Guidebook for Monitoring VMware Tanzu Kubernetes Grid Integrated Environment

Sajal Debnath

Staff Architect – Advanced Customer Engagements

vmware®

Version History

VERSION HISTORY				
DATE	REV	AUTHOR	DESCRIPTION	REVIEWERS
05-30-2020	0.1	Sajal Debnath	Initial Draft	Rag Ramanathan
06-01-2020	1.0	Sajal Debnath	First Publication	Rag Ramanathan
07-02-2020	1.1	Sajal Debnath	First update	
08-19-2020	1.2	Sajal Debnath	Minor corrections	

VMware, Inc.
3401 Hillview Ave
Palo Alto, CA 94304
www.vmware.com

Table of contents

1 Purpose

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

The question under consideration is what to monitor for in the deployed TKGI environment and how to monitor these. The next question is what to do or how to react to the issues we face in the monitored components.

This document provides answers to all these questions. It guides to what to monitor and how to monitor in a datacenter hosting TKGI environment. Also, this document provides best practices and general guidelines for monitoring such an environment. Though there are many ways we can monitor a TKGI environment, this document covers the VMware suite of products that is used to monitor TKGI environment including vRealize Network Insight, vRealize Operations Manager, and vRealize Log Insight.

2 Topics covered

The document is divided into below-mentioned four main categories.

- **Part A**: Monitoring containerized world
  - Monitoring philosophy
  - Monitoring K8s environment
- **Part B**: Monitoring a TKGI environment
  - What to monitor
  - How to monitor
- **Part C**: Handling Failures

In the following sections we will discuss the above-mentioned topics in detail.

Part A

3 Monitoring containerized world

Before we discuss about monitoring a containerized world, I wanted to take a step back and discuss an often-discussed topic i.e. about monitoring, it's importance and why it is more important in today's world. This will also act as background to the main topics discussed in this document.

3.1 Monitoring Philosophy

In any organization, application drives the infrastructure. The infrastructure exists because of applications. If an organization decides only to use public cloud solution, then they will not need any infrastructure (in terms of Server and Storage).

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

- We moved from monolithic applications to modular applications. Which in turn lead 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 advance service offerings we now have cloud native applications which exhibits certain key attributes.
- Also, all these led to change in application development methodology. With the above advancements' organizations can now adopt agile process for application development. An organization can implement agile methodologies through adoption of DevOps process and implementing CI/CD pipelines.

All these lead to a requirement where all the current applications developed by any organization needs to be able to run on a cloud platform. Because sooner or later it will run on a cloud environment be it public, private, hybrid or a multi-cloud environment.

3.1.1 Agile process

Cloud native operations practice Agile and DevOps. Agile process with fast iterations requires well-established feedback loop. Look 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. Also note, this is a continuous process. Meaning we take continuous feedback and improve upon the existing application through the continuous feedback loop.

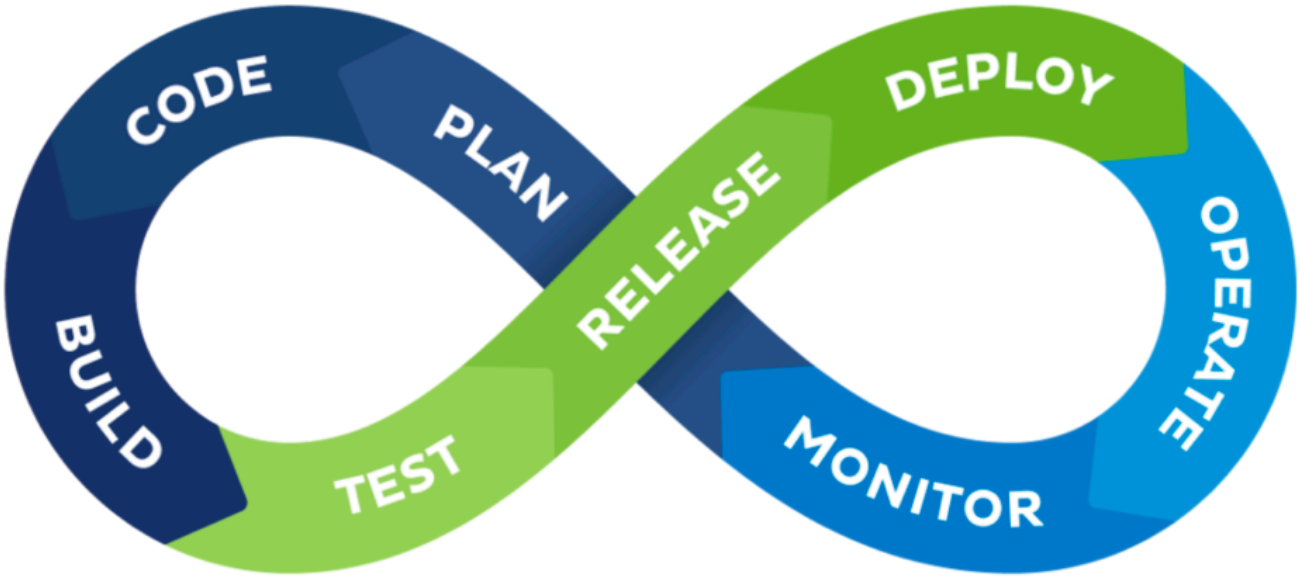

Figure 1. Cloud native operations with agile process

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

3.1.2 Requirement for healthy operations

A healthy operation of applications and platform has the following three layers.

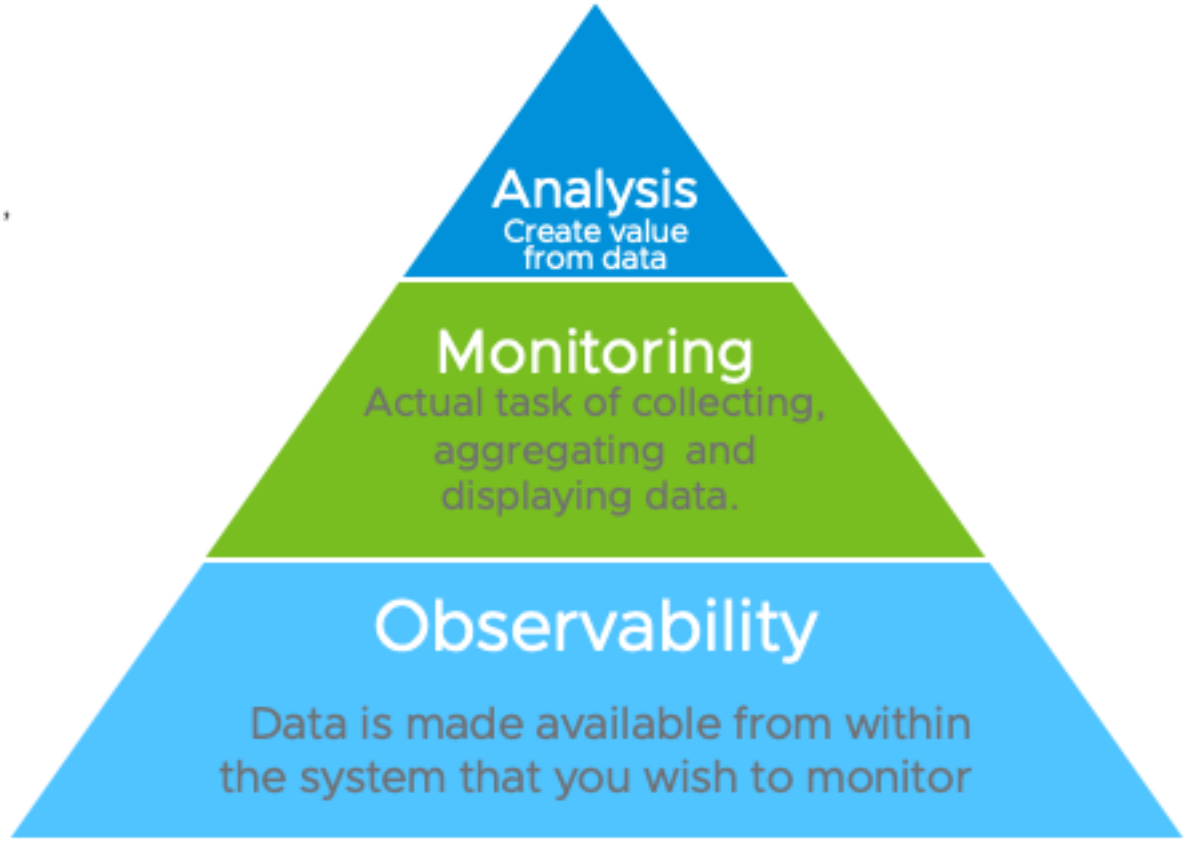

Figure 2. Healthy operations criterions

- **Observability**: It requires monitoring related data made available from the applications and other components. If observability is not built-in, data should be gathered by traditional ways.
- **Monitoring**: This collects, aggregates and displays observability data.
- **Analysis:** It defines the automatic or manual analysis of collected monitoring data to instrument and take actions

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 following.
  - Know when things go wrong
  - Problem Detection, Alerting and notifications
  - Be able to debug and gain insight
  - Detect changes over time and drive technical/ business decisions
  - 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, it's different component/layers and different players in those areas.

3.2.1 Containerized env. components

Following picture depicts the different layers of such an environment and major players in those areas.

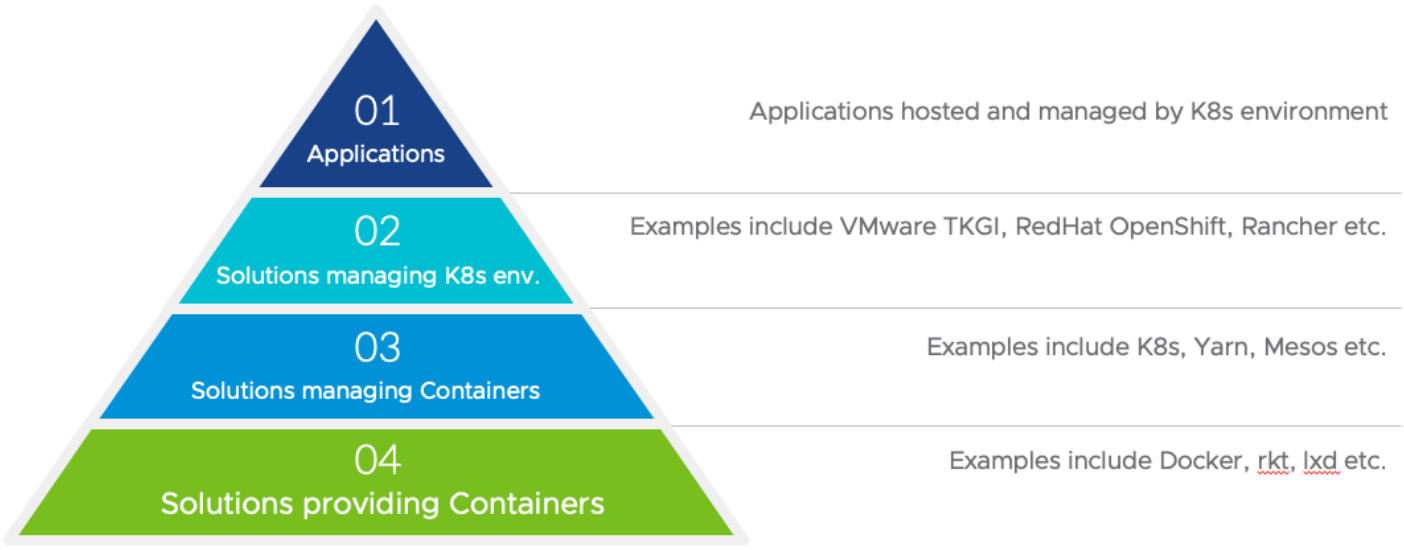

Figure 3. Container environment components

- **Layer 4**: It is 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 provides 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 and run and manage thousands or 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, managing these managers is also not easy. These are complex solutions requires 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 TKGI, RedHat OpenShift, Rancher etc.
- **Layer 1**: At the top of the pyramid we have applications. Remember application drives everything. So, we host and manage many applications on top of the other layers.

When the container technology started to take shape, there were many players offering solutions in that area. With the passing of time Kubernetes became the undisputed leader in container manager space with Docker winning the container technology race. So, we will limit our discussion to Kubernetes (K8s) environments only.

3.2.2 End to end monitoring of K8s env.

Now let's look at the required solutions to host 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.

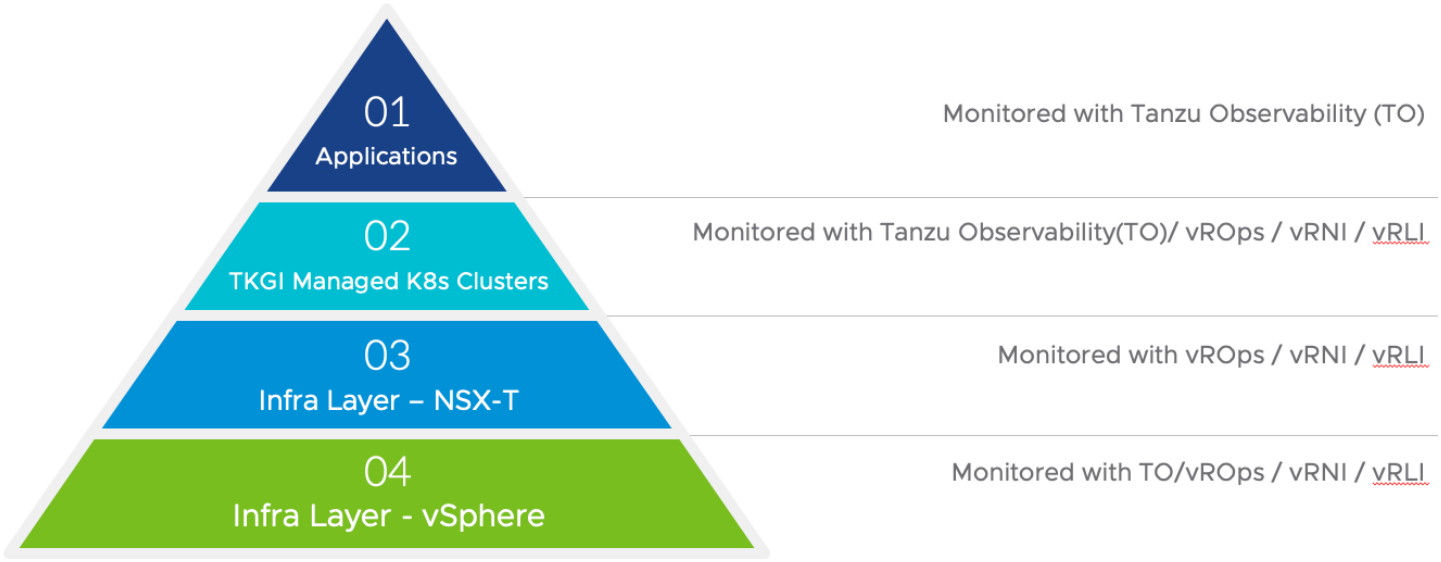

Figure 4. Layers of a K8s hosting environment

Above picture depicts the different layers involved in a K8s hosting environment and the solutions of available/used for monitoring them.

** Please note, these products have overlapping capabilities and depending on persona or teams, can be used accordingly. For example, while Tanzu Observability monitors end to end stack, for virtualization and cloud admin perspective, using vROps to monitor infra layer in context of the solution described in this whitepaper is more appropriate as it adds more values. However, even within any solution, an SRE may find additional value by using Wavefront.

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

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

In the below sections we will discuss in detail about end to end monitoring a TKGI environment.

Part B

4 Monitoring a TKGI environment

Before we delve into the details of monitoring a TKGI environment, it is imperative to understand the different aspects of a TKGI 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 TKGI environment.

4.1 Who monitors what?

Provided below is a list showing the different layers in a TKGI environment and who monitors which layers. Also, it provides a sample of the monitoring areas these roles may monitor.

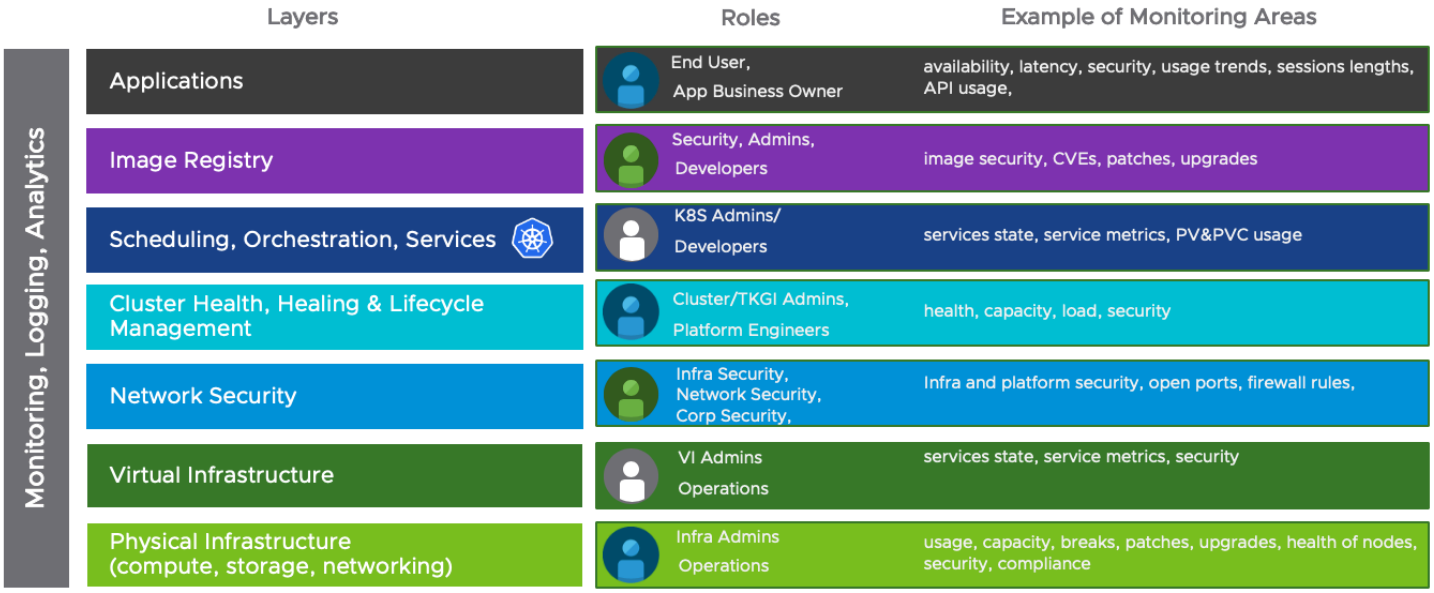

Figure 5. Different TKGI environment layers and the monitoring responsibility of the different teams

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

4.2 Example Metrics

The following picture contains a list of metrics to monitor a different layer of a TKGI environment. This list is an example list and does not include all the possible metrics to watch in the environment.

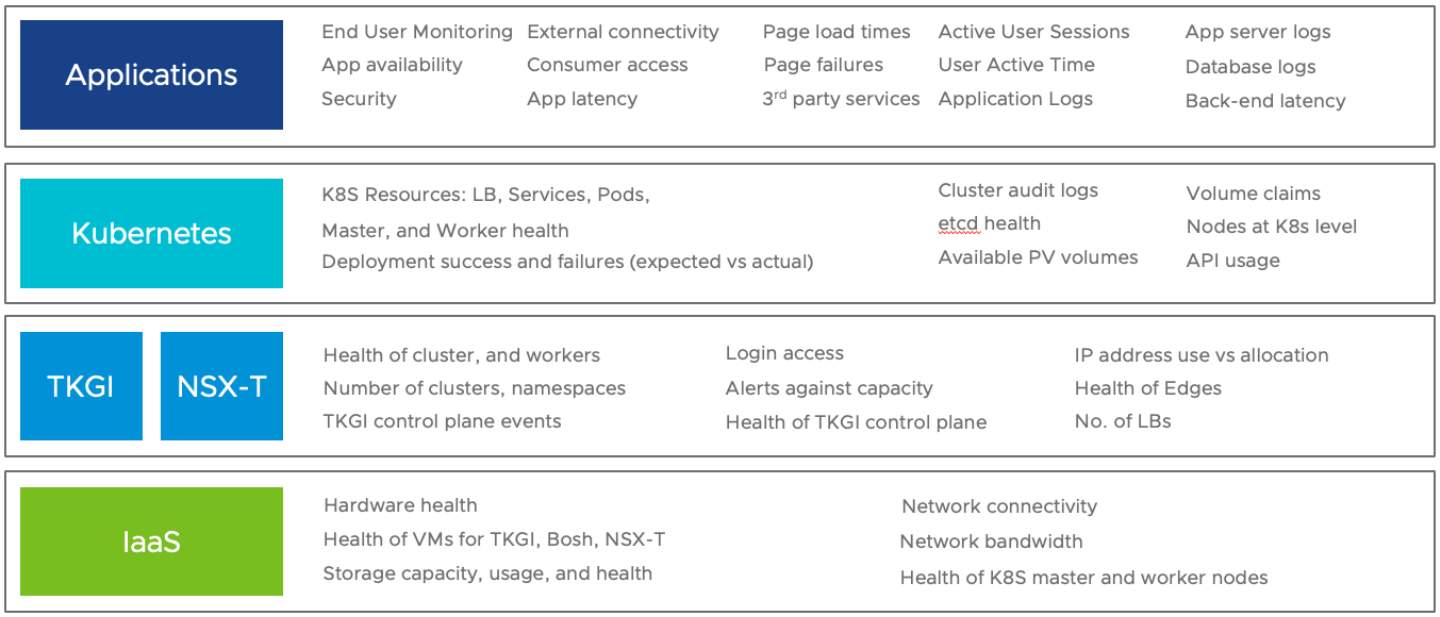

Figure 6: A sample list of metrics to monitor in the different TKGI environment layers

4.3 Different tools for different layers

The below picture provides a pictorial view of different TKGI layers and the choice of VMware tools used to monitor those layers.

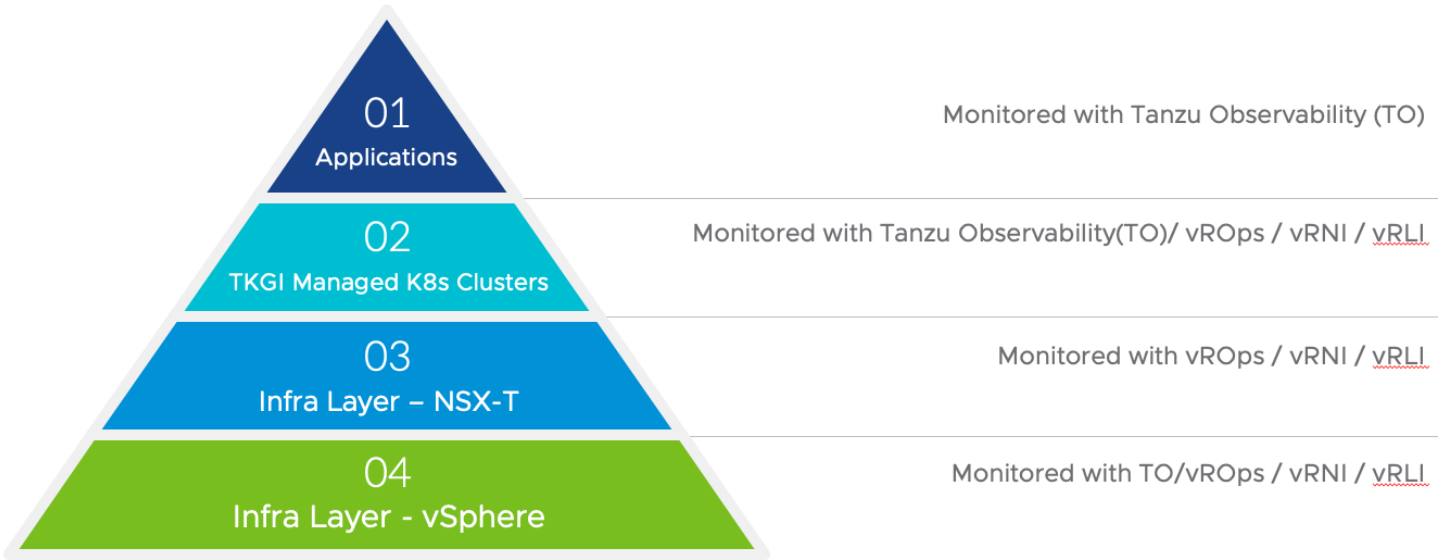

Figure 7: Different TKGI environment layers and the VMware tools to monitor those

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 the performance-related issues are 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 responsible for them, VMware tools used to monitor those, and the general flow of troubleshooting issues in TKGI overall environment.

