{ github.event.repository.name }}* ### Accepts Multiple Values

polaris\_assessment\_types: "SAST,SCA"

### Uncomment below configuration if Synopsys Bridge diagnostic files needs to be uploaded # include\_diagnostics: true

**List of mandatory and optional parameters for Polaris:**

| **Input Parameter** | **Description** | **Mandatory / Optional** |
| --- | --- | --- |
| polaris\_serverUrl | Polaris URL | Mandatory |
| polaris\_accessToken | Polaris Access token | Mandatory |
| polaris\_application\_name | Polaris Application name | Mandatory |
| polaris\_project\_name | Polaris Project name | Mandatory |
| polaris\_assessment\_types | Polaris assessment types.  Example: SCA or SAST or SAST,SCA | Mandatory |

## Using Synopsys GitHub Action for Black Duck

The Synopsys Action supports both self-hosted (e.g. on-prem) and Synopsys-hosted Black Duck Hub instances.

In the default Black Duck Hub permission model, projects and project versions are created on the fly and as needed. Ensure that permissions needed to create projects and project versions are granted on Black Duck Hub.

Synopsys action runs full “intelligent” Black Duck scans on SCM push events and “rapid” ephemeral scans for SCM pull request events as shown in the example below.



**Note:**

Detect specific options can be passed to Synopsys Bridge thru Detect environment variables.

Synopsys Bridge CLI Guide | 4 - GitHub - Synopsys Action | 38

Below is an example of a *workflow*.yml file configured for Black Duck.

name: bd-sig-action

on:

push:

branches: [ main, master, develop, stage, release ] pull\_request:

branches: [ main, master, develop, stage, release ] workflow\_dispatch:

jobs:

build:

runs-on: [ ubuntu-latest ] steps:

* name: Checkout Source uses: actions/checkout@v3
* name: Black Duck Full Scan

if: *${{ github.event\_name != 'pull\_request' }}*

uses: [synopsys-sig/synopsys-action@v1.2.0](mailto:synopsys-sig/synopsys-action@v1.2.0)

### Use below configuration to set specific detect environment variables env:

DETECT\_PROJECT\_NAME: *${{ github.event.repository.name }}*

with:

blackduck\_url: *${{ secrets.BLACKDUCK\_URL }}* blackduck\_apiToken: *${{ secrets.BLACKDUCK\_API\_TOKEN }}* blackduck\_scan\_full: true

### Accepts Multiple Values

blackduck\_scan\_failure\_severities: 'BLOCKER,CRITICAL'

Synopsys Bridge CLI Guide | 4 - GitHub - Synopsys Action | 39

### Uncomment below configuration to enable automatic fix pull request creation if vulnerabilities are

reported

# blackduck\_automation\_fixpr: true

# github\_token: *${{ secrets.GITHUB\_TOKEN }}* # Mandatory when blackduck\_automation\_fixpr is set to 'true' ### Uncomment below configuration if Synopsys Bridge diagnostic files needs to be uploaded

# include\_diagnostics: true

- name: Black Duck PR Scan

if: *${{ github.event\_name == 'pull\_request' }}*

uses: [synopsys-sig/synopsys-action@v1.2.0](mailto:synopsys-sig/synopsys-action@v1.2.0)

### Use below configuration to set specific detect environment variables env:

DETECT\_PROJECT\_NAME: *${{ github.event.repository.name }}*

with:

blackduck\_url: *${{ secrets.BLACKDUCK\_URL }}* blackduck\_apiToken: *${{ secrets.BLACKDUCK\_API\_TOKEN }}* blackduck\_scan\_full: false

### Below configuration is used to enable automatic pull request comment based on Black Duck scan result blackduck\_automation\_prcomment: true

github\_token: *${{ secrets.GITHUB\_TOKEN }}* # Mandatory when blackduck\_automation\_prcomment is set to 'true' ### Uncomment below configuration if Synopsys Bridge diagnostic files needs to be uploaded

# include\_diagnostics: true

* Synopsys Actionfor vulnerable direct dependencies.
* noteatGitHub may limit the number of pull requests that are created by Synopsys Action

**List of mandatory and optional parameters for Black Duck:**

| **Input Parameter** | **Description** | **Mandatory / Optional** |
| --- | --- | --- |
| blackduck\_url | Black Duck URL | Mandatory |
| blackduck\_apiToken | Black Duck API token | Mandatory |
| blackduck\_install\_directory | Installation directory for Black Duck | Optional |
| blackduck\_scan\_full | Specifies whether full scan is required or not.  Full "intelligent" scan is to be used for push events and rapid scan for pull request events.  Supported values: true or false | Optional |
| blackduck\_scan\_failure\_severities | Black Duck scan failure severities.  Supported values: ALL, NONE, BLOCKER, CRITICAL, MAJOR, MINOR, OK, TRIVIAL, UNSPECIFIED | Optional |
| blackduck\_automation\_prcomment | Option to enable automatic creation pull request comments for new issues found in the pull request.  Merge Request must be created first from feature branch to main branch to run Black Duck PR Comment.  Default: false | Optional |
| blackduck\_automation\_fixpr | Flag to enable automatic creation for fix pull requests for vulnerable direct dependencies.  Default: false  **Black Duck automation fix pull request is currently supported for NPM projects only.** | Optional |
| github\_token | GitHub Access Token  Example: github\_token: ${{ secrets.GITHUB\_TOKEN }} | Mandatory if blackduck\_automation\_fixpr or blackduck\_automation\_prcomment is set as true |

## Using Synopsys GitHub Action for Coverity Cloud Deployment with Thin Client

Synopsys GitHub Action only supports the Kubernetes-based Coverity cloud deployment model, which uses a small footprint thin client to capture the source code and submit an analysis job running on the server. This removes the need for a multi-gigabyte software installation in your GitHub Runner.

On push events, a full Coverity scan will be run and results are committed to the Coverity Connect database.

