Security workshop with Sysdig on Red Hat OpenShift

Introduction:

Shifting to containers accelerates application delivery. It also tends to complicate security, visibility and compliance. DevOps teams are learning that they can't solve these new business challenges with traditional approaches. Moving to cloud-native infrastructure brings about critical security issues that can be avoided with the right solutions in place.

During this workshop, discover how Sysdig solutions on Red Hat® OpenShift® work together to protect businesses and avoid security and compliance issues.

This workshop consists of the following four (4) labs:

- Installing the Sysdig Agent
- Runtime Security
- Securing your Images at Runtime
- Securing your Image Pipeline

Table of Contents

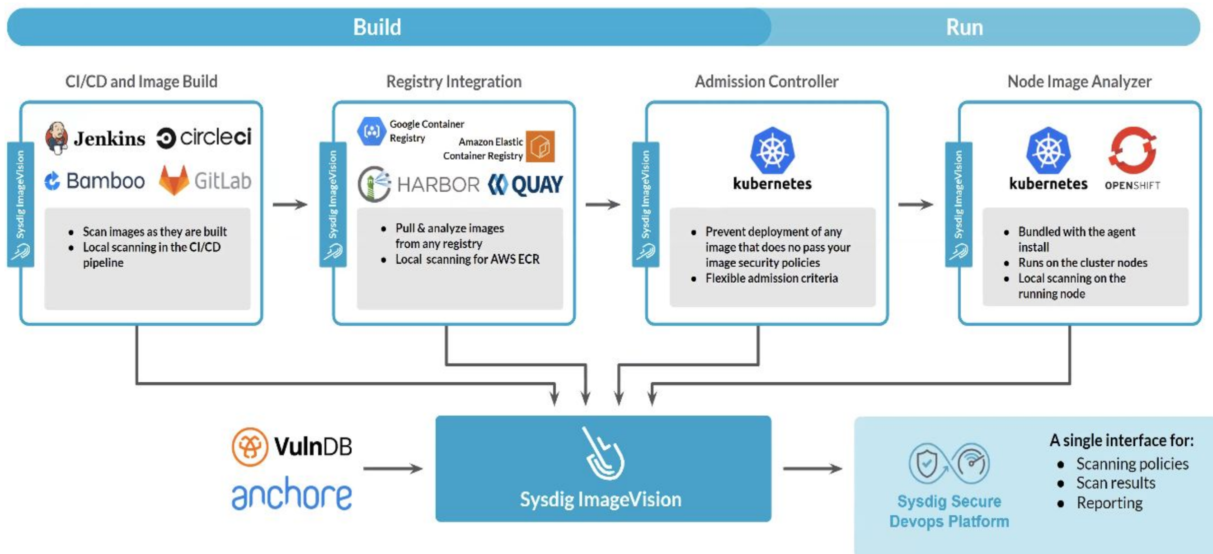
Introduction

It is imperative that you maintain a high level of security & compliance in your entire application environment. Not doing so can result in your system being compromised. This can incur significant costs and can lead to commercial & business issues, and failed compliance tests leading to a loss of trust with customers and monetary fines and/or settlement fees.

In a cloud native environment, the security & compliance posture of your application stack is dependent largely upon the security of your containers, but not exclusively - it also depends upon the infrastructure upon which it runs.

Sysdig Secure embeds security and compliance into the build and runtime stages of the container lifecycle. Sysdig Secure is a unified data platform that provides image scanning, vulnerability management, compliance, runtime security, and forensics for enterprise cloud native environments at scale.

Image Scanning can be performed at various points in the container lifecycle. In this workshop, we will look specifically at image scanning using the Node Image Analyser on the host, and scanning within a CI/CD pipeline, as well as runtime threat detection.



For a detailed description of Image Scanning, please refer to [Appendix 1: Image Scanning Technical Description.](#)

What you will learn:

You will learn how image scanning can provide the security insights you need without affecting the level of flexibility you desire. Along the way we will simulate a number of roles as we progress through this workshop, depending upon the task at hand.

- DevOps Engineer
- DevSecOps Engineer
- Bad Actor

Who should attend:

- Cloud architects
- Developers and DevOps engineers
- Security architects

Prerequisites

1. You must have a Sysdig account for this workshop. You can sign up for a free trial here.
2. Pull down the required lab files & move them to the workshop directory.

```
git clone https://github.com/johnfitzpatrick/ocp-workshop.git workshop && cd
$_
```

3. Download and install the OpenShift `oc` command-line tools and add them to your PATH:
   a. Linux: https://mirror.openshift.com/pub/openshift-v4/clients/ocp/latest/openshift-client-linux.tar.gz
   b. MacOS: https://mirror.openshift.com/pub/openshift-v4/clients/ocp/latest/openshift-client-mac.tar.gz
   c. Windows: https://mirror.openshift.com/pub/openshift-v4/clients/ocp/latest/openshift-client-windows.zip

   [here](https://sysdig.com/company/free-trial/).

```
wget
http://<Openshift_Master_Console_URL>/pub/openshift-v4/clients/ocp/4.6.3/ope
nshift-client-linux-4.6.3.tar.gz
```

Note: If you're using Windows, then please refer to the instructions here https://www.openshift.com/blog/installing-oc-tools-windows.

Lab 1: Install the Sysdig Agent

Description:

The Sysdig agent is a set of processes running in your host that perform the capture, processing and transmission of metrics, as well as policy monitoring and enforcement. It does this mostly by listening to every **syscall** on the host's kernel. Since every request on the host is handled by a single kernel means only one agent needs to be installed per host, no matter the container density of that host.

Objectives:

After completing this lab, you will be able to:

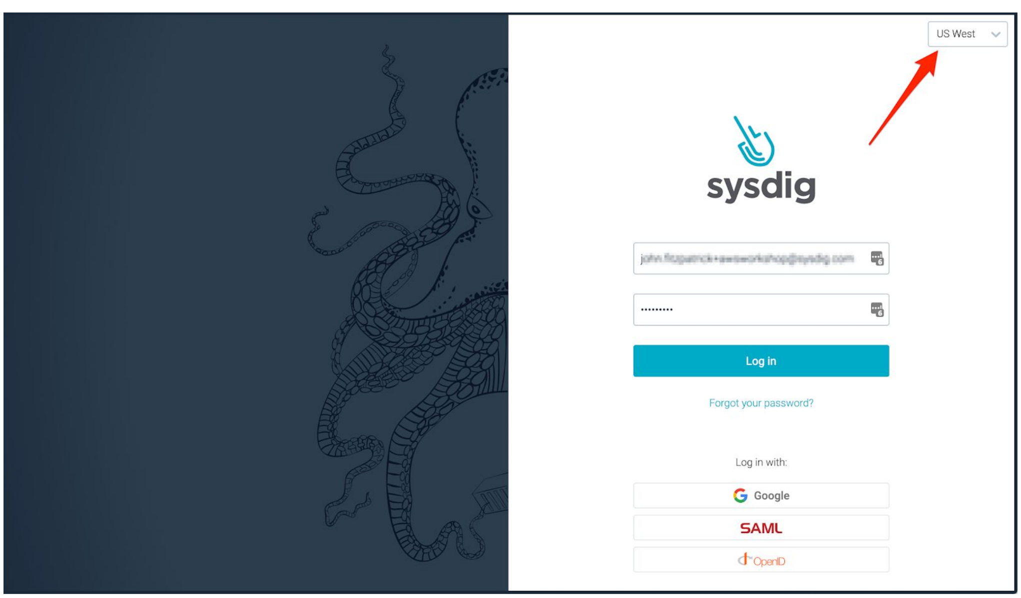
- Get started with Sysdig
- Log into Sysdig Secure
- Deploy the Sysdig Agent

Task 1: Login to Sysdig Secure

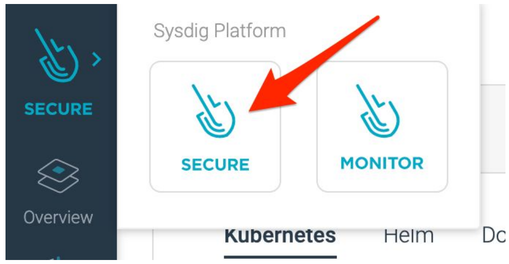
Role: DevSecOps Engineer

1. Log into Sysdig Secure by browsing to the URL in the email you received when you signed up for your trial account.

If you do not have this at hand, then browse to https://secure.sysdig.com/. You may need to select the region your account is hosted in from the dropdown list on the top right of the screen.



2. Make sure you are logged into Sysdig Secure, and not Sysdig Monitor. Click on the Sysdig icon in the upper left and choose Sysdig Secure.



Task 2: Deploy the Sysdig Agent on your cluster

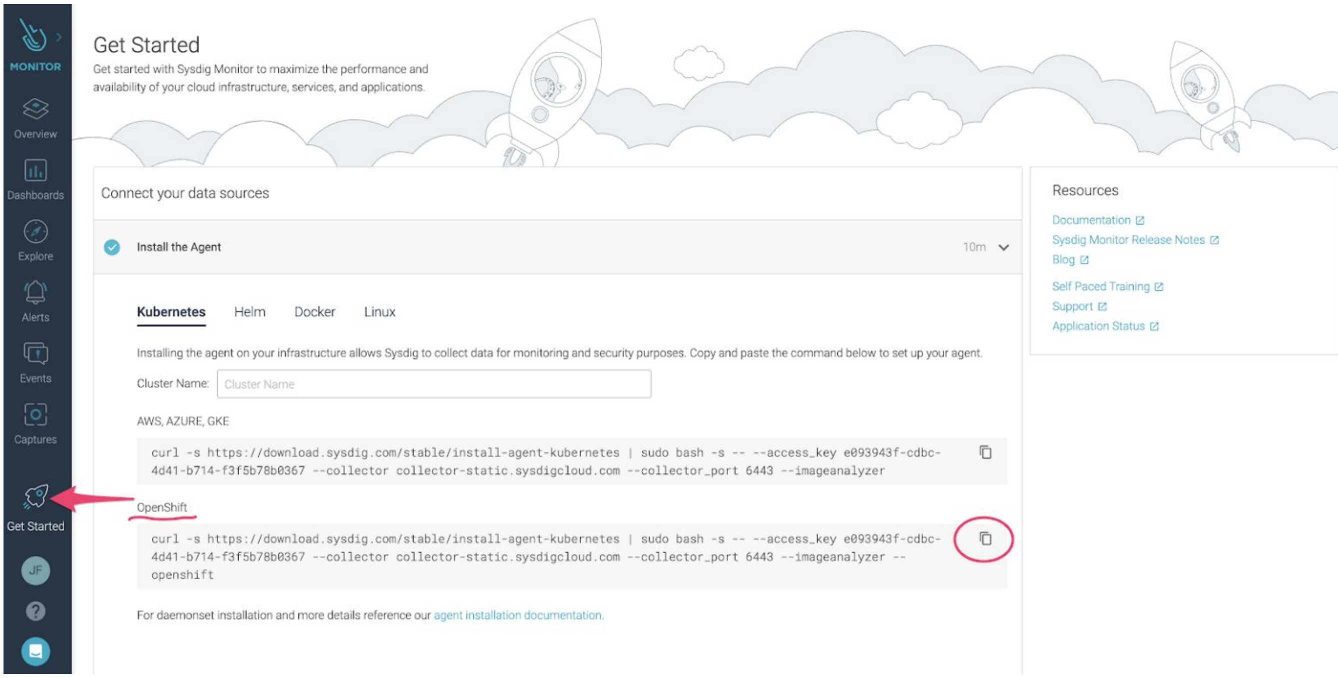
Role: DevSecOps Engineer

The Sysdig Agent is responsible for processing metrics, analysis, aggregation, and transmission of data to the Sysdig backend.

1. Log into the OCP cluster as user 'admin'.

```
$ oc login https://<cluster-url>:6443
Authentication required for
https://api.cluster-1e5a.1e5a.example.opentlc.com:6443 (openshift)
Username: <admin>
Password: <admin-password>
Login successful.
```

2. Check you're logged in and your cluster is ready.
3. Click on the 'Get Started' link on the left in Sysdig Secure, then expand the 'Install the Agent' section.
4. Copy the installation command for Openshift.



5. Paste the command into the command line.

```
$ curl -s https://download.sysdig.com/stable/install-agent-kubernetes | sudo
bash -s -- --access_key <YOUR KEY HERE> --collector
collector-static.sysdigcloud.com --collector_port 6443 --imageanalyzer
--openshift
```

The output should be similar to the following:

```
* Detecting operating system
* Downloading yamls files to the temp directory:
/tmp/sysdig-agent-k8s.5IrByT
* Downloading Sysdig cluster role yaml
* Downloading Sysdig config map yaml
* Downloading Sysdig daemonset v2 yaml
* Downloading Sysdig daemonset slim v2 yaml
* Downloading Sysdig Image Analyzer config map yaml
* Downloading Sysdig Image Analyzer daemonset v1 yaml
* Creating project: sysdig-agent
node/ip-10-0-138-21.eu-central-1.compute.internal labeled
node/ip-10-0-155-177.eu-central-1.compute.internal labeled
```

```
node/ip-10-0-189-3.eu-central-1.compute.internal labeled
node/ip-10-0-190-23.eu-central-1.compute.internal labeled
node/ip-10-0-199-146.eu-central-1.compute.internal labeled
* Creating sysdig-agent serviceaccount in project: sysdig-agent
* Creating sysdig-agent access policies
* Creating sysdig-agent secret using the ACCESS_KEY provided
* Updating agent configmap and applying to cluster
* Setting collector endpoint
* Setting collector port
configmap/sysdig-agent created
* Setting Analysis Manager endpoint for Image Analyzer
configmap/sysdig-image-analyzer created
Trying to detect COS (Container-Optimized OS) - to enable eBPF
COS not detected.
* Deploying the sysdig agent
daemonset.apps/sysdig-agent created

The list of agent pods deployed in the namespace "sysdig-agent" are:
sysdig-agent-89vhj   0/1     ContainerCreating   0          0s
sysdig-agent-9d4rk   0/1     ContainerCreating   0          0s
sysdig-agent-j69t5   0/1     ContainerCreating   0          0s
sysdig-agent-rrcwf   0/1     ContainerCreating   0          0s
sysdig-agent-xsjxw   0/1     ContainerCreating   0          0s

Make sure the above pods all turn to "Running" state before continuing
Should any pod not reach the "Running" state, further info can be obtained
from logs as follows
'kubectl logs <agent-pod-name> -n sysdig-agent'
* Deploying the Image Analyzer
daemonset.apps/sysdig-image-analyzer created

The list of Image Analyzers pods deployed in the namespace "sysdig-agent"
are:
sysdig-image-analyzer-gwqbn   0/1     ContainerCreating   0          0s
sysdig-image-analyzer-kc9c6   0/1     ContainerCreating   0          0s
```

```
sysdig-image-analyzer-vdgfk   0/1     ContainerCreating   0          0s
sysdig-image-analyzer-wh52r   0/1     ContainerCreating   0          0s
sysdig-image-analyzer-x4s4s   0/1     ContainerCreating   0          0s
```

Once complete you can check the status of the pods.

```
$ oc get pods -n sysdig-agent
NAME                          READY   STATUS              RESTARTS   AGE
sysdig-agent-89vhj            0/1     ContainerCreating   0          19s
sysdig-agent-9d4rk            0/1     ContainerCreating   0          19s
sysdig-agent-j69t5            0/1     ContainerCreating   0          19s
sysdig-agent-rrcwf            0/1     ContainerCreating   0          19s
sysdig-agent-xsjxw            0/1     ContainerCreating   0          19s
sysdig-image-analyzer-gwqbn   0/1     ContainerCreating   0          18s
sysdig-image-analyzer-kc9c6   0/1     ContainerCreating   0          18s
sysdig-image-analyzer-vdgfk   0/1     ContainerCreating   0          18s
sysdig-image-analyzer-wh52r   0/1     ContainerCreating   0          18s
sysdig-image-analyzer-x4s4s   0/1     ContainerCreating   0          18s
```

In the next lab, we will see how **Sysdig Agent** allows us to monitor and alert on activity within the cluster. We will look at **Sysdig Image Analyser** later in this workshop.

Lab 2: Monitoring Security at Runtime

Description:

What would you do if you are having security issues at your application level, or a bad actor is performing nefarious activities on your system, either within a container or within your orchestration environment? For this, we need to monitor what's happening in your container in real-time.

