Tenable.io Container Security

Last Updated: April 26, 2018

Table of Contents

Getting Started with Tenable.io Container Security

Last Updated: April 26, 2018

This user guide describes Tenable.io Container Security. Tenable.io Container Security stores and scans container images as the images are built, before production. It provides vulnerability and malware detection, along with continuous monitoring of container images. By integrating with the continuous integration and continuous deployment (CI/CD) systems that build container images, Tenable.io Container Security ensures every container reaching production is secure and compliant with enterprise policy.

> **Tip:** If you are new to Tenable.io Container Security, see the workflow to get started.

Other Tenable.io Products

Tenable.io Vulnerability Management

See the User Guide

Tenable.io Vulnerability Management allows security and audit teams to share multiple Nessus scanners, scan schedules, scan policies, and scan results with an unlimited set of users or groups.

By making multiple resources available for sharing among users and groups, Tenable.io Vulnerability Management provides endless possibilities for creating customized workflows for vulnerability management programs, while accommodating the numerous regulatory or compliance drivers that demand you keep your business secure.

Tenable.io Vulnerability Management can schedule scans, push policies, view scan findings, and control multiple Nessus scanners from the cloud. This enables the deployment of Nessus scanners throughout networks to both public clouds, private clouds, and physical locations.

Tenable.io Web Application Scanning

See the User Guide

Tenable.io Web Application Scanning offers significant improvements over the existing **Web Application Tests** policy template provided by the Nessus scanner which is incompatible with modern web applications that rely on Javascript and are built on HTML5. This leaves you with an incomplete understanding of your web application security posture.

Tenable.io Web Application Scanning provides comprehensive vulnerability scanning for modern web applications. Tenable.io Web Application Scanning has accurate vulnerability coverage that minimizes

false positives and false negatives, ensuring that security teams understand the true security risks in their web applications. The product offers safe external scanning that ensures production web applications are not disrupted or delayed, including those built using HTML5 and AJAX frameworks.

Tenable.io Container Security Workflow

Get Started with Tenable.io Container Security

1. Review container terminology.
2. Activate your account and log in to the web portal.
3. Generate Access and Secret keys for the Tenable.io API.
4. Push a container image from Docker Hub to Tenable.io Container Security.
5. Import container images.
6. Pull a container image from Tenable.io Container Security's built in container registry.

Push a Container Image to Tenable.io Container Security

> **Note:** The first step of this procedure assumes your docker image is hosted at Docker Hub. However, there are different ways to create docker images, and your organization may host its own docker registry. You should obtain the docker image in the way most appropriate for your organization.

1. Download the image (for example, **`rabbitmq`**) from Docker Hub:

```
$ sudo docker pull rabbitmq:latest
latest: Pulling from library/rabbitmq

Digest: sha256:6499945e41389896bc3a304698f8b8d72668e2f5904b11d133937b1aa810936a
Status: Image is up to date for rabbitmq:latest
$
```

2. Use the `docker login` command with your Tenable.io API Access and Secret keys to authenticate with Tenable.io Container Security:

```
$ sudo docker login -u $TENABLE_IO_ACCESS_KEY -p $TENABLE_IO_SECRET_KEY
registry.cloud.tenable.com

WARNING: login credentials saved in /home/user/.docker/config.json
Login Succeeded
$
```

3. Tag the container image:

```
$ sudo docker tag rabbitmq:latest
registry.cloud.tenable.com/rabbitmq/rabbitmq:latest
$
```

4. Push the tagged image to Tenable.io Container Security:

```
$ sudo docker push registry.cloud.tenable.com/rabbitmq/rabbitmq:latest

The push refers to a repository [registry.cloud.tenable.com/rabbitmq/rabbitmq]
(len: 1)
de0397c6625e: Pushed
f8016aaf3b9f: Pushed
83758cb91f4f: Pushed
c9d5389102ad: Pushed
a5e498845191: Pushed
1683aa1aee8b: Pushed
e1253242cc77: Pushed
b102fbc1499f: Pushed
36a89f2fe626: Pushed
ca89d4e16f6d: Pushed
3318a6a8de2c: Pushed
6dcbaff01d16: Pushed
7d39e0a61f2e: Pushed
523ef1d23f22: Pushed
latest: digest:
sha256:834c17047b310836c64b903a33b0757e581124b56cf367a51910dcf4a789c9dc size:
35888
$
```

Now you can view the report on the **Scan Results** page in Tenable.io Container Security.