On pull request events, comments are added to pull requests for new issues found by the scan if coverity\_automation\_prcomment is set to true (see example below). Note that scan results are not committed to Coverity Connect database in this case.

Before running the pipeline with Synopsys Action, make sure the specified project and stream exist in your Coverity Connect server environment.

Below is an example of a workflow.yml file configured for Coverity Cloud Deployment.

name: cnc-sig-action on:

push:

branches: [ main, master, develop, stage, release ] pull\_request:

branches: [ main, master, develop, stage, release ] workflow\_dispatch:

jobs:

build:

runs-on: [ ubuntu-latest ] steps:

* name: Checkout Source uses: actions/checkout@v3
* name: Coverity Full Scan

if: *${{ github.event\_name != 'pull\_request' }}* uses: [synopsys-sig/synopsys-action@v1.2.0](mailto:synopsys-sig/synopsys-action@v1.2.0) with:

coverity\_url: *${{ secrets.COVERITY\_URL }}* coverity\_user: *${{ secrets.COVERITY\_USER }}* coverity\_passphrase: *${{ secrets.COVERITY\_PASSPHRASE }}*

coverity\_project\_name: *${{ github.event.repository.name }}* coverity\_stream\_name: *${{ github.event.repository.name }}*-*${{ github.ref\_name }}* coverity\_policy\_view: 'Outstanding Issues'

### Uncomment below configuration if Synopsys Bridge diagnostic files needs to be uploaded # include\_diagnostics: true

* name: Coverity PR Scan

if: *${{ github.event\_name == 'pull\_request' }}* uses: [synopsys-sig/synopsys-action@v1.2.0](mailto:synopsys-sig/synopsys-action@v1.2.0) with:

coverity\_url: *${{ secrets.COVERITY\_URL }}*

coverity\_user: *${{ secrets.COVERITY\_USER }}*

Synopsys Bridge CLI Guide | 4 - GitHub - Synopsys Action | 41

coverity\_passphrase: *${{ secrets.COVERITY\_PASSPHRASE }}*

coverity\_project\_name: *${{ github.event.repository.name }}*

coverity\_stream\_name: *${{ github.event.repository.name }}*-*${{ github.base\_ref }}*

### Below configuration is used to enable feedback from Coverity security testing as pull request comment coverity\_automation\_prcomment: true

github\_token: *${{ secrets.GITHUB\_TOKEN }}* # Mandatory when coverity\_automation\_prcomment is set to 'true' ### Uncomment below configuration if Synopsys Bridge diagnostic files needs to be uploaded

# include\_diagnostics: true

**List of mandatory and optional parameters for Coverity:**

| **Input Parameter** | **Description** | **Mandatory / Optional** |
| --- | --- | --- |
| coverity\_url | Coverity URL | Mandatory |
| coverity\_user | Coverity username | Mandatory |
| coverity\_passphrase | Coverity passphrase | Mandatory |
| coverity\_project\_name | Coverity project name.   Tip: Many customers prefer to set their Coverity project and stream names to match the GitHub repository name | Mandatory |
| coverity\_stream\_name | Coverity stream name | Mandatory |
| coverity\_install\_directory | Installation directory of Coverity | Optional |
| coverity\_policy\_view | ID or name of policy view to be used to enforce the “break the build” policy.  If issues are found in the specified this view, build will be failed.  Example: coverity\_policy\_view: '100001' or coverity\_policy\_view: 'Outstanding Issues' | Optional |
| coverity\_automation\_prcomment | Option to enable automatic creation pull request comments for new issues found in the pull request.  Merge Request must be created first from feature branch to main branch to run Coverity PR Comment.   Default: false | Optional |
| github\_token | GitHub Access Token  Example: github\_token: ${{ secrets.GITHUB\_TOKEN }} | Mandatory if coverity\_automation\_prcomment is set as true |

## Additional GitHub Configuration

The following parameters can be used for Polaris, Black Duck or Coverity Connect.

* synopsys\_bridge\_path: Provides the path to Synopsys Bridge.



**Note:**

If this is not explicitly specified, then the integration defaults to $HOME/synopsys-bridge. If the installed version of Synopsys Bridge is not the latest, then the latest version of Synopsys Bridge is downloaded unless you specify the version to use explicitly (as documented below).

* bridge\_download\_url: Specifies the URL to the Synopsys Bridge zip file to be downloaded and used.



**Note:**

If bridge\_download\_url is not provided, Synopsys GitHub Action downloads the latest version of Synopsys Bridge from the default SIG-REPO download location.

* bridge\_download\_version: Specifies the Synopsys Bridge version to use. If provided, the specified version of Synopsys Bridge will be automatically downloaded and used. If not, the latest version is downloaded and used.
* include\_diagnostics: When set to true, Synopsys Bridge diagnostic files are created and posted to GitHub. Additionally, diagnostics\_retention\_days can be used to specify the number of days the diagnostics files are retained for. Default value is 90. Accepted range of values is from 1 to 90.

# Chapter 5. GitLab – Synopsys Template

The Synopsys GitLab Template leverages Synopsys Bridge, allowing you to configure your GitLab pipeline to run Synopsys security testing and act on the results.

The Synopsys GitLab Template Marketplace link is <https://gitlab.com/synopsys/synopsys-template>.

#### Additional information

For additional GitLab integration information, see:

* [GitLab Prerequisites *(on*](#_bookmark26)[*page 42)*](#_bookmark26)
* GitLab Runner Setup *(on page )*
* [Using Synopsys GitLab Template with Polaris *(on*](#_bookmark27)[*page 43)*](#_bookmark27)
* [Using the Synopsys GitLab Template with Black Duck *(on*](#_bookmark28)[*page 44)*](#_bookmark28)
* [Using the Synopsys GitLab Template for Coverity Cloud Deployment with Thin Client *(on*](#_bookmark29)[*page 47)*](#_bookmark29)
* [Additional GitLab Configuration *(on*](#_bookmark30)[*page 48)*](#_bookmark30)

## GitLab Prerequisites

Before configuring Synopsys Template into your GitLab pipeline, set up the following.