The Sysdig agent continuously monitors activity on the host by listening to every syscall on the kernel. This way, along with extensive tagging, you can monitor and alert on every conceivable event that occurs across all containers running on that host.

Objectives:

After completing this lab, you will be able to:

- Enable Runtime Policies
- Configure individual policies
- Enable Notifications in Your Account
- Investigate incidents
- View Activity Audit

Task 1: Deploy a Demo Application on Your Cluster

Role: DevOps Engineer

First, let's deploy an application to the cluster.

1. Run the following to deploy an Nginx application (based on nginx:1.16.0).

```
$ oc create ns web-app
$ oc adm policy add-scc-to-user anyuid system:serviceaccount:web-app:default
$ oc apply -f lab/manifests/nginx-1.yaml -n web-app
```

2. Once deployed, you can check the status to make sure everything is up and running.

```
$ oc get pods -n web-app
NAME                        READY  STATUS   RESTARTS   AGE
nginx-56d5598888-njgpp       1/1          Running    0         1m
```

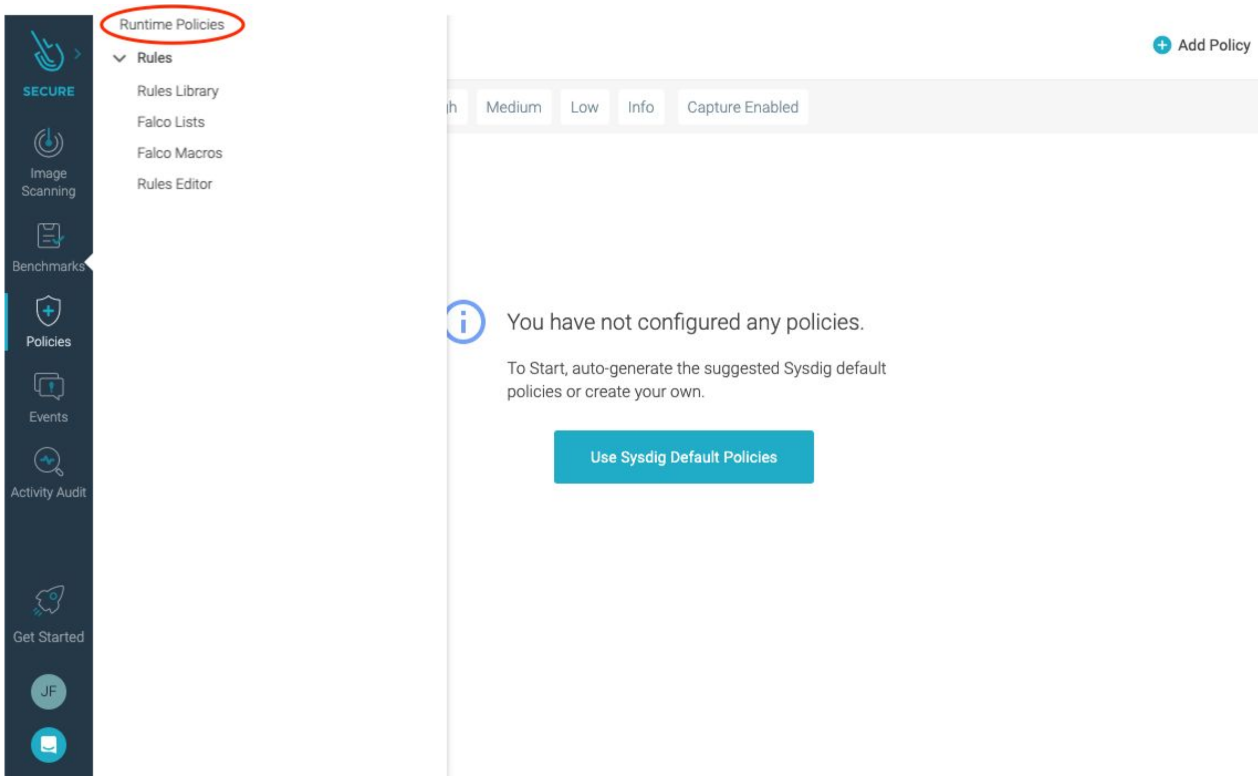
Task 2: Enable & View Runtime Policies

Role: DevSecOps Engineer

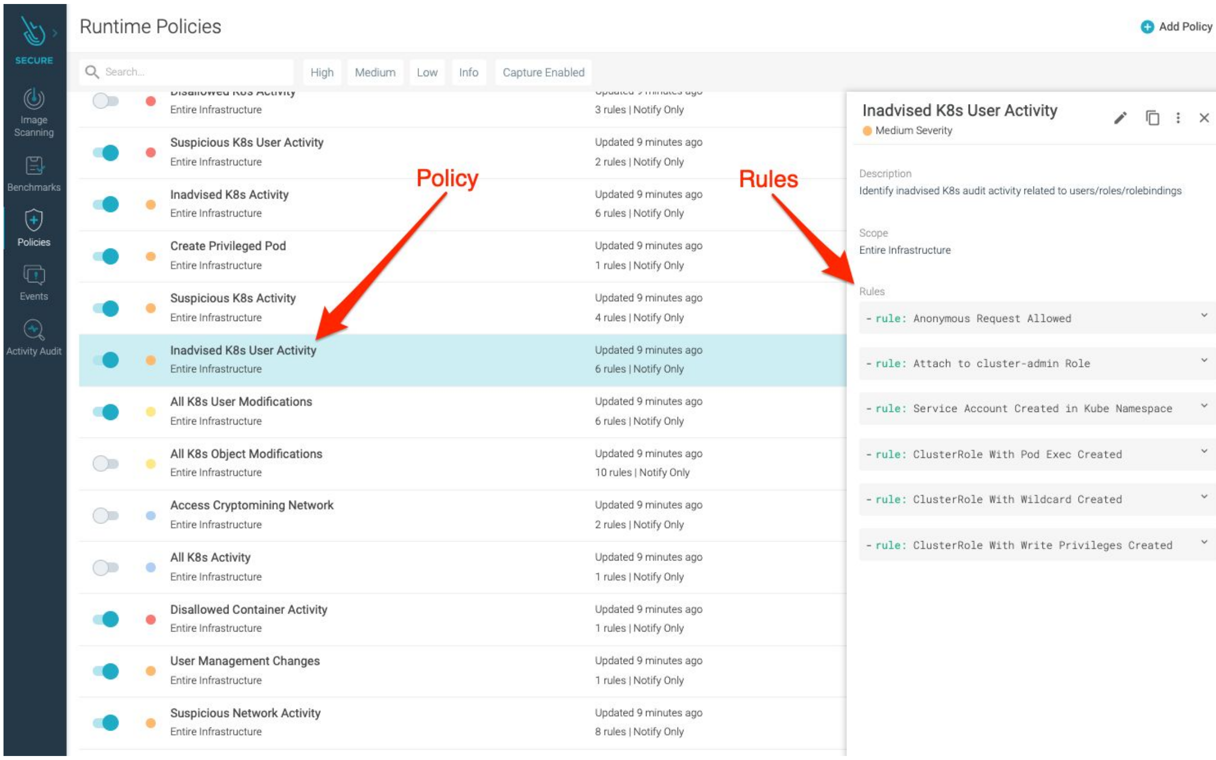
A Sysdig Secure Runtime Policy is a combination of rules about activities an enterprise wants to detect in an environment, the actions that should be taken if the policy rule is breached, and, potentially, the notifications or other actions that should be invoked. Examples of these may be, someone logging into a container, suspicious activity on your cluster, etc.

1. Click on 'Policies' on the left menu, then 'Runtime Policies'.
   For a new account, you will be presented with 'You have not configured any policies'.

2. Go ahead and click 'Use Sysdig Default Policies'.
   This will deploy the default policy set, which is a generic set designed to cover 95% of common detection scenarios.

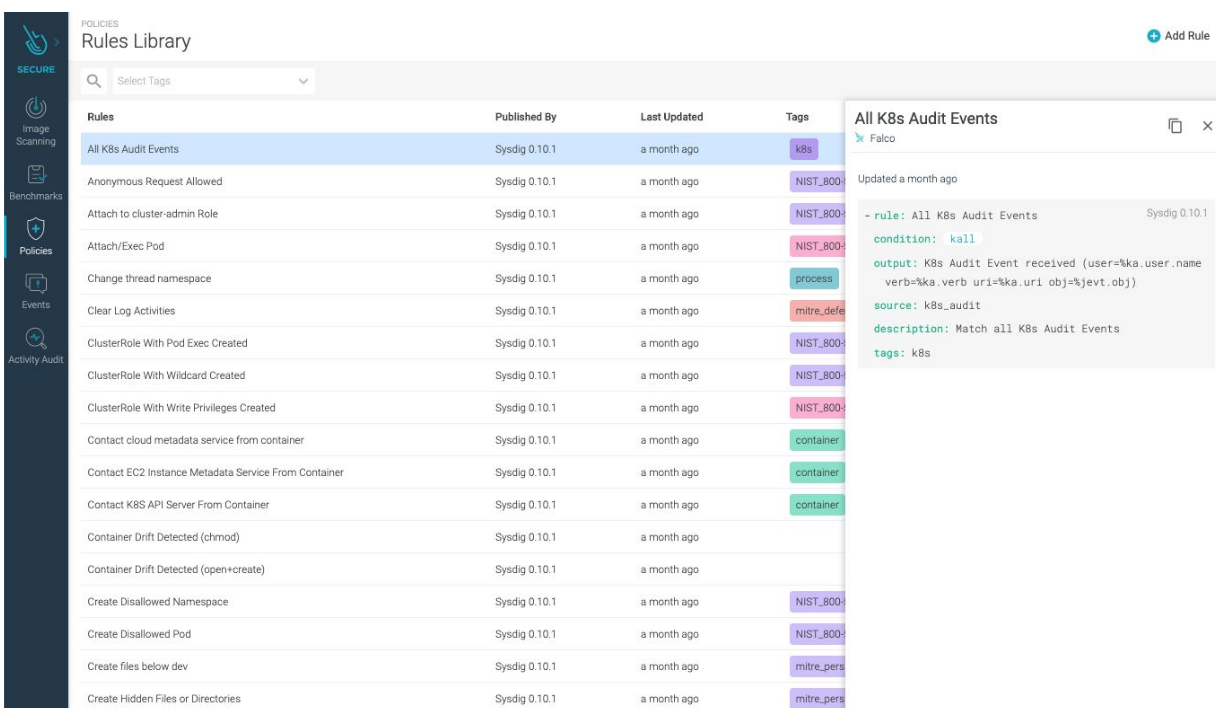


Common policies are enabled by default, but no notification or preventative actions are configured, so only events within the Sysdig Platform will be created.



Each policy represents a collection of discreet rules, mostly based on Falco.

3. Browse to ‘Policies > Rules Library’ to see these rule descriptions.



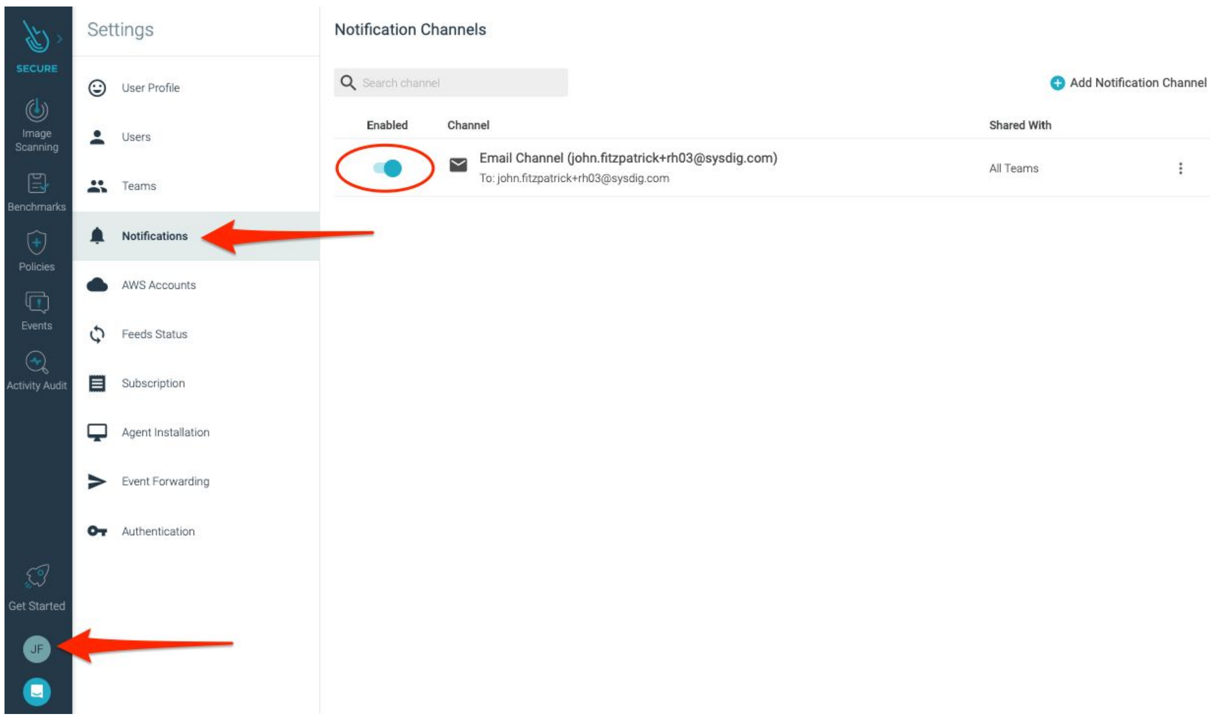
These rules are totally configurable and based on Container, File System, Network, Process, Syscall, or Falco rules.

Task 3: Configure 'Suspicious Container Activity' Policy

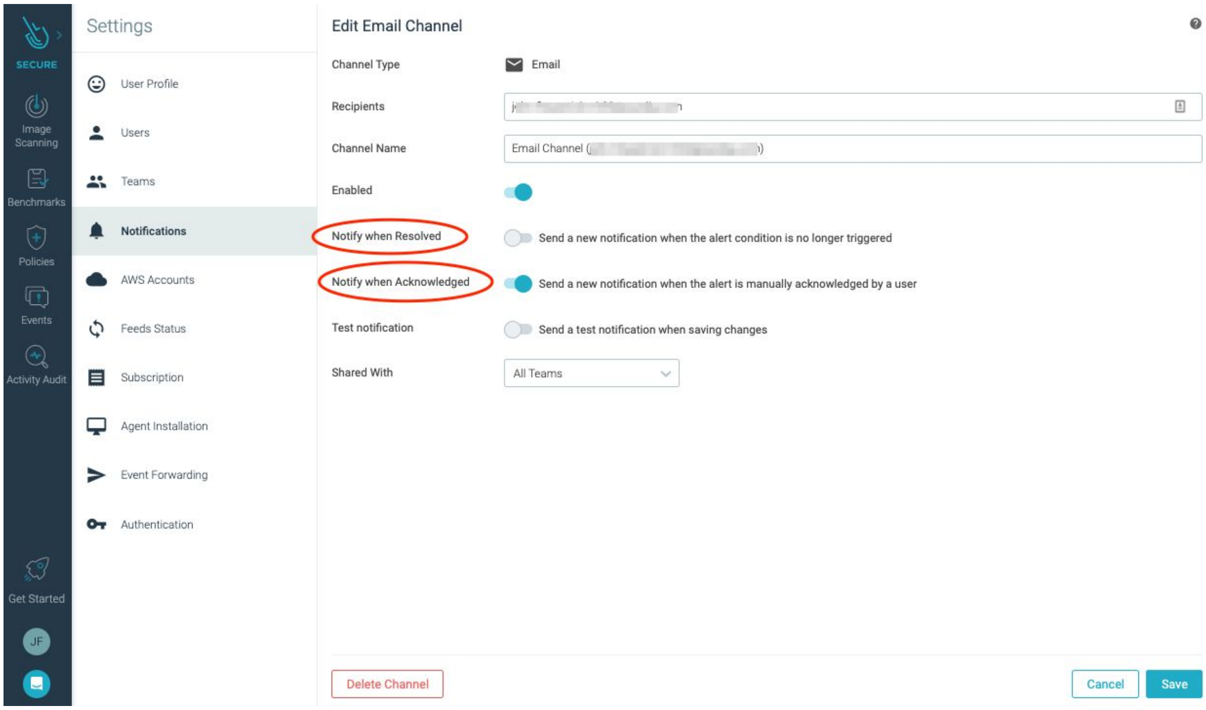
Role: DevSecOps Engineer

We will now configure one of our policies to raise an alert when fired.

