VMware Tanzu
Observability

Comprehensive Kubernetes Observability at Scale

A guide to enterprise observability
for Kubernetes environment full-stack.

vmware®

Table of Contents

Introduction

One of the main forces of business innovation today are modern applications. They are deployed in lightweight and portable containers together with their dependencies and configurations. As they scale quickly to meet the needs of millions of users in milliseconds, they span multiple containers across multiple hosts. Operating these huge number of containers in real time becomes significantly more complex.

Kubernetes, an industry standard for container orchestration, manages containerized applications' complexities. It discovers services, incorporates load balancing, tracks resource allocation, scales based on utilization, and checks the health of individual resources. It also enables applications to self-heal by automatically restarting containers wherever applications run—from public clouds (AWS, Azure, GCP) to private (VMware) and hybrid clouds, on-premises, or any combinations of infrastructures.

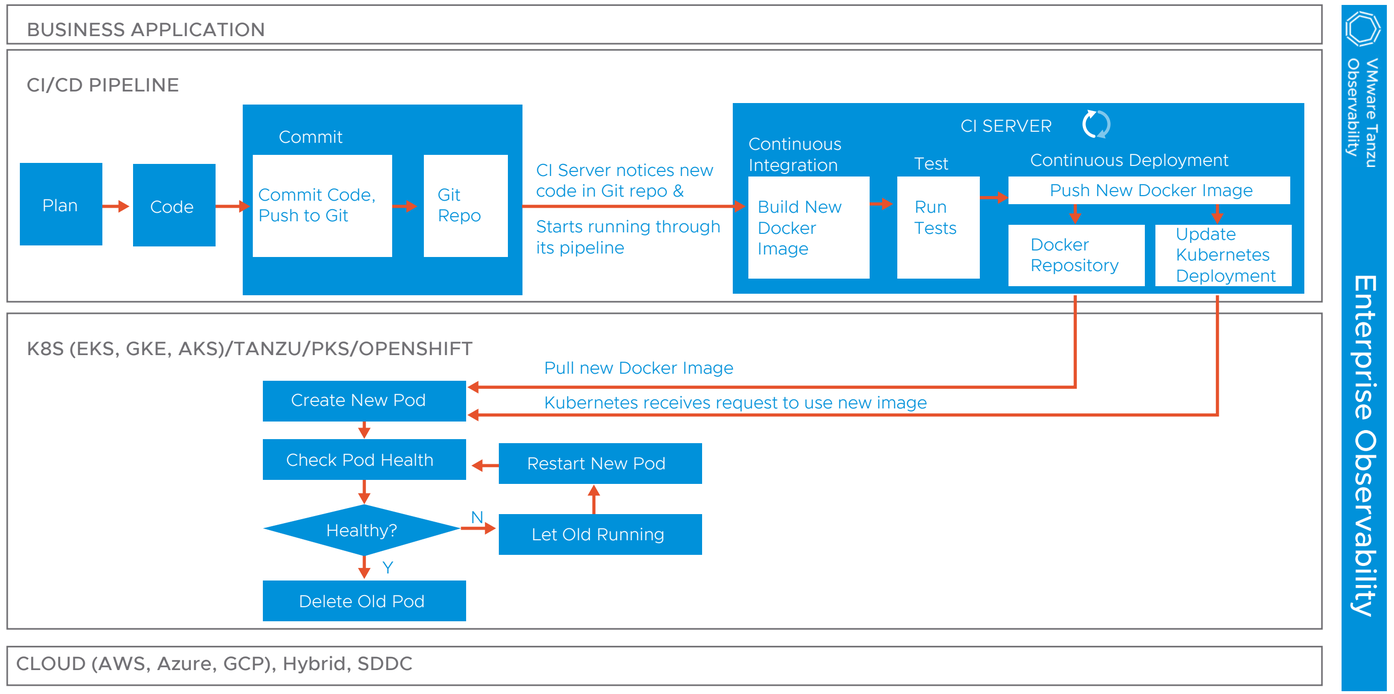


FIGURE 1: CI/CD Pipeline Workflow with Kubernetes Zero-Downtime Deployment and Full-Stack Observability

However, Kubernetes also introduces new operational and observability challenges. In this eBook, we will talk about what can go wrong with Kubernetes, and what you, as Kubernetes architect, operator, SRE, or developer, should monitor and look for in an enterprise observability platform for Kubernetes. We will provide you with real-world Kubernetes case studies and show the implementation of automated Kubernetes observability with Tanzu Observability. We hope that you will find this eBook helpful, as it includes practical tips on optimizing reliable performance, fast pinpointing of performance issues, and minimizing the time for troubleshooting and resolving the issues of microservices deployed on Kubernetes.

What's Needed for Maintaining Kubernetes

Kubernetes simplifies deployment, scaling, and management of applications, but is complex to install, configure, and manage. You will quickly realize that observability and monitoring are crucial for understanding relationships between infrastructure, Kubernetes and application services of this dynamic environment. There will be many questions and to answer them, you will need to correlate Kubernetes with application data, aggregate data across Kubernetes clusters, or isolate data for each Kubernetes service. Sometimes, Kubernetes recovery will work so well, that you would not have to isolate root cause and quickly respond to an incident. For example, Kubernetes might restart your application and you would not notice that some of your containers crashed multiple times due to memory leak. Still, you should know about this.

To keep track of how things are running, you should have dashboards that monitor Kubernetes nodes, individual pods, and all application's services with load, throughput, application errors, and other statistics. In the illustration on the right, the components interact as follows:

Master Node—a gateway and brain for the cluster

- Kube-apiserver—allows a user to configure Kubernetes workloads and organizational units
- Kube-scheduler—assigns workloads to specific worker nodes
- Controller—regulates the state of the cluster (a group of servers)
- Etcd—stores configuration data that is accessed by each node of the cluster

Worker Node—a server that accepts and runs workloads

- Kubelet—interacts with master server components
- Kube-Proxy—forwards requests to the containers
- Pod—consists of main and helper containers that operate closely together
- Container—deploys applications

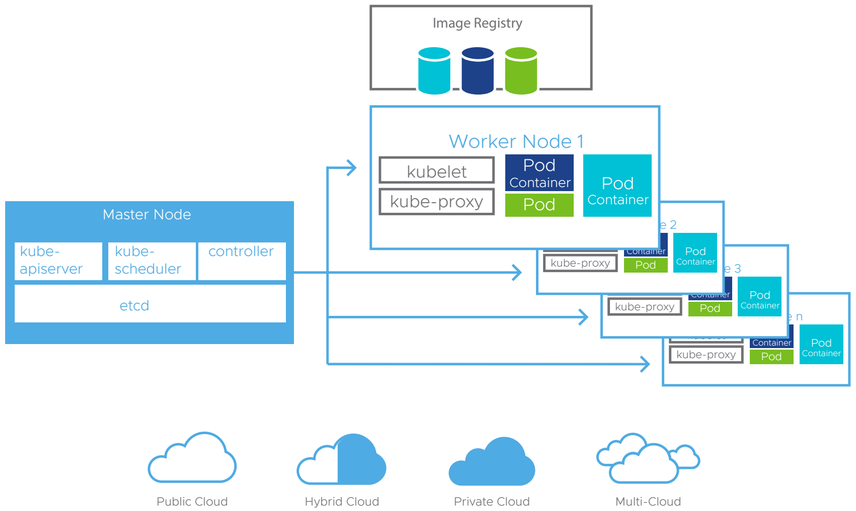


FIGURE 2: Kubernetes Cluster with Master and Worker Nodes Components

What Can Go Wrong with Kubernetes?

After making a change to the environment, SREs and developers need to correlate the application and the Kubernetes data in order to understand the impact of those changes on the Kubernetes environment. They need to understand the dependencies between their applications and the Kubernetes environment on which they run. Application performance depends on having enough resources and on the successful operation of the Kubernetes components and services on which the application runs.

For example, if some of the physical or virtual machines in the Kubernetes clusters are overloaded or misconfigured, all other operations within the cluster and the deployed application on top may stop working as well. A Kubernetes API server outage can affect running workloads. Similarly, if there are problems with a service, those services that are upstream and downstream of the application may have issues as well.

While SREs and platform engineers need to operate the Kubernetes orchestration layer, developers must understand the impact of the Kubernetes orchestration layer on their applications and microservices. Therefore, their challenges are different and are listed in the table below.

Though their challenges are different, both types of users need to ingest, observe, be alert about, and understand the data flow and the metrics exposed through Kubernetes.

SREs' AND PLATFORM OPERATORS' CHALLENGES
• Assure that the Kubernetes components operate as expected—e.g., over or under allocated resources may cause non-starting or crashing pods under high workloads • Run within cloud provider limits—e.g., if a container exceeds its memory limit, Kubernetes might terminate it • Use resources per bus network policies—e.g., traffic spikes can be amplified by network latencies • Pull images from trusted locations • Understand etcd load per object and call because major etcd failure will cripple or take down the container infrastructure • Predict capacity across application load • Manage components certification expiration, cluster version mismatch, and updates—e.g., if blue-green deployments are not used, requests can fail during the software update due to still coming upstream traffic to already stopped service • Avoid node hotspots—e.g., nodes with low memory or disk capacity
DEVELOPERS' CHALLENGES
• Operate workloads within specified resource limits for their applications or microservices—e.g., increased number of pods per node can lead to service latencies or working in proximity of CPU limit might significantly slow down the application • Take care of upstream and downstream dependencies • Resolve mismatch between desired and actual state (applications/ workloads, container images, number of replicas, network and disk resources) for better resources usage and application performance • Use all defined services and delete ones that connect to nothing since when high-scaled, the kubelet might get bogged down • Prevent pod failures

TABLE 1: Platform Operators' vs. Developers' Challenges in Monitoring Kubernetes Environment

Kubernetes Observability vs. Monitoring Kubernetes

Monitoring Kubernetes shows you whether a Kubernetes environment and all its layers—clusters, nodes, pods, containers, and application workloads are operating as expected. The most popular open-source solution for monitoring Kubernetes is Prometheus.

Prometheus provides real-time monitoring, alerting, and time-series database functionalities for modern applications. It can collect Kubernetes layers and is a simple and flexible monitoring toolkit that measures symptoms and enables you to determine what is wrong.

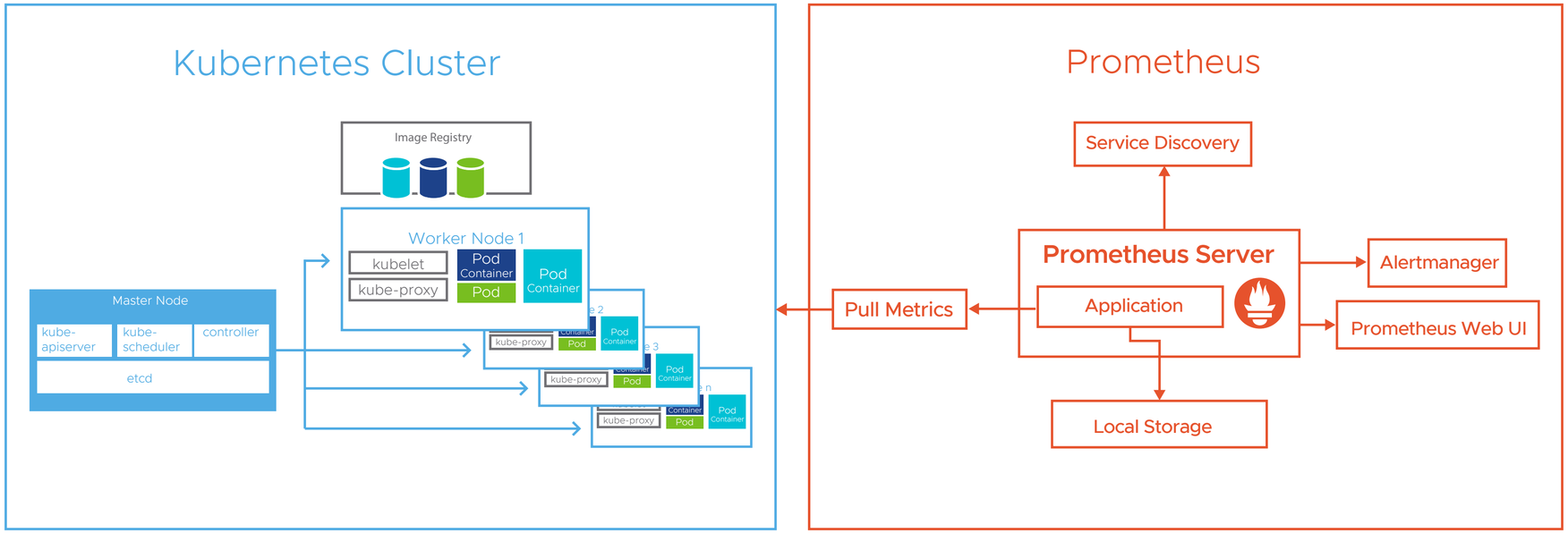


FIGURE 3: Monitoring Kubernetes with Prometheus

Monitoring is a great start. However, to get a context on how Kubernetes components influence the performance of Kubernetes applications, and correct problems before they become end-user problems, you need Kubernetes observability. Kubernetes observability provides engineers with a complete picture and all the information necessary for increasing performance and improving the stability and resiliency of applications, the Kubernetes components, and the underlying infrastructure. Kubernetes's observability tools include comprehensive overviews, intelligent summaries and alerts, and ways to correlate metrics changes with Kubernetes events. For example, they can provide more insight if a container failure is caused by a Docker image pull up or application issue.

How Tanzu Observability Helps Developers,SREs and Platform Operators

Tanzu Observability is an observability platform that is specifically designed for enterprises that need monitoring, observability, and analytics for their modern applications. Tanzu Observability delivers full-stack observability for Kubernetes with advanced analytics on metrics, traces, histograms, and span logs that come from the applications themselves, from application services, container services, and your multi-cloud.