#### GitLab Runner Setup

* GitLab Runner is an application that works with GitLab CI/CD to run jobs in a pipeline. To use GitLab Runner in your project, you must have the maintainer or owner role for the project.
* A GitLab runner can be self-managed or SaaS runners managed by GitLab.
* A GitLab self-managed runner can be installed and used on GNU/Linux, macOS and Windows. For more details refer: [Install GitLab Runner](https://docs.gitlab.com/runner/install/)
* To set up project specific self-managed runner, go to (**Project Settings** → **CI/CD** → **Runners**) and

configure.

* During runner registration, choose executor as shell.
* Make sure you have curl and unzip package tools installed in self-managed/SaaS runner (Linux/ Mac).
* Synopsys Template supports both Project runners and Shared runners (except Shared Mac Runners).

Synopsys Bridge CLI Guide | 5 - GitLab – Synopsys Template | 43

#### Configure GitLab Variables

* Sensitive data such as access tokens, user names, passwords and even URLs must be configured using GitLab variables.
* These can be added at the Project, Group or Global scopes (Global for self-managed GitLab instances only).
* To add variables, go to **Settings** → **CI/CD** → **Variables**. Be sure to mask passwords and tokens to

avoid them being exposed in logs. For more details see [GitLab CI/CD variables](https://docs.gitlab.com/ee/ci/variables).

#### Configure Gitlab User Token

* BRIDGE\_GITLAB\_USER\_TOKEN is required as input when running Black Duck Fix PR, Black Duck/ Coverity PR Comment.
* Generate a Personal Access Token (PAT) from GitLab (**User Settings** → **Access Tokens**) and store

it as secret variable or store and fetch it from vault.

* PAT must have **api** scope to perform Black Duck Fix PR or Black Duck/Coverity PR Comment. For more details, see: [Personal access tokens](https://docs.gitlab.com/ee/user/profile/personal_access_tokens.html)

#### Create a .gitlab-ci.yml file

* Before running a pipeline using the Synopsys Template, add a .gitlab-ci.yml file to your project by adding an include entry.
* Push those changes and a GitLab runner picks up the job and initiates the pipeline.

## Using the Synopsys GitLab Template with Polaris

Before running a pipeline using the Synopsys Template and Polaris, add .gitlab-ci.yml to your project by adding an include entry, as in the example below.

include:

- project: synopsys/synopsys-template ref: v1.1.0

file: templates/synopsys-template.yml

### Use below configuration for accessing synopsys-template in GitLab self-managed

# - remote: 'https://gitlab.com/synopsys/synopsys-template/-/raw/v1.0.0/templates/synopsys-template.yml' variables:

BRIDGE\_POLARIS\_SERVERURL: *$POLARIS\_SERVER\_URL* BRIDGE\_POLARIS\_ACCESSTOKEN: *$POLARIS\_ACCESS\_TOKEN* BRIDGE\_POLARIS\_APPLICATION\_NAME: *$CI\_PROJECT\_NAME*

BRIDGE\_POLARIS\_PROJECT\_NAME: *$CI\_PROJECT\_NAME*

Synopsys Bridge CLI Guide | 5 - GitLab – Synopsys Template | 44

### Accepts Multiple Values BRIDGE\_POLARIS\_ASSESSMENT\_TYPES: 'SCA,SAST'

stages:

- polaris\_scan synopsys\_template\_execution:

stage: polaris\_scan tags:

- linux # Name of your GitLab runner extends: .run-synopsys-tools # Used for bash

# extends: .run-synopsys-tools-powershell # Used for powershell

### Uncomment below configuration if Synopsys Bridge diagnostic files needs to be uploaded # variables:

# INCLUDE\_DIAGNOSTICS: 'true'

# artifacts:

# when: always # paths:

# - .bridge

Configure sensitive data like access tokens and URLs using GitLab secrets.



**Note:**

Polaris does not currently support the analysis of merge requests. We recommend running the Synopsys GitLab Template on pushes to main branches.

Push those changes and an active runner will pick up the job and initiate the pipeline.

## Using the Synopsys GitLab Template with Black Duck

Synopsys GitLab Template supports both self-hosted (on-prem) and Synopsys-hosted Black Duck Hub instances.

In the default Black Duck Hub permission model, projects and project versions are created on the fly as needed.

Before running a pipeline using the Synopsys GitLab Template and Black Duck, add .gitlab-ci.yml to your project by adding an include entry, as in the example below.

Synopsys Bridge CLI Guide | 5 - GitLab – Synopsys Template | 45

include:

- project: synopsys/synopsys-template

ref: v1.1.0

file: templates/synopsys-template.yml

### Use below configuration for accessing synopsys-template in Gitlab self-managed

# - remote: 'https://gitlab.com/synopsys/synopsys-template/-/raw/v1.1.0/templates/synopsys-template.yml'

stages:

* blackduck\_scan

variables:

SCAN\_BRANCHES: "/^(main|master|develop|stage|release|feature\_branch)$/" # Add branches where you want to run Black Duck scan

synopsys\_template\_execution:

stage: blackduck\_scan

variables:

BRIDGE\_BLACKDUCK\_URL: *$BLACKDUCK\_URL*

BRIDGE\_BLACKDUCK\_TOKEN: *$BLACKDUCK\_API\_TOKEN*

### Use below configuration to set specific detect environment variables

DETECT\_PROJECT\_NAME: *$CI\_PROJECT\_NAME*

### Uncomment below configuration if Synopsys Bridge diagnostic files needs to be uploaded

# INCLUDE\_DIAGNOSTICS: 'true'

# artifacts:

# when: always

# paths:

# - .bridge

rules:

### Use below configuration to run Black Duck full scan

- if: (*$CI\_COMMIT\_BRANCH* =~ *$SCAN\_BRANCHES* && *$CI\_PIPELINE\_SOURCE* != 'merge\_request\_event')

variables:

BRIDGE\_BLACKDUCK\_SCAN\_FULL: 'true'

### Accepts Multiple Values

BRIDGE\_BLACKDUCK\_SCAN\_FAILURE\_SEVERITIES: 'BLOCKER,CRITICAL'

### Uncomment below configuration to enable automatic fix pull request creation if vulnerabilities are reported

# BRIDGE\_BLACKDUCK\_AUTOMATION\_FIXPR: 'true'

Synopsys Bridge CLI Guide | 5 - GitLab – Synopsys Template | 46

BRIDGE\_GITLAB\_USER\_TOKEN: *$GITLAB\_USER\_TOKEN* # Mandatory when BRIDGE\_BLACKDUCK\_AUTOMATION\_FIXPR is set to

'true'

### Use below configuration to run Black Duck PR scan

- if: (*$CI\_MERGE\_REQUEST\_TARGET\_BRANCH\_NAME* =~ *$SCAN\_BRANCHES* && *$CI\_PIPELINE\_SOURCE* == 'merge\_request\_event') variables:

BRIDGE\_BLACKDUCK\_SCAN\_FULL: 'false' BRIDGE\_BLACKDUCK\_AUTOMATION\_PRCOMMENT: 'true'

BRIDGE\_GITLAB\_USER\_TOKEN: $GITLAB\_USER\_TOKEN

tags:

- linux # Name of your Gitlab runner extends: .run-synopsys-tools # Used for bash.

#extends: .run-synopsys-tools-powershell # Used for powershell

#### Optional Parameters for Black Duck

* + BRIDGE\_BLACKDUCK\_SCAN\_FULL: To enable full scan set it as true. By default, pushes will initiate a full "intelligent" scan and pull requests will initiate a rapid scan.
  + BRIDGE\_BLACKDUCK\_INSTALL\_DIRECTORY: Pass the directory path to install Black Duck on the runner.
  + BRIDGE\_BLACKDUCK\_SCAN\_FAILURE\_SEVERITIES: Values are ALL|NONE|BLOCKER|CRITICAL|MAJOR|MINOR|

OK|TRIVIAL|UNSPECIFIED. Single parameter as ALL and multiple parameters as ,

CRITICAL

BLOCKER,

TRIVIAL.

* + BRIDGE\_BLACKDUCK\_AUTOMATION\_PRCOMMENT: Set true to enable feedback from Black Duck testing as pull request comment.
  + BRIDGE\_BLACKDUCK\_AUTOMATION\_FIXPR: Set true to enable automatc fix pull request creation.
  + BRIDGE\_GITLAB\_USER\_TOKEN: It is mandatory when BRIDGE\_BLACKDUCK\_AUTOMATION\_PRCOMMENT is set to true, or when BRIDGE\_BLACKDUCK\_AUTOMATION\_FIXPR is set to true. Generate Personal Access Token (PAT) from GitLab and store it as secret variable or store and fetch it from vault. The PAT must have api scope. For more info, see - <https://docs.gitlab.com/ee/ci/secrets/index.html>"



**Note:**

**About Detect command line parameters**. Any command line parameters passed to detect can be passed through variables. This is a standard capability of Detect. For example, if you want to only report newly found policy violations on rapid scans, you would normally use the command line -- detect.blackduck.rapid.compare.mode=BOM\_COMPARE\_STRICT. You can replace this by setting the DETECT\_BLACKDUCK\_RAPID\_COMPARE\_MODE variable to BOM\_COMPARE\_STRICT.

Synopsys Bridge CLI Guide | 5 - GitLab – Synopsys Template | 47

## Using the Synopsys GitLab Template for Coverity Cloud Deployment with Thin Client

Before running Coverity using the Synopsys Template, ensure the appropriate project and stream are set in your Coverity Connect server environment, as in the example below.



**Note:**

Currently, Synopsys Template only supports the Coverity thin client/cloud deployment model, which removes the need for a large footprint GitLab Runner installation.

include:

- project: synopsys/synopsys-template ref: v1.1.0

file: templates/synopsys-template.yml

### Use below configuration for accessing synopsys-template in Gitlab self-managed

# - remote: 'https://gitlab.com/synopsys/synopsys-template/-/raw/v1.1.0/templates/synopsys-template.yml'

stages:

- coverity\_scan

variables:

SCAN\_BRANCHES: "/^(main|master|develop|stage|release|feature\_branch)$/" # Add branches where you want to run Coverity scan

synopsys\_template\_execution: stage: coverity\_scan variables:

BRIDGE\_COVERITY\_CONNECT\_URL: *$COVERITY\_URL* BRIDGE\_COVERITY\_CONNECT\_USER\_NAME: *$COVERITY\_USER* BRIDGE\_COVERITY\_CONNECT\_USER\_PASSWORD: *$COVERITY\_PASSWORD* BRIDGE\_COVERITY\_CONNECT\_PROJECT\_NAME: *$CI\_PROJECT\_NAME*

### Uncomment below configuration if Synopsys Bridge diagnostic files needs to be uploaded # INCLUDE\_DIAGNOSTICS: 'true'

# artifacts:

# when: always # paths:

# - .bridge

rules:

Synopsys Bridge CLI Guide | 5 - GitLab – Synopsys Template | 48

* if: (*$CI\_COMMIT\_BRANCH* =~ *$SCAN\_BRANCHES* && *$CI\_PIPELINE\_SOURCE* != 'merge\_request\_event') variables:

BRIDGE\_COVERITY\_CONNECT\_STREAM\_NAME: *$CI\_PROJECT\_NAME*-*$CI\_COMMIT\_BRANCH*

BRIDGE\_COVERITY\_CONNECT\_POLICY\_VIEW: 'Outstanding Issues' ### Use below configuration to run Coverity PR scan

