Continuous Security Intelligence

Proof of Concept

Why DeployHub

With an obfuscated software supply chain, IT teams struggle to know all the pieces of software they deliver to their end-users. Without insights, its hard to confirm that the software delivered is safe for consumption. The software supply chain includes thousands of open-source packages, consumed across hundreds of containers that make up the software applicaitons delivered to end-users.

DeployHub's Continuous Security Intelligence unifies supply chain forensics to expose all the pieces of software, making it easy to respond to threats and vulnerabilities within hours not months. DeployHub collects supply chain and DevOps intelligence generated by the DevOps Pipeline. DeployHub clarifies 'logical' application composition, aggregates SBOM and CVE reports from lower-level dependencies, and tracks open-source inventory across all environments.

Table of Contents

DeployHub POC Success Criteria

Implementing DeployHub's Continuous Security Intelligence will ensure that IT teams can deliver secure, high-quality software at scale by exposing the following:

1 Versioning and Component to Application Dependency Management

DeployHub will track updates and create new versions of components (containers, DB objects, File based objects) that are being continuously pushed across the supply chain.

DeployHub will automatically create new logical application versions based on changes occurring at the lower component dependency level.

DeployHub will show the 'many-to-many' relationships between components and the logical applications that consume them.

2 Supply Chain Security

DeployHub will integrate into the DevOps pipeline consuming component-level SBOMs and producing CVE reports for each new version of a component.

DeployHub will produce application-level SBOMs and CVE reports for all logical applications impacted by a lower-level component change.

3 Service Ownership and Organization

DeployHub will track component ownership and provide a simple method of knowing whom to call when a lower-level object has an issue that impacts multiple teams.

4 Component and Open-Source Usage and Inventory

DeployHub will provide the ability to search for open-source packages across all logical applications.

The DeployHub SaaS or On-Premise Installation Options

DeployHub offers a SaaS model provding you operational management and a cost effective way to implement your supply chain catalog. Alternatively you can install DeployHub on premise. Pricing is the same for both methods. For terms of use see: https://www.deployhub.com/terms-of-use/

SaaS Signup

Signup for the SaaS option at https://www.deployhub.com/deployhub-team

You will recieve an email that provides you instructions for accessing the SaaS portal at https://console.deployhub.com/

To access the SaaS portal you will be asked to enter a UserID/Password, Company and Project name. Your UserID/Password and Company name are unique. Once you login, your Project will be a found under your Company‘s high-level Domain.

On-Prem Installation

DeployHub can be installed into your own cloud environment, or onto a hosted cloud environment. DeplyHub uses Helm to manage and perform the installation. The process includes the installation of multiple containers. Note: You will not need the Reverse Proxy if you are installing into your own enviornment.

The DeployHub on-premise Helm chart and instructions can be found at ArtifactHub. This is the location for the most up to date instructions for downloading and running the DeployHub Helm chart. (https://artifacthub.io/packages/helm/deployhub/deployhub)

Installing the CI/CD CLI for Pipeline Automation

Regardless if you are running the SaaS version or an on-premise version, you will need to install the CI/CD Command Line Interface (CLI) to automate the gather of supply chain data from your pipeline workflows.

DeployHub integrates into your CI/CD process using the Ortelius Open-Source Command Line (CLI). The Ortelius CLI gathers supply chain data based on a single pipeline workflow at the build and deploy steps. The CLI will support any CI/CD engine, but does require Python. The build step gathers Swagger, SBOM, Readme, licenses, Git data, Docker image, and other build output. The deploy step records when a release occurs, what was sent and where the objects were sent to.

To complete your POC you will need to install the Ortelius CLI where your CI/CD server is running. Refer to the Ortelius GitHub CLI Documentation (https://github.com/Ortelius/cli/blob/main/doc/dh.md) for installation instructions.

The Ortelius (https://Ortelius.io) CLI is maintained by the Ortelius Open Source Community under the governance of the Linux Foundation's Continuous Delivery Foundation.

Ortelius CLI Data Gathering using the .toml File

The Ortelius CLI reads from a .toml file. The .toml file contains non-derived information for each artifact that you create at your build step. In DeployHub, an artifact is referred to as a Component. A Component is a Container, DB Object, or file object (.jar, Lamda Function, Apex file, etc.). The .toml file will provide the 'non-derived' data for the Component your are tracking in DeployHub which includes the Component name, owner, Component type, and owner contact details. The Ortelius CLI will read the .toml file from the Git Repository associated to your pipeline. If you are using a Mono Repository for your entire codebase, you will need a separate Component.toml file for each Component, managed in sub-directories.

