
Space Telescope Environment (`stenv`)

Release 2024.08.14.post3.dev0+2f0c3e4

Space Telescope Science Institute

Sep 30, 2024

CONTENTS

1	Getting Started	3
2	Frequently Asked Questions	9
3	Developer Notes	11
4	Pipeline Releases	13
5	Windows Support	17

`stenv` is an installable Conda environment maintained by the [Space Telescope Science Institute \(STScI\)](#) in Baltimore, Maryland. This environment provides tools and utilities required to process and analyze data from the Hubble Space Telescope (HST), James Webb Space Telescope (JWST), and others.

GETTING STARTED

1.1 Conda Basics

`stenv` defines a Conda environment, which is a set of packages installed together at specific versions. A Conda environment is designed to be isolated from system packages, and can be **activated** to switch the current context (PATH, environment variables, available binaries, Python installation, etc.) to an isolated instance that is separate from the system. (This is similar to using `source bin/activate`, if you are familiar with Python `virtualenvs`). This has the advantage of allowing several separate installations of Python packages and other tools without cluttering the system installation, allowing switching between use cases or package contexts at will.

1.2 Installation

1.2.1 Install Conda

A Conda distribution provides the `micromamba` / `mamba` / `conda` command, which lets you create, manage, and switch to (activate) environments. Try running `micromamba`, `mamba`, or `conda` in your terminal. If you get `command not found` (or similar), see below to install.

`mamba` is a rewrite of `conda` that is much faster at resolving dependencies with near-parity of commands. `micromamba` is `mamba` packaged into a single binary, which makes installation and maintenance much easier. For these reasons, I recommended you use `micromamba`.

Run the following in your terminal to install `micromamba`:

```
"${SHELL}" <(curl -L micro.mamba.pm/install.sh)
```

On macOS, you can alternatively install `micromamba` using the [Homebrew package manager](#), if you have it installed:

```
brew install micromamba
micromamba shell init
```

You may also follow [these installation instructions](#).

Follow [these instructions to install Miniforge](#), which includes the `mamba` command in its base environment.

Important

Remember to run `mamba init` after installing. This is required in order to set up your shell to activate and deactivate environments.

```
mamba init
```

 **Caution**

The Anaconda organization has [updated their terms of service](#) to indicate that any usage of their services requires a paid license, if used by an organization of more than 200 users. This includes pulling packages from the Anaconda `defaults` channels, as well as installing `conda` itself.

We recommend that you use `mamba` and pull packages from the `conda-forge` channel, instead of using `conda` and the `defaults` channels.

Follow [these instructions](#) to install `Miniconda` which includes the `conda` command in its base environment.

 **Important**

Remember to run `conda init` after installing. This is required in order to set up your shell to activate and deactivate environments.

```
conda init
```

1.2.2 Choose an `stenv` release

Now that you have a Conda installation, you should choose a release of `stenv` from the [Releases page](#) and choose the environment definition file from the `Assets` section that corresponds with your platform.

The screenshot shows the GitHub release page for the repository `spacetelescope/stenv`. The release version is `2023.02.16`, marked as the latest. The page includes a navigation bar with links for Code, Issues (6), Pull requests (4), Discussions, Actions, Security, and Insights. A search bar is present in the top right. The release details section shows the release was made 2 weeks ago by `zacharyburnett` with commit `2a421e5`. A "Compare" dropdown is available. The "What's Changed" section lists several updates, including consolidating environment variables, using `extra-specs` instead of `sed`, caching CRDS tests, adding a version updater, removing `stable` and `dev` environments, adding explicit references to scientific packages, testing `jwt` from source, pinning `stcal` and `stdatamodels`, and using `conda env export` for micromamba-built environments. A link to the full changelog is provided. The "Contributors" section lists `zacharyburnett`. The "Assets" section lists 8 assets, including environment files for Linux and macOS with Python 3.10, 3.11, and 3.9, source code (zip and tar.gz), and their respective sizes and upload times.

Every release is available for several combinations of operating system and Python version. The name of the release file indicates which is which. For example, a release of `stenv` for Python 3.11 on Linux will be named something like `stenv-Linux-X64-py3.11-YYYY.MM.DD.yaml` (where `YYYY.MM.DD` is the date of the release). Unless you have particular requirements, you should choose the newest (highest-numbered) Python version available.

