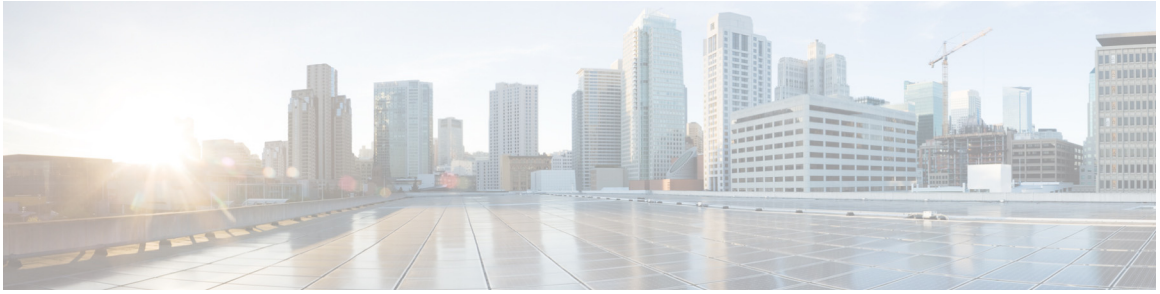
Cisco Media Transformer 1.0 User Guide

February 1, 2018

Cisco Systems, Inc.
www.cisco.com

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco website at www.cisco.com/go/offices.



Preface

The following User Guide provides some background theory on the Cisco's Media Transformer (CMT) solution. It also provides a tour of the reference Grafana dashboards for monitoring system metrics.

New and Changed Information

Given that this is a new product release, all information within this document is also new. Return to this section in future releases to determine what has changed.

Audience

This guide is intended for use by administrators responsible for using and monitoring the CMT solution and related software components. We expect that the reader will already be familiar with Linux, OpenShift, Kubernetes, Docker, and containerized software in general. Additionally, an understanding of VOD, OTT, and Legacy TV network infrastructure is beneficial, though, in places, we will review relevant concepts within this guide.

Document Organization

This document contains the following chapters and appendices:

Chapter or Appendix	Description
Cisco Media Transformer Overview	Introduces the theory behind CMT along with key terminology and concepts.
Dashboards	This chapter covers the procedures for configuring Grafana for use as well as configuring the available reference dashboards for use. Lastly, the reader is given a tour of the dashboards, showing the breadth of the metrics that are being presented.
Log Messages	This appendix provides details on Log Messages that can be generated by the Cisco Media Transformer.
Alert Manager	This appendix provides information on software that manages alerts for CMT.

Document Conventions

This document uses the following conventions:

Convention	Indication
bold font	Commands and keywords and user-entered text appear in **bold** font.
italic font	Document titles, new or emphasized terms, and arguments for which you supply values are in *italic* font.
[ ]	Elements in square brackets are optional.
{x | y | z }	Required alternative keywords are grouped in braces and separated by vertical bars.
[x | y | z]	Optional alternative keywords are grouped in brackets and separated by vertical bars.
string	A non-quoted set of characters. Do not use quotation marks around the string or the string will include the quotation marks.
`courier` font	Terminal sessions and information the system displays appear in `courier` font.
< >	Non-printing characters such as passwords are in angle brackets.
[ ]	Default responses to system prompts are in square brackets.
!, #	An exclamation point (!) or a pound sign (#) at the beginning of a line of code indicates a comment line.

Note Means *reader take note*. Notes contain helpful suggestions or references to material not covered in the manual.

Tip Means *the following information will help you solve a problem*. The tips information might not be troubleshooting or even an action, but could be useful information, similar to a Timesaver.

Caution Means *reader be careful*. In this situation, you might perform an action that could result in equipment damage or loss of data.

Timesaver Means *the described action saves time*. You can save time by performing the action described in the paragraph.

Warning **IMPORTANT SAFETY INSTRUCTIONS**

This warning symbol means danger. You are in a situation that could cause bodily injury. Before you work on any equipment, be aware of the hazards involved with electrical circuitry and be familiar with standard practices for preventing accidents. Use the statement number provided at the end of each warning to locate its translation in the translated safety warnings that accompanied this device.

SAVE THESE INSTRUCTIONS

Warning **Statements using this symbol are provided for additional information and to comply with regulatory and customer requirements.**

Related Publications

Refer to the following documents for additional information about CMT 1.0:

- *Cisco Media Transformer 1.0 Release Notes*
- *Cisco Media Transformer 1.0 Installation Guide*
- *Open Source used in Cisco Media Transformer 1.0*

Cisco Media Transformer Overview

This chapter includes the following topics to introduce you to the Cisco Media Transformer (CMT) solution:

Product Overview

Cisco's Media Transformer (CMT) is a part of the OMD (Open Media Distribution) Suite of products. The CMT solution provides fill-agent functionality to VDS-TV VoD streamers and transforms MPEG DASH TS (segmented-ABR) content to MPEG-2 TS-compliant streams, which allows playback of ABR content on legacy set-top boxes that require CBR input. This approach effectively allows Service Providers to fully leverage their existing QAM-based set-top box infrastructure, while giving them a path to transition to IP-based set top boxes over a longer timeframe.

Note During the development stages, Cisco Media Transformer has undergone a name change from ABR2TS. That older acronym may still appear in configuration files, console output, and other locations. Additionally, the product is occasionally referred to as the more generic "VoD Gateway" that describes its overall functionality. For all intents and purposes, please consider ABR2TS, VoD Gateway, and Media Transformer the same product.

Containerized Deployment

The CMT solution is deployed in a clustered environment utilizing the OpenShift Container Platform for node and container management. The solution consists of a set of microservices that run in Docker containers. These containers are deployed to the cluster nodes and managed via the Kubernetes

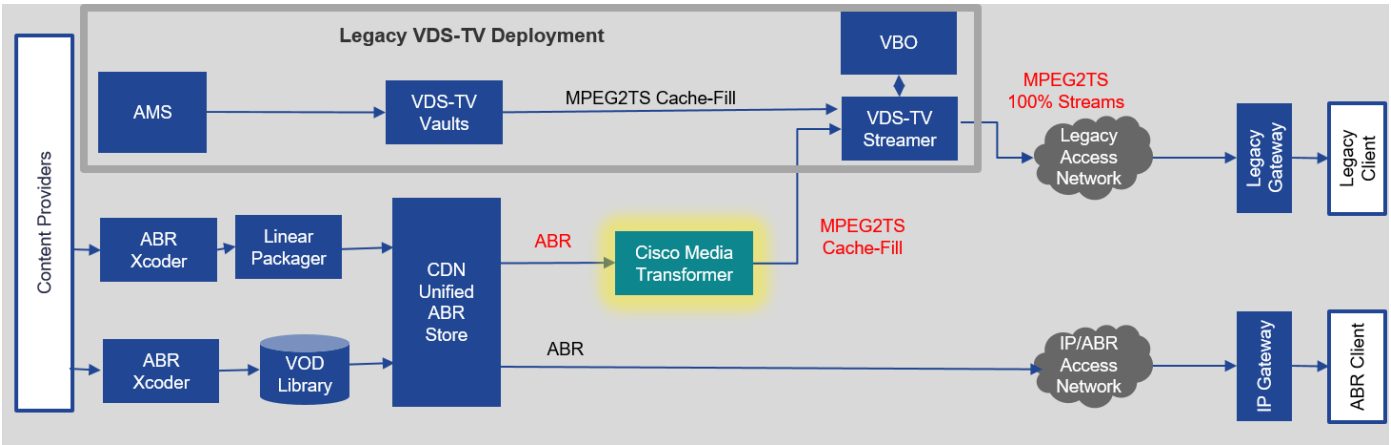
orchestration layer of the OpenShift platform. This approach leverages the benefits and flexibility of container technology, such as high availability, auto-recovery, horizontal scalability, and ease of deployment.

Functional Overview

With respect to Cisco Media Transformer, the process starts in the following manner:

- The user of a set-top box requests specific content to the Video Back Office (VBO)
- VBO communicates with a master streamer, which selects the appropriate streamer that will serve the request. If the requested content is not cached on any of the streamers, then content will need to be pulled from the vaults, otherwise it will be served directly from the streamers.
- The system sees that the content is located at a URL and is not traditional VoD content. A conversion will need to take place.
- The system will pass the content URL, CBR bitrate, and starting/ending offsets to Media Transformer. The Media Transformer then fetches the manifest file from the CDN.
- The manifest provides a few key pieces of information to Media Transformer: representations, segment timeline, and segment location. Using the information in the manifest file along with the information provided in the request, Media Transformer is able to determine what segments need to be fetched from the CDN.
- The appropriate MPEG DASH segments are fetched, and transformed in real-time to be an MPEG-2 TS-compliant CBR stream, and delivered at a specified rate to the requesting system.
- The VDS-TV system will cache the CBR stream while delivering it to the QAM-based STB.

Figure 1-1 CMT Functional Overview



Worker Node Deployment

Worker nodes enable the core ABR to CBR conversion functionality within Media Transformer. ABR to CBR content transformation happens as part of the real-time streaming process. When the VDS-TV system detects that it does not have content in cache, it issues a request to Media Transformer to provide

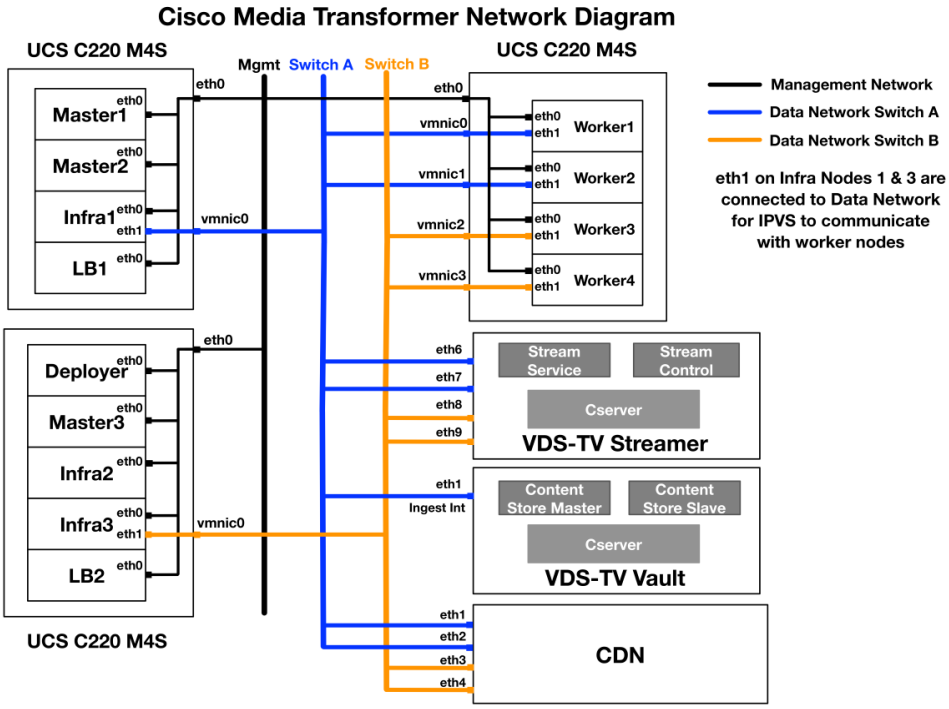
the content. This request will be directed to one of the Media Transformer pods for immediate processing. Since this is part of the real-time streaming process, the ABR content must be fetched, transformed, and delivered at a guaranteed rate specified by the VDS-TV system. A failure to deliver at rate will cause a VoD stream failure at the QAM or STB.