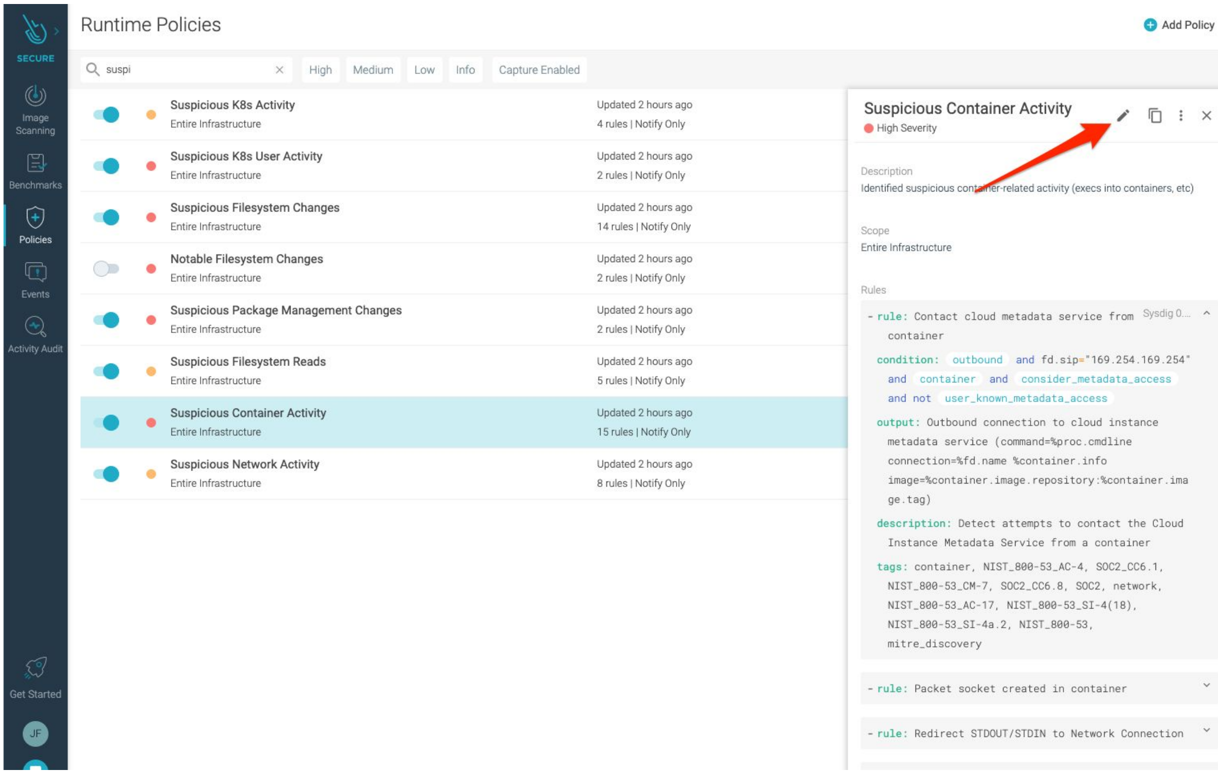
1. Note, in order to receive the email, make sure notifications are enabled at the account level under 'Settings > Notifications'.



2. Optionally, you can configure the notification channel details and when you receive notifications by clicking the notification.



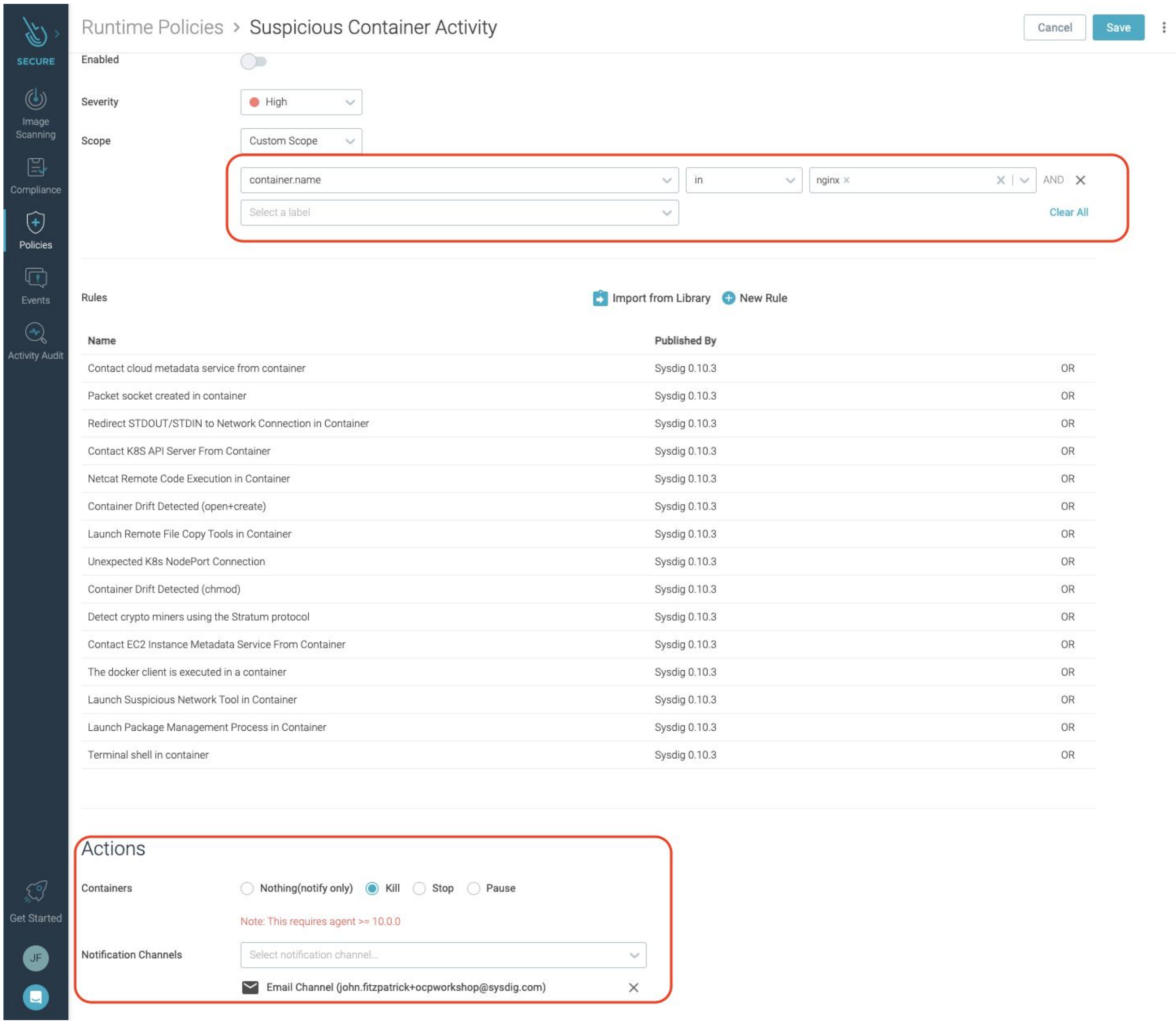
3. Click ‘Save’.
4. Go back to ‘Policies’ > ‘Runtime Policies’.
5. Highlight ‘Suspicious Container Activity’, and you will see the list of Falco rules that make up this policy on the left.
   As the name suggests, this policy will trigger when something untoward occurs in the container. Shortly we will log into the container to fire an Event based on this Policy.



6. Click the edit icon identified.
7. Set the Scope to 'container.name in nginx'.

****NOTE** Make sure you set the scope to JUST 'container.name in nginx', otherwise it will affect your entire OpenShift cluster!!**

8. Set the Actions to Kill the container and send a Notification to your email, as illustrated.



9. Click '**Save**'.

Task 4: Launch an Attack

Role: Hacker/Bad Actor

Let's play the attacker and spawn a shell in our Nginx container and run a command.

1. First, ensure check that the Nginx pod is running.

```
$ oc get pods -n web-app
NAME                        READY  STATUS RESTARTS   AGE
nginx-56d5598888-njgpp       1/1         Running    0         1m
```

2. Let's get the name of the actual container that's running inside the pod - we will compare against this later.

```
$ NGINXPOD=$(oc get pods -n web-app | grep nginx | awk '{print $1}')

$ oc describe pod $NGINXPOD -n web-app | grep "Container ID"
Container ID:
cri-o://678bb7e0bc00674c974c210ee3212ede370cfd54f5771d8e6199cb82a7d2a988
```

3. Now let's log into the pod and run a few commands.

```
$ oc -n web-app exec -it $NGINXPOD -- /bin/bash -c 'echo This a virus >
/tmp/tempfile && cat /tmp/tempfile'

This is a virus
```

4. Then **quickly** view the pods status as the container is killed and restarts.

```
oc get pods -n web-app
NAME                        READY   STATUS   RESTARTS   AGE
nginx-5c445d5c56-brgjk      0/1    Error  3           4m45s
oc get pods -n web-app
NAME                        READY   STATUS     RESTARTS   AGE
nginx-5c445d5c56-brgjk      1/1    Running    1         55s
```

5. Check the container has been killed and respawned (with a new container id, as configured in our alert).

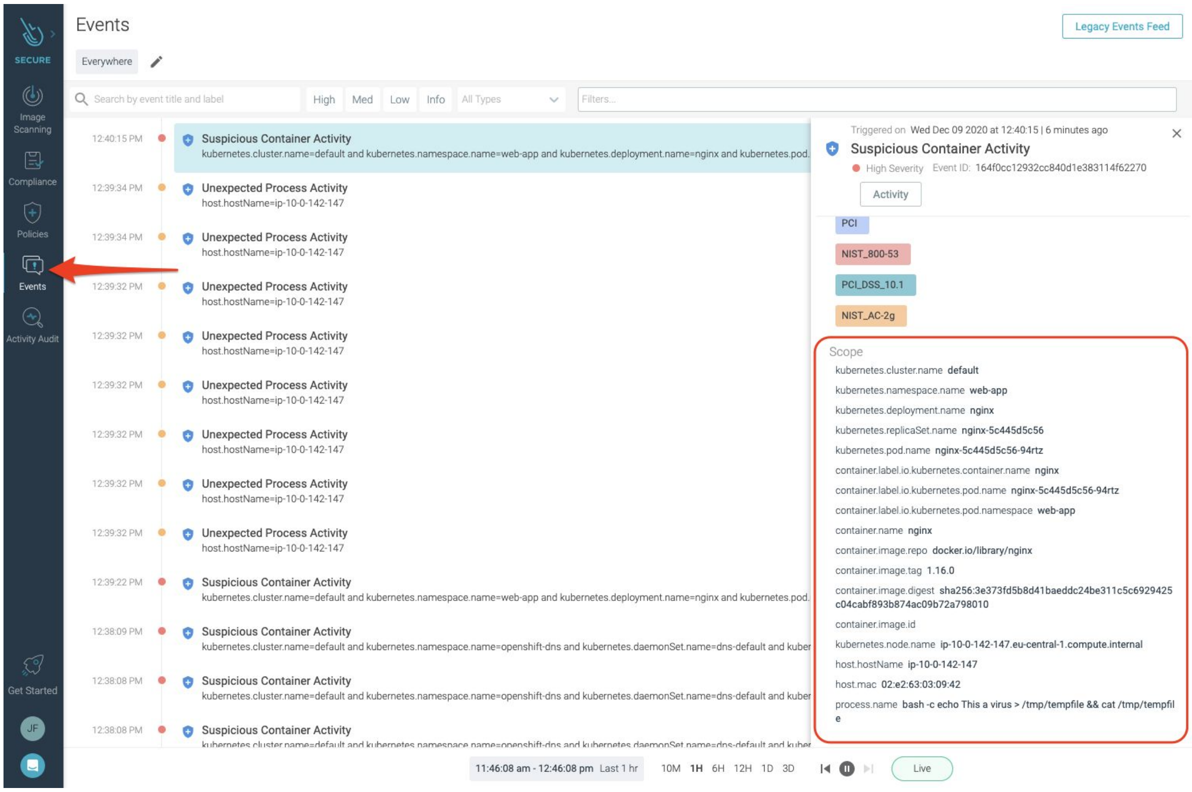
```
$ oc describe pod $NGINXPOD -n web-app | grep "Container ID"
Container ID:
cri-o://f52222b328c6debc862aec896ada25bbecc62af318beb28134af465e49a0cddb
```

Task 5: Play the DevSecOps Role

Role: DevSecOps Engineer

Let's now play the DevSecOps Engineer role and investigate the incident.

1. Browse to Events.
   You will see details of the activity and in particular the hostname from where the event originated.



2. You should also receive an email alerting you to the event, e.g.

Sysdig Notifications Inbox - j...ysdig.com 12:32

Suspicious Container Activity triggered at 12/09/2020 12:32:02.876 PM UTC

To: john.fitzpatrick+ocpworkshop@sysdig.com

Policy event triggered at 12/09/2020 12:32:02.876 PM UTC .

Policy Suspicious Container Activity
Identified suspicious container-related activity (execs into containers, etc)

Triggered at 12/09/2020 12:32:02.876 PM UTC

Severity High

Scope host.mac: 06:92:24:4a:be:52
process.name: dig -t A @172.30.0.10 +short image-registry.openshift-image-registry.svc.cluster.local
container.label.io.kubernetes.pod.name: dns-default-lx4k7
kubernetes.namespace.name: openshift-dns
container.image.tag:
container.name: dns-node-resolver
kubernetes.daemonSet.name: dns-default
container.image.id:
container.label.io.kubernetes.container.name: dns-node-resolver
container.image.digest:
sha256:476573bea77709c0e40f5cbd97244bf3fa1c7c5cb76396cf6917c91e2a808114
container.image.repo: quay.io/openshift-release-dev/ocp-v4.0-art-dev
host.hostName: ip-10-0-164-236
kubernetes.node.name: ip-10-0-164-236.eu-central-1.compute.internal
kubernetes.cluster.name: default
kubernetes.pod.name: dns-default-lx4k7
kubernetes.service.name: dns-default
container.label.io.kubernetes.pod.namespace: openshift-dns

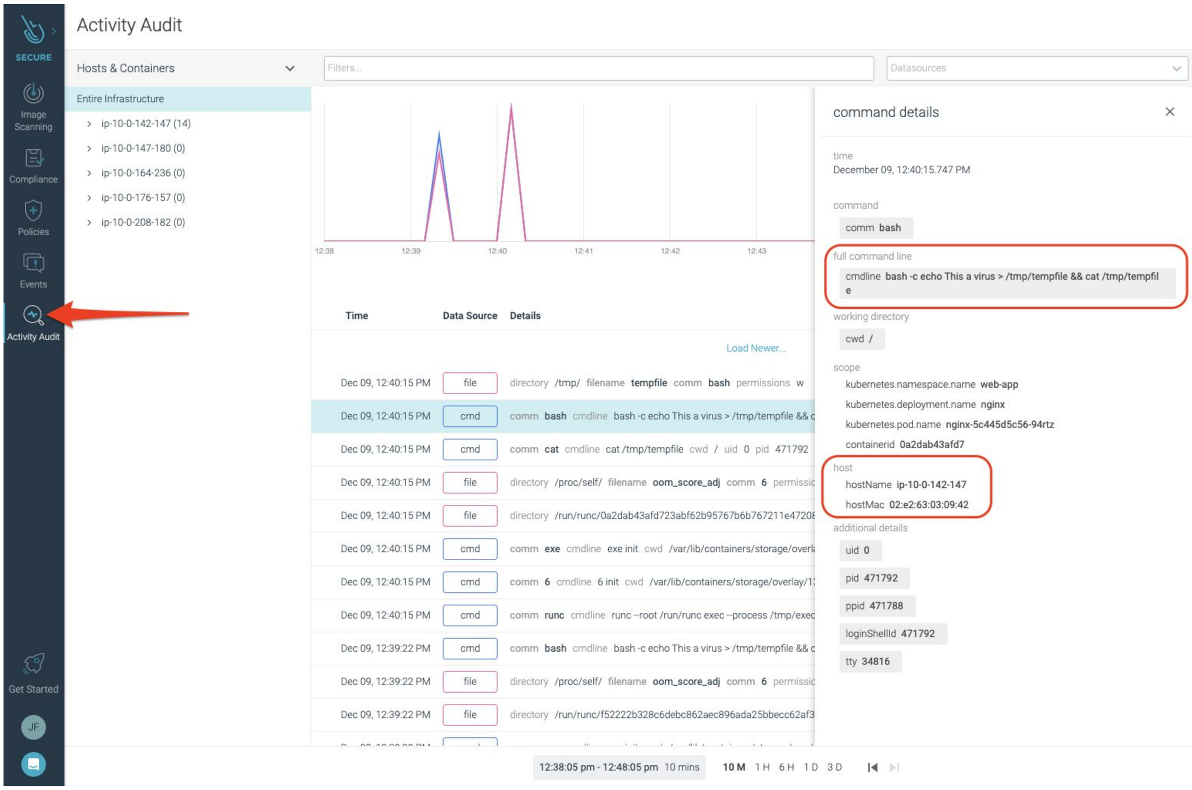
Actions No actions performed

Details Network tool launched in container (user=root user_loginuid=-1 command=dig -t A @172.30.0.10 +short image-registry.openshift-image-registry.svc.cluster.local parent_process=bash container_id=7dfea3d5f93d container_name=dns-node-resolver image=quay.io/openshift-release-dev/ocp-v4.0-art-dev:<NA>)

