Abstract
Grafana is a powerful tool to visualize time-series performance data. This technical report describes how to build a fully customizable Grafana instance to visualize performance statistics for NetApp® SolidFire®, VMware, and NetApp HCI systems.

This solution is completely open source. IT uses Grafana for graphing performance data, Docker for containerizing the applications, Trident for Docker plugin for persistent storage of metrics and container state, and Graphite for storing the time-series data.
1 Overview

Effective monitoring of critical infrastructure is the keystone in maintaining operational readiness and is a key enabler of risk mitigation. Monitoring critical infrastructure for atypical workloads and events can help prevent small issues from becoming outages.

NetApp® HCI includes access to the NetApp Active IQ® platform, which is a cloud-based performance monitoring and alerting platform. Active IQ provides a rich set of preconfigured dashboards that displays real-time cluster performance and alerts and that presents a view of historical data (Figure 1).

Figure 1) Active IQ dashboard.

To aid with future planning and budgeting, Active IQ also enables capacity modeling and forecasts as to when additional capacity should be added to the cluster (Figure 2).
Active IQ is the preferred method for monitoring and alerting for NetApp HCI and SolidFire® systems. However, for Active IQ to function, it requires outbound connectivity from your site to the cloud. Active IQ is not an available option for sites that do not allow outbound connections (dark sites).

The HCICollector is a community open-source project that replicates a subset of the Active IQ functionality in a collection of Docker containers. This functionality can be run on local infrastructure, removing the need for external internet connectivity once configured. The HCI Collector assembles time-series data from both SolidFire and NetApp HCI components, including VMware vSphere, and it presents those metrics through a collection of preconfigured Grafana dashboards.

### 1.1 HCICollector Components

The HCICollector is composed of several individual components. Figure 3 shows how the following components work together:

- **SFCollector.** A Docker container that hosts a Python script that collects performance data from SolidFire systems or NetApp HCI storage nodes and sends the data to a Graphite time-series database.
- **Graphite.** A Docker container that hosts the time-series database for holding performance data that is collected from storage and compute hosts.
- **VMwCollector.** A Docker container that hosts a statistics collector for VMware components, written in Golang.
- **Grafana.** A Docker container that hosts the front end that is used to graphically visualize the time-series data in the Graphite database.
- **Docker host.** An Ubuntu 16.04 LTS virtual machine (VM) that hosts the HCICollector containers. It is assumed the Docker host exists in the environment.
- **Trident.** An (optional) NetApp Docker Volume Plugin that runs in the Docker host that automates the creation and presentation of persistent storage volumes for the containers that constitute the HCICollector. You can use local host volumes as well, but their configuration is not covered in this guide.
1.2 How It All Works Together

As implemented in this guide, the HCICollector functions as follows:

- Some initial setup is required to provide IP addresses and credentials to the components of the collector. This process is partially automated by the included `install_hcicollector.sh` shell script. Note that this configuration expects and requires DNS resolution of VMware vCenter and ESXi hosts.
- The Trident for Docker plugin creates an iSCSI volume on a SolidFire system for storing the Whisper database files for Graphite. The use of an external data source allows portability of the collected data. It also enables protection of the collected data through snapshots, clones, or replication to another SolidFire or NetApp FAS system.
- The Docker host mounts the volume that Trident provides and passes this volume through to the Graphite container for persistent storage of collected metrics. Container management is handled by Docker Compose. Docker Compose handles container configuration and provisions the networks and external volumes that the containers use.
- The SFCollector connects to one or more SolidFire storage back ends and collects statistics every 60 seconds and sends them to Graphite under the `netapp.solidfire` namespace. A list of statistics that are collected is available in Appendix B.
- The VMvCollector connects to one or more vCenter instances and collects statistics every 60 seconds and sends them to Graphite under the `vsphere` namespace. A list of statistics that are collected is available in the `.vmvcollector/vsphere-graphite.json` file in the “Metrics” section.
- Grafana accesses data from the Graphite database and uses it to draw a set of preconfigured dashboards. To aid in setup, the dashboards and data source are automatically configured and populated by using the provisioning functionality that was introduced in Grafana 5.
1.3 Primary Use Cases

The HCICollector can be used in isolation, but the primary use case is to augment the data that Active IQ provides. You can leverage the HCICollector in the following ways:

- Create custom dashboards for visualizing data that is not present in Active IQ.
- Create multisystem reporting dashboards. Currently, Active IQ reporting is performed per system.
- Visualize more complete VMware statistics.
- Extend the collectors to capture additional data from switches or from other infrastructure that is of interest.