CMT Network Overview

Each UCS C220 M4 server is configured with four - 10GB network cards. The first two boards are connected to a Data A router, while the other two boards are connected to a Data B router. These data pathways are where the data from Media Transformer will be sent to the VDS-TV streamers. The purpose of having two data pathways is to provide high-availability functionality, so that if one router goes offline, then the other router will pick up the work and provide the required data stream.

Additionally, a 1GB network interface runs throughout the system to provide management functionality to Media Transformer - a task that requires less bandwidth than the data processing aspect. Figure 1-2 illustrates the Media Transformer network topology.

Figure 1-2 Media Transformer Network Diagram

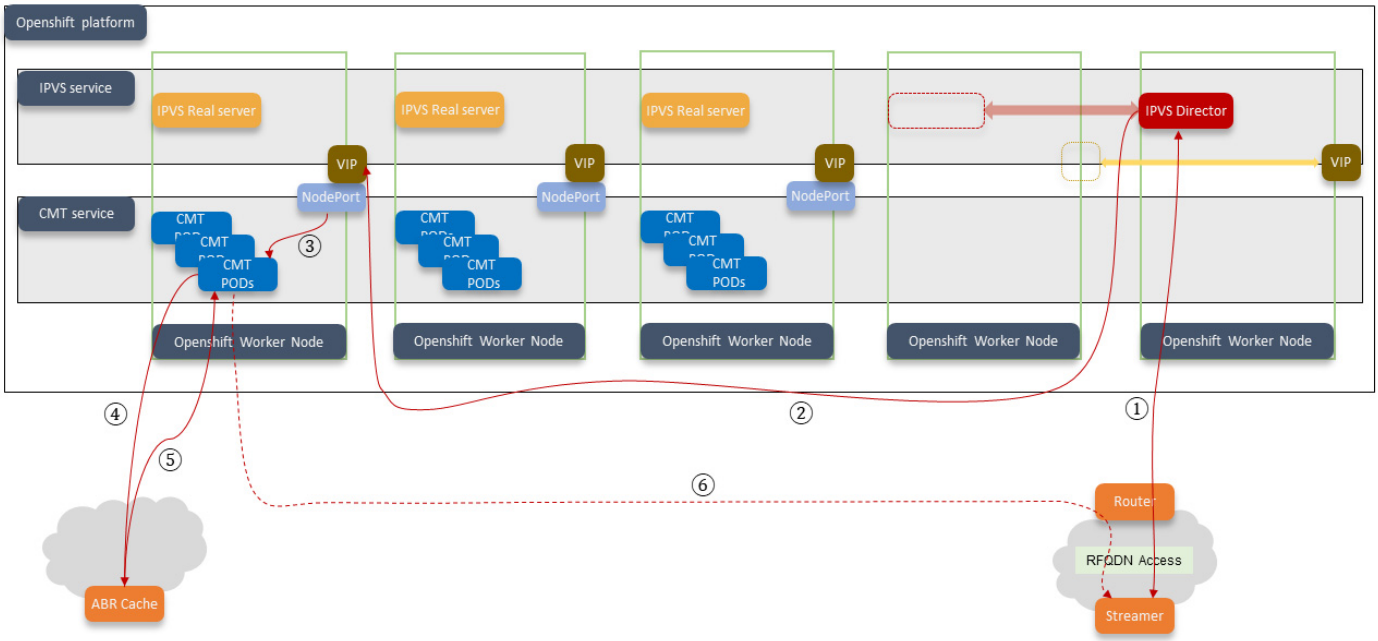


LB Request Example

All VDS-TV requests (API and client calls) to Media Transformer will first be sent to the IPVS load balancer. IPVS then redirects the calls onto different Media Transformer virtual machines (4 VMs exist per physical server). The Kubernetes instance on each virtual will then allocate the video processing load onto one of the five pod services (Docker containers) that it is managing.

After the Media Transformer pods have performed their work, they send the data back directly to the VDS-TV streamer, thereby bypassing the IPVS load balancer.

Figure 1-3 CMT Load Balance Solution



Virtual Machine Types

CMT consists of a set of virtual machines, each of which performs specific functions within the cluster and is packaged in an OVA file that encapsulates all functionality and optimal system configuration settings for each node type. An explanation of the virtual machine types and their resource requirements follows.

Master VMs

The OpenShift Master is the virtual machine that manages the entire cluster by communicating control messages to all of the cluster VM nodes. These services provide functionality related to pod management and the replication of nodes, authentication, data store, and scheduling.

Table 1-1 Master Node Virtual Machine Settings

Resource	Configuration
CPU	4 Cores
Memory	8GB
Disks	60GB disk space consisting of 2 - 30GB disks Operating System (30GB) & Docker (30GB)

Note We recommend 3 Master virtual machines within a cluster to fulfill high-availability requirements.

Deployer VMs

The OpenShift Deployer virtual machine stores the images and deployment scripts used to deploy and install all of the OpenShift images required for the initial cluster setup. This is a non-critical function for high-availability, so the cluster only needs a single Deployer node..

Table 1-2 Deployer Node Virtual Machine Settings

Resource	Configuration
CPU	4 Cores
Memory	8GB
Disks	100GB disk space consisting of 2 - 50GB disks Operating System (50GB) & Docker (50GB)

Load Balancer VMs

The Load Balancer virtual machines define a node that is used to manage the OpenShift cluster. A load balancer Virtual IP is used to access the OpenShift cluster.

Table 1-3 Load Balancer Virtual Machine Settings

Resource	Configuration
CPU	2 Cores
Memory	4GB
Disks	20GB disk space Operating System (20GB)

Note We recommend 2 Load Balancer virtual machines within the cluster to fulfill high availability requirements. They will serve Master/Slave roles.

Infrastructure VMs

The Infrastructure virtual machines define a node that contains the IPVS load balancer (for CMT use), logging queue, and other infrastructure-related services such as those providing monitoring and alert functionality. The composition of these services will evolve over time.

Table 1-4 Infra Virtual Machine Settings

Resource	Configuration
CPU	8 Cores
Memory	16GB
Disks	60GB disk space consisting of 2 - 30GB disks Operating System (30GB) & Docker (30GB)

Note A minimum of 3 infrastructure (Infra) nodes are required for a high availability system deployment.

Table 1-5 Recommended Infrastructure Service Allocation

Infrastructure VM 1	Infrastructure VM 2	Infrastructure VM 3
IPVS Director (Master)	Proxytoservice	IPVS Director (Standby)
Kafka	Kafka	Kafka
Zookeeper	Zookeeper	Zookeeper
Logstash	Logstash	Logstash

Worker VMs

OpenShift Worker virtual machines perform the primary functionality of CMT, which is to run multiple pods that convert adaptive bitrate (ABR) content to constant bitrate (CBR) content in real time with no latency or caching. As such, the CPU and memory resource requirements are considerable, relative to the rest of the system.

Table 1-6 Worker Virtual Machine Settings

Resource	Configuration
CPU	7 Cores
Memory	60GB
Disks	60GB disk space consisting of 2 - 30GB disks Operating System (30GB) & Docker (30GB)

Note Swap memory will be set to 0 (meaning physical memory only is used) and hyper-threading should be disabled. Hyper-threading introduces some scheduling challenges into the system, so we have found that a more consistent throughput is achieved when using non-virtualized cores. Configuration instructions will be provided within this guide.

System Hardware Requirements

Media Transformer runs on general-purpose computing hardware, and is optimized for the Cisco Unified Computing System (UCS) server platform. Table 1-7 lists the recommended hardware configuration for a single Media Transformer server. For more detailed hardware requirements, refer to your Bill of Materials (BOM) or contact your Cisco Systems representative.

Note The recommended configuration for a CMT deployment is a minimum of 3UCS C220 M4 servers.

Table 1-7 Media Transformer Server Recommended Hardware Configuration

Description	Quantity
UCS C220 M4 Server	1
2.6GHz E5-2690 v4 CPUs	2
32GB DDR4 RAM	8
600 GB SAS 10K RPM HDD	2
Dual-port 10Gb Network Interface Cards	2

Terms and Definitions

Table 1-8 lists terms and definitions used in describing CMT or related concepts

Table 1-8 Terms and Definitions

Term	Definition
ABR2TS	The previous name for Cisco Media Transformer. This acronym still appears in various places throughout the installation process and therefore will also appear in this guide.
ABS	Adaptive Bitrate Streaming is where video content is streamed at the maximum rate and highest quality at which the network will allow at any given moment.
CBR	Constant Bitrate is where video content is streamed at a constant rate across a network.
Docker	A service used by Kubernetes to deploy containerized applications, such as the CMT solution.
IPVS	Linux IP Virtual Servers run on a host and act as a load balancer in front of a cluster of servers.

Table 1-8 Terms and Definitions

Term	Definition
Kubernetes	Management system for containerized applications deployed across a cluster of nodes.
Load Balancer Node	Two types of load balancers exist within the Media Transformer solution: 1) An IPVS Load Balancer directs external VDS-TV requests to different CMT virtual machines. 2) A Kubernetes instance on each virtual machine allocates the video processing load onto one of five Worker pods that it manages.
OMD Suite	Open Media Distribution Suite OMD is a suite of products for Service Providers to efficiently distribute and cache multi-screen video to managed & un-managed devices on managed & un-managed networks. Cisco Media Transformer is a part of OMD Suite.
POD	Are Docker containers that run microservices and that, in the Media Transformer solution, are managed by Kubernetes.
VDS-TV	The streamer component to which Media Transformer streams
Video BackOffice	Video BackOffice is a solution that provides a managed video control plane to service providers.

Grafana Dashboards

The Grafana dashboards present a customizable reference user interface that provides system insights via detailed metrics for the CMT solution.

This chapter will provide a tour of the CMT dashboards, and includes:

Monitoring Stack Overview

The CMT monitoring stack consists of a Prometheus backend coupled with a Grafana user interface, AlertManager, and the Heapster cluster monitoring tool.

Prometheus is used to collect various metrics, such as network, memory, and CPU utilization, from the CMT cluster by scraping information from the endpoints. That information is stored locally so that rules can be run against it, or the data can be aggregated, if necessary.

Granafa provides a customizable dashboard user interface to view the node and cluster metrics collected by Prometheus. The focus of this chapter will be to provide a tour of the reference dashboards that we have made available within Grafana.

Tip For details on using, creating, or customizing dashboards, visit *http://docs.grafana.org/*

Configuring Grafana

The following section details the steps that are required to configure the Grafana interface for use with the reference dashboards provided by Cisco.

Step 1 Edit the `/etc/hosts` file on the machine from which you will be accessing the Grafana user interface. Add the Grafana hostname to the load balancer VIP IP. For example:

```
##
# Host Database
#
# localhost is used to configure the loopback interface
# when the system is booting.  Do not change this entry.
##
127.0.0.1localhost
255.255.255.255broadcasthost
::1             localhost

172.22.102.170 grafana.cmt.cisco.com  ' Load Balancer VIP IP
172.22.102.170 alertmanager.cmt.cisco.com  ' Load Balancer VIP IP
172.22.102.170 prometheus.cmt.cisco.com  ' Load Balancer VIP IP
```

Step 2 Using the previous hostname setting as an example, log into the Grafana interface on port 3000. The credentials are `username: admin` / `password: admin`.

```
http://grafana.cmt.cisco.com:3000
```

Step 3 Navigate to **Add data source**.

Step 4 Enter the following values onto the data source configuration page.

