Anchore Container Image Scanner Plugin

Plugin Information
View Anchore Container Image Scanner on the plugin site for more information.

Older versions of this plugin may not be safe to use. Please review the following warnings before using an older version:

- Password stored in plain text

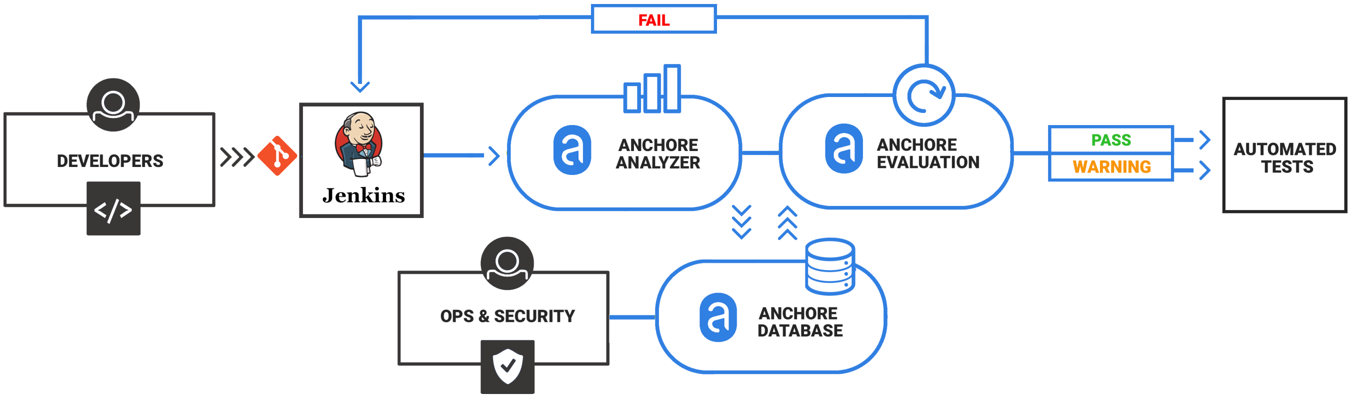
Description

Anchore is a container inspection and analytics platform that enables operators to analyze, inspect, perform security scans, and evaluate custom policies against container images. The Anchore plugin can be used in a Pipeline job or added as a build step to a Freestyle job to automate the process of running an anchore analysis, evaluating custom anchore policies against images, and performing image anchore security scans.

Anchore Jenkins Plugin

Anchore has been designed to plug seamlessly into the CI/CD workflow. A developer commits code into the source code management system. This change triggers Jenkins to start a build which creates a container image.

In the typical workflow this container image is then run through automated testing. If an image does not meet your organization’s requirements for security or compliance then it makes little sense to invest the time required to perform automated tests on the image, it would be better to “learn fast” by failing the build and returning the appropriate reports back to the developer to allow the issue to be addressed.



Plugin Modes

Before getting started, one of two plugin operating 'modes' must be selected. Selecting a mode will configure the plugin to either use a direct anchore scanner (which must be installed on each Jenkins worker node), or a second mode which configures the plugin to interact with the Anchore Engine service API (where the Anchore engine has been installed with its service API accessible from the worker nodes). Depending on which mode you select, the initial configuration/usage will differ.

Anchore Engine Mode (recommended)

In this mode, the usage model generally conforms to the following flow:

1. A Jenkins job will build a container image, and push the image to a registry that is pre-configured in the anchore engine service (see pre-requisites below)
2. The anchore build step will interact with the anchore engine by 'adding' the image (which instructs the anchore engine to pull the image from the registry), and then performing a policy evaluation check on the image. The build step can optionally be configured to fail the build if the policy evaluation results in a 'STOP' action.
3. the plugin will store the resulting policy evaluation results with the job, for later inspection/review

The plugin can be used in Freestyle and Pipeline jobs.