In a cloud-native, microservice architecture there are many, if not hundreds, of Components. Organizing your Components within DeployHub is done in two ways. They are grouped based on a subject Domain and assigned to a logical Application. Not all Components need to be assigned to an Application, but they should be stored in a subject matter Domain so they can be easily found and reused.

A logical Application is a collection of Components that make up a complete software systems consumed by an end user. Applications are composed of shared Components and Application specific Components, and are a logical representation of what Components need to be deployed in order for the software system to run.

Note: Once created, your .toml file does not need to be updated unless the non-derived information changes, or

Steps for Running the Proof of Concept

To automate DeployHub, you will need to add it's data gathering to your CI/CD pipeline. The following steps will guide you through the process of implementing the Ortelius CLI to implement your Proof of Concept. Be sure you have installed the Ortelius CLI before you start.

Note: This POC does not include data gathering of the deployment for inventory tracking.

Step 1 - Define Your DeployHub Pipeline Variables

The following variables should be set at the beginning of your Pipeline.

DHURL - URL to DeployHub Login
DHUSER - The ID used to log into DeployHub
DHPASS - The password used to log into DeployHub. This can encrypted based on the CI/CD solution.
DOCKERREPO -Name of your Docker Repository .For Components that are Docker Images. Not needed for non-docker objects.
IMAGE_TAG - Tag for the Docker Image if used . For Components that are Docker Images. Not needed for non-docker objects.

Example:

```
export DHURL=https://console.deployhub.com
export DHUSER=Stella99
export DHPASS=chasinghorses
export DOCKERREPO=quay.io/DeployHub/hello-world
export IMAGE_TAG=1.0.0
```

Step 2 - Create your Component.toml file

Cut and paste the following into a component.toml file, update ‘your’ information, and commit/push it to your Git Repository.

```
# Application Name and Version - optional. If not used the Component will not be associated to an Application

Application = “GLOBAL.your Application Name”
Application_Version = “your Application Version”

# Define Component Name, Variant and Version - required
Name = “GLOBAL.your Component Name”
Variant = “${GIT_BRANCH}”
Version = “vyour Component Version.${BUILD_NUM}-g${SHORT_SHA}”

# Key/Values to associate to the Component Version
[Attributes]
    DockerBuildDate = “${BLDDATE}”
    DockerRepo = “${DOCKERREPO}”
    DockerSha = “${DIGEST}”
    DockerTag = “${IMAGE_TAG}”
    DiscordChannel = “Your Discord Channel”  or SlackChannel=”Your Slack Channel”
    ServiceOwner= “${DHUSER}”
    ServiceOwnerEmail = “Your Component Owner Email”
```

Example:

```
# Application Name and Version
Application = “GLOBAL.Santa Fe Software.Online Store Company.Hipster Store.Prod.helloworld app”
Application_Version = “1”

# Define Component Name, Variant and Version
Name = “GLOBAL.Santa Fe Software.Online Store Company”
Variant = “${GIT_BRANCH}”
Version = “v1.0.0.${BUILD_NUM}-g${SHORT_SHA}”

# Key/Values to associate to the Component Version
[Attributes]
   DockerBuildDate = “${BLDDATE}”
   DockerRepo = “${DOCKERREPO}”
   DockerSha = “${DIGEST}”
   DockerTag = “${IMAGE_TAG}”
   DiscordChannel = “ttps://discord.gg/wM4b5yEFzS”
   ServiceOwner= “${DHUSER}”
   ServiceOwnerEmail = “stella@DeployHub.io”
```

Note: For SaaS users, you will have a second high-level qualifier that was created as part of your sign-up. This second high-level qualifier must be used as the start of your Application Name and Component Name. For example: GLOBAL.Santa Fe Software.Online Store.

Step 3 - Add a step in your pipeline to run Syft if you are not generating SBOMS (Optional)

DeployHub can consume any SPDX and CycloneDX formatted SBOM. If you are already generating SBOMs, you will pass the name of the SBOM results to DeployHub is step 4 below. If you are not generating SBOMs as part of your pipeline process, you will need to add SBOM generation to collect the lower dependency data. Following is how to add Syft to your workflow to include the collection of SBOM data.

