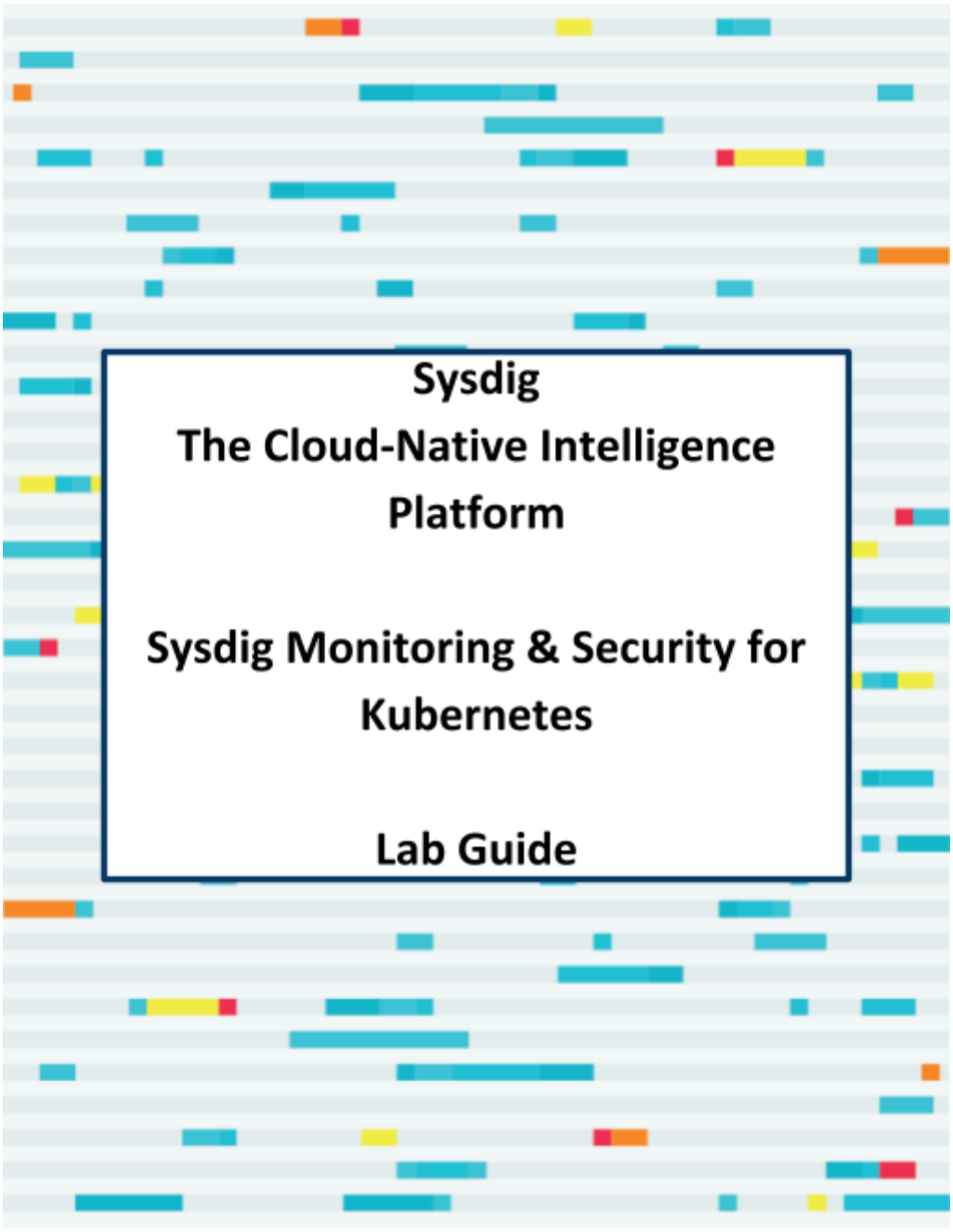

Sysdig
The Cloud-Native Intelligence Platform
Sysdig Monitoring & Security for Kubernetes
Lab Guide

Sysdig

Table of Contents

Document Control & Purpose

Document Control

Version	Release Date	Release Notes	Author	Reviewer
v0.1	25th March 2019	Internal Draft	Chris Kranz	
v1.0	4th April 2019	First Release	Chris Kranz	Brett Egloff Payal Chakravarty Jain Alex Lawrence
v2.0	31st May 2019	Second Release	Chris Kranz	Dan Papandrea Matthew Andersen
v2.1	12th July 2019	Minor Revisions	Chris Kranz	Eric Magnus Denis Maligin
v2.2	15th July 2019	Product Updates	Chris Kranz	

Document Purpose

The purpose of this document is to complement an instructor led training course delivered by a certified trainer. It is provided as a reference to an instructor led lesson, and also includes important details such as source file locations and connectivity details.

As the Sysdig Platform is constantly evolving and information changes, this document includes a lot of links to external resources which are updated each time a release introduces changes or new features. All efforts are made to keep this training course up-to-date with the latest available features and technical details, but sometimes there can be a delay.

Intended Audience

This document is intended for anyone looking to understand more about the Sysdig Platform through instructor led training. This document (and as such, the training course itself) is intended for individuals with an existing understanding of microservices, container technologies and container orchestrators. Primarily this course focused on Docker container runtime engine and Kubernetes container orchestrator, but many of the concepts are universal across runtimes and orchestrators. The audience is expected to have a good

understanding of the Linux command line and basic Linux troubleshooting (viewing logs, identifying IP addresses, etc.).

Introduction

Expected Competencies

The student needs to understand the basic concepts of microservices applications, containers and Kubernetes. They will be able to check the status of deployed Kubernetes resources (i.e. pods, deployments, services) and nodes, and be able to do basic container and Linux troubleshooting (i.e. tail logs, view IP addresses, start/stop/list containers, etc.). It is also helpful if the student is familiar with cloud services and Kubernetes ecosystem services like networking and storage.

Training Workshop Goals

- Deploy a new microservices application
- Explore a microservices application using Sysdig Monitor:
    - Where is each container running?
    - How does this microservices application work?
    - Which services talk to each other?
    - What's running inside each container and microservice?
    - How many nodes and containers do I have on my cluster?
- Monitor a web services application. Key metrics and monitoring approaches: Golden Signals, Utilization Saturation and Errors (USE) resource monitoring and Kubernetes orchestration.
- Identify performance bottlenecks in the application.
- Scale the application to handle the required load while monitoring the changes.
- Tailor Sysdig to provide application assurance for this microservices application
    - Custom dashboards
    - Alerts for proactive actions
- Dig deeper into the application using custom metrics from the following data sources:
    - Kubernetes state metrics
    - Prometheus instrumented metrics
    - Statsd
    - JMX
    - Events & custom events
- Authentication & Access Control - limit access to different resources

- Image scanning
- Compliance benchmarks
- Runtime Security
- Forensic Troubleshooting

Training Workshop Prerequisites

In order to run through this workshop, you will need the following:

- Access to Sysdig Monitor & Secure (trial accounts are fine, but Secure should be enabled also)
- A Kubernetes cluster deployed and running
    - This may have been provided to you, please check with your instructor
    - We recommend a minimum of 2-3 nodes running, approximately 2x vCPU and 8GB RAM
- If you are providing your own Kubernetes cluster, then you will also need the following:
    - Download Sysdig lab files local to where kubectl will be run
    - The kubectl CLI client, configured to connect to your cluster with admin privileges.
- If a Kubernetes cluster is provided for you, then kubectl and the lab files have been pre-configured also.

Please run through sections 1 & 2 of this lab guide before the workshop in order to confirm everything is setup and ready. Some setup tasks can take a while, so it is best to have them ready before the workshop starts.

Agenda

Timings	Runtime	Chapter	Chapter	Content Type
9:00	0:15	Welcome & Introductions	n/a	Slides
9:15	0:15	The Industry is Changing	n/a	Slides
9:30	0:15	Lab Access and Pre-Requisites	1 & 2	Slides & Workbook
9:45	0:15	Deploying the Sysdig Agent	3	Slides & Workbook
10:00	0:15	Coffee Break		
10:15	1:00	Lab 1: Application Health	4	Slides & Workbook

11:15	0:30	Lab 2: Digging Deeper	5	Slides & Workbook
11:45	0:30	Lab 3: Deployment Health	6	Slides & Workbook
12:15	0:30	Lab 4: Authentication and Access Control	7	Slides & Workbook
12:45	1:00	Lunch		
13:45	1:00	Lab 5: Securing your Images	8	Slides & Workbook
14:45	0:30	Lab 6: Containerised Compliance	9	Slides & Workbook
15:15	1:00	Lab 7: Runtime Security	10	Slides & Workbook
16:15	0:15	Coffee Break		
16:30	1:00	Lab 8: Forensic Troubleshooting	11	Slides & Workbook
17:30		Q&A into Finish		

Scenario

You work for a large pet shop retailer with stores across the country. You have been asked with working with an external web agency to deploy a business critical application for understanding customer preference.

In this scenario we will deploy a microservices app called 'voting-app' in Kubernetes that issues, collects, and stores votes for your favourite: dogs or cats. Results are stored in a database and then a report can be generated with the results. The example application generates its own traffic in order to simulate actual users.

Suddenly the app becomes slower and slower. In this lab we will find the microservices bottleneck and check how we can scale this application to handle the required load. The external agency is offshore in a different timezone so you have no direct contact with them at the moment, but the business is putting pressure on you to resolve the issues at hand as this application is of critical business importance.

You are then told by the project lead that you need to own a 'java-app'. Once again, you know little about it and need to create some operational guides to support this application. The

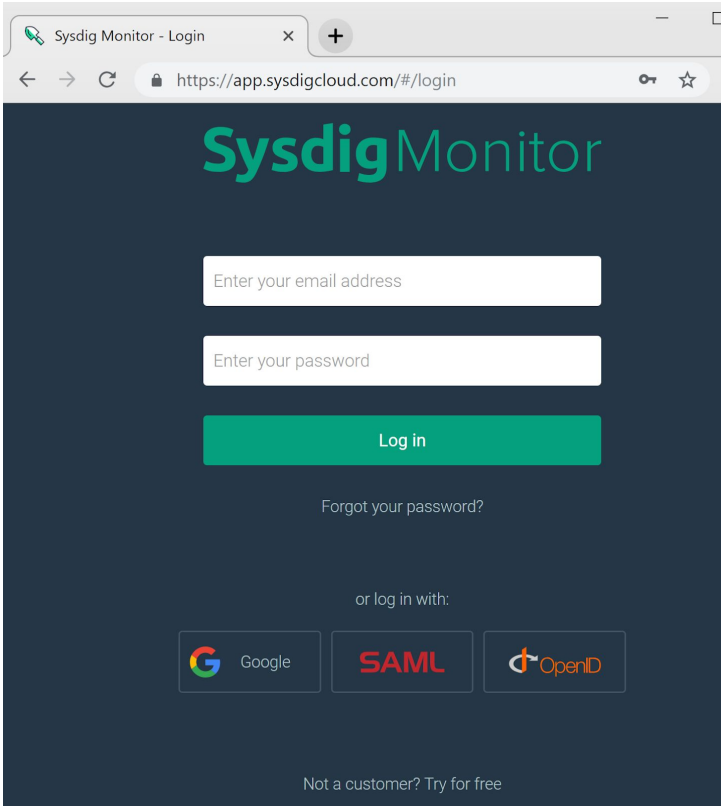
application is still under development, but the business is looking to have this operationally ready in the next few weeks.

The company is looking to gain ISO27001, so a security audit is looming. You need to make sure the platform and applications (even the ones you have no idea about), are secure and protected.

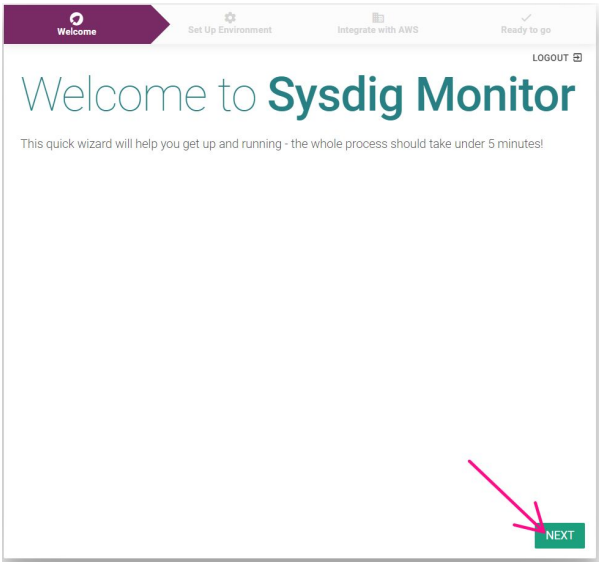
A web application is rushed out the door for a new marketing initiative, so check to ensure it hasn't got any holes or issues. If any issues are detected, ensure you can gather evidence to definitively identify and report the issues, as well as prevent them going forward.

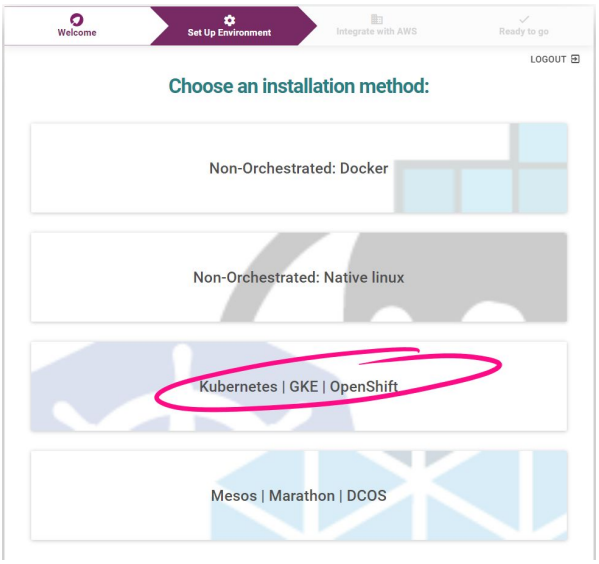
1. Logging into Sysdig

Point your browser at https://app.sysdigcloud.com, and log into the account you are going to use for the training.

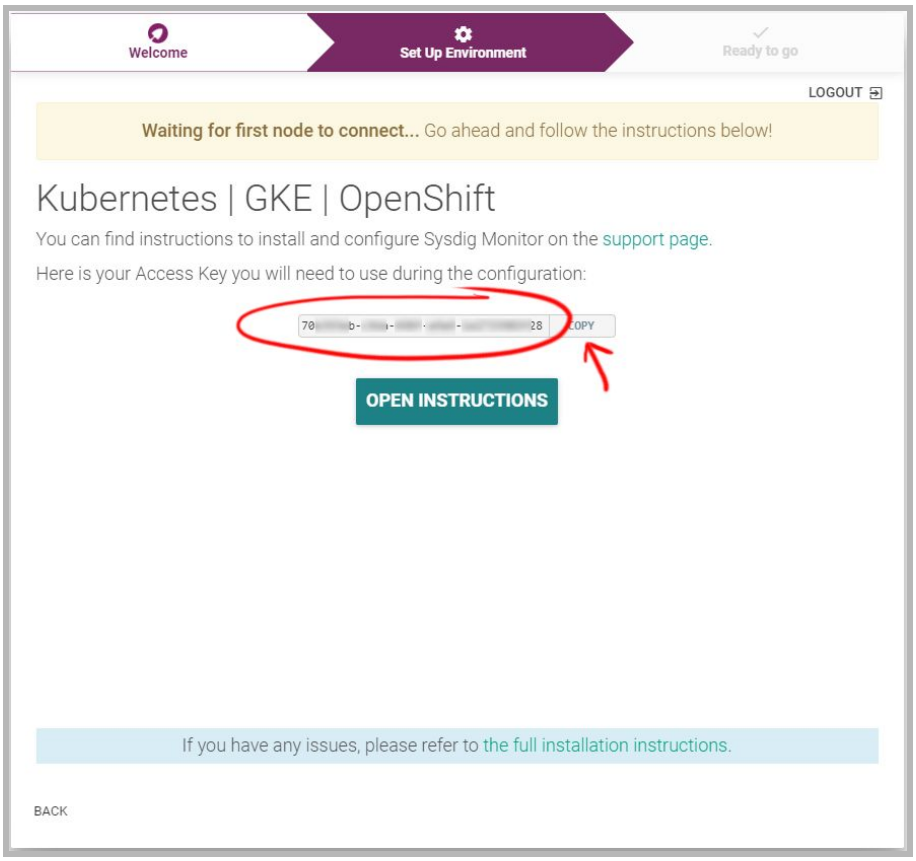


If this is the first time you've used this account, you will be greeted by the initial setup Wizard. This will walk you through installing your first agents. Select the 'Kubernetes | GKE | OpenShift' method button.





For now, we simply want to grab the Access Key, so run through the wizard (selecting 'Kubernetes | GKE | OpenShift') until you are shown your personal Access Key. Copy this somewhere you can recall quickly as we'll use it when installing the Sysdig Agent shortly. You can recover this key later if required. Leave this window/tab open as we'll return here to finish the installation later.



Once you have agents sending metrics into the Sysdig Platform, this wizard will prompt you to continue. You can leave this window open for now, we'll return to it once we have the Sysdig Agent installed.

2. Making sure we are prepared

If you don't have an existing Kubernetes environment to use, or one hasn't been provided to you, we recommend setting up a GKE cluster in Google Cloud. It's quick and the costs can be covered by the introductory credits.

2.1 Access provided Kubernetes cluster

If a Kubernetes cluster has been provided to you, use the SSH key that has also been provided to connect to the cluster.

2.2 Confirm access to kubectl

- Check that you are registered:

```
$ kubectl get nodes
```

- Check that your are admin:

```
$ kubectl auth can-i create node
                yes
```

2.3 Download Sysdig Lab resources

https://drive.google.com/drive/folders/1cpsfhZ7h0wyCN4Pb1eat18jSq8XaFUID?usp=sharing

OR

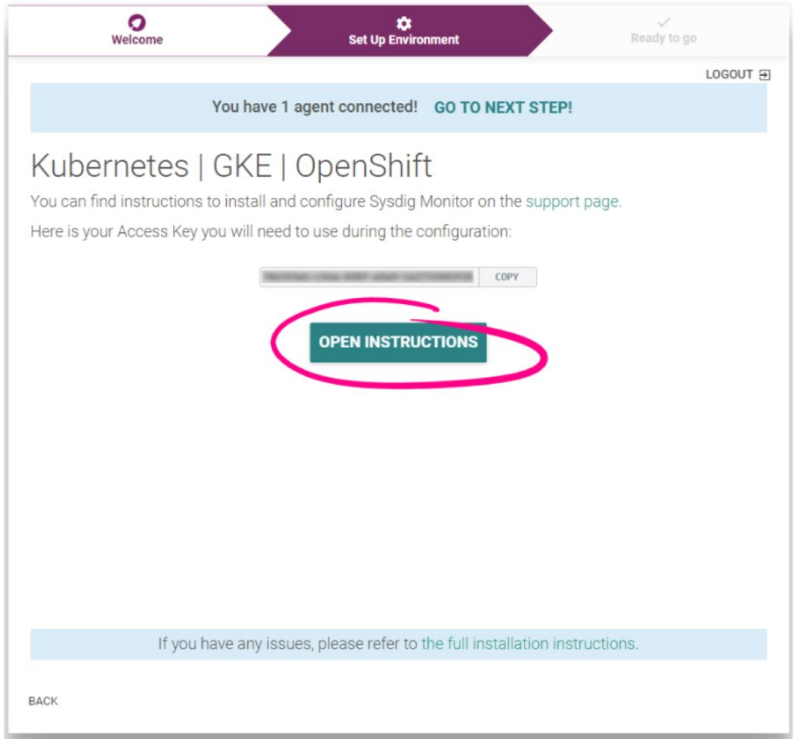
https://tinyurl.com/SysdigWizard

3. Deploying the Sysdig Agent

There are several ways to install the Sysdig agent in your Kubernetes cluster, but for this training the preferred method is via a Kubernetes Daemonset using the provided installation scripts. You will need your Sysdig Access Key, from what you noted down previously, from the previous screen we left open, or click on your user initials in the bottom left, go to Settings, then goto Agent Installation.

3.1 Kubernetes Daemonset

On the Kubernetes setup page, click 'Open Instructions', or you can click directly here for the Kubernetes install instructions: [Agent Installation - Kubernetes]



This will take you to the agent installation page. Jump to the Kubernetes page (if not already there) and then go to the '[Installation Steps]' section. Run through the installation steps, with the addition of enabling Kube State Metrics and Prometheus in the agent configmap.

First create a directory to contain all the agent files, then obtain the 3 yaml files used for the install.

```
wizard@Sysdig:~$ mkdir sysdig-agent
wizard@Sysdig:~$ cd sysdig-agent
wizard@Sysdig:~$ wget
https://raw.githubusercontent.com/draios/sysdig-cloud-scripts/master/
agent_deploy/kubernetes/sysdig-agent-clusterrole.yaml
[... clipped response content ...]
wizard@Sysdig:~$ wget
https://raw.githubusercontent.com/draios/sysdig-cloud-scripts/master/
agent_deploy/kubernetes/sysdig-agent-daemonset-v2.yaml
[... clipped response content ...]
wizard@Sysdig:~$ wget
https://raw.githubusercontent.com/draios/sysdig-cloud-scripts/master/
agent_deploy/kubernetes/sysdig-agent-configmap.yaml
[... clipped response content ...]
```

Now create a namespace, and setup the cluster role for the agent.

```
wizard@Sysdig:~$ kubectl create ns sysdig-agent
namespace/sysdig-agent created
wizard@Sysdig:~$ kubectl create secret generic sysdig-agent
--from-literal=access-key=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx -n
sysdig-agent
secret/sysdig-agent created
wizard@Sysdig:~$ kubectl apply -f sysdig-agent-clusterrole.yaml -n
sysdig-agent
clusterrole.rbac.authorization.k8s.io/sysdig-agent created
wizard@Sysdig:~$ kubectl create serviceaccount sysdig-agent -n
sysdig-agent
serviceaccount/sysdig-agent created
wizard@Sysdig:~$ kubectl create clusterrolebinding sysdig-agent
--clusterrole=sysdig-agent --serviceaccount=sysdig-agent:sysdig-agent
clusterrolebinding.rbac.authorization.k8s.io/sysdig-agent created
```

Now edit the sysdig-agent-configmap.yaml file to include the following lines at the bottom (please take note this is YAML formatted and correct indentation is required):

```
    new_k8s: true
    k8s_cluster_name: Sysdig_Wizard_Training
    prometheus:
        enabled: true
```

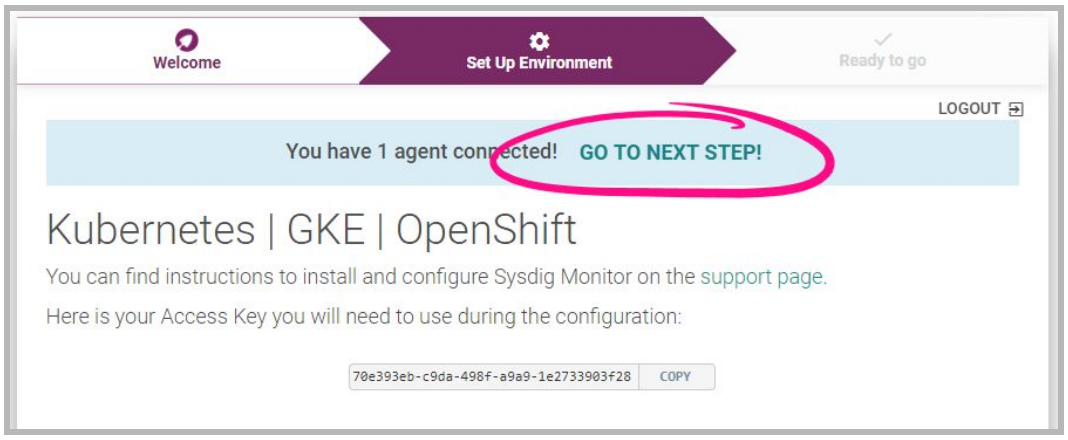
Finally load the modified configmap and deploy the daemonset.

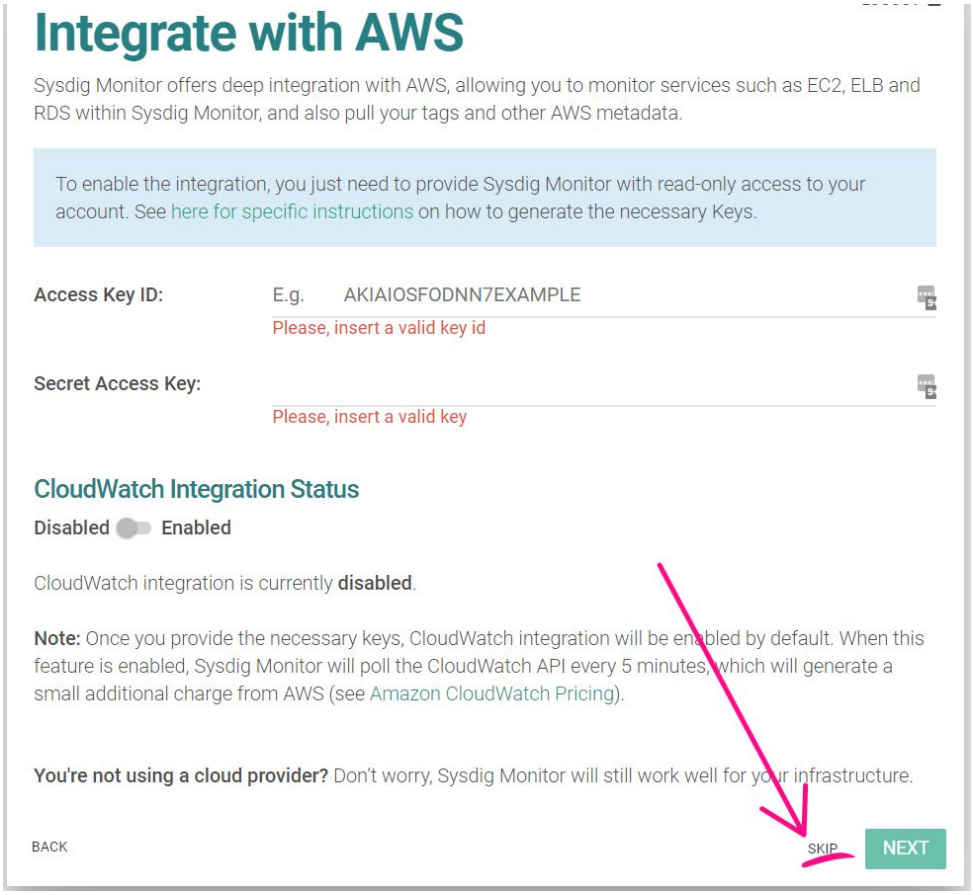
```
wizard@Sysdig:~$ kubectl apply -f sysdig-agent-configmap.yaml -n
sysdig-agent
configmap/sysdig-agent created
wizard@Sysdig:~$ kubectl apply -f sysdig-agent-daemonset-v2.yaml -n
sysdig-agent
daemonset.extensions/sysdig-agent created
```

You can use ‘kubectl get pods -n sysdig-agent’ to keep an eye on the status of the daemonset deployment, or go back to the Sysdig setup Wizard web page and wait for the top banner to change to show at least one agent has connected.

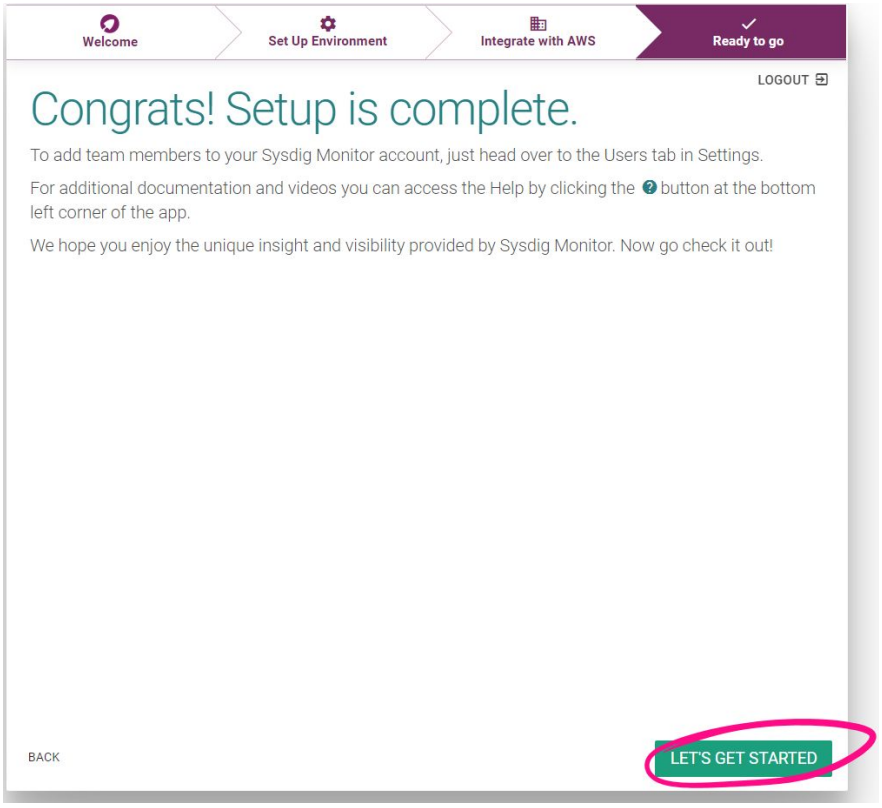
```
wizard@Sysdig:~$ kubectl get pods -n sysdig-agent
NAME                  READY   STATUS    RESTARTS   AGE
sysdig-agent-fn5jt    1/1     Running   0          9m17s
sysdig-agent-sf6c5    1/1     Running   0          9m17s
```

It may take a minute or two for the agent to download, run, and start reporting metrics, but once it has you will see the Wizard page change to show [d] agents connected. Now you can click ‘GO TO NEXT STEP’. We won’t be configuring any AWS integrations just now, so on the next page just click ‘SKIP’ in the bottom right, and then in the bottom right you can click ‘LET’S GET STARTED’.


Welcome
Set Up Environment
Ready to go
LOGOUT
You have 1 agent connected!
GO TO NEXT STEP!
Kubernetes | GKE | OpenShift
You can find instructions to install and configure Sysdig Monitor on the support page.
Here is your Access Key you will need to use during the configuration:
70e393eb-c9da-498f-a9a9-1e2733903f28
COPY


Integrate with AWS
Sysdig Monitor offers deep integration with AWS, allowing you to monitor services such as EC2, ELB and RDS within Sysdig Monitor, and also pull your tags and other AWS metadata.
To enable the integration, you just need to provide Sysdig Monitor with read-only access to your account. See here for specific instructions on how to generate the necessary Keys.
Access Key ID:
E.g. AKIAIOSFODNN7EXAMPLE
Please, insert a valid key id
Secret Access Key:
Please, insert a valid key
CloudWatch Integration Status
Disabled
Enabled
CloudWatch integration is currently disabled.
Note: Once you provide the necessary keys, CloudWatch integration will be enabled by default. When this feature is enabled, Sysdig Monitor will poll the CloudWatch API every 5 minutes, which will generate a small additional charge from AWS (see Amazon CloudWatch Pricing).
You're not using a cloud provider? Don't worry, Sysdig Monitor will still work well for your infrastructure.
BACK
SKIP
NEXT

Sysdig



3.2 Modifying the Configuration

Start by seeing what configuration maps where installed. Although we should know that already, it's good practice to double check in case you're picking up a cluster someone else has deployed.

```
$ kubectl get ns
$ kubectl get ds --all-namespaces | grep -i sysdig-agent
$ kubectl get configmaps -n sysdig-agent
```

This last command will show us the configuration map created as part of the install.

```
$ kubectl edit configmap -n sysdig-agent sysdig-agent
```

This will bring up a text editor (usually VI). The only area we want to focus on is the 'dragent.yaml' indented section, this is the Sysdig Agent configuration file. Take care when editing as this is a YAML file and indentation is key (spaces not tabs).

