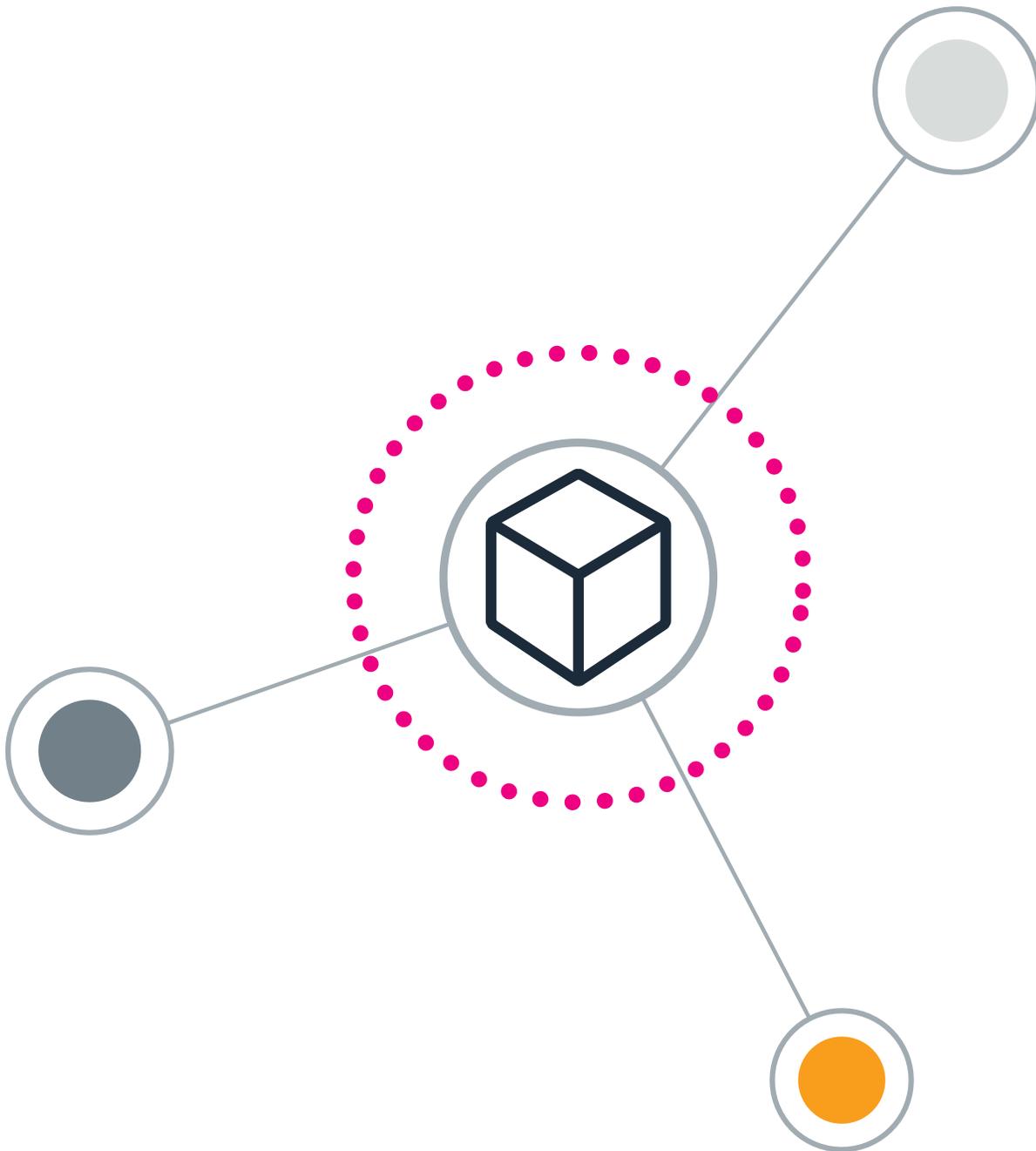


# Container Monitoring and Observability



## Introduction

With more than 50 percent of the world now connected to the Internet via mobile phones, developing new applications has become a core requirement for any digital business. The advent of this core requirement has brought about a paradigm shift in the infrastructure space. As a part of their cloud journey, companies are adopting cloud-native technologies to increase the speed of developing these new applications. A survey conducted by the Cloud Native Computing Foundation shows that the biggest benefit of adopting cloud-native technologies is faster deployment time. This was followed by improved scalability, cloud portability and improved availability.

Containers have become one of the most popular cloud-native infrastructure components to adopt, allowing for faster development and testing of new applications. Alongside container technologies like Docker, DevOps teams are also leveraging container orchestration technologies such as Kubernetes to manage and operate containers at scale.

This whitepaper highlights the role of monitoring on Amazon Web Services (AWS) container environments and key considerations for effectively monitoring those container environments. It also examines factors in deciding which AWS container service would be best suited for microservices-oriented applications. And finally, it lays out the importance of a monitoring and observability strategy for containers.

## Splunk and AWS: Transforming real-time enterprise monitoring of containerized applications

Splunk observability solutions include Splunk Infrastructure Monitoring, Splunk Microservices APM, and Splunk Cloud. Splunk Solutions works across your AWS environment to give you:

- Real real-time (1-second) metrics and monitoring for your entire environment including infrastructure, applications, microservices, containers, and AWS Lambda
- Insight to help you optimize the cost and capacity of your AWS environment
- Direct interoperability with AWS services, including pre-built dashboards
- Consolidated visibility and monitoring for everyone across your organization

Splunk makes it easier for DevOps and SRE teams to monitor and manage container environments and optimize resources and costs on premises or on the cloud. Customers using both containers and functions (serverless) on AWS can have an end-to-end real-time monitoring solution for their entire environment. Besides monitoring, Splunk Infrastructure Monitoring also offers a cost optimizer tool to gain direct visibility into the cost and allocation of resources for both reserved and on-demand Amazon Elastic Cloud Compute (Amazon EC2) instances.

Splunk has been helping customers with complex container environments comprising of applications running in Docker on Amazon EC2 instances managed by Amazon Elastic Container Service (Amazon ECS), stateful applications such as databases running in Docker on EC2 instances without schedulers, stateless applications running on ECS, and streaming applications such as Apache Spark running on Mesosphere (now D2iQ). Besides Amazon Elastic Kubernetes Service (Amazon EKS) and AWS Fargate, Splunk natively supports Amazon EC2, Amazon ECS along with more than 200 out-of-the-box integrations for technologies and applications across the cloud landscape.

## Containers 101

### What are containers

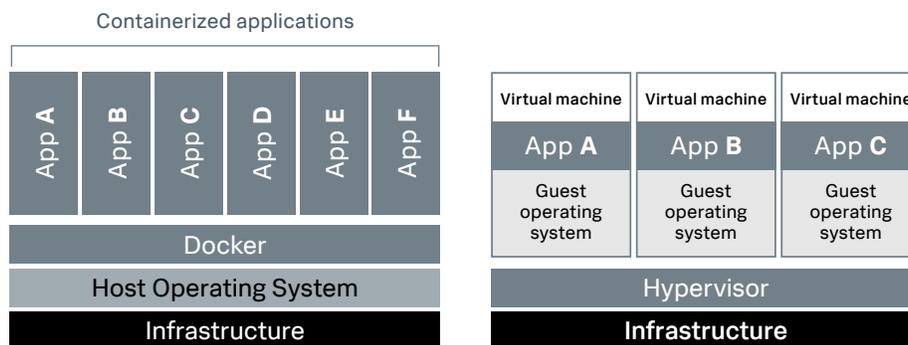
Containers enable a new agile way to develop software that empowers developers to have application portability and the ability to run multiple applications on the same OS without sharing dependencies. Containers provide a standard way to package your application’s code, configurations, and dependencies into a single object. As you can see in the figure, each container shares the host OS kernel and its binaries and libraries, making them extremely light (megabytes). With this level of abstraction, containers can be spun up and spun down in seconds, unlocking the ability to run immutable infrastructure.

### Formats and open standards

In 2013, Docker launched as an open-source container runtime project—thus beginning the container adoption sprawl. Since then, a multitude of open standards has arisen around the technology. Some are complementary, others are competitive, but a few have become the cream of the crop. Here are some of the more popular open standards:

- **OCI** (Open Container Initiative) Managed by the Linux Foundation, OCI Standards are supported by many vendors and govern image and runtime specifications.
- **CNI** (Container Network Interface) A CNCF (Cloud Native Computing Foundation) project consisting of specifications and libraries for writing plugins to configure network interfaces in Linux containers.
- **Kubernetes CRI** (Container Runtime Initiative) While Docker is the most popular container runtime, the space continues to evolve. CRI enables developers to use a wide variety of container runtimes.

Running containers at an enterprise scale requires decisions about which container runtime to use, as well as which container engine, orchestrator, storage, and network solutions should be adopted.



Containerized vs. Virtualized application stacks

## Containers on the cloud

If you are using cloud infrastructure to power your enterprise applications, the AWS Cloud offers infrastructure resources optimized for running containers, as well as a set of orchestration services that make it easy for you to build and run containerized applications in production. These services allow storing, managing, and running containers as per your requirements, from small experiments to critical production applications. AWS supports **OCI-compliant containers**, including **Docker** containers.

## AWS container management tools

Container management tools can be broken down into three categories as shown in the figure below: registry, compute, and orchestration. AWS offers services that give you a secure place to store and manage your container images, orchestration that manages when and where your containers run, and flexible compute engines to power your containers. AWS can help manage your containers and their deployments for you, so you don't have to worry about the underlying infrastructure.

### Registry

#### Amazon Elastic Container Registry

*Highly available and secure container registry that makes it easy to store and manage container images.*

[Learn more »](#)

### Orchestration

#### Amazon Elastic Container Service

*Fully managed container orchestration, seamlessly integrated with other AWS services.*

[Learn more »](#)

#### Amazon Elastic Kubernetes Service

*Easily deploy, orchestrate, and scale containerized applications using Kubernetes.*

[Learn more »](#)

### Compute

#### AWS Fargate

*Serverless compute engine built to run containers in production at any scale.*

[Learn more »](#)

#### Amazon EC2

*Run containers on virtual machine infrastructure with full control over configuration and scaling.*

[Learn more »](#)

## AWS container services considerations

Container orchestrators handle the deployment, maintenance, and scaling of containerized workloads that assist companies in operating containers. Kubernetes (K8s), born two years after Docker launched, has taken the community by storm to become the dominant container orchestration technology. Kubernetes was spun out of Google and was built by engineers who were tasked with managing more than two billion containers that were spun up and down each week. Kubernetes is by far the most popular container orchestration technology. In CNCF survey, 78% of respondents said they are using Kubernetes in production.

AWS offers infrastructure resources optimized for running containers, as well as a set of orchestration services that make it easy for you to build and run containerized applications in production. This includes a number of managed services around Kubernetes. Amazon EKS, for instance, is the AWS version of the Kubernetes orchestration service and runs the Kubernetes management infrastructure for you across multiple AWS availability zones to eliminate a single point of failure. Amazon EKS is Certified Kubernetes so you can use existing tooling and plugins from partners and the Kubernetes community.

AWS recommendations on which use case benefits from each type of AWS container service:

### Splunk Offers Built-in Monitoring for AWS Container Services

If you want to...	Consider using	Because...
Store, encrypt, and manage container images	Amazon ECR	ECR compresses and encrypts your container images, making them fast to start and available to run anywhere.
Run containerized applications or build microservices	Amazon ECS	ECS eliminates the need for you to install and operate your own container orchestration software or manage and scale a cluster of virtual machines.
Manage containers with Kubernetes	Amazon EKS	EKS provisions and manages a conformant, upstream Kubernetes control plane and is integrated with many AWS services to provide scalability and security for your applications.
Run containers without managing servers	AWS Fargate	Fargate scales and manages the infrastructure required to run your containers. This removes the need to choose server types, decide when to scale your clusters, or optimize cluster packing.
Run containers with server-level control	Amazon EC2	EC2 virtual machines gives you control of your server clusters and provide a broad range of customization options.

**Built-in Monitoring with Splunk**

When choosing one or more of the container services and tools offered by AWS, besides your primary use case, you need to keep the following considerations in mind:

- Use case (application, microservices architecture, hybrid- or multi-cloud requirement)
- Entities that need to be administered by users
- Cluster configuration flexibility requirements
- Cluster management needs
- Control of worker nodes in Kubernetes environment
- Control over cloud server control
- Freedom of choice of container orchestration tool
- Observability needs, real-time application performance monitoring needs, and scale of monitoring
- Pricing considerations

The table below provides details regarding strengths and recommended use for each of the AWS container services. It also highlights key metrics and considerations for different container usage scenarios that are served best for the service. Based on your use case and level of control over infrastructure and container components, you can choose these services.

Container service	Strengths	Recommended use	CloudWatch metrics
<p><b>Amazon Elastic Container Service (Amazon ECS)</b></p>	<ul style="list-style-type: none"> <li>• Much simpler than Kubernetes interface for managing containers</li> <li>• Closely resembles server-based workloads making it easier to migrate to cloud based containers</li> <li>• Well integrated with AWS services and provides networking and support components via AWS service components such as Application Load Balancers (ALBs), Route 53, and Amazon CloudWatch</li> </ul>	<ul style="list-style-type: none"> <li>• Learning curve with ECS is much smaller for organizations with limited DevOps resources who do not wish to rearchitect applications around the concepts of Pods in containers</li> <li>• If you are leveraging “software defined infrastructure” such as Terraform, AWS Elastic Beanstalk, you will generally find ECS well supported in these apps</li> <li>• If you need to control scaling at the task level then you need to use ECS, otherwise AWS Fargate abstracts container and server level scaling needs to a much simpler task scaling model</li> <li>• AWS controls and manages the clusters, allowing some cluster configuration by customers which is helpful if your application requires full control over Docker; however, container orchestration is AWS proprietary using open source schedulers</li> <li>• Pricing: You pay for Amazon EC2 instances, Amazon Elastic Block Store volumes, and any additional AWS services used for deploying and setting up your application servers — there is no additional charge for ECS but if you use the AWS Fargate for ECS option, you pay for the amount of vCPU and memory resources that your containerized application requests</li> </ul>	<ul style="list-style-type: none"> <li>• CPU and memory reservation and utilization metrics for containers, disk and networks</li> <li>• Amazon ECS clusters and ECS service, Running Task counts</li> </ul>

While most managed Kubernetes providers are releasing hybrid versions of their container solutions, multi-cloud support remains negligible. This is driving many users to continue to adopt Kubernetes as a standalone environment that they manage themselves. If you need more control over container placement and traffic routing decisions for your application, you could choose **Amazon ECS** or **Amazon EC2** -based container environments.

Container service	Strengths	Recommended use	CloudWatch metrics
<p><b>Amazon Elastic Compute Cloud (Amazon EC2)</b></p>	<ul style="list-style-type: none"> <li>• Proven, highly available, AWS cloud server instances to host your managed container application environment</li> <li>• Flexibility in choosing container management software as per your application needs</li> <li>• Highest level of control over cloud server instances and applications that run on it, enabling user to manage provisioning, patching, and scaling clusters of servers – you can decide which type of server to use, which applications and how many containers to run in a cluster to optimize utilization, and when you should add or remove servers from a cluster</li> </ul>	<ul style="list-style-type: none"> <li>• If your application requires extremely robust, scalable, and tightly controlled container environment that is hybrid – on cloud as well as on premises, using Kubernetes on Amazon EC2 server instances is a good option</li> <li>• You have full freedom to choose your preferred orchestration tooling, such as Kubernetes or, if you need simple container management and wish to manage it yourself, Docker Swarm</li> <li>• Users have full control over server clusters and a broader range of customization options for your container environment, which might be required to support some specific applications or compliance and government requirements</li> <li>• Using a self-managed Amazon EC2 cluster for deploying containers is better suited for application portability</li> <li>• Pricing: You pay for Amazon EC2 instances, Amazon EBS volumes, and any additional AWS services used for deploying and setting up your application servers such as storage, networking, LBS, messaging etc.</li> </ul>	<ul style="list-style-type: none"> <li>• Amazon EC2 instance – CPU and memory utilization, memory reservation, at the cluster, service and task level</li> <li>• Availability and AWS Zone metrics</li> <li>• Disks, network and port monitoring metrics</li> <li>• Scalability metrics</li> </ul>

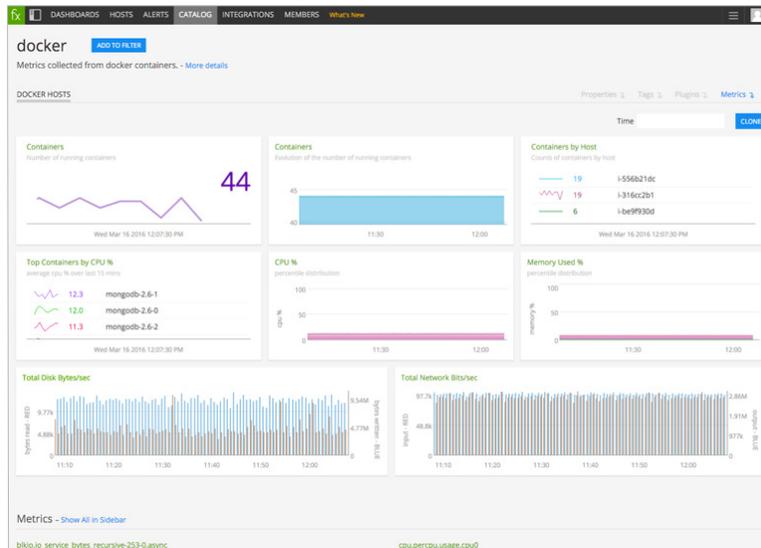
Container service	Strengths	Recommended use	CloudWatch metrics
<p><b>Amazon Elastic Kubernetes Service (Amazon EKS)</b></p>	<ul style="list-style-type: none"> <li>• Tight integration with Amazon services</li> <li>• Amazon manages your Kubernetes cluster control plane</li> <li>• Integrated with IAM, VPC, auto scale models</li> </ul>	<ul style="list-style-type: none"> <li>• If you are already using an on premises or cloud-provided host for Kubernetes, or are looking to utilize a de-facto industry standard for open source orchestration of containers, EKS can provide many benefits of Kubernetes without the operations responsibility of hosting and configuring the Kubernetes environment</li> <li>• If you are leveraging AWS but want to keep your infrastructure portable for other cloud providers or have a need for multi-cloud infrastructure</li> <li>• You have requisite DevOps skills and your apps need to control how containers are managed</li> <li>• Pricing: There are additional EKS control plane costs on top of the costs for Amazon EC2, Amazon EBS and other AWS services that your application uses besides the containers</li> </ul>	<ul style="list-style-type: none"> <li>• Amazon CloudWatch helps to monitor Kubernetes control plane logs including: audit, API server, authenticator, controller-manager, and scheduler logs</li> <li>• State of containers, pods, nodes, network, cluster, namespace, and AWS services</li> </ul>

Container service	Strengths	Recommended use	CloudWatch metrics
<b>AWS Fargate</b>	<ul style="list-style-type: none"> <li>Serverless containerized environment for modern applications that lets developers focus on writing code instead of worrying about container provisioning and scaling</li> <li>AWS Fargate is a completely managed solution where users simply setup their application tasks, and deploy them on AWS-managed ECS instances, which allows users to run containerized applications without having to manage servers or clusters</li> <li>With AWS Fargate, users simply define the amount of CPU and memory that their Task or Service requires to operate, and only pay for the CPU and memory time rather than the Amazon EC2 instance</li> </ul>	<ul style="list-style-type: none"> <li>AWS Fargate is recommended for containerized applications that are complex to auto scale efficiently and often end up with a lot of unused CPU and memory resources while using the Amazon EC2 or ECS</li> <li>Good option for tasks that run in response to an event, on-demand, or on a schedule and do not require a dedicated Amazon EC2 server instance whether it is used directly or as part of ECS option</li> <li>Best suited for small microservices, on-demand jobs that do not require any control over either the cluster or server or container orchestration and very limited Docker control</li> <li>Pricing: You only pay for the resources that are defined per task – pricing is calculated based on the vCPU and memory resources used from the time you start to download your container image (Docker pull) until the AWS task terminates, rounded up to the nearest second (with a minimum charge of 1 minute)</li> </ul>	<ul style="list-style-type: none"> <li>Amazon CloudWatch Log Streams for AWS Fargate</li> <li>Metrics for AWS Fargate by clusters by tasks, containers, and services (<a href="#">Preview mode as on July 2019</a>)</li> </ul>

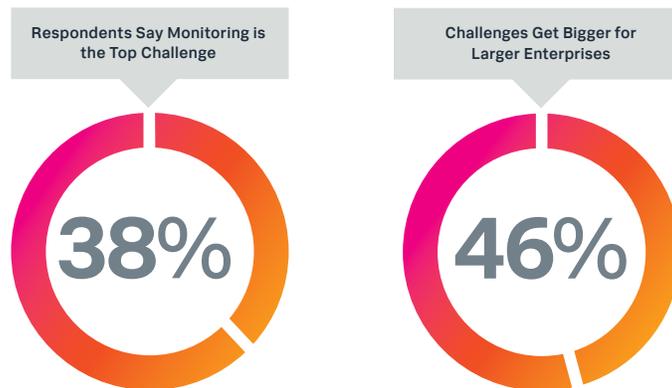
Container service	Strengths	Recommended use	CloudWatch metrics
<b>Amazon Elastic Container Registry (ECR)</b>	<ul style="list-style-type: none"> <li>Amazon ECR supports private Docker repositories with resource-based permissions using AWS IAM so that specific users or Amazon EC2 instances can access repositories and images</li> <li>Tight integration with Amazon ECS and the Docker CLI, allowing you to simplify your development and production workflows</li> </ul>	<ul style="list-style-type: none"> <li>Developers can use the Docker CLI to push, pull, and manage container-based application deployment images</li> <li>If your application infrastructure is fully AWS based, it is beneficial to use ECR as it eliminates the need to operate your own container repositories or worry about scaling the underlying infrastructure.</li> </ul>	<ul style="list-style-type: none"> <li>Monitor 'Put Image' notification events for new images added to the repository</li> </ul>

## Container monitoring and observability strategy

Choosing the right container monitoring and observability strategy can be as daunting as choosing the right runtime and orchestration technologies. It is one of the most critical components of getting your container architecture production-ready. If you're pushing new containers into production without a fool-proof monitoring and observability strategy in place, you're going to have problems.



Monitoring and observability have become highest cited concerns for DevOps teams that are adopting new cloud-native architectures, according to a CNCF survey of nearly 2,400 CNCF members.



Survey results: Container monitoring is the top challenge

Source: <https://www.cncf.io/blog/2018/08/29/cncf-survey-use-of-cloud-native-technologies-in-production-has-grown-over-200-percent/>

## You need to consider the following when planning and executing your container monitoring and observability strategy.

### Consideration #1: Interoperability

There are many container technologies to choose from, just as there are many varieties of cloud-native technologies. Knowing how well your container technology integrates within your broader cloud-native stack is critical.

For example, if your **Amazon Elasticsearch Service** data is in monitoring system A and your container telemetry data is in monitoring system B, there's not much chance of success. To ensure you have all the data to make effective decisions, you need a monitoring platform that has the full interoperability you need to capture and correlate data, with a single source of truth for observability. You need to look at robustness, breadth, and coverage when it comes to integration.

### Recommended strategy

Choose a monitoring vendor that is highly connected into the cloud-native ecosystem, e.g. Splunk combined with AWS.

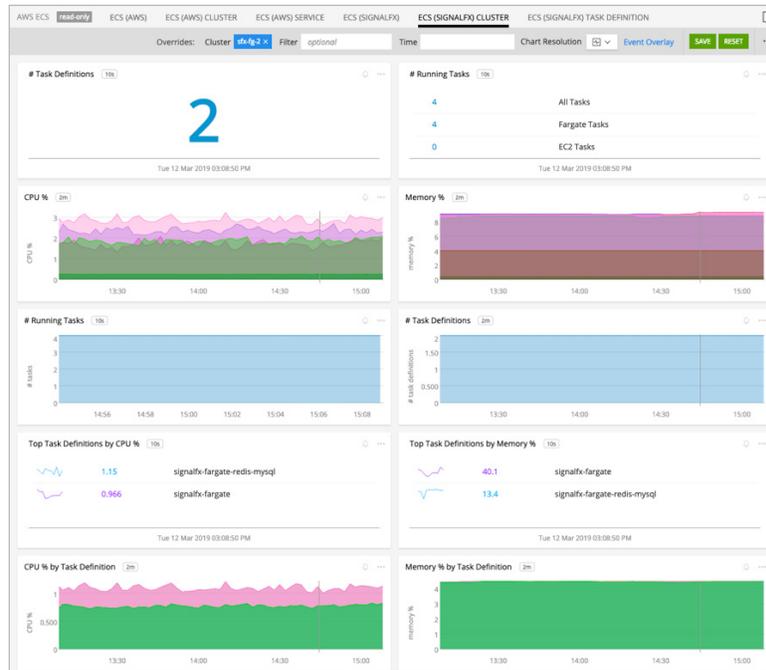
### Consideration #2: Discovery and ingest

Since containers can be extremely short-lived (seconds/minutes), it's paramount to discover new containers as they are spun up and down, and associate them with the correct host (Amazon EC2, etc.) and microservice for faster MTTR. Containers are lightweight, and as such it doesn't make sense to run heavyweight agents on them that are used by traditional monitoring vendors.

### Recommended strategy

Choose a vendor that employs a lightweight agent to auto-discover new containers and services. With Splunk Infrastructure Monitoring, you can see detailed metrics that can offer real-time proactive troubleshooting insights using **Splunk container resource monitoring dashboards**. Select a solution based on open standards rather than one that makes use of its heavyweight and/or proprietary agents. This alleviates the concern of vendor lock-in and reaps the benefits of being ecosystem friendly.

Splunk Infrastructure Monitoring dashboard below shows performance metrics related to Amazon ECS tasks:



Splunk Infrastructure Monitoring Dashboard for Amazon ECS

### Consideration #3: Scale

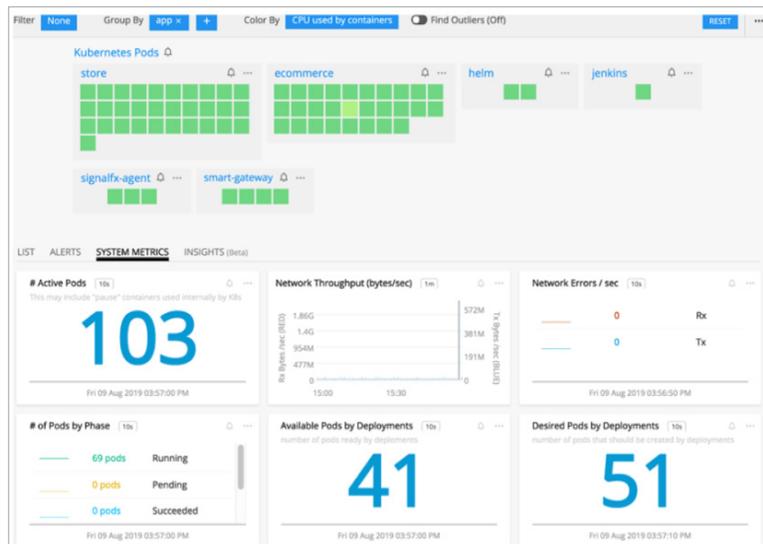
The proliferation of microservices and containers has created an explosion of data points to analyze. Current trends will continue to increase the number of data points: Bin-packing is a common strategy that increases the number of containers per host which means more data per host. Teams are doing this to help with cost savings on cloud compute. Also, right-sizing of compute instances means there are many more instances than the traditional large hosts.

Granular observability use cases (such as monitoring/alerting on a per customer or category basis) creates performance degradation of traditional monitoring tools. High data volume prevents important use cases (e.g. searching across 1,000 containers over the course of a year translates to 31 billion data points), capacity planning, and even slows root-cause analysis during outages.

## Recommended strategy

Building your own solution to handle scale presents serious trade-off considerations, and ultimately might not be worth the headache. Remember, as you continue to scale your containers in the future, data volumes will continue to grow rapidly. A home-grown system must not only handle your data scale today but continue to perform well in the future. It will require serious investments, care and feeding. You'll not only have to scale data storage so it doesn't need to be fragmented among multiple siloed databases, but queries must continue to respond quickly at higher data volumes. Finally, data burstiness means peak loads are much higher than normal and provisioning for peak capacity wastes a lot of money during off-peak hours.

Commercial options are available. Open source TSDBs (Time Series Database) are often chosen when leveraging K8s and containers for a lot of monitoring teams. However, at some point you will feel the pain of the issues discussed above. It may be a better use of your time, developers, and money to focus on your core business and simply leverage a managed monitoring solution like Splunk Infrastructure Monitoring that is built with containers in mind as shown below.



Splunk Dashboard showing system metrics for container environment

## Consideration #4: Latency

Traditional batch monitoring systems that run the full query and load the full dataset every time an alert is evaluated are not adequate for modern needs. These systems are unable to refresh queries more often than once every few minutes, which means end users can't track applications with fine granularity. They'll also miss important alerts and analysis on containers due to their ephemeral (short-lived) nature.

### Recommended strategy

Streaming analytics technology, when properly implemented, is superior to a batch approach when it comes to monitoring. Not only can streaming analytics support bigger data volumes and concurrent users/queries, but it can also trigger updates and alerts in real-time. We recommend that you build or buy a monitoring solution that streams data so you can alert on and analyse container data within seconds. This delivers the ability to act on problems faster and even trigger alerts via webhooks to perform automated rollbacks and other operations to remediate issues at machine time.

## Consideration #5: Churn

Churn is the most underestimated, and potentially the most painful problem in monitoring. From a technical standpoint, churn occurs when a source of data is replaced by another equivalent one. This churning of containers creates an explosion of metadata because the identities of the data sources keep changing. Some use cases that cause this churn are CI/CD deployments (some organizations blue-green their entire environment), auto-scaling, short-lived spot instance on the cloud, and large-scale tagging that affects many metrics (e.g., adding a new tag like 'customer\_id' to all your metrics). Not only does churn cause accumulation of metadata over time, but sometimes it also happens in bursts (e.g., while blue-greening an entire application or service). Many monitoring systems are unable to handle this sudden volume of metadata and cannot process or index it quickly enough.

### Recommended strategy

Churn can be dangerous for your monitoring system, and the effects can become progressively worse over time. If you're pushing code daily, you'll have to handle 365x more metadata after a year, and it's much more challenging with containers.

If you've already built or plan to build a system, here are some of the things you will want to consider are your plans for provisioning capacity for handling bursts or slow down your code pushes to smoothen bursts if historically querying of your data is an important use case for you, you'll probably end up implementing some sort of 'pre-aggregation' or 'pre-compute' scheme on a per-service or app basis to deal with this. However, that will introduce the trade-off between timeliness and accuracy of your alerts. Most open source tools won't let you have both.

Churn issues are also widely present in today's monitoring vendors. They might work great in a small POC (Proof of Concept), but performance will degrade with time and scale of your environments. Choose a vendor that has built its system with the problem of churn in mind.

## Consideration #6: Visibility

Finding the root cause of an issue is difficult in today's distributed, containerized environments. Having a solution that can help you visualize your data across your stack is important in teasing the signal from the noise when incidents pop up.

### Recommended strategy

Leveraging Grafana for general visualization or Sysdig to get deep into the visualization of your containers down to the kernel level are options but they can only solve one piece of the observability puzzle. Be careful you don't stack up too many point tools. Instead, consider consolidating your tooling as much as possible for cost savings. You may want to choose an end-to-end observability platform, rather than cobbled together point tools that can cover the breadth of your full-stack, with pre-built dashboards for fast time to value.

## Consideration #7: Troubleshooting

It's important to correlate your infrastructure to the applications that infrastructure is servicing. As some companies adopt containers, they're doing so with the mindset to break up their monolith app into microservices or start anew with a distributed services architecture for their greenfield app. This makes it critical to also be able to capture the transaction data as it traverses these services your containers support.

### Recommended strategy

Leverage a tail-based distributed tracing solution that observes all the transactions that traverse your application layer and store the p90 and p99 traces for historical analysis. Also, make sure your solution will help your team troubleshoot issues in a guided fashion. The distributed nature of today's systems creates interdependencies between your services that are critical to observe and require the ability to tease out where the bulk of the latency is in the communication between your services. You also need to quickly answer whether the underlying infrastructure is contributing to increased latency or errors, so you need a solution that ties your infrastructure metrics and application traces together.

## Splunk as the core of your observability strategy for containers

Splunk provides unified monitoring for Amazon Web Services infrastructure, Kubernetes platform, App Mesh data plane — Envoy, Docker containers and microservices from a single-pane-of-glass. Whether you deploy App Mesh on Amazon EKS, Amazon ECS or Kubernetes on Amazon EC2, Splunk provides comprehensive full-stack monitoring.

Splunk Infrastructure Monitoring provides a robust interoperability with Amazon CloudWatch, has a CloudWatch-powered mode for the **Infrastructure Navigator**, and includes many **built-in-dashboards** to help you get started monitoring AWS. You can also monitor AWS instances and the services running on them by using the **Smart Agent**. The Smart Agent offers a much higher degree of customization than is possible with CloudWatch, and may be preferable for instances where you want to see metrics at a finer resolution, or where detailed control over the metrics sent matters. The Smart Agent is a metric agent written in Go for monitoring infrastructure and application services in a variety of different environments. It installs with more than 100 bundled monitors for gathering data, including Python-based plug-ins such as Mongo, Redis, and Docker. Initially developed as a Kubernetes integration, the Smart Agent now gathers host performance, application, and service-level metrics from both containerized and non-container environments.

For a complete list of AWS services and infrastructure monitored by SignalFx see: <https://docs.signalfx.com/en/latest/integrations/amazon-web-services.html>

As you will see in the tables on the following pages, Splunk technology is uniquely positioned to meet the new monitoring criteria of the cloud-native stack, and help you take centralized control over your observability strategy.

Considerations	Splunk	Open source	Traditional monitoring solutions
<b>Integrations</b>	Hundreds of integrations into the most popular cloud-native technologies (including Prometheus)	Most open source tools lack pre-built integrations, except for Prometheus, which has some	Hundreds of integrations into the most popular cloud-native technologies
<b>Discovery and ingest</b>	Smart Agent auto-discovery and configuration for Docker, Kubernetes, and other popular cloud-native technologies	Most open source tools lack auto-discovery and auto-configuration capabilities, however, Prometheus does provide some limited auto-discovery capabilities	Proprietary agents and limited agent auto-discovery and configuration for containers and Kubernetes
<b>Scale</b>	Streaming architecture handles 100,000s of components seamlessly with high resolution	Open source TSDBs not built for horizontal scalability and see performance degradation around 5,000 components	Batch architecture limits scalability to <10,000 components
<b>Latency</b>	Streaming architecture enables alerts to fire and charts to update in seconds when new containers are spun up or down	All open source tools have pull-based (batch) architectures that poll data every few minutes to update charts and fire alerts, which miss critical data from ephemeral containers that dynamically spin up and down data sets	Batch architecture limits the timeliness of alerts, creates delays in updating charts (on the order of minutes), and cause queries to take minutes to hours to complete, limiting ability to effectively monitor containers-limiting key use case
<b>Churn</b>	Splunk has a purpose-built metadata store (separate from the TSDB) built to handle large metadata volumes and high cardinality associated with high container churn	Open source TSDBs only have one database for time series and metadata and often limit the number of tags you can apply to metrics (<1,000)	Discourages the use of more than 1,000 tags on metrics due to degradation of overall performance

Continued ▼

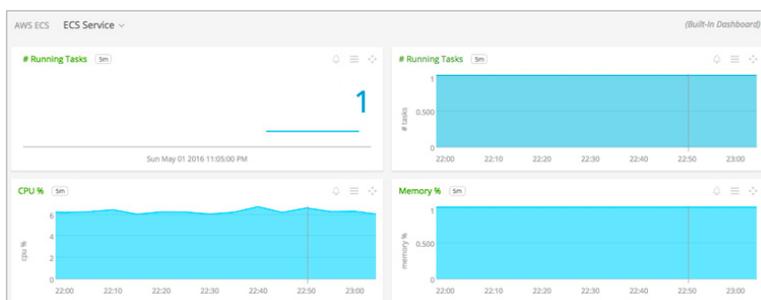
As you will see in the tables on the following pages, Splunk technology is uniquely positioned to meet the new monitoring criteria of the cloud-native stack, and help you take centralized control over your observability strategy.

Considerations	Splunk	Open source	Traditional monitoring solutions
<b>Visibility</b>	<ul style="list-style-type: none"> <li>• Full-stack observability with native support for metrics, distributed traces, and deep-linking functionality into best-of-breed solutions for logs</li> <li>• Mirrored dashboard functionality cuts down on dashboard sprawl and promotes best practice content across teams and organizations</li> </ul>	<ul style="list-style-type: none"> <li>• Limited visibility into container metrics hinders full observability</li> <li>• Point solutions for metrics, traces and logs that lack full set of capabilities and require significant effort to individually configure and integrate together</li> </ul>	<ul style="list-style-type: none"> <li>• Mix of point solutions and lightly integrated suites for visibility across metrics, traces, and logs</li> <li>• Limited visibility into distributed traces with random sampling hinders full observability</li> </ul>
<b>Troubleshooting</b>	<ul style="list-style-type: none"> <li>• Outlier Analyzer™ uses data science to quickly analyze related traces to identify contributing factors to high latency, significantly lowering MTTR</li> <li>• Tightly integrated metrics and distributed traces platform enables context correlation and rapid problem isolation, enabled by trace/span metricization (RED+) for unique identification, historical baselining, and calculating aggregates and percentiles</li> </ul>	<ul style="list-style-type: none"> <li>• Open source tracing tools such as Zipkin and Jaeger lack user-friendly powerful UIs. They often force trace by trace analysis which is time-consuming and drives down MTTR</li> <li>• Random sampling of transactions (to reduce data volumes to manageable levels) misses key outliers and anomalous traces, preventing understanding and mitigation of P99 issues</li> <li>• Siloed tools with APM only – users are not able to correlate transaction performance with infrastructure health</li> </ul>	<ul style="list-style-type: none"> <li>• Lack of built-in analytics and manual trace-by-trace analysis which drives down MTTR</li> <li>• Random sampling misses key outliers and anomalous traces, preventing understanding and mitigation of P99 issues</li> </ul>

## Monitoring AWS container services with Splunk Infrastructure Monitoring

Monitoring and observability is one of the key considerations when choosing any of the AWS container services. The highly dynamic and ephemeral nature of containerized environments makes them very hard to monitor using traditional infrastructure monitoring or APM solutions. With hundreds — if not thousands — of components being spun up and down every day or even hour, batch-based monitoring solutions cannot keep up with the churn and in many cases cannot even see the new components, let alone enabling multi-dimensional real-time analysis.

