Table of Contents

About this Guide ........................................................................................................ 5
  About Qualys .............................................................................................................. 5
  Qualys Support ......................................................................................................... 5

Container Security Overview ................................................................................ 6
  Concepts and Terminologies ..................................................................................... 7

Get Started ........................................................................................................... 10
  Qualys Subscription and Modules required .......................................................... 10
  System support ....................................................................................................... 10
  Deploying Container Sensor .................................................................................... 10
  Sensor network configuration ................................................................................ 13

Securing Container Assets .................................................................................... 14
  Asset Inventory ...................................................................................................... 14
  Asset Details .......................................................................................................... 14
  Vulnerability scanning of Docker Images ............................................................ 17
  Vulnerability scanning of Docker Containers ..................................................... 19
  Vulnerability scanning of Docker Hosts ............................................................... 20
  Working with Events ............................................................................................... 20

Registry Scanning .................................................................................................. 21
  Docker host requirements ....................................................................................... 21
  Installing Registry Sensor ..................................................................................... 21
  Adding a new registry to scan ............................................................................... 22
  Creating a registry scan schedule ......................................................................... 23
  Viewing vulnerable registry images ...................................................................... 24

Container Security APIs ....................................................................................... 25
  Accessing the APIs ................................................................................................. 25
  Qualys Platform URL to use ................................................................................ 26
  List of Container Security APIs .......................................................................... 27
  API Samples ........................................................................................................... 28

Administration ...................................................................................................... 47
  Sensor updates ....................................................................................................... 47
  How to uninstall sensor ...................................................................................... 48

Troubleshooting ..................................................................................................... 49
  Check sensor logs ................................................................................................. 49
Appendix

Qualys Vulnerability Analysis Plugin for Jenkins

Qualys Vulnerability Analysis Plugin for Bamboo

Installing the sensor on a MAC

Deploying sensor in Kubernetes

Deploying sensor in Docker Swarm

Deploying sensor in AWS ECS Cluster

Deploying sensor in Mesosphere DC/OS
About this Guide

Welcome to Qualys Container Security! We’ll help you get acquainted with the Qualys solutions for securing your Container environments like Images, Containers and Docker Hosts using the Qualys Cloud Security Platform.

About Qualys

Qualys, Inc. (NASDAQ: QLYS) is a pioneer and leading provider of cloud-based security and compliance solutions. The Qualys Cloud Platform and its integrated apps help businesses simplify security operations and lower the cost of compliance by delivering critical security intelligence on demand and automating the full spectrum of auditing, compliance and protection for IT systems and web applications.

Founded in 1999, Qualys has established strategic partnerships with leading managed service providers and consulting organizations including Accenture, BT, Cognizant Technology Solutions, Deutsche Telekom, Fujitsu, HCL, HP Enterprise, IBM, Infosys, NTT, Optiv, SecureWorks, Tata Communications, Verizon and Wipro. The company is also founding member of the Cloud Security Alliance (CSA). For more information, please visit www.qualys.com

Qualys Support

Qualys is committed to providing you with the most thorough support. Through online documentation, telephone help, and direct email support, Qualys ensures that your questions will be answered in the fastest time possible. We support you 7 days a week, 24 hours a day. Access online support information at www.qualys.com/support/.
Container Security Overview

Qualys Container Security provides discovery, tracking, and continuously protecting container environments. This addresses vulnerability management for images and containers in their DevOps pipeline and deployments across cloud and on-premise environments.

With this version, Qualys Container Security supports
- Discovery, inventory, and near-real time tracking of container environments
- Vulnerability analysis for images and containers
- Vulnerability analysis for registries
- Integration with CI/CD pipeline using APIs (DevOps flow)
- Uses new ‘Container Sensor’ – providing native container support, distributed as docker image
**Docker Image**

A Docker image is a read-only template. For example, an image could contain an Ubuntu operating system with Apache and your web application installed. Images are used to create Docker containers. Docker provides a simple way to build new images or update existing images, or you can download Docker images that other people have already created. Docker images are the build component of Docker.

An image is a static specification what the container should be in runtime, including the application code inside the container and runtime configuration settings. Docker images contain read-only layers, which means once an image is created it is never modified.

Image is tracked within Qualys Container Security module using Image Id and also a unique identifier generated by Qualys called Image UUID.

**Docker Registry**

Docker registries hold images. These are public or private stores from which you upload or download images. The public Docker registry is provided with the Docker Hub, Quay.io or from cloud providers like AWS ECR, Azure Container Registry or Google Container Registry. It serves a huge collection of existing images for your use. These can be images you create yourself or you can use images that others have previously created. Docker registries are the distribution component of Docker.

**Docker Containers**

Docker containers are similar to a directory. A Docker container holds everything that is needed for an application to run. Each container is created from a Docker image. Docker containers can be run, started, stopped, moved, and deleted. Each container is an isolated and secure application platform. Docker containers are the run component of Docker.

A running Docker container is an instantiation of an image. Containers derived from the same image are identical to each other in terms of their application code and runtime dependencies. But unlike images that are read-only, each running container includes a writable layer (a.k.a. the container layer) on top of the read-only content. Runtime
changes, including any writes and updates to data and files, are saved in the container layer only. Thus multiple concurrent running containers that share the same underlying image may have different container layers.

Containers are tracked within Qualys Container Security module using Container Id and also a unique identifier generated by Qualys called Container UUID.

**Docker Host**

Hosts or servers running docker daemon and hosting containers and images. Qualys tracks them as Host Assets, collects the metadata including IP address, DNS and other attributes of the Host. A host in Qualys is identified by a unique identifier Host UUID. The UUID is also stored in a marker file under `/usr/local/qualys` directory by the Agent or a scan with authentication via a Scanner Appliance.

**Qualys Container Sensor**

The new sensor from Qualys designed for native support of Docker environments. Sensor is packaged and delivered as a Docker Image. Download the image and deploy it as a Container alongside with other application containers on the host.

These are docker based, can be deployed on hosts in your data center or cloud environments like AWS ECS, Azure Container Service or Google Container Service. Sensor currently is only supported on Linux Operating systems like CentOS, Ubuntu, RHEL, Debian and requires docker daemon of version 1.12 and higher to be available.

Since they are docker based, the sensor can be deployed into orchestration tool environments like Kubernetes, Mesos or Docker Swarm just like any other application container.

Upon installation, the sensor does automatic discovery of Images and Containers on the deployed host, provides a vulnerability analysis of them, and additionally it monitors and reports on the docker related events on the host. The sensor lists and scans registries for vulnerable images. The sensor container runs in non-privileged mode. It requires a persistent storage for storing and caching files.
Currently, the sensor only scans Images and Containers, for getting a vulnerability posture on the Host, you would require Qualys Cloud Agents or a scan through Qualys Virtual Scanner Appliance. Currently doesn’t do inventory collection specific to orchestration tools and identifies the nodes/slaves as just docker hosts.
Get Started

Follow the steps to get started with Container Security.

Qualys Subscription and Modules required

You would require “Container Security” (CS) module enabled for your account. Additionally, in order to get vulnerabilities for the hosts that run the containers, you would need to enable Vulnerability Management (VM), either via Scanner Appliance or Cloud Agent.

System support

Container Security supports these systems running Docker version 1.12 or later.
- Ubuntu
- Red Hat Enterprise Linux
- Debian
- CentOS

Deploying Container Sensor

Log into your Qualys portal with your user credentials. Select Container Security from the module picker.

As a first time user, you’ll land directly into the Getting Started page.

Want to scan images in build, registry or host? Expand a use case to get to the required steps quickly.
Go to Configurations > Sensors, and then click Add New Sensor to download the sensor tar file. You can see various sensor types:

- **General Sensor**: Scan any host other than registry / build (CI/CD).
- **Registry Sensor**: Scan images in a registry (public / private).
- **CI/CD Sensor**: Scan images on CI/CD pipeline (Jenkins / Bamboo).

For Registry you need to append the install command with `--registry-sensor` or `-r`.

For CI/CD you need to append the install command with `--cicd-deployed-sensor` or `-c`.

Download the QualysContainerSensor.tar.xz file and run the commands generated directly from the screen on the docker host. Note the requirements for installing the sensor, the sensor needs a minimum of 1 GB persistent storage on the host.

A quick overview of the “installsensor.sh” script command line parameters options:

- **ActivationId**: Activation Id for the container sensor, auto-generated based on your subscription.
- **CustomerId**: Qualys subscription’s customerId, auto-generated based on your subscription.
- **Storage**: Directory where the sensor would store the files. Default: /usr/local/qualys/sensor/data. Create it if not already available or you can specify a custom directory location.

- **ImageFile**: Location of the Sensor ImageFile, defaults to the local directory [Optional]

- **LogLevel**: Configuration to set the logging level for sensor, accepts 0 to 5 [Optional]

- **HostIdSearchDir**: Directory to map the marker file created by Qualys Agent or Scanner appliance on the host, update if modified [Optional]

- **CPUUsageLimit**: CPU usage limit in percentage for sensor. Valid range is in between 0-100 [Optional]

- **ConcurrentScan**: Number of docker/registry asset scans to run in parallel [Optional]

- **Proxy**: IPv4/IPv6 address or FQDN of the proxy server [Optional]

- **ProxyCertFile**: Proxy certificate file path [Optional]
  
  ProxyCertFile is applicable only if Proxy has valid certificate file. If this option is not provided then Sensor would try to connect to the server with given https Proxy settings only.

  If only ProxyCertFile is provided without Proxy then Sensor would simply ignore the ProxyCertFile and it would try to connect to the server without any https proxy settings.

- **--silent** or **-s**: Run installsensor.sh in non-interactive mode [Optional]

- **--disable-auto-update**: Do not let sensor update itself automatically [Optional]

- **--cicd-deployed-sensor** or **-c**: Run Sensor in CI/CD environment

- **--registry-sensor** or **-r**: Run sensor to list and scan registry assets

For more information on installing the registry sensor, see [Registry Scanning](#).

For information on deploying the sensor in CI/CD environments, MAC, and various orchestrators and cloud environments, see the [Appendix](#).

**Proxy Support**

The install script asks for proxy configuration. You need to provide the IP Address/FQDN and port number along with the proxy certificate file path. For example,

Do you want connection via Proxy [y/N]: y
Enter Https Proxy settings [<IP Address>:<Port #] : 10.xxx.xx.xx:3xxx
Enter Https Proxy certificate file path: /etc/qualys/cloud-agent/cert/ca-bundle.crt

Your proxy server must provide access to the Qualys Cloud Platform (or the Qualys Private Cloud Platform) over HTTPS port 443. Go to Help > About to see the URL your hosts need to access.
Sensor network configuration

The sensor is pre-configured with the Qualys URL and the subscription details it needs to communicate to. In order for the sensor to communicate to Qualys, the network configuration and firewall needs to provide accessibility to Qualys domain over port 443.

After successful installation of the Sensor, the sensor is listed under Configurations > Sensors where you can see its version, status, etc. and access details.

Additionally, you can Download the sensor from the link under Configurations > Sensors.
Securing Container Assets

Asset Inventory

Upon installation of the sensor, it automatically scans the host for the images and containers that are present on the host. The inventory and the metadata of the inventory is pushed to Qualys portal.

