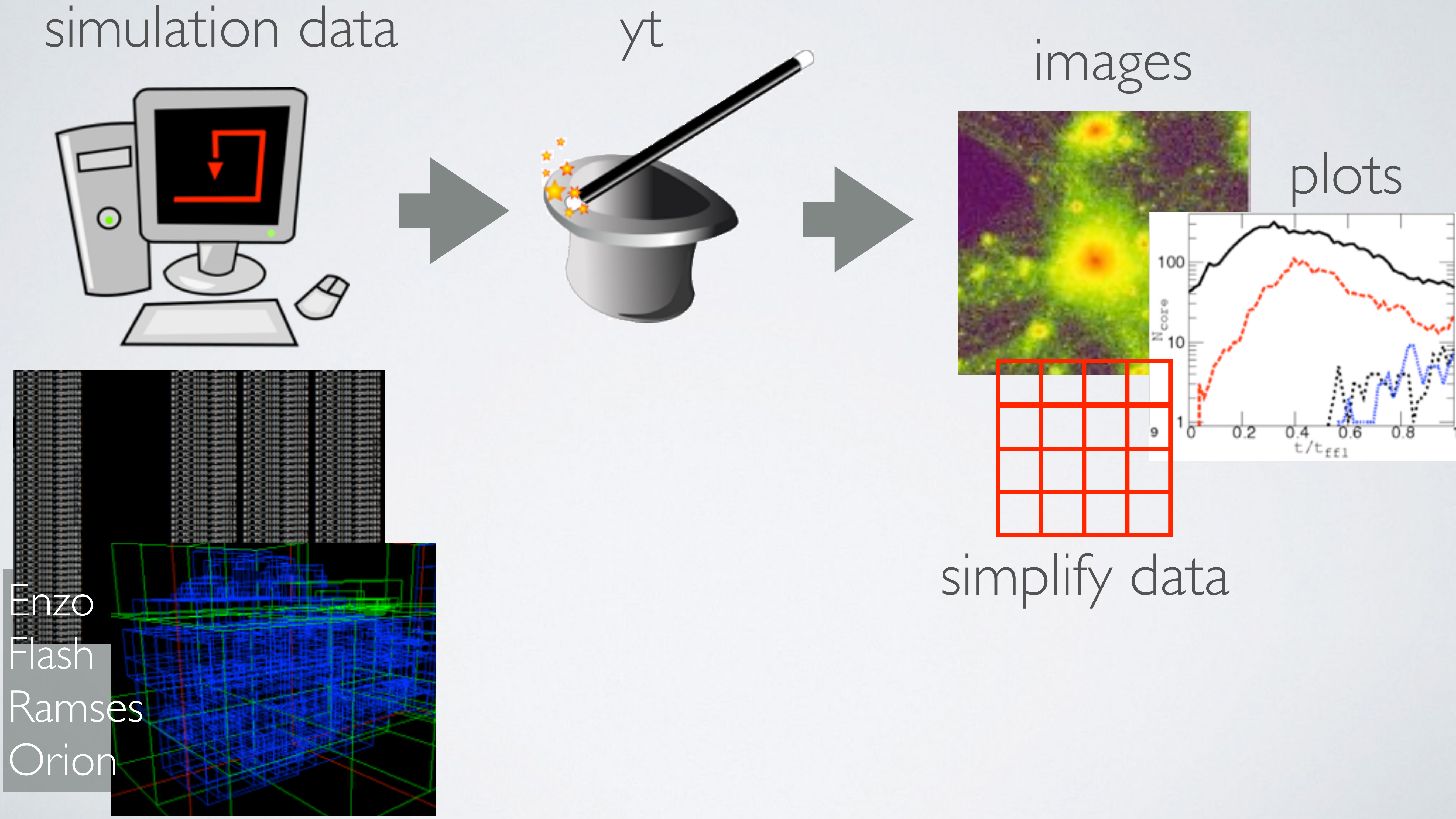
YT

An introduction

Contents

- What is **yt**?
- Installing **yt**
- Using **yt**
  - `> yt from the command line`
  - **yt** with the iPython notebook
  - scripting **yt**
- **yt**'s Cookbook

What is **yt**?



simulation data

yt

images

plots

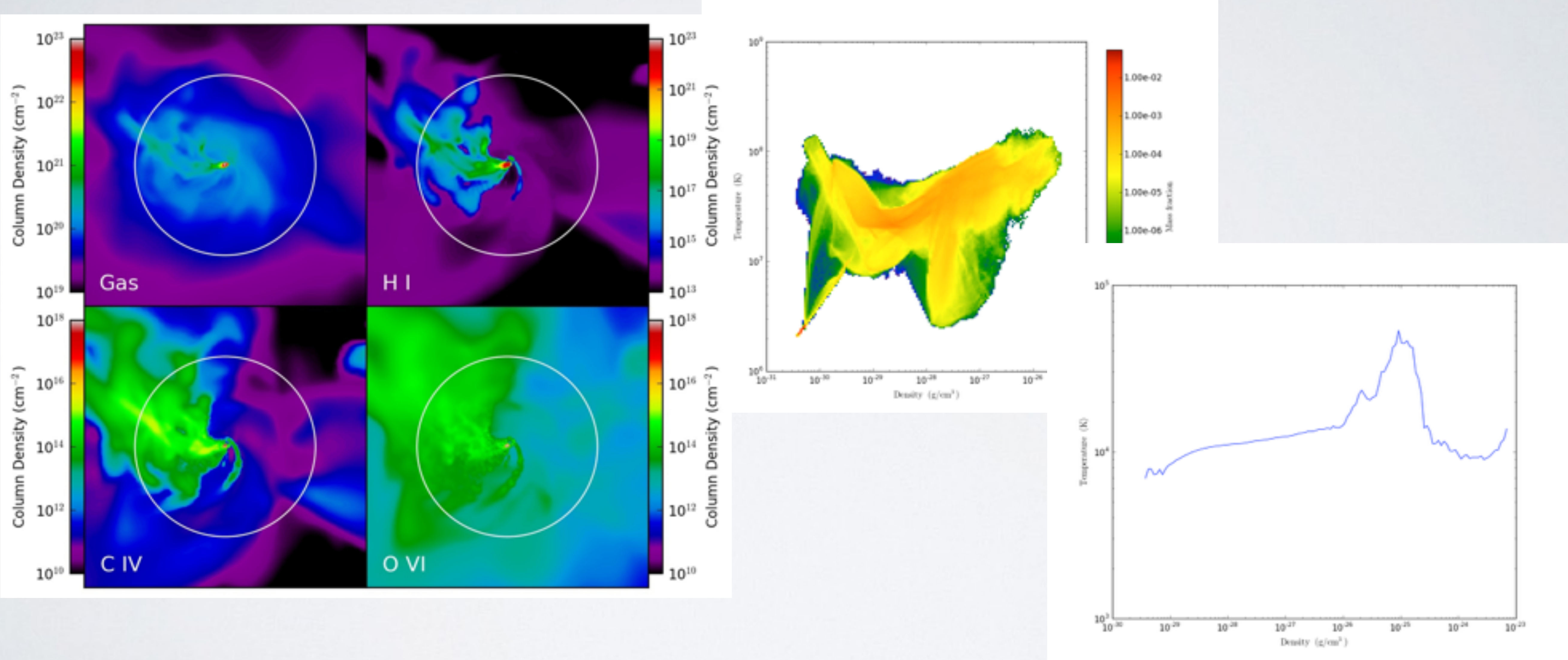
simplify data

Enzo
Flash
Ramses
Orion

What is **yt**?

Analysis basics:

(Plots you always need to create)



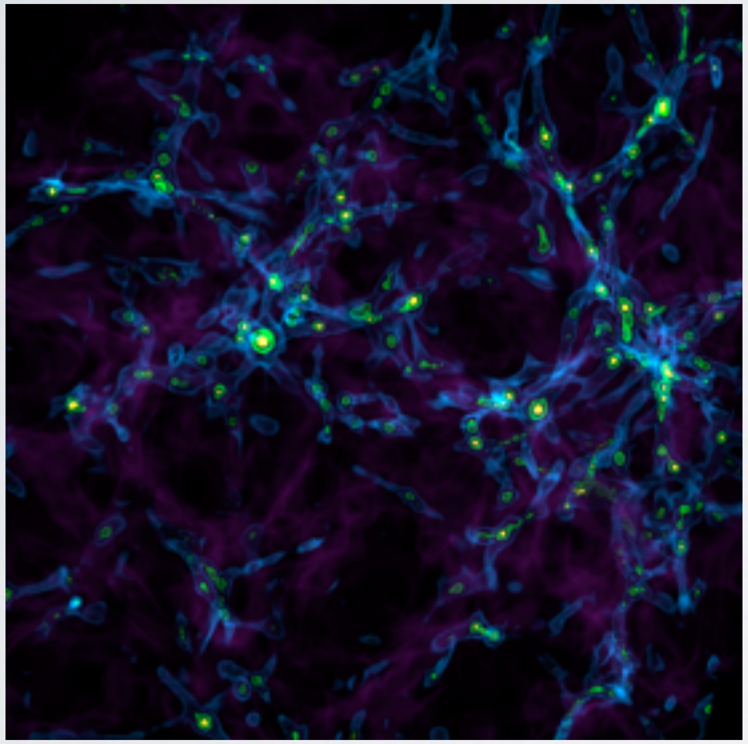
Very easy to make

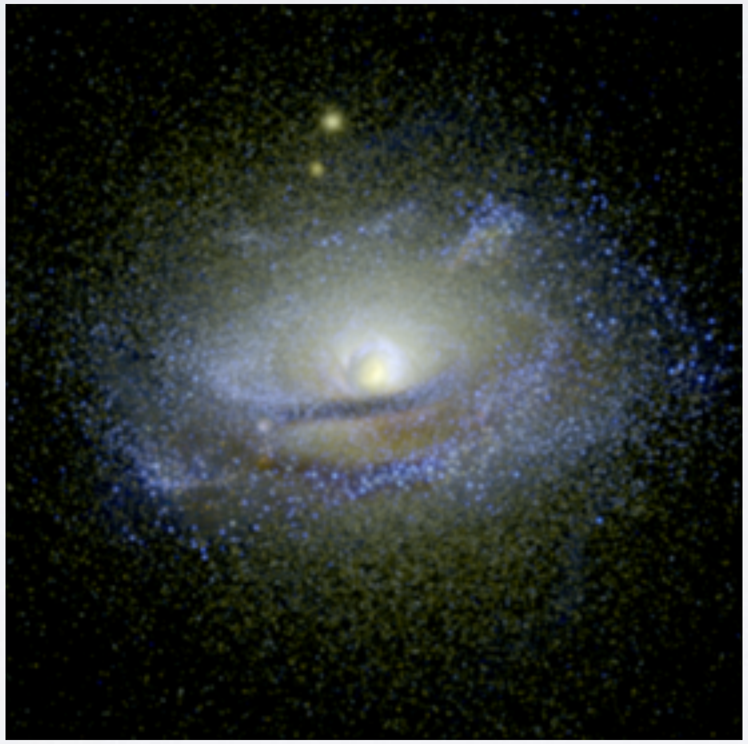
Slices, projections, 2D plots, 1D profiles....

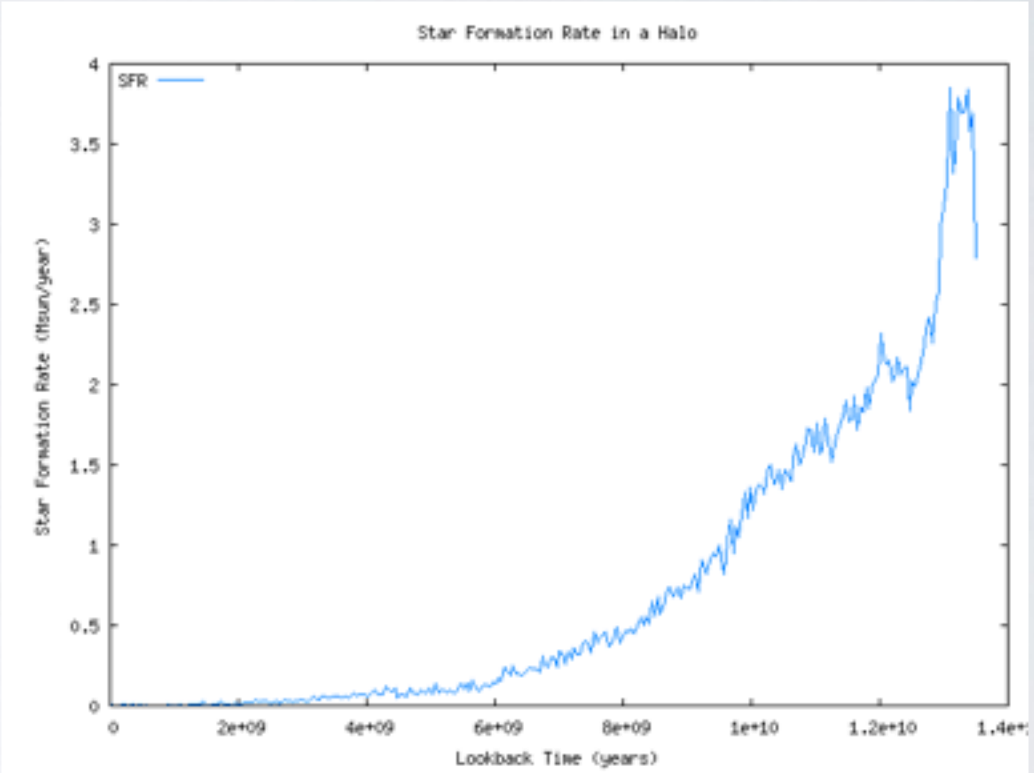
What is **yt**?