Below is an example (skipping the default commented section). If your config does not have these uncommented items, add them now:

```
apiVersion: v1
data:
  dragent.yaml: |
    k8s_cluster_name: Sysdig_Training
    new_k8s: true
    prometheus:
        enabled: true
```

Once you've made modifications to the configuration file, the agents will need to be restarted to pick this up. Confirm that no other pods run inside the namespace with

```
$ kubectl get pods -n sysdig-agent
```

and if it is just the sysdig-agent pods, you can run

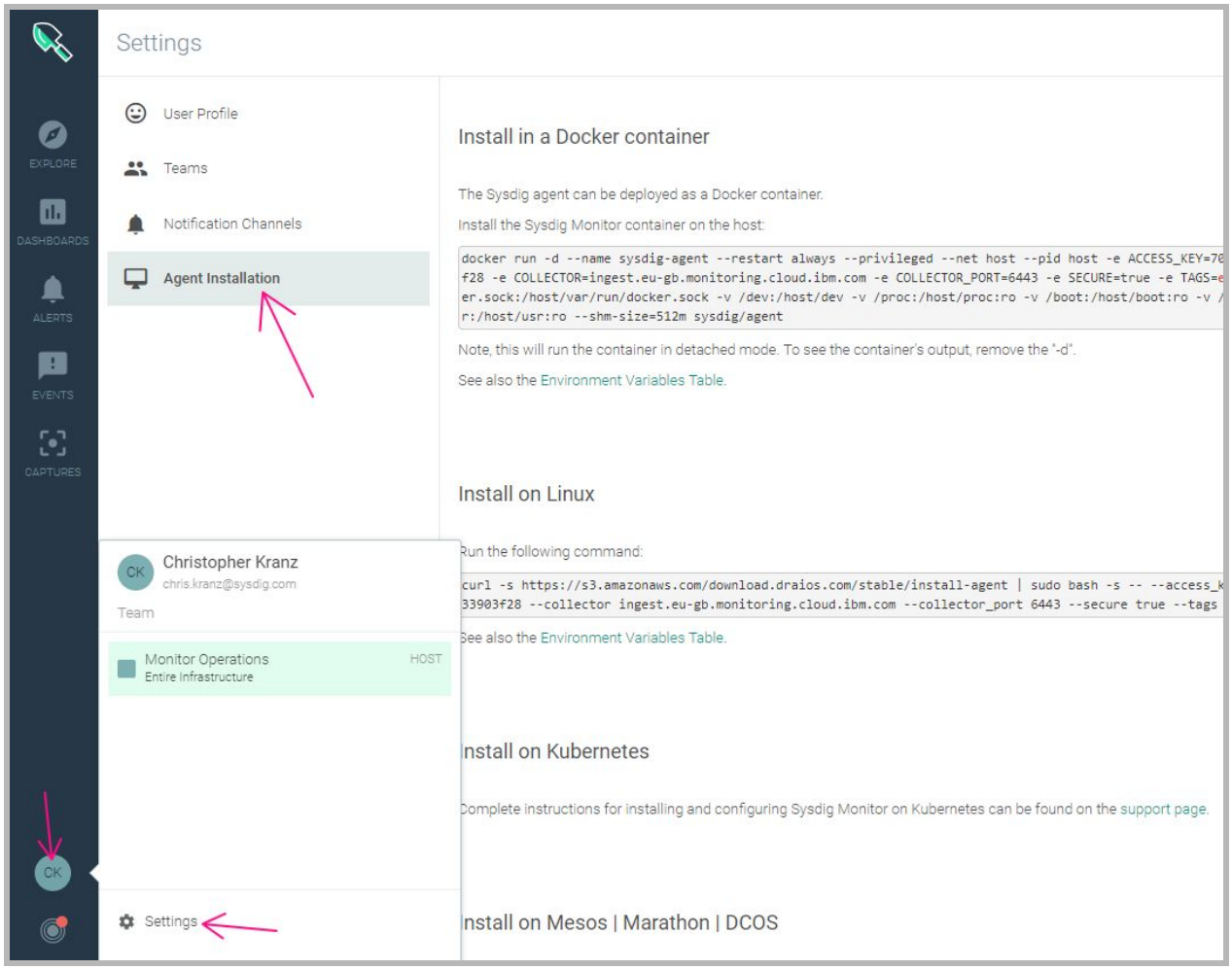
```
$ kubectl delete pods -n sysdig-agent --all
```

If other pods exist in this namespace, delete the sysdig-agent pods manually.

```
$ kubectl delete pods -n sysdig-agent sysdig-agent-fn5jt
sysdig-agent-22z6x
```

3.3 Alternative Installation Options

While we won't cover this during the lab exercises, the Sysdig Agent can be installed via a variety of methods, depending on the environment. You can find the other installation instructions by going to Settings -> Agent Installation.


Settings
User Profile
Teams
Notification Channels
Agent Installation
EXPLORE
DASHBOARDS
ALERTS
EVENTS
CAPTURES
Install in a Docker container
The Sysdig agent can be deployed as a Docker container.
Install the Sysdig Monitor container on the host:
docker run -d --name sysdig-agent --restart always --privileged --net host --pid host -e ACCESS_KEY=70
f28 -e COLLECTOR=ingest.eu-gb.monitoring.cloud.ibm.com -e COLLECTOR_PORT=6443 -e SECURE=true -e TAGS=e
er.sock:/host/var/run/docker.sock -v /dev:/host/dev -v /proc:/host/proc:ro -v /boot:/host/boot:ro -v /
r:/host/usr:ro --shm-size=512m sysdig/agent
Note, this will run the container in detached mode. To see the container's output, remove the "-d".
See also the Environment Variables Table.
Install on Linux
Run the following command:
curl -s https://s3.amazonaws.com/download.draios.com/stable/install-agent | sudo bash -s -- --access_k
33903f28 --collector ingest.eu-gb.monitoring.cloud.ibm.com --collector_port 6443 --secure true --tags
See also the Environment Variables Table.
Install on Kubernetes
Complete instructions for installing and configuring Sysdig Monitor on Kubernetes can be found on the support page.
Install on Mesos | Marathon | DCOS
CK
Christopher Kranz
chris.kranz@sysdig.com
Team
Monitor Operations
Entire Infrastructure
HOST
CK
Settings

4. Lab 1: Application Assurance

4.1 Deploy the Demo Application

From the location of where you have downloaded the demo application to, run the following:

```
$ kubectl create ns voting-app
$ kubectl apply -f lab1/manifests/ -n voting-app
```

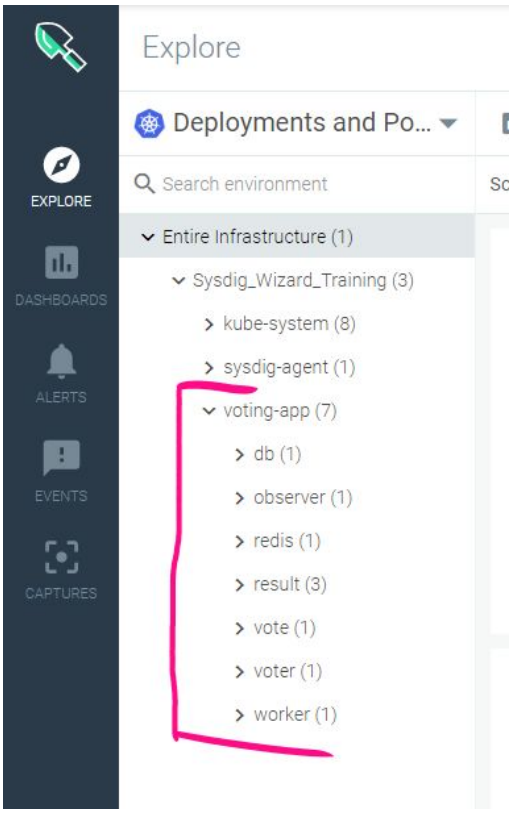
Once deployed, you can check the status to make sure everything is up and running.

```
$ kubectl get pods -n voting-app
```

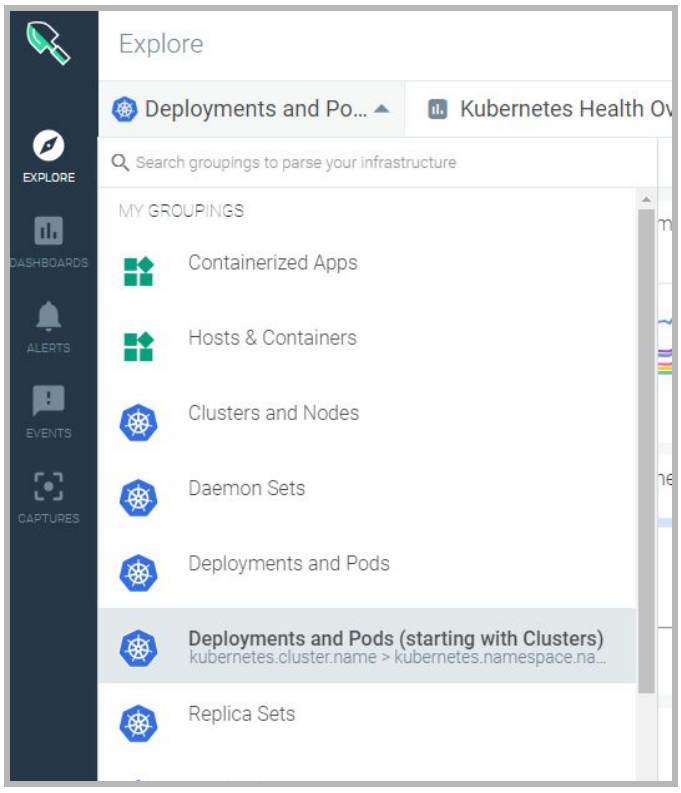
```
ckranz@Office:~/academy-labs/lab1$ kubectl get pods -n voting-app
NAME                      READY   STATUS    RESTARTS   AGE
db-6c54bb5bf6-lpqgm       1/1     Running   0          5m34s
observer-5d84488869-tqxpq 1/1     Running   0          5m34s
redis-85b778d67d-bxq4w    1/1     Running   0          5m34s
result-6576d95cc4-ddvp5   1/1     Running   0          4m42s
result-6576d95cc4-hndcl   1/1     Running   0          4m42s
result-6576d95cc4-m9572   1/1     Running   0          4m42s
vote-58c584946d-gr5kf     1/1     Running   0          5m34s
voter-5b5fcf6769-lqsnh    1/1     Running   0          5m34s
worker-69549dc66d-ndsfk   1/1     Running   0          5m34s
```

4.2 Understanding the application

If you switch to the Sysdig Monitor UI, you should see the voting-app deployment showing.



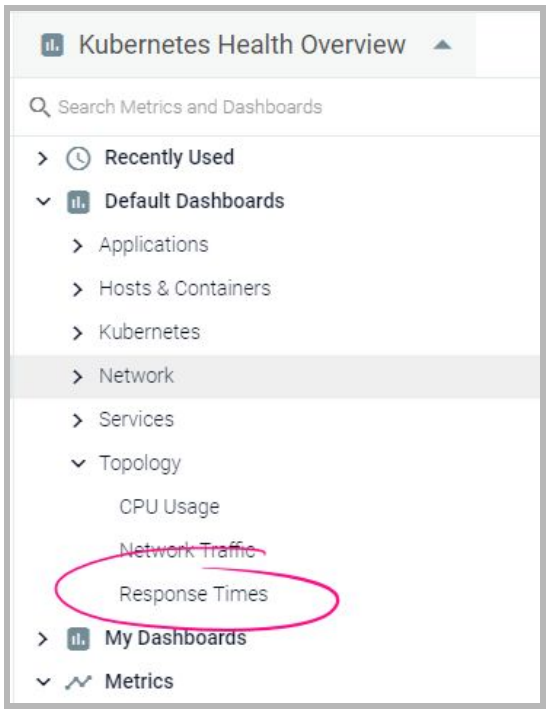
As this application comes from a 3rd party supplier, we don't yet know much about it, but lets see what Sysdig can tell us about this particular application. If your default view isn't 'Deployments and Pods (starting with Clusters)', select it from the drop-down list at the upper left of the main screen. This will group all our objects by Kubernetes Cluster, then Namespace, then Deployment, then Pod, then container.

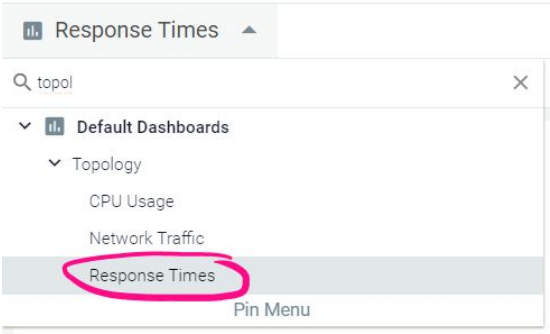


If you expand the grouping hierarchy on the left hand side, we can find out 'voting-app' in the list and then by clicking on it we'll focus all our dashboards to just show data that is relevant to this one object. From the main dashboard page, we can see what default dashboards are available out-of-the-box from Sysdig. Many more default dashboards automatically become available when the Sysdig Agent detects new applications that are of relevance, but for now this shows all the defaults that are relevant to the applications we have running.

Topology Map

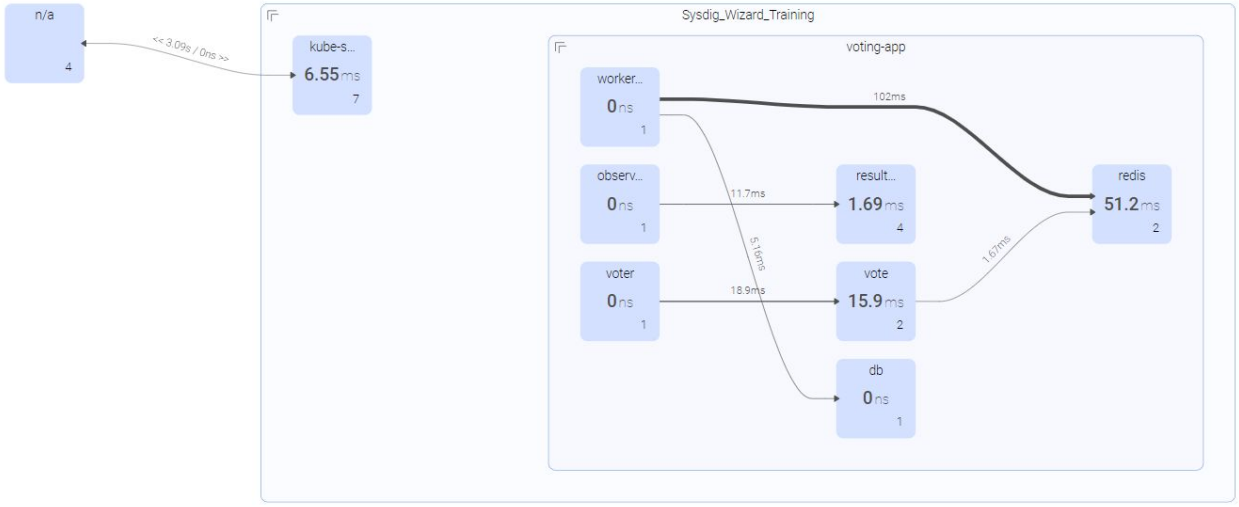
In order to understand more about our application, let's have a look at a topology view. Under 'Default Dashboards', find 'Topology', then click on 'Response Times'. Alternatively, just click 'Default Dashboards' and start typing 'topology' and the available dashboards will filter.





This brings up a topology map where the groupings are dynamically generated based on the groupings you have chosen on the left hand side. All the lines connecting the various objects are also dynamically drawn based on real network traffic that the Sysdig Agent is observing between the different objects. This network traffic may purely be service to service within a single host rather than actual network traffic. If you expand the boxes you should end up with a view similar to the below, showing all the different components of our voting-app

deployment.



Questions using this dashboard:

1. What processes are any of these deployments running?
    a. _____
2. How many pods are running in each deployment?
    a. _____
3. Make some notes about how you think this application operates.
    a. _____
4. If you have more than one host, from this dashboard alone can you see how many nodes you have (hint: you may want to switch the grouping around)
    a. _____
5. If you have more than one host in your cluster, can you see which pods are running on each host?
    a. _____
6. Can you see any external traffic?
    a. _____
7. You may see a grouping marked as ‘n/a’, what do you think this is?
    a. _____

Identifying processes

While we can see the individual processes on the Topology maps, it can be nice to see this as a list instead rather than graphically. From the dashboard dropdown selection, type in 'process' and see what dashboards are available.

- 'Top Processes' is simply a table showing a list of top processes along with relevant metrics to each process.
- 'Overview by Process' gives a richer dashboard, showing overall resource usage, breakdown and remaining resources.

Questions using one of these dashboards?

1. What is the database technology used? Are multiple technologies?
    a. _____
2. What is the web server technology used? Are multiple technologies?
    a. _____
3. What are the largest resource consumers?
    a. _____
4. Are there currently any resource constraints?
    a. _____

Golden Signals

Golden Signals is a concept developed by Google with it's Site Reliability Engineering role. The idea is that there are 4 core metric groups that help to understand the health of an application, that is the following:

- Latency - In Sysdig we generally record this as response time
- Traffic - In Sysdig we generally record this as request rate
- Errors - In Sysdig we also show this as errors
- Saturation - In Sysdig this is generally a view across different resource usage metrics such as CPU, memory, network and file.

By understanding these metrics, we get a good understanding of the health of the application. Try to find a dashboard that might show some (or all) of these metrics.

For more reading on Google's SRE take on monitoring distributed systems, have a look at the following chapter in the Google Site Reliability Engineering book: https://landing.google.com/sre/sre-book/chapters/monitoring-distributed-systems/

USE Monitoring

USE Monitoring may look very similar to Golden Signals, here we are monitoring resources of Utilisation, Saturation and Errors, but the subtle difference here is that the USE method can be more beneficial for understanding the underlying host health, whereas Golden Signals is more focused on application health.

In a DevOps where siloed responsibility is much less common, it's increasingly important to understand both of these observability methods as depending on the specifics of an issue, you may need to approach problems from different angles (or even from multiple angles).

You can read more about USE method monitoring from Brendon Gregg: http://www.brendangregg.com/usemethod.html

Investigation Summary

Now we should have a pretty good understanding of this application, from knowing nothing other than what it's called, hopefully you can now describe the traffic flow, the technologies used, and start thinking about how this could potentially be easily scaled. Let's test that new knowledge:

1. What is the main database technology / technologies used by this application?
    a. _____
2. What does it appear the voter component is doing?
    a. _____
3. Are results available in realtime?
    a. _____
4. What do you think the worker is doing?
    a. _____
5. What technology does the 'vote' web service use?
    a. _____
6. What technology does the 'result' web service use?
    a. _____
7. Are there currently any bottlenecks in the application?
    a. _____
8. What is the normal response rate of the service?

a. _____

9. What is a normal request rate?
    a. _____
10. What is a normal error rate?
    a. _____
11. Do the host(s) have sufficient resources to support the application requirements?
    a. _____

4.3 Breaking the Application

From within the folder you downloaded the course scripts to, execute the 'problem.sh' file. **No spoilers, don't view this file first**!

```
$ ./lab1/problem.sh
```

This will have created a situation with our application that we now need to understand. The problem led to the following support call:

'Somethings wrong with our application, Twitter sentiment has taken a negative dive and we need to understand what's wrong. We don't yet know what the issue looks like, just that our users are not happy on social media at the moment, and some of our key paid influencers are under fire. We've got execs shouting, we need you to sort this as quickly as possible!'

4.4 Troubleshooting an Unknown Issue

Application Health

All we know is that we have unhappy users of our application, and as this is an application problem, so which health & observability strategy do you think we should try first?

A. Golden Signals
B. USE method

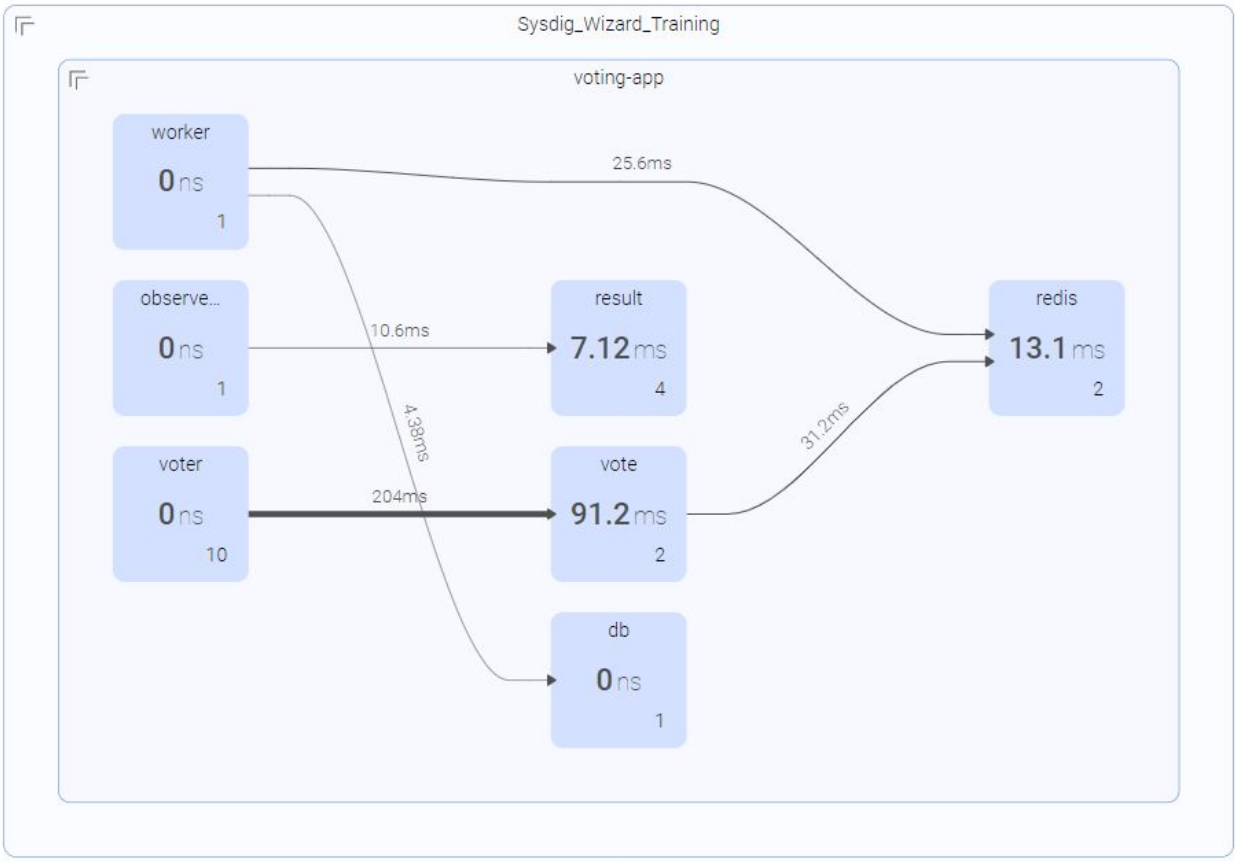
From whichever dashboard you chose, can you answer the following questions?

1. What does user response look like?
    a. _____
2. Is there an increase in errors?
    a. _____

3. What does resource usage look like?
    a. _____
4. Is there anything that suggests a change to the application profile?
    a. _____
5. Do you have enough information to diagnose what changed?
    a. _____
6. Do you have enough information to suggest a fix?
    a. _____

Application Topology

Hopefully you have a good understanding of what's going on, but lets have a look at the topology maps to see if this helps us paint a better picture.

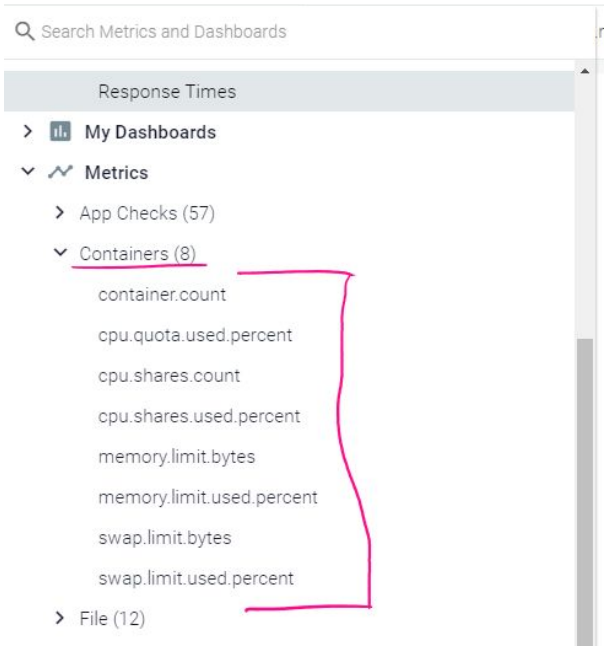


Hopefully it's quite obvious, we can see some high latency between the voter and the vote services. With real traffic we wouldn't have a voter service and see this increase, but we can

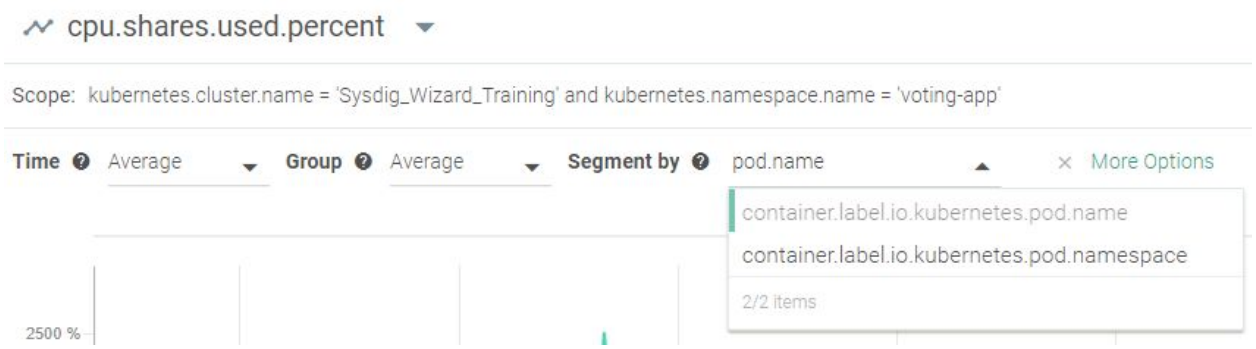
see the latency and that our problem creator script scaled up the number of users hitting our vote application. This had a negative impact on the response time from the application.

Finding Specific Metrics

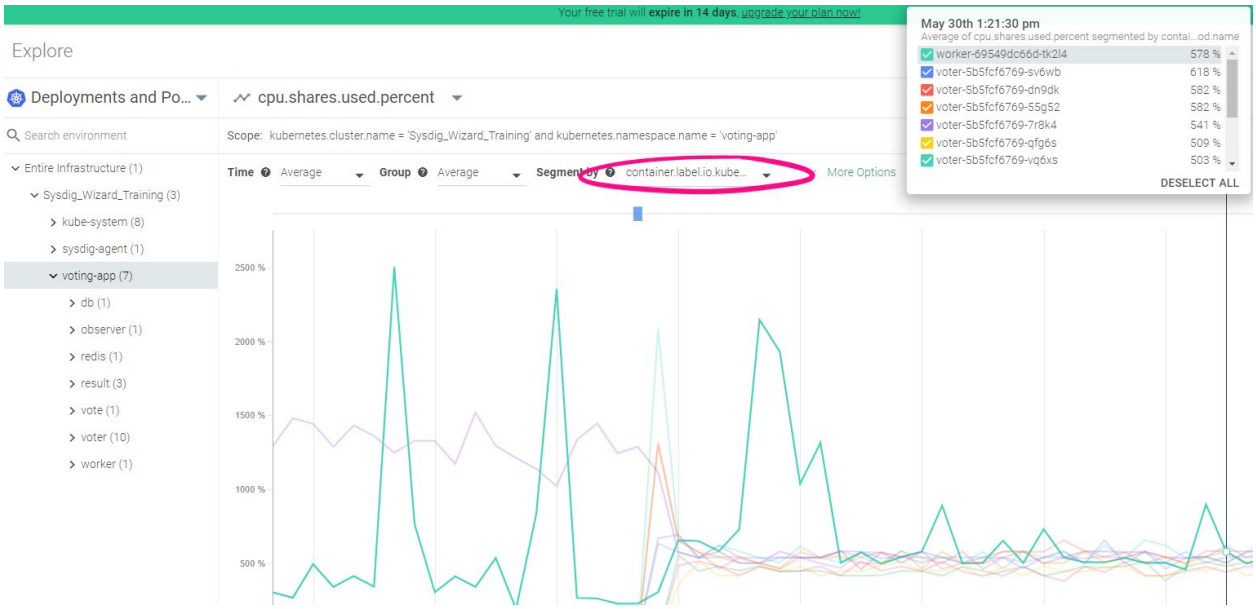
In addition to lots of default dashboards and displays, we're collecting all sorts of additional metrics which can be used in custom dashboards. Lets see if we can find out more about this application issue by interrogating what other metrics are available. From the dashboard dropdown, scroll down to metrics, then expand the 'Containers' group. This is all the containerd specific metrics we're collecting.



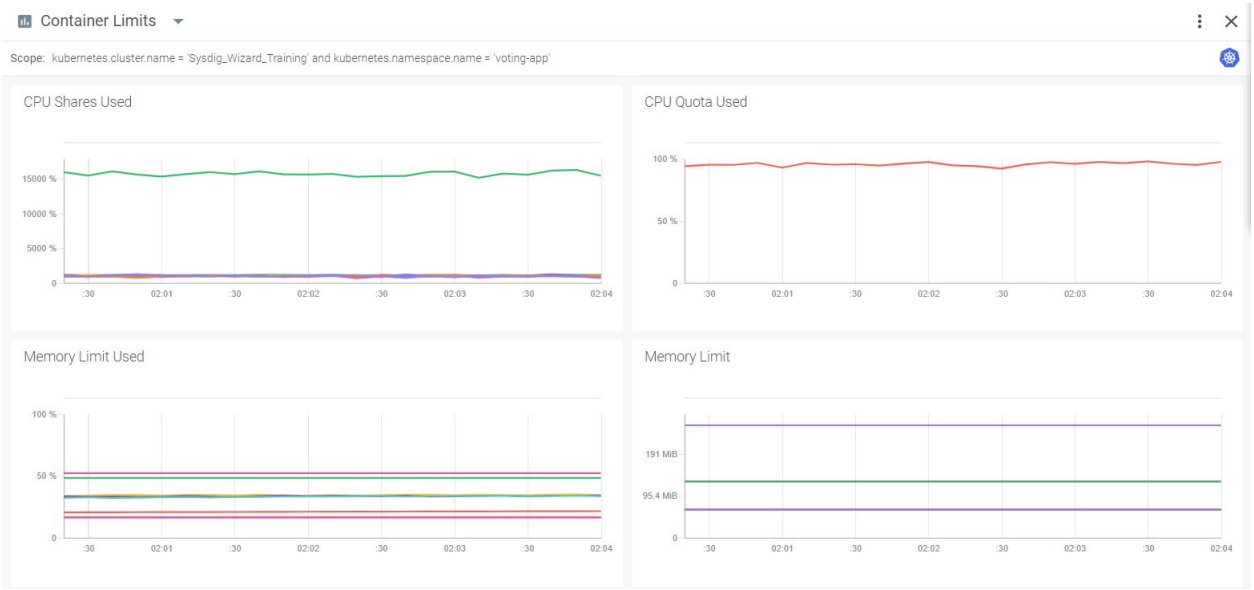
Now if we select 'cpu.shares.used.percent' we'll see a specific chart for that. But seeing this as an average across the entire application isn't very useful, so we want to add a segmentation. In the top, click inside the 'Segment by' input box and type in 'pod.name' and then select the annotation labelled 'kubernetes.pod.name' (or 'container.label.io.kubernetes.pod.name').



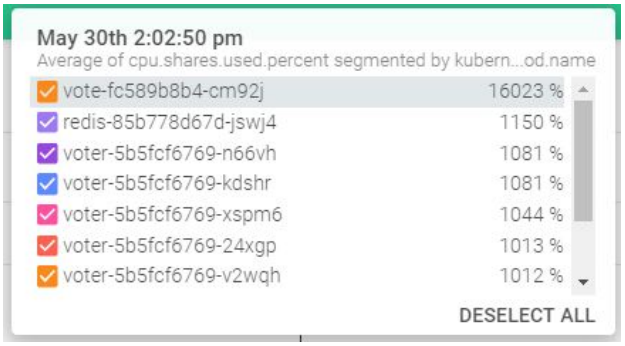
This will split the graph up so that each line represents a different pod up to a certain limit (10 is the default, but this can be configured also).



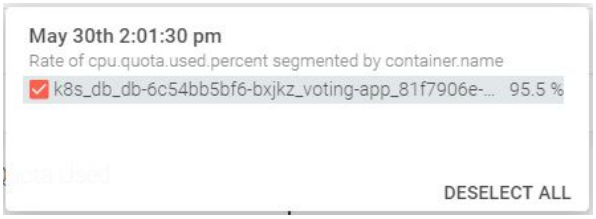
Also have a look at the metric ‘cpu.quota.used.percent’ and see if there is anything of concern there. We also already have a default dashboard that shows these metrics side-by-side, so search for a dashboard with the word ‘limit’.



Now we can see the individual container share usage, and we can see that we have containers using way more CPU shares than they have assigned.



We can also easily see that we have a container using almost its maximum CPU limit.