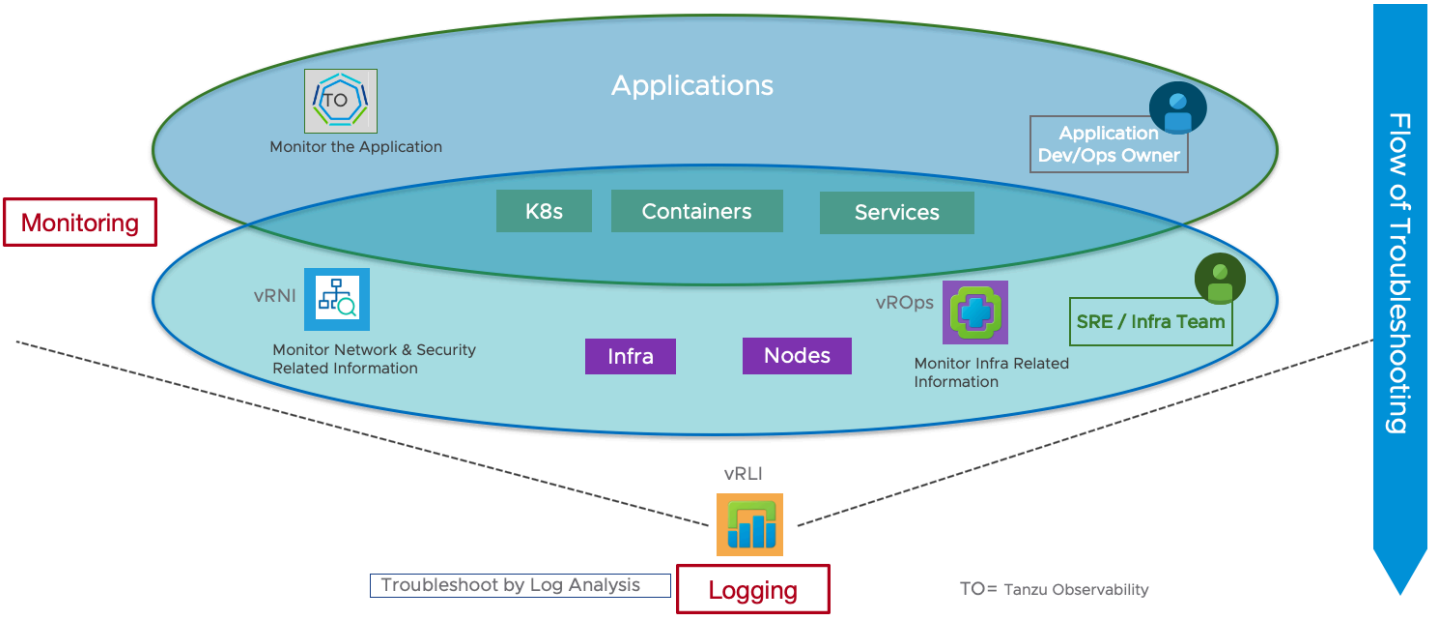

Figure 8: General flow of troubleshooting issues in TKGI overall environment

4.5 Currently available options

To showcase and explore existing options please follow the path, products and information provided in the below picture.

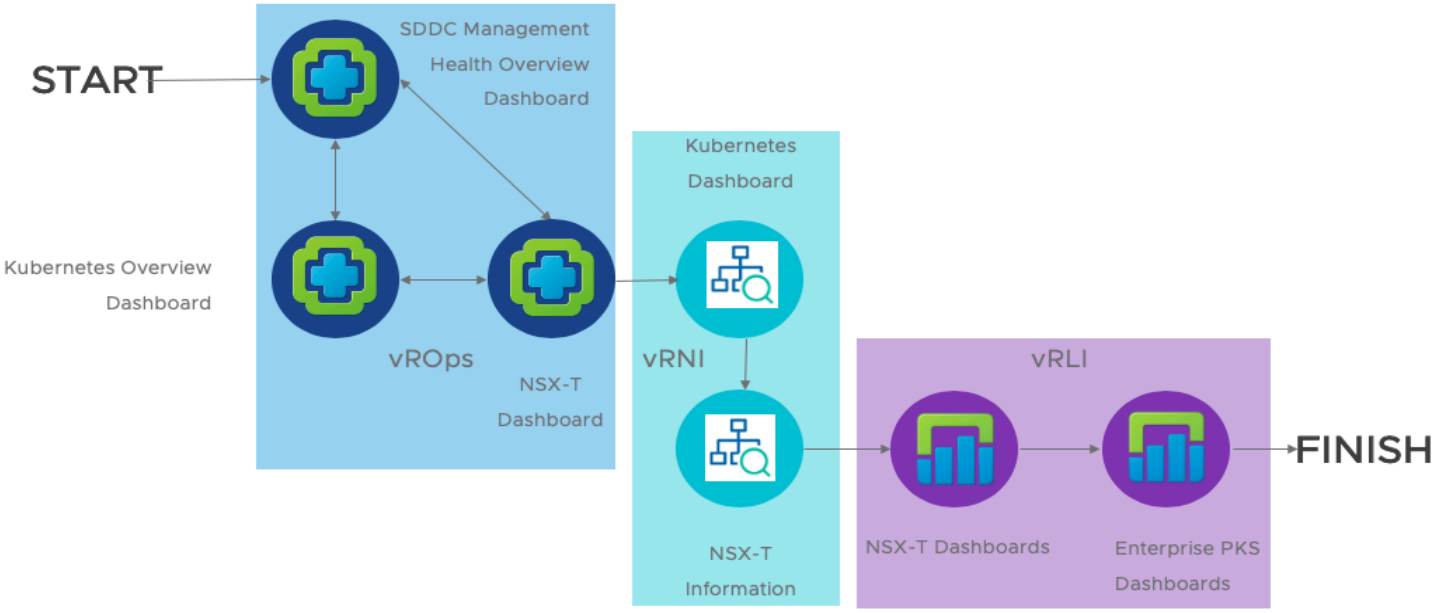

Figure 9: Existing options in different VMware products to monitor a TKGI environment

4.6 Missing Layer from Monitoring

In Pic 3, we have defined the layers present in the TKGI environment and the current monitoring options for them. But if we look closely, we can see there is a layer missing from default monitoring options. That layer is the TKGI Control Plane layer. Currently, there is no default or out of the box solution available to monitor specifically the TKGI control plane/management plane components. Let's focus on that layer now.

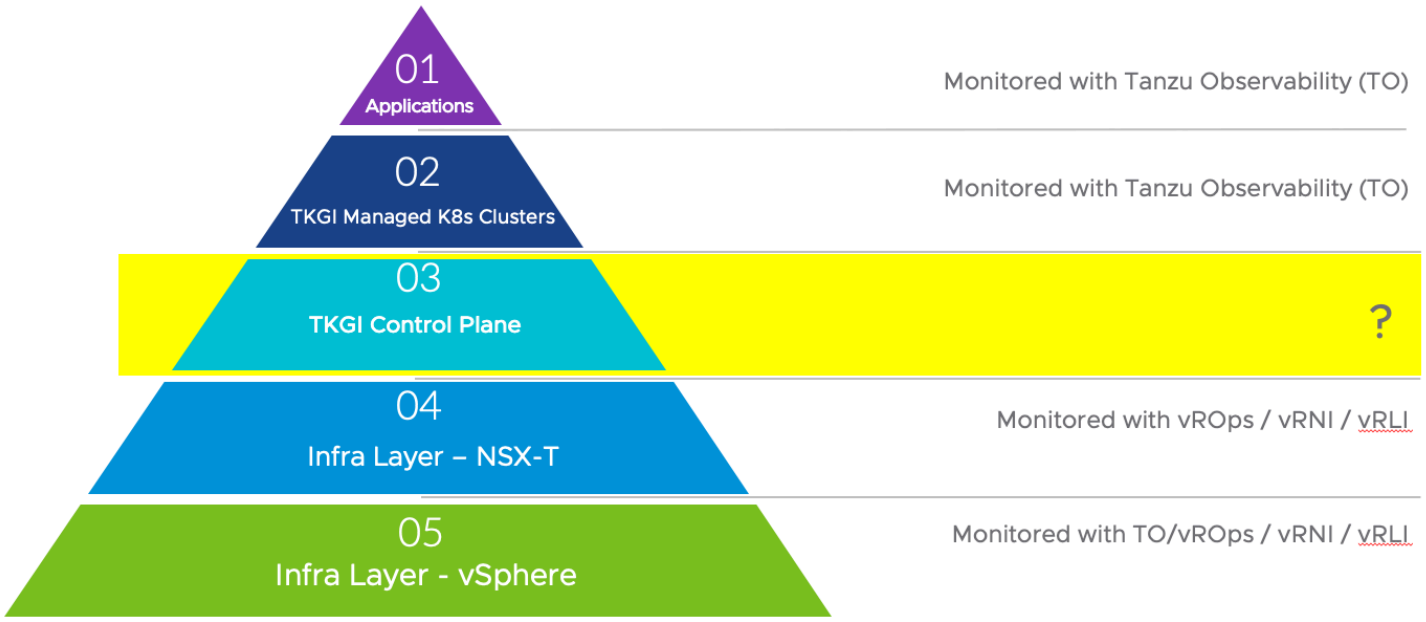

Figure 10: Missing layer in TKGI monitoring

In this document, we describe the missing layer and how to monitor it. We provide details on how to customize the different existing VMware products to monitor that layer in detail.

5 TKGI Control Plane

Let's discuss TKGI control plane components in detail and the different failure areas in those components. Provided below is a picture showing the different components and the primary services that we need to monitor.

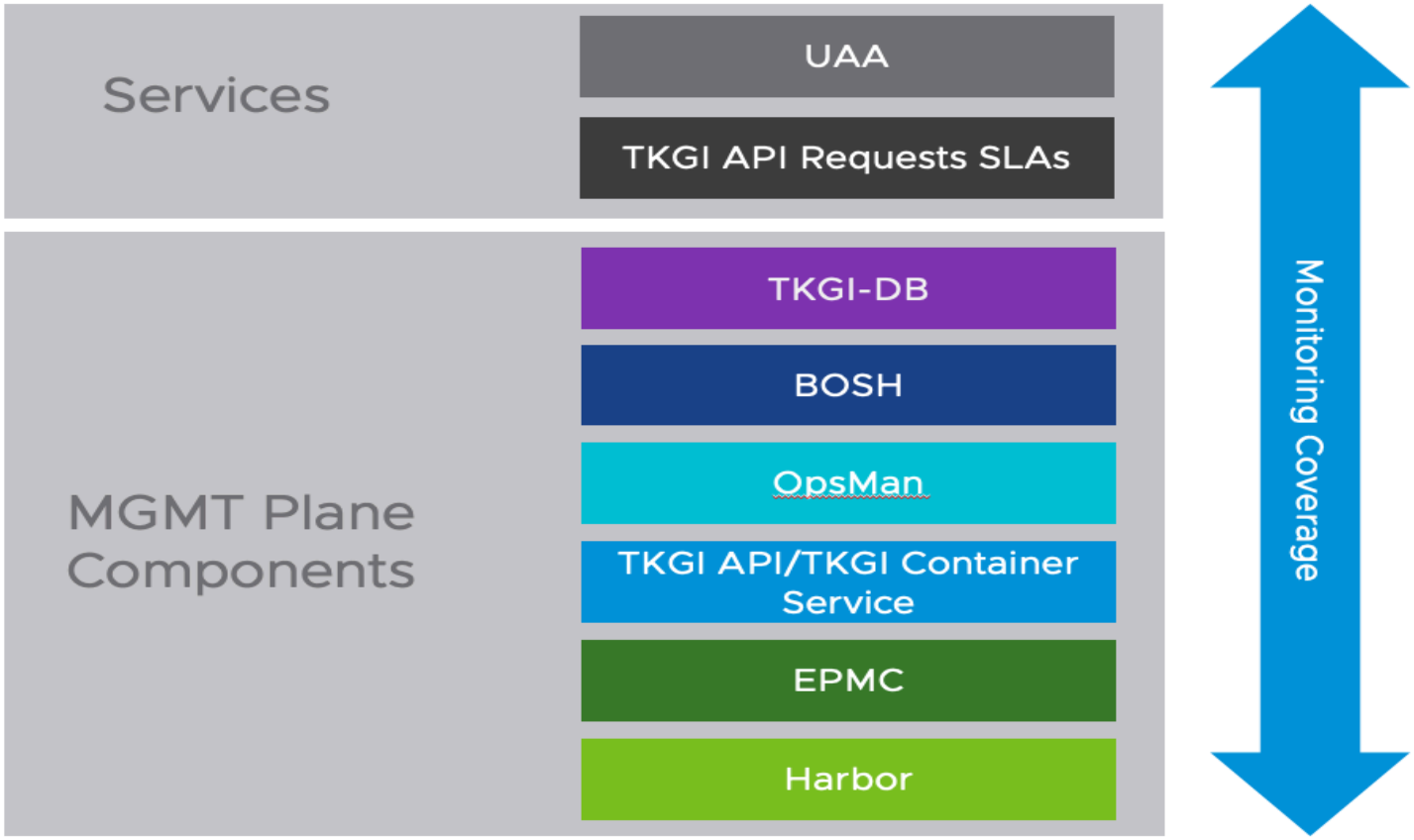

Figure 11: TKGI Control Plane components and major services

5.1 Metrics to Monitor

Attached is an excel document with a list of the major metrics to monitor for these components. You can download and check the file from *https://github.com/sajaldebnath/tkgi-monitoring/blob/master/PKS%20Environment%20Monitoring.xlsx*.

In the below sections, we will discuss custom options and how to configure those in the following three VMware products.

- vRealize Operations Manager (vROps)
- vRealize Network Insight (vRNI)
- vRealize Log Insight (vRLI)

6 Assumptions

The following assumptions are made in the guide:

- A TKGI environment is deployed and ready for monitoring.
- This guide will use the following VMware tools for monitoring.
    - VMware vRealize Operations Manager
    - VMware vRealize Log Insight
    - VMware vRealize Network Insight
- It is assumed the above-mentioned VMware products are deployed and integrated together.

7 Environment

This document was prepared based on the following products and versions.

Table 1: Used products, components and versions

Product	Component	Version
vRealize Operations Manager	Core Product	8.1
	VMware vRealize Operations Management Pack for Container Monitoring	1.4.3.15987816
	SDDC Management Health	8.1.15995854
	NSX-T Management Pack (Default out of the box)	8.1.15972155
vRealize Network Insight	Core Product	5.2.0.1585846638
vRealize Log Insight	Core Product	8.1.0-15994158
	VMware - Enterprise PKS (Community Supported)	1.0
	VMware - NSX-T	3.8.2

8 Result – How it would look

Before we start discussing about various configuration steps, let's have a look at what we want to achieve. Provided below is a picture showing the flow of the custom solutions in different products.

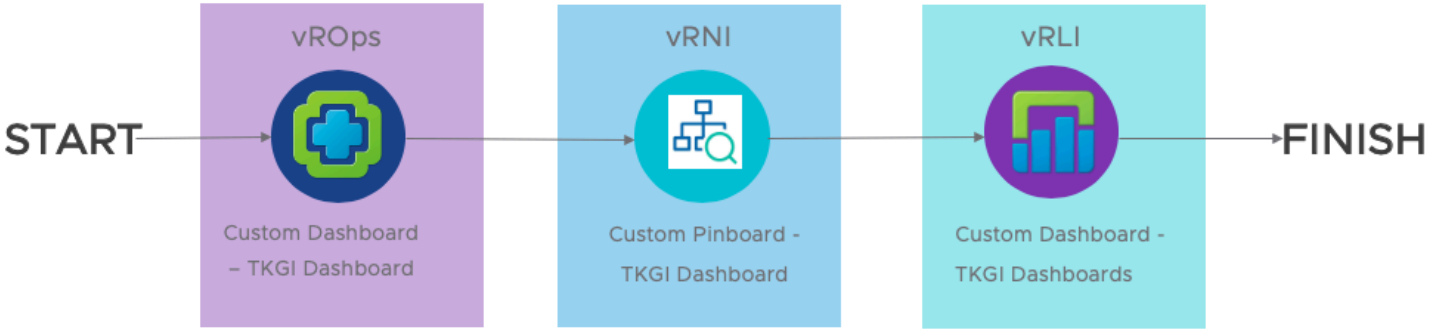

Figure 12: Flow of different custom dashboard in various products

8.1 vRealize Operations Manager

Let's start with a Customer Dashboard in vRealize Operations Manager (vROps).

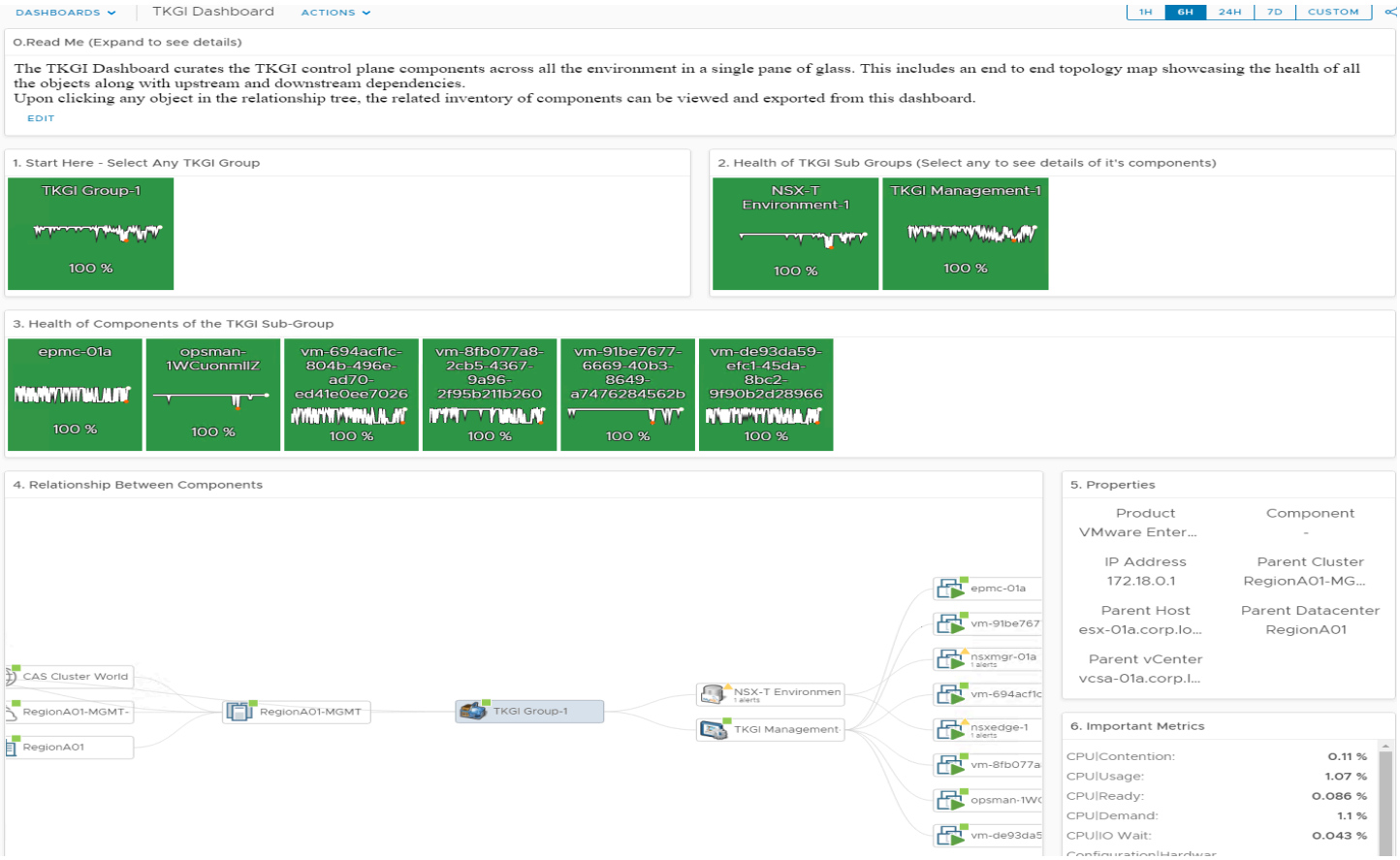

Figure 13: TKGI Dashboard – Custom Dashboard in vROps

8.2 vRealize Network Insight

Next, we have a custom dashboard configured in vRealize Network Insight.