Glossary of Terms

The following terms are used throughout Tenable.io Container Security product documentation.

Term	Description
CD System	A Continuous Deployment system - typically used to monitor for successful builds that have passed tests, and to take those successful builds and push them to production environments, thus automating the deployment of the successful builds.
CI System	A Continuous Integration system - typically used to monitor source control commits, such as merged pull requests in GitHub, to automatically trigger a build (to test) as the change in source control is detected.
CI/CD System	A Continuous Integration and Continuous Deployment system - typically used to monitor source control commits, such as merged pull requests in GitHub, to automatically trigger a build (to test) as the change in source control is detected, and upon successful completion of the build and test phase, to take those successful builds and push them to production environments, thus automating the deployment of the successful build.
Container	A running instance of a container image. A container image that has been started or otherwise executed.
Container Image	An application hosted inside of a container image file (e.g., ubuntu:14.04).
Container Image Tag	A specific release or version of an application hosted inside of a container (e.g., 14.04).
Container Registry	A storage location for Container Images. Provides developers and continuous integration systems the ability to store containers that are pushed.
Continuous Deployment	A development practice where operations (or DevOps) automatically push successfully tested builds to production environments, making them immediately available.
Continuous Integration	A development practice where developers integrate code into a shared source control repository, regularly, as changes are made.
Image	An application hosted inside of a container image file (e.g., ubuntu:14.04).
Image Tag	A specific release or version of an application hosted inside of a container (e.g.,

Term	Description
	14.04).
Organization Admin	The role assigned to the first user registering for Tenable.io Container Security, at the time the Organization is created. If you have registered without an invitation, you were automatically assigned the role of Organization Admin and a new Organization was created for your account.
Registry	A storage location for Container Images. Provides developers and continuous integration systems the ability to store containers that are pushed.
Repository	A storage location or namespace, within the registry, for an image (e.g., /org/tenable_io_container_security/approved/).
Tag	A specific release or version of an application hosted inside of a container (e.g., 14.04).
User	The role assigned to invited users registering for Tenable.io Container Security, for pre-existing Organizations. If you have registered via an invitation, you were automatically assigned the role of User and you were added to the same Organization of the user who invited you.

Log in to Tenable.io Container Security

Log in to the Web Console

1. In a browser, access https://cloud.tenable.com/app.html.

   The Tenable.io login page appears.

2. Type the user name and password that you defined during registration.

3. To remain logged in until you click **Log Out** or close the browser, select the **Remember Me** check box. Otherwise, you will be logged out after a period of inactivity.

4. Click **Log In**.

   Tenable.io Container Security **Dashboard** page appears.

   -or-

   If you have a Tenable.io Vulnerability Management subscription, the Tenable.io Vulnerability Management **Dashboards** page appears. To access Tenable.io Container Security, in the top navigation bar, click **Vulnerability Management > Container Security**.

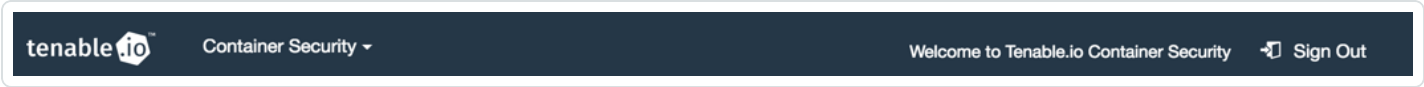
Log in Via Docker

You can log in to Tenable.io Container Security using the **`docker login`** command:

```
$ docker login -u $TENABLE_IO_ACCESS_KEY -p $TENABLE_IO_SECRET_KEY
registry.cloud.tenable.com
WARNING: login credentials saved in /home/user/.docker/config.json
Login Succeeded
```

Navigation

The top navigation bar displays a toggle to switch between Tenable.io Container Security and Tenable.io Vulnerability Management.



The left navigation bar displays the main pages available in Tenable.io Container Security: **Dashboard**, **Repositories**, **Scan Results**, **Policies**, and **Usage**. All of the Tenable.io Container Security primary tasks can be performed using these pages. Click a page name to open the corresponding page.

The default landing page in Tenable.io Container Security is the **Dashboard** page. This page displays usage and security metrics for the container images hosted in your registry. If some metrics are not available, you need to push a container image to the registry. If you are new to Tenable.io Container Security, see the workflow for additional steps.