Dashboard

Container security application provides out-of-the-box default Security Overview Dashboard providing summary of inventory and security posture across container assets.

The default dashboard provides
- An inventory list of Images, Containers and the Sensors
- Vulnerability summary for all of the images
- Vulnerability summary for all the containers
- Events and their trend over the last 24 hrs
- Listing of the most popular Images and Containers by Labels

Asset Details

Assets tab list the Images and Containers discovered along with their metadata information like ports, networks, services, users, installed software, etc. The assets are listed along with their associations like associated containers and hosts for an image, other containers from the same parent image. Users can search for images and containers based on their attributes.
Clicking View Details in the Quick Actions menu for an image in the Assets > Images tab, displays comprehensive information about the image.

You can view detailed information about that image, its associations with containers, rogue containers, and hosts. Installed Software panel displays software having vulnerabilities, and for which fixes (patches) are available. You can view the vulnerability information such as confirmed vulnerabilities, potential vulnerabilities with their severity. You can view the age of a vulnerability. The age value is displayed in days. Age is calculated from the point Qualys published the vulnerability. The Layers panel displays a list of layers the image is made of.
Securing Container Assets

Asset Details

Clicking View Details in the Quick Actions menu for a container in the Assets > Containers tab, displays comprehensive information about the container.

You can view detailed information about that container, its associations with an image, rogue containers, and hosts. Installed Software panel displays software having vulnerabilities, and for which fixes (patches) are available. You can view the vulnerability information such as confirmed vulnerabilities, potential vulnerabilities with their severity. You can view the age of a vulnerability. The age value is displayed in days. Age is calculated from the point Qualys published the vulnerability. Services/Users panel displays the list of services available in the container and users associated with the container.

Container “State” is updated based on the docker events (exec_start, kill, destroy, stop) that Qualys Sensor reports to Qualys Cloud Platform.
Host Details

Access the details page for a host from the Sensor details page. Asset Details view displays information about the host on which the sensor is deployed. Besides system, network, and port information, the Asset Details view also displays a list of software installed on the host, vulnerabilities present, certificates, and Threat Protection RTIs (when Qualys TP app is enabled). Container Security panel shows all containers installed on the host, their status, and the images from which the containers are spawned.

Vulnerability scanning of Docker Images

The docker images are scanned to check the presence of any vulnerabilities by the Qualys container sensor. The vulnerabilities panel in Image Details provides a list of vulnerabilities with Severity along with their QIDs. Select **Show Patchable Vulnerabilities** to view vulnerabilities with available patches.
Qualys scans the docker images for vulnerabilities not through static analysis but via a non-static method, where it looks at the Image as a complete entity. This process is more effective and has lesser false positives (FP) than the more commonly used Static Analysis.

Docker Images are found distributed across the environment from developer laptops, build systems, Image Registry to being cached on the docker hosts running Containers. To scan for vulnerabilities you would need the Container Sensor deployed on the host asset.

To get an inventory of the images and scan them for vulnerabilities, deploy the container sensor on the host. Refer to Deploying Container Sensor for the install instructions and system requirements.

**On the local host or laptops**

To get an inventory of the images and scan them for vulnerabilities, deploy the container sensor on the local host. Refer to Deploying Container Sensor for the install instructions and system requirements

To deploy the Sensor on the Mac laptops, there are additional install steps - follow the instructions in the Appendix. See Installing the sensor on a MAC.

Upon Installation the sensor automatically detects the images, and provides -inventory and vulnerability scans of the image.

**In the CI/CD pipeline**

Doing a complete check of vulnerabilities in an image during the build time ensures a lot cleaner operating environment. Qualys Container Security provides a plugin for Jenkins and Bamboo to get the vulnerability analysis of images in the build environment. If you are using other tools you can use the REST APIs available to perform vulnerability analysis on the images.

To start, deploy the Container Sensor on the Build host where the images are being created. The sensor upon install would automatically trigger a vulnerability analysis of the new images found. Use the API or the plug-in to look for vulnerabilities in the Images.
Securing Container Assets

Vulnerability scanning of Docker Containers

If you are in Jenkins or Bamboo environment, the plug-in would provide detail list of the vulnerabilities and its details directly within the plug-in, you could optionally access your Qualys subscription to view the full report.

In the Registry

Currently, the Qualys Container Sensor doesn’t automatically poll or pull images to do an analysis. Rather you would be needed to deploy the sensor on the host that is configured to pull images from the registry. Either manually or via a cron pull the new images to the host. The sensor does an automatic analysis as soon as it finds a new image. Use the APIs or the Qualys portal to query for the vulnerabilities identified.

Vulnerability scanning of Docker Containers

The containers are scanned to check the presence of any vulnerabilities within the containers. The Vulnerabilities panel in Container Details provides a list of vulnerabilities with Severity along with their QIDs. Select Show Patchable Vulnerabilities to view vulnerabilities with available patches.

Good to know!

**Rogue Containers** are those which contain vulnerabilities or software, not found in the image from which the container is spawned.

**Rogue Vulnerabilities** are classified as either New, Fixed or Varied. New are those which are newly found on the containers, but were not present in the image from which the container is spawned. Fixed, are the vulnerabilities that are not found in the container but in the image. Varied, are the vulnerabilities that are found in both Containers and Images but the detection varies between them.

**Rogue Software** are classified as new or removed. New, software which are found in the Container but not in the image from which the container is spawned. Fixed, Software not seen in the Container but is present in the parent Image.
Vulnerability scanning of Docker Hosts

Container Security Sensor scans Images and Containers for vulnerabilities, and not the actual host machine. You can scan the host via Scanner Appliance or Cloud Agent. Configurations required on the host for using the Cloud Agent are independent of the Sensor. For example, proxy configuration.

Working with Events

The Events tab displays a log for all activities that you perform on the images and containers installed on the host. These activities could be tagging of images, create, destroy or commit containers. The events list displays the timestamp of the event along with the event ID, type, and the host on which the activities took place.
Registry Scanning

Registry images are scanned to check the presence of any vulnerabilities by the Qualys container sensor. You can scan public and private registries for vulnerable images. Public registries are those hosted on cloud providers such as amazon, azure or google. Private registries are on-premise such as those hosted using artifactory or nexus.

Docker host requirements

As a prerequisite you must install the registry sensor on a docker host which has access to the registry to pull images to scan.

Docker host configuration

Docker version - 1.12 or later.

Disk space on docker host - Minimum 20 GB of free space on the partition where docker is installed. This is required to scan registry images. Additionally, 1 GB of free space is required for persistent storage.

Connectivity - Docker host should have connectivity to the Registry to be scanned.

To validate connectivity, perform a successful docker login from the host to the Registry.

docker login <registryurl> (No protocol)

For Example,

docker login myregistry.com:5001

Installing Registry Sensor

To download the sensor, simply go to Configurations > Sensors, click Add New Sensor and then select Registry Sensor.

You need to append --registry-sensor or -r to the sensor install command to install the sensor for registry scan.
Adding a new registry to scan

You need to add a registry in order to scan it for vulnerabilities. Go to Assets > Registries, and click New Registry. Ensure that registry sensor deployed on the docker host is in running state.

In order to perform vulnerability analysis you need to connect to the registries using credentials. You need different types of credentials to connect to different registries. Credential types supported are Token, BasicAuth, DockerHub, AWS.
For AWS ECR, you can create a connector to connect to your AWS account.

Creating a registry scan schedule

You can choose to scan immediately (On Demand scan) or on an ongoing basis (Automatic scan).

**On Demand** scan allows you to scan repositories as well as specific images within those repositories.

With **Automatic** scan, you can scan entire repositories at a set time every day.
Viewing vulnerable registry images

Once you connect to the registry, Container Security pulls the inventory data and performs vulnerability scans on repositories and images within the registries. Vulnerable images are listed on the Images tab.

To get the total count of vulnerable images in a registry, go to Registries tab, and click View Details in the Quick Actions Menu of a registry. The Scan Jobs panel shows a list of schedules created for scanning the registry.
Container Security APIs

Accessing the APIs

All features of Container Security are available through REST APIs. Equivalent Rest API request for each tab is provided on the UI.

Click Rest Reference to launch the Swagger UI, where you can try out the Rest APIs.
Where’s the API documentation?
You can directly access the Swagger UI from the following URL

https://<QualysURL>/csapi/swagger-ui.html

For example, if your account is on US Platform 2


Do I need to Authenticate?
Authentication to the Qualys Cloud Platform is necessary before you try out the APIs.
Simply, click Authorize and provide the user name and password. You can now try out the APIs!

Permissions required to use APIs
- User must have the Container module enabled
- User must have API ACCESS permission

Qualys Platform URL to use
Qualys maintains multiple platforms. The Qualys platform URL that you should use for API requests depends on the platform where your account is located.

<table>
<thead>
<tr>
<th>Account Location</th>
<th>Platform URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Qualys US Platform 1</td>
<td><a href="https://qualysguard.qualys.com">https://qualysguard.qualys.com</a></td>
</tr>
<tr>
<td>Qualys US Platform 2</td>
<td><a href="https://qualysguard.qg2.apps.qualys.com">https://qualysguard.qg2.apps.qualys.com</a></td>
</tr>
<tr>
<td>Qualys US Platform 3</td>
<td><a href="https://qualysguard.qg3.apps.qualys.com">https://qualysguard.qg3.apps.qualys.com</a></td>
</tr>
<tr>
<td>Qualys EU Platform 1</td>
<td><a href="https://qualysguard.qualys.eu">https://qualysguard.qualys.eu</a></td>
</tr>
<tr>
<td>Qualys EU Platform 2</td>
<td><a href="https://qualysapi.qg2.apps.qualys.eu">https://qualysapi.qg2.apps.qualys.eu</a></td>
</tr>
<tr>
<td>Qualys India Platform 1</td>
<td><a href="https://qualysguard.qg1.apps.qualys.in">https://qualysguard.qg1.apps.qualys.in</a></td>
</tr>
</tbody>
</table>
# List of Container Security APIs

Here is the list of the APIs we currently support:

<table>
<thead>
<tr>
<th>API Objective</th>
<th>Operator</th>
<th>API Path</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Container</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Show a list of containers in your account</td>
<td>GET</td>
<td>/csapi/v1.1/containers</td>
</tr>
<tr>
<td>Show details of a container</td>
<td>GET</td>
<td>/csapi/v1.1/containers/{containerId}</td>
</tr>
<tr>
<td>Show software installed on a container</td>
<td>GET</td>
<td>/csapi/v1.1/containers/{containerId}/software</td>
</tr>
<tr>
<td>Show vulnerability details for a container</td>
<td>GET</td>
<td>/csapi/v1.1/containers/{containerId}/vuln</td>
</tr>
<tr>
<td>Show vulnerability count for a container</td>
<td>GET</td>
<td>/csapi/v1.1/containers/{containerId}/vuln/count</td>
</tr>
<tr>
<td>Delete containers in your account</td>
<td>DELETE</td>
<td>/csapi/v1.1/containers</td>
</tr>
<tr>
<td><strong>Event</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Show a list of events in your account</td>
<td>GET</td>
<td>/csapi/v1.1/events</td>
</tr>
<tr>
<td><strong>Image</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Show a list of images in your account</td>
<td>GET</td>
<td>/csapi/v1.1/images</td>
</tr>
<tr>
<td>Show details of an image</td>
<td>GET</td>
<td>/csapi/v1.1/images/{imageId}</td>
</tr>
<tr>
<td>Show associations for an image</td>
<td>GET</td>
<td>/csapi/v1.1/images/{imageId}/association</td>
</tr>
<tr>
<td>Show software installed on an image</td>
<td>GET</td>
<td>/csapi/v1.1/images/{imageId}/software</td>
</tr>
<tr>
<td>Show vulnerability details for an image</td>
<td>GET</td>
<td>/csapi/v1.1/images/{imageId}/vuln</td>
</tr>
<tr>
<td>Show vulnerability count for an image</td>
<td>GET</td>
<td>/csapi/v1.1/images/{imageId}/vuln/count</td>
</tr>
<tr>
<td>Delete images in your account</td>
<td>DELETE</td>
<td>/csapi/v1.1/images</td>
</tr>
<tr>
<td><strong>Registry</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Show a list of registries in your account</td>
<td>GET</td>
<td>/csapi/v1.1/registry</td>
</tr>
<tr>
<td>Show details of a registry</td>
<td>GET</td>
<td>/csapi/v1.1/registry/{registryId}</td>
</tr>
<tr>
<td>Fetch AWS account ID and External ID for your account</td>
<td>GET</td>
<td>/csapi/v1.1/registry/aws-base</td>
</tr>
<tr>
<td>Show a list of AWS connectors in your account</td>
<td>GET</td>
<td>/csapi/v1.1/registry/aws/connectors</td>
</tr>
<tr>
<td>Show a list of AWS connectors for an AWS account ID</td>
<td>GET</td>
<td>/csapi/v1.1/registry/aws/connectors/{accountId}</td>
</tr>
</tbody>
</table>
### Container Security APIs

**API Samples**

<table>
<thead>
<tr>
<th>Sample 1</th>
<th>Sample 2</th>
<th>Sample 3</th>
<th>Sample 4</th>
<th>Sample 5</th>
<th>Sample 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get list of currently running containers in your account</td>
<td>Get details for container ID c4905cc26d35</td>
<td>Show a list of events occurred between 30 June 2018 and 11 July 2018</td>
<td>Show images with severity 5 vulnerabilities</td>
<td>Create registry</td>
<td>Create AWS connector for registry</td>
</tr>
</tbody>
</table>

In the API response, 

- `associatedContainersCount` shows count of containers in RUNNING or STOPPED state.
- `associatedHostsCount` shows count of hosts where Qualys sensor AND the image is installed.

<table>
<thead>
<tr>
<th>Create new AWS connector</th>
<th>POST</th>
<th>/csapi/v1.1/registry/aws/connector</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate information for new registry</td>
<td>POST</td>
<td>/csapi/v1.1/registry/validate</td>
</tr>
<tr>
<td>Create a new registry</td>
<td>POST</td>
<td>/csapi/v1.1/registry</td>
</tr>
<tr>
<td>Update existing registry in your account</td>
<td>PUT</td>
<td>/csapi/v1.1/registry/{registryId}</td>
</tr>
<tr>
<td>Show a list of repositories in a registry</td>
<td>GET</td>
<td>/csapi/v1.1/registry/{registryId}/repository</td>
</tr>
<tr>
<td>Show a list of schedules created for a registry</td>
<td>GET</td>
<td>/csapi/v1.1/registry/{registryId}/schedule</td>
</tr>
<tr>
<td>Create a new registry scan schedule</td>
<td>POST</td>
<td>/csapi/v1.1/registry/{registryId}/schedule</td>
</tr>
<tr>
<td>Update existing registry schedule in your account</td>
<td>PUT</td>
<td>/csapi/v1.1/registry/{registryId}/schedule/{scheduleId}</td>
</tr>
<tr>
<td>Delete registry in your account</td>
<td>DELETE</td>
<td>/csapi/v1.1/registry/{registryId}</td>
</tr>
<tr>
<td>Delete multiple registries in your account</td>
<td>DELETE</td>
<td>/csapi/v1.1/registry</td>
</tr>
<tr>
<td>Delete registry schedule in your account</td>
<td>DELETE</td>
<td>/csapi/v1.1/registry/{registryId}/schedule/{scheduleId}</td>
</tr>
<tr>
<td>Delete multiple registry schedules in your account</td>
<td>DELETE</td>
<td>/csapi/v1.1/registry/{registryId}/schedule/</td>
</tr>
</tbody>
</table>

**Sensor**

| Show a list of sensors in your account | GET | /csapi/v1.1/sensors |
| Show details of a sensor | GET | /csapi/v1.1/sensors/{sensorId} |
| Delete sensors in your account | DELETE | /csapi/v1.1/sensors |

28
**Sample 7 - Delete sensors in your account**

**Sample 8 - Delete images in your account**

**Sample 9 - Delete multiple registries (bulk delete) in your account**

### Sample 1 - Get list of currently running containers in your account

```
/csapi/v1.1/containers
```

[GET]

**Input Parameters:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filter</td>
<td>Filter the containers list by providing a query using Qualys syntax. Refer to the &quot;How to Search&quot; topic in the online help for assistance with creating your query.</td>
</tr>
<tr>
<td>pageNo</td>
<td>(Required) The page to be returned. Page numbers start with 0.</td>
</tr>
<tr>
<td>pageSize</td>
<td>(Required) The number of records per page to be included in the response.</td>
</tr>
<tr>
<td>sort</td>
<td>Sort the results using a Qualys token. For example <code>created:desc</code>. Refer to the &quot;Sortable tokens&quot; topic in the online help for more information.</td>
</tr>
</tbody>
</table>

**API request:**

```
```

**Response:**

```
{
    "data": [
        {
            "imageId": "61205f6444f9",
            "created": "1529489437000",
            "uuid": "6d74becc-cd23-3c78-af16-5be95eb19dc7",
            "name": "conta",
            "host": {
                "sensorUuid": "9a40cc0f-4b6e-436d-bacf-a58c337d627b",
                "hostname": "qubu1604ltssdp4",
                "ipAddress": "10.11.62.176",
                "uuid": "878b906c-9382-48dd-8d68-d4768db5c801",
                "runningContainerCount": null,
                "stoppedContainerCount": null,
                "runningImageContainerCount": null,
                "stoppedImageContainerCount": null
            }
        }
    ]
}```
"state": "RUNNING",
"imageUuid": "2cd08101-dc49-3572-8fbd-a7ca85a7e596",
"containerId": "c4905cc26d35",
"stateChanged": "1529963367464",
"lastScanned": "1531310274033",
"vulnerabilities": {
  "severity5Count": 0,
  "severity3Count": 4,
  "severity4Count": 0,
  "severity1Count": 0,
  "severity2Count": 2
},
"rogueVulnerabilities": null,
"rogueSoftware": null,
"softwareCount": null
},

{
  "imageId": "49ae92f15a2c",
  "created": "1529396311000",
  "uuid": "1b5c7c10-5b7d-3935-b2c3-b9d5c0e2b80d",
  "name": "contb",
  "host": {
    "sensorUuid": "9a40cc0f-4b6e-436d-bacf-a58c337d627b",
    "hostname": "qubu1604ltssdp4",
    "ipAddress": "10.11.62.176",
    "uuid": "878b906c-9382-48dd-8d68-d4768db5c801",
    "runningContainerCount": null,
    "stoppedContainerCount": null,
    "runningImageContainerCount": null,
    "stoppedImageContainerCount": null
  },
  "state": "RUNNING",
  "imageUuid": "ab3dc427-0d70-349d-9a6f-68855c4173c0",
  "containerId": "b38166df28b1",
  "stateChanged": "1529963367650",
  "lastScanned": "1531310273409",
  "vulnerabilities": {
    "severity5Count": 0,
    "severity3Count": 2,
    "severity4Count": 0,
    "severity1Count": 0,
    "severity2Count": 2
  },
  "rogueVulnerabilities": null,
}
Sample 2 - Get details for container ID c4905cc26d35
/csapi/v1.1/containers/{containerId}

[GET]
Parameters list:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>containerId</td>
<td>Specify the container ID of a specific container in the user’s scope.</td>
</tr>
</tbody>
</table>

API request:

```
curl -X GET --header 'Accept: application/json' --header 'Authorization: Basic cXVheXNfYmczOnFhdGVtcEAxMjM=' 'https://<QualysURL>/csapi/v1.1/containers/c4905cc26d35'
```

Response:

```
{
  "portMapping": [
    {
      "protocol": "tcp",
      "port": 8080,
      "hostIp": null,
      "hostPort": null
    }
  ],
  "imageId": "61205f6444f9",
  "created": "1529489437000",
  "label": null,
  "uuid": "6d74becc-cd23-3c78-af16-5be95eb19db7",
  "sha": "c4905cc26d35d7bacb85991cfaaffc0603e095b869e74b2c0185f57c7e685076",
  "privileged": false,
  "sensorUuid": "9a40cc0f-4b6e-436d-bacf-a58c337d627b",
  "path": "catalina.sh",
  "imageSha": "61205f6444f9e917faec4aa5d29e3c4b81c5b47fb206535ea035900a89ae38"
}
```
"macAddress": "02:42:ac:11:00:25",
"customerUuid": "81c490ed-34cc-f026-810d-6f38f0e2d1e7",
"ipv4": "172.17.0.37",
"ipv6": null,
"name": "conta",
"host": {
"sensorUuid": "9a40cc0f-4b6e-436d-bac9-a58c337d627b",
"hostname": "qubu1604ltsssdp4",
"ipAddress": "10.11.62.176",
"uuid": "878b906c-9382-48dd-8d68-d476d887ad81",
"runningContainerCount": null,
"stoppedContainerCount": null,
"runningImageContainerCount": null,
"stoppedImageContainerCount": null
},
"id": null,
"state": "RUNNING",
"imageUuid": "2cd08101-dc4f-3572-8fbd-a7ca85a7e596",
"containerId": "c4905cc26d35",
"stateChanged": "1529963367464",
"hostname": null,
"services": null,
"users": [
  "root"
],
"operatingSystem": "Debian Linux 9.4",
"lastScanned": "1531310274033",
"stateCount": null,
"vulnCategoryCount": null,
"totalVulCount": null,
"environment": [
"JAVA_HOME=/docker-java-home/jre",
"CATALINA_HOME=/usr/local/tomcat",
"TOMCAT_ASC_URLS=https://www.apache.org/dyn/closer.cgi?action=download&filename=tomcat/tomcat-8/v8.5.31/bin/apache-tomcat-8.5.31.tar.gz"
8.5.31.tar.gz.asc https://www-us.apache.org/dist/tomcat/tomcat-8/v8.5.31/bin/apache-tomcat-8.5.31.tar.gz.asc
https://www.apache.org/dist/tomcat/tomcat-8/v8.5.31/bin/apache-tomcat-8.5.31.tar.gz.asc
https://archive.apache.org/dist/tomcat/tomcat-8/v8.5.31/bin/apache-tomcat-8.5.31.tar.gz.asc

"PATH=/usr/local/tomcat/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin",
"OPENSSL_VERSION=1.1.0f-3+deb9u2",
"TOMCAT_MAJOR=8",
"TOMCAT_NATIVE_LIBDIR=/usr/local/tomcat/native-jni-lib",
"GPG_KEYS=05AB33110949707C93A279E3D3EFE6B686867BA6 07E48665A34DCAFEE522E5E6266191C37C037D42 473090207DB18FFD8D3DF83F1931D684307A10A5 541FBEE7DF78B25E055D3EE13C370389288584E7 61B832AC2F1CA90F0F9B00A1C506407564C17A3 713DA88BE50911535FE716F52080A1D63011C7 79F7026C690BA50B92CD8B66A3AD3F4E2244FED 9BA44C2621385CB966CA58F72C284D731FABEE A27677289986DB50844682F8ACB77FC2E86E29AC A9C5DF40D2399998D9875A5110C015A2F6059E7 DCFD35E0B8F8CA7344752DE8B6FB21E8933C0243 F3A04C595DB56A5F1ECA43E3B7BBB100D811BBE F7DA4B8648CBB84ECB7EE6935CD23C10498E23",
"JAVA_DEBIAN_VERSION=8u171-b11-1-deb9u1",
"JAVA_VERSION=8u171",
"TOMCAT_VERSION=8.5.31",
"CA_CERTIFICATES_JAVA_VERSION=20170531+nmu1",
"TOMCAT_SHA512=a961eedc4b0c0729f1fb96dafb75eb48e000502233b849f47c8 4a6355873bc96d131b112400587e96391262e0659df9b991b4e66a78fda74168f9 39c4ab5af88",
"LANG=C.UTF-8",
"LD_LIBRARY_PATH=/usr/local/tomcat/native-jni-lib"
],
"arguments": [
  "run"
],
"command": "catalina.sh run",
"rogue": {
  "category": ["Vulnerability"
  ],
  "reason": ["New"
  ]
}
"software": [],
"vulnerability": [
{
  "vulnerability": null,
  "result": "sysctl net.ipv4.ip_forward",
  "lastFound": "1531310274031",
  "firstFound": "1531310274031",
  "fixed": null,
  "severity": 2,
  "customerSeverity": 2,
  "port": null,
  "typeDetected": "POTENTIAL",
  "status": null,
  "nonRunningKernel": null,
  "nonExploitableConfig": null,
  "runningService": null,
  "risk": 20,
  "category": "Local",
  "os": null,
  "discoveryType": [
    "AUTHENTICATED"
  ],
  "authType": [
    "UNIX_AUTH"
  ],
  "supportedBy": [
    "VM",
    "CA-Linux Agent",
    "CA-Mac Agent"
  ],
  "product": [],
  "vendor": [],
  "cveids": [
    "CVE-1999-0511"
  ],
  "threatIntel": {
    "activeAttacks": null,
    "zeroDay": null,
    "publicExploit": null,
    "highLateralMovement": null,
    "easyExploit": true,
    "highDataLoss": null,
    "noPatch": true,
    "denialOfService": null,
    "malware": null,
  }
}
"exploitKit": null,
"publicExploitNames": null,
"malwareNames": null,
"exploitKitNames": null
},
"qid": 115284,
"title": "IP Forwarding Enabled",
"cvssInfo": {
  "baseScore": "7.5",
  "temporalScore": "6.8",
  "accessVector": "Network"
},
"cvss3Info": {
  "baseScore": "5.3",
  "temporalScore": "4.6"
},
"patchAvailable": false,
"software": null,
"source": "CONTAINER",
"reason": "New",
"imageResult": null,
"containerResult": "sysctl net.ipv4.ip_forward"
}
},
{
  "name": "perl-base",
  "version": "5.24.1-3+deb9u3",
  "fixVersion": "5.24.1-3+deb9u4",
  "vulnerabilities": [
    {
      "vulnerability": null,
      "result": "perl-base 5.24.1-3+deb9u3 5.24.1-3+deb9u4",
      "lastFound": "1531310274031",
      "firstFound": "1529969711115",
      "fixed": null,
      "severity": 3,
      "customerSeverity": 3,
      "port": null,
      "typeDetected": "CONFIRMED",
      "status": null,
      "nonRunningKernel": null,
      "nonExploitableConfig": null,
      "runningService": null,
      "risk": 30,
    }
  ]
}
"category": "Debian",
"os": null,
"discoveryType": [
  "AUTHENTICATED"
],
"authType": [
  "UNIX_AUTH"
],
"supportedBy": [
  "VM",
  "CA-Linux Agent"
],
"product": [null],
"vendor": [null],
"cveids": [null],
"threatIntel": {null},
"qid": 176409,
"title": "Debian Security Update for perl (DSA 4226-1)",
"cvssInfo": {null},
"cvss3Info": {null}
"temporalScore": "4.4",
},
"patchAvailable": true,
"software": null
}
],
...
{
  "name": "libxtst6:amd64",
  "version": "2:1.2.3-1",
  "fixVersion": null,
  "vulnerabilities": null
},
{
  "name": "libunistring0:amd64",
  "version": "0.9.6+really0.9.3-0.1",
  "fixVersion": null,
  "vulnerabilities": null
},
...
"isRogue": true
}

Sample 3 - Show a list of events occurred between 30 June 2018 and 11 July 2018
/csapi/v1.1/events

[GET]

Parameters list:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filter</td>
<td>Filter the containers list by providing a query using Qualys syntax. Refer to the “How to Search” topic in the online help for assistance with creating your query.</td>
</tr>
<tr>
<td>pageNo</td>
<td>(Required) The page to be returned. Page numbers start with 0.</td>
</tr>
<tr>
<td>pageSize</td>
<td>(Required) The number of records per page to be included in the response.</td>
</tr>
<tr>
<td>sort</td>
<td>Sort the results using a Qualys token. For example eventOccurred:desc. Refer to the “Sortable tokens” topic in the online help for more information.</td>
</tr>
<tr>
<td>fromDate</td>
<td>Show events logged after a certain date and time. Supports epoch time / unix timestamp.</td>
</tr>
<tr>
<td>toDate</td>
<td>Show events logged until a certain date and time. Supports epoch time / unix timestamp</td>
</tr>
</tbody>
</table>
API request:

curl -X GET --header 'Accept: application/json' --header
'Authorization: Basic cKVheXNfYmczOnPhdGVtcEAxMjM='
'https://<QualysURL>/csapi/v1.1/events?pageNo=0&pageSize=50&sort=eventOccurred%3Adesc&fromDate=1530347611&toDate=1531298011'

Note: fromDate and toDate contain equivalent epoch time for 30 June 2018 and 11 July 2018.

Response:

```
{
  "data": [
    {
      "containerUuid": "84f2197e-65c5-31c6-92d4-5bbd21523202",
      "imageId": "d23bdf5b1b1b",
      "eventOccurred": 1531287743910,
      "containerSha": "f6ca5d84e28b4042be85d422e4d7bd3c50b3f367146cc027ef56ec8a46496056",
      "uuid": "463e0792-a9e4-3d70-81b6-23a064cd39df",
      "sensorUuid": "9a40cc0f-4b6e-436d-bacf-a58c337d627b",
      "imageSha": "d23bdf5b1b1b1afce5f1d0fd33e7ed8afbc084b594b9ccf742a5b27080d8a4a8",
      "customerUuid": "81c490ed-34cc-f026-810d-6f38f0e2d1e7",
      "host": {
        "sensorUuid": "9a40cc0f-4b6e-436d-bacf-a58c337d627b",
        "hostname": "qubu1604ltsssdp4",
        "ipAddress": "10.11.62.176",
        "uuid": "878b906c-9382-48dd-8d68-d4768db5c801",
        "runningContainerCount": null,
        "stoppedContainerCount": null,
        "runningImageContainerCount": null,
        "stoppedImageContainerCount": null
      },
      "event": "exec_start",
      "type": "CONTAINER",
      "context": "/bin/bash",
      "name": "abc-java",
      "containerId": "f6ca5d84e28b",
      "category": "DOCKER",
      "imageUuid": "641ef0bf-d474-32a7-b89a-01c69d4762c1"
    },
    ...
    {
      "containerUuid": "5f6223f3-6fa6-3f8b-afa4-e150b277884f",
```
"imageId": "abd4f451db7",
"eventOccurred": 1531275600922,
"containerSha": "5517f25e5d2f7e28ef7f86474e06734bc3283861d17517ee6374f7b326d2f",
"uuid": "d68c828f-9875-3bac-9a0b-b6d27df4b8e8",
"sensorUuid": "9a40cc0f-4b6e-436d-bac0a58c337d627b",
"imageSha": "abd4f451db706a38e68a36d695456a515cdd6f9581b7a8348a380030a6fd7689",
"customerUuid": "81c490ed-34cc-f026-6f3f0e2d1e7",
"host": {
  "sensorUuid": "9a40cc0f-4b6e-436d-bac0a58c337d627b",
  "hostname": "qubu1604ltssdp4",
  "ipAddress": "10.11.62.176",
  "uuid": "87b906c-9382-48dd-8d68-d4768db5c801",
  "runningContainerCount": null,
  "stoppedContainerCount": null,
  "runningImageContainerCount": null,
  "stoppedImageContainerCount": null
},
"event": "exec_start",
"type": "CONTAINER",
"context": "sh",
"name": "abc-bad-image",
"containerId": "5517f25e5d2f",
"category": "DOCKER",
"imageUuid": "236f66f7-dc80-3e58-9a16-88ab987fc20f"
},
"count": 319,
"groups": {}
Container Security APIs

API Samples

<table>
<thead>
<tr>
<th>pageSize</th>
<th>(Required) The number of records per page to be included in the response.</th>
</tr>
</thead>
<tbody>
<tr>
<td>sort</td>
<td>Sort the results using a Qualys token. For example \texttt{eventOccurred:desc}. Refer to the &quot;Sortable tokens&quot; topic in the online help for more information.</td>
</tr>
</tbody>
</table>

**API request:**

```
curl -X GET --header 'Accept: application/json' --header 'Authorization: Basic cXVheXNfYmczOnFhdGVtcEAsMjM=' 'https://<QualysURL>/csapi/v1.1/images?filter=vulnerabilities.severity%20%3A%205&pageNumber=0&pageSize=50&sort=created%3Adesc'
```

**Response:**