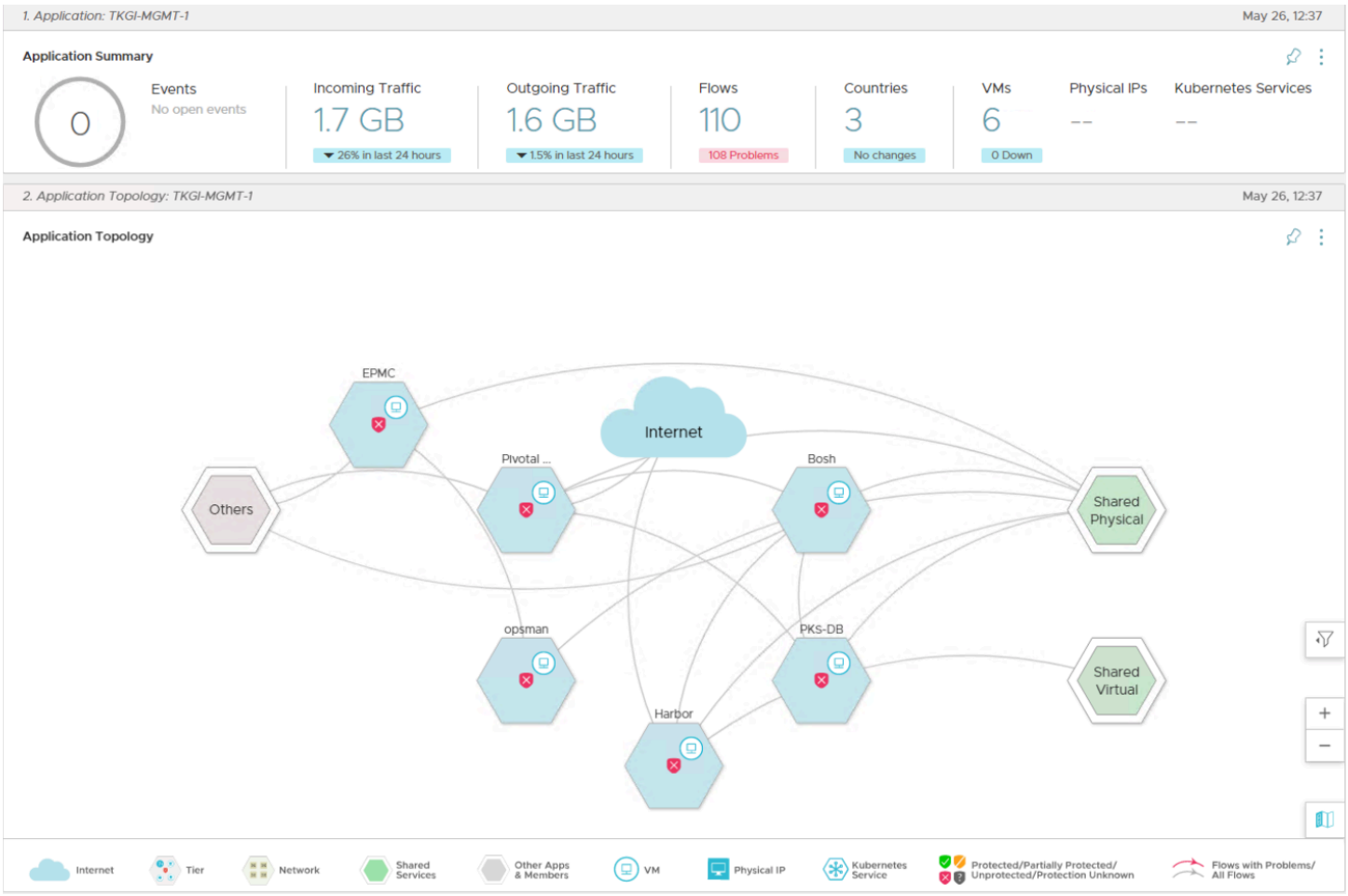

Figure 14: TKGI-Group-1 Dashboard – Custom Dashboard in vRNI – pic 1

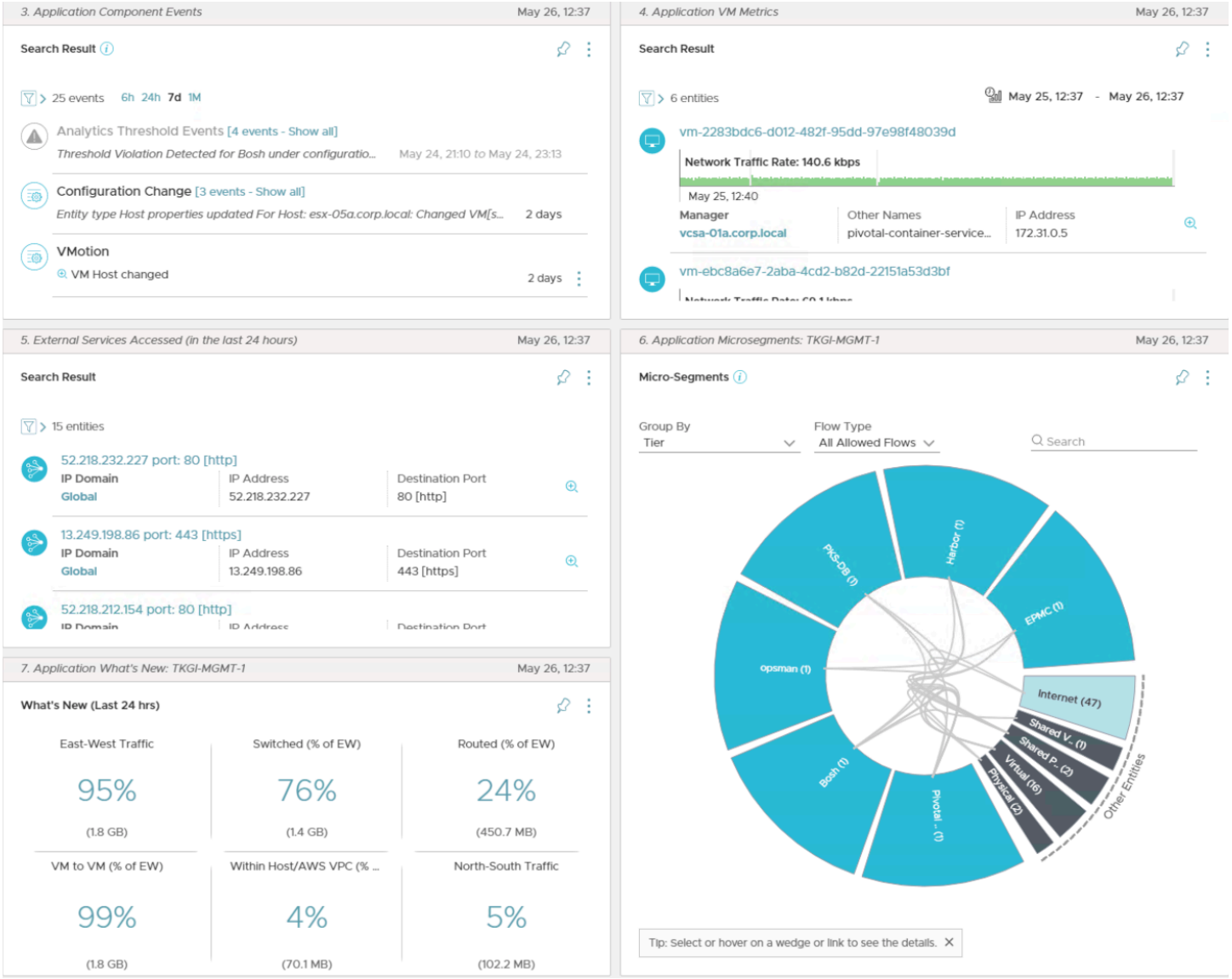

Figure 15: TKGI-Group-1 Dashboard – Custom Dashboard in vRNI – pic 2

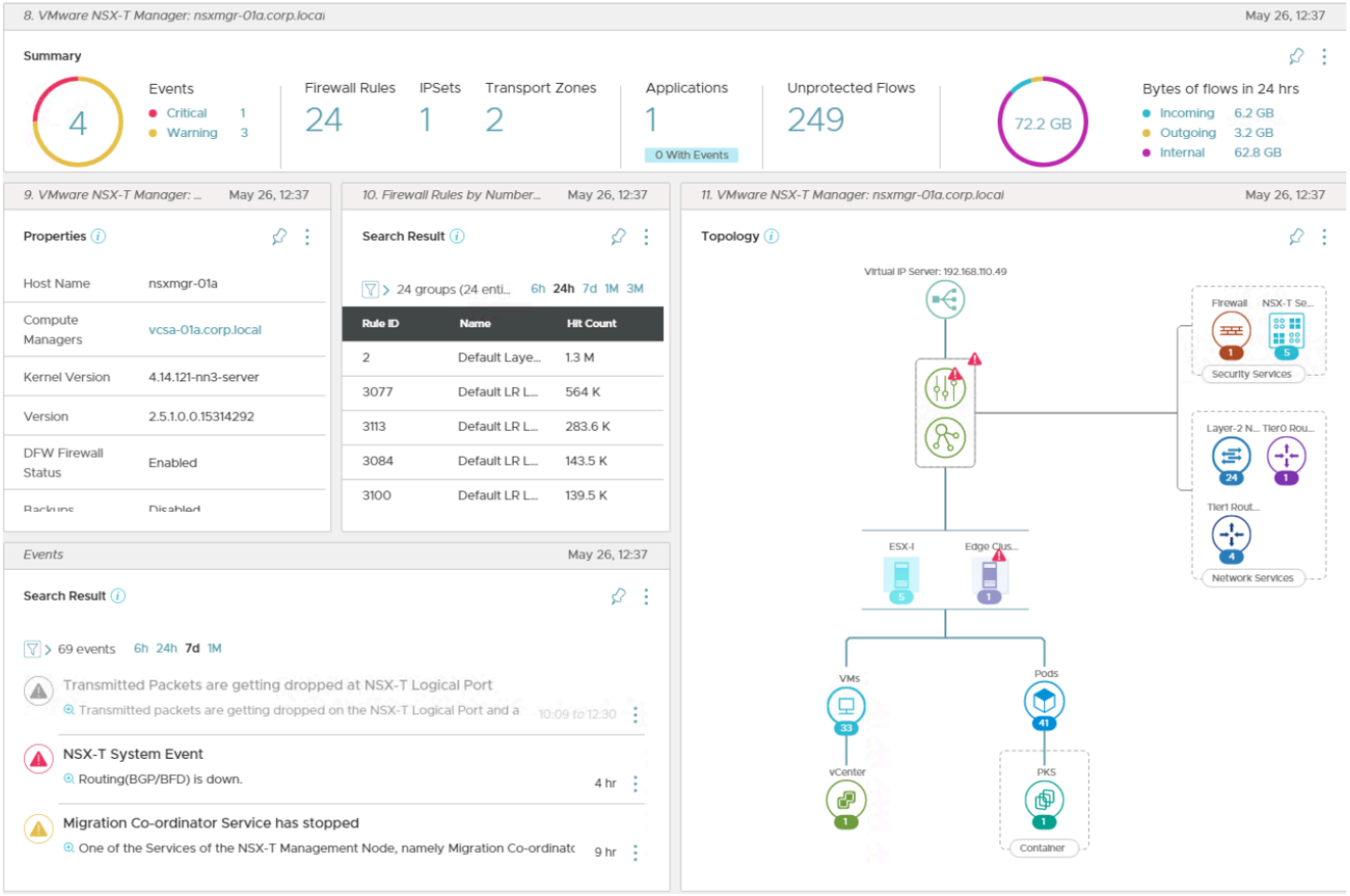

Figure 16: TKGI-Group-1 Dashboard – Custom Dashboard in vRNI – pic 3

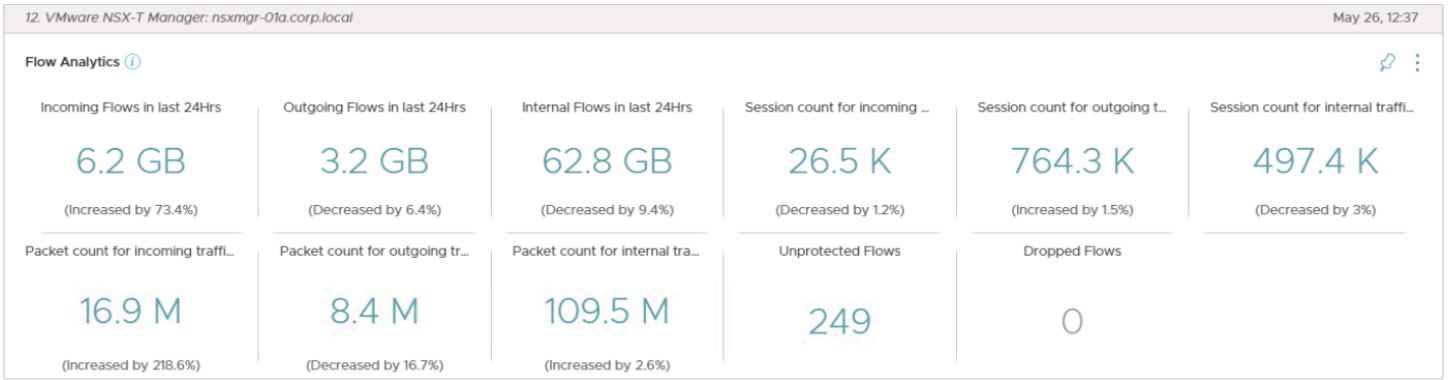

Figure 17: TKGI-Group-1 Dashboard – Custom Dashboard in vRNI – pic 4

8.3 vRealize Log Insight

Next, we have a custom dashboard configured in vRealize Log Insight.

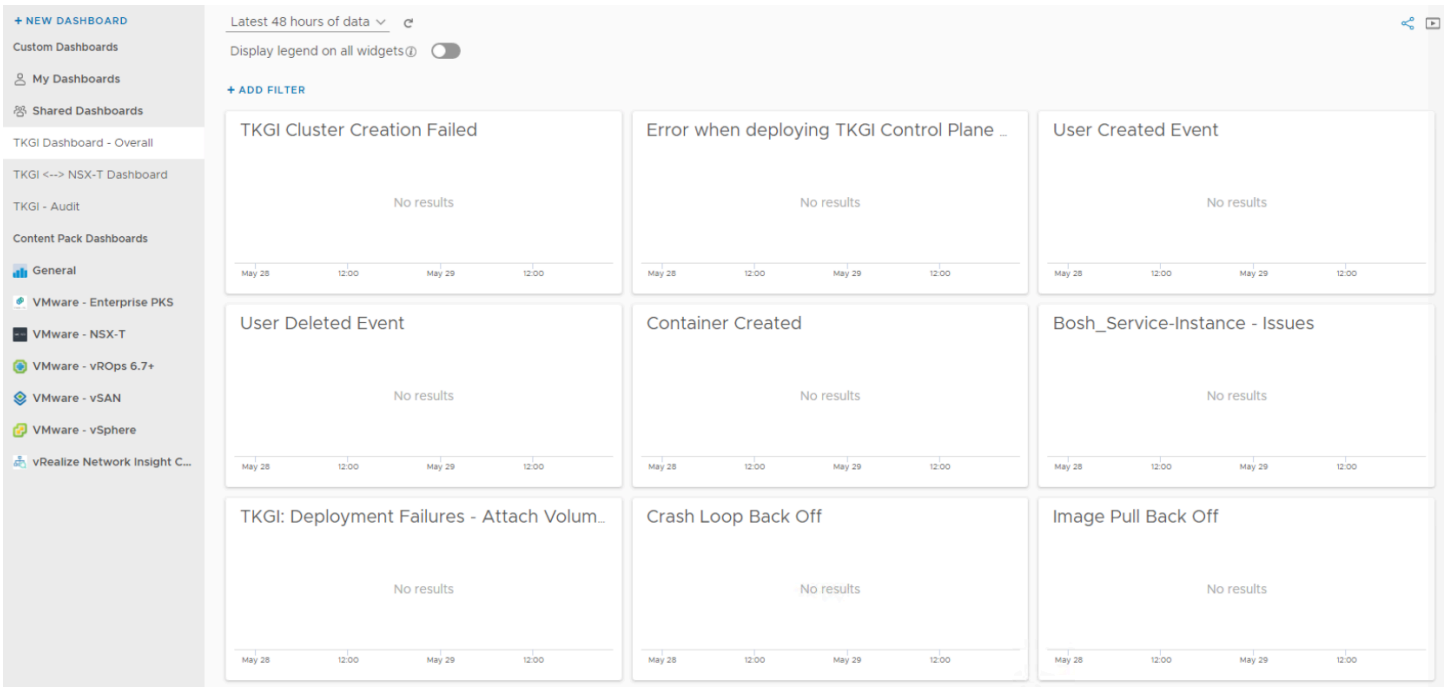

Figure 18: TKGI Dashboard – Overall – Custom Dashboard in vRLI – pic 1

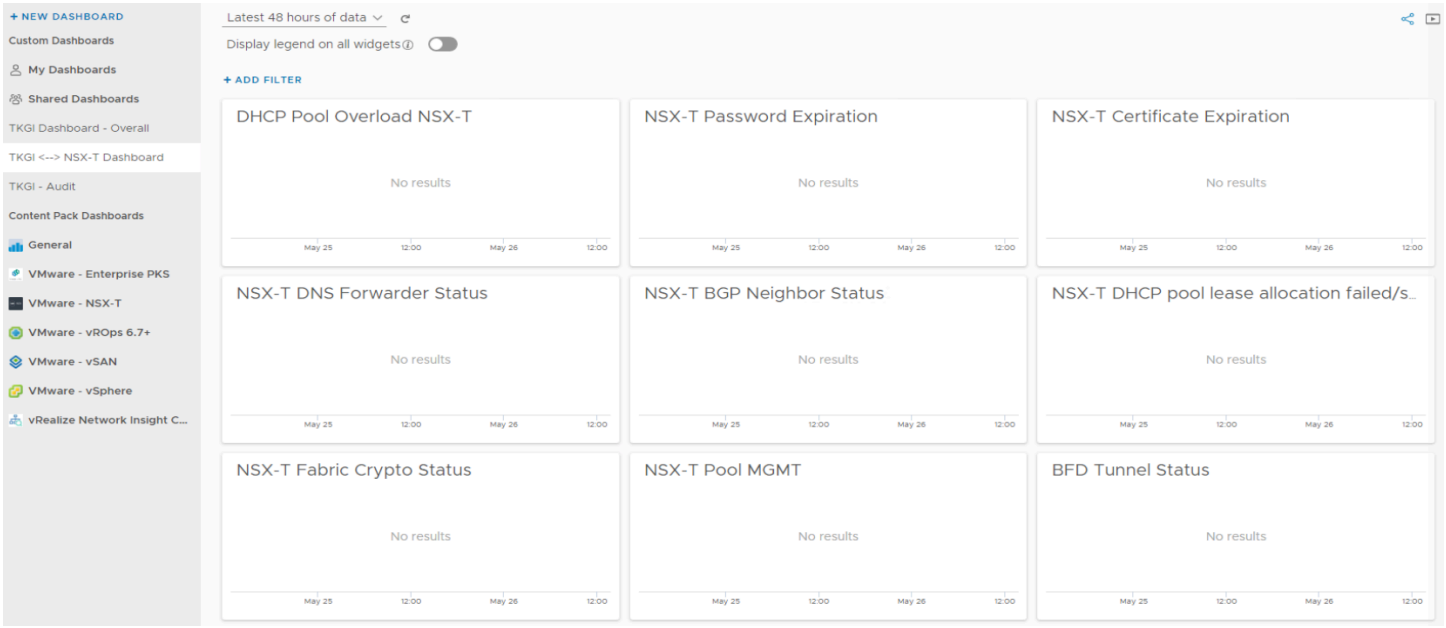

Figure 19: TKGI ←→NSX-T Dashboard – Custom Dashboard in vRLI – pic 2

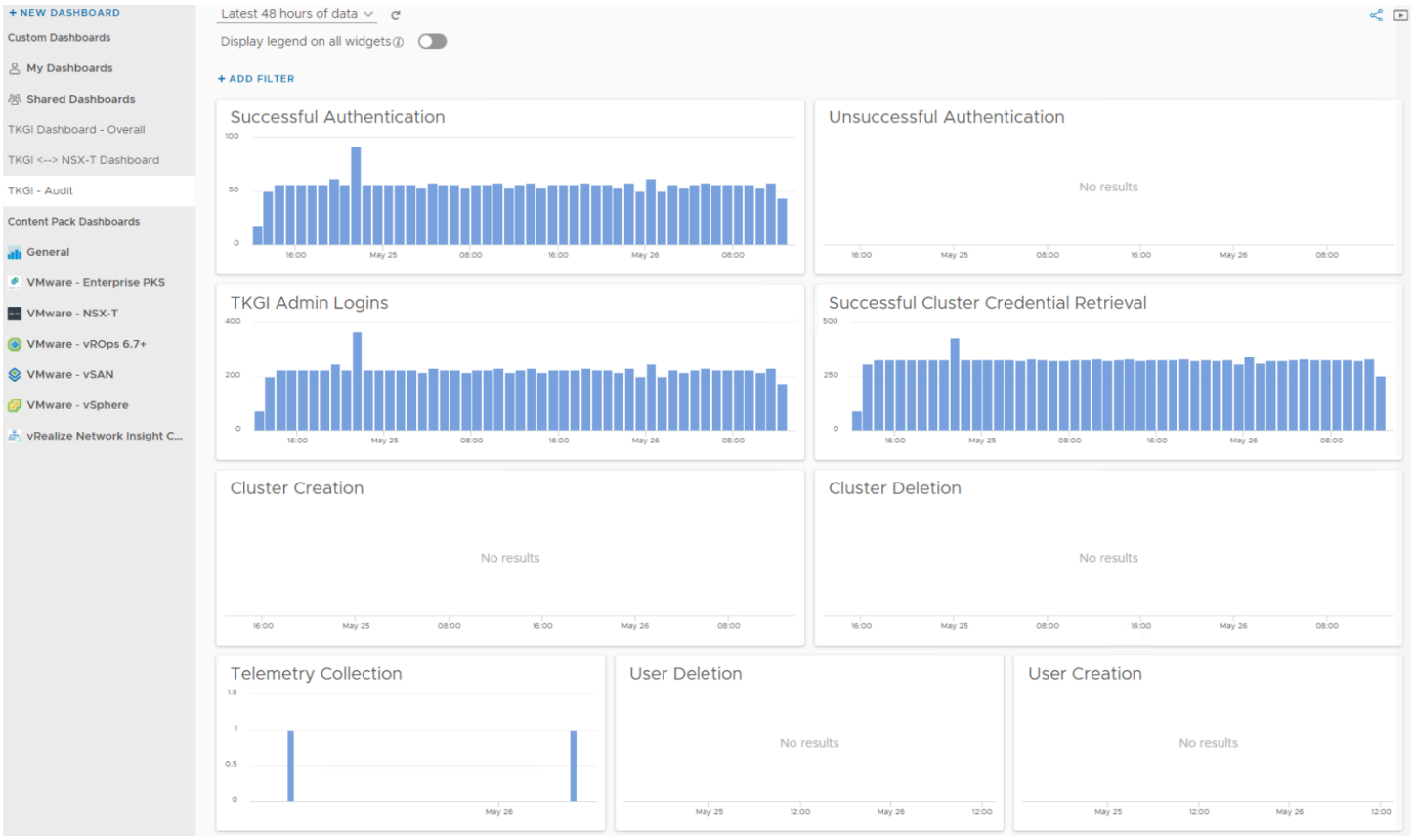

Figure 20: TKGI - Audit Dashboard Custom Dashboard in vRLI – pic 3

9 Custom configuration details

Here we will discuss the various configuration details required in the VMware products to get the desired result.

9.1 vRealize Operations Manager

Let's start the discussion with configuring a custom dashboard to monitor the TKGI Control plane components in vROps. Before we start, provided below is a picture depicting different TKGI layers and their relationships in the dashboard.

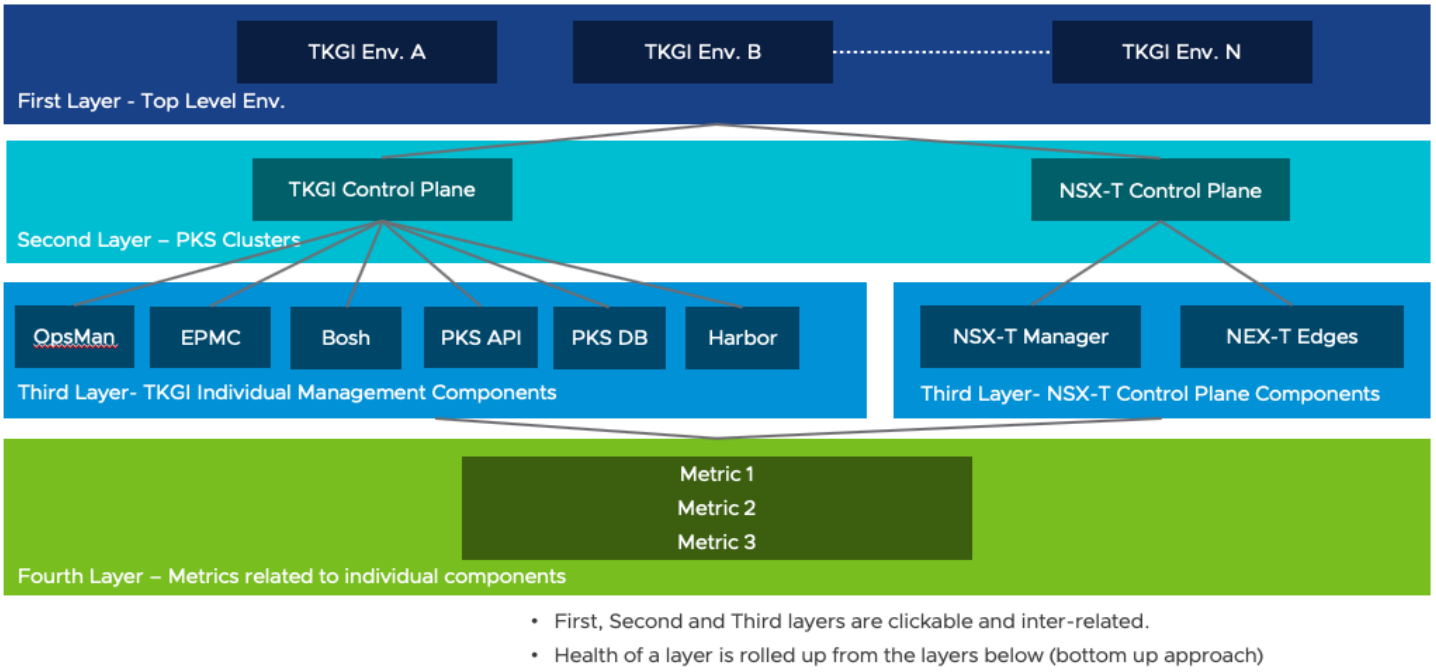

Figure 21: Custom dashboard components and relationships

Explanation of the dashboard components:

- The top layer shows as list of different TKGI environment (if the environment has more than one TKGI clusters, it will be listed here)
- Once clicked, it will show list in second layer. The underlying NSX-T control plane and the TKGI control plane
- Clicking on individual items in the second layer will show the list of components under that group in third layer
- Clicking on the items in third layer will show the individual component metrics

Above is a relation of the components and groups. Apart from that, there will be events view and a relationship view as well.

How to Build:

To build this view we need to follow and configure the following components:

- Group Types
- Custom Groups
- Object Relationships
- Custom Dashboard

The purpose is to build a custom dashboard that shows all the TKGI Control Plane components. By looking at this dashboard, we should be able to tell at a glance whether the environment is healthy or not.

9.1.1 Group Types

We start the configuration with custom group types. We need to create three custom group type. Create three custom group type by going to Administration → Configuration → Group Types → ADD. Name and purpose for the group types are provided below.

Table 2: Group Type details

Name of the group type	Purpose
NSX-T MGMT	Will be used to group together all the NSX-T environments
TKGI-MGMT	Will be used to group together all the control plane objects of TKGI Cluster
TKGI Group	Will be used to group together the "NSX-T MGMT" and TKGI-MGMT type group objects

9.1.2 Custom Groups

Next, we will configure custom groups. Go to Environment → Environment Overview → Custom Groups → ADD to add custom groups. We will configure three custom groups.

Table 3: Custom Group Type details

Name of the custom group	Purpose
TKGI Group-1	Used to represent Layer 1 objects depicted in Pic 17. Groups together TKGI MGMT and NSX-T environment components. Shows overall health of an environment.
TKGI Management-1	Used to represent Layer 2 objects depicted in Pic 17. Groups together TKGI MGMT components. Shows overall health of TKGI control plane components.
NSX-T Environment-1	Used to represent Layer 2 objects depicted in Pic 17. Groups together NSX-T control plane components. Shows overall health of NSX-T control plane objects.

Note, since I have only one TKGI environment, I have 3 groups. If you have more than one environment, then, you need to create that many groups. For example, if you have 3 different TKGI environment, then you need to create 3 x 3 = 9 total custom groups. The purpose of base 3 groups are same.

Provided below are the details of the individual groups.

TKGI Management-1

Purpose of this custom group is to group all the TKGI control plane elements. This is a second layer entity. Details of the group is provided below.

Name: TKGI Management-1

Group Type: TKGI-MGMT

Define Membership Criteria:

- Ops Manager
  - Object Type: Virtual Machine
  - Properties→(Summary|Configuration|Product Name) → contains → Ops Manager

  AND
  - Relationship→ Descendant of → is → Cluster Name (e.g. RegionA01-MGMT)
- Bosh
  - Object Type: Virtual Machine
  - Properties→(Summary|Custom Tag:instance_group|Value) → contains → bosh

  AND
  - Relationship→ Descendant of → is → Cluster Name (e.g. RegionA01-MGMT)
- Harbor App
  - Object Type: Virtual Machine
  - Properties→(Summary|Custom Tag:instance_group|Value) → contains → harbor-app

  AND
  - Relationship→ Descendant of → is → Cluster Name (e.g. RegionA01-MGMT)
- PKS DB
  - Object Type: Virtual Machine
  - Properties→(Summary| Custom Tag:instance_group|Value) → contains → pks-db

  AND
  - Relationship→ Descendant of → is → Cluster Name (e.g. RegionA01-MGMT)
- Pivotal-container-service (PKS API)
  - Object Type: Virtual Machine
  - Properties→(Summary| Custom Tag:instance_group|Value) → contains → pivotal-container-service

  AND
  - Relationship→ Descendant of → is → Cluster Name (e.g. RegionA01-MGMT)
- Enterprise PKS (EPMC)
  - Object Type: Virtual Machine
  - Properties→(Summary|Configuration|Product Name) → contains → Enterprise PKS

  AND
  - Relationship→ Descendant of → is → Cluster Name (e.g. RegionA01-MGMT)

Please note, in my case, the distinction between the environments is the cluster name. In a cluster I will host only a single TKGI environment. That is the reason I got Cluster as a defining criterion. In your case if something else is defining criteria, please select that

instead of cluster. For example, if you have multiple datacenters or vCenter servers and decide to create a TKGI environment per Datacenter or vCenter Server, then please select Datacenter or vCenter Server as the deciding factor.

Anyways, for multiple TKGI environment, create multiple groups like “TKGI Management-1”, “TKGI Management-2”, “TKGI Management-n” etc. with each group containing the control plane components pertaining to that group only (based on the selection criteria).

A sample screenshot is provided below.