How To

This section describes how to perform the following primary functions of Tenable.io Container Security:

- Pull from the Registry
- Push to the Registry
- Import Container Images
- Add a Repository
- Modify a Repository
- Delete a Repository
- Add a Policy
- Modify a Policy
- Delete a Policy

Push to the Registry

Steps

1. Download a container image from Docker Hub. In this example, the image is **`rabbitmq:latest`**.

```
docker pull rabbitmq:latest
latest: Pulling from library/rabbitmq
a6a21504c62e: Pull complete
bf055db89267: Pull complete
dbd43ef2ea53: Pull complete
600a59e46bd7: Pull complete
7232626642cd: Pull complete
cc290cb80573: Pull complete
f997bab46fd6: Pull complete
641325f6925e: Pull complete
58e1b05f4459: Pull complete
2dec2d1eea4e: Pull complete
847202ab44fc: Pull complete
66e2c8882002: Pull complete
fbcaa7c864ad: Pull complete
99bc733dbe0a: Pull complete
63c04263ad5e: Pull complete
61e945908a53: Pull complete
a90ffec32ce7: Pull complete
3001d59cf838: Pull complete
be1fddfeb5f6: Pull complete
Digest: sha256:bb54a2ca1c5e390d1cea32748cc2c9cf927e05740d4962bf889c9e62bcdd3788
Status: Downloaded newer image for rabbitmq:latest
```

2. Log in to Tenable.io Container Security using **`docker login`**:

```
$ docker login -u $TENABLE_IO_ACCESS_KEY -p $TENABLE_IO_SECRET_KEY
registry.cloud.tenable.com
WARNING: login credentials saved in /home/user/.docker/config.json
Login Succeeded
```

3. Get the IMAGE ID using the **`docker images`** command:

```
$ docker images|grep rabbitmq|grep latest
rabbitmq     305 MB            latest          be1fddfeb5f6        2 days ago
```

The results are in the following format:

```
REPOSITORY   VIRTUAL SIZE      TAG             IMAGE ID            CREATED
```

4. Tag the image using **`docker tag:`**

```
$ docker tag be1fddfeb5f6 registry.cloud.tenable.com/rabbitmq/rabbitmq
```

> **Tip:** By default. **`docker push`** pushes an image to Docker's Central Registry. When you tag an image, then you can use **`docker push`** to push the image to the Tenable.io Container Security registry.

5. Push the image to Tenable.io Container Security using **`docker push`**:

```
$ docker push registry.cloud.tenable.com/rabbitmq/rabbitmq
The push refers to a repository [registry.cloud.tenable.com/rabbitmq/rabbitmq]
(len: 1)
be1fddfeb5f6: Image already exists
61e945908a53: Pushed
63c04263ad5e: Pushed
fbcaa7c864ad: Pushed
847202ab44fc: Pushed
58e1b05f4459: Pushed
641325f6925e: Pushed
cc290cb80573: Pushed
7232626642cd: Pushed
600a59e46bd7: Pushed
dbd43ef2ea53: Pushed
bf055db89267: Pushed
a6a21504c62e: Pushed
9ee13ca3b908: Pushed
latest: digest:
sha256:aa6f181d836ce0e91dfeaf08ca671b4e997926919f141c10b2abd37c2af42fe1 size:
35888
```

Now **`rabbitmq:latest`** is stored in your private instance of Tenable.io Container Security.

Pull from the Registry

To pull a container image with the name **rabbitmq/rabbitmq:latest**, pull the image using the following command:

```
$ docker login -u $TENABLE_IO_ACCESS_KEY -p $TENABLE_IO_SECRET_KEY
registry.cloud.tenable.com
WARNING: login credentials saved in /home/user/.docker/config.json
Login Succeeded
$ docker pull registry.cloud.tenable.com/rabbitmq/rabbitmq:latest
Using default tag: latest
latest: Pulling from rabbitmq/rabbitmq
Digest: sha256:aa6f181d836ce0e91dfeaf08ca671b4e997926919f141c10b2abd37c2af42fe1
Status: Image is up to date for registry.cloud.tenable.com/rabbitmq/rabbitmq:latest
```

Import Container Images

Note: Tenable.io Container Security does not support registry import from Docker Hub.

Import an Existing Container Image

These steps assume a container registry exists in your organization.

1. In Tenable.io Container Security, in the left navigation bar, in the **Quick Start** section, click **Import Docker Images**.

   The **Import** page appears.

2. In the **Source** box, select the type of container registry your organization uses.

   Note: If you use a container registry that is not listed, contact Tenable, Inc. at support@tenable.com to let us know that you would like your container registry to be officially supported. In the meantime, select **Docker Registry**, since it is a generic placeholder for a variety of container registries.