```
{
  "data": [
    {
      "eventOccurred": null,
      "created": "1526592592000",
      "author": null,
      "repo": [
        {
          "registry": "docker.io",
          "tag": "baddoc",
          "repository": "qualysdemo/checkoutapp"
        }
      ],
      "label": null,
      "layersCount": null,
      "uuid": "3fe02a98-36cf-317b-8e76-89a012c37235",
      "sha": null,
      "operatingSystem": null,
      "parentUuid": null,
      "sensorUuid": null,
      "customerUuid": null,
      "dockerVersion": null,
      "size": 108256183,
      "layers": null,
      "host": null,
      "vulnerabilities": {
        "severity5Count": 5,
        "severity3Count": 14,
        "severity4Count": 19,
        "severity1Count": 1,
        "severity2Count": 3
      }
    }
  ]
}
```
"event": null,
"architecture": null,
"imageId": "93c55587b0a5",
"associatedContainersCount": 4,
"associatedHostsCount": 1,
"lastScanned": "1531308873830",
"softwares": null
},
{
"eventOccurred": null,
"created": "1509562706000",
"author": null,
"repo": [
{
"registry": "docker.io",
"tag": "latest",
"repository": "imiell/bad-dockerfile"
},
{
"registry": "docker.io",
"tag": "baddockerfile",
"repository": "qualysdemo/checkoutapp"
}
],
"label": null,
"layersCount": null,
"uuid": "236f66f7-dc80-3e58-9a16-88ab987fc20f",
"sha": null,
"operatingSystem": null,
"parentUuid": null,
"sensorUuid": null,
"customerUuid": null,
"dockerVersion": null,
"size": 1082855961,
"layers": null,
"host": null,
"vulnerabilities": {
  "severity5Count": 5,
  "severity3Count": 14,
  "severity4Count": 19,
  "severity1Count": 1,
  "severity2Count": 3
},
"event": null,
"architecture": null,
Sample 5 - Create registry

/csapi/v1.1/registry

[POST]

Use this API to create a new registry.

Input Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>accountId</td>
<td>Provide the AWS account Id if your registry will be hosted on AWS. Parameters accountId, arn, and region are required when the registryType is AWS ECR and you want to create a new AWS connector.</td>
</tr>
<tr>
<td>arn</td>
<td>ARN number of the account ID. Specify the ARN if you want to use an existing AWS connector, or if you want to create a new connector.</td>
</tr>
<tr>
<td>region</td>
<td>Region where your AWS account belong to.</td>
</tr>
<tr>
<td>username</td>
<td>Username to connect to the registry. Should be in base64 format.</td>
</tr>
<tr>
<td>password</td>
<td>Password to connect to the registry. Should be in base64 format.</td>
</tr>
<tr>
<td>credentialType</td>
<td>None, Token, BasicAuth, DockerHub, AWS.</td>
</tr>
<tr>
<td>dockerHubOrgName</td>
<td>(Optional) Organization name if the registryType is DockerHub.</td>
</tr>
<tr>
<td>registryType</td>
<td>AWS ECR, DockerHub, Docker V2, Docker V2-Private.</td>
</tr>
<tr>
<td>registryUri</td>
<td>URL of the registry to connect to.</td>
</tr>
</tbody>
</table>

Input parameters can be provided in following format if you are using swagger:

```json
{
    "aws": {
        "accountId": "383031258652",
        "arn": "arn:aws:iam::383031258652:role/testabcd",
        "region": "us-east-2"
    }
}
```
Container Security APIs

API Samples

```
{
    "credentialType": "AWS",
    "registryType": "AWS",
    "registryUri": "https://383031258652.dkr.ecr.us-east-2.amazonaws.com"
}
```

**API request:**
```
```

**Response:**
```
{"registryUuid":"95b715e0-0fc7-4dac-b4de-2e1b92fc527d"}
```

**Sample 6 - Create AWS connector for registry**

/`csapi/v1.1/registry/aws/connector`

[POST]

Use this API to create a new aws connector for a registry.

**Input Parameters:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>arn</td>
<td>ARN number of the account ID.</td>
</tr>
<tr>
<td>externalId</td>
<td>The externalId of your organization.</td>
</tr>
<tr>
<td>name</td>
<td>Connector name.</td>
</tr>
<tr>
<td>description</td>
<td>Connector description.</td>
</tr>
</tbody>
</table>

Input parameters can be provided in following format if you are using swagger:

```
{
    "arn": "arn:aws:iam::205767712438:role/abcd",
    "externalId": "903805594",
    "name": "TestAWS",
    "description": "Testing of AWS account"
}
```

**API request:**
```
```
Response:
response code 200

Sample 7 - Delete sensors in your account
/csapi/v1.1/sensors

[DELETE]

Use this API to delete existing sensors in your account. You can only delete sensors with UNKNOWN status.

Input Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sensorDeleteRequest</td>
<td>(Required) user filters to query sensors or provide one or more sensor UUIDs to delete. Filter can be applied by providing a query using Qualys syntax. Refer to the &quot;How to Search&quot; topic in the online help for assistance with creating your query.</td>
</tr>
</tbody>
</table>

Input parameters can be provided in following format if you are using swagger:

```json
{
  "filter": "hostname:cms"
}
```

API request:

```bash
```

Response:

Returns {
  "deletionJobId": "bbaac4c7-6263-4e2f-b391-bcb032975206"
}

response code 200
Sample 8 - Delete images in your account
/csapi/v1.1/images

[DELETE]

Use this API to delete existing images in your account. Images with active containers (CREATED, RUNNING, STOPPED, PAUSED) associated with them, cannot be deleted.

Input Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>imageDeleteRequest</td>
<td>(Required) user filters to query images or provide one or more image UUIDs to delete. Filter can be applied by providing a query using Qualys syntax. Refer to the &quot;How to Search&quot; topic in the online help for assistance with creating your query.</td>
</tr>
</tbody>
</table>

Input parameters can be provided in following format if you are using swagger:

```
{
    "imageIds": [
        "e3e4cca0-8305-3835-810a-b334dcb65a33"
    ]
}
```

API request:

```
```

Response:

Returns "deletionJobId":"980ce235-5677-4997-81ca-3905e63471bb"

response code 200

Sample 9 - Delete multiple registries (bulk delete) in your account
/csapi/v1.1/registry

[DELETE]

Use this API to delete multiple registries in your account.

Input Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>registryIds</td>
<td>(Required) ID/UUIDs of the registries you want to delete. Should be in the form of an array. Note: You cannot delete registries whose schedules are in &quot;Running&quot; state.</td>
</tr>
</tbody>
</table>
Input parameters can be provided in following format if you are using swagger:

"fc129b85-e23c-4236-9fd2-47a257746208", "fe066970-0efc-4b04-91f4-b21870c61136"

API request:

curl -X DELETE --header 'Content-Type: application/json' --header 'Accept: text/plain' --header 'Authorization: Basic cXVhXNfO02ZwMTIz' -d '{"fc129b85-e23c-4236-9fd2-47a257746208", "fe066970-0efc-4b04-91f4-b21870c61136"}' 'https://qualysguard.qualys.com/csapi/v1.1/registry'

Response:

Returns \{"deletedRegistryUuids":\{"fc129b85-e23c-4236-9fd2-47a257746208","fe066970-0efc-4b04-91f4-b21870c61136\}\}

response code 200
Administration

Sensor updates

When an update is available you'll see "Update Available" next to the sensor name.

Container sensor update is otherwise automatic, however if you are currently using the beta version of the sensor you need to update to the latest sensor version manually. Automatic update kicks off once you are on a version higher than the beta.

To manually update the sensor from beta to the latest version, download the QualysContainerSensor.tar.xz file from Qualys Cloud Portal and then run the following commands directly from the screen on the docker host.

Untar the sensor package:

```
sudo tar -xvf QualysContainerSensor.tar.xz
```

Launch the new sensor:

```
sudo ./installsensor.sh ActivationId=5e7e422a-a1ca-403f-9274-506622dc5b28 CustomerId=a8cf7043-0245-6f1d-82f8-97f784652b93 Storage=/usr/local/qualys/sensor/data -s
```

Enter Y at the prompt asking you to upgrade 'Qualys-Container-Sensor' from version x.x.x to x.x.x.

The install script asks for proxy configuration. If you want to configure proxy, see Proxy Support.

Note: Once you have upgraded from the beta version to a higher version, future updates of Sensor are automatic.
How to uninstall sensor

The QualysContainerSensor.tar.xz file (which you download for sensor installation from Qualys Cloud Platform) has the script `uninstallsensor.sh` for uninstalling the sensor.

To uninstall a sensor:

If the docker host is configured to communicate over docker.sock, use the following command:

```
./ uninstallsensor.sh -s
```

If the docker host is configured to communicate over TCP socket then provide the address on which docker daemon is configured to listen:

```
./uninstallsensor.sh DockerHost=<IPv4 address or FQDN>:<Port#> -s
```

For example,

