Documentation on Log Analytics Installation in a HDInsight cluster

AIM

This documentation aims to do a detailed explanation of installation of log analytics in a HDinsight cluster using a script file. It describes how it is installed in each node (head , worker and zookeeper), and diagrammatically show how the code is flowing.

The Log analytics installation is done using a script file which is run through script action on all nodes(head, worker and zookeeper). The log_analytics_insall.sh file has other file dependency which will also be explained in this documentation.

Script Link:
https://hdiconfigactions.blob.core.windows.net/loganalyticsmonitoring/log_analytics_install.sh

Dependent files run inside the script:

1. Log Analytics Constant Python file
   Link:
   https://hdiconfigactions.blob.core.windows.net/loganalyticsmonitoring/LogAnalyticsConstants.py

2. Log Analytics Utilities Python file
   Link:
   https://hdiconfigactions.blob.core.windows.net/loganalyticsmonitoring/log_analytics_utilities.py

3. Log Analytics Uninstall file
   Link:
   https://hdiconfigactions.blob.core.windows.net/loganalyticsmonitoring/log_analytics_uninstall.py

4. Log Analytics Install Python file
   Link:
   https://hdiconfigactions.blob.core.windows.net/loganalyticsmonitoring/log_analytics_install.py

CONTENTS

1. What is Azure HDInsight?
2. What is Log Analytics?
3. What is Script action?
4. Understanding script action parameters.
    - Valid cluster types
    - Workspace ID and Key
    - Resource Group
    - Storage Account
5. Working of Log Analytics install script
    - Validate script input parameters
    - Download Constant and Utilities files
    - Run the uninstall script
    - Install Geneva Monitoring pipeline
6. Explaining log analytics python file used to install Geneva monitoring
    - Table explaining 38 functions and its actions
    - Explaining the configuration files installed in which node types

A) What is Azure HDInsight?

Azure HDInsight is a cloud distribution of Hadoop components. Azure HDInsight makes it easy, fast, and cost-effective to process massive amounts of data in a customizable environment.

HDInsight offers following cluster types: Apache Hadoop, Apache Spark, Apache Hbase, Apache Storm, Apache Interactive Query and Apache Kafka.

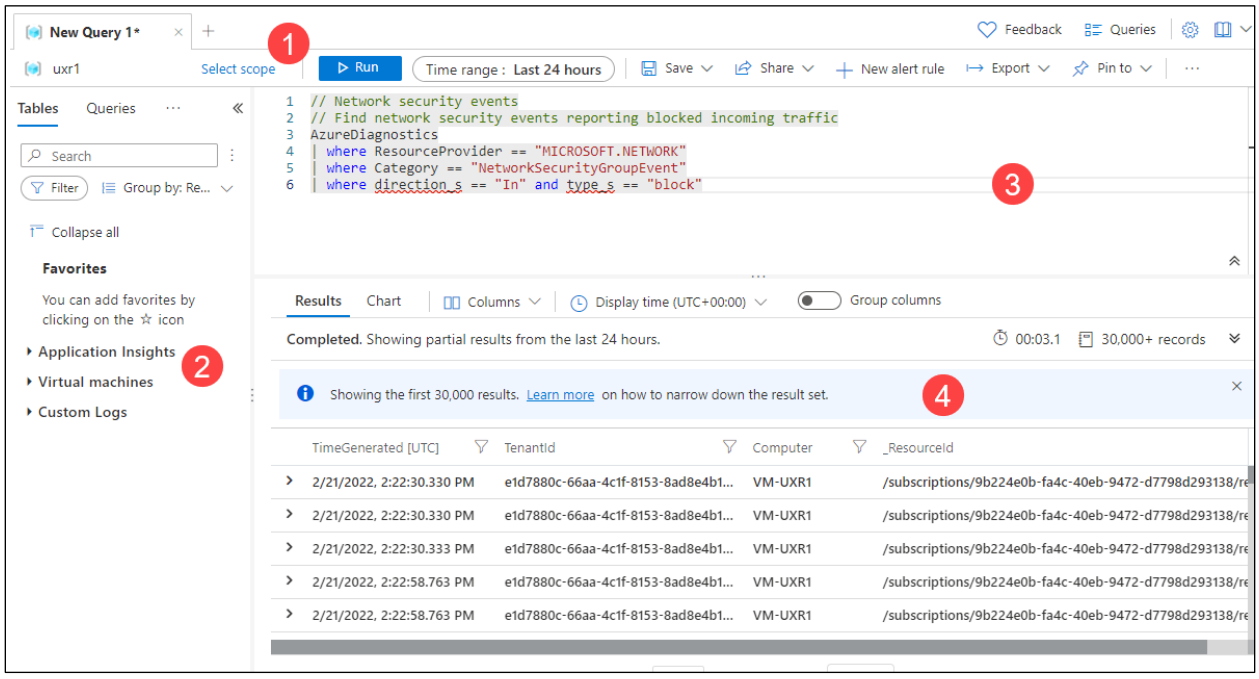
B) What is Log Analytics?

