Kubernetes Monitoring in vRealize Operations

What to Monitor
“Many organizations now require all newly developed applications to be able to run on a cloud platform - because inevitably, they will run on a cloud environment - whether public, private, hybrid, or multi-cloud.”
In order to maintain agility with DevOps, monitoring of infrastructure performance is essential, as it provides crucial insights into the health of a given TKG environment.

1 Purpose
With the ever-increasing use of containerized applications, we see a significant proliferation of related technologies in modern data centers. These latest technologies have brought newer responsibilities for data center teams. For example, monitoring these environments for continuous operation and performance. Since the architecture of these related technologies are different, their components and monitoring process are also different. One such example is VMware Tanzu Kubernetes Grid (TKG). VMware Tanzu Kubernetes Grid, or TKG, is a Kubernetes-based container solution with advanced networking, a private container registry, and life cycle management. TKG simplifies the deployment and operation of Kubernetes clusters, so you can run and manage containers at scale on private and public clouds.

The questions under consideration are what to monitor in a containerized world, why monitoring is important for container operations, and specifically what to monitor in a deployed TKG environment.

Though there are many ways we can monitor a TKG environment, this document covers the VMware suite of products that is used to monitor TKG environments including vRealize Operations, vRealize Log Insight, and vRealize Network Insight.

2 Topics covered
The document is divided into below-mentioned two main categories.

• Monitoring containerized world
  • Monitoring philosophy
  • Monitoring K8s environment
  • Monitoring a TKG environment
  • What to monitor

Part A:

3 Monitoring containerized world
Before we discuss monitoring a containerized world, let’s discuss the idea of monitoring in general - its importance and why it is more important in today’s world. This will also act as a background to the main topics discussed in this document.

3.1 Monitoring Philosophy
If an organization decides only to use a public cloud solution, then they will not need any on-premises infrastructure (in terms of Server and Storage).

With the advent of newer technologies, the nature of applications has changed. The change is more visible specifically in the following areas:

• We moved from monolithic applications to modular applications, which in turn led to microservices. With advancements in container technology, we can now easily adopt the microservices architecture and build containerized applications.
• With the improvements in public cloud technologies and various advanced service offerings, we now have cloud native applications which exhibit certain key attributes.
• These changes led to a change in application development methodology. With the above advancements, organizations can now adopt agile processes for application development. An organization can implement agile methodologies through the adoption of DevOps processes and implementing Continuous Integration/Continuous Deployment (“CI/CD”) pipelines.

Many organizations now require all newly developed applications to be able to run on a cloud platform - because inevitably, they will run on a cloud environment - whether public, private, hybrid, or multi-cloud.
A healthy operation of applications and platform has the following three layers: observability, monitoring, and analysis.

3.1.1 Agile process
Cloud native operations practice Agile and DevOps. Agile processes with fast iterations require well-established feedback loops. Looking at the following figure, if we consider “code” as the starting point, then the process follows Code → Build → Test → Release → Deploy → Operate → Monitor → Plan path. This is a continuous process, which means we take continuous feedback and improve upon the existing application through the feedback loop.

![Figure 1: Cloud native operations with agile process](image)

In order to maintain agility with DevOps, monitoring of infrastructure performance is essential, as it provides crucial insights into the health of a given TKG environment.

Monitoring is an all-important step for consumers and providers alike, despite their differing roles and responsibilities. Consumers are typically the developers and application operators that will consume and operate Kubernetes, while providers are typically the platform engineers and cloud administers who must provide Kubernetes to whomever wants to consume it.

In section 4 of this document, we will further discuss which specific levels of a TKG environment each individual, depending upon their unique role, is responsible for monitoring and why.

For now, we can see that monitoring is a critical element of the DevOps practice.

3.1.2 Requirement for healthy operations
A healthy operation of applications and platform has the following three layers.

![Figure 2: Healthy operations criteria](image)
• Observability: It requires extracting data made available from the applications and other components
• Monitoring: Collecting, aggregating, and displaying observability data
• Analysis: The automatic or manual processing of collected monitoring data to provide meaning and take action

By following the above-mentioned layer implementations, we can ensure the healthy operation of any environment.