* if: (*$CI\_MERGE\_REQUEST\_TARGET\_BRANCH\_NAME* =~ *$SCAN\_BRANCHES* && *$CI\_PIPELINE\_SOURCE* == 'merge\_request\_event') variables:

BRIDGE\_COVERITY\_CONNECT\_STREAM\_NAME: *$CI\_PROJECT\_NAME*-*$CI\_MERGE\_REQUEST\_TARGET\_BRANCH\_NAME*

### Below configuration is used to enable feedback from Coverity security testing as pull request comment BRIDGE\_COVERITY\_AUTOMATION\_PRCOMMENT: 'true'

BRIDGE\_GITLAB\_USER\_TOKEN: *$GITLAB\_USER\_TOKEN* # Mandatory when BRIDGE\_COVERITY\_AUTOMATION\_PRCOMMENT is set to

'true' tags:

* linux # Name of your Gitlab runner extends: .run-synopsys-tools # Used for bash.

#extends: .run-synopsys-tools-powershell # Used for powershell

#### Optional Parameters for Cloud Coverity

* + BRIDGE\_COVERITY\_INSTALL\_DIRECTORY: Specifies an install directory other than the default to install Coverity.
  + BRIDGE\_COVERITY\_CONNECT\_POLICY\_VIEW: Policy view in Coverity. ID number/Name of a saved view to apply as a break the build policy. If any defects are found within this view when applied to the

project, the build fails with an exit code. Example: BRIDGE\_COVERITY\_CONNECT\_POLICY\_VIEW: '100001'

or BRIDGE\_COVERITY\_CONNECT\_POLICY\_VIEW:'Outstanding Issues'.

* + BRIDGE\_COVERITY\_AUTOMATION\_PRCOMMENT: Enables feedback from Coverity security testing as pull request comment. Merge request must be created first from feature branch to main branch to run Coverity PR Comment. Values are true or false.
  + BRIDGE\_GITLAB\_USER\_TOKEN: Gitlab User Access Token Example: BRIDGE\_GITLAB\_USER\_TOKEN:

$GITLAB\_USER\_TOKEN. Mandatory when BRIDGE\_COVERITY\_AUTOMATION\_PRCOMMENT is set as true.

## Additional GitLab Configuration

The following parameters can be used for Polaris, Black Duck or Coverity Connect.

Synopsys Bridge CLI Guide | 5 - GitLab – Synopsys Template | 49

* + SYNOPSYS\_BRIDGE\_PATH: Provide a path, where you want to configure or already configured Synopsys Bridge. Optional.



**Note:**

If you don't provide any path, then the configuration path defaults to $HOME/synopsys- bridge.

* + DOWNLOAD\_BRIDGE\_URL: Provides a URL to the Synopsys Bridge zip file. If provided, Synopsys Bridge is automatically downloaded and configured.
  + DOWNLOAD\_BRIDGE\_VERSION: Provide a Synopsys Bridge version. If provided, the specified version of Synopsys Bridge is automatically downloaded and configured.



**Note:**

If DOWNLOAD\_BRIDGE\_VERSION or DOWNLOAD\_BRIDGE\_URL are not provided, Synopsys Template automatically downloads and configures the latest version of Synopsys Bridge. As per current behavior, the existing directory will be cleaned and then Synopsys Template automatically downloads and configures Synopsys Bridge every time.

* + INCLUDE\_DIAGNOSTICS: Pass true to upload Synopsys Bridge diagnostic files.



**Note:**

While including Synopsys Bridge diagnostic files, default expiry time for uploaded artifacts is 30 days. Expiry time for job artifacts can be updated accordingly. Refer to SCM documentation for more details :<https://docs.gitlab.com/ee/ci/jobs/job_artifacts.html>.

# Chapter 6. Azure DevOps - Synopsys Security Scan

Synopsys Security Scan Extension for Azure DevOps enables you to configure your Azure pipeline to run Synopsys security testing and take action on the results. Synopsys Security Scan leverages Synopsys Bridge, allowing you to run tests for several Synopsys products from the command line.

The Synopsys Security Scan for Azure DevOps Visual Studio Marketplace link is [https://](https://marketplace.visualstudio.com/items?itemName=synopsys-security-scan.synopsys-security-scan) [marketplace.visualstudio.com/items?itemName=synopsys-security-scan.synopsys-security-scan](https://marketplace.visualstudio.com/items?itemName=synopsys-security-scan.synopsys-security-scan).



**Note:**

Synopsys Security Scan for Azure DevOps requires appropriate licenses for all Synopsys applications used.

#### Additional Info

For additional Azure integration information, see:

* + [Azure Prerequisites *(on*](#_bookmark32)[*page 50)*](#_bookmark32)
  + [Using Azure DevOps Extension with Polaris *(on*](#_bookmark33)[*page 51)*](#_bookmark33)
  + [Using Azure DevOps Extension with Black Duck *(on*](#_bookmark34)[*page 52)*](#_bookmark34)
  + [Using Azure DevOps Extension with Coverity Connect with Thin Client *(on*](#_bookmark35)[*page 54)*](#_bookmark35)
  + [Additional Azure Configuration *(on*](#_bookmark30)[*page 48)*](#_bookmark30)

## Azure Prerequisites

Before configuring Synopsys Security Scan into your azure pipeline, note the following prerequisites:

#### Azure Agent Setup

Agents can be installed and used on GNU/Linux, macOS, Windows and Docker. See [https://](https://learn.microsoft.com/en-us/azure/devops/pipelines/agents/agents?view=azure-devops&tabs=browser) [learn.microsoft.com/en-us/azure/devops/pipelines/agents/agents?view=azure-devops&tabs=browser](https://learn.microsoft.com/en-us/azure/devops/pipelines/agents/agents?view=azure-devops&tabs=browser) for details. You can use Microsoft-hosted agents as well to scan your code using Azure Pipelines.