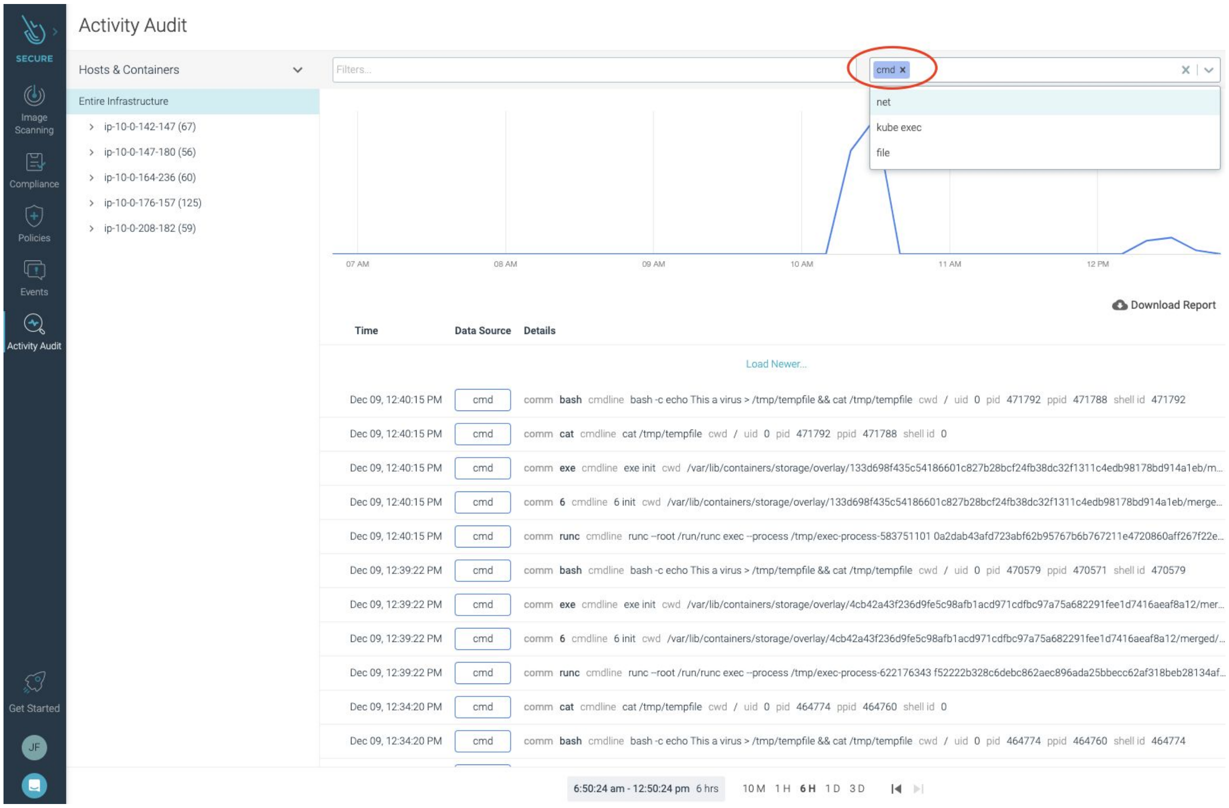
Task 6: View Activity Audit

Role: DevSecOps Engineer

1. Now browse to Activity Audit.
   You will see a complete trail of the command history, not just within the container, but also activity with OpenShift itself.



2. Click on each entry to drill down further into the activity.
   In our case we can see full details of the commands that were run, as well as the host the commands originated from.
3. Select a ‘Datasource’ on the top right to narrow the view to only command, file, network, ‘cmd’ activity.


Activity Audit
Hosts & Containers
Entire Infrastructure
cmd
net
kube exec
file
Download Report
Time
Data Source
Details
Load Newer...

Lab 3: Secure your Images at Runtime

Description:

In the previous lab, we looked at runtime security. But what about the security of the image itself – how we can detect and alert on configuration and vulnerability concerns?

In this lab, we'll see how images can be scanned automatically upon deployment into our cluster.

Objectives:

After completing this lab, you will be able to:

- Describe Node Image Analyser
- Inspect Node Image Analyser Scan Results
- Create scanning Policies
- Create Policy Assignments
- Create an Alert

Task 1: Reviewing Node Image Analyser Scans

Role: DevSecOps Engineer

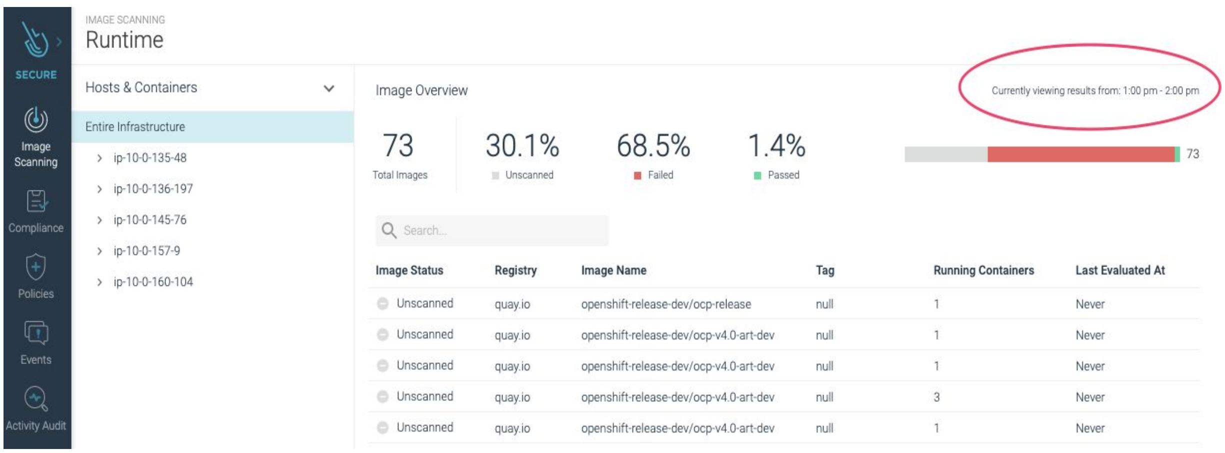
The Sysdig Image Analyser is deployed as a separate pod at the same time as the Sysdig Agent. Once deployed, it will scan deployed images automatically.

1. Run the following command to view the Sysdig Image Analyser pod.

```
$ oc get pods
NAME                           READY   STATUS    RESTARTS   AGE
sysdig-agent-2tg2v            1/1     Running   0          1h40m
sysdig-agent-9ltnr            1/1     Running   0          1h40m
sysdig-agent-s7wvs            1/1     Running   0          1h40m
sysdig-agent-tt4bw            1/1     Running   0          1h40m
sysdig-agent-w8ztj            1/1     Running   0          1h40m
sysdig-image-analyzer-8pqnl   1/1     Running   0          1h37m
```

```
sysdig-image-analyzer-ln48p   1/1   Running   0   1h37m
sysdig-image-analyzer-mzdwm   1/1   Running   0   1h37m
sysdig-image-analyzer-t9v5z   1/1   Running   0   1h37m
sysdig-image-analyzer-wg7zs   1/1   Running   0   1h37m
```

2. In Sysdig Secure UI, click on 'Image Scanning' on the left side menu, then 'Runtime'.
3. Select 'Hosts & Containers' view and highlight 'Entire Infrastructure'.
   You will see a list of running images that are present on your system, their scan status, along with the time frame in which the image was scanned.

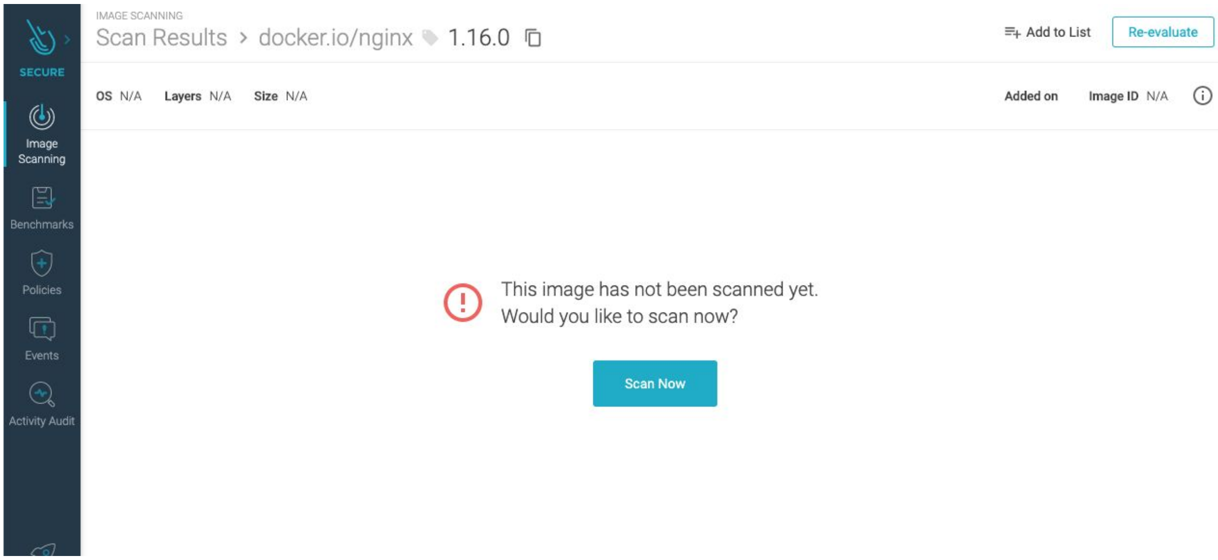


Note: It may take several minutes for all containers to be scanned and for the results to come through.

You can change the scope of what is displayed by clicking on the drop-down on the top left side which defaults to 'Hosts & Containers' or change the groupings with the dropdown.

The Node Image Analyser on the node cycles through and scans all running containers. Remember, your OpenShift cluster itself consists of a number of containers, so these too will be scanned, so it might take a little while for your Nginx container to be scanned.

If this is not already scanned, you can expedite a scan by highlighting the container and clicking 'Scan Now'.



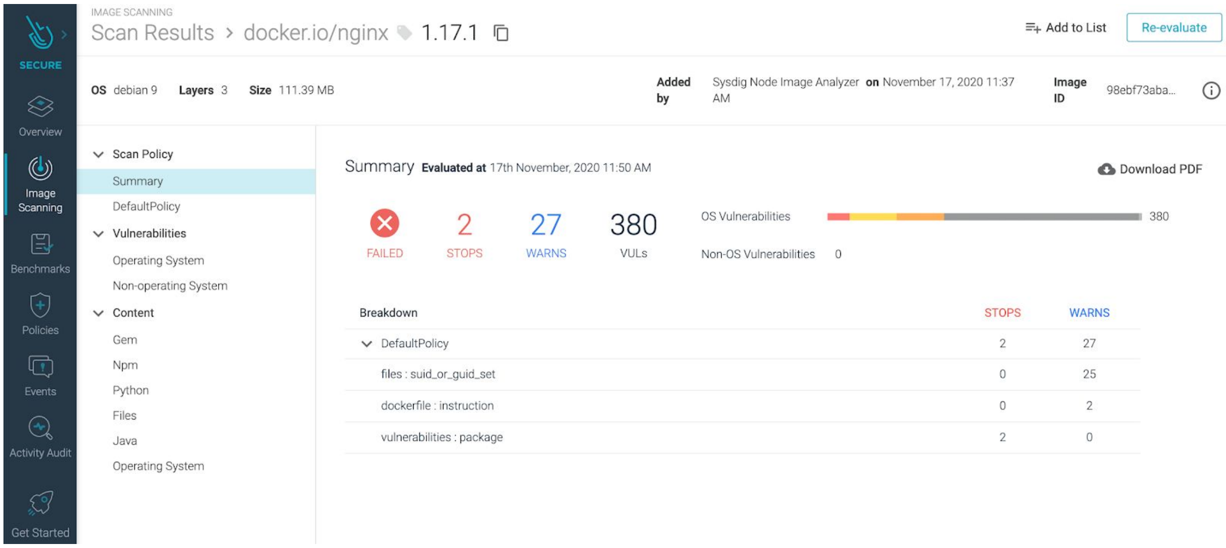
Note, you can check which node your 'nginx' pod is running on by running '**oc get pods -n web-app -o wide**', then click on that node.

```
$ oc get pods -n web-app -o wide
NAME                         READY   STATUS    RESTARTS   AGE   IP
NODE                                                 NOMINATED NODE   READINESS
GATES
nginx-5c445d5c56-szdbh   1/1   Running   3       38m   10.128.2.234
ip-10-0-147-128.eu-central-1.compute.internal   <none>        <none>
```

Task 2: Inspect Scan Results

Role: DevSecOps Engineer

1. Click on one of the failed images, for example 'nginx 1.16.0' to see the results of the scan. You will see details of both OS and non-OS vulnerabilities, as well as details of packages and files on the image itself.



Note: Zero-day vulnerabilities, i.e. newly discovered vulnerabilities since the image was deployed, will be automatically added to the vulnerability database. There is no need to rescan the image to check for these vulnerabilities as Sysdig already knows exactly what the image contains, so you only need to re-evaluate it against the updated database. You can do this by clicking 'Re-evaluate' button on the top right of the screen.

2. Under "Scan Policy" on the left menu click on "DefaultPolicy".
   You will see two tabs - 'Evaluation' and 'Rules'. Let's look a little more about what we mean by policies and rules.

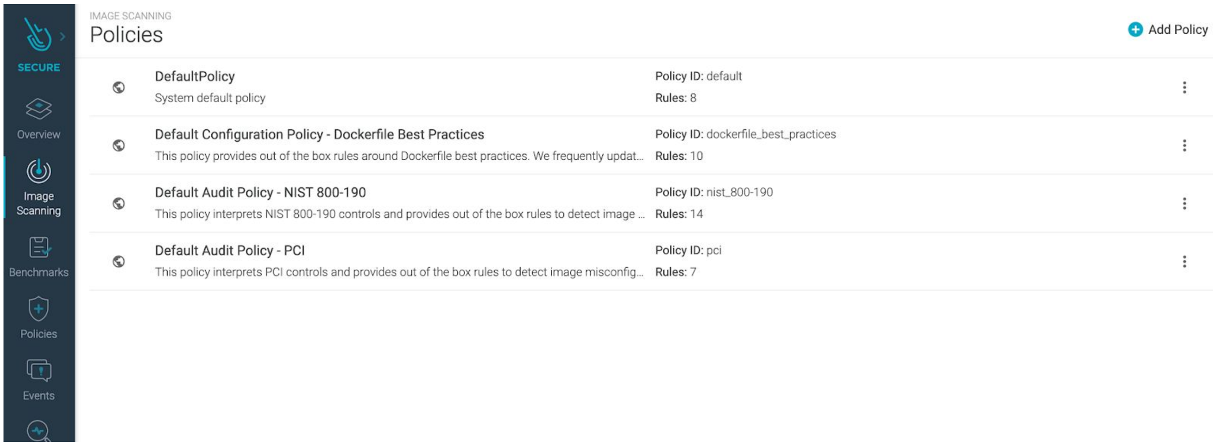
Task 3: Assign the Nginx Application a Policy & Re-evaluate

Role: DevSecOps Engineer

Scanning Policies

Policies define the rules used to detect anomalies in an image, the actions that should be taken or the notifications that should be sent if the policy rule is breached.