3. Type your connection details in the **IP Address**, **Port**, **Username**, and **Password** boxes.

4. Select or clear the **Requires SSL** check box as necessary.

   Ensure the account you choose has the necessary permissions to pull the container images out of the registry.

5. Click **Import**.

   The container images appear in Tenable.io Container Security with scan results.

Import an AWS Elastic Container Registry

These steps assume your container image exists in an Amazon Web Services (AWS) Elastic Container Registry (ECR) deployment hosted within the AWS Elastic Container Service (ECS).

1. In Tenable.io Container Security, in the left navigation bar, in the **Quick Start** section, click **Import Docker Images**.

   The **Import** page appears.

2. In the **Source** box, select **AWS Elastic Container Registry**.

3. In the **IP Address** box, type the fully-qualified domain name of your ECR deployment (e.g., **`579133718396.dkr.ecr.us-east-2.amazonaws.com`**).
4. In the **Port** box, type *443*.
5. In the **Username** box, type *AWS*.
6. In the **Password** box, type the base64-encoded password used in the **`docker login`** command, which is generated by AWS CLI.

   > **Tip:** If your ECR is in the us-east-2 region, you can run the **`aws ecr get-login --region us-east-2`** command to get the **`docker login`** command.

7. Select the **Requires SSL** check box.
8. Click **Import**.

   The container images appear in Tenable.io Container Security with scan results.

Add a Repository

Steps

1. In Tenable.io Container Security, in the left navigation bar, in the **Workbench** section, click **Repositories**.

   The **Repositories** page appears.

2. Click the **Add Repository** button.

   The **Add Repository** page appears.

3. In the **Repository Name** box, type a name for the repository.

   > **Note:** Repository names may consist of lowercase characters, numbers, and separators. Acceptable separators are periods, one or two underscores, and minus signs. A name cannot begin or end with a separator.

4. Configure the options, then click **Create Repository**.

   The new repository appears in the list of repositories on the **Repositories** page.

Modify a Repository

Steps

1. In Tenable.io Container Security, in the left navigation bar, in the **Workbench** section, click **Repositories**.

   The **Repositories** page appears.

2. Select the repository that you want to modify.

   The **Container Images** page appears.

3. Click **Edit Repository**.

   The **Edit Repository** page appears.

4. Modify the settings as necessary, and click **Update**.

   Your changes are saved.

   > **Note:** Repository names may consist of lowercase characters, numbers, and separators. Acceptable separators are periods, one or two underscores, and minus signs. A name cannot begin or end with a separator.

Delete a Repository

Steps

1. In Tenable.io Container Security, in the left navigation bar, in the **Workbench** section, click **Repositories**.

   The **Repositories** page appears.

2. Select the repository that you want to delete.

   The **Container Images** page appears.

3. Click the **Remove this repository** button.

   A dialog box appears, confirming your selection to delete the repository.

4. Click the **Yes, remove it!** button.

   The repository is deleted.

Add a Policy

Steps

1. In Tenable.io Container Security, in the left navigation bar, under **Workbench**, click **Policies**.

   The **Policies** page appears.

2. Click the **Add Policy** button.

   The **Add Policy** page appears.

3. Configure the options as necessary, then click **Create Policy**.

   The new policy appears in the list of policies on the **Policies** page.

Modify a Policy

Steps

1. In Tenable.io Container Security, in the left navigation bar, under **Workbench**, click **Policies**.

   The **Policies** page appears.

2. Select the policy that you want to modify.

   The **Edit Policy** page appears.

3. Modify the settings as necessary, and click **Update**.

   Your changes are saved.

Delete a Policy

Steps

1. In Tenable.io Container Security, on the left bar, in the **Workbench** section, click **Policies**.

   The **Policies** page appears.

2. Click the policy that you want to delete.

   The **Edit Policy** page appears.

3. Click the **Remove this policy** button.

   A dialog box appears, confirming your selection to delete the policy.

4. Click the **Yes, remove it!** button.

   The policy is deleted.

Integration Guides

This section describes how Tenable.io Container Security integrates with the following major CI/CD systems:

- Bamboo
- CircleCI
- Codeship
- Distelli
- Drone.io
- Jenkins
- Shippable
- Solano Labs
- Travis CI
- Wercker

Bamboo

Before You Begin

These instructions describe how to push a Docker image from Bamboo to Tenable.io Container Security.