A feature of Azure Monitor that collects and analyzes data generated by resources in cloud and on-premises environments.

Log Analytics is a tool in the Azure portal used to edit and run log queries with data in Azure Monitor Logs. You may write a simple query that returns a set of records and then use features of Log Analytics to sort, filter, and analyze them. Or you may write a more advanced query to perform statistical analysis and visualize the results in a chart to identify a particular trend. Whether you work with the results of your queries interactively or use them with other Azure Monitor features such as log query alerts or workbooks, Log Analytics is the tool that you're going to use write and test them.

Log Analytics Interface

The following image identifies the different components of Log Analytics



1 – The top action bar which shows scope, time range and run button.

2 - It is the sidebar which lists tables in the workspace, sample queries and filter options for the current query.

3 – Query Window

The query window is where you edit your query. This includes intellisense for KQL commands and color coding to enhance readability. Click the + at the top of the window to open another tab.

4 – Results Window

The results of the query are displayed in the results window. By default, the results are displayed as a table. To display as a chart, either select Chart in the results window, or add a render command to your query.

C) What is Script Action?

A script action is Bash script that runs on the nodes in an HDInsight cluster. Characteristics and features of script actions are as follows:

The Bash script URI (the location to access the file) has to be accessible from the HDInsight resource provider and the cluster.

D) Understanding script action parameters

Link:
https://hdiconfigactions.blob.core.windows.net/loganalyticsmonitoring/log_analytics_install.sh

What does the script do?

The job of this script file is to install the log analytics in the cluster by installing Geneva Monitoring Pipeline to send information to Log Analytics.

Parameters

The script takes 5 parameters which are Cluster Type, Workspace ID, Workspace Key, Resource Group and Storage Account.

Parameters must be in this order only:

[clusterType, workspaceID, workspaceKey, resourceGroup, storageAccount]

