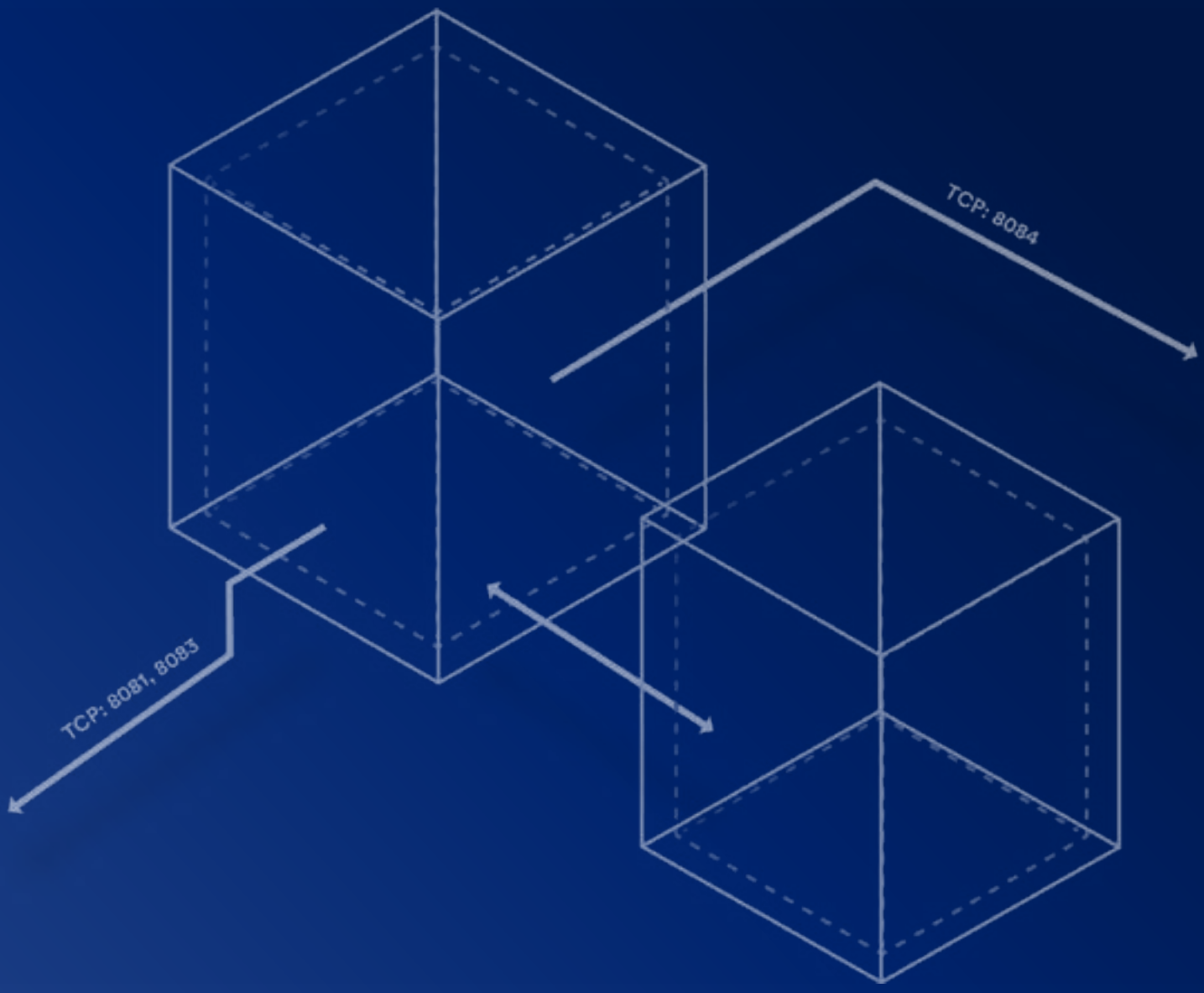
Twistlock
A PALO ALTO NETWORKS® COMPANY

Twistlock Reference Architecture 19.07



Contents

Twistlock
A PALO ALTO NETWORKS® COMPANY

Reference Architecture Objectives

The Twistlock Reference Architecture provides guidance to Enterprise and Security Architects on how to deploy Twistlock and integrate with systems commonly found in the enterprise stack and across the elements of their cloud workloads.

Solution Overview

Twistlock protects your cloud native assets anywhere they operate—whether you're running containers, serverless functions, non-container hosts, or any combination of them. Advanced threat intelligence and machine learning enable protection of your entire cloud native stack, whether it runs in the public cloud, private cloud, or air-gapped environment.

Twistlock provides an agentless architecture that requires no changes to your host, container engine, or applications. Twistlock is deployed as a set of containers, as a service on your hosts, or as a runtime component of your serverless function. For environments that do not support deployment of Twistlock as a privileged peer, we offer runtime application self protection (RASP) capabilities.

Upon deployment, Twistlock immediately begins working to secure your container and cloud environment. Twistlock supports discovery of assets within your cloud environment, allowing you to easily identify assets which are not protected and add them.

Twistlock is easily integrated into your container build process with support for continuous integration (CI) systems and registry/serverless repository scanning capabilities.

Twistlock Host, Container, Virtual Machine, and Serverless Function Support

Twistlock supports the full stack and lifecycle of your cloud native workloads. With Twistlock, you can protect mixed workload environments. Whether you're running standalone hosts, containers, serverless functions, or any combination of the above, Twistlock allows you to manage your environment with a single interface across the entirety of the lifecycle — from development to runtime.

Twistlock protects the hosts you're working with, whether you are using a Linux variant or using Windows Server 2019, and the applications they run.

Twistlock can also protect your serverless functions and applications on AWS Fargate. As you look to move beyond using containers and identify workloads that can be run as functions, Twistlock can protect and report on these functions alongside your other workloads regardless of cloud service provider. Finally, Twistlock supports the Docker and OCI compatible container runtimes, as well as any functions you may run across any platform.

Twistlock Platform Components

Twistlock Console

Twistlock Console serves as the user interface within Twistlock. The graphical user interface (GUI) lets you define policy, configure and control your Twistlock deployment, and view the overall health (from a security perspective) of your container environment. Console also provides an API for customers that want to control Twistlock programmatically to build out their own integrations or custom tooling. The API is thoroughly documented. Endpoints are provided for all features, functions, and controls offered in the GUI.

Regardless of how Console is installed and where it operates, Console requires access to persistent storage. Console can be deployed using either your orchestrator's native HA capabilities or Twistlock's built-in high availability (HA) capabilities.

When installing Twistlock, install Console first, then install Defender. Defender is the component of Twistlock that runs on each host, more detail is provided below. Defender can be installed from the deployment tabs in Console's graphical user interface. Defender, as the initiator of the connection, requires network connectivity to the Console.

Twistlock provides automation in the product that generates the required artifacts for common orchestration platforms such as Kubernetes, OpenShift and Swarm. Twistlock can also generate Helm charts to ease deployment for organizations who have adopted Helm as their packaging standard.