Pre-Requisites:

1. The anchore engine service must be installed within your environment, with its service API being accessible from all Jenkins workers. See https://github.com/anchore/anchore-engine to get started
2. A docker registry must exist and be configured within anchore engine, as the plugin will be instructing the anchore engine to pull images from a registry in this mode.
3. All authentication credentials/anchore engine API endpoint information must be available as input to the plugin at configuration time.

Anchore Local Mode (deprecated)

Local mode is deprecated and is included for backwards compatibility. Future releases of the Anchore Jenkins plugin will remove support for Local Mode. All new deployments should use *Engine Mode.* The following installation instructions covers the installation and configuration of the Anchore Jenkins plugin in Engine mode.

In this mode, the usage model generally conforms to the following flow:

1. A Jenkins job will build a container image on some worker node, and the name of the locally available image is written to a file (default anchore_images)
2. The anchore build step will read the images from the anchore_images file and perform analysis/policy evaluation, by calling out to a locally running container with anchore pre-installed (see pre-requisites in the guide linked below). The build step can optionally be configured to fail the build if the policy evaluation results in a 'STOP' action.
3. The plugin will store the resulting policy evaluation and image query results with the job, for later inspection/review

Pre-Requisites:

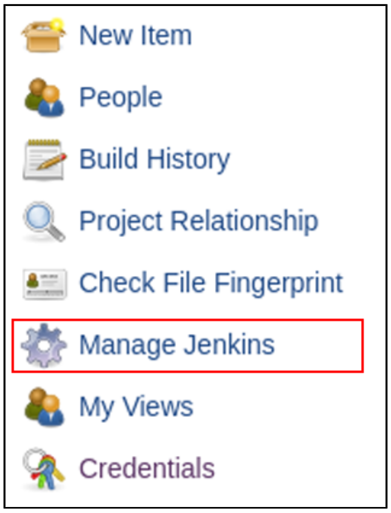
1. Jenkins installed and configured either as a single system, or with multiple configured jenkins worker nodes
2. Each host on which jenkins jobs will run must have docker installed and the jenkins user (or whichever user you have configured jenkins to run jobs as) must be allowed to interact with docker (either directly or via sudo)
3. Each host on which jenkins jobs will run must have the latest anchore container image installed in the local docker host. To install, run 'docker pull anchore/jenkins:latest' on each jenkins host to make the image available to the plugin. The plugin will start an instance of the anchore/jenkins:latest docker container named 'jenkins_anchore' by default, on each host that runs a jenkins job that includes an anchore container image scanner build step.

Once installed, one of two 'modes' must be selected (Manage Jenkins -> Configure System -> Anchore Plugin Mode), which will configure the plugin to either use a direct anchore scanner (which must be installed on each Jenkins worker node), or a second mode which configures the plugin to interact with the anchore engine service API (where the anchore engine has been installed with its service API accessible from the worker nodes). Depending on which mode you select, the initial configuration/usage will differ.

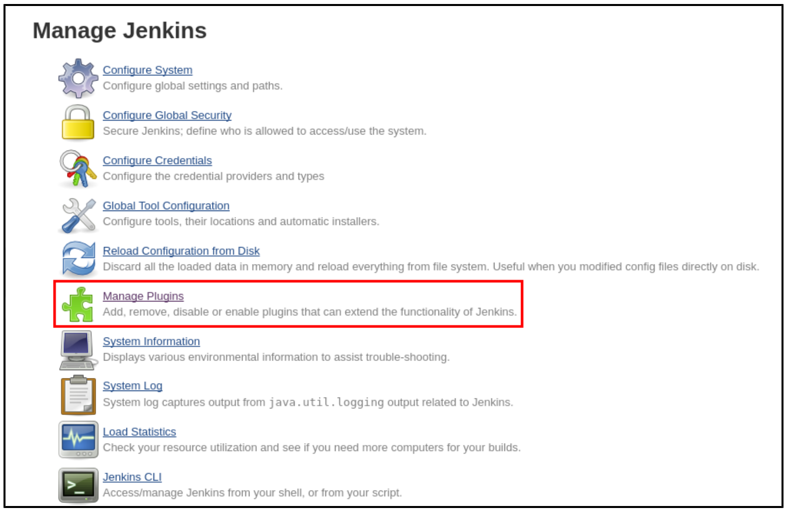
Installation

The Anchore plugin has been published in the Jenkins plugin registry and is available for installation on any Jenkins server.

