



GUIDE TO MONITORING MICROSERVICES

For organizations adopting Kubernetes

Summary

In order to anticipate problems & discover bottlenecks state in production environment, monitoring an application's current is most effective. Yet, it is one of the biggest challenges faced by many software development organizations. Inexperience, lac of focus, and management are some of the obstacles that can touch how successful organizations are in overcoming these challenges. The growing uses of microservices makes monitoring even more complex giving rise to the need of the complex Kubernetes monitoring.

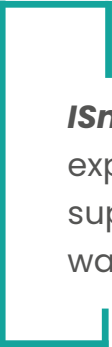


In Kubernetes environment, a large number of applications – distributed & diversified – are communicating with each other. A single point of failure can seize the entire process & identifying it is very difficult. And monitoring is just one challenge that microservices pose, managing availability, performance, and deployment are pushing the organizations to use orchestrators to handle all the services & servers. Though, there are several cluster orchestration tools, Kubernetes is becoming very popular than its competitors. Kubernetes manages containers in several computers & eliminates the complexity of handling distributed processing. But how to monitor such a tool? In this E-Book, we will discuss the importance of monitoring microservices on Kubernetes, and also discuss several tools that can be used for this purpose.

Why Kubernetes is important?

The way the development & operations team test & deploy today's software is not the same as it was years ago thanks to the container-based microservices architecture. Containers facilitate the modernization initiatives of small & large companies by letting the creation of an entirely new infrastructure ecosystem. Can you imagine the complexity of the scale that today's companies have to manage given that thousands of container instances are getting deployed daily? So, how it is done? That's where we look at Kubernetes. It's an open-source container orchestration platform created to automate the deployment, scaling, and management of containerized applications. In a microservice architecture, Kubernetes makes it easy to deploy & work with applications. For doing so, it creates an abstraction layer on top of a cluster of hosts. In this way, development teams deploy their application while Kubernetes manages the following activities.

- Controlling the resource which is consumed by the application or team.
- Spreading the application load evenly across a host infrastructure.
- Handling load balancing requests automatically across the different instances of an application.
- When any application starts consuming too much resource, it stops the application automatically, monitors the resource limit, and then, restarts the application again.
- If a host is short on resources or if it's dying, then the application instance has to be moved from one instance to another.
- Automatically providing additional resources when a new host is placed in the cluster.
- Doing canary deployments & rollbacks.



ISmile Technologies is a vetted Kubernetes service provider with extensive experience in helping enterprises successfully adopt Kubernetes. We provide support, consulting, professional services, and training for the companies that want to adopt Kubernetes.

As organizations move to microservices & cloud-native architecture, practitioners are adopting Kubernetes for the following four reasons.

Kubernetes helps you move quickly

Kubernetes helps in the creation of a hardware abstraction layer for the development teams. Whenever the development team needs any resources, they can quickly & efficiently place the requests for them. If they are facing an additional load, and they need more resources, they can get them very quickly as all the resources come from an infrastructure shared across all your team. Kubernetes automates the packaging, deployment, and testing. There is no need to fill out forms and wait for requesting new machines to run your application.

Kubernetes is cost efficient

Kubernetes & containers are known for much better resource utilization than hypervisors & VMs. Since the containers are very lightweight, they require very less memory & CPU to function.

Kubernetes is cloud-agnostic

Kubernetes runs just fine on Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP). You can also run it on-premises. With it, workloads can be easily moved without redesigning your application, completely rethinking your infrastructure. This helps in on-platform standardization & vendor lock-in.

You don't have to worry about managing Kubernetes

Cloud providers will manage it for you. Kubernetes has emerged as the standard for container orchestration tools. And therefore, major cloud providers are giving plenty of Kubernetes-as-a-service offerings. So all major players such as Amazon EKS, Google Cloud Kubernetes Engine, IBM Cloud Kubernetes Service, and Azure Kubernetes Service – all provide full Kubernetes platform management. This allows you to focus you on your core competency – shipping applications for your users.

What is microservices & why are businesses adopting it for application building?

In software development, small things are easier to code, easier to test, simple to maintain, and easier to deploy. Applications built this way are scalable, flexible, and can deliver value at great speed. Microservices support this type of development. It is a building block of designing today's distributed enterprise system & is an enabler in the digital transformation journey. It gives the agility, reliability, scalability, and deployability that helps organizations in digital transformation process.