Graphical Interface

Console's graphical user interface can be accessed using a web browser on ports 8081 (HTTP) or 8083 (HTTPS). We recommend that you access Console over HTTPS so that sensitive information, such as admin passwords, is encrypted while in transit. By default, self-signed certificates are used to secure access to Console but you can configure your own certificate to prove your server's identity to browser clients. For more information on this topic, see our support article here.

API

The Twistlock API is REST-based and can be accessed over HTTP or HTTPS on ports 8081 and 8083 respectively. For more information about the Twistlock API, see the support article here.

Twistlock Defender

Twistlock Defenders enforce the policies defined in Console and send event data up to the Console for correlation. There are several types of Defenders, and depending on the assets in your environment that require protection you may end up deploying all of them or a subset. Defenders support the full variety of workloads in cloud native environments:

- Container Defender: This Defender type is deployed as a container on every asset running containers in your infrastructure.
- Host Defender: This Defender type is deployed for Virtual Machines that do not run containers.
- Fargate Defender: This Defender type deploys as part of your Fargate deployment.
- Serverless Defender: This Defender type deploys as part of your serverless function.

The Defender can be installed one of the following ways:

- One at a time, on each host that you want to protect. Use this method when you're not using an orchestrator, or for simple proof-of-concept environments. You can also install Defenders via whatever configuration management or automation tools you are already using, Ansible, Puppet, or Chef for example.
- As a orchestrator-native construct. For example, you can deploy Defender as a DaemonSet in Kubernetes and OpenShift environments or as a global service in Docker Swarm environments. Orchestrator-native constructs ensure that Defender is automatically deployed to every node in the cluster, even as the cluster dynamically scales up or down.
- As a systemd service on hosts that do not have Docker.
- As a windows system service on hosts that do not have Docker.
- As a part of your Fargate deployment or Serverless function deployment.

By default, Defender establishes a connection to Console on TCP port 8084 but you can customize the port to meet the needs of your environment. All traffic between the Defender and the console is TLS encrypted.

To use Twistlock registry scanning capabilities, different container Defenders in your environment can be designated to scan each registry, allowing you to balance registry scanning based on geographic or other concerns; container based Defenders can simultaneously protect its host and scan registry images. These Defenders must be able to connect to the registries over the network, and the type (Linux or Windows) must match the kind of images you want it to scan. If you have a hybrid Linux and Windows environment, one Defender of each type must be running.

As with Console, Twistlock provides automation in the product to generate the artifacts required to deploy Defender across a variety of environments.

Twistlock Intelligence Stream

The Twistlock Intelligence Stream is a real-time threat feed delivered from the Twistlock content delivery network (CDN) to our customers' installations. This service gathers, analyzes, and prepares threat data for distribution to the Console located on your network. Console pulls data down from the Threat Intelligence Stream using HTTPS requests. The Intelligence Stream is Console's only required external dependency.

Note: No customer data ever leaves your network or environment. Twistlock does not gather data from our customers unless you choose to opt in and contribute.

If you operate in an air-gapped environment, data in the Intelligence Stream can be downloaded and transferred to Console using the *twistcli* tool we ship with the product using whatever operational processes you wish to put in place.

The twistcli Tool

The twistcli tool is a command-line control and configuration tool. It ships with your Twistlock release and can be found in the Twistlock release tarball. Support is provided for both Linux and MacOS.

The twistcli tool provides a number of functions:

- Scanning images for vulnerabilities and compliance issues. This is useful when you're building custom tooling, or when you're using a CI tool for which Twistlock does not provide a native plugin.
- Deploying (installing and uninstalling) Console and Defender across all environments.
- Downloading the latest threat data from the Intelligence Stream for transfer to an air-gapped environment.
- Packaging log files and other relevant data from your environment and optionally uploading that data so that Twistlock Support can help debug issues.
- Interacting with Serverless and Fargate artifacts to automatically produce the artifacts necessary to run workloads in Serverless or Fargate.

For more information about twistcli, see the support article here.

Authenticated Registry

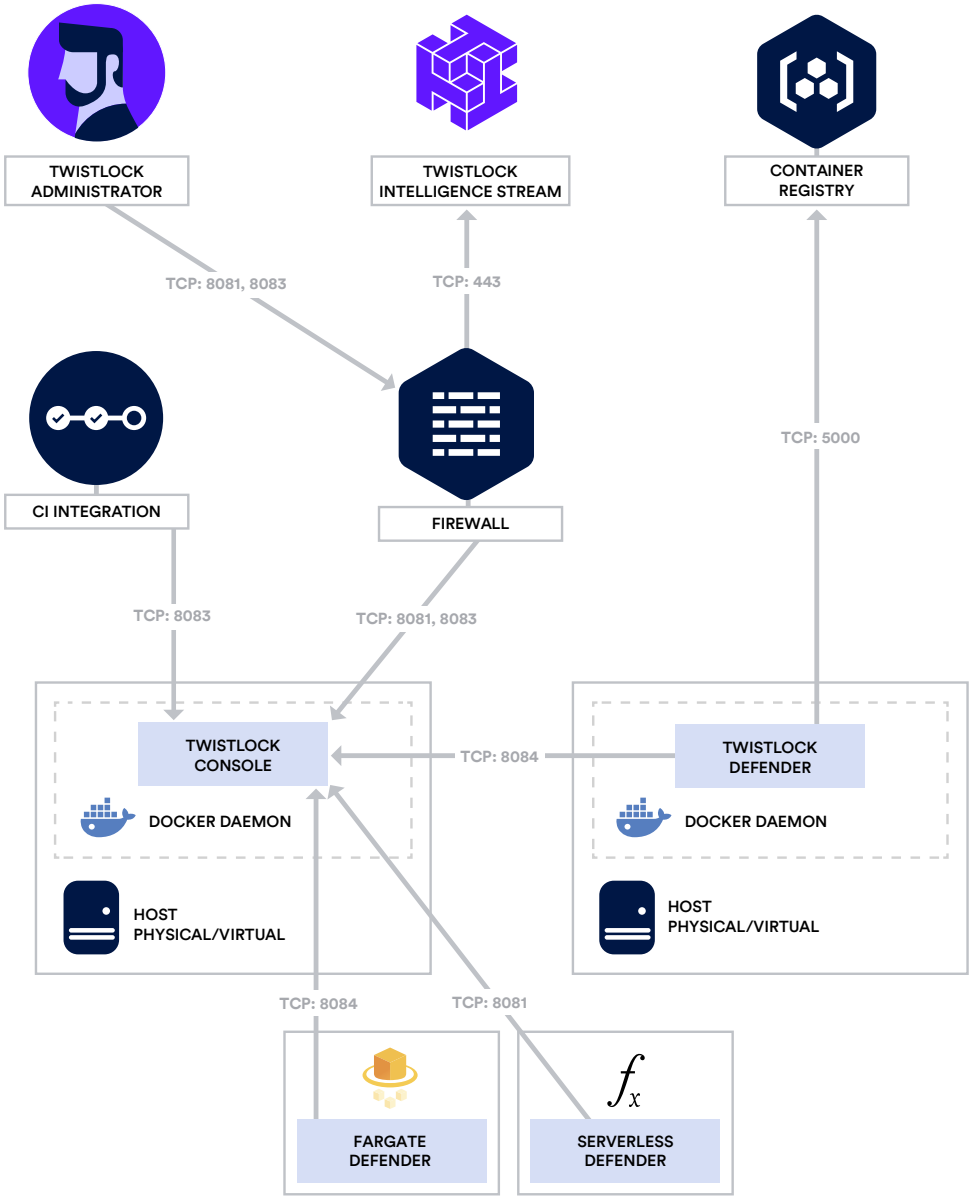
For our customers, Twistlock hosts the images required for deployment of Twistlock in a secured and highly available container registry they can leverage to automate their deployments. Use of this registry is not required and is provided as a convenience to our customers.

