NetApp ToolChest

NetApp Harvest Extension Manager Installation & Administration Guide v1.0

Georg Mey, NetApp
20 September 2018

Abstract

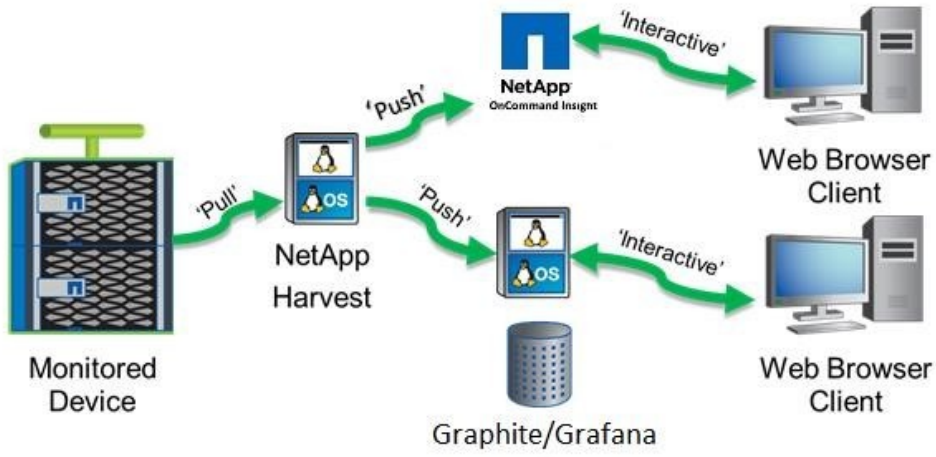
This guide discusses installation and administration steps for the NetApp Harvest Extensions

TABLE OF CONTENTS

1 Introduction to Harvest Extension Manager

1.1 About NetApp Harvest

NetApp Harvest is software that connects to a remote host, collects data, calculates and summarizes the data, and posts it to a metrics server. It offers default collection templates for performance information from ONTAP and Data ONTAP 7-mode every 1 minute, and storage capacity information from OnCommand Unified Manager 6+ every 15 minutes. Further customization is possible in terms of the detail of data collected, the summarization that occurs, and the frequency of collection.
Harvest can be used together with Graphite (time-series db) and Grafana (dashboards), and/or with NetApp OnCommand Insight as shown in the diagram below:



1.2 Why do we need Extensions

In order to avoid to overload the Harvest Poller Program **netapp-worker** with additional tasks, we have decided to create an interface to trigger Harvest Extensions during a poll cycle. Thanks to ***Chris Madden*** (original developer of Harvest) this is an easy task.
You can create "virtual" counter objects in Harvest templates with an associated plugin. For this object no counters will be retrieved, but the plugin will be called. As counter object names will be sorted before the polling cycle starts, it's pretty easy to define an additonal counter object at the beginning or the end of the counter object list.

We have created a generic plugin which will be called at the beginning and/or end of the poll cycle. This plugin will read a list of pre- or post-exec commands from the HEM template file.

All relevant **Harvest** config paramters of the poller task will be passed via shell environment variables. This provides the flexibility to call simple shell scripts as well as complex programs written in any language.

2 Installation of Harvest Extension Manager

2.1 Components

The Framework has three components (which need to be installed in the **Harvest** installation tree – in this case in **/opt/netapp-harvest/**).