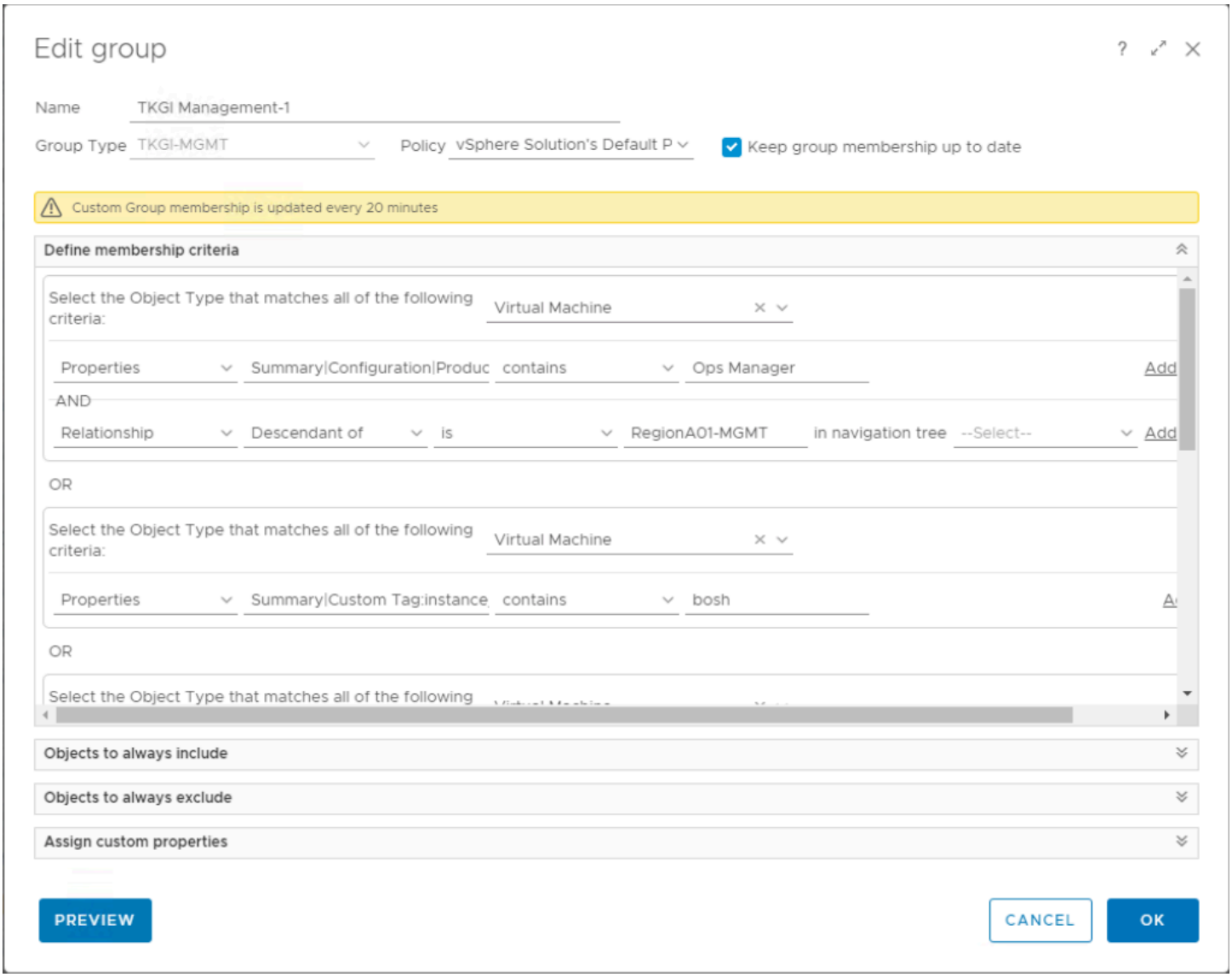

Figure 22: Details of custom group TKGI Management-1

NSX-T Environment-1

Purpose of this custom group is to group all the NSX-T control plane elements. This is a second layer entity. Details of the group is provided below.

Name: NSX-T Environment-1

Group Type: NSX-T MGMT

Define Membership Criteria:

- NSX-T Managers
    - Object Type: Virtual Machine
    - Properties→(Summary|Configuration|Product Name) → contains → nsx-unified-appliance

  AND
    - Relationship→ Descendant of → is → Cluster Name (e.g. RegionA01-MGMT)
- NSX Edges
    - Object Type: Virtual Machine
    - Properties→(Summary|Configuration|Product Name) → contains → nsx-edge

  AND
    - Relationship→ Descendant of → is → Cluster Name (e.g. RegionA01-MGMT)

Please note, in my case, the distinction between the environments is the cluster name. In a cluster, I will host only a single NSX-T environment. That is the reason, I got Cluster as a defining criterion. In your case, if something else is defining criteria, please select that instead of cluster. For example, if you have multiple datacenters or vCenter servers and decide to create an NSX-T environment per Datacenter or vCenter Server, then please select Datacenter or vCenter Server as the deciding factor.

Anyways, for multiple NSX-T environment, create multiple groups like “NSX-T Environment-1”, “NSX-T Environment-2”, “NSX-T Environment-n” etc. - with each group containing the control plane components pertaining to that group only (based on the selection criteria).

A sample screenshot is provided below:

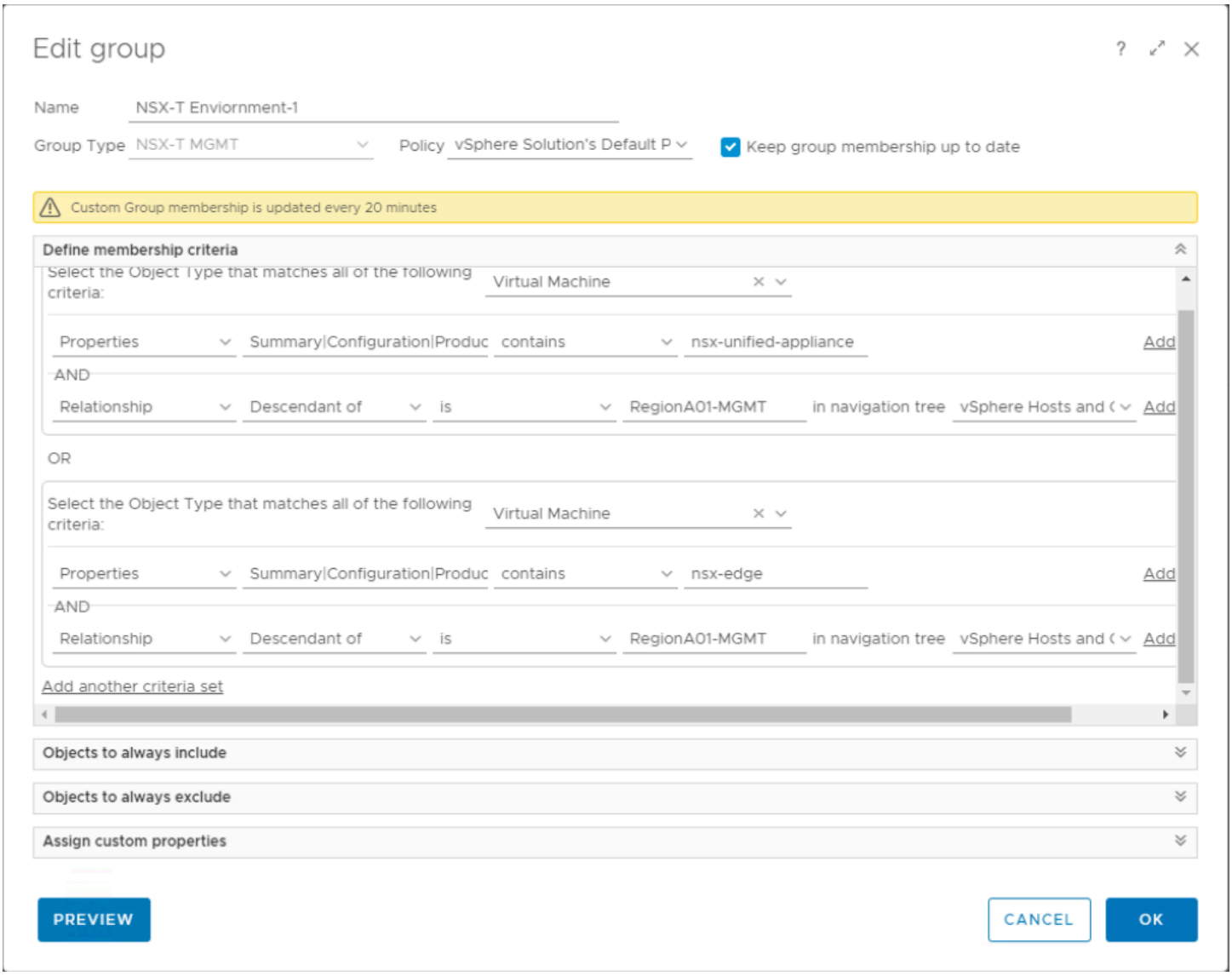

Figure 23: Details of custom group NSX-T Environment-1

TKGI Group-1

Purpose of this custom item is to group all the elements of the TKGI environment. Details for the custom group is provided below:

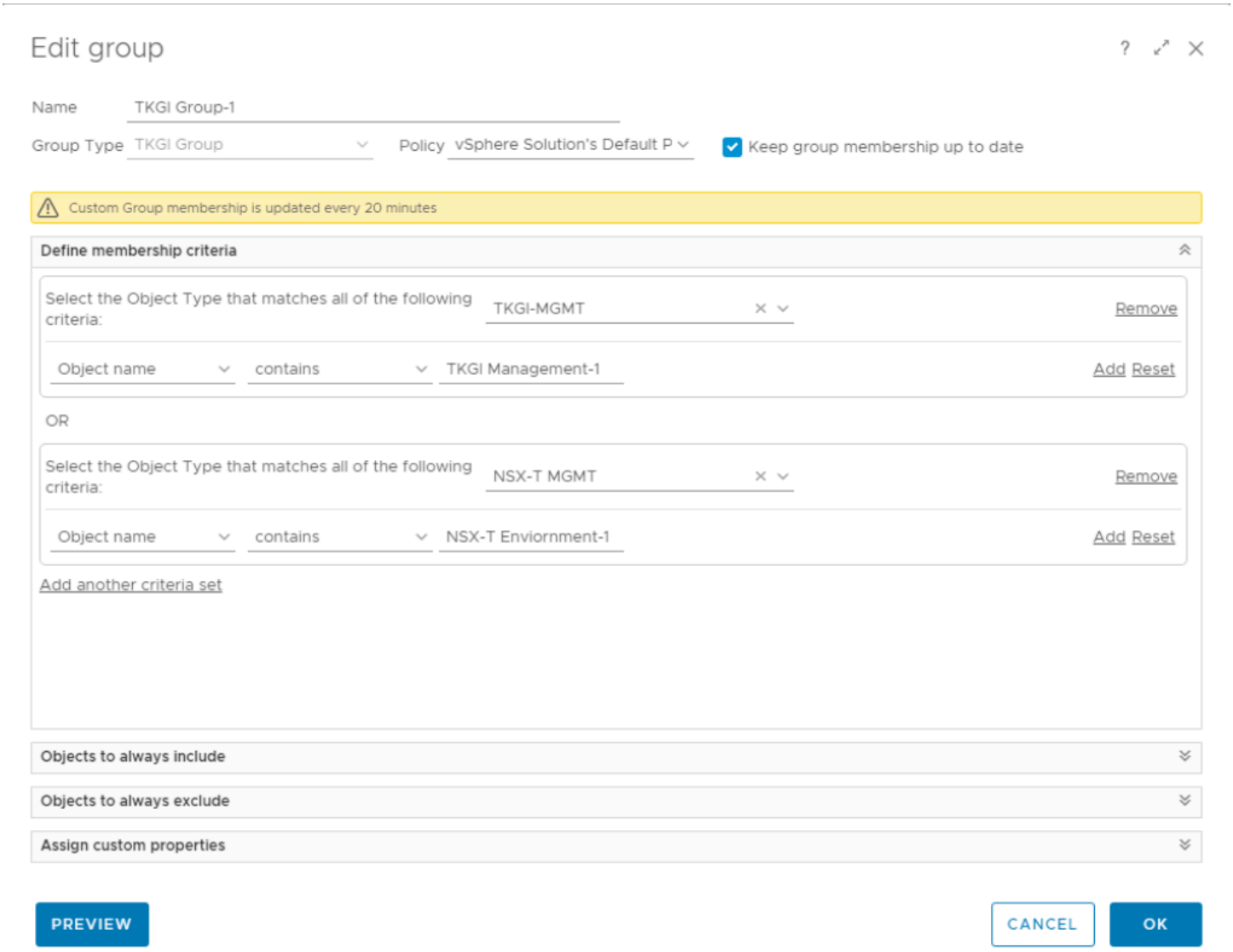

Figure 24: Details of custom group TKGI Group-1

Please note, the Group Type is "TKGI Group". Also, under the membership criteria we selected "TKGI Management-1" and "NSX-T Environment-1". For multiple TKGI environments, create multiple custom groups and name them accordingly so that They can be

"TKGI Group-2" →("TKGI Management-2" + "NSX-T Environment-2")

"TKGI Group-n" →("TKGI Management-n" + "NSX-T Environment-n")

and so on.

9.1.3 Object Relationship

Next, we will define a customer object relationship between the defined custom groups and the environment.

Go to **Administration → Configuration → Object Relationship** and make sure the following relationships exist:

Table 4: Object Relationship details

Parent Selection	Children
TKGI Group-1 (under TKGI Group)	• NSX-T Environment-1 • TKGI Management-1
TKGI Management-1 (under TKGI-MGMT)	Shows all the 6 components of TKGI control plane (epmc, opsman, bosh, harbor, pks-db, pks api)
NSX-T Environment-1 (under NSX-T MGMT)	Shows all the management component of NSX-T environment (nsx manager and nsx edge)
RegionA01-MGMT (cluster under Cluster Compute Resource)	Should contain TKGI Group-1 apart from other children

Please note, the above relationship is for one group only. If you have more than one group (environment) then check all of them accordingly. The major relationship that you need to configure is add "TKGI Group-1" under the Cluster (RegionA01-MGMT in my case) by dragging it from right side list to the top of the box.

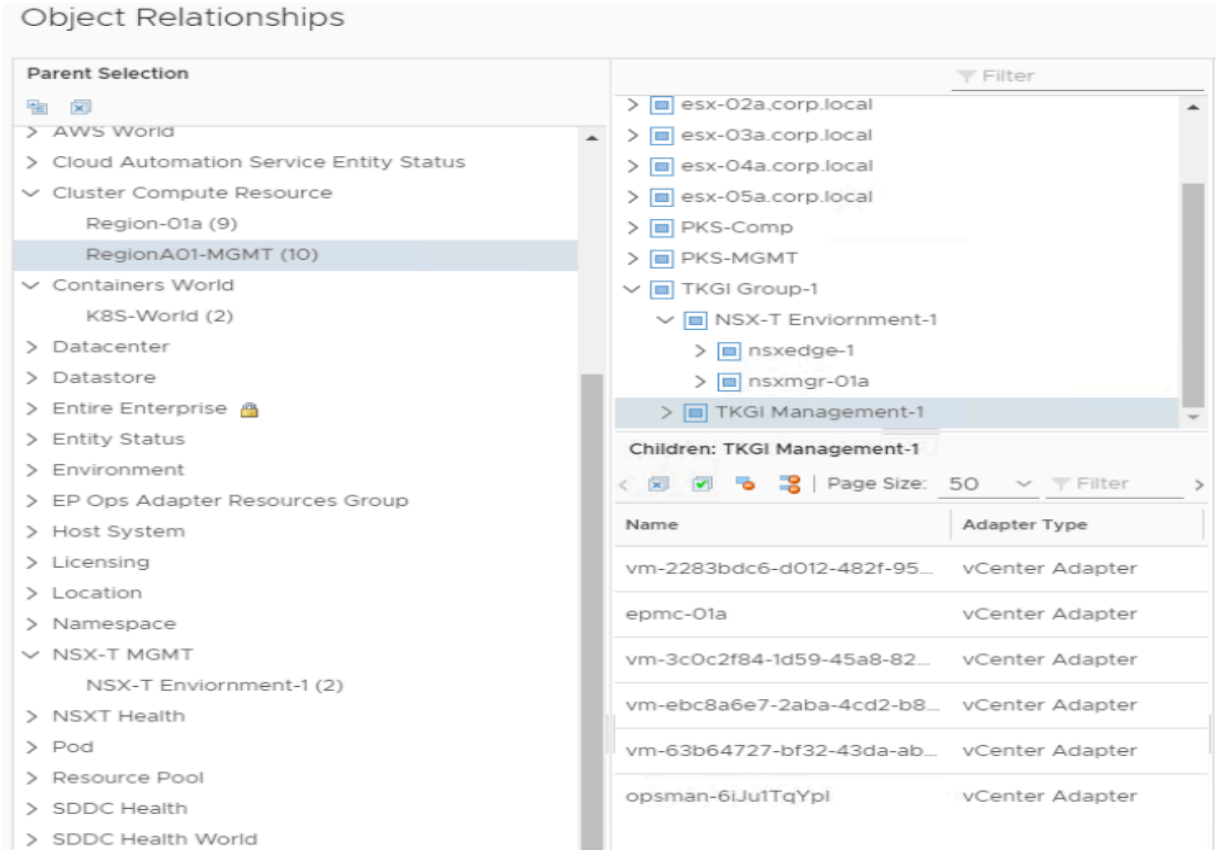

Figure 25: A sample screenshot is provided

Please use the following steps to configure the relationship for TKGI Group-1 and the cluster hosting that group.

Under Parent Selection pane select “Datacenter” and select the Datacenter under that. In my case it is,

Datacenter → RegionA01

Once you select the required Datacenter the components under that will be seen at the top of the middle pane. Select the cluster there (in my case it is RegionA01-MGMT). Once you select the cluster all its children will be seen at the bottom of the middle pane. Select TKGI Group-1 from right side pane and drag it under the Children section in the middle pane.

This will make TKGI Group-1 a children of cluster RegionA01-MGMT.

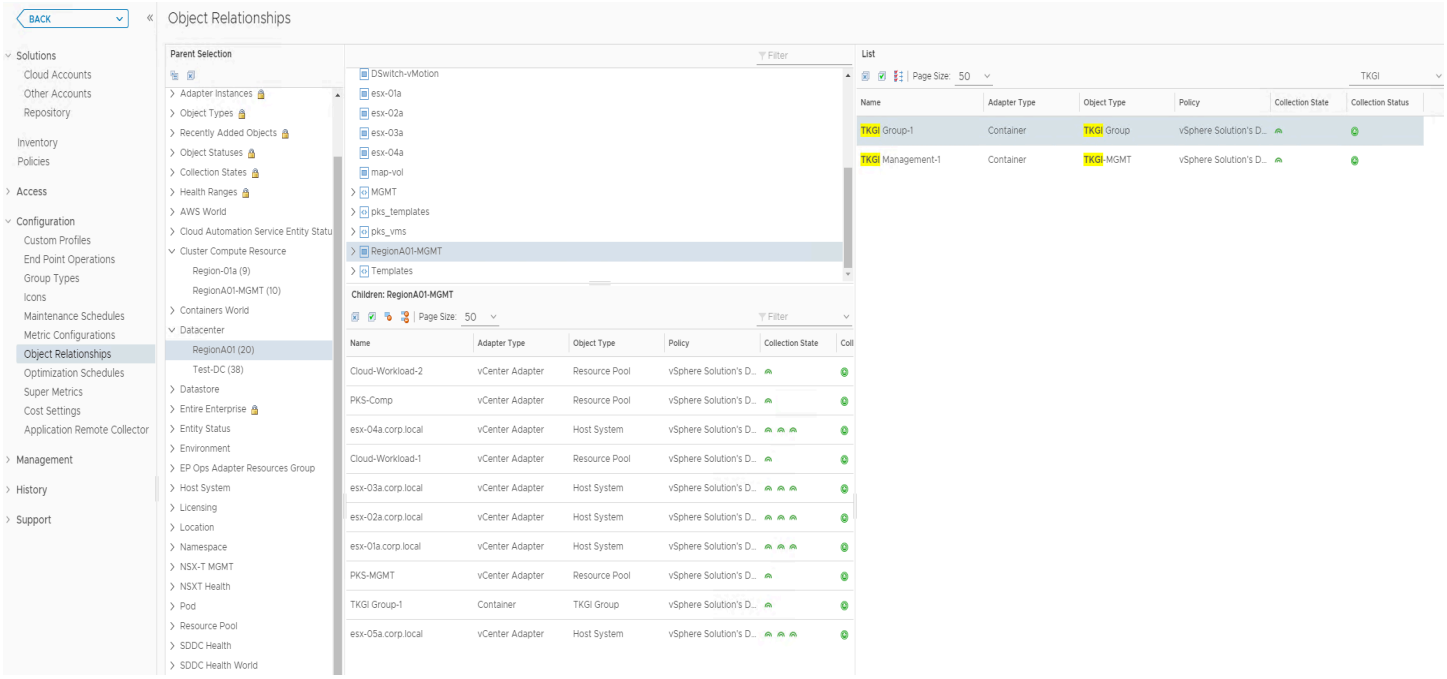

Figure 26: A sample screenshot of how to configure the relationship

9.1.4 Custom Dashboard

The last step is to configure the custom dashboard. Download the Dashboard zip file from GitHub repository - *https://github.com/sajaldebnath/tkgi-monitoring/blob/master/TKGI-Dashboard-vROps-v2.0.zip*. The file name is “TKGI-Dashboard-v2.zip”. Once downloaded, import it in vROps by going to Dashboards → Manage Dashboards → Import. Once all these steps are done, the dashboard for TKGI components are ready in vRealize Operations Manager.

Note: If the imported dashboard does not work, then, delete the dashboard and follow the steps provided in Appendix A to configure the dashboard manually. If the above steps are completed correctly, then the manual dashboard creation will take less than 30 minutes.

9.2 vRealize Network Insight

Next product is vRealize Network Insight (vRNI). We need to configure this in two steps. In the first step, we need to define an application. In the second step, we need to define a pinboard and add widgets from application and NSX-T information to this pinboard.

Remember, dashboards in vRNI is called pinboards. So vROps dashboards and vRNI pinboards are essentially the same thing.

9.2.1 Defining the Application

Let's define the application first. Go to Plan & Assess → Applications → Add Application (manual)

We will define 6 Tiers for the 6 components, namely opsman, Bosh, PKS-DB, Harbor, Pivotal Container Service, EPMC. Provided below are the details of each Tier.

- Name: opsman
    - Member: Custom VM Search → VMs where → Name like opsman and Cluster like 'RegionA01-MGMT'
- Name: EPMC
    - Member: Custom VM Search → VMs where → Name like epmc and Cluster like 'RegionA01-MGMT'
- Name: Bosh
    - Member: VM Names → 'vm-ebc8a6e7-2aba-4cd2-b82d-22151a53d3bf'
- Name: PKS-DB
    - Member: VM Names → 'vm-3c0c2f84-1d59-45a8-82d0-dcdaa3d5e8a4'
- Name: Harbor
    - Member: VM Names → 'vm-63b64727-bf32-43da-ab90-947825fd9708'
- Name: Pivotal Container Service
    - Member: VM Names → 'vm-2283bdc6-d012-482f-95dd-97e98f48039d'

Note: The VM name for Bosh, PKS-DB, Harbor and Pivotal Container Service will be different from environment to environment. So, please make changes accordingly. Also, vRNI does not have visibility into the VM to search for the name inside the VM (like we can in vROps). So, we will have to either depend on the VM naming pattern in vCenter or get the exact VM name. For OpsMan and EPMC the naming pattern exists and so we can search for them in a cluster but not for other components. If you want to find the VMs by the VM name and dynamically update the VM membership, then following the instructions given at *https://bosh.io/docs/vsphere-human-readable-names/*. With this feature enabled, when a new VM is created, it will be assigned to a name like ***instance-group-name_deployment-name_a81a26b3a9a8*** instead of a name like *vm-d6f0f537-18cd-4a1b-b0f5-ae03e8f590e8*. If you do not have that enabled, then, for the other VM's get their names from vROps and provide the value here. A sample screenshot is provided below:

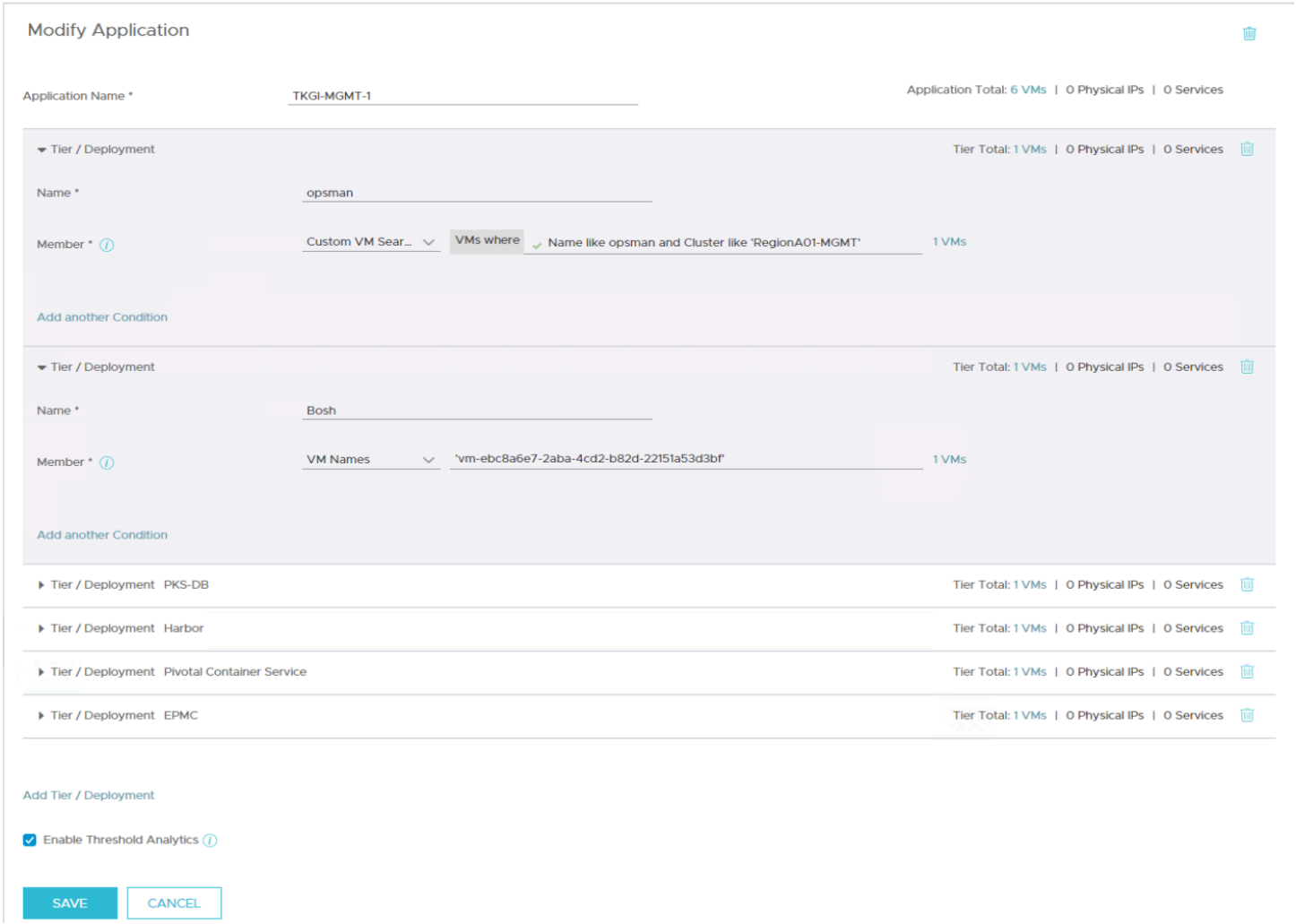

Figure 27: Sample picture of Application TKGI-MGMT-1

If you have more than one environment, then create a custom application for each of the environments.