Twistlock Connectivity Flows

In order to operate, Twistlock requires a number of connections between components. The following diagram shows the ports and connections required. To fit the needs of various customer environments and deployments, all ports are configurable at install time (for more information, see the inline comments in twistlock.cfg).

Firewalls

Customers typically place Console in a management security zone or other segregated part of their network. Some customers might also want to place a firewall between Console and Defender. Twistlock can interoperate with firewalls wherever necessary, provided the required TCP ports are open.



Istio

When Defender DaemonSets are deployed with Istio monitoring enabled, Twistlock can discover the service mesh and show you the RBAC capabilities for each service (e.g. this pod can read service X using REST/grpc on the following endpoints). Services integrated with Istio display the Istio logo.

Load Balancers

A common configuration involves placing a load balancer in front of Console for access to the GUI and the API. Twistlock can interoperate with traditional hardware or software load balancers, as well as load balancers from all major cloud service providers.

High Availability

Console can be deployed in an HA configuration using either your orchestrator's HA capabilities or Twistlock built-in HA capability. Twistlock built-in HA capability are provided for customers that want to run Console on hosts that are not under the control of a cluster manager.

In general, we recommend that you use your orchestrator's native availability features. If you are using an orchestrator for HA, do not enable Console's built-in HA capabilities.

When you deploy Console on multiple hosts, Twistlock built-in HA ensures that one Console is always available, even if a host fails. Twistlock HA creates a high availability clusters from redundant hosts running Console in an active-passive configuration, and automatically manages cluster membership, leader election, and failover.

To learn about Twistlock's built-in HA capabilities for Console, see here.

Operational Concerns

System Requirements

Before deploying Twistlock Console, Defenders, and registry scanners, be sure that your hosts meet the minimum requirements detailed here.

Monitoring

Twistlock provides API endpoints to monitor the health and availability of deployed components.

We recommend you monitor the following aspects of your Twistlock installation:

Console Service Availability:

Monitor the Twistlock API and ensure the ping API returns “200 ok”

- API endpoint: GET /api/v1/_ping
- Example command: curl -u admin:Password ‘https:<console-ip>:8083/api/v1/_ping

Intelligence Stream Connectivity:

The Intelligence Stream is used to pull down threat and CVE data. From the Console or host, monitor the following:

- https://intelligence.twistlock.com/api/v1/_ping

Disk Space:

Console writes files to disk both database and logging. We recommend customers monitor the health and space available on the volume that contains /var/lib/twistlock. Typical customers with normal usage should raise an alert when 10 GB or less of disk space is available on the volume. This location can be modified in twistlock.cfg at install time.

Both Console and Defender containers are configured with Docker Health Check, a best practice, which gives you insight into the containers’ health.

Backup

Twistlock includes full backup and restore functionality in the Console. Twistlock automatically backs up on a Daily, Weekly, and Monthly cadence.

Supported Schedulers and Deployment Patterns

You can either download and manage all Twistlock container images by yourself or you can access them from our hosted registry.

When you download our software from the Releases page, you get a tarball that can be used to install Twistlock. You can also pull the images from the Twistlock Registry, for more information please refer to the documentation.

You can push the Twistlock images to your own private registry, and manage them as you see fit. The Console image is delivered as a .tar.gz file in the release tarball. After Console is installed, the Defender image is accessible from the dashboard under Manage > Defenders > Deploy, where deployment scripts retrieve the Defender image from Console using the /api/v1/images/twistlock_defender.tar.gz API endpoint.

You can also retrieve Twistlock images from our hosted registry, which is available to all current customers with a valid access token. This option simplifies a lot of workflows, especially the initial install flow.

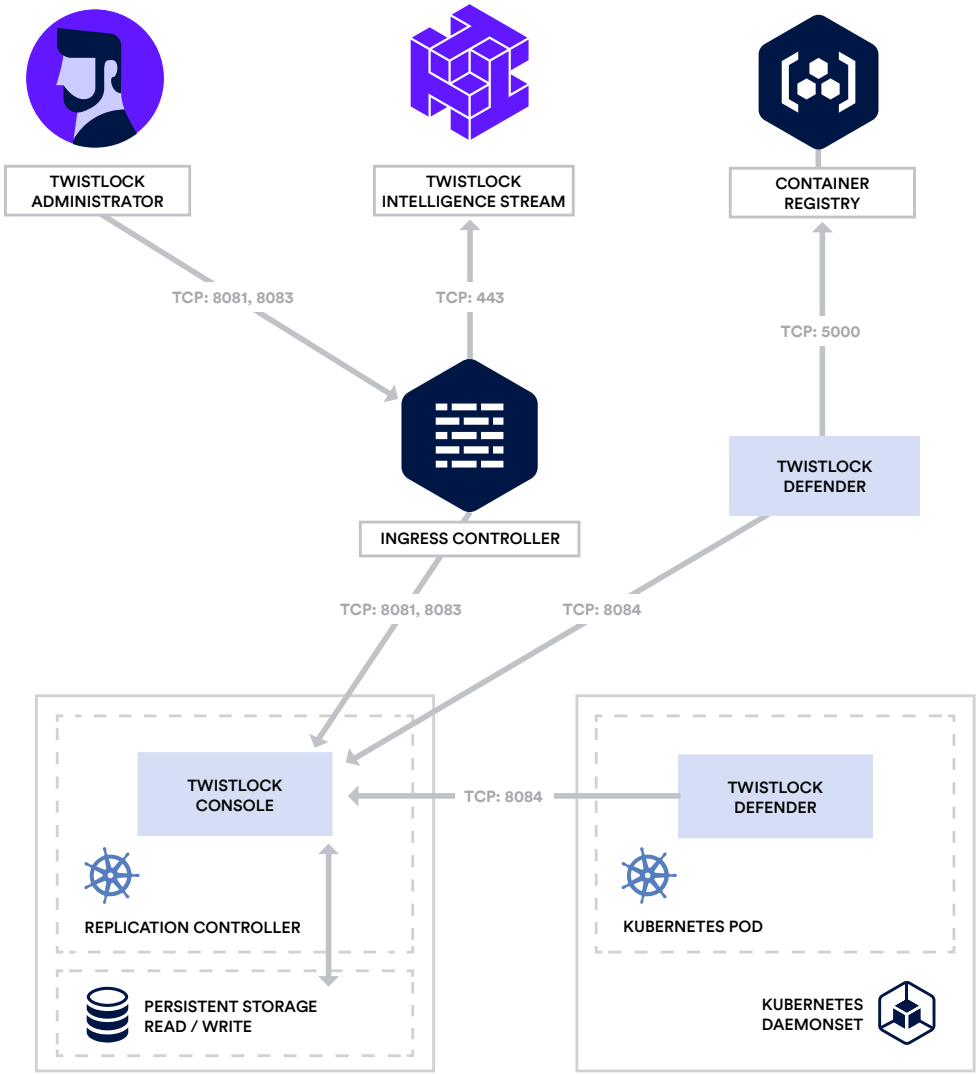
Twistlock on Kubernetes

Twistlock supports deploying Console and Defenders into Kubernetes clusters.

