# Chapter 1

# Software Installation/Getting Started

## 1.1 Introduction

This chapter describes how to download and set up METplus. METplus has been developed and tested on the Debian Linux operating system.

## 1.2 Supported architectures

METplus was developed on Debian Linux and is supported on this platform.

## 1.3 Programming/scripting languages

METplus is written in Python 2.7. METplus is intended to be a tool for the modeling community to use and adapt. As users make upgrades and improvements to the tools, they are encouraged to offer those upgrades to the broader community by offering feedback to the developers.

## 1.4 Pre-requisites

The following software is required to run METplus:

- Python 2.7

- R version 3.2.5 [1]
- nco (netCDF operators)
- MET version 6.1 or above
- Basic familiarity with MET
- GitHub account (if you plan on contributing code to METplus)

## 1.5 METplus directory structure

Once you have cloned the METplus from the GitHub repository at https://github.com/NCAR/METplus to a location on your host, change directories to the METplus directory. You should have the following directory structure:

```
METplus
    doc
    internal_tests
    parm
    sorc
    ush
    README.md
```

The top-level METplus directory consists of a README.md file and several subdirectories.

The doc/ directory contains documentation for users (PDF) and Doxygen files that are used to create the developer documentaton. The Doxygen documentation can be created and viewed via web browser if the developer has Doxygen installed on the host.

The internal_tests/ directory contains unit test scripts that are only relevant to METplus developers and contributors.

The parm/ directory contains all the configuration files for MET and METplus.

The sorc/ directory contains Doxygen executables to generate documentation for developers.

The src/ directory contains the source code for each of the wrappers in METplus.

The ush/ directory contains the Python wrappers to the MET tools.

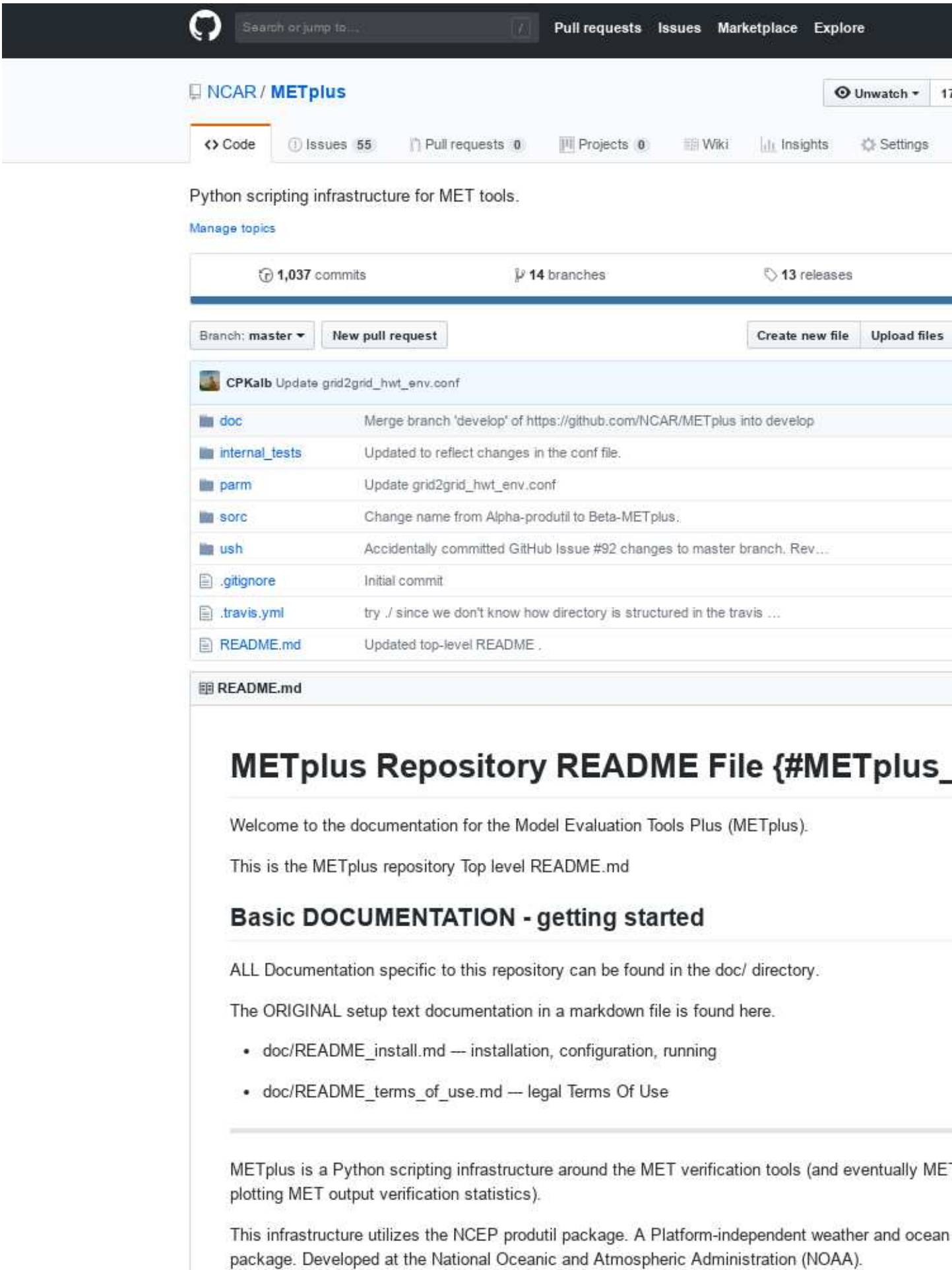
[1] R version 3.2.5 is required when the tcmpr_plotter_wrapper.py wraps the plot_tcmpr.R script. Please refer to Chapter 21 Plotting and Graphics Support for more information about plot_tcmpr.R.