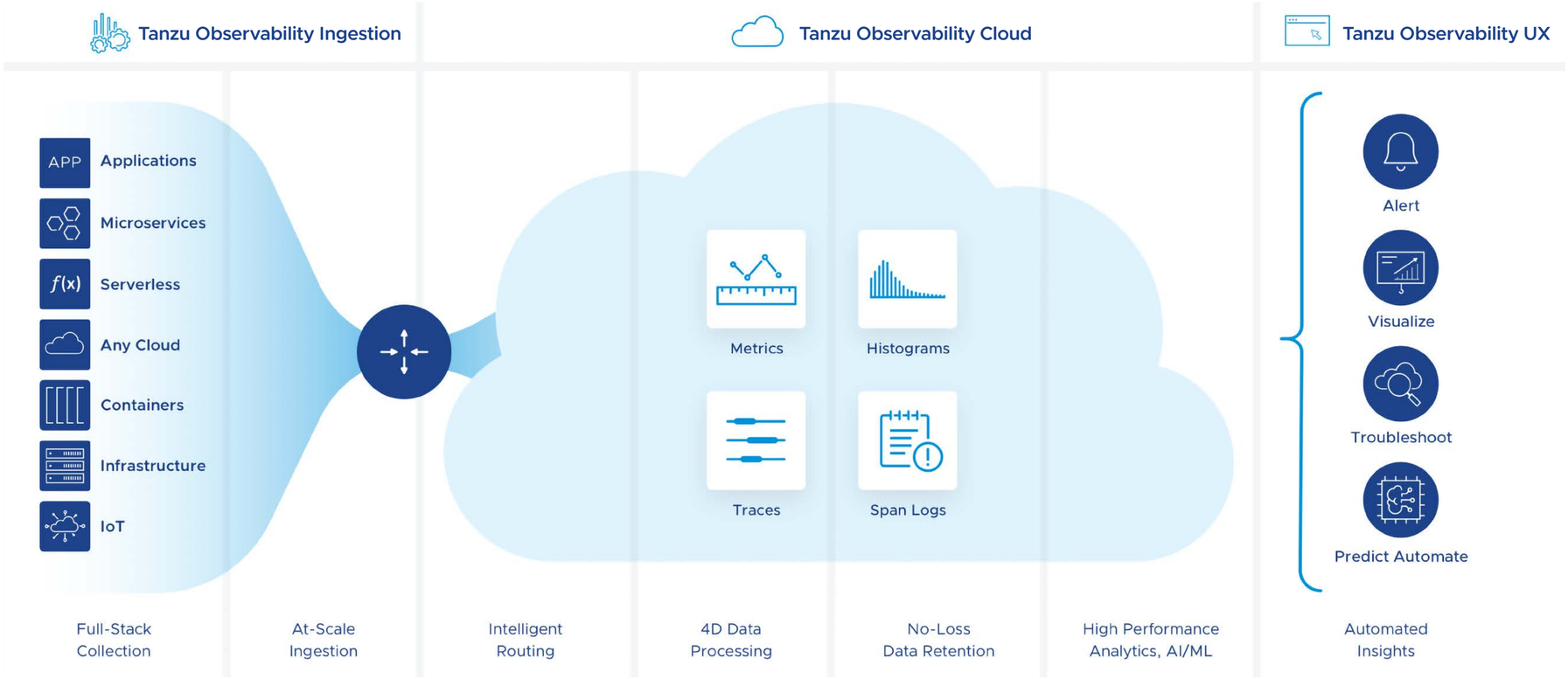


FIGURE 4: Tanzu Observability—Enterprise Observability Platform for Modern Applications on Kubernetes and Multi-Cloud

With Tanzu Observability enterprise-grade observability for Kubernetes, developers, SREs, and Kubernetes platform operators get accelerated time to value with automated and unified comprehensive insights into the health, state, and performance of their application, Kubernetes and multi-cloud environments. Engineers use Tanzu Observability to proactively alert on problems so that they can troubleshoot and optimize the performance of their modern applications rapidly. They can get answers to questions such as:

- Was there a performance regression with the latest code update?
- Are there any errors?
- Are there opportunities for performance optimization?

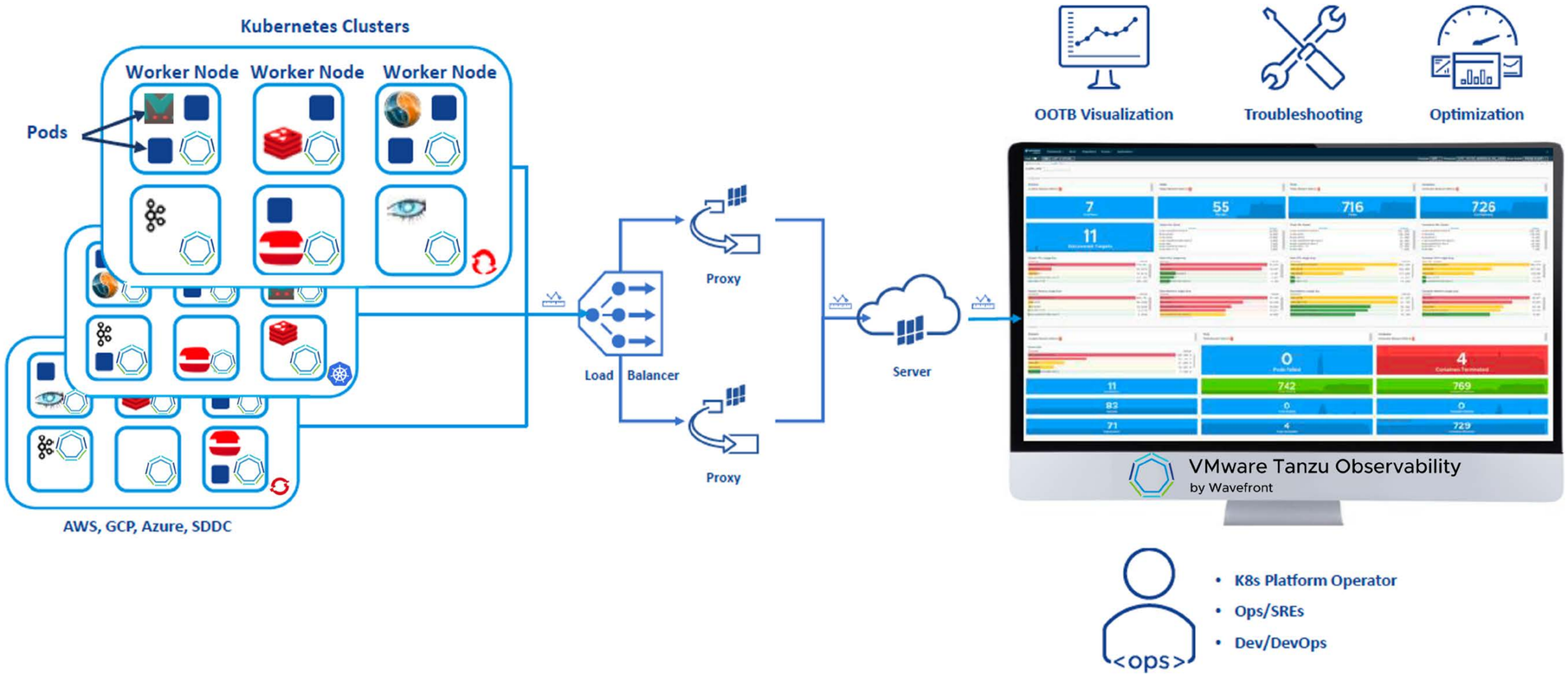


FIGURE 5: The Tanzu Observability Architecture of Kubernetes Observability with Automated Service Discovery and Full-Stack Analytics

Tanzu Observability for Kubernetes starts with a zero-configuration installation. Next, Tanzu Observability automatically recognizes Kubernetes services, *discovers*, and instruments Java-based services across any clouds. With the proven scale of up to 200,000+ containers per cluster, Tanzu Observability ingests and visualizes with no interruptions:

- *Core Kubernetes metrics* (CPU, memory, network, storage, uptime, restarts)
- *Host level metrics*
- *Application metrics from popular applications* (Redis, RabbitMQ, PostgreSQL, etc.)
- Prometheus endpoint data from the API Server, NGINX and etcd
- *Comprehensive metrics of the Tanzu Observability Collector for Kubernetes*.

Visualization is available through automatically populated prebuilt dashboards with full-stack metrics from all Kubernetes layers—clusters, nodes, pods, containers, and system metrics. All this data is continuously available with per 1-second granularity (or sub-second through histograms), no matter how often the container fleet is refreshed because of code deployments and updates. To meet dynamic end-user demand, Tanzu Observability can also horizontally autoscale Kubernetes pods based on ingested custom and external metrics APIs using the *Kubernetes HPA (Horizontal Pod Autoscaler) adapter*.

Engineering teams can rapidly identify root causes with predefined Tanzu Observability alerts and events for Kubernetes operation. Tanzu Observability gathers detailed information about the operation of the Kubernetes environment, such as utilization extremes, SLA violations, excessive pod restarts, and paused or stuck deployments

Engineers can use the Tanzu Observability auto-configured set of Kubernetes-related alerts but are also empowered to configure instant real-time alerting across multi-cloud Kubernetes environments at a large scale. All these alerts enable engineers to detect and resolve issues proactively before service disruption.

For fast and accurate troubleshooting, Tanzu Observability offers packaged Kubernetes troubleshooting dashboards with high fidelity metrics without roll-ups.

Kubernetes troubleshooting dashboards keep track of many important metrics and events, such as memory utilization, scheduling issues, and crash loop detection. If there are no issues, these dashboards provide status only. When important Kubernetes metrics are out of the range or a critical Kubernetes event is fired, the dashboards will show the metrics and information that engineers need to troubleshoot the issue. By only focusing on relevant metrics in critical times, Tanzu Observability protects engineering teams from being overwhelmed, enabling them to focus on the issue at hand.

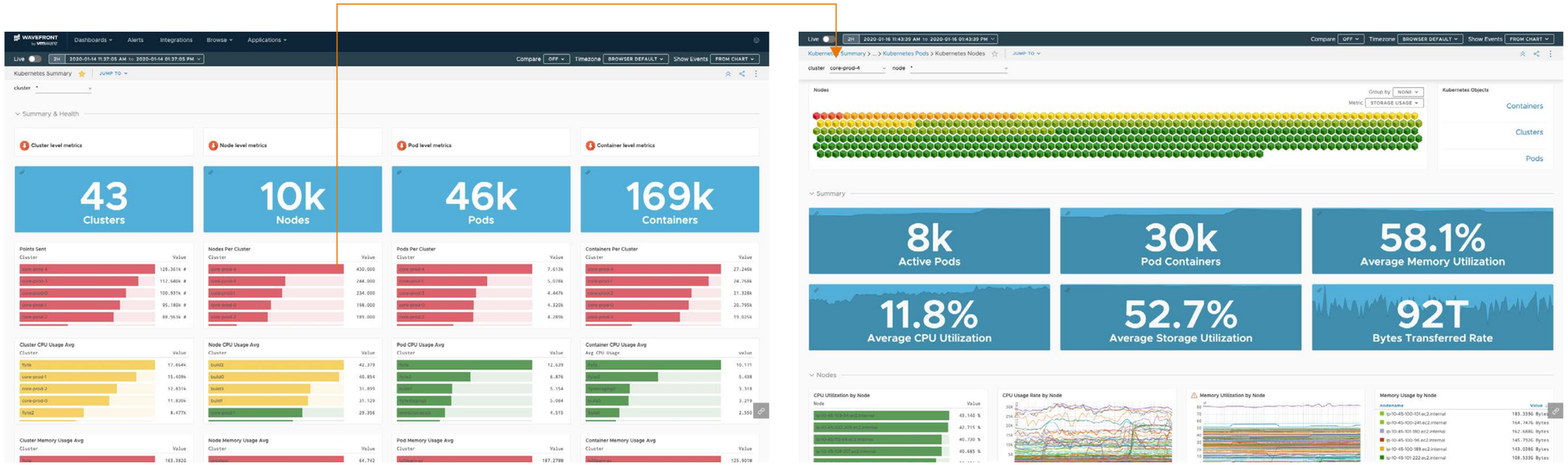


FIGURE 6: Kubernetes Observability Dashboards of One of the Top Two Transportation as a Service Companies

BENEFIT	DESCRIPTION
Automated Detection of Kubernetes Clusters4	✓ Autodiscovers all Kubernetes clusters ✓ Gives one-liner for zero-configuration installation
Zero-configuration Installation	✓ Installs Wavefront Collector for Kubernetes (WCK) for: – Open-source-based Kubernetes implementations (EKS, GKE, AKS) with Wavefront operator for Kubernetes – OpenShift environment with Red Hat Certified Wavefront operator for OpenShift 4.x – Hybrid environments – On-prem environments
Services Autodiscovery	✓ Recognizes what services are up with plugins: – Prometheus – for Prometheus metric endpoints – *Telegraf – for applications supported by Telegraf* – Systemd – for system components for Linux OS ✓ Collects key metrics from containerized applications running within Kubernetes ✓ Instruments Java-based applications via Wavefront Java Tracing Agent
Automated Instrumentation of Java-Based Applications	✓ Automatically attaches the Wavefront Java Tracing Agent (WJTA) to a Java service running on a pod inside Kubernetes ✓ Instruments Java-based applications and gathers traces
Pre-packaged Kubernetes Dashboards	✓ Populates pre-packaged Kubernetes dashboards: – Kubernetes Metrics – Kubernetes Metrics by Namespace – Kube-State Metrics – Wavefront Collector Metrics

TABLE 2: Key Functionalities of Tanzu Enterprise Observability for Kubernetes

BENEFIT	DESCRIPTION
Pre-packaged Kubernetes Dashboards	✓ Provides full-stack metrics from all Kubernetes layers: – Core Kubernetes – Clusters – Hosts/Worker Nodes – Pods – Systems – Custom Applications ✓ Enables further Kubernetes dashboard customization
Annotated Relationships Between Full-Stack Layers	✓ Enables easier and quicker drill-downs for faster triaging and incident resolution
Autoscale of Kubernetes Pods	✓ Autoscales based on any custom or external metrics APIs in Tanzu Observability via horizontal pod autoscaler (HPA) adapter so that end-user dynamic demand is met (see *HPA-examples*)
Metrics and Events Filtering	✓ *Filters metrics and events before they are reported to Tanzu Observability*
High scale with up to 200,000+ Concurrently Running Containers per Kubernetes Cluster	✓ Ingests, visualizes, and analyzes telemetry – Real-time, with up to 1-second resolution – Sub-second rsolution through histograms ✓ Solves high cardinality issues with indexing strategy, patented functionality, and high-performance database – Fast queries at high scale – Reliable and uninterrupted observability during code deployments and updates – Instant real-time alerting
Automated Kubernetes Operational Intelligence	✓ Provides Kubernetes troubleshooting dashboards with an auto-configured set of Kubernetes-related events and alerts

TABLE 2: Key Functionalities of Tanzu Enterprise Observability for Kubernetes

Comprehensive Solution for All Popular Kubernetes Implementations

Tanzu Observability provides automated and unified enterprise observability for Kubernetes, and applications for all the most popular Kubernetes implementations: *Tanzu*, *open-source Kubernetes*, *Amazon EKS*, *Google Kubernetes Engine (GKE)*, *Azure Kubernetes Service (AKS)*, *VMware Cloud PKS*, and *OpenShift*.