It helps enterprise with the following factors & that is the reason why businesses are using it:

- It helps you to focus on the business needs.
- Microservices gives agility, by which organizations can introduce new products or features into the market quickly.
- It helps in boosting the customer experience.
- It makes systems highly reliable; if a service is down users will not be able to use the functionality related to that service, but the rest of the system will be perfectly usable.
- It makes applications highly scalable. The applications can auto-scale as per requirements & during the off-peak hours it can go back to its normal operations resulting in considerable cost savings.
- Microservices can be developed & deployed without reliance on other services.
- Microservices are preferred when organizations are moving from on-premise system to cloud based systems.
- Helps in developing highly distributed systems.
- When the functionality has to be highly available.
- For designing systems that require integration of business capability from disparate systems & models.
- It is preferred for incremental development & continuous deployment.



We help enterprises with microservices development to update their legacy IT systems with independent, agile, modular, and secure microservices application.

Our Microservices consulting service helps in building projects faster & with simple systems that enable quick repairs & upgrade with no downtime.

[Request a Free Consultation](#)

Benefits of Kubernetes for microservices

Kubernetes orchestration & management does containerization which supports microservices. We will help you decide if Kubernetes is a fit for your specific case, so, here we are discussing some high-level advantages of Kubernetes for microservice architecture.

Declarative configuration management & version control

YAML files are used to store Kubernetes configurations. These files can be version controlled with a source control software such as Git. These stored configurations can be applied to create or update resources.

No downtime

Kubernetes creates additional pods when releasing a new image, and it doesn't destroy the existing containers to make sure there is no downtime. After the new containers have been created and are functioning well, teams can roll out updates and do away with the old containers. In case, the new containers fail, the IT admins can immediately roll back the changes with no downtime.

Scalability

Kubernetes has full provision for scalability issues. Whenever there is a high demand or increase in load, Kubernetes increases the number of containers that run a microservice to prevent performance issues.

Load balancing the service exposing

Kubernetes use IP addresses or DNS to expose containers in Pods or groups of Pods. In this way, other microservices can easily consume those resources. It also allows IT admins to load balance logical POD groupings without putting in too much effort.

Self-healing

In the development project, it may so happen that a container fails or becomes unhealthy. In such a scenario, there is no need to panic for you as Kubernetes replaces it automatically to keep the state configuration at the desired level without disturbing the overall health of the application or its functioning.

Multi-cloud & hybrid cloud

Kubernetes allows IT teams to choose a platform on which to put the workload. It can be any cloud such as Google Cloud Platform, Microsoft Azure, and AWS. This not only gives a lot of options as far as the cloud is concerned but also helps in preventing vendor lock-in.