From the main Jenkins menu select Manage Jenkins



Next select Manage Plugins



Manage Jenkins

Configure System
Configure global settings and paths.

Configure Global Security
Secure Jenkins; define who is allowed to access/use the system.

Configure Credentials
Configure the credential providers and types

Global Tool Configuration
Configure tools, their locations and automatic installers.

Reload Configuration from Disk
Discard all the loaded data in memory and reload everything from file system. Useful when you modified config files directly on disk.

Manage Plugins
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.

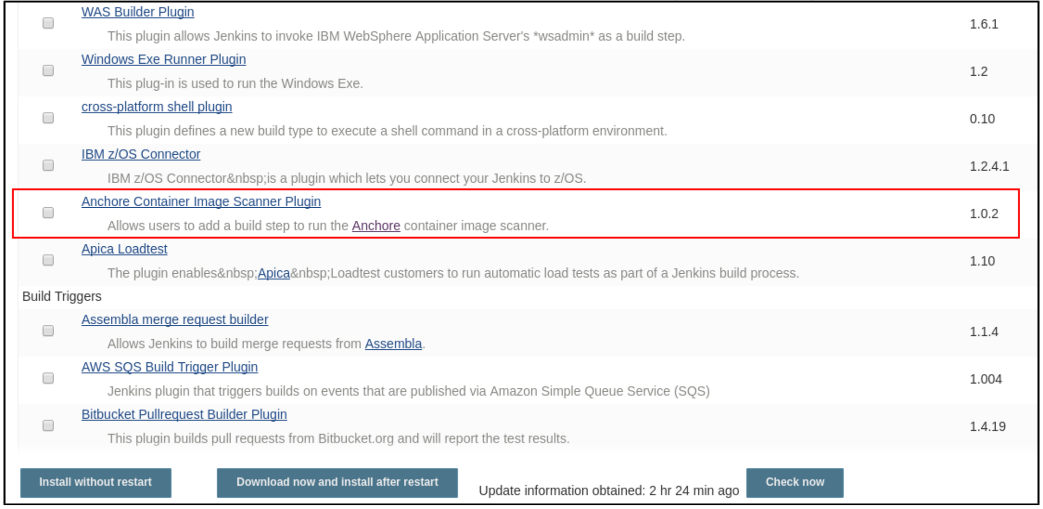
System Information
Displays various environmental information to assist trouble-shooting.

System Log
System log captures output from `java.util.logging` output related to Jenkins.

Load Statistics
Check your resource utilization and see if you need more computers for your builds.

Jenkins CLI
Access/manage Jenkins from your shell, or from your script.

Click on the *Available* tab and scroll down to select “Anchore Container Image Scanner Plugin” and click on “Install without restart”