```
./uninstallsensor.sh DockerHost=10.115.27.54:3128 -s
```

Follow the on-screen prompts to uninstall the sensor. Qualys recommends not to clear the persistent storage.
Troubleshooting

Check sensor logs

The sensor log file is located at (by default):
/usr/local/qualys/sensor/data/logs/qpa.log

Diagnostic script

Qualys provides a script to collect diagnostic information about the sensor. You must run the script on the host on which you want to collect the diagnostic information from.

The diagnostic script is present in the QualysContainerSensor.tar.xz that you downloaded for installing the sensor.

The script is called Sensor_Diagnostic_Script.py. You must have Python installed on the host in order to run the script. The script collects the following information from the host and puts it in a tar file called SensorDiagnostic.tar. You can send that file to Qualys Support for further assistance.

The SensorDiagnostic.tar includes 'ScanInfo.json', 'qpa.log' of qualys-container-sensor from given persistent storage, docker logs of qualys-container-sensor, and all information described below in the 'SensorDiagnostic.log' file. If 'ScanInfo.json' and Sensor logs are not available on the Docker host then this script creates empty 'ScanInfo.json' and qpa.log files, and appends "File not found" to them.

- Operating System Information (Type of OS i.e. Linux or Mac and other details)
- Proxy Configuration (Type of proxy set e.g. system, docker, cloud-agent proxy)
- CPU Architecture (Details about model, CPUs, cores, etc)
- RAM Usage (Memory allocation and utilization on host)
- Docker Version (Docker version installed on host)
- Socket Configuration (Docker socket configuration on host e.g. TCP/unix domain)
- Number of docker images (Count of all docker images and their details)
- Number of docker containers (Count of all docker containers and their details)
- CPU and Memory usage of running containers (First result of all resource usage statistics)
Sensor crashes during upgrade

Use installsensor.sh to reinstall Qualys container sensor keeping the "Storage" value as it was for earlier Sensor. This will ensure that the new sensor will not be marked as another Sensor and will simply upgrade the existing one.

For help on install command, see Deploying Container Sensor.

Note: At any given point in time, DO NOT delete the persistent storage. Else, the sensor deployed thereafter will be marked as a new sensor.

What if sensor restarts?

The Sensor is designed to handle restart scenarios and will continue functioning normally after restart. No customer intervention is needed until the sensor crashes.

Note: The Qualys container sensor will fail to restart if it has exited due to a fatal error before the docker host/service restarts.

Known Issues and Limitations

Known limitations include Automated Registry Scanning, Deletion of Sensors from the pages, and raising of alerts for Events. These features are planned to be added soon.

The sensor only works for Docker and in Linux environments.
Appendix

Qualys Vulnerability Analysis Plugin for Jenkins
Qualys Vulnerability Analysis Plugin for Bamboo
Installing the sensor on a MAC
Deploying sensor in Kubernetes
Deploying sensor in Docker Swarm
Deploying sensor in AWS ECS Cluster
Deploying sensor in Mesosphere DC/OS

Qualys Vulnerability Analysis Plugin for Jenkins

Qualys Container Security provides a plugin for Jenkins to get the security posture for the docker images built via the tool. The plugin can be configured to fail or pass the docker image builds based on the vulnerabilities detected.

What you'll need
- A valid Qualys subscription with the Container Security application activated.
- Access to Qualys Container Security application API endpoint from your build host.
- Requires the container sensor for CI/CD environment to be installed on the jenkins build host. Refer to Deploying Container Sensor for instructions on installing the container cicc sensor. You must pass the following parameter while deploying the sensor for CI/CD environment --cicd-deployed-sensor or -c.

Jenkins plugin automatically tags images built out of CI/CD pipeline with the tag `qualys_scan_target:<image-id>` to mark them for scanning and only those images are scanned for vulnerabilities. Once the scanning is over, Qualys Container Sensor will remove the tag. However, if an image has no other tag applied to it other than 'qualys_scan_target:<image-id>', the sensor will retain the tag to avoid removal of the image from the host.

Install the Plugin

1) On Qualys Cloud Platform, go to Configurations > Integrations to download the Jenkins plugin.

2) Upload the plugin to your Jenkins tool. Go to Manage Jenkins > Manage Plugins > Advanced > Upload Plugin, then restart it.
Scanning CI/CD images
Configure the Jenkins plugin to automatically tag CI/CD images with 'qualys_scan_target:<image-id>'. This should be done as part of global configuration. Go to Manage Jenkins > Configure System, then scroll down to the Qualys Container Security section > Advanced, and provide the following details.

**Docker URL**: Docker REST API URL - Docker socket path. Only unix:/// and tcp:// protocols allowed.

**Cert File Path**: If you are using remote server enabled https, you can provide a specific folder location which contains the files ca.pem, cert.pem and key.pem. For example, /var/jenkins_home/certs.

Note: A Job Specific (local) configuration will use the Docker URL and Cert File Path configured in global configuration for tagging CI/CD images.

Start Using the Plugin
This plugin provides a build step and a post-build action. You can use it with pipeline type projects (for CI/CD pipeline) as well as freestyle projects. We'll describe both in the sections that follow.

- Pipeline Project
- Freestyle Project

**Pipeline Project**
With pipeline projects, you provide the docker image Id(s) to the plugin via a command argument. Use our Snippet Generator to generate this command, and then copy/paste it into your pipeline script (Jenkinsfile).

See How to configure?

**Freestyle Project**
For Freestyle projects you can use a POST-BUILD step provided by the plugin.

You'll provide the docker image Id(s) using the environment variable. Note – The variable name must be defined correctly or the plugin will not work. When Jenkins executes the post-build steps the plugin will only pull a report for the image Ids you've specified.

In Post-Build Actions, select “Get docker image vulnerabilities from Qualys”. This will open a form similar to the one shown for pipeline projects. Provide configuration details and test the connection to make sure it’s successful. See How to configure?
Somewhere in your build steps you must set the IMAGE_ID environment variable to the docker image IDs you want to report on. IMAGE_ID can be a single string value like 'a1b2c3d4e5f6' or a comma-separated list like 'a1b2c3d4e5f6,abcdef123456'.

**How to configure?**

You can either provide a global configuration or a job specific configuration. Global configuration can be set once and used for multiple projects: both Pipeline and Freestyle.

To set a global configuration, go to Manage Jenkins > Configure System, then scroll down to the Qualys Container Security section, and provide the configuration details listed below.

If you want to set a job specific configuration, in the Snippet Generator, select “getImageVulnsFromQualys - Get Image Vulns From Qualys”, and then select the Use Job Specific Configuration option. Note: Selecting the “Use Global(Jenkins) Configuration” option here will let the job use the global configuration you have set under Manage Jenkins > Configure System > Qualys Container Security.

**Define docker image IDs**

You’ll notice an argument called imageIds. Set this to the docker image IDs you want to report on. The plugin will only pull a report for the image IDs you specify.

Enter a single string value like imageIds: ‘a1b2c3d4e5f6’ or a comma-separated list like imageIds: ‘a1b2c3d4e5f6,abcdef123456’. You can also define docker image IDs in a variable and specify the variable as the value.

You can provide image ids through an environment variable. Get the image ids of the images programmatically created in earlier stages of the build and provide these ids in the 'imageIds' argument. For example, in pipeline script, you can get the image ids by executing shell script and store it in environment variable. And then use the same environment variable in 'ImageIds' argument to provide the image ids.

**Using the WebHook**

You can forward Jenkins job results to a WebHook URL.

You can set a global WebHook URL under Manage Jenkins > Configure System > Qualys Container Security, or a Job Specific WebHook URL under Snippet Generator by selecting “getImageVulnsFromQualys - Get Image Vulns From Qualys”, and then clicking Advanced.

Note: WebHook URL specified under Snippet Generator, for a particular project, will always take preference over the global WebHook URL specified under Manage Jenkins > Configure System > Qualys Container Security.
Configuration Details

Provide the following configuration details: (1) API login information (Select Use Proxy Settings to provide proxy information). (2) Click Test Connection to verify that the plugin can call the Qualys Container Security API. (3) data collection frequency. (4) build fail conditions. (5) forward Jenkins job results to a WebHook URL.

When you’re ready, click Generate Pipeline Script to get the script command.
If you are setting a global configuration, you can select a user from the Credential Store to authenticate to the API Server. In case of Job specific configuration, you can provide the credentials in the pipeline / freestyle script.

Note: Use global configuration for scanning images in CI/CD pipeline. See Scanning CI/CD images.

For information on what API Server URL to use, see Container Security APIs.

**View Your Qualys Report**

In either case - pipeline project or freestyle project - the plugin will generate one report for each docker image in the build. In other words, multiple image IDs will result in multiple report links.
Click any report link to see vulnerability details for the docker image.

The Build Summary provides a dashboard view of your security posture. Go to Vulnerabilities for a list of detected QIDs, Installed Software to see software detected on the docker image, and Layers to view a list of layers the image is made of.

**Debugging and Troubleshooting**

**HTTP codes in API response**

All API calls and their responses are logged by the plugin and are visible in the Console Output. Here are the HTTP response codes you may see during plugin execution.

<table>
<thead>
<tr>
<th>Code</th>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>204</td>
<td>No content</td>
<td>Qualys sensor is processing data. You’ll see 200 OK when complete.</td>
</tr>
<tr>
<td>200</td>
<td>OK</td>
<td>You would see this code in two situations: 1) You might have received partial data from Qualys where image details are available but vulnerability data is not available. 2) Vulnerability data is also available. This is usually the last call, after which the thread for that image Id stops.</td>
</tr>
<tr>
<td>500</td>
<td>Internal server error</td>
<td>Qualys service is down or there was an issue processing data.</td>
</tr>
<tr>
<td>400</td>
<td>Bad request</td>
<td>Qualys API server is unable to understand the request.</td>
</tr>
<tr>
<td>401</td>
<td>Unauthorized</td>
<td>The credentials used for Qualys API server are incorrect or the user does not have access to the APIs.</td>
</tr>
</tbody>
</table>
If you don't see any API calls being made...

Make sure you're correctly passing image Ids to the plugin. When the plugin starts the execution, it will print the image Ids provided and you can see this in the Console Output. Check that the docker image Ids you provided are printed.

Plugin times out, no report seen

The plugin is designed to keep polling the Qualys API until the configured timeout period is reached. If it does not get vulnerability data from Qualys within this period, it stops. In this case, the plugin will fail the build only if you have set any fail-on conditions. Otherwise, it will not fail the build. You will not see any report links since the plugin could not get vulnerability data, and could not prepare a report.
Qualys Vulnerability Analysis Plugin for Bamboo

Qualys Container Security provides a plugin for Bamboo to get the security posture for the docker images built via the tool. The plugin can be configured to fail or pass the docker image builds based on the vulnerabilities detected.

What you’ll need
- A valid Qualys subscription with the Container Security application activated.
- Access to Qualys Container Security application API endpoint from your build host.
- Requires the container sensor for CI/CD environment to be installed on the Bamboo build host. Refer to Deploying Container Sensor for instructions on installing the container cicd sensor. You must pass the following parameter while deploying the sensor for CI/CD environment --cicd-deployed-sensor or -c.
- Bamboo CICD tool version 5.14.0.1 or later.

Bamboo plugin automatically tags images built out of CI/CD pipeline with the tag qualys_scan_target:<image-id> to mark them for scanning and only those images are scanned for vulnerabilities. Once the scanning is over, Qualys Container Sensor will remove the tag. However, if an image has no other tag applied to it other than 'qualys_scan_target:<image-id>', the sensor will retain the tag to avoid removal of the image from the host.

Install the Plugin
1) On Qualys Cloud Platform, go to Configurations > Integrations to download the Bamboo plugin.
2) Upload the plugin to your Bamboo tool. Go to Administration > Add-ons and upload the Qualys bamboo plugin jar file.

Scanning CI/CD images
Configure the Bamboo plugin to automatically tag CI/CD images with 'qualys_scan_target:<image-id>'. This can be done as part of global or local configuration.

Docker URL: Docker REST API URL - Docker socket path. Only unix:/// and tcp:// protocols allowed.

Cert File Path: If you are using remote server enabled https, you can provide a specific folder location which contains the files ca.pem, cert.pem and key.pem. For example, /var/bamboo_home/certs.
Start Using the Plugin

You can use this plugin as a task in your bamboo plan. In the Tasks tab, click Add Task, and simply search for “Qualys” to get the Qualys Container Security addon you uploaded earlier. Click the Qualys addon to add it as a task.

You can either provide a **global configuration** or a **local configuration** for Qualys Container Security. Global configuration can be set once and used for multiple projects.

To set a global configuration, go to Administration > Add-ons, then in the left pane under ADD-ONS, find and click Qualys Container Security Plugin. Then, provide the configuration details listed below.

If you want to set a local configuration, in the Tasks tab, select Qualys Container Security, and then select the Configure locally option. Note: Selecting the “Use Global Config” option here will let the task use the global configuration you have set under Administration > Add-ons > Qualys Container Security Plugin.

See Configuration Details
Define docker image IDs

You’ll notice a field called Image IDs. This field is only available for local configuration. Set this to the docker image IDs you want to report on. The plugin will only pull a report for the image IDs you specify.

Enter a single string value like imageIds: 'a1b2c3d4e5f6' or a comma-separated list like imageIds: 'a1b2c3d4e5f6,abcdef123456'. You can also define docker image IDs in a variable and specify the variable as the value. Alternatively, you can inject bamboo variables using a task.

Using the WebHook

You can forward Bamboo job results to a WebHook URL.

You can set a global WebHook URL under Administration > Add-ons > Qualys Container Security, or a WebHook URL for local configuration in the Tasks tab for a plan, by selecting Qualys Container Security.

Note: WebHook URL specified under local configuration, for a particular project, will always take preference over the global WebHook URL specified under Administration > Add-ons > Qualys Container Security.
Appendix

Qualys Vulnerability Analysis Plugin for Bamboo

Configuration Details

Provide the following configuration details: (1) API login information (Select Use Proxy to provide proxy information). (2) Click Test Connection to verify that the plugin can call the Qualys Container Security API. (3) data collection frequency, and (4) build failure conditions. (5) forward Bamboo job results to a WebHook URL. When you’re ready, click Save Configuration.

For information on what API Server URL to use, see Container Security APIs.
View Your Qualys Report

The plugin will generate one report for each docker image in the build. Multiple image Ids will result in multiple reports. In a build, click the job which includes Qualys plugin, to see vulnerability details for the docker image.

Note: If case of multiple image Ids, the build fails even if one image Id matches the fail condition. The build summary will show details of the image which matched the fail condition.

The Build Summary provides a dashboard view of your security posture. Go to Vulnerabilities for a list of detected QIDs, Installed Software to see software detected on the docker image, and Layers to view a list of layers the image is made of.

Debugging and Troubleshooting

HTTP codes in API response

All API calls and their responses are logged by the plugin and are visible in the Console Output. Here are the HTTP response codes you may see during plugin execution.

<table>
<thead>
<tr>
<th>Code</th>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>204</td>
<td>No content</td>
<td>Qualys sensor is processing data. You’ll see 200 OK when complete.</td>
</tr>
<tr>
<td>200</td>
<td>OK</td>
<td>You would see this code in two situations: 1) You might have received partial data from Qualys where image details are available but vulnerability data is not available. 2) Vulnerability data is also available. This is usually the last call, after which the thread for that image Id stops.</td>
</tr>
<tr>
<td>500</td>
<td>Internal server error</td>
<td>Qualys service is down or there was an issue processing data.</td>
</tr>
</tbody>
</table>
If you don't see any API calls being made...

Make sure you're correctly passing image IDs to the plugin. When the plugin starts the execution, it will print the image IDs provided and you can see this in the Console Output. Check that the docker image IDs you provided are printed.

Plugin times out, no report seen

The plugin is designed to keep polling the Qualys API until the configured timeout period is reached. If it does not get vulnerability data from Qualys within this period, it stops. In this case, the plugin will fail the build only if you have set any fail-on conditions. Otherwise, it will not fail the build. You will not see any report links since the plugin could not get vulnerability data, and could not prepare a report.

<table>
<thead>
<tr>
<th>Code</th>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400</td>
<td>Bad request</td>
<td>Qualys API server is unable to understand the request.</td>
</tr>
<tr>
<td>401</td>
<td>Unauthorized</td>
<td>The credentials used for Qualys API server are incorrect or the user does not have access to the APIs.</td>
</tr>
</tbody>
</table>
**Installing the sensor on a MAC**

You can install the Qualys Container Sensor on a MAC.

Here are the steps:

Download the QualysContainerSensor.tar.xz file using the “Download and Install Qualys Container Sensor” link on the Get Started page or from the Configurations > Sensors tab on Qualys Cloud Platform.

Copy the file to the target MAC host.

Once you copy the file on the target host, run the following commands in sequence:

This command extracts the tar file.