Maybe there is a reason behind the container limits, but we have a couple of ways to fix our application now:

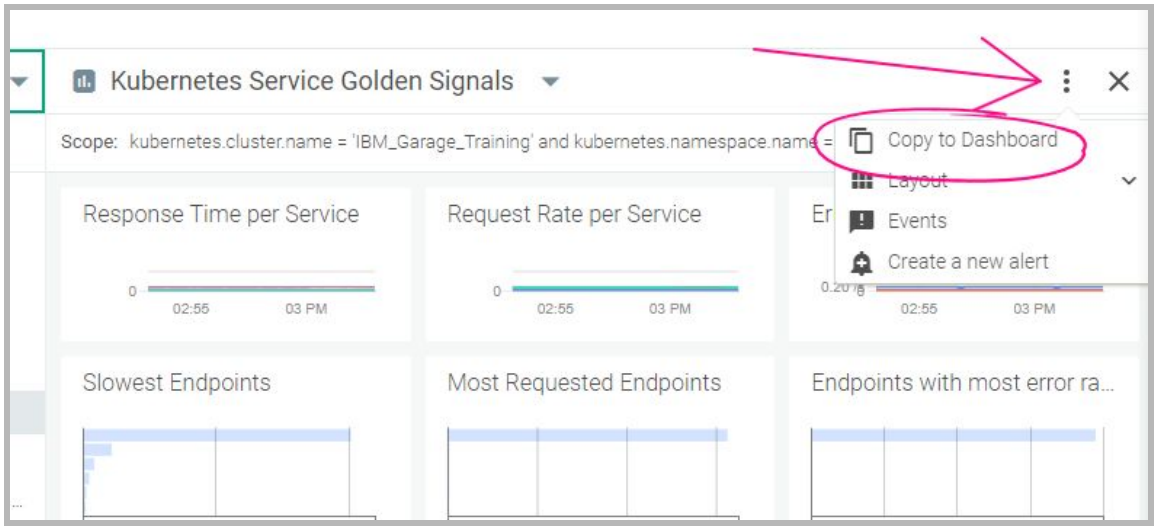
1. Increase the CPU limits applied to the deployments
2. Scale up the vote deployment to deal with the load

Optional questions:

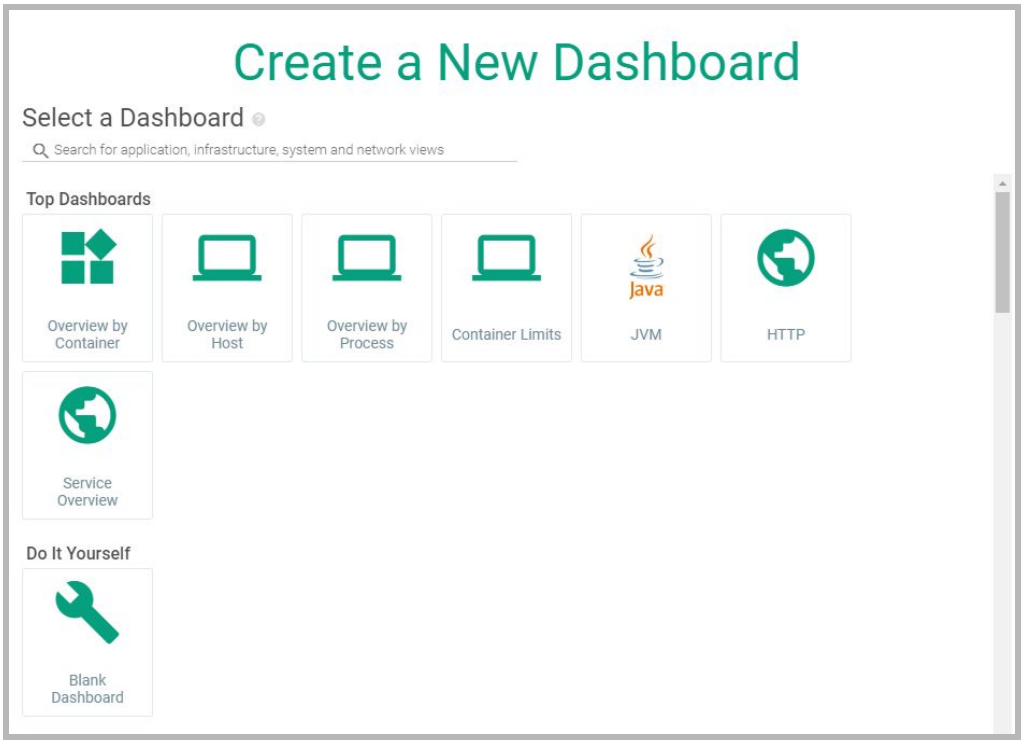
1. The db deployment also has a CPU limit applied to it, is this affecting user experience?
    a. _____
2. Would it be simple enough to scale the db service out as we could the vote deployment?
    a. _____

4.5 Custom Dashboards

We've used several different dashboards to learn about our application, we want to try to simplify this for next time or for someone else that might not have learned about this application. You can clone any default dashboard or metric using the menu in the top right of any of the default dashboards and selecting 'Copy to Dashboard'.



You can also build a new dashboard from scratch or from a template, simply click on ‘Dashboards’ on the left hand side, then in the upper right of the pop-out menu choose the green + icon - ‘Add Dashboard’ and you’ll be presented with a creation wizard.



From here you can customise the panels, resize them, change the metric grouping or aggregation. You can either create a pre-scoped dashboard which can be useful for sharing, or create a dashboard designed for exploring where the scope is dynamically set depending on the user. Documentation for creating and editing dashboards can be found here: https://sysdigdocs.atlassian.net/wiki/spaces/Monitor/pages/205488166/Dashboards

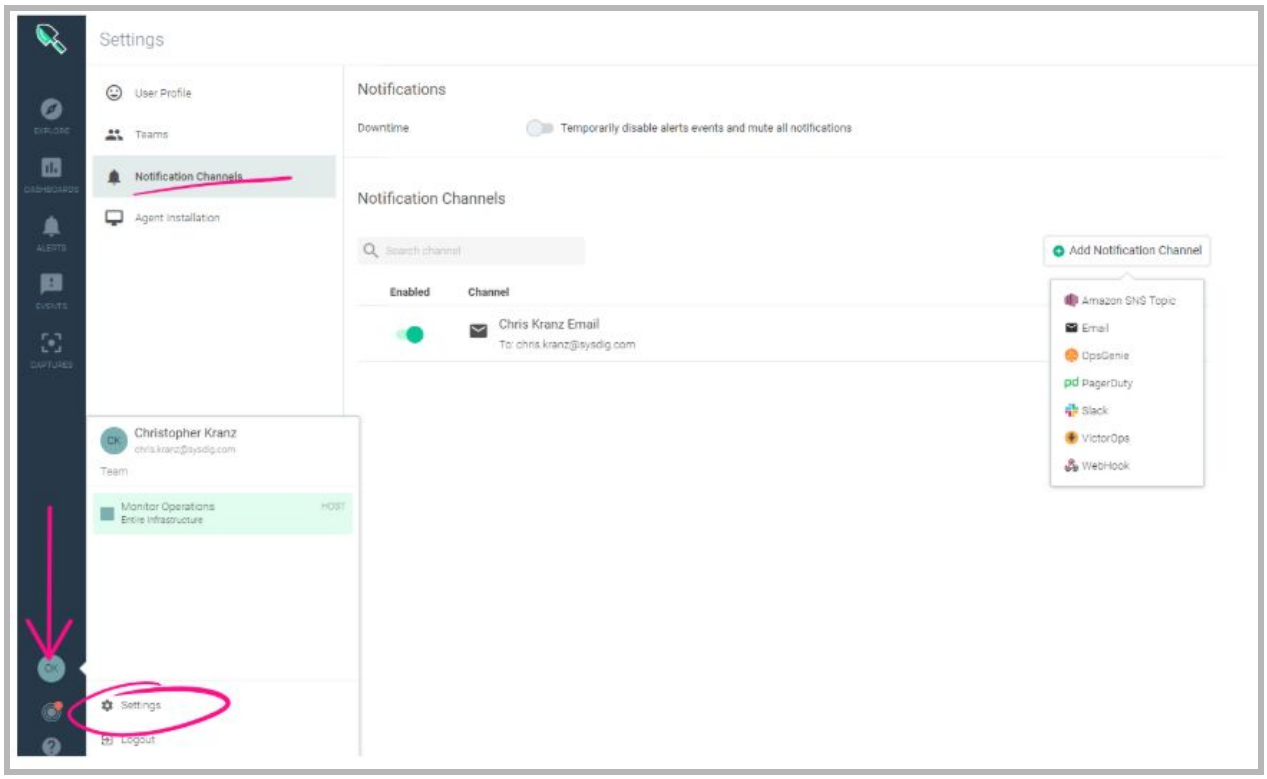
Task

Create a custom dashboard for the ‘voting-app’ application that can help with future troubleshooting of issues. Try to focus on relevant information and reducing unnecessary information overload.

4.6 Alerts

Notification Channels

First we'll need to setup a simple Notification Channel. There are a few options here, but for simplicity let's just create a simple email notification. Goto Settings, then Notification Channels.

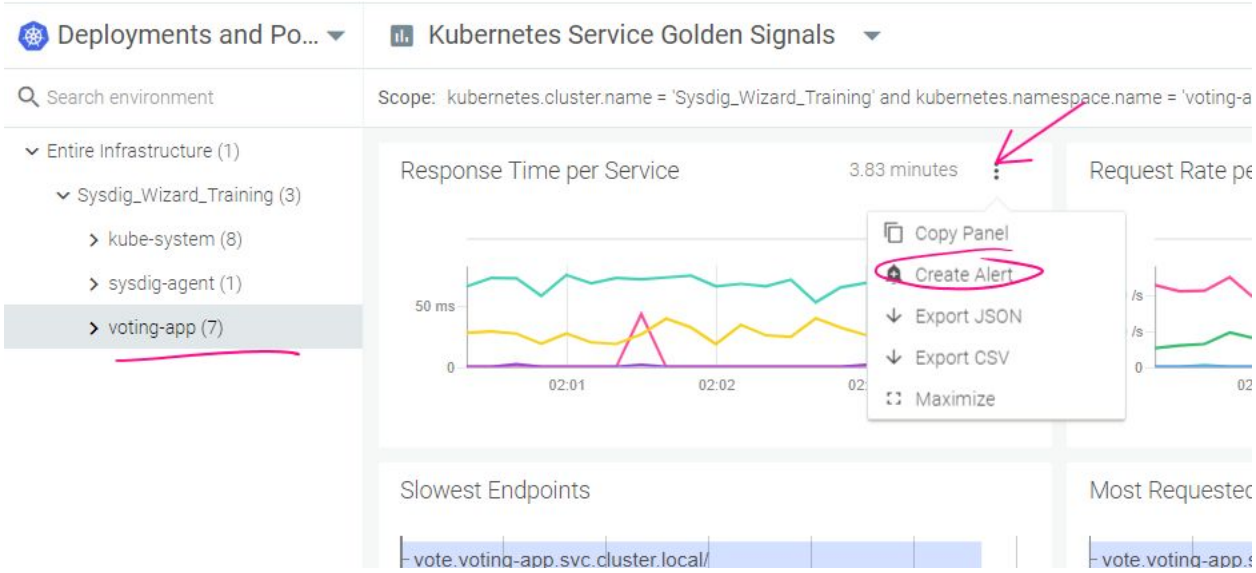


Now click 'Add Notification Channel' and you'll see the different channels available. For now just choose 'Email' and run through the options to configure your own email address for the notification.

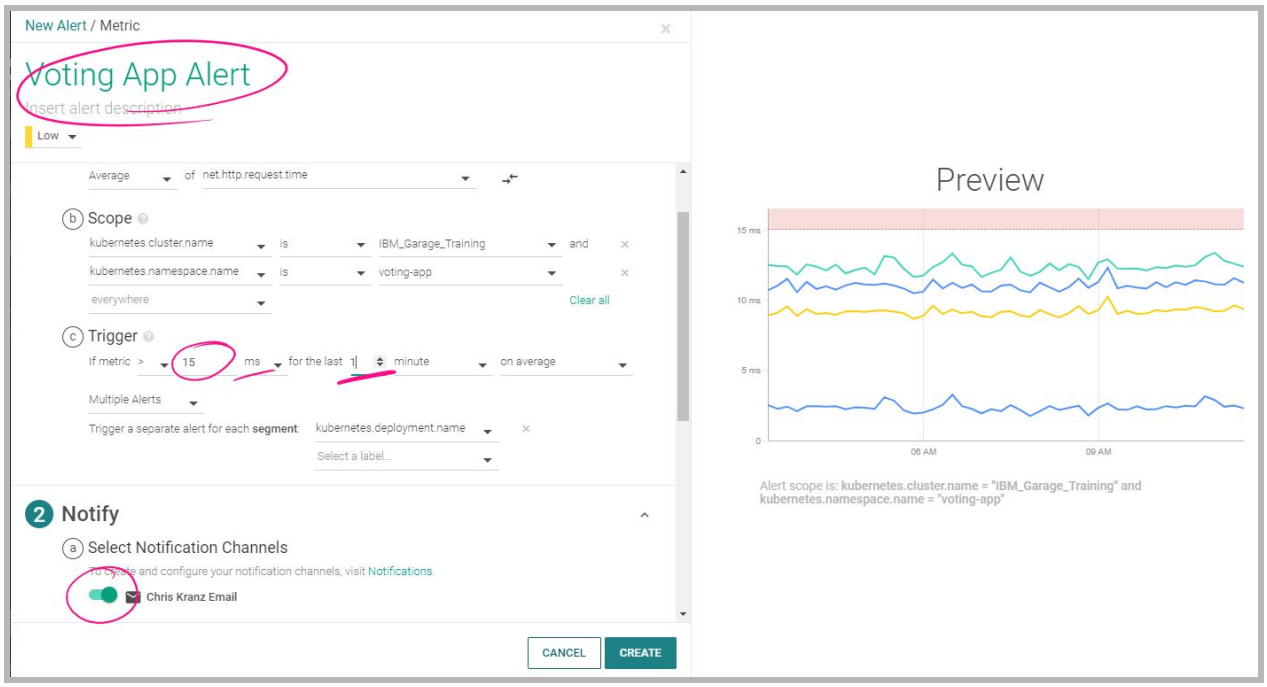
Metric Alerts

Having focused dashboards is really useful to get a view of application health, but a dashboard isn't very noisy when something goes wrong. Lets create an alert or two to help be more proactive the next time this happens.

From the Explore view, make sure you have the ‘voting-app’ chosen and then choose your dashboard you created, select a metric of your choice, click the 3 vertical dots on the metric panel, then choose ‘Create Alert’.



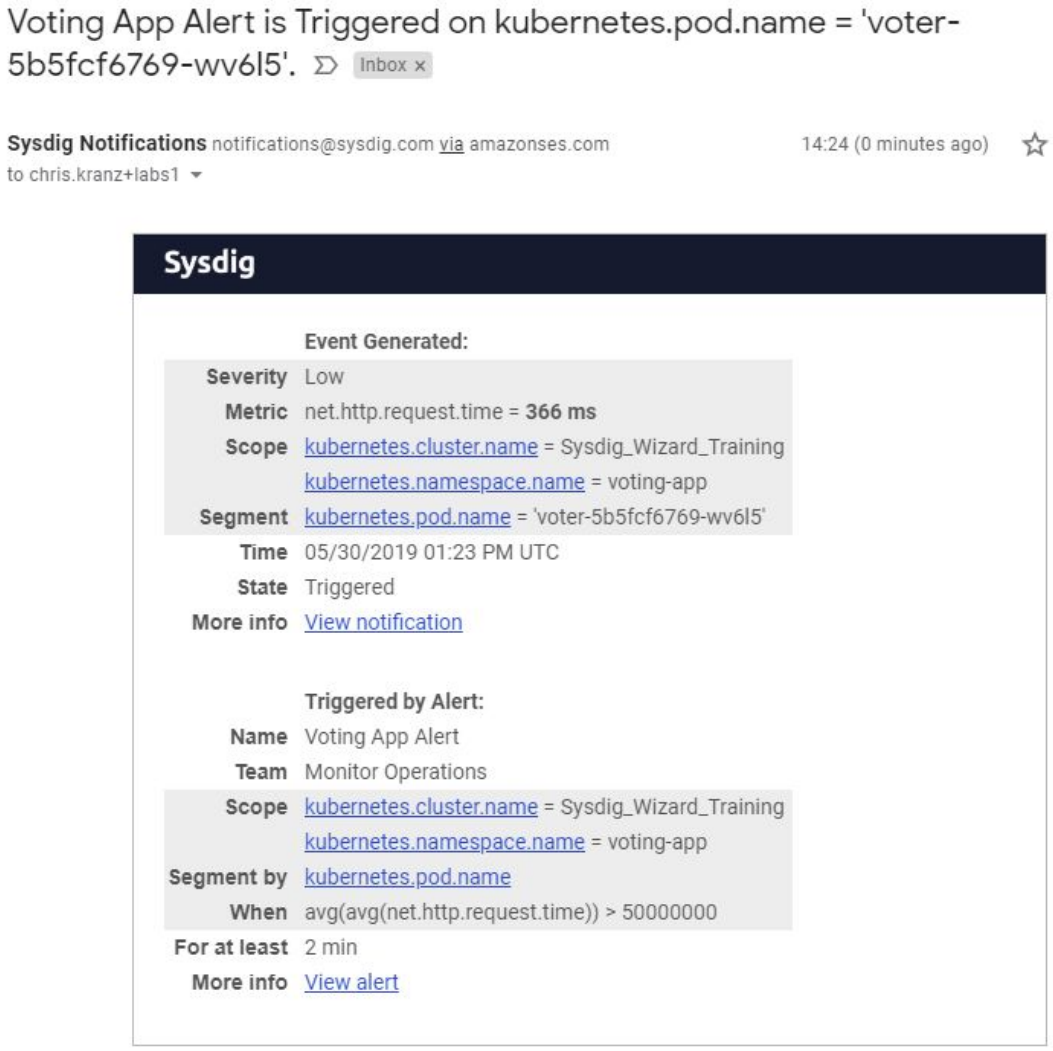
Now on the New Metric Alert pop-up, configure a relevant threshold and enable the notification channel that we just setup. You should get visual feedback to the right hand side of where this alert will be triggered. There are a lot of different options here, you can create quite complex alerts using multiple metrics and segmented by multiple objects (i.e. so each individual pod will trigger against a threshold rather than an entire deployment). Notice that the metric alert is already scoped to the ‘voting-app’ and the segmentation matches what was on your dashboard. If you had no segmentation, try segment the alert by deployment, pod or container. It might also be useful to give the new alert a name rather than the default ‘New Metric Alert’. For testing purposes, it might also be useful to change the trigger duration to only a minute or two, unless there is a coffee break coming up! In the real world we will tweak this to get a balance between brief spikes, false positives and early warning.



If you click on the Alerts menu item on the left hand side, you should be able to find your newly created alert and confirm that it is enabled.

Alert Trigger

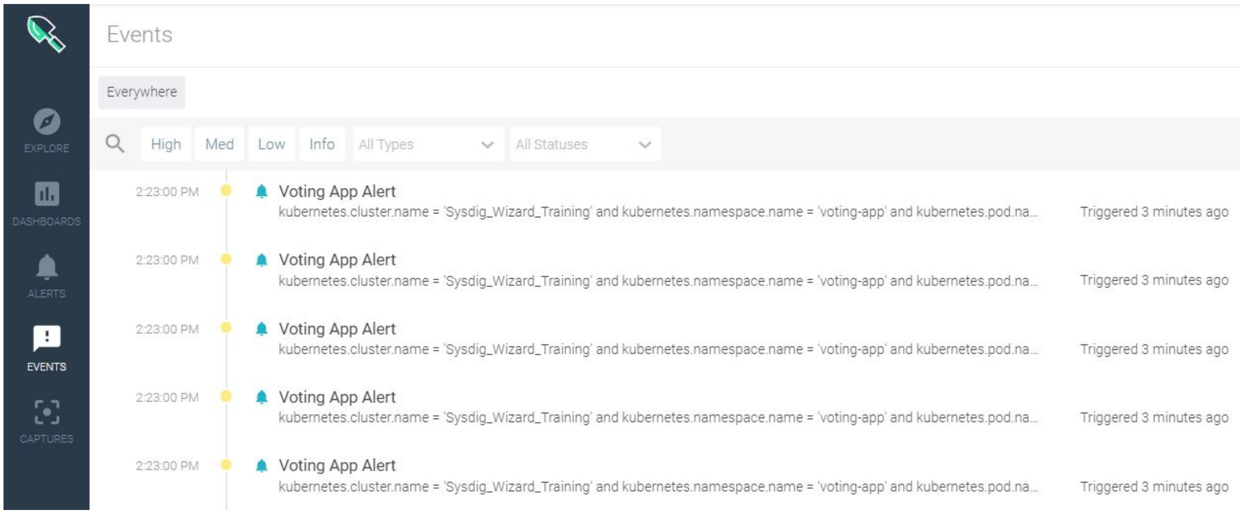
As our problem still exists and has pushed our application over an acceptable threshold, your alert should trigger. Hopefully you should receive at least 1 email, but depending on the what, where and how you applied the trigger, you may get several.



You can either click on the 'More info: View alert' link, or from within the Sysdig UI click on 'Events'.

Events

From the Events page we can see all the alerts that can be triggered, along with other events that Sysdig is ingesting. We'll natively pull in events from the container runtime (Docker, CRI-O, or ContainerD) and from the orchestrator (Kubernetes, OpenShift, GKE/EKS/AKS/IKS, etc.), but other events can be posted to the Sysdig platform also. Here is a snapshot of my events being triggered by my alert, obviously not very tuned and I get multiple alerts for the single event.



Now run './lab1/fix.sh'.

```
$ ./lab1/fix.sh
```

In a minute or two have a look to see if the events stream changes at all, or if you receive any new emails. You can also click on any of the events on the events stream for more information surrounding it.

5 Lab 2: Digging Deeper

5.1 Deploy the Demo Application

We're going to deploy a second application that has a little more variation to it. From the location of where you have downloaded the demo application to, run the following:

```
$ kubectl create ns java-app
$ kubectl apply -f lab2/manifests/ -n java-app
```

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

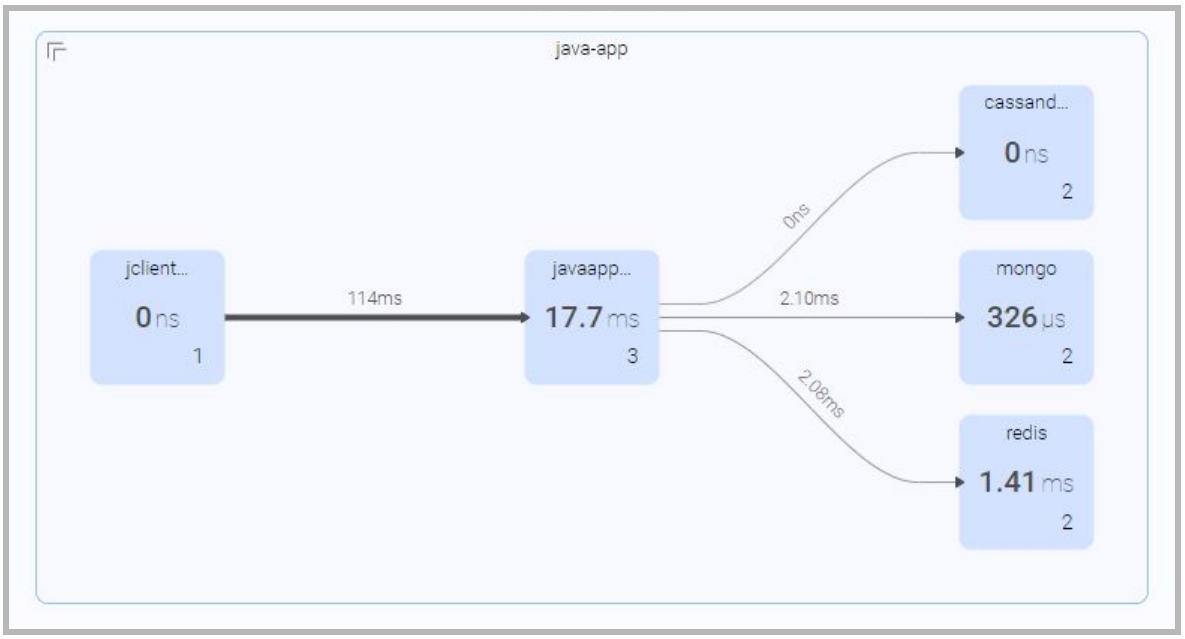
```
$ kubectl get pods -n java-app
```

```
wizard@Sysdig:$ kubectl get pods -n java-app
NAME                          READY   STATUS    RESTARTS   AGE
cassandra-64578cf66c-jzjqb    1/1     Running   0          3m45s
javaapp-5fd4987676-bfs9n      1/1     Running   0          3m45s
javaapp-5fd4987676-ts8jr      1/1     Running   1          3m45s
jclient-f5856486f-qlgn6       1/1     Running   0          3m45s
mongo-7f9fdbbfc-nh8r4         2/2     Running   0          3m45s
redis-f59bfbf48-kk9wz         1/1     Running   0          3m44s
```

5.2 Discovering the application

This application gives nothing away about what it actually does, but it is a lot more helpful in the way it's labelled than our previous 'db' application. It's fairly obvious that it's a Java based application and it appears to be writing to 3 different databases:

- Cassandra
- Mongo
- Redis



This can be confirmed by looking again at one of the processes dashboards, and this also helps show us that the Java web application is running on a Tomcat server.

Investigate if there are any application specific dashboards for the different technologies used by this application. See if you can learn more about what it does and how it's built.

5.3 Data Sources

Default Data Sources

During Lab 1 we walked through a fairly generic scenario using the default telemetry data that the Sysdig agent collects without any additional configuration. If you dig around the metrics lists you'll notice that by default Sysdig collects the following metric groups:

- Container metrics
- Kubernetes state metrics
- Host metrics
- Statsd
- JVM
- JMX

Additional Data Sources

Sysdig can additionally be configured to collect data from a variety of additional data sources, but these generally need some level of configuration. These vary between simple permissions defined in the agent configuration, configuring application specific exporters, to writing customer integrations. The following additional integrations can also be configured:

- Prometheus (the agent by default is looking for Prometheus metrics, but Prometheus exporters may need to be configured)
- App Checks (the agent needs to be provided application specific metrics)
- JMX & statsd can be additionally instrumented by pushing out custom metrics directly from your application if that is required.

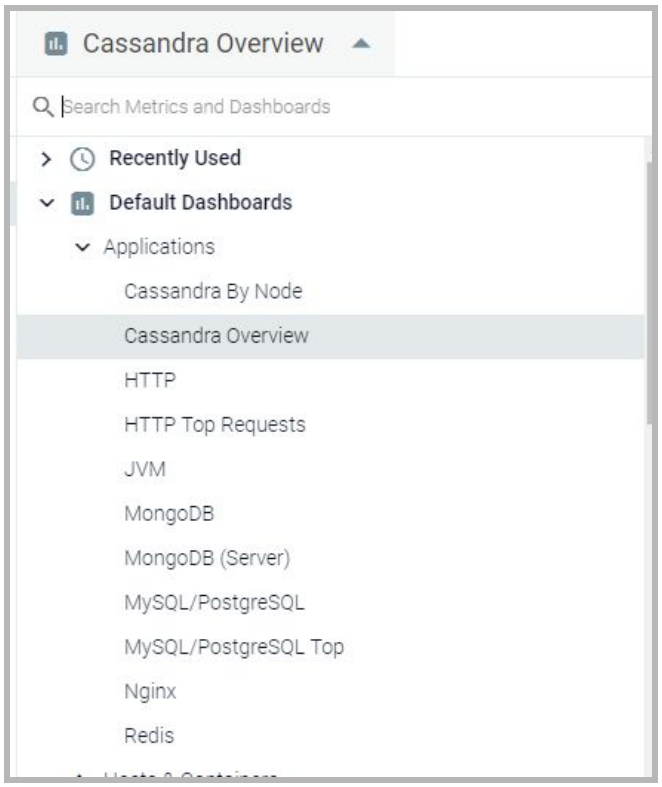
Prometheus Exporters

If we go back and look at the manifest for the lab1 results deployment, we can see that we already have this application instrumented for Prometheus. The additional annotations in the manifest are simple enough, we're exposing the metrics on port 80, using the default metric path as it isn't defined (i.e. '/metrics'), and we're configured the metrics to allow scraping. If you setup a new Kubernetes cluster for this training session, it's likely that you won't have a Prometheus installation, but all the Sysdig Agent requires is the Prometheus exporter and the Agent will auto-detect and start scraping these metrics. You'll notice under the Metrics section of the dashboard dropdown in Explore that we are already collecting a number of Prometheus metrics. These can be used alongside any other metrics. Add any of the Prometheus metrics to the dashboard you built at the end of the last exercise.

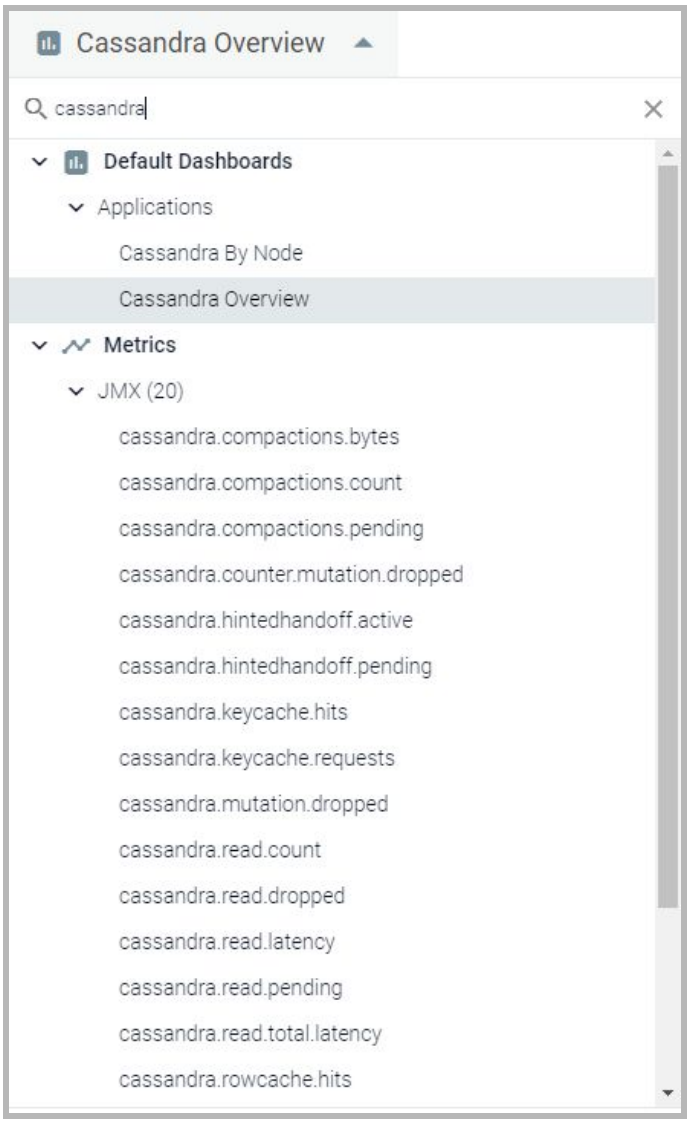
JMX and JVM

Now that we have a few more Java based applications installed, let's quickly explore what we can gather from these. The Sysdig agent will auto-detect Java applications based on the Java runtime, and where possible also auto-detect the specific application (i.e. Cassandra, Kafka, etc.). When the Sysdig Agent auto-detects the JVM process, it also tests if JMX can be used, and where it can the Agent will automatically enable and disable JMX with each metric collection cycle. This minimises the normal overhead of running JMX without any loss of data cardinality.

If you click on the dashboard dropdown in Explore, you can see we've detected Cassandra and MongoDB as new applications.

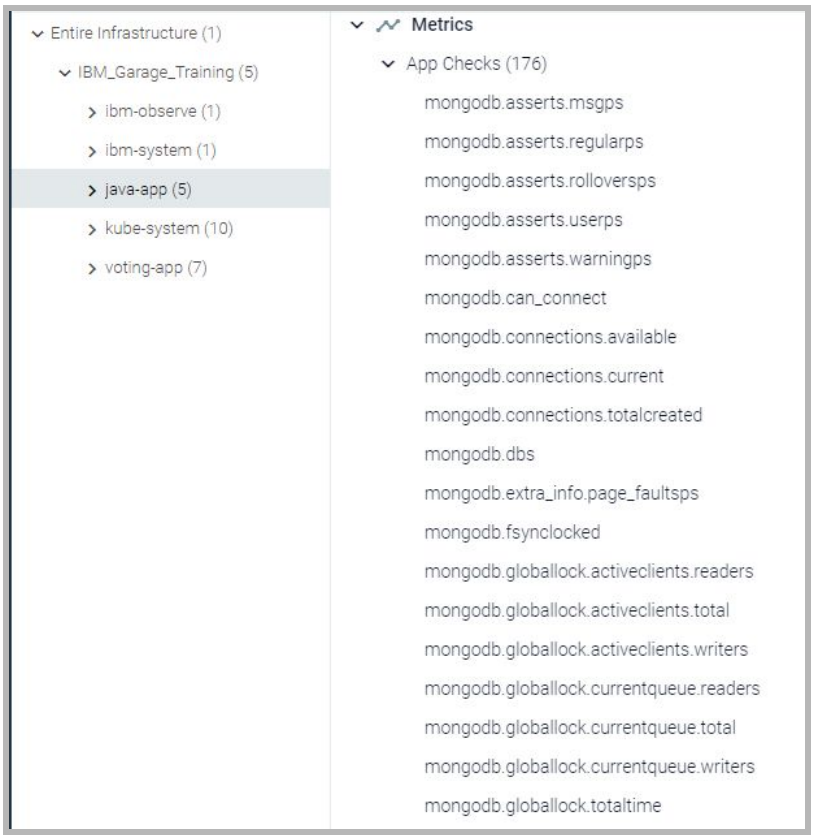


Clicking into these we can drill into more details on each of these specific applications. Using the dashboard search box, we can look for any other Cassandra metrics / dashboards for example, and we see that there are a number of JMX based Cassandra specific metrics.