All these Kubernetes implementations are supported by the Tanzu Observability enhanced Kubernetes Integration that uses:

- the *Wavefront Collector for Kubernetes (WCK)* to collect metrics from Kubernetes clusters
- the *Wavefront Java Tracing Agent (WJTA)* to automatically instrument Java-based applications

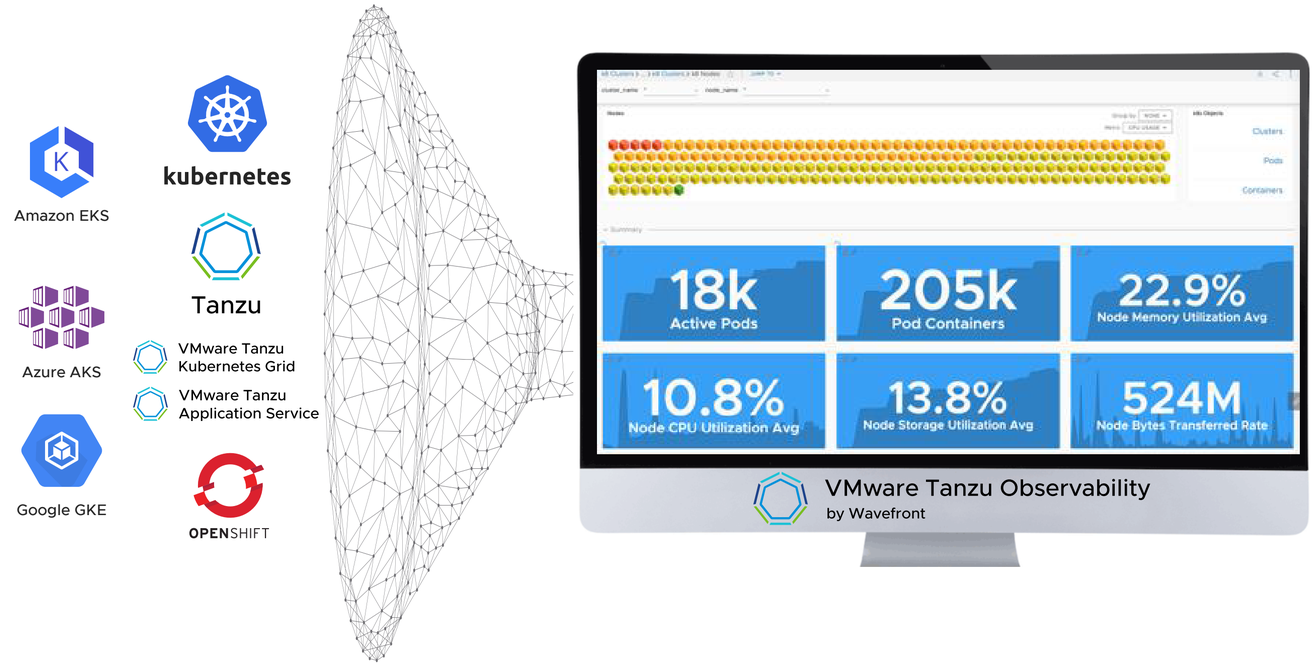


FIGURE 7: Tanzu Observability Platform for All Popular Kubernetes Implementations

The Wavefront Collector for Kubernetes (WCK) makes it fast and easy for Ops/SRE and DevOps/Dev to monitor and manage Kubernetes environments. The collector is a cluster-level agent that runs as a pod within a Kubernetes cluster. It can be installed:

- *Using Helm*—for easy and consistent installation of the WCK and Tanzu Observability proxies
- *Manually*—a 4-step procedure often used for testing and development

Once deployed, the WCK discovers the nodes in the Kubernetes cluster and starts collecting the metrics from the Kubelet Summary API on each node.

RESOURCE	METRICS
Cluster	CPU, Memory
Namespace	CPU, Memory
Nodes	CPU, Memory, Network, Filesystem, Storage, Uptime
Pods	CPU, Memory, Network, Filesystem, Storage, Uptime, Restarts
Pod Containers	CPU, Memory, Network, Filesystem, Storage, Uptime
System Containers	CPU, Memory

TABLE 3: Kubelet Summary of API Metrics Collected by the Wavefront Collector for Kubernetes on Each Node

Additionally, the WCK collects *kube-state-metrics* about the state of the Kubernetes objects (deployments, nodes, pods) to provide total visibility into the state of Kubernetes resources.

The WCK also supports the auto-discovery of pods and services that expose Prometheus metric endpoints based on *annotations* (default) or discovery rules.

ANNOTATION	DESCRIPTION
prometheus.io/scrape	If `true` pod/service will be autodiscovered
prometheus.io/scheme	Defaults to `http`
prometheus.io/path	Defaults to `/metrics`
prometheus.io/port	Defaults to a port-free target if omitted
prometheus.io/prefix	Prefix for reported metrics: defaults to an empty string
prometheus.io/includeLabels	Include pod/service labels as metric tags: defaults to true
prometheus.io/source	Source for the reported metrics: defaults to the name of the Kubernetes resource
prometheus.io/collectionsInterval	Custom collection interval; defaults to 1min

TABLE 4: The Wavefront Collector for Kubernetes (WCK) Annotations

The WCK also supports:

- Rules that enable discovery, based on labels and namespaces, by automatically detecting configuration changes without the need to restart it. The rules support Prometheus scrape. They are configured as a *ConfigMap* and provided to the collector using the optional -discovery-config flag.
- Scraping Prometheus targets based on static configurations using the -source flag.
- Metrics about its internal health.

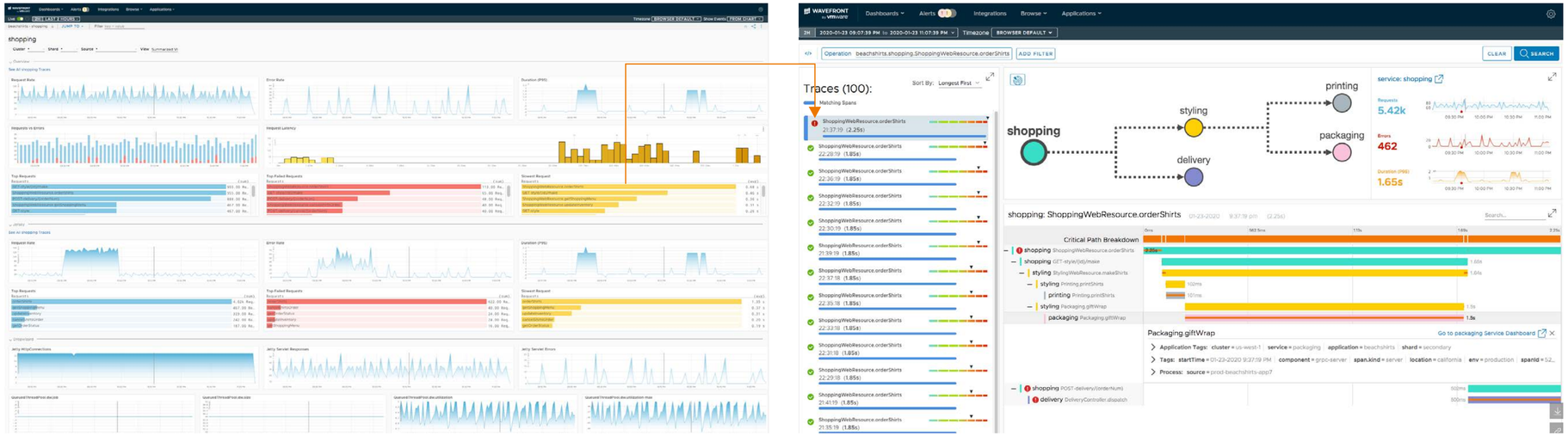


FIGURE 8: Out-Of-The-Box Kubernetes Application Performance Visibility provided by Wavefront Java Tracing Agent

The Wavefront Java Tracing Agent provides an easy, automated experience for achieving application observability without code changes. The Wavefront Java Tracing Agent is based on the *Java Special Agent*, which is an open-source tracing agent for Java. The Wavefront Java Tracing Agent automatically instruments third-party libraries in Java, providing out-of-the-box visibility into traces, RED metrics, and span logs for Java-based applications. Tanzu Observability unified observability for Kubernetes and applications results in a more in-depth understanding and faster troubleshooting of Kubernetes environments and the application workloads running on them.

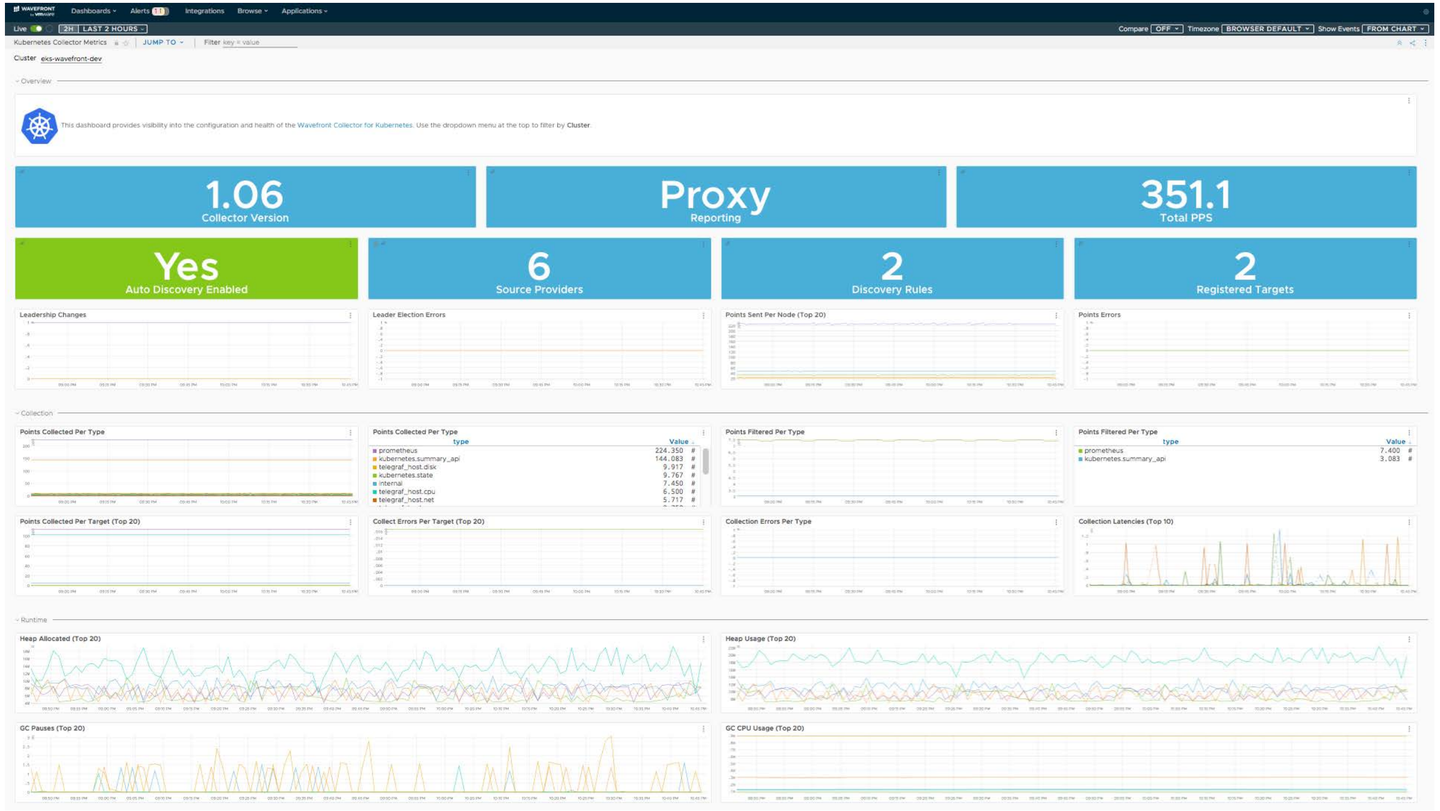


FIGURE 9: The Wavefront Kubernetes Collector Internal Health Dashboard

Tanzu Observability for VMware Tanzu

VMware Tanzu is a portfolio of products and services that helps enterprises build, run, and manage applications on Kubernetes. Tanzu helps build modern applications with the leading developer-centric platforms (Spring, Build Service, Function Service, Application CI/CD Service) and with Bitnami's application packaging solution. Bitnami has the most extensive catalog of click-to-deploy applications and development stacks for major cloud and Kubernetes environments.