The Twistlock Console is installed as a replication controller with persistent storage, allowing the Console to be resilient to node failures.

Defenders are deployed to Kubernetes nodes using DaemonSets. DaemonSets make Defender deployment simple and automatic, regardless of how large your cluster or how frequently you add nodes to it. With DaemonSets, rather than manually installing Twistlock Defenders on each node, Twistlock generates a configuration file that you load into your Kubernetes Master. Kubernetes uses the configuration to ensure that every node in the cluster runs a Defender. As new nodes are added, Defenders are automatically installed on them. Deploying Defenders with DaemonSets guarantees that every node in your environment is protected, without having to manually intervene when node membership changes.

The diagram below illustrates a basic Twistlock deployment on Kubernetes:



Notes on Installing Console

Before installing Twistlock Console as a pod, your Kubernetes cluster should be built out and persistent storage should be provisioned and formatted. As part of the install process, you'll need to update twistlock.cfg with the specifics of your environment. Complete instructions can be found on the Twistlock Support Site here. We recommend that you use a namespace other than "twistlock" when deploying Console.

Notes on Installing Defender as a DaemonSet

When installing the Twistlock Defender as a DaemonSet, we recommend you use the twistcli to generate a daemonset.yaml as described on the Support Site here.

Twistlock on Amazon EKS

You can install Twistlock Console as a ReplicationController and Defenders as a DaemonSet in EKS. More details on this configuration can be found in our Support Site here.

Twistlock on Amazon ECS

Twistlock supports deploying Console and Defenders into an ECS cluster.

In an ECS cluster deployment, Console is deployed directly using a task definition. The Twistlock Console requires persistent storage to store its data. ECS does not have a way to dynamically attach storage to a node that is running a particular task, so deploying Console requires attaching an EFS mount to each node in the cluster. Console runs on a single node at a time, and ECS can schedule it to run on any available node, so a shared data store is recommended.

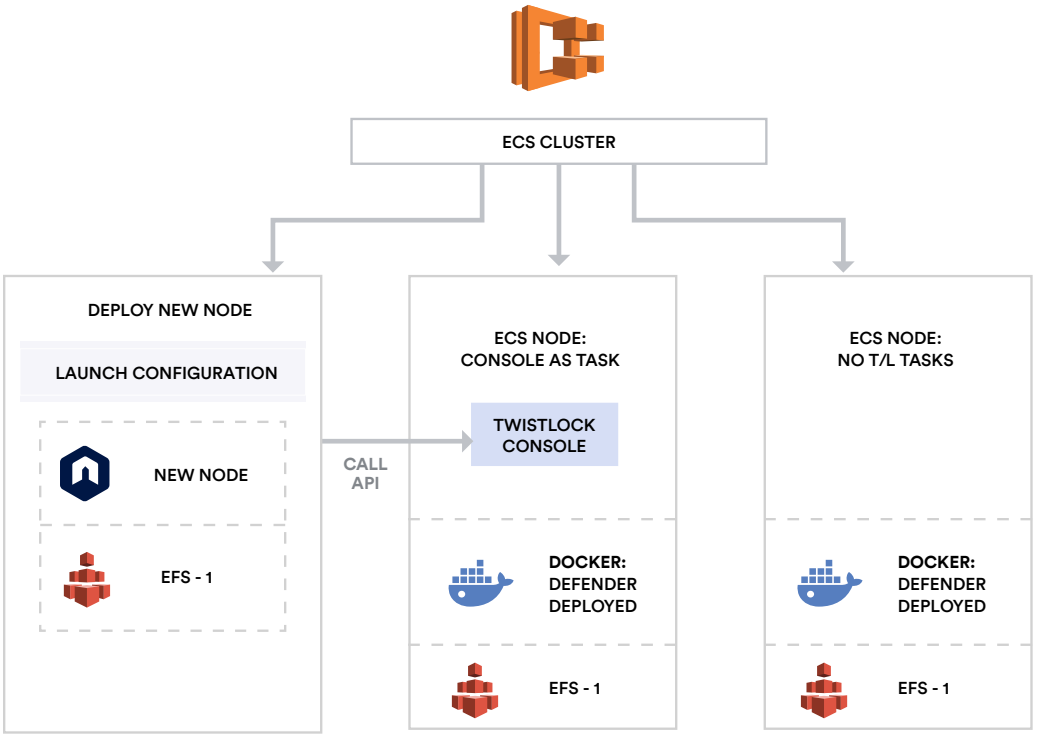
Defenders are not deployed as tasks due to a few limitations in the parameters available to define ECS tasks. Ideally, you want a Defender to run on each node in your cluster. In order to accomplish this, you deploy Defenders as part of an AWS launch configuration that is used by the ECS cluster to provision new nodes. In the launch configuration, you define a user data script that calls our API to install Defender.

Due to Host PID limitations imposed by Amazon, ECS Daemon Scheduling is not currently supported.

The final piece is to set up an ELB to forward traffic to the Console node and give the Defenders a static endpoint they can connect to. This should be configured as a standard ELB with TCP forwarding for port 8084. Additionally you will want to set up a health check so the ELB knows where to forward traffic. Because this is a TCP forward, there is no health check that can be performed on the websocket so we will configure a health check to call our HTTPS API endpoint and ping /api/v1/_ping. Only the node that is running the Console will be respond to the health check and the ELB will forward all traffic to the console node.