2 Installation

This section describes the steps to deploy the HCICollector. It is assumed that a Docker host is available. These instructions use an Ubuntu 16.04 LTS VM as an example. It is also assumed that Docker CE 17.03+ and Docker Compose are installed.

The instructions for installing Docker components are available on https://docs.docker.com.

Note: Docker version 18.03.0 breaks plugins. This problem has been resolved in version 18.03.1. The earlier versions are unaffected. Docker 17.12.1 is used in this guide.

2.1 Preparation

Before you continue, be sure to carry out the following preparation:

- Verify that you have a suitable Docker host machine available and that the machine has a supported version of Docker installed, as well as any support tools that you might require.
- Confirm that DNS records exist for the equipment that you will be monitoring.
- Confirm that you have the following environment information:
  - vCenter host name, fully qualified domain name (FQDN), user name, and password
  - SolidFire management virtual IP address (MVIP address), storage virtual IP address (SVIP address), user name, and password
  - Docker host IP and login information
- Verify that the https://github.com/jedimt/hcicollector repository has been cloned into /opt/github/hcicollector on the Docker host.
- If you use Trident, confirm that the Docker host can connect to the SolidFire SVIP address.

After the repository has been cloned, the following high-level directory structure should be in place:

root@sfps-grafana-dev:/opt/github/hcicollector# tree -d -L 3

  ├── sfcollector        #Container configuration for the SFCollector components
  │   └── grafana        #Container configuration for the Grafana components
  │       ├── dashboards #Contains all the preconfigured dashboards
  │       └── provisioning #Configuration files for Grafana automated provisioning
  │             ├── dashboards #Dashboard provisioning configuration
  │             └── datasources #Datasource provisioning configuration
  └── graphite         #Container configuration for Graphite database
  └── vmmwcollector    #Contain configuration for the vSphere-Graphite collector

Note: This tree view is abbreviated to show the directories of interest.
2.2 HCICollector Installation: Scripted

The HCICollector includes a rudimentary bash install script (install_hcicollector.sh) that performs the following tasks:

1. Prompts the user for required information.
2. Writes the Trident configuration file to /etc/netappdvp/config.json and installs Trident 18.04.
3. Creates the Docker volume for the Graphite database to mount.
4. Writes out the following configuration files:
   - ./docker-compose.yml
   - ./sfcollector/wrapper.sh file for the SolidFire collector
   - ./vmwcollector/vsphere-graphite.json
5. Sets the correct data source for the included dashboards in the ./grafana/dashboards directory.

When you use the script to drive the installation, the workflow is as follows:

1. Create the directory to house the GitHub repository, for example /opt/github/hcicollector.
2. Clone the https://github.com/jedimt/hcicollector GitHub repository into the desired directory.
3. Execute the install_hcicollector.sh script and provide the requested inputs.
4. Start the collector by running docker-compose up -d from the /opt/github/hcicollector directory. Starting the containers for the first time requires about 10 minutes on most systems.

Figure 4) Scripted installation with the install_hcicollector.sh script.

```
root@fpe-grafana-dev-temp:/opt/github/hcicollector$ ./install_hcicollector.sh

Enter the SolidFire management virtual IP (MVIP):
10.193.136.240
Enter the SolidFire storage virtual IP (SVIF):
10.193.139.43
Enter the SolidFire username (case sensitive):
netapp
Enter the SolidFire password:
Enter the tenant account to use for Trident:
docker-dev
Enter the volume name to create for Graphite:
graphite-db-dev
Enter the password to use for the Grafana admin account:
Enter the vCenter username:
administrator@vsphere.local
Enter the vCenter password:
Enter the vCenter hostname. Ex. vsa:
vspp-prototype-vasa
Enter the vCenter domain. Ex. vsa.openenglab.netapp.com:
vsa.openenglab.netapp.com
Enter the IP address of this Docker host:
10.193.136.230

Beginning Install
Installing Trident and creating the volume
00:00:01 Pulling from netapp/trident-plugin
50c600252068a: Download complete
Digest: sha256:306c6ddc6c46822f2db5c67e2d278bb5254e4d4b4f4fa3662945f1790a3555
Status: Downloaded newer image for netapp/trident-plugin:18.01
Installed plugin netapp/trident-plugin:18.01
graphite-db-dev
Creating the docker-compose.yml file
Creating the SolidFire collector wrapper.sh script
Making wrapper.sh as executable
Creating the storage-schemas.conf file
Creating the vsphere-graphite.json file
Modifying the default 'datasource' values in the pre-packaged dashboards
```
If you plan to customize the collector, you can modify the `install_hciscollector.sh` script. Alternatively, the manual setup instructions are also covered in section 2, HCICollector Installation: Manual.

