MFLib

Release 0.0.0b0

Fabric UKY Team

Jan 31, 2023

CONTENTS:

MFLib Measurement Framework Library makes it easy to add monitoring to your FABRIC Slice.

CHAPTER

ONE

MFLIB MEASUREMENT FRAMEWORK LIBRARY

Welcome to the FABRIC Measurement Framework Library. MFLib makes it easy to install monitoring systems to a FABRIC experimenter's slice. The monitoring system makes extensive use of industry standards such as Prometheus, Grafana, Elastic Search and Kibana while adding customized monitoring tools and dashboards for quick setup and visualization.

1.1 Documentation Resources

For more information about FABRIC vist fabric-testbed.net

1.1.1 Example Jupyter Notebooks

FABRIC Jupyter Examples GitHub repository contains many examples for using FABRIC from simple slice setup to advanced networking setups. Look for the MFLib section. These notebooks are designed to be easily used on the FABRIC JupyterHub

1.1.2 FABRIC Learn Site

FABRIC Knowledge Base

1.1.3 MFLib Python Package Documentation

See MFLib.pdf for package documentation.

1.2 MFLib Installation

S

MFLib may be installed using pip. `pip install --user fabrictestbed-mflib`

1.2.1 Installing via Source Code

If you need a development version, clone the git repo, then use pip to install.

```
git clone https://github.com/fabric-testbed/mflib.git
cd mflib
pip install --user .
```

1.3 Building & Deploying

1.3.1 Spinx Documentation

This package is documented using sphinx. The `source` directories are already created and populated with reStructuredText (.rst) files. The `build` directories are deleted and/or are not included in the repository,

Build HTML Documents

The sphinx theme furo is used. This may need to be installed using
`pip install furo`
To parse the markdown files (README.md) sphinx needs myst-parser.
`pip install myst-parser`
Build the documentation by running the following command from the root directory of the repo. `sphinx-build -b html docs/source/ docs/build/html`
The completed documentation may be accessed by clicking on `/docs/build/html/index.html`

Build PDF Document

Latex must be installed. For Debian use:

```
sudo apt install texlive-latex-extra
sudo apt install latexmk
```

Run the bash script to create the MFLIB.pdf documentation. MFLIB.pdf will be placed in the root directory of the repository. `./create_pdf_doc.sh`

Building Distribution Package

To build python package for PyPi run
`python setup.py sdist`

Uploading to PyPI

First test the package by uploading to test.pypi.org then test the install. `twine upload --repository-url https://test.pypi.org/legacy/ dist/*` Once install is good, upload to PiPy
`twine upload dist/*`

CHAPTER

TWO

MFLIB CORE

MFLib's Core functions are defined in this class. This class is the base class for all MFLib classes and is not meant to be used directly.

class mflib.core.**Core**(*local_storage_directory='/tmp/mflib'*)

MFLib core contains the core methods needed to create and interact with the Measurement Framework installed in a slice. It is not intended to be used by itself, but rather, it is the base object for creating Measurement Framework Library objects.

property bootstrap_status_file

The full path to the local copy of the bootstrap status file.

Returns:
String: The full path tp the local copy of the bootsrap status file.

property common_hosts_file

The full path to a local copy of the hosts.ini file.

Returns:
String: The full path to a local copy of the hosts.ini file.

core_class_version = '1.0.30'

An updatable version for debugging purposes to make sure the correct version of this file is being used. Anyone can update this value as they see fit. Should always be increasing.

Returns:
String: Version.sub-version.build

create(*service*, *data=None*, *files=[]*)

Creates a new service for the slice. :param service: The name of the service. :type service: String :param data: Data to be passed to a JSON file place in the service's meas node directory. :type data: JSON serializable object. :param files: List of filepaths to be uploaded. :type files: List of Strings

download_log_file(*service*, *method*)

Download the log file for the given service and method. Downloaded file will be stored locally for future reference. :param service: The name of the service. :type service: String :param method: The method name such as create, update, info, start, stop, remove. :type method: String :return: Writes file to local storage and returns text of the log file. :rtype: String

get_bootstrap_status(*force=True*)

Returns the bootstrap status for the slice. Default setting of force will always download the most recent file from the meas node. :param force: If downloaded file already exists locally, it will not be downloaded unless force is True. The downloaded file will be stored locally for future reference. :return: Bootstrap dict if any type of bootstraping has occured, None otherwise. :rtype: dict

`get_mfuser_private_key`(*force=True*)

Returns the mfuser private key. Default setting of force will always download the most recent file from the meas node. :param force: If downloaded file already exists locally, it will not be downloaded unless force is True. The downloaded file will be stored locally for future reference. :return: True if file is found, false otherwise. :rtype: Boolean

`property grafana_tunnel`

Returns the command for createing an SSH tunnel for accesing Grafana.

Returns:
String: ssh command

`property grafana_tunnel_local_port`

If a tunnel is used for grafana, this value must be set for the port.

Returns:
String: port number

`info`(*service*, *data=None*)

Gets inormation from an existing service. Strictly gets information, does not change how the service is running. :param service: The name of the service. :type service: String :param data: Data to be passed to a JSON file place in the service's meas node directory. :type data: JSON serializable object.