#### Configure Variables

Sensitive data such as access tokens, user names, passwords and even URLs must be configured using variable groups (**Project** → **Pipelines** → **Library** → **New Variable Group**).

AZURE\_TOKEN is required as input when running Black Duck Fix PR, Black Duck/Coverity PR Comment. There are two different types of tokens to pass to AZURE\_TOKEN:

Synopsys Bridge CLI Guide | 6 - Azure DevOps - Synopsys Security Scan | 51

* + To use AZURE\_TOKEN: $(System.AccessToken), you must enable this in the Azure interface. Go to **Project** → **Project Settings** → **Repository** → **Security** → **Build Service** and set **Contribute to pull requests**, **Create branch** and **Delete or disable repository** to **Allow**. Confirm System.AccessToken has **Contribute to PR** permissions (**Project** → **Project Settings** → **Repositories** → **Security** → **Build Service User**).
  + To use AZURE\_TOKEN: $(PAT\_TOKEN), PAT token should have minimum permissions **Code - Full** and

**Pull Request Threads - Read & write**. See [Use personal access tokens](https://learn.microsoft.com/en-us/azure/devops/organizations/accounts/use-personal-access-tokens-to-authenticate?view=azure-devops&tabs=Windows) for more details.

For Black Duck and Coverity PR comments, enable **Build validation policy** (**Project** → **Project Settings** → **Repositories** → **Branch Policy** → **Add branch protection**) to trigger the pipeline on raising PR or any push event to existing branch (usually on main or master branch). See [Build Validation](https://learn.microsoft.com/en-us/azure/devops/repos/git/branch-policies?view=azure-devops&tabs=browser&build-validation) for more details.

#### Configure Azure Pipeline

Create a new pipeline or use existing pipeline (**Project** → **Pipelines** → **New Pipeline**) and configure required fields. Push those changes and agent will pick up the job and initiate the pipeline.

## Using Azure DevOps Extension with Polaris

Before running a pipeline using the Synopsys Security Scan and Polaris, add azure-pipelines.yml to your project. Configure sensitive data like usernames, passwords and URLs using pipeline variables. Push those changes and agent will pick up the job and initiate the pipeline. An example of the changes you need to add to azure-pipelines.yml is shown below.

trigger:

- main

pool:

vmImage: ubuntu-latest

variables:

- group: polaris

steps:

- task: [SynopsysSecurityScan@1.0.0](mailto:SynopsysSecurityScan@1.0.0) displayName: 'Polaris Scan' inputs:

BRIDGE\_POLARIS\_SERVERURL: *$(POLARIS\_SERVER\_URL)*

BRIDGE\_POLARIS\_ACCESSTOKEN: *$(POLARIS\_ACCESS\_TOKEN)*

Synopsys Bridge CLI Guide | 6 - Azure DevOps - Synopsys Security Scan | 52

BRIDGE\_POLARIS\_APPLICATION\_NAME: *$(Build.Repository.Name)* BRIDGE\_POLARIS\_PROJECT\_NAME: *$(Build.Repository.Name)* ### Accepts Multiple Values BRIDGE\_POLARIS\_ASSESSMENT\_TYPES: 'SCA,SAST'

### Uncomment below configuration if Synopsys Bridge diagnostic files needs to be uploaded

# INCLUDE\_DIAGNOSTICS: 'true'

Descriptions of these arguments are shown in the [Complete List of Synopsys Bridge Arguments *(on*](#_bookmark16)[*page*](#_bookmark16)[*18)*](#_bookmark16).

## Using Azure DevOps Extension with Black Duck

Synopsys Security Scan supports both self-hosted (e.g. on-prem) and Synopsys-hosted Black Duck Hub instances. In the default Black Duck Hub permission model, projects and project versions are created on the fly as needed. Configure sensitive data like usernames, passwords and URLs using pipeline variables. An example of additions to azure-pipelines.yml to run Black Duck is shown below.

trigger:

- main

pool:

vmImage: ubuntu-latest

variables:

- group: blackduck

steps:

- task: [SynopsysSecurityScan@1.0.0](mailto:SynopsysSecurityScan@1.0.0) displayName: 'Black Duck Full Scan'

condition: not(eq(variables['Build.Reason'], 'PullRequest'))

### Use below configuration to set specific detect environment variables env:

DETECT\_PROJECT\_NAME: *$(Build.Repository.Name)*

inputs:

BRIDGE\_BLACKDUCK\_URL: *$(BLACKDUCK\_URL)*

BRIDGE\_BLACKDUCK\_TOKEN: $*(BLACKDUCK\_TOKEN)*

BRIDGE\_BLACKDUCK\_SCAN\_FULL: true

### Accepts Multiple Values BRIDGE\_BLACKDUCK\_SCAN\_FAILURE\_SEVERITIES: 'BLOCKER,CRITICAL'

Synopsys Bridge CLI Guide | 6 - Azure DevOps - Synopsys Security Scan | 53

### Uncomment below configuration to enable automatic fix pull request creation if vulnerabilities are reported # BRIDGE\_BLACKDUCK\_AUTOMATION\_FIXPR: true

# AZURE\_TOKEN: *$(System.AccessToken)* # Mandatory when BRIDGE\_BLACKDUCK\_AUTOMATION\_FIXPR is set to 'true' ### Uncomment below configuration if Synopsys Bridge diagnostic files needs to be uploaded

# INCLUDE\_DIAGNOSTICS: true

- task: [SynopsysSecurityScan@1.0.0](mailto:SynopsysSecurityScan@1.0.0) displayName: 'Black Duck PR Scan'

condition: eq(variables['Build.Reason'], 'PullRequest')

### Use below configuration to set specific detect environment variables env:

DETECT\_PROJECT\_NAME: $*(Build.Repository.Name)* inputs:

BRIDGE\_BLACKDUCK\_URL: *$(BLACKDUCK\_URL)*

BRIDGE\_BLACKDUCK\_TOKEN: *$(BLACKDUCK\_API\_TOKEN)*

BRIDGE\_BLACKDUCK\_SCAN\_FULL: false

### Below configuration is used to enable automatic pull request comment based on Black Duck scan result BRIDGE\_BLACKDUCK\_AUTOMATION\_PRCOMMENT: true

AZURE\_TOKEN: *$(System.AccessToken)* # Mandatory when BRIDGE\_BLACKDUCK\_AUTOMATION\_PRCOMMENT is set to 'true' ### Uncomment below configuration if Synopsys Bridge diagnostic files needs to be uploaded

# INCLUDE\_DIAGNOSTICS: true

#### Optional Parameters for Black Duck

* + BRIDGE\_BLACKDUCK\_SCAN\_FULL: To enable full scan set it as true. By default, pushes will initiate a full "intelligent" scan and pull requests will initiate a rapid scan.
  + BRIDGE\_BLACKDUCK\_INSTALL\_DIRECTORY: Pass the directory path to install Black Duck on the runner.
  + BRIDGE\_BLACKDUCK\_SCAN\_FAILURE\_SEVERITIES: Values are ALL|NONE|BLOCKER|CRITICAL|MAJOR|MINOR|

OK|TRIVIAL|UNSPECIFIED. Single parameter as ALL and multiple parameters as ,

CRITICAL

BLOCKER,

TRIVIAL.

* + BRIDGE\_BLACKDUCK\_AUTOMATION\_PRCOMMENT: Set true to enable feedback from Black Duck testing as pull request comment.
  + BRIDGE\_BLACKDUCK\_AUTOMATION\_FIXPR: Set true to enable automatc fix pull request creation.



**Note:**

**About Detect command line parameters**. Any command line parameters passed to detect can be passed through variables. This is a standard capability of Detect. For example, if you want to only report newly found policy violations on rapid scans, you would normally use the command line --

Synopsys Bridge CLI Guide | 6 - Azure DevOps - Synopsys Security Scan | 54



detect.blackduck.rapid.compare.mode=BOM\_COMPARE\_STRICT. You can replace this by setting the

DETECT\_BLACKDUCK\_RAPID\_COMPARE\_MODE variable to BOM\_COMPARE\_STRICT.

See the [Complete List of Synopsys Bridge Arguments *(on*](#_bookmark16)[*page 18)*](#_bookmark16) for details of Black Duck arguments.

If bridge\_blackduck\_automation\_prcomment or bridge\_blackduck\_automation\_fixpr is set to true, you must also pass azure\_token with the required permissions. (Example: azure\_token: *(System.AccessToken)*.)

## Using Azure DevOps Extension with Coverity Connect with Thin Client

Currently, Synopsys Security Scan only supports the Coverity Connect with thin client deployment model, which removes the need for a large footprint installation in your agent. Before running Coverity Connect using the Synopsys Security Scan for Azure DevOps Extension, ensure the appropriate project and stream are set in your Coverity Connect server environment. Configure sensitive data like usernames, passwords and URLs using pipeline variables. An example of additions to azure-pipelines.yml to run Coverity Connect with Thin Client is shown below.

trigger:

- main

pool:

vmImage: ubuntu-latest

variables:

- group: coverity

steps:

- task: [SynopsysSecurityScan@1.0.0](mailto:SynopsysSecurityScan@1.0.0) displayName: 'Coverity Full Scan'

condition: not(eq(variables['Build.Reason'], 'PullRequest')) inputs:

BRIDGE\_COVERITY\_CONNECT\_URL: *$(COVERITY\_URL)* BRIDGE\_COVERITY\_CONNECT\_USER\_NAME: *$(COVERITY\_USER)* BRIDGE\_COVERITY\_CONNECT\_USER\_PASSWORD: *$(COVERITY\_PASSPHRASE)*

BRIDGE\_COVERITY\_CONNECT\_PROJECT\_NAME: *$(Build.Repository.Name)* BRIDGE\_COVERITY\_CONNECT\_STREAM\_NAME: *$(Build.Repository.Name)*-*$(Build.SourceBranchName)* BRIDGE\_COVERITY\_CONNECT\_POLICY\_VIEW: 'Outstanding Issues'

### Uncomment below configuration if Synopsys Bridge diagnostic files needs to be uploaded

Synopsys Bridge CLI Guide | 6 - Azure DevOps - Synopsys Security Scan | 55

# include\_diagnostics: true

- task: [SynopsysSecurityScan@1.0.0](mailto:SynopsysSecurityScan@1.0.0) displayName: 'Coverity PR Scan'

condition: eq(variables['Build.Reason'], 'PullRequest') inputs:

BRIDGE\_COVERITY\_CONNECT\_URL: *$(COVERITY\_URL)* BRIDGE\_COVERITY\_CONNECT\_USER\_NAME: *$(COVERITY\_USER)* BRIDGE\_COVERITY\_CONNECT\_USER\_PASSWORD: *$(COVERITY\_PASSPHRASE)*

BRIDGE\_COVERITY\_CONNECT\_PROJECT\_NAME: *$(Build.Repository.Name)*

BRIDGE\_COVERITY\_CONNECT\_STREAM\_NAME: *$(Build.Repository.Name)*-*$(Build.targetBranchName)*

### Below configuration is used to enable feedback from Coverity security testing as pull request comment BRIDGE\_COVERITY\_AUTOMATION\_PRCOMMENT: true

AZURE\_TOKEN: $(System.AccessToken) # Mandatory when BRIDGE\_COVERITY\_AUTOMATION\_PRCOMMENT is set to 'true' ### Uncomment below configuration if Synopsys Bridge diagnostic files needs to be uploaded

# include\_diagnostics: true

#### Optional Parameters for Cloud Coverity

* + BRIDGE\_COVERITY\_INSTALL\_DIRECTORY: Specifies an install directory other than the default to install Coverity.
  + BRIDGE\_COVERITY\_CONNECT\_POLICY\_VIEW: Policy view in Coverity. ID number/Name of a saved view to apply as a break the build policy. If any defects are found within this view when applied to the

project, the build fails with an exit code. Example: BRIDGE\_COVERITY\_CONNECT\_POLICY\_VIEW: '100001'

or BRIDGE\_COVERITY\_CONNECT\_POLICY\_VIEW:'Outstanding Issues'.