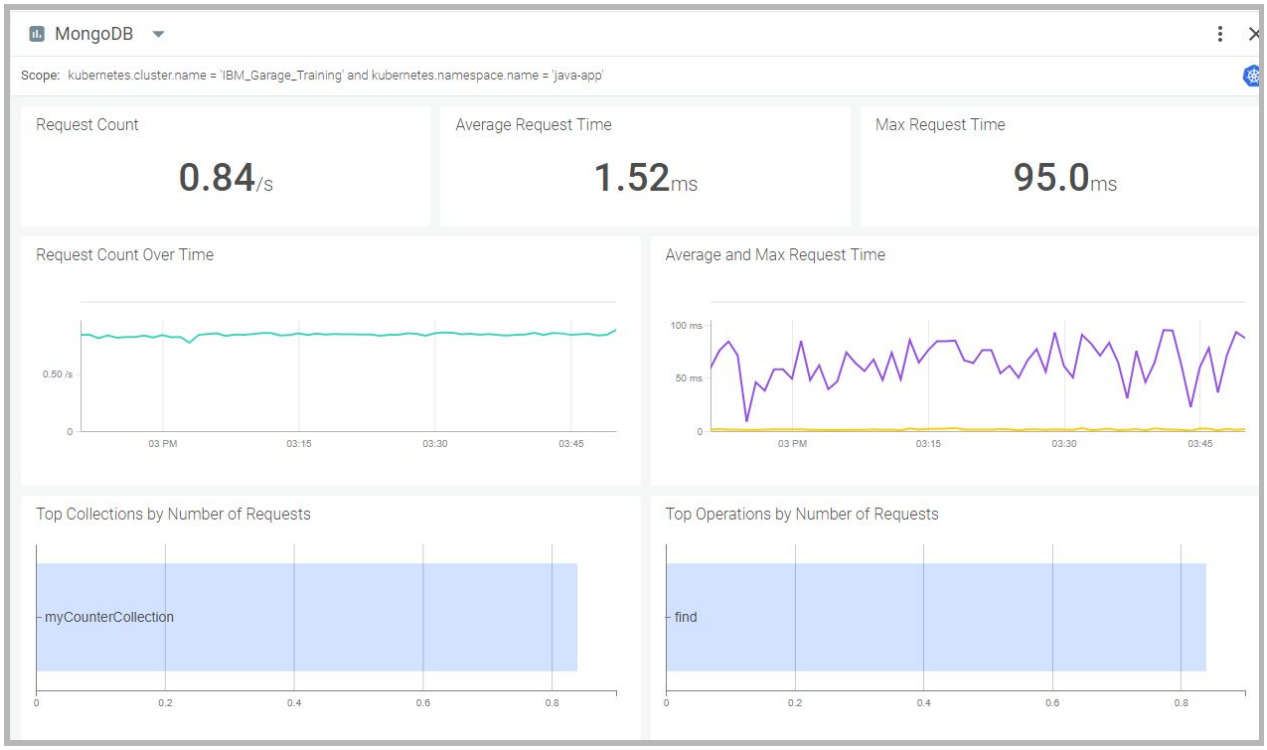
AppChecks

AppChecks are Sysdig's way of auto-detecting applications, and where possible collecting application specific metrics. Many applications are auto-detected and auto-collection needs no additional configuration. Some applications (for example: RabbitMQ and MySQL) have a mechanism for a user to login and gain additional metrics, and this can be provided to the Sysdig Agent configuration. Full documentation on configuration AppChecks can be found here:
https://sysdigdocs.atlassian.net/wiki/spaces/Monitor/pages/204767363/Integrate+Applications+Default+App+Checks



If you look at the dashboard dropdown in Explore view, and scroll down to the Metrics section, you'll see the AppCheck specific metrics that are being collected by the Sysdig Agent. You'll notice that with no additional configuration we should already be collecting metrics for the following applications:

- MongoDB
- NGINX
- PostgreSQL
- Redis

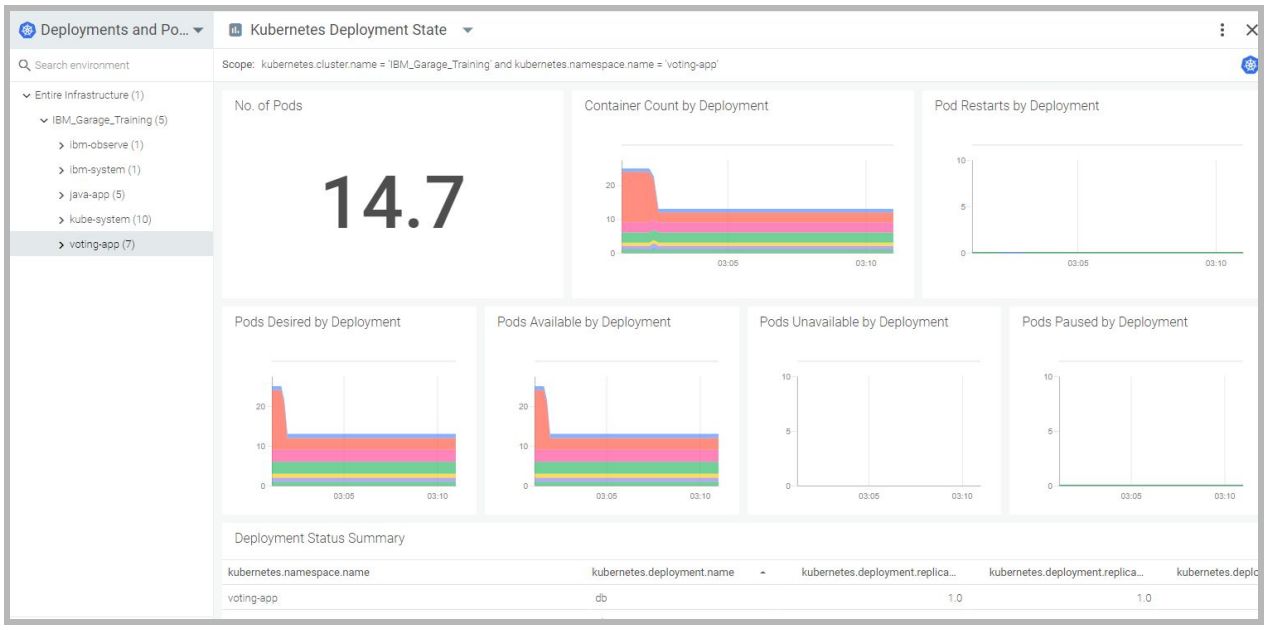


Kube State Metrics

Kube State Metrics give us information about what is deployed into Kubernetes. The Sysdig Agent automatically collects these from Kubernetes, and you can see these from the default dashboards under Kubernetes. This can be really useful to understand the health of Kubernetes and the Kubernetes deployed applications directly.

If we look at 'Kubernetes Deployment State' we can see details of the state of the different deployments, including the number of pods desired vs. available. This can be important when understanding why auto-scaling isn't working, or perhaps to get a quick view of what each application is consuming.

In the below screenshot we can clearly see the 'fix' script from the last exercise.

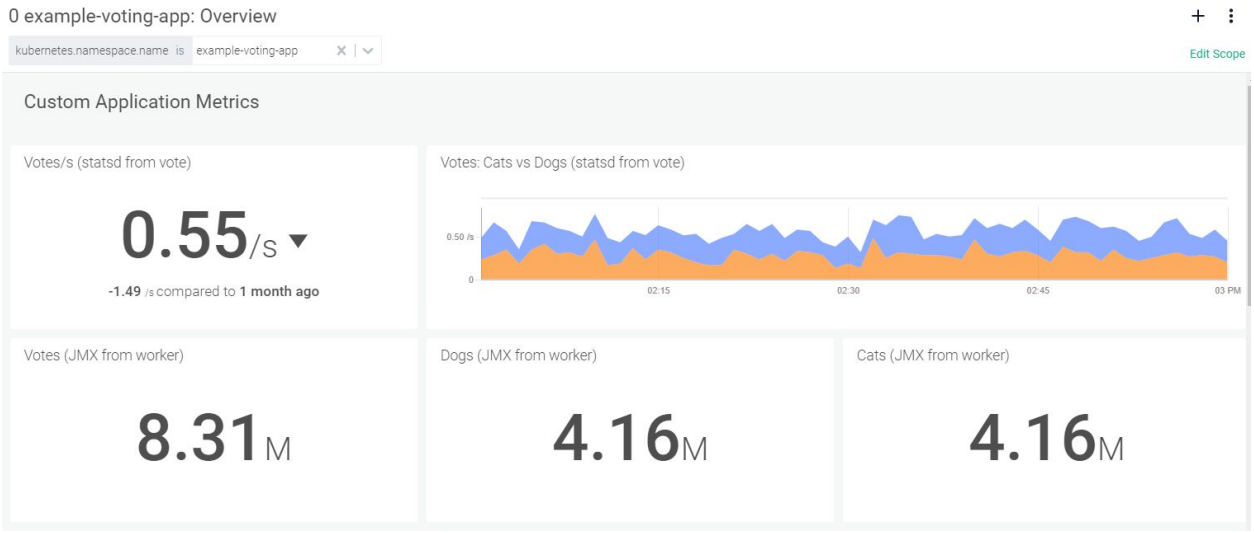


5.4 Create a Java-App dashboard

Using what we've learned about the java-app, and about the different data sources available, create a new dashboard to show the application health. If you can, try to include at least the following metric sources:

- AppCheck metrics
- Kube State Metrics
- JMX metrics

Here is an example dashboard modelled around multiple data sources for the voting application, see if you can discover similar different sources to build up a view of the java-app health.


0 example-voting-app: Overview
kubernetes.namespace.name is example-voting-app
Edit Scope
Custom Application Metrics
Votes/s (statsd from vote)
0.55/s
-1.49 /s compared to 1 month ago
Votes: Cats vs Dogs (statsd from vote)
0.50 /s
0
02:15
02:30
02:45
03 PM
Votes (JMX from worker)
8.31M
Dogs (JMX from worker)
4.16M
Cats (JMX from worker)
4.16M

Sysdig

6 Lab 3: Deployment Health

6.1 Deploy the Demo Application

We're going to deploy a second application that has a little more variation to it. From the location of where you have downloaded the demo application to, run the following:

```
$ kubectl create ns stress
$ kubectl apply -f lab3/manifests/ -n stress
```

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

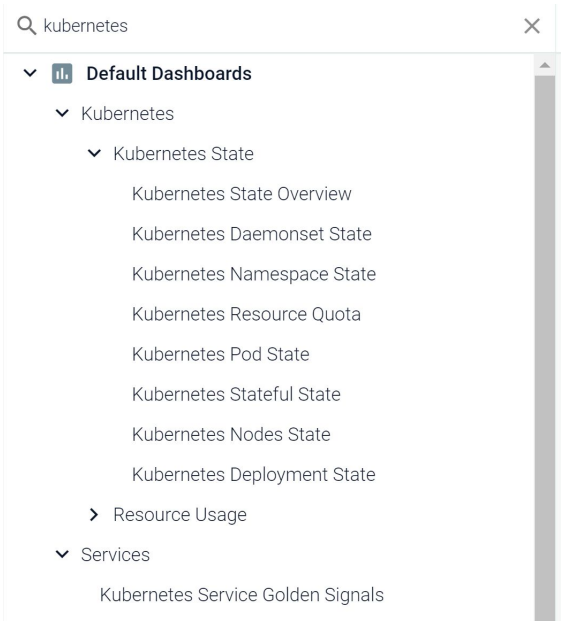
```
$ kubectl get pods -n stress
```

Don't wait for all the pods to hit 'Running' status, just confirm that they have been deployed.

```
wizard@Sysdig:$ kubectl get pods -n stress
NAME                            READY   STATUS    RESTARTS   AGE
k8s-stress-568fc7b786-dlgb4     0/1     Pending   0          7s
k8s-stress-568fc7b786-grcvz     0/1     Pending   0          7s
k8s-stress-568fc7b786-h726c     0/1     Pending   0          7s
k8s-stress-568fc7b786-hv85h     0/1     Pending   0          7s
k8s-stress-568fc7b786-mq5kh     0/1     Pending   0          7s
k8s-stress-568fc7b786-xfjrd     0/1     Pending   0          7s
```

6.2 Kubernetes State Metrics

Kubernetes State Metrics (or kube-state-metrics, or KSM) are generated by Kubernetes itself and reporting against the state of various components and objects in the cluster. In the dropdown dashboard explorer, either search for 'kubernetes' or browse to Kubernetes -> Kubernetes State and find the various default dashboards available.



Have a look through the various dashboards and see if our new 'stress' application deployment looks healthy.

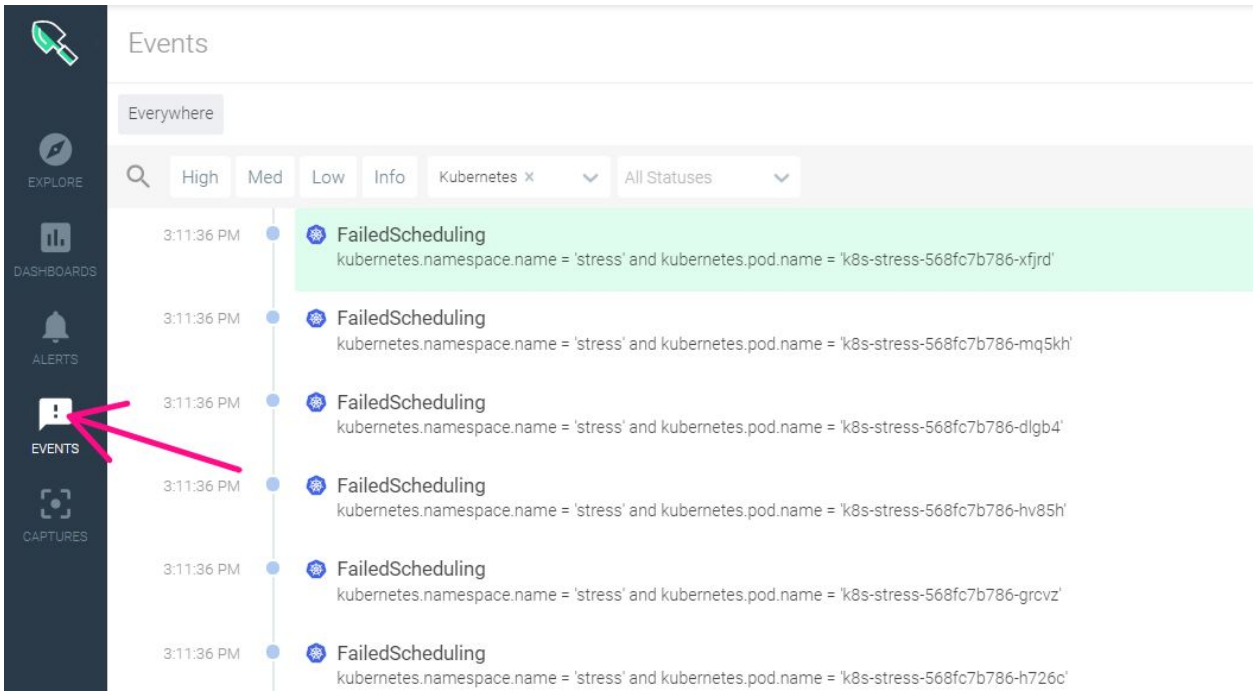
1. What is the current state of the application?
2. How many pods should it be running vs how many are running?
3. Can you see any visible resource constraints on the application?

Also explore the Kubernetes -> Resource Usage dashboards to discover more about available vs. allocated vs. consumed resources.

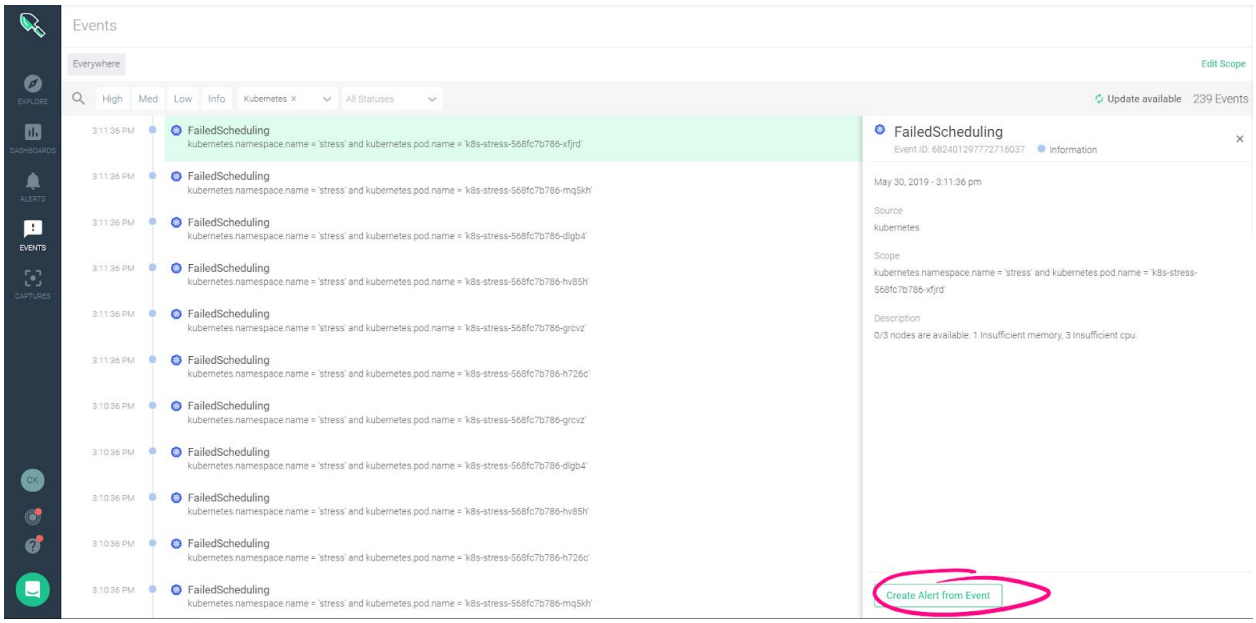
6.3 Events

We did a little digging with events earlier, but we're going to use them again to try and diagnose our deployment. From the last section of discovery we saw that our deployment wasn't exactly healthy, and while we may have got a hint at what the issue is, we don't have the full picture just yet. Let's have a look at 'Events' to see if there is anything useful here.

Depending how busy you made your cluster from previous lessons will depend what you see, so the quickest thing to do is to clear up our events list. In the top filter menu, click the dropdown for 'All Types' and choose 'Kubernetes'. You should see some 'FailedScheduling' events. If we click on one, can you see what might be causing this event?



Create an alert for this event. In theory you may want this going back to the development / application team directly, but for now just create a simple alert for this event. This event may become noisy, so watch it closely.



6.4 Fixing the Deployment

Having discovered the root cause of the deployment issue, how would you propose solving this issue? Please keep infrastructure costs in mind when solving the issue, and consider actual resource usage. Make your fix only using the kubectl command and a file editor.

7 Lab 4: Authentication and Access Control

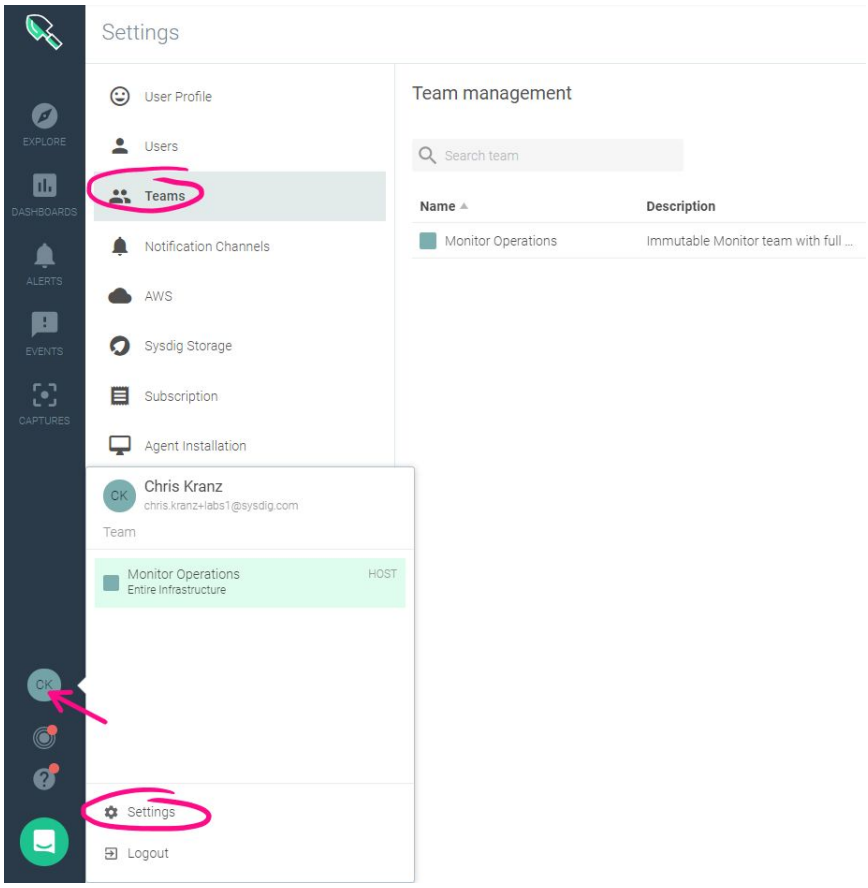
7.1 Authentication methods

In this chapter we will review the different methods for authenticating against Sysdig. As configuring different authentication providers can have complex dependencies (setting up authentication realms), we won't be doing that here, simply reviewing what is possible. The following authentication providers are available:

- LDAP (on-premises backend deployment only)
- OpenID
- SAML
- Google

7.2 Sysdig Teams

Teams are Sysdig's implementation of role based access control. This allows grouping of users by various metadata objects and allows limited scope of visibility. On one side this allows users to focus on what's important to them without the noise of everything else, on the other side this allows for secure setup and limiting groups or individuals to only view what they have permission to view. Get to teams by clicking on your login initials, then click Settings, then Teams.



7.3 Creating Teams

Create a new team that only has access to the voting application we deployed in Lab 1 / Chapter 4. Switch your own context to the new team to confirm it works. Add someone else in the room to your team, keep in mind that logins use email as a unique field, so you may need something like chris.kranz+team2@sysdig.com when adding new users. Alternatively you can simply add yourself with a unique identifier. This user has to login at least once before you can assign any permissions or teams, so make sure the instructions in the welcome email are first followed.

Team management

Search team

Add team

Name	Description	Scope By	Scope	Default Team
Monitor Operations	Immutable Monitor team with full ...	Host		✓

New Team

Name: voting-app

Description: Team that only has access to view the 'voting-app' namespace

Default Team: Users with no designated team will be added to this team by default

Default Entry Point: Explore

Visibility

Scope By: Host / Container

Scope: kubernetes.namespace.name | is | voting-app | AND

Select a label

Clear All

Additional Permissions: Sysdig Captures, Infrastructure Events, AWS Data

Team Users

Assign User

Name	Role

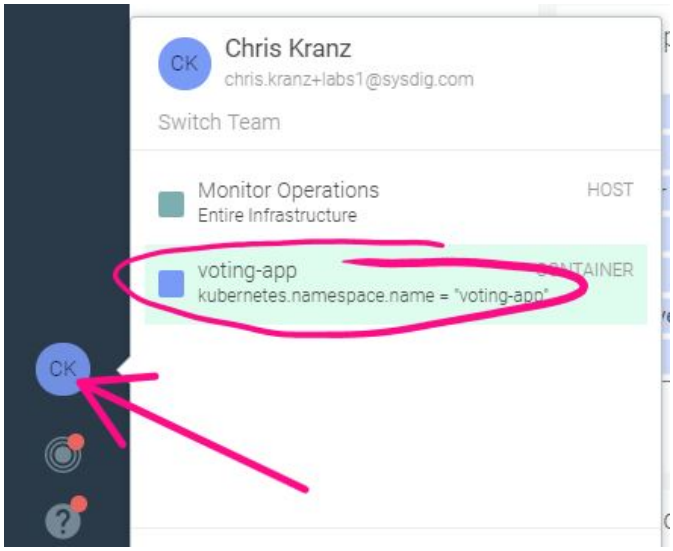
Team Users

Assign User

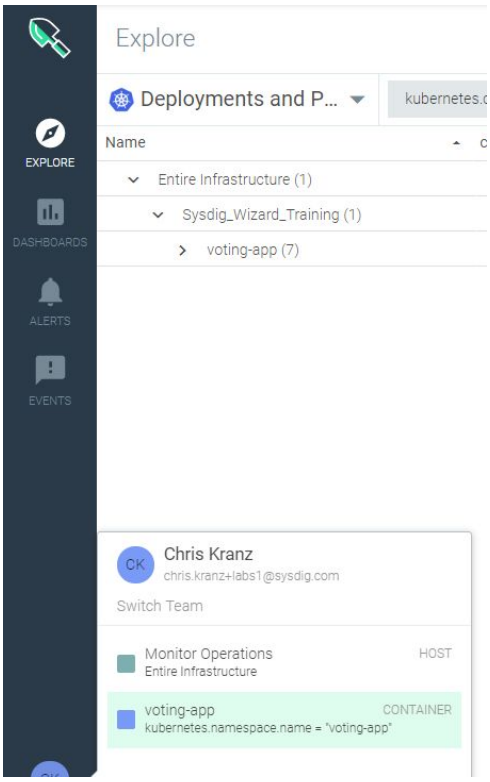
Name	Role
chris.kranz+labs2@sysdig.com	Advanced user

Sysdig

As an administrator, you automatically have access to all teams so don't need to assign yourself. Once a team is created you can switch your view to test that the scope has been set correctly. Click on your login initials and then select your newly created team. You may need to refresh the screen in order to clear out your previously cached views of the infrastructure.



Once logged into the restricted team, confirm the view is confined to the relevant permissioned areas.



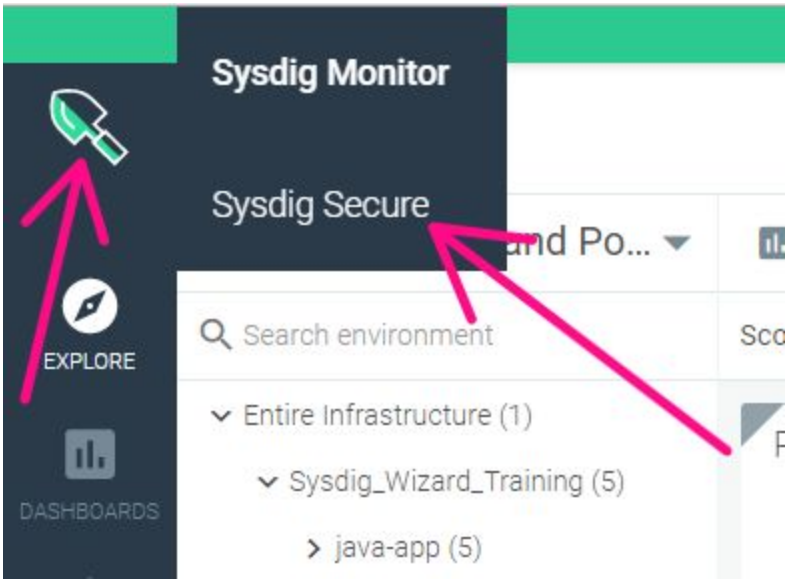
7.4 Additional Teams

1. Create a team that is limited to the java-app, add yourself and switch teams to test
2. Create a team that scopes out a grouping to include both the java-app and the voting-app, but nothing else. Add yourself and switch teams to test
3. Create a team that has visibility of the sysdig-agent only. Add yourself and switch teams to test

8 Lab 5: Securing your Images

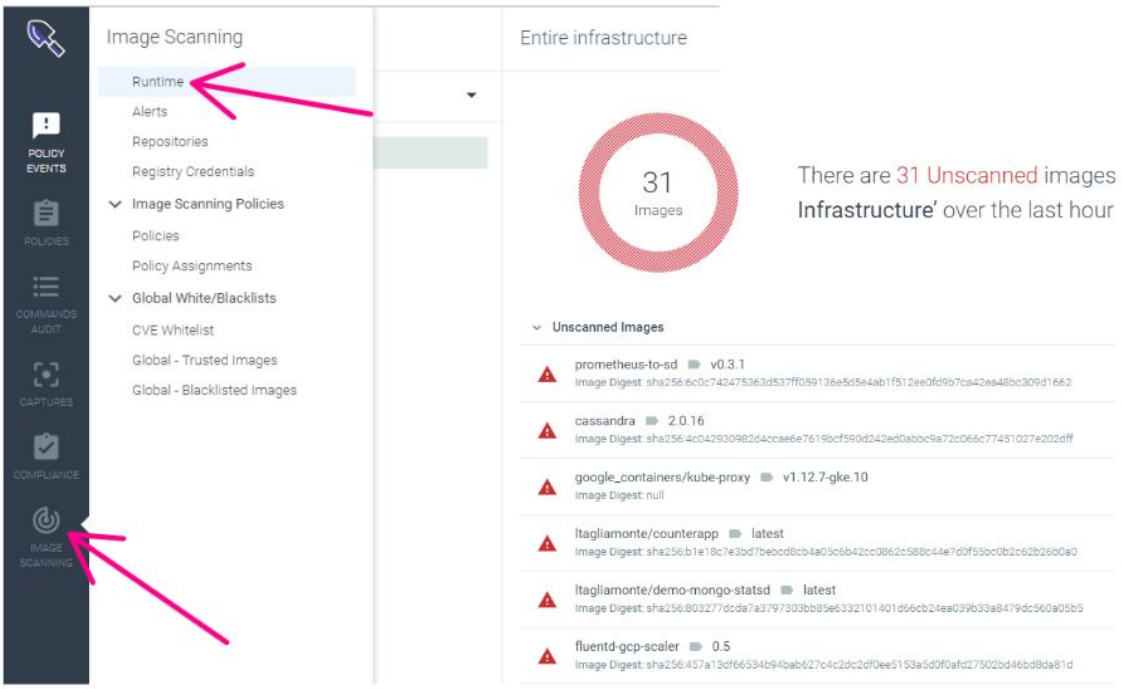
8.1 Login to Secure

Click on the upper left spade icon on Sysdig Monitor and choose Sysdig Secure, this will open up in a new window/tab and you may be asked to login. Use the same credentials you have been using for Sysdig Monitor so far (not any team scoped login you may have created in the last lab).



Now click on 'Image Scanning on the left hand side menu.

If you click on 'Image Scanning' on the left hand side, you'll be taken to a view that shows the current security posture of your environment.



None of the images should have been scanned already (the exact number of images may vary from the screenshot here), so you'll see a view that shows all images as unscanned. You can also limit the scope of what is displayed in the same way you can in Sysdig Monitor by clicking on the drop down on the top left hand side which defaults to 'Hosts & Containers', or change the groupings with the dropdown.

8.2 Deploy the Demo Application

We're going to deploy a second application that has a little more variation to it. From the location of where you have downloaded the demo application to, run the following:

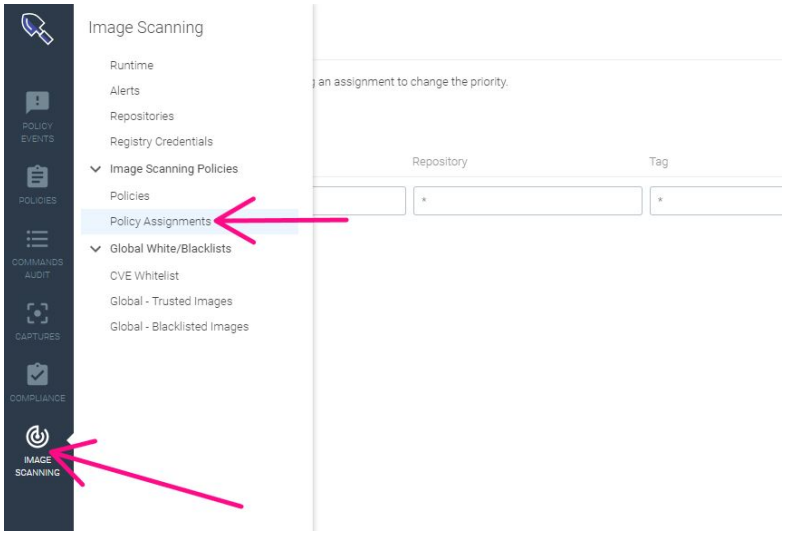
```
$ kubectl create ns web-app
$ kubectl apply -f lab5/manifests/nginx-1.yaml -n web-app
```

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

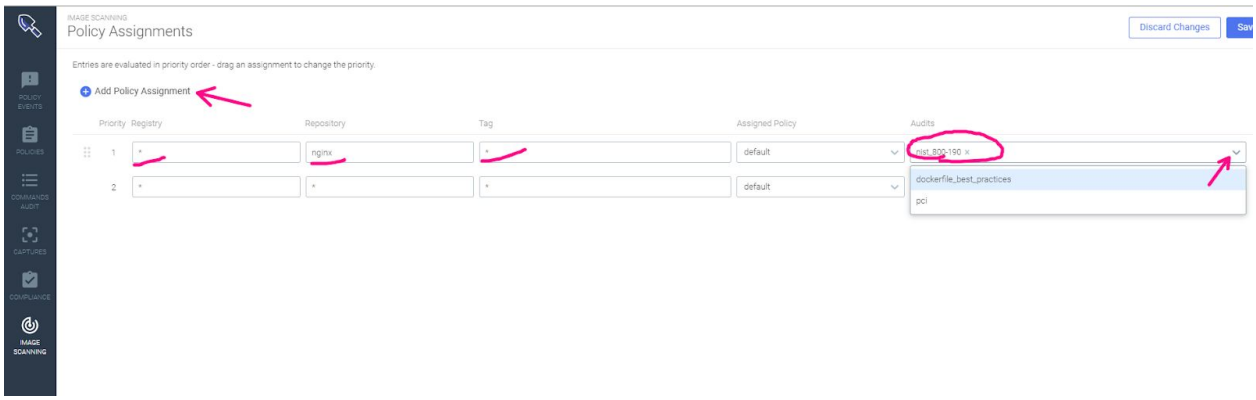
```
wizard@Sysdig:$ kubectl get pods -n web-app
NAME                     READY   STATUS    RESTARTS   AGE
nginx-767988fcc8-zk4xr   1/1     Running   0          6s
```

8.3 Assign the NGINX application a policy

By default, all images will get scanned against the default policy, but this can be configured. From the 'Image Scanning' screen, click on 'Scanning Policies', then 'Policy Assignment'.