1. Click 'Image Scanning > Scanning Policies > Policies' to see the list of policies.
   There are four default policies available out of the box. These are
   - DefaultPolicy
   - Default Configuration Policy - Dockerfile Best Practices
   - Default Audit Policy - NIST 800-190
   - Default Audit Policy - PCI

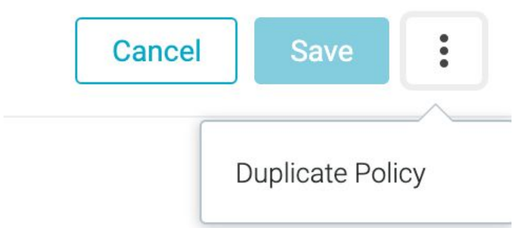


By default, all images will get scanned against the DefaultPolicy.

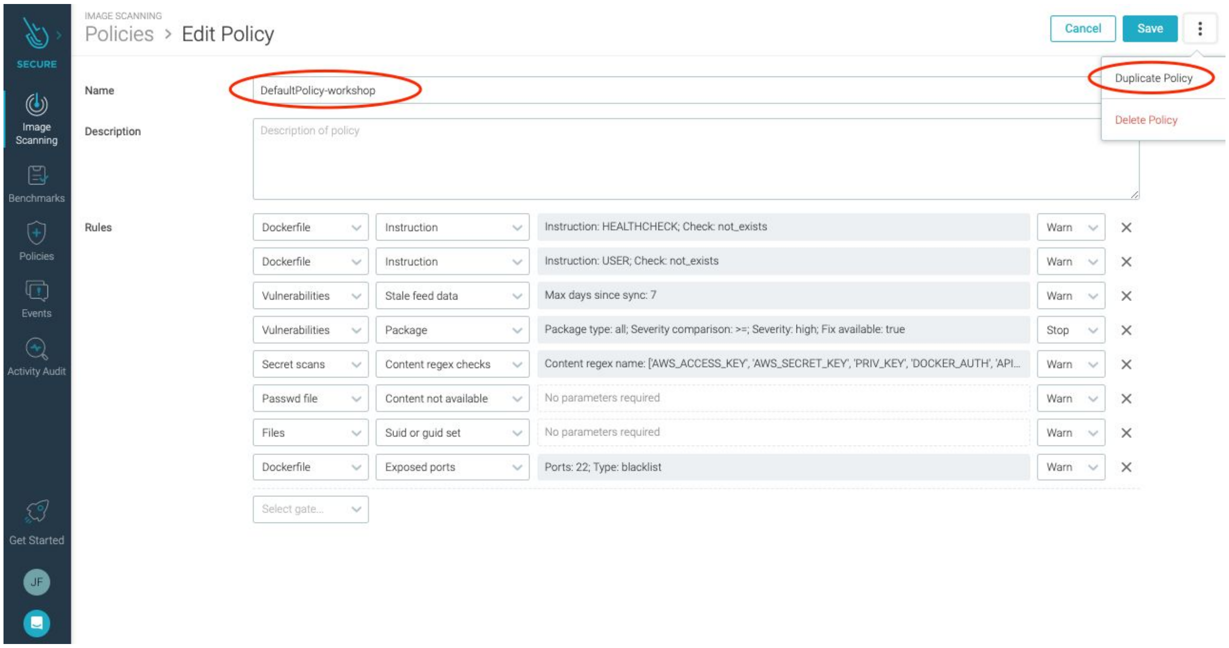
2. Click on 'DefaultPolicy'.
   You'll see the policy is made up of a number of rules, each relating to different aspects of the image – Dockerfile settings, files, packages, licenses, etc. Each of these 'gates' can have triggers, or parameters, to refine the rule.
   Setting the rule to 'Stop' means the scan will fail if this condition is met. Please note, it does not actually stop the container.
3. Let's change the scanning policy. As an example, we would like the scan to fail if the file '**/etc/passwd**' contains an entry for '**irc**', as we do not want IRC traffic on our container.

All the default policies are read only, but you can edit a duplicate of them.

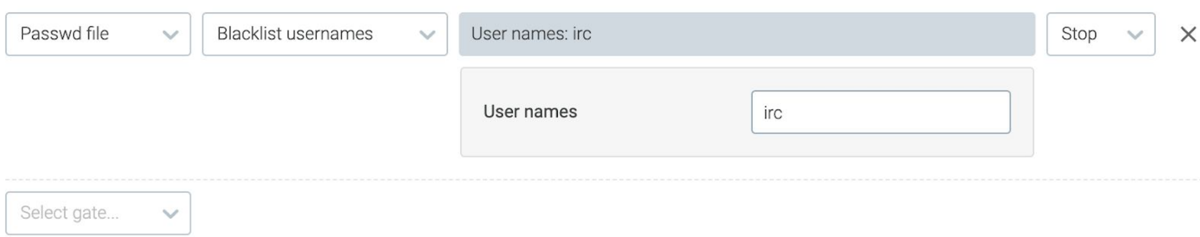
4. Click the vertical ellipsis on the top right, then 'Duplicate Policy'.



5. Call the duplicate policy '**DefaultPolicy-workshop**'.



6. At the bottom click 'Select gate', then add the information illustrated below, then click Save.



Note: Setting the action to '**Stop**' means a build pipeline will be stopped when the rule is breached.

7. Click '**Save**'.

So how do we ensure that **'DefaultPolicy-workshop'** is the policy that is used during the scan? For this, we define "Policy Assignments"'.

Policy Assignments

1. Navigate to 'Image Scanning > Policy Assignments'.

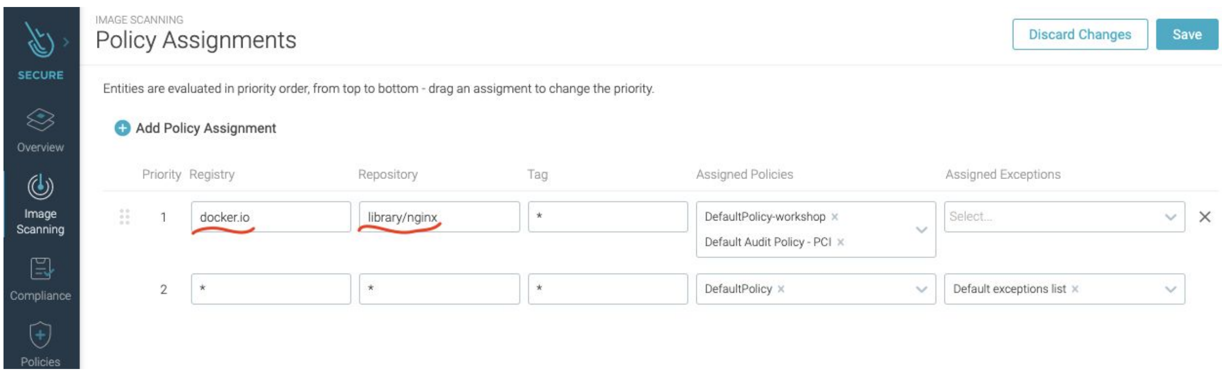
'Policy Assignments' define the policies that are implemented during a particular scan. In our case, we want to implement our new **'DefaultPolicy-workshop'** policy.

Let's also add the **'Default Audit Policy – PCI'** policy to scan all images for PCI Compliance.

2. From the 'Image Scanning' screen, click on 'Scanning Policies', then 'Policy Assignment'.
3. Click 'Add Policy Assignment'.
4. Configure as follows:
   - **Registry** - 'docker.io'
   - **Repository** – 'library/nginx'
   - **Tag** - '*'
   - **Assigned Policies** - 'DefaultPolicy-workshop' and 'Default Audit Policy – PCI' policies

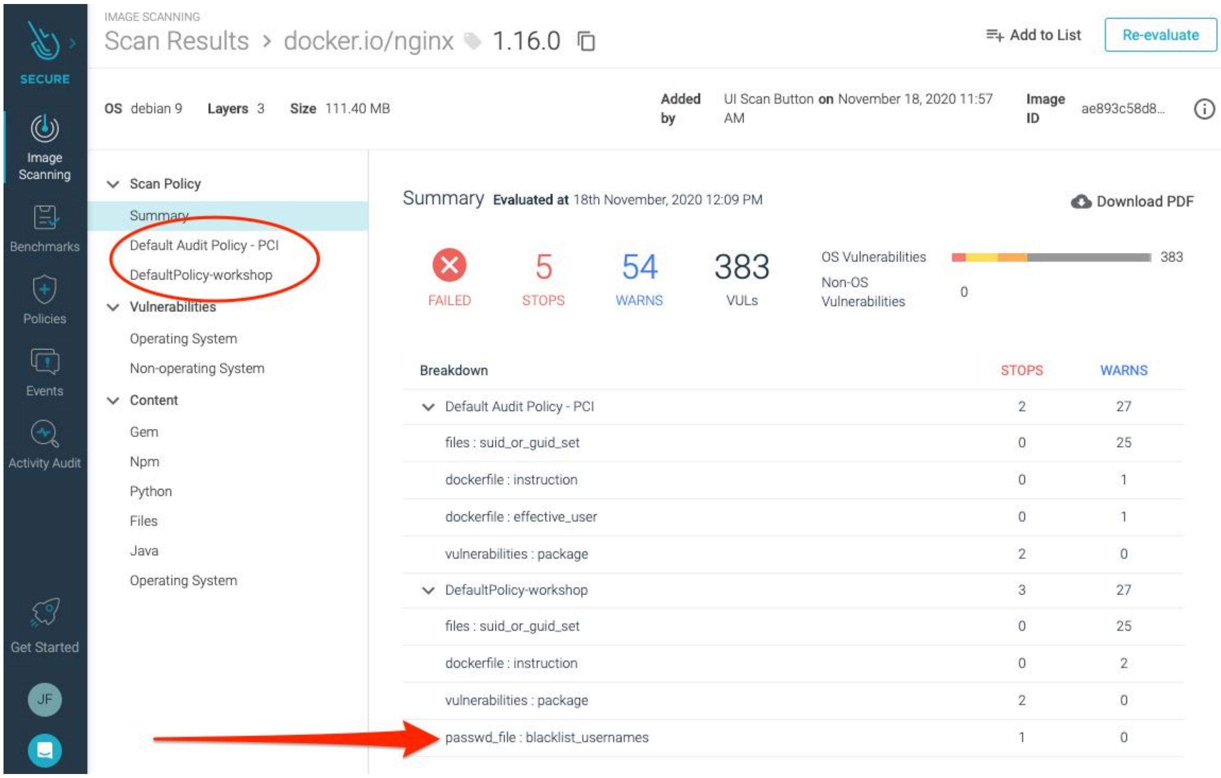
   The registry or repository does not have to refer to a specific image, but a wildcard ('*') can be used in either of these fields.

****NOTE**** Make sure you set **Registry** & **Repository** as indicated, otherwise, it will globally affect your entire OpenShift cluster!!



These policies are implemented in order, and once a Registry, Repository, or Tag match is found then, the subsequent policies are not evaluated. The DefaultPolicy is a catch-all that is implemented as a last resort.

5. Click '**Save**'.
6. Now go to 'Image Scanning > Scan Results' and find your Nginx image, and click on it, then click 'Re-evaluate'.



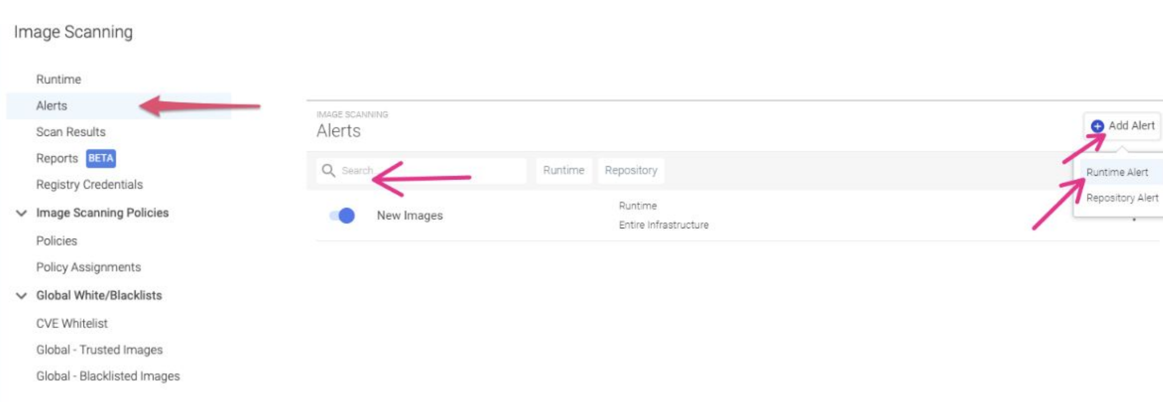
Note what just happened there. The image had already been scanned and the metadata stored within Sysdig. So when the scan parameters changed, we didn't have to rescan the image, but just reevaluated it against the stored metadata.

Task 4: Create an Alert

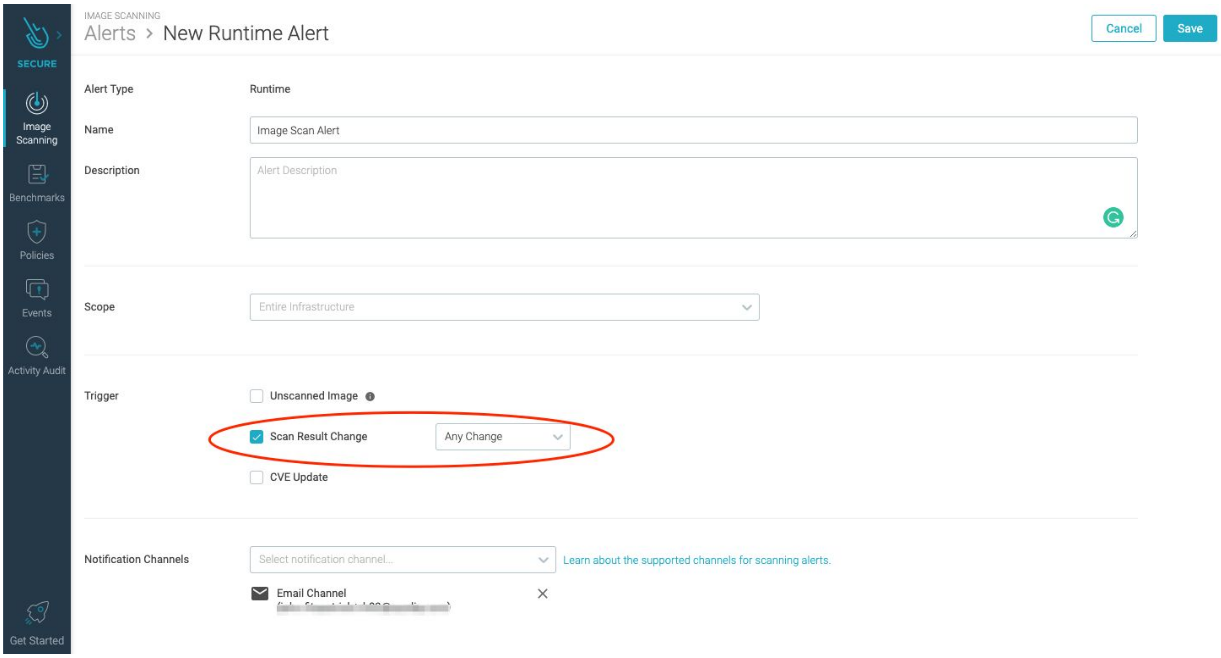
Role: DevSecOps Engineer

You can set up an alert that detects any new image that hasn't been scanned, and create an action that automatically scans the image.