Advanced tools:

(Complicated analysis in easy-to-use routines)







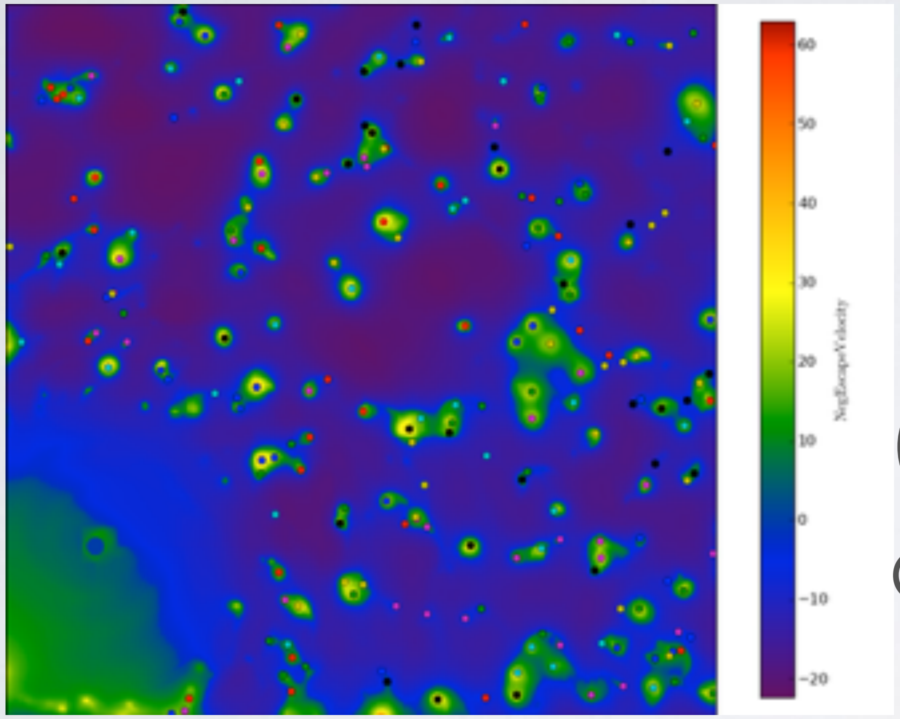
e.g. Dark matter halo finder and gas clump finder,

‘Synthetic observations’ with Sunrise (radiative transfer)

Calculate star formation rates in any region

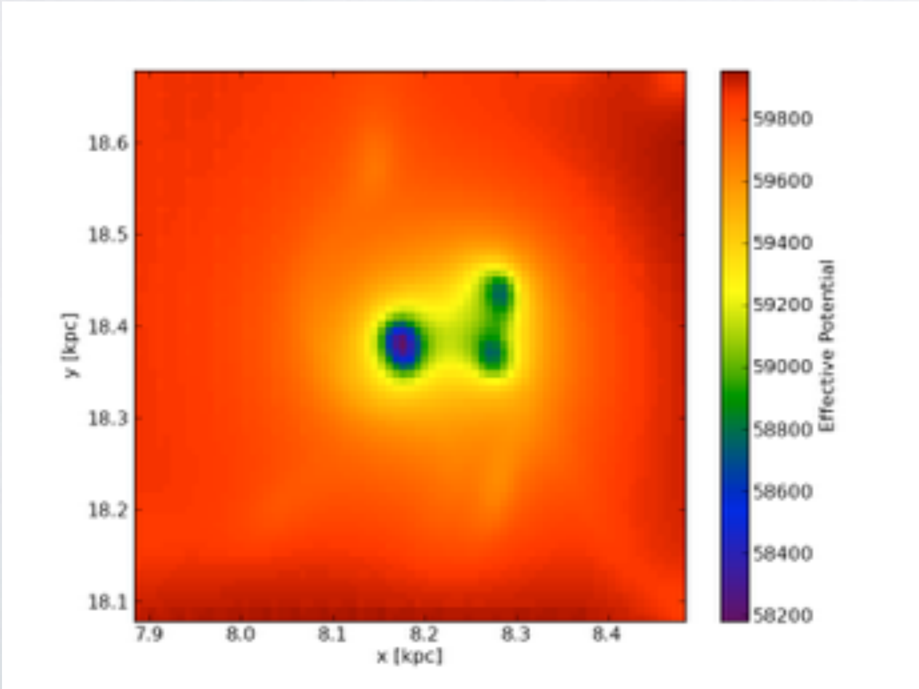
What is **yt**?

Use as part of your own analysis programmes

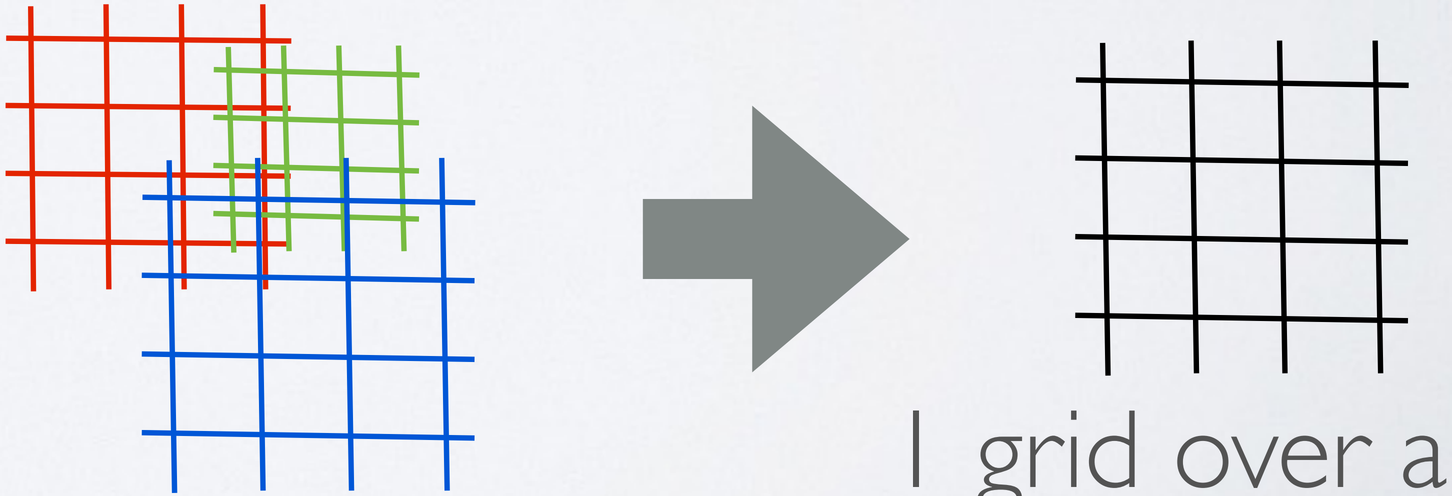


e.g. easily view new properties in images, plots etc

(escape velocity, density^2, mass x time, dinosaurs/cm^3 ...)



e.g. make data simple

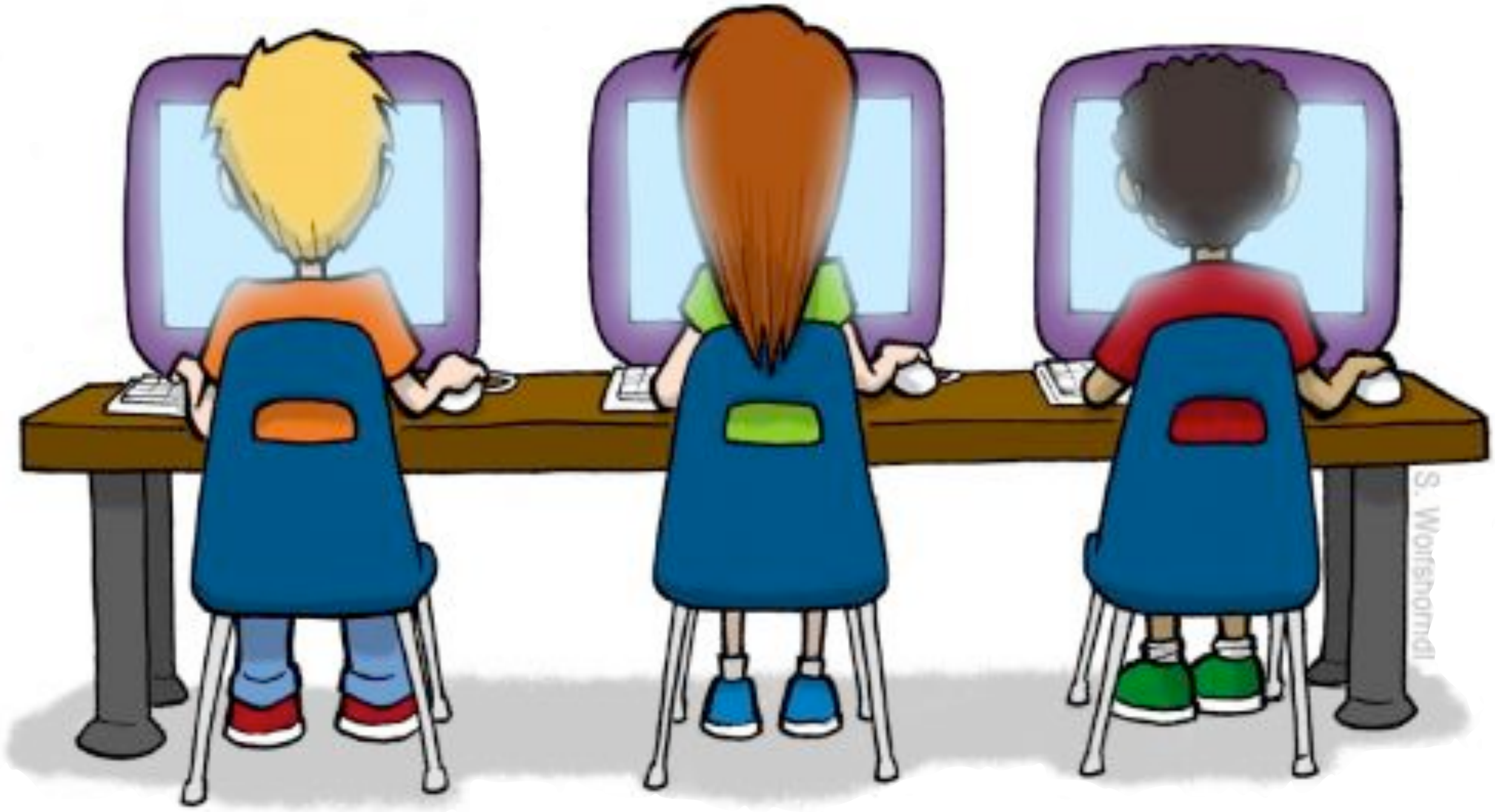


1000s of grids...