- **/opt/netapp-harvest/plugin/pre-post-exec**
  this is the plugin called by **Harvest**

- **/opt/netapp-harvest/extensions.conf**
  the **HEM** template with pre- and post-exec commands

- **/opt/netapp-harvest/extension/**
  location of executable extension scripts

2.2 Plugin "pre-post-exec"

This plugin is simple and straight forward. You may not need to make any changes.
The plugin will export the following environment variables:

Name	Content	Harvest Source (Perl)
_HARVEST_GRAPHITE_HOST	DNS name or IP address of the Graphite Server	$connection{graphite_server}
_HARVEST_GRAPHITE_ROOT	**netapp.perf.<group>.<cluster>**	$connection{graphite_root}
_HARVEST_GROUP	group name of the cluster	$connection{group}
_HARVEST_HOSTNAME	DNS name or IP address of the cluster	$connection{hostname}
_HARVEST_USERNAME	cluster username used by **Harvest**	$connection{username}
_HARVEST_PASSWORD	password of the user above	$connection{password}
_HARVEST_INSTALL_DIR	Typically /opt/netapp-harvest	Dynamically generated by plugin
_HARVEST_POLL_EPOCH	Timestamp for poll cycle	$poller{*}{data_nextrun_timestamp}
_HARVEST_VERBOSE	Verbose option	$options{'v'}

The **netapp-worker** option **-v** will be exported via the environment variable **$_HARVEST_VERBOSE**.
This can be used to control extended logging to **Harvest** itself, the Plugins and all Extensions centrally.

2.3 Template "extensions.conf"

You can add these two object entries to an existing template. With the template cascading feature in Harvest we do recommend to create an own template file like this:

```
# ============================================================================
# Adds pre- and post-exec plugins
# ============================================================================
%poller = (
    'AAAA' => {
            plugin => 'pre-post-exec',
            command_list => [
               "/opt/netapp-harvest/extension/snapmirror_replications.py"
            ],
            data_update_freq => 60,
            enabled => '1',
    },
    'zzzz' => {
```

```
            plugin => 'pre-post-exec',
            command_list => [
                    "/opt/netapp-harvest/extension/nfs-connections.sh"
            ],
            data_update_freq => 900,
            enabled => '1'
    }
);
```

You have to use the "virtual" Object names **AAAA** and **zzzz**. The name of the object will be passed by **Harvest** to the plugin. The plugin executes commands for **AAAA** and commands for **zzzz** At the beginning and end of a polling session respectively.

Take this template and merge it with the current template(s) used by **Harvest** for this cluster. To configure this, you need to add a template parameter in the Havest configuration file **netapp-harvest.conf**:

```
[mycluster]
hostname         = 192.168.1.2
site             = DC1
username         = admin
password         = secrect
data_update_freq = 60
template         = default,extensions.conf
```

2.5 A few notes for Extension Developers

Harvest Extensions can be used for a variety of tasks. There are a few requirements programmers have to keep in mind:

- As the default poll cycle is 60 seconds - the extension should have finished work before the next poll cycle begins – the **pre-post-exec** plugin starts extension asynchronously and does a **`waitpid()`** for <defunct> extension at each poll cycle.

- There can be many extensions running in parallel

- Any type of persistent information used during the run or across runs needs to be addressed by an instance related naming scheme

- For debugging – logfiles should be placed in the central Harvest logging directory /opt/netapp-harvest/log. Good practice would be the following naming convention:

  `<cluster-name>_ netapp-harvest.log` logfile of the poller
  `<cluster-name>_ netapp-harvest_<extension-name>.log` logfile of an exstension

- Deployment options:

  - You can add all the components to an existing **Harvest** installation – when doing so, keep in mind that you may have to re-install the extension when you touch your current Harvest setup.

  - You can use a different host/VM or a different instance of Harvest on the same host/VM using a different installation directory. This method does make the extensions independent from the normal polling cycles of performance counters. This may be a good approach especially for large environments, when 1000s of objects need to be polled per cycle

3 Overview of currently available Extensions from NetApp

3.1 nfs-connect

Purpose

This extension retrieves the number of NFSv3 connections and NFSv3 hosts via an ONTAP CLI command (network connections active show -node <node> -service nfs*) for each node in the cluster and generates the following new metric entries in the Whisper timeseries database of Graphite.