### 2.3 HCICollector Installation: Manual

This section describes the manual installation of the collector on a Docker host.

If you plan to use multipathing, follow the steps that are outlined at [https://netapp-trident.readthedocs.io/en/stable-v18.04/docker/install/host_config.html#host-configuration](https://netapp-trident.readthedocs.io/en/stable-v18.04/docker/install/host_config.html#host-configuration).

#### Trident Installation and Configuration

To install and configure Trident, complete the following steps:

1. **Clone the GitHub repository to `/opt/github`**.
   ```bash
   mkdir -p /opt/github
   git clone https://github.com/jedimt/hciscollector /opt/github/hciscollector
   ```

2. **Create a location to store the Trident for Docker plugin configuration files**.
   ```bash
   sudo mkdir -p /etc/netappdvp
   ```

3. **Create the configuration file for SolidFire**.
   ```bash
   cat << EOF > /etc/netappdvp/config.json
   {
   "version": 1,
   "storageDriverName": "solidfire-san",
   "Endpoint": "https://admin:solidfire@10.193.136.240/json-rpc/9.0",
   "SVIP": "10.193.137.240:3260",
   "TenantName": "docker",
   "InitiatorIFace": "default",
   "Types": [
   {
   "Type": "docker-default",
   "Qos": {
   "minIOPS": 1000,
   "maxIOPS": 2000,
   "burstIOPS": 4000
   }
   },
   {
   "Type": "docker-app",
   "Qos": {
   "minIOPS": 4000,
   "maxIOPS": 6000,
   "burstIOPS": 8000
   }
   },
   {
   "Type": "docker-db",
   "Qos": {
   "minIOPS": 6000,
   "maxIOPS": 8000,
   "burstIOPS": 10000
   }
   }
   }
   }
   EOF
   ```

4. **Install the Trident plugin**.
   ```bash
   docker plugin install --grant-all-permissions --alias netapp netapp/trident-plugin:18.04
   config-config.json
   ```
5. Verify that the plugin is installed and is enabled.

<table>
<thead>
<tr>
<th>ID</th>
<th>NAME</th>
<th>DESCRIPTION</th>
<th>ENABLED</th>
</tr>
</thead>
<tbody>
<tr>
<td>047ac2d0663f</td>
<td>netapp:latest</td>
<td>Trident - NetApp Docker Volume Plugin</td>
<td>true</td>
</tr>
</tbody>
</table>

6. Create Docker volumes to be used for Graphite persistent storage.

```
# Create Graphite docker volume
docker volume create -d netapp --name graphite-db -o type=docker-db -o size=100G

#show volume
docker volume list --filter 'driver=netapp:latest'
```

Docker Container Setup and Configuration

To install and configure the Docker containers, complete the following steps.

1. Install Docker Compose if it is not already installed.

   ```
   apt install docker-compose
   ```

2. Create the `docker-compose.yml`, specifying the persistent data volumes to use for Graphite and for Grafana and the password to secure the Grafana web interface.

   ```
   cat << EOF > /opt/github/hcicollector/docker-compose.yml
   version: "2"
   services:
     graphite:
       build: ./graphite
       container_name: graphite-v.6
       restart: always
       ports:
       - "8080:80"
       - "8125:8125/udp"
       - "8126:8126"
       - "2003:2003"
       - "2004:2004"
       volumes: #Trident or local volumes for persistent storage
       - graphite-db:/opt/graphite/storage/whisper
     networks:
       - net_hcicollector
     grafana:
       build: ./grafana
       container_name: grafana-v.6
       restart: always
       ports:
       - "80:3000"
     networks:
       - net_hcicollector
     environment:
     #Set password for Grafana web interface
     - GF_SECURITY_ADMIN_PASSWORD=<your password>
     #Optional SMTP configuration for alert queries
     - GF_SMTP_ENABLED=true
     - GF_SMTP_HOST=smtp.gmail.com:465
     - GF_SMTP_USER=<email address>
     - GF_SMTP_PASSWORD=<email password>
     - GF_SMTP_SKIP_VERIFY=true
     sfcollector:
       build: ./sfcollector
       container_name: sfcollector-v.6
       restart: always
       networks:
       - net_hcicollector
```
vmwcollector:
  build: ./vmwcollector
  container_name: vmwcollector-v.6
  restart: always
  networks:
    - net_hcicollector
  depends_on:
    - graphite

networks:
  net_hcicollector:
    driver: bridge

volumes:
  graphite-db:
    external: true

EOF

3. Create the /opt/github/hcicollector/sfcollector/wrapper.sh script with the appropriate SolidFire cluster MVIP address, user name, and password. If you changed the Graphite container name, specify the new host name by using the -g option.

```bash
cat << EOF > /opt/github/hcicollector/sfcollector/wrapper.sh
#!/usr/bin/env bash
while true
  do
    /usr/bin/python /solidfire_graphite_collector.py -s 10.193.136.39 -u admin -p <yourpassword> -g graphite &
    sleep 60
  done
EOF
```