1 grid over any volume

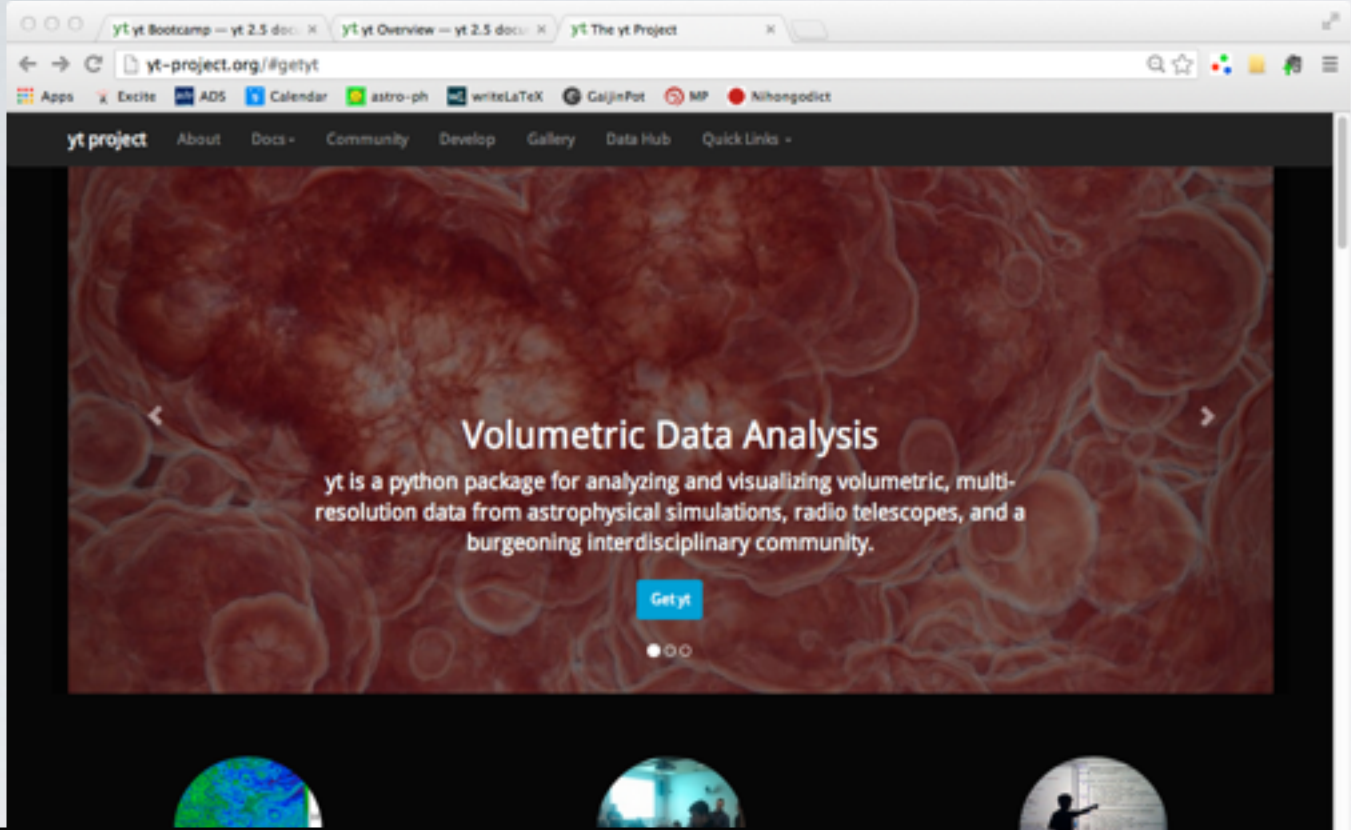
Installing **yt**

Let's do this together....



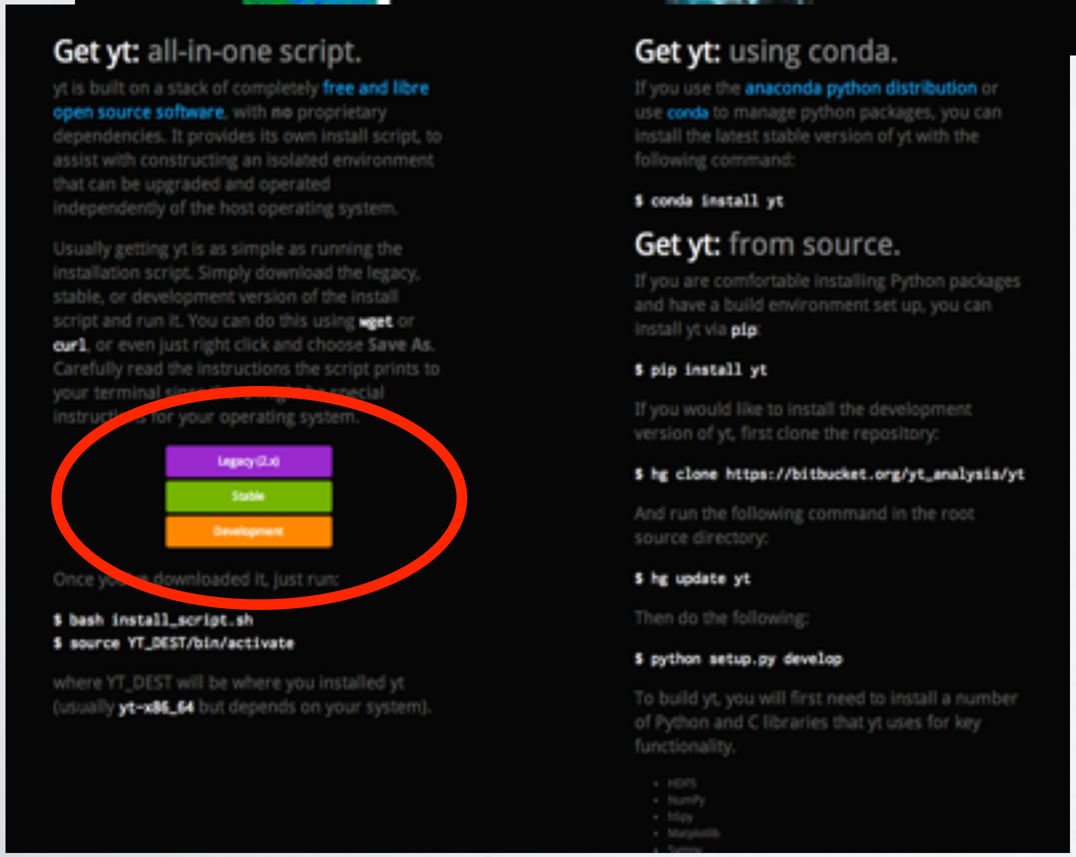
Can everyone connect to the WWW?

Installing **yt**



Volumetric Data Analysis

yt is a python package for analyzing and visualizing volumetric, multi-resolution data from astrophysical simulations, radio telescopes, and a burgeoning interdisciplinary community.



Get yt: all-in-one script.

yt is built on a stack of completely free and libre open source software, with no proprietary dependencies. It provides its own install script, to assist with constructing an isolated environment that can be upgraded and operated independently of the host operating system.

Usually getting yt is as simple as running the installation script. Simply download the legacy, stable, or development version of the install script and run it. You can do this using **wget** or **curl**, or even just right click and choose Save As. Carefully read the instructions the script prints to your terminal since there might be special instructions for your operating system.

Once you've downloaded it, just run:

```
$ bash install_script.sh
$ source YT_DEST/bin/activate
```

where YT_DEST will be where you installed yt (usually **yt-x86_64** but depends on your system).

Get yt: using conda.

If you use the anaconda python distribution or use conda to manage python packages, you can install the latest stable version of yt with the following command:

```
$ conda install yt
```

Get yt: from source.

If you are comfortable installing Python packages and have a build environment set up, you can install yt via **pip**:

```
$ pip install yt
```

If you would like to install the development version of yt, first clone the repository:

```
$ hg clone https://bitbucket.org/yt_analysis/yt
```

And run the following command in the root source directory:

```
$ hg update yt
```

Then do the following:

```
$ python setup.py develop
```

To build yt, you will first need to install a number of Python and C libraries that yt uses for key functionality.

- HDF5
- NumPy
- h5py
- Matplotlib
- Cython

yt webpage:

http://yt-project.org

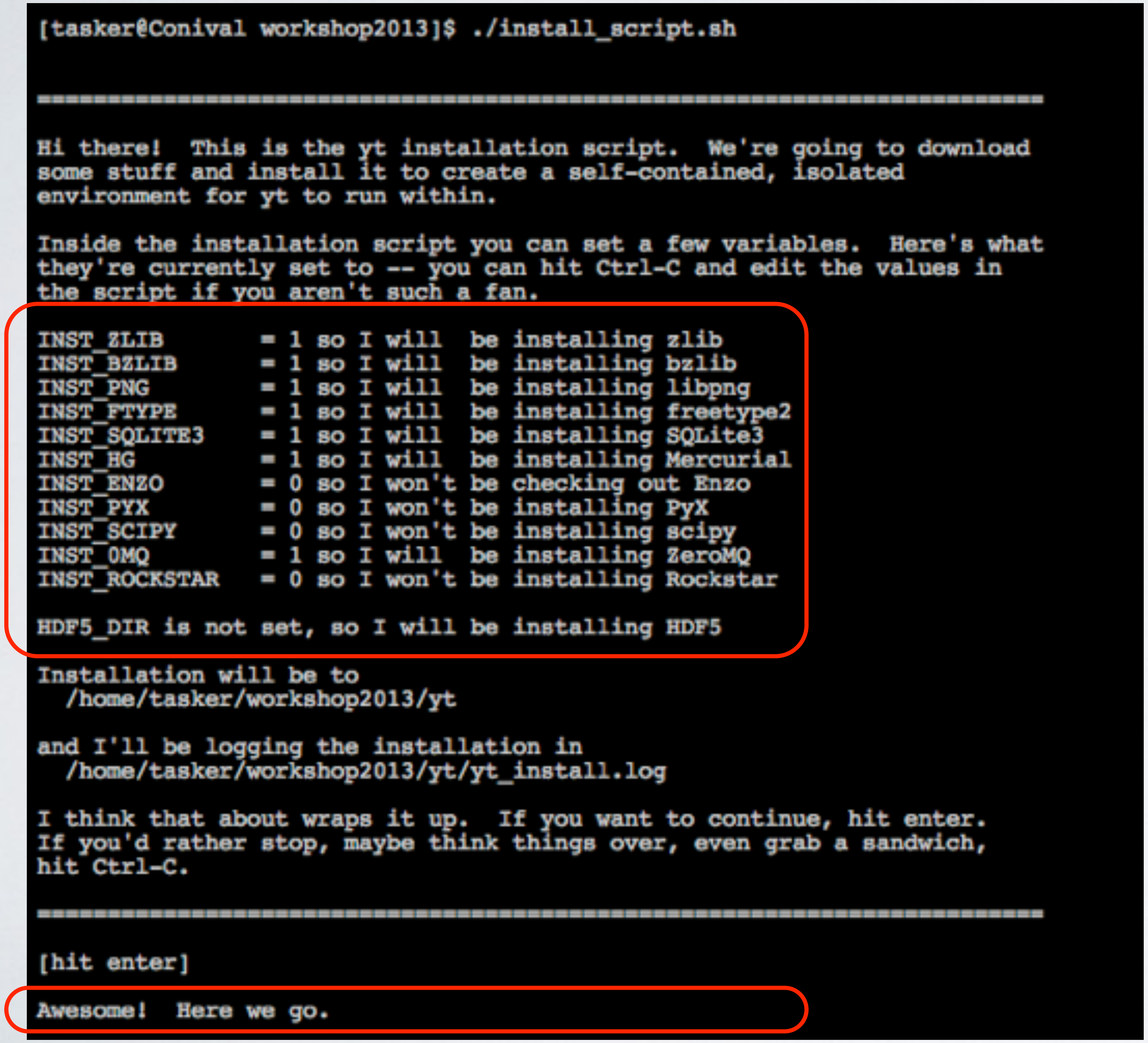
Download installation script

and run...

```
./install_script.sh
```

Installing **yt**

install_script.sh



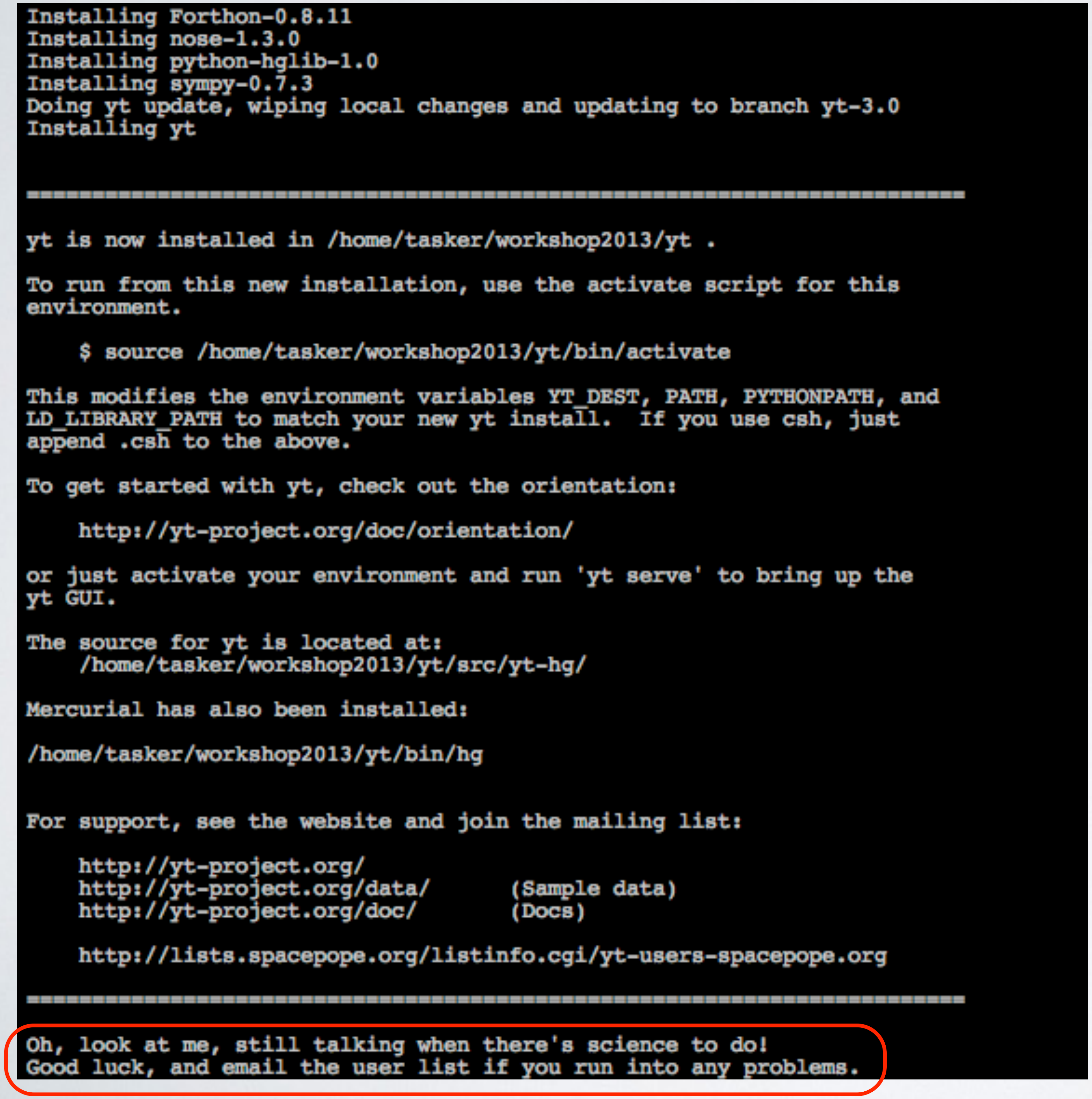
```
[tasker@Conival workshop2013]$ ./install_script.sh


========================================================================

Hi there!  This is the yt installation script.  We're going to download
some stuff and install it to create a self-contained, isolated
environment for yt to run within.

Inside the installation script you can set a few variables.  Here's what
they're currently set to -- you can hit Ctrl-C and edit the values in
the script if you aren't such a fan.

INST_ZLIB       = 1 so I will  be installing zlib
INST_BZLIB      = 1 so I will  be installing bzlib
INST_PNG        = 1 so I will  be installing libpng
INST_FTYPE      = 1 so I will  be installing freetype2
INST_SQLITE3    = 1 so I will  be installing SQLite3
INST_HG         = 1 so I will  be installing Mercurial
INST_ENZO       = 0 so I won't be checking out Enzo
INST_PYX        = 0 so I won't be installing PyX
INST_SCIPY      = 0 so I won't be installing scipy
INST_0MQ        = 1 so I will  be installing ZeroMQ
INST_ROCKSTAR   = 0 so I won't be installing Rockstar

HDF5_DIR is not set, so I will be installing HDF5

Installation will be to
  /home/tasker/workshop2013/yt

and I'll be logging the installation in
  /home/tasker/workshop2013/yt/yt_install.log

I think that about wraps it up.  If you want to continue, hit enter.
If you'd rather stop, maybe think things over, even grab a sandwich,
hit Ctrl-C.

========================================================================

[hit enter]

Awesome!  Here we go.
```

installs all necessary packages

very friendly!

Installing **yt**

install_script.sh

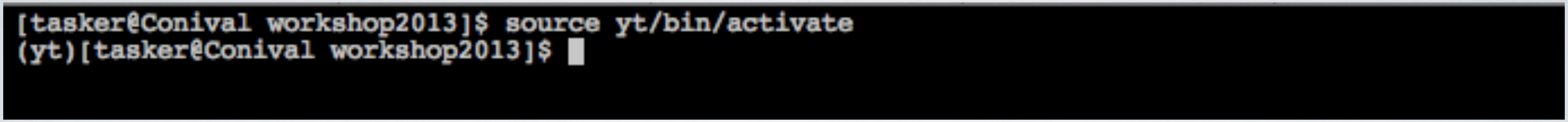


```
Installing Forthon-0.8.11
Installing nose-1.3.0
Installing python-hglib-1.0
Installing sympy-0.7.3
Doing yt update, wiping local changes and updating to branch yt-3.0
Installing yt


========================================================================

yt is now installed in /home/tasker/workshop2013/yt .

To run from this new installation, use the activate script for this
environment.

    $ source /home/tasker/workshop2013/yt/bin/activate

This modifies the environment variables YT_DEST, PATH, PYTHONPATH, and
LD_LIBRARY_PATH to match your new yt install.  If you use csh, just
append .csh to the above.

To get started with yt, check out the orientation:

    http://yt-project.org/doc/orientation/

or just activate your environment and run 'yt serve' to bring up the
yt GUI.

The source for yt is located at:
    /home/tasker/workshop2013/yt/src/yt-hg/

Mercurial has also been installed:

/home/tasker/workshop2013/yt/bin/hg


For support, see the website and join the mailing list:

    http://yt-project.org/
    http://yt-project.org/data/      (Sample data)
    http://yt-project.org/doc/       (Docs)

    http://lists.spacepope.org/listinfo.cgi/yt-users-spacepope.org

========================================================================

Oh, look at me, still talking when there's science to do!
Good luck, and email the user list if you run into any problems.
```

time

Finished!

Installing **yt**



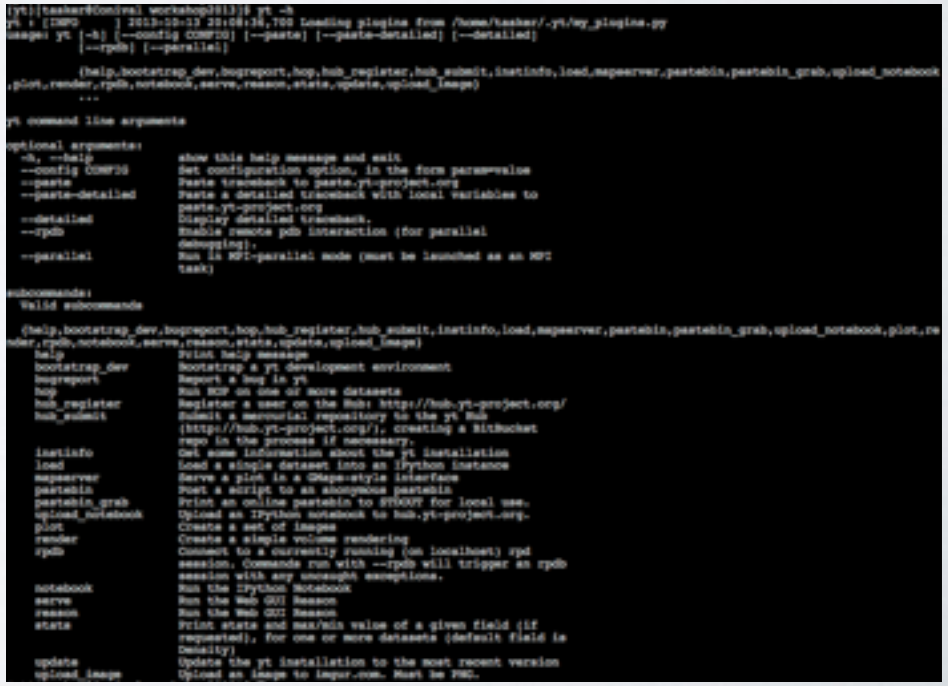
```
[tasker@Conival workshop2013]$ source yt/bin/activate
(yt)[tasker@Conival workshop2013]$ █
```

```
> source yt-x86_64/bin/activate
```

path to yt

```
> yt -h
```

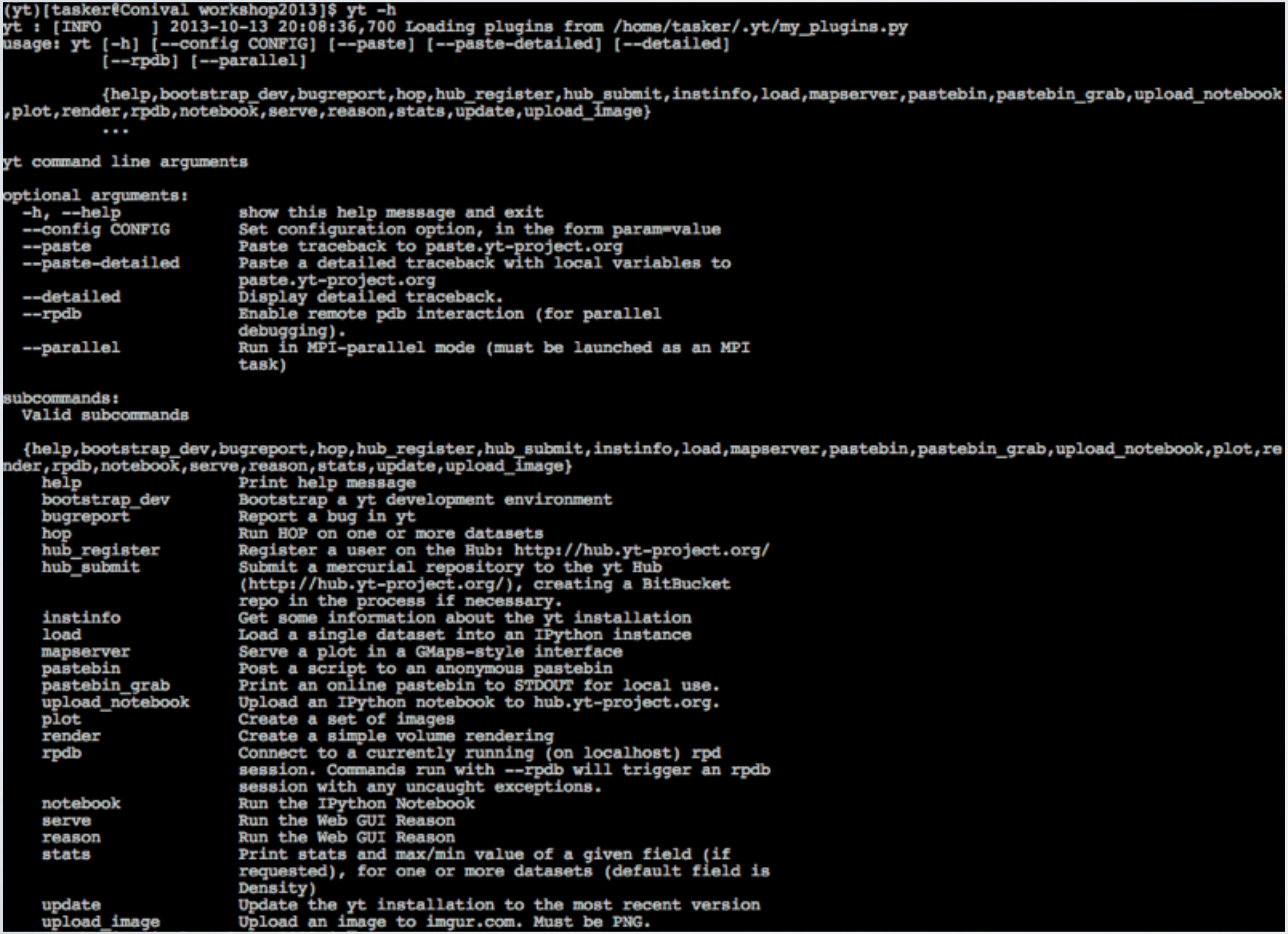
yt command line options



```
(yt)[tasker@Conival workshop2013]$ yt -h
yt : [INFO     ] 2013-10-13 20:08:36,700 Loading plugins from /home/tasker/.yt/my_plugins.py
usage: yt [-h] [--config CONFIG] [--paste] [--paste-detailed] [--detailed]
          [--rpdb] [--parallel]

          {help,bootstrap_dev,bugreport,hop,hub_register,hub_submit,instinfo,load,mapserver,pastebin,pastebin_grab,upload_notebook
,plot,render,rpdb,notebook,serve,reason,stats,update,upload_image}
          ...

yt command line arguments

optional arguments:
  -h, --help            show this help message and exit
  --config CONFIG       Set configuration option, in the form param=value
  --paste               Paste traceback to paste.yt-project.org
  --paste-detailed      Paste a detailed traceback with local variables to
                        paste.yt-project.org
  --detailed            Display detailed traceback.
  --rpdb                Enable remote pdb interaction (for parallel
                        debugging).
  --parallel            Run in MPI-parallel mode (must be launched as an MPI
                        task)

subcommands:
  Valid subcommands

  {help,bootstrap_dev,bugreport,hop,hub_register,hub_submit,instinfo,load,mapserver,pastebin,pastebin_grab,upload_notebook,plot,re
nder,rpdb,notebook,serve,reason,stats,update,upload_image}
    help                Print help message
    bootstrap_dev       Bootstrap a yt development environment
    bugreport           Report a bug in yt
    hop                 Run HOP on one or more datasets
    hub_register        Register a user on the Hub: http://hub.yt-project.org/
    hub_submit          Submit a mercurial repository to the yt Hub
                        (https://hub.yt-project.org/), creating a BitBucket
                        repo in the process if necessary.
    instinfo            Get some information about the yt installation
    load                Load a single dataset into an IPython instance
    mapserver           Serve a plot in a GMaps-style interface
    pastebin            Post a script to an anonymous pastebin
    pastebin_grab       Print an online pastebin to STDOUT for local use.
    upload_notebook     Upload an IPython notebook to hub.yt-project.org.
    plot                Create a set of images
    render              Create a simple volume rendering
    rpdb                Connect to a currently running (on localhost) rpd
                        session. Commands run with --rpdb will trigger an rpdb
                        session with any uncaught exceptions.
    notebook            Run the IPython Notebook
    serve               Run the Web GUI Reason
    reason              Run the Web GUI Reason
    stats               Print stats and max/min value of a given field (if
                        requested), for one or more datasets (default field is
                        Density)
    update              Update the yt installation to the most recent version
    upload_image        Upload an image to imgur.com. Must be PNG.
```