Note

Version numbers aren't real numbers; a hypothetical Python 3.20 would be newer than Python 3.2.

Warning

Can't find the release you need? Building and testing environments on supported platforms may take several minutes; for new releases, you may need to wait for the [associated workflow job](#) to finish before environment files

are available.

Note

Every Conda environment has a name, specified by the `--name` or `-n` option. If you include the version numbers in the name, it will be easier to keep track of which version of `stenv` you have. Therefore, I recommend using a more descriptive name than `stenv` for your environment; for example, use something like `stenv-py3.11-2023.01.01` (changed as needed to match the version you chose).

Download the file corresponding to your platform and desired Python version, then run the following command in a terminal using the file you downloaded (in this example `stenv-Linux-py3.10-2023.02.16.yaml`):

```
micromamba env create --name stenv --file ~/Downloads/stenv-Linux-py3.10-2023.02.16.yaml
```

```
mamba env create --name stenv --file ~/Downloads/stenv-Linux-py3.10-2023.02.16.yaml
```

```
conda env create --name stenv --file ~/Downloads/stenv-Linux-py3.10-2023.02.16.yaml
```

Note

If the build does not succeed on your system, please refer to [stenv doesn't build on my system; what do I do?](#)

1.3 Activating an environment

Environments let you install packages while isolating them from the rest of your system, and even each other. Even though we just created an environment, we will not be able to import the new packages yet:

```
$ python -c 'import jwst; print("ok")'
Traceback (most recent call last):
  File "<string>", line 1, in <module>
ModuleNotFoundError: No module named 'jwst'
```

In order to access the packages in `stenv`, you must first activate the environment you just created:

Important

If you chose another name when creating the environment, use that here instead.

```
micromamba activate stenv
```

```
mamba activate stenv
```

```
conda activate stenv
```

Activating a Conda environment changes which Python interpreter and packages are in use for that session (i.e. terminal window). Now, if you try to `import jwst`:

```
(stenv) $ python -c 'import jwst; print("ok")'  
ok
```

Every time you open a new terminal window, you will need to activate the environment before you can use software included in stenv.

Note

You can show installed packages available within a Conda environment with `conda list`:

```
micromamba list
```

```
mamba list
```

```
conda list
```

To deactivate an environment and return your shell to normal, close your terminal window or run `conda deactivate`:

```
micromamba deactivate
```

```
mamba deactivate
```

```
conda deactivate
```

1.4 Deleting an environment

To delete an environment with all of its packages, run `conda env remove --name <name>`:

Important

If you chose another name when creating the environment, use that here instead.

```
micromamba env remove --name stenv
```

```
mamba env remove --name stenv
```

```
conda env remove --name stenv
```


FREQUENTLY ASKED QUESTIONS

2.1 stenv doesn't build on my system; what do I do?

You can use the environment definition YAML file (*environment.yaml*) in the root of the repository:

```
curl -L https://raw.githubusercontent.com/spacetelescope/stenv/main/environment.yaml -o ~
↳/Downloads/stenv.yaml
micromamba env create --name stenv --file ~/Downloads/stenv.yaml
```

```
mamba env create --name stenv --file https://raw.githubusercontent.com/spacetelescope/
↳stenv/main/environment.yaml
```

```
conda env create --name stenv --file https://raw.githubusercontent.com/spacetelescope/
↳stenv/main/environment.yaml
```

This environment is unpinned, meaning it may take some time to resolve dependency versions. Additionally, the resulting package versions may not have been tested for your platform.

Warning

stenv does not currently support a native Windows installation. To build stenv on Windows, see [Windows Support](#).

2.2 Why isn't _____ package in stenv?

Not all STScI packages are included in the base stenv environment; some packages are not supported and / or deprecated, and some are deemed too niche (or dependent on too many extra packages) to be included for all users.

To install a package in your local environment, you can use `pip install` while the environment is activated:

```
micromamba activate stenv
pip install <package_name>
```

```
mamba activate stenv
pip install <package_name>
```

```
conda activate stenv
pip install <package_name>
```

To request that a new package be added to `stenv` (*environment.yaml*) for all users, see *Adding a package to stenv*.