Click on Save button to save the application.

Next, we click on the application name and it opens the application on another tab.

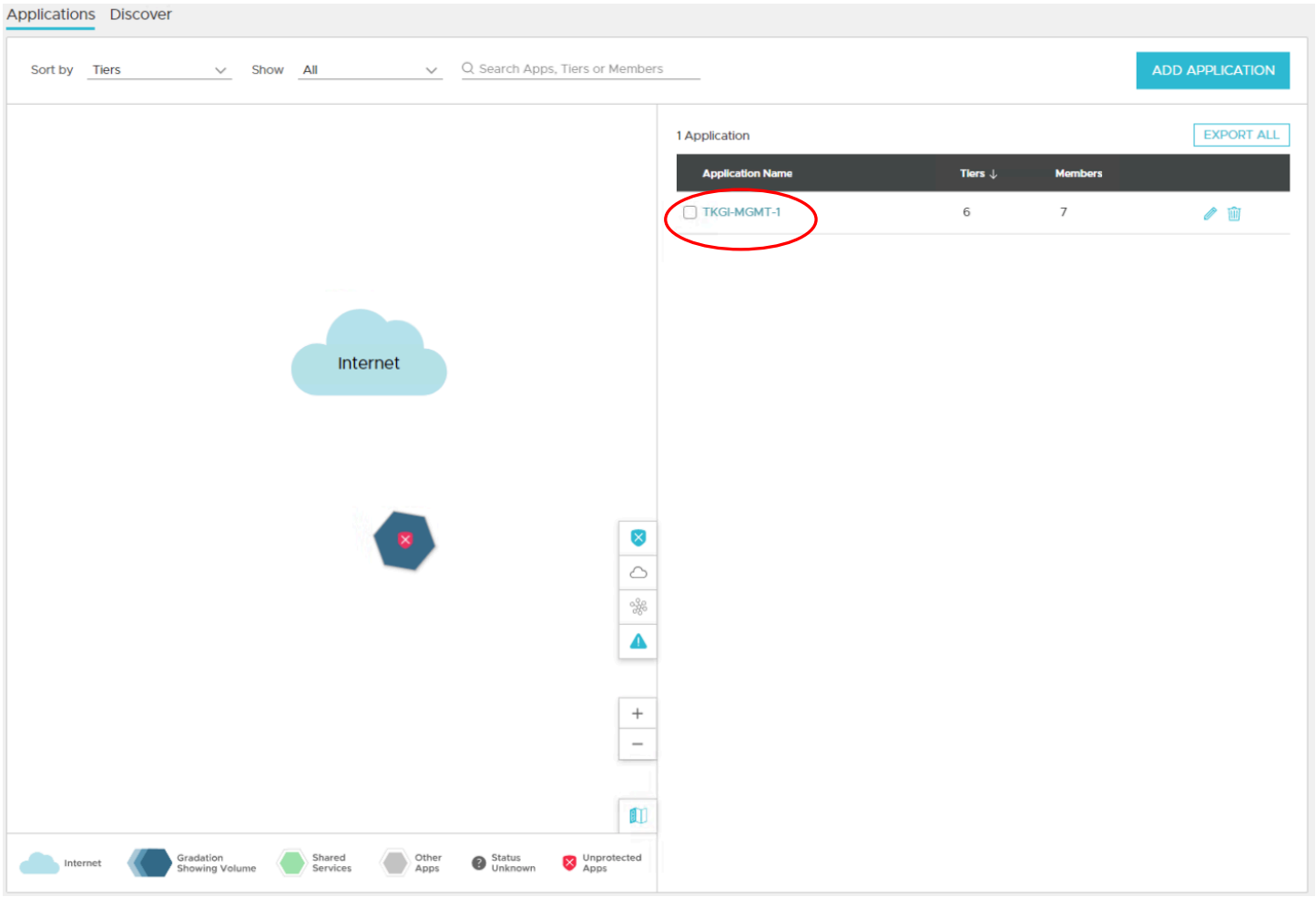

Figure 28: Click on the application name

In the newly opened page, we have all the details of the application. One thing remains is to create a pinboard and add the required views in the new pinboard.

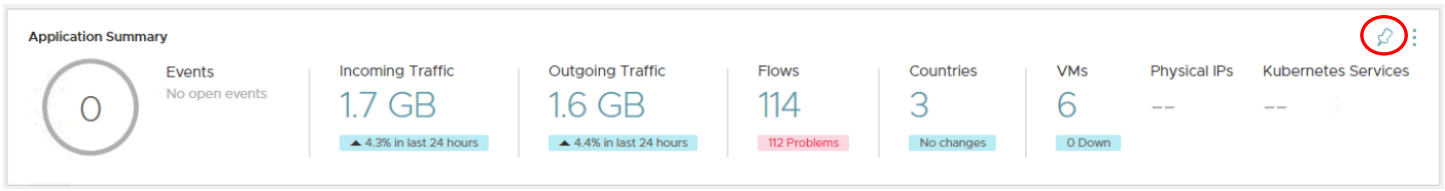

Figure 29: Widgets in the new application page

In the widget “Application Summary” click on the pin icon on the top right corner. This will ask you to add the widget to an existing pinboard or create a new pinboard. Click on Create New Pinboard link.

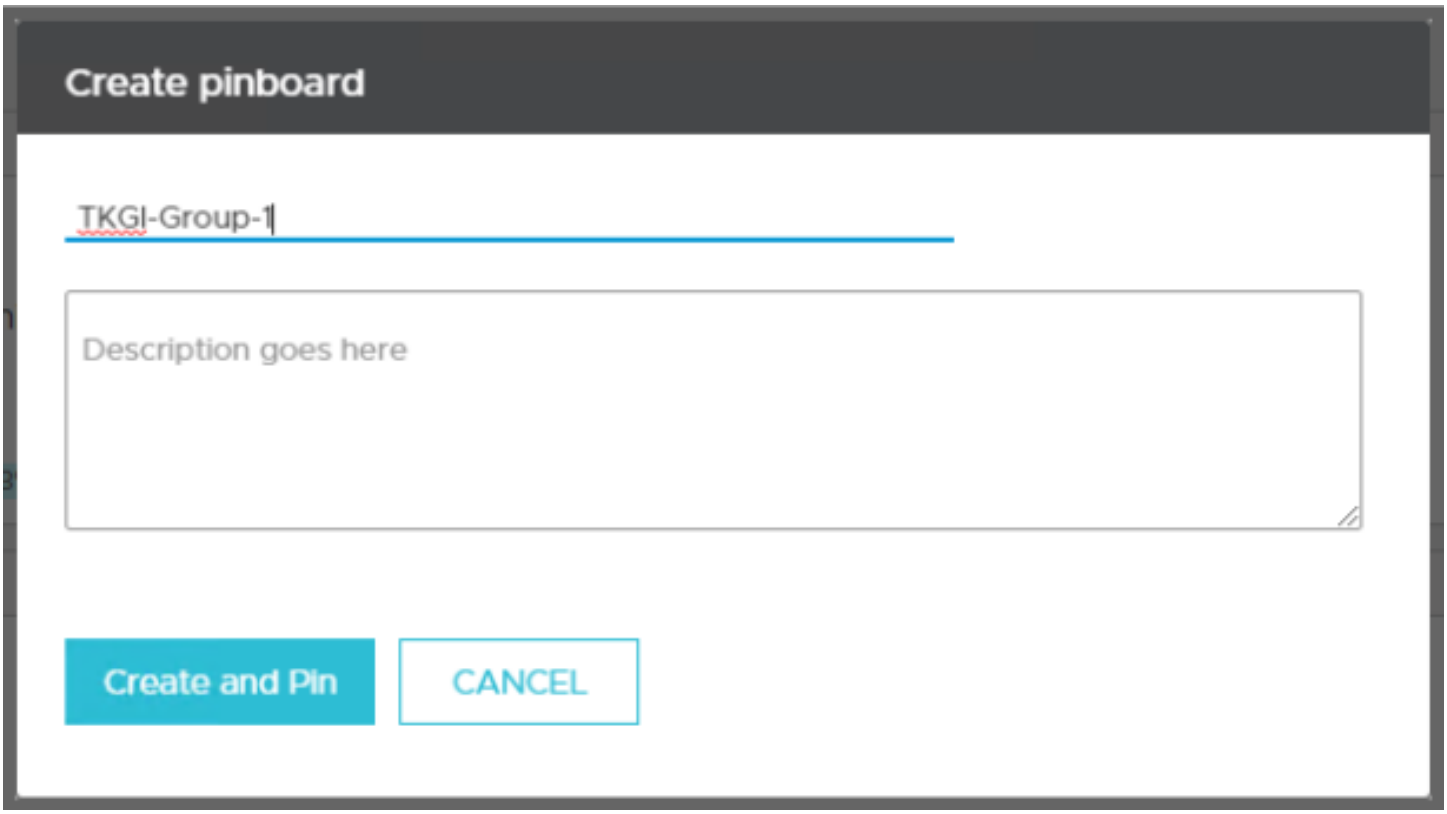

Figure 30: Create new pinboard

Provide the name as "TKGI-Group-1" as the pinboard name and click on Create and Pin.

From next time onwards just add the widgets to this existing pinboard.

Add the following widgets to the newly created pinboard.

- Application Summary
- Application Topology
- Events
- Application Members
- VM Metrics
- What's New (Last 24 Hrs)
- Micro segmentation

Next, search for the NSX-T Manager related to that environment. vRNI by default shows a lot of information about the NSX-T Manager. So, search for the NSX-T Manager in the search tab and click on the name of the manager in the results page. This will open the details of the NSX-T Manager page. **Remember** in the result, the NSX-T manager will be reported as a VM as well as NSX-T manager. Click on the link for the NSX-T Manager.

From the new page add the following widgets to the pinboard “TKGI-Group-1”.

- Summary
- Properties
- Firewall Rules by Number of Hits
- Topology
- Events
- Flow Analytics

Once this is done our Pinboard is ready.

Follow the similar procedure to configure pinboards for other environments (if you have more than one environment).

9.3 vRealize Log Insight

vRealize Log Insight has two management packs to showcase NSX-T and PKS information. The content pack for PKS is not released officially and only available at code.vmware.com. So, download it from *https://code.vmware.com/samples/7104/vmware-enterprise-pks-v1.0-content-pack?h=VMware%20-%20Enterprise%20PKS*.

To get the custom dashboards for vRealize Log Insight simply download the dashboards from GitHub repository *https://github.com/sajaldebnath/tkgi-monitoring/blob/master/TKGI-Dashboards-vRLI-v1.0.vlcp* and import it in the vRLI environment. The filename is “TKGI-Dashboards-vRLI-v1.0.vlcp”.

Follow the steps to import the content pack into vRLI. Go to Content Packs → IMPORT CONTENT PACK

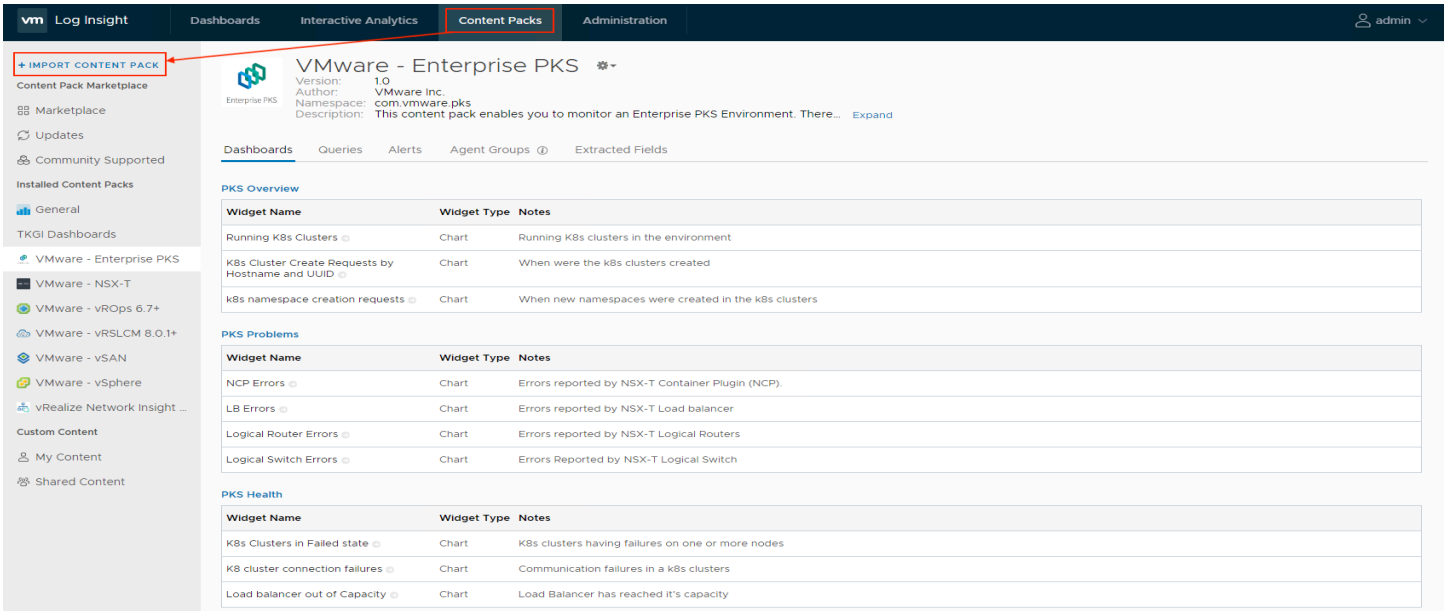

Figure 31: Importing content pack in vRLI – step 1

In the next step, in new windows, click on *Browse* to browse and select the downloaded "*TKGI-Dashboards-vRLI-v1.0.vlcp*" file and click on *Import* to import the pack.

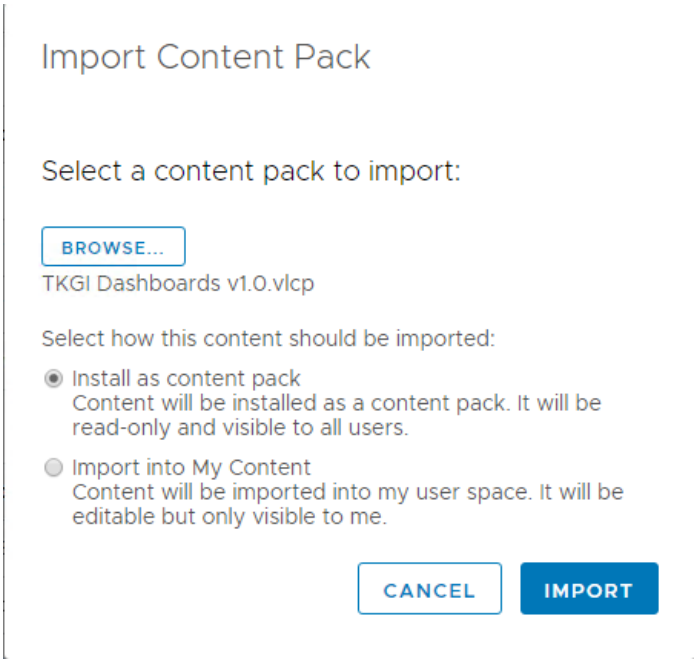

Figure 32: Importing content pack in vRLI – step 2

Part C

10 Failure Scenarios – Monitoring & Remediation

For many reasons, failure may occur at any of the TKGI layers and components. Question is, how do we monitor those failures and remediate them once they occur. Thus, reducing overall MTTR. In this last segment we will discuss about such various failure scenarios, and how to monitor and remediate them.

10.1 Different important layers and components

The following picture depicts the different layers of a TKGI environment and the components in it. All these layers and components needs to be monitored proactively to look out for possible issues.

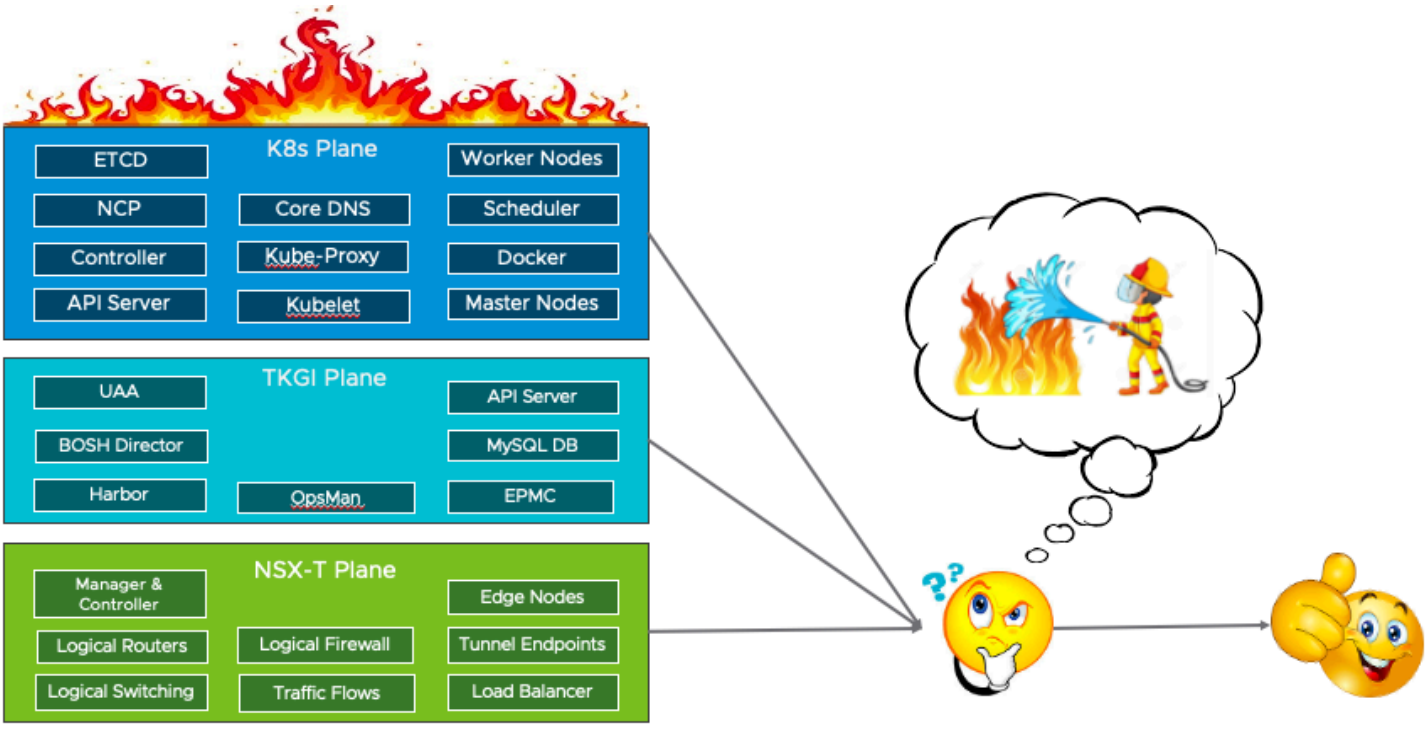

Figure 33: Different TKGI layers and important components in it

10.2 Failure points & Remediation

Provided below is a list of probable failure points and their remediations.

10.2.1 K8s failure points & remediation

Provided below is a list of failure scenarios their probable remediations for K8s layer.

Component	Env.	Monitored by	Failure Impact	Failure Mitigation	Remediation
NCP	Runs on Master Node VMs	Monitored by BOSH	Service Impacting. Cannot create any new resources. Network connectivity of existing pods might also be impacted.	TKGI control plane (BOSH) will monitor and restart.	Follow best practices to deploy minimum 3 Master node. Use Valero for complete failure and recovery of K8S cluster
Core DNS	Runs On K8 worker nodes	VRNI, Monitored by BOSH	Service impacting. When CoreDNS is down, applications are unable to use DNS for service discovery.	Use more than one worker node , spread pods across nodes.	Always have more than one replica of a pod running on different nodes.
Scheduler	Runs on Master node VMs.	Monitored by BOSH, vRNI	Not service impacting but cannot place new workloads and move existing workloads to other nodes.	TKGI control plane (BOSH) will monitor and restart.	Follow best practices to deploy minimum 3 Master node. Use Valero for complete failure and recovery of K8S cluster
Controller	Runs on Master node VMs.	vRNI, Monitored by BOSH	Not service impacting but critical to ensure the cluster can reconcile the current state of the cluster with the users' desired state.	TKGI control plane (BOSH) will monitor and restart.	Follow best practices to deploy minimum 3 Master node. Use Valero for complete failure and recovery of K8S cluster
Kube-proxy	Runs On K8 worker nodes and	vRNI, Monitored by BOSH	Critical to ensure workloads can access Pods and Services running on other nodes. Services are not reflected on a node's iptables or IPVS configuration	BOSH will monitor and restart. But this could be because of an underlying problem	Have at least 3 worker nodes Spread workload replicas across nodes. Use Valero for complete failure and recovery of K8S cluster
Docker	Runs On K8 worker nodes and	vRNI, Monitored by BOSH	Service Impacting, pods will crash. New pods will not startup	BOSH will monitor and restart. But this could be because of an underlying problem	Have at least 3 worker nodes Spread workload replicas across nodes. Use Valero for complete failure and recovery of K8S cluster
API Server	Runs on Master node VMs.	vRNI, Monitored by BOSH	Not Service Impacting. Unable to stop, update, or start new pods, services, replication controller	TKGI control plane (BOSH) will monitor and restart.	Follow best practices to deploy minimum 3 Master node. Use Valero for complete failure and recovery of K8S cluster
kubelet	Runs On K8 worker nodes and	vRNI, Monitored by BOSH	Crashing kubelet cannot start new pods on the node	BOSH will monitor and restart. But this could be because of an underlying problem	Have at least 3 worker nodes. Use Valero for complete failure and recovery of K8S cluster

***Service:** Application running on K8S cluster provisioned and managed by TKGI. Use HA deployment (2 sites) for mission critical services.

Component	Env.	Monitored by	Failure Impact	Failure Mitigation	Remediation
ETCD	Runs on Master node VMs.	BOSH	None.	Covered by remaining two ETCD copies. TKGI control plane (BOSH) will monitor and restart.	Follow best practices to deploy minimum 3 Master node. Use Valero for complete failure and recovery of K8S cluster
Master Node(s)	VMs	vRNI – Primary vROps – at VM level only	None.	Covered by remaining two Master nodes. TKGI control plane (BOSH) will monitor and restart.	Follow best practices to deploy minimum 3 Master nodes. Use Valero for complete failure and recovery of K8S cluster
Worker Node(s)	VMs	vRNI – Primary vROps – at VM level	Depending on the workload, there maybe momentary response slow down.	Covered by remaining Worker nodes. TKGI control plane (BOSH) will monitor and restart.	Select the number of Worker nodes based on workload and responsiveness needs. Define minimum two Worker nodes.

10.2.2 TKGI failure points & remediation

Provided below is a list of failure scenarios for TKGI layer and their probable remediations.

Component	Env.	Monitored by	Failure Impact	Failure Mitigation	Remediation
BOSH Director	VM	vROps/vRLI/vRNI	Operations only. Could impact service - remediation of other failures: e.g. Workers.	VM protection from vSphere.	Use BBR to backup and restore.
API Server	VM(1)	vROps/vRLI/vRNI	Operations only. Not service impacting.	BOSH monitors and respawn and restarts.	Use BBR to backup and restore.
MySQL	VM(2)	vROps/vRLI/vRNI	Operations only. Not service impacting.	BOSH monitors and respawn and restarts.	Use BBR to backup and restore.
UAA	VM(1)	vROps/vRLI/vRNI	Operations only. Not service impacting. Cannot validate to the cluster	BOSH monitors and respawn and restarts.	Use BBR to backup and restore.
OpsMan	VM	vROps/vRLI/vRNI	Operations only. Not service impacting.	VM protection from vSphere.	vSphere snapshot, or export configurations
EPMC	VM(3)	vROps/vRLI/vRNI	Operations only. Not service impacting.	VM protection from vSphere.	vSphere snapshot
Harbor	VMs	vROps/vRLI/vRNI	Operational – deployment time.	VM protection from vSphere.	Use harbor in HA mode with replication strategy for images

***Service:** Application running on K8S cluster provisioned and managed by TKGI. Use HA deployment (2 sites) for mission critical services.

(1,2) As of TKGI 1.7 UAA and API Server run in their own VM (single VM for both); MySQL separated to another VM. Until 1.7, all 3 were in a single VM.

UAA/API HA (multiple instances of UAA and API running at the same time) will be supported in TKGI 1.8 or 1.9

(3) EPMC is used as an optional TKGI installation and K8S cluster management tool.

10.2.3 NSX-T failure points & remediation

Provided below is a list of failure scenarios for NSX-T layer and their probable remediations.