Syft SBOM tool (https://github.com/anchore/syft) will generate Software Bill of Material Reports for popular coding languages and package managers, including Docker images.

The following code example scans a Docker Image to generate the SBOM. See Syft Options (https://github.com/anchore/syft#supported-sources) to scan other objects and coding languages.

```
# install Syft
curl -sSfL https://raw.githubusercontent.com/anchore/syft/main/install.sh | sh -s -- -b $PWD

# create the SBOM
../syft packages $DOCKERREPO:$IMAGE_TAG --scope all-layers -o cyclonedx-json > cyclonedx.json

# display the SBOM
cat cyclonedx.json
```

Step 4 - Run the Ortelius CLI to add Your Component and Create an Application

Execute the following calls to the Ortelius CLI as part of your workflow. It should be called after the build and SBOM generation:

With CycloneDX SBOM

dh updatecomp --rsp component.toml --deppkg “cyclonedx@name of your SBOM file”

Example:
dh updatecomp --rsp component.toml --deppkg “cyclonedx@cyclonedx.json”

With SPDX SBOM

dh updatecomp --rsp component.toml --deppkg “spdx@name of your SBOM file. “

Example:
dh updatecomp --rsp component.toml --deppkg “spdx@spdx.json”

Without SBOM

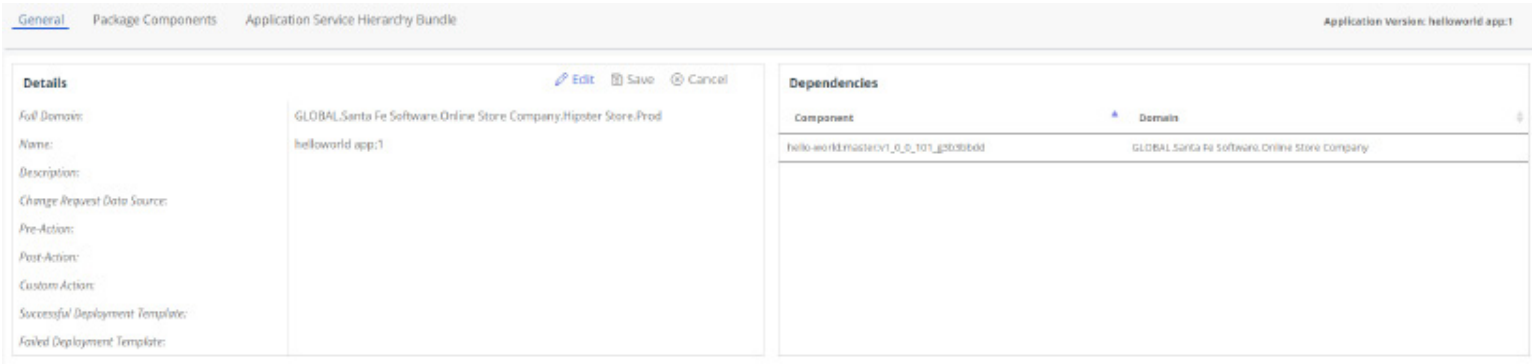
dh updatecomp --rsp component.toml

Expected Results

Bring up your DeployHub URL and login using the DHUSER and DHPASS from Step 1.

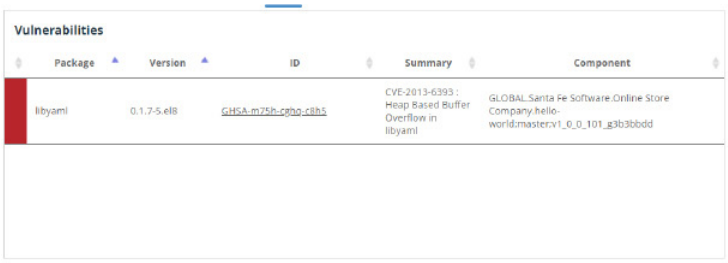
Application to Component Dependencies

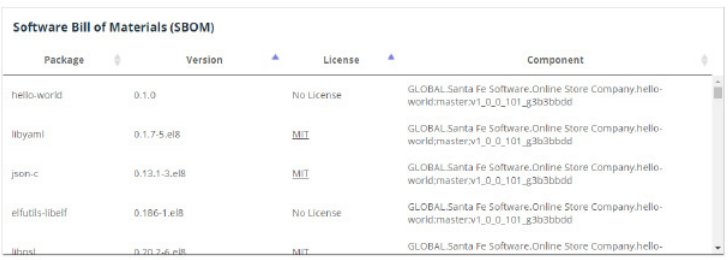
Select Your Application from the ‘Application View.’ It should show you one Component as a dependency.