`property kibana_tunnel`

Returns the command for createing an SSH tunnel for accesing Kibana.

Returns:
String: ssh command

`property kibana_tunnel_local_port`

If a tunnel is used for Kibana, this value must be set for the port

`property local_mfuser_private_key_filename`

The local copy of the private ssh key for the mfuser account.

Returns:
String: The local copy of the private ssh key for the mfuser account.

`property local_mfuser_public_key_filename`

The local copy of the public ssh key for the mfuser account.

Returns:
String: The local copy of the public ssh key for the mfuser account.

`property local_slice_directory`

The directory where local files associated with the slice are stored.

Returns:
str: The directory where local files associated files are stored.

`property log_directory`

The full path for the log directory.

Returns:
String: The full path to the log directory.

`property meas_node`

The fablib node object for the Measurement Node in the slice.

Returns:
fablib.node: The fablib node object for the Measurement Node in the slice.

`property meas_node_ip`

The management ip address for the Measurement Node

Returns:
String: ip address

`mf_repo_branch = 'main'`

The git branch to be used for cloning the MeasurementFramework branch to the Measusrement Node.

`remove`(*services=[]*)

Stops a service running and removes anything setup on the experiment's nodes. Service will then need to be re-created using the create command before service can be started again.

`set_core_logger`()

Sets up the core logging file. Note that the self.logging_filename will be set with the slice name when the slice is set. Args: filename (_type_, optional): _description_. Defaults to None.

`property slice_name`

Returns the name of the slice associated with this object.

Returns:
String: The name of the slice.

`property slice_username`

The default username for the Measurement Node for the slice.

Returns:
String: username

`start`(*services=[]*)

Restarts a stopped service using existing configs on meas node.

`stop`(*services=[]*)

Stops a service, does not remove the service, just stops it from using resources.

`property tunnel_host`

If a tunnel is used, this value must be set for the localhost, Otherwise it is set to empty string.

Returns:
String: tunnel hostname

`update`(*service*, *data=None*, *files=[]*)

Updates an existing service for the slice. :param service: The name of the service. :type service: String :param data: Data to be passed to a JSON file place in the service's meas node directory. :type data: JSON serializable object. :param files: List of filepaths to be uploaded. :type files: List of Strings

CHAPTER

THREE

MFLIB

MFLib is the main class that is used to interact with the Measurement Framework set up in a user's FABRIC Experiment.

class mflib.mflib.**MFLib**(*slice_name=''*, *local_storage_directory='/tmp/mflib'*)

MFLib allows for adding and controlling the MeasurementFramework in a Fabric experiementers slice.

static addMeasNode(*slice*, *cores=4*, *ram=16*, *disk=500*)

Adds Measurement node and measurement network to unsubmitted slice object.

Args:
slice (fablib.slice): Slice object already set with experiment topology. cores (int, optional): Cores for measurement node. Defaults to 4 cores. ram (int, optional): _description_. Defaults to 16 GB ram. disk (int, optional): _description_. Defaults to 500 GB disk.

add_mflib_log_handler(*log_handler*)

Adds the given log handler to the mflib_logger. Note log handler needs to be created with set_mflib_logger first.

Args:
log_handler (logging handler): Log handler to add to the mflib_logger.

download_common_hosts()

Downloads hosts.ini file and returns file text. Downloaded hosts.ini file will be stored locally for future reference. :param service: The name of the service. :type service: String :param method: The method name such as create, update, info, start, stop, remove. :type method: String :return: Writes file to local storage and returns text of the log file. :rtype: String

init(*slice_name*)

Sets up the slice to ensure it can be monitored. Sets up basic software on Measurement Node and experiment nodes. Slice must already have a Measurement Node. See log file for details of init output.

Args:
slice_name (str): The name of the slice to be monitored.

Returns:
Bool: False if no Measure Node found or a init process fails. True otherwise.

instrumentize()

Instrumentize the slice. This is a convenience method that sets up & starts the monitoring of the slice. Sets up Prometheus, ELK & Grafana.

Returns:
dict : The output from each phase of instrumetizing.

remove_mflib_log_handler(*log_handler*)

Removes the given log handler from the mflib_logger.

Args:
log_handler (logging handler): Log handler to remove from mflib_logger

`restore_DNS_all_nodes()`

Restores the DNS to default if previously set. See set_DNS_all_nodes.

Returns:
string: "restored" if restored, "not needed" if not needed

`set_DNS(node)`

Sets the DNS on IPv6 only nodes to enable access to IPv4 sites.

`set_DNS_all_nodes()`

Sets DNS for nodes to allow them to access ipv4 networks.

Returns:
string: "set" if DNS set, "not needed" otherwise.

`set_mflib_logger()`

Sets up the mflib logging file. Filename is created from the self.logging_filename. Note that the self.logging_filename will be set with the slice when the slice name is set.

This method uses the logging filename inherited from Core.

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex

PYTHON MODULE INDEX

m

INDEX