```
sudo tar -xvf QualysContainerSensor.tar.xz
```

This command creates the directory where the sensor data like configuration, manifest, logs, and setup is stored.

```
sudo mkdir -p /tmp/qualys/sensor/data
```

This command provides the required permissions to the directory to run the installer script.

```
sudo chmod -R 777 /tmp/qualys/sensor/data
```

If you want to specify a custom location for storage, ensure that the Docker’s File Sharing is enabled for the same. On your MAC host, go to Docker > Preferences > File Sharing, add the custom path e.g. /usr/local/qualys/sensor/data, then click Apply & Restart.

Enabling file sharing is required only if the custom location is NOT from /Users, /Volumes, /private or /tmp.

To avoid this step, we recommend using Storage=/tmp/qualys/sensor/data and HostIdSearchDir=/private/etc/qualys during sensor install.

That way you can leverage the existing shared location with docker, without the need of additional configuration to launch the CS Sensor.
If you are using a custom location, provide permissions to the directory to run the installer script.

For example,

```
sudo chmod -R 777 /usr/local/qualys/sensor/data
```

The following commands install the sensor. Notice that the command includes the Activation ID and your Customer ID, both generated based on your subscription. The Storage parameter specifies where to install the sensor. Ensure that the HostIdSearchDir exists, otherwise the installer script will throw an error.

Use the following command to install a General Sensor:

```
./installsensor.sh ActivationId=d5814d5f-5fd2-44ec-8969-e03cc58a4ef5
CustomerId=6f35826e-4430-d75e-8356-c444a0abbb31
HostIdSearchDir=/private/etc/qualys Storage=/tmp/qualys/sensor/data -s
```

Use the following command to install a Registry Sensor:

```
./installsensor.sh ActivationId=d5814d5f-5fd2-44ec-8969-e03cc58a4ef5
CustomerId=6f35826e-4430-d75e-8356-c444a0abbb31
HostIdSearchDir=/private/etc/qualys Storage=/tmp/qualys/sensor/data -s -registry-sensor
```

Use the following command to install a CI/CD Sensor:

```
./installsensor.sh ActivationId=d5814d5f-5fd2-44ec-8969-e03cc58a4ef5
CustomerId=6f35826e-4430-d75e-8356-c444a0abbb31
HostIdSearchDir=/private/etc/qualys Storage=/tmp/qualys/sensor/data -s -cicd-deployed-sensor
```
Deploying sensor in Kubernetes

Integrate the Container Sensor into the DaemonSet like other application containers and set the replication factor to 1 to ensure there is always a sensor deployed on the Docker Host. This information is applicable for Amazon Elastic Container Service for Kubernetes (Amazon EKS), Google Kubernetes Engine (GKE), and Azure Kubernetes Service (AKS).

Perform the following steps for creating a DaemonSet for the Qualys sensor to be deployed in Kubernetes.

Download the **QualysContainerSensor.tar.xz** file from Qualys Cloud Portal on a Linux computer.

Untar the sensor package:

```
sudo tar -xvf QualysContainerSensor.tar.xz
```

Use the following commands to push the qualys sensor image to a repository common to all nodes in the Kubernetes cluster:

```
sudo docker load -i qualys-sensor.tar
```

```
sudo docker tag <IMAGE NAME/ID> <URL to push image to the repository>
```
For example,

```
sudo docker tag c3fa63a818df mycloudregistry.com/container-sensor:qualys-sensor-xxx
```

```
sudo docker push <URL to push image to the repository>
```
For example,

```
sudo docker push mycloudregistry.com/container-sensor:qualys-sensor-xxx
```

**Note:** Do not use the examples as it is. You need to replace the registry/image path with your own.

Modify the **cssensor-ds.yml** file (extracted from QualysContainerSensor.tar.xz) to provide values for the following parameters. In order for the yml file to work properly, ensure that you do not remove/comment the respective sections mentioned below.

Ensure that all Kubernetes nodes have the latest Qualys sensor image from the URL provided.

```
containers:
- name: qualys-container-sensor
  image: mycloudregistry.com/container-sensor:qualys-sensor-xxx
  args: ['--k8s-mode']
```

If you want to deploy the sensor for CI/CD environment provide the **args** value as:

```
args: ['--k8s-mode','--cicd-deployed-sensor']
```
If you want to deploy a Registry Sensor provide the **args** value as:

```bash
args: ["--k8s-mode","--registry-sensor"]
```

To restrict the cpu usage to a certain value, change the following: (Optional)

Under **resources** specify the following:

```yaml
limits:
  cpu: "0.2" # Default CPU usage limit(20% of overall CPU available) on each node for sensor.
```

For example,

For limiting the cpu usage to 5%, set `resources:limits:cpu: "0.05"`, this limits overall cpu usage to 5% of a CPU on a node.

If there are multiple processors on a node, set the `resources:limits:cpu` value accordingly.

For example,

You have 5 CPUs on system and you want to set 5% of overall capacity of system, set the CPU usage limit to 5 x 0.05 = "0.25".

To disable any CPU usage limit, set `resources:limits:cpu` value to 0.

Under **env** specify the following:

**Activation ID (Required)**

```yaml
- name: ACTIVATIONID
  value: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
```

**Customer ID (Required)**

```yaml
- name: CUSTOMERID
  value: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
```

Specify proxy information, or remove if not required:

```yaml
- name: qualys_https_proxy
  value: proxy.localnet.com:3128
```

Under **volumes** specify the proxy cert path, or remove if not required:

```yaml
- name: proxy-cert-path
  hostPath:
    path: /root/cert/proxy-certificate.crt
    type: File
```

Activation ID and Customer ID are required. Use the Activation ID and Customer ID from your subscription.
Appendix
Deploying sensor in Kubernetes

If you are using a proxy, ensure that all Kubernetes nodes have a valid certificate file for the sensor to communicate with the Container Management Server.

If you are not using a proxy and you have removed the above mentioned parts, you can remove the following part from **volumeMounts** as well:

```yaml
- mountPath: /etc/qualys/qpa/cert/custom-ca.crt
  name: proxy-cert-path
```

Once you have modified the **cssensor-ds.yml** file, run the following command on Kubernetes master to create a DaemonSet:

```bash
cubectl create -f cssensor-ds.yml
```

If you need to uninstall Qualys Container Sensor, run the following command on Kubernetes master:

```bash
cubectl delete -f cssensor-ds.yml
```
Deploying sensor in Docker Swarm

Integrate the Container Sensor into the DaemonSet like other application containers and set the replication factor to 1 to ensure there is always a sensor deployed on the Docker Host.

Perform the following steps for creating a DaemonSet for the Qualys sensor to be deployed in Docker Swarm.

Download the `QualysContainerSensor.tar.xz` file from Qualys Cloud Portal on a Linux computer.

Untar the sensor package:

```
sudo tar -xvf QualysContainerSensor.tar.xz
```

Use the following commands to push the qualys sensor image to a repository common to all nodes in the Docker Swarm cluster:

```
sudo docker load -i qualys-sensor.tar

sudo docker tag <IMAGE NAME/ID> <URL to push image to the repository>
```

For example,

```
sudo docker tag c3fa63a818df myregistry.com/qualys_sensor:xxx
```

```
sudo docker push <URL to push image to the repository>
```

For example,

```
sudo docker push myregistry.com/qualys_sensor:xxx
```

**Note:** Do not use the examples as it is. You need to replace the registry/image path with your own.

Modify the `cssensor-swarm-ds.yml` file (extracted from QualysContainerSensor.tar.xz) to provide values for the following parameters. In order for the yml file to work properly, ensure that you do not remove/comment the respective sections mentioned below.

Ensure that all masters and worker nodes have the latest Qualys sensor image from the URL provided.

```
qualys-container-sensor:
  image: myregistry.com/qualys_sensor:xxx
  deploy:
    mode: global  # Deploy 1 container on each node == DaemonSet
    command: ["--swrm-mode"]
```

If you want to deploy the sensor for CI/CD environment provide the `command` value as:

```
  command: ["--swrm-mode","--cicd-deployed-sensor"]
```
If you want to deploy a Registry Sensor provide the **command** value as:

```command: ["--swrm-mode","--registry-sensor"]```

To restrict the cpu usage to a certain value, change the following: (Optional)

Under **deploy** specify the following:

```mode: global  # Deploy 1 container on each node == DaemonSet
resources:
  limits:
    cpus: '0.20' # Default CPU usage limit(20% of overall CPU available) on each node for sensor.
```

For example,

For limiting the cpu usage to 5%, set `deploy:resources:limits:cpus: '0.05'`, this limits overall cpu usage to 5% of a CPU on a node.