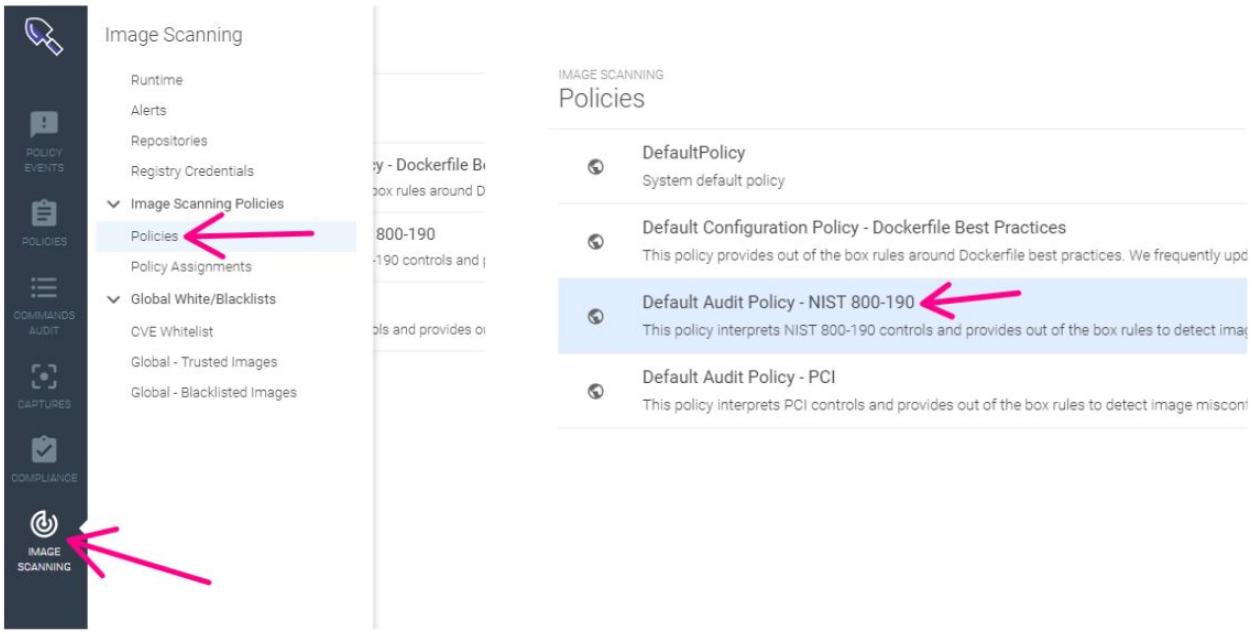


Add a new Policy Assignment and then fill it out in order to pickup the nginx repository and assign the policy ‘nist_800-190’. Put a star in other boxes to match against anything. Then click ‘Save’ in the top right.

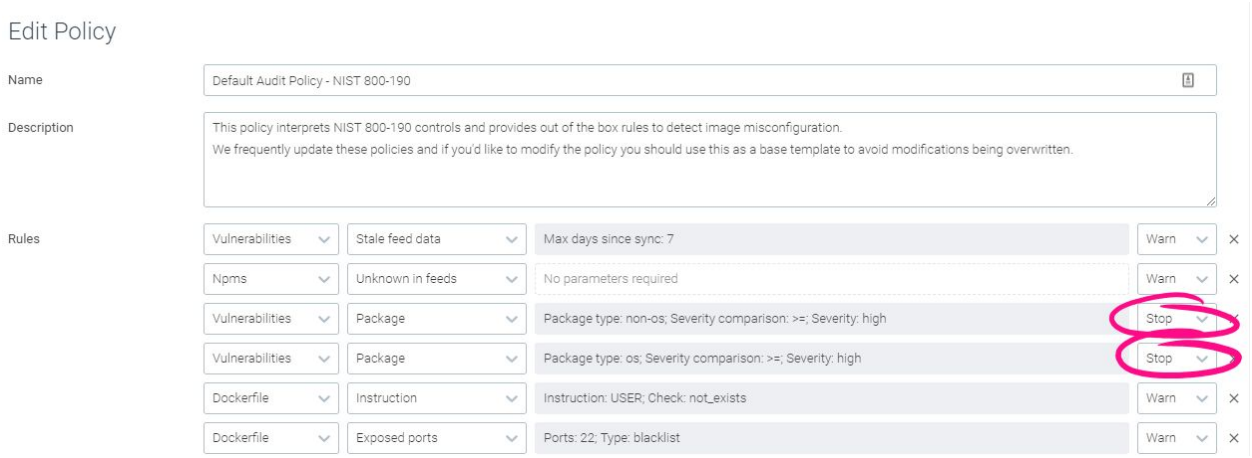


8.4 Change the policy evaluation severity

Under ‘Image Scanning’ -> ‘Policies’, then select the ‘NIST 800-190’ policy:



Against the OS & non-OS vulnerability checks, change the Warn/Stop option to 'Stop'.

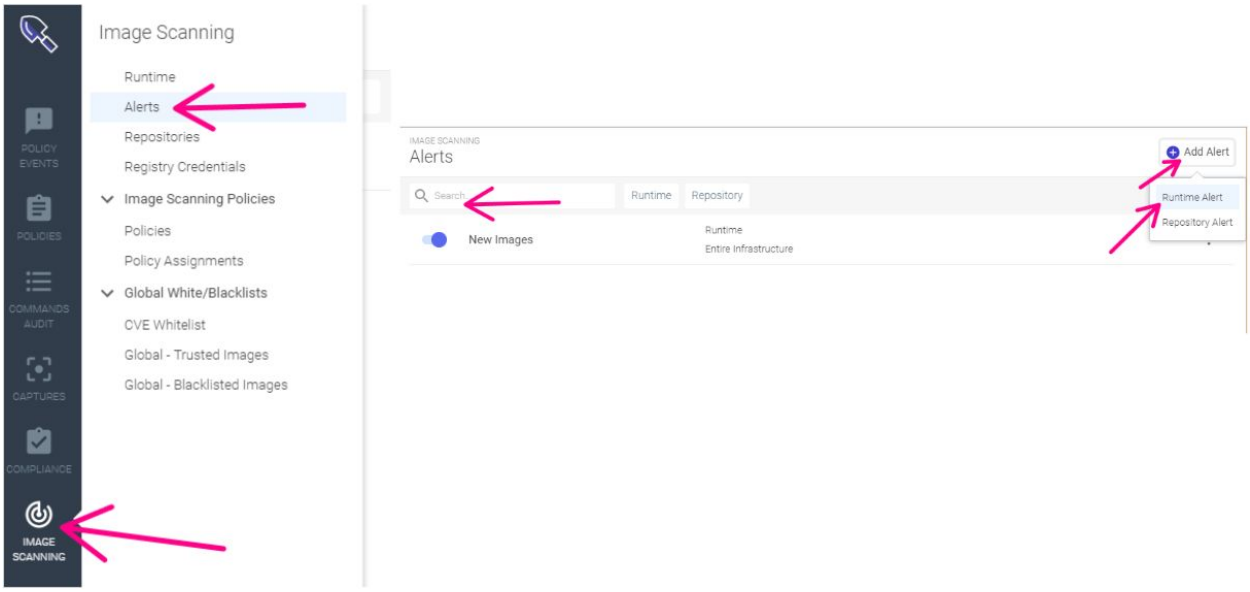


Feel free to make other changes and see what options are available to choose from and modify.

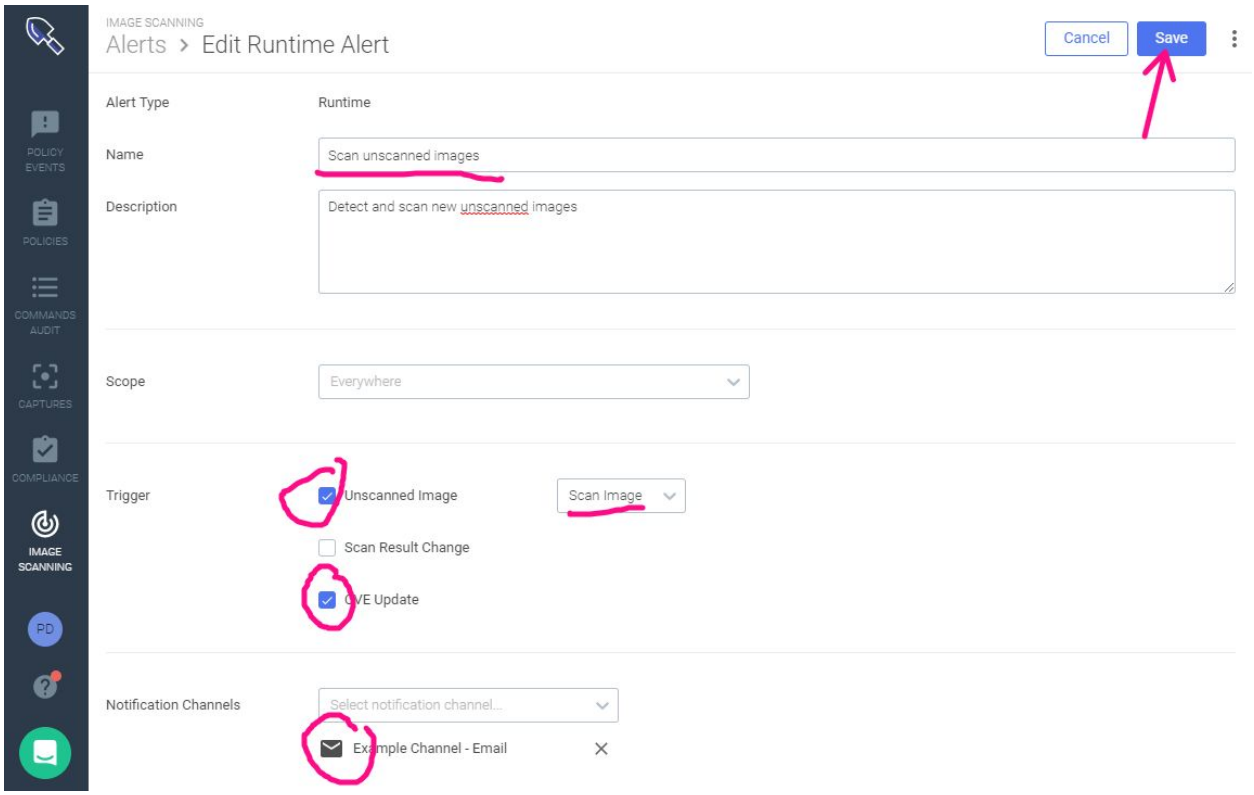
8.5 Scan all Images

You can setup an alert that detects any new image that hasn't been scanned, and create an action that automatically scans the image. Lets do that so that all your existing images get scanned.

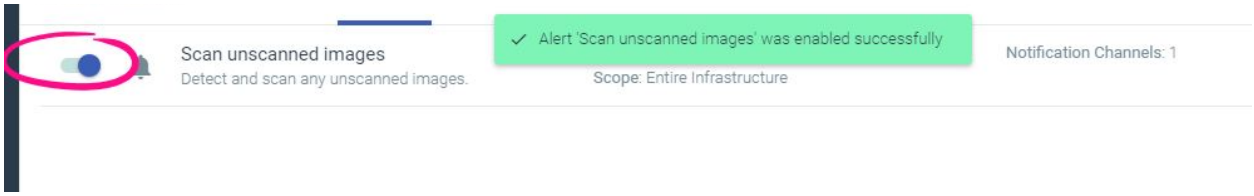
Click on 'Image Scanning' -> 'Alerts', then 'Add Alert'.



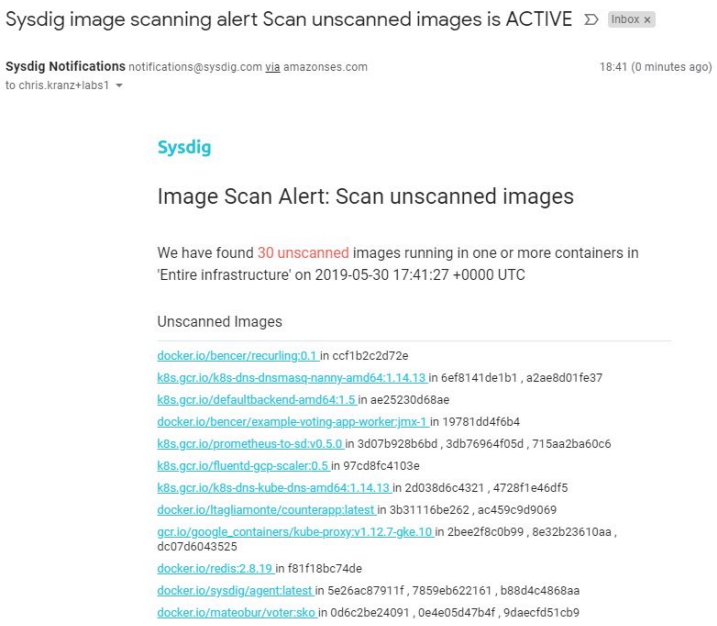
Fill out the form, make sure to select a trigger to 'Scan Image'. If you like you can also set a notification to email yourself.



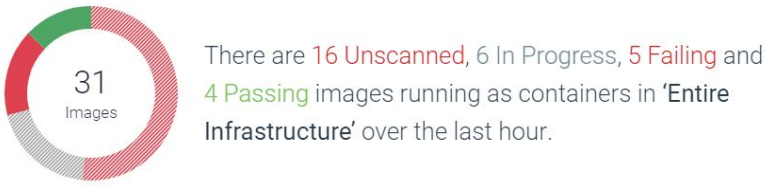
Make sure the alert is enabled after you've created it (it should be by default). This will now scan all images against the default policy when it detects they haven't yet been scanned.



In a minute or two you should get a notification (if you created on) that unscanned images were found.

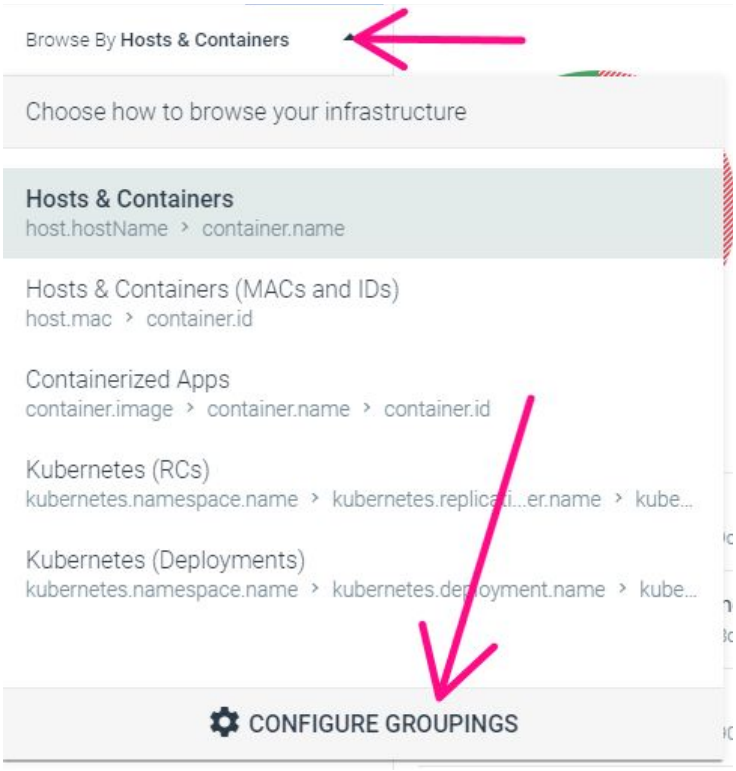


And slowly the images will be scanned and the results page updated to show the scan progress and results.

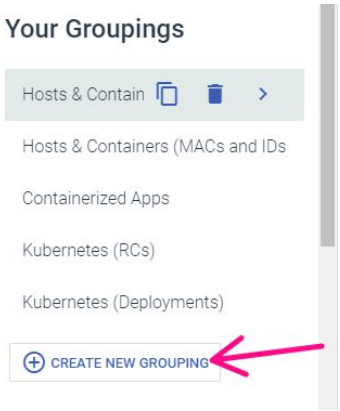


8.6 Configure Groupings

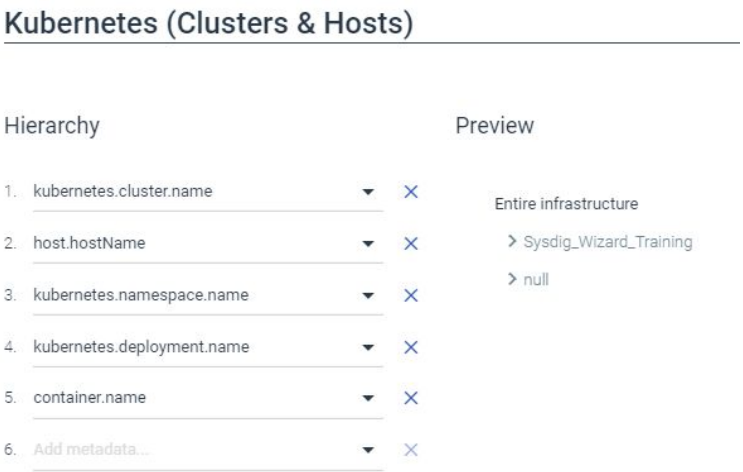
The default groupings of images might not be helpful, so create your own grouping to group against namespaces and deployments. From the 'Runtime' page of Image Scanning, click the dropdown on the top left and then click 'Configure Groupings'.



From here, click on ‘Create New Grouping’

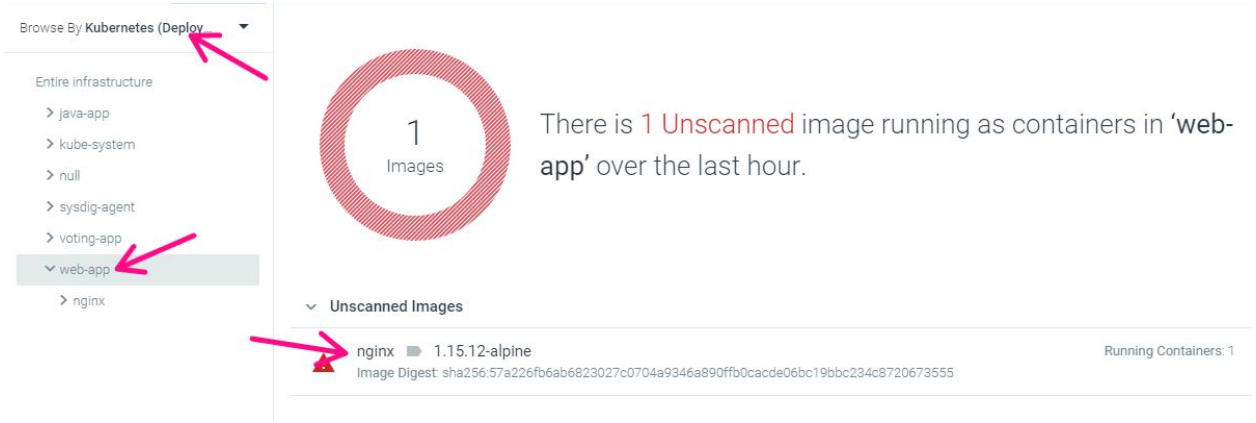


Create a grouping of your choice. Feel free to play around with different levels. Use the search feature by just typing ‘namespace’ or ‘cluster’ and then selecting the full annotation. Give the grouping your own name that makes sense when returning to it later. Below is an example.

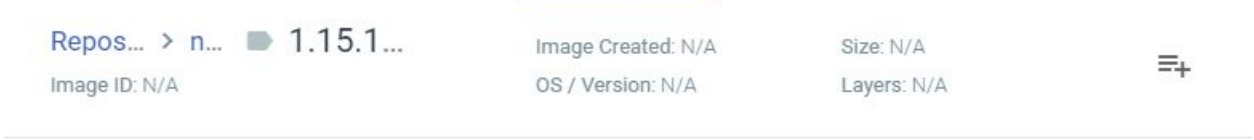


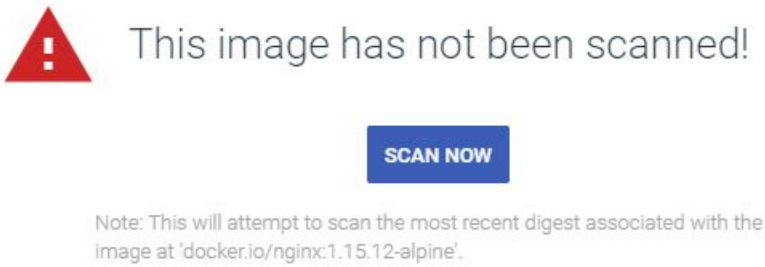
8.7 Scan Now!

Choose a grouping that reveals the namespaces or deployments, find the web-app or nginx deployment.

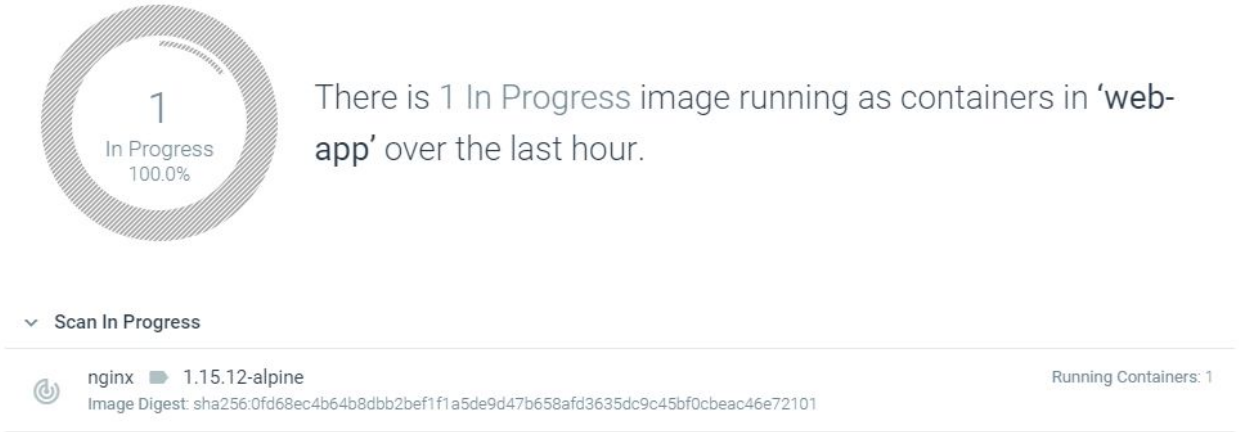


Click on the unscanned image name and you can now choose 'Scan Now'. Again, this will scan against the default policy.





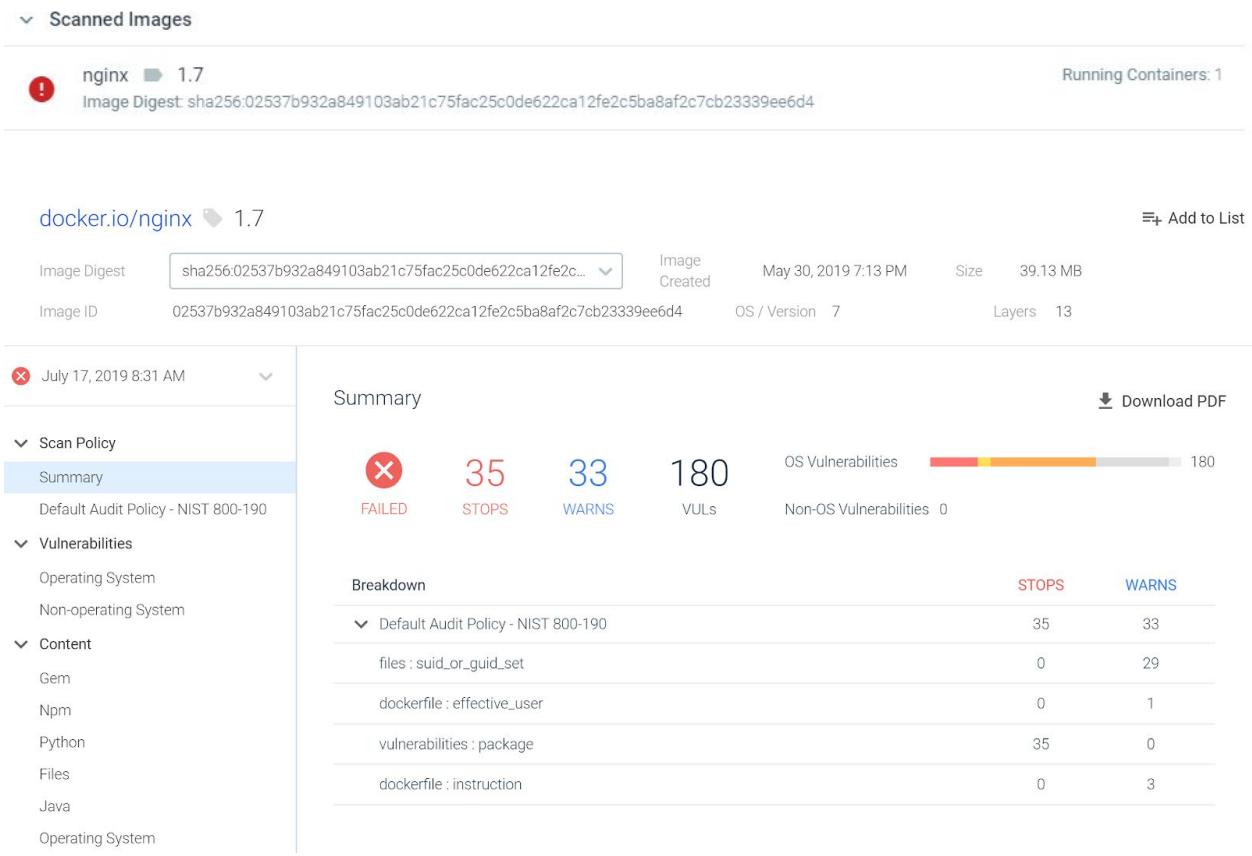
Even though the alert we configured earlier will scan this image eventually, we don't want to wait so by clicking 'Scan Now' we send it immediately to the scanning engine rather than it being somewhere in the queue. You'll now see the status changed to 'In Progress'.



And if you wait a minute or two, then refresh the page this should change to show the results of the scan, which should be 'Failing'.

8.8 Review the Results

Click on the failing image and you'll be taken to the results page.



From here you can see that this image is being evaluated against the NIST 800-190 policy rather than the default policy, and we can see the 'Stop' actions being highlighted as failing the image scan. A contextual link is provided to the various vulnerabilities that have been detected. You can explore the results of the image scan, what the policy is evaluating against, details against the vulnerabilities and even the contents of the image.

If a new vulnerability is released, or if you change the policy assignment, the image doesn't need to be scanned again as the contents are stored in the image scanning database. This makes updating against policy changes or new vulnerabilities incredibly fast and efficient.

8.9 Patch the image

For simplicity, to patch the image we have provided an updated YAML. Deploy this to overwrite the existing deployment.

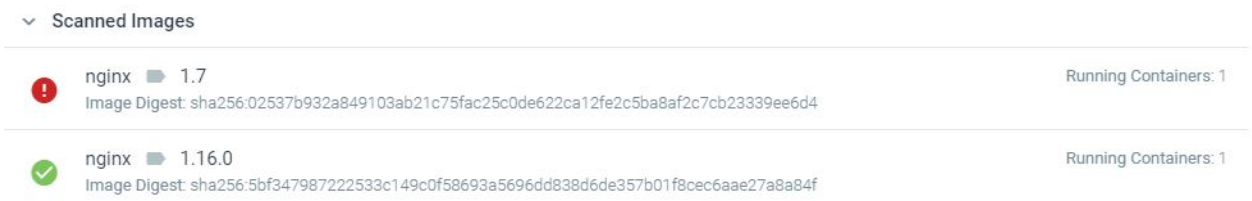
```
$ kubectl apply -f lab5/manifests/nginx-2.yaml -n web-app
```

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

```
wizard@Sysdig:$ kubectl get pods -n web-app
NAME                     READY   STATUS    RESTARTS   AGE
nginx-74d54c49c4-b56pz   1/1     Running   0          6s
```

8.9 Rescan and Review

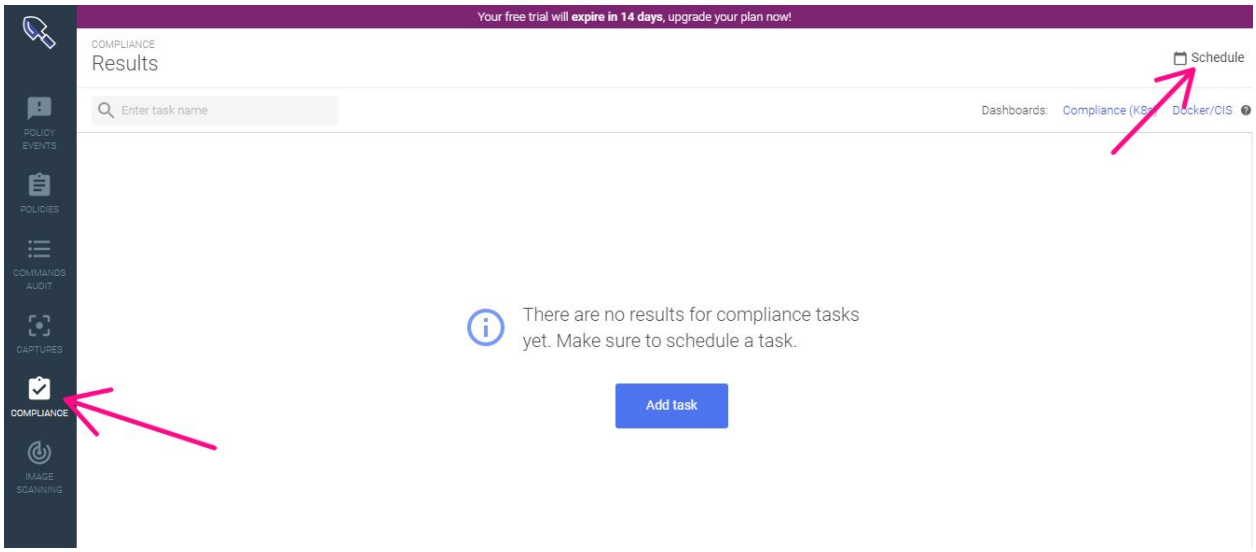
You can either way for the alert to detect the new image and automatically scan it, or once the image appears in the Sysdig Secure UI, again click on it and choose 'Scan Now'. It may take a minute or two for the image to appear. The UI will show both the updated image as well as the old image as it shows everything that has existed over the past hour. You should now see that the image no longer has high or critical vulnerabilities.



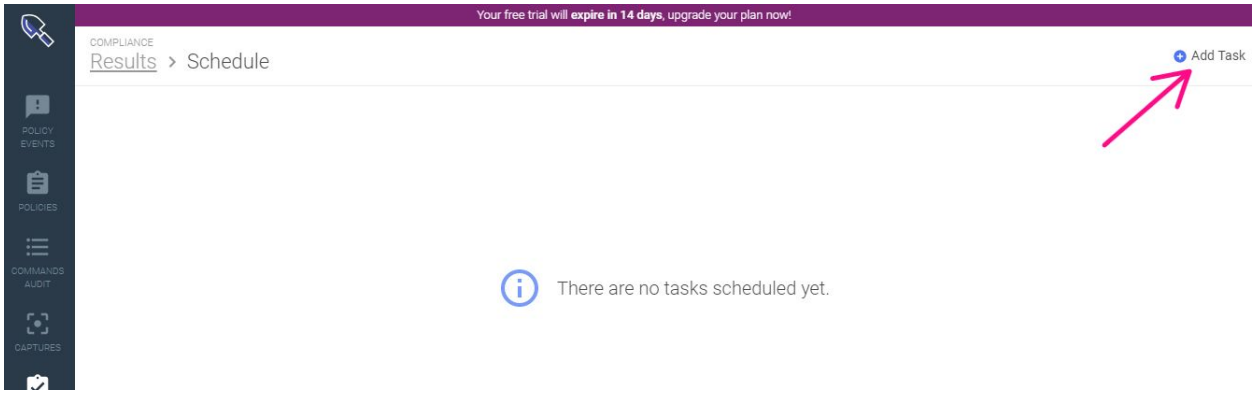
9 Lab 6: Containerized Compliance

9.1 Create a Compliance Schedule

Click on ‘Compliance’ in the left hand menu, then click the ‘Schedule’ button at the top right so we can create a basic schedule to start doing compliance scanning against the environment.

