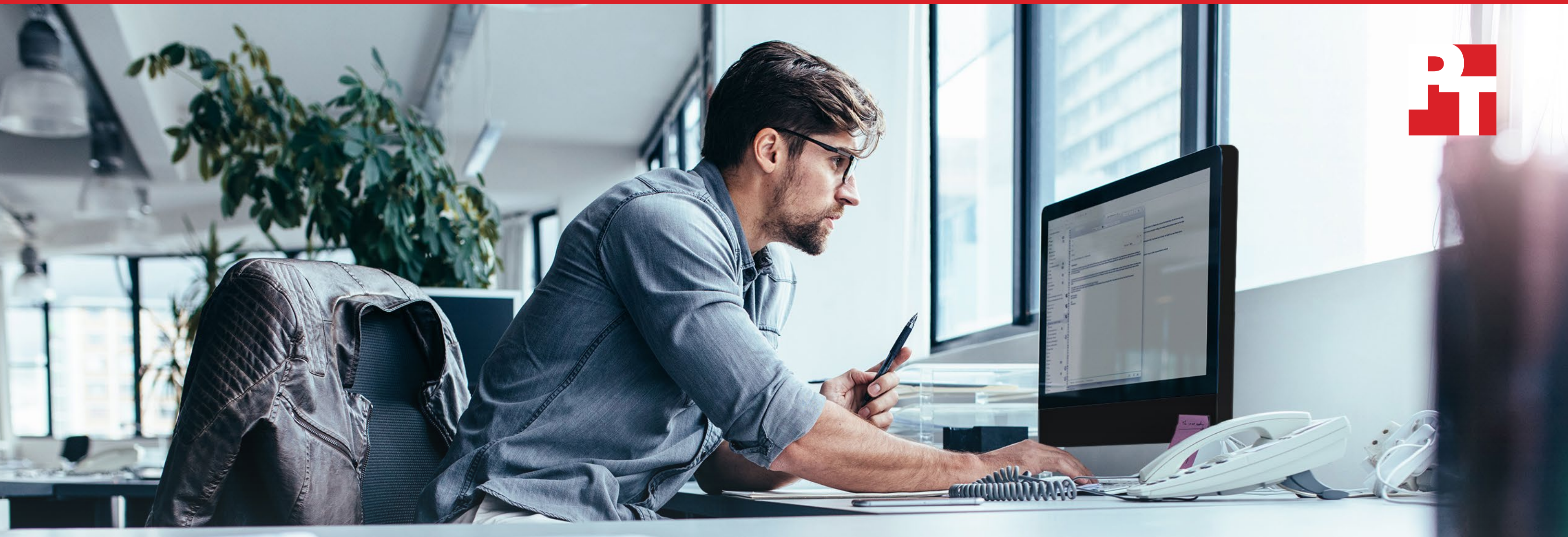


Create useful data center health visualizations with Dell iDRAC9 Telemetry reference toolset and Elastic Stack

Dell EMC™ has recently released reference tools for the Integrated Dell Remote Access Controller version 9 (iDRAC9) Telemetry feature. The Telemetry reference toolset enables data center engineers to ingest and structure telemetry data streams for visualization with Elastic Stack software. This report serves as a how-to guide for setting up and configuring these components within your own environment.

At Principled Technologies, we set up the iDRAC9 Telemetry reference toolset with a containerized Elastic Stack that comprised Elasticsearch and Kibana on a Dell EMC PowerEdge™ R6515 with a 3rd Gen AMD EPYC™ processor. We did not encounter any significant issues, and within a single business day, our data center technician was able to go from initial setup to creating Kibana visualizations using real telemetry data streams from multiple iDRAC9 sources. Additionally, to show the value of running this environment on the Dell EMC and AMD hardware, we stress-tested our telemetry environment with the Elasticsearch benchmarking tool Rally. Telemetry ingest remained uninterrupted during the stress test.

Get started quickly with iDRAC9 Telemetry Streaming and Elastic Stack

Using the new iDRAC9 Telemetry reference toolset and an Elastic Stack container deployment, our team was able to query and visualize telemetry data starting from scratch in less than a day.

Continue to ingest telemetry data even during heavy server load

Telemetry ingest remained uninterrupted even while our environment ran a workload that stressed Elastic Stack and the system CPU.

Table of contents

Dell EMC PowerEdge R6515 servers

According to Dell Technologies, these single-socket 1U servers offer the following high-level specifications:

- Up to 64 high performance 3rd Gen AMD cores
- PCIe Gen4 for high throughput connectivity
- Support for up to 10 SAS/SATA/NVMe drives
- Improved VM density and SQL performance

Learn more at https://www.dell.com/en-us/work/shop/povw/poweredge-r6515.

3rd Gen AMD EPYC processors

The latest offering from AMD, 3rd Gen EPYC processors offer increased I/O with "up to 32MB L3 cache per core," 7nm x86 technology, and new security features like Secure Encrypted Virtualization - Secure Nested Paging (SEV-SNP) and Encrypted State (SEV-ES). AMD positions the EPYC 75F3 model as being well suited for high frequency use cases such as VM density, virtualization, and VDI. For more information, visit https://www.amd.com/en/processors/epyc-7003-series.

Overview: iDRAC9 and Elastic Stack

In this how-to guide, we will describe the steps required to set up and use the new Dell iDRAC9 Telemetry tool with a containerized Elastic Stack deployment.

What is iDRAC9 Telemetry Streaming?

iDRAC9 v4.0 introduced the Telemetry Streaming feature, which allows system administrators to stream data from system hardware sensors and general health reporting to use in data analytics. (Note: Telemetry Streaming is available for Dell EMC PowerEdge systems with iDRAC9 Datacenter licenses.)

Collecting telemetry data via streaming is a newer and improved method compared to polling, a process that can be challenging to scale across a large data center. According to Dell EMC, the polling method is also prone to sampling errors and delays, whereas streaming is said to be more accurate and efficient.[1]

Administrators can use telemetry data with an analytics solution to predict hardware failures (thus enabling staff to proactively replace devices in danger of failing); to determine how best to improve system operations by analyzing data for key insights; and to tighten security measures, among other use cases. Because each telemetry stream in iDRAC9 contains more than 180 data points ranging from thermal data to system resource utilization, administrators will have many different ways in which to monitor their data centers. (For more information on iDRAC9 Telemetry Streaming, visit https://www.delltechnologies.com/resources/en-us/asset/white-papers/products/software/direct-from-development-datacenter-telemetry.pdf.)

While iDRAC9 produces the telemetry report streams, it does not analyze or visualize data on its own. With the iDRAC9 Telemetry reference toolset, however, administrators can use other components of Elastic Stack to perform those functions.

What is Elastic Stack?

Elastic Stack is a group of open-source software that enables users to format, search, analyze, and visualize real-time data from virtually any type of source. Elastic Stack consists of three components: Elasticsearch, Kibana, and an ingest tool such as Logstash.

Elasticsearch is a search and analytics engine that uses indices to store and retrieve data in JSON documents. This data can come in a variety of forms (structured, unstructured, textual, numerical, etc.), and can come from a number of sources, such as an application log or iDRAC9 telemetry data.

To ingest and enrich the data before it is indexed in Elasticsearch, Elastic Stack uses Logstash as their ingestion tool. In our guide, the iDRAC9 Telemetry reference toolset takes care of that ingestion and enrichment step before the data is indexed.

The final piece in the Elastic Stack is Kibana, a free and open-source front end application that enables visualization and analysis of data stored in an elastic index or indices. With Kibana's built-in visualization tools, you can create dashboards that are a collection of graphs, maps, charts, and tables. We finish off our guide with instructions on how to query the data for meaningful reports and dashboards full of charts and graphs.

Deploying Elastic Stack

There are several ways to deploy Elastic Stack. Depending upon your environment, you may need to use a different method for installation and configuration. For this guide, we elected to deploy Elastic Stack and the iDRAC9 Telemetry reference toolset in Docker containers using Docker-Compose to automate that process. We chose containers rather than VMs for a couple of reasons. First, in our testing, we had only a single server to work with. Running Elastic Stack and the iDRAC9 Telemetry reference toolset in containers meant that we could deploy the entire environment on a single server. Containers also provide a more portable environment, allowing you to easily move your applications and data to another location if necessary. On their GitHub repository, Dell includes the Docker-Compose YAML files we used to deploy the Elastic Stack environment (https://github.com/dell/iDRAC-Telemetry-Reference-Tools/).

We start off with instructions for installing Ubuntu 20.04.1 LTS (focal) and the containerized Elastic Stack prerequisites Docker, Docker-Compose, and ActiveMQ on the Dell EMC PowerEdge R6515 server powered by a 3rd Gen AMD EPYC processor.

Installing and configuring Ubuntu 20.04.1 LTS

1. Boot to the installation media.
2. Select English.
3. If prompted, update to the new installer.
4. Select the keyboard layout, and click Done.
5. Configure the Network connections, and click Done.
6. To skip past the proxy address screen, click Done.
7. Leave the mirror as default, and click Done.
8. Configure the storage. For our testing, we used the entire disk.
9. Click Done.
10. Setup a username, hostname, and password. Click Done.
11. To install OpenSSH Server, check the box, and click Done.
12. Install any additional packages, and click Done.
13. When the installation is complete, select Reboot Now.
14. Log into the server.
15. Update and upgrade the OS:
    ```
    sudo apt update
    sudo apt upgrade
    ```
16. Create a partition on the disks for Docker containers:
    ```
    for i in {0..5};do sudo gdisk /dev/nvme$in1; done
    ```
17. Create physical volumes:
    ```
    for i in {0..5};do sudo pvcreate /dev/nvme$in1; done
    ```
18. Create a virtual group:
    ```
    sudo vgcreate data /dev/nvme0n1 /dev/nvme1n1 /dev/nvme2n1 /dev/nvme3n1 /dev/nvme4n1 /dev/nvme5n1
    ```
19. Create a logical volume:
    ```
    sudo lvcreate -type raid10 -L 2T data
    ```
20. Create an EXT4 filesystem on the logical volume:
    ```
    sudo mkfs.ext4 /dev/data/lvol0
    ```
21. Create a directory for Docker:
    ```
    sudo mkdir /var/lib/docker
    ```
22. Mount the logical volume to the Docker directory:
    ```
    sudo mount /dev/data/lvol0 /var/lib/docker
    ```
23. Edit /etc/fstab, and add the mounted volume for persistence across reboots.
24. Reboot.
25. Once you have installed and configured the OS, you will need to install Docker and Docker-Compose in order to deploy Elastic Stack and the iDRAC9 Telemetry reference toolset within a containerized environment.

Installing Docker

1. Uninstall any Docker components that are already installed.

```
sudo apt-get remove docker docker-engine docker.io containerd runc
```

2. Install all prerequisites.

```
sudo apt-get install apt-transport-https ca-certificates curl gnupg-agent software-properties-common
```

3. Add the official Docker gpg key:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

4. Set up the stable repository:

```
sudo add-apt-repository    "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) \
stable"
```

5. Update the apt index:

```
sudo apt-get update
```

6. Install Docker:

```
sudo apt-get install docker-ce docker-ce-cli containerd
```

7. To use Docker as a non-root user, add your user to the Docker group:

```
sudo usermod -aG docker <user>
```

Installing Docker-Compose

1. Download the latest release of Docker-Compose:

```
sudo curl -L "https://github.com/docker/compose/releases/download/1.28.0/docker-compose-$(uname -s)-
$(uname -m)" -o /usr/local/bin/docker-compose
```

2. Change the permissions of the binary to allow execution:

```
sudo chmod +x /usr/local/bin/docker-compose
```

The iDRAC9 Telemetry reference toolset has two prerequisites: ActiveMQ and Go. Because we chose to deploy in a containerized environment, we did not need to perform a local installation of either. Note: If you are using multiple systems, you will need to install Go to build and start the required components.

Enabling the Telemetry Streaming service in iDRAC9

There are several ways to enable the iDRAC9 Telemetry Streaming service, such as via iDRAC9 RACADM. To simplify this process, Dell has provided sample scripts on their GitHub repository. Below are the steps we used to enable telemetry reporting on several target iDRAC9 instances within our environment.

1. Clone the GitHub repository that contains the iDRAC9 Telemetry utilities:

```
wget https://github.com/dell/iDRAC-Telemetry-Scripting/archive/master.zip -O iDRAC-
TelemetryScripting-master.zip
```

2. Unzip the file, and change directories:

```
unzip iDRAC-TelemetryScripting-master.zip
cd iDRAC-TelemetryScripting-master/
```

3. Enable telemetry reports and alerts on each iDRAC9 server:

```
python3 ./ConfigurationScripts/EnableOrDisableAllTelemetryReports.py -ip <IP ADDRESS> -u <USER> -P
<PASSWORD> -s Enabled
python3 ./ConfigurationScripts/SubscriptionManagementREDFISH.py -ip <IP ADDRESS> -u <USER> -p
<PASSWORD> -c y
```

Setting up the iDRAC9 Telemetry reference toolset

1. Clone the GitHub repository that contains the iDRAC9 Telemetry reference toolset:

   ```
   git clone https://github.com/dell/iDRAC-Telemetry-Reference-Tools.git
   ```

2. Change directories into the iDRAC9 Telemetry reference toolset:

   ```
   cd iDRAC-Telemetry-Reference-Tools
   ```

3. In the config.ini file, add the iDRAC9 information for the systems you wish to monitor. We have included a reference in the Appendix.

Starting the Elastic Stack and iDRAC9 Telemetry reference toolset

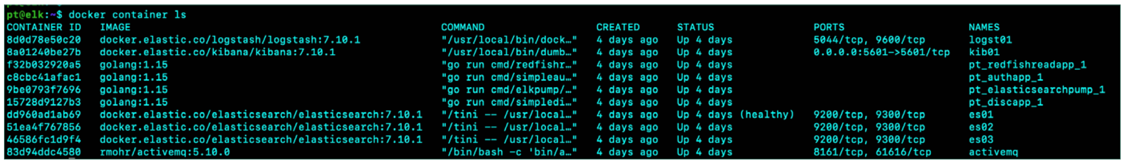
We have provided a reference Docker-Compose file in the Appendix to help guide you through the basic requirements for starting up the Elastic Stack. You may need to alter certain specifications such as volume location, networks, ports, or naming schemes based on your specific environment. Note: In our reference Docker-Compose file we do not have security enabled for the Elastic Stack. Elastic recommends securing your Elastic Stack after successfully deploying your initial cluster. See the following guide for more information: https://www.elastic.co/guide/en/elasticsearch/reference/current/configuring-security.html

1. Navigate to the location of your Docker-Compose file.
2. Start up the Elastic stack and iDRAC9 Telemetry reference toolset:

   ```
   docker-compose -f elk.yml up -d
   ```

3. After the cluster starts successfully, check to make sure all of your containers are still running:

   ```
   docker container ls
   ```



```
pt@elk:~$ docker container ls
CONTAINER ID   IMAGE                                                  COMMAND                  CREATED      STATUS                  PORTS                        NAMES
8d0d78e50c20   docker.elastic.co/logstash/logstash:7.10.1             "/usr/local/bin/dock…"   4 days ago   Up 4 days               5044/tcp, 9600/tcp           logst01
8a01240be27b   docker.elastic.co/kibana/kibana:7.10.1                 "/usr/local/bin/dumb…"   4 days ago   Up 4 days               0.0.0.0:5601->5601/tcp       kib01
f32b032920a5   golang:1.15                                            "go run cmd/redfishr…"   4 days ago   Up 4 days                                            pt_redfishreadapp_1
c8cbc41afac1   golang:1.15                                            "go run cmd/simpleau…"   4 days ago   Up 4 days                                            pt_authapp_1
9be0793f7696   golang:1.15                                            "go run cmd/elkpump/…"   4 days ago   Up 4 days                                            pt_elasticsearchpump_1
15728d9127b3   golang:1.15                                            "go run cmd/simpledi…"   4 days ago   Up 4 days                                            pt_discapp_1
dd960ad1ab69   docker.elastic.co/elasticsearch/elasticsearch:7.10.1   "/tini -- /usr/local…"   4 days ago   Up 4 days (healthy)     9200/tcp, 9300/tcp           es01
51ea4f767856   docker.elastic.co/elasticsearch/elasticsearch:7.10.1   "/tini -- /usr/local…"   4 days ago   Up 4 days               9200/tcp, 9300/tcp           es02
46586fc1d9f4   docker.elastic.co/elasticsearch/elasticsearch:7.10.1   "/tini -- /usr/local…"   4 days ago   Up 4 days               9200/tcp, 9300/tcp           es03
83d94ddc4580   rmohr/activemq:5.10.0                                  "/bin/bash -c 'bin/a…"   4 days ago   Up 4 days               8161/tcp, 61616/tcp          activemq
```

Figure 1: Docker output. Source: Principled Technologies.

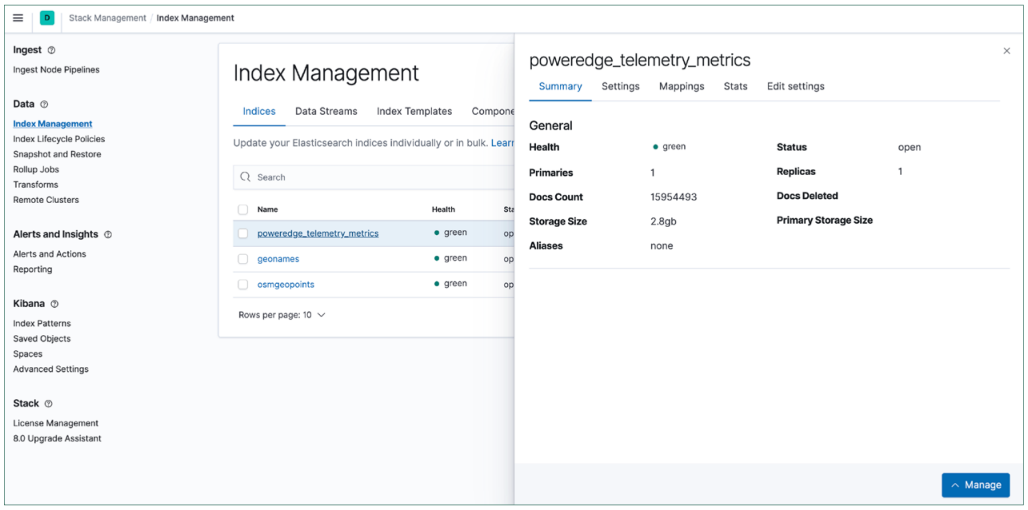
Creating an index pattern in Kibana to query index

In order to search an index, Kibana requires you to create an index pattern. Index patterns allow you to define properties on the fields from the selected data.

1. In a web browser, enter the address for your Kibana container:

   ```
   <IP_ADDRESS>:5601
   ```

2. In the top-left, to expand the options menu, click the hamburger symbol.
3. Scroll down to the Management section, and click Stack Management.
4. On the left, under Data, click Index Management. You should see an index named poweredge_telemetry_metrics with green health, the status set to open, and multiple documents.



Update your Elasticsearch indices individually or in bulk. Learn

Name	Health	Sta
poweredge_telemetry_metrics	● green	op
geonames	● green	op
osmgeopoints	● green	op

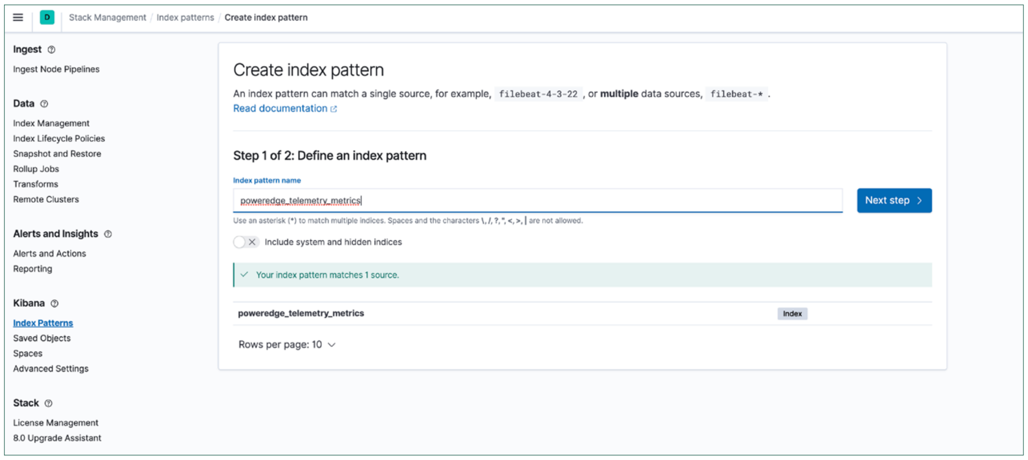
General

Health	● green	Status	open
Primaries	1	Replicas	1
Docs Count	15954493	Docs Deleted	
Storage Size	2.8gb	Primary Storage Size	
Aliases	none		

Figure 2: Index Management within Kibana. Source: Principled Technologies.

5. After verifying that the index exists, you will need to create an index pattern with which to query the index for information for visualizations. To do this, click Index Patterns under Kibana.

6. Click Create index pattern.
7. Type the name of the index into the index pattern name followed by an asterisk (*) to match all indices with that name. (An asterisk is not necessary if you are only using a single index.)



Stack Management / Index patterns / Create index pattern

Ingest
Ingest Node Pipelines

Data
Index Management
Index Lifecycle Policies
Snapshot and Restore
Rollup Jobs
Transforms
Remote Clusters

Alerts and Insights
Alerts and Actions
Reporting

Kibana
Index Patterns
Saved Objects
Spaces
Advanced Settings

Stack
License Management
8.0 Upgrade Assistant

Create index pattern

An index pattern can match a single source, for example, `filebeat-4-3-22` , or **multiple** data sources, `filebeat-*` .
Read documentation

Step 1 of 2: Define an index pattern

Index pattern name: poweredge_telemetry_metrics

Next step

Use an asterisk (*) to match multiple indices. Spaces and the characters \, /, ?, ", <, >, | are not allowed.

Include system and hidden indices

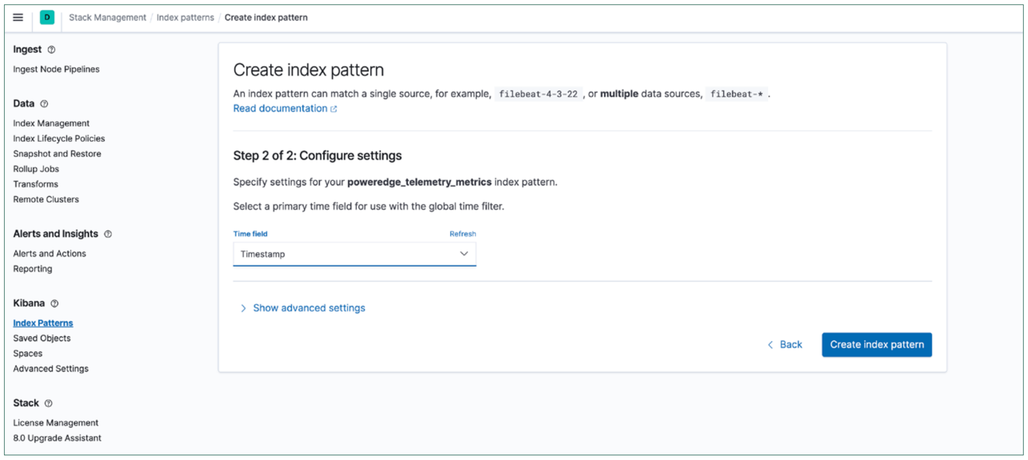
Your index pattern matches 1 source.

poweredge_telemetry_metrics	Index

Rows per page: 10

Figure 3: Index pattern definition within Kibana. Source: Principled Technologies.

8. Click Next step.
9. Under Time field, from the drop-down menu, select Timestamp. Click Create index pattern.



Stack Management / Index patterns / Create index pattern

Ingest
Ingest Node Pipelines

Data
Index Management
Index Lifecycle Policies
Snapshot and Restore
Rollup Jobs
Transforms
Remote Clusters

Alerts and Insights
Alerts and Actions
Reporting

Kibana
Index Patterns
Saved Objects
Spaces
Advanced Settings

Stack
License Management
8.0 Upgrade Assistant

Create index pattern

An index pattern can match a single source, for example, `filebeat-4-3-22` , or **multiple** data sources, `filebeat-*` .
Read documentation

Step 2 of 2: Configure settings

Specify settings for your **poweredge_telemetry_metrics** index pattern.

Select a primary time field for use with the global time filter.

Time field: Timestamp (Refresh)

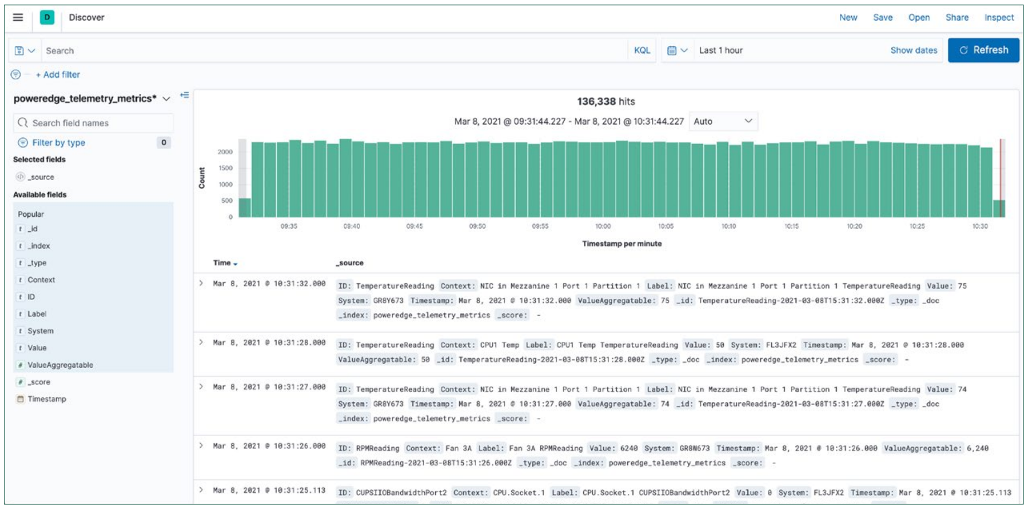
Show advanced settings

Back | Create index pattern

Figure 4: Index pattern configuration within Kibana. Source: Principled Technologies.

10. Expand the left hand menu. Under the Kibana section, click Discover.

11. Expand the time range to 1 hour.



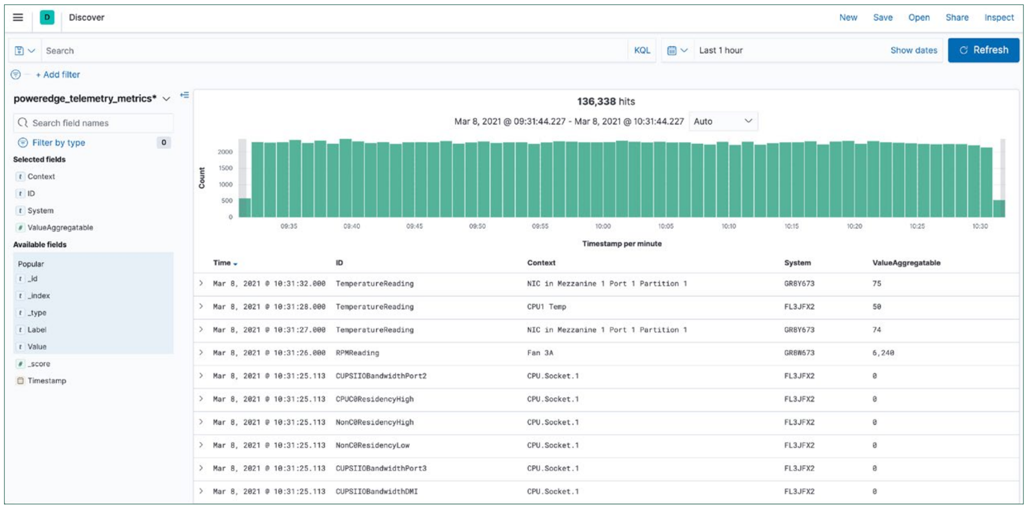
```
Time    _source
> Mar 8, 2021 @ 10:31:32.000  ID: TemperatureReading Context: NIC in Mezzanine 1 Port 1 Partition 1 Label: NIC in Mezzanine 1 Port 1 Partition 1 TemperatureReading Value: 75
                              System: GR8Y673 Timestamp: Mar 8, 2021 @ 10:31:32.000 ValueAggregatable: 75 _id: TemperatureReading-2021-03-08T15:31:32.000Z _type: _doc
                              _index: poweredge_telemetry_metrics _score: -
> Mar 8, 2021 @ 10:31:28.000  ID: TemperatureReading Context: CPU1 Temp Label: CPU1 Temp TemperatureReading Value: 50 System: FL3JFX2 Timestamp: Mar 8, 2021 @ 10:31:28.000
                              ValueAggregatable: 50 _id: TemperatureReading-2021-03-08T15:31:28.000Z _type: _doc _index: poweredge_telemetry_metrics _score: -
> Mar 8, 2021 @ 10:31:27.000  ID: TemperatureReading Context: NIC in Mezzanine 1 Port 1 Partition 1 Label: NIC in Mezzanine 1 Port 1 Partition 1 TemperatureReading Value: 74
                              System: GR8Y673 Timestamp: Mar 8, 2021 @ 10:31:27.000 ValueAggregatable: 74 _id: TemperatureReading-2021-03-08T15:31:27.000Z _type: _doc
                              _index: poweredge_telemetry_metrics _score: -
> Mar 8, 2021 @ 10:31:26.000  ID: RPMReading Context: Fan 3A Label: Fan 3A RPMReading Value: 6240 System: GR8W673 Timestamp: Mar 8, 2021 @ 10:31:26.000 ValueAggregatable: 6,240
                              _id: RPMReading-2021-03-08T15:31:26.000Z _type: _doc _index: poweredge_telemetry_metrics _score: -
> Mar 8, 2021 @ 10:31:25.113  ID: CUPSIIOBandwidthPort2 Context: CPU.Socket.1 Label: CPU.Socket.1 CUPSIIOBandwidthPort2 Value: 0 System: FL3JFX2 Timestamp: Mar 8, 2021 @ 10:31:25.113
```

Figure 5: Kibana data discovery page with no filters applied. Source: Principled Technologies.

12. Verify that the data displayed include a timestamp and data from _source.
13. Click the plus signs next to the following fields to add them:
    a. Context
    b. ID
    c. System
    d. ValueAggregatable

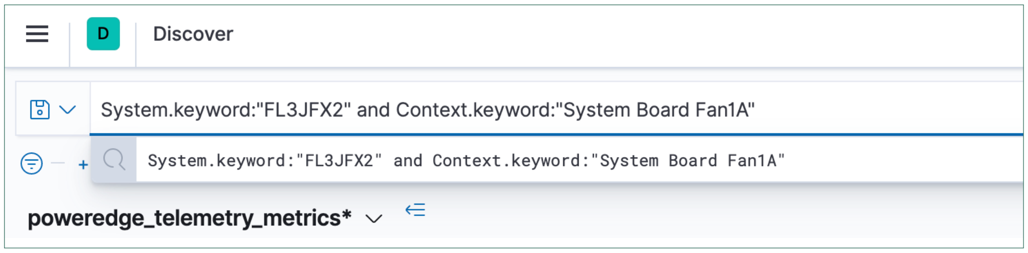
Now that your data is in a more readable format, you can use the search bar and Kibana Query Language to query the index for specific results. Below is an example query you could use to see the RPM of a particular fan in your server:

```
System.keyword:"<IDRAC SERVICE TAG>" and Context.keyword:"System Board Fan1A"
```



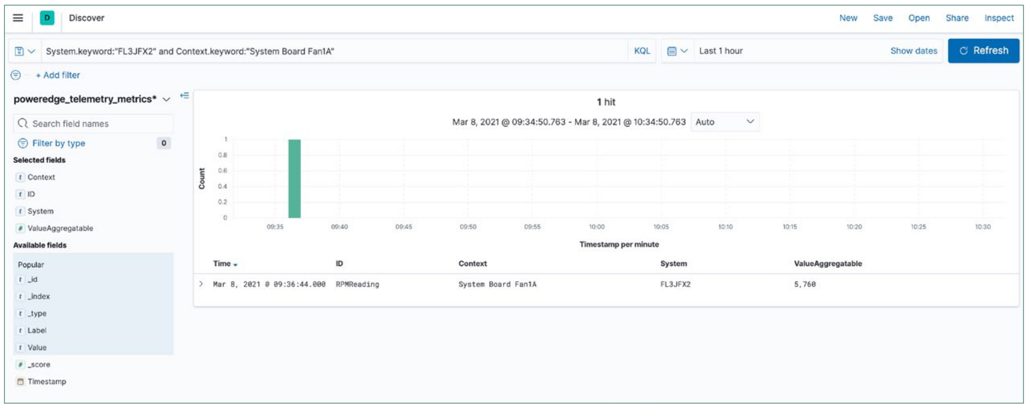
Time	ID	Context	System	ValueAggregatable
Mar 8, 2021 @ 10:31:32.000	TemperatureReading	NIC in Mezzanine 1 Port 1 Partition 1	GR8Y673	75
Mar 8, 2021 @ 10:31:28.000	TemperatureReading	CPU1 Temp	FL3JFX2	50
Mar 8, 2021 @ 10:31:27.000	TemperatureReading	NIC in Mezzanine 1 Port 1 Partition 1	GR8Y673	74
Mar 8, 2021 @ 10:31:26.000	RPMReading	Fan 3A	GR8W673	6,240
Mar 8, 2021 @ 10:31:25.113	CUPSIIOBandwidthPort2	CPU.Socket.1	FL3JFX2	0
Mar 8, 2021 @ 10:31:25.113	CPUC0ResidencyHigh	CPU.Socket.1	FL3JFX2	0
Mar 8, 2021 @ 10:31:25.113	NonC0ResidencyHigh	CPU.Socket.1	FL3JFX2	0
Mar 8, 2021 @ 10:31:25.113	NonC0ResidencyLow	CPU.Socket.1	FL3JFX2	0
Mar 8, 2021 @ 10:31:25.113	CUPSIIOBandwidthPort3	CPU.Socket.1	FL3JFX2	0
Mar 8, 2021 @ 10:31:25.113	CUPSIIOBandwidthDMI	CPU.Socket.1	FL3JFX2	0

Figure 6: Kibana data discovery page with field filters. Source: Principled Technologies.



```
System.keyword:"FL3JFX2" and Context.keyword:"System Board Fan1A"
```

Figure 7: A basic query within Kibana. Source: Principled Technologies.



Time	ID	Context	System	ValueAggregatable
Mar 8, 2021 @ 09:36:44.000	RPMReading	System Board Fan1A	FL3JFX2	5,760

Figure 8: The resulting visualization based on the query in Figure 7. Source: Principled Technologies.

To save a query, click the Save drop-down menu beside the search bar and click Save current query. Give the query a name and description, if necessary, and click Save.

You can also use filters to further carve up the search results. To do so, click the Add filter button under the search bar. Select a field and an operator. If a value is required, enter the value, and click Save.

Creating visualizations and dashboards with Kibana

Having the capability to ingest and index data is great, but it's more useful to perform analysis on data and create visualizations from it. With Kibana, you can create visualizations specific to key areas you wish to monitor and group these with dashboards. This not only reduces the time required to evaluate the status of your hardware or software; it also provides you with insight into potential problem areas.

1. In the left-hand pane, under the Kibana section, select Dashboard.

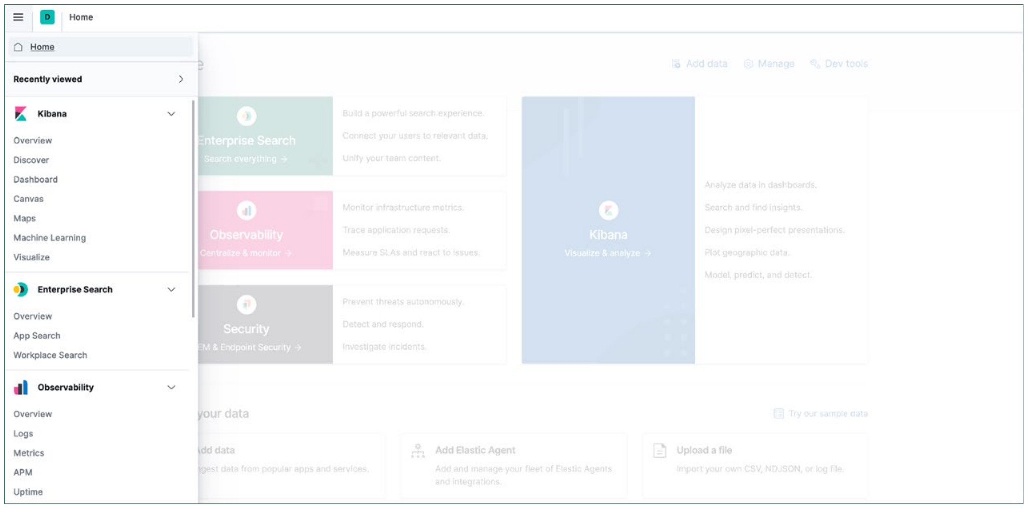


Figure 9: The Kibana navigation pane. Source: Principled Technologies.

2. Click Create dashboard.
3. To create a new visualization for the dashboard, click Create new.
4. Select the type of visualization you want. If you are new to Kibana, the Lens visualization type will be the easiest to start with.

Covering the total number of options available for creating visualizations in Kibana is outside the scope of this document. You should experiment with all of the different query and filter options as well as the different visualization types to create the most appropriate visualizations for your purposes. Below is an example of how to create a basic visualization of the average RPM of a single fan with Lens.

1. Drag and drop the ValueAggregatable field from the left-hand side to the middle section to begin. The X-axis should automatically populate with Timestamp, and the Y-axis should be Average of ValueAggregatable.
2. On the right, click the value under Y-axis.
3. Change the display name to Average Fan RPM, and click Close.
4. To filter for a specific fan, we need to add a field under Break down by section on the right side of the screen. Click Drop a field or click to add.
5. Select Filters.
6. Click the field that says "= All records" to query the index.
7. We will use the same query that we saved from earlier to get the average fan speed from the index for Fan 1A.

```
System.keyword:"<IDRAC SERVICE TAG>" and Context.keyword:"System Board Fan1A"
```

8. Give the query a label. For example, `Fan 1A`
9. To add another fan to the visualization, add another filter, and repeat the query, replacing 1A with 1B.
10. To change the chart type, click the drop-down menu, or select one of the suggestions that Kibana provides below the chart.
11. When you are finished with the visualization, to save the visualization, click Save.

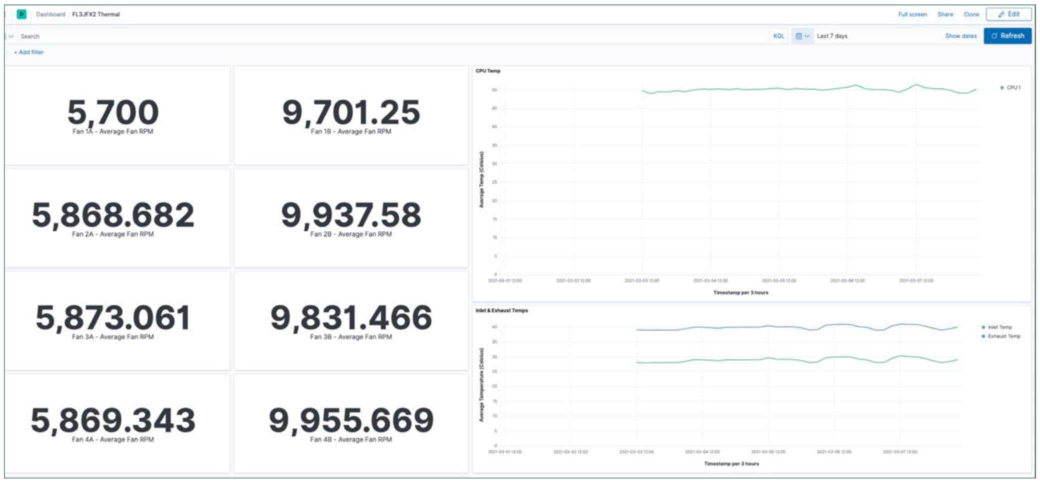


Figure 10: Visualization of thermal telemetry data within Kibana. Source: Principled Technologies.

12. Give the visualization a title and description, select Add to Dashboard after saving, and click Save and return.
13. Repeat the visualization creation process as many times as necessary. To save your dashboard, in the top-right, click Save.
14. Give the dashboard a title and description, and click Save.

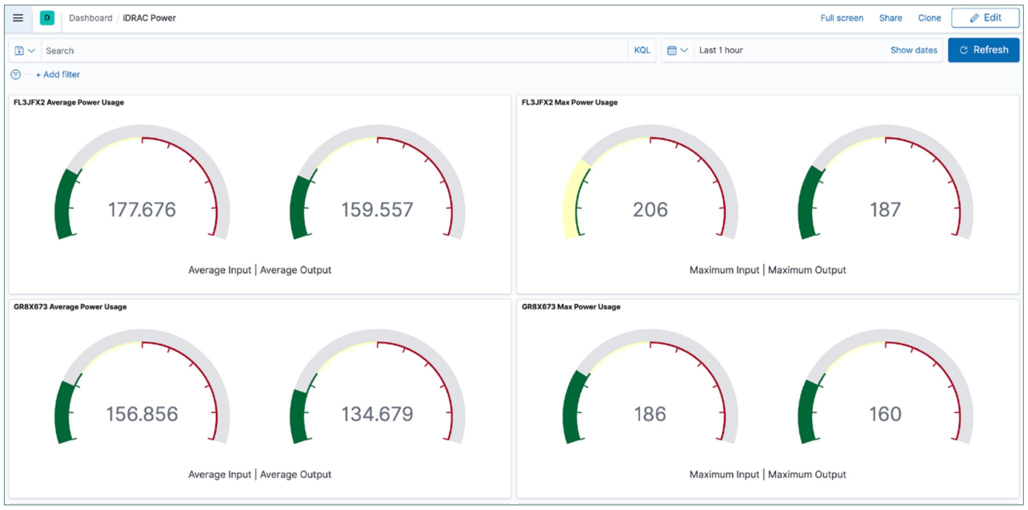


Figure 11: An example of a Kibana dashboard using power usage telemetry data. Source: Principled Technologies.

Running the Rally benchmark

In order to demonstrate the capability of the iDRAC9 Telemetry reference toolset to continue functioning under load with the support of the Dell EMC PowerEdge R6515 server powered by a 3rd Gen AMD EPYC processor, we produced a heavy load on the Elastic Stack and CPU by running the Rally benchmark in containers with the same track running in two terminals simultaneously. While the benchmark was running, CPU utilization was at or above 90 percent, and the iDRAC9 Telemetry reference toolset continued to push data to the Elastic Stack without interruption. We used the following steps to run Rally:

1. On the server under test, open a terminal window.
2. Create two directories, one for each Rally track:

```
mkdir rally1
mkdir rally2
```

3. To add the Rally paths to your environment, run the following commands:

```
echo "RALLY1=[Path to rally1]" > sudo tee -a .profile
echo "RALLY2=[Path to rally2]" > sudo tee -a .profile
```

4. To set the new environment variables, source the profile:

```
source .profile
```

5. Open a second terminal window.
6. In the first terminal, type the following (but don't start the test yet):

```
docker run --network=pt_elastic --rm -v $RALLY:/rally/.rally elastic/rally --track=nyc_taxis
--pipeline=benchmark-only --target-hosts=es01:9200
```

7. In the second terminal, type the following (but don't start the test yet):

```
docker run --network=pt_elastic --rm -v $RALLY:/rally/.rally elastic/rally --track=nyc_taxis
--pipeline=benchmark-only --target-hosts=es01:9200
```

8. Once you have both commands queued and ready to go, hit the return key in each terminal to start the Rally benchmarks.

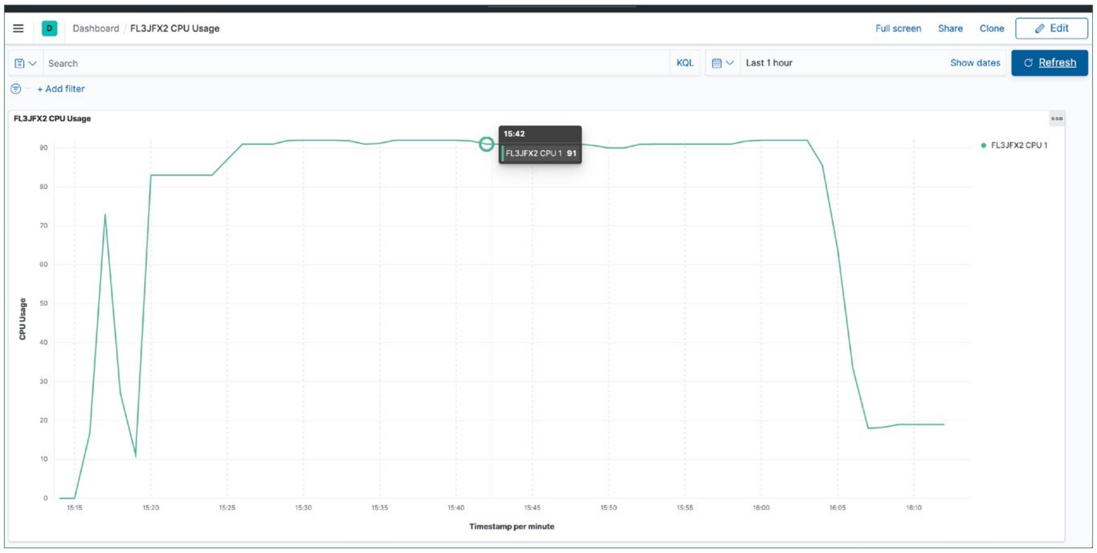
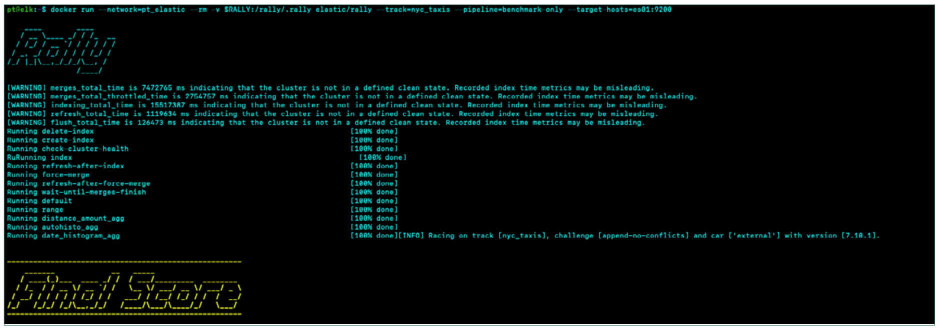


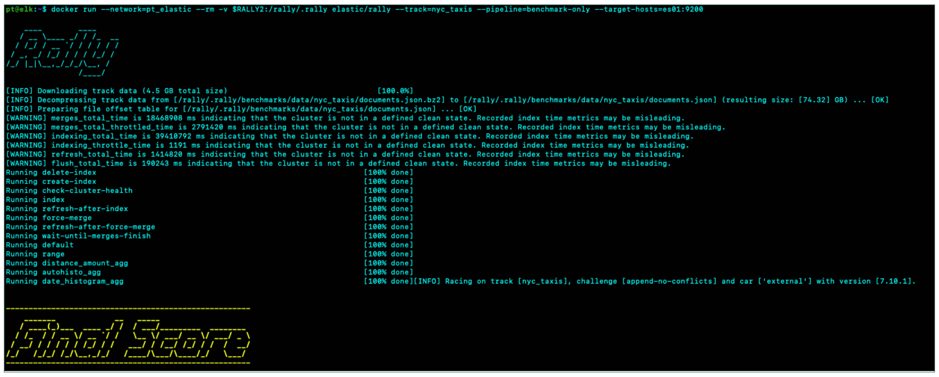
Figure 12: Elastic Stack CPU dashboard during the Rally benchmark test. Source: Principled Technologies.



```
pt@elk:~$ docker run --network=pt_elastic --rm -v $RALLY:/rally/.rally elastic/rally --track=nyc_taxis --pipeline=benchmark-only --target-hosts=es01:9200

[WARNING] merges_total_time is 7472765 ms indicating that the cluster is not in a defined clean state. Recorded index time metrics may be misleading.
[WARNING] merges_total_throttled_time is 2754757 ms indicating that the cluster is not in a defined clean state. Recorded index time metrics may be misleading.
[WARNING] indexing_total_time is 15517387 ms indicating that the cluster is not in a defined clean state. Recorded index time metrics may be misleading.
[WARNING] refresh_total_time is 1119634 ms indicating that the cluster is not in a defined clean state. Recorded index time metrics may be misleading.
[WARNING] flush_total_time is 126473 ms indicating that the cluster is not in a defined clean state. Recorded index time metrics may be misleading.
Running delete-index                                                      [100% done]
Running create-index                                                      [100% done]
Running check-cluster-health                                              [100% done]
RuRunning index                                                             [100% done]
Running refresh-after-index                                               [100% done]
Running force-merge                                                       [100% done]
Running refresh-after-force-merge                                         [100% done]
Running wait-until-merges-finish                                          [100% done]
Running default                                                           [100% done]
Running range                                                             [100% done]
Running distance_amount_agg                                               [100% done]
Running autohisto_agg                                                     [100% done]
Running date_histogram_agg                                                [100% done][INFO] Racing on track [nyc_taxis], challenge [append-no-conflicts] and car ['external'] with version [7.10.1].
```

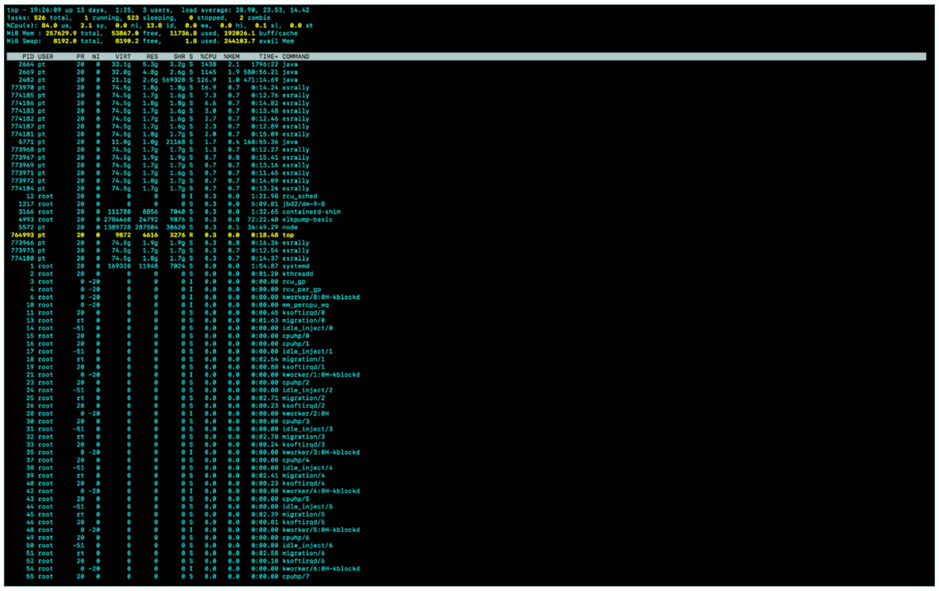
Figure 13: Rally benchmarking tool. Source: Principled Technologies.



```
pt@elk:~$ docker run --network=pt_elastic --rm -v $RALLY2:/rally/.rally elastic/rally --track=nyc_taxis --pipeline=benchmark-only --target-hosts=es01:9200

[INFO] Downloading track data (4.5 GB total size)                          [100.0%]
[INFO] Decompressing track data from [/rally/.rally/benchmarks/data/nyc_taxis/documents.json.bz2] to [/rally/.rally/benchmarks/data/nyc_taxis/documents.json] (resulting size: [74.32] GB) ... [OK]
[INFO] Preparing file offset table for [/rally/.rally/benchmarks/data/nyc_taxis/documents.json] ... [OK]
[WARNING] merges_total_time is 18468908 ms indicating that the cluster is not in a defined clean state. Recorded index time metrics may be misleading.
[WARNING] merges_total_throttled_time is 2791420 ms indicating that the cluster is not in a defined clean state. Recorded index time metrics may be misleading.
[WARNING] indexing_total_time is 39410792 ms indicating that the cluster is not in a defined clean state. Recorded index time metrics may be misleading.
[WARNING] indexing_throttle_time is 1191 ms indicating that the cluster is not in a defined clean state. Recorded index time metrics may be misleading.
[WARNING] refresh_total_time is 1414820 ms indicating that the cluster is not in a defined clean state. Recorded index time metrics may be misleading.
[WARNING] flush_total_time is 190243 ms indicating that the cluster is not in a defined clean state. Recorded index time metrics may be misleading.
Running delete-index                                                      [100% done]
Running create-index                                                      [100% done]
Running check-cluster-health                                              [100% done]
Running index                                                             [100% done]
Running refresh-after-index                                               [100% done]
Running force-merge                                                       [100% done]
Running refresh-after-force-merge                                         [100% done]
Running wait-until-merges-finish                                          [100% done]
Running default                                                           [100% done]
Running range                                                             [100% done]
Running distance_amount_agg                                               [100% done]
Running autohisto_agg                                                     [100% done]
Running date_histogram_agg                                                [100% done][INFO] Racing on track [nyc_taxis], challenge [append-no-conflicts] and car ['external'] with version [7.10.1].
```

Figure 14: Rally benchmarking tool. Source: Principled Technologies.



```
top - 19:26:09 up 13 days,  1:25,  3 users,  load average: 28.90, 23.53, 14.42
Tasks: 526 total,   1 running, 523 sleeping,   0 stopped,   2 zombie
%Cpu(s): 84.0 us,  2.1 sy,  0.0 ni, 13.8 id,  0.0 wa,  0.0 hi,  0.1 si,  0.0 st
MiB Mem : 257629.9 total,  53867.0 free,  11736.8 used, 192026.1 buff/cache
MiB Swap:   8192.0 total,   8190.2 free,      1.8 used. 244103.7 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   2664 pt        20   0   33.1g   5.3g   3.2g S  1438   2.1   1796:22 java
   2669 pt        20   0   32.0g   4.8g   2.6g S  1145   1.9 580:56.21 java
   2482 pt        20   0   21.1g   2.6g 569328 S 126.9   1.0 471:14.69 java
 773970 pt        20   0   74.5g   1.8g   1.8g S  16.9   0.7   0:14.24 esrally
 774185 pt        20   0   74.5g   1.7g   1.6g S   7.3   0.7   0:12.76 esrally
 774186 pt        20   0   74.5g   1.8g   1.8g S   6.6   0.7   0:14.82 esrally
 774183 pt        20   0   74.5g   1.7g   1.6g S   3.0   0.7   0:13.48 esrally
 774182 pt        20   0   74.5g   1.7g   1.6g S   2.7   0.7   0:12.46 esrally
 774187 pt        20   0   74.5g   1.7g   1.6g S   2.3   0.7   0:12.89 esrally
 774181 pt        20   0   74.5g   1.8g   1.7g S   2.0   0.7   0:15.09 esrally
   5771 pt        20   0   11.0g   1.0g  21168 S   1.7   0.4 168:55.36 java
 773968 pt        20   0   74.5g   1.7g   1.7g S   1.3   0.7   0:12.27 esrally
 773967 pt        20   0   74.5g   1.9g   1.9g S   0.7   0.8   0:15.41 esrally
 773969 pt        20   0   74.5g   1.7g   1.7g S   0.7   0.7   0:13.16 esrally
 773971 pt        20   0   74.5g   1.7g   1.6g S   0.7   0.7   0:11.45 esrally
 773972 pt        20   0   74.5g   1.8g   1.7g S   0.7   0.7   0:14.09 esrally
 774184 pt        20   0   74.5g   1.7g   1.7g S   0.7   0.7   0:13.26 esrally
     12 root      20   0       0      0      0 I   0.3   0.0   1:21.98 rcu_sched
   1217 root      20   0       0      0      0 S   0.3   0.0   5:09.81 jbd2/dm-9-8
   3166 root      20   0  111780   8856   7040 S   0.3   0.0   1:32.65 containerd-shim
   4993 root      20   0 2784460  24792   9876 S   0.3   0.0  72:22.40 elkpump-basic
   5572 pt        20   0 1389728 287504  30620 S   0.3   0.1  36:49.29 node
 764993 pt        20   0    9872   4616   3276 R   0.3   0.0   0:18.48 top
 773966 pt        20   0   74.5g   1.9g   1.9g S   0.3   0.8   0:16.36 esrally
 773973 pt        20   0   74.5g   1.7g   1.7g S   0.3   0.7   0:12.54 esrally
 774180 pt        20   0   74.5g   1.8g   1.7g S   0.3   0.7   0:14.37 esrally
      1 root      20   0  169320  11948   7024 S   0.0   0.0   1:54.87 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:01.20 kthreadd
      3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
      4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_par_gp
      6 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/0:0H-kblockd
     10 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 mm_percpu_wq
     11 root      20   0       0      0      0 S   0.0   0.0   0:00.45 ksoftirqd/0
     13 root      rt   0       0      0      0 S   0.0   0.0   0:01.63 migration/0
     14 root     -51   0       0      0      0 S   0.0   0.0   0:00.00 idle_inject/0
     15 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/0
     16 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/1
     17 root     -51   0       0      0      0 S   0.0   0.0   0:00.00 idle_inject/1
     18 root      rt   0       0      0      0 S   0.0   0.0   0:02.54 migration/1
     19 root      20   0       0      0      0 S   0.0   0.0   0:00.88 ksoftirqd/1
     21 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/1:0H-kblockd
     23 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/2
     24 root     -51   0       0      0      0 S   0.0   0.0   0:00.00 idle_inject/2
     25 root      rt   0       0      0      0 S   0.0   0.0   0:02.71 migration/2
     26 root      20   0       0      0      0 S   0.0   0.0   0:00.23 ksoftirqd/2
     28 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/2:0H
     30 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/3
     31 root     -51   0       0      0      0 S   0.0   0.0   0:00.00 idle_inject/3
     32 root      rt   0       0      0      0 S   0.0   0.0   0:02.78 migration/3
     33 root      20   0       0      0      0 S   0.0   0.0   0:00.24 ksoftirqd/3
     35 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/3:0H-kblockd
     37 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/4
     38 root     -51   0       0      0      0 S   0.0   0.0   0:00.00 idle_inject/4
     39 root      rt   0       0      0      0 S   0.0   0.0   0:02.41 migration/4
     40 root      20   0       0      0      0 S   0.0   0.0   0:00.23 ksoftirqd/4
     42 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/4:0H-kblockd
     43 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/5
     44 root     -51   0       0      0      0 S   0.0   0.0   0:00.00 idle_inject/5
     45 root      rt   0       0      0      0 S   0.0   0.0   0:02.39 migration/5
     46 root      20   0       0      0      0 S   0.0   0.0   0:00.81 ksoftirqd/5
     48 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/5:0H-kblockd
     49 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/6
     50 root     -51   0       0      0      0 S   0.0   0.0   0:00.00 idle_inject/6
     51 root      rt   0       0      0      0 S   0.0   0.0   0:02.58 migration/6
     52 root      20   0       0      0      0 S   0.0   0.0   0:00.18 ksoftirqd/6
     54 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/6:0H-kblockd
     55 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/7
```

Figure 15: Top task manager output during the Rally benchmark test. Source: Principled Technologies.