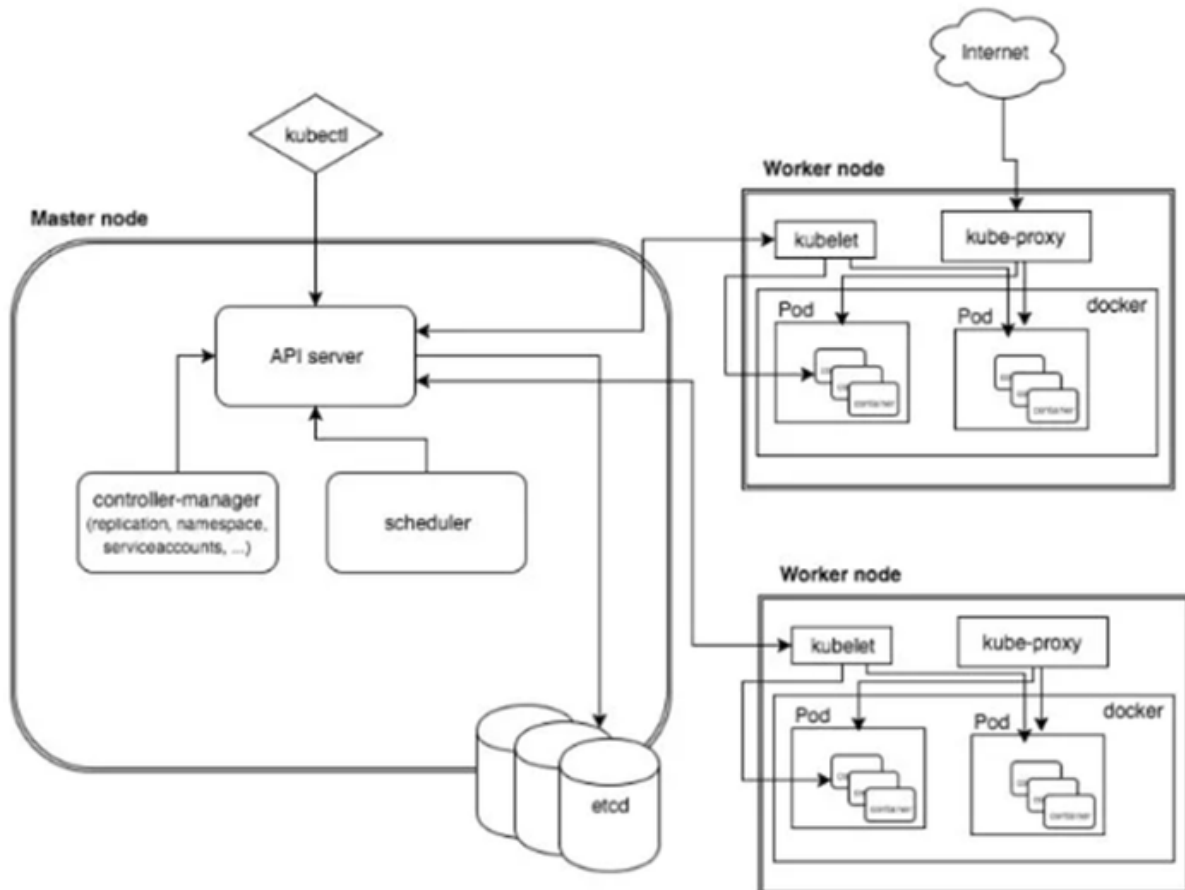
Protects information

Kubernetes can keep the secrets. It helps in preventing the exposure of sensitive information such as login credentials, in container images and maintains secret objects with help from etcd datastore.

ISmile Technologies prefers using Kubernetes for microservices because it ensures high availability as applications have no downtime & are always up & running. It offers high scalability – applications can be scaled-up or scaled-down as per your requirements. And it has fail-proof disaster recovery feature.

An overview of the Kubernetes architecture

High level Kubernetes Architecture diagram showing the cluster with a Master & two working nodes.



Kubernetes runs on a client-server architecture. You can have a multi-master setup (to ensure high availability), but by default, you have only one master server which functions as a controlling node and also as a point of contact. The master server has various components such as Kube-apiserver, a Kube-controller-manager, an etcd storage, a cloud controller manager, a DNS server, and a Kube-scheduler for Kubernetes services. The node server consists of a kubelet and a Kube-proxy on top of Docker.

Master Components

The following are the main components in the master node:

Etcd Cluster

It is a simple, distributed key value storage which is utilized for storing the Kubernetes cluster data like number of pods, states they are in, namespace etc., API objects, and service discovery details. You can access it only from the API server for security reasons. etcd takes help of the watchers to send notifications to the cluster about configuration changes. Notifications are nothing but the API requests on each etcd cluster to trigger information update in the node's storage.

Kube-apiserver

Kubernetes API server acts as an entity for central management which receives all the REST requests for any changes done to (pods, services, controllers/replication sets) and serves as the frontend to the clusters. Furthermore, it is the only component that talks to the etcd cluster, ensuring that data is stored in etcd and it agrees with the service details of the pods which have been deployed.

Cloud-controller-manager

This is responsible for doing the management of the controller processes which have dependencies on the underlying cloud provider. One example of the task which is handled by the cloud-controller-manager is when a controller needs to know if a node was terminated or it wants to set up routes, load balancers, or volumes in the cloud infrastructure.

Kube-controller-manager

It runs various controller processes in the background to do the regulation of the shared state of the cluster and do routine tasks. It runs a replication controller for controlling the number of replicas in the pod, it runs an endpoints controller to populate endpoint objects such as services, pods, and others. Whenever there is a change in the service configuration, like the image from which the pods are running is replaced, or the parameters in the configuration YAML file are changed, the change is spotted by the controller and it starts working towards the new desired state.

Kube-scheduler

It helps in scheduling the pods on the different nodes based on how the resources are utilized. A pod is a co-located container group within which our application processes are running. It determines the operational requirements of the services and schedules them on the best fit node. Let's suppose an application needs 2 GB memory & 4 CPU cores, then the pods for this application will be scheduled on a node having at least this much of resources. Whenever there is a need to schedule pods, the scheduler runs. Therefore, the scheduler must know what are the total available resources and the total resources allocated to existing workloads on each node.