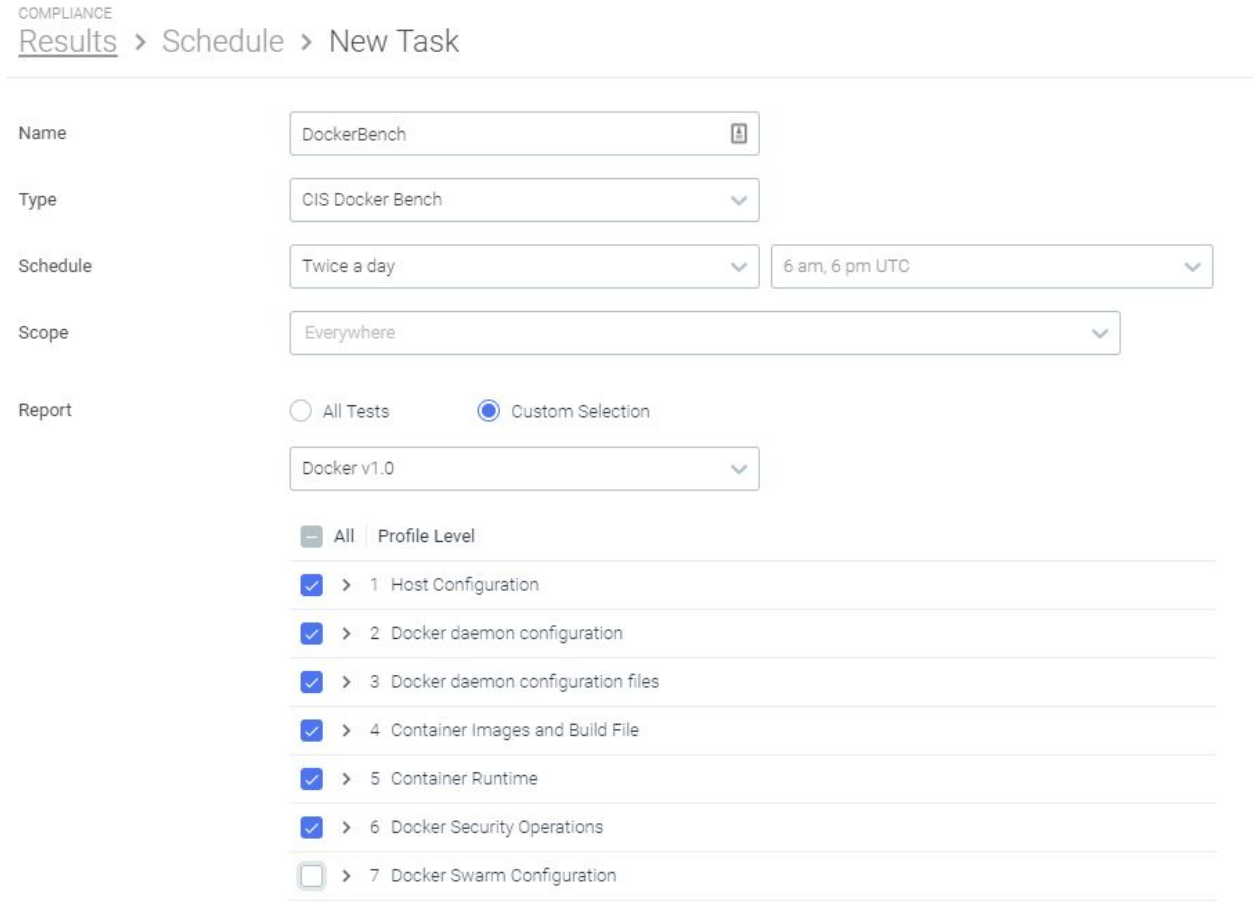


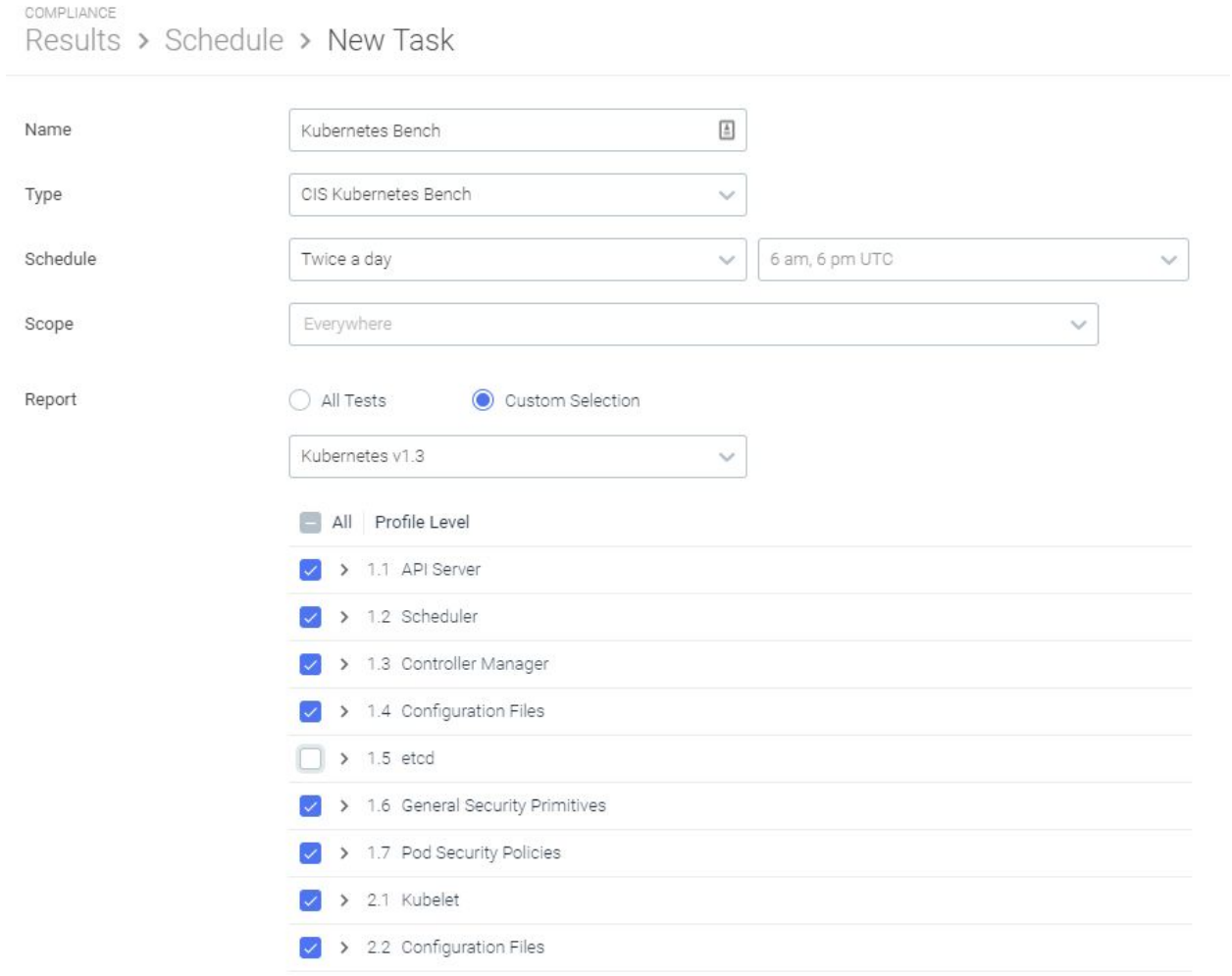
Then click ‘Add Task’



Run through the form to create 2 scheduled tasks, one for DockerBench, one for KubeBench. Look at the ‘Custom Selection’ for each job type, remove anything that seems irrelevant. The

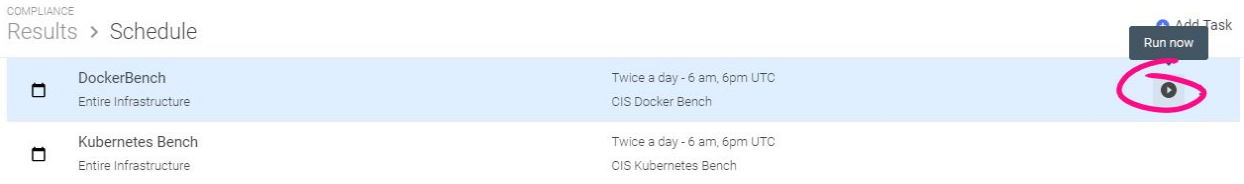
cluster used to create the screenshots is running in GKE, so it seems relevant to ignore Docker Swarm and etcd checks (as we won't have access to the master nodes).

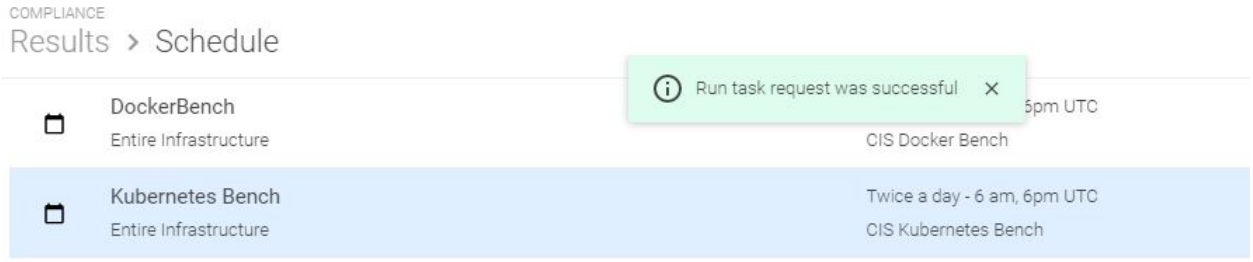




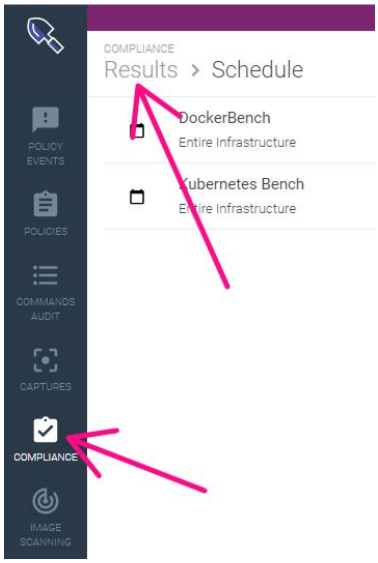
9.2 Run Now!

On the Schedule page, hover over either policy and on the right hand side click the 'Run Now' button (a play icon).

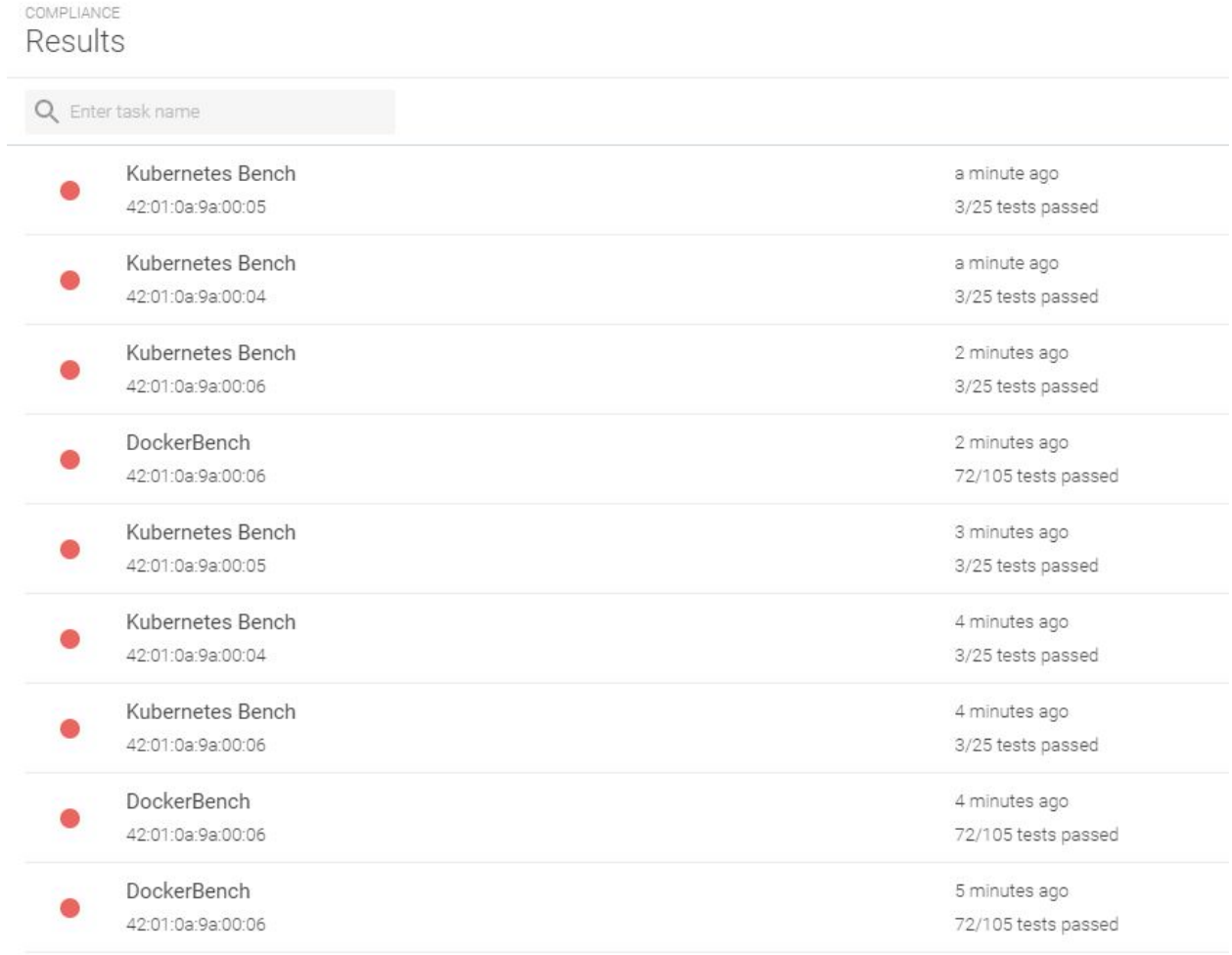




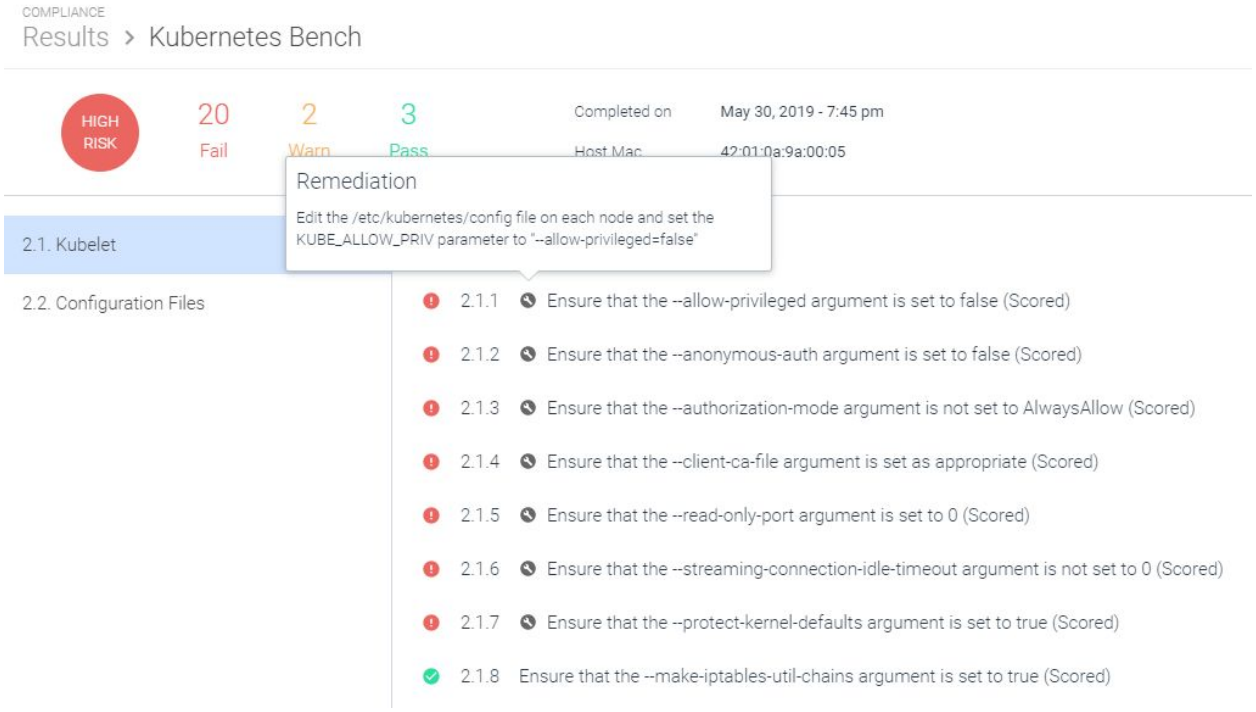
It may take a few minutes for the results to show, but if you go back to the first page for Compliance (either click on ‘Results’ in the top breadcrumb, or just click on ‘Compliance’ in the left again).



9.3 Review Results



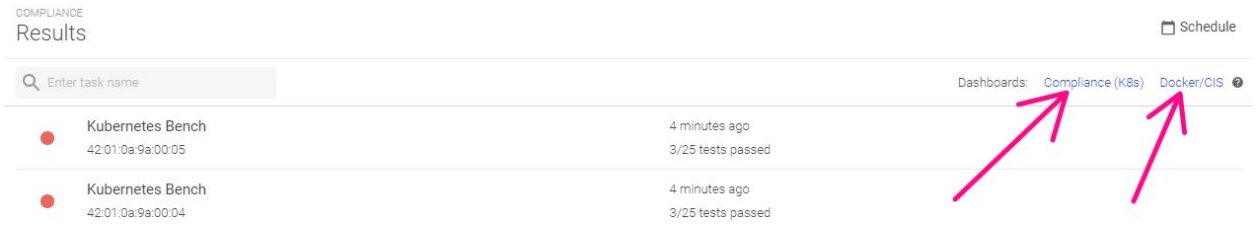
You will see a result for each host, as the compliance checks run against every host in the cluster. Click on any of the results pages and you'll be able to review the results. Note that any failures also provide remediation hints. Depending on the Kubernetes setup, some checks may be blocked, for example in a managed Kubernetes environment such as GKE or EKS, some checks are trying to interrogate the control plane and masters which aren't fully exposed.



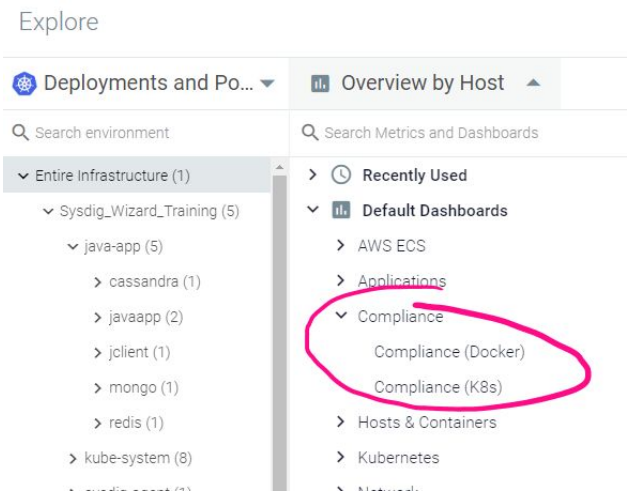
9.4 Continuous Compliance

All results of the compliance checks are stored as metrics, this allows trending over time and alerting if critical checks change status.

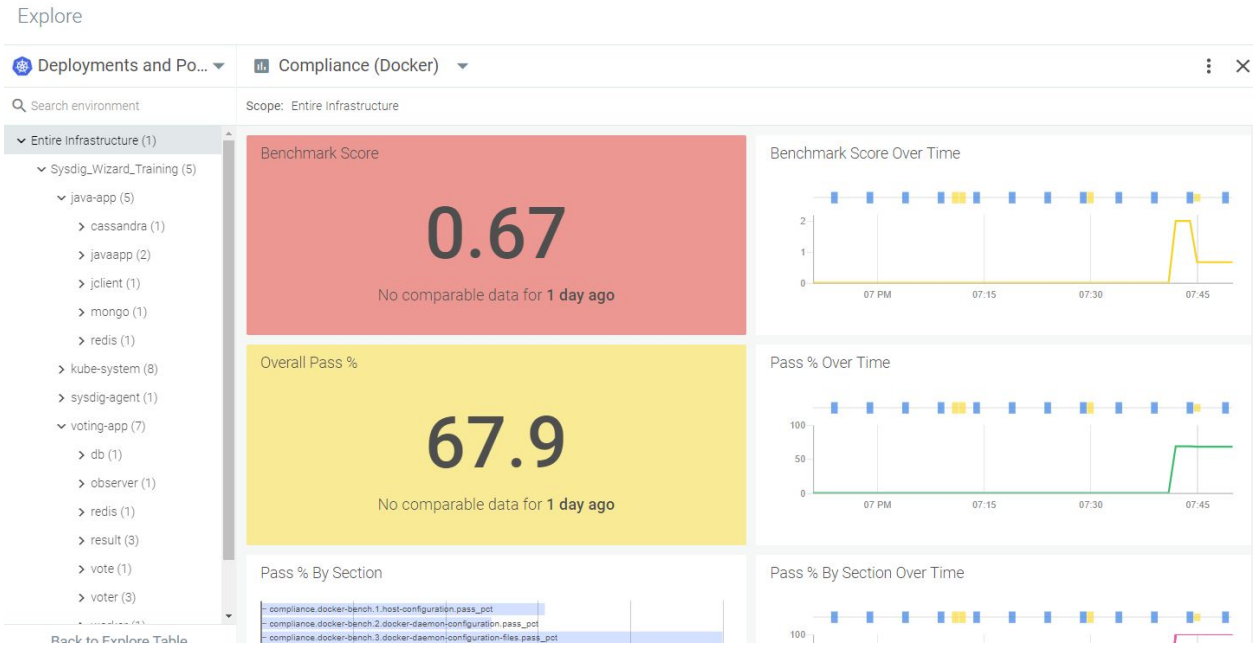
You can get to the compliance dashboard either from the Compliance Results page (top right, click on one of the Dashboard links)



Alternatively from within Sysdig Monitor, you can find the Compliance dashboard under the Default Dashboards dropdown.



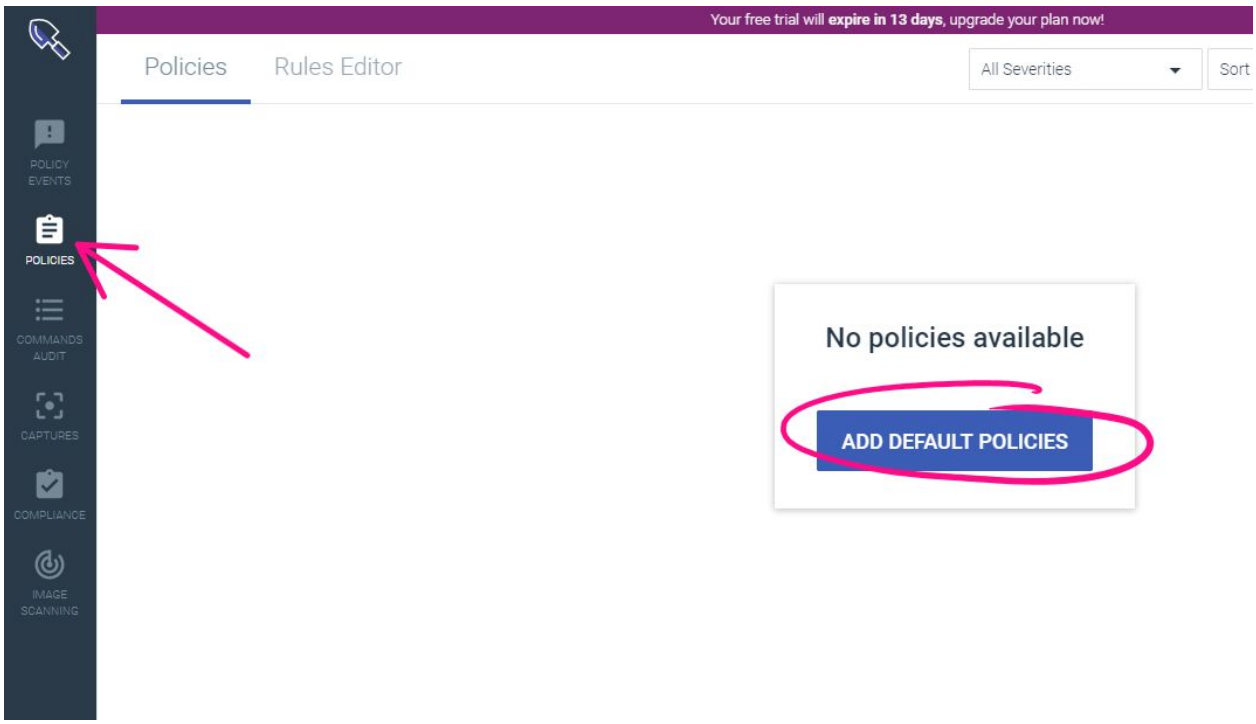
Within the scope of this lab exercise, the trending of results won't be very exciting as we'll only be able to generate a few data points and limited to making changes. If you have time within the day, please attempt some of the remediation fixes and analyse the changes to the results.



10 Lab 7: Runtime Security

10.1 Enable default policies

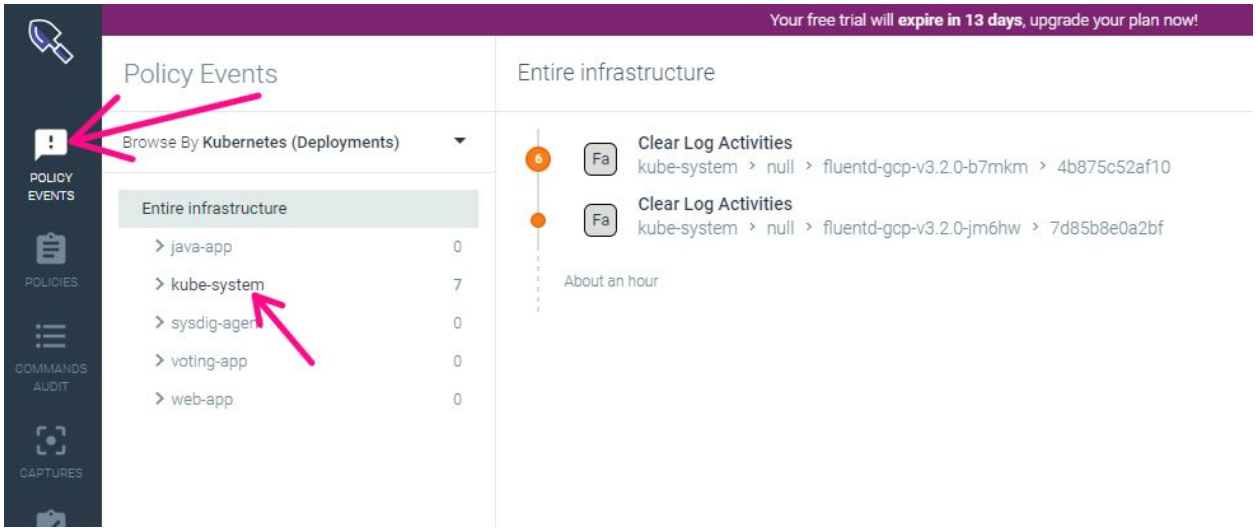
Click on ‘Policies’ on the left hand menu, you should be shown ‘No policies available’, so go ahead and click ‘Add Default Policies’. This will deploy the default policy set, which is a generic set designed to cover 95% of common detection scenarios. The policies are enabled by default, but no notification or preventative actions are configured, so only events within the Sysdig Platform will be created.



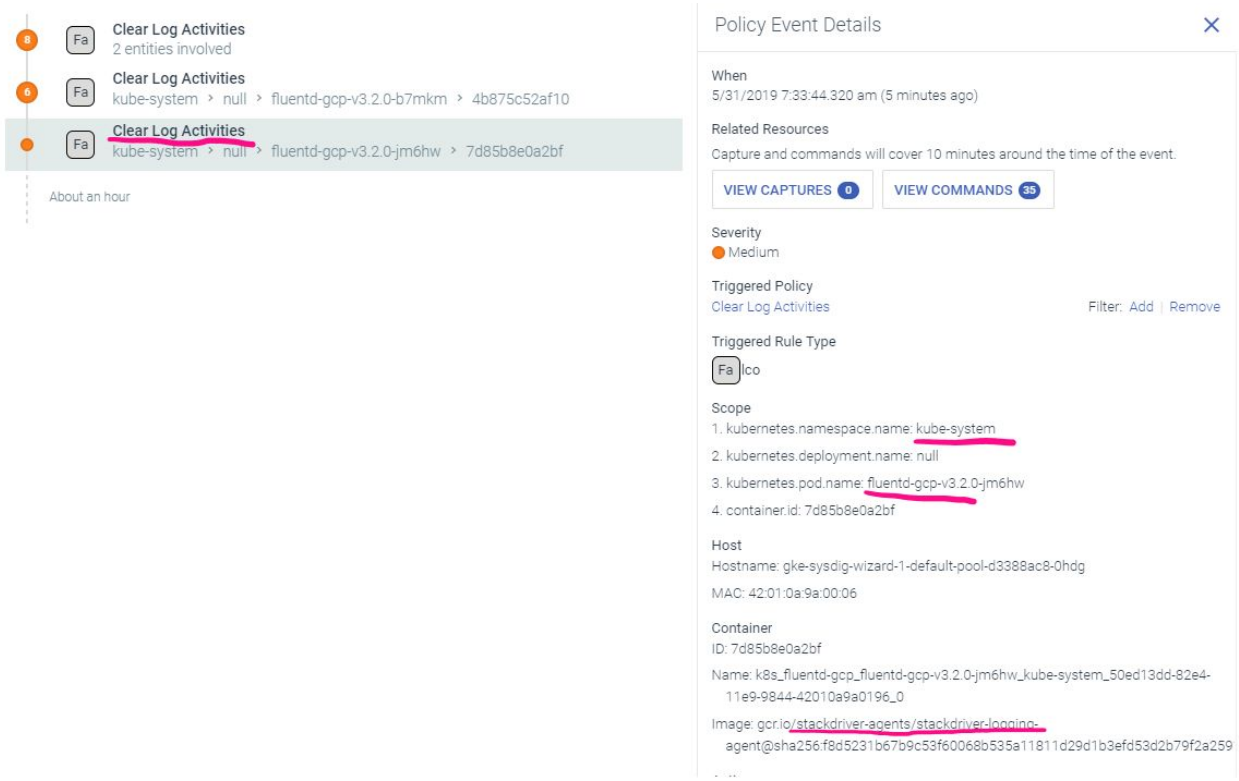
10.2 Whitelisting

Depending on your Kubernetes cluster setup, you may notice that some events start to get triggered straight away. Click on ‘Policy Events in the left hand menu and you may start to see events being generated. The cluster used to create these screenshots is based in GCP and we can see a rule already detecting that logs are being cleared down. As this log clearing is being done by Fluentd, we can guess this is a desirable action, so we’ll run through

whitelisting this. Your deployment may differ from this, so try whitelist something that appears to be working as expected to reduce the noise.

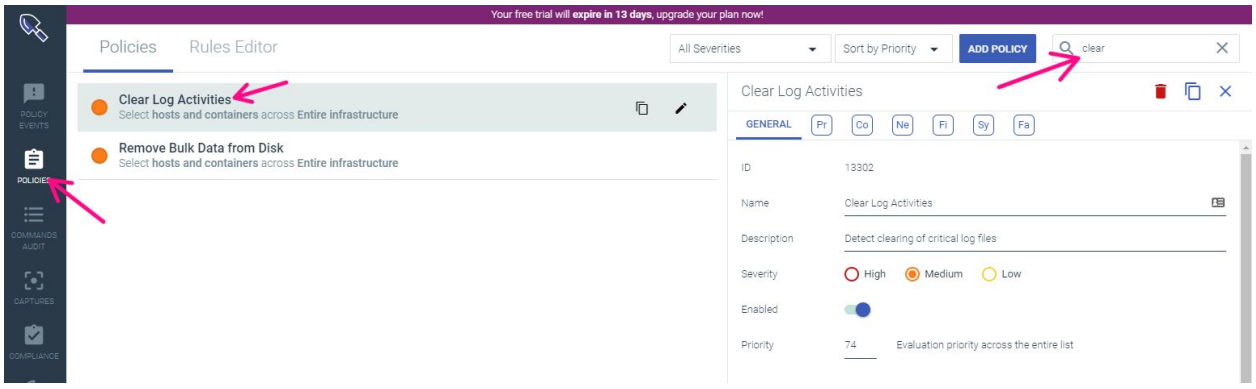


Clicking on one of the events we see the event details. From here we can discover details about the event in order to build up information on what we want to whitelist.



Triggering rule: 'Clear Log Activities'
Kubernetes Namespace: kube-system
Kubernetes Pod: fluentd-gcp-xxxxxxx
Image Name: gcr.io/stackdriver-agents/stackdriver-logging-agent

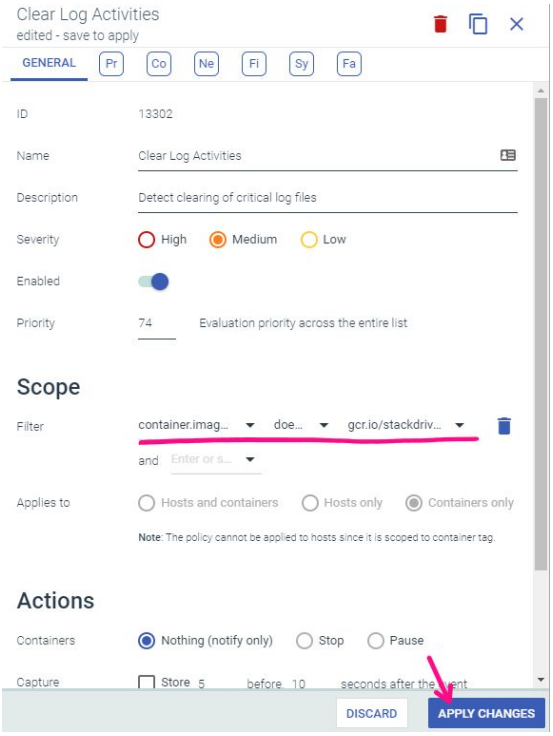
Armed with this information, let's go back to the policies view, and find the policy that is being triggered. Here we've used the search field to quickly find the policy we want.



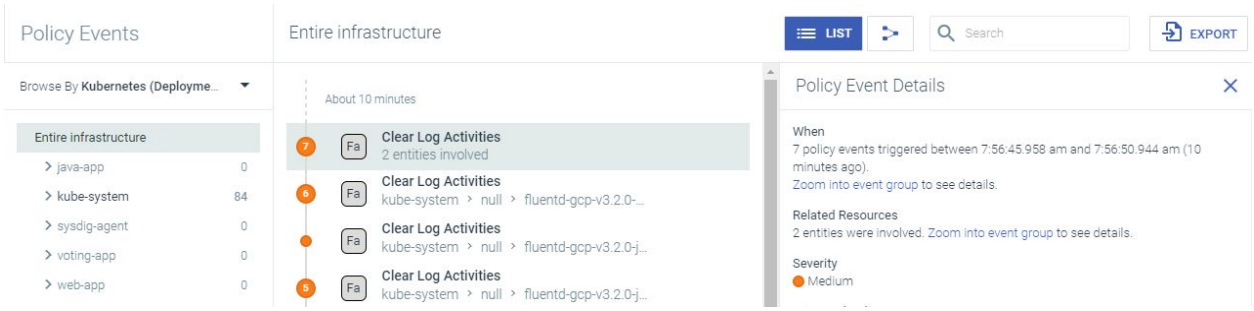
Sysdig

While we could whitelist the entire kube-system namespace, that may represent an obvious attack target, so we're going to specific whitelist the image source here, including the gcr.io repository so we can effectively trust any future stackdriver-agents coming from that repository.