4. If you want to adjust the retention period for the NetApp statistics, edit the /opt/github/hcicollector/graphite/storage-schemas.conf file. By default, the following retention is set, which keeps 1-minute statistics for 7 days, 5-minute statistics for 28 days, and 10-minute statistics for 1 year. If you are also collecting statistics from vCenter, add the [vsphere] section as well. This must be done before starting the containers for the first time.

```yaml
[netapp]
  pattern = ^netapp\.*
  retentions = 1m:7d,5m:28d,10m:1y

[vsphere]
  pattern = ^vsphere\.*
  retentions = 1m:7d,5m:28d,10m:1y
```

5. Create the /opt/github/hcicollector/vmwcollector/vsphere-graphite.json file and add your vCenter details.

```json
cat << EOF > /opt/github/hcicollector/vmwcollector/vsphere-graphite.json
{
  "Domain": "<yourdomain>",
  "Interval": 60,
  "FlushSize": 100,
  "VCenters": [
    { "Username": "administrator@vsphere.local", "Password": "<yourpassword>", "Hostname": "sfps-prototype-vcsa" }
  ],
  "Backend": {
    "Type": "graphite",
    "Hostname": "graphite",
    "Port": 2003
  },
  "Metrics": [
    { "ObjectType": [ "VirtualMachine", "HostSystem" ],
      "Definition": [
        { "Metric": "cpu.usage.average", "Instances": "" },
        { "Metric": "cpu.usage.maximum", "Instances": "" }
      ]
  ]
}
EOF
```
6. Create the `datasource.yml` file at the following location, providing the IP address for the Docker host in the `url` field:

```yaml
apiVersion: 1
datasources:
- name: graphite-db-dev
type: graphite
access: proxy
orgId: 1
isDefault: true
version: 1
editable: true
```
7. Bring up the containers by using `docker-compose`. This task takes several minutes to complete.

```bash
docker-compose -f /opt/github/hcicollector/docker-compose.yml up -d
```

**Note:** If you want to view the logs, you can bring up the containers the first time with the `-d` flag omitted.

### 2.4 Grafana Configuration

To configure Grafana, complete the following steps:

1. After the Docker Compose process completes, launch a web browser to `http://<Docker VM IP Addr>`.
   The Grafana web interface appears. Log in as an admin user and use the password that was configured in the `docker-compose.yml` file.

   **Note:** If you open the Grafana interface before any metrics have been collected, all dashboards might display “N/A” values for all counters. Wait 2 minutes and reload the dashboard, and the issue should resolve itself.

2. Verify that the Graphite data source is automatically provisioned (Figure 5).
3. Verify that the dashboards are automatically provisioned and functional. Figure 6 shows a list, and Figure 7 shows an example dashboard.
Figure 6) Grafana dashboards.
2.5 Graph Conventions That Are Used

The following conventions are used in the system graphs.

- Dashboards expect the host objects (vCenter, ESXi hosts, and so on) to be pulled in by their FQDN. Because Graphite uses "." to delimit metrics, pulling in an IP address breaks the dashboard templating. Alternatively, you can change the dashboard templating to account for IP addresses.
- Null values are shown as null for most graphs, enabling you to spot objects that fail to report statistics. Null values are augmented by keepLastValue.
- keepLastValue(5) continues the line with the last received value when gaps (null values) appear in your data, rather than breaking your line. If there are more than five consecutive missed reporting periods, a break shows in the graph. Removing this option shows a graph with breaks for any object that has no statistics for the reporting period (Figure 8). A value of 5 minutes was chosen because that is the break point for evicting a SolidFire node from the storage cluster (Figure 9). Note the difference in the following screenshots.
Appendix A: Troubleshooting

This section includes some troubleshooting steps that you can use when you have issues with the configuration of the Docker Collector.

Validating Metrics in the Graphite Database

The Graphite API that was used in this project does not include the graphical front end for Graphite, so the render API for Graphite can be used to verify that metrics are being pushed into the Graphite database. The format for displaying cluster metrics is: http://<docker VM IP>:8080/render?target=netapp.solidfire.cluster.<cluster name>.<metric>.
For example, to see the cluster `activeSessions`:  


Figure 10) Using the render API for Graphite.

