signal fx

DOCKER + SIGNALFX INTEGRATION

What is Docker?

Docker is an open-source technology that packages an application with all its dependencies into software containers. It provides code, runtime, system tools, and system libraries to be directly managed by the kernel, thereby allowing software to run as microservices in seemingly independent, lightweight systems, on top of the host operating system. Docker is used to build, test, run, and deploy applications quickly, reliably, and consistently regardless of underlying environment—cloud, VM, or bare metal. Docker helps make workloads more portable and flexible, which is particularly useful in a cloud environment.

Sending Docker Metrics to SignalFx

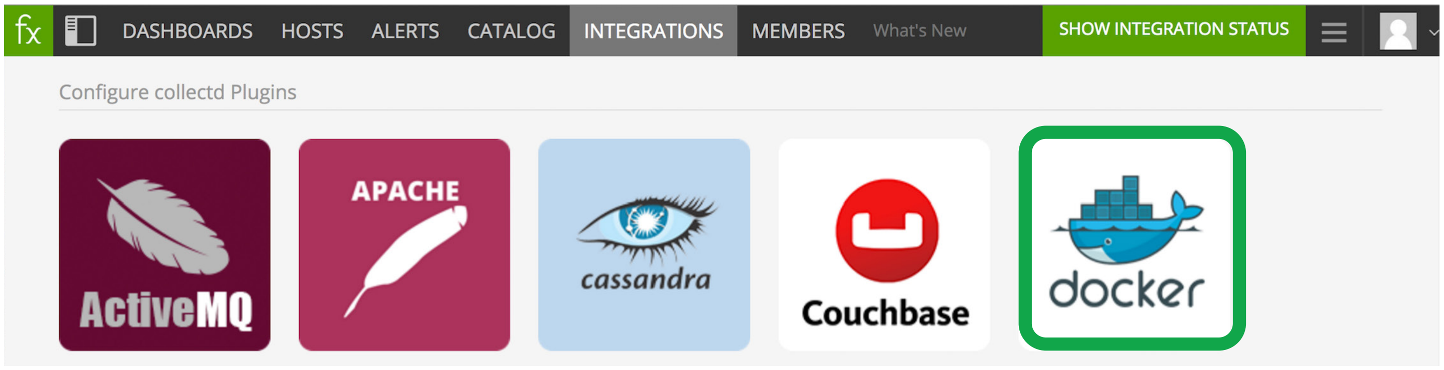
Use collectd with the collectd-docker plugin to capture metrics about the Docker containers running on the system using Docker's stats API. It reports container metrics about CPU utilization, memory consumption, and network and disk activity. SignalFx provides built-in dashboards to display the most useful metrics for running Docker containers in production. You can also add dimensions to the metadata to easily aggregate, filter, and group metrics by any properties you choose.

Monitoring Docker

The adoption of Docker containers and the associated desire to build microservices has changed the requirements of monitoring tools. There are three sources of challenges associated with monitoring Docker containers in production environments:

- Working with various cluster managers
- Managing containers at scale
- Configuring monitoring agents for service discovery

OPTIONALITY FOR CLUSTER MANAGERS: Since the commercial launch of Docker, the ecosystem of operational tools has rapidly evolved. The proliferation of cluster manager tools—including Marathon, Kubernetes, Aurora, and Swarm—gives customers a variety of choices to orchestrate, cluster, and manage containers based on the specific needs of their environment or application. However, this optionality of various cluster managers implies that operations and development teams must be aware of the challenges associated with each of the underlying components.



COMPLEXITIES AT SCALE: The lightweight nature of Docker containers allows for ease of rapid deployment, but an increased number of datapoints to measure and increased velocity of change within a containerized environments creates complexity. The ephemeral, dynamic nature of containers only exacerbates the complexities of monitoring. Containers make it easy to reverse updates in isolation without affecting the rest of the environment, but this means the life of a container could last as little as hours or even minutes. With containers moving in and out of the environment at any given moment, it becomes increasingly difficult to track and monitor all containers in production.

Docker Metrics

Asynchronous Block I/O Volume	CPU Usage Total
Read Volume from Block Devices	User CPU Usage
Synchronous Block I/O Volume	Memory Limit
Total Block I/O Volume	Maximum Memory Usage
Write Volume to Block Devices	Memory Usage
Asynchronous Block I/O Requests	Network Bytes Received
Read Requests from Block Devices	Outbound Network Packets Dropped
Synchronous Block I/O Requests	Network Reception Errors
Write Requests to Block Devices	Network Packets Received
Block I/O Volume	Network Transmission Errors
Per-Core CPU Usage	Outbound Network Packets Dropped
Kernel CPU Usage	Network Bytes Sent
System CPU Usage	Network Packets Sent

NO SET SERVICE DISCOVERY OR REGISTRATION: Service discovery allows Docker containers to adapt to their current environment for scalable and flexible deployment. Without requiring administrator intervention, containers can find connection information for components they must interact with and register themselves to tools that are available.

However, monitoring in this type of environment requires an upfront decision on the configuration of how monitoring agents are deployed—either inside the container or alongside in a separate container. Configuring a monitoring agent within the container is straightforward, but requires operations teams to manage two distinct processes and create policies for the process lifecycle. On the other hand, creating an independent and privileged container for the purpose of hosting the monitoring agent allows the Docker image to consist of only the microservice application, but requires orchestrating a persistent link between the application and monitoring container.