Under 'Scope' we selected 'container.image.repo, using the operator 'does not contain' and then the start of the image repository name, i.e. 'gcr.io/stackdriver-agents/stackdriver-logging-agent', so it'll match if they add versions later. Then we hit 'Apply Changes'.



Go back to policy events, we see a bunch more events have been created and on average these are being triggered around every few minutes. Lets see if any new events are now created. We can see that after around 10 minutes, no new events have been triggered by this pod, so we've successfully whitelisted it.



10.3 Detecting command injection

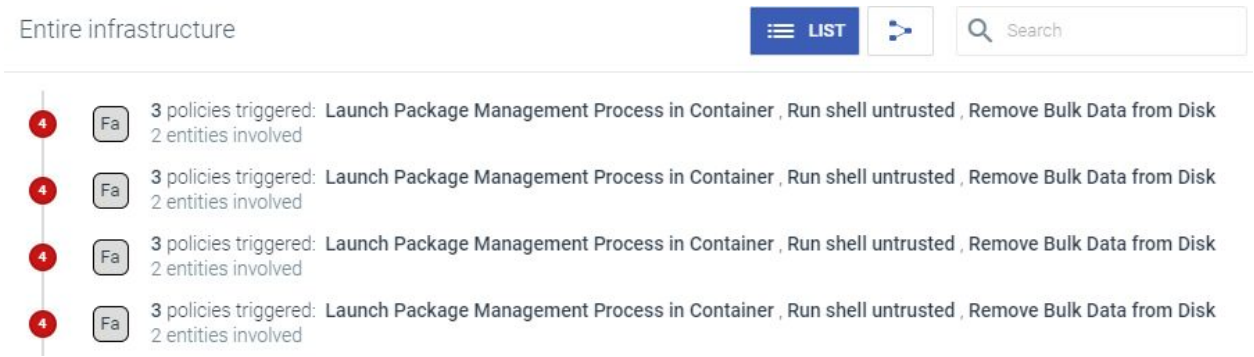
We're going to extend our existing web application here, so deploy 2 new components into the web-app namespace.

```
kubectl apply -f lab7/manifests/ -n web-app
```

Once deployed, you can check the status to make sure everything is up and running. 'Actor' is a cronjob and will not continually run, having a status of 'Completed' is expected.

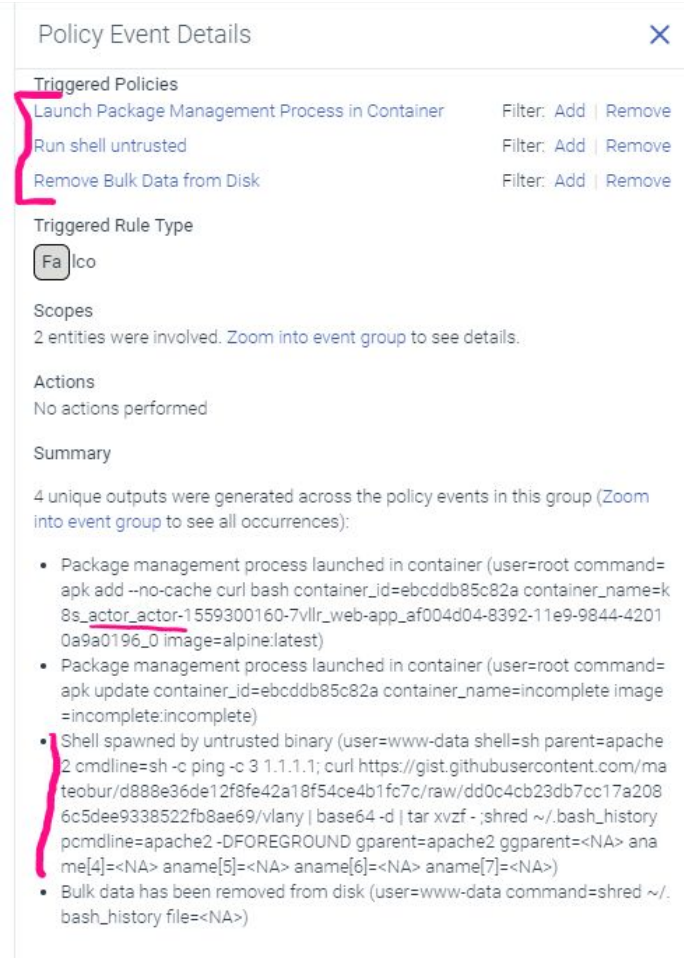
```
wizard@Sysdig:$ kubectl get pods -n web-app
NAME                     READY   STATUS      RESTARTS   AGE
actor-1559300040-25ccv   0/1     Completed   0          57s
nginx-74d54c49c4-b56pz   1/1     Running     0          15h
www-679595d8fd-8vbxw     1/1     Running     0          25m
```

Now in Sysdig Secure UI, have a look at the policy event triggers and see what is happening with the new web service and the actor cronjob. You should see a number of policy violations.

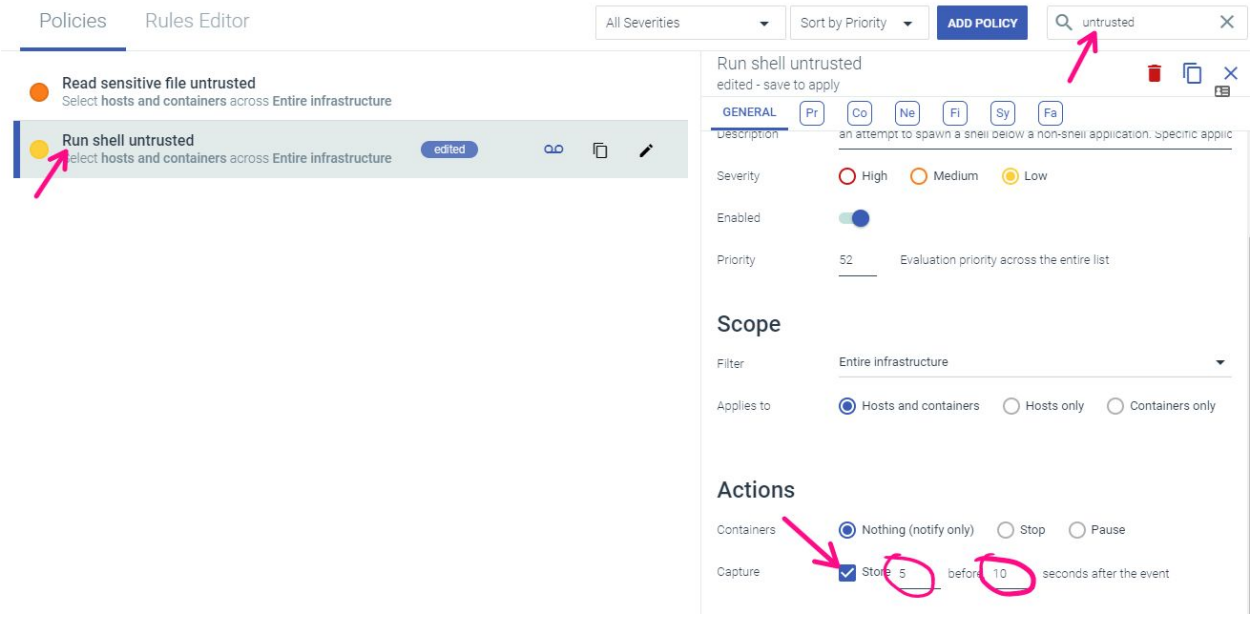


If you click on one of these, you'll get more verbose information. We can see at least 3 policies have been triggered, these are listed at the top of the event details. We can see the

details of which container triggered the ‘Launch Package Management Process’ and we can see the commands run for both the ‘Run shell untrusted’ and the ‘Remove Bulk Data from Disk’. Some fairly suspicious activity, all detected with the default rules!



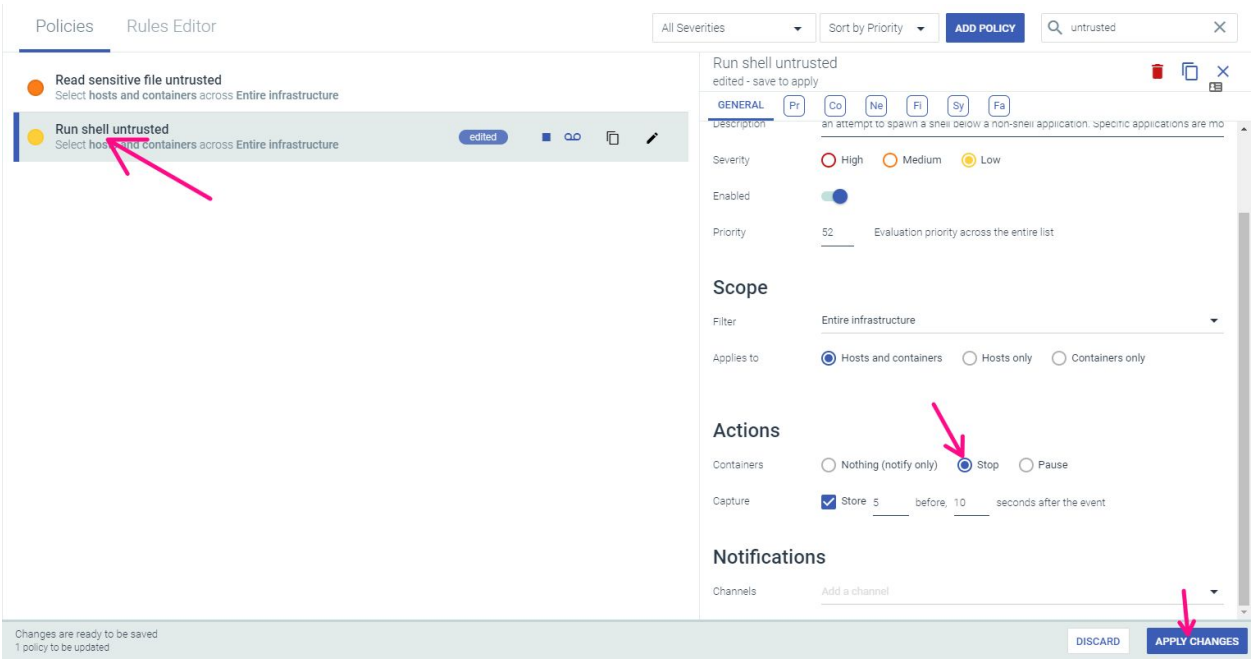
We may want to investigate this further later, so let's modify the ‘Run shell untrusted’ policy to take a Sysdig Capture. You can do this by clicking the policy name in the policy event details page we just opened, or under the Policies page, use the search filter to find any policy with ‘shell’ or ‘untrusted’ (or the full name!). Under ‘Actions’, click ‘Capture’ and leave the default values (5 seconds before, 10 seconds after). This should hopefully be sufficient for investigations, but you can play with these values to see if there is anything else useful. If you want, you can also configure a notification, just be aware this may be noisy (the cronjob runs every minute). Make sure to click ‘Apply Changes’.



There is a good chance this will result in a good number of captures, so after 5-10 minutes, you may want to disable the captures again.

10.4 Preventing command injection

On the Policies page again, once again find the ‘Run shell untrusted’ policy. Under ‘Actions’, click ‘Stop’ and then ‘Apply Changes’.



What you should find is that now your Kubernetes CronJob and web service is having a problem running, both are being restarted each time Kubernetes detects that they have failed (i.e. the Docker container has been killed by the Sysdig Agent).

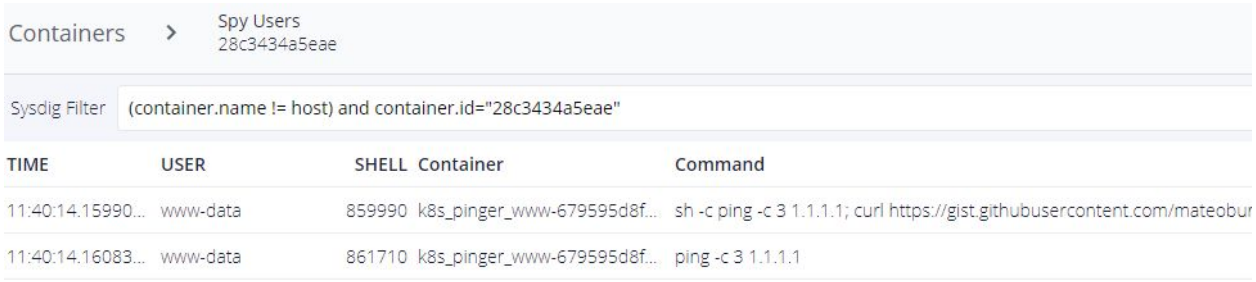
```
wizard@Sysdig:$ kubectl get pods -n web-app
NAME                     READY   STATUS             RESTARTS   AGE
actor-1559302320-c676s   0/1     Completed          0          4m32s
actor-1559302380-q6tg5   0/1     Completed          0          3m32s
actor-1559302440-js8jd   0/1     Completed          0          2m32s
actor-1559302500-dskw7   0/1     CrashLoopBackOff   3          92s
actor-1559302560-csnk9   0/1     Error              2          32s
nginx-dfff5979-pk79c     1/1     Running            0          44m
www-679595d8fd-8vbxw     0/1     Completed          3          113m
```

If the www service was scaled out, then this would have no disruption to service, the exploited container would be killed and then replaced, but overall the service should have sufficient resilience to maintain service. The below example illustrates what this may look like. Only a single pod is offline at any one time, so the overall service remains up.

```
wizard@Sysdig:$ kubectl get pods -n web-app
NAME                     READY   STATUS             RESTARTS   AGE
actor-1559302320-c676s   0/1     Completed          0          7m15s
```

```
actor-1559302380-q6tg5   0/1     Completed          0          6m15s
actor-1559302440-js8jd   0/1     Completed          0          5m15s
actor-1559302500-dskw7   0/1     CrashLoopBackOff   5          4m15s
actor-1559302560-csnk9   0/1     CrashLoopBackOff   4          3m15s
actor-1559302620-pqrg5   0/1     CrashLoopBackOff   4          2m14s
actor-1559302680-v9tjk   0/1     Error              3          74s
actor-1559302740-25zd5   0/1     Error              1          14s
nginx-dfff5979-pk79c     1/1     Running            0          46m
www-679595d8fd-4fgxx     1/1     Running            1          19s
www-679595d8fd-8vbxw     0/1     Completed          5          115m
www-679595d8fd-pzxdc     1/1     Running            0          19s
www-679595d8fd-rp7zj     1/1     Running            0          19s
www-679595d8fd-v6q25     1/1     Running            0          19s
```

If you notice from the policy events, we are no longer triggering the policy 'Remove Bulk Data from Disk'. This shows how quickly the Sysdig Agent reacts and processes the 'Stop' action we told it too, the executed command doesn't have time to complete. We'll open up the captures later, but just to illustrate, here is the commands that were successfully run, notice that as soon as the ping command is attempted, the action to shut down the container is run and no other command is executed, so here we've detected the attack, prevented it, and gathered evidence to fix the application to avoid it happening in the future.

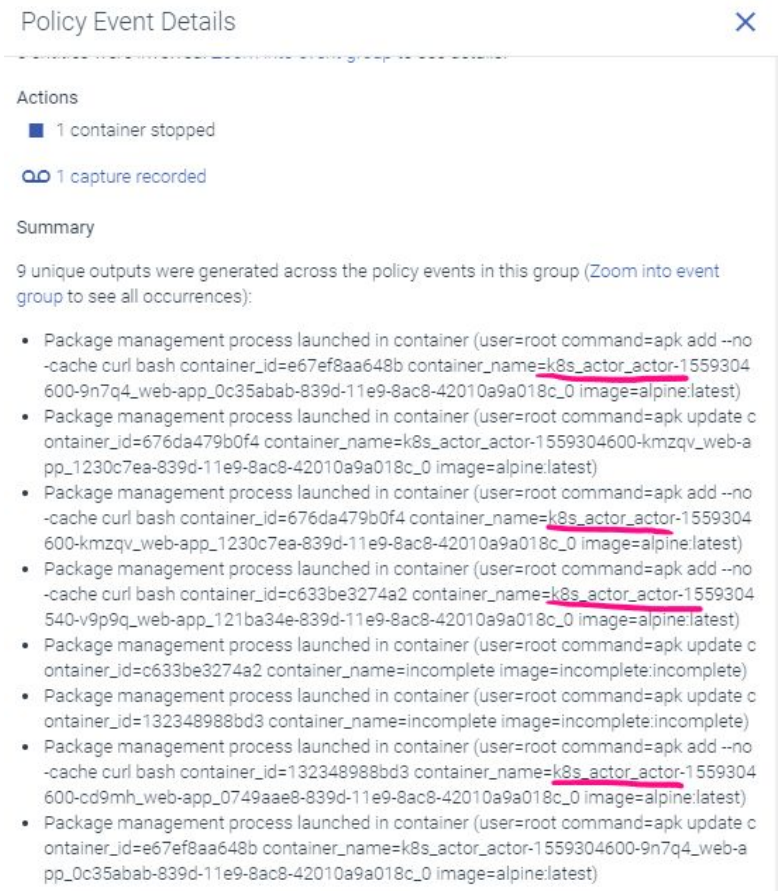


10.5 Blacklist the malicious pod

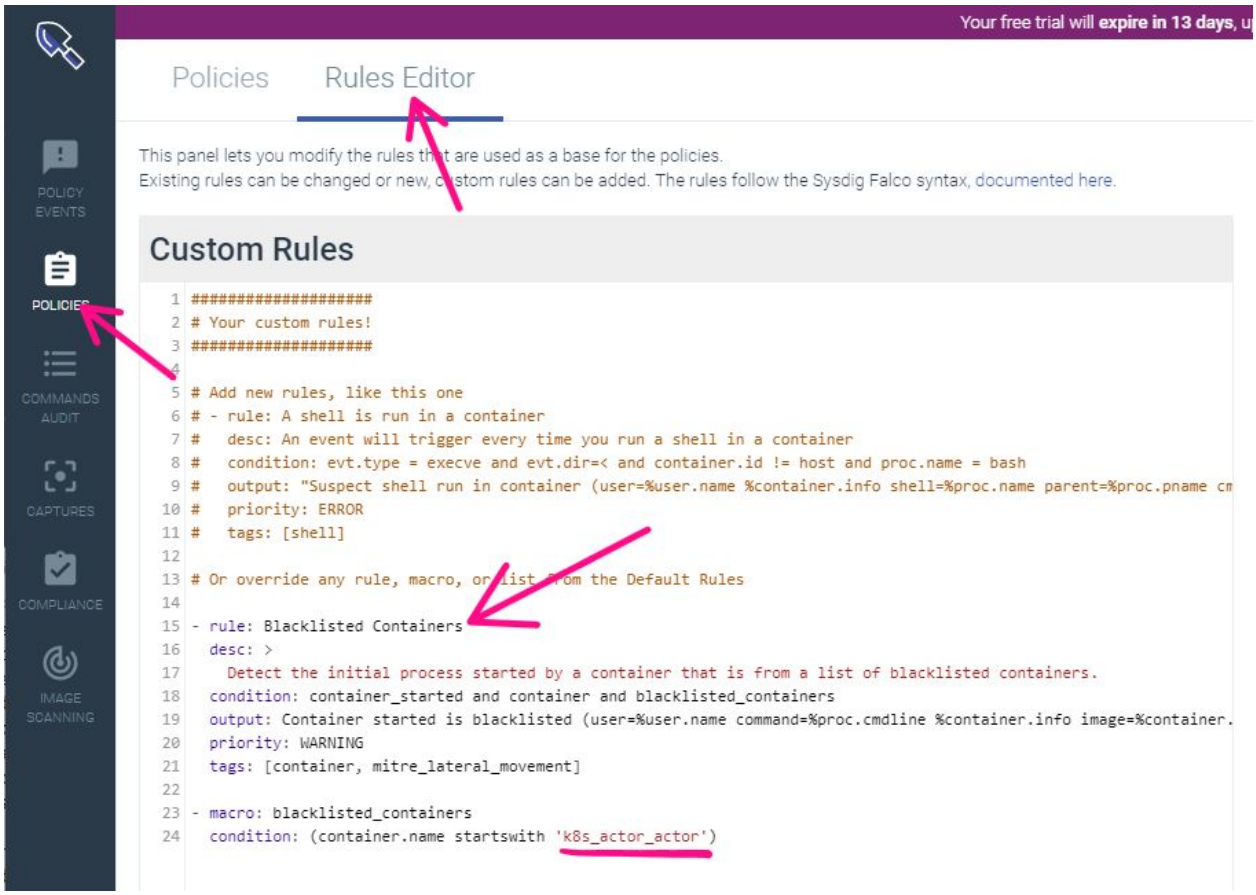
We know we protect ourselves against zero day attacks that surface as gaining shell access, but let's take earlier action against this specific container that is attacking our cluster. Lets blacklist this specific pod from actually starting. To do this we're going to create a new Falco rule and a new macro to match the container.id of the image we detected. Although a default

rule already exists to disallow containers, this is based on a known whitelist, we instead want to do our blocking based on a blacklist, so we'll create a new rule and policy.

First, lets find the container name from within the Sysdig Secure UI. Click on one of the policies, then towards the bottom of the event details page find the container name for the actor pod. In this cluster the container name always starts with 'k8s_actor_actor' so we'll use this as our detection.



Under 'Policies', click on the 'Rules Editor' tab along the top.



We want to copy and paste the following rule and macro into the ‘Custom Rules’ section.

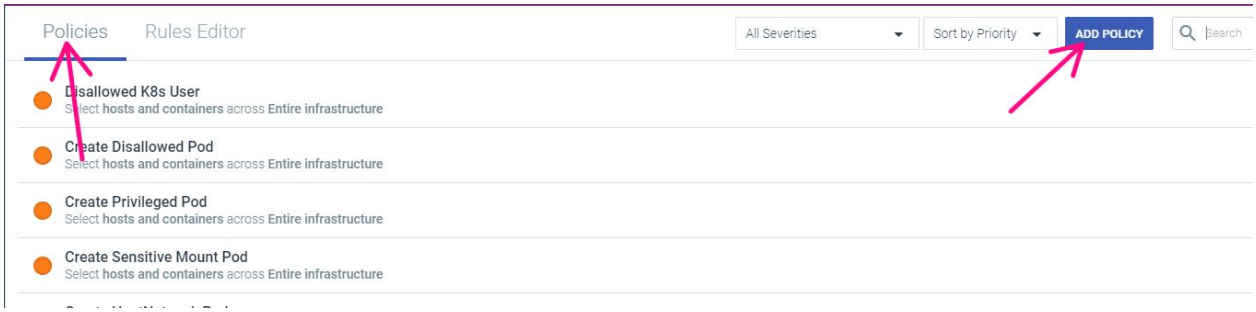
```
- rule: Blacklisted Containers
  desc: >
    Detect the initial process started by a container that is from a list of blacklisted containers.
  condition: container_started and container and blacklisted_containers
  output: Container started is blacklisted (user=%user.name command=%proc.cmdline %container.info image=%container.image.repository:%container.image.tag)
  priority: WARNING
  tags: [container, mitre_lateral_movement]

- macro: blacklisted_containers
  condition: (container.name startswith 'k8s_actor_actor')
```

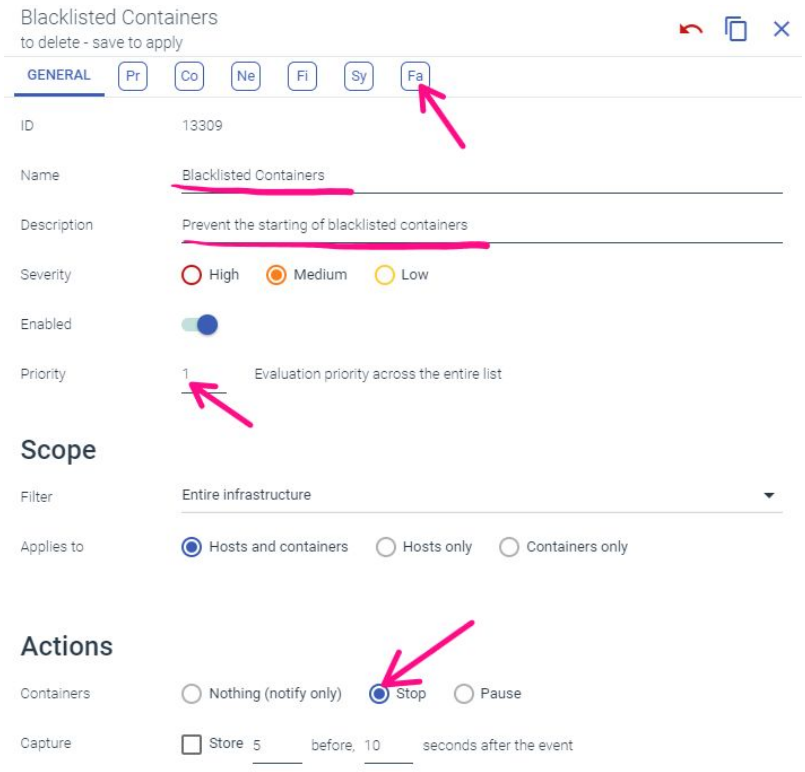
The conditions here detect that a container_started event was detected, that it was indeed a

container, and then we use the ‘blacklisted_containers’ macro as a 3rd check. The ‘blacklisted_containers’ macro simply contains a boolean check for the single container ID we saw earlier. As we want to future proof this, and we may want to be able to check for blacklisted containers using different techniques, a macro will prove more powerful than a list here. Click the ‘Save’ button and then return to the ‘Policies’ tab.

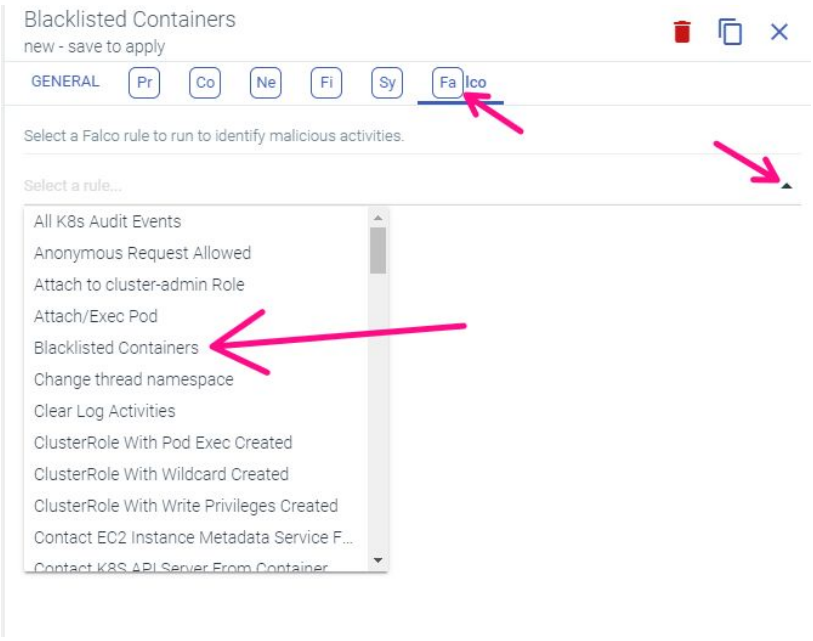
Now we want to create a policy that uses this new Falco rule. Click ‘Add Policy’.



On the first tab, give the new policy a name, description and set a ‘Stop’ action. We want to push this up the priority list, so change the ‘Priority’ to 1. Then click on the ‘Fa’ icon to select the Falco rule we just created.

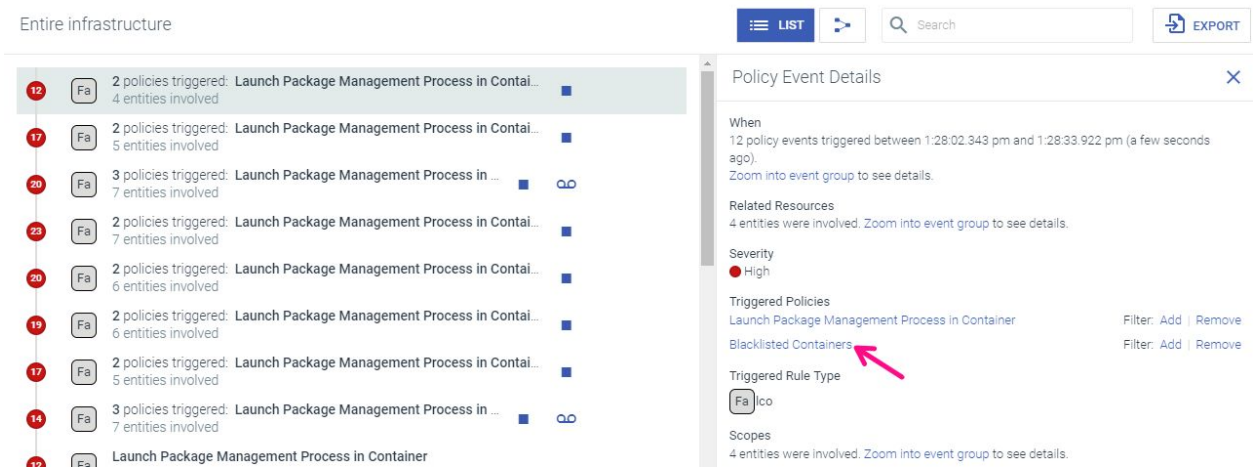


From the Falco rules dropdown, select the ‘Blacklisted Containers’ rule we just created. Finally go back to ‘Policy Events’ and see if we’re now blocking the starting of this container earlier in its lifecycle (it may take a minute or two for the cronjob to restart and trigger the policy).



Now finally click ‘Apply Changes’.

Having a look a few minutes later from the ‘Policy Events’ page we can see that we are now generally preventing the container from starting and no longer getting to the shell being executed.



Have a play with different methods of detecting and blacklisting containers.

10.6 Cleaning up

In this lab we created a noisy and malicious cronjob that is triggered a number of security policies, which in turn is creating a lot of noise in the Sysdig interface. We could simply whitelist this application, but as we now know it is malicious, it makes more sense to remove the cronjob. Simply run the following to remove the job.

```
wizard@Sysdig:$ kubectl delete cronjob actor -n web-app
```

10.7 Homework

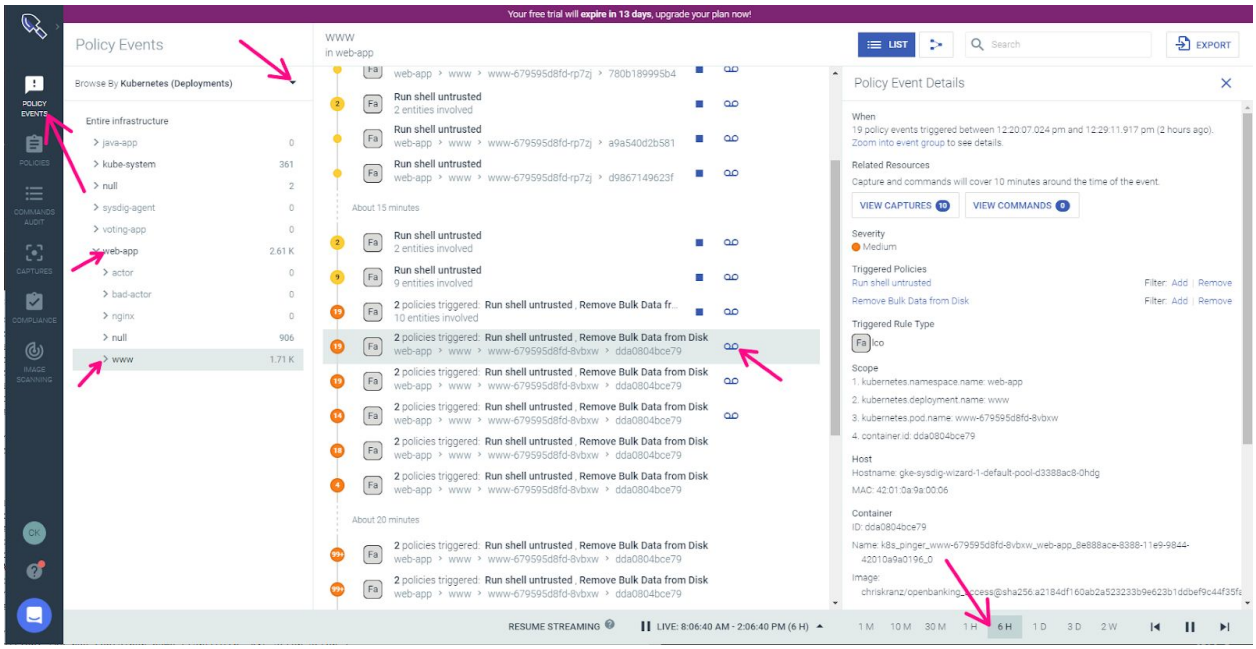
If you have time, or as homework, investigate this application in more detail, try to create a rule that will block this application earlier on without danger of preventing regular activity or applications. Investigate kube-audit policies to see if you can prevent this sort of pod being launched in the first place.

11 Lab 8: Forensic Troubleshooting

11.1 Investigating a Capture

In the last lab we created some very noisy captures and events, so first thing now is to drill in to find a specific capture that is going to be useful. The captures created in this lab are going to be noisy, so we will need to do some discovery and work around the data set.

In the 'Policy Events' page, make sure the grouping is set to 'Kubernetes (Deployments)', then browse to find web-app > www and click on this. If this displays a 0, move the time control at the bottom of the page to discover some events for this application. Find one of the early events created that has a capture (like a voicemail icon).