These steps assume you are already comfortable using Bamboo and are already pushing Docker images to a public or private registry. If you are already using Bamboo, but have not built Docker container images, familiarize yourself with the Bamboo documentation Configuring the Docker task in Bamboo.

Steps

1. Create a new Docker task for the relevant job.
2. In the **Task** box, type a description for the task.
3. Depending on whether you want the task to run, select or clear the **Disable this task** check box.
4. Select **Push a Docker image to a Docker registry command** and complete the settings.

   Bamboo builds are sent to Tenable.io Container Security for storage, distribution, vulnerability scanning, and malicious code scanning.

CircleCI

Before You Begin

These instructions describe how to push a Docker image from CircleCI to Tenable.io Container Security.

These steps assume you are already comfortable using CircleCI and are already pushing Docker images to a public or private registry. If you are already using CircleCI, but have not built Docker container images, familiarize yourself with the CircleCI documentation Continuous Integration and Delivery with Docker.

Click here for information about the `circle.yml` file.

If you are using CircleCI to build Docker container images, you should have a `circle.yml` file in your project source control repository that looks similar to the following example:

```
machine:
services:
- docker

dependencies:
override:
- docker info
- docker build -t circleci/elasticsearch .

test:
override:
- docker run -d -p 9200:9200 circleci/elasticsearch; sleep 10
- curl --retry 10 --retry-delay 5 -v http://localhost:9200

deployment:
hub:
branch: master
commands:
- docker push circleci/elasticsearch
```

The following lines in `circle.yml` instruct CircleCI to leverage Docker for the build process:

```
machine:
```

```
services:
- docker
```

The following lines in `circle.yml` instruct CircleCI to build the elasticsearch image in the `circleci/` repository:

```
dependencies:
override:
- docker info
- docker build -t circleci/elasticsearch .
```

The following are the most important lines for adding Tenable.io Container Security integration to CircleCI environments. These lines instruct CircleCI to use Docker to log in to the registry (in this case to Docker Hub, since no private registry is specified) and push `cicleci/elasticsearch` to the registry:

```
deployment:
hub:
branch: master
commands:
- docker login -u $DOCKER_USER -p $DOCKER_PASS
- docker push circleci/elasticsearch
```

Steps

1. To add environment variables for the project in the CircleCI console, open the project, click **Project Settings**, then click **Environment Variables**.
2. Define the following variables:

Variable	Description
TENABLE_IO_ CONTAINER_ SECURITY_EMAIL	The email that you use to log in to Tenable.io Container Security.
TENABLE_IO_ CONTAINER_	The user name that you use to log in to Tenable.io Container Security. You can find this on the **Settings** page in Tenable.io Container

Variable	Description
SECURITY_USER	Security.
TENABLE_IO_CONTAINER_SECURITY_ENDPOINT	For hosted cloud users of Tenable.io Container Security, this value is `registry.cloud.tenable.com`.

3. To add support for Tenable.io Container Security, update the `circle.yml` file as follows:

```
machine:
environment:
VERSION: 2.1.1
TAG: ${VERSION}
services:
- docker

dependencies:
override:
- docker info
- docker version
- docker build -t $TENABLE_IO_CONTAINER_SECURITY_
ENDPOINT/circleci/elasticsearch .

test:
override:
- docker run -d -p 9200:9200 $TENABLE_IO_CONTAINER_SECURITY_
ENDPOINT/circleci/elasticsearch; sleep 10
- curl --retry 10 --retry-delay 5 -v registry.cloud.tenable.com

deployment:
hub:
branch: master
commands:
- docker login -u $TENABLE_IO_ACCESS_KEY -p $TENABLE_IO_SECRET_KEY
- docker tag $TENABLE_IO_CONTAINER_SECURITY_ENDPOINT/circleci/elasticsearch
$TENABLE_IO_CONTAINER_SECURITY_ENDPOINT/circleci/elasticsearch:${TAG}
- docker push $TENABLE_IO_CONTAINER_SECURITY_
ENDPOINT/circleci/elasticsearch:${TAG}
- docker logout
```

CircleCI builds are sent to Tenable.io Container Security for storage, distribution, vulnerability scanning, and malicious code scanning.

Codeship

Before You Begin

These instructions describe how to push a Docker image from Codeship to Tenable.io Container Security.

These steps assume you are already comfortable using Codeship and are already pushing Docker images to a public or private registry. If you are already using Codeship, but have not built Docker container images, familiarize yourself with the Codeship documentation Pushing to a remote registry.