The diagram below illustrates the layers each component runs in and how a new node

is provisioned with a Defender:

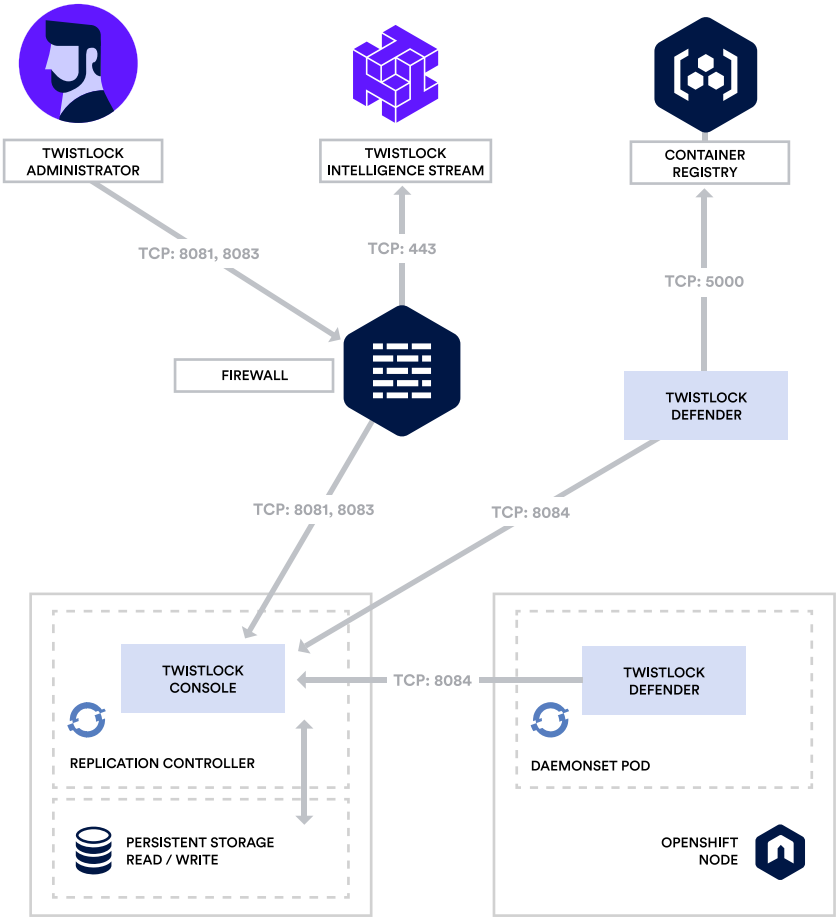


Twistlock on OpenShift

Twistlock supports deploying Console and Defenders into an OpenShift cluster.

OpenShift makes Defender deployment simple and automatic, regardless of how large your cluster is or how frequently you add nodes to it. With DaemonSet pods, rather than installing Twistlock Defenders on each node individually, Twistlock generates a configuration file that you load into your OpenShift master. OpenShift uses this configuration to ensures that every node in the cluster runs a Defender. As new nodes are added, Defenders are installed on them as well. Deploying Defenders with DaemonSets guarantees that every node in your environment is protected, without having to manually intervene when membership changes.

The diagram below illustrates a basic Twistlock deployment on OpenShift:



Notes on Console as a Replication Controller

Before deploying Twistlock Console to a Replication Controller, configure and format your persistent storage. Note the hostpath and labels used when defining the persistent storage YAML file. Both these values are required when using twistcli and twistlock.cfg to generate the Console YAML file.

If you are installing the Twistlock Console and Defender into different namespaces, specify Services and Routes for TCP ports 8081, 8083, and 8084 between Console and DaemonSet pods.

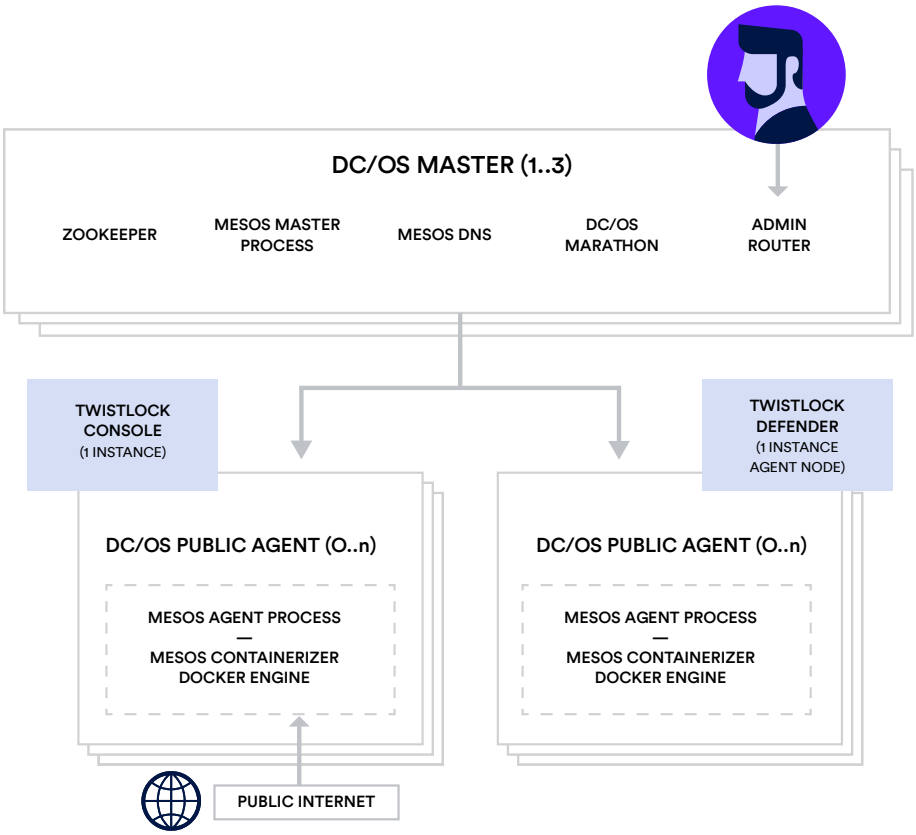
Twistlock on DC/OS

Twistlock supports deploying Console and Defender to a DC/OS or Mesos environment, using either the Marathon or Kubernetes scheduler. Twistlock recommends Kubernetes, please refer to the section on Kubernetes in this document.

The deployments for all environments are essentially the same. Console runs as a standard Docker container on one of your hosts, and a Defender instance runs on each agent node.

Before installation, load the Console and Defenders images, tag them, and then push them to a registry that can be accessed during the deployment.

The diagram below illustrates a basic Twistlock deployment on DC/OS:



Notes on Installing Console

We recommend that you create network mountable durable storage for Console's data. Mount this storage on any host that might run Console. This way, Console has access to its data no matter where it is deployed. We recommended that you use /var/lib/twistlock as the mount point on the host.