* + BRIDGE\_COVERITY\_AUTOMATION\_PRCOMMENT: Enables feedback from Coverity security testing as pull request comment. Merge request must be created first from feature branch to main branch to run Coverity PR Comment. Values are true or false.

See the [Complete List of Synopsys Bridge Arguments *(on*](#_bookmark16)[*page 18)*](#_bookmark16) for details of Coverity Connect arguments.

If bridge\_coverity\_automation\_prcomment is set to true, you must also pass azure\_token with the required permissions. (Example: azure\_token: $*(System.AccessToken)*.)

## Additional Azure Configuration

The following parameters can be used for Polaris, Black Duck or Coverity Connect.

Synopsys Bridge CLI Guide | 6 - Azure DevOps - Synopsys Security Scan | 56

* + BRIDGE\_DOWNLOAD\_URL: Provides URL to Synopsys Bridge zip file. If provided, Synopsys Bridge is automatically downloaded and configured.
  + BRIDGE\_DOWNLOAD\_VERSION: Provides Synopsys Bridge version. If provided, the specified version of Synopsys Bridge is automatically downloaded and configured.



**Note:**

If DOWNLOAD\_BRIDGE\_VERSION or DOWNLOAD\_BRIDGE\_URL are not provided, Synopsys Template automatically downloads and configures the latest version of Synopsys Bridge. As per current behavior, the existing directory will be cleaned and then Synopsys Template automatically downloads and configures Synopsys Bridge every time.

* + SYNOPSYS\_BRIDGE\_PATH: Provide a path, where you want to configure or already configured Synopsys Bridge. Optional.



**Note:**

If you don't provide any path, then the configuration path defaults to $HOME/synopsys- bridge.

* + include\_diagnostics: Pass true to upload Synopsys Bridge diagnostic files. Azure DevOps no longer supports per-pipeline retention rules. The only way to configure retention policies for YAML and classic pipelines is through the project settings. For more details, see [Set run retention policies](https://learn.microsoft.com/en-us/azure/devops/pipelines/policies/retention?view=azure-devops&tabs=yaml&set-run-retention-policies).

# Chapter 7. Glossary

Here are terms and concepts used by Synopsys Bridge and the various Synopsys programs with which it interfaces.

|  |  |
| --- | --- |
| **Term** | **Definition** |
| **Application** | The software security tool used to scan code. |
| **Application Security** | Application security is enhancing software features to functionality to prevent security threats. These include denial of service attacks, unauthorized data access, privilege escalation attacks, etc. Applica­ tion security is one of several levels of security used to protect sys­ tems. |
| **BDSA** | Black Duck Security Advisory, highly detailed open source vulnerabil­ ity records that are hand-crafted by the Synopsys Cybersecurity Re­ search Center (CyRC) |
| **Black Duck** | Software composition analysis (SCA) security scanning tool. Helps manage the security, quality, and license compliance risks of open source and third-party code in applications and containers. Bridge in­ tegrates with Black Duck. |
| **CI/CD** | Continuous Integration/Continuous Deployment, the process by which new checked-in code is automatically built, checked for securi­ ty issues, and packaged for deployment. |
| **CLI** | Command Line Interface |
| **Coverity** | Static analysis scanning tool (SAST), which scans source code for security flaws and coding standards compliance. Bridge does not in­ tegrate with Coverity, but does integrate with Coverity Connect and CNC. |
| **Coverity Connect** | A web-based platform for Coverity. Bridge supports Coverity Con­ nect. |
| **Coverity cloud deployment** | A cloud-native version of Coverity. Bridge supports Coverity cloud de­ ployment, and every place in this manual that references "Coverity Connect" also applies to Coverity cloud deployment. |

Synopsys Bridge CLI Guide | 7 - Glossary | 58

|  |  |
| --- | --- |
| **Term** | **Definition** |
| **CVE** | Common Vulnerabilities and Exposures. A database of publicly iden­ tified, defined, and cataloged cybersecurity vulnerabilities. |
| **EULM** | End User License Management agreement |
| **GUI** | Graphic User Interface |
| **IAST** | Interactive application security testing (IAST) solutions help organi­ zations identify and manage security risks associated with vulnera­ bilities discovered in running web applications by continuously an­ alyzing all application interactions initiated by manual and/or auto­ mated tests to identify vulnerabilities in real time. |
| **Polaris** | Polaris is a cloud-native application security testing solution that pro­ vides both best-in-class SAST and SCA, making it easier to manage application security testing. Bridge integrates with Polaris. |
| **Rapid Scan Static (Sigma)** | Rapid Scan Static using the Sigma engine is a headless Static Appli­ cation Security Testing (SAST) scanner. |
| **RSQL** | REST Query Language |
| **Runner** | An application that runs a pipeline job from a CI/CD platform like GitHub or GitLab. |
| **SAST** | Static Analysis Security Testing (SAST), or static analysis, is a testing methodology that analyzes source code to find security vulnerabili­ ties. SAST scans an application before the code is compiled. Coveri­ ty is a SAST tool. |
| **SCA** | Software Composition Analysis (SCA) is an automated process iden­ tifying open source software in a codebase to evaluate security, li­ cense compliance, and code quality. Black Duck is an SCA tool. |
| **SCM** | Source Code Management. This usually refers to an online CI/CD SCM repo like GitHub, GitLab or Azure, all of which Synopsys offers integrations adaptors for. |