Steps

1. Edit the **`codeship-services.yml`** file to use the repository name and image name specified in Tenable.io Container Security.

   ```
   app:
   build:
   image: repository_name/image_name
   dockerfile_path: Dockerfile
   ```

   Note: If this is the first time you are pushing an image into the repository, there is not a pre-configured image name. The image name is added automatically after the push from Codeship.

2. Edit the service section of the the **`codeship-steps.yml`** file to look similar to the following example:

   ```
   service:
   app type: push
   image_name: repository_name/image_name
   registry: registry.cloud.tenable.com
   encrypted_dockercfg_path: dockercfg.encrypted
   ```

   Codeship builds are sent to Tenable.io Container Security for storage, distribution, vulnerability scanning, and malicious code scanning.

Distelli

Before You Begin

These instructions describe how to push a Docker image from Distelli to Tenable.io Container Security using the Distelli WebUI Manifest.

These steps assume you are already comfortable using Distelli and are already pushing Docker images to a public or private registry. If you are already using Distelli, but have not built Docker container images, familiarize yourself with the Distelli documentation on the Distelli Manifest. You can use the Distelli manifest file by either using the Distelli WebUI Manifest, or by editing the **`distelli-manifest.yml`** file directly.

Steps

1. Log in to Distelli and navigate to an application.
2. Click the **Manifest** tab.

   The **Build** section displays content similar to the following example:

   ```
   docker build --quiet=false -t $DOCKER_REPO:$DISTELLI_BUILDNUM .
   docker login -u $DOCKER_USERNAME -p $DOCKER_PW
   docker push $DOCKER_REPO:$DISTELLI_BUILDNUM
   ```

3. To add support for Tenable.io Container Security, modify the **Build** section to look like the following example:

   ```
   bash docker build --quiet=false -t $TENABLE_IO_CONTAINER_SECURITY_
   REPO:$DISTELLI_BUILDNUM . docker login -u $TENABLE_IO_ACCESS_KEY -p $TENABLE_
   IO_SECRET_KEY registry.cloud.tenable.com docker push $TENABLE_IO_CONTAINER_
   SECURITY_REPO:$DISTELLI_BUILDNUM
   ```

   This modification adds the Tenable.io Container Security URI to docker login.

   Distelli builds are sent to Tenable.io Container Security for storage, distribution, vulnerability scanning, and malicious code scanning.

Drone.io

Before You Begin

These instructions describe how to push a Docker image from Drone.io to Tenable.io Container Security.

These steps assume you are already comfortable using Drone.io and are already pushing Docker images to a public or private registry. If you are already using Drone.io, but have not built Docker container images, familiarize yourself with the Drone.io documentation How to build and publish Docker images.

If you use Drone.io to build Docker container images, you should already have a build script (usually a **`build.sh`** file) that looks like the following:

```
$ docker build -t docker-registry/image-name .
$ docker push docker-registry/image-name
```

Steps

1. Open the **`build.sh`** file.
2. Append a docker login directive before the docker push directive in the script, as in the following example:

   ```
   $ docker build -t docker-registry/image-name .
   $ docker login -u $TENABLE_IO_ACCESS_KEY -p $TENABLE_IO_SECRET_KEY
   registry.cloud.tenable.com
   $ docker push docker-registry/image-name
   ```

   Drone.io builds for this project are sent to Tenable.io Container Security for storage, distribution, vulnerability scanning, and malicious code scanning.

Jenkins

Before You Begin

These instructions describe how to push a Docker image from Jenkins to Tenable.io Container Security.

These steps assume you are already comfortable using Jenkins and are already pushing Docker images to a public or private registry. If you are already using Jenkins, but have not built Docker container images, familiarize yourself with the documentation for the Jenkins CloudBees Docker Build and Publish plugin.

Click here for instructions on how to install the CloudBees Docker Build and Publish plugin.

1. Log in to Jenkins.
2. Click **Manage Jenkins**, then click **Manage Plugins**.
3. Click **Installed**.

   A list of installed plugins appears.
4. Click **Available**.
5. In the **Filter** box, type **CloudBees Docker Build and Publish plugin**.
6. Select the check box that corresponds to the plugin.
7. Install the plugin.

   The CloudBees Docker Build and Publish plugin is installed and ready for use by Jenkins jobs.

Steps

1. On the Jenkins dashboard, select the job you want to modify.
2. Click **Configure**.
3. In the **Build** section, click **Add build step**.
4. In the drop down box, select **Docker Build and Publish**.