Command line **yt**

Quickest way to use **yt**

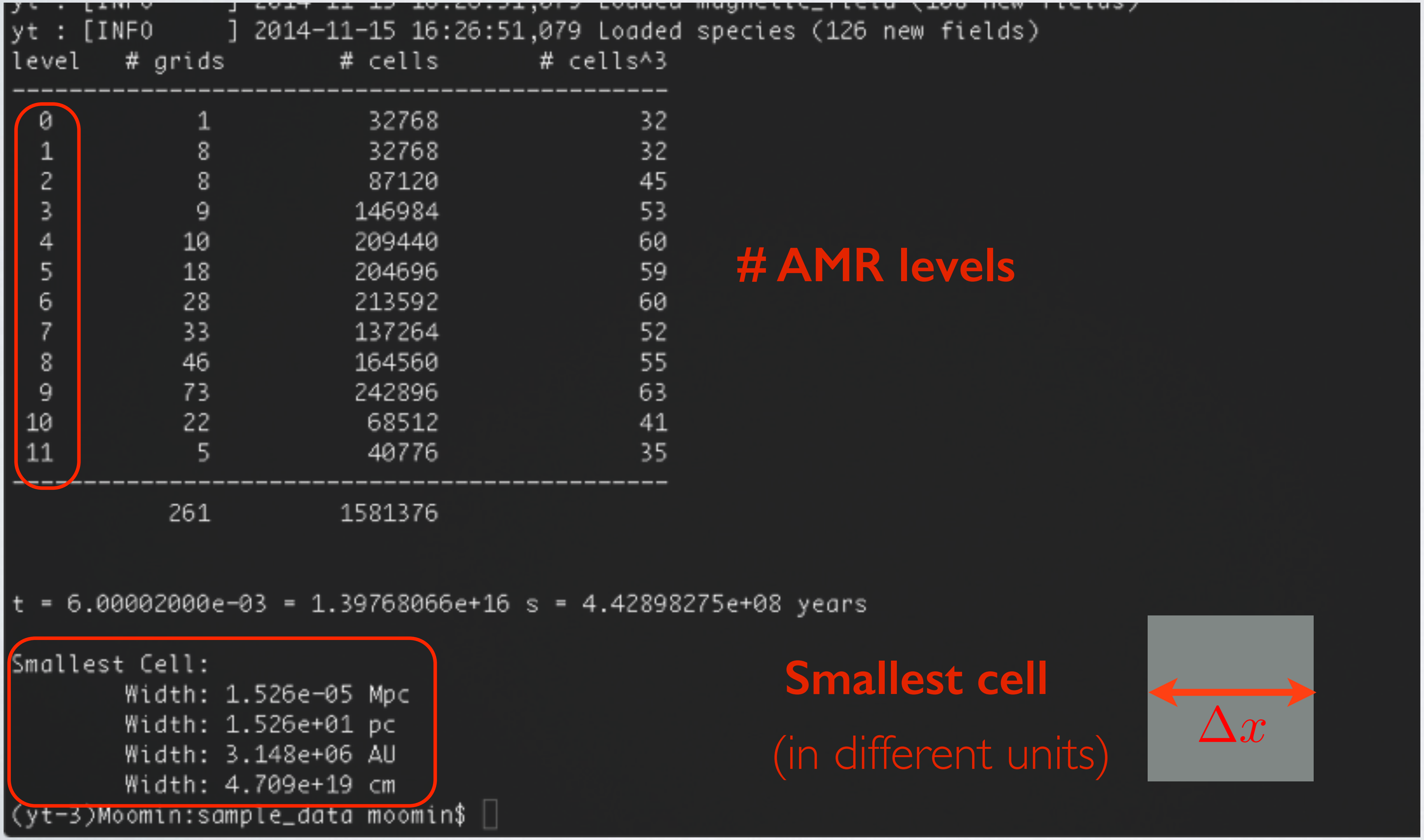


```
(yt)[tasker@Conival workshop2013]$ yt -h
yt : [INFO     ] 2013-10-13 20:08:36,700 Loading plugins from /home/tasker/.yt/my_plugins.py
usage: yt [-h] [--config CONFIG] [--paste] [--paste-detailed] [--detailed]
          [--rpdb] [--parallel]

          {help,bootstrap_dev,bugreport,hop,hub_register,hub_submit,instinfo,load,mapserver,pastebin,pastebin_grab,upload_notebook
,plot,render,rpdb,notebook,serve,reason,stats,update,upload_image}
          ...

yt command line arguments

optional arguments:
  -h, --help            show this help message and exit
  --config CONFIG       Set configuration option, in the form param=value
  --paste               Paste traceback to paste.yt-project.org
  --paste-detailed      Paste a detailed traceback with local variables to
                        paste.yt-project.org
  --detailed            Display detailed traceback.
  --rpdb                Enable remote pdb interaction (for parallel
                        debugging).
  --parallel            Run in MPI-parallel mode (must be launched as an MPI
                        task)

subcommands:
  Valid subcommands

  {help,bootstrap_dev,bugreport,hop,hub_register,hub_submit,instinfo,load,mapserver,pastebin,pastebin_grab,upload_notebook,plot,re
nder,rpdb,notebook,serve,reason,stats,update,upload_image}
    help                Print help message
    bootstrap_dev       Bootstrap a yt development environment
    bugreport           Report a bug in yt
    hop                 Run HOP on one or more datasets
    hub_register        Register a user on the Hub: http://hub.yt-project.org/
    hub_submit          Submit a mercurial repository to the yt Hub
                        (http://hub.yt-project.org/), creating a BitBucket
                        repo in the process if necessary.
    instinfo            Get some information about the yt installation
    load                Load a single dataset into an IPython instance
    mapserver           Serve a plot in a GMaps-style interface
    pastebin            Post a script to an anonymous pastebin
    pastebin_grab       Print an online pastebin to STDOUT for local use.
    upload_notebook     Upload an IPython notebook to hub.yt-project.org.
    plot                Create a set of images
    render              Create a simple volume rendering
    rpdb                Connect to a currently running (on localhost) rpd
                        session. Commands run with --rpdb will trigger an rpdb
                        session with any uncaught exceptions.
    notebook            Run the IPython Notebook
    serve               Run the Web GUI Reason
    reason              Run the Web GUI Reason
    stats               Print stats and max/min value of a given field (if
                        requested), for one or more datasets (default field is
                        Density)
    update              Update the yt installation to the most recent version
    upload_image        Upload an image to imgur.com. Must be PNG.
```

Command line **yt**

```
(yt)> cd workshop2014

(yt)> yt stats sample_data/IsolatedGravity/galaxy0030/galaxy0030
```

Enzo data output



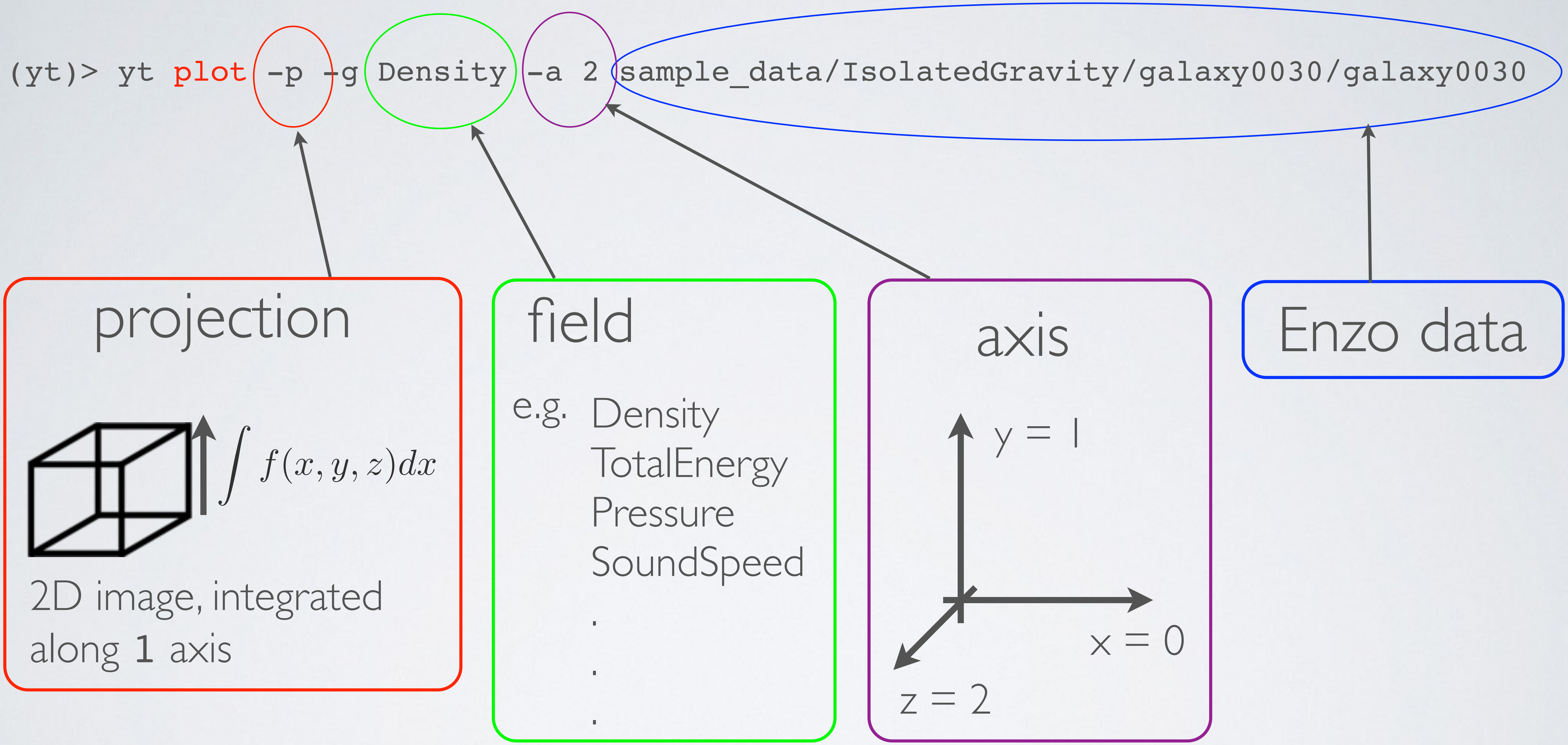
```
yt : [INFO     ] 2014-11-15 16:26:51,079 Loaded magnetic_field (108 new fields)
yt : [INFO     ] 2014-11-15 16:26:51,079 Loaded species (126 new fields)
level   # grids         # cells        # cells^3
----------------------------------------------
  0          1           32768              32
  1          8           32768              32
  2          8           87120              45
  3          9          146984              53
  4         10          209440              60
  5         18          204696              59
  6         28          213592              60
  7         33          137264              52
  8         46          164560              55
  9         73          242896              63
 10         22           68512              41
 11          5           40776              35
----------------------------------------------
           261         1581376


t = 6.00002000e-03 = 1.39768066e+16 s = 4.42898275e+08 years

Smallest Cell:
        Width: 1.526e-05 Mpc
        Width: 1.526e+01 pc
        Width: 3.148e+06 AU
        Width: 4.709e+19 cm
(yt-3)Moomin:sample_data moomin$
```