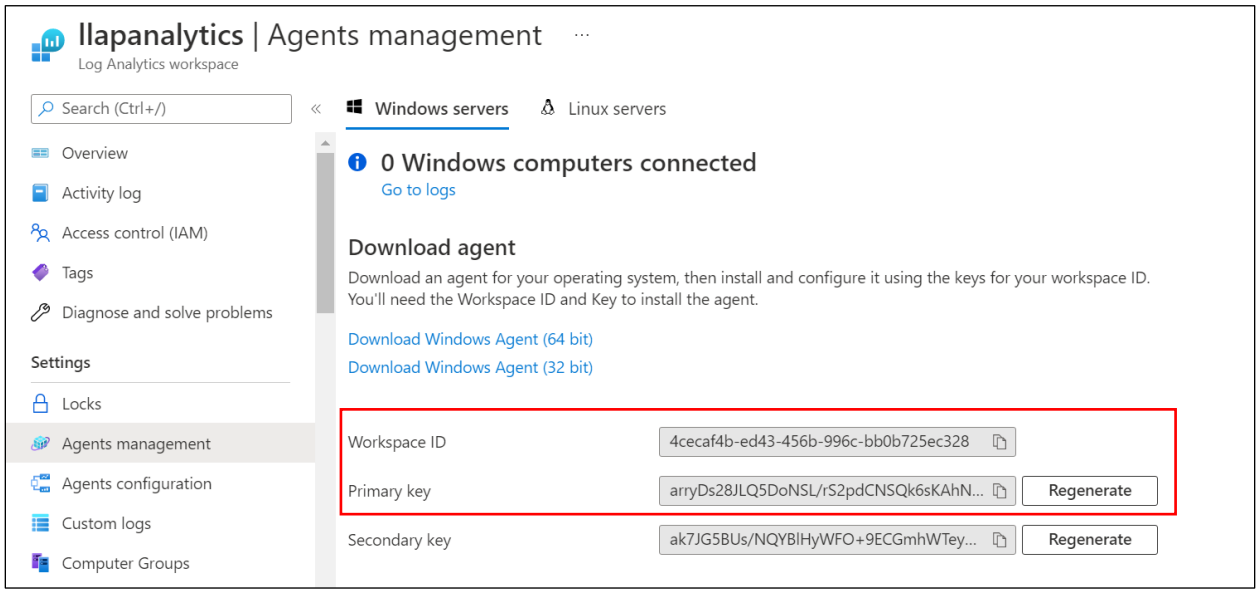
Valid Cluster Types

Valid cluster types in 1st parameter are :

Cluster Names	Cluster_name used in parameter
Apache Hadoop	“hadoop”
Apache Hbase	“hbase”
Apache Hive	“hive”
Apache Interactive Query	“interactivehive”
Apache Kafka	“kafka”
Apache Spark	“spark”
Apache Storm	“storm”

Workspace ID and Workspace Key

Both ID and key of a log analytics workspace can be found in Agent Management under Settings category in sidebar

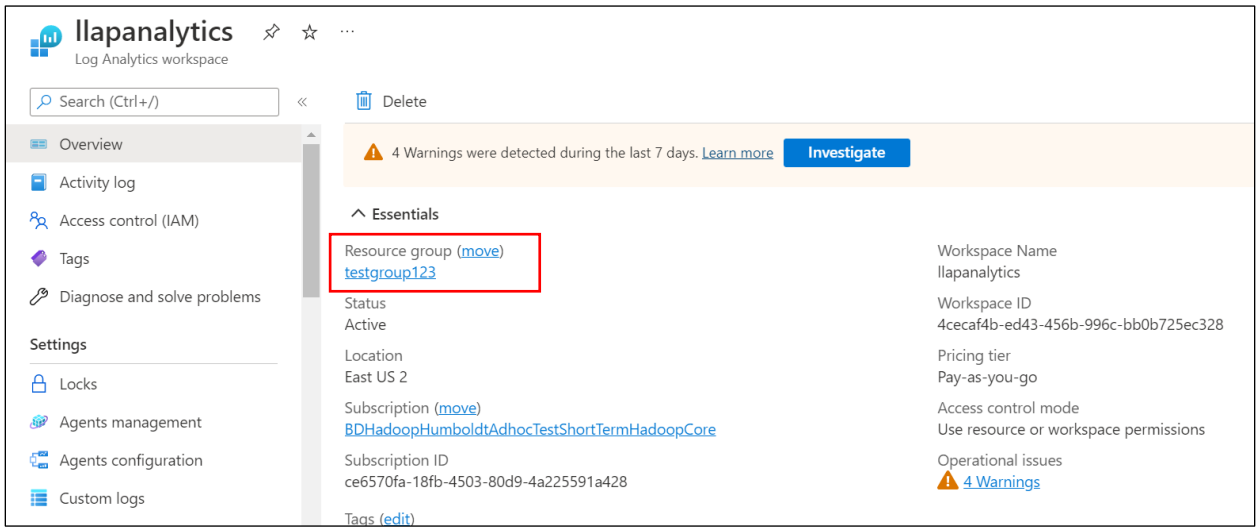


Workspace ID and key are show in the sample image above

Resource Group

The Resource group is the 4th parameter for this log analytics install script.

The Resource group in which the log analytics workspace exists which you want to connect to the cluster.



Resource Group for this workspace is shown in red box

Storage Account

The Storage account is the 5th parameter for the install script. It specifies the name of the storage account where the script exists.

You can give the name of the storage account as “null” as it has been already handled inside the script.

```
STORAGE_ACCT=${5:-hdiconfigactions}
if [ "$STORAGE_ACCT" == "null" ];
then
    STORAGE_ACCT=hdiconfigactions
fi
```

If the 5th parameter has not been entered or is “null” then in that case storage account will be “hdiconfigactions”.