3.1.3 Purpose of monitoring
Here, we will try to find answers to the following two questions.

• **Why Monitoring** - The purpose of monitoring is as follows:
  • Know when things go wrong
  • Detect problems via alerts and notifications
  • Debug issues and gain insight
  • Detect changes over time and drive technical/business decisions
  • Ultimately, the MONITOR section of the DevOps loop provides the all-important feedback that drives future iterations

• **Components of Monitoring** - Listed below are the components of monitoring:
  • Instrumentation (generating metrics)
  • Metrics collection and storage
  • Querying, alerting, and dashboarding

So, to successfully monitor any environment, we need to ensure the availability of all the above-mentioned three components.

3.2 Monitoring a K8s environment
Before we discuss a K8s environment, let’s have a look at a typical containerized environment, its different components/layers, and different players in those areas.

3.2.1 Containerized env. components
The diagram below depicts the different layers of a containerized environment and major players in those areas.

![Figure 3: Components of a containerized environment](image-url)
• Layer 4: The foundation for the entire solution. This layer provides the actual containers. Among many different solutions—Docker, rkt, and LXD are few of the major players.

• Layer 3: While running applications in small scale with basic container providers are fine, it becomes very hard to manage in large environments. With thousands of containers running in an environment, we need a better way to manage all those containers. This requirement leads to a manager for containers. Hence the rise of container managers which run and manage thousands of containers efficiently. Major players providing these solutions are Kubernetes, Yarn, and Mesos.

• Layer 2: While the container managers make it easy to manage containers at scale, container management in itself is not easy. These are complex solutions requiring specialized skills and building such an environment requires heavy time and effort. So, there are again solutions which can deploy and manage the manager for containers. Players in this are VMware TKG, RedHat OpenShift, Rancher, etc.

• Layer 1: At the top of the pyramid, we have applications. Remember, the application drives everything. So, we host and manage many applications on top of the other layers.

When container technology started to take shape, there were many players offering solutions in that area. Today, Kubernetes is the undisputed leader in container management (with Docker winning the container technology race). Therefore, we will limit our discussion to Kubernetes (K8s) environments only.

3.2.2 End to end monitoring of K8s environment

Now, let’s look at the recommended solutions for hosting and optimizing a Kubernetes environment. Though there are many ways to host a Kubernetes environment, for this discussion and document, we will limit ourselves to available options from VMware only.

![Figure 4: Layers of a K8s hosting environment](image)

The above diagram depicts the different layers involved in a K8s hosting environment and the solutions that are available/used for monitoring them.

By using the solutions mentioned above, we can monitor and optimize such an environment to a great length.

Since we are limiting this discussion to VMware provided solutions only, we will discuss VMware Tanzu Kubernetes Grid (TKG) as the manager for K8s environment.

In the following section, we will give more detail about the end-to-end monitoring of a TKG environment.
Part B:

4 Monitoring a TKG environment

Before we delve into the details of monitoring a TKG environment, it is imperative to understand the different aspects of a TKG environment. Also, it is important to understand the different roles and responsibilities for monitoring the environment. This section covers about those details of monitoring a TKG environment.

4.1 Who monitors what?

Provided below is a list showing the different layers in a TKG environment and the roles that may be responsible for monitoring certain layers.