# AMR levels

Smallest cell

(in different units)

Command line **yt**



```
(yt)> yt plot -p -g Density -a 2 sample_data/IsolatedGravity/galaxy0030/galaxy0030
```

projection

$$\int f(x,y,z)dx$$

2D image, integrated along 1 axis

field

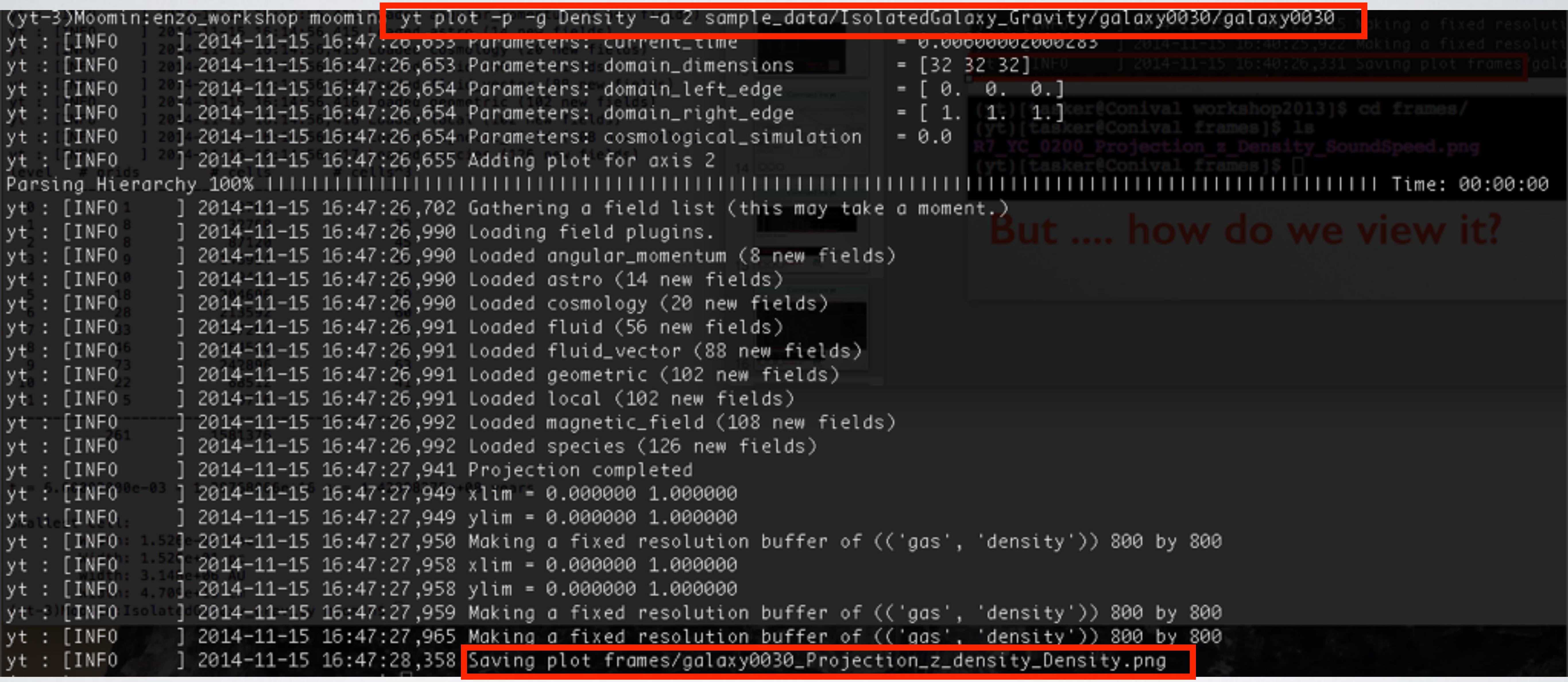
e.g. Density
TotalEnergy
Pressure
SoundSpeed
.
.
.

axis

Enzo data

Command line **yt**

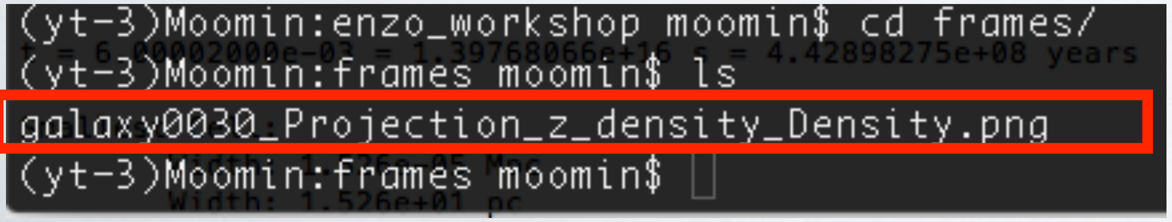
```
(yt)> yt plot -p -g Density -a 2 sample_data/IsolatedGravity/galaxy0030/galaxy0030
```



```
(yt-3)Moomin:enzo_workshop moomin$ yt plot -p -g Density -a 2 sample_data/IsolatedGalaxy_Gravity/galaxy0030/galaxy0030
yt : [INFO     ] 2014-11-15 16:47:26,653 Parameters: current_time              = 0.00600002000283
yt : [INFO     ] 2014-11-15 16:47:26,653 Parameters: domain_dimensions         = [32 32 32]
yt : [INFO     ] 2014-11-15 16:47:26,654 Parameters: domain_left_edge          = [ 0.  0.  0.]
yt : [INFO     ] 2014-11-15 16:47:26,654 Parameters: domain_right_edge         = [ 1.  1.  1.]
yt : [INFO     ] 2014-11-15 16:47:26,654 Parameters: cosmological_simulation   = 0.0
yt : [INFO     ] 2014-11-15 16:47:26,655 Adding plot for axis 2
Parsing Hierarchy 100% |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| Time: 00:00:00
yt : [INFO     ] 2014-11-15 16:47:26,702 Gathering a field list (this may take a moment.)
yt : [INFO     ] 2014-11-15 16:47:26,990 Loading field plugins.
yt : [INFO     ] 2014-11-15 16:47:26,990 Loaded angular_momentum (8 new fields)
yt : [INFO     ] 2014-11-15 16:47:26,990 Loaded astro (14 new fields)
yt : [INFO     ] 2014-11-15 16:47:26,990 Loaded cosmology (20 new fields)
yt : [INFO     ] 2014-11-15 16:47:26,991 Loaded fluid (56 new fields)
yt : [INFO     ] 2014-11-15 16:47:26,991 Loaded fluid_vector (88 new fields)
yt : [INFO     ] 2014-11-15 16:47:26,991 Loaded geometric (102 new fields)
yt : [INFO     ] 2014-11-15 16:47:26,991 Loaded local (102 new fields)
yt : [INFO     ] 2014-11-15 16:47:26,992 Loaded magnetic_field (108 new fields)
yt : [INFO     ] 2014-11-15 16:47:26,992 Loaded species (126 new fields)
yt : [INFO     ] 2014-11-15 16:47:27,941 Projection completed
yt : [INFO     ] 2014-11-15 16:47:27,949 xlim = 0.000000 1.000000
yt : [INFO     ] 2014-11-15 16:47:27,949 ylim = 0.000000 1.000000
yt : [INFO     ] 2014-11-15 16:47:27,950 Making a fixed resolution buffer of (('gas', 'density')) 800 by 800
yt : [INFO     ] 2014-11-15 16:47:27,958 xlim = 0.000000 1.000000
yt : [INFO     ] 2014-11-15 16:47:27,958 ylim = 0.000000 1.000000
yt : [INFO     ] 2014-11-15 16:47:27,959 Making a fixed resolution buffer of (('gas', 'density')) 800 by 800
yt : [INFO     ] 2014-11-15 16:47:27,965 Making a fixed resolution buffer of (('gas', 'density')) 800 by 800
yt : [INFO     ] 2014-11-15 16:47:28,358 Saving plot frames/galaxy0030_Projection_z_density_Density.png
```

image!

```
(yt)>cd frames/
```

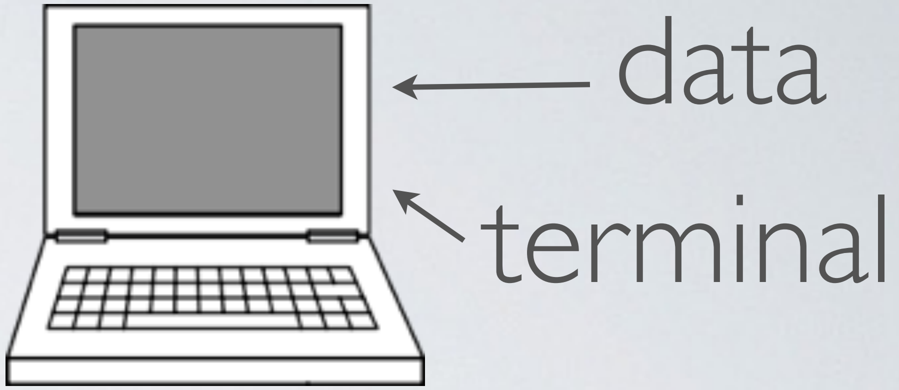


```
(yt-3)Moomin:enzo_workshop moomin$ cd frames/
(yt-3)Moomin:frames moomin$ ls
galaxy0030_Projection_z_density_Density.png
(yt-3)Moomin:frames moomin$
```

But .... how do we view it?

Command line **yt**

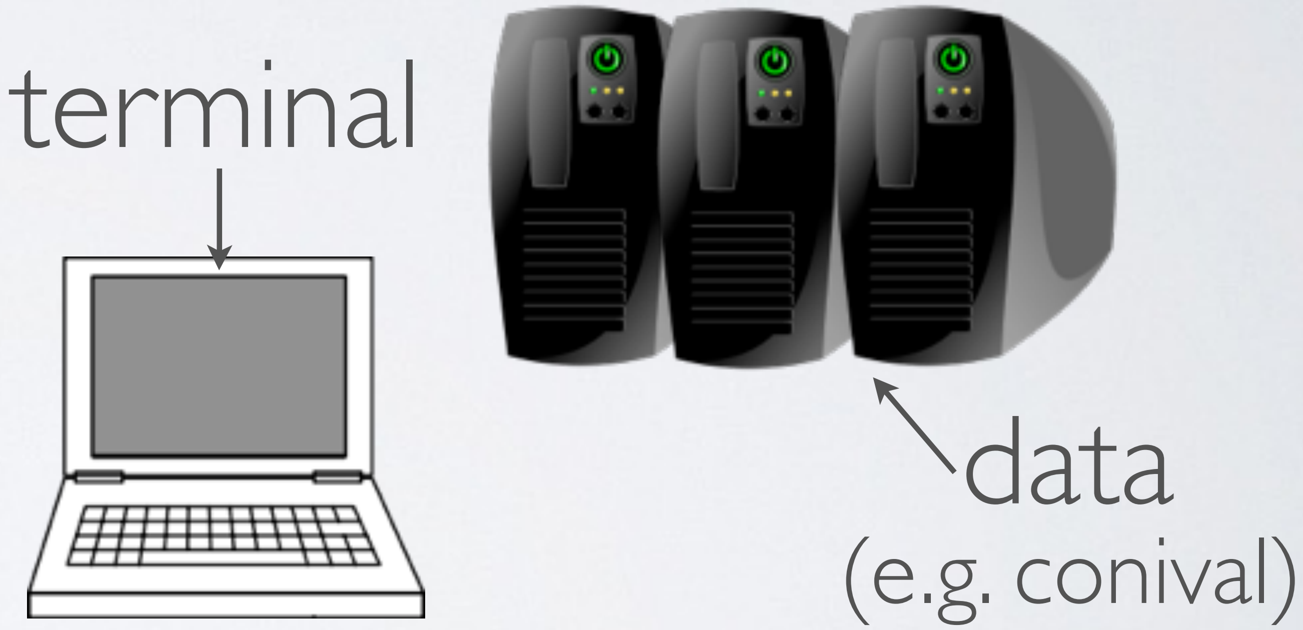
If data is local, viewing the image is easy!



e.g.

```
(yt)>display galaxy0030_Projection_z_density_Density.png
```

If data is not local....



Can use scp ....