With *VMware Tanzu Kubernetes Grid (TKG)*, Tanzu enables running Kubernetes across any infrastructure: vSphere, VMC on AWS, public clouds, and edge. It runs Kubernetes consistently with a fully integrated cluster lifecycle management and open-source container image registry.

Through Tanzu Mission Control, VMware Tanzu manages the entire Kubernetes footprint across clouds, clusters, and teams from a single point of control. Tanzu empowers both developers and Kubernetes operators. Developers have self-service access to resources, and Kubernetes operators operate multiple clusters running across multiple clouds easier and faster and have more control over cost.

When using Tanzu with Tanzu Observability, developers, SREs, and Kubernetes platform operators:

- Get full-stack observability into applications running across clouds, as well as their associated namespaces, clusters, pods, containers, and configuration code
- Release faster code updates with an automated development CI/CD pipeline

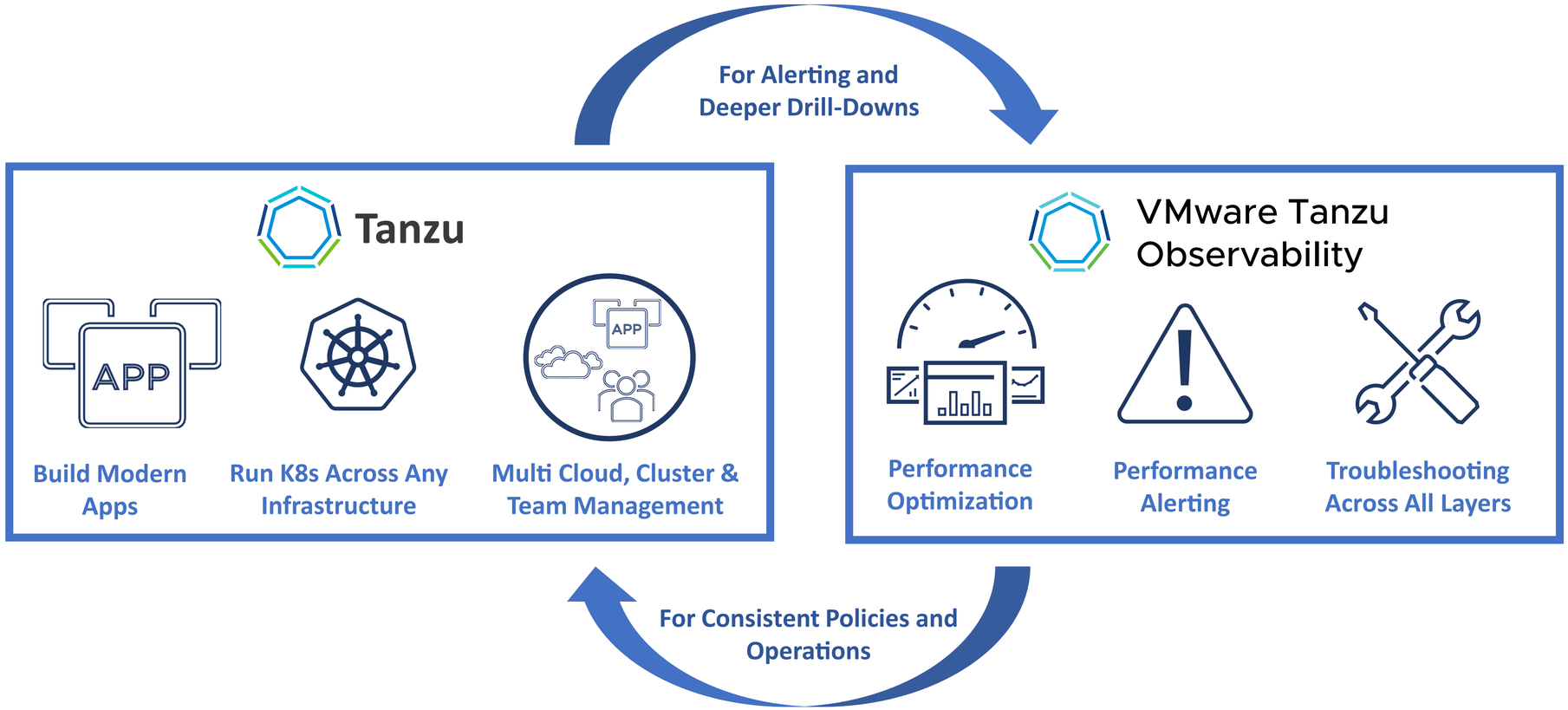


FIGURE 10: Tanzu Observability for Tanzu: Proactive Applications and Kubernetes Observability

- Quickly identify abnormalities across cluster fleets and applications with Tanzu Observability's powerful AI/ML-driven analytics
- Drill down, troubleshoot, and resolve issues faster with Tanzu Observability-integrated metrics, distributed tracing, histograms, and span logs
- Reduce MTTR to optimize code performance
- Predict future resource needs to meet customer demands in a timely manner
- Collaborate better because of the unified applications and Kubernetes observability

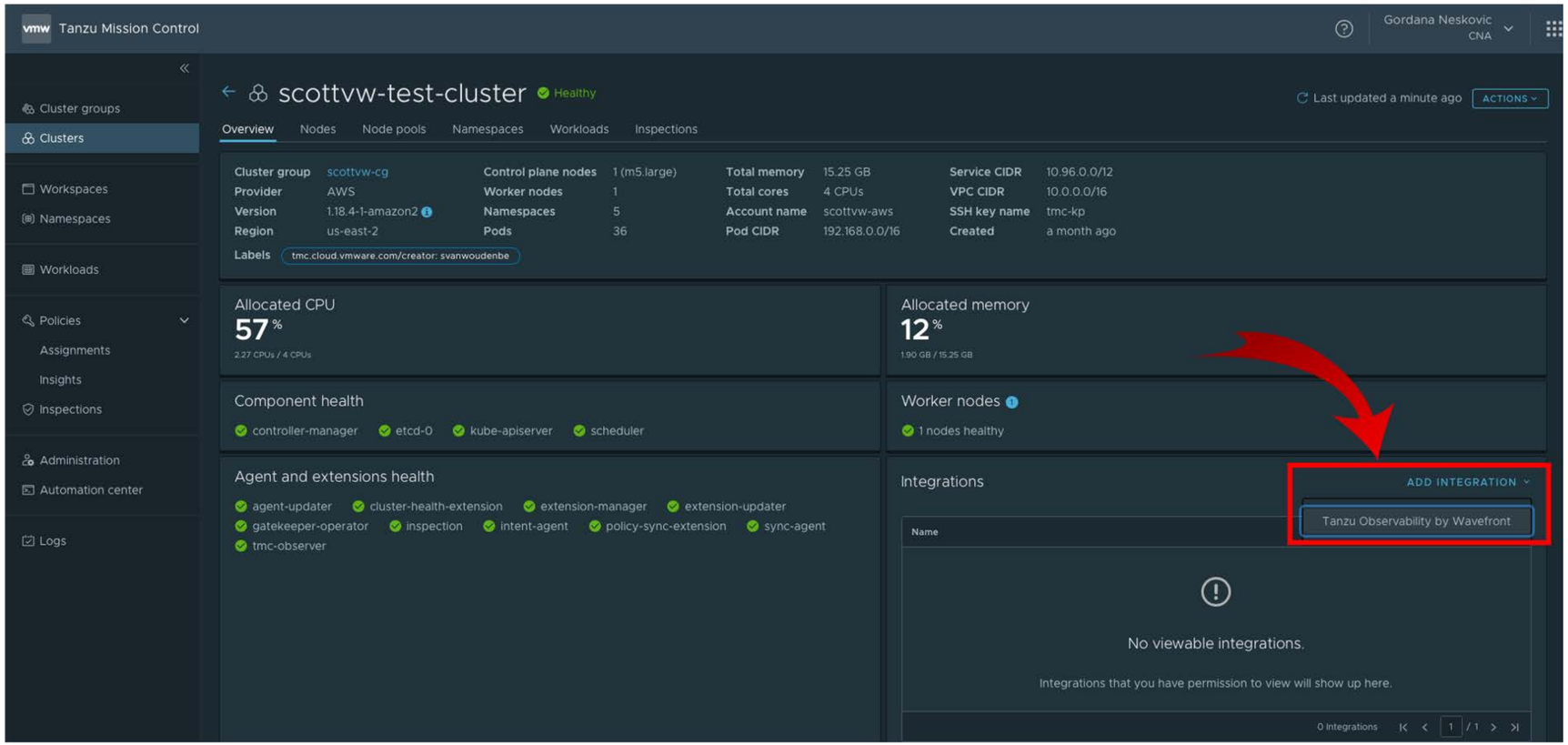


FIGURE 11: Use One-Click Drill-down on Tanzu Observability Dashboards for Troubleshooting and Faster Resolution of Issues

VMware Tanzu Kubernetes Grid Built-In Integration with Tanzu Observability for Holistic Visibility

VMware Tanzu Kubernetes Grid (TKG) enables provisioning, operations, and management of enterprise-grade Kubernetes clusters. It takes care of container deployments from the application layer to the infrastructure layer with built-in high availability, autoscaling, health-checks, self-healing, and rolling upgrades for Kubernetes clusters.

VMware TKG delivers a built-in *integration* with Tanzu Observability for complete Kubernetes at-scale observability. This integration uses the Tanzu Observability Collector for Kubernetes and *kube-state-metrics* to collect detailed metrics about containers, namespaces, nodes, pods, deployments, services, and the cluster itself. It also provides a set of predefined monitoring alerts for important KPIs across TKG Kubernetes clusters and several predefined dashboards that you can further customize.

Tanzu Observability built-in integration with TKG enables developers and DevOps teams to:

- Find leading problem indicators of containerized applications—at any scale—using real-time analytics
- Improve productivity with self-serving, customizable Kubernetes container and applications dashboards and metrics
- Detect containerized cloud service resource bottlenecks proactively with intelligent, query-driven alerts
- Correlate top-level application metrics with Kubernetes orchestration metrics, down to the container and resource level

FIGURE 12: Tanzu Observability Built-In Integration Dashboard—Complete Visibility into Each Level of VMware TKG Cluster

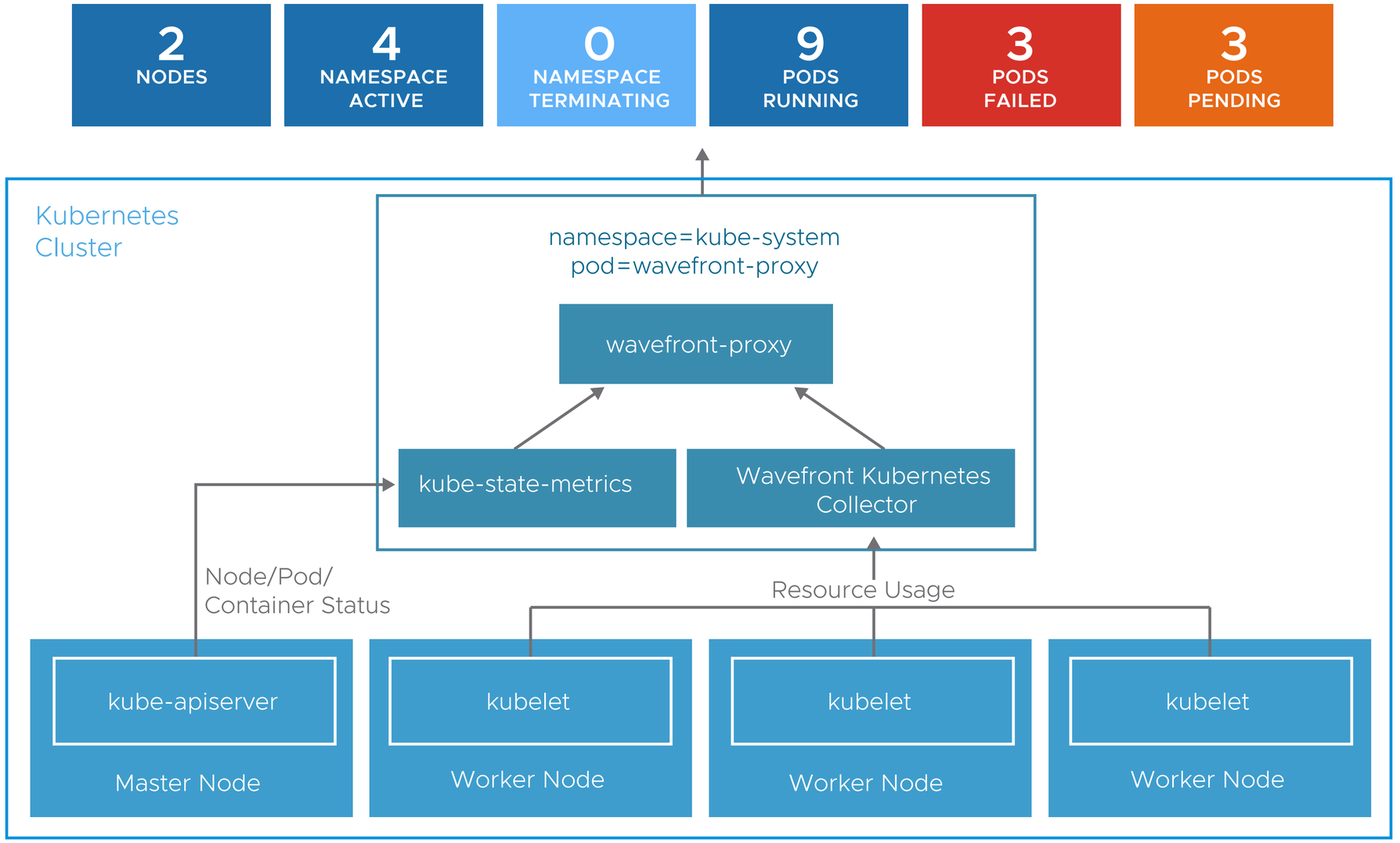


FIGURE 13: The Architecture of Tanzu Observability for VMware TKG—Tanzu Observability Runs as a Wavefront Proxy Pod with Four Containers Inside Each TKG-Created Kubernetes Cluster.