Custom reporting

Dell has added the ability to create custom report definitions for iDRAC9 telemetry data. With this new capability, data center technicians can use the Redfish API to create, edit, and delete custom reports. These custom reports allow users to set different metric values, collection intervals, and collection functions that are unique to their organization's interests or concerns. For example, your technician could create a report that collects relevant thermal metrics via temperature and fan speed readings. We were able to successfully upload a custom report following the steps below:

1. Create a JSON document with the relevant fields as in the example below.

```
{
    "Id": "TemperatureAndRPMReading",
    "Name": "All Temperature and RPM Readings",
    "Description": "A report with all relevant 
thermal metrics - Temperature reading and FAN 
RPM readings",
    "MetricReportDefinitionEnabled": true,
    "MetricReportDefinitionType": "Periodic",
    "MetricReportHeartbeatInterval": 
"PT0H0M0S",
    "SuppressRepeatedMetricValue": false,
    "ReportTimespan": "PT0H1M0S",
    "ReportUpdates": "Overwrite",
    "Schedule": {
        "RecurrenceInterval": "PT0H1M0S"
    },
    "Metrics": [
            {"MetricId": "TemperatureReading"},
            {"MetricId": "RPMReading"}
    ],
    "Metrics@odata.count": 2
}
```

2. To create the custom report, run the following command:

```
curl -s -k -u <username>:<password> 
-X POST https://<iDRACIP>/redfish/v1/
TelemetryService/MetricReportDefinitions -H 
'Content-Type: application/json' -d  @<custom 
file path>
```

Appendix: Reference files

iDRAC9 configuration file

Below are the full contents of the config.ini file we used when setting up the iDRAC9 Telemetry reference toolset.

config.ini

```
[General]
StompHost=activemq
StompPort=61613



[Services]
Types=iDRAC,iDRAC,iDRAC,iDRAC,iDRAC
IPs=IDRAC_IP01,IDRAC_IP02,IDRAC_IP03,IDRAC_IP04,IDRAC_IP05

[IDRAC_IP01]
username=<USERNAME>
password=<PASSWORD>

[IDRAC_IP02]
username=<USERNAME>
password=<PASSWORD>

[IDRAC_IP03]
username=<USERNAME>
password=<PASSWORD>

[IDRAC_IP04]
username=<USERNAME>
password=<PASSWORD>

[IDRAC_IP05]
username=<USERNAME>
password=<PASSWORD>
```

Docker-Compose reference file

Below are the contents of the file we used when setting up Docker-Compose.

elk.yml

```
version: '3.5'

services:

  es01:
    image: docker.elastic.co/elasticsearch/elasticsearch:7.10.1
    container_name: es01
    environment:
      - node.name=es01
      - cluster.name=es-docker-cluster
      - discovery.seed_hosts=es02,es03
      - cluster.initial_master_nodes=es01,es02,es03
      - bootstrap.memory_lock=true
    ulimits:
      memlock:
        soft: -1
        hard: -1
    volumes:
      - <PATH TO ES01 DATA>:/usr/share/elasticsearch/data
    networks:
      - elastic

    healthcheck:
      test: curl http://localhost:9200 >/dev/null; if [[ $$? == 52 ]]; then echo 0; else echo 1; fi
      interval: 30s
      timeout: 10s
      retries: 5

  es02:
    image: docker.elastic.co/elasticsearch/elasticsearch:7.10.1
    container_name: es02
    environment:
      - node.name=es02
      - cluster.name=es-docker-cluster
      - discovery.seed_hosts=es01,es03
      - cluster.initial_master_nodes=es01,es02,es03
      - bootstrap.memory_lock=true
    ulimits:
      memlock:
        soft: -1
        hard: -1
    volumes:
      - <PATH TO ES02 DATA>:/usr/share/elasticsearch/data
    networks:
      - elastic

  es03:
    image: docker.elastic.co/elasticsearch/elasticsearch:7.10.1
    container_name: es03
    environment:
      - node.name=es03
      - cluster.name=es-docker-cluster
      - discovery.seed_hosts=es01,es02
      - cluster.initial_master_nodes=es01,es02,es03
      - bootstrap.memory_lock=true
    ulimits:
      memlock:
        soft: -1
        hard: -1
    volumes:
      - <PATH TO ES03 DATA>:/usr/share/elasticsearch/data
    networks:
      - elastic

  kib01:
    image: docker.elastic.co/kibana/kibana:7.10.1
    container_name: kib01
    depends_on: {"es01": {"condition": "service_healthy"}}
    environment:
      ElasticSEARCH_URL: http://es01:9200
      ElasticSEARCH_HOSTS: http://es01:9200
    ports:
      - 5601:5601
    networks:
      - elastic

  activemq:
    image: rmohr/activemq:5.10.0
    container_name: activemq
    hostname: activemq
    working_dir:
      /opt/activemq/
    networks:
      - elastic

  discapp:
    image: golang:1.15
    volumes:
      - <PATH TO iDRAC-Telemetry-Reference-Tools>:/go/src/github.com/telemetryservice
    working_dir:
      /go/src/github.com/telemetryservice
    command:
      go run cmd/simpledisc/simpledisc.go
    networks:
      - elastic
    depends_on:
      - "activemq"

  authapp:
    image: golang:1.15
    volumes:
      - <PATH TO iDRAC-Telemetry-Reference-Tools>:/go/src/github.com/telemetryservice
    working_dir:
      /go/src/github.com/telemetryservice
    command:
      go run cmd/simpleauth/simpleauth.go
    networks:
      - elastic
    depends_on:
      - "activemq"

  redfishreadapp:
    image: golang:1.15
    volumes:
      - <PATH TO iDRAC-Telemetry-Reference-Tools>:/go/src/github.com/telemetryservice
    working_dir:
      /go/src/github.com/telemetryservice
    command:
      go run cmd/redfishread/redfishread.go
    networks:
      - elastic
    depends_on:
      - "activemq"
  elasticsearchpump:
    image: golang:1.15
    volumes:
      - <PATH TO iDRAC-Telemetry-Reference-Tools>:/go/src/github.com/telemetryservice
    working_dir:
      /go/src/github.com/telemetryservice
    command:
      go run cmd/Elasticpump/Elasticpump-basic.go
    networks:
      - elastic
    depends_on: {"es01": {"condition": "service_healthy"}}
    depends_on:
      - "activemq"
    environment:
      ElasticSEARCH_URL: http://es01:9200


networks:
  elastic:
    driver: bridge
```

Appendix: System configuration information

Table 1: Detailed information on the system we tested.

System configuration information	Dell EMC PowerEdge R6515
BIOS name and version	Dell 2.0.3
Non-default BIOS settings	N/A
Operating system name and version/build number	Ubuntu 20.04.1 LTS
Date of last OS updates/patches applied	02/27/2021
Power management policy	System Profile=Performance
Processor	
Number of processors	1
Vendor and model	AMD EPYC 7313
Core count (per processor)	16
Core frequency (GHz)	3.0
Stepping	Model 1 Stepping 1
Memory module(s)	
Total memory in system (GB)	256
Number of memory modules	16
Vendor and model	Hynix HMA82GR7CJR8N-XN
Size (GB)	16
Type	DDR-4
Speed (MHz)	2,933
Speed running in the server (MHz)	2,933
Storage controller	
Vendor and model	PERC H730p Mini
Cache size (GB)	2 GB
Firmware version	25.5.7.0005
Local storage 1	
Number of drives	1
Drive vendor and model	Micron MTFDDAK240TCB
Drive size (GB)	240
Drive information (speed, interface, type)	6Gbps, SATA, SSD

System configuration information	Dell EMC PowerEdge R6515
Local storage 2	
Number of drives	6
Drive vendor and model	Dell Express Flash PM1725b
Drive size (GB)	1,600
Drive information (interface and type)	NVMe SSD
Network adapter	
Vendor and model	Broadcom BCM5720
Number and type of ports	2 x 1Gb
Driver version	21.60.8
Cooling fans	
Vendor and model	Dell 16W7D-A01
Number of cooling fans	6
Power supplies	
Vendor and model	Dell 06V43GA00
Number of power supplies	2
Wattage of each (W)	550

1 Dell EMC, "Transforming Datacenter Analytics with iDRAC9 Telemetry Streaming," accessed March 15, 2021, https://www.delltechnologies.com/resources/en-us/asset/white-papers/products/software/direct-from-development-datacenter-telemetry.pdf.

This project was commissioned by Dell Technologies.