Table 2-1 Add/Edit Data Source

Field	Value
Name	abr2ts
Type	Prometheus
URL	http://{master_node_IP}:9090

Importing Grafana Dashboards

The following procedures will import the Grafana Dashboards, allowing you to monitor metrics for the Kubernetes cluster and for the Worker nodes.

Step 1 Copy the `Media-Transformer-Workers-Dashboard.json` and `Media-Transformer-Cluster-Monitoring.json` files to the localhost from where you will be opening the Grafana user interface. These files will need to be importing these file to create the Grafana dashboards. The json files are located on the Deployer node at: `/root/abr2ts-deployment/platform/resources/config/grafana`

Note Whenever you restart the Monitoring Stack, you will need to re-import the `Media-Transformer-Workers-Dashboard.json` and `Media-Transformer-Cluster-Monitoring.json` files in order to view the dashboards again.

Step 2 Navigate to **Dashboards > Import**.

Step 3 Import the `Media-Transformer-Workers-Dashboard.json`.

Step 4 Select "`abr2ts`" as the Prometheus data source.

Step 5 Verify that the dashboard shows all of the CMT pod data, such as: transmit/receive/memory/CPU usage.

Step 6 Navigate to **Dashboards > Import** once again.

Step 7 Import the `Media-Transformer-Cluster-Monitoring.json` file to the dashboard.

Step 8 Select "`abr2ts`" as the Prometheus data source.

Step 9 Verify that the dashboard shows cluster node metrics, such as network I/O, memory, CPU, and filesystem usage.

Dashboard Tour

The following section will give a tour of the reference Grafana dashboards provided by Cisco. The tour is divided into two major parts:

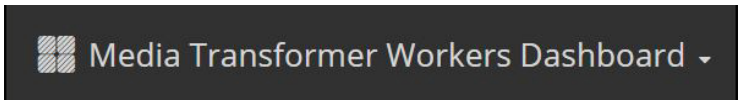
1. **Workers Dashboards** - dashboards providing raw metrics for Media Transformer Worker nodes.
2. **Cluster Dashboards** - dashboards providing raw metrics for the Kubernetes cluster.

Tip To properly monitor Cisco Media Transformer, we recommend that all system operators acquaint themselves with the range of metrics presented by the different Grafana dashboards.

Media Transformer Workers Dashboards

From the Grafana home page, click the **Home > Media Transformer Workers Dashboard** near at the upper left of the interface. This will bring up a page showing all of the Worker Dashboards.

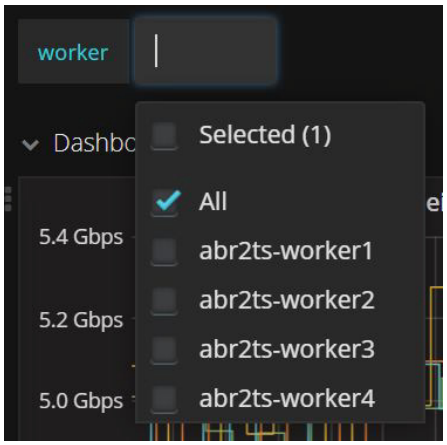
Figure 2-1 CMT Worker Dashboard Button



Worker Node Selection

The **worker** drop-down option allows you to select the set of Worker nodes for which you would like to view metrics. You can select all Worker nodes within cluster, or select on or more individual nodes as required.

Figure 2-2 CMT Worker Node Selection

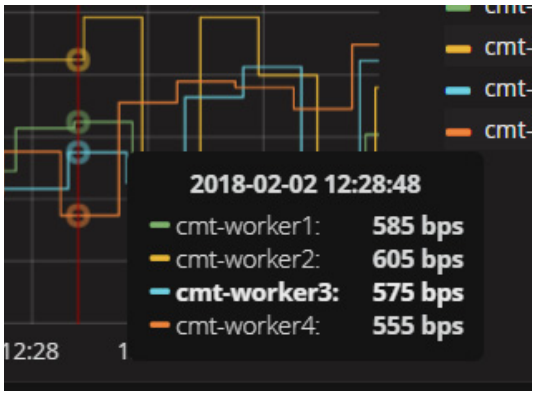


Tip As you navigate around the interface, you will notice that the following dashboards are available. The dashboards are clearly labeled at the top, so their purpose should be self-evident.

Viewing Graph Details

As you view the various graphs, you can move your mouse over the lines to see a pop-up listing of specific data points. Figure 2-3 shows dots highlighting specific point in time. The focus is on cmt-worker3 node, with data received at 575 bps. Moving the mouse to a different point in the graph will change the focus onto other nodes. The functionality here applies to all similar graphs shown in the Grafana dashboards.

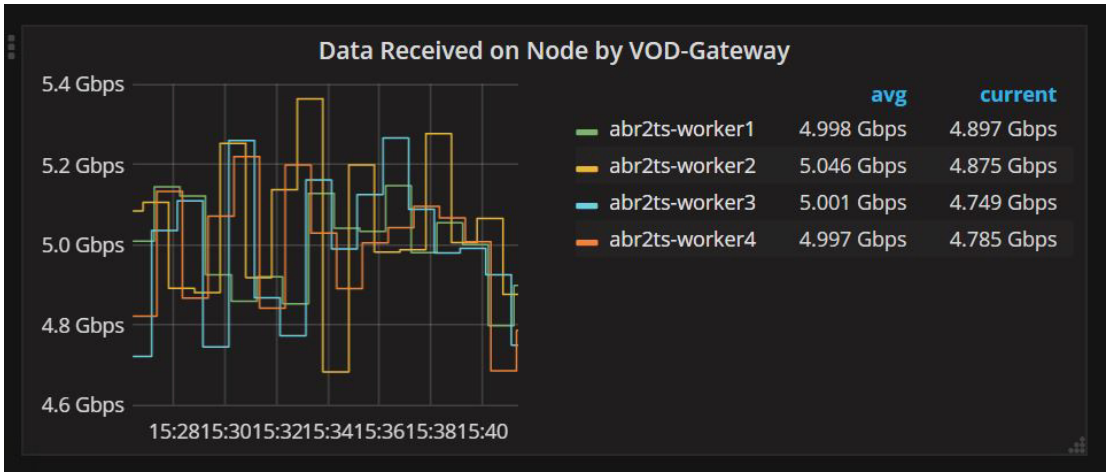
Figure 2-3 Viewing Graph Details



Data Received on Node by VOD-Gateway

The **Data Received on Node by VOD-Gateway** dashboard (Figure 2-4) shows graphical and numeric representations of the Average and Current Gbps flow of traffic being received by the Worker nodes.

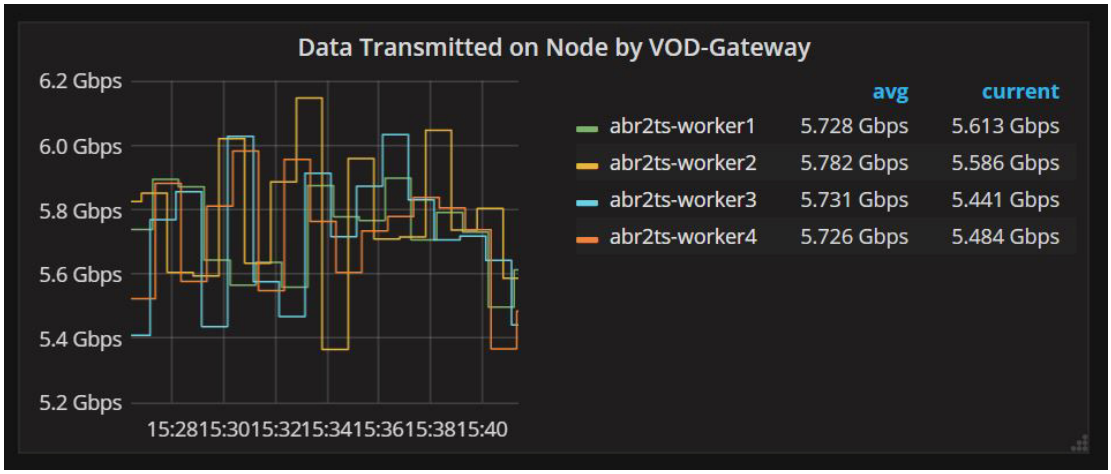
Figure 2-4 Figure CMT Worker Data Received by Node



Data Transmitted on Node by VOD-Gateway

The **Data Transmitted on Node by VOD-Gateway** dashboard (Figure 2-5) shows graphical and numeric representations of the Average and Current Gbps flow of traffic being transmitted by the Worker nodes.

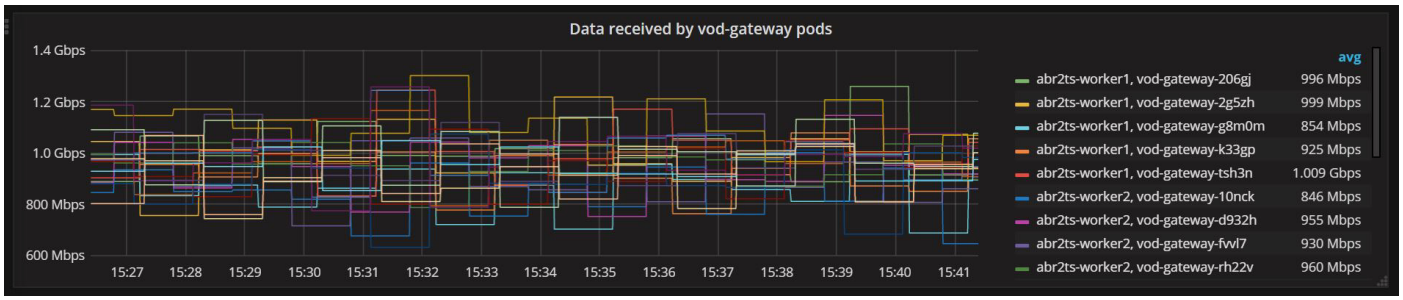
Figure 2-5 CMT Worker Data Transmitted by Node



Data Received VOD-Gateway Pods

The **Data Received VOD-Gateway Pods** dashboard (Figure 2-6) shows graphical and numeric representations of the Average and Current Mbps flows of traffic being received by the Worker pods. Like with other dashboards, time is shown on the x-axis, while throughput is shown on the y-axis.

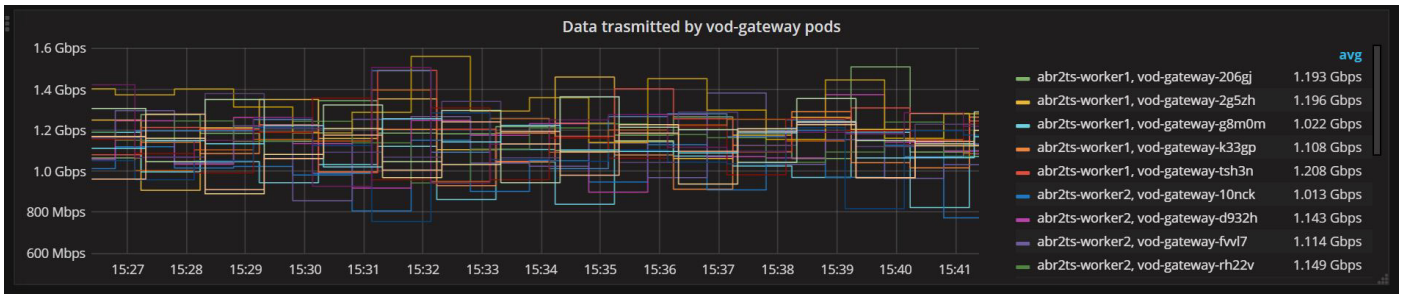
Figure 2-6 CMT Worker Data Received by VoD Gateway



Data Transmitted by VOD-Gateway Pods

The **Data Transmitted by VOD-Gateway Pods** dashboard (Figure 2-7) shows graphical and numeric representations of the Average and Current Gbps flow of traffic being transmitted by the Worker pods. Like with other dashboards, time is shown on the x-axis, while throughput is shown on the y-axis.