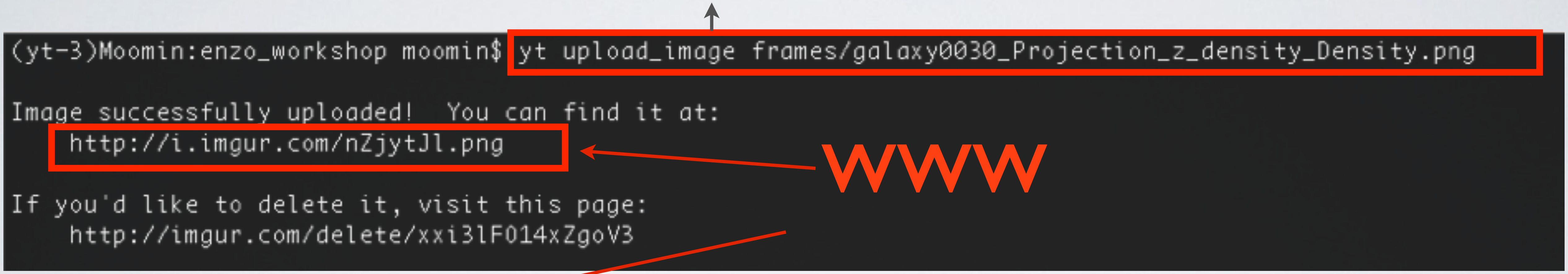
e.g.

```
(yt)>scp tasker@conival:workshop2014/frames/
galaxy0030_Projection_z_density_Density.png
```

But this can be slow

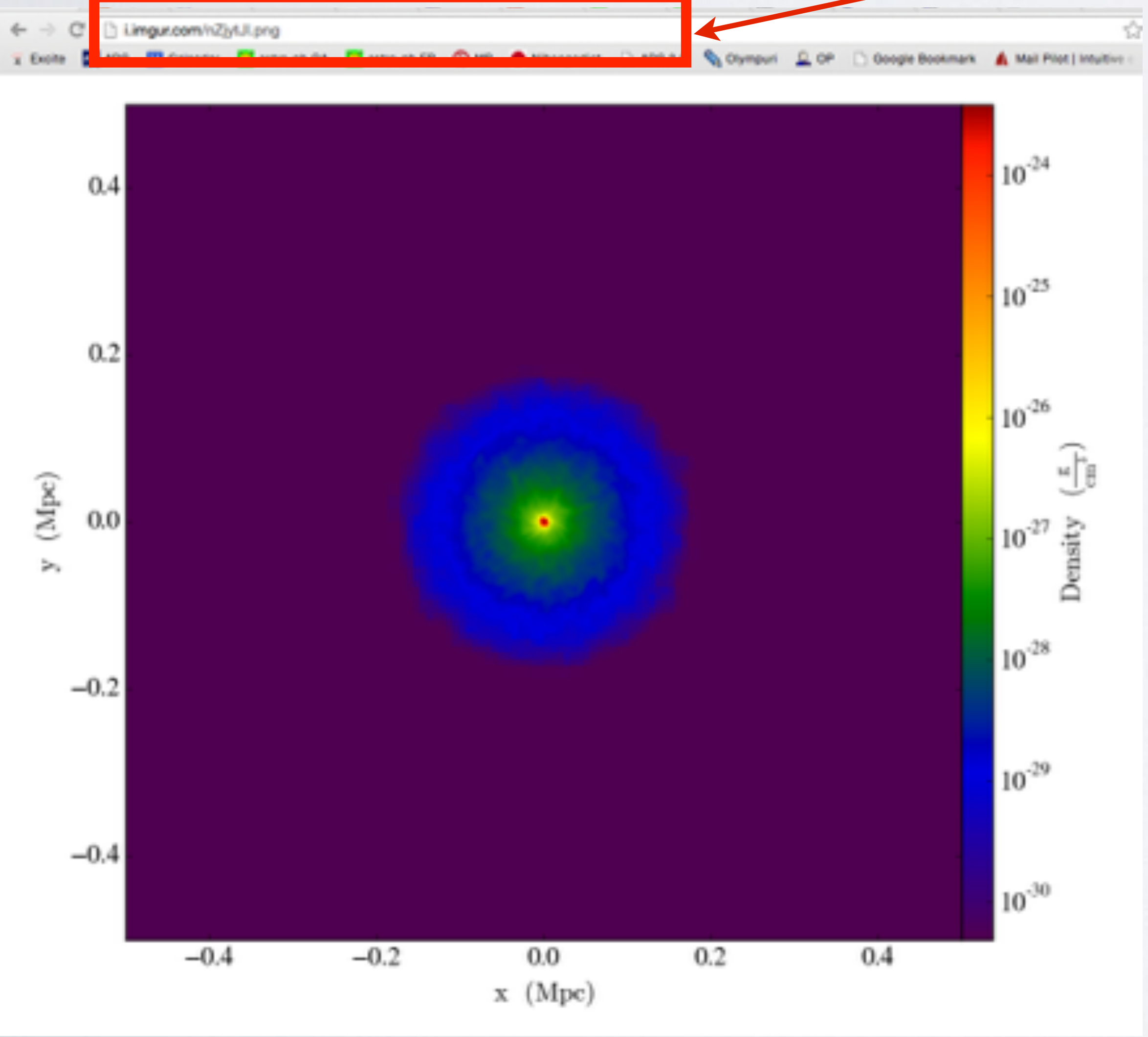
Command line **yt**

```
(yt)> yt upload_image frames/
galaxy0030_Projection_z_density_Density.png
```



```
(yt-3)Moomin:enzo_workshop moomin$ yt upload_image frames/galaxy0030_Projection_z_density_Density.png

Image successfully uploaded!  You can find it at:
    http://i.imgur.com/nZjytJl.png

If you'd like to delete it, visit this page:
    http://imgur.com/delete/xxi3lF014xZgoV3
```

www



Upload to imgur.com

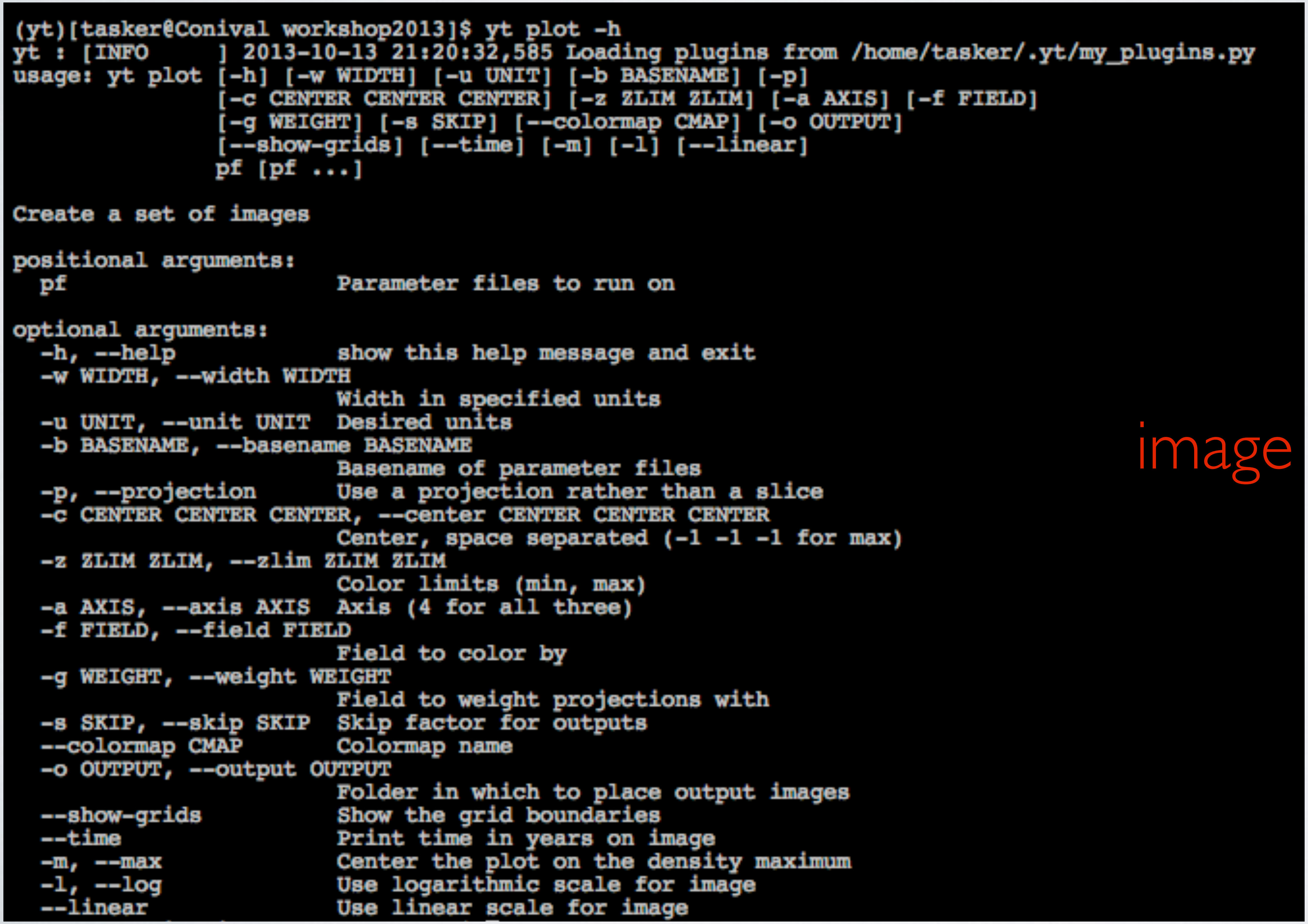
Easy to view,

easy to share

Command line **yt**

Image changes

```
(yt)> yt plot -h
```



```
(yt)[tasker@Conival workshop2013]$ yt plot -h
yt : [INFO     ] 2013-10-13 21:20:32,585 Loading plugins from /home/tasker/.yt/my_plugins.py
usage: yt plot [-h] [-w WIDTH] [-u UNIT] [-b BASENAME] [-p]
               [-c CENTER CENTER CENTER] [-z ZLIM ZLIM] [-a AXIS] [-f FIELD]
               [-g WEIGHT] [-s SKIP] [--colormap CMAP] [-o OUTPUT]
               [--show-grids] [--time] [-m] [-l] [--linear]
               pf [pf ...]

Create a set of images

positional arguments:
  pf                    Parameter files to run on

optional arguments:
  -h, --help            show this help message and exit
  -w WIDTH, --width WIDTH
                        Width in specified units
  -u UNIT, --unit UNIT  Desired units
  -b BASENAME, --basename BASENAME
                        Basename of parameter files
  -p, --projection      Use a projection rather than a slice
  -c CENTER CENTER CENTER, --center CENTER CENTER CENTER
                        Center, space separated (-1 -1 -1 for max)
  -z ZLIM ZLIM, --zlim ZLIM ZLIM
                        Color limits (min, max)
  -a AXIS, --axis AXIS  Axis (4 for all three)
  -f FIELD, --field FIELD
                        Field to color by
  -g WEIGHT, --weight WEIGHT
                        Field to weight projections with
  -s SKIP, --skip SKIP  Skip factor for outputs
  --colormap CMAP       Colormap name
  -o OUTPUT, --output OUTPUT
                        Folder in which to place output images
  --show-grids          Show the grid boundaries
  --time                Print time in years on image
  -m, --max             Center the plot on the density maximum
  -l, --log             Use logarithmic scale for image
  --linear              Use linear scale for image
```

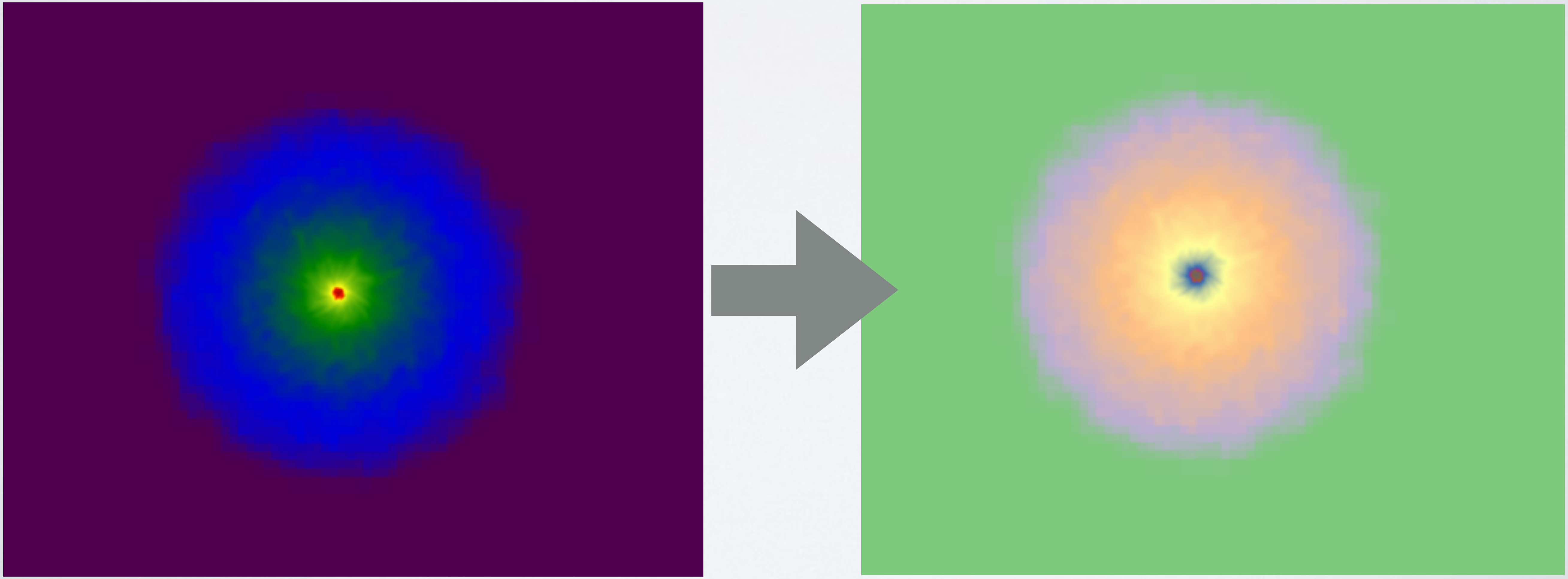
image options

Command line **yt**

Image changes

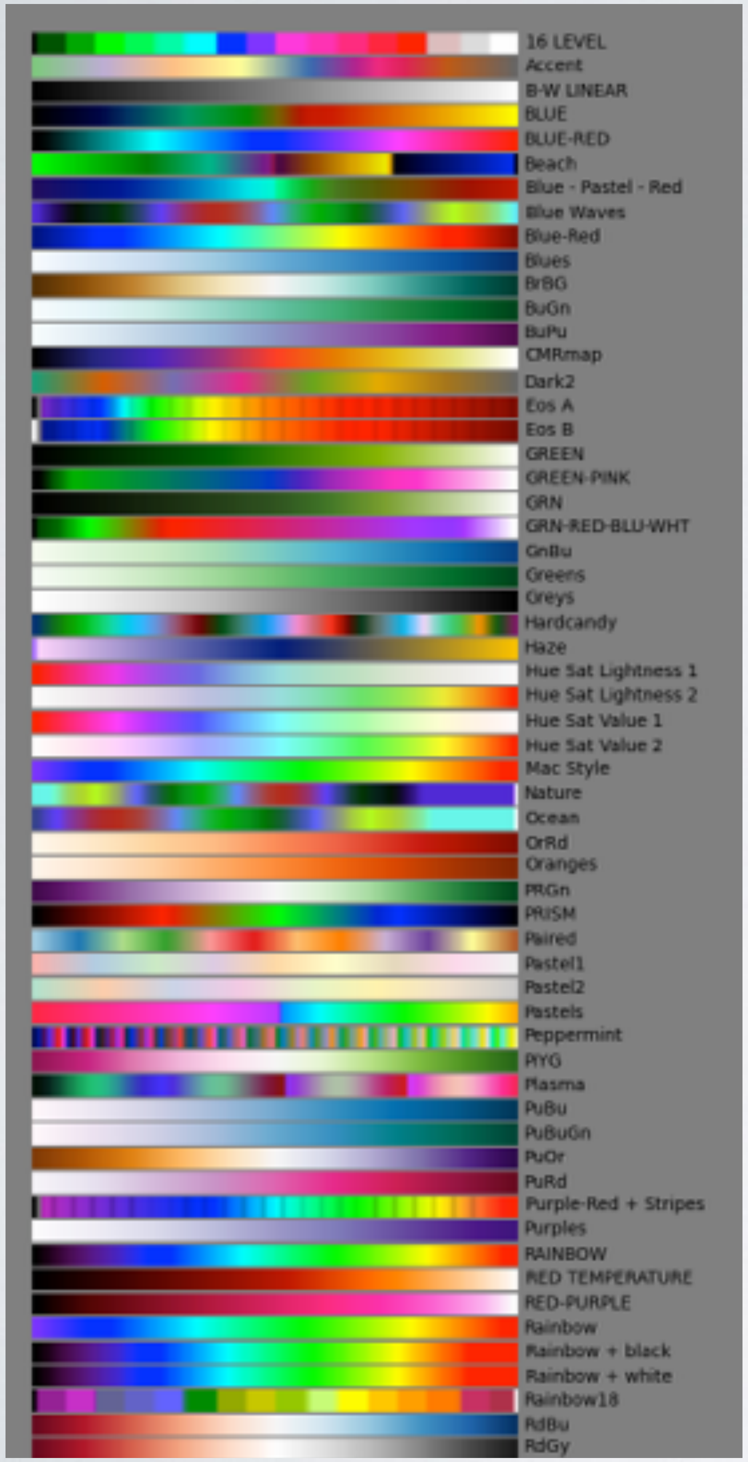
e.g.

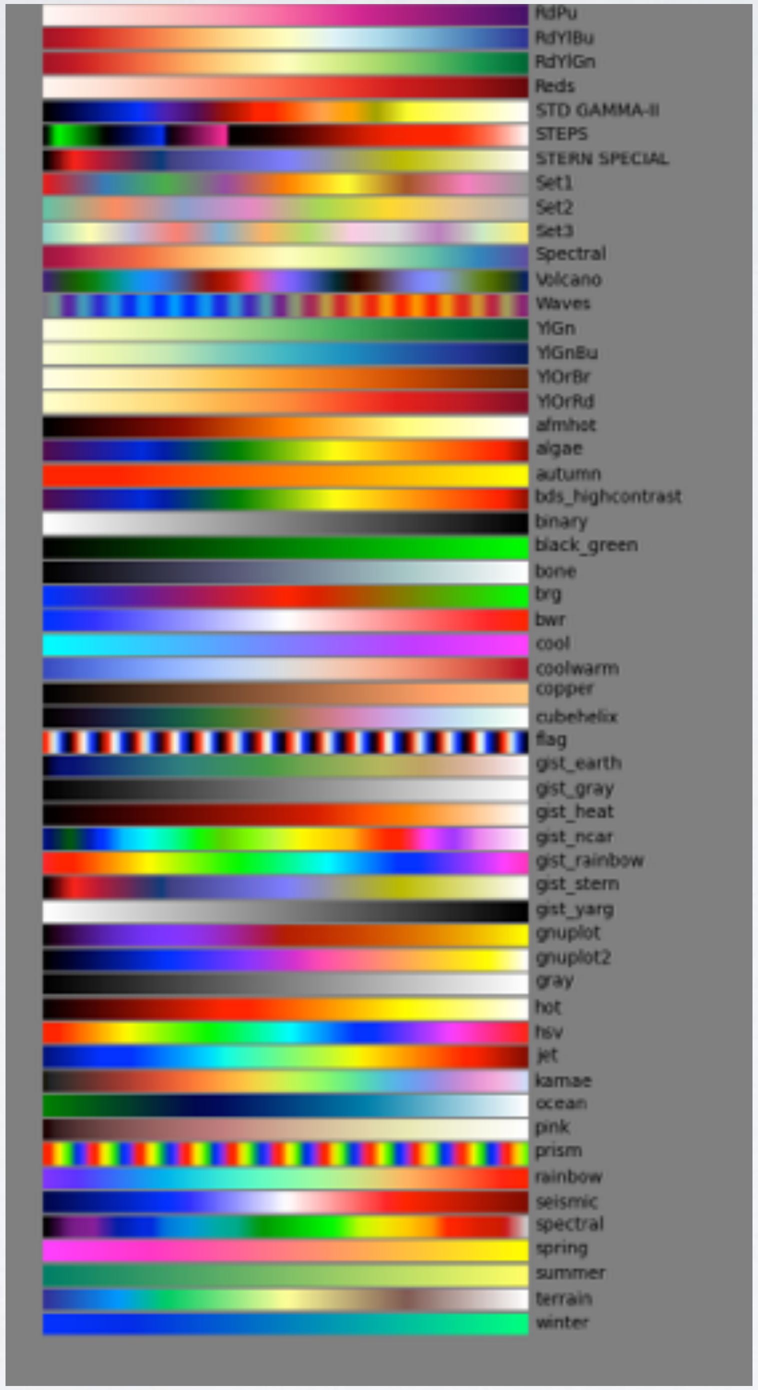
```
(yt)> yt plot --colormap Accent -p -g Density -a 2
sample_data/IsolatedGravity/galaxy0030/galaxy0030
```



Command line **yt**

Image changes





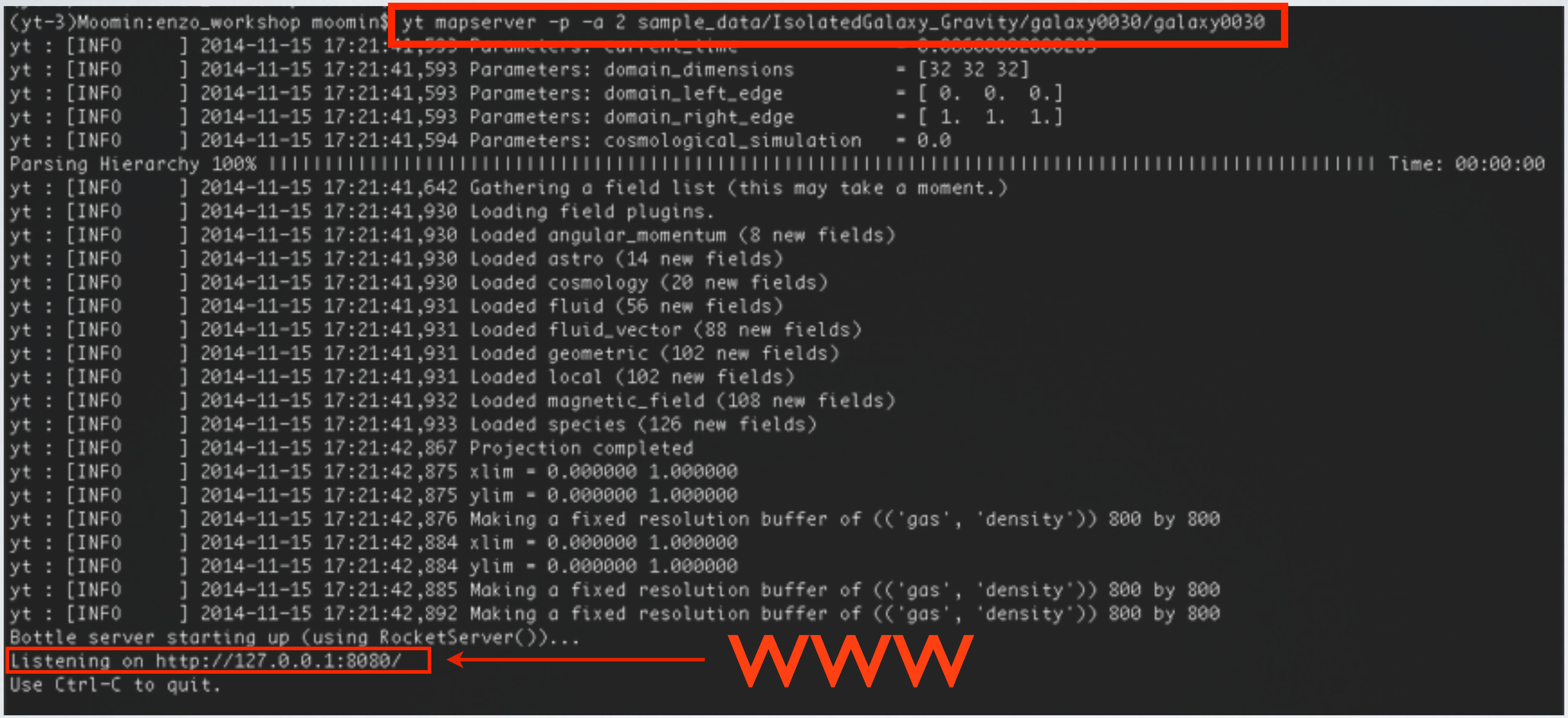
color maps

`http://yt-project.org/docs/dev/visualizing/colormaps/index.html`

Command line **yt**

Mapserver

```
(yt)> yt mapserver -p -a 2 sample_data/IsolatedGravity/galaxy0030/galaxy0030
```