To display only the metrics from the past hour, add the `&from=<time window>` argument:  


Figure 11) Viewing statistics from the past hour.
Checking Collector Logs

If you have to connect to the SFCollector to troubleshoot, you must override the `entrypoint` for the container.

```
docker run --entrypoint "/bin/bash" -it sfcollector-v.6
```

The logs for the collector are stored in the `/tmp` directory of the container.

Rebuilding a Container

If you need to change a single container in the Docker Compose setup (for instance, to change the collector wrapper script), you can make that change without taking down all the containers.

```
#List the services
root@hci-grafana01:/opt/github/hcicollector# docker-compose config --services
  graphite
  grafana
  sfcollector
  vmwcollector

#Stop the service
docker-compose stop sfcollector #this is the service name
<make changes>

#Start the service
docker-compose up -d --no-deps --build sfcollector
```

Removing Stale Metrics

If the Whisper database has stale metrics, you must remove the corresponding metric files from the Graphite container persistent storage. You can remove them either from the perspective of the container (method 1) or from the Docker host (method 2).

For example, to remove all the metrics for the `ultron` cluster from Graphite, use one of the following procedures.

Method 1: From the Container Perspective

```
#Using Trident - Deleting stale stats from the container’s perspective
#Stop the sfcollector container
root@vmgrafana01-0:/opt/github/hcicollector/# docker-compose down graphite
Stopping graphite-v.6 ... done
Removing graphite-v.6 ... done

#Start the graphite container interactively with persistent storage
docker run --rm -it --entrypoint "/bin/bash" --volume graphite-db:/opt/graphite/storage/whisper graphite-v.6

#Remove old stats for the ultron cluster
root@ed7dbf28f424:/# ls /opt/graphite/storage/whisper/netapp/solidfire/cluster/
ultron  wolverine

root@ed7dbf28f424:/# rm -rf /opt/graphite/storage/whisper/netapp/solidfire/cluster/ultron
```

Method 2: From the Docker Host Perspective

```
#Using Trident - Deleting stats from the Docker host’s perspective
root@sfps-grafana01:/opt/github/hcicollector/# docker-compose down graphite
Stopping graphite-v.6 ... done
Removing graphite-v.6 ... done

#Find the “Id” of the Trident plugin
root@sfps-grafana01:/opt/github/hcicollector/# docker plugin list
ID    NAME                        DESCRIPTION ENABLED
```
Automatically Purging Stale Data

To automate the removal of stale data from the Graphite database, you can use a cron job to run a cleanup script at a set period. The following example runs every day and removes metrics that are over 30 days old that have not had an update:

```
#Add the following cron job (crontab -e)
@daily /root/graphite-whisper-cleanup.sh
```

The contents of the `graphite-whisper-cleanup.sh` script are as follows:

```
root@sfps-grafana01:~# cat graphite-whisper-cleanup.sh
#variable for long pathname
graphitedb=/var/lib/docker/plugins/50cf6ba66948f4c7e329be406d070c25ebe103b49f36382e76888e807c8
a1/rootfs/var/lib/docker-volumes/netapp/graphite-db/

# how much space to reclaim if we delete files not updated in last 30 days?
# find $graphitedb
# -mtime -30 -exec echo -n -e '{0}' ";" | du -h --files0-from=-
# delete the files!
find $graphitedb -type f -mtime +30 -name ".wsp" -exec rm '{}'

# delete empty directories
find $graphitedb -type d -empty -delete
```

The variable for `graphitedb` needs to be changed for your environment. The path should be structured as `/var/lib/docker/plugins/<Trident ID>/rootfs/var/lib/docker-volumes/netapp/graphite-db/`. You can find the Trident ID by running the following `find` command:

```
root@sfps-grafana01:~# docker plugin list
ID         NAME                DESCRIPTION                             ENABLED
5d2382b6be6a  netapp:latest       Trident - NetApp Docker Volume Plugin   true
```

```
root@sfps-grafana01:~# docker plugin inspect 5d2382b6be6a | grep Id
"Id": "5d2382b6be6ad67fb873ae6f02b9af042ff32029b3b4d4f7e176eeeb58a3ef40",
```
Appendix B: SFCollector Statistics

Table 1 lists the (potentially non-exhaustive) statistics that SFCollector uses.

Table 1) SFCollector statistics.