1. Click on 'Image Scanning' -> 'Alerts', then 'Add Alert' -> 'Runtime'.



2. Fill out the form, make sure to select a trigger to ‘Scan Result Change’ and set it to ‘Any Change’. If you like you can also set a notification to email yourself.



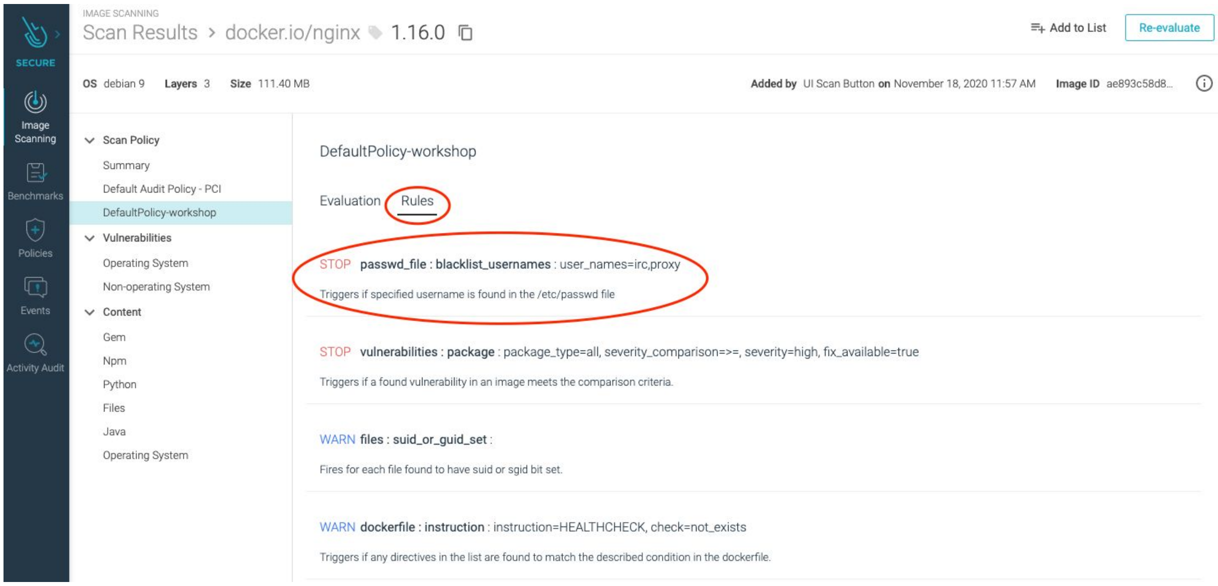
3. Click ‘**Save**’.

Update Policy to Trigger Alert

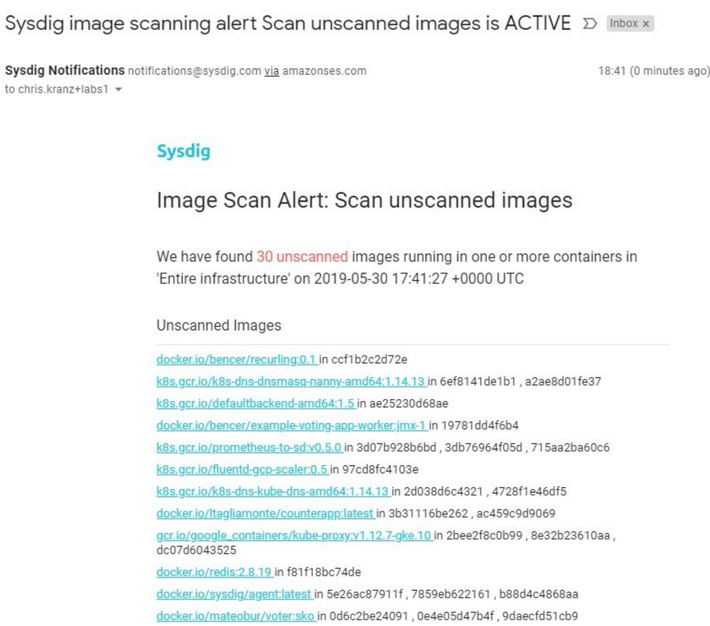
1. Now let's update the rule we added to policy ‘DefaultPolicy-workshop’ to include ‘proxy’.

2. Click ‘**Save**’.
3. Now go back to ‘Image Scanning > Scan Results’ and click on ‘docker.io/library/nginx’ image.
4. Click on “Re-evaluate”.
5. Now click on ‘**DefaultPolicy-workshop**’ and then click on the ‘**Rules**’ tab.

You will see the new trigger has been implemented.



In a minute or two, you should get an email notification an image scan has failed.



Lab 4: Secure your Image Pipeline with Inline Scanner

Description:

Runtime security ensures the containers currently running adhere to your compliance and corporate regulations. However, prevention is better than cure, so you will want to catch issues during the development phase before they go into production.

Sysdig's ImageVision technology identifies vulnerabilities and misconfigurations by automating scanning within CI/CD pipelines. It also blocks vulnerabilities pre-production and helps you map critical vulnerabilities back to an application and dev team.

Objectives:

After completing this lab, you will be able to:

- Install Jenkins & Log In
- Configure Jenkins Credentials
- Configure Jenkins Pipeline
- Run the Jenkins Pipeline
- View Results in Sysdig Secure

Task 1: Install Jenkins & Log In

Role: DevOps Engineer

1. Copy and paste the following code to the command line to install Jenkins.

```
$ oc new-project myproject  --description="Project for Jenkins lab"
--display-name="myproject"

$ oc new-app jenkins-ephemeral
```

Jenkins will take a few minutes to install.

2. Whilst installing, you will see a `jenkins-1-deploy` pod.

```
$ oc get pods
NAME                     READY   STATUS        RESTARTS   AGE
jenkins-1-deploy         1/1    Running        0            2m10s
```

```
jenkins-1-pmz2s          0/1   ContainerCreating   0          2m4s
```

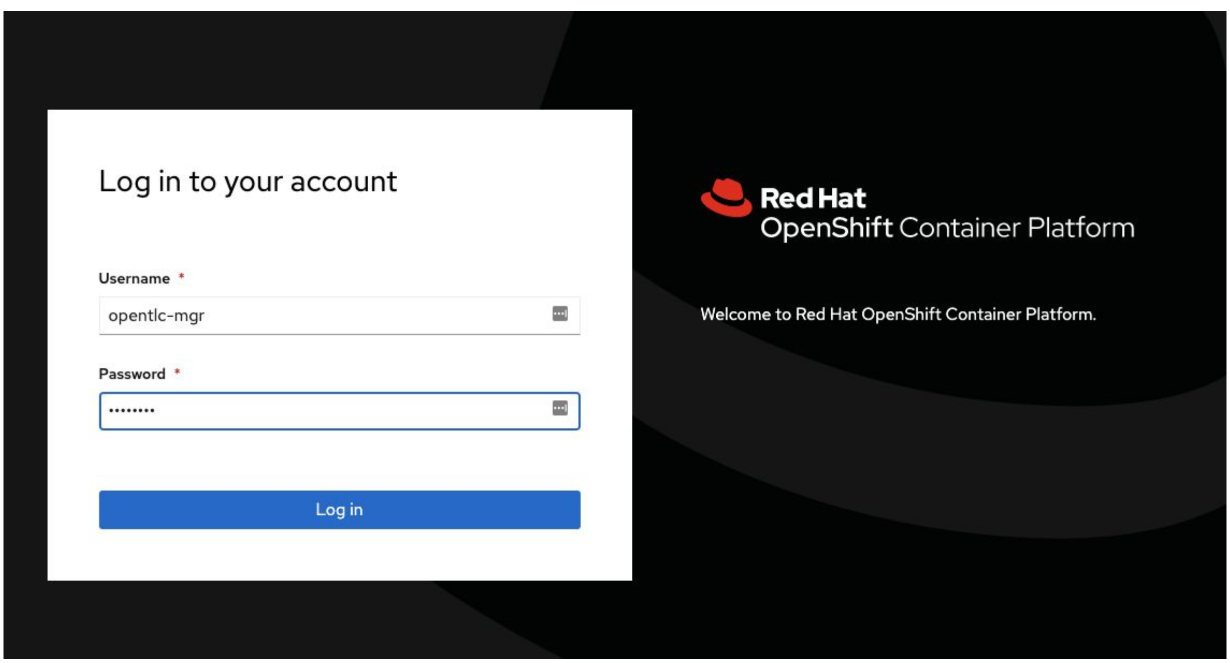
3. The Jenkins server will be available when the 'jenkins-1-deploy' has disappeared and the other jenkins pod is running.

```
$ oc get pods
NAME                     READY   STATUS             RESTARTS   AGE
jenkins-1-deploy         0/1   Completed           0          3m59s
jenkins-1-pmz2s          1/1   Running              0         3m53s
```

4. Once it's ready, run the following command to get your URL.

```
$ oc get routes
NAME   HOST/PORT                                                      PATH
SERVICES    PORT    TERMINATION  WILDCARD
jenkins   jenkins-myproject.apps.cluster-1e5a.1e5a.example.opentlc.com
      jenkins      <all>    edge/Redirect    None
```

5. Browse to the host/port (e.g. jenkins-myproject.apps.cluster-1e5a.1e5a.example.opentlc.com).
6. Click 'Log in with OpenShift' and log in using the 'cluster admin' credentials.



You can accept any warnings about the connection not being secure.

Task 2: Configure Jenkins Credentials

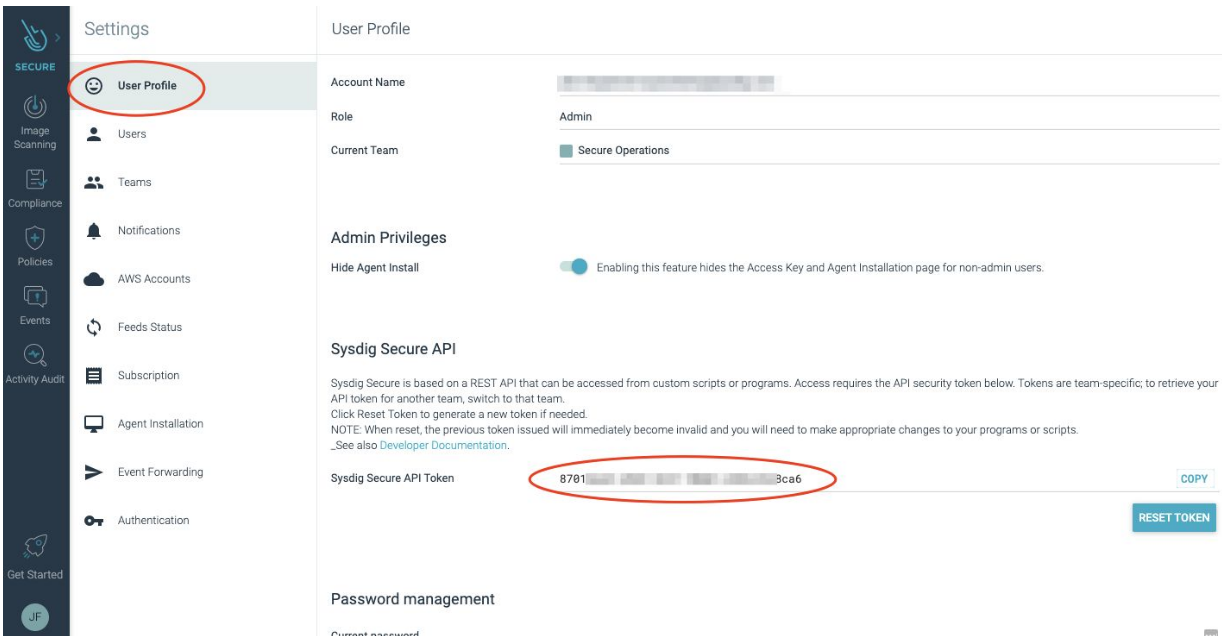
Role: DevSecOps Engineer

The pipeline pulls scanning policies from, and posts scan results to, Sysdig Secure. So, you must provide your Sysdig Secure API key as a credential within Jenkins.

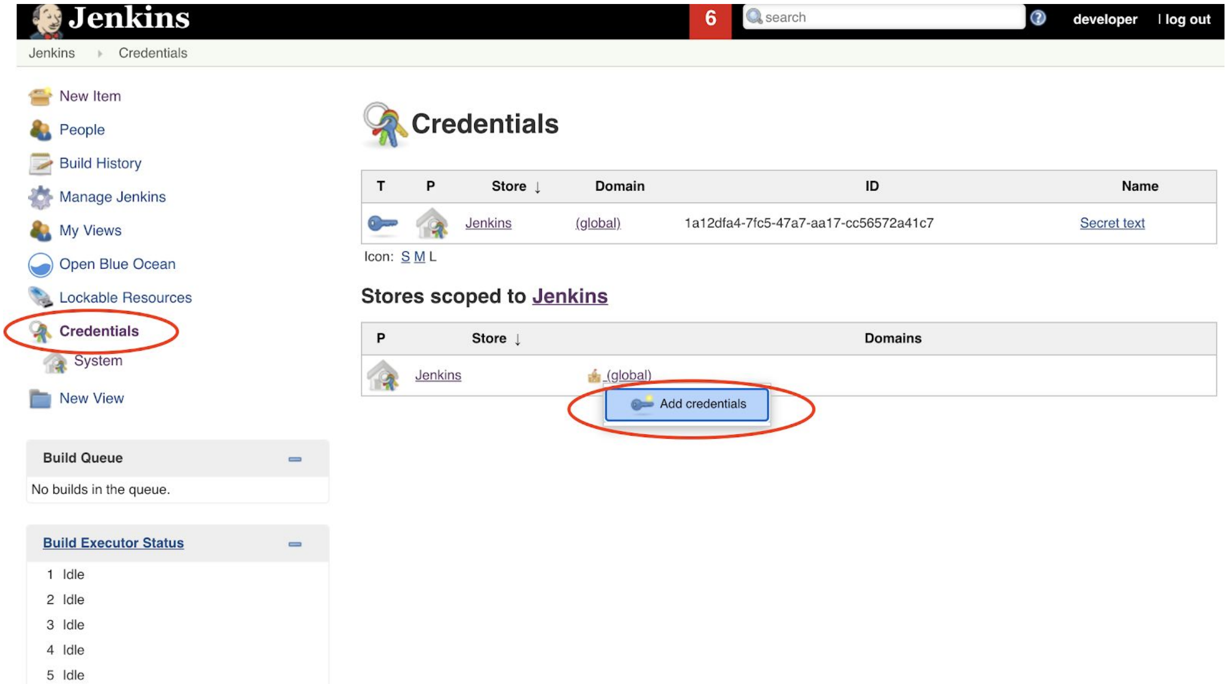
Note: The steps below require your Sysdig Secure API key which you used previously.

Before we configure Jenkins, we need to get our Sysdig Secure API key

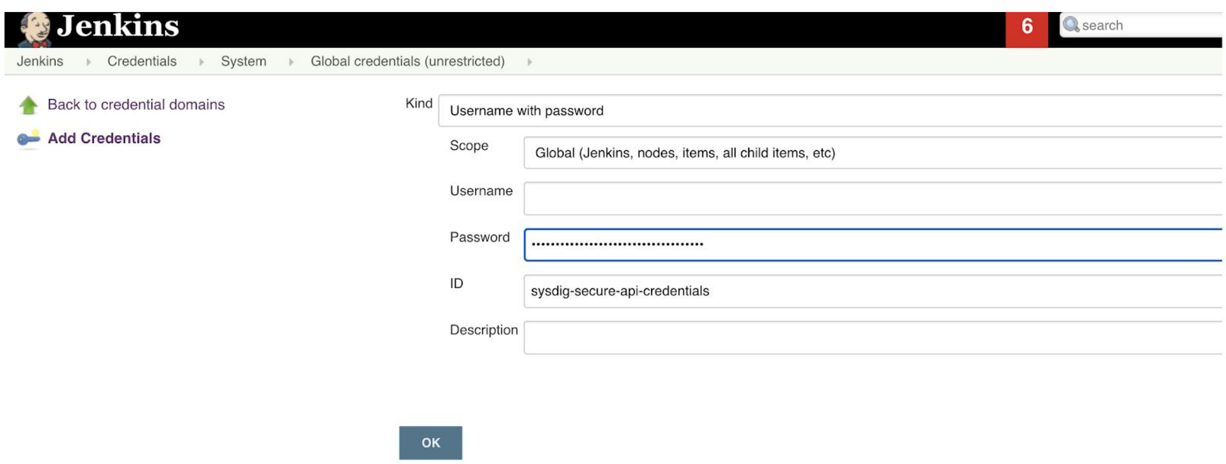
1. Browse to Settings > User Profile and copy your **API Key.**



2. Now, under 'Jenkins' click on 'Credentials'.
3. Under '(global)', click 'Add credentials'.



4. In the `Kind` field select "Username with password".
5. Leave 'Scope' as 'Global'.
6. Leave the Username field empty.
7. Enter the API key as the Password.
8. In the ID field, enter the text '**sysdig-secure-api-credentials**' (Note, it must be this exact text).