E) What is mdsd?

Mdsd is the **Linux logging infrastructure for Azure services**. It connects various log outputs to Azure monitoring service (Geneva warm path).The mdsd output plugin is a buffered fluentd plugin. https://github.com/Azure/fluentd-plugin-mdsd

F) What is td-agent?

Treasure agent (td-agent) is a data collection daemon. It collects logs from various data sources and uploads them to Treasure Data.

Logs usually rotate on an hourly or daily basis based on time or size. This system quickly produces many large log files that need to be batch imported for further analysis. This is an outdated approach. Logs are better treated as continuously generated *STREAMS* as opposed to files.

G) Working of log analytics install script

Some useful built-in commands used in the script

Comamnd	Explanation
Set -ex	This command sets the script behaviour to Exit immediately if a command exits with a non-zero status and also print commands and their arguments as they are executed.
Exec	Replace the shell with the given command. After the command completes exits the shell.
logger	Enter messages into the system log.

The functioning of the log_analytics_install.sh script can be divided into 4 steps:

1. **Validate the script input parameters**

```
#Validate the script input paramters
if [ -z "$CLUSTER_TYPE" ];
then
    logger "$SCRIPTNAME - Cluster type parameter must be provided"
    exit 1
fi

if  [ -z "$WORKSPACE_ID" ]; then
    logger "$SCRIPTNAME - Workspace ID parameter must be provided."
    exit 1
fi

if  [ -z "$WORKSPACE_KEY" ]; then
    logger "$SCRIPTNAME - Workspace Key parameter must be provided."
    exit 1
fi
```

The script checks if the parameter is not blank and Cluster type, Workspace ID and Workspace Key have been provided by the user, since they are essential for the installation of the Geneva monitoring.

```
if [ -z "$CLUSTER_TYPE" ];
```

The -z flag is used to **check if the string has Zero length.**

2. **Download Constant and Utilities used by the install and uninstall scripts**.

Storage Account base URI is set using the **STORAGE_ACCT** parameter.

```
STORAGE_ACCT_BASE_URI=https://$STORAGE_ACCT.blob.core.windows.net/loganalyticsmon
itoring/
```

Since STORAGE_ACCT is set by default so it becomes,

```
https://hdiconfigactions.blob.core.windows.net/loganalyticsmonitoring/
```

```
#Download Contstants and Utilities used by the install and uninstall scripts
REMOTE_LOG_ANALYTICS_CONSTANTS_PY=$STORAGE_ACCT_BASE_URI"LogAnalyticsConstants.py"
LOCAL_LOG_ANALYTICS_CONSTANTS_PY=/mnt/LogAnalyticsConstants.py
wget -T 60 $REMOTE_LOG_ANALYTICS_CONSTANTS_PY -O $LOCAL_LOG_ANALYTICS_CONSTANTS_PY

REMOTE_LOG_ANALYTICS_UTILITIES_PY=$STORAGE_ACCT_BASE_URI"log_analytics_utilities.py"
LOCAL_LOG_ANALYTICS_UTILITIES_PY=/mnt/log_analytics_utilities.py
wget -T 60 $REMOTE_LOG_ANALYTICS_UTILITIES_PY -O $LOCAL_LOG_ANALYTICS_UTILITIES_PY
```

The script downloads **LogAnalyticsConstant.py file** and **log_analytics_utilities.py file** with a maximum try of 60 in case the download fails.

3. **Run the uninstallation script to make sure node starts installation from a clean state**

The **log_analytics_uninstall.py file** is downloaded using the same storage account base URI.

```
#Run the uninstall script to make sure node starts installation from clean state
REMOTE_LOG_ANALYTICS_UNINSTALL_PY=$STORAGE_ACCT_BASE_URI"log_analytics_uninstall.py"
LOCAL_LOG_ANALYTICS_UNINSTALL_PY=/mnt/log_analytics_uninstall.py
wget -T 60 $REMOTE_LOG_ANALYTICS_UNINSTALL_PY -O $LOCAL_LOG_ANALYTICS_UNINSTALL_PY

logger "Finished downloading log analytics uninstall python script"

logger "Running uninstall script to clean previous configurations"

python $LOCAL_LOG_ANALYTICS_UNINSTALL_PY --preinstall

logger "Finished running log analytics uninstall python script"
```