2.3 What about Astroconda?

Astroconda, historically maintained by STScI as a Conda software channel, provides data analysis tools and pipelines via the Conda package management system.

Warning

Astroconda is no longer supported as of **February 1st, 2023**.

`stenv` supersedes Astroconda as a STScI software distribution; it supports most of the packages in Astroconda, works with all current versions of Python, and provides a common environment for both the Hubble Space Telescope (HST) and James Webb Space Telescope (JWST) pipelines. Additionally, while Astroconda primarily uses Conda recipes to build and serve packages, which need to be updated separately from PyPI releases, `stenv` draws most of its packages directly from PyPI with `pip` (though it still requires use of a Conda environment for `hstcal` and `fitsverify`, which are provided by `conda-forge`).

DEVELOPER NOTES

stenv consists of several parts:

1. an unpinned Conda environment definition YAML file *environment.yaml*
2. a GitHub Actions CI workflow that automatically builds and tests the environment on several platforms
3. regular GitHub releases with attached constrained Conda environment definition YAML files for every tested platform

3.1 environment.yaml

```
# this is the base unpinned environment for `stenv`; to manually resolve an environment,
↳ from this file, run the following:
# conda env create -n stenv -f https://raw.githubusercontent.com/spacetelescope/stenv/
↳ main/environment.yaml

channels:
  - conda-forge

dependencies:
  - astropy
  - dask
  - fitsverify
  - hstcal>=3.0.3
  - h5py
  - ipython
  - jupyter
  - numpy<2.0
  - pip
  - python>=3.10
  - scipy
  - pip:
    - acstools
    - asdf
    - calcos
    - ccdproc
    - costools
    - crds
    - drizzlepac
    - ginga
```

(continues on next page)

(continued from previous page)

```

- jwst
- nictools
- pysynphot
- reftools
- stcal
- stdatamodels
- stistools
- stsynphot
- stregion
- synphot
- wfc3tools
- wfpc2tools

```

To build an environment from this unpinned environment definition, run the following:

```

curl -L https://raw.githubusercontent.com/spacetelescope/stenv/main/environment.yaml -o ~
↪/Downloads/stenv.yaml
micromamba env create --name stenv --file ~/Downloads/stenv.yaml

```

```

mamba env create --name stenv --file https://raw.githubusercontent.com/spacetelescope/
↪stenv/main/environment.yaml

```

```

conda env create --name stenv --file https://raw.githubusercontent.com/spacetelescope/
↪stenv/main/environment.yaml

```

3.2 Adding a package to stenv

To request that a new package be added to `stenv`, please [create a new issue in the repository](#).

Issue: Request new package

Add a new package to `stenv` If this doesn't look right, [choose a different type](#).

add ``<package>`` to environment

Write Preview

<!-- Feel free to modify this placeholder text to be relevant to your request: -->
`<package>` should be added to the base `stenv` environment. It can be found at
<https://github.com/spacetelescope/<package>>

<!-- The default environment of `stenv` represents the "basic" software stack for work with space telescope data. If you would like to add a package to this environment, please consider the following: -->
- [] This package is useful or relevant to a significant amount of `stenv` users.
- [] This package does not require specific versions of other packages in `stenv` (inclusion would not introduce significant backward constraints to the requirements).
- [] This package's tests do not take an overly long time to complete, and test data files are not significantly large.

<!-- If any of the above are not true, this package might not be suitable for inclusion in the base environment;
Attach files by dragging & dropping, selecting or pasting them.

Styling with Markdown is supported

Submit new issue

Assignees: No one—assign yourself

Labels: add package

Projects: None yet

Milestone: No milestone

Development: Shows branches and pull requests linked to this issue.

Helpful resources: [GitHub Community Guidelines](#)

PIPELINE RELEASES

 **Note**

A working conda installation (Miniconda, Mamba, Anaconda, etc.) is required (see *Install Conda*).

 **Warning**

stenv does not support Python 2.

 **Warning**

stenv does not support 32-bit operating systems.

Pipeline releases differ from the standard software stack and serve a different purpose. The release files, described below, are immutable snapshots of STScI operational software and can be used to replicate the environment used by STScI to perform mission-specific data processing. Be aware that upgrading packages with `conda update` is not recommended, as it will likely introduce unwanted bugs and / or break the environment all together.

