
MFLib

Release 0.0.0dev4

Fabric UKY Team

Apr 03, 2023

CONTENTS

- 1 Documentation Resources 2**
 - 1.1 Example Jupyter Notebooks 2
 - 1.2 FABRIC Learn Site 2
 - 1.3 MFLib Python Package Documentation 2
- 2 MFLib Installation 3**
 - 2.1 Instalng via PIP 3
 - 2.2 Installing via Source Code 3
- 3 Building & Deploying 4**
 - 3.1 Spinx Documentation 4
 - 3.2 Distribution Package 5
- 4 MFLib 6**
- 5 MFLib Core 8**
- Python Module Index 13**

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.

DOCUMENTATION RESOURCES

For more information about FABRIC visit fabric-testbed.net

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](#)

1.2 FABRIC Learn Site

[FABRIC Knowledge Base](#)

1.3 MFLib Python Package Documentation

Documentation for the package is presented in several different forms (and maybe include later in this document):

- [ReadTheDocs](#)
- [MFLib.pdf](#)
- Or you may build the documentation from the source code. See Sphinx Documentation later in this document.

MFLIB INSTALLATION

2.1 Instalng via PIP

MFLib may be installed using PIP and PyPI [fabrictestbed-mflib](#)

```
pip install --user fabrictestbed-mflib
```

2.2 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 .
```

BUILDING & DEPLOYING

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,

API documentation can also be found at <https://fabrictestbed-mflib.readthedocs.io/>.

3.1.1 Build HTML Documents

Install the extra packages required to build API docs: (sphinx, furo theme, and myst-parser for parsing markdown files):

```
pip install -r docs/requirements.txt
```

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

3.1.2 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
```

If error “! LaTeX Error: Unknown graphics extension: .svg.” Just type enter to continue.

3.2 Distribution Package

MFLib package is created using [Flit](#) Be sure to create and commit the PDF documentation to GitHub before building and publishing to PyPi. The MFLib.pdf is included in the distribution.

```
python3 -m pip install flit
```

To build python package for PyPi run

```
flit build
```

3.2.1 Uploading to PyPI

First test the package by uploading to test.pypi.org then test the install.

```
flit publish --repository testpypi
```

Once install is good, upload to PiPy

```
flit publish --repository pypi
```

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', mf_repo_branch='main')
```

Bases: *Core*

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

```
static addMeasNode(slice, cores=4, ram=16, disk=500, network_type='FABNetv4', site='NCSA')
```

Adds Measurement node and measurement network to an unsubmitted slice object.

Parameters

- **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.
- **network_type** (*string, optional*) – *_description_*. Defaults to FABNetv4.
- **site** (*string, optional*) – *_description_*. Defaults to NCSA.

```
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.

Parameters

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.

```
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.

Parameters

slice_name (*str*) – The name of the slice to be monitored.

Returns

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

Return type

Bool

instrumentize()

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

Returns

The output from each phase of instrumentizing.

Return type

dict

mflib_class_version = '1.0.32'

remove_mflib_log_handler(*log_handler*)

Removes the given log handler from the mflib_logger.

Parameters

log_handler (*logging handler*) – Log handler to remove from mflib_logger

restore_DNS(*node*)**restore_DNS_all_nodes()**

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

Returns

“restored” if restored, “not needed” if not needed

Return type

string

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

“set” if DNS set, “not needed” otherwise.

Return type

string

set_mflib_logger()

Sets up the mflib logging file. The 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.

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', *mf_repo_branch*='main')

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

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

Return type

String

property `common_hosts_file`

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

Returns

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

Return type

String

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

Version.sub-version.build

Return type

String

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

Creates a new service for the slice.

Parameters

- **service** (*String*) – The name of the service.
- **data** (*JSON serializable object*) –
- **files** (*List of Strings*) – List of filepaths to be uploaded.

Returns

Dictionary of creation results.

Return type

dict

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. The downloaded file will be stored locally for future reference at self.bootstrap_status_file.

Parameters

force (*Boolean*) – If downloaded file already exists locally, it will not be downloaded unless force is True. .

Returns

Bootstrap dict if any type of bootstrapping has occurred, Empty dict otherwise.

Return type

Dictionary

get_mfuser_private_key(*force=True*)

Downloads the mfuser private key. Default setting of force will always download the most recent file from the meas node. The downloaded file will be stored locally for future reference at self.local_mfuser_private_key_filename.

Parameters

force (*Boolean*) – If downloaded file already exists locally, it will not be downloaded unless force is True.

Returns

True if file is found, false otherwise.

Return type

Boolean

property grafana_tunnel

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

Returns

ssh command

Return type

String

property grafana_tunnel_local_port

If a tunnel is used for grafana, this value must be set for the port. :returns: port number :rtype: String

info(*service, data=None*)

Gets inormation from an existing service. Strictly gets information, does not change how the service is running.

Parameters

- **service** (*String*) – The name of the service.

- **data** (*JSON Serializable Object*) – Data to be passed to a JSON file place in the service's meas node directory.

Returns

Dictionary of info results.

Return type

dict

property kibana_tunnel

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

Returns

ssh command

Return type

String

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

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

Return type

String

property local_mfuser_public_key_filename

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

Returns

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

Return type

String

property local_slice_directory

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

Returns

The directory where local files associated files are stored.

Return type

str

property log_directory

The full path for the log directory.

Returns

The full path to the log directory.

Return type

String

property meas_node

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

Returns

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

Return type
fablib.node

property meas_node_ip

The management ip address for the Measurement Node

Returns
ip address

Return type
String

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.

Parameters
services (*List of Strings*) – The names of the services to be removed.

Returns
List of remove result dictionaries.

Return type
List

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
The name of the slice.

Return type
String

property slice_username

The default username for the Measurement Node for the slice.

Returns
username

Return type
String

start(services=[])

Restarts a stopped service using existing configs on meas node.

Parameters
services (*List of Strings*) – The name of the services to be restarted.

Returns
List of start result dictionaries.

Return type
List

stop(services=[])

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

Parameters

services (*List of Strings*) – The names of the services to be stopped.

Returns

List of stop result dictionaries.

Return type

List

property tunnel_host

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

Returns

tunnel hostname

Return type

String

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

Updates an existing service for the slice.

Parameters

- **service** (*String*) – The name of the service.
- **data** (*JSON Serializable Object*) – Data to be passed to a JSON file place in the service's meas node directory.
- **files** (*List of Strings*) – List of filepaths to be uploaded.

Returns

Dictionary of update results.

Return type

dict

PYTHON MODULE INDEX

m

`mflib.core`, [8](#)

`mflib.mflib`, [6](#)

A

`add_mflib_log_handler()` (*mflib.mflib.MFLib method*), 6
`addMeasNode()` (*mflib.mflib.MFLib static method*), 6

B

`bootstrap_status_file` (*mflib.core.Core property*), 8

C

`common_hosts_file` (*mflib.core.Core property*), 8
`Core` (*class in mflib.core*), 8
`core_class_version` (*mflib.core.Core attribute*), 8
`create()` (*mflib.core.Core method*), 8

D

`download_common_hosts()` (*mflib.mflib.MFLib method*), 6
`download_log_file()` (*mflib.core.Core method*), 9

G

`get_bootstrap_status()` (*mflib.core.Core method*), 9
`get_mfuser_private_key()` (*mflib.core.Core method*), 9
`grafana_tunnel` (*mflib.core.Core property*), 9
`grafana_tunnel_local_port` (*mflib.core.Core property*), 9

I

`info()` (*mflib.core.Core method*), 9
`init()` (*mflib.mflib.MFLib method*), 6
`instrumentize()` (*mflib.mflib.MFLib method*), 6

K

`kibana_tunnel` (*mflib.core.Core property*), 10
`kibana_tunnel_local_port` (*mflib.core.Core property*), 10

L

`local_mfuser_private_key_filename` (*mflib.core.Core property*), 10

`local_mfuser_public_key_filename` (*mflib.core.Core property*), 10
`local_slice_directory` (*mflib.core.Core property*), 10
`log_directory` (*mflib.core.Core property*), 10

M

`meas_node` (*mflib.core.Core property*), 10
`meas_node_ip` (*mflib.core.Core property*), 11
`MFLib` (*class in mflib.mflib*), 6
`mflib.core`
 module, 8
`mflib.mflib`
 module, 6
`mflib_class_version` (*mflib.mflib.MFLib attribute*), 7
module
 mflib.core, 8
 mflib.mflib, 6

R

`remove()` (*mflib.core.Core method*), 11
`remove_mflib_log_handler()` (*mflib.mflib.MFLib method*), 7
`restore_DNS()` (*mflib.mflib.MFLib method*), 7
`restore_DNS_all_nodes()` (*mflib.mflib.MFLib method*), 7

S

`set_core_logger()` (*mflib.core.Core method*), 11
`set_DNS()` (*mflib.mflib.MFLib method*), 7
`set_DNS_all_nodes()` (*mflib.mflib.MFLib method*), 7
`set_mflib_logger()` (*mflib.mflib.MFLib method*), 7
`slice_name` (*mflib.core.Core property*), 11
`slice_username` (*mflib.core.Core property*), 11
`start()` (*mflib.core.Core method*), 11
`stop()` (*mflib.core.Core method*), 11

T

`tunnel_host` (*mflib.core.Core property*), 12

U

`update()` (*mflib.core.Core method*), 12