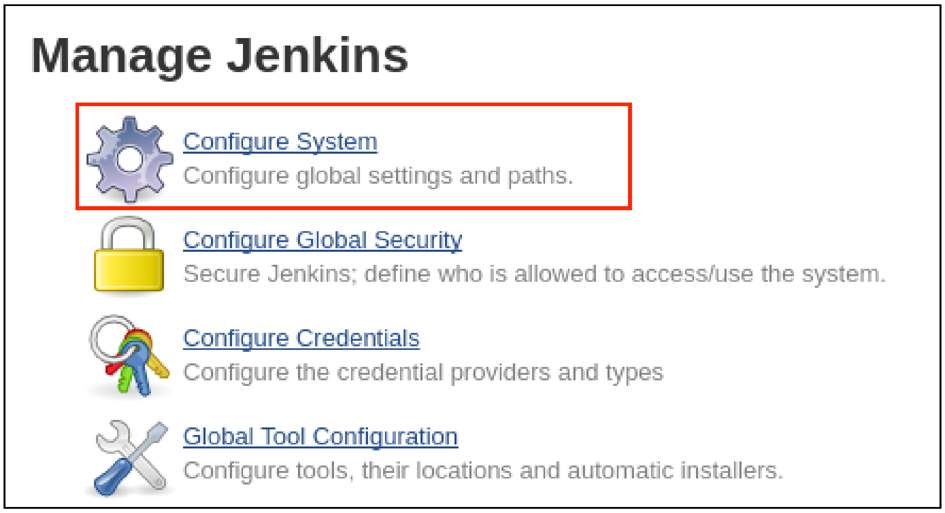


□	WAS Builder Plugin This plugin allows Jenkins to invoke IBM WebSphere Application Server's *wsadmin* as a build step.	1.6.1
□	Windows Exe Runner Plugin This plug-in is used to run the Windows Exe.	1.2
□	cross-platform shell plugin This plugin defines a new build type to execute a shell command in a cross-platform environment.	0.10
□	IBM z/OS Connector IBM z/OS Connector is a plugin which lets you connect your Jenkins to z/OS.	1.2.4.1
□	Anchore Container Image Scanner Plugin Allows users to add a build step to run the Anchore container image scanner.	1.0.2
□	Apica Loadtest The plugin enables Apica Loadtest customers to run automatic load tests as part of a Jenkins build process.	1.10
Build Triggers		
□	Assembla merge request builder Allows Jenkins to build merge requests from Assembla.	1.1.4
□	AWS SQS Build Trigger Plugin Jenkins plugin that triggers builds on events that are published via Amazon Simple Queue Service (SQS)	1.004
□	Bitbucket Pullrequest Builder Plugin This plugin builds pull requests from Bitbucket.org and will report the test results.	1.4.19

Update information obtained: 2 hr 24 min ago

Configuration

From the Manage Jenkins menu select Configure System



Manage Jenkins

Configure System
Configure global settings and paths.

Configure Global Security
Secure Jenkins; define who is allowed to access/use the system.

Configure Credentials
Configure the credential providers and types

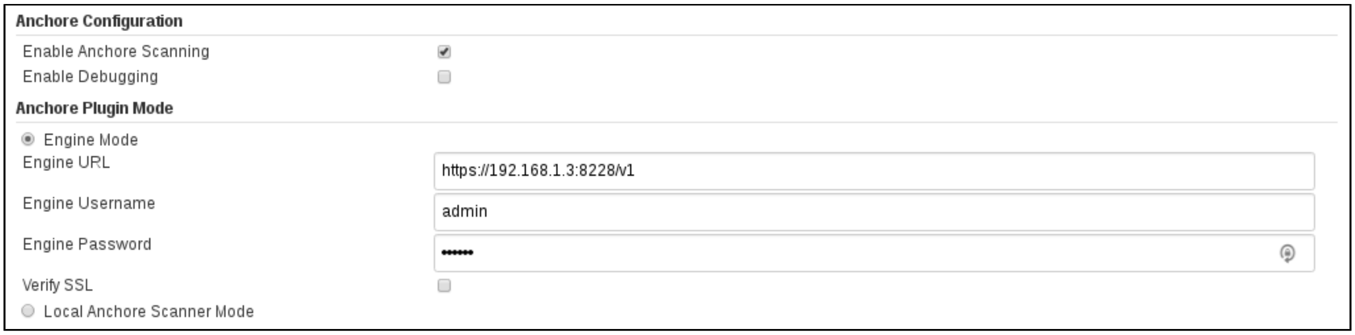
Global Tool Configuration
Configure tools, their locations and automatic installers.

Scroll down to the **Anchore Configuration** section

- Ensure that *Enable Anchore Scanning* is enabled
- Select *Engine Mode*

- Update *Engine URL* to point to the Anchore Engine
  Note: Ensure that the /v1 route is included in the URL
- Enter *Engine Username*.
  eg. Admin
- Enter *Engine Password*

If the Anchore Engine uses a user created certificate that is not signed by a standard certificate authority then select uncheck *Verify SSL*



Anchore Configuration

Enable Anchore Scanning ☑

Enable Debugging □

Anchore Plugin Mode

● Engine Mode

Engine URL: https://192.168.1.3:8228/v1

Engine Username: admin

Engine Password: ••••••

Verify SSL □

○ Local Anchore Scanner Mode

Adding Anchore Scanning to Jenkins Build

The Anchore plugin can be added a build step for a Freestyle or Pipeline build. Typically the flow is as follows.