## 1.6 Getting the METplus source code

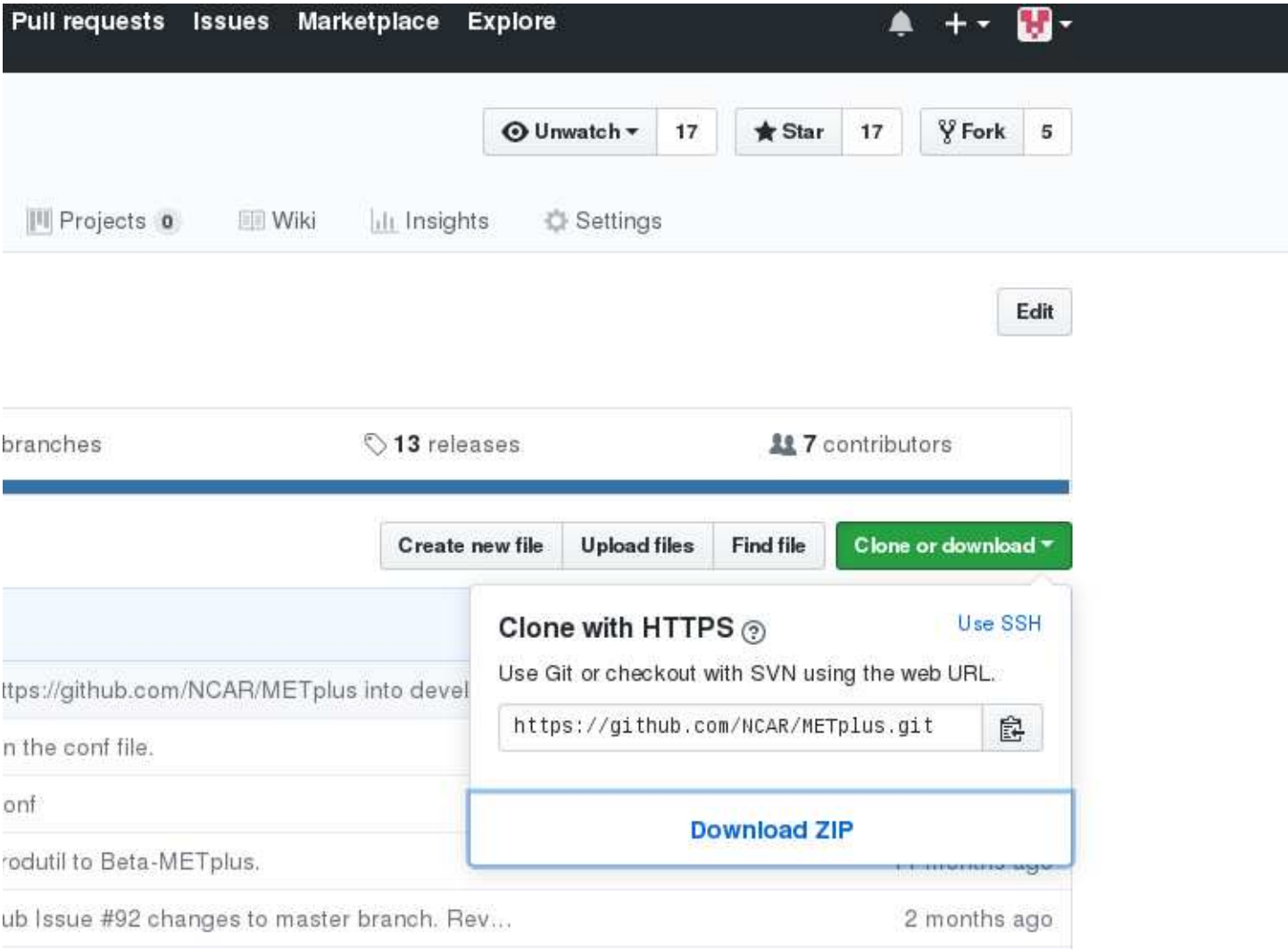
The METplus source code is available for download from a public GitHub repository.