OpenShift Observability—Automated, Full-Stack, and Unified

Red Hat *OpenShift* is a hybrid cloud, enterprise Kubernetes application platform for the development, deployment, and management of modern applications on any cloud infrastructure. It includes software-defined networking, integrated private container registry, streamlined workflows, and easy access to services (service brokers, validated third-party solutions, and Kubernetes operators).

The Tanzu Observability OpenShift integration gives operators, SRE, and DevOps teams complete insight into the health and performance of their large scale OpenShift Kubernetes implementation across containerized applications, Kubernetes, and the underlying infrastructure. As a result of Tanzu Observability's collaboration with Red Hat, automated enterprise observability for OpenShift is now available as part of the Red Hat Certified Tanzu Observability Operator for OpenShift 4.x in *Operator Hub*, a registry for finding Kubernetes operator-backed services. This operator installs, upgrades, and continuously checks the health of the *Wavefront Collector for Kubernetes*. It reduces costs and risks of managing observability for OpenShift environments and applications at scale. It also improves time-to-value by pretesting and expediting the Wavefront Collector for Kubernetes deployment and configuration.

With the Tanzu Observability OpenShift observability and the Red Hat OpenShift Certified Tanzu Observability Operator, engineers, developers, and OpenShift operators get:

- Accelerated and automated transition into Kubernetes and application observability
- Streamlined Day-2 observability operations—from deployments of the Wavefront Collector for Kubernetes and Tanzu Observability proxies to managed configurations and upgrades
- *Automated full-stack enterprise observability* and deep insight analytics across OpenShift environments
- Uninterrupted data ingestion during the spin up and down of hundreds of containers with new IDs
- No running blind during a major update/code deployment that refreshes the container fleet
- Instant alert using fast metric queries across multiple clouds and at a large scale
- Quick and correct troubleshooting without pre-aggregation or inline aggregation of the containers' data

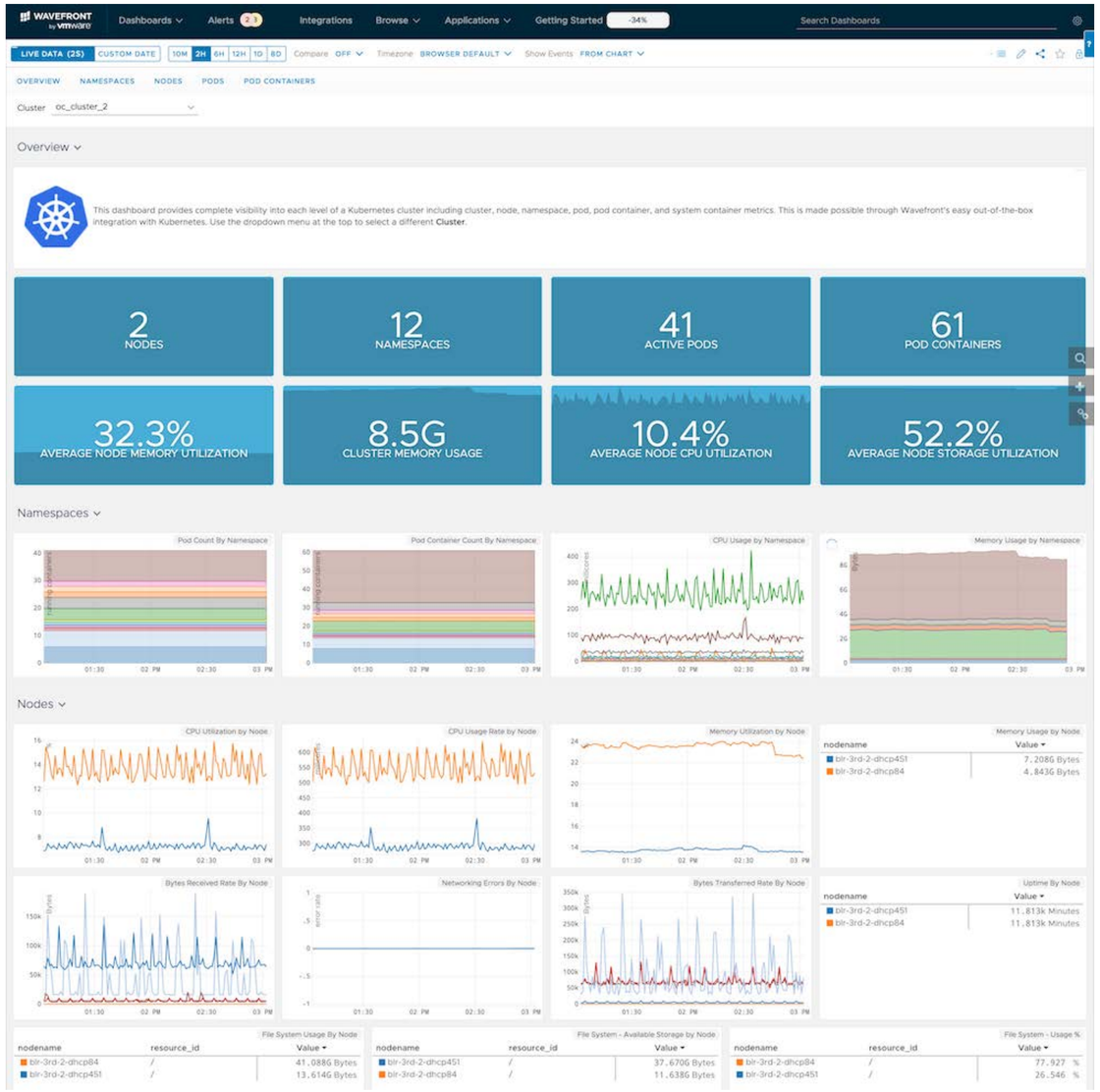


FIGURE 14: Tanzu Observability Provides Complete Visibility into OpenShift Clusters, Nodes, Namespaces, Pods, Containers

How Tanzu Observability Makes Prometheus Enterprise-Ready

As mentioned above, Prometheus is an open-source tool to get started with metrics monitoring quickly. It was designed to handle ephemeral Kubernetes workloads. It is freely available on GitHub as well as sponsored and maintained by the Cloud Native Computing Foundation (CNCF). Prometheus scrapes HTTP endpoints to collect metrics from Kubernetes along with instrumented services and applications. The Prometheus server collects metrics, stores them locally, and provides a simple web UI. Alerts and dashboards have to be managed through separate service or different tool.

Running Prometheus at scale is challenging. It doesn't provide a single place for all your metrics, persistent data storage, robust data analysis tools, smart-flexible alerting, or high availability configuration.

FEATURE	PROMETHEUS	TANZU OBSERVABILITY
Maintenance	Requires dedicated HW & engineering resources to run	No HW or engineering resources for Tanzu Observability as a hosted solution—ingests data from both on-prem and multi-cloud
Enterprise Ready	No	✓ Granular RBAC and detailed policy control ✓ Supports 1,000s users/tenant ✓ Per team usage consumption metering & reporting ✓ Fully managed with ultra-high reliability and fail-over redundancy (multi-zone, multi-region)
UI	Simple	Enterprise-grade: ✓ Automatic metrics discovery ✓ Drag-and-drop chart creation ✓ Widget-based dashboard customization
Dashboards	Simple	Rich, analytics-driven
Scope	Kubernetes	Full stack: from applications to infrastructure

TABLE 5: Prometheus vs. Tanzu Observability Features Comparison

FEATURE	PROMETHEUS	TANZU OBSERVABILITY
Scale	For a few components	Up to 2,000,000 pps, 200,000+ containers
Granularity	Limited	Up to 1 sec (sub-second with histograms)
Data Retention	• 15 days • 2-hours block compaction • Local on-disk storage: ✗ Not clustered ✗ Not replicated	• 18 months • 100% full fidelity (no downsampling) • Up to petabytes of retained data: ✓ Clustered ✓ Replicated
Configuration	Prometheus server	• Telegraf agent with Prometheus plugin (replacement) • Prometheus storage adapter (extension)
Telemetry	Metrics	Metrics, traces, histograms, span logs
Analytics	PromQL (39 functions)	110+ analytical functions, flexible query-builder
Alerting	Basic	Smart, analytics-driven, and flexible
Environments	Static	Static and dynamic
Security	Completely lacking, with: ✗ No authentication ✗ No authorization ✗ No data encryption	Enterprise-ready security with: ✓ Authentication ✓ Authorization ✓ Encryption
Performance	Unoptimized: ✗ Data blackouts on a large scale ✗ Slow queries on a large scale ✗ Must be provisioned for burst peak loads	Optimized: ✓ Uninterrupted on a large scale ✓ Fast queries on a large scale (high *cardinality* resilience) ✓ High availability (HPA included)

TABLE 5: Prometheus vs. Tanzu Observability Features Comparison

Tanzu Observability can quickly resolve these problems. You can replace the Prometheus server (static integration) or extend Prometheus into an enterprise-grade and hyper-scale solution (dynamic integration). Both *Tanzu Observability integrations* treat Prometheus metrics the same as any other data, and they offer access to a reliable database that supports fast querying, analytics-driven dashboards, and a powerful alerting interface.

- Static integration is excellent for monitoring static environments where the list of HTTP endpoints does not change often. It replaces the Prometheus server with a Telegraf agent (and the *Prometheus input plug-in* for Telegraf), which scrapes the HTTP end points directly, converts the data to the *Wavefront format*, and sends those data to the Tanzu Observability service.
- Dynamic integration preserves existing service discovery by keeping the Prometheus server as a collector agent that forwards metrics into Tanzu Observability through the *Prometheus storage adapter*.

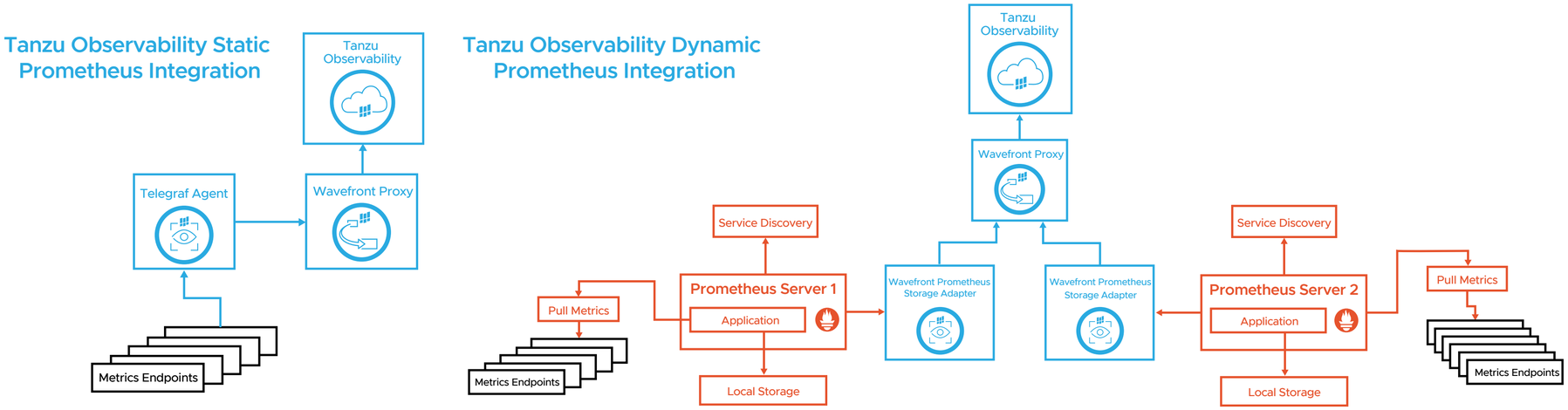
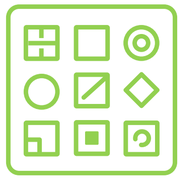


FIGURE 15: Tanzu Observability Static and Dynamic Integrations for Upgrading Prometheus Monitoring into Enterprise-Ready Observability

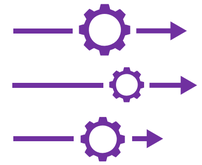
With the Tanzu Observability integrations, engineers can take advantage of hyper-scale clustering, long-term storage, and enterprise readiness controls. They also have access to intelligent dashboards, powerful querying, and smart, flexible alerting. Most importantly, no matter whether it's a few Prometheus endpoints or hundreds of Prometheus servers, Tanzu Observability reliably scales with your growing monitoring and observability needs.



Scale (built-in) beyond monitoring 100s containers/hosts with HA

Perform 100x better at handling inherent cardinality

Use the past to help predict and troubleshoot the future



High-fidelity monitoring of ALL metrics in one place w/context

Customers who matured their observability from Prometheus to Tanzu Observability Enterprise

workday. Postmates DOORDASH BOXEVER okta STEELHOUSE

FIGURE 16: Enterprise Observability of Tanzu Observability

How to Use Tanzu Observability for Kubernetes Observability

Alerting Across the Full Stack

Alerting is one of the main pillars of Kubernetes observability. Alerts fire when KPI Kubernetes metrics and Kubernetes events of interest indicate issues or potential issues, as and when they occur. If the Kubernetes problem is noticed only when the user is complaining about failures of an application running on Kubernetes, there is a Kubernetes monitoring and alerting gap. Incidents are inevitable, but customer-reported incidents should not exist in a Kubernetes that is observed continuously.