If you have any questions, comments, or concerns related to pipeline releases, please feel free to contact help@stsci.edu

4.1 Installation

Pipeline release installations use the following `conda create` command format:

```
conda create -n demo_2016.1 --file http://ssb.stsci.edu/releases/hstdp/2016.1/hstdp-2016.1-  
↳1-linux-py35.0.txt  
conda activate demo_2016.1
```

 **Warning**

The URL shown in this example does not reflect the latest iteration available. Please consult the *File URLs* section to ensure you are installing the correct release.

4.2 File URLs

Select the URL that matches your intended platform and environment.

4.2.1 HST Data Processing (HSTDP)

HSTDP was previously known as *OPUS*.

Instructions for installation of each delivery may be found in the respective subdirectories of the releases repository: <https://github.com/astroconda/astroconda-releases/tree/master/caldp>

Warning

The repository is located within the Astroconda GitHub organization for legacy reasons. Astroconda is no longer supported as of **February 1st, 2023**.

```
20221010
20220527
20220406
20220214
20220127
20211129
20211119
20210928
20210827
20210721
20210505
20210415
20210323
20201208
20201012
20200812
20200708
20200611
20200421
20200323
```

Historical deliveries used an older naming convention: <https://github.com/astroconda/astroconda-releases/tree/master/hstdp>

```
2019.5.2
2019.5.1
2019.5
2019.4
2019.3c
2019.3b
2019.3a
2019.3
2019.2
2018.3a
2018.3
```

(continues on next page)

(continued from previous page)

```

2018.1
2017.3
2017.2a
2017.2
2017.1
2016.2
2016.1

```

4.3 Continuous Integration

This example BASH function provides a starting point for users intending to execute pipeline software from within a continuous integration environment. This installation method is unsupported and your mileage may vary. Use at your own risk.

```

function get_pipeline()
{
    # Do we have enough arguments?
    if [[ $# < 3 ]]; then
        echo "Not enough arguments."
        return 1
    fi

    # Setup basic argument list      & Example Input(s)
    local conda_env="$1"             # hst_env
    local name="$2"                  # hstdp, ...
    local build="$3"                 # 2017.2, 2016.2 ...
    local python_version="$4"        # py[35, 27, ...]
    local iteration="$5"             # final | post[0, 1, 2, ...]

    # Detect platform
    local _platform=$(uname -s)
    local platform=""

    # Convert platform string to match file naming convention
    if [[ ${_platform} == Linux ]]; then
        platform="linux"
    elif [[ ${_platform} == Darwin ]]; then
        platform="osx"
    else
        echo "Unsupported platform: ${_platform}"
        return 1
    fi
    unset _platform

    # Handle optional arguments.
    if [[ -z ${python_version} ]]; then
        # Notice the "py" prefix and condensed version here
        python_version="py35"
    fi
}

```

(continues on next page)

(continued from previous page)

```
if [[ -z ${iteration} ]]; then
    iteration="final"
fi

# Assemble pipeline spec file URL
local ac_root="http://ssb.stsci.edu/releases"
local ac_base="${ac_root}/${name}/${build}"
local ac_spec="${name}-${build}-${platform}-${python_version}.${iteration}.txt"
local ac_url="${ac_base}/${ac_spec}"

# Perform installation
conda create -q -n "${conda_env}" --file "${ac_url}"
return $?
}

#
# Usage example:
#
# Silently generate a pipeline environment called "hst_env"
get_pipeline hst_env hstdp 2017.2

# Enter environment
source activate hst_env

# ... do work ...
# EOF
```

WINDOWS SUPPORT

`stenv` does not currently support Windows, as `hstcal`, `fitsverify`, and the `jwst` calibration pipeline are not built or tested on Windows platforms.

If you would like to run `stenv` on Windows, you can use the *Windows Subsystem for Linux (WSL)*, an optional Windows feature that provides a functioning Linux terminal with access to the host operating system.

After installing your Linux system, install `stenv` by following the *Getting Started* instructions.