Worker Components

The following are the main components in the worker node:

kubelet

This is the main service on a node. It regularly takes in new or modified pod specifications & ensures that pods & the containers within them are healthy and are running in the desired state. This component also reports to the master about the health of the host where it is running.

kube-proxy

It's a proxy service running on each worker node, and it deals with individual host subnetting and also exposes services to the outside world. It also does request forwarding to the right containers/pods across the different isolated networks inside a cluster.

What are monitoring challenges in Kubernetes?

Kubernetes monitoring is important for managing application performance, uptime of the services, and troubleshooting. Though Kubernetes simplifies the deployment of the application in containers & across clouds, it also introduces a new set of challenges in managing & securing applications.

Navigating dynamic containers

Container-based applications are dynamic in nature, and one will have to navigate these dynamics for monitoring which could be very challenging. The containers are very short-lived with a life span ranging from 10 seconds to 5 minutes. This leads to a lot of churn. When a container dies, everything inside it is gone, you cannot find anything by looking at logs. Containers ease operations as they help in packaging & isolating applications for deploying them everywhere. But then, they look like black boxes, and they are hard to troubleshoot.

Microservices architecture

Along with the increase in the infrastructure complexity, the applications are getting designed using microservices, and there the components population is increased by an order of magnitude. You can distribute each service across multiple instances & move the containers across your infrastructure as needed. You will have to ensure that Kubernetes is keeping all the service instances up & running, and for that monitoring the Kubernetes orchestration state is very important.

The explosion of the components & scale requirement

Upon the adoption of the containerized architecture, the number of components & the monitoring metrics grows exponentially. A monitoring system must keep up with this explosion. In legacy application architecture, the number of instances of each service component & their location are known. But that is not the case for containerized architecture. Multidimensional levels like cluster, node, service, etc. are added by the Kubernetes. The result is, different perspectives that have to be monitored can become staggering in volume.

Infrastructure complexity increases

Kubernetes being an orchestration platform enhances efficiency but also increases the complexity of monitoring & troubleshooting. Hence, one will have to find a new approach to gain the visibility one needs for the deployment & management of the cloud services.



Despite having better tools geared towards Kubernetes, it is very difficult to reach to the root cause of any given problem.

Logs are dependent on developer insights to determine what to log. Even if properly implemented, most tools on their own are incapable of answering the 'why' question. ISmile technologies with its deep expertise helps in overcoming monitoring Kubernetes challenges.

[Request a Free Consultation](#)

Key performance metrics that you should consider monitoring

KUBERNETES CLUSTER METRICS

Node resource utilization	This include some metrics such as bandwidth, CPU utilization, memory, and disk utilization. These metrics can be used to determine if number & size of cluster nodes should be increased or decreased.
The number of nodes	This metric will tell you that cloud provider is billing what resources & how the cluster is being used.
Running pods	If you are tracking the number of running pods, you will ne able to tell if the available nodes are enough to handle the current work load in case a node fails.

CONTAINER METRICS

Container CPU usage	Tells you how much CPU your containers are using against defined pod limits.
Container memory utilization	Learn how much memory your containers are using against the defined pod limits.
Network usage	Find out about sent & received data packets & how much bandwidth is being used.

KUBERNETES POD METRICS

Kubernetes metrics	This will help you to see how the orchestrator is handling & deploying an individual pod. You can monitor number of instances in a pod at any given time compared to the number of instances you expect (a lower number means the cluster has run out of the resources). In-progress deployments, health of your pods, and network data can also be tracked.
Pod container metrics	cAdvisor provides them & Heapster exposes them which also queries each nodes about the containers that are running.
Application specific metrics	The actual application itself develops them & are related to specific business rules. A database application may expose metrics on the state of an index, and relational statistics while an ecommerce application might provide the number of online customers & the revenue generated in a given timeframe. The application can be linked to a monitoring tool to track them mor closely.

WORKER STATE METRICS

Persistent Volumes (PV)	It is the storage resource specified on the cluster & provided as persistent storage for the pods that request it. PVs are associated with a certain pod during their lifecycle. When the pod no longer needs the PV, it is reclaimed. Monitoring PVs will tell you when the reclamation process fails, which indicates that something is not right with your PV.
-------------------------	---

Disk pressure	This occurs when too much disk space is used by a node or when it uses disk space too quickly. A configurable threshold defines disk pressure. Monitoring this metric will tell you, if the application truly needs additional disk space or if it prematurely fills up the disk unanticipatedly.
Crash loop	It happens when a pod starts, crashes, and then gets stuck in a loop of trying to restart without success. When a crash loop happens, the application fails to run. It is usually caused by crashing app within the pod, a pod misconfiguration, or a deployment issue. As the possibilities are many, debugging a crashed loop is quite tricky. However, you must learn about the crash immediately to quickly mitigate or implement emergency measures to keep the app available.
Jobs	These are components which are designed to temporarily run pods. Monitoring job failures tell you when your application is not accessible.