Component	Env.	Monitored by	Failure Impact	Failure Mitigation	Remediation
Manager. & Controller	VMs	vROps/vRNI/vRLI	Operations only. Not service impacting.	Cluster of 3. Other two will cover. Failed VM will be restarted by vSphere?	Use recommended best practices for clustered deployment.
Edge Nodes	VMs	vROps/vRNI/vRLI	Service Impact	vSphere will restart Edge VM	Configure NSX-T Edges in Active/Standby mode and monitor availability
Tunnel Endpoints		vRNI	East-West traffic is impacted	Change in MTU or other physical network changes should be closely monitored	Ensure adequate capacity in the the vSphere cluster to migrate workload
Logical Routers	NSX-T/ESXi Host	vROps/vRNI/vRLI	Impact network resources connected to the switch	NSX-T monitors and reports	Review Configuration limits regularly
Logical Firewall	NSX-T/ESXi Host	vROps/vRNI/vRLI	Impact on protected resources	NSX-T monitors	Review Configuration limits regularly
Load Balancer	NSX-T/ESXi Host	vRNI/vROps	Impact on all load balancers include TKGI clusters, etc. This has an impact on services	NSX-T monitors	Review Configuration limits regularly
Logical Switching	NSX-T/ESXi Host	vRNI/vROps	Impact on all traffic	NSX-T monitors	Review Configuration limits regularly
Traffic Flows	NSX-T/ESXi Host	vRNI	None	NSX-T monitors	Review Configuration limits regularly

***Service:** Application running on K8S cluster provisioned and managed by TKGI. Use HA deployment (2 sites) for mission critical services.

11 Conclusion

We hope this document was useful. As you try these configuration steps, please provide any feedback or questions in the comments section of code.vmware.com. Also, do let us know if you have any suggestions or if you would like to see guidance on other topics.

12 Acknowledgement

The following people helped in creating the content for this document:

- Rag Ramanathan
- Raghu Pemmaraju
- Riaz Mohammed
- Ron Walski
- Francis Guillier

13 Appendix

13.1 Appendix A

Provided below are detailed steps for manually creating the vROps dashboard. If the imported dashboard did not work, then to recreate the dashboard using the below-mentioned steps.

Start with creating a new Dashboard. Name the dashboard “TKGI Dashboard”.

13.1.1 Read Me widget

Start by adding a “Text Display” widget. Edit the scoreboard and name it “*0. Read Me (Expand to see details)*”

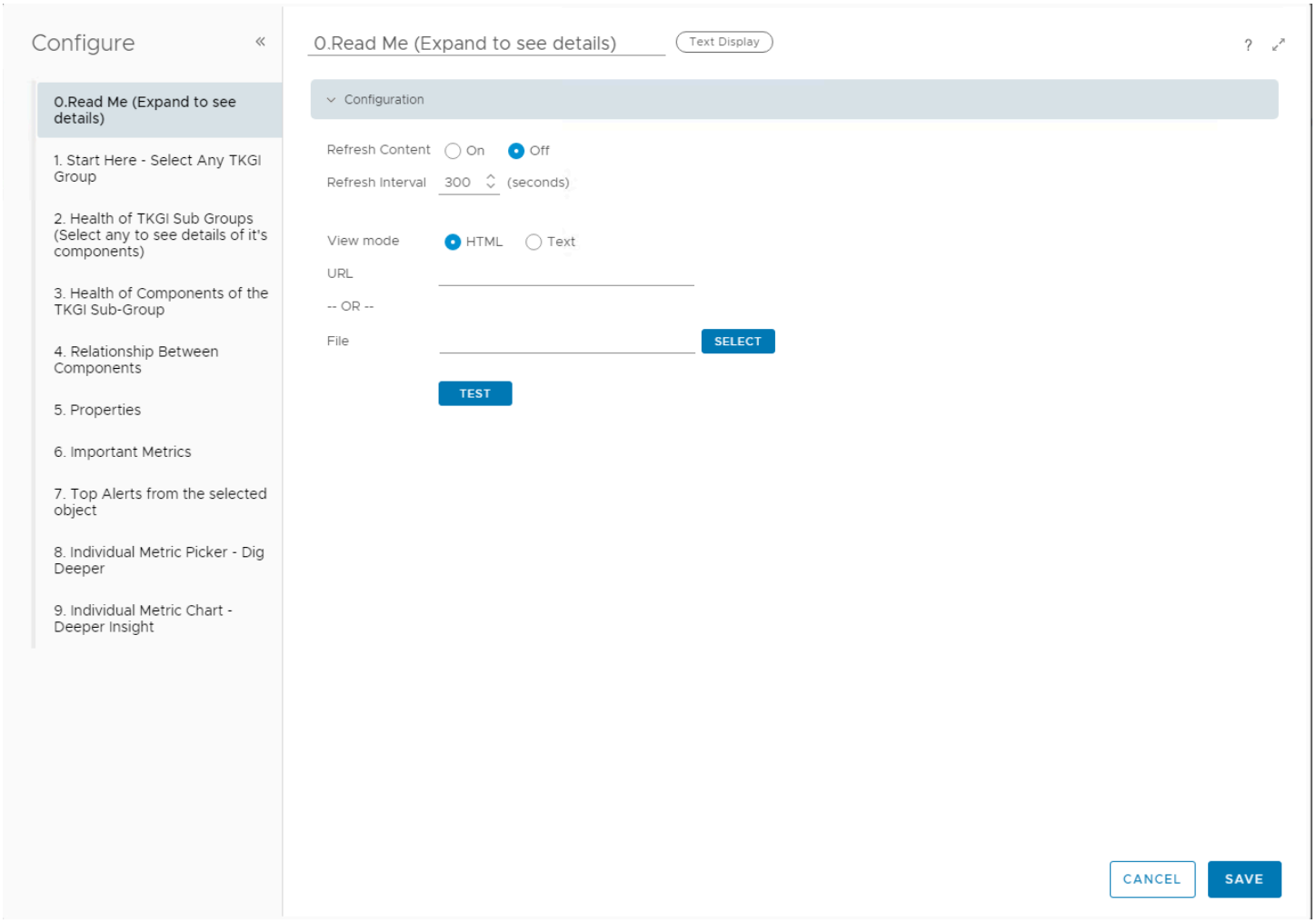

Figure 34: Read Me widget

Once added, Edit the widget in the dashboard to add the text. In my case, I added the following texts.
“The TKGI Dashboard curates the TKGI control plane components across all the environment in a single pane of glass. This includes an end to end topology map showcasing the health of all the objects along with upstream and downstream dependencies.

Upon clicking any object in the relationship tree, the related inventory of components can be viewed and exported from this dashboard."

13.1.2 Start Here widget

Next, add a scoreboard. Edit the scoreboard and name it "*1. Start Here - Select Any TKGI Group*"

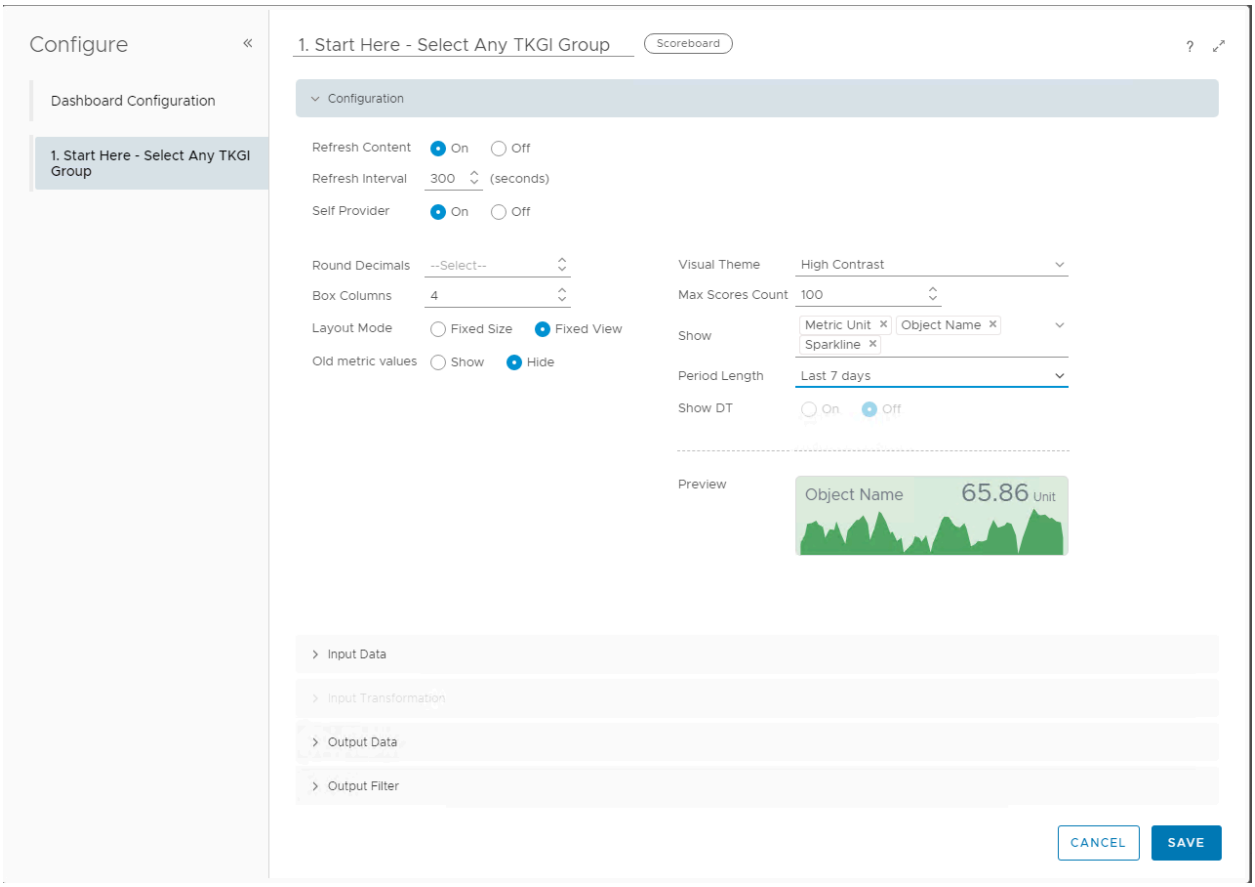

Figure 35: Start Here widget

Make sure the Configuration page looks exactly like the above screenshot. Specially change the following:

- Refresh Content → On
- Self-Provider → On
- Box Columns → 4
- Visual Theme → High Contrast
- Show →Object Name, Metric Unit, Sparkle
- Period Length → 7 days (optional)

Input Data → All

Output Data → In the Output Data page, click on ✚ sign to add a new Metric. In the new window type "TKGI" in the left side pane to filter out the objects and select "TKGI Group". Once selected, all the metrics of the object will be listed on the right-hand side of the pane. Select All Metrics → Health → Average Score of All Group Members. Click OK.

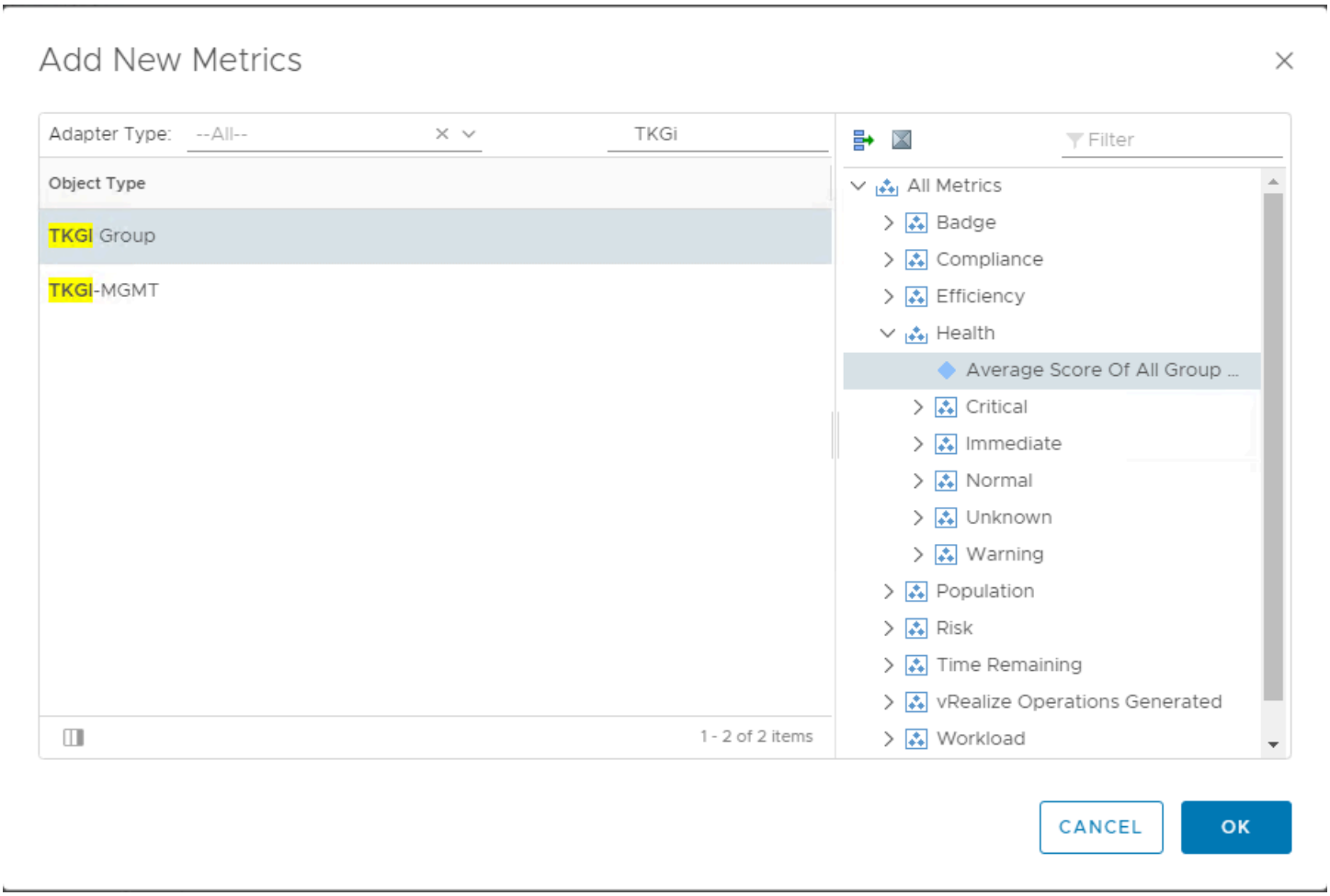

Figure 36: Metric Selection

Once done, make the following changes to the selected metric.

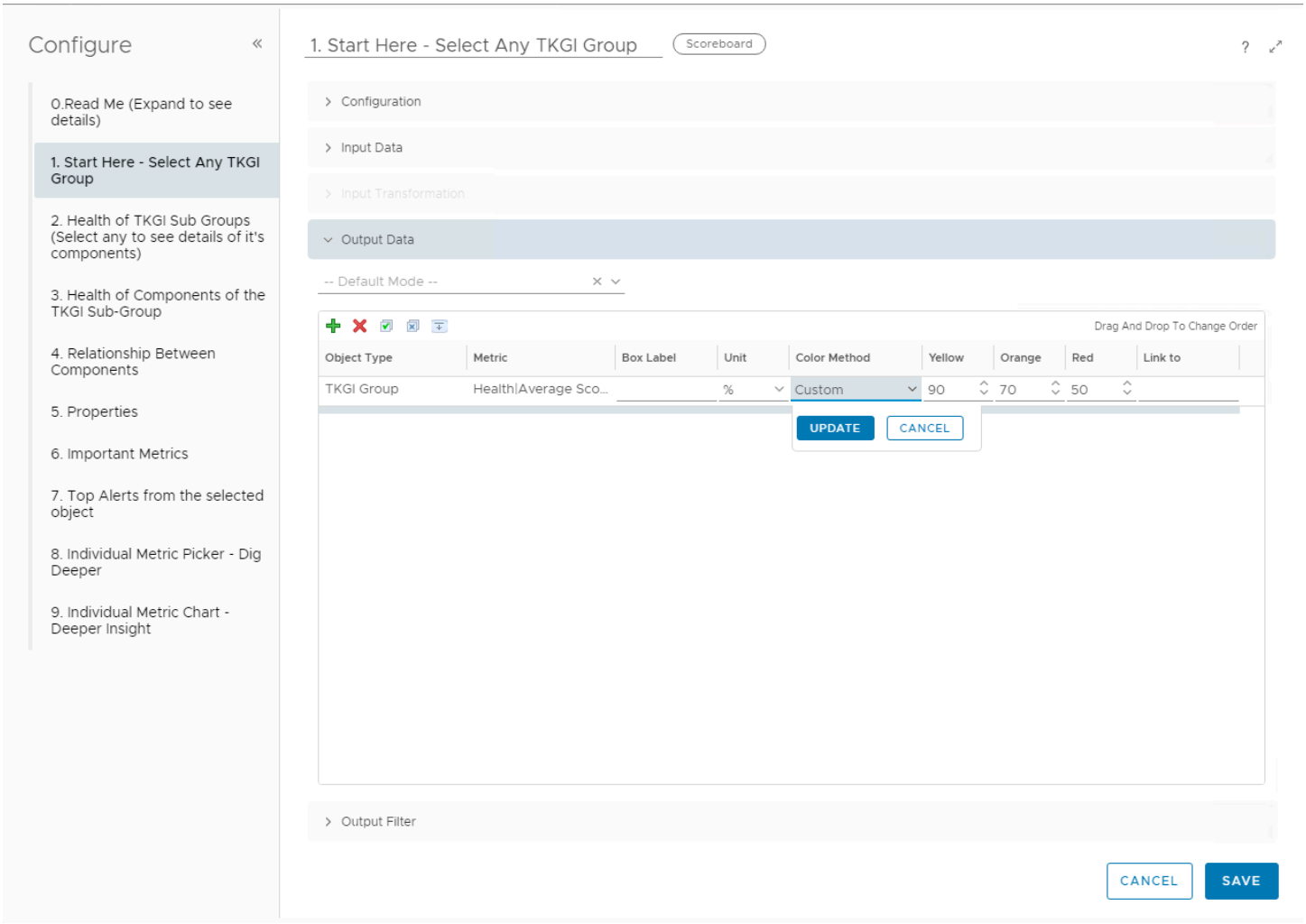

Figure 37: Fine adjustments

Unit → %
Color Method: Custom
Yello → 90
Orange → 70
Red → 50
Modify the values as per your requirement. Select SAVE to save the widget.

13.1.3 Health of TKGI Subgroups

Next, add a scoreboard. Edit the scoreboard and name it "*2. Health of TKGI Subgroups (Select any to see details of its components)*"
Make sure the Configuration page looks exactly like the below screenshot. Specially change the following:

- Refresh Content → On
- Self-Provider → Off
- Box Columns → 4

- Visual Theme → High Contrast
- Show →Object Name, Metric Unit, Sparkle
- Period Length → 7 days (optional)

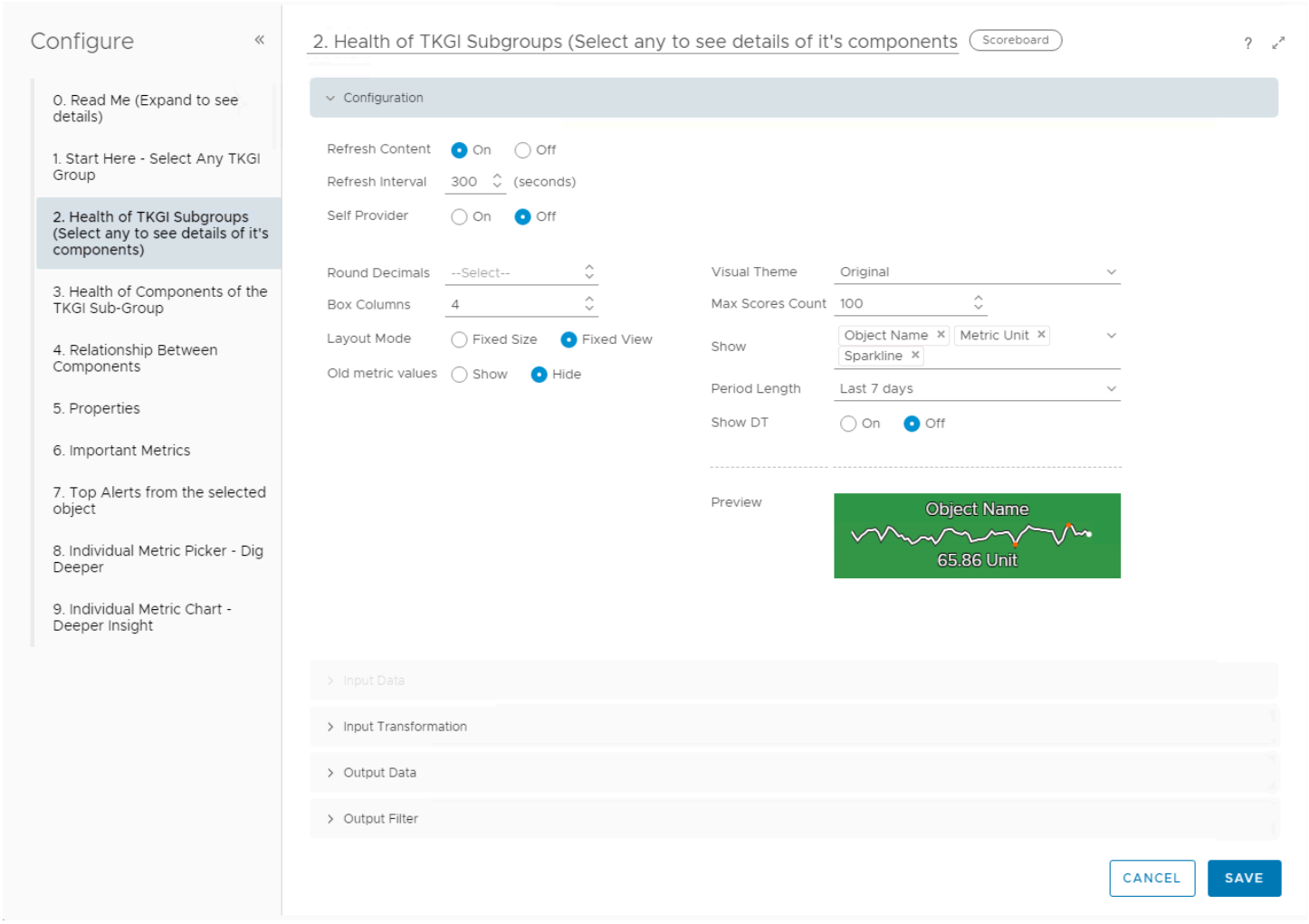

Figure 38: Health of TKGI Subgroups

Input Transformation → Children → Depth 1

Output Data → In the Output Data page, follow the process mentioned in step 13.1.2. Select the Metrics for "*TKGI-MGMT*" and "*NSX-T MGMT*". The Metric to be selected is again "*Health → Average Score of all group members*". Also, edit the default values to match as given in the following screenshot. Click OK to save.

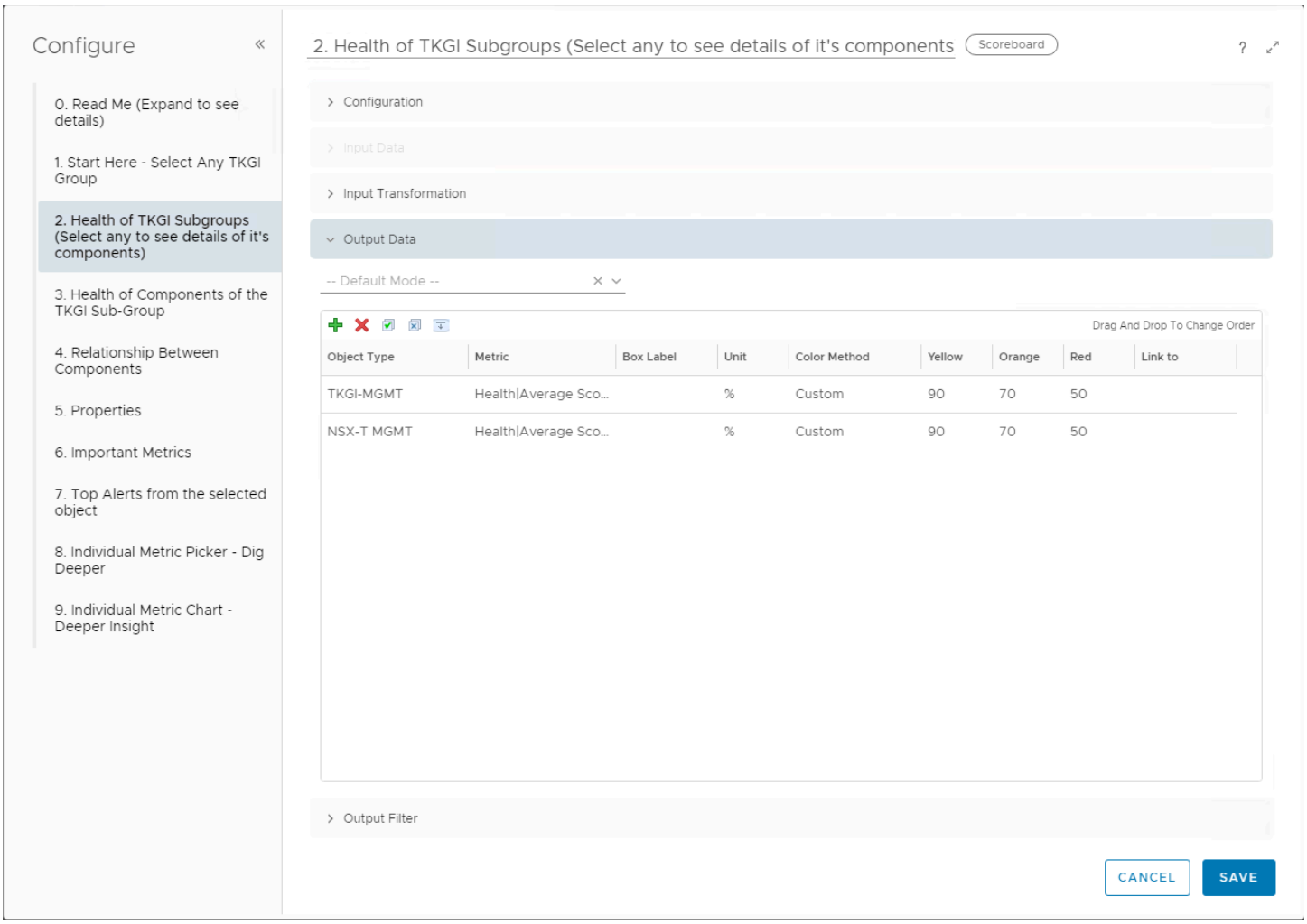

Figure 39: Output Data

Select SAVE to save the widget.