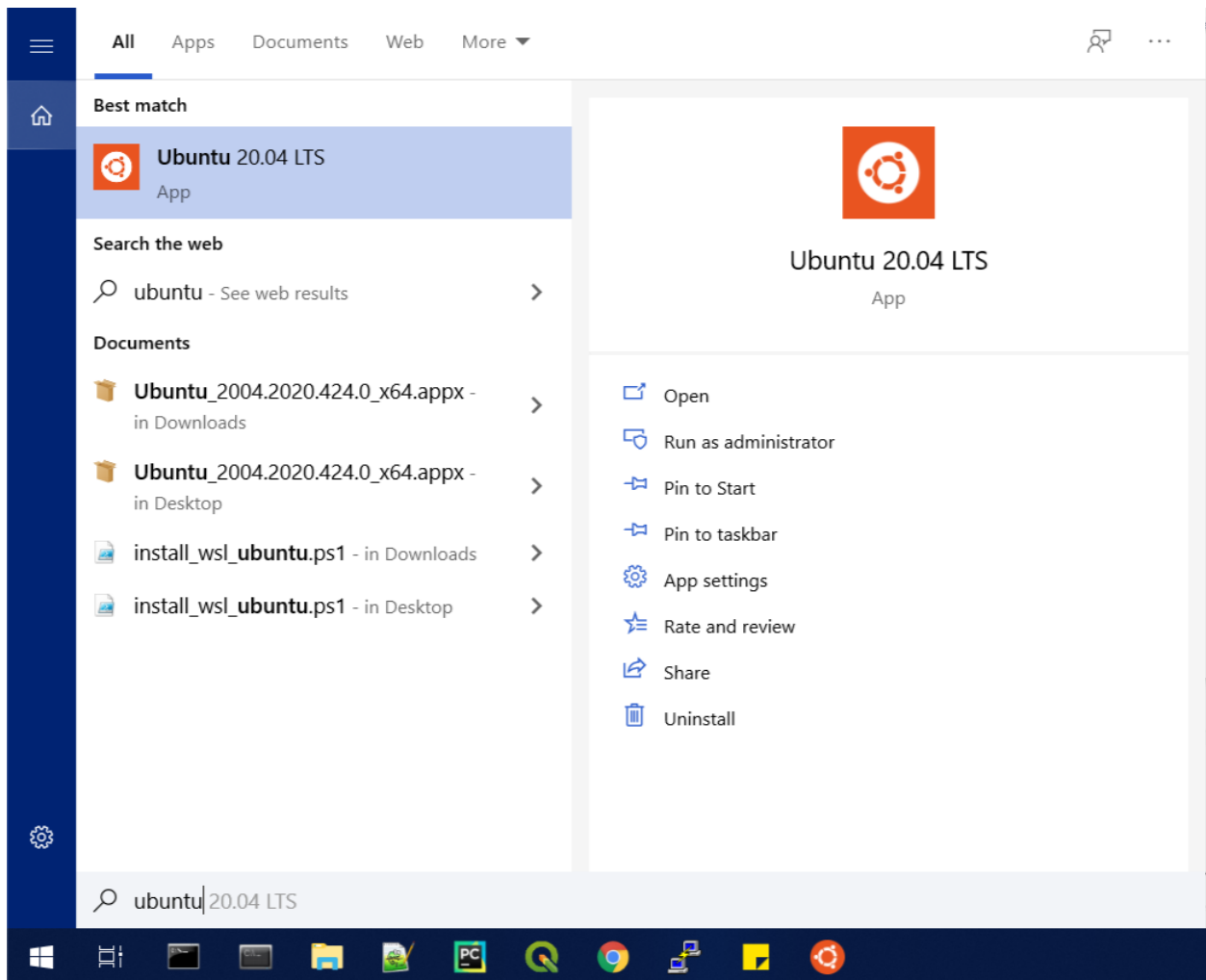
5.1 Windows Subsystem for Linux (WSL)

Windows Subsystem for Linux (WSL) is a platform provided by Microsoft onto which a Linux distribution (i.e. Ubuntu) can be installed. It provides a Linux shell within Windows that contains a functional Linux operating system, and thus can compile and run programs meant for Linux within that environment.