The uninstall python file is run after it is downloaded like the constant files above

4. **Install Geneva Monitoring pipeline using given workspace key**

```
#Install Geneva Monitoring pipeline using given workspace and key
REMOTE_LOG_ANALYTICS_INSTALL_PY=$STORAGE_ACCT_BASE_URI"log_analytics_install.py"
LOCAL_LOG_ANALYTICS_INSTALL_PY=/mnt/log_analytics_install.py
wget -T 60 $REMOTE_LOG_ANALYTICS_INSTALL_PY -O $LOCAL_LOG_ANALYTICS_INSTALL_PY

logger "Finished downloading log analytics install python script"

python $LOCAL_LOG_ANALYTICS_INSTALL_PY $CLUSTER_TYPE $WORKSPACE_ID $WORKSPACE_KEY $RESOURCE_GROUP $STORAGE_ACCT

logger "Finished running log analytics install python script"
exit $?
```

The **log_analytics_install.py file** is downloaded and then run to finish the log analytics install.

H)Explanation of Log Analytics Python file (log_analytics_install.py)

There are a total of **38 functions** inside this python file , which have different uses and it will be explained briefly ahead.

Name of all the functions

Sno	**Function name (def)**	Explanation of the function
1.	**main()**	This function calls 8 functions inside it: a) hdinsightlogging.initialize_root_logger() b) init_global_params() c) make_log_analytics_image_changes_if_necessary() d) get_and_set_cluster_properties_files() e) configure_mdsd_with_workspace_and_resource_id() f) download_and_configure_events() g) set_up_ambari_authentication() h) start_pipeline()
2.	**Init_global_params()**	It is used to initialize : a) global variables b) command line arguments c) Cluster manifest file d) Cluster type,name e) Variables related to Genva Control Plane Service f) Get Node Type, Resource ID
3.	**Get_cluster_type_mapping()**	It maps Hadoop_spark3_ha to spark. Note: This is a temporary fix because spark 3.0 does not map ambari blueprint name correctly
4.	**Create_resource_id()**	Gets resource url in the following form: /subscriptions/{SUBSCRIPTION}/resourceGroups/{Resource Group}/providers/{Provider Name}/clusters/{cluster Name}
5.	**get_node_type()**	Gets the type of node (Head, Worker, Zookeeper) using hostname.
6.	**Get_is_secure_cluster()**	Returns manifest security if it is true or false.

7.	**Determine_storage_endpoint_suffix()**	If the deployment environment is among these “Current”, “Dogfood”,”Production”,”TIP” then windows.core.net is returned otherwise the Geneva monitoring is not supported for the development environment.
8.	**Make_log_analytics_image_change_if_necessary()**	It calls two functions: a) Configure_mdsd_for_log_analytics() b) Configure_td_agent_for_log_analytics()
9.	**Configure_mdsd_for_log_analytics()**	It calls 4 functions: a) Make_syslog_owner_of_mdsd b) Remove_mdsd_default_read_access_for_others() c) Make_mdsd_conf_dir_writable() d) Add_import_statement_block_to_mdsd_conf()
10.	**Add_import_statement_block_to_mdsd_conf()**	If import element is not found then import element is inserted in the mdsd xml file
11.	**Make_syslog_owner_of_mdsd()**	Make syslog owner of mdsd file
12.	**Remove_mdsd_default_read_access_for_others()**	Removing read access for others to the mdsd file in /etc/default/mdsd
13.	**Make_mdsd_conf_dir_writable()**	Make /etc/mdsd.d file writable for syslog
14.	**Configure_td_agent_for_log_analytics()**	It calls 4 functions: a) Create_log_analytics_configuration_directory() b) Set_td_agent_conf_to_read_conf_subfiles() c) Create_log_analytics_cluster_properties_directory() d) Download_and_configure_td_agent_sudoers_file()
15.	**Create_log_analytics_configuration_directory()**	Creates 2 directories if they doesn’t exist a) /etc/td-agent/CurrentLogAnalyticsSourceConfiguration b) /etc/td-agent/CurrentLogAnalyticsOutputConfiguration
16.	**Set_td_agent_conf_to_read_conf_subfiles()**	Creates td agent conf file if it not exists at /etc/td-agent/td-agent.conf Adds the @inlcude statement to read from source and output configuration files
17.	**Create_log_analytics_cluster_properties_directory()**	Create a folder for cluster properties at /etc/td-agent/LogAnalyticsClusterProperties