The Splunk streaming analytics engine is the only one that can keep up with the dynamic nature of containerized environments without compromising performance and being overwhelmed by alert storms. Some of our customers are running the most demanding container based production environments in the industry, with millions of components being churned on a daily basis.

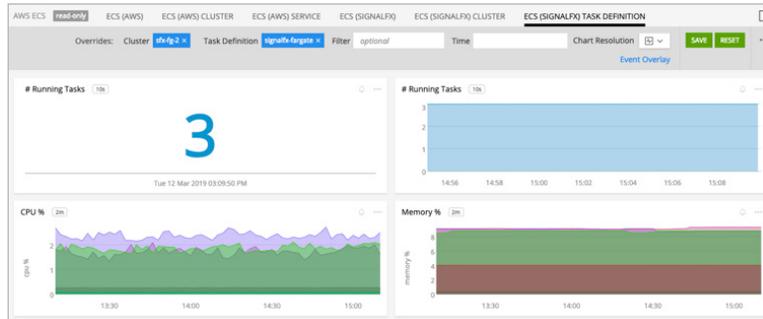


**Metrics Monitoring** — Focus on single Amazon ECS service using Amazon CloudWatch

### Create your monitoring plan geared for your specific container environment

Besides the container considerations listed earlier, we recommend that you should create a monitoring plan that addresses the specifics related to your container environments such as:

- Monitoring goals
- List of resource types and services that need to be monitored
- Monitoring frequency
- Tools and integration requirements
- Monitoring roles and responsibilities
- Process to notify and handle alerts and level of automation



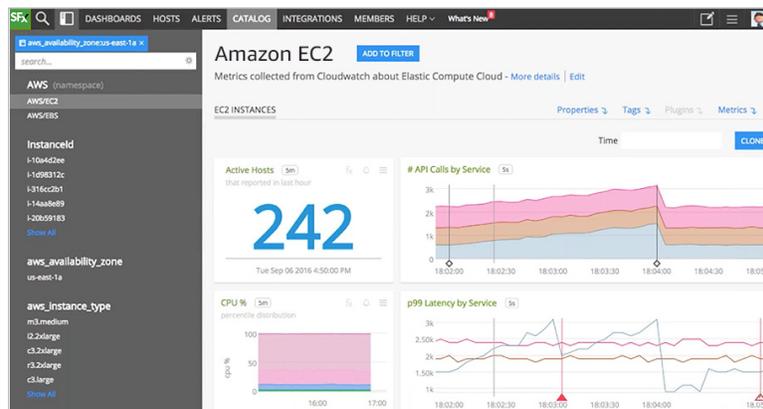
**Metrics monitoring** — Focus on single Amazon ECS task definition using smart agent

Splunk helps you closely monitor AWS container services and resources in real-time. Use Splunk to monitor Amazon ECS via our **out-of-the-box alignment with AWS**. For greater insight into your Amazon ECS environment, **Smart Agent** can auto-discover services and provide more in-depth metrics about your containers that are running on ECS.

Splunk can help you better monitor and manage your container environments. It bridges the gap between what you require to monitor and what Amazon CloudWatch offers. Also, it provides a simpler way to visualize your monitored data as opposed to the programming interface of CloudWatch. When Amazon ECS collects metrics, it collects multiple data points every minute. It then aggregates them to one data point before sending the data to CloudWatch. So with CloudWatch, one sample count is actually the aggregate of multiple data points during one minute. Splunk allows you to monitor your container metrics at a more detailed frequency of one second. AWS allows you to access historical metrics and logs for a period of two weeks. This data can be used for baselining and identifying patterns for troubleshooting. With Splunk, you can have more rich, fine-grained metrics roll ups, **resolution and data retention of up to a year** based on your subscription plan. SignalFx AWS Optimizer gives you actionable insight into cost-saving opportunities and underutilized Amazon EC2 investments. You can see usage patterns and cost attribution by InstanceType, AWS Region, AWS Availability Zone, as well as categories specific to your setup, such as service, team, or any other dimensions that are sourced from Amazon EC2 instance tags.

Splunk provides a robust combination with Amazon CloudWatch, has a CloudWatch-powered mode for the Infrastructure Navigator, and includes many built-in-dashboards to help you get started monitoring Amazon Web Services. You can also monitor AWS instances and the services running on them by using the Smart Agent. The Smart Agent offers a much higher degree of customization than is possible with CloudWatch, and may be preferable for instances where you want to see metrics at a finer resolution, or where detailed control over the metrics sent matters.

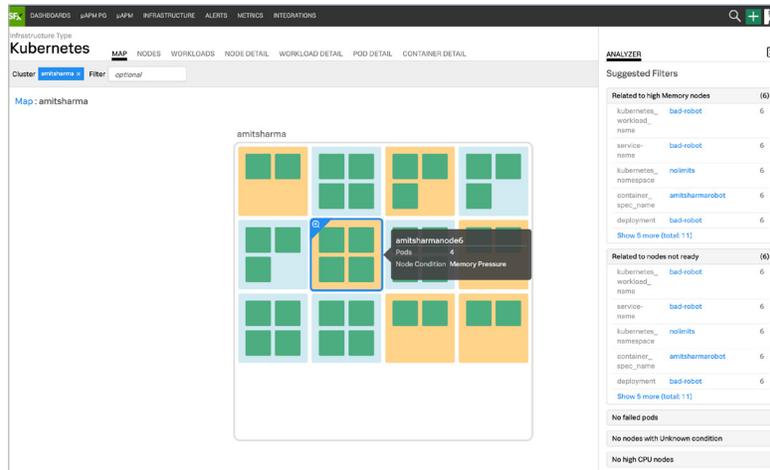
If you wish to know how to deploy Smart Agent in Amazon ECS, see the [Smart Agent Amazon ECS Deployment Guide](#) for instructions. Also, Smart Agent can be deployed on ECS task to monitor AWS Fargate containers. See the [Smart Agent AWS Fargate Deployment Guide](#) for more detailed deployment instructions.