Application Level SBOM and CVE

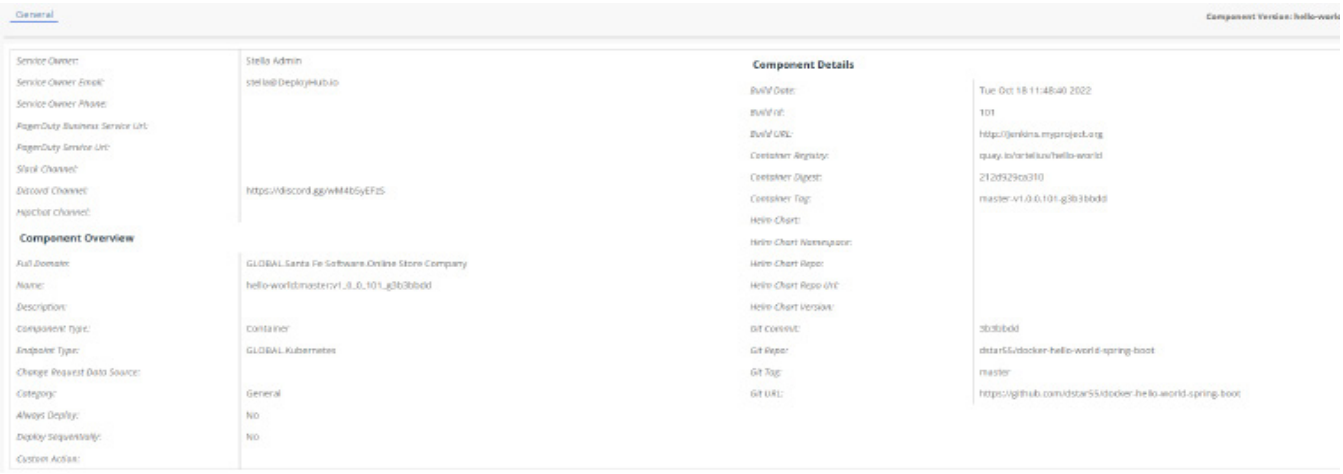
Review the Applicaiton SBOM and vulnerabilities. *Note: CVE Results may vary depending on the time of the scan.*





Component Ownership

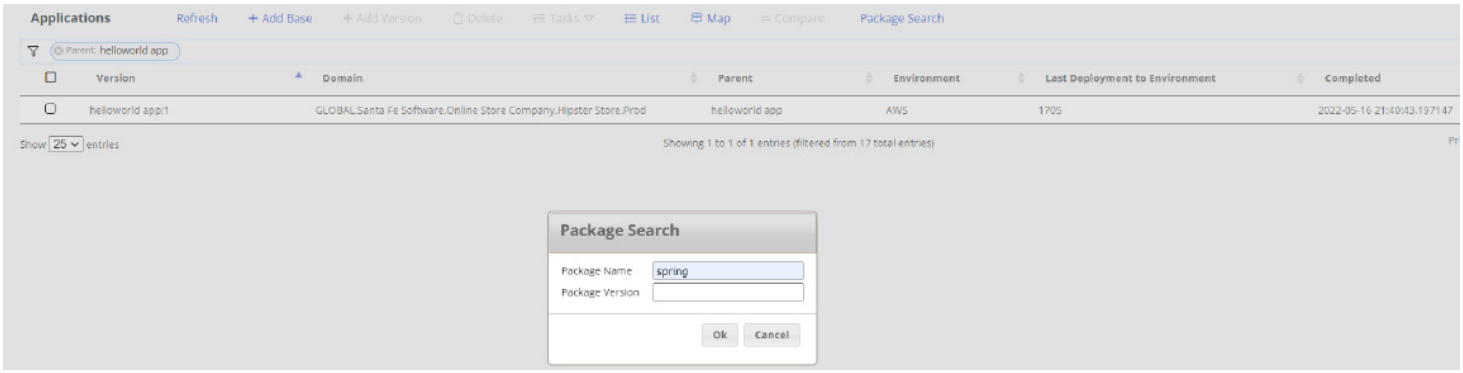
Go to the ‘Component View’. You should see your Component Ownership and Detail, including its SBOM and vulnerabilities.