![Figure 5: Different TKG environment layers and the monitoring responsibility of the different teams](image)

<table>
<thead>
<tr>
<th>Layers</th>
<th>Roles</th>
<th>Examples of Monitoring Areas</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applications</td>
<td>End User, App Business Owner</td>
<td>availability, latency, security, usage trends, sessions lengths, API usage</td>
</tr>
<tr>
<td>Image Registry</td>
<td>Security, Admins, Developers</td>
<td>image security, CVEs, patches, upgrades</td>
</tr>
<tr>
<td>Scheduling, Orchestration, Services</td>
<td>Kubernetes Admins/Developers</td>
<td>services state, service metrics, PV &amp; PVC usage</td>
</tr>
<tr>
<td>Cluster Health, Healing &amp; Lifecycle Management</td>
<td>Cluster/TKG Admins, Platform Engineers</td>
<td>health, capacity, load, security</td>
</tr>
<tr>
<td>Virtual Infrastructure</td>
<td>VI Operations</td>
<td>services state, service metrics, security</td>
</tr>
<tr>
<td>Physical Infrastructure (compute, storage, networking)</td>
<td>Infra Operations</td>
<td>usage, capacity, breaks, patches, upgrades, health of nodes, security, compliance</td>
</tr>
</tbody>
</table>

Note, the roles need not be mutually exclusive, and they may slightly overlap at times based on organizational structures.

As mentioned in Section 3.1.1. and shown in the graphic above, these roles can be categorized as either “consumer operations” (i.e., those consuming the Kubernetes services), or “provider operations” (i.e., those providing the Kubernetes services).

Consumers, or the developers and application operators, may be responsible for delivering their applications to the end user. Therefore, they are focused on maintaining the application’s security, performance, and overall usability. In terms of monitoring, they are mostly concerned with response times between layers of the application itself.

On the other hand, producers, or the platform engineers and cloud administrators, are responsible for providing the platform to their consumers whether it be physical, virtual, storage, or networking infrastructure. Their main concern is maintaining availability, performance, and capacity of the platform, while staying aligned with business needs, such as budgetary constraints, all which require monitoring.

Now, we can see how monitoring needs vary based upon the monitor’s role and their main concerns.

4.2 Example Metrics

The following picture contains a list of metrics to monitor within the different layers of a TKG environment. This is an example list and does not include all the possible metrics to watch in the environment.
Figure 6: A sample list of metrics to monitor in the different TKG environment layers

4.3 Different tools for different layers

The diagram below offers a pictorial view of different TKG layers and the optimal VMware tools for monitoring those layers.

```
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Kubernetes</td>
<td>K8s Resources: LB, Services, Pods</td>
<td>Cluster Audit Logs</td>
<td>etcd Health</td>
<td>Available PV Volumes</td>
<td>Volume Claims</td>
<td>Nodes at KS Level</td>
<td>API Usage</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TKG NSX-T</td>
<td>Health of Cluster, and Workers</td>
<td>Number of Clusters, Namespaces</td>
<td>Login Access</td>
<td>Alerts Against Capacity</td>
<td>IP Address Use vs Allocation</td>
<td>Health of Edges</td>
<td>No. of LBs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IaaS</td>
<td>Hardware Health</td>
<td>Health of VMs for TKG, Bosh, NSX-T</td>
<td>Network Connectivity</td>
<td>Network Bandwidth</td>
<td>Health of KSs Control Plane and Worker Nodes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Monitoring needs vary based upon the monitor’s role and their main concerns.

```
As you can see in the image above, “consumer ops”, who are responsible for providing apps to end users, are primarily concerned with monitoring the top 2 layers of a TKG environment, since those layers are tied to application performance.

Meanwhile, “provider ops” are primarily concerned with monitoring the bottom 2 layers, which are tied to infrastructure and platforms. This is because “provider ops” are responsible for maintaining the health and availability of those resources, which they are providing to consumers.

Therefore, it is important to note that user personas dictate which specific layers of a given TKG environment one might be responsible for monitoring.

Figure 7: Different TKG environment layers and associated VMware monitoring tools

Monitored with Tanzu Kubernetes Observability (WaveFront)
Monitored with vRealize Operations/vRealize Network Insight/vRealize Log Insight
Monitored with vRealize Operations/vRealize Network Insight/vRealize Log Insight
Monitored with vRealize Operations/vRealize Network Insight/vRealize Log Insight

---|---|---|---|---|---|---|---|---|---|---|---|---|---|---
4.3 Different tools for different layers
The diagram below offers a pictorial view of different TKG layers and the optimal VMware tools for monitoring those layers.
4.4 Flow of monitoring and troubleshooting

Usually, for different layers, the respective teams monitor and maintain the health of that layer. If something breaks, teams quickly observe and rectify issues. But performance-related issues can be hard to troubleshoot. Typically, the end-users or developers notify other teams about the problem they face. Provided below is a pictorial view of the different layers, teams typically responsible for them, VMware tools used to monitor those, and the general flow of troubleshooting issues in TKG overall environment.