For effective Kubernetes environment monitoring, you can alert on the following metrics:

- Application layer metrics—service response time, service availability up or down, SLA compliance, successful or error requests per second
- API metrics such as error, request, and latency—high response time, dropped requests, and large variance in response time point to service component degradation
- REST API requests on the services load balancers
- Services running on Kubernetes—HTTP requests, database connections and replication, threads/file descriptors/connections, middleware specific metrics such as JVM heap size
- Kubernetes infrastructure—meeting requirements, up and running
- Host/Node layer—up & down/unreachable, resources availability (CPU, memory, disk)
- High disk usage—the most common, usually indicating an application issue
- Master nodes degradation—points to DNS bottlenecks, network overload, etcd issues
- Pod's lifecycle—pending, running, succeeded, failed, CrashLoopBackoff
- Pattern changes on the set of containers—it's noisy to measure individual container resources

Tanzu Observability gathers detailed information about the operation in the Kubernetes environment. For example, Tanzu Observability informs engineering teams about utilization extremes, SLA violations due to increased latency or error rate per service, excessive pod restarts, and paused or stuck deployments. Tanzu Observability auto-configures a set of Kubernetes-related alerts. These alerts enable engineering teams to detect and resolve issues before they can impact production.

With Tanzu Observability, engineers can create alerts that dynamically filter noise and capture true anomalies:

- Smart streaming real-time alerts—for faster automation
- Regular *multi-threshold alerts*—for aggregating data across different hosts or time series

Upon receiving an alert, engineers are directed to the Tanzu Observability alert viewer—a single landing page for an alert—for fast incident triaging and resolution. Tanzu Observability alert viewer powered by AI/ML algorithms surfaces related alerts and tags. It automatically highlights point tags shared by affected time series, giving a clear indication of what might be going wrong, and empowers developers and SREs to spot dependencies across firing alerts instantly. Based on correlation and AI/ML algorithms, it also lists and ranks other synchronously-firing alerts and events across the full stack in application services, Kubernetes, and the enterprise multi-cloud. This enables engineers to easily search and filter through key evidence and quickly find the root cause of the incident.

"Wavefront has a great UI for creating truly intelligent, dynamic alerts. Its query language is the best out there, and we love how alert creation is so well integrated right within the dashboard, not as some separate tool within the platform."

JULIEN LEMOINE
CO-FOUNDER AND CTO
ALGOLIA

Guided Troubleshooting

Troubleshooting Kubernetes is complex, and it is difficult to account for all situations. Kubernetes API may not be reachable, etcd may fail on one or more master nodes, the Kubernetes cluster may not see persistent volumes, the deployment service may not be reachable at all, or only internally, and the pod may run, but be unresponsive. For all these situations, the Tanzu Observability curated out-of-the-box Kubernetes dashboards can help you understand the status of your services and to resolve incidents faster.

"Wavefront's powerful query language allows us to visualize and debug our time series telemetry data easily. It's well-tuned alerting helps us lower the MTTD. Our engineers customize their metrics to monitor the health and performance of their systems fully."

JING ZHAO
DEVOPS SOFTWARE ENGINEER
DOORDASH

Tanzu Observability out-of-the-box Kubernetes dashboards provide Kubernetes platform operators, SREs and developers with:

- Resource metrics—important for investigating and diagnosing utilization, saturation, errors, and availability issues
- Events—provides a context for understanding changes in the system's behavior such as code releases, builds, and scaling. They are used for troubleshooting issues and correlating information across systems
- Labeling—enables filtering and aggregating data across different time zones, instance types, software versions, services, roles, etc.

These metrics and events, which are labeled for important dimensions (regions, software versions, etc.), are simple to understand, they capture system behavior and are retained, making it possible to define seasonality and standard behavior.

Engineers usually start with the high-level metrics that point to the issue in the system. The next step is to examine the physical system resources: physical resources, software, and external services. Tanzu Observability Kubernetes dashboards display high-level and resource key metrics of each subsystem/component layer with overlaid relevant events. Easy navigation makes it possible to investigate the issue quickly. While investigating Tanzu Observability users can examine related alerts and events that are reported in the same time window—they might be correlated and connected with the problem they are troubleshooting.

The Tanzu Observability Kubernetes troubleshooting dashboards leverage the data that Tanzu Observability already collects. When certain important metrics are out of predefined thresholds or when a known event occurs, Tanzu Observability surfaces the metrics and data engineers need to troubleshoot the issue. By only displaying relevant information, Tanzu Observability protects users from being overwhelmed in times of crisis; therefore, they can focus on the problem at hand.

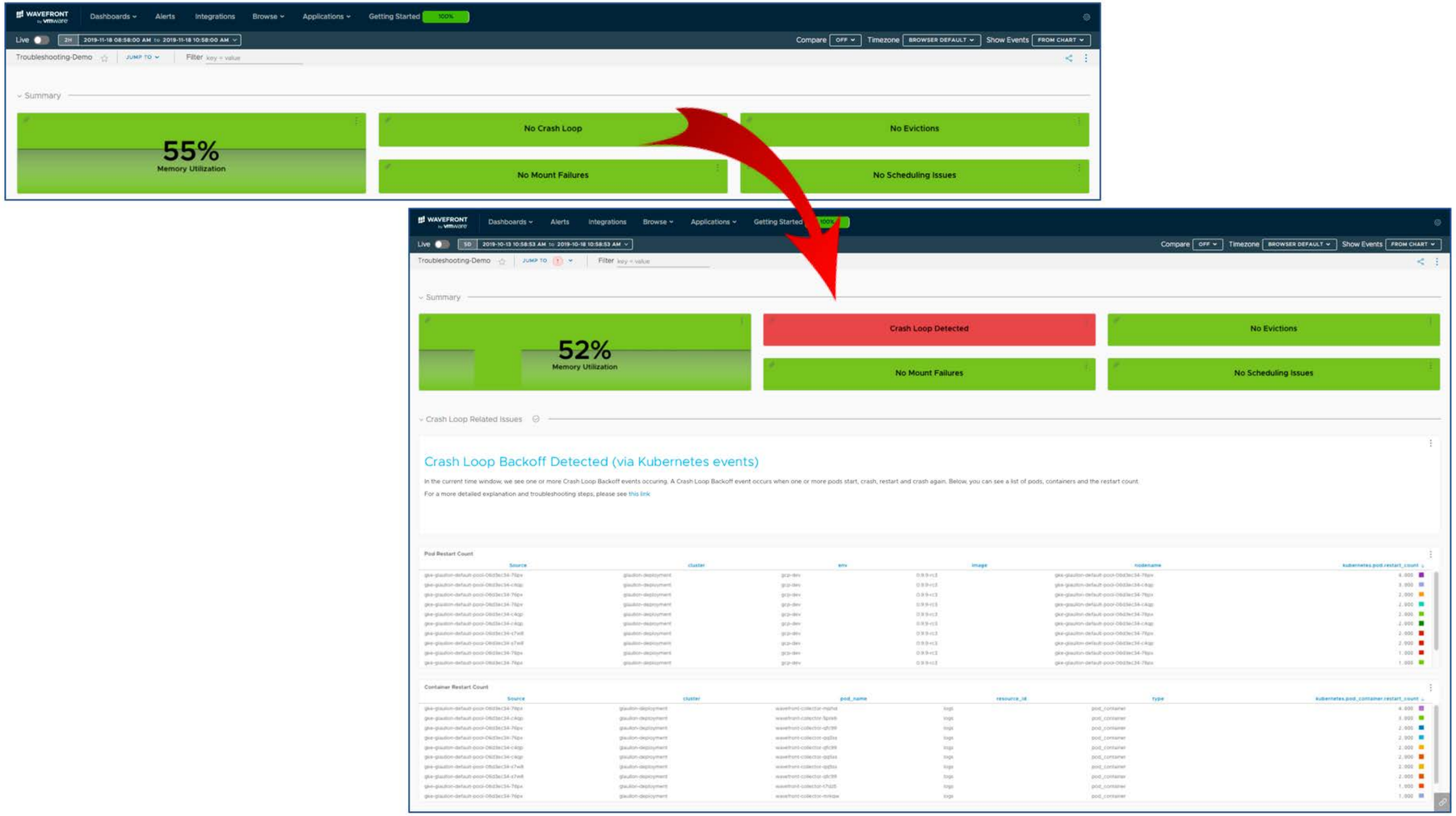


FIGURE 17: Tanzu Observability Packaged Kubernetes Troubleshooting Dashboard Surfacing Information in Time of Crisis

Optimizing with Deep Understanding

When dealing with a complex Kubernetes-based infrastructure, performance tuning must be done all across the stack, including the host, cluster, container, networking, and the applications on top. For example, all containers and pods are usually not of the same nature. Some need more CPU-optimized instances, and others may run more effectively on I/O-optimized instances. Because Kubernetes does not optimize performance on its own, engineers have to identify the types of instances.

Scaling/Tuning	Description
Horizontal Cluster Scaling	Adding nodes
Vertical Cluster Scaling	Using a different type of EC2 instances
Horizontal Pods Scaling	Increasing the replica count
Vertical Pod Scaling	Increasing CPU and memory limits
Horizontal Ingress Scaling	Increasing the replica count
Vertical Ingress Scaling	Increasing CPU and memory limits
Tuning NGINX	Tuning parameters: timeouts, worker processes logs, HTTP
Optimizing Databases	Optimizing queries and indexes

TABLE 4: Scaling and Optimizing a Kubernetes Cluster and its Components

To address optimization, Kubernetes operators and architects must understand Kubernetes performance features and how to work with them. They want to know the answer to questions such as:

- Do some pods have more latency than their peers in a cluster?
- Are there hotspots?
- Are there performance hits from multitenancy?
- Is autoscaling the reason behind performance lag?
- Is my overall CPU usage fine, but are some nodes CPU starved?
- Are my applications and underlying dependencies hampering performance?

Engineers perform ramp-up tests to adjust resource limits close to optimal performance and duration tests to identify memory leaks and hidden queuing. Engineers perform all these tests to understand an application's different failure modes before reaching production.

Also, one part of optimization is related to changing the system of service to avoid problems that have been previously troubleshooted and corrected. Engineer teams want to prevent similar issues in the future.

"Wavefront handles high-velocity metrics with relative ease. It retains and visualizes historical data with full fidelity together with real-time data, helping us track and anticipate seasonalities. These unique capabilities allow us to be even more operationally responsive to our business."

KEVIN CANTONI
VP PRODUCT DEVELOPMENT
WORKDAY

Tanzu Observability automated and unified enterprise observability for Kubernetes and applications running on multi-cloud environments, enabling developers, SREs, and Kubernetes operators to get the most from their environments while providing the best user experience:

- See the real-time impact of code in production—deliver faster high-quality code with full-stack visibility of released code by examining the impact of changes on every aspect of Kubernetes environment
- Run the Kubernetes environment with tighter margins—scale and optimize your Kubernetes cluster and its components dynamically with no performance degradation
- Map Kubernetes cloud resources to pricing and cut your public cloud bill—get visibility and optimize Kubernetes cloud resources through Tanzu Observability integrations with leading cloud providers
- Improve Kubernetes capacity forecasting with AI Genie automation—auto-detect anomalies and hidden trends, and get automatic prediction with no parameter tune-up
- Discover Kubernetes applications performance bottlenecks—get instant visibility into request flow, service dependencies and performance bottlenecks with OOTB application observability with the *Wavefront Java Tracing Agent*

Tanzu Observability provides support for optimization of your Kubernetes environment by delivering observability during all stages of Kubernetes adoption, from Day-0 getting CI/CD pipeline ready and Day-1 mapping performance to resource needs to Day-2 scaling reliably and efficiently in production.

Monitoring of Service Level Objectives for All Teams

It's the joint responsibility of all engineering teams to make sure that overall Service Level Objectives (SLOs) are being met.

- Developers translate business requirements from new features and bug fixes into a code. For improving software quality, developers define SLOs using metrics such as page load times, caching behavior, and transaction times
- Kubernetes operators and SREs take care of platform services, such as workload orchestration and database operations that are critical dependencies to applications and services on top. They use SLOs to support a requested quality of high-velocity software release lifecycle

Data-driven decisions to meet end-user expectations, are made based on correlating developers, SREs, and Kubernetes operators SLOs with business SLOs based on metrics such as new customer registrations or active time spent with the application.

To achieve a good Kubernetes application SLO, i.e., clear target around Kubernetes application performance, which will balance feature development with platform stability, an nterprise observability platform like Tanzu Observability is essential.

With Tanzu Observability, SLOs are comprehensively monitored and shared as a service across all teams in the enterprise. They give service providers the information they need to control product enhancements, maintenance, and cost efficiency within the error budget.

"Wavefront products provide unparalleled visibility across all of our Kubernetes-enabled, containerized apps, helping all of our developers become more productive focusing on innovation, while enabling 8x8 to deliver exceptional SLAs and eliminate any issues with our cloud services."

DEJAN DEKLICH,
CHIEF PRODUCTS OFFICER
8X8

Tanzu Observability enables Kubernetes platform operators, SREs, and developers to track, manage, and monitor Kubernetes application SLOs such as the ratio of successful requests for availability or the 99th percentile of request times for latency in one place.

Histograms are a variation of metrics that record the detailed metric distribution over set time periods. Histograms are ideal for high-velocity metrics and for situations where it is essential to retain outlier information that becomes lost in averages, e.g., latencies, and percentile metrics used to track service level indicators/objectives. The *Tanzu Observability histogram* is the most accurate solution for monitoring high-velocity data from different sources without compromising scale or losing valuable reporting data. With Tanzu Observability histograms, engineers can reliably measure and aggregate quantiles/percentiles of high-velocity metrics such as application response times and service SLOs.

