CD.FOUNDATION

ORTELIUS

Govern the Microservice Supply Chain and deliver secure, high-quality microservices at scale.

Why Use Ortelius

With microservices at scale, Developers, DevOps, and Security teams struggle to confirm that the software they deliver to end users is safe for consumption. A microservice environment obfuscates application-level security reporting, impact analysis, inventory, or even knowing whom to call when an end-user reports an issue. Ortelius is an open-source, centralized 'evidence store' of supply chain data that clarifies these complexities, making what's hard about cloud-native computing easy. Ortelius is governed by the Continuous Delivery Foundation a part of the Linux Foundation.

Ortelius collects supply chain and DevOps intelligence data generated by the DevOps Pipeline. The data gathered gives IT Teams critical insights about the software they deliver to end users. Ortelius clarifies 'logical' application composition, aggregates SBOM and CVE reports from lower-level dependencies and tracks open-source usage across all environments.

Table of Contents

Ortelius POC Success Criteria

Implementing the Ortelius centralized catalog of supply chain evidence will ensure that teams can deliver secure, high-quality microservices at scale by exposing the following:

1 Versioning and Component to Application Dependency Management

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

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

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

2 Supply Chain Security

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

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

3 Service Ownership and Organization

Ortelius 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

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

Installing Ortelius One-Premise

Ortelius can be installed into your own cloud environment, or onto a hosted cloud environment. Ortelius uses Helm to manage and perform the installation. The process includes the installation of multiple containers. The Ortelius 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 Ortelius Helm chart. (https://artifacthub.io/packages/helm/ortelius/ortelius)

A SaaS Option

DeployHub, the core contributors of Ortelius, offers a free SaaS option called DeployHub Team. Sign-up for the DeployHub SaaS option at deployhub.com/microservice-dashboard. 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.

Note: If another user signs up with the same Company name, they will be informed that they must contact the Administrator for access to your DeployHub account. The Administrator is the first person who signed up to DeployHub with that Company name.

Installing the CI/CD CLI for Pipeline Automation

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

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, Read-me, 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.

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 Ortelius, 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 Ortelius 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 Ortelius 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 you want to reorganize to which Applications or Domains the Component has been assigned. For example, a Component has been reassigned to a new owner and new team represented by a Domain or Application.

Steps for Running the Proof of Concept

To automate Ortelius, 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 Ortelius Pipeline Variables

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

DHURL - URL to Ortelius Login
DHUSER - The ID used to log into Ortelius
DHPASS - The password used to log into Ortelius. 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.ortelius.com
export DHUSER=Stella99
export DHPASS=chasinghorses
export DOCKERREPO=quay.io/Ortelius/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)

Ortelius can consume any SPDX and CycloneDX formatted SBOM. If you are already generating SBOMs, you will pass the name of the SBOM results to Ortelius 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

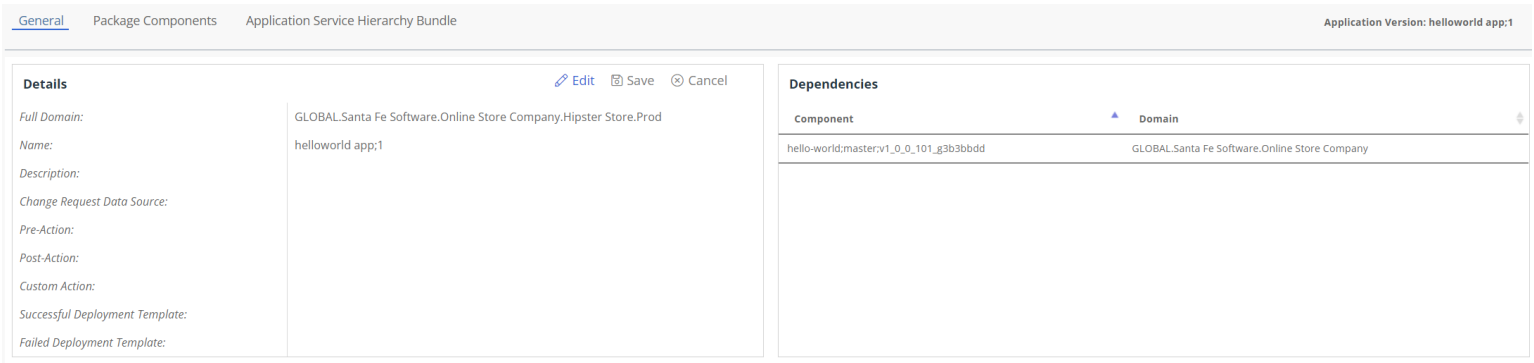
dh updatecomp --rsp component.toml

Expected Results

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

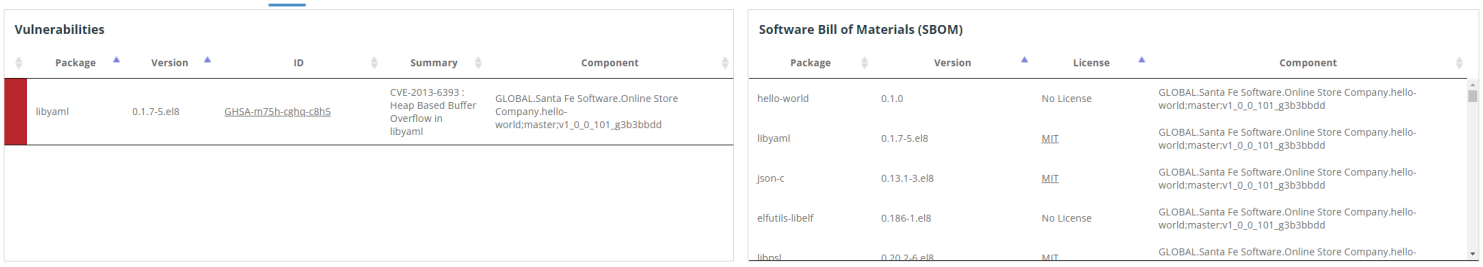
Application to Component Dependencies

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



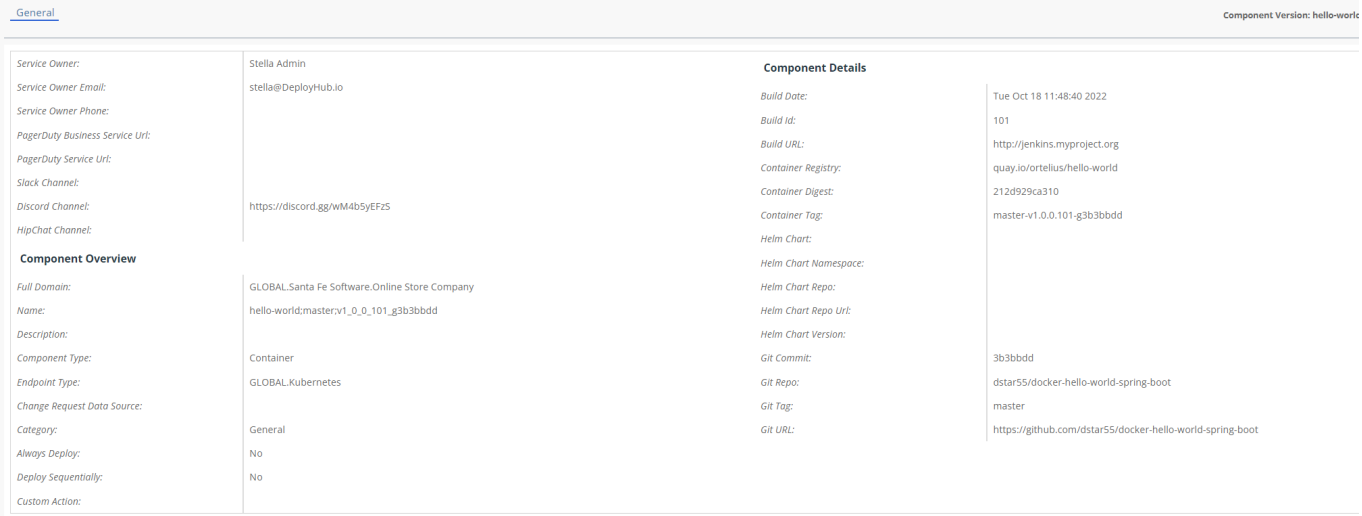
Application Level SBOM and CVE

Review the Application 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.



General

Component Version: hello-world

Service Owner:	Stella Admin
Service Owner Email:	stella@DeployHub.io
Service Owner Phone:	
PagerDuty Business Service Url:	
PagerDuty Service Url:	
Slack Channel:	
Discord Channel:	https://discord.gg/wM4b5yEFzS
HipChat Channel:	

Component Overview

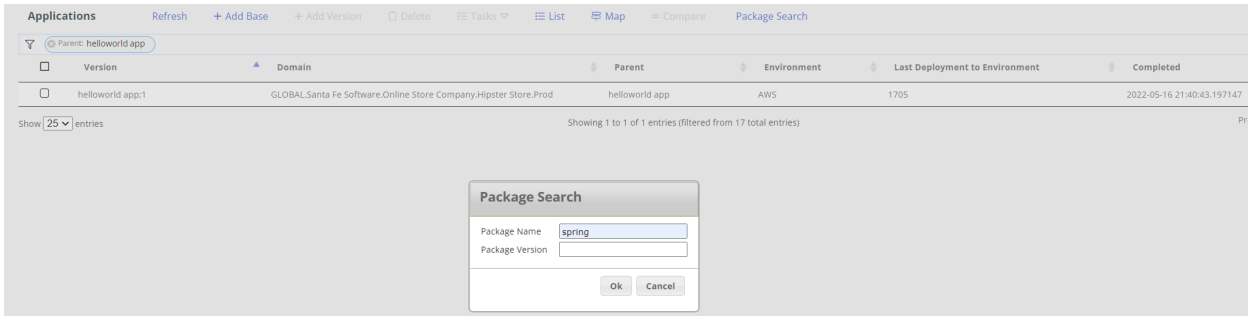
Full Domain:	GLOBAL.Santa Fe Software.Online Store Company
Name:	hello-world:master:v1_0_0_101_g3b3bbdd
Description:	
Component Type:	Container
Endpoint Type:	GLOBAL.Kubernetes
Change Request Data Source:	
Category:	General
Always Deploy:	No
Deploy Sequentially:	No
Custom Action:	

Component Details

Build Date:	Tue Oct 18 11:48:40 2022
Build Id:	101
Build URL:	http://jenkins.myproject.org
Container Registry:	quay.io/ortelius/hello-world
Container Digest:	212d929ca310
Container Tag:	master-v1.0.0.101-g3b3bbdd
Helm Chart:	
Helm Chart Namespace:	
Helm Chart Repo:	
Helm Chart Repo Url:	
Helm Chart Version:	
Git Commit:	3b3bbdd
Git Repo:	dstar55/docker-hello-world-spring-boot
Git Tag:	master
Git URL:	https://github.com/dstar55/docker-hello-world-spring-boot

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.



Package Consumption by Components and Application Versions

Package	Package Version	Component	Application
jersey-spring3	2.23.1	GLOBAL.Santa Fe Software.Online Store Company.Store Services.Recommendation Service;main;v1_2_2_38_gb47fa32	GLOBAL.Santa Fe Software.Online Store Company.Hipster Store;July 4th Sale;1_2_9_1
jersey-spring3	2.23.1	GLOBAL.Santa Fe Software.Online Store Company.Store Services.Recommendation Service.recommendationservice;main;v1_2_2_38_gb47fa32	GLOBAL.Santa Fe Software.Online Store Company.Hipster Store;July 4th Sale;1_2_9_1
spring-aop	5.3.18	GLOBAL.Santa Fe Software.Online Store Company.hello-world;master;v1_0_0_101_g3b3bbdd	GLOBAL.Santa Fe Software.Online Store Company.Hipster Store.Prod.helloworld app;1
spring-aop	4.3.2.RELEASE	GLOBAL.Santa Fe Software.Online Store Company.Store Services.Recommendation Service;main;v1_2_2_38_gb47fa32	GLOBAL.Santa Fe Software.Online Store Company.Hipster Store;July 4th Sale;1_2_9_1
spring-aop	4.3.2.RELEASE	GLOBAL.Santa Fe Software.Online Store Company.Store Services.Recommendation Service.recommendationservice;main;v1_2_2_38_gb47fa32	GLOBAL.Santa Fe Software.Online Store Company.Hipster Store;July 4th Sale;1_2_9_1
spring-aspects	4.3.2.RELEASE	GLOBAL.Santa Fe Software.Online Store Company.Store Services.Recommendation Service;main;v1_2_2_38_gb47fa32	GLOBAL.Santa Fe Software.Online Store Company.Hipster Store;July 4th Sale;1_2_9_1
spring-aspects	4.3.2.RELEASE	GLOBAL.Santa Fe Software.Online Store Company.Store Services.Recommendation Service.recommendationservice;main;v1_2_2_38_gb47fa32	GLOBAL.Santa Fe Software.Online Store Company.Hipster Store;July 4th Sale;1_2_9_1
spring-beans	5.3.18	GLOBAL.Santa Fe Software.Online Store Company.hello-world;master;v1_0_0_101_g3b3bbdd	GLOBAL.Santa Fe Software.Online Store Company.Hipster Store.Prod.helloworld app;1
spring-beans	4.3.2.RELEASE	GLOBAL.Santa Fe Software.Online Store Company.Store Services.Recommendation Service;main;v1_2_2_38_gb47fa32	GLOBAL.Santa Fe Software.Online Store Company.Hipster Store;July 4th Sale;1_2_9_1
spring-beans	4.3.2.RELEASE	GLOBAL.Santa Fe Software.Online Store Company.Store Services.Recommendation Service.recommendationservice;main;v1_2_2_38_gb47fa32	GLOBAL.Santa Fe Software.Online Store Company.Hipster Store;July 4th Sale;1_2_9_1
spring-boot	2.6.6	GLOBAL.Santa Fe Software.Online Store Company.hello-world;master;v1_0_0_101_g3b3bbdd	GLOBAL.Santa Fe Software.Online Store Company.Hipster Store.Prod.helloworld app;1
spring-boot	1.4.0.RELEASE	GLOBAL.Santa Fe Software.Online Store Company.Store Services.Recommendation Service;main;v1_2_2_38_gb47fa32	GLOBAL.Santa Fe Software.Online Store Company.Hipster Store;July 4th Sale;1_2_9_1
spring-boot	1.4.0.RELEASE	GLOBAL.Santa Fe Software.Online Store Company.Store Services.Recommendation Service.recommendationservice;main;v1_2_2_38_gb47fa32	GLOBAL.Santa Fe Software.Online Store Company.Hipster Store;July 4th Sale;1_2_9_1
spring-boot-autoconfigure	2.6.6	GLOBAL.Santa Fe Software.Online Store Company.hello-world;master;v1_0_0_101_g3b3bbdd	GLOBAL.Santa Fe Software.Online Store Company.Hipster Store.Prod.helloworld app;1
spring-boot-autoconfigure	1.4.0.RELEASE	GLOBAL.Santa Fe Software.Online Store Company.Store Services.Recommendation Service;main;v1_2_2_38_gb47fa32	GLOBAL.Santa Fe Software.Online Store Company.Hipster Store;July 4th Sale;1_2_9_1
spring-boot-autoconfigure	1.4.0.RELEASE	GLOBAL.Santa Fe Software.Online Store Company.Store Services.Recommendation Service.recommendationservice;main;v1_2_2_38_gb47fa32	GLOBAL.Santa Fe Software.Online Store Company.Hipster Store;July 4th Sale;1_2_9_1
spring-boot-jarmode-layertools	2.6.6	GLOBAL.Santa Fe Software.Online Store Company.hello-world;master;v1_0_0_101_g3b3bbdd	GLOBAL.Santa Fe Software.Online Store Company.Hipster Store.Prod.helloworld app;1
spring-boot-starter	1.4.0.RELEASE	GLOBAL.Santa Fe Software.Online Store Company.Store Services.Recommendation Service;main;v1_2_2_38_gb47fa32	GLOBAL.Santa Fe Software.Online Store Company.Hipster Store;July 4th Sale;1_2_9_1
spring-boot-starter	1.4.0.RELEASE	GLOBAL.Santa Fe Software.Online Store Company.Store Services.Recommendation Service.recommendationservice;main;v1_2_2_38_gb47fa32	GLOBAL.Santa Fe Software.Online Store Company.Hipster Store;July 4th Sale;1_2_9_1
spring-boot-starter-aop	1.4.0.RELEASE	GLOBAL.Santa Fe Software.Online Store Company.Store Services.Recommendation Service;main;v1_2_2_38_gb47fa32	GLOBAL.Santa Fe Software.Online Store Company.Hipster Store;July 4th Sale;1_2_9_1

Show All entries
Showing 1 to 77 of 77 entries
Previous 1 Next CSV Excel PDF Print

Next Steps

After completing these initial POC steps, you can add additional Components to your Application, update them via your pipeline, and view how Ortelius 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 Ortelius.

Get Help

Report an Issue: https://github.com/ortelius/ortelius/issues

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

Ortelius Documentation: https://docs.ortelius.io/guides/

Get Involved in Open-Source

Help us create the best, open source microservice 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.

CD.FOUNDATION

About the CD Foundation

The Continuous Delivery Foundation (CDF) serves as the vendor-neutral home of many of the fastest-growing projects for continuous integration/continuous delivery (CI/CD). It fosters vendor-neutral collaboration between the industry's top developers, end users and vendors to further CI/CD best practices and industry specifications. Its mission is to grow and sustain projects that are part of the broad and growing continuous delivery ecosystem.

Learn more about the Continuous Delivery Foundation at CD.Foundation