netapp.perf.<group-name>.<cluster-name>.node.<node-name>.nfsv3.nfs_connections
netapp.perf.<group-name>.<cluster-name>.node.<node-name>.nfsv3.nfs_hosts

Installation and Prerequisits

This extension is written with the Bash scripting language. You do need the tool sshpass being installed. For Debian based installs – this can be done via **sudo apt-get install sshpass**
Installation is done in few simple steps:

1. Copy **nfs-connect.sh** to **/opt/netapp-harvest/extension/nfs-connections.sh**
2. Check the execute permissions of the script **nfs-connections.sh**
3. Add the command to **/opt/netapp-harvest/extension.conf**

Testing and debugging

The script checks the **netapp-worker/netapp-manager** command line option **-v**. If set, debug messages will be sent to this logfile:

/opt/netapp-harvest/log/<cluster-name>_netapp-harvest_nfs-connections.log

Log entries per poll cycle will look like this:

```
[2019-08-01 14:08:13] [DEBUG  ] Session started
[2019-08-01 14:08:14] [DEBUG  ] M= netapp.perf.CBC.cuba.node.cuba-01.nfsv3.nfs_connections 89.0 1537369500
[2019-08-01 14:08:15] [DEBUG  ] M= netapp.perf.CBC.cuba.node.cuba-01.nfsv3.nfs_hosts 60.0 1537369500
[2019-08-01 14:08:16] [DEBUG  ] M= netapp.perf.CBC.cuba.node.cuba-02.nfsv3.nfs_connections 32.0 1537369500
[2019-08-01 14:08:16] [DEBUG  ] M= netapp.perf.CBC.cuba.node.cuba-02.nfsv3.nfs_hosts 22.0 1537369500
[2019-08-01 14:08:17] [DEBUG  ] M= netapp.perf.CBC.cuba.node.cuba-03.nfsv3.nfs_connections 68.0 1537369500
[2019-08-01 14:08:18] [DEBUG  ] M= netapp.perf.CBC.cuba.node.cuba-03.nfsv3.nfs_hosts 23.0 1537369500
[2019-08-01 14:08:19] [DEBUG  ] M= netapp.perf.CBC.cuba.node.cuba-04.nfsv3.nfs_connections 49.0 1537369500
[2019-08-01 14:08:20] [DEBUG  ] M= netapp.perf.CBC.cuba.node.cuba-04.nfsv3.nfs_hosts 19.0 1537369500
[2019-08-01 14:08:20] [DEBUG  ] Session ended
```

You can run the script **nfs-connection.sh** outside Harvest by either setting environment variables like outlined in chapter 2.2 or by using runstring parameters:

```
# ./nfs-connections.sh -h

NetApp Harvest Extensions - nfs-connections.sh

usage:

      nfs-connections.sh [general options] [required options]

general options:

      -h | --help
      -t | --test
      -v | --verbose

required options:

      -H | --ghost            Fills environment variable _HARVEST_GRAPHITE_HOST
      -R | --ghroot           Fills environment variable _HARVEST_GRAPHITE_ROOT
```

```
      -G | --group           Fills environment variable _HARVEST_GROUP
      -C | --cluster         Fills environment variable _HARVEST_HOSTNAME
      -U | --user            Fills environment variable _HARVEST_USERNAME
      -P | --password        Fills environment variable _HARVEST_PASSWORD
      -I | --installdir      Fills environment variable _HARVEST_INSTALL_DIR
      -E | --epoch           Fills environment variable _HARVEST_POLL_EPOCH
```

With option **--test** retrieved data will be send to STDOUT only:

```
# ./nfs-connections.sh –test –ghost=192.168.1.2 . . .

node                 NFS-connections      NFS-clients
-------------------- -------------------- --------------------
cuba-01              89                   60
cuba-02              33                   23
cuba-03              70                   24
cuba-04              52                   20
```

Testing and debugging

3.2 snapmirror-replications.py

See documentation in extension (**extension/snapmirror-replications.py**)