A Jenkins job will:

1. Build a container image
2. Push the image to a Docker Registry, typically a staging registry for QA
3. Use Anchore Plugin in a Pipeline job or add Anchore Container Image Scanner build step to a Freestyle job to instruct the Anchore Engine to analyze the image
   a. The Anchore Engine downloads (pulls) the image layers from the staging registry
   b. The Anchore Engine performs analysis on the image
   c. The Anchore Engine performs a policy evaluation on the image.
4. The Anchore Plugin polls the Anchore Engine for a user defined period until the analysis and policy evaluation is complete
5. Based on user configuration, the Anchore Plugin may fail the build in the case of a Policy violation or allow the built to continue with warnings.

When run, the Anchore Plugin will look for a file named *anchore_images* in the project workspace. This file should contain the name(s) of containers to be scanned and optionally include the Dockerfile.

Freestyle

In the example below an *Execute Shell* build step is used to build and push a container image to a local registry.

```
TAG=$(date "+%H%M%S%d%m%Y")
IMAGENAME=build.example.com/myapp
docker build -t $IMAGENAME:$TAG .
docker push $IMAGENAME:$TAG
```

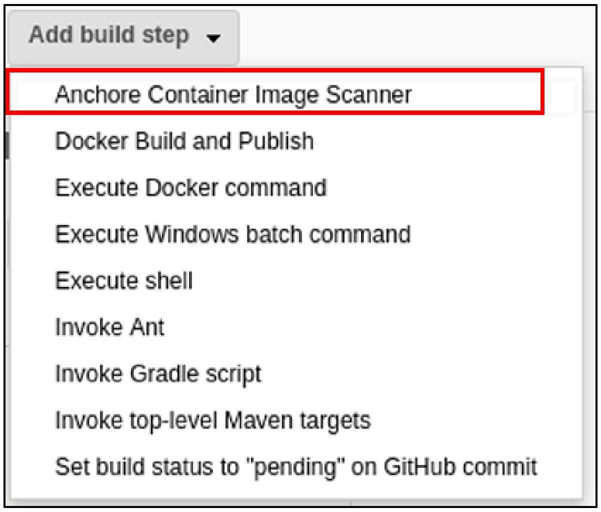
We will add a single line to create the *anchore_images* file that is read by the Anchore Plugin

Note: Multiple lines can be added if the build produces more than a single container image.

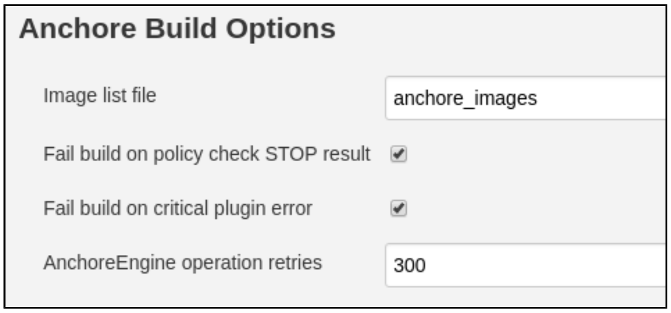
```
TAG=$(date "+%H%M%S%d%m%Y")
IMAGENAME=build.example.com/myapp
docker build -t $IMAGENAME:$TAG .
docker push $IMAGENAME:$TAG

# Line added to create anchore_images file
echo "$IMAGENAME:$TAG ${WORKSPACE}/Dockerfile " > anchore_images
```

After the image has been built and pushed to the staging registry the Anchore Scanner should be called.