APPLICATION METRICS

Application availability	Tells you about the application's uptime & response time. This metric can help you to assess optimal UX & performance.
Application health & performance	Tells you about user experience issues like performance, latency, responsiveness, and others.

Kubernetes Monitoring Best Practices

Using levels of abstraction like pods & services, Kubernetes frees you from the worry that where are your apps running, or if they have resources in sufficient amount to run efficiently. But still, to ensure optimal performance, you must monitor your apps, the containers running on them, and the Kubernetes itself. Here are the important Kubernetes monitoring best practices –

Use Kubernetes DaemonSets

While running Kubernetes, you may like to run a single pod on all your nodes, such as, while running a monitoring process like the AppDynamics agent or FluentD, a data collector to collect logs. The AppDynamics agent that monitors containerized apps running inside pods, is used as a daemonset in every node. A daemonset is a workload object which ensures that a specific pod is running on every node, or on some subset of nodes. By using daemonset, we tell Kubernetes to ensure there is one instance of a pod on every node.

Tags & labels are important

As Kubernetes manages container orchestration, labels become important for monitoring, because they are the only way to interact with pods & containers. To enhance the usefulness of your metrics, it's important to define your labels with a logical & consistent schema.

Ensure you are monitoring important metrics

As per Kubernetes, several types of metrics should be closely tracked. They are running pods, resource metrics, container-native metrics, and application metrics.

Use monitoring system with service discovery

Since Kubernetes schedules apps according to scheduling policy, you might be unaware where your apps are running, but any ways you need to monitor them. You may require a monitoring system with service discovery which adapts metric collection to moving containers. This way you can continuously monitor your apps without any interruption.



Even if you have a complex & multi-layered system, there should be a consistent monitoring strategy throughout the entire cluster.

This implies, that metric collection must happen in the same way no matter where your nodes are deployed. If you lack the expertise in Kubernetes monitoring as that is not your core competency, rely on ISmile Technologies for the same.

[Request a Free Consultation](#)

Tools for Monitoring Microservices

On Kubernetes

Kubernetes is a complicated environment, and containerized apps can be distributed across multiple environments. The monitoring system should be able to collect metrics from across distributed environments, and deals with containerized resources' ephemeral nature. Given below are the important tools for the containerized environment.

Prometheus

It is a popular monitoring tool developed by SoundCloud. It provides alerts & detailed analysis of Kubernetes & Docker. It has been designed for monitoring apps & container-based microservices running at scale.

Kiali

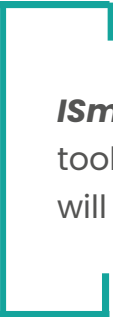
Kiali provides a UI for managing service mesh architecture. You get dashboards for visualization and operate the mesh with the capability to configure & validate. Kiali offers detailed metrics to visualize the health of your mesh, and also integrates with other monitoring tools.

Grafana

It is an open-source platform for visualizing metrics & analytics. It has four built-in dashboards for Cluster, Nodes, Pods, and deployment. It allows creating data-rich dashboards using the info sources from Prometheus.

Jaeger

It's an open-source tracking system developed by Uber. You can use it to monitor & troubleshoot distributed transactions. This tool is used to address software issues related to distributed context propagation and latency optimization.




ISmile Technologies has deep expertise in using the Kubernetes monitoring tools for best performance. We know which tool to use in which situation. We will use the right tool that is best suited as per your specific requirements.

About ISmile Technologies



ISmile Technologies, delivering automation-enabled Intelligent Cloud Solutions and Managed IT services, is your Multi-Cloud Technology Advisor & key Implementation Partner. Operating globally, we leverage disruptive technologies alongside deep expertise to deliver business-specific Cloud Solutions. We maximize impact at an unparalleled value and securely accelerate business agility while infusing competitive excellence.

 [+1 \(844\) 845-9236](tel:+18448459236)

 <https://www.ismiletechnologies.com>

 service@ISmileTechnologies.com

 [501 S Weber Rd Unit 108, Bolingbrook, IL 60490](#)