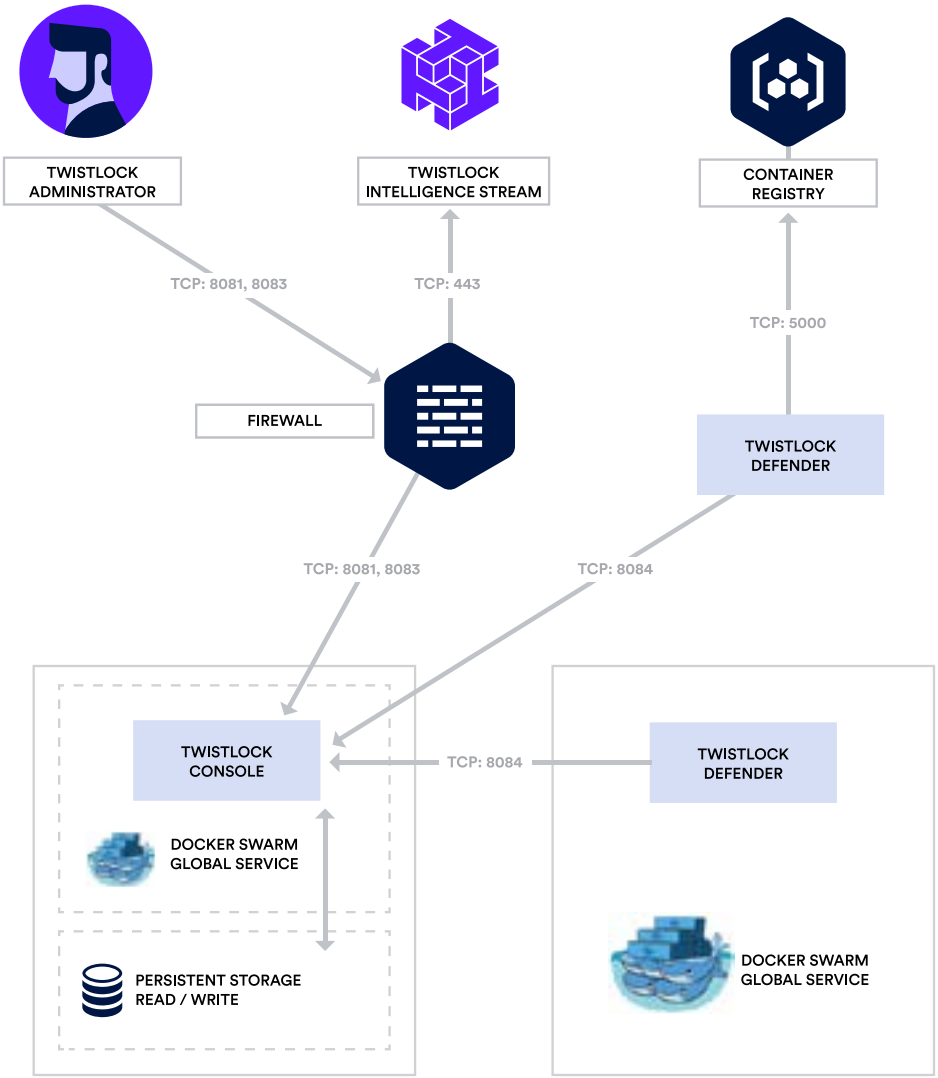
Notes on Installing Defender

Twistlock Defenders are deployed to each agent node using Marathon's application construct. Marathon applications are defined in JSON. Before loading the Defender application, update Defender image with the certificates it needs to securely communicate with Console. To do this, load the Defender image, open an interactive shell to a running instance of the Defender image, install curl into the container, then create and populate the directory that holds the certs. For complete details, see the support article here.

Twistlock on Swarm

Twistlock supports installation on Docker Swarm using Swarm-native features. You deploy Console as a service with the number of replicas limited to 1 and rely on Swarm to provide built-in high availability. You also deploy Defender as a global service. The global service guarantees that Defender is automatically deployed to each worker node in the cluster.

The diagram below illustrates a basic Twistlock deployment on Docker Swarm:



Notes on Deploying Console in Swarm

You'll use the twistcli tool provided in the install package to install Console, which will push the console image to your registry. Console uses Swarm's routing mesh networking, so the Console service is available on the target port on every node. More detail on installing Console in Swarm can be found here.

Twistlock on Pivotal Cloud Foundry: PAS and PKS

Twistlock can be used to secure applications on both Pivotal Application Service (PAS) and Pivotal Container Service (PKS).

Twistlock on Pivotal Application Service (PAS)

The PCF Defender is delivered as a tile. Initially, Twistlock supports scanning droplets and blobstores. To use Twistlock on PAS, go to the PCF Ops Manager Installation Dashboard to install the tile.

For more complete details, see the support article on PAS here.

Twistlock on Pivotal Container Service (PKS)

PKS provides on-demand deployment of Kubernetes clusters with the command line (CLI) and API using BOSH. Twistlock supports the deployment of both Console and Defender containers on your PKS nodes.

For more complete details, see the support article on PKS here.

Automating Twistlock Installs for Other Environments

To enable deployment across environments we host our Twistlock images in a repository designed for reliability and availability. This repository is accessible using your Access Token supplied as part of your license. More information on accessing our hosted registry is available here.

Details on deployment will vary according to the environment but the high level steps are much like the other deployments above.

The Console image can be pulled from Twistlock. It can be started on your platform with persistent storage, appropriate network connectivity and configured using the Web UI.

Defender deployments will vary but as the Defender is usually pulled from the Console it's as simple as calling the API. Detailed instructions can be found here.

Twistlock
A PALO ALTO NETWORKS® COMPANY

Cloud Discovery and Service Account Monitoring

As cloud platforms continue to add new services, it's becoming more difficult and impractical to ensure the apps running on them are protected. Consider that you might be using multiple cloud platforms, and that you have many separate accounts per platform, such as different accounts per business unit or geography. You could easily have hundreds of combinations of providers, accounts, and regions where cloud native services are deployed.

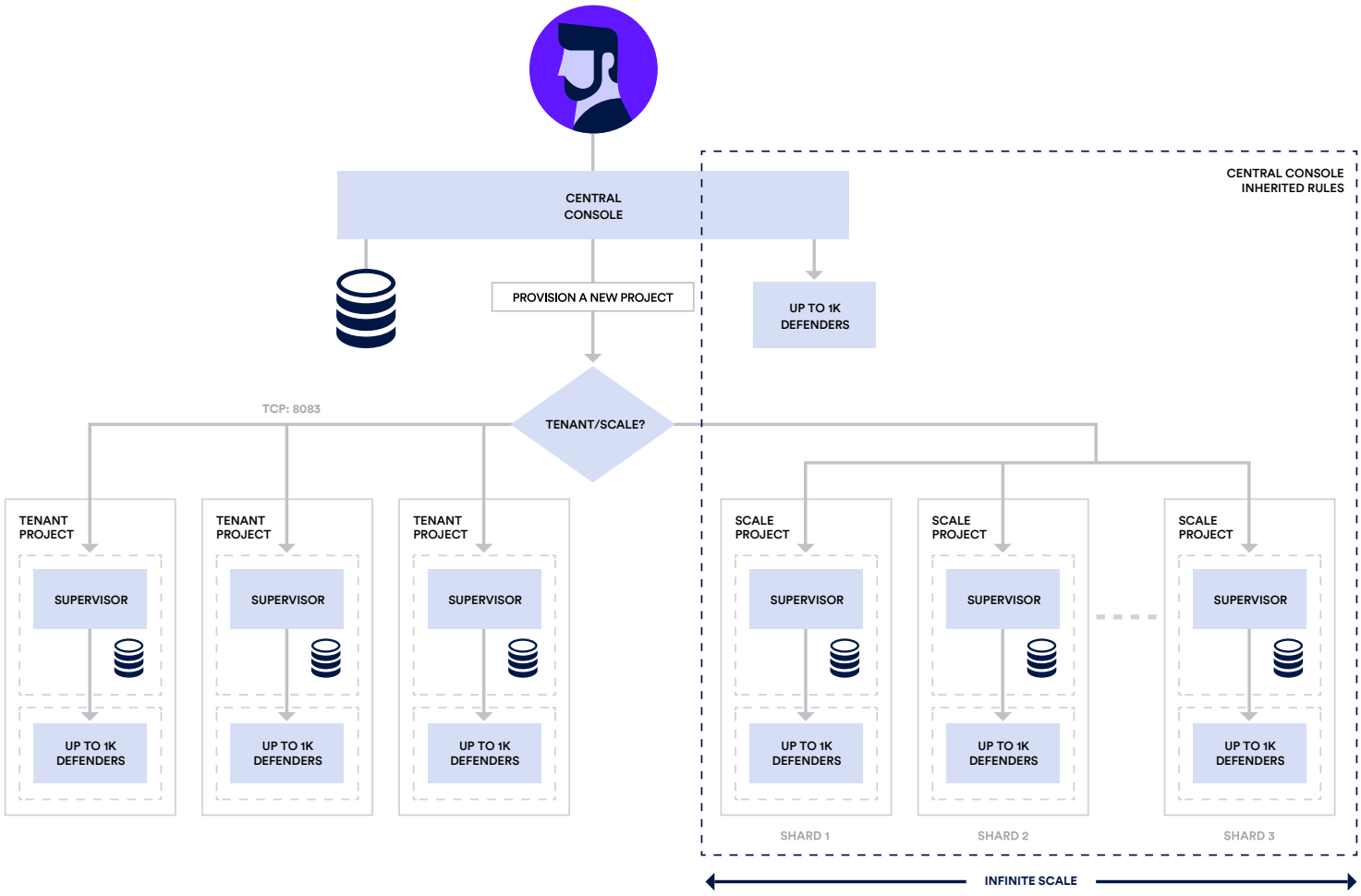
Cloud Platform Compliance helps you centrally discover all the cloud-native services used in AWS, Azure, and Google Cloud, across all regions and accounts. Cloud Provider Compliance continuously monitors these accounts, detects when new services are added, and reports which services are unprotected. It can help you mitigate risks introduced by rogue deployments, abandoned environments, and environments not protected by Twistlock.