If there are multiple processors on a node, set the `deploy:resources:limits:cpus` value accordingly.

For example,

You have 5 CPUs on system and want to set 5% of overall capacity of system, set the CPU usage limit to `5 x 0.05 = '0.25'`.

To disable any CPU usage limit, set `deploy:resources:limits:cpus` value to 0.

Under **environment** specify the following:

```environment:
  ACTIVATIONID: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
  CUSTOMERID: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
  qualys_https_proxy: proxy.qualys.com:3128
```

Activation ID and Customer ID are required. Use the Activation ID and Customer ID from your subscription. You can remove the proxy information if not required.

Under **volumes** ensure that you provide the following information:

```volumes:
  - type: bind
    source: /var/run/
    target: /var/run/
  - type: volume
    source: persistent-volume
    target: /usr/local/qualys/qpa/data/
  - type: bind
    source: /etc/qualys  # Must exist!
    target: /usr/local/qualys/qpa/data/conf/agent-data
```
Keep source as "persistent-volume". This ensures that the source directory in volume mapping is set to docker swarm root directory (i.e. /data/docker/volumes).

/etc/qualys directory must exist on all masters and worker nodes for successful volume mapping.

volumes:
  persistent-volume:

Under **configs** ensure that you provide the following information:

configs:
  proxy-cert-path:
    file: /root/cert/proxy-certificate.crt

If you are using a proxy, ensure that all masters and worker nodes have a valid certificate file for the sensor to communicate with the Container Management Server.

If you are not using a proxy and you have removed `qualys_https_proxy` from **environment**, you can remove the following parts as well:

configs:
  - source: proxy-cert-path
    target: /etc/qualys/qpa/cert/custom-ca.crt

configs:
  proxy-cert-path:
    file: /root/cert/proxy-certificate.crt

Once you have modified the **cssensor-swarm-ds.yml** file, run the following command on docker swarm master/leader to create a stack:

docker stack deploy -c cssensor-swarm-ds.yml qualys-container-sensor

If you need to uninstall Qualys Container Sensor, run the following command on docker swarm master/leader:

docker stack rm qualys-container-sensor
Deploying sensor in AWS ECS Cluster

Perform the following steps to deploy Qualys Container Sensor as a daemon service in Amazon ECS cluster.

**Prerequisites:** AWS ECS Cluster should be up and running.

Download the `QualysContainerSensor.tar.xz` file from Qualys Cloud Portal on a Linux computer.

Untar the sensor package:

```
sudo tar -xvf QualysContainerSensor.tar.xz
```

Use the following commands to push the qualys sensor image to a repository common to all nodes in the cluster:

```
sudo docker load -i qualys-sensor.tar
```

```
sudo docker tag <IMAGE NAME/ID> <URL to push image to the repository>
```
For example,

```
sudo docker tag c3fa63a818df 20576712438.dr.ecr.us-east-1.amazonaws.com/container-sensor:qualys-sensor-xxx
```

```
sudo docker push <URL to push image to the repository>
```
For example,

```
sudo docker push 20576712438.dr.ecr.us-east-1.amazonaws.com/container-sensor:qualys-sensor-xxx
```

**Note:** Do not use the examples as it is. You need to replace the registry/image path with your own.

Modify the `cssensor-aws-ecs.json` file (extracted from QualysContainerSensor.tar.xz) to provide values for the following parameters. In order for the json file to work properly, ensure that you do not remove/comment the respective sections mentioned below.

```
"containerDefinitions": [
  {
    "name": "qualys-container-sensor",
    "image": "20576712438.dr.ecr.us-east-1.amazonaws.com/container-sensor:qualys-sensor-xxx",
    "cpu": 10,
    "memory": 512,
    "essential": true,
    "command": [
      "--ecs-mode"
    ],
```
Specify appropriate values for **cpu** (no. of vcpu) and **memory** (size in MB).

If you want to deploy the sensor for CI/CD environment provide the **command** value as:

```json
"command": [
  "--ecs-mode",
  "--cicd-deployed-sensor"
],
```

If you want to deploy a Registry Sensor provide the **command** value as:

```json
"command": [
  "--ecs-mode",
  "--registry-sensor"
],
```

Under **environment** specify the following:

```json
"environment": [
  {
    "name": "ACTIVATIONID",
    "value": "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX"
  },
  {
    "name": "CUSTOMERID",
    "value": "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX"
  },
  {
    "name": "qualys_https_proxy",
    "value": "proxy.qualys.com:3128"
  }
]
```

Activation ID and Customer ID are required. Use the Activation ID and Customer ID from your subscription. Specify proxy information, or remove the section if not required. If you remove the proxy section, ensure that json indentation is correct.

If you are not using a proxy and you have removed **qualys_https_proxy** from **environment**, you can remove the following parts from **mountPoints** and **volumes**:

```yaml
configs:
- source: proxy-cert-path
  target: /etc/qualys/qpa/cert/custom-ca.crt

configs:
  proxy-cert-path:
    file: /root/cert/proxy-certificate.crt
```

If proxy section is removed from environment, then remove proxy-cert-path sections under mountPoints and volumes as well:
Appendix

Deploying sensor in AWS ECS Cluster

"mountPoints": [
  {
    "sourceVolume": "proxy-cert-path",
    "containerPath": "/etc/qualys/qpa/cert/custom-ca.crt"
  },

  "volumes": [
    {
      "name": "proxy-cert-path",
      "host": {
        "sourcePath": "/root/cert/proxy-certificate.crt"
      }
    }
  ]

Under volumes, provide information for persistent_volume. If you specify a custom location for persistent_volume, it would get created if not already available on the Docker Host.

Once you are done with the changes, save the cssensor-aws-ecs.json file.

**Import the json file into Amazon ECS UI to complete the sensor deployment**

On the Amazon ECS UI, under Task Definitions, click Create New Task Definition.

Select the launch type compatibility whether Fargate or EC2. Provide the Task Definition name, and then provide Task Role, Network Mode, and Task Execution Role if applicable.

Scroll to the bottom of the page and select Configure via JSON option. Remove any existing content and then copy-paste the entire contents of the cssensor-aws-ecs.json file.

Click Create to create the Task Definition. Once created, it should get listed under Task Definitions.

Now go to Clusters, and click the cluster name on which you want to deploy the sensor.

Under Services tab, click Create.

Select the launch type whether Fargate or EC2. Select the Task Definition you created above and its revision, and then select a cluster. Provide the Service name, Service type as “DAEMON”, and then configure Network, Load Balancing, and Auto Scaling if applicable.

Review the provided information, and then click Create to create the Service. Once created, it should get listed under Services.

Verify that the service status is Active. In the tasks tab, verify that tasks are running on all ECS containers.
Stopping Qualys sensor on Amazon ECS Cluster

If you want to stop the Qualys container sensor from running on all containers, simply delete the service from the Services tab. This will kill the qualys-container-sensor service, but will not remove the sensor from the AWS ECS instances.
Deploying sensor in Mesosphere DC/OS

Perform the following steps to deploy Qualys Container Sensor as an application in DC/OS Marathon.

**Prerequisites:** A running DC/OS cluster with the DC/OS CLI installed.

Download the `QualysContainerSensor.tar.xz` file from Qualys Cloud Portal on DC/OS master.

Untar the sensor package:

```
sudo tar -xvf QualysContainerSensor.tar.xz
```

Use the following commands to push the qualys sensor image to a repository common to all nodes in the cluster:

```
sudo docker load -i qualys-sensor.tar

sudo docker tag <IMAGE NAME/ID> <URL to push image to the repository>
For example,
sudo docker tag c3fa63a818df myregistry.com/qualys_sensor:xxx

sudo docker push <URL to push image to the repository>
For example,
sudo docker push myregistry.com/qualys_sensor:xxx
```

**Note:** Do not use the examples as it is. You need to replace the registry/image path with your own.

Modify the `cssensor-dcos.json` file (extracted from QualysContainerSensor.tar.xz) to provide values for the following parameters. In order for the json file to work properly, ensure that you do not remove/comment the respective sections mentioned below.

```
"id": "/qualys-container-sensor",
"args": ["--dcos-mode"],
"cpus": 1,
"mem": 128,
"disk": 0,
"instances": 1,
"acceptedResourceRoles": ["*"],
```

Specify appropriate values for **cpus** (no. of vcpu), **mem** (size in MiB) and **disk** (size in MiB).

If you want to deploy the sensor for CI/CD environment provide the **args** value as:

```
"args": ["--dcos-mode","--cicd-deployed-sensor"],
```
If you want to deploy a Registry Sensor provide the `args` value as:

```json
"args": ["--dcos-mode","--registry-sensor"],
```

Ensure that `instances` value is the number of nodes in the cluster. This ensures that the container Sensor runs on each cluster node.

```json
"container": {
    "type": "DOCKER",
    "docker": {
        "forcePullImage": true,
        "image": "myregistry.com/qualys_sensor:xxx",
        "parameters": [],
        "privileged": false
    }
},
```

Under `env` specify the following:

```json
"env": {
    "ACTIVATIONID": "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX",
    "CUSTOMERID": "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX",
    "qualys_https_proxy": "proxy.qualys.com:3128"
},
```

Activation ID and Customer ID are required. Use the Activation ID and Customer ID from your subscription. If you are using a proxy, ensure that all nodes have a valid certificate file for the sensor to communicate with the Container Management Server.

Under `volumes` specify the following:

```json
"volumes": [
    {
        "containerPath": "/usr/local/qualys/qpa/data",
        "hostPath": "/usr/local/qualys/sensor/data",
        "mode": "RW"
    },
    {
        "containerPath": "/var/run",
        "hostPath": "/var/run",
        "mode": "RW"
    },
    {
        "containerPath": "/usr/local/qualys/qpa/data/conf/agent-data",
        "hostPath": "/etc/qualys",
        "mode": "RW"
    },
    {
        "containerPath": "/etc/qualys/qpa/cert/custom-ca.crt",
```
The directories specified for the `hostPath` are automatically created if not already available on the nodes. Ensure to provide a valid `proxy-certificate.crt` file path if you want to deploy the Sensor using a proxy.

If you are not using a proxy and you have removed `qualys_https_proxy` from `env`, you can remove the following from `volumes` as well, while ensuring that json indentation is correct:

```json
{
    "containerPath": "/etc/qualys/qpa/cert/custom-ca.crt",
    "hostPath": "/root/cert/proxy-certificate.crt",
    "mode": "RO"
}
```

Under `portDefinitions` specify the following:

```json
"portDefinitions": [
    {
        "port": 10000,
        "protocol": "tcp"
    }
]
```

Specify a valid port number. Replace port number 10000, if already in use.

Once you have modified the `cssensor-dcos.json` file, run the following command on DC/OS master to add the qualys-container-sensor application to Marathon:

```
dcos marathon app add cssensor-dcos.json
```

Use this command to verify that the application is added successfully:

```
dcos marathon app list
```

If you need to uninstall Qualys Container Sensor from Marathon, run the following command on DC/OS master:

```
dcos marathon app remove --force /qualys-container-sensor
```