Finally, *Tanzu Observability SLO* alerts can be set for different burn rates or how fast a service consumes the error budget together with multiple time windows—shorter for immediate alerts and longer for long-term observations that lead to SLO breaches over time.

Real-World Kubernetes Observability Case Studies

Observability for the Front Internet Page

A company known for its social news aggregation, web content rating, and discussion website, which has over hundreds of millions of monthly visitors and ranks within the top 10 most visited websites in the United States and worldwide, recently migrated to containerized microservices and Kubernetes recently and hired a team of new software engineers that wanted more visibility into their production code. The migration led to a massive increase for metrics and a realization that its open source-based monitoring platform was not scaling and wasn't easy to support.

FIGURE 18: Tanzu Observability Unified Kubernetes and Applications Observability for Well-Known Social News Aggregation, Web-Content Rating, and Discussion Website Company

The company transitioned to Tanzu Observability because they needed Tanzu Observability's powerful and flexible handling of metrics time series data. All metrics from Tanzu Observability integrations, custom application metrics reported through *Telegraf* or *StatsD*, and a mix of *Prometheus*/Kubernetes system metrics from multiple Kubernetes clusters running on *AWS* and *GCP* are fed through a Tanzu Observability proxy into Tanzu Observability for visualization and storage.

Engineers find different ways of data visualization that help identify potential issues. For example, color-node maps for Kubernetes health allow them to monitor a farm of machines and at a glance, identify which nodes have potential issues. Tanzu Observability also helps manage complexity and the sheer amount of Kubernetes metrics by making metrics useful and more understandable with layered, connected dashboards with more granularity as engineers drill down deeper. Tanzu Observability standardized dashboards simplify dealing with unfamiliar environments, services, and dependency, making debugging edge cases easier and faster. Also, Tanzu Observability's advanced analytics engine and *query language* accelerate anomaly detection for alerting and troubleshooting their Kubernetes environment of tens of thousands of pods.

The powerful Tanzu Observability query language was instrumental in calculating complex SLAs. Engineers have specific procedures for keeping all their services within SLAs. When developers want to launch a new service, they follow the best practices for generating SLA charts and alerts that they have to monitor. SREs review the Tanzu Observability SLA dashboards as well to identify hot spots in Kubernetes environments. Increased team confidence resulted in taking more ownership of performance and monitoring SLAs.

Tanzu Observability metrics dashboards for developers, SREs, and Kubernetes operators have overlaps. All these users monitor systems' Kubernetes-based metrics such as CPU, memory, I/O, the number of instances, network metrics, and Kubernetes metrics. However, SREs are interested in behavior across the whole fleet, while developers want to understand which changes might be caused by issues with the code.

Developers own unique sets of application microservices metrics that depend on business logic, and they want to know if resource spikes are due to more load or an introduced bug. For example, if developers are running a web server and serving many requests per second, and then they do a code release, and the size of Kubernetes infrastructure goes up, but the request performance doesn't, then they know that the release code increased the resources used.

With Tanzu Observability engineers have the metrics and ability to monitor their services within the Kubernetes environment. They extensively use Tanzu Observability's *programmatic application of dashboards*, *alert backtesting*, alert customization, and integration with PagerDuty and Slack. They use *Wavefront CLI* to see what metrics are used within the Tanzu Observability ecosystem and what percentage is relevant for developers on a daily basis. They also find the URL share mechanism very helpful for communicating with the team during incident response.

The company has rolled out Tanzu Observability to all software engineering teams of several hundred engineers and operations teams. With Tanzu Observability's comprehensive dashboards, powerful analytics, and intelligent summaries and alerts, engineers have a complete picture and all the information they need to increase performance, stability, and resiliency of applications, Kubernetes components, and underlying infrastructure.

How VMware Cloud Engineering Team Exceeds SLAs

The VMware cloud engineering team is responsible for delivering critical cloud services with strict SLAs to internal and external stakeholders. They are using Tanzu Observability to observe their environment:

- 650 cloud and on-prem pipelines
- Multiple GEOs
- 5,000 daily-deployed containers across 12+ Kubernetes clusters with 100+ application microservices

Their Tanzu Observability instance includes 600 alerts and over 500 dashboards.

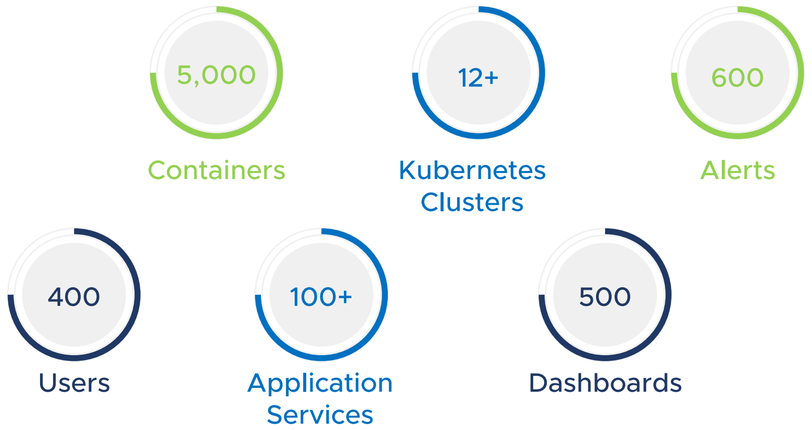


FIGURE 19: Enterprise Observability of Tanzu Observability for Cloud Management Platform at Scale

Tanzu Observability is their primary platform for unified applications and Kubernetes's full-stack observability. They use Tanzu Observability for reliability and health analysis across all microservices, the build pipelines, and the hybrid cloud infrastructure. They use the Kubernetes integration to observe Kubernetes on-prem and on cloud performance health and to correlate across containerized microservices and underlying infrastructure.

With 900 pipeline executions per day, the SRE team automated the entire pipeline. Now they need constant visibility to make sure that nothing is stuck at any point. Tanzu Observability dashboards show all the pipeline metrics such as Jenkins server and worker metrics and Gerrit code review metrics. By analyzing services, Kubernetes, and supporting infrastructure, the SRE team can guarantee continuous running while hitting SLAs.

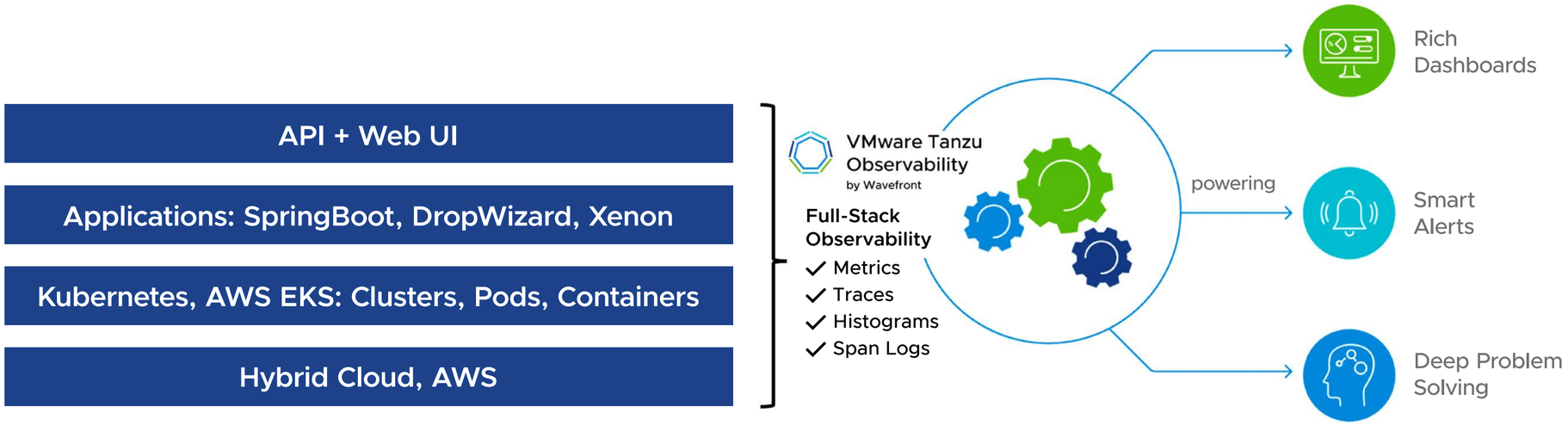


FIGURE 20: Tanzu Observability Kubernetes and AWS EKS Observability Full Stack

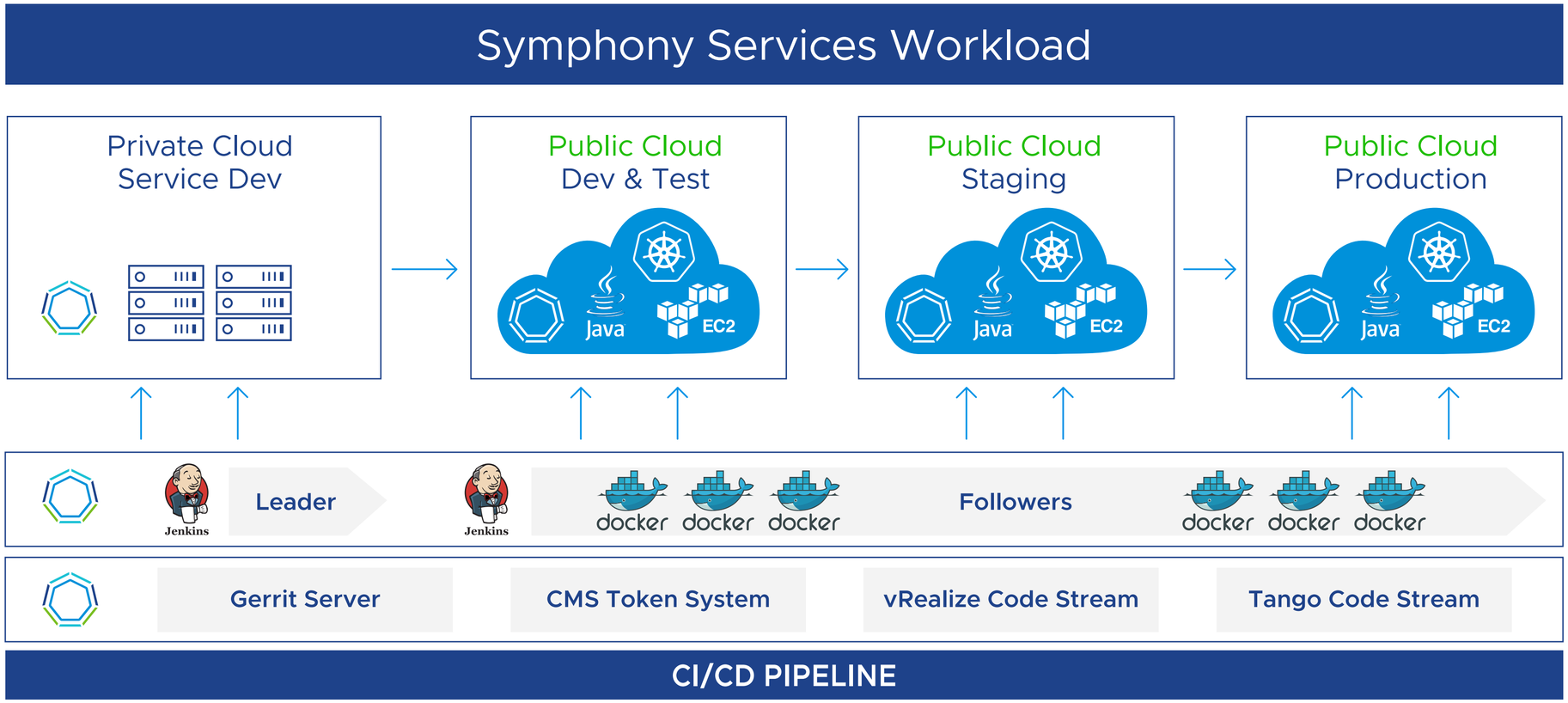


FIGURE 21: CI/CD On-Prem and Cloud Pipeline Observability with Tanzu Observability

Recently, the team has used Tanzu Observability to compare the success rate of Python scripts deployments and Helm charts deployments. The results allowed the teams to agree to switch to Helm on EKS. Tanzu Observability dashboards enabled them to go and look at pipeline failure runs, drill into the most frequently failing jobs, and figure out the root cause of failures. They were able to find out if failures were across the board, and if the problems were in the Kubernetes clusters or with the particular service.