<table>
<thead>
<tr>
<th>API</th>
<th>Statistic Name</th>
<th>Description</th>
<th>Calc.</th>
<th>Type</th>
<th>Ver.</th>
</tr>
</thead>
<tbody>
<tr>
<td>clusterStats</td>
<td>actualIOPS</td>
<td>Current actual IOPS for the entire cluster in the last 500ms</td>
<td>Point in time</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>clientQueueDepth</td>
<td>Number of outstanding read and write operations to the cluster</td>
<td>N/A</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>clusterUtilization</td>
<td>Cluster capacity being utilized</td>
<td>N/A</td>
<td>Float</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>latencyUSec</td>
<td>Average time, in microseconds, to complete operations to a cluster in the last 500ms</td>
<td>Point in time</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>normalizedIOPS</td>
<td>Average number of IOPS for the entire cluster in the last 500ms</td>
<td>Point in time</td>
<td>Integer</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>readBytes</td>
<td>Total cumulative bytes read from the cluster since the creation of the cluster</td>
<td>Monotonic</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>readBytesLastSample</td>
<td>Total number of bytes read from the cluster during the last sample period</td>
<td>Point in time</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>readLatencyUSec</td>
<td>Average time, in microseconds, to complete read operations to the cluster in the last 500ms</td>
<td>Point in time</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>readOps</td>
<td>Total cumulative read operations to the cluster since the creation of the cluster</td>
<td>Monotonic</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>readOpsLastSample</td>
<td>Total number of read operations during the last sample period</td>
<td>Point in time</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>unalignedReads</td>
<td>Total cumulative unaligned read operations to a cluster since the creation of the cluster</td>
<td>Monotonic</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td>API</td>
<td>Statistic Name</td>
<td>Description</td>
<td>Calc.</td>
<td>Type</td>
<td>Ver.</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------</td>
<td>-----------</td>
<td>------</td>
</tr>
<tr>
<td></td>
<td>unalignedWrites</td>
<td>Total cumulative unaligned write operations to a cluster since the creation of the cluster</td>
<td>Monotonic</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>writeLatencyUSec</td>
<td>Average time, in microseconds, to complete write operations to the cluster in the last 500ms</td>
<td>Point in time</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>writeOps</td>
<td>Total cumulative write operations to the cluster since the creation of the cluster</td>
<td>Monotonic</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>writeOpsLastSample</td>
<td>Total number of write operations during the last sample period</td>
<td>Point in time</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>writeBytes</td>
<td>Total cumulative bytes written to the cluster since the creation of the cluster</td>
<td>Monotonic</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>writeBytesLastSample</td>
<td>Total number of bytes written from the cluster during the previous sample period</td>
<td>Point in time</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>clusterCapacity</td>
<td>activeBlockSpace</td>
<td>N/A</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Amount of space on the block drives, and including additional information such as metadata entries and space that can be cleaned up</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>activeSessions</td>
<td>Number of active iSCSI sessions</td>
<td>N/A</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>averageIOPS</td>
<td>Average IOPS for the cluster since midnight UTC</td>
<td>N/A</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>ClusterRecentIOSize</td>
<td>Average size of IOPS to all volumes in the cluster</td>
<td>N/A</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>currentIOPS</td>
<td>Average IOPS for all volumes in the cluster over the last five seconds</td>
<td>N/A</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>maxIOPS</td>
<td>Estimated maximum IOPS capability of the current cluster</td>
<td>N/A</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td>API</td>
<td>Statistic Name</td>
<td>Description</td>
<td>Calc.</td>
<td>Type</td>
<td>Ver.</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------</td>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>maxOverProvisionableSpace</td>
<td>Maximum amount of provisionable space</td>
<td>N/A</td>
<td>Integer</td>
<td>9,10</td>
<td></td>
</tr>
<tr>
<td>maxProvisionedSpace</td>
<td>Total amount of provisionable space if all volumes are filled to 100%</td>
<td>N/A</td>
<td>Integer</td>
<td>9,10</td>
<td></td>
</tr>
<tr>
<td>maxUsedMetadataSpace</td>
<td>Number of bytes on volume drives that are used to store metadata</td>
<td>N/A</td>
<td>Integer</td>
<td>9,10</td>
<td></td>
</tr>
<tr>
<td>maxUsedSpace</td>
<td>Total amount of space on all active block drives</td>
<td>N/A</td>
<td>Integer</td>
<td>9,10</td>
<td></td>
</tr>
<tr>
<td>nonZeroBlock</td>
<td>Total number of 4KiB blocks that contain data after the last garbage collection</td>
<td>N/A</td>
<td>Integer</td>
<td>9,10</td>
<td></td>
</tr>
<tr>
<td>peakActiveSessions</td>
<td>Peak number of iSCSI connections since midnight UTC</td>
<td>N/A</td>
<td>Integer</td>
<td>9,10</td>
<td></td>
</tr>
<tr>
<td>peakIOPS</td>
<td>Highest value for currentIOPS since midnight UTC</td>
<td>N/A</td>
<td>Integer</td>
<td>9,10</td>
<td></td>
</tr>
<tr>
<td>provisionedSpace</td>
<td>Total amount of space that is provisioned in all volumes on the cluster</td>
<td>N/A</td>
<td>Integer</td>
<td>9,10</td>
<td></td>
</tr>
<tr>
<td>totalOps</td>
<td>Total number of I/O operations that are performed throughout the lifetime of the cluster</td>
<td>N/A</td>
<td>Integer</td>
<td>9,10</td>
<td></td>
</tr>
<tr>
<td>uniqueBlocks</td>
<td>Total number of blocks that are stored on the block drives, including replicated blocks</td>
<td>N/A</td>
<td>Integer</td>
<td>9,10</td>
<td></td>
</tr>
<tr>
<td>uniqueBlocksUsedSpace</td>
<td>Total amount of data that the uniqueBlocks take up on the block drives</td>
<td>N/A</td>
<td>Integer</td>
<td>9,10</td>
<td></td>
</tr>
<tr>
<td>usedMetadataSpace</td>
<td>Total number of bytes on volume drives that are used to store metadata</td>
<td>N/A</td>
<td>Integer</td>
<td>9,10</td>
<td></td>
</tr>
<tr>
<td>usedMetadataSpaceInSnapshots</td>
<td>Number of bytes on volumes drives that are used for storing unique data in snapshots; provides an estimate of how much metadata</td>
<td>N/A</td>
<td>Integer</td>
<td>9,10</td>
<td></td>
</tr>
<tr>
<td>API</td>
<td>Statistic Name</td>
<td>Description</td>
<td>Calc.</td>
<td>Type</td>
<td>Ver.</td>
</tr>
<tr>
<td>--------------------</td>
<td>----------------</td>
<td>------------------------------------------------------------------------------</td>
<td>-------</td>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td></td>
<td>usedSpace</td>
<td>Total amount of space that is used by all block drives in the system</td>
<td>N/A</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>zeroBlocks</td>
<td>Total number of empty 4KiB blocks without data after the last round of garbage collection.</td>
<td>N/A</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td>nodeStats</td>
<td>cpu</td>
<td>CPU usage in percent (%)</td>
<td>N/A</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>cBytesIn</td>
<td>Bytes in on the cluster interface</td>
<td>N/A</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>cBytesOut</td>
<td>Bytes out on the cluster interface</td>
<td>N/A</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>sBytesIn</td>
<td>Bytes in on the storage interface</td>
<td>N/A</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>sBytesOut</td>
<td>Bytes out on the storage interface</td>
<td>N/A</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>mBytesIn</td>
<td>Bytes in on the management interface</td>
<td>N/A</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>mBytesOut</td>
<td>Bytes out on the management interface</td>
<td>N/A</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>networkUtilizationCluster</td>
<td>Network interface utilization (%) for the cluster network interface</td>
<td>N/A</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>networkUtilizationStorage</td>
<td>Network interface utilization (%) for the storage network interface</td>
<td>N/A</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>readOps</td>
<td>Monotonically increasing value of the total read operations to a node</td>
<td>N/A</td>
<td>Integer</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>usedMemory</td>
<td>Total usage in bytes</td>
<td>N/A</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>writeOps</td>
<td>Monotonically increasing value of the total write operations to a node</td>
<td>N/A</td>
<td>Integer</td>
<td>10</td>
</tr>
<tr>
<td>volumeStats</td>
<td>accountID</td>
<td>ID of the account of the volume owner</td>
<td>N/A</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td>API</td>
<td>Statistic Name</td>
<td>Description</td>
<td>Calc.</td>
<td>Type</td>
<td>Ver.</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------</td>
<td>---------</td>
<td>------</td>
</tr>
<tr>
<td></td>
<td>actualIOPS</td>
<td>Current actual IOPS to the volume in the last 500ms</td>
<td>Point in time</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>averageIOPSize</td>
<td>Average size in bytes of the recent I/O to the volume in the last 500ms</td>
<td>Point in time</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>burstIOPSCredit</td>
<td>Total number of IOPS credits available</td>
<td>N/A</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>clientQueueDepth</td>
<td>Number of outstanding read and write operations to the volume</td>
<td>N/A</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>latencyUSec</td>
<td>Average time, in microseconds, to complete operations to the volume in the last 500ms</td>
<td>Point in time</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>bytesRead</td>
<td>Total cumulative bytes read from the volume since the creation of the volume</td>
<td>Monotonic</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>bytesReadLastSample</td>
<td>Total number of bytes read from the volume during the last sample period</td>
<td>Point in time</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>readLatencyUSec</td>
<td>Average time, in microseconds, to complete read operations to the volume in the last 500ms</td>
<td>Point in time</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>readOps</td>
<td>Total cumulative read operations to the volume since the creation of the volume</td>
<td>Monotonic</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>readOpsLastSample</td>
<td>Total number of read operations during the last sample period.</td>
<td>Point in time</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>throttle</td>
<td>A floating value between 0 and 1 that represents how much the system is throttling clients below their maxIOPS due to replication of data, transient errors, and snapshots created</td>
<td>N/A</td>
<td>Float</td>
<td>9,10</td>
</tr>
<tr>
<td>API</td>
<td>Statistic Name</td>
<td>Description</td>
<td>Calc.</td>
<td>Type</td>
<td>Ver.</td>
</tr>
<tr>
<td>----------------</td>
<td>------------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>-------------</td>
<td>-----------</td>
<td>------</td>
</tr>
<tr>
<td></td>
<td>unalignedReads</td>
<td>Total cumulative unaligned read operations to a volume since the creation of the volume</td>
<td>Monotonic</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>unalignedWrites</td>
<td>Total cumulative unaligned write operations to a volume since the creation of the volume</td>
<td>Monotonic</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>volumeUtilization</td>
<td>Floating value that describes how much the client is using the volume: 0 = the client is not using the volume; 1 = the client is using their maximum IOPS; &gt;1 = the client is using their burst IOPS</td>
<td>N/A</td>
<td>Float</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>writeLatencyUSec</td>
<td>Average time, in microseconds, to complete write operations to the volume in the last 500ms</td>
<td>Point in time</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>writeOps</td>
<td>Total cumulative write operations to the volume since the creation of the volume</td>
<td>Monotonic</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>writeOpsLastSample</td>
<td>Total number of write operations during the last sample period</td>
<td>Point in time</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>writeBytes</td>
<td>Total cumulative bytes written from the volume since the creation of the volume</td>
<td>Monotonic</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>writeBytesLastSample</td>
<td>Total number of bytes written from the volume during the last sample period</td>
<td>Point in time</td>
<td>Integer</td>
<td>9,10</td>
</tr>
<tr>
<td></td>
<td>zeroBlocks</td>
<td>Total number of empty 4KiB blocks without data after the last round of garbage collection.</td>
<td>N/A</td>
<td>Integer</td>
<td>9,10</td>
</tr>
</tbody>
</table>
Where to Find Additional Information

To learn more about the information that is described in this document, see the following documents or websites:

- Updates to the HCICollector can be found at https://github.com/jedimt/hcicollector
- This blog has some excellent troubleshooting steps for a Graphite + Grafana configuration: http://dieter.plaetinck.be/post/25-graphite-grafana-statsd-gotchas
- How to install Docker https://docs.docker.com/engine/installation/linux/ubuntu
- Composing a Graphite server with Docker https://thepracticalsysadmin.com/composing-a-graphite-server-with-docker/
- SolidFire Collector for Graphite https://github.com/cbiebers/solidfire-graphite-collector
- vSphere Graphite collector https://github.com/cblomart/vsphere-graphite

Version History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Document Version History</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version 1.0</td>
<td>May 2018</td>
<td>Initial release</td>
</tr>
</tbody>
</table>
Refer to the Interoperability Matrix Tool (IMT) on the NetApp Support site to validate that the exact product and feature versions described in this document are supported for your specific environment. The NetApp IMT defines the product components and versions that can be used to construct configurations that are supported by NetApp. Specific results depend on each customer’s installation in accordance with published specifications.

Copyright Information

Copyright © 2018 NetApp, Inc. All rights reserved. Printed in the U.S. No part of this document covered by copyright may be reproduced in any form or by any means—graphic, electronic, or mechanical, including photocopying, recording, taping, or storage in an electronic retrieval system—without prior written permission of the copyright owner.

Software derived from copyrighted NetApp material is subject to the following license and disclaimer:

THIS SOFTWARE IS PROVIDED BY NETAPP “AS IS” AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WHICH ARE HEREBY DISCLAIMED. IN NO EVENT SHALL NETAPP BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

NetApp reserves the right to change any products described herein at any time, and without notice. NetApp assumes no responsibility or liability arising from the use of products described herein, except as expressly agreed to in writing by NetApp. The use or purchase of this product does not convey a license under any patent rights, trademark rights, or any other intellectual property rights of NetApp.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.277-7103 (October 1988) and FAR 52-227-19 (June 1987).

Trademark Information

NETAPP, the NETAPP logo, and the marks listed at http://www.netapp.com/TM are trademarks of NetApp, Inc. Other company and product names may be trademarks of their respective owners.