5. Type the details for the following configuration parameters:

   - **Repository Name:** The repository name and image name. For example, if you build a rabbitmq container image, you can name the repository rabbitmq and the image rabbitmq. In this example, in the **Repository Name** box, type *rabbitmq/rabbitmq*.
   - **Tag:** The tag name. The simplest tag name to use is *latest*.
   - **Docker Host URI:** The Jenkins path to the Docker Host. If the Docker Host is running on localhost, then in the **Docker Host URI** box, type *tcp://127.0.0.1:4243*.
   - **Docker registry URL:** The Tenable.io Container Security API endpoint, which in this case is *registry.cloud.tenable.com*.
   - **Registry credentials:** The registry credentials that you select from the box.

     ### Adding registry credentials

     1. Click **Add**.
     2. Click **Username with password**.
     3. In the **Username** box, type your Tenable.io Container Security user name.
     4. In the **Password** box, type your Tenable.io Container Security password.
     5. Click **Add**.

        The credentials are added.

6. Click **Save**.

   Jenkins builds are sent to Tenable.io Container Security for storage, distribution, vulnerability scanning, and malicious code scanning.

Shippable

Before You Begin

These instructions describe how to push a Docker image from Shippable to Tenable.io Container Security.

These steps assume you are already comfortable using Shippable and are already pushing Docker images to a public or private registry. If you are already using Shippable, but have not built Docker container images, familiarize yourself with the Shippable documentation Building a Docker image.

Steps

1. Log in to Shippable.
2. In the upper right corner of the screen, click the **Account Settings** button.
3. Click **Integrations**, and then click **Add Integration**.
4. In the **Master Integration** section, click **Private Docker Registry**.
5. In the **Name** box, type **Tenable.io Container Security**.
6. In the **URL** box, type **registry.cloud.tenable.com**.
7. In the **Username** box, type your Tenable.io Container Security user name.
8. In the **Password** box, type your Tenable.io Container Security password.
9. In the **Email** box, type the email address associated with your Tenable.io Container Security account.
10. Click **Save**.

    Your Tenable.io Container Security account is now available for hosting container images built by Shippable.
11. Access your project page, and click **Settings**.
12. Click **Hub**, and select the Tenable.io Container Security integration that you just created.
13. In the **Push Build** field, click **Yes**.

14. In the **Push image to** box, type the name of your repository and image in Tenable.io Container Security (e.g., `testrepo/nodejs`).
15. In the **Push Image Tag** box, select from the following options: **default**, **commitsha**, or **latest**.
16. Click **Save**.

    Shippable builds are sent to Tenable.io Container Security for storage, distribution, vulnerability scanning, and malicious code scanning.

Solano Labs

Before You Begin

These instructions describe how to push a Docker image from Solano Labs to Tenable.io Container Security.

These steps assume you are already comfortable using Solano Labs and are already pushing Docker images to a public or private registry. If you are already using Solano Labs, but have not built Docker container images, familiarize yourself with the Solano Labs documentation.

Note: Solano Labs support for building Docker container images is in private beta. For customers interested in participating, Solano Labs recommends contacting Solano Labs support.

Steps

1. Open the **`solano.yml`** file, which should look similar to the following example:

```
# Use docker-enabled workers (currently private beta - contact
support@solanolabs.com)
system:
docker: true
python:
python_version: 2.7
hooks:
pre_setup: |
set -ex
sudo apt-get update -qq
sudo docker pull jenkins
sudo docker build -t myrepo/jenkins-dsl-ready:my .
tests:
- python -m doctest build/resolve_jenkins_plugins_dependencies.py
```

2. Add a `post_build` phase with your Tenable.io Container Security user name.

```
# Use docker-enabled workers (currently private beta - contact
support@solanolabs.com)
system:
docker: true
```

```
python:
python_version: 2.7
hooks:
pre_setup: |
set -ex
sudo apt-get update -qq
sudo docker pull jenkins
sudo docker build -t myrepo/jenkins-dsl-ready .
post_build: |
docker login -u $TENABLE_IO_ACCESS_KEY -p $TENABLE_IO_SECRET_KEY
registry.cloud.tenable.com
docker push myrepo/jenkins-dsl-ready
tests:
- python -m doctest build/resolve_jenkins_plugins_dependencies.py
```

Solano Labs builds are sent to Tenable.io Container Security for storage, distribution, vulnerability scanning, and malicious code scanning.

