Maximizing infrastructure performance in Kubernetes environments

In your organization, you use Kubernetes for container orchestration. It’s essential for you to monitor and maximize the performance of your Kubernetes environments. However, microservices environments can present challenges that monolithic environments do not, since requests traverse between different layers of the stack and across multiple services. You need to monitor these interrelated layers, while efficiently correlating application and infrastructure behavior to streamline troubleshooting.

Open-source monitoring tools exist, and while these can supply high-level performance metrics to help you understand your deployment, they can lack the sophisticated performance analytics capabilities and persistence for historical trends to help maximize performance and cost for your deployments.

Data and add-ons required

- Splunk OpenTelemetry collector

How to use Splunk software for this use case

Use the Splunk OpenTelemetry collector to identify performance issues

To maximize performance for Kubernetes, understanding both your microservices infrastructure needs and metrics is important. Optimizing your pods, which run on nodes, then grouped together by clusters, is key to a successful implementation. You’ll want to make sure that you are keeping an eye on metrics that include which nodes pods are scheduled to, their CPU and memory consumption, and resource limits for each of your workloads.

The Splunk OpenTelemetry Collector provides integrated collection/forwarding for all Kubernetes telemetry types and is deployed using a Helm chart for Kubernetes.

1. Navigate to Infrastructure and click Kubernetes to find the essential metrics to monitor your Kubernetes deployment, starting with the Kubernetes cluster map. This gives you an overview of cluster resource usage, node, and pod availability, and health, as well as if you have any missing or failed pods.
2. Access the Kubernetes Analyzer to view suggested filters for your deployment. You can access filters related to high memory nodes and nodes containing pods found not ready.
3. Jump to the namespaces that may have possible deployment issues. When selecting a namespace affected with high memory utilization, the Kubernetes cluster map filters to the cluster affected by high memory utilization and only shows the nodes and pods in this state, allowing you to drill down and see what’s going on with each node.
4. Once you have identified the affected node in the cluster, filter through the Kubernetes navigator to learn more about the workload, node, and pod details:

The information provided in Splunk Lantern is intended for informational and educational purposes only. All information is provided in good faith, however, Splunk disclaims any and all representations and warranties, express and implied, regarding the information provided, including without limitation any warranties and representations regarding the completeness, adequacy or accuracy of the information. You agree to take full responsibility for the results arising from the use of the information provided.
Node Details allows you to filter by the affected node and display detailed charts.

Workflow information allows you to filter by the affected Workload Name and discover other pods deployed with this workload.

Pod Detail allows you to view detailed information about each pod running the workload. Search for the Workload (App) name to see critical metrics in this area, for example to be able to view whether the workload was deployed with no resource limits.

---

**Applying resource limits to fix performance issues**

Resource limits define a hard limit of resources (CPU and Memory) a workload can take when deployed, making sure the process doesn't consume all resources in the node.

You can set resource limits within the Kubernetes deployment file. In most cases, you'll need to define a request and limit for most applications.

- Requests define guaranteed resources the application must have when deployed with Kubernetes. Kubernetes only schedules (deploys) workloads on a node that can give it the available resource requested.
- Limits define the value a workload is limited to. The workload is only allowed to go up to that limit of resources and restricted to surpass that limit.

Here is an example of what a Kubernetes deployment file looks like when requesting and limiting resources:

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: httpgooglechecker
spec:
  containers:
    - name: httpgooglechecker
      image: docker.io/astro7982/curlappstatic
      command: ['/bin/sh']
      args: ['run.sh']
      resources:
        requests:
          cpu: 100m
          memory: 64Mi
        limits:
          cpu: 1
          memory: 509Mi
```

When defining CPU limits, you should usually keep the CPU request at 1 core or below and run more replicas to scale it out, unless your workload is specifically designed to leverage multiple cores. This results in more flexibility and reliability.

For memory resources, they are defined in bytes. In most cases, you give a mebibyte value for memory, but you can provide anything from bytes to petabytes.

When configuring resources, if both the CPU and memory request is larger than the amount of memory on your nodes,
the pod will never be deployed by the scheduler.

After the pod is deployed with the example workload, you can use Splunk Infrastructure Monitoring to identify if resource limits were set on the pod when scheduled and the active resource metrics used based on those limits.

Next steps

The content in this guide comes from a previously published blog, one of the thousands of Splunk resources available to help users succeed. In addition, these Splunk resources might help you understand and implement this use case:

- .Conf Talk: Real-world strategies for Kubernetes & Docker integrations with Splunk
- Blog: Strategies for monitoring Docker and Kubernetes environments
- Splunk Docs: Kubernetes cluster
- Tech Talk: Splunk Operator for Kubernetes: Effortlessly deploy, scale, and manage Splunk
- Tech Talk: Maximizing the performance of your Kubernetes deployment