Monitoring Docker with SignalFx

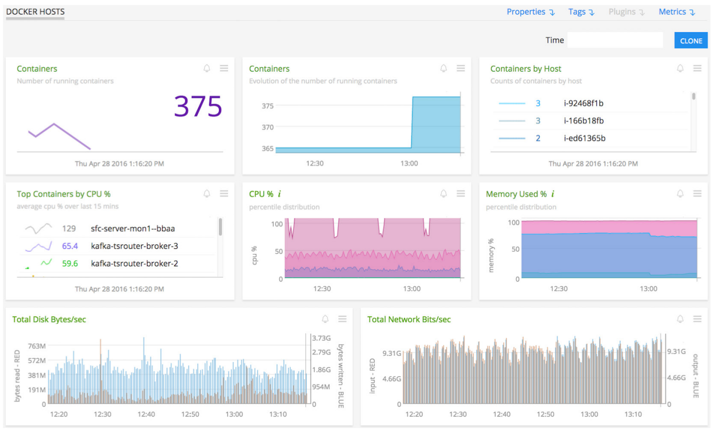
MONITORING AT ALL LEVELS: When it comes to operating Docker environments at scale, understanding performance relies on dependencies among the various layers within the container and the larger architecture. With SignalFx, you can monitor from host to container to microservice application in a single dashboard. Instant visibility across all layers of the Docker infrastructure gives you both the flexibility to gain a service-wide view of performance and the power to explore individual details.

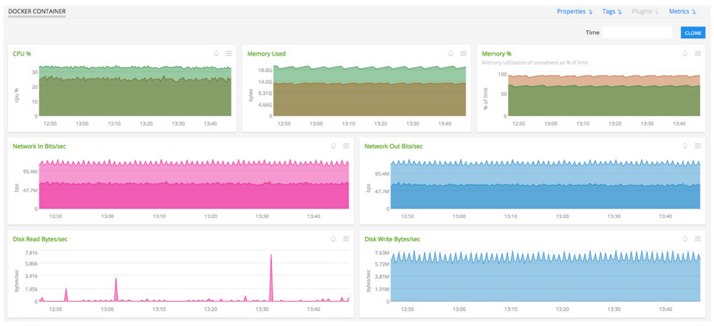
In SignalFx, start with an aggregated view of Docker containers across all hosts in the environment, containers per host, and aggregate infrastructure metrics like CPU utilization in percentile. Easily drill down into a Docker host to view infrastructure metrics broken down by container, and compare performance of the individual host against the population. Correlate performance metrics down to the specific container, and evaluate whether the application is impacted at the service level.

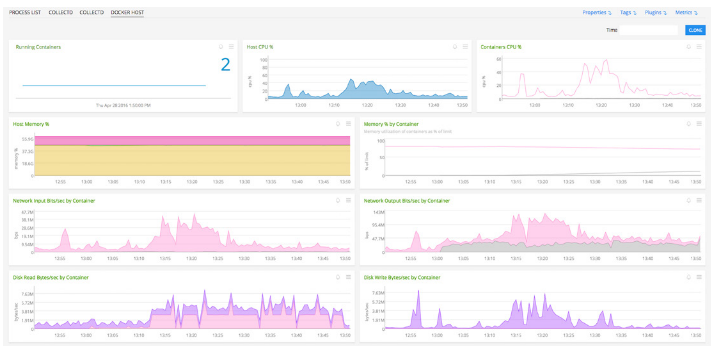
AUTOMATIC CONFIGURATION: Monitoring Docker requires real-time insight into the hosts, containers, and microservices that make up your environment even as the population changes. Visibility into any containerized environment relies on the endurance of your monitoring configurations without constant maintenance and interruption. With SignalFx, configuration is automatic as your Docker infrastructure evolves. All your charts, dashboards, and alert detectors adapt to service membership and instantly reflect the current state of your environment.

For example, new containers entering the environment are automatically picked up without any manual intervention. Docker hosts with the collectd plugin will report any new containers to SignalFx without any of the additional configuration that was typical of traditional on-prem time series databases. Metrics flowing from an application in the container with the collectd plugin installed—or privileged containers set up to access the new application instance as part of the build configuration—will also automatically be captured and reflected.

INSTANT VISIBILITY: SignalFx provides out-of-the-box insights across the Docker metrics that matter. Built-in dashboards for Docker provide a running start to monitoring your modern environment. SignalFx also curates data from the other applications and cloud services in your environment enabling you to embed your own best practices for monitoring and alerting across services important to your specific use case.






PROCESS LIST
COLLECTD
COLLECTD
DOCKER HOST
Running Containers
2
Host CPU %
Containers CPU %
Host Memory %
Memory % by Container
Network Input Bits/sec by Container
Network Output Bits/sec by Container
Disk Read Bytes/sec by Container
Disk Write Bytes/sec by Container

About SignalFx

SignalFx is the most advanced monitoring solution for modern infrastructure. Our mission is to help cloud-ready organizations drive high levels of availability in today's elastic, agile, distributed environments. With SignalFx, development and operations teams gain a real-time view of, interact with, and take action on the infrastructure and application metrics that matter. We have enterprise customers including Yelp, Cisco, Zuora, and Hubspot and thousands of users analyzing billions of metrics every day. SignalFx was founded in 2013 by former Facebook and VMware executives, launched in 2015, and is backed by Andreessen Horowitz and Charles River Ventures.