13.1.4 Health of Components of the TKGI Sub-Group

Next, add another scoreboard. Edit the scoreboard and name it "*3. Health of Components of the TKGI Sub-Group*".
Make sure the Configuration page looks exactly like the below screenshot. Specially change the following:

- Refresh Content → On
- Self-Provider → Off
- Box Columns → 10
- Visual Theme → High Contrast
- Show →Object Name, Metric Unit, Sparkle
- Period Length → 7 days (optional)

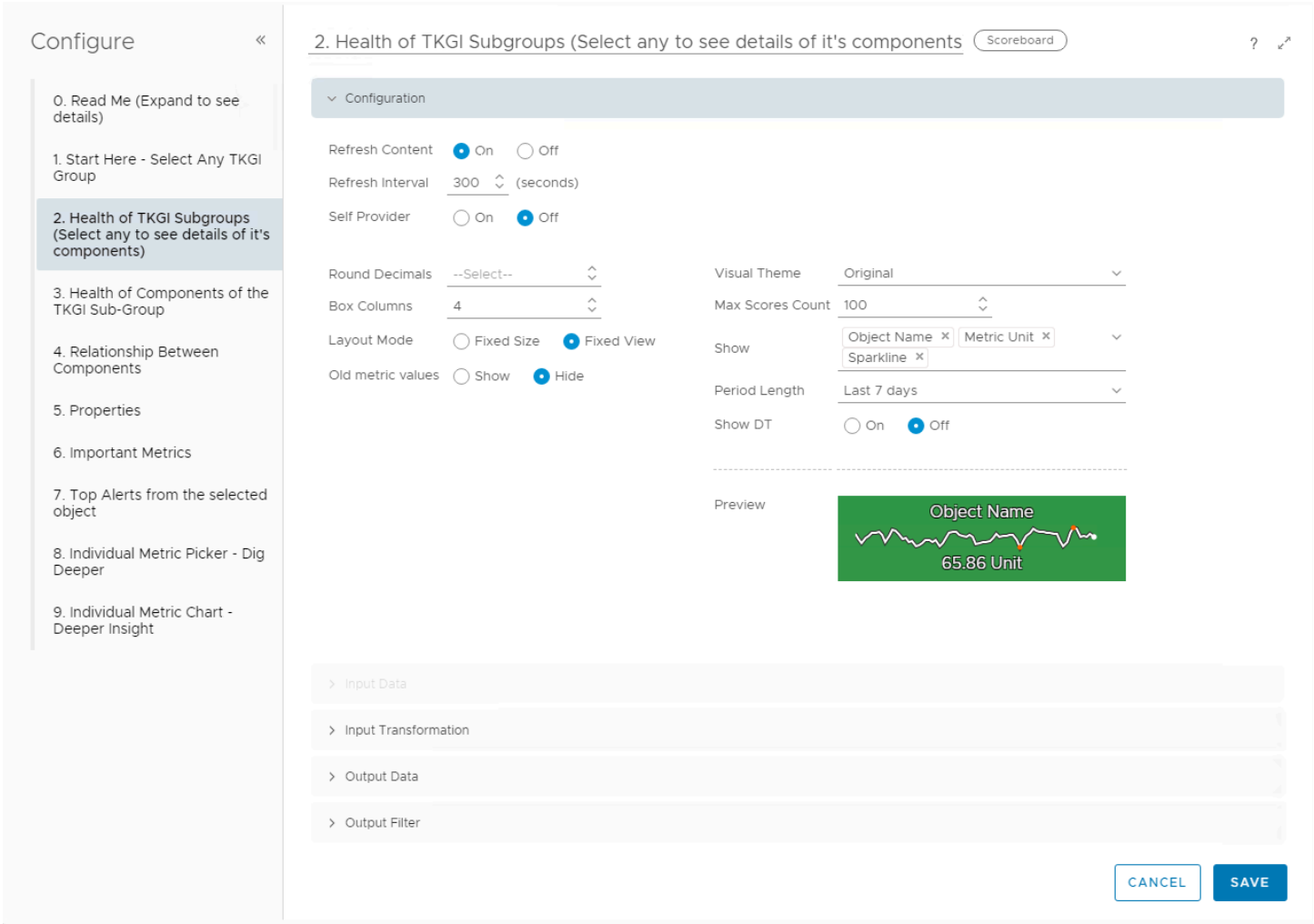

Figure 40: Health of TKGI Subgroups

Input Transformation → Children → Depth 1

Output Data → In the Output Data page, follow the process mentioned in step 13.1.2. Select the Metrics for "*Virtual Machine*". The Metric to be selected is "*Badge → Health*". Do not make any changes to the default color values. Click SAVE to save.

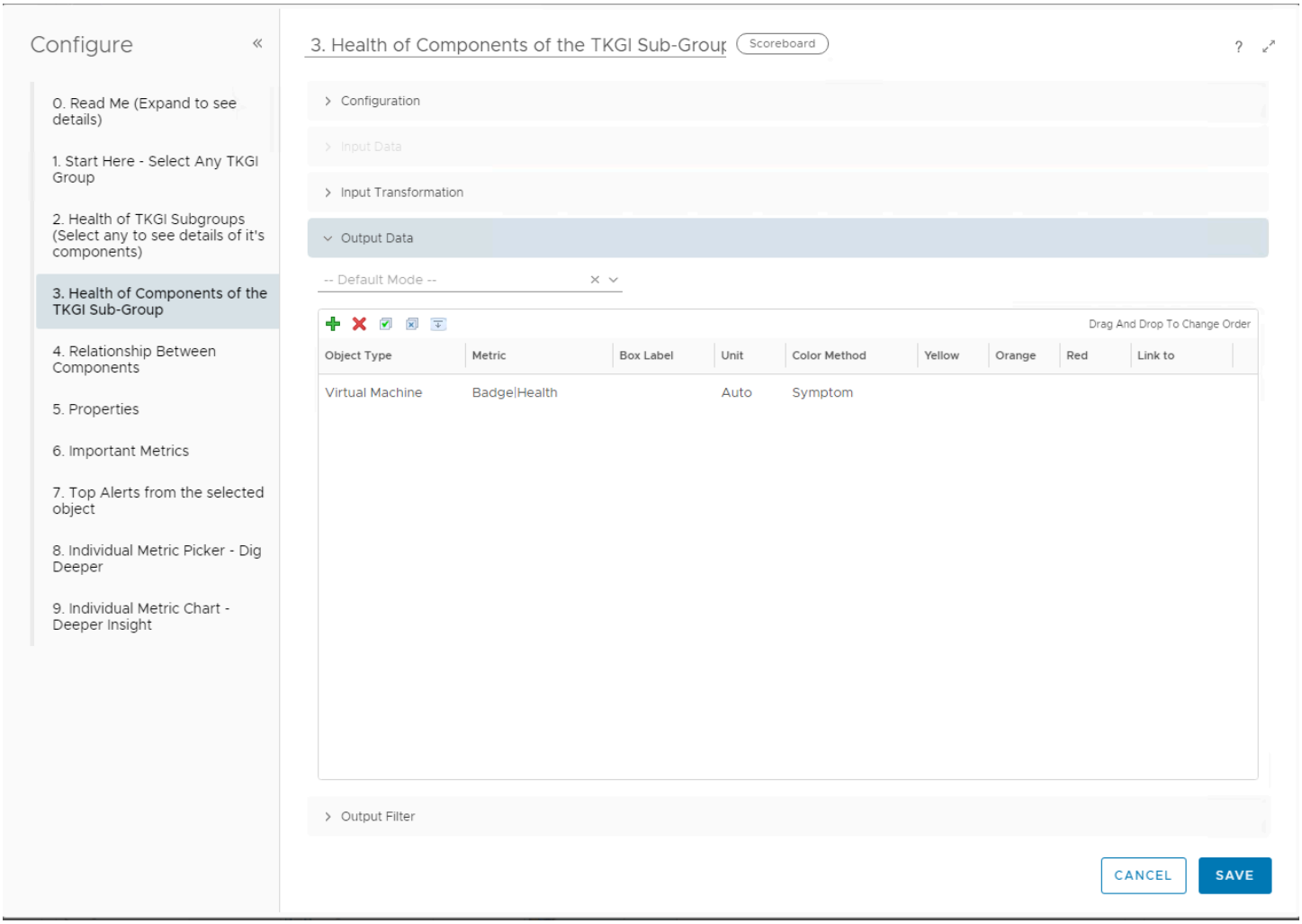

Figure 41: Output Data

Select SAVE to save the widget.

13.1.5 Health of Components of the TKGI Sub-Group

Next, add a "*Object Relationship (Advanced)*" widget. Edit and name the widget "*4. Relationship Between Components*". Select the parameters as following.

- Refresh Content → On
- Self-Provider → Off
- Parents Depth → 3
- Children Depth → 3
- Inventory trees → All Objects

Click on SAVE to save the widget.

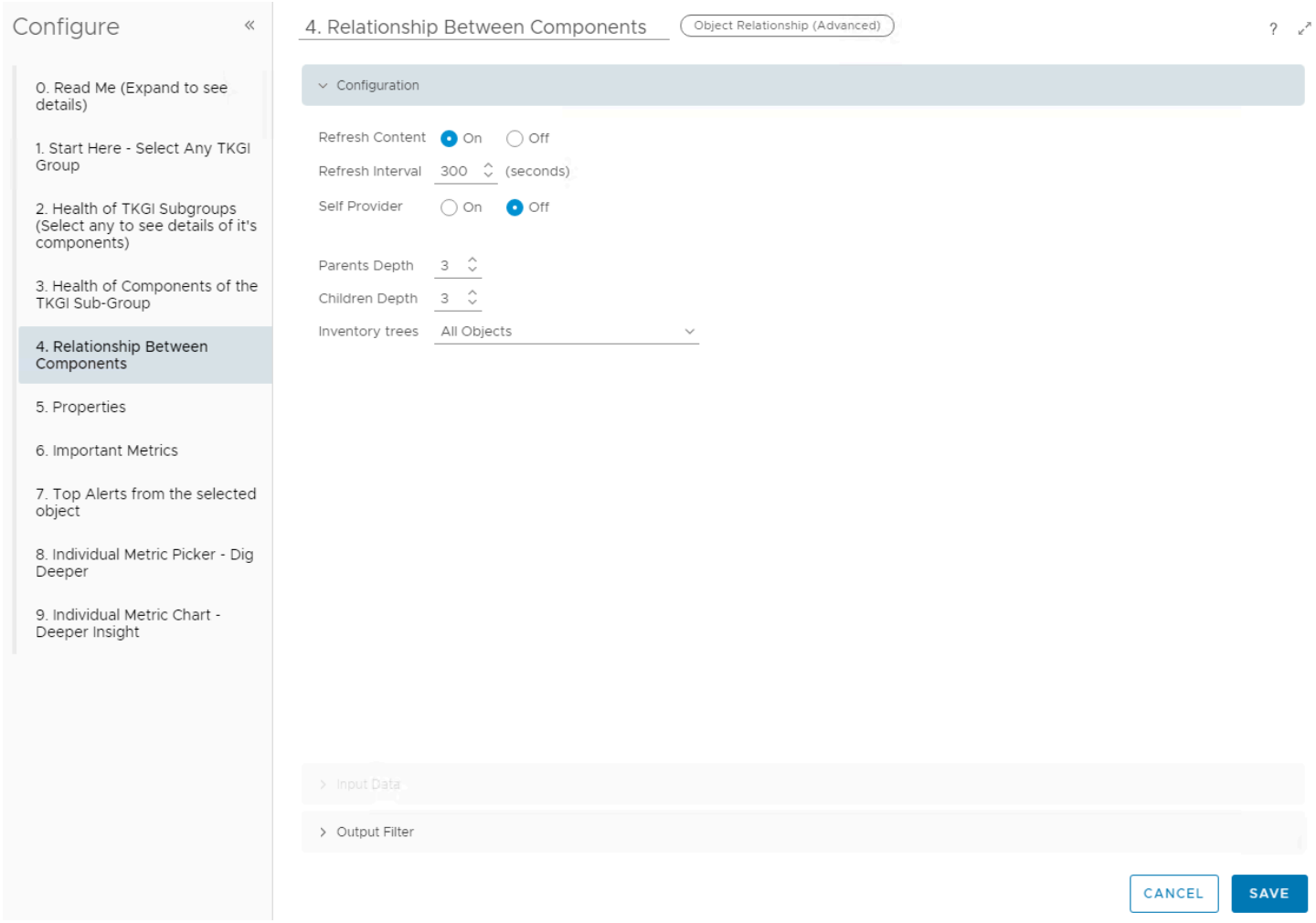

Figure 42: Relationship widget

13.1.6 Properties

Next, add a "*Scoreboard"* widget. Edit and name the widget "*5. Properties*". Select the parameters as following.

- Refresh Content → Off
- Self-Provider → Off
- Box Columns → 2
- Visual Theme → Original
- Metric Scores Count → 1000
- Show → Metric Name, Metric Unit
- Old Metric values → Hide

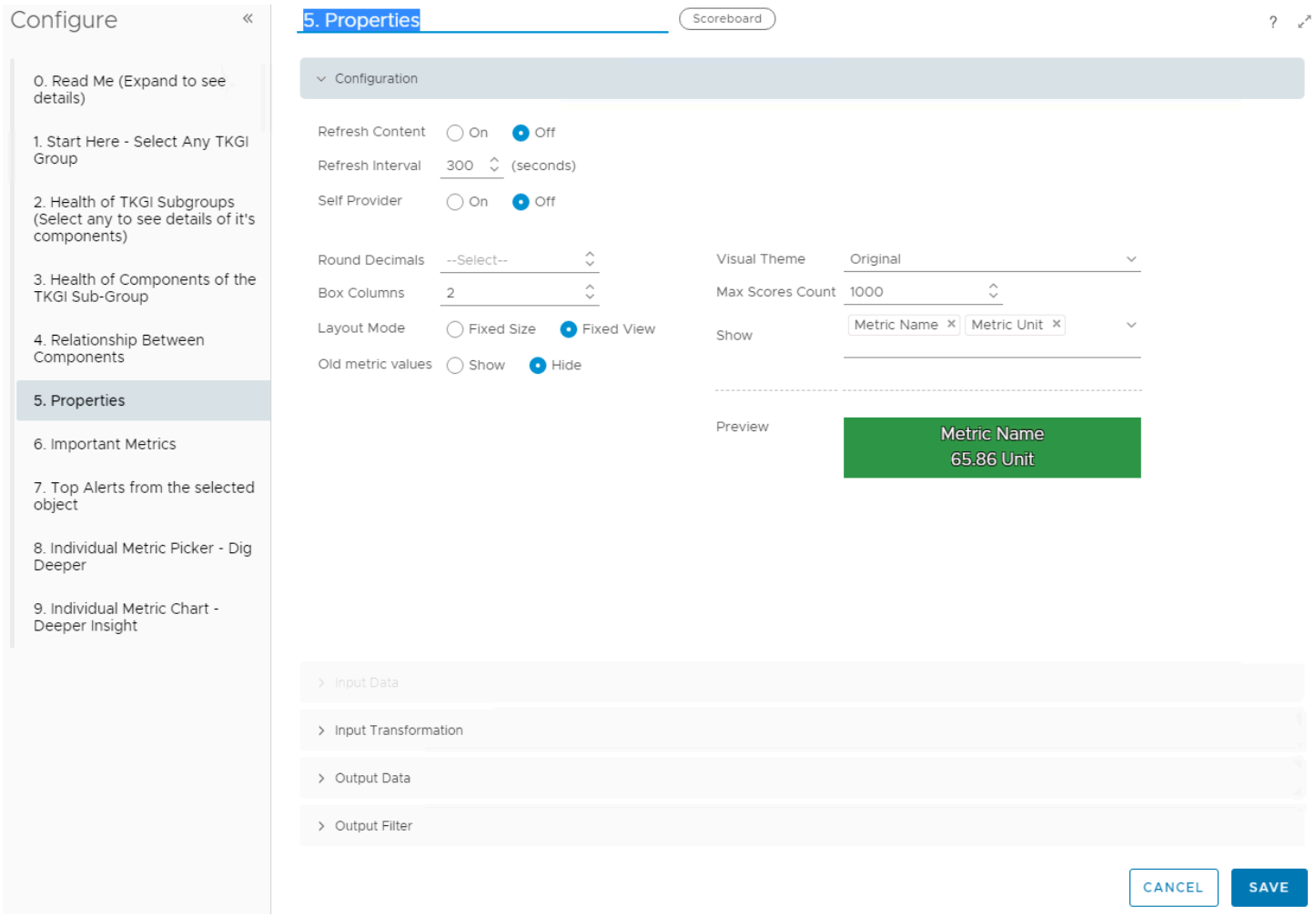

Figure 43: Properties

Input Transformation → Self

Output Data → Select the metrics by following the process mentioned in 13.1.2. Select the following metrics as given in the screenshot.

Summary → Configuration → Product Name

Summary → Custom Tag:Instance_group → Value

Summary → Guest Operating System → OS IP Address

Summary → Parent Cluster

Summary → Parent Host

Summary → Parent DataCenter

Summary → Parent vCenter

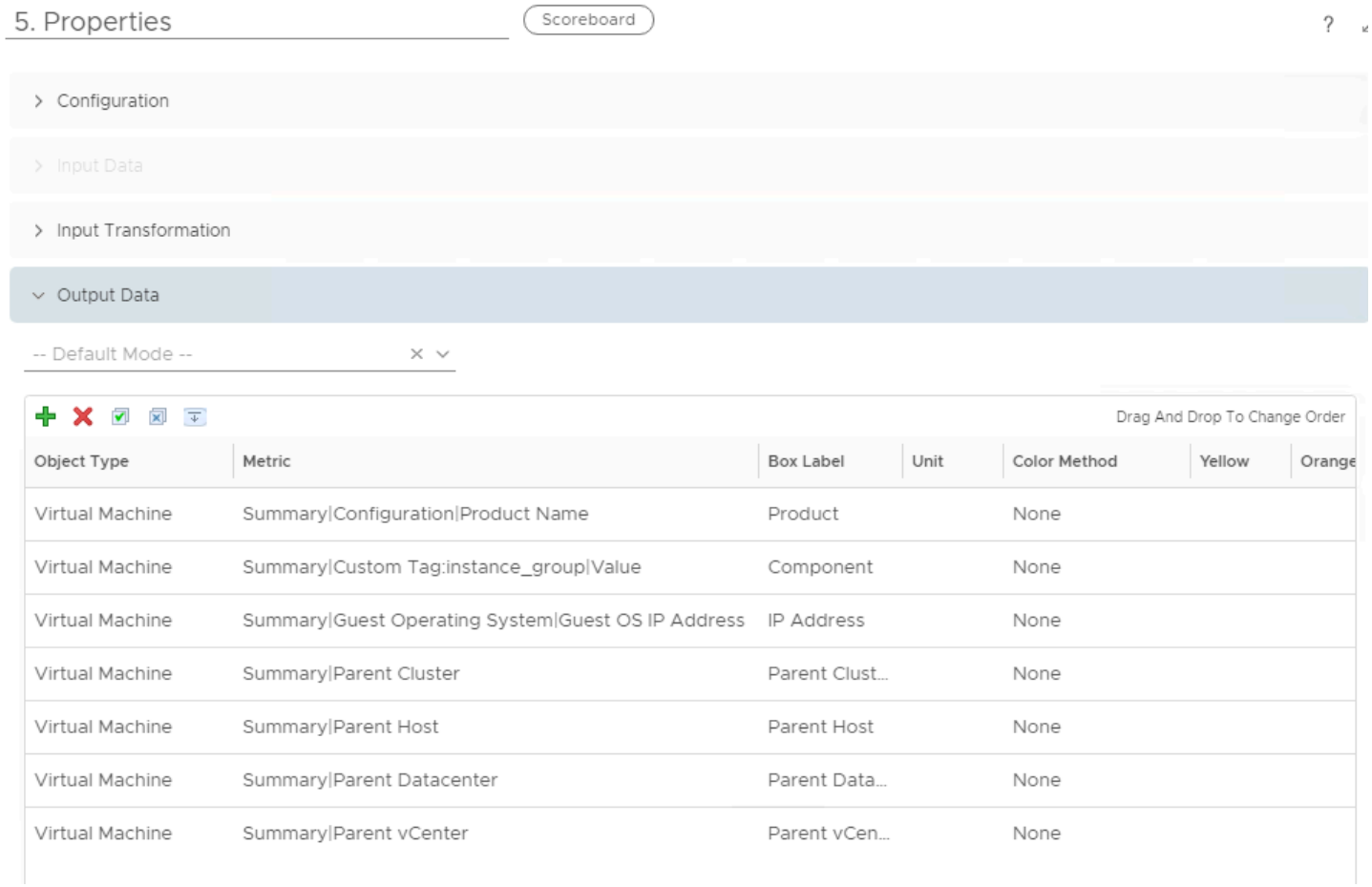

Figure 44: Property list

Click on SAVE to save the widget.

13.1.7 Important Metrics

Next, add a "*Property List*" widget. Edit and name the widget "*6. Important Metrics*". Select the parameters as following.

- Refresh Content → Off
- Self-Provider → Off
- Box Columns → 2
- Visual Theme → Compact
- Metric Scores Count → 1000
- Show Metric Full Name → On

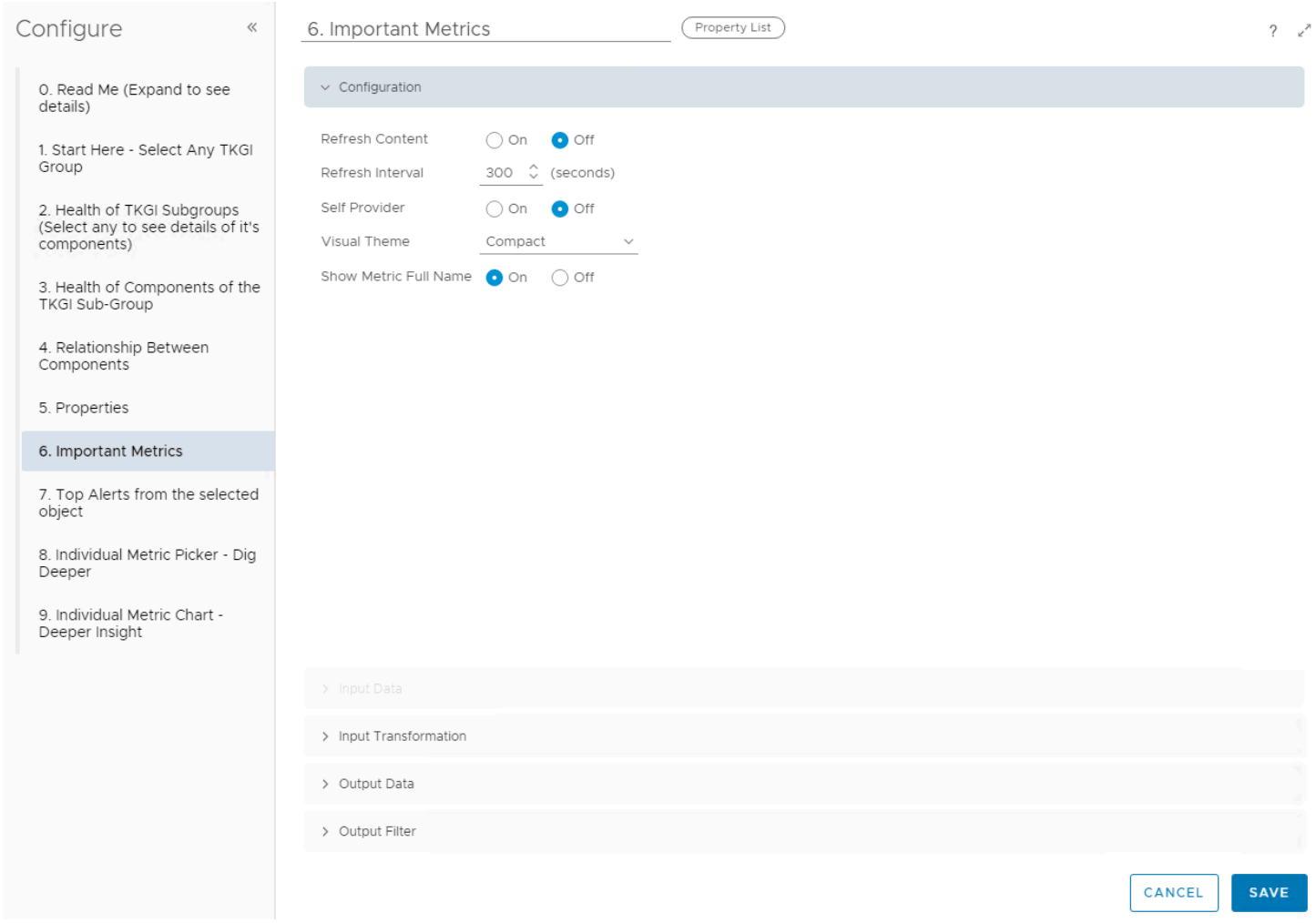

Figure 45: Important Metrics

Input Transformation → Self → Depth 1

Output Data → Select the metrics by following the process mentioned in 13.1.2. Select the following metrics as given in the screenshot.