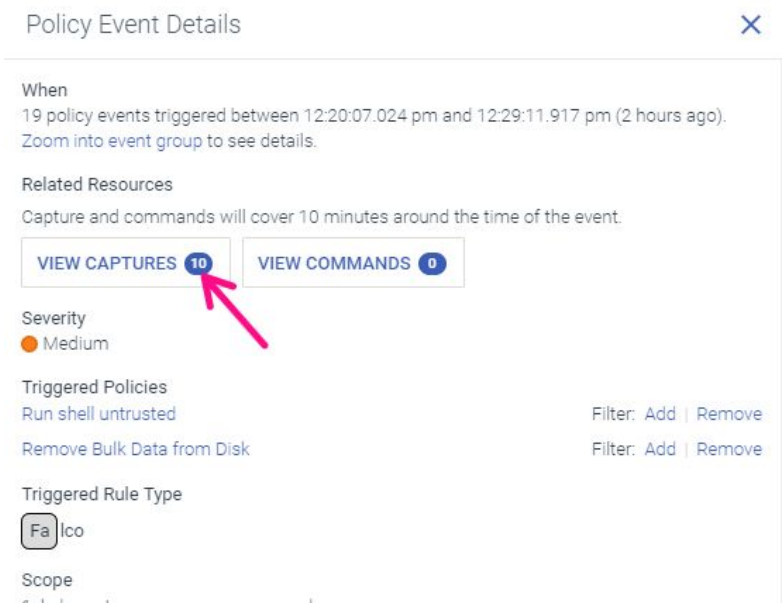


From here you can discover a little more about this particular event and the scenario around the capture. This will be useful information to note down first. In this scenario, I've made the following notes:

- Policy Triggers: Run shell untrusted & Remove Bulk Data from Disk
- Kubernetes Pod Name: www-679595d8fd-8vbxw

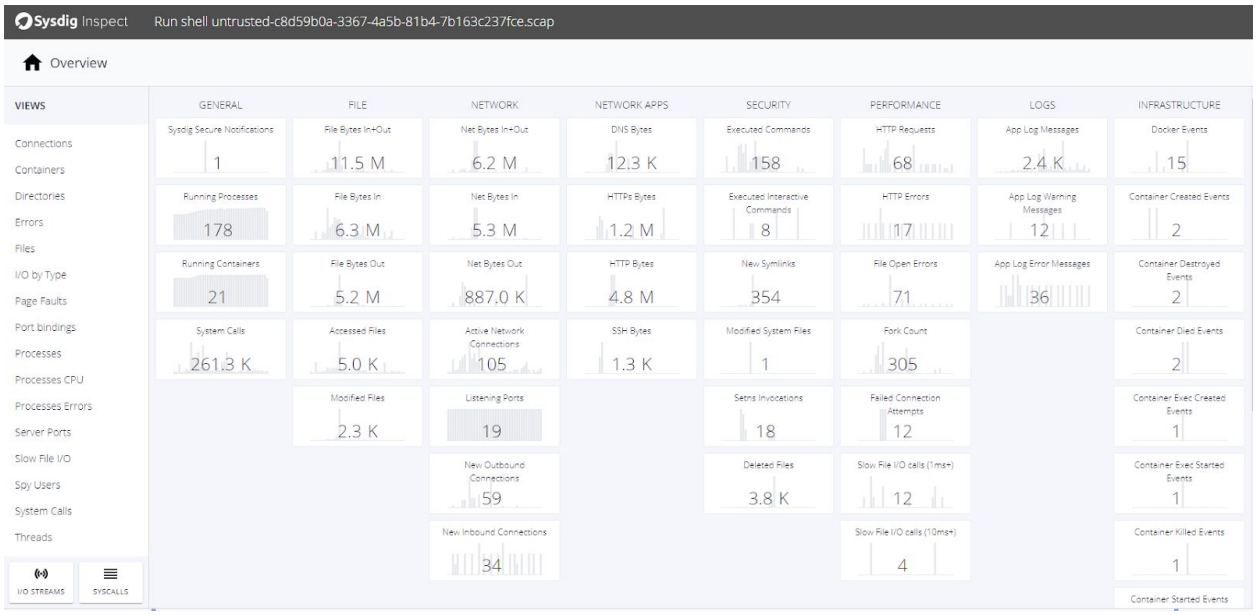
- Container Name: k8s_pinger_www-679595d8fd-8vbxw_web-app_8e888ace-8388-11e9-9844-42010a9a0196_0
- Container ID: dda0804bce79
- Summary info:
    - Shell spawned by untrusted binary (user=www-data shell=sh parent=apache2 cmdline=sh -c ping -c 3 1.1.1.1;
    - Bulk data has been removed from disk (user=www-data command=shred ~/.bash_history file

Now click 'View Captures', there may be a number of them.

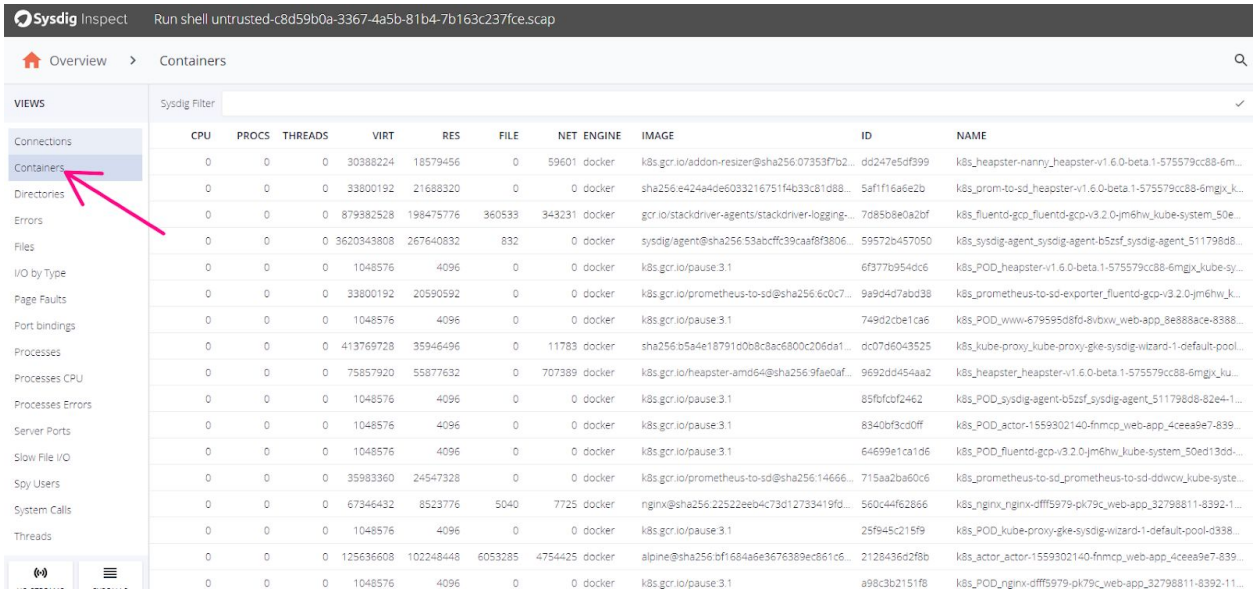


And we'll get a list of all the captures that relate to this policy event. Multiple events have been combined here as they are all the same, happening at a very similar time. Choose any of the captures by mousing over and then clicking the Sysdig Inspect spade icon.

This will open up Sysdig Inspect and allow you to start troubleshooting this particular event to discover exactly what happened. There is a lot of information here, so we want to understand how to filter and narrow down our search to relevant information.

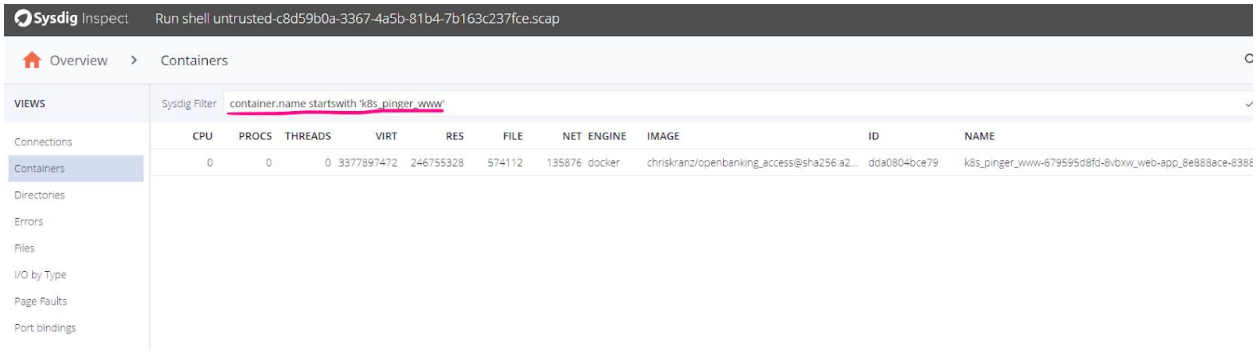


First, lets find the container we identified as being the target for this event trigger. Click ‘Containers’ on the left hand side to get a list of all containers running on this host.

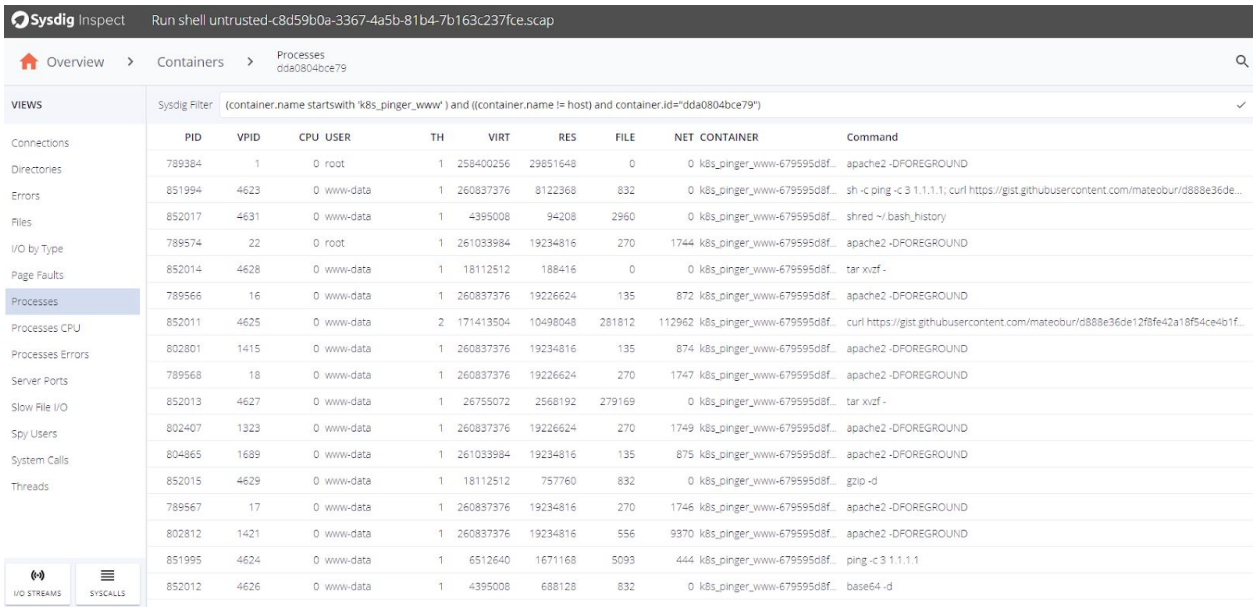


That's still a big list of containers, so use the search field in the top bar to narrow down our search. From our notes we're looking for a container that starts with 'k8s_pinger_www', so lets use that in the search field.

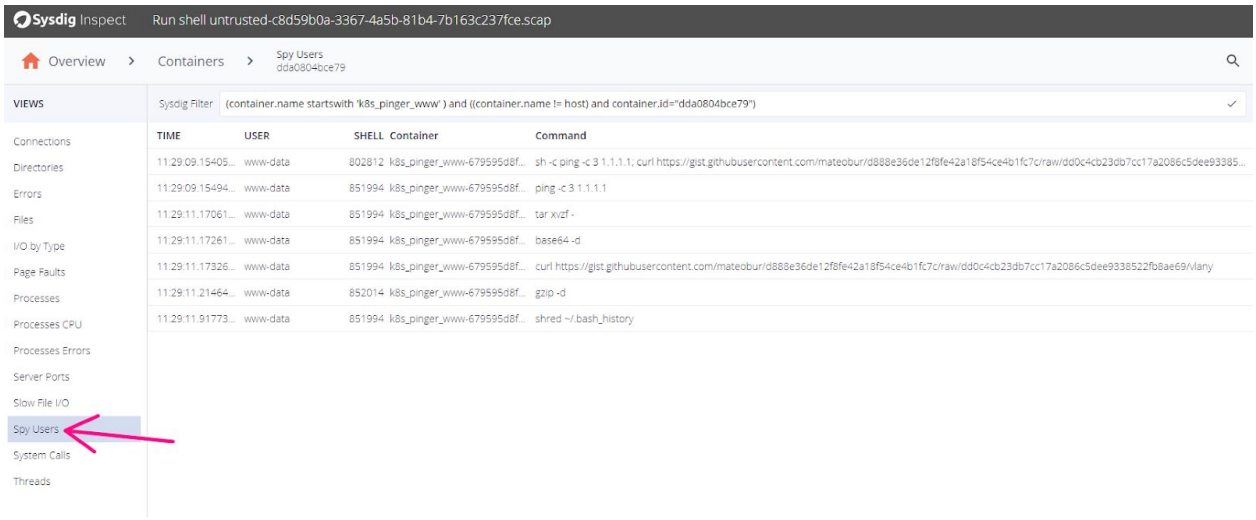
Sysdig Filter: container.name startswith 'k8s_pinger_www'



If we double click on this container, we'll set the search scope to just this single container and we can start exploring what this one container was doing. So we've quickly filtered out the noise. It'll default to showing us the processes. For an apache server, this is doing a lot more than just running apache! Immediately you can probably spot a few questionable commands and activity.

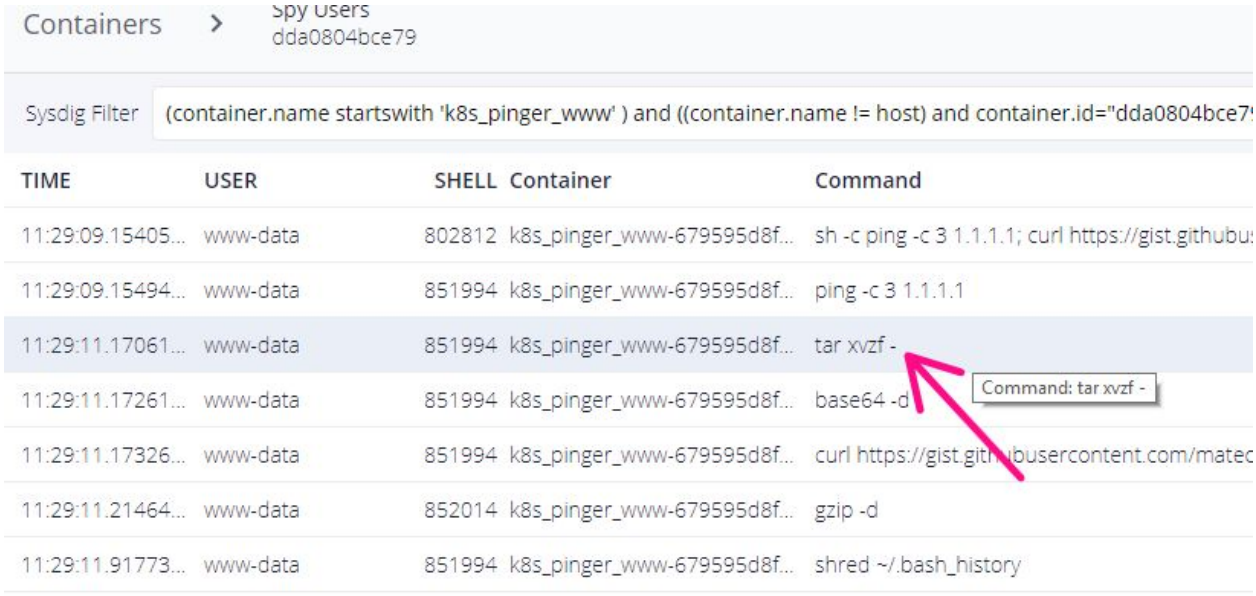


Lets keep investigating. We know this was triggered by 'Run shell untrusted', so lets see what commands were run within the shell. Click on 'Spy Users' on the left hand side. Notice that our Sysdig filter has stuck and we're still just seeing this single container that matches both the container name, and container ID, and is not running directly on the host.

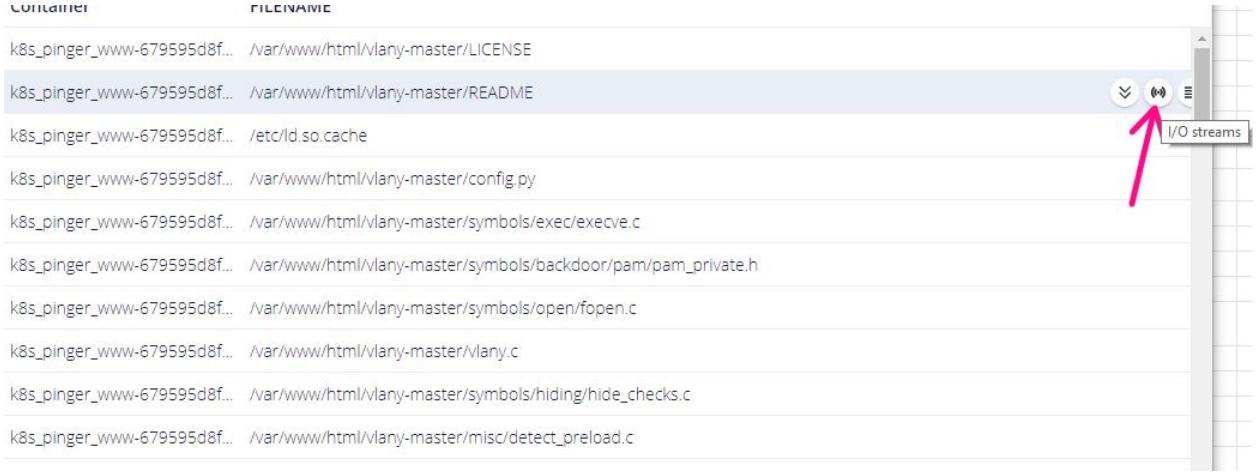


Now we have a clear picture of what was executed, including the shred ~/.bash_history which is triggering the second policy 'Remove Bulk Data from Disk'. But this doesn't tell us enough yet, we could always go and grab these files from the internet, but then we'll need an isolated host to unpack and examine these. Continue here for now to see if we can discover

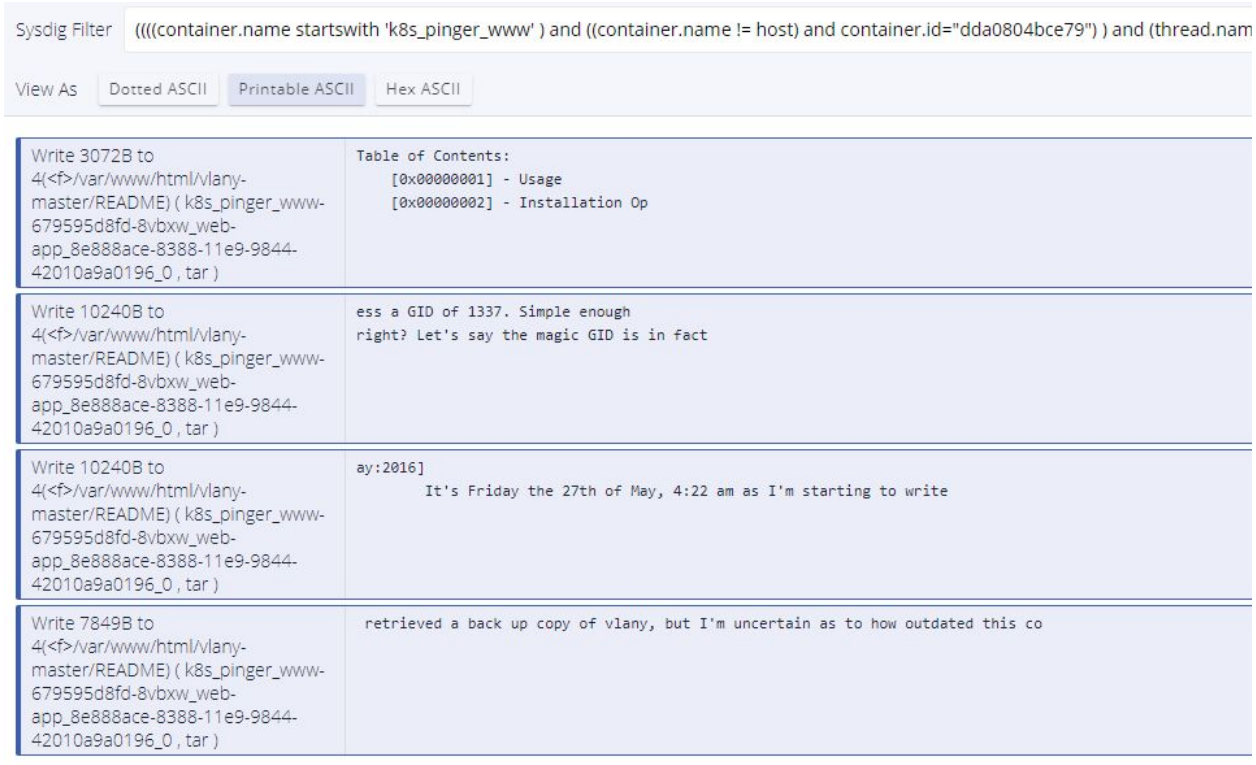
more. One of the commands was extracting a tarball, double click on this, and then when the page loads double click on the tarball command again. The way the tarball command runs is it's effectively a command within a command, so we drill down 2 levels before discovering the next set of information.



Now we can see every single file that was extracted as part of this tarball, which is useful! We can probably take a good guess now, Google 'vlany' or identifying filenames. But again, lets try digging deeper. On any of the files shown, if you mouse-over you get a few actions on the right hand side. One of these is 'I/O Streams'. Choose any of the files that looks like it might be interesting.

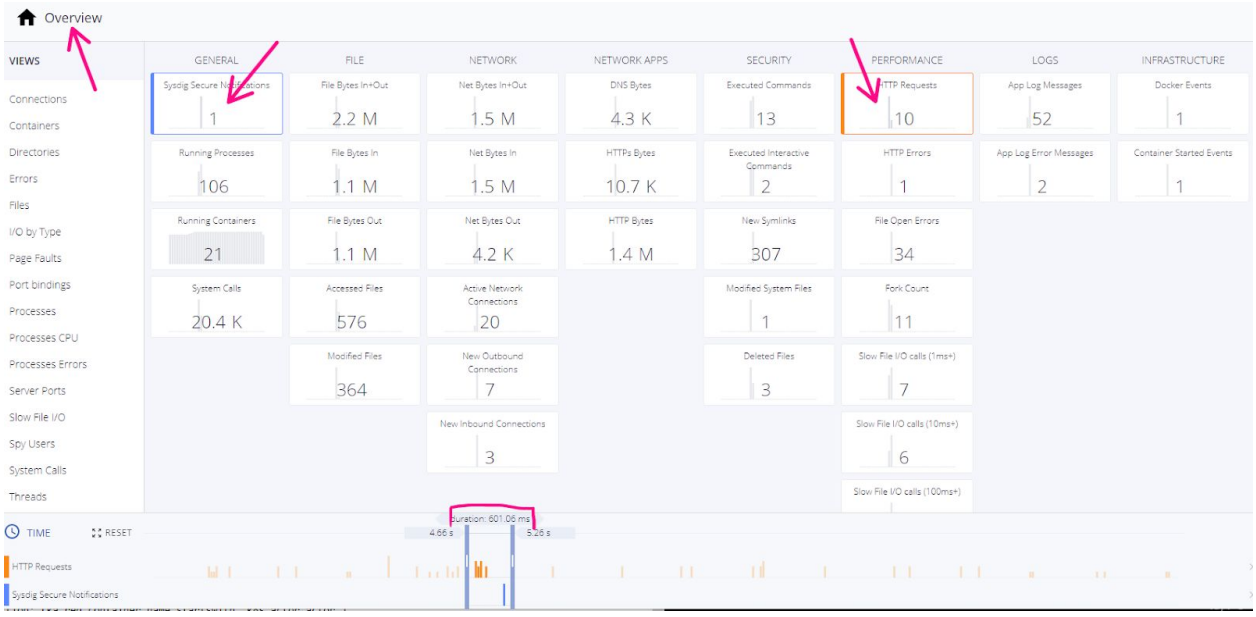


Once the I/O stream loads, you'll want to switch to 'Printable ASCII' so it's formatted a little easier to read. Now we can see snippets of the file contents. Although it isn't the entire file contents (file sizes would be difficult to manage), it is hopefully enough to continue the investigation.

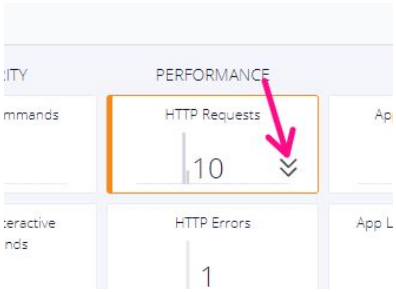


You can use the breadcrumbs at the top to go back to the ‘Files’ view so you can open a different file. As you investigate (and with the help of Googling ‘vlany’, you’ll hopefully discover this is an LD_PRELOAD rootkit.

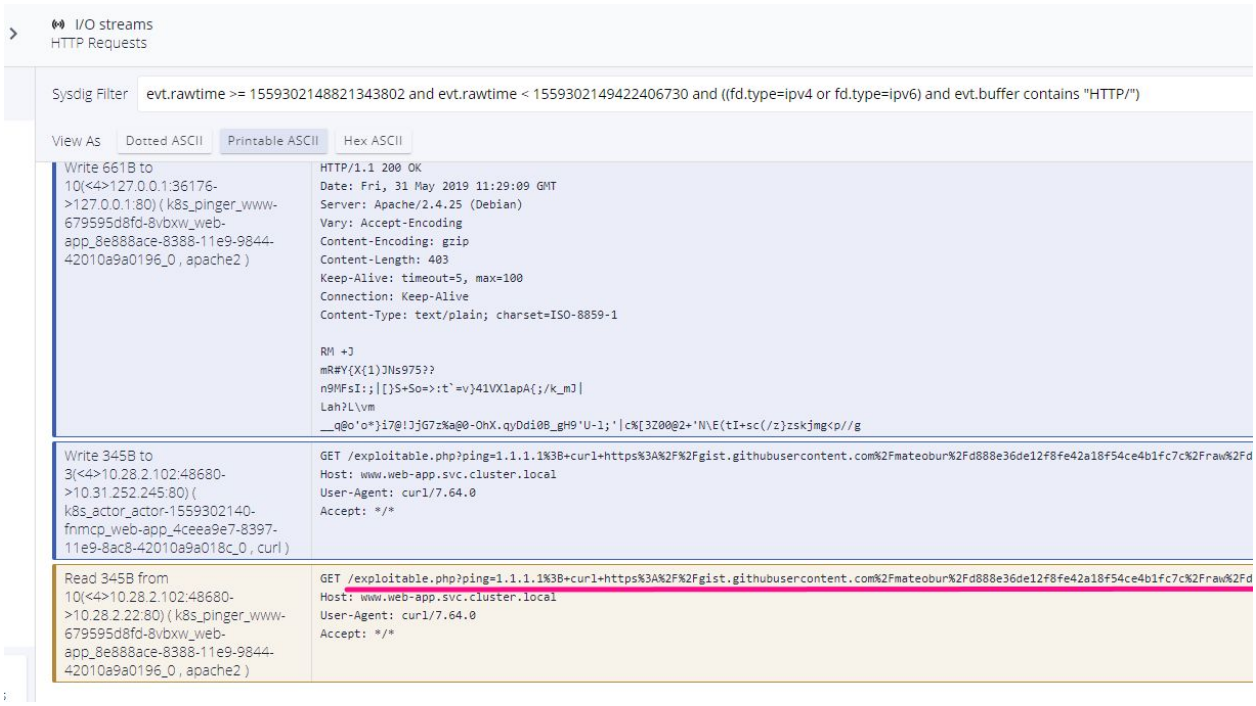
But still the question remains, how did they get shell access? Go back to Overview (top left), this clears all our filters / searches. Now single click on the ‘Sysdig Secure Notifications’ and ‘HTTP Requests’ panels. This will overlay activity at the bottom. From here we can either drag in the left and right time markers, or click and hold to drag across a specific time period. Highlight the few HTTP requests immediately before the Sysdig Secure Notification was triggered.



This has narrowed down the information that’s being displayed greatly, so now lets have a look at the 10 (or so) HTTP requests made just before the policy event was triggered. Hover over ‘HTTP Requests’ and click the drill down icon.



The page should still be on ‘Printable ASCII’ from setting this previously, but if it isn’t please click this button to make the output more readable. Scrolling down there’s a lot of fairly standard traffic, we can see outbound traffic getting various alpine modules (this will be what is triggering the ‘Launch Package Management Process in Container’ policy!). Right at the bottom of this example capture, I can see the exploit being executed. A command injection into a form that hasn’t got the input adequately filtered.



In this scenario we see the read and the write because the ‘actor’ cronjob that is executing the exploit is running on the same host as the service that is being exploited. This wouldn’t usually happen, but it does show that the capture is recording both incoming and outgoing

traffic, so you can also detect if you are being used as for any malicious activity to other victims external to you.

Appendix

Further Reading - Sysdig Platform

When you first log in to your Sysdig UI, you'll be presented with a brief interactive tour to show you the main areas. This document points to a few of our documentation pages to help guide you to our reference material. Our user documentation is already ordered to help guide you through the platform, so it is provided simply as a reference here so you always have access to the latest documentation.

https://sysdigdocs.atlassian.net/wiki/spaces/Platform/overview

Sysdig Agent

https://sysdigdocs.atlassian.net/wiki/spaces/Platform/pages/204734652/Agent

Sysdig Monitor

https://app.sysdigcloud.com

Getting Started

https://sysdigdocs.atlassian.net/wiki/spaces/Monitor/pages/199721030/Getting+Started

Sysdig Secure

https://secure.sysdig.com

Getting Started

https://sysdigdocs.atlassian.net/wiki/spaces/Secure/pages/261881870/Getting+Started

Troubleshooting

We recommend visiting the Sysdig Support site (https://sysdig.com/support/) for a comprehensive set of Sysdig Platform documentation and troubleshooting guides.

Sysdig Common Use-Cases

Kubernetes Monitoring and Troubleshooting

https://sysdig.com/use-cases/kubernetes-troubleshooting/

Microservices Monitoring

https://sysdig.com/use-cases/microservices-monitoring/

Prometheus Monitoring

https://sysdig.com/use-cases/prometheus-monitoring/

Continuous Security

https://sysdig.com/use-cases/continuous-security/

Container Forensics

https://sysdig.com/use-cases/container-forensics/

Glossary of Terms

Term	Meaning
Kernel	The kernel is a computer program that is the core of a computer's operating system, with complete control over everything in the system. On most systems, it is one of the first programs loaded on start-up. It handles input/output requests from software, translating them into data-processing instructions for the central processing unit.
System Call / Syscall	Syscalls are the input/output requests made from software to the kernel
Tracepoint / Trace	A tracepoint provides a hook to call a function (probe) that you can provide at runtime. Sysdig uses traces and tracepoints as part of the measurement process for collecting low-level metrics.
Container	A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another.
Container Image	A container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.
Orchestrator	In modern development, applications are no longer monolithic, but instead are composed of dozens or hundreds of loosely coupled, containerised components that need to work together to allow a given app to function as designed. Container orchestration refers to the process of organising the work of individual components and application layers.
Metric / Telemetry	This is data about performance and events, i.e. CPU load, disk I/O, application response times, and so on. This is generally time-series coded data, i.e. CPU load at 8:47am on 1/1/2019m average disk I/O between 7:00-8:00 on 1/1/2019, and so on.
Instrumentation	This is configuration changes need to enable a functionality. This could mean giving credentials to be able to log in, or it could be making changes to code in order to support external tracepoints.
Inspect / Capture	This is a Sysdig technology used to create a forensic copy of all syscalls and the system state of a host at a particular point in time.

Reference Reading

Sysdig Resources

Document	URL
Sysdig Monitor Product Page	https://sysdig.com/products/monitor/
Sysdig Secure Product Page	https://sysdig.com/products/secure/
Sysdig Platform Product Page	https://sysdig.com/platform/
Sysdig Falco Product Page	https://sysdig.com/opensource/falco/
Sysdig & Inspect Product Page	https://sysdig.com/opensource/inspect/
Teachables Secure 101	https://sysdig.teachable.com/p/sysdig-secure-101
Teachables Monitor 101	https://sysdig.teachable.com/p/sysdig-101
Recorded Webinar Library	https://www.brighttalk.com/channel/16287/sysdig
Vulnerability Management	https://go.sysdig.com/vulnerability-management
Overview of Sysdig Inspect	https://sysdig.com/blog/sysdig-inspect/
User Guides	https://support.sysdig.com/hc/en-us
Sysdig Metric & Notification Integrations	https://support.sysdig.com/hc/en-us/sections/201820476-Integrations
Sysdig Custom Notifications	https://sysdig.com/blog/kubernetes-docker-elasticsearch-splunk/
Sysdig Release Notes	https://sysdigdocs.atlassian.net/wiki/spaces/SRN/overview

Other Useful Resources

Document	URL
Google's Site Reliability Engineering	https://landing.google.com/sre/book.html
Google's Golden Signals	https://landing.google.com/sre/sre-book/chapters/monitoring-distributed-systems/
USE Method Monitoring	http://www.brendangregg.com/usemethod.html
A Philosophy on Alerting	https://docs.google.com/document/d/199PqyG3UsyXlwieHaqbGiWVa8eMWi8zzAn0YfcApr8Q/edit

Sysdig