Dropdown *Add build step* and select the *Anchore Container Image Scanner*


Add build step
Anchore Container Image Scanner
Docker Build and Publish
Execute Docker command
Execute Windows batch command
Execute shell
Invoke Ant
Invoke Gradle script
Invoke top-level Maven targets
Set build status to "pending" on GitHub commit

A new build step labeled **Anchore Build Options** will appear in your job.



Anchore Build Options

Image list file: anchore_images

Fail build on policy check STOP result ☑

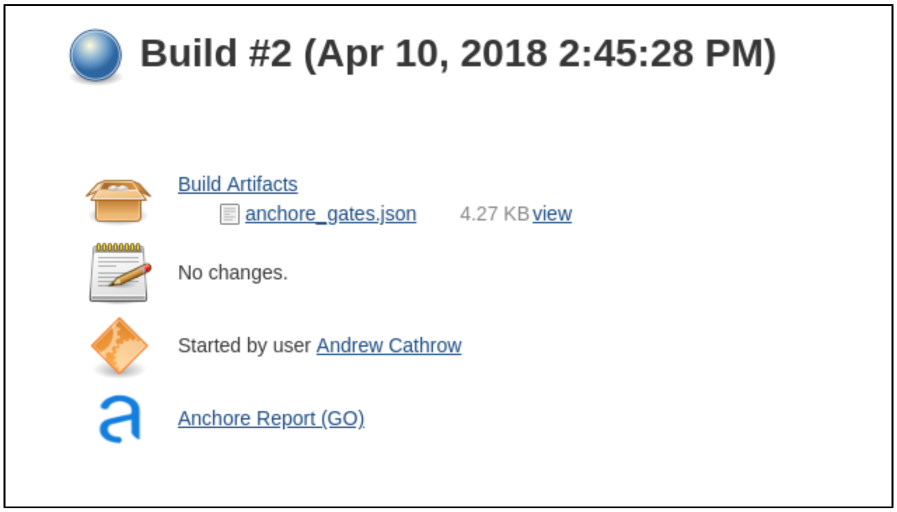
Fail build on critical plugin error ☑

AnchoreEngine operation retries: 300

Option	Description
Image list file	Name of the file, present in workspace that contains the image name and optionally Dockerfile location
Fail build on policy check STOP result	If the Anchore Engine policy evaluate returns a fail (STOP) then the Jenkins job should be failed. If this is not selected then a failed policy evaluation will allow the build to continue.
Fail build on critical plugin error	If selected and the Anchore Plugin experiences a critical error the the build will be failed. This is typically used to ensure that a fault with the Anchore Engine (eg. service not available) does not permit a failing image to be promoted to production.
AnchoreEngine operation retries	How long in seconds the Anchore Plugin waits until timing out image analysis. The Plugin will continue operation once the image has been analyzed but will time out if this period is exceeded.

The Anchore Plugin creates an Anchore Report directory that includes a JSON file including the results of the policy evaluation.

The Plugin renders this in the Jenkins UI showing the status of the build (GO = Pass, STOP = Fail, WARN=Warning)



Build #2 (Apr 10, 2018 2:45:28 PM)