Travis CI

Before You Begin

These instructions describe how to push a Docker image from Travis CI to Tenable.io Container Security.

These steps assume you are already comfortable using Travis CI and are already pushing Docker images to a public or private registry. If you are already using Travis CI, but have not built Docker container images, familiarize yourself with the Travis CI documentation Using Docker in Builds.

Click here for information about the **travis.yml** file.

If you are using Travis CI to build Docker container images, you should have a **travis.yml** file in your project source control repository that looks similar to:

```
sudo: required
language: ruby
services:
- docker
before_install:
- docker build -t carlad/sinatra .
- docker run -d -p 127.0.0.1:80:4567 carlad/sinatra /bin/sh -c "cd /root/sinatra;
bundle exec foreman start;"
- docker ps -a
- docker run carlad/sinatra /bin/sh -c "cd /root/sinatra; bundle exec rake test"
script:
- bundle exec rake test
```

The following lines in **travis.yml** instruct Travis CI to leverage Docker for the build process:

```
sudo: required
services:
- docker
```

The following lines in **travis.yml** instruct Travis CI to build the sinatra image in the carlad/ repository:

```
before_install:
```

```
- docker build -t carlad/sinatra .
```

Steps

1. Open the **`travis.yml`** file.
2. Add your Tenable.io Container Security credentials.

   ```
   $ travis encrypt TENABLE_IO_CONTAINER_SECURITY_EMAIL=email@organization.com
   $ travis encrypt TENABLE_IO_CONTAINER_SECURITY_USER=username
   $ travis encrypt TENABLE_IO_CONTAINER_SECURITY_PASSWORD=password
   ```

3. Add your environment variables.

   ```
   env:
   global:
   - secure: "UkF2CHX0lUZ...VI/LE=" # TENABLE_IO_CONTAINER_SECURITY_EMAIL
   - secure: "Z3fdBNPt5hR...VI/LE=" # TENABLE_IO_CONTAINER_SECURITY_USER
   - secure: "F4XbD6WybHC...VI/LE=" # TENABLE_IO_CONTAINER_SECURITY_PASSWORD
   - COMMIT=${TRAVIS_COMMIT::8}
   ```

4. Add your connection information.

   ```
   after_success:
   - docker login -u $TENABLE_IO_ACCESS_KEY -p $TENABLE_IO_SECRET_KEY
   registry.cloud.tenable.com
   - export REPO=sebestblog/travis-demo
   - export TAG=`if [ "$TRAVIS_BRANCH" == "master" ]; then echo "latest"; else
   echo $TRAVIS_BRANCH ; fi`
   - docker build -f Dockerfile -t $REPO:$COMMIT .
   - docker tag $REPO:$COMMIT $REPO:$TAG
   - docker tag $REPO:$COMMIT $REPO:travis-$TRAVIS_BUILD_NUMBER
   - docker push $REPO
   ```

   Travis CI builds are sent to Tenable.io Container Security for storage, distribution, vulnerability scanning, and malicious code scanning.

Wercker

Before You Begin

These instructions describe how to push a Docker image from Wercker to Tenable.io Container Security.

These steps assume you are already comfortable using Wercker and are already pushing Docker images to a public or private registry. If you are already using Wercker, but have not built Docker container images, familiarize yourself with the Wercker documentation Containers.

Steps

1. In your project source control repository, open the **`wercker.yml`** file.
2. Add support for Tenable.io Container Security by changing the deploy directive as follows:

```
deploy:
steps:
- internal/docker-push:
username: $USERNAME
password: $PASSWORD
tag: my-amazing-tag
repository: turing/bar
registry: registry.cloud.tenable.com
```

Wercker builds for this project are sent to Tenable.io Container Security for storage, distribution, vulnerability scanning, and malicious code scanning.

Troubleshooting

Scan Result Missing

If you have successfully created a new repository and pushed a container image, you may want to view the scan result for the container image or for one or more of its layers. If you cannot find the report you want to view, sort by date/timestamp to see the most recent reports made available to you.

If you still cannot find the report you want to view, it could be because program analysis of (potentially) large container images (and layers) can take time. Tenable.io Container Security is the fastest container image scanning service, but it is possible to look for a report faster than it is made available. On average, Tenable.io Container Security analyzes new container images in approximately 30 seconds. If a few minutes have already passed, ensure that the image was pushed correctly by checking the **Repositories** page and refreshing the **Scan Results** page. If you still have trouble, contact support@tenable.com.