Kubernetes has a rich RBAC model based around the notion of service and cluster roles. This model is fundamental to the secure operation of the entire cluster because these roles control access to resources and services within namespaces and across the cluster. While these service accounts can be manually inspected with kubectl, this manual approach can be difficult to visualize and understand service account scope at scale.

Twistlock Radar provides a discovery and monitoring tool for service accounts. Every service account associated with a resource in a cluster can easily be inspected. For each account, Twistlock shows detailed metadata describing the resources it has access to and the level of access it has to each of them. This visualization makes it easy for security staff to understand role configuration, assess the level of access provided to each service account, and mitigate risks associated with overly broad permissions.

Multitenancy and Scale Projects

Twistlock supports multitenancy and unlimited scale. We accomplish this with our Projects capabilities. Twistlock support two types of Projects: Tenant projects and Scale projects. For more information refer to the guide below or access our documentation on the feature here.



Multitenancy — Tenant Projects

The Central Console has full visibility into the entire estate. You can then setup tenant projects which act as a self contained Console and Defender setup. Users can only see and administer their subsection of the estate.

Tenant projects are like silos. They each have their own rules and settings that are created and maintained separately from all other projects.

This is represented in the left hand side of the above diagram.

Scale — Scale Projects

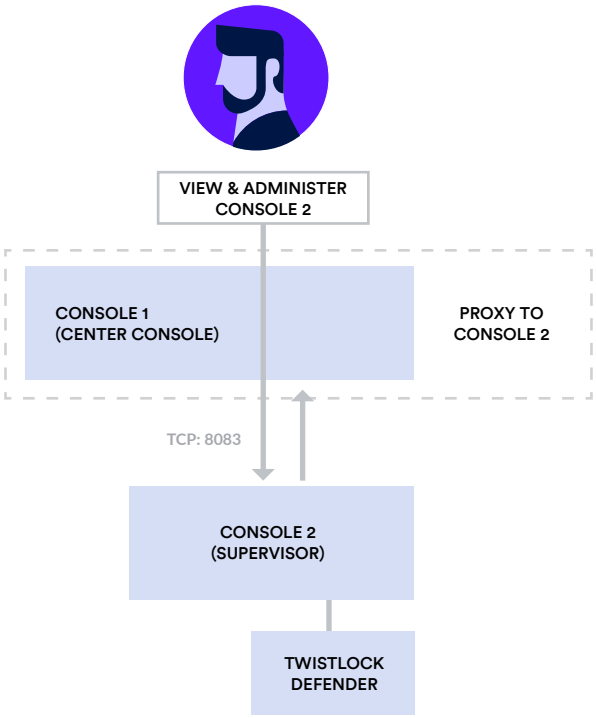
Each Console can support 1,000 Defenders. By utilizing Scale Projects, we can allocate Consoles to a Central Console. This enables an unlimited number of Defenders.

Defenders communicate to the scale project Console (1,000 Defenders per scale project Console) and the scale project Console aggregates and sends to a Central Console.

Policies and rules are inherited by the scale project from the Central Console. Users and administrators operate the Central Console which then pushes changes to the scale projects.

Configuration of Projects

Detailed setup instructions can be found here, In essence, you deploy the Console you want to become the Central Console and connect that to another Console via the User Interface. Twistlock will then configure it appropriately.



By default, the master and its supervisor Consoles communicate over port 8083. You can configure a different port by setting MANAGEMENT_PORT_HTTPS in twistlock.cfg at install time. All Consoles must use the same value for MANAGEMENT_PORT_HTTPS.

Role Based Access Control (RBAC)

The Twistlock Console can be accessed via the graphical user interface and the application programming interface (API).

The Twistlock Console supports the following authentication methods:

- Username / Password
- Lightweight Directory Access Protocol (LDAP)
- Security Assertion Markup Language v2.0 (SAML2.0)
- X.509 smart cards

Twistlock can apply password complexity rules for user accounts created within Twistlock. For the authentication of external identities, Twistlock supports LDAP and SAML2.0. LDAP authentication supports the OpenLDAP and Active Directory directories. Twistlock Console can be configured as an SAML2.0 Service Provider. The SAML2.0 Identity Providers that have been successfully federated with the Twistlock Console are Okta, G Suite, Ping, Shibboleth and Azure Active Directory. Smart card authentication to the Twistlock Console requires configuring Twistlock with the smart card's chain of trust and matching the smart card's SubjectAlternativeName's PrincipalName value to user's corresponding Twistlock username.