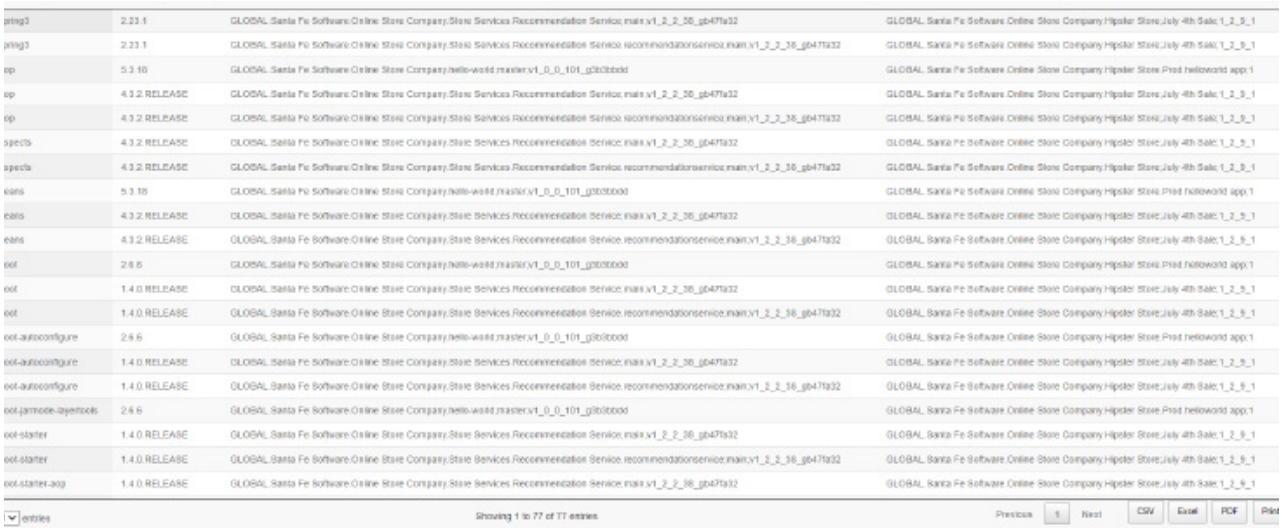


Supply Chain “Package” Search

Package Search

Go to the ‘Application View.’ Select ‘Package Search’ from the high-level menu. Enter a package name such as ‘spring’ to identify all locations where the package is used.





Next Steps

After completing these initial POC steps, you can add additional Components to your Application, update them via your pipeline, and view how DeployHub creates new versions of both Components and Applications overtime. Each time a Component is updated, you will see that a new version of all impacted “logical” Applications have been captured, showing you what changed.

You can also add CLI integration to your deployments and begin tracking your service inventory across all clusters, controlling drift and proactively understanding your ‘blast radius” caused by a single service update.

Thank you for your interest in DeployHub.

Get Help

Email us at: request-info@deployhub.com

Report an Issue: github.com/DeployHubProject/DeployHub-Pro/issues

Community Discord Channel: https://discord.gg/wM4b5yEFzS

DeployHub Documentation: https://docs.deployhub.com/userguide/

Get Involved in Open-Source

Help us create the best, open source supply chain management catalog available at ortelius.io. We believe everyone has something to offer in solving the microservice management puzzle. We would love to have you on board.

About the Author:

Tracy Ragan, CEO and Co-Founder DeployHub

Tracy is a DevOps and Open-Source security evangelist with expertise in software configuration management, and supply chain security. She has served on Boards at the Open Source Security Foundation (OpenSSF) and the Continuous Delivery Foundation. She was a founding member of the Eclipse organization and served on the board for 5 years. She is a recognized leader in open-source and has been published in multiple industry publications as well as presenting to audiences at industry conferences. Tracy co-founded DeployHub in 2019 to improve security in a the software supply chain.

Visit us at:
DeployHub.com