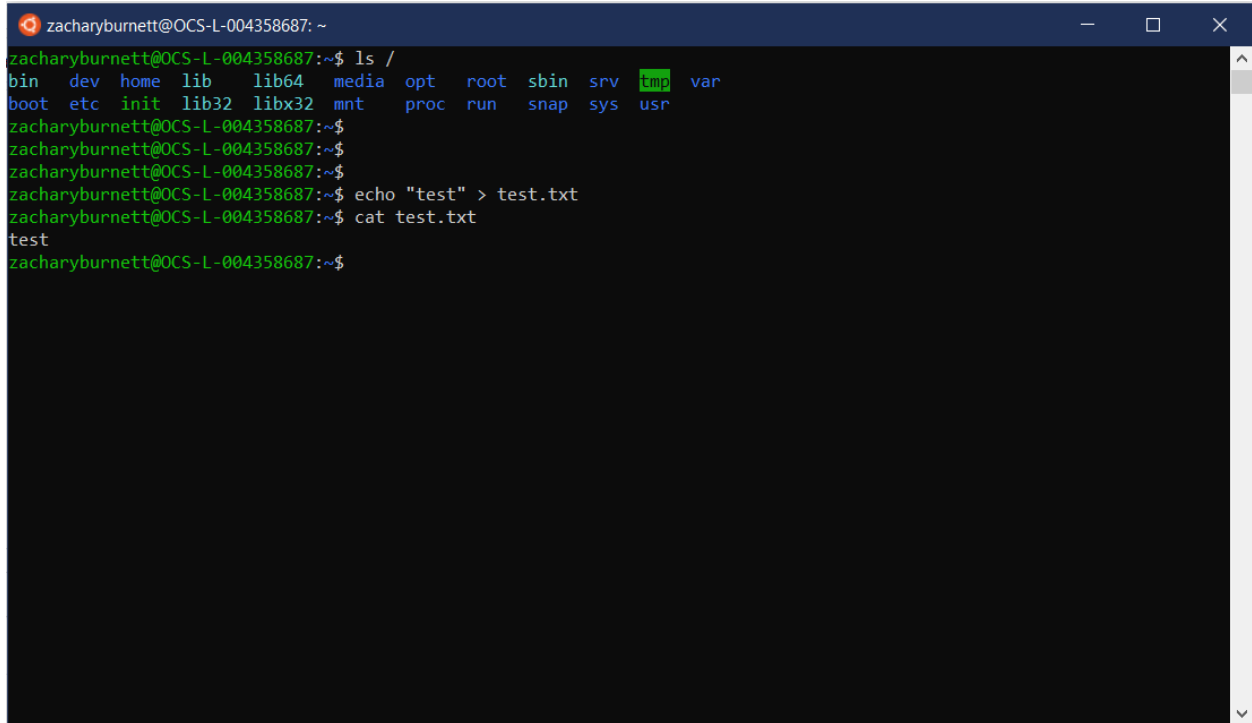
To install WSL on your machine, follow [Microsoft's provided installation instructions](#):

```
wsl --install
```

Now, type Ubuntu (or whatever other distribution you chose) into the Start Menu:



This will boot the operating system and start the installation within WSL, after which it will prompt you to set a username and password.

A terminal window titled 'zacharyburnett@OCS-L-004358687: ~' with standard window controls. The terminal shows the following commands and output:

```
zacharyburnett@OCS-L-004358687:~$ ls /
bin  dev  home  lib  lib64  media  opt  root  sbin  srv  tmp  var
boot  etc  init  lib32  libx32  mnt  proc  run  snap  sys  usr
zacharyburnett@OCS-L-004358687:~$
zacharyburnett@OCS-L-004358687:~$
zacharyburnett@OCS-L-004358687:~$ echo "test" > test.txt
zacharyburnett@OCS-L-004358687:~$ cat test.txt
test
zacharyburnett@OCS-L-004358687:~$
```

After installation completes, you now have a functional Linux operating system that you can access in the same way.

You have access to many Windows utilities from within the Linux shell, and many WSL utilities from the Windows shell. By default, the Windows partition is mounted at `/mnt/c`.

Microsoft provides [documentation on interoperability between Windows and WSL](#).

If you have issues with `stenv`, please [create a new GitHub issue](#) or contact one of the following help desks:

- **HST Help Desk:** <https://stsci.service-now.com/hst>
- **JWST Help Desk:** <https://stsci.service-now.com/jwst>