9. Click **OK.**

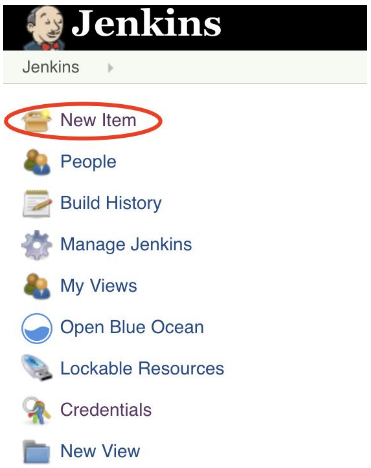
Task 3: Configure Jenkins Pipeline

Role: DevSecOps Engineer

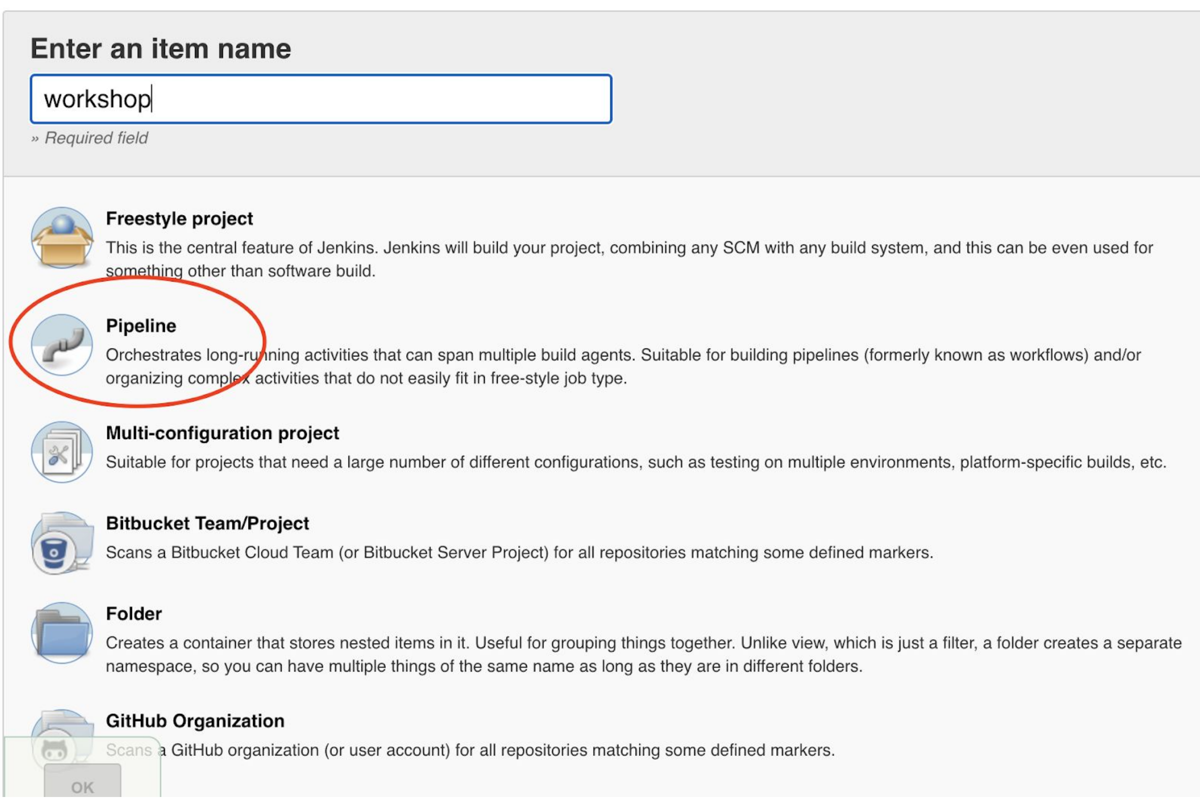
Now we are going to create the pipeline that will scan an image.

Although the plugin is able to scan locally built images, this example will pull an existing image from a repository, scan it locally, and send the results back to Sysdig Secure.

1. From the frontpage of Jenkins, click on **New Item.**



2. Give it the name '**workshop**'.
3. Select **Pipeline** as type:



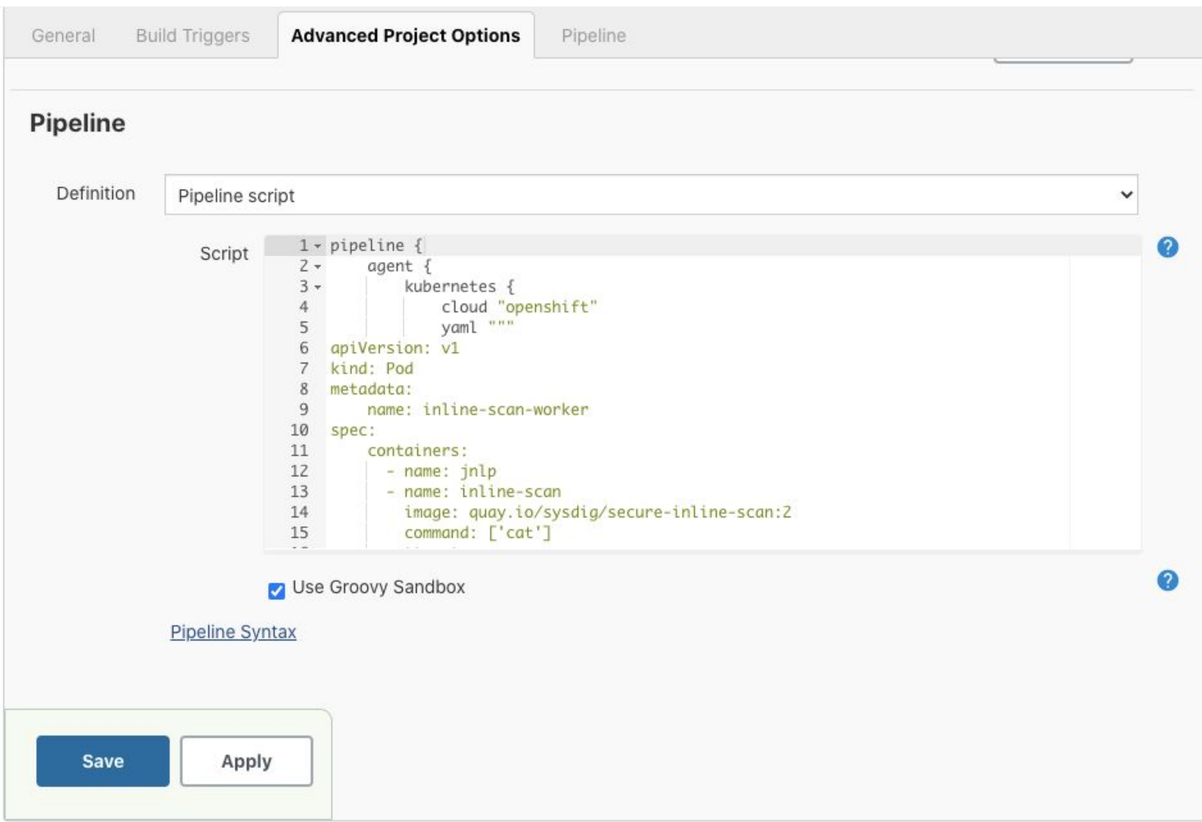
4. Click **OK.**

5. Scroll down to the **Pipeline** section.
6. Leave the 'Definition' as *Pipeline script*.
   We will use the following Jenkins pipeline configuration file.

```
pipeline {
      agent {
      kubernetes {
            cloud "openshift"
            yaml """
apiVersion: v1
kind: Pod
metadata:
      name: inline-scan-worker
spec:
      containers:
      - name: jnlp
      - name: inline-scan
      image: quay.io/sysdig/secure-inline-scan:2
      command: ['cat']
      tty: true
"""
      }
   }
      parameters {
      string(name: 'IMAGE_NAME', defaultValue: 'sysdiglabs/dummy-vuln-app',
description: 'Name of the image to be built andscanned (e.g.:
myrepo/dummy-app)')
      string(name: 'BACKEND', defaultValue: 'https://secure.sysdig.com',
description: 'Secure Backend Endpoint (e.g. https://us2.app.sysdig.com or
https://eu1.app.sysdig.com)')
      }
      environment {
      SECURE_API_KEY = credentials('sysdig-secure-api-credentials')
      }
      stages {
      stage('Scanning Image pulled from repository') {
```

```
            steps {
            container("inline-scan") {
                  sh "/sysdig-inline-scan.sh --sysdig-url '${BACKEND}' -k
${SECURE_API_KEY_PSW} ${IMAGE_NAME}"
            }
            }
      }
   }
}
```

Copy this pipeline script & paste it into the **Script** field (available from this gist **https://tinyurl.com/y9hlwbeg**).



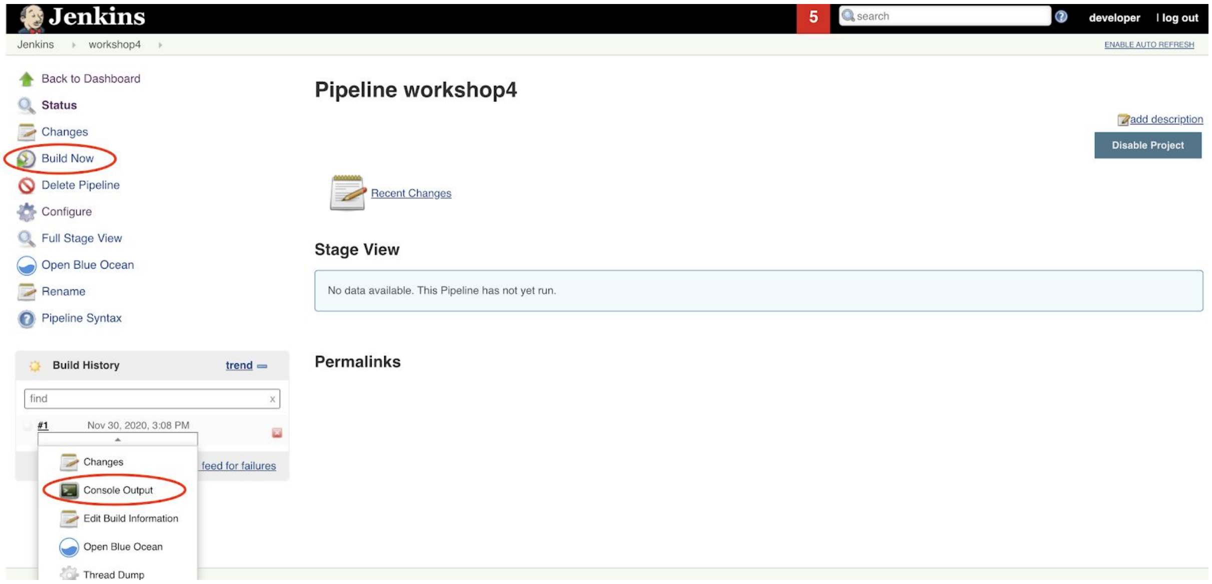
7. Click **Save.**

Task 4: Run the Jenkins Pipeline

Role: DevSecOps Engineer

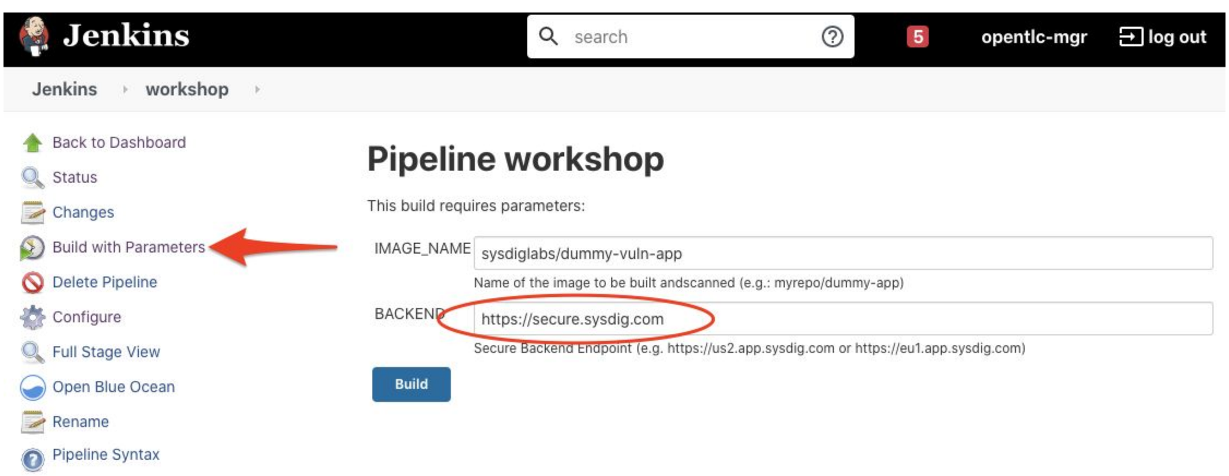
Now we are going to run the pipeline and watch as the image gets scanned.

1. Click 'Build Now'.
2. Click **Console Output** beneath the build history to watch progress of the build.

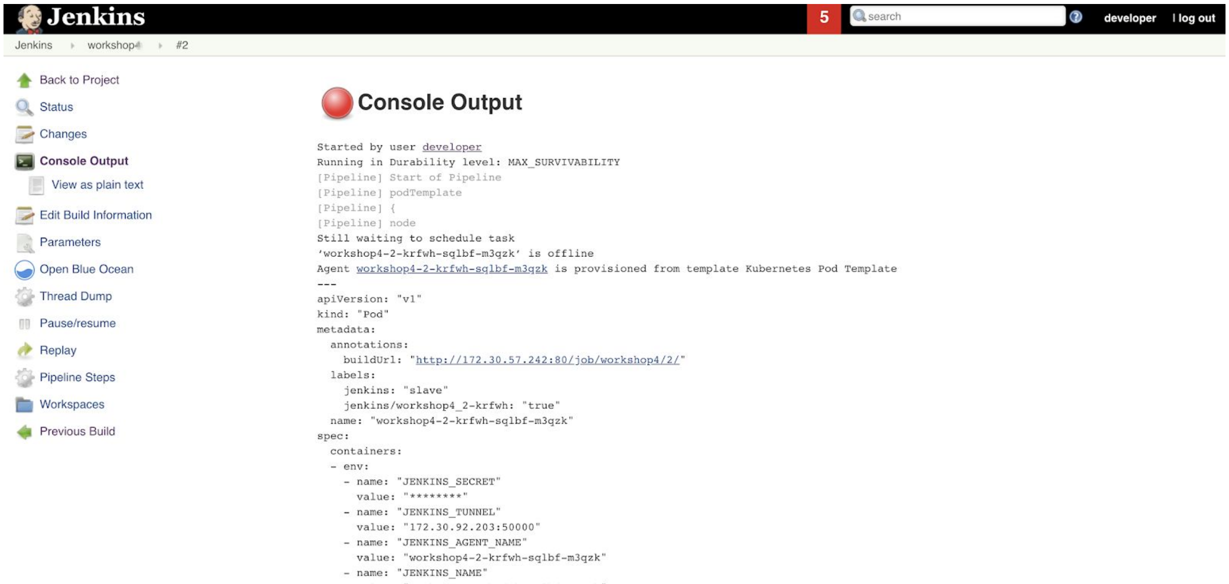


Please Note, this **FIRST BUILD WILL FAIL**, due to a known issue with parameterised builds in Jenkins - See **https://tinyurl.com/y229yo4e**.