Twistlock supports group based authorization and defines the following roles:

Role	Access Level
Administrator	• Full read-write access to all Twistlock settings and data
Operator	• Read-write access to all rules and data • Read-only access to user and group management and role assignments
Defender Manager	• Read-only access to all rules and data • Can install / uninstall Twistlock Defenders • Used for Automating Defender installs via Bearer Token or Basic Auth
Auditor	• Read-only access to all Twistlock rules and data
DevOps User	• Read-only access to vulnerability scan data.
Access User	• Install personal certificates required for access to Defender protected nodes
CI User	• Run the Continuous Integration plugin • No Twistlock Console access

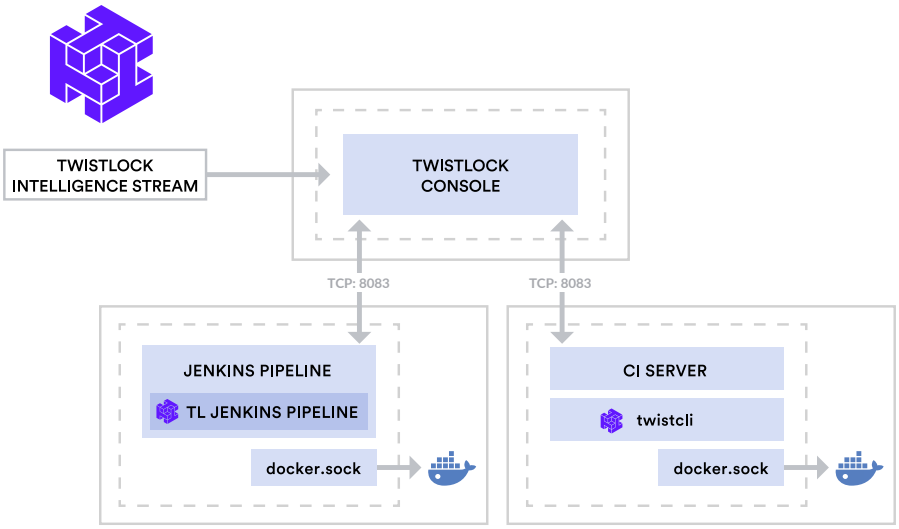
Group membership can be assigned within the Twistlock Console, as an SAML 2.0 role claim, or LDAP group membership value.

Integration with the CI Pipeline

Overview

Engineering teams can integrate Twistlock vulnerability and compliance scanning capabilities into their development process. Twistlock provides a native Jenkins plugin, as well as a stand-alone command-line tool called twistcli, for integration with your continuous integration (CI) pipeline.

Twistlock Integration with Single Node Jenkins or Other CI Server



Twistlock CI integration enables automatic scans of your custom Docker images at build time. Scans can detect vulnerabilities and compliance issues before your images are pushed to the registry and deployed into production. Thresholds can be specified to fail builds of images that have issues that exceed a specified severity.

The results of the scans via Jenkins or twistcli are available in the Console.

Twistlock Jenkins Plugin

The Twistlock Jenkins plugin is compatible with Jenkins version 1.58 or higher. Twistlock Jenkins plugin must be able to reach Twistlock Console over the network. The Twistlock plugin depends on two other Jenkins plugins: `Static Analysis Utilities` and `Dashboard View`.

Scan reports show detailed information for each vulnerability, including information that can assist with remediation (i.e which package versions fix the vulnerability). Trend charts show how the number of security issues has changed over time.

If your Jenkins server runs as a container, mount the docker socket from the host into the Jenkins container at runtime using:

"`-v /var/run/docker.sock:/var/run/docker.sock`".

This enables the Twistlock plugin to run docker commands via the host's Docker installation.

Twistlock Integration with other CI Servers

The Twistlock Command Line Interface, twistcli, allows for integration into most any other CI tools. Simply create a post-build step and use the return value to optionally fail a build based on thresholds for compliance and/or vulnerabilities. See this Support article for more details here.

Twistlock Registry and Serverless Repository Scanning

After you build your Docker images or Serverless functions, you need somewhere to store them. You could use an on-premises registry, a hosted solution, or some combination of both. Either way, you will want to continually scan these images against the latest available threat data.

To scan a registry, Twistlock uses the registry's APIs to list and pull the images to the local system. Twistlock then deconstructs and inventories each image, cross-referencing the threat data from the Intelligence Stream to identify vulnerable packages. By default, registry images are scanned once per day, but the period is configurable.

Twistlock must have the same network connectivity and credentials that would be required run docker pull. Firewalls must be configured correctly and the correct access credentials must be provided.

Twistlock supports Docker v1 and v2 registries, as well as the registries provided by Amazon, Google and Microsoft. For registries that implement the Catalog API, you can use wildcards to specify which repositories to scan - or you can leave the repository name or label blank and all repositories will be scanned. Twistlock also supports PCF Blobstores, the rough equivalent of a Docker container registry.

Finally, Twistlock supports webhook integration. When using Docker Hub and Docker Trusted Registry, Twistlock scans can be triggered as soon as an image is uploaded. For more information about setting up webhooks, see this support article.

Twistlock can also automatically watch your serverless repositories, much like the registry scanning capabilities Twistlock will continually monitor your serverless function repos for changes and scan them automatically, providing you with vulnerability and compliance reporting against the policies you define. For more information on this capability, see this support article.

Container Secrets

Twistlock can be configured to retrieve secrets from your secrets store and inject them into the containers that need them. Twistlock supports a variety of secrets stores:

- AWS Systems Parameter Store
- AWS Secrets Manager
- Azure Key Vault
- CyberArk Enterprise Password Vault
- Hashicorp Vault

Twistlock securely retrieves secrets from your designated secrets store and can inject them as either environment variables or files into the containers you designate. Twistlock provides a granular rule-driven system for defining how and where secrets are injected. To protect your secrets, configure your rules restrictively, using the principle of least-privilege access. For more information about configuring Twistlock to perform secrets injection, see this support article.

Retrieving Data From Twistlock

Twistlock can provide data from the assets it's protecting in a number of ways. The mechanisms for providing data fall in to a few categories:

- Event Driven Messaging
- Log files
- API Calls

Event Driven Messaging

Twistlock provides comprehensive event driven alerting capabilities that allow customers to integrate Twistlock with assets they may already have investments in. Twistlock has integrations with the following:

- Generic Webhook Provider
- Email
- Jira
- Slack
- PagerDuty
- Google Cloud Security Command Center
- AWS Security Hub
- IBM Security Advisor

For more information see our Push Alerts documentation.

Kubernetes Audits

Twistlock can be configured as an audit sink for your Kubernetes logs, allowing you to use Twistlock to generate alerts based on events detected in the logs.

Log Files

Twistlock writes to RFC compliant syslog, this syslog data can be injected by any number of time series data solutions as well as common SIEM solutions. Twitlock offers a few layers of detail in our syslog messages which you can select to increase the verbosity of the messages. Twistlock recommends leaving the messages at their defaults Twistlock can also write to STDOUT as well as feed in to Prometheus. For more information see our logging documentation.

API Calls

As mentioned above, Twistlock provides a stable, well documented RESTful API for customer to leverage when automating or extracting data from the platform. For more information see our API documentation.

Twistlock
A PALO ALTO NETWORKS® COMPANY

Trusted by 35% of the Fortune 100, Twistlock is the world's first truly comprehensive cloud native security platform - providing holistic coverage across hosts, containers, and serverless in a single platform. Twistlock is cloud-native and API-enabled itself, protecting all your workloads regardless of what underlying compute technology powers them.

Follow Twistlock

Twitter

Facebook

LinkedIn