It should be noted that self-driving operations tools like vRealize Operations and vRealize Network Insight are designed to self-optimize infrastructure performance and the speed of remediation when issues do crop up, so as much reactive “firefighting” as possible is mitigated in this flow.

![Figure 8: General flow of troubleshooting issues in TKG overall environment](image)

4.5 VMware Solutions for TKG Monitoring and Troubleshooting

To explore existing troubleshooting options please follow the path, products and information provided in the below diagram:

![Figure 9: Existing options in different VMware products to monitor a TKG environment](image)
Self-driving operations tools like vRealize Operations and vRealize Network insight are designed to self-optimize infrastructure performance and the speed of remediation.

4.6 vRealize Operations Capabilities for TKG Monitoring and Troubleshooting

VMware vRealize Operations delivers self-driving IT operations management for private, hybrid, and multi-cloud environments in a unified, AI-powered platform. It is available either as an on-premises (vRealize Operations) or SaaS (vRealize Operations Cloud) offering.

Recently, container operations have been a heavy point of emphasis in the vRealize monitoring story. Just as vRealize Operations has been instrumental for years in consistent operations regardless of virtual/cloud infrastructure, today’s vRealize Operations delivers consistent operations for consistent Kubernetes.

vRealize Operations is able to monitor multiple Kubernetes solutions, whether VMware TKG, RedHat OpenShift, or Kubernetes on AWS EC2, Azure, or Google Virtual Machines. However, for this section, we will focus specifically on the native TKG integration.

As central IT teams start to manage more Kubernetes environments, they can simply extend their existing investment in vRealize products to this new world and empower their teams to have an end to end visibility into business applications comprised of traditional VM based apps and modern microservices-based apps. As with the “traditional” infrastructure constructs that vRealize Operations has monitored for years, the primary goals are to support application performance from the infrastructure perspective, troubleshoot and find the root cause quickly and easily when things go wrong (in many cases providing airtight “proof of innocence” for infrastructure teams), determine capacity needs for the environment, and ensure integrated compliance with organizational policies.

To clear up a potential question - vRealize Operations is designed for “provider operations” - (the infrastructure teams providing infrastructure resources to application teams) whereas APM tools like Wavefront are designed for “consumer operations” (developer/devops teams consuming those resources).

Figure 10: Discoverable/Monitorable K8s Constructs in vRealize Operations
The below is a summary of the monitoring, troubleshooting, and capacity management capabilities vRealize Operations delivers for TKG K8s clusters:

- Auto-Discovery of K8s clusters
- End-to-end visualization of K8s cluster topology, including namespaces, clusters, replica sets, nodes, pods, and containers
- KPIs for performance monitoring
- OOTB overview dashboards (Workload Management Inventory, Workload Management Configuration, etc) with troubleshooting workflows
- Multiple alerts to help monitor the K8s infrastructure
- Mapping of K8s nodes to VM objects
- Capacity planning/management for K8s clusters
- Reporting for cluster/pod capacity, configuration, inventory, and more

For a detailed guide on how to get the most out of these capabilities, check out our follow-up deep dive whitepaper Kubernetes Monitoring in vRealize Operations: How to Monitor.

VMware vRealize Operations delivers self-driving IT operations management for private, hybrid, and multi-cloud environments in a unified, AI-powered platform.

As central IT teams start to manage more Kubernetes environments, they can simply extend their existing investment in vRealize products to this new world and empower their teams to have an end to end visibility into business applications comprised of traditional VM based apps and modern microservices-based apps.