### 1.6.1 Get the source code via your Web Browser

- On your local host (or wherever you wish to install the METplus code) create a directory where you want the code to reside
- Open the browser of your choice and navigate to https://github.com/NCAR/METplus. You will see something like the following:


Search or jump to...
Pull requests
Issues
Marketplace
Explore
NCAR / METplus
Unwatch
Code
Issues 55
Pull requests 0
Projects 0
Wiki
Insights
Settings
Python scripting infrastructure for MET tools.
Manage topics
1,037 commits
14 branches
13 releases
Branch: master
New pull request
Create new file
Upload files
CPKalb Update grid2grid_hwt_env.conf
doc
Merge branch 'develop' of https://github.com/NCAR/METplus into develop
internal_tests
Updated to reflect changes in the conf file.
parm
Update grid2grid_hwt_env.conf
sorc
Change name from Alpha-produtil to Beta-METplus.
ush
Accidentally committed GitHub Issue #92 changes to master branch. Rev...
.gitignore
Initial commit
.travis.yml
try ./ since we don't know how directory is structured in the travis ...
README.md
Updated top-level README .
README.md
METplus Repository README File {#METplus_
Welcome to the documentation for the Model Evaluation Tools Plus (METplus).
This is the METplus repository Top level README.md
Basic DOCUMENTATION - getting started
ALL Documentation specific to this repository can be found in the doc/ directory.
The ORIGINAL setup text documentation in a markdown file is found here.
doc/README_install.md — installation, configuration, running
doc/README_terms_of_use.md — legal Terms Of Use
METplus is a Python scripting infrastructure around the MET verification tools (and eventually MET
plotting MET output verification statistics).
This infrastructure utilizes the NCEP produtil package. A Platform-independent weather and ocean
package. Developed at the National Oceanic and Atmospheric Administration (NOAA).

- You should be directed to the 'master' branch, verify this by looking at the button labelled 'Branch' in the upper left corner of your window, directly beneath the solid blue horizontal line.
- Click on the green "Clone or download" button near the top right of the page.
- A box appears with "Clone with HTTPS" label
- Click on the blue text: "Download Zip" :




- Your browser should prompt you on what to do with this file. Save it to the directory you created above
- cd to the directory where you saved the code. You should see the file METplus-master.zip
- Uncompress the file:
  - Linux/Unix:
  - unzip METplus-master.zip

  - You should now have a METplus-master directory
    * If you downloaded the code via the command line, you will get a METplus directory rather than METplus-master.
    * GitHub appends the '-master' to the name to emphasize that is is from the master branch
    * To avoid clutter and confusion, you can now remove the METplus-master.zip (optional)

### 1.6.2 Get the source code via Command line

- On your local host (or wherever you wish to install the METplus code) create a directory where you want the code to reside
- cd to the directory you just created.
- On the command line, enter the following:
  - git clone https://github.com/NCAR/METplus
  - The source code should appear under the METplus directory
- To update your copy, cd to your METplus install directory: */path/to/*METplus and enter *git pull* at the command line

## 1.7 Set up your environment

Environment variables need to be set to allow the METplus application to be run from any directory and for locating the necessary Python modules. There is an option to set the JLOGFILE environment variable, which indicates where JLOGS will be saved. JLOGS provide information pertinent to the configuration-file framework. If this environment is unset, then output from the configuration framework will be directed to stdout (your display).

Add the following information to your .cshrc (C shell) or .bashrc (Bash shell):

**.cshrc:**

- Open your .cshrc file and do the following:
- To your PATH, add: *full-path-to-METplus*/ush
- To your PYTHONPATH, add: *full-path-to-METplus*/ush:*full-path-to-METplus*/parm
- Optional: add JLOGFILE variable and set to *full-path-to-save-jlog-files*
- close your .cshrc file and run `source ~/.cshrc`

- *e.g.*
  set path = (other_path_entries /home/username/METplus/ush
  setenv PYTHONPATH /home/username/METplus/ush:/home/username/METplus/parm:$PYTHONPATH
  # optional
  setenv JLOGFILE /home/username/jlog_out

**.bashrc:**

- Open your .bashrc file and do the following:
- To your PATH, add : *full-path-to-METplus*/ush
- To your PYTHONPATH, add *full-path-to-METplus*/parm
- Optional: add a JLOGFILE environment variable and set it to the directory where you want the logs to reside
- close your .bashrc file and run `source ~/.bashrc`
- *e.g.*
  export PATH=/home/username/METplus/ush:$PATH
  export PYTHONPATH="/home/username/METplus/ush:/home/username/METplus/parm:/$PYTHONPATH"
  #optional
  export JLOGFILE=/home/username

## 1.8 Running METplus

Running METplus involves invoking the Python script master_metplus.py from any directory followed by a list of configurationfiles (file path relative to the *path_to_METplus_install_dir*/METplus/parm directory).

Example: Using a default configuration
>master_metplus.py
Does nothing, a usage message appears, indicating that other config files are required to perform useful tasks.

Example: Using a use-case configuration
>master_metplus.py -c use_cases/feature_relative/feature_relative.conf \
Runs METplus using the defaults set in the three config files found in parm/metplus_config. Any variables defined in these three config files can be over-ridden in the parm/use_cases/feature_relative/feature_relative.conf file. METplus will run using the values specified in the feature_relative.conf file.

Example: Using example configuration to perform specific evaluation (e.g. Model 1 vs. Obs1, Model 1 vs Obs 2, Model 2 vs. Obs 1, etc.)
>master_metplus.py -c use_cases/feature_relative/feature_relative.conf \
-c use_cases/feature_relative/example/series_by_lead_all_fhrs.conf
This runs METplus using the defaults set in the three config files found in parm/metplus_config, where variables can be over-ridden by parm/use_cases/feature_relative/feature_relative.conf or in parm/use_cases/feature_relative/example/series_by_lead_all_fhrs.conf. The order in which conf files are called is important. Variables that are defined in intermediate conf files will be over-ridden by the same variables set in the conf file following it, or the last conf file.