Build Artifacts

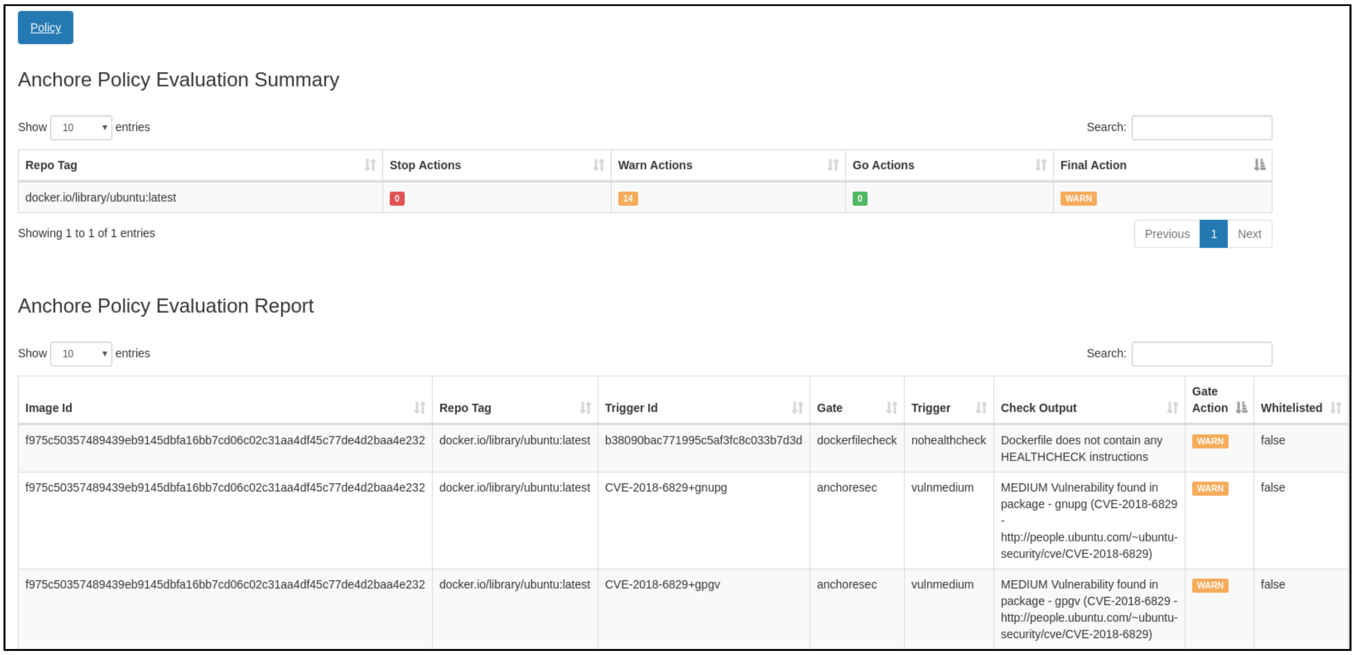
anchore_gates.json 4.27 KB view

No changes.

Started by user Andrew Cathrow

Anchore Report (GO)

Clicking on the Anchore Report link will display a graphical policy reporting showing the summary information and a detailed list of policy checks and results.



Anchore Policy Evaluation Summary

Show 10 entries

Search:

Repo Tag	Stop Actions	Warn Actions	Go Actions	Final Action
docker.io/library/ubuntu:latest	0	14	0	WARN

Showing 1 to 1 of 1 entries

Anchore Policy Evaluation Report

Show 10 entries

Search:

Image Id	Repo Tag	Trigger Id	Gate	Trigger	Check Output	Gate Action	Whitelisted
f975c50357489439eb9145dbfa16bb7cd06c02c31aa4df45c77de4d2baa4e232	docker.io/library/ubuntu:latest	b38090bac771995c5af3fc8c033b7d3d	dockerfilecheck	nohealthcheck	Dockerfile does not contain any HEALTHCHECK instructions	WARN	false
f975c50357489439eb9145dbfa16bb7cd06c02c31aa4df45c77de4d2baa4e232	docker.io/library/ubuntu:latest	CVE-2018-6829+gnupg	anchoresec	vulnmedium	MEDIUM Vulnerability found in package - gnupg (CVE-2018-6829 - http://people.ubuntu.com/~ubuntu-security/cve/CVE-2018-6829)	WARN	false
f975c50357489439eb9145dbfa16bb7cd06c02c31aa4df45c77de4d2baa4e232	docker.io/library/ubuntu:latest	CVE-2018-6829+gpgv	anchoresec	vulnmedium	MEDIUM Vulnerability found in package - gpgv (CVE-2018-6829 - http://people.ubuntu.com/~ubuntu-security/cve/CVE-2018-6829)	WARN	false

Pipeline

Following is a sample code snippet for using Anchore Plugin in a pipeline script. For more options refer to Pipeline Syntax and try the Snippet Generator

```
node {
  def imageLine = 'debian:latest'
  writeFile file: 'anchore_images', text: imageLine
  anchore name: 'anchore_images'
}
```