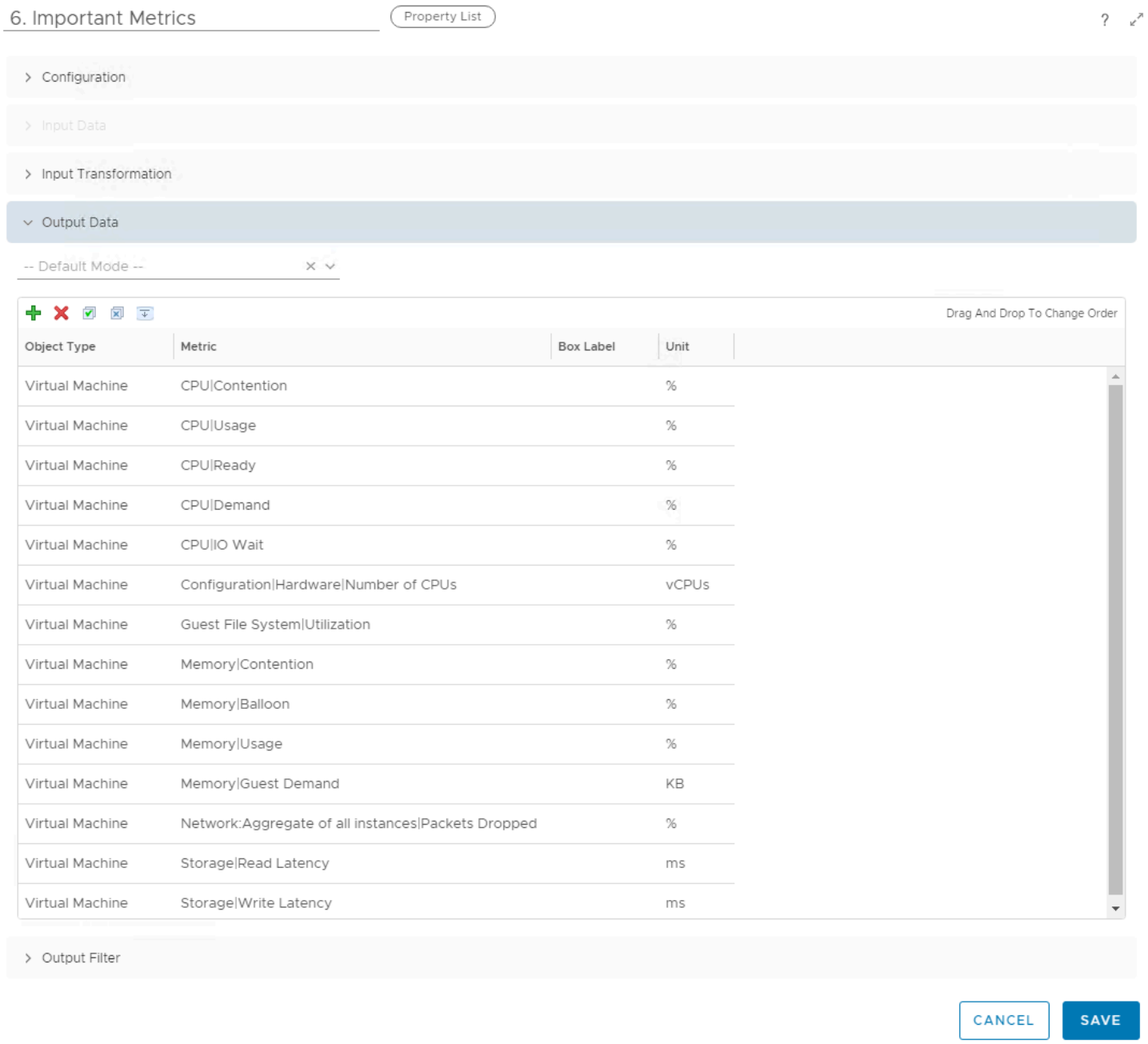

Figure 46: Property list

Click on SAVE to save the widget.

13.1.8 Top Alerts

Next, add a "*Top Alerts*" widget. Edit and name the widget "*7. Top Alerts from the selected object*". Select the parameters as following.

- Refresh Content → On
- Self-Provider → Off
- Impact Badge → Any
- Number of Alerts → 5

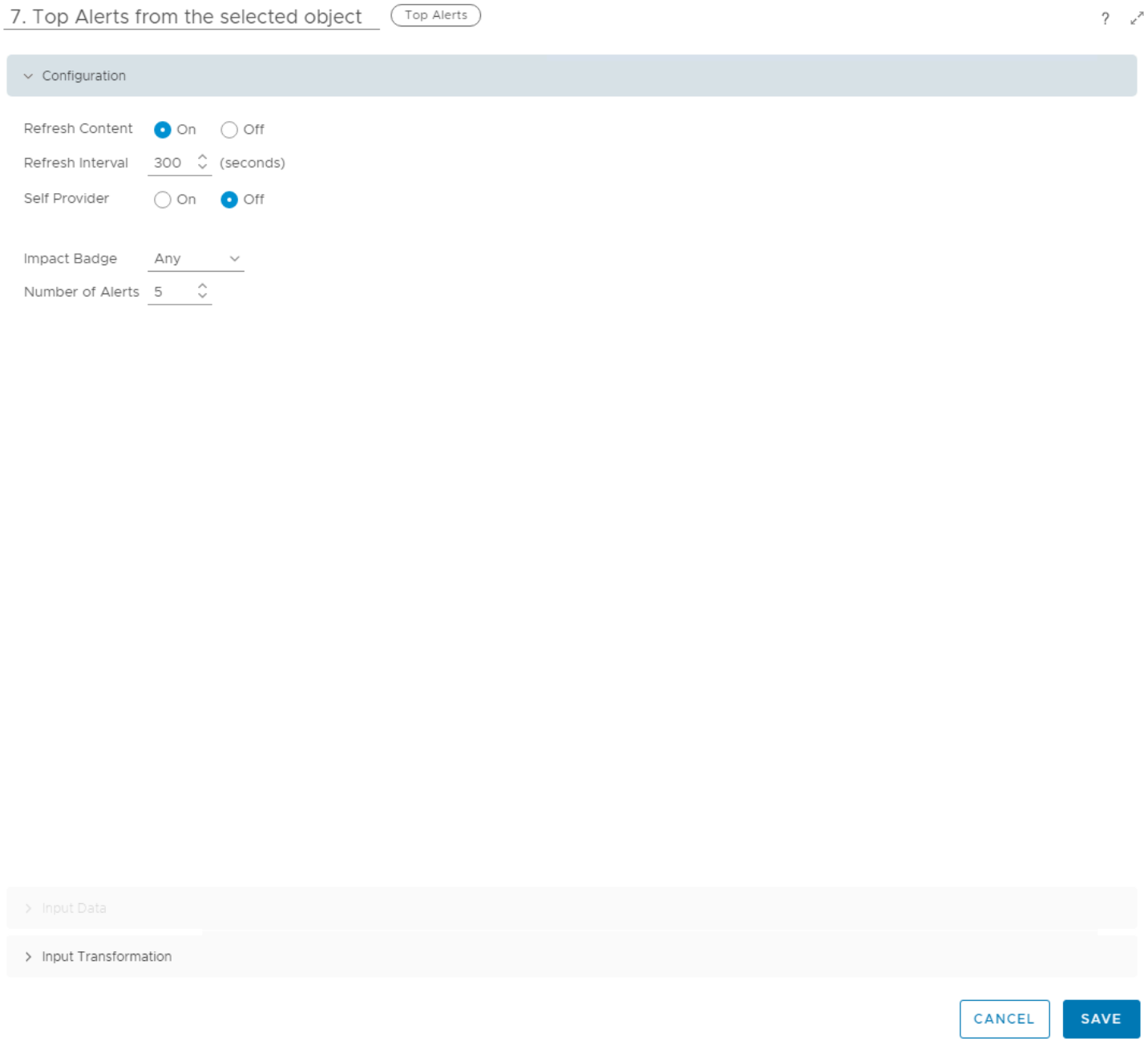

Figure 47: Top Alerts

Input Transformation → Relationship → Children → Self

Figure 48: Top alerts

Click on SAVE to save the widget.

13.1.9 Individual Metric Picker

Next, add a "*Metric Picker*" widget. Edit and name the widget "*8. Individual Metric Picker - Dig Deeper*". Select the parameters as following.

- Refresh Content → Off

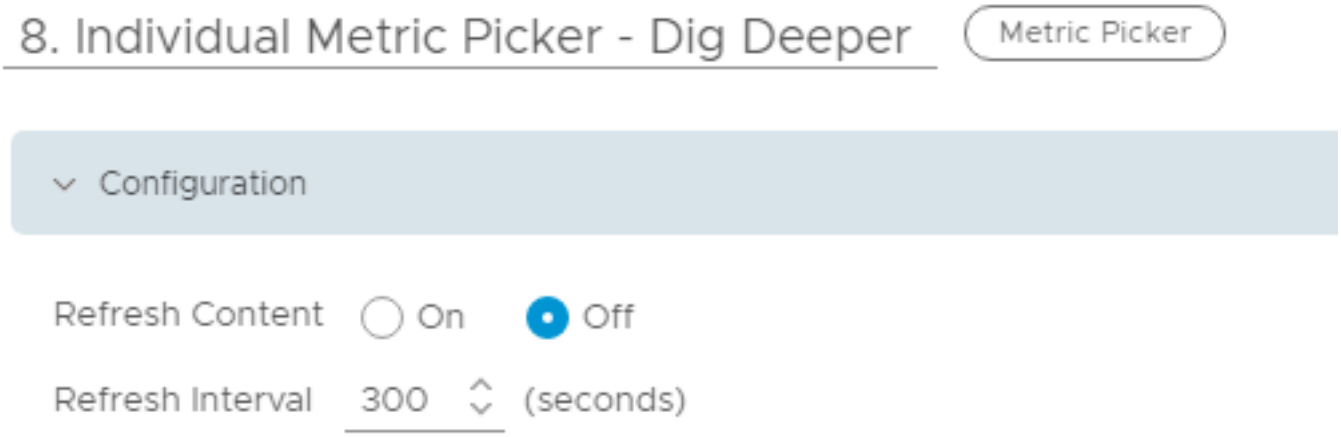

Figure 49: Metric Picker

Click on SAVE to save the widget.

13.1.10 Individual Metric Picker

Next, add a "*Metric Chart*" widget. Edit and name the widget "*9. Individual Metric Chart - Deeper Insight*". Select the parameters as following.

- Refresh Content → Off
- Self Provider → Off

Figure 50: Metric Chart

Input Transformation → Self

Click on SAVE to save the widget.

13.1.11 Interactions

The last step is to set the interactions. Click on the Show interactions tab to show the interactions view.

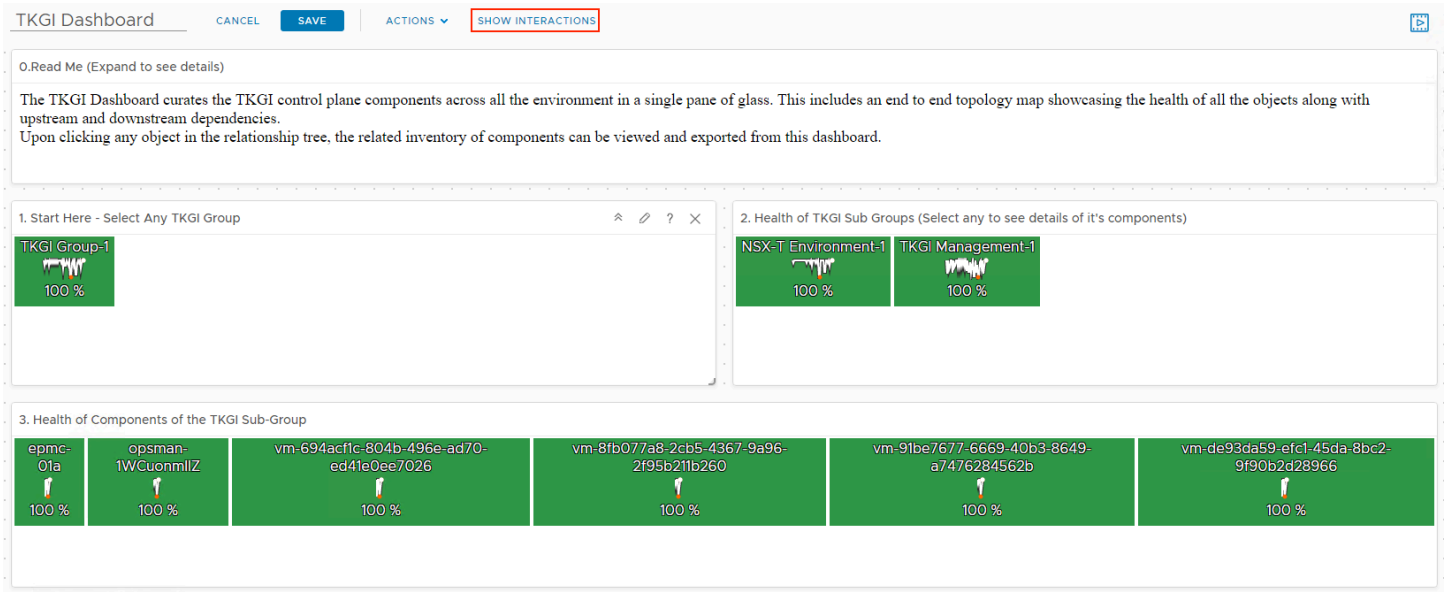

Figure 51: Show interactions

Set the interactions as shown in the below picture.

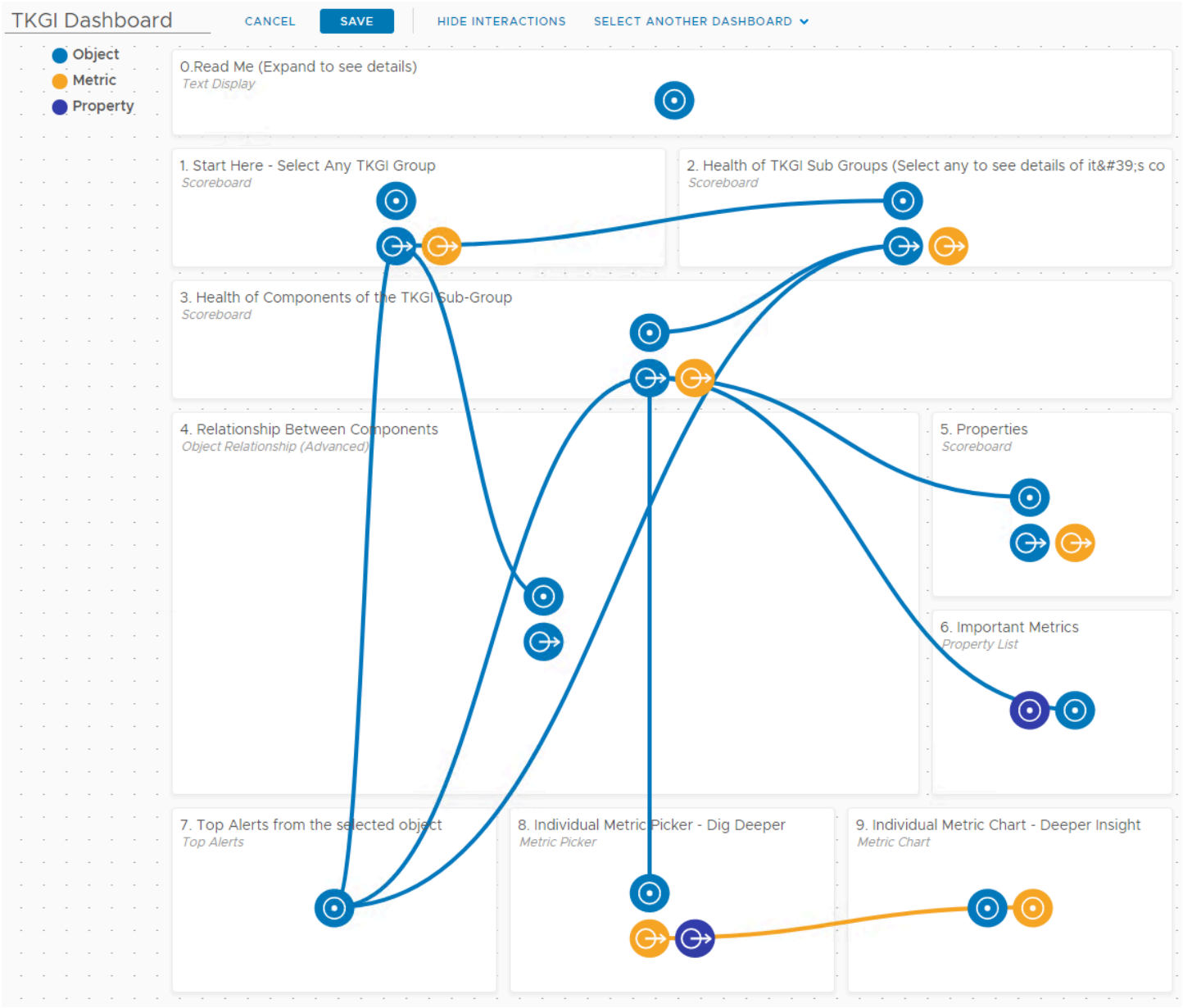

Figure 52: Interactions

Widget 0 → 0. Read Me →Stand alone
Widget 1 → Stand alone
Widget 2 depends on → Widget 1
Widget 3 depends on → widget 2
Widget 4 depends on → Widget 1
Widget 5 depends on → Widget 3
Widget 6 depends on → Widget 3
Widget 7 depends on → Widget 1, Widget 2 and Widget 3

Widget 8 depends on → Widget 3
Widget 9 depends on → Widget 8 (select the Metric field)

Note: Apart from widget 9, all the other widget relations are based on Object.

Click on SAVE, and the dashboard should be ready.

13.2 Appendix B

Provided below are the steps to manually create the dashboards in vRealize Log Insight.
Here we are going to create three new dashboards.

- TKGI Dashboard – Overall
- TKGI ←→ NSX-T Dashboard
- TKGI - Audit

TKGI Dashboard – Overall
We have total 9 widgets in this dashboard. The widgets and their queries are provided below:

1. TKGI Cluster Creation Failed
    - Search by "bosh"
    - Filters
        - Text → contains → description: create deployment result: action failed
        - Text → contains → error create deployment for instance

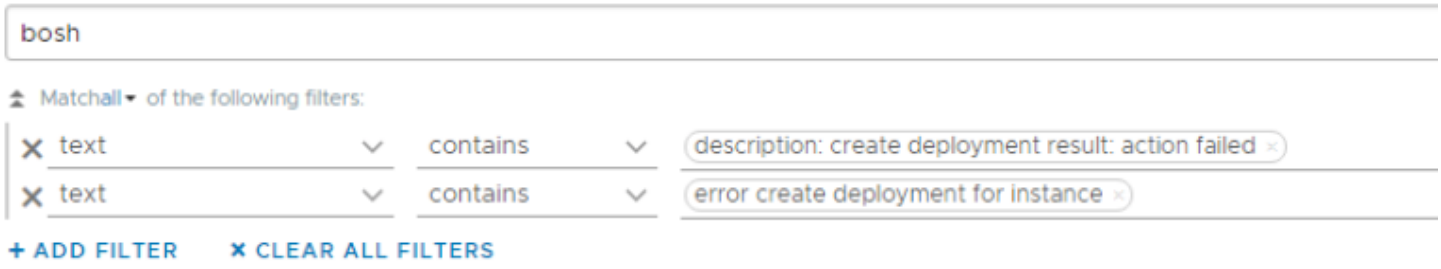

Figure 53: TKGI Cluster Creation Failed widget query

2. Error when deploying TKGI Control Plane VMs
    - Filter
        - Text → contains → Error: Unknown CPI error 'Unknown' with message 'execution expired' in 'create_stemcell' CPI method

3. User Created Event
    - Search by "UserCreatedEvent"

4. User Deleted Event
    - Search by "UserDeletedEvent"

5. Container Created
    - Search by "Started container"

6. Bosh_Service-Instance - Issues
    - Filters
        - bosh_deployment → contains → service-instance_
        - text → contains → failing unresponsive changed from running to failing

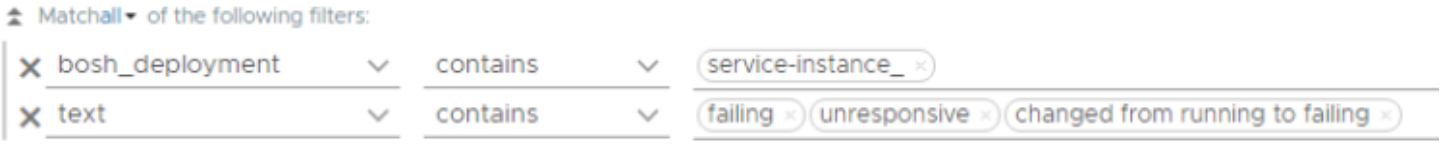

Figure 54: Bosh service instance issues query

7. TKGI: Deployment Failures - Attach Volume Issues
    - Filters
        - text → contains → FailedAttachVolume

8. Crash Loop Back Off
    - Search by "Back-off restarting failed container"

9. Image Pull Back Off
    - Search by "Error:ErrImagePull"

TKGI ←→ NSX-T Dashboard

We have total 9 widgets in this dashboard. The widgets and their queries are provided below:

1. DHCP Pool Overload NSX-T
    - Filters
        - vmw_nsxt_eventid → contains → vmwNSXDhcpPoolUsageOverloadedEvent
        - vmw_nsxt_event_state → matches regex → ^1$

2. NSX-T Password Expiration
    - Filters
        - vmw_nsxt_eventid → contains → vmwNSXPlatformPasswordExpiryStatus
        - vmw_nsxt_event_state → matches regex → ^[12]?[0-9]|-\d+$

3. NSX-T Certificate Expiration
    - Filters
        - vmw_nsxt_eventid → contains → vmwNSXPlatformCertificateExpiryStatus
        - vmw_nsxt_event_state → matches regex → ^[12]?[0-9]|-\d+$

4. NSX-T DNS Forwarder Status
    - Filters
        - vmw_nsxt_eventid → contains → vmwNSXDnsForwaderStatus
        - vmw_nsxt_event_state → matches regex → ^[12]?[0-9]|-\d+$

5. NSX-T BGP Neighbor Status
    - Filters
        - vmw_nsxt_eventid → contains → vmwNSXRoutingBgpNeighborStatus
        - vmw_nsxt_event_state → matches regex → ^0$

6. NSX-T DHCP pool lease allocation failed/succeeded
    - Filters
        - vmw_nsxt_eventid → contains → vmwNSXDhcpPoolUsageOveroadedEvent
        - vmw_nsxt_event_state → matches regex → ^1$

7. NSX-T Fabric Crypto Status
    - Filters
        - vmw_nsxt_eventid → contains → vmwNSXFabricCryptoStatus
        - vmw_nsxt_event_state → matches regex → ^[2-9]|\d{2,}$

8. NSX-T Pool MGMT
    - Filters
        - Text → contains → pool-mgmt
        - Text → contains → poolusage
        - Text → contains → nsxmgr*
        - Text → contains → 100 ips out of 239 total ips

9. BFD Tunnel Status
    - Filters
        - vmw_nsxt_eventid → contains → vmwNSXBfdTunnelStatus
        - vmw_nsxt_event_state → matches regex → ^[3-9]|\d{2,}$

TKGI - Audit

We have total 9 widgets in this dashboard. The widgets and their queries are provided below:

1. Successful Authentication
    - Search by “UserAuthenticationSuccess”

2. Unsuccessful Authentication
    - Search by “UserAuthenticationFailure”

3. TKGI Admin Logins
    - Filters
        - Text → contains → pks.cluster.admin pks-admin pks_cli

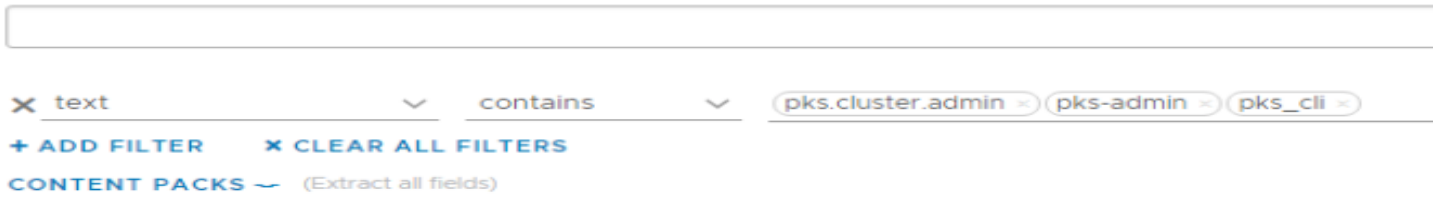

Figure 55: TKGI Admin login query

4. Successful Cluster Credential Retrieval
    - Search by “ClientAuthenticationSuccess”

5. Cluster Creation
    - Search by “Action 'create-cluster'”

6. Cluster Deletion
    - Search by “delete deployment for instance”

7. Telemetry Collection
    - Search by “telemetry-server”

8. User Deletion
    - Search by “UserDeletedEvent”

9. User Creation
    - Search by “UserCreatedEvent”

14 Glossary

VMware Acronyms	Detailed Explanation
Acronyms	Explanation
TKGI	Tanzu Kubernetes Grid Integrated
TKGO	Tanzu Kubernetes Grid Observability
vROps	vRealize Operations Manager
vRNI	vRealize Network Insight
vRLI	vRealize Log Insight

VMware, Inc. 3401 Hillview Avenue Palo Alto CA 94304 USA Tel 877-486-9273 Fax 650-427-5001 www.vmware.com.