Splunk Amazon EC2 monitoring dashboard

## Kubernetes Navigator

Included with Splunk Infrastructure Monitoring, Kubernetes Navigator is an enterprise-grade and turn-key Kubernetes monitoring solution that provides an easy and intuitive way to understand and manage the performance of Amazon Elastic Kubernetes Service (EKS) environments. Working alongside hundreds of customers, we designed a solution that works for every team—irrespective of their maturity and experience with Kubernetes. Kubernetes Navigator brings immediate value to teams that are starting on their cloud-native journey while also addressing monitoring challenges for the world's most complex Kubernetes deployments at scale.



With Kubernetes Navigator, your teams can detect, triage, and resolve performance issues faster than ever before. DevOps and SRE teams can successfully navigate the complexity associated with operating Kubernetes at scale by taking advantage of these features:

- Dynamic Cluster Map: An intuitive way to instantly understand the health of Kubernetes clusters
- Drill-downs: Faster and effective troubleshooting with quick drill-downs
- Logs in context: Deep linking to contextual logs to gain granular insights, eliminate context switching and accelerate root cause analysis
- Kubernetes Analyzer: AI-driven analytics to expedite troubleshooting

## Dynamic cluster map

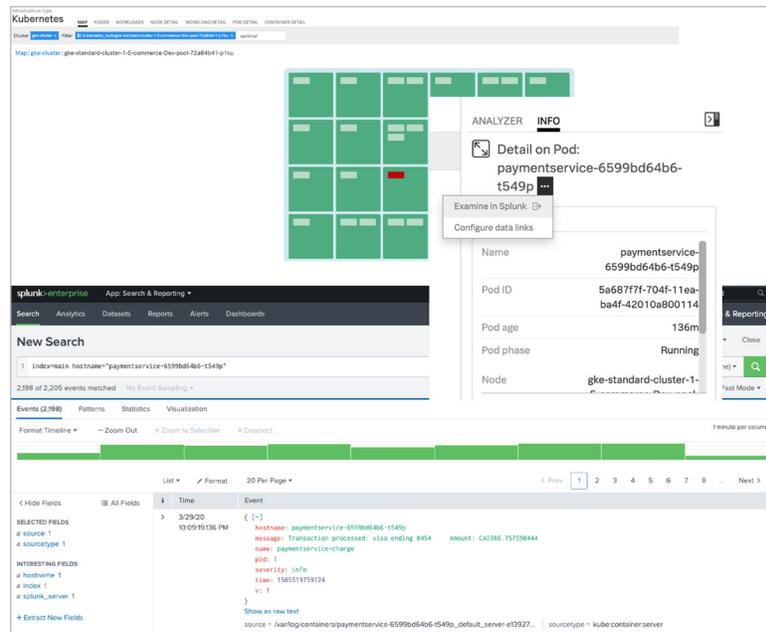
Starting with the bird's eye view, Kubernetes Navigator enables teams to quickly understand the performance of the entire Kubernetes environment with intuitive and hierarchical navigation. Select, filter, or search for any Kubernetes entity, e.g., node, pod, and container level within seconds. Splunk automatically discovers Kubernetes components and containerized services to instantly monitor your entire stack. Understand relationships between dynamic Kubernetes components and quickly fix interdependent performance issues.

## Drill-downs

A global, at-a-glance view into the entire Kubernetes environment helps teams understand how the overall system is performing. It is equally important to have a granular, detailed view into individual components as teams narrow down to the source of the problem—drilling down from nodes to pods to containers to workloads. Our streaming architecture enables in-depth analysis with search and filters within seconds at a massive scale.

## Logs in context

Seamlessly pivot to logs and get granular visibility into application, Kubernetes, and container logs to correlate performance across the entire stack without any context switching. Visibility into lifecycle events of Kubernetes and API Server Audit logs help you understand and maintain your security and compliance postures.



The screenshot displays the Splunk Kubernetes Analyzer interface. At the top, there's a navigation bar with 'Kubernetes' and various tool icons. Below it, a search bar shows a query: 'index=main hostname="payment-service-6599bd64b6-t549p"'. The search results indicate 2,198 of 2,205 events matched. A 'Detail on Pod' panel is open, showing information for 'payment-service-6599bd64b6-t549p', including its Pod ID, age (136m), phase (Running), and node (gke-standard-cluster-1-...). Below the pod details, a log entry is visible with a timestamp of 3/25/20 at 10:05:55.136 PM. The log message is 'Transaction processed: visa ending 0454 Amount: CAD385.75758444'. The interface also includes a 'New Search' section, a 'Format Timeline' section, and a 'List' section showing the selected fields and the event details.

## Kubernetes Analyzer

To understand the “why” behind performance anomalies, Kubernetes Navigator leverages AI-driven analytics, which automatically surfaces insights and recommendations to precisely answer, in real-time, what is causing anomalies across the entire Kubernetes cluster — nodes, pods, containers, and workloads. In the example below, Kubernetes Analyzer automatically detects a pattern causing memory pressure on some of the Kubernetes nodes. In this case, a container with an unlimited memory limit ends up consuming all the available memory on these nodes draining other pods scheduled by Kubernetes. This scenario is commonly known as a noisy neighbor issue. Following suggested filters, SRE teams can narrow down to the underlying issue within minutes. Sophisticated algorithms, including Historical Performance Baselines and Sudden Change, detect system-level issues such as a sudden increase in Goroutines or container restarts and alert within seconds.

## Summary

While new cloud-native technologies like containers are allowing faster innovation with more nimble and resilient application development, these new technologies are also increasing the level of complexity for monitoring and troubleshooting use cases. It's critical to implement the correct monitoring, troubleshooting and overall observability strategy as you adopt containers. Splunk supports AWS container infrastructure and services integration making it easier for companies to manage their application performance in Kubernetes environments and maximize benefits of adopting containers at scale.

Customers are increasingly adopting containers, orchestration and microservices architecture-based applications to deliver innovation faster and to make applications more resilient. AWS container services and Splunk platform complement the efforts of Site Reliability Engineers and operations teams with streaming intelligence to assist with problem detection and troubleshooting in container environments. Together, AWS and Splunk can also accelerate code deployment and allow for automated performance data capture for applications. The Splunk solution offers pre-built dashboards featuring performance metrics and other visualizations to customers with system-wide monitoring, observability and directed troubleshooting — all critical requirements for confidently adopting containers at scale in production environments.

Future-proof your observability investment with a proven solution trusted by enterprises for most advanced use cases at a massive scale. [Sign up for a free trial.](#)