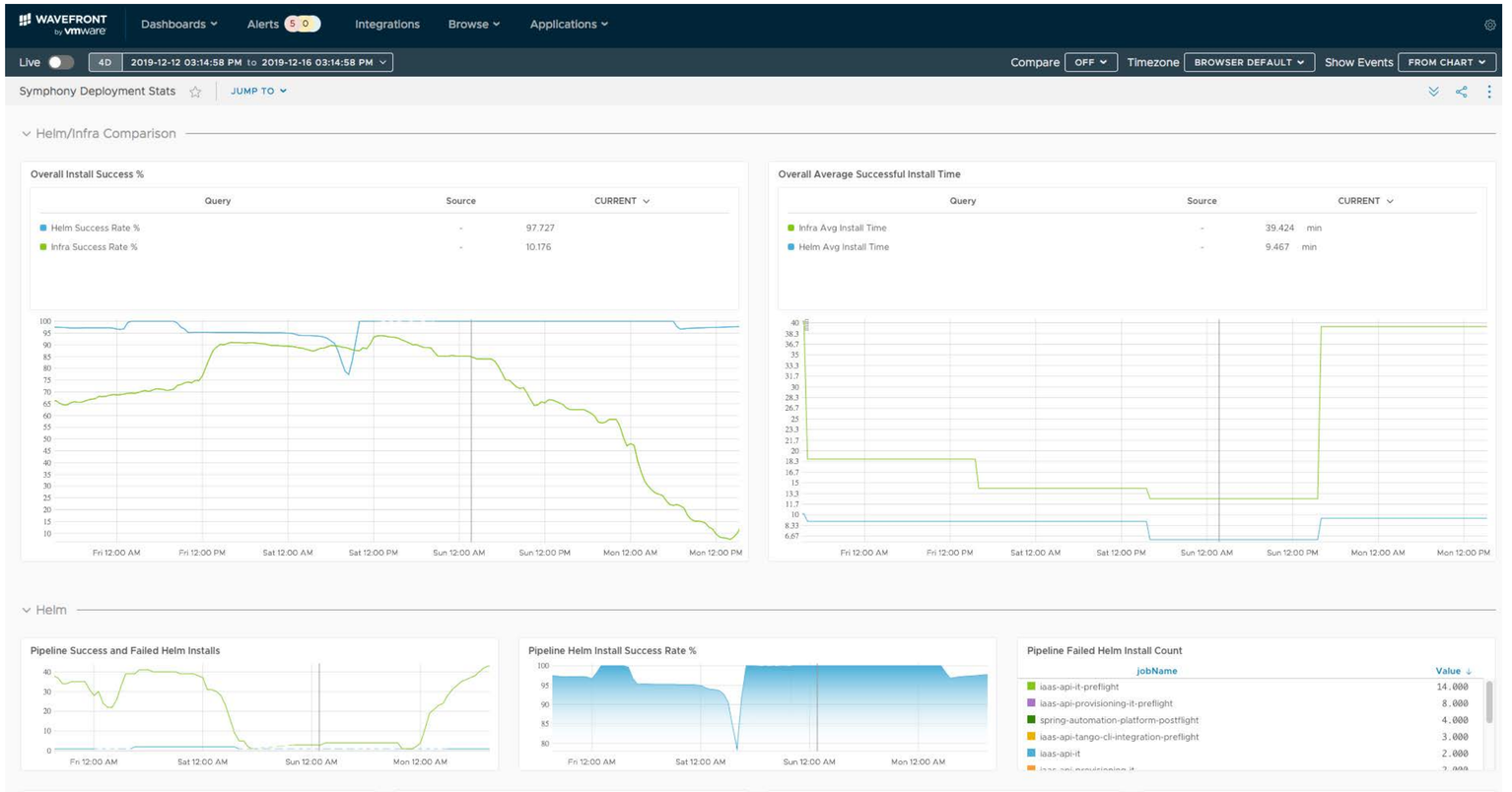


FIGURE 22: Switching to Helm on EKS Based on Tanzu Observability Tracked Success Rate

The Tanzu Observability Kubernetes integration enables Kubernetes operators and SREs to understand how Kubernetes behaves across all levels and to perform guided root cause analysis (RCA). For example, they can easily drill down and find out if a pod restart is due to an uncaught exception, CPU/memory limit, or restart of the node itself.

Developers see what happens with the service they own from code check-in through production under load. With Tanzu Observability, all engineering teams have full insight into the Kubernetes application environment, and can easily collaborate, and find issues quickly.

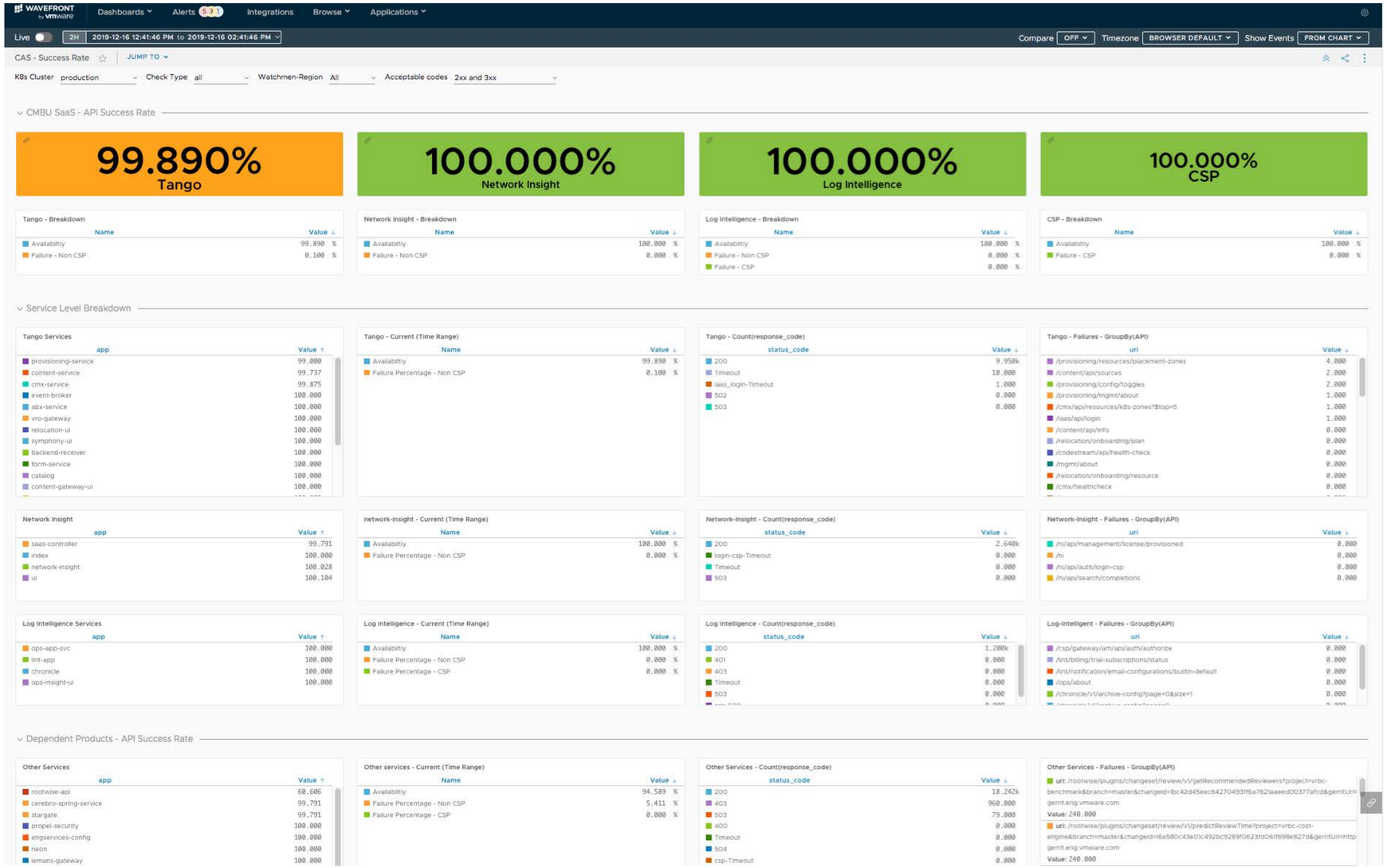


FIGURE 23: A Glance into the Health of All Deployed Kubernetes Services

What to Look for in an Enterprise Observability Platform

To confidently deliver observability across the organization, Kubernetes Operators, SREs, and developers should look for several features:

Real-Time Performance at Scale:

The Tanzu Observability patented architecture is proven to scale to millions of metrics per second. Many customers with thousands of active users use Tanzu Observability to set alerts, run queries, and view dashboards. The platform is optimized to handle high *cardinality* metrics at scale, delivering real-time visibility regardless of the telemetry volume.

Detailed Platform Self-Reporting:

Tanzu Observability provides a real-time portal on its service status and usage. Track how Tanzu Observability is consumed by team, project, and domain. Set usage quotas per team report to maintain cost budgets.

User Customization at Scale:

Tanzu Observability enables user teams to self-service their telemetry and create their alerts and dashboards at scale. Intuitive query builders and creation tools for alerts and dashboards make it easy for all teams to deliver on SLOs for their services.

High Availability Architecture:

Tanzu Observability has a high availability architecture with a 99.95% SLA using multiple availability zones across multiple regions. It keeps 4 copies of your data at all times, and the proxy queues your data even if your WAN link goes down. The Tanzu Observability Cloud auto-indexes and shards, adapting to your data shape.

Granular Policy Control:

Tanzu Observability enables DevOps teams to support 100s of teams based on policy, define access to assets with granular user groups, protect sensitive metrics with permission controls, encrypt data at multiple levels, and support multi-tenant SSO.

Full Programmable Automation:

Tanzu Observability provides a complete API and CLI for all its UX capabilities, with version control, facilitating automation and monitoring as code. Tanzu Observability integrates with run-book automation and continuous delivery platforms to automate remediation and rollbacks.

What to Monitor in a Kubernetes Environment

Kubernetes is not self-monitoring, and all the components in the Kubernetes environment, from user experience to infrastructure need to be included for effective monitoring of Kubernetes deployments. A layered approach to monitoring makes it easy to identify the problem by drilling down through the layers until the issues are identified.

Finally, a different level of aggregation is essential for getting comprehensive observability and successful troubleshooting. For example, when aggregating per node, infrastructure issues might easier surface, while aggregation per application allows easier microservices root causing.

User Experience Metrics:

Get insight into the user experience by monitoring response time, errors, load time, and availability of mobile applications and browsers.

Application Metrics:

Gain insight into the health and performance of Kubernetes applications by monitoring custom metrics (requires deep application understanding) and RED (Rate, Errors, Duration) metrics.

Network I/O Metrics:

Monitor bytes and errors received and transmitted because they are good indicators of an underperforming network that influences application performance.

Kubernetes Metrics:

Clusters—CPU and memory usage, pod/container counts

Nodes—CPU and memory usage, network, storage, uptime, filesystem

Pods—CPU and memory usage, network, storage, uptime

Containers—CPU and memory usage, storage, uptime, filesystem

Infrastructure Metrics:

Put the accent on the utilization of resources by monitoring USE (Utilization, Saturation, Errors) metrics.

Want to Learn More?

Videos and Webinars:

- *Tanzu Observability and Kubernetes*
- *Tanzu Observability and Kubernetes: The Next Generation*
- *The Next Generation Kubernetes Experience in Tanzu Observability*
- *Container Monitoring Best Practices*

Docs and GitHub:

- *Monitor and Scale Kubernetes with Tanzu Observability*
- *Kubernetes Integration*
- *GitHub: Wavefront Collector for Kubernetes*
- *VMware TKG Integration*

Blogs:

- *Tanzu Observability's Next-Gen Collector for Kubernetes*
- *Tanzu Observability Automates Observability for Enterprise Kubernetes*
- *Tanzu Observability's Kubernetes Observability Extends Beyond PKS, Cloud TKG, Amazon EKS, Now to OpenShift, GKE, AKS and More*
- *How to Instrument and Monitor Your Spring Boot 2 Application in Kubernetes Using Tanzu Observability*
- *Got penShift? Need Observability That's Automated, Full Stack, and Unified? Make a Shift to Tanzu Observability!*
- *Application Metrics Collection in Kubernetes via Telegraf Sidecars and Tanzu Observability*
- *How to Make Prometheus Monitoring Enterprise Ready*
- *How Moving From Prometheus Monitoring to Enterprise Observability Helped Secure State Deliver Exceptional Cloud Security Services*
- *Integrating Prometheus with Tanzu Observability for Easy Scaling and Failover*

Glossary

Alert Automation	alerts life-cycle management, i.e., creating and maintaining alerts
Annotations	metadata attached to Kubernetes objects
Auto-Remediation	self-healing, resolving alerts automatically
Availability	perform as expected at a given point in time
Burn Rate	how fast relative to SLO the service consumes the error budget
Canary	testing a new release on small subset of typical workload
Capacity Cache	a cache that serves precomputed results for API calls/queries to a service
Cardinality	a number of values in a set
ConfigMap	handles configuration data
Container Image	a collection of all the files that make up an executable application
Control Manager	daemon process that implements Kubernetes control loops (rolling deployments, replica sets, number of worker nodes)
CRD (Custom Resource Definition)	an extension of Kubernetes API for storing and retrieving data
Deployments	manages ReplicaSet, pod definitions etc.
Etcd	key-value store of all Kubernetes objects
Endpoint	source of metrics that can be scraped
Error Budget	determines how unreliable the service is allowed to be i.e. the number of errors within SLO
Exporter	exports metrics from third-party systems as desired metrics
Failover	handling failure by automatically routing incoming requests to a different instance
Fallback	alternative source when given component is unavailable
Helm Chart	Kubernetes specific package manager
Histograms	shows data distribution
Ingress	manager of external HTTP traffic to hosted service

Glossary

Instrumentation	metrics endpoint is embedded within an existing application for measuring and recording quantities and states
Kube-APIServer	gateway to Kubernetes cluster
Kubectl	command line for Kubernetes
Kube-DNS	resolves DNS of all services in Kubernetes cluster
Kube-State Metrics	monitor state of Kubernetes Objects (nodes, pods, jobs, deployments)
Kubelet	initiates pods, interacts with containers
Kube-Proxy	network proxy and load balancer for Kubernetes services
Namespace	virtual segmentation of single clusters
Nodes	host of Kubernetes worker or master components
Operator	domain specific controller
Pods	group of one or more containers
ReplicaSet	continuous loop that ensures that requested number of pods are running
Rollback	revert of a set of previously rolled out changes (configurations/services)
Rollout	service/configuration deployment
RPC	remote procedure call
Saturation	% of maximum capacity
Scheduler	assigns workloads to specific nodes
Service	logical layer that provides IP, DNS etc. persistence to dynamic pos
Sharding	splitting a data structure/service into shards
SLA (Service Level Agreement)	legally binding of keeping certain SLO over certain period; if not penalties paid
SLI (Service Level Indicator)	key metrics as a measurement of service health and performance
SLO (Service Level Objective)	target level of SLIs
Tail latency	a long tail of very large outlier
Traffic	the amount of use of a service per unit of time

Get Started Today

Try Tanzu Observability By VMware For Free

Learn More

Join us online: