A Complete Introduction to Monitoring Kubernetes with New Relic

The fundamentals you need to know to effectively monitor Kubernetes deployments
Introduction

Before Kubernetes took over the world, cluster administrators, DevOps engineers, application developers, and operations teams had to perform many manual tasks in order to schedule, deploy, and manage their containerized applications. The rise of the Kubernetes container orchestration platform has altered many of these responsibilities.

Kubernetes makes it easy to deploy and operate applications in a microservice architecture. It does so by creating an abstraction layer on top of a group of hosts, so that development teams can deploy their applications and let Kubernetes manage:

- Controlling resource consumption by application or team
- Evenly spreading application load across a host infrastructure
- Automatically load balancing requests across the different instances of an application
- Monitoring resource consumption and resource limits to automatically stop applications from consuming too many resources and restarting the applications again
- Moving an application instance from one host to another if there is a shortage of resources in a host, or if the host dies
- Automatically leveraging additional resources made available when a new host is added to the cluster
- Easily performing canary deployments and rollback

But such capabilities also give teams new things to worry about. For example:

- There are a lot more layers to monitor.
- The ephemeral and dynamic nature of Kubernetes makes it a lot more complex to troubleshoot.
- Automatic scheduling of pods can cause capacity issues, especially if you’re not monitoring resource availability.

In effect, while Kubernetes solves old problems, it can also create new ones. Specifically, adopting containers and container orchestration requires teams to rethink and adapt their monitoring strategies to account for the new infrastructure layers introduced in a distributed Kubernetes environment.

With that in mind, we designed this guide to highlight the fundamentals of what you need to know to effectively monitor Kubernetes deployments with New Relic. This guide outlines some best practices for monitoring Kubernetes in general, and provides detailed advice for how to do so with the New Relic platform.

Whether you’re a cluster admin, an application developer, an infrastructure engineer, or DevOps practitioner working on the Kubernetes platform, by the end of this guide, you should be able to use New Relic to monitor the health and capacity of Kubernetes components and resources, correlate events in Kubernetes with contextual insights to help you troubleshoot issues, understand how to monitor applications running in your cluster, and know how to track end-user experience from those apps.
Monitoring Kubernetes with New Relic

In addition to providing visibility into operational data—such as the number of resources used and namespaces per cluster and per pod—an important part of monitoring Kubernetes is having the ability to see the relationships between objects in a cluster using Kubernetes’ built-in labeling system.

New Relic integrates with Kubernetes in a number of ways:

• **Amazon Elastic Container Service for Kubernetes (Amazon EKS):** Amazon EKS provides Kubernetes as a managed service on AWS. It helps make it easier to deploy, manage, and scale containerized applications on Kubernetes.

• **Google Kubernetes Engine (GKE):** GKE provides an environment for deploying, managing, and scaling your containerized applications using Google-supplied infrastructure.

• **Microsoft Azure Kubernetes Service (AKS):** AKS manages your hosted Kubernetes environment, making it easier to deploy and manage containerized applications without container orchestration expertise. It also eliminates the burden of ongoing operations and maintenance by provisioning, upgrading, and scaling resources on demand, without taking your applications offline.

• **Red Hat OpenShift:** OpenShift provides developers with an integrated development environment (IDE) for building and deploying Docker-formatted containers, and then managing them with Kubernetes.

• **Pivotal Container Service (PKS):** Available as part of Pivotal Cloud Foundry or as a standalone product, PKS provides the infrastructure and resources to reliably deploy and run containerized workloads across private and public clouds.

Getting started

To get started monitoring Kubernetes with New Relic, you’ll need to activate the Kubernetes integration by deploying the newrelic-infra agent onto your Kubernetes cluster. The New Relic Kubernetes integration brings in system-level metrics, allowing you to view, troubleshoot, and alert on the most important parts of your cluster. Use the integration’s out-of-the-box dashboard to inspect a single container, or scale up to see large, multi-cluster deployments across different Kubernetes entities, including nodes, pods, namespaces, and containers.

1. New Relic uses **kube-state-metrics**—a simple service that listens to the Kubernetes API server and generates metrics—to gather information about the state of Kubernetes objects. Install kube-state-metrics in your cluster:

   ```bash
   curl -o kubectl-state-metrics-1.7.2.zip https://codeload.github.com/kubernetes/kube-state-metrics/zip/release-1.7.2 && unzip kubectl-state-metrics-1.7.2.zip && oc apply -f kube-state-metrics-release-1.7.2/kubernetes
   ```

2. Download the Kubernetes integration configuration file:

   ```bash
   curl -o
   ```

3. In the configuration file, add your **New Relic license key** and a cluster name to identify your Kubernetes cluster. Both values are required. Be sure to update `<YOUR_LICENSE_KEY>` with your license key and `<YOUR_CLUSTER_NAME>` with the name of your cluster.
env:
- name: NRIA_LICENSE_KEY
  value: <YOUR_LICENSE_KEY>
- name: CLUSTER_NAME
  value: <YOUR_CLUSTER_NAME>

4. Add any additional configuration, as documented in the Kubernetes integration instructions.

5. To confirm that kube-state-metrics is installed, run this command:
   
   kubectl get pods --all-namespaces | grep kube-state-metrics

6. To create the DaemonSet, run this command:
   
   kubectl create -f newrelic-infrastructure-k8s-latest.yaml

7. Confirm that the DaemonSet has been created successfully by looking for newrelic-infra in the results generated by this command:
   
   kubectl get daemons

8. Go to one.newrelic.com, and select the Kubernetes cluster explorer launcher.

Exploring clusters with the Kubernetes cluster explorer

New Relic’s Kubernetes cluster explorer provides a multi-dimensional representation of a Kubernetes cluster from which you can explore your namespaces, deployments, nodes, pods, containers, and applications. With the cluster explorer, you will be able to easily retrieve the data and metadata of these elements, and understand how they are related.

From the Kubernetes cluster explorer, you can:

1. Select the cluster you want to explore
2. Filter by namespace or deployment
3. Select specific pods or nodes for status details

The cluster explorer has two main parts:

1. A visual display of the status of a cluster, up to 24 nodes. Within the visual display, the cluster explorer shows the nodes that have the most issues in a series of four concentric rings:
   • The outer ring shows the nodes of the cluster, with each node displaying CPU, memory, and storage performance metrics.
   • The next innermost ring displays the distribution and status of the non-alerting pods associated with that node.
   • The third innermost ring displays the pods on alert and that may have health issues even if they are still running.
   • Finally, the innermost ring displays pods that are pending or that Kubernetes is unable to run.

2. You can select any pod to see its details, such as its namespace, deployment, containers, alert status, CPU usage, memory usage, and more.

3. The cluster explorer node table displays all the nodes of the selected cluster/namespace/deployment, and can be sorted according to node name, node status, pod, pod status, container, CPU% vs. Limit and MEM% vs. Limit.
Benefits of monitoring with the cluster explorer

The cluster explorer expands the Kubernetes monitoring capabilities already built into the New Relic platform. Use the cluster explorer’s advanced capabilities to filter, sort, and search for Kubernetes entities, so you can better understand the relationships and dependencies within an environment. The default data visualizations of your cluster provides a fast and intuitive path to getting answers and understanding their Kubernetes environments, so you can contain the complexity associated with running Kubernetes at scale.

When your team adopts cluster explorer, you can expect improved performance and consistency and quicker resolutions when troubleshooting errors. New Relic can help ensure that your clusters are running as expected and quickly detect performance issues within your cluster—even before they have a noticeable impact on your customers.
We recommend that Kubernetes observability begins with these five practices:

1. Visualizing services
2. Monitoring health and capacity
3. Correlating Kubernetes events
4. Understanding APM correlation
5. Investigating end-user experience

Visualizing services

When working in a Kubernetes environment, it can be difficult to untangle the dependencies between applications and infrastructure, and navigate all of the entities—containers, pods, nodes, deployments, namespaces, and so on—that may be involved in a troubleshooting effort. You need to observe performance and dependencies across the entire Kubernetes environment.

You should be able to visualize key parts of your services, including:

- The structure of your application and its dependencies
- The interactions between various microservices, even those that are intermingled across your machine cluster

How New Relic helps

The cluster explorer provides a multi-dimensional representation of a Kubernetes cluster that allows teams to drill down into Kubernetes data and metadata in a high-fidelity, curated UI that simplifies complex environments. Teams can use cluster explorer to more quickly troubleshoot failures, bottlenecks, and other abnormal behavior across their Kubernetes environments.

Suggested alerting

When deploying the New Relic Kubernetes integration for the first time in an account, a default set of alert conditions is deployed to the account. The alert policy is configured without a notification channel to avoid unwanted alerts.

You can customize the alert conditions' thresholds to your environment and update the alert policy to send notifications. For more, see the New Relic Infrastructure alerts documentation.

Monitoring cluster health and capacity

Kubernetes environments vary from deployment to deployment, but they have all a handful of key components, resources, and potential errors in common. The following sections introduce best practices, including tips for how to use New Relic and alerts, for monitoring the health and capacity of any Kubernetes environment:

- Track cluster resource usage
- Monitor node resource consumption
- Monitor for missing pods
- Find pods that aren’t running
- Troubleshoot container restarts
- Track container resource usage
- Monitor storage volumes
- Monitor the control plane: etcd, the API server, the scheduler, and the controller manager

Track cluster resource usage

When you administer clusters, you need enough usable resources in your cluster to avoid running into issues when scheduling pods or deploying containers. If you don’t have
enough capacity to meet the minimum resource requirements of all your containers, scale up your nodes' capacity or add more nodes to distribute the workload.

You should know:
- What percentage of cluster resources you’re using at any given time
- If your clusters are over- or under-provisioned
- How much demand have you’ve placed on your systems

How New Relic helps

Our Kubernetes integration monitors and tracks aggregated core and memory usage across all nodes in your cluster. This allows you to meet resource requirements for optimal application performance.

### Resources Used

<table>
<thead>
<tr>
<th>Since 60 minutes ago until 1 minute ago</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cores: 1.46</td>
</tr>
</tbody>
</table>

### Suggested alerting

Set alerts on the cores and memory usage of the hosts in your cluster.

Monitor node resource consumption

Beyond simply keeping track of nodes in your cluster, you need to monitor the CPU, memory, and disk usage for Kubernetes nodes (workers and masters) to ensure all nodes in your cluster are healthy.

Use this data to ensure:
- You have enough nodes in your cluster
- The resource allocations to existing nodes is sufficient for deployed applications
- You’re not hitting any resource limits

How New Relic helps

New Relic tracks resource consumption (used cores and memory) for each Kubernetes node. That lets you track the number of network requests sent across containers on different nodes within a distributed service. You can also track resource metrics for all containers on a specific node—regardless of which service they belong to.

Always ensure your current deployment has sufficient resources to scale. You don’t want new node deployments blocked by a lack of resources.

### Suggested alerting

Set alerts so you’ll be notified if hosts stop reporting or if a node’s CPU or memory usage drops below a desired threshold.

Monitor for missing pods

From time to time, you may find your cluster is missing a pod. A pod can go missing if the engineers did not provide sufficient resources when they scheduled it. The pod may have never started; it’s in a restart loop; or it went missing because of an error in its configuration.

To make sure Kubernetes does its job properly, you need to confirm the health and availability of pod deployments. A pod deployment defines the number of instances that need to be present for each pod, including backup instances. (In Kubernetes, this is referred to as a **Replica-Set**). Sometimes the number of active pods is not specified in the Replicas field on each deployment. Even if they are specified, Kubernetes may determine if it can run another instance based on resources the administrator has defined.

```text
forbidden: exceeded quota: compute-resources, requested: pods=1, used: pods=1, limited: pods=1
```
How New Relic helps

New Relic makes it easier to avoid this issue by knowing the resource limitations of the cluster.

If you don’t have enough resources to schedule a pod, add more container instances to the cluster or exchange a container instance for one with the appropriate amount of resources. In general, you can use the New Relic Kubernetes integration to monitor for missing pods and immediately identify deployments that require attention. This often creates an opportunity to resolve resource or configuration issues before they affect application availability or performance.

Suggested alerting

Set an alert for when a deployment’s missing pods value rises above a certain threshold for a certain period. If the number of available pods for a deployment falls below the number of pods you specified when you created the deployment, the alert will trigger. The alert will be applied to each deployment that matches the filters you set.

Find pods that aren’t running

Kubernetes dynamically schedules pods into the cluster, but if you have resource issues or configuration errors, scheduling will likely fail. If a pod isn’t running or even scheduled, that means there’s an issue with either the pod or the cluster, or with your entire Kubernetes deployment.

When you see that pods aren’t running, you’ll want to know:
- If there are any pods in a restart loop
- How often are requests failing
- If there are resource issues or configuration errors

How New Relic helps

As noted, if you have resource issues or configuration errors, Kubernetes may not be able to schedule the pods. In such cases, you want to check the health of your deployments, and identify configuration errors or resource issues.

With the New Relic Infrastructure Kubernetes integration, you can use default deployment data to discover and track pods that may not be running and sort them by cluster and namespace.

Additionally, you can analyze further root causes of terminated pods, with the terminated pods metric. For example, if a pod is terminated because its application memory has reached the memory limit set on the containers, it will be killed by the out of memory (OOM) killer. In such cases, New Relic will expose the reason for pod termination.

Suggested alerting

Set alerts on the status of your pods. Alerts should trigger when a pod has a status of “Failed,” “Pending,” or “Unknown” for the period of time you specify.
Troubleshoot container restarts

In normal conditions, containers should not restart. Container restarts are a sign that you’re likely hitting a memory limit in your containers. Restarts can also indicate an issue with either the container itself or its host. Additionally, because of the way Kubernetes schedules containers, it can be difficult to troubleshoot container resource issues since Kubernetes will restart—or kill—containers when they hit their limits.

Monitoring container restarts helps you understand:

- If any containers are in a restart loop
- How many container restarts occurred in X amount of time
- Why containers are restarting

How New Relic helps

A running count of container restarts is part of the default container data New Relic gathers with the Kubernetes integration.

When monitoring container resource usage, you’ll want to know:

- If your containers are hitting resource limits and affecting the performance of their applications
- If there are spikes in resource consumption
- If there is a pattern to the distribution of errors per container

How New Relic helps

First, identify the minimum amount of CPU and memory a container requires to run—which needs to be guaranteed by the cluster—and monitor those resources with New Relic.

Second, monitor container resource limits. These are the maximum amounts of resources that the container will be allowed to consume. In Kubernetes, resource limits are unbounded by default.

Suggested alerting

This an optional alerting scenario. Kubernetes automatically restarts containers, and setting up an alert will give you an immediate, useful notification, but don’t let container restarts interrupt your sleep.

Track container resource usage

Monitoring container resource usage helps you ensure that their containers and applications remain healthy. For example, if a container hits its limit for memory usage, the `kubelet` agent might kill it.
This type of monitoring can help proactively resolve resource usage issues before they affect your application.

**Suggested alerting**

Set alerts on container CPU and memory usage and on limits for those metrics.

**Monitor storage volumes**

One thing you definitely want to avoid when running applications on Kubernetes is data loss or application crashes because you’ve run out of space on your storage volumes.

In Kubernetes, storage volumes are allocated to pods and possess the same lifecycle as the pod; in other words, if a container is restarted, the volume is unaffected, but if a pod is terminated, the volume is destroyed with the pod. This works well for stateless applications or batch processing where the data doesn’t outlive a transaction.

Persistent volumes, on the other hand, are used for stateful applications and when data must be preserved beyond the lifespan of a pod. Persistent volumes are well suited for database instances or messaging queues.

To monitor Kubernetes volumes, you’ll want to:

- Ensure your application has enough disk space so pods don’t run out of space
- View volume usage and adjust either the amount of data generated by the application or the size of the volume according to usage
- Identify persistent volumes and apply a different alert threshold or notification for these volumes, which likely hold important application data

**How New Relic helps**

You’ll want to monitor and alert on disk volume issues, especially in the context of persistent volumes where data must be made available to stateful applications persistently, so that it’s not destroyed if a specific pod is rescheduled or recreated—like when a container image is updated to a new version, for example.

With Kubernetes volume monitoring in New Relic, you can monitor your volumes, and set alerts on them so that you get informed as soon as a volume reaches a certain threshold—a proactive approach to limiting issues with application performance or availability.

**Suggested alerting**

Set alerts on available bytes, capacity, and node usage in your cluster.

**Monitor etcd**

This is where the current and desired state of your cluster is stored, including information about all pods, deployments, services, secrets, etc. This is the only place where Kubernetes stores its information.

To monitor etcd, you’ll want to track:

- Leader existence and change rate
- Committed, applied, pending, and failed proposals
- gRPC performance

**Suggested alerting**

Set alerts to be notified if pending or failed proposals reach inappropriate thresholds.

**Monitor the API server**

The central RESTful HTTP API handles all requests coming from users, nodes, control plane components, and automation. The API server handles authentication, authorization, validation of all objects, and is responsible for storing said objects in etcd. It’s the only component that talks with etcd.
To monitor the API server, you’ll want to track:
- Rate and number of HTTP requests
- Rate and number of apiserver requests

**Suggested alerting**
Set alerts to trigger if the rate or number of HTTP requests crosses a desired threshold.

**Monitor the scheduler**
The scheduler is responsible for assigning newly created pods to worker nodes capable of running said pods. To do so, the scheduler updates pod definitions through the API server.

The scheduler takes several factors into consideration when selecting a worker node, such as requested CPU/memory vs. what’s available on the node.

To monitor the scheduler, you’ll want to track:
- Rate, number, and latency of HTTP requests
- Scheduling latency
- Scheduling attempts by result
- End-to-end scheduling latency (sum of scheduling)

**Suggested alerting**
Set alerts to trigger if the rate or number of HTTP requests crosses a desired threshold.

**Correlating Kubernetes events with cluster health**
To speed troubleshooting and issue resolution, you can correlate the health status of your cluster and other objects with Kubernetes events. If you run complex Kubernetes environments or don’t have command-line access to your cluster, Kubernetes events provide the insights you need to understand what’s happening inside your cluster.

For example, let’s say you have a pod that doesn’t get properly scheduled and won’t start because the node it’s assigned to wasn’t allocated enough memory. In this case, the node can’t accommodate the pod, so the pod stays in pending status, but no other metrics or metadata provide deeper insight into the issue. With Kubernetes events, you’d get a clear message:

```
FailedScheduling [...] 0 nodes are available: Insufficient memory
```

If you’re managing a Kubernetes deployment, or developing on top of one, you need:
- Visibility into the Kubernetes events for a cluster
- Visibility into the Kubernetes events related to a specific object, such as a pod or a node
- Alerting on Kubernetes events
How New Relic helps

As shown in the example, Kubernetes events provide additional contextual information that is not provided by metrics and metadata. When using Kubernetes events alongside the cluster explorer, you get a holistic view of the health of your platform.

When troubleshooting an issue in a pod, Kubernetes events more readily point toward root causes with useful context. New Relic also layers each event with useful details, so you can determine if an event affects several pods or nodes in cluster, such as when a ReplicaSet is scaled or when a StatefulSet creates a new pod.

You can query Kubernetes events with New Relic chart builder, or view them from the cluster explorer.

Suggested alerting

Set alerts for specific types of events on objects and resources in your cluster. For example, New Relic can send alerts if an expected autoscaling action doesn’t occur.

Integrating with APM data

A key benefit of Kubernetes is that it decouples your application and its business logic from the specific details of its runtime environment. That means if you ever have to shift the underlying infrastructure to a new Linux version, for example, you won’t have to completely rewrite the application code.

When you’re monitoring applications that are managed by an orchestration layer, being able to relate an application error trace, for instance, to the container, pod, or the host it’s running in can be very useful for debugging or troubleshooting.

At the application layer, you need to monitor the performance and availability of applications running inside your Kubernetes cluster. You do that by tracking such metrics as request rate, throughput, and error rate.

New Relic APM lets you add custom attributes, and that metadata is available in transaction traces gathered from your application. You can create custom attributes to collect information about the exact Kubernetes node, pod, or namespace where a transaction occurred.

To get started monitoring applications running in Kubernetes, you will need to instrument your application with the Kubernetes Downward API. We’ve created a sample app to demonstrate how this works in a Node.js application—fork this repo for your own use. (Our Monitoring Application Performance in Kubernetes blog post explains how to add this type of Kubernetes metadata to APM-monitored application transactions.)

The following sections introduce key parts of your Kubernetes-hosted applications to monitor:

- Application health
- Prevent errors

Monitor application health

When you run applications in Kubernetes, the containers the apps run in often move around throughout your cluster as instances scale up or down. This scheduling happens automatically in Kubernetes but could affect your application’s performance or availability. If you’re an application developer, being able to correlate Kubernetes objects to applications is important for debugging and troubleshooting.

You’ll want to know:

- What applications are associated with which cluster
- How many transactions are happening within a given pod
- What’s the service latency or throughput for a production application running in a pod
How New Relic helps

To monitor transaction traces in Kubernetes, you need a code-centric view of your applications. You need to correlate applications with the container, pod, or host it’s running in. You also need to identify pod-specific performance issues for any application’s workload.

Knowing the names of the pod and node where the error occurred can speed your troubleshooting. Visibility into transaction traces will quickly highlight any abnormalities in your Kubernetes-hosted application.

Additionally, distributed tracing allows you to inspect the distributed traces captured for any application running in your cluster. If you click on an individual span in a distributed trace, you can quickly see the relevant Kubernetes attributes for that application; for example, you can find out which pod, cluster, and deployment an individual span belongs to.

New Relic distributed tracing provides automated anomaly detection to identify slow spans and bottlenecks. You should also set alerts on key transactions and communications with third-party APIs.

To learn about how to gain visibility into transaction traces in Kubernetes, see the blog post, Monitoring Application Performance in Kubernetes.

Suggested alerting

You’ll want to set up alerts for all applications running in production. Specifically, you’ll want to alert on API service requests, transactions, service latencies, uptime, and throughput, sending alerts when any of these metrics fall below the thresholds you define.

Prevent errors

If a single pod or particular pod IP starts failing or throwing errors, you need to troubleshoot before those errors harm your cluster or application. When something goes wrong, zero in on root causes as quickly as possible.

You’ll want to know:

• In which namespace/host/pod did a transaction fail
• If your app is performing as expected in all pods
• The performance of application X running on pod Y
How New Relic helps

With New Relic, you can get a code-centric view of the applications running inside your cluster, monitor your Kubernetes-hosted applications for performance outliers, and track down any individual errors.

The Monitoring Application Performance in Kubernetes blog post can help you pinpoint applications that might be causing performance issues in your cluster.

Suggested alerting

Set alerts to track error rates for any applications running in production environments in Kubernetes.

Integrating with Prometheus metrics

Prometheus is an open-source toolkit that provides monitoring and alerting for services and applications running in containers, and it’s widely used to collect metrics data from Kubernetes environments. In fact, Prometheus’ scheme for exposing metrics has become the de facto standard for Kubernetes.

Prometheus uses a pull-based system to pull multidimensional timeseries metrics from services over HTTP endpoints, instead of relying on services to push metrics out. Because of this pull-based system, third-parties, like New Relic, can build integrations that work with Prometheus’ metric exporters to gather that valuable data for storage and visualization.

How New Relic helps

The New Relic Prometheus OpenMetrics integration collects telemetry data from the many services (such as Traefik, Envoy, and etcd) that expose metrics in a format compatible with Prometheus. In fact, with this integration you’ll be able to monitor key aspects of your Kubernetes environments, such as, etcd performance and health metrics, Kubernetes Horizontal Pod Autoscaler (HPA) capacity, and node readiness.

The integration supports both Docker and Kubernetes, using Prometheus version 2.

Installing the Prometheus OpenMetrics integration within a Kubernetes cluster is as easy as changing two variables in a manifest and deploying it in the cluster:

1. Download the integration manifest YAML file:
   ```
   curl -O https://download.newrelic.com/infrastructure_agent/integrations/kubernetes/nri-prometheus-latest.yaml
   ```

2. Edit the `nri-prometheus-latest.yaml` manifest file, and add a cluster name to identify your Kubernetes cluster (required) and your New Relic license key (required).
   ```
   env: - name: LICENSE_KEY
     value: ""
   ```
   config.yaml: |
     cluster_name: ""

3. Deploy the integration in your Kubernetes cluster:
   ```
   kubectl apply -f nri-prometheus-latest.yaml
   ```

Once you’ve installed the integration for Docker or Kubernetes, you can begin building queries to track and visualize your Prometheus data in New Relic. When troubleshooting issues in your Kubernetes clusters, the metrics collected by this integration are accessible alongside those gathered from New Relic APM and New Relic Infrastructure.

See the New Relic docs for more on compatibility and requirements, installation options, data limits, configuration, metric queries, troubleshooting, metric transformation, and more.

Using Prometheus data in New Relic

There are any number of ways to use Prometheus data in New Relic, but consider the following use cases:
Monitoring etcd

Etcd is a key-value data store that’s essential for running Kubernetes clusters. Prometheus pulls metrics from etcd, so to ensure your clusters are healthy, you can use the Prometheus OpenMetrics Integration to monitor etcd server, disk, and network metrics such as:

- etcd_server_has_leader
- etcd_server_proposals_failed_total
- etcd_network_peer_sent_bytes_total
- etcd_disk_wal_fsync_duration_seconds

Kubernetes Horizontal Pod Autoscaler (HPA)

HPA automatically scales up a Kubernetes deployment based on user-configured limits. After installing the Prometheus OpenMetrics Integration, you can use the following query in the New Relic One chart builder (or New Relic Insights) to build a dashboard widget and monitor the remaining HPA capacity.

```
FROM Metric select
latest(kube_hpa_status_current_replicas),
latest(kube_hpa_spec_max_replicas)
where clusterName = '<YOUR CLUSTER NAME>'
facet hpa
```

Node readiness

Set alerts to track error rates for any applications running in production environments in Kubernetes.

In Kubernetes, a node is marked ready when it can accept workloads (pods). If a node is having issues, Kubernetes will label it as “not ready.” To create an alert condition for this, using the integration, use the following query:

```
FROM Metric select latest(kube_node_status_condition) where condition='Ready' and
status = 'true' and clusterName = '<YOUR CLUSTER NAME>'
facet nodeName
```

Monitoring logs in context

With New Relic Logs, you can bring all your Kubernetes data together into one view.

Available in the Kubernetes cluster explorer, Logs provides a near-instant search with full contextual log information. And when you configure logs in context, you can correlate those log messages with application, infrastructure, Kubernetes, and event data.

For example, you can easily correlate application log messages with a related distributed trace in New Relic APM. New Relic appends trace IDs to the corresponding application logs and automatically filters these logs from the distributed trace UIs. Bringing all of this data together in a single tool, you'll more quickly get to the root cause of issues—narrowing down from all of your logs, to finding the exact log lines that you need to identify and resolve a problem.

This gives you end-to-end visibility, as well as a level of depth and detail that simply isn't available when you work with siloed sources of log data.

How New Relic helps

New Relic offers a Fluent Bit output plugin to enable New Relic Logs for Kubernetes to collect cluster log data. After downloading the plugin, you can deploy it as a Helm chart or deploy it manually via the command line.

Here's how to deploy the plugin manually:

1. `Clone or download the New Relic kubernetes-logging project from GitHub.`
2. In the `new-relic-fluent-plugin.yml`, edit the `env:` section to replace the placeholder value `<LICENSE_KEY>` with your New Relic license key.

   - `name: LICENSE_KEY`
   - `value: <YOUR_LICENSE_KEY>`

3. Load the logging plugin into your k8s environment:

   `kubectl apply -f` .

4. Go to one.newrelic.com, and select the Logs launcher.

5. In the query field, add the following:

   `plugin source "kubernetes"`

---

### How New Relic Helps

For example, when developers first migrate to Kubernetes, they can set up a pre-migration baseline to compare their front-end application's average load time before and after the migration. Developers can use the same strategies detailed in the [Monitoring Application Performance in Kubernetes](https://newrelic.com/blog) blog post to gain insight into key indicators such as response time and errors for mobile application and browser performance. It's also imperative to monitor load time and availability to ensure customer satisfaction.

**New Relic Browser** and **New Relic Mobile** are built to give you that crucial view into your users’ experiences.

---

### Investigating End-User Experience

If you order an item from a delivery service, and it arrives at your house broken or late, do you really care what part of the delivery process failed? Whether it was the fault of the manufacturer, distributor, or the delivery service, the end result is equally frustrating.

The same logic applies to companies hosting apps in Kubernetes: If a customer navigates to their website and a page doesn't load or takes too long, the customer isn't interested in the technical reasons why. That's why it's not enough to track your own systems' performance—it's also essential to monitor the front-end performance of your applications to understand what your customers are actually experiencing.

Even though your application is running in Kubernetes, you can still identify and track the key indicators of customer experience, and clarify how the mobile or browser performance of their application is affecting business.

Additionally, developers and operators both need to understand the availability of any Kubernetes-hosted service, often from locations all around the world. That's why **New Relic Synthetics** is designed to track application availability and performance from a wide variety of locations.

New Relic brings together business-level information and performance data into a single pane of glass. That helps customers from development, operations, product, and customer support teams identify potential areas of improvement in the product and find better ways to debug errors that may be affecting specific customers.
**Suggested alerting**

New Relic Mobile alerts:
- Mobile network error rate and response time

New Relic Browser alerts:
- Browser session count drop to indicate availability issues
- Browser javascript error rate or error count
- Browser interaction duration

New Relic Synthetics alerts:
- Synthetics error rate or error count
- Synthetics response times
- Synthetics ping checks

---

**Scaling Kubernetes with Success: A Real-World Example**

As you begin your Kubernetes journey, it may help to understand how another organization’s approach to monitoring enabled them to be successful.

Since its inception in 1997, Phlexglobal has been helping life sciences companies streamline clinical trials by enabling them to take charge of their trial master file (TMF)—i.e., the data repository for all documentation related to a clinical trial.

**Challenges**

- Scaling, monitoring and managing the performance of Phlexglobal’s trial master file (TMF) platform while migrating workloads onto Kubernetes.
- Ensuring this platform is well maintained is critical not only to prove key compliance with industry and government regulations, but also to facilitate and improve collaboration among a clinical trial’s many partners.

**Solution**

The team was searching for a tool to facilitate an agile organization with specific needs from the development and IT operations teams, so Phlexglobal looked to New Relic to get a system-wide view into performance that would enable proactive monitoring. Explore their full monitoring story to appreciate the impact and results.