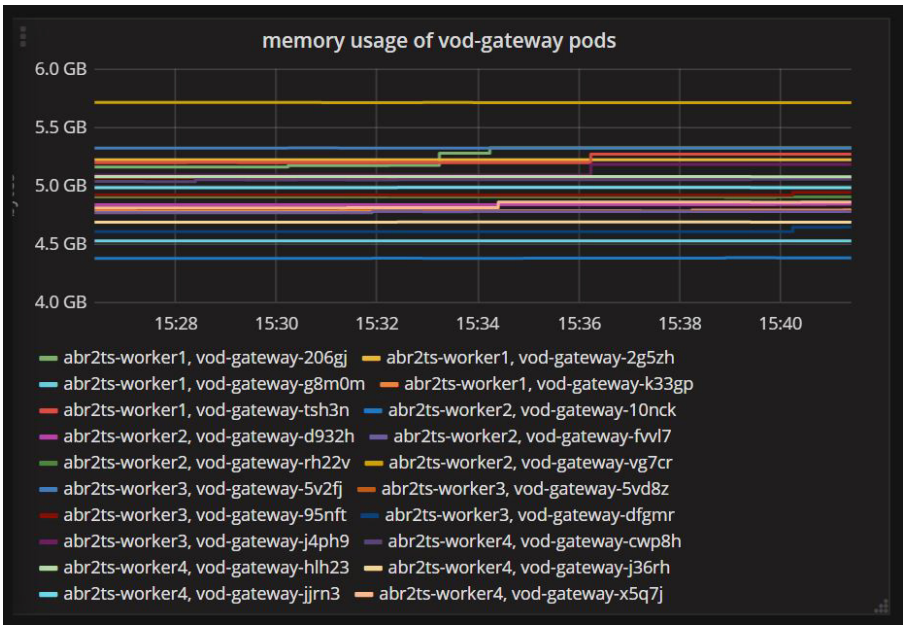
Figure 2-7 CMT Worker Data Transmitted by VoD Gateway Pods



Memory Usage of VOD-Gateway Pods

The **Memory Usage of VOD-Gateway Pods** dashboard (Figure 2-8) shows a graphical representation of the ongoing memory use of the Worker pods. Once again, time is shown on the x-axis, with GB throughput shown on the y-axis.

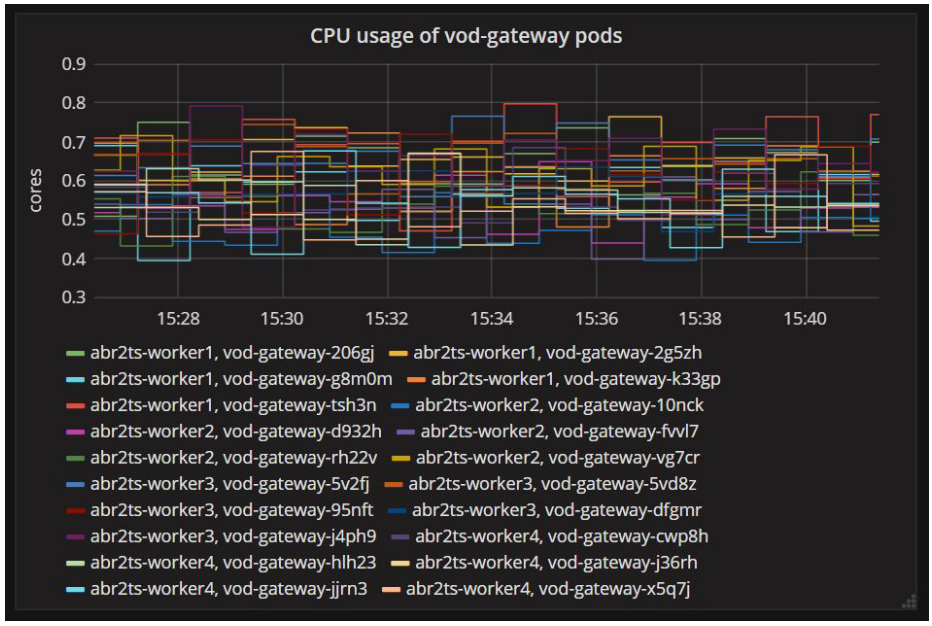
Figure 2-8 CMT Memory Usage of VoD Gateway Pods



CPU Usage of VOD-Gateway Pods

The **CPU Usage of VOD-Gateway Pods** dashboard (Figure 2-9) shows a graphical representation of the ongoing CPU usage of the Worker pods. Time is shown on the x-axis, with CPU cores shown on the y-axis.

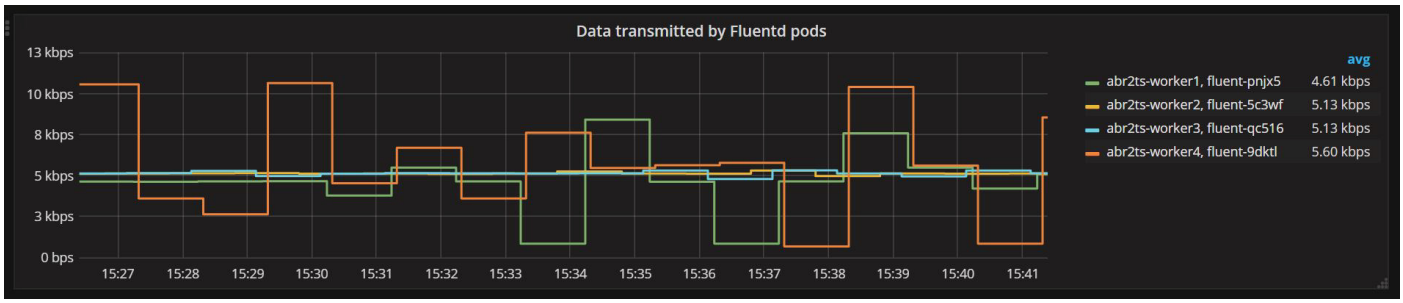
Figure 2-9 CPU Usage of VoD Gateway Pods



Data Transmitted by Fluentd Pods

The **Data Transmitted by Fluentd Pods** dashboard (Figure 2-10) shows graphical and numeric representations of the ongoing data in kbps that is being transmitted by the Fluentd logging pods.

Figure 2-10 CMT Worker Data Transmited by Fluentd Pods



Media Transformer Cluster Monitoring Dashboard

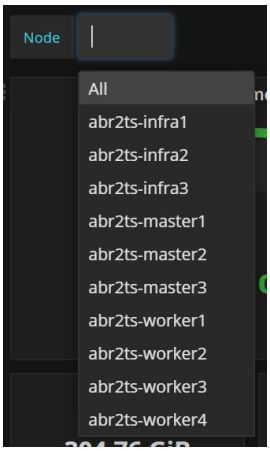
From the Grafana home page, click the **Home > Media Transformer Cluster Monitoring** near at the upper left of the interface. This will bring up a page showing all of the Cluster Monitoring Dashboards.

Figure 2-11 CMT Cluster Monitoring Icon

Node Selection

The **node** drop-down option allows you to select the nodes within the cluster whose metrics you would like to appear on the dashboard. You can select all nodes within the cluster, or individual nodes (only), as required.

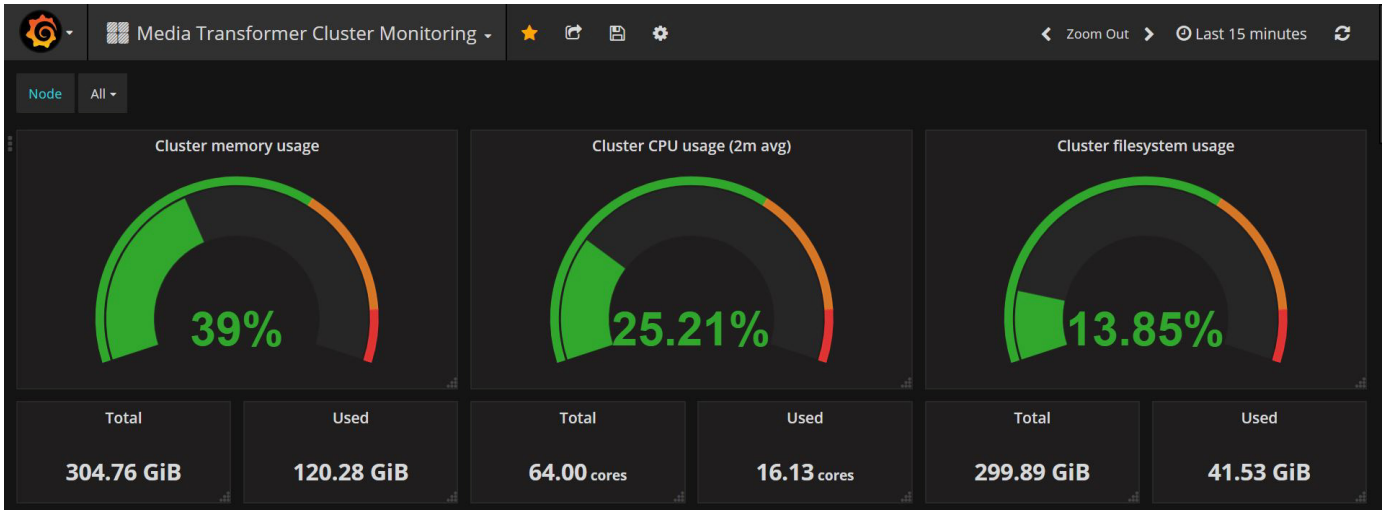
Figure 2-12 CMT Cluster Monitoring - Node Selection



Odometer Style Dashboards

The Media Transformer Cluster Monitoring collection of dashboards include some odometer-style widgets for display of some of the metrics (see Figure 2-13). Dashboard components can be moved around and sized, if desired. Figure 2-13 shows the default layout of the odometer-style dashboard components.

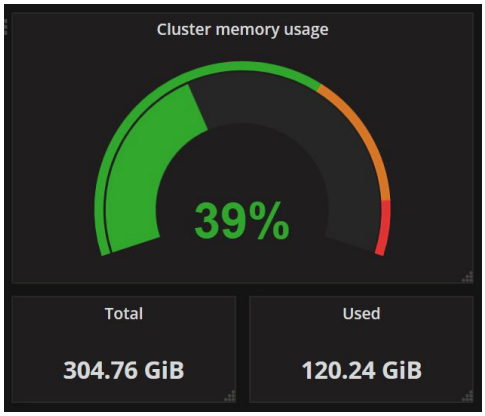
Figure 2-13 Cluster Memory/CPU/Filesystem Usage



Cluster Memory Usage

The **Cluster Memory Usage** dashboard (Figure 2-14) shows a graphical representation of the ongoing cluster memory usage in GiB (gibibytes). Total and used memory is summarized at the bottom of the dashboard.

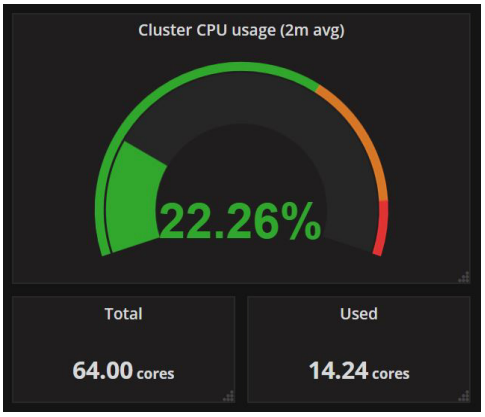
Figure 2-14 CMT Cluster Monitoring - Memory



Cluster CPU Usage

The **Cluster CPU Usage** dashboard (Figure 2-15) shows a graphical representation of the ongoing total cluster CPU usage in cores. A total and used cores summary is shown at the bottom of the dashboard.

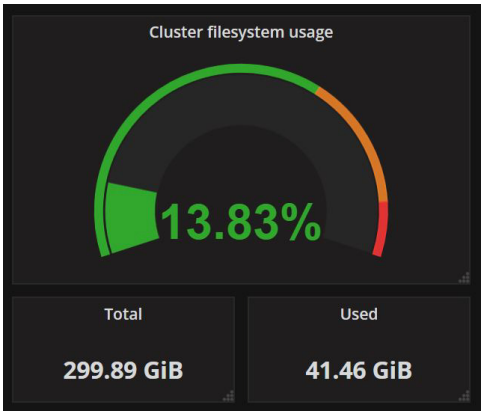
Figure 2-15 CMT Cluster Monitoring - CPU



Cluster Filesystem Usage

The **Cluster Filesystem Usage** dashboard (Figure 2-16) shows a graphical representation of the ongoing total file usage within the cluster. A total and used filesystem summary is shown at the bottom of the dashboard.

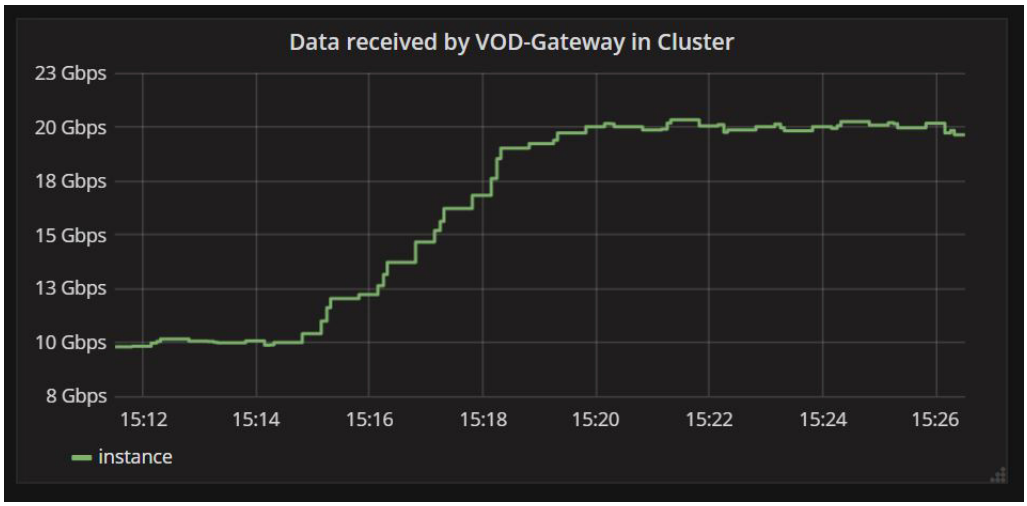
Figure 2-16 CMT Cluster Monitoring - Filesystem Usage



Data Received by VoD Gateway in Cluster

The **Data Received by VOD-Gateway in Cluster** dashboard (Figure 2-17) shows a graphical representation of the data received by the selected VoD Gateway nodes in the cluster. Time is shown on the x-axis, with Gbps throughput shown on the y-axis.

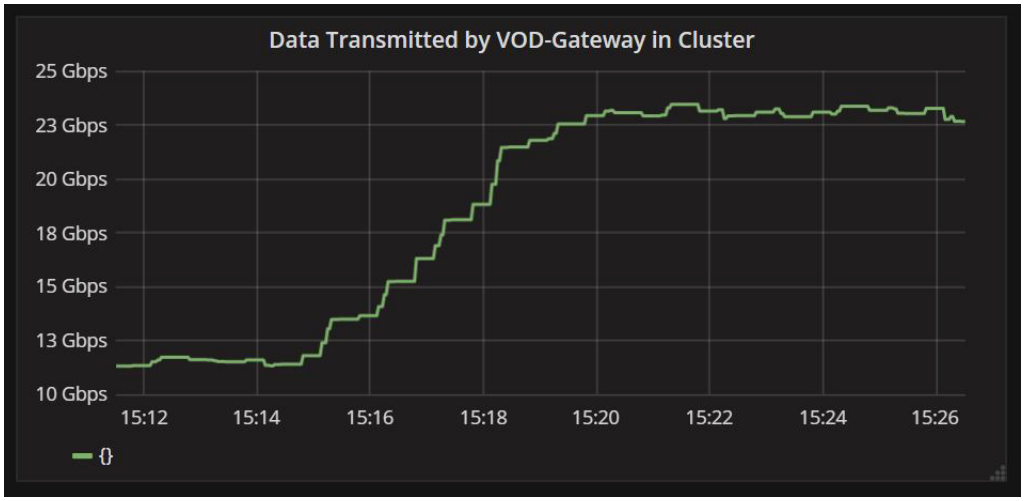
Figure 2-17 CMT Cluster Monitoring - Data Received by VoD Gateway



Data Transmitted by VoD Gateway in Cluster

The **Data Transmitted by VOD-Gateway in Cluster** dashboard (Figure 2-18) shows a graphical representation of the data transmitted by the selected VoD Gateway nodes in the cluster. Time is shown on the x-axis, with Gbps throughput shown on the y-axis.

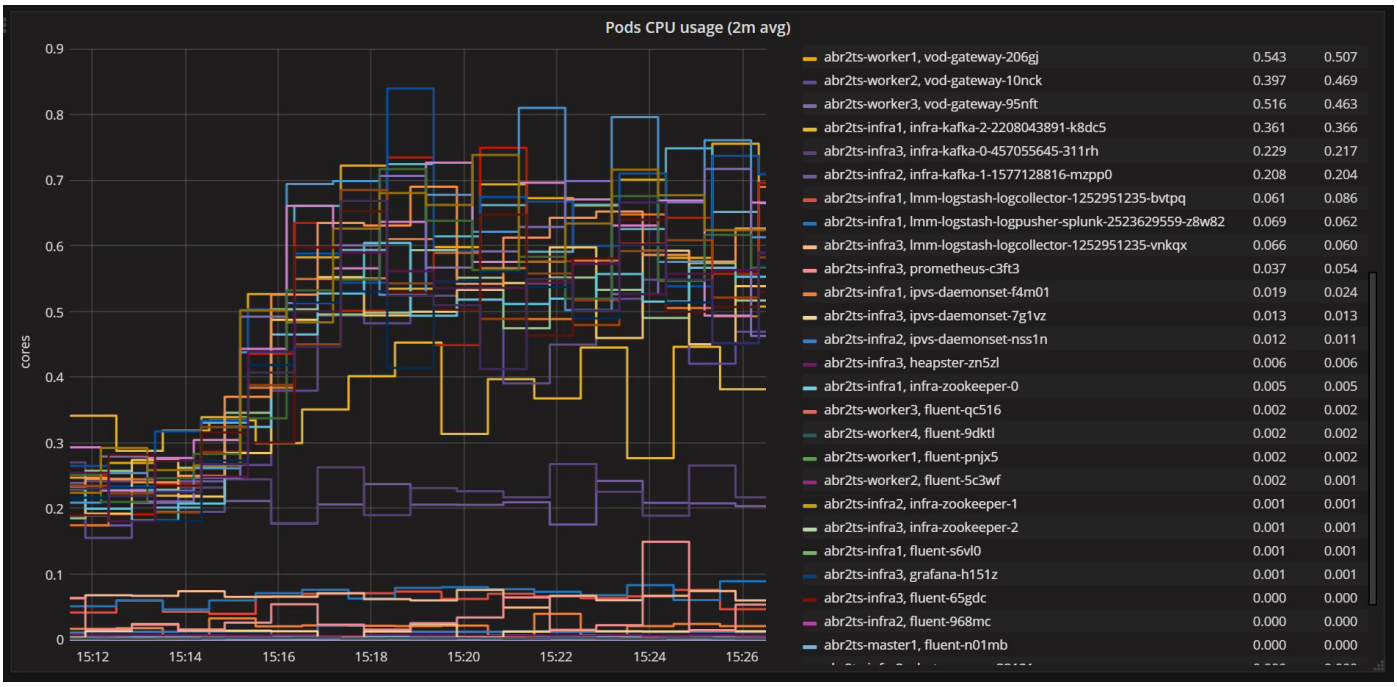
Figure 2-18 CMT Cluster Monitoring - Data Transmitted by VoD Gateway



POD CPU Usage

The **Pods CPU Usage** dashboard (Figure 2-19) shows a graphical representation of the ongoing CPU usage of the CMT pods as averaged over a 2 minute period. Time is shown on the x-axis, with CPU cores used shown on the y-axis.

Figure 2-19 CMT Cluster Monitoring - POD CPU Usage

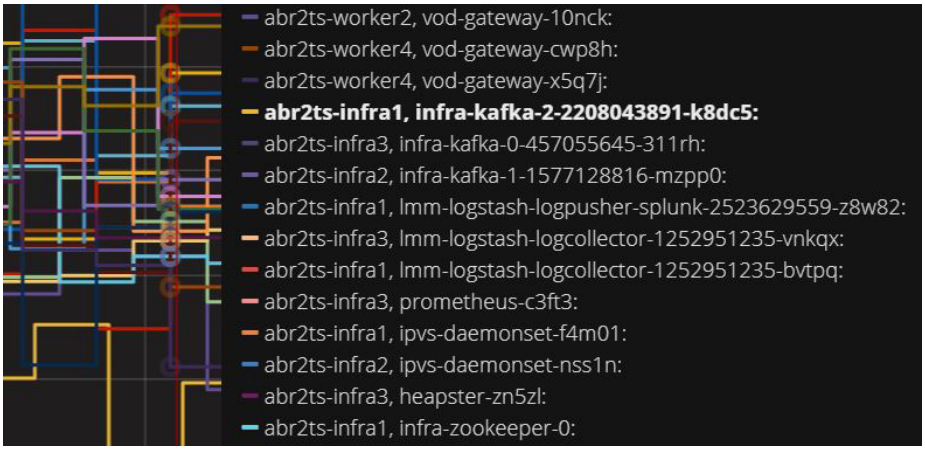


POD CPU Usage Details

Viewing Graph Details

Similar to the worker dashboards, as you view the various graphs, you can move your mouse over the lines to see a pop-up listing of specific data points. Figure 2-20 shows dots highlighting specific point in time. The focus is on **abr2ts-infra1, infra-kafka-2** pod. Data details will be shown at the right of pop up and will vary depending on the nature of the graph being viewed. Moving the mouse to a different point in the graph will change the focus to other pods. The functionality here applies to all similar graphs.

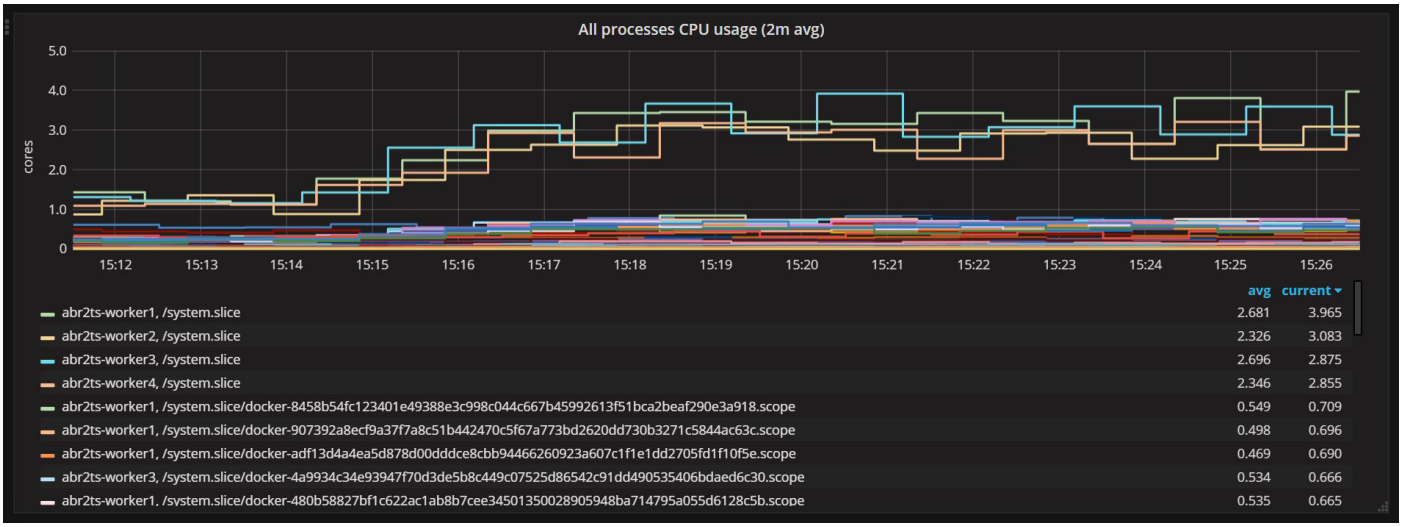
Figure 2-20 CMT Cluster Monitoring - POD CPU Usage Graph Detail



All Processes CPU Usage

The **All processes CPU usage** dashboard (Figure 2-21) shows a graphical representation of the ongoing CPU usage for all Linux processes on each node averaged over a 2-minute time period. Time is shown on the x-axis, with CPU cores used being shown on the y-axis. Average and current core usage is summarize for each node at the bottom of the dashboard.

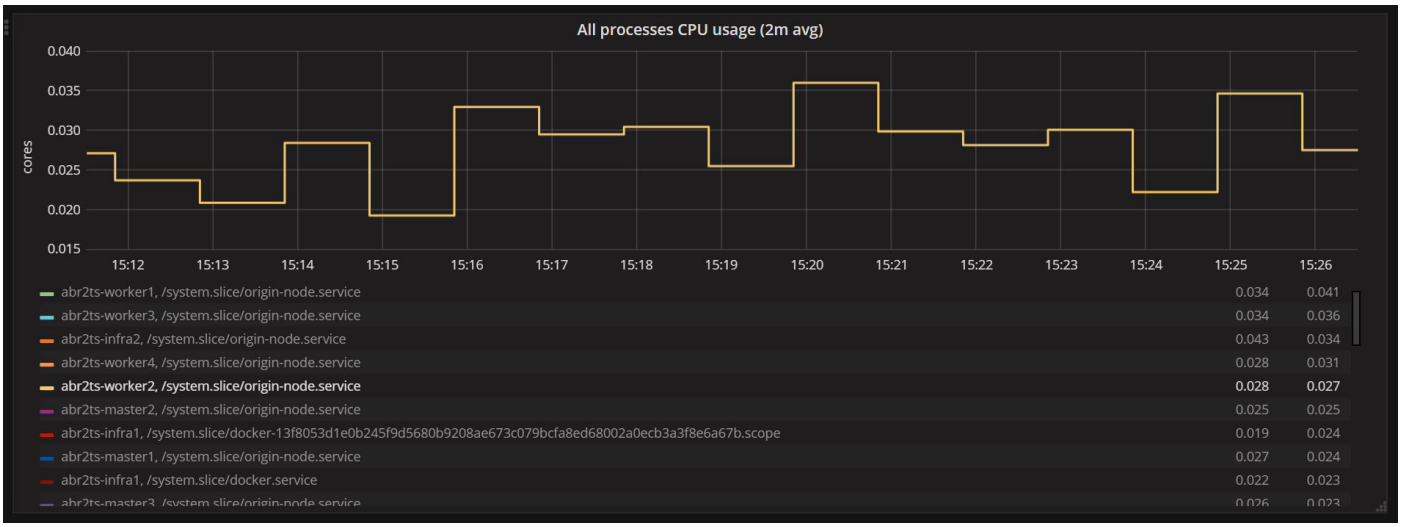
Figure 2-21 CMT Cluster Monitoring - All Processes CPU Usage



Selecting Individual Services

For the **All processes CPU usage** dashboard (Figure 2-22), clicking on any process in the listing at the bottom will show a graphical representation of the metrics for that process in a node.

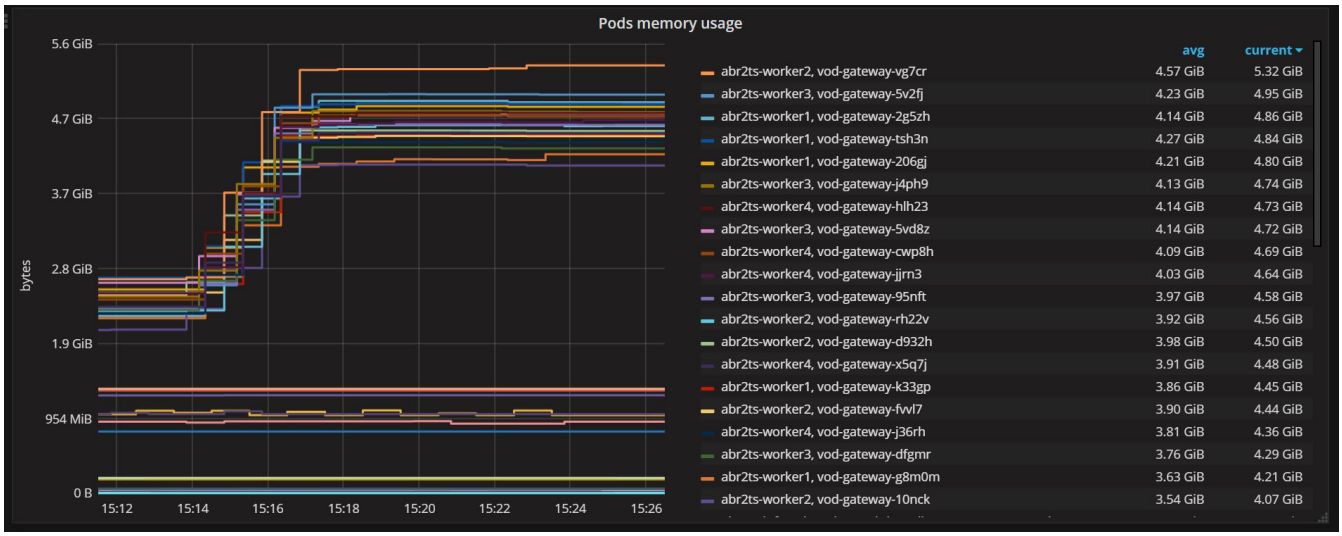
Figure 2-22 CMT Cluster Monitoring - All Processes CPUs Usage - Individual Service Selected



Pod Memory Usage

The **Pods Memory Usage** dashboard (Figure 2-23) shows a graphical representation of the ongoing memory usage for all processes for each pod. Time is shown on the x-axis, with Gibibytes (GiB) being shown on the y-axis. Average and current memory usage is summarize for each pod to the right of the dashboard.

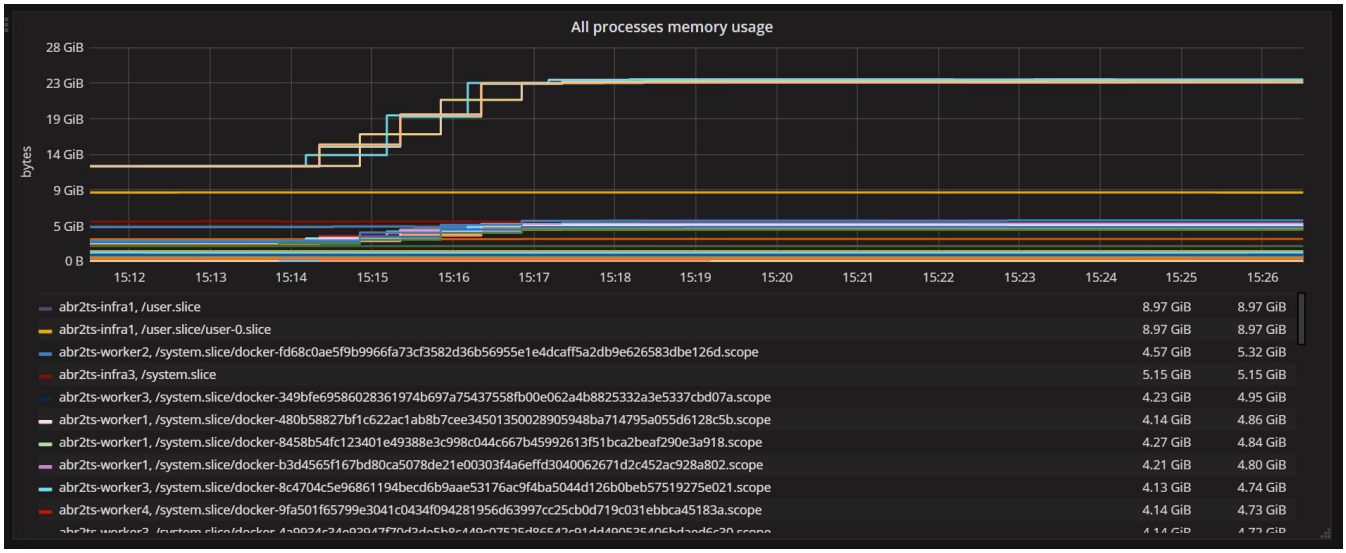
Figure 2-23 CMT Cluster Monitoring - POD Memory Usage



All Processes Memory Usage

The **All Processes Memory Usage** dashboard (Figure 2-24) shows a graphical representation of the ongoing memory usage for all processes for each pod. Time is shown on the x-axis, with Gibibytes (GiB) being shown on the y-axis. Average and current memory usage is summarize for each process at the bottom of the dashboard.

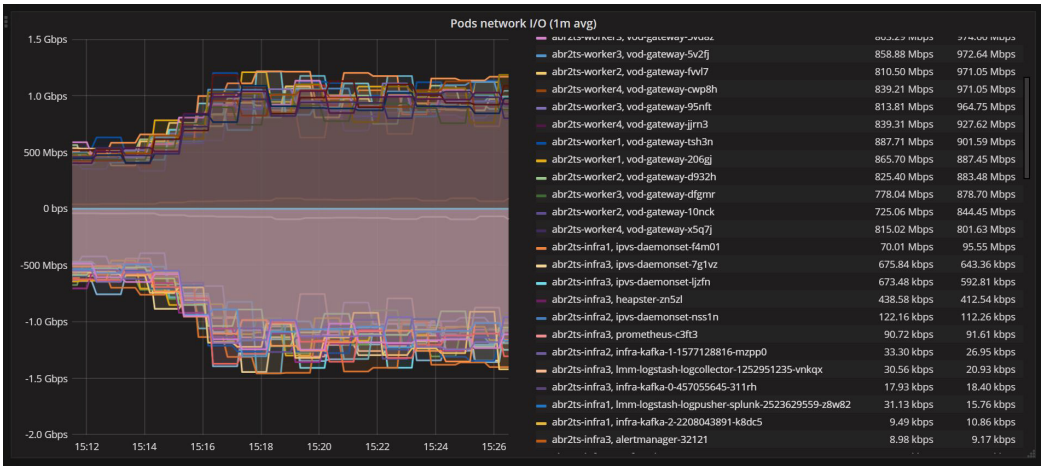
Figure 2-24 CMT Cluster Monitoring - All Processes Memory Usage



POD Network I/O

The **Pods Network I/O** dashboard (Figure 2-25) shows a graphical representation of the ongoing data throughput for all processes for each pod as averaged over a 1 minute time period. Time is shown on the x-axis, with Gibibytes (GiB) being shown on the y-axis. Average and current data throughput is summarize for each pod at the right of the dashboard.

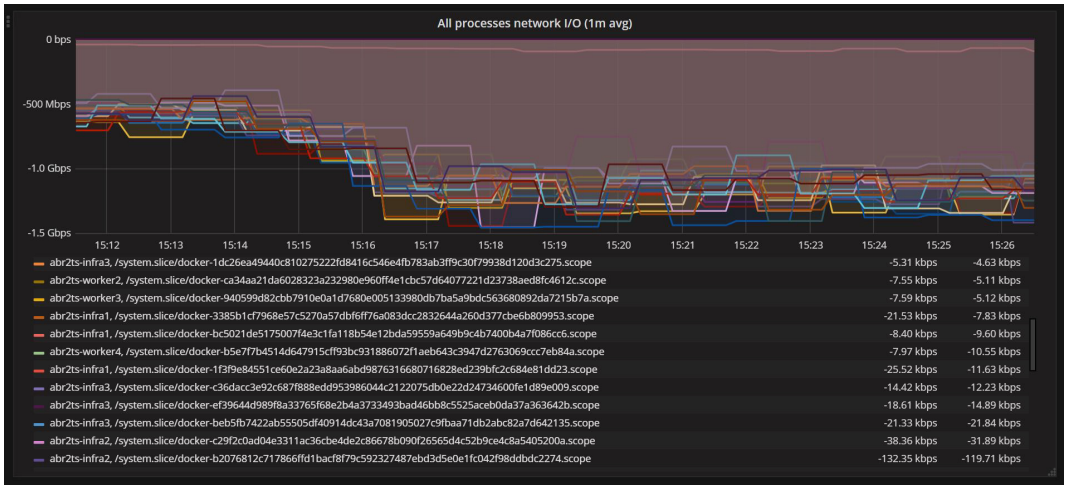
Figure 2-25 CMT Cluster Monitoring - POD Network I/O



All Processes Network I/O

The **All Processes Network I/O** dashboard (Figure 2-25) shows a graphical representation of the ongoing data throughput for all processes for the entire cluster as averaged over a 1 minute time period. Time is shown on the x-axis, with megabits per second (Mbps) or gigabits per second (Gbps) being shown on the y-axis. Average and current data throughput is summarize for the entire cluster at the bottom of the dashboard.

Figure 2-26 CMT Cluster Monitoring - All Processes Network I/O



Log Messages

Log Message Overview

This is a comprehensive list of log/error messages that can be generated by the VoD gateway application.

Note Message specifics are included in the "message" portion of the JSON or "message" field in the key/value pairs, which are added by the application.

Data Dictionary

Each of the gateway operations - bitrate, index, cbr, iframe - will log an informational message upon completion of the operation. The purpose of the message is to indicate success or failure of the operation. Additional key/value pairs are added to the informational message to help with analytics gathering. The following table describes the "data dictionary" or set of key/value pairs.

Table A-1 Data Dictionary Keys/Values/Definition

Key	Value	Definition
result	success | failure	The result of the operation.
timestamp	<datetime>	The date and time the operation completed.
FCID	<string>	Flow Control ID
httpCode	<status code>	HTTP status code returned by the operation.
api	Bitrate | Index | CBR | IFrame	The API called that generated this message.
httpMethod	GET	Only GET is currently supported.
url	<url>	The request URL, including the query string.
remoteAddr	<ip> | <hostname>	The value of the remote host that submitted the request
duration	<milliseconds>	The time it took to complete the request in milliseconds

Table A-1 Data Dictionary Keys/Values/Definition

Key	Value	Definition
appError	<error>	Error code, reason and message if result is failure.
level	ERROR | WARN | INFO	The logging level of the message.

Success Example

```
{ "timeStamp" : "2017-11-22T11:22:20.677Z" ,"component" : "abr2ts-vg" ,"module" :
"abr2ts-vg" ,"level" : "INFO" ,"FCID" : "5d0293ce-19f7-420f-9901-4a95d74b042f" ,"api" :
"IFrame" ,"duration" : 28 ,"httpCode" : 200 ,"httpMethod" : "GET" ,"remoteAddr" :
"192.165.3.232:49165" ,"result" : "success" ,"url" : "abr2ts-vgserver.com/iframe?
mpd=http://abr2ts-vgserver.com/manifest.mpd&bitrate=2178622&offsets=10092005" }
```

Failure Example

```
{ "timeStamp" : "2017-11-22T11:22:22.685Z" ,"component" : "abr2tsvg"
,"module" : "abr2ts-vg" ,"level" : "ERROR" ,"FCID" : "aaf1b3e5-7df1-4118-
b8a7-20e773d647c2" ,"api" : "IFrame" ,"appError" : {"errorCode":1204,"reason":"HTTP
write issue","message":"write tcp4 10.129.0.17:8000->192.165.3.232:49171: write: broken
pipe"} ,"duration" : 1032 ,"httpCode" : 200 ,"httpMethod" : "GET" ,"remoteAddr" :
"192.165.3.232:49171" ,"result" : "failure" ,"url" : "abr2ts-vgserver.com/iframe?
mpd=http://abr2ts-vgserver.com/manifest.mpd&bitrate=2178622&offsets=10184805" }
```

Configuring Logging Options

The VoD gateway application log level can be changed from within a JSON configuration file. By default, the log level is set to “INFO”. To change log level for the VoD-gateway application:

Step 1 SSH into the deployer node as root.

Step 2 Open the following file for editing:

```
/root/abr2ts-deployment/platform/resources/config/vod-gateway/vod-gateway-rc.json
```

Step 3 Set the following environment variable to the desired log level:

```
{
"name": "LOG_LEVEL",
"value": "INFO"
}
```

Info Log Level Format Example

```
2017-08-13T08:21:20.896Z component=abr2tsvg,module=httpServer,
level=INFO,FCID=b39a4055-6ef7-4591-899eac77e795ff84,api=CBR,
httpMethod=GET,url=localhost:8080/cbr
```

JSON Format Example

```
{ "timeStamp" : "2017-08-12T16:10:59.78Z" ,"component" : "abr2ts-vg" ,"module" :
"httpServer" ,"level" : "INFO" ,"FCID" : "f6e4ce5f-2a8d-4080-a69f-28fa32346244" ,"api" :
"CBR" ,"httpMethod" : "GET" ,"url" : "localhost:8080/cbr" }
```

HTTP Requests

HTTP requests are logged by an internal library within CMT. The "FCID" parameter can be matched with any errors that are logged during the processing of a request (the example error messages below can be matched to the example requests found here). This allows execution paths to be traceable.

Examples:

```
{ "timeStamp" : "2017-08-12T16:00:32.076Z" ,"component" : "abr2ts-vg" ,"module" :
"httpServer" ,"level" : "INFO" ,"FCID" : "575fe9e4-0166-40e5-8072-35bdad2800fc" ,"api" :
"Bitrate" ,"httpMethod" : "GET" ,"url" : "localhost:8080/bitrate" }
```

Gateway Log Messages:

```
{ "timeStamp" : "2017-08-12T16:09:29.057Z" ,"component" : "abr2ts-vg" ,"module" :
"httpServer" ,"level" : "INFO" ,"FCID" : "95f98964-bec4-4fc0-9786-23adff0b49f9" ,"api" :
"Bitrate" ,"httpMethod" : "GET" ,"url" : "localhost:8080/bitrate" }
{ "timeStamp" : "2017-08-12T16:10:59.78Z" ,"component" : "abr2ts-vg" ,"module" :
"httpServer" ,"level" : "INFO" ,"FCID" : "f6e4ce5f-2a8d-4080-a69f-28fa32346244" ,"api" :
"CBR" ,"httpMethod" : "GET" ,"url" : "localhost:8080/cbr" }
```

Error Messages

Config Parameter Processing

The error messages for config parameter reporting are:

Table A-2

Code	Reason	Message
1001	Invalid config parameter	<name>='<value>' is invalid

Example:

{ "timeStamp" : "2017-11-21T13:19:24.202Z" ,"component" : "abr2ts-vg" ,"module" : "abr2ts-vg" ,"level" : "ERROR" ,"appError" : {"errorCode":1001,"reason":"Invalid config parameter","message":"abr2ts.bitrate.Listing='foo' is invalid."} }

Note Some of the config parameters are handled by an internal library within CMT and are logged outside of the control of the gateway.

Example:

invalid value "foo" for flag -abr2ts.queue.Jitter: strconv.ParseInt: parsing "foo": invalid syntax

Query Parameter Processing

The error messages for query parameter reporting are:

Table A-3 Query Parameter Reporting Error Messages

Code	Reason	Message
1001	Missing query parameter	'mpd' is mandatory
1001	Missing query parameter	'version' is mandatory
1001	Missing query parameter	'bitrate' is mandatory
1001	Missing query parameter	'offsets' is mandatory
1102	Invalid query parameter	'mpd' <value> is not a valid URL
1102	Invalid query parameter	'bitrate' - strconv.ParseInt: parsing \"<value>\": invalid syntax
1102	Invalid query parameter	'start' - strconv.ParseInt: parsing \"<value>\": invalid syntax
1102	Invalid query parameter	'start' - less than zero
1102	Invalid query parameter	'end' - strconv.ParseInt: parsing \"<value>\": invalid syntax
1102	Invalid query parameter	'end' - less than or equal to zero
1102	Invalid query parameter	'start' larger than 'end'
1102	Invalid query parameter	'rated' - unsupported value '<value>'
1102	Invalid query parameter	'xrate' - strconv.ParseInt: parsing \"<value>\": invalid syntax
1102	Invalid query parameter	'xrate' - less than zero
1102	Invalid query parameter	'xrate' - found but 'output' is unrated
1102	Invalid query parameter	'brate' - required if 'btime' is set
1102	Invalid query parameter	'brate' - found but 'output' is unrated
1102	Invalid query parameter	'btime' - required if 'brate' is set
1102	Invalid query parameter	'btime' - found but 'output' is unrated
1102	Invalid query parameter	'brate' - strconv.ParseInt: parsing \"<value>\": invalid syntax
1102	Invalid query parameter	'brate' - less than zero
1102	Invalid query parameter	'btime' - strconv.ParseInt: parsing \"<value>\": invalid syntax
1102	Invalid query parameter	'btime' - less than zero
1102	Invalid query parameter	'brate'/'btime' - found but 'output' is unrated
1102	Invalid query parameter	'start' is beyond the end of the content
1102	Invalid query parameter	'end' is beyond the end of the content
1102	Invalid query parameter	'offsets' cannot contain whitespace
1102	Invalid query parameter	'offsets' - strconv.ParseInt: parsing \"<value>\": invalid syntax
1102	Invalid query parameter	'offsets' exceeds maximum of 64 elements

Table A-3 Query Parameter Reporting Error Messages

Code	Reason	Message
1102	Invalid query parameter	'offsets' is required to be in ascending/descending order
1102	Invalid query parameter	'fastfill' - strconv.ParseInt: parsing \"<value>\": invalid syntax
1102	Invalid query parameter	'fastfill' - less than zero
1102	Invalid query parameter	'offsets' [values] are not found in the content
1102	Invalid query parameter	No 'offsets' are found in the content
1103	Unknown query parameter	Key '<value>' found but not supported for current operation

Example HTTP response bodies:

```
{"Status":"Fail"ssskype,"error":{"errorCode":101,"reason":"Missing query
parameter","message":"'mpd' is mandatory"}}
{"Status":"Fail","error":{"errorCode":102,"reason":"Invalid query
parameter","message":"'bitrate' -
strconv.ParseInt: parsing \"xyz\": invalid syntax"}}
{"Status":"Fail","error":{"errorCode":103,"reason":"Unknown query
parameter","message":"fooooo"}}
```

Example log message entries:

```
{ "timeStamp" : "2017-08-12T16:09:29.057Z" ,"component" : "abr2ts-vg" ,"module" :
"httpServer" ,"level" : "ERROR" ,"FCID" : "95f98964-bec4-4fc0-9786-23adff0b49f9" ,"api" :
"Bitrate" ,"appError" : {"errorCode":101,"reason":"Missing query
parameter","message":"'mpd' is mandatory"} ,"bytes" : 109 ,"duration" : 0 ,"httpCode" :
400 ,"httpMethod" : "GET" ,"msg" : null ,"remoteAddr" : "127.0.0.1" ,"result" :
"failure" ,"url" : "localhost:8080/bitrate" }
{ "timeStamp" : "2017-08-12T16:10:59.806Z" ,"component" : "abr2tsvg"
,"module" : "httpServer" ,"level" : "ERROR" ,"FCID" : "f6e4ce5f-2a8d-4080-
a69f-28fa32346244" ,"api" : "CBR" ,"appError" : {"errorCode":102,"reason":"Invalid
query parameter","message":"'bitrate' - strconv.ParseInt: parsing \"xyz\": invalid
syntax"} ,"bytes" : 152 ,"duration" : 25 ,"httpCode" : 400 ,"httpMethod" : "GET" ,"msg" :
null ,"remoteAddr" : "127.0.0.1" ,"result" : "failure" ,"url" : "localhost:8080/cbr?
mpd=http://172.22.102.107/sample/manifest.mpd&bitrate=xyz" }
{ "timeStamp" : "2017-08-12T16:00:32.077Z" ,"component" : "abr2ts-vg" ,"module" :
"httpServer" ,"level" : "ERROR" ,"FCID" : "575fe9e4-0166-40e5-8072-35bdad2800fc" ,"api" :
"Bitrate" ,"appError" : {"errorCode":103,"reason":"Unknown query
parameter","message":"fooooo"} ,"bytes" : 98 ,"duration" : 0 ,"httpCode" :
400 ,"httpMethod" : "GET" ,"msg" : null ,"remoteAddr" : "127.0.0.1" ,"result" :
"failure" ,"url" : "localhost:8080/bitrate?mpd=http://172.22.102.107/test/
manifest.mpd&fooooo=hey" }
```

HTTP Handling

The error messages for HTTP handling are::

Table A-4 HTTP Handling Error Messages

Code	Reason	Message
1201	HTTP Get failed	Status code for <url> is <code>
1201	HTTP Get failed	Could only read <partial> bytes from <url>, expected <total> bytes (<error>)
1201	HTTP Get failed	<OS specific error message>
1202	HTTP Head failed	Status code for <url> is <code>
1202	HTTP Head failed	<OS specific error message>
1203	HTTP read failed	does not contain enough data to continue processing
1203	HTTP read failed	Could only read <partial> bytes of packet, expected <total> bytes
1203	HTTP read failed	bytes.Buffer: too large
1204	HTTP write failed	Write truncated - wrote <partial> bytes out of <total>
1204	HTTP write failed	<OS specific error message>

MPEG Processing

The error messages for MPEG-related processing are::

Table A-5 MPEG-related Processing Error Messages

Code	Reason	Message
1301	PSI error	PMT byte zero (<value>) is missing sync byte
1301	PSI error	PMT is missing PUSI
1301	PSI error	PMT has PCR
1301	PSI error	PMT is missing payload
1301	PSI error	PMT has invalid pointer field <value>
1301	PSI error	PMT has invalid table id <value>
1301	PSI error	PMT has invalid section length <value>
1301	PSI error	PMT has invalid program info length <value>
1301	PSI error	PMT has invalid esinfo length <value>
1301	PSI error	Could not find video PID in PMT
1302	AVC error	bytes.Buffer: too large
1302	AVC error	Cannot locate SEQ for pid <value>
1302	AVC error	Cannot locate SEQ EXT for pid <value>
1302	AVC error	Unsupported SPS profile <id>

Table A-5 MPEG-related Processing Error Messages

Code	Reason	Message
1302	AVC error	Invalid seq_parameter_set_id <id> >= 32
1302	AVC error	Invalid seq_parameter_set_id <id> != 0
1302	AVC error	Invalid pic_order_cnt_type == 1
1302	AVC error	Cannot determine hsize or vsize
1303	CBR error	Not enough spacing - <bytes> (total) < <bytes> (required)
1303	CBR error	<count> nulls left to fill
1303	CBR error	<count> nulls overfilled
1303	CBR error	<count> bytes left to convert
1304	MPEG-TS error	Cannot find PTS, packet has no payload
1304	MPEG-TS error	Cannot find PTS in packet payload
1304	MPEG-TS error	bytes.Buffer: too large
1304	MPEG-TS error	Could only read <count> bytes of packet, expected <total> bytes

ABR Processing

The error messages for ABR (DASH) related processing are::

Table A-6 ABR Processing-Related Error Messages

Code	Reason	Message
1401	MPD error	No segment template found for representation <id>
1401	MPD error	No video/mp2t representation found
1401	MPD error	Media attribute missing from SegmentTemplate
1401	MPD error	ID attribute missing from Representation
1401	MPD error	No segments generated from SegmentTimeline
1401	MPD error	<MPD parsing error>
1402	Segment error	No segments available to index
1402	Segment error	Segment <url> does not have exact packet boundary of <value> bytes
1403	Version error	Version <version> is not supported
1404	Index error	Cannot find index entry for

Alert Manager

Alert Manager Overview

Prometheus allows users to define alert conditions based upon predefined expressions within an Alert Rules file. It then notifies an external service (AlertManager in our case) to fire alerts once specific thresholds have been reached. Whenever an alert expression is evaluated as true, that alert becomes active.

Updating Alert Rules

The following process is used to update the Alert Rules file, update necessary configuration settings, and then restart the system so that the changes take effect. To update Alert Rules:

Step 1 If necessary, SSH as root into the Deployer node.

Step 2 The rules file is located at:
`/root/abr2ts-deployment/platform/resources/config/prometheus/alert.rules`

Step 3 Make a backup of the rules file.

Step 4 Edit the file to set the parameters and thresholds that you need monitored.
For background information on the Prometheus querying language, rules, conventions, and available metrics, see Alert Rules Reference Materials, page B-2.

Step 5 Navigate to `/root/abr2ts-deployment/scripts/`

Step 6 Run this command to stop the Infra node.

```
./abr2ts_infra.sh stop
```

Step 7 Run this command to update the Alert Manager configuration settings on the Infra node.

```
./abr2ts_infra.sh config
```

Step 8 Run this command to start the Infra node. The rules file will automatically be loaded at start up.

```
./abr2ts_infra.sh start
```

Alert Rules Reference Materials

The following section provides links to background information that you will find useful when creating or editing Alert Rules:

- For details on how Alert Rules are defined, refer to:
  `https://prometheus.io/docs/prometheus/latest/configuration/alerting_rules/`
- For details on the Prometheus querying language, refer to:
  `https://prometheus.io/docs/prometheus/latest/querying/basics/`
- Metrics probed by the querying functions are provided by the Kubernetes API. Information related to metrics and monitoring is available at this URL:
  `https://coreos.com/blog/monitoring-kubernetes-with-prometheus.html`

Sample Alert Rule

The following section lists a sample Alert Rule for your reference.

Sample Alert Rule

```
ALERT ClusterContainerMemoryUsage

  IF sum (container_memory_working_set_bytes{id="/",kubernetes_io_hostname=~"abr2ts-.*"})
/ sum (machine_memory_bytes{kubernetes_io_hostname=~"abr2ts-.*"}) * 100  > 50

  FOR 10s

  LABELS { severity = "critical" }

  ANNOTATIONS {

     summary = "cluster containers consuming high level of memory",

     description = ""

  }
```

Alert Rule Commands

The following table provides explanations for some commands used when creating Alert Rules.

Table B-1 Alert Rule Commands

Label	Possible Values	Description
ALERT	QueryContainerMemoryUsage	Name of alert rule
ANNOTATIONS	summary = “…….” Description = “…….”	Annotations for the alert
FOR	10s	The optional for clause causes Prometheus to wait for a certain duration between first encountering a new matching condition.

Table B-1 Alert Rule Commands

Label	Possible Values	Description
IF	sum(container_memory_working_set_bytes {id="/",kubernetes_io_hostname=~"abr2ts-.*"})	sum is a Prometheus query function
	/ sum (machine_memory_bytes{kubernetes_io_hostname=~"abr2ts-.*"}) * 100 > 90	container_memory_working_set_bytes and machine_memory_bytes are kubernetes metrics expression checks whether memory usage is greater than 90%
LABELS	severity=”critical”	One or more labels for the alert

Inspecting Alerts at Runtime

To manually view the exact label sets for which alerts are active (meaning pending or firing), navigate to the "Alerts" tab within Prometheus. The alert value is set to 1 as long as the alert remains in an active state. When the alert transitions to an inactive state, the alert value will be changed to 0 by the system.

Sending Alert Notifications

Prometheus' Alert Rules are suitable for basically assessing what is going wrong at a given time. An additional component is required to add summarization, notification rate limiting, silencing, and other features on top of the provided simple alert definitions. The AlertManager component takes on this task. Prometheus is configured to periodically send information about alert states to the AlertManager instance, which is then responsible for dispatching the right notifications.

The following page shows how alerts are pushed out from Prometheus to AlertManager:

Figure B-1 Prometheus UI Alert Tab - showing pending alerts

Figure B-2 Prometheus UI Alert Tab - showing firing alerts

Sample Alert Notifications

The following default sample alerts will be packaged with the CMT release.

Table B-2 Sample Alert Notifications

Label	Description	Default Duration
NodeDown	A node in Media Transformer is down for n minutes	5 minutes
VODGatewayTotalMemoryUsage	VOD Gateway memory usage exceeded a certain threshold on a node.	10 minutes
VODGatewayPercentageMemory Usage	VOD Gateway node memory usage exceeded a threshold percentage (default=90%)	10 minutes
VODGatewayCPUUsage	VOD Gateway node CPU usage exceeded a threshold percentage (default=80%)	10 minutes
ClusterContainerMemoryUsage	Overall memory usage of containers in the cluster exceeded a certain threshold percentage (default=90%)	10 minutes
ApiServerDown	The Kubernetes server is down. This indicates that the system is probably unstable.	5 minutes