18.	**Download_and_configure_td_agent_sudoers_file()**	It calls two functions: a) Download_td_agent_sudoers_file() b) Install_td_agent_sudoers_file()
19.	**Download_td_agent_sudoers_file()**	Download suders file in the destination /mnt/td-agent.txt
20.	**Ensure_td_agent_sudoers_file_valid()**	Check the validity of sudoer file with visudo -cf command.
21.	**Install_td_agent_sudoers_file()**	Install td-agent sudoers file in the path /etc/sudoers.d/td-agent
22.	**Get_and_set_cluster_properties_files()**	Adds two files related to cluster properties: a) Fully Qualified domain of the cluster b) Cluster Name If ESP Cluster then sets watch dog user details
23.	**Run_keytab_setup()**	This is run only when the cluster is ESP. Runs keytab files
24.	**Configure_mdsd_with_workspace_and_resource_id()**	It calls two function: a) Add_workspace_and_resource_id_details() b) Set_cluster_environment_endpoint()
25.	**Add_workspace_and_resource_id_details()**	Adds workspace id , workspace key and resource id to /etc/default/mdsd
26.	**Set_cluster_environment_endpoint()**	Sets cluster environment endpoint
27.	**Download_and_configure_events()**	It downloads a json file with all the configuration source and output files for that particular node type by calling **Download_node_type_configuration_jsons()** It then separates different type of configuration files list like td_source_confs, td_output_confs, mdsd_confs etc. Using the given list, configuration files of each type is downloaded and saved in their respective destinations inside the /etc/td-agent folder for td-agent conf files. Downloads and Configure td_agent_plugins
28.	**Download_node_type_configuration_jsons()**	Downloads json file containing list of different configuration files for different node types
29.	**Parse_required_node_type_configurations()**	Used to parse the json file and separate different configuration lists like mdsd_confs, td_source_confs, td_output_confs , etc.

30.	**Download_configurations()**	Downloads configuration files from the configuration lists and stores them in their respective destination paths.
31.	**Configure_event_files_with_workspace_info()**	Mdsd events are linked to Workspace ID
32.	**Add_event_import_statements_to_mdsd_conf()**	The mdsd events are added as elements in the mdsd xml file in /etc/mdsd.d/mdsd.xml
33.	**Download_td_agent_plugins()**	Downloads td_agent plugins from the list available in Constants file.
34.	**Make_td_agent_plugins_readable()**	The permissions are set using chmod recursively so that they are readable.
35.	**Set_up_ambari_authentication()**	Sets up ambari authentication
36.	**Remove_unsupported_configs()**	The config files which are not supported are removed from the list passed as parameter
37.	**Get_cluster_shape()**	Cluster shape is made up of cluster type and hdi_version.
38.	**Start_pipeline()**	It calls 3 functions: a) Restarts mdsd b) Restart td-agent service c) Restart collectd service

After this python file is run successfully the configuration files are downloaded as per the node type in which it is installed.

The output configuration files installed in each node can be checked using this link:

https://hdiconfigactions.blob.core.windows.net/loganalyticsmonitoring/node_type_configurations/{VALID_COMPONENT_TYPES}/{VALID_NODE_TYPES}.json

where

```
VALID_NODE_TYPES = ["interactivehivehead", "interactivehiveworker",
"interactivehivezookeeper",
                    "hbasehead", "hbaseworker", "hbasezookeeper",
                    "kafkahead", "kafkaworker", "kafkazookeeper",
                    "stormhead", "stormworker", "stormzookeeper",
"lowestfqdnstormhead",
                    "sparkhead", "sparkworker", "sparkzookeeper",
                    "hadoophead", "hadoopworker", "hadoopzookeeper",
                    "head","worker", "zookeeper",
                    "rangerhead"]
```

```
VALID_COMPONENT_TYPES = ["hadoop", "hbase", "hive", "interactivehive",
"kafka", "oozie", "spark", "storm", "yarn", "appstats"]
```