3. Once the initial build completes, go back to the Dashboard and click 'Build with Parameters'.
4. In the 'Pipeline workshop' window, enter your Sysdig Secure url as appropriate based on your account, i.e. one of the following:
   - https://secure.sysdig.com
   - https://us2.app.sysdig.com
   - https://eu1.app.sysdig.com



5. Click **Build.**



6. Scrolling down you will see details relating to the image scan, and results getting sent to Sysdig Secure, for example.

```
+ /sysdig-inline-scan.sh -s https://us2.app.sysdig.com -k ****
sysdiglabs/dummy-vuln-app
Inspecting image from remote repository -- sysdiglabs/dummy-vuln-app:latest
  Full image:  docker.io/sysdiglabs/dummy-vuln-app
  Full tag: docker.io/sysdiglabs/dummy-vuln-app:latest
  Repo digest:
sha256:bc86e8ba5741ab71ce50f13fbf89a1f27dc4e1d3b0c3345cee8e3238bc30022b
  Image id:
b670c067178c876d17363baec279d483ae07384351d1a0be7646230442471ac6


Image digest found on Sysdig Secure, skipping analysis.
Trying to add alias for image
Added successfully alias tag for image


Scan Report


Last Evaluation: 2020-11-30T14:12:45Z
Final action:      stop
```

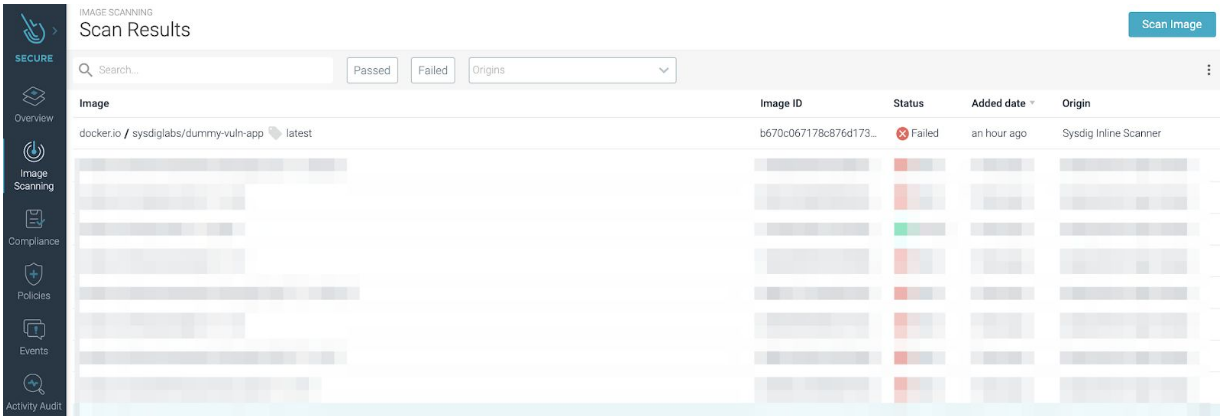
```
Status:            fail

Evaluation results
 - warn dockerfile:instruction Dockerfile directive 'HEALTHCHECK' not found,
matching condition 'not_exists' check
 - warn dockerfile:instruction Dockerfile directive 'USER' not found,
matching condition 'not_exists' check
 - warn dockerfile:exposed_ports Dockerfile exposes port (22) which is in
policy file DENIEDPORTS list
 - stop vulnerabilities:package HIGH Vulnerability found in non-os package
type (python) - /usr/share/pyshared/pyxdg (fixed in: 0.26)(VULNDB-204097 -
http://sysdigcloud-anchore-core:8228/v1/query/vulnerabilities?id=VULNDB-2040
97)
 - stop vulnerabilities:package CRITICAL Vulnerability found in non-os
package type (python) - /usr/lib/python2.7/dist-packages/pyasn1 (fixed in:
0.2.1)(VULNDB-205522 -
http://sysdigcloud-anchore-core:8228/v1/query/vulnerabilities?id=VULNDB-2055
22)
 - stop vulnerabilities:package HIGH Vulnerability found in os package type
(dpkg) - linux-libc-dev (fixed in: 4.9.240-1)(CVE-2019-19074 -
https://security-tracker.debian.org/tracker/CVE-2019-19074)
```

Task 5: View Results in Sysdig Secure

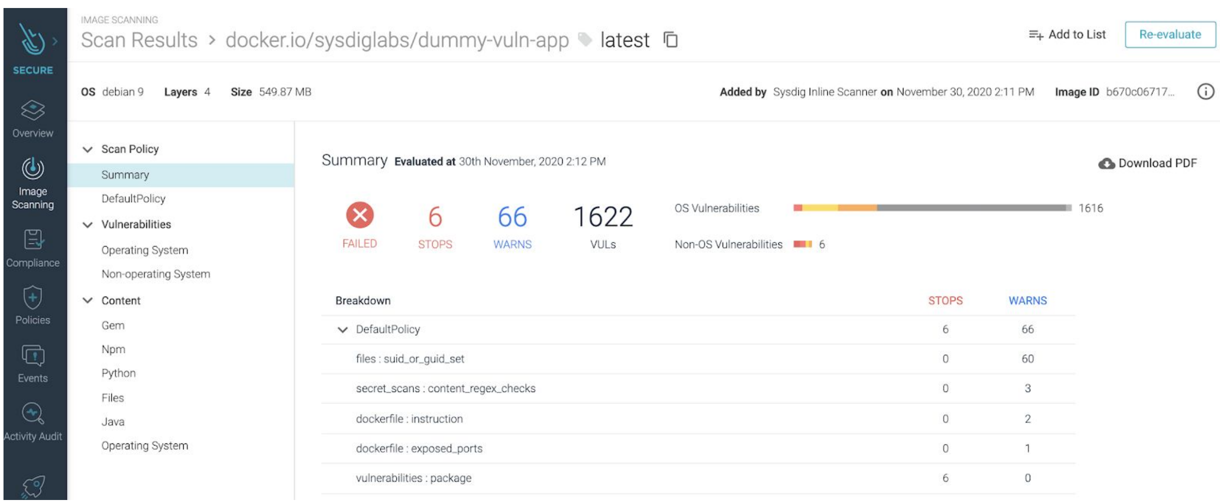
Role: DevSecOps Engineer

Now view the results of the scan in Sysdig.

1. In Sysdig Secure UI, click on **'Image Scanning'** on the left side menu, then **Scan Results.**
2. Look for details of the previous scan, the Origin being **Sysdig Inline Scanner.**



3. Click on this entry to see details of the scan.



Appendix 1: Image Scanning Technical Description

There are two general approaches to scanning images in Sysdig - backend scanning or inline scanning. The reasons why you might choose one over the other is best explained by an understanding of how scanning works under the hood.

With Sysdig, there are two phases in scanning an image.

1. Analysis of contents.
2. Evaluation against policies and vulnerabilities.

Phase 1 - Analysis of Contents

During the analysis phase of the scan, the worker first loads the image. Each image is built upon a series of other images, called 'layers', each one referred to by specific SHA value in the manifest file. Each layer is pulled down, they are then "flattened", and the composition of all layers is analysed and a complete list of all the files, packages, package versions, etc. across all the layers is generated.

This phase of the scan is quite process-heavy and accounts for approximately 90% of the Time/CPU consumed during the entire scan, and the output of this is a JSON document containing metadata on all aspects of the image.

During an inline scan, this phase happens in the image's environment - in a CI/CD pipeline, by an Admission Controller, or even on a Kubernetes worker node, as illustrated below.

Going A Little Deeper...

The scanner process uses the Anchore engine and runs between 15 and 20 different analysers, each performing different functions. Some of these analysers create a bill of materials for the image (operating system and non-operating system packages), while others retrieve file contents, file sizes, search for secrets, open ports, etc. Each analyser generates a JSON report containing metadata, all of which are then combined.

To give you a better idea of how this looks, below is a sample of the data generated in the analysis phase.


```
{
   "document": [
      {
      "image": {
      "imageId":
```

```
"f35646e83998b844c3f067e5a2cff84cdf0967627031aeda3042d78996b68d35",
      "imagedata": {
            "analysis_report": {
            "analyzer_meta": {
            "analyzer_meta": {
                  "base": {
                   "DISTRO": "debian",
                  "DISTROVERS": "10",
                  "LIKEDISTRO": "debian"
                  }
            }
            },
            "file_checksums": {
            "files.md5sums": {
                  "base": {
                  "/bin": "DIRECTORY_OR_OTHER",
                  "/bin/bash": "4600132e6a7ae0d451566943a9e79736",
                  "/bin/cat": "44b8726219e0d2929e9150210bfbb544",
                  "/bin/chgrp": "2befb2d66eee50af3fd5eb0b30102841",
                  "/bin/chmod": "737ae4345da6e93c44fe9b11b12defe1",
                  "/bin/chown": "8680c8e619194af847c009a97fd4ebe2",
                  "/bin/cp": "d38d5be99452fb23cce11fc7756c1594",
                  "/bin/dash": "895aea5b87d9d6cbd73537a9b2d45cff",
                  "/bin/date": "b175b76c42bf04d764f3f5d7e4f3c69c",
                  "/bin/dd": "1f90de0a1b75febeda1936a1ed9e1066",
                  "/bin/df": "b50d93d2ab75977d129baf0078becb96",
                  "/bin/dir": "3c76bcda677ed3ff9901d6e770ebca3d",
                  "/bin/dmesg": "ea95ebcd2794014a5f933f7b6434e31c",
...
```


You can see each binary, file etc is specifically referenced with a md5 hash next to each entry which relates to its version etc.

The results of this analysis are combined into a single JSON analysis report, or metadata document, and it is this metadata document that is used to evaluate the scan against the defines security policy.

For a backend scan, this process all happens within Sysdig, however during an 'inline scan', this metadata is then forwarded to the Sysdig Backend for evaluation using the API - this is the reason you must supply the **Sysdig Secure API Token** later in this workshop!

Phase 2 - Evaluation

Once the Sysdig Backend retrieves the metadata it is checked against the assigned policy definitions as well as the vulnerability database. A Sysdig Secure policy is a combination of rules about activities an enterprise wants to detect in an environment, the actions that should be taken if the policy rule is breached, and potentially the notifications that should be sent. These policies are configured through the Sysdig UI, and may differ from image to image.

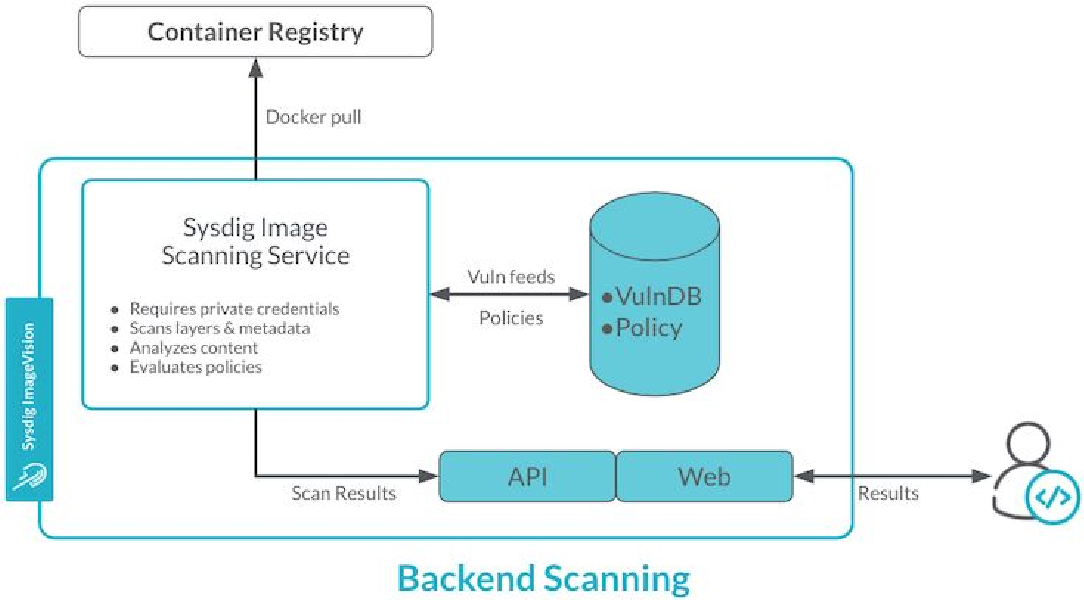
Individual policy rules may relate to

- Open ports
- File permissions
- Exposed passwords
- Etc.

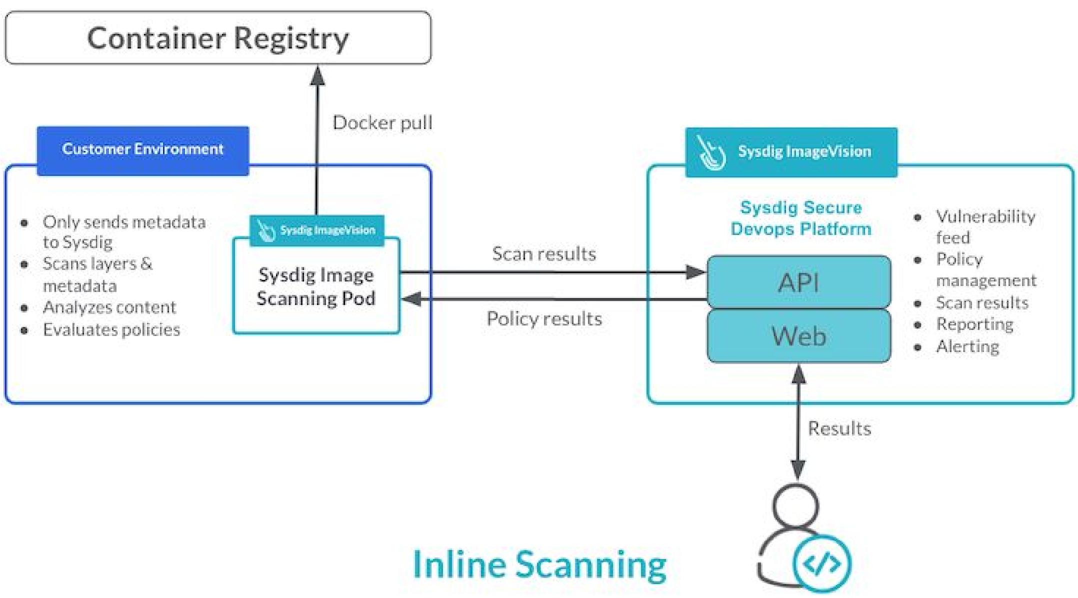
A number of policies are delivered out-of-the-box and can be used as-is, duplicated, or edited as needed. These relate specifically to 'PCI' and 'NIST 800-190' compliance, as well as general Dockerfile best practices. You can also create policies from scratch, using either predefined rules or creating custom rules.

Inline Scanning vs Backend Scanning

With Backend Scanning, both phases of the scan, i.e. the analysis of contents followed by the evaluation against policies and vulnerabilities, are performed on Sysdig's servers. This requires that images be transferred to Sysdig to be scanned and evaluated, therefore Sysdig must store your repository's credentials in order to have access to it. This may be problematic in a secure environment and/or when you're using SaaS.



However, with Inline Scanning the scan and subsequent analysis occur where the image resides, maybe as part of a CI/CD pipeline, or within your cloud environment, for example with ECR. In this case Sysdig does not need access to your repository and only the metadata is sent back to the Sysdig backend, hence no registry keys are exposed to Sysdig.



Inline Scanning is considered best practice and the better approach to scanning over backend scanning. The benefits of scanning inline include:

- Images don't leave their own environment

- SaaS users don't send images and proprietary code to Sysdig's SaaS service
- Registries don't have to be exposed
- Images can be scanned in parallel more easily
- Images can be scanned before they hit the registry, which can cut down on registry costs and simplify the build pipeline
- Existing scan metadata can be checked against new vulnerabilities, so images do no need to be rescanned, for example upon detection of zero-day vulnerabilities

End of Workshop!

Links/contacts for more information

Red Hat with Sysdig: https://sysdig.com/partners/red-hat/

Red Hat Marketplace: https://marketplace.redhat.com/en-us/products/sysdig-monitor

Red Hat Advance Cluster Manager with Sysdig:
https://www.openshift.com/blog/securing-kubernetes-clusters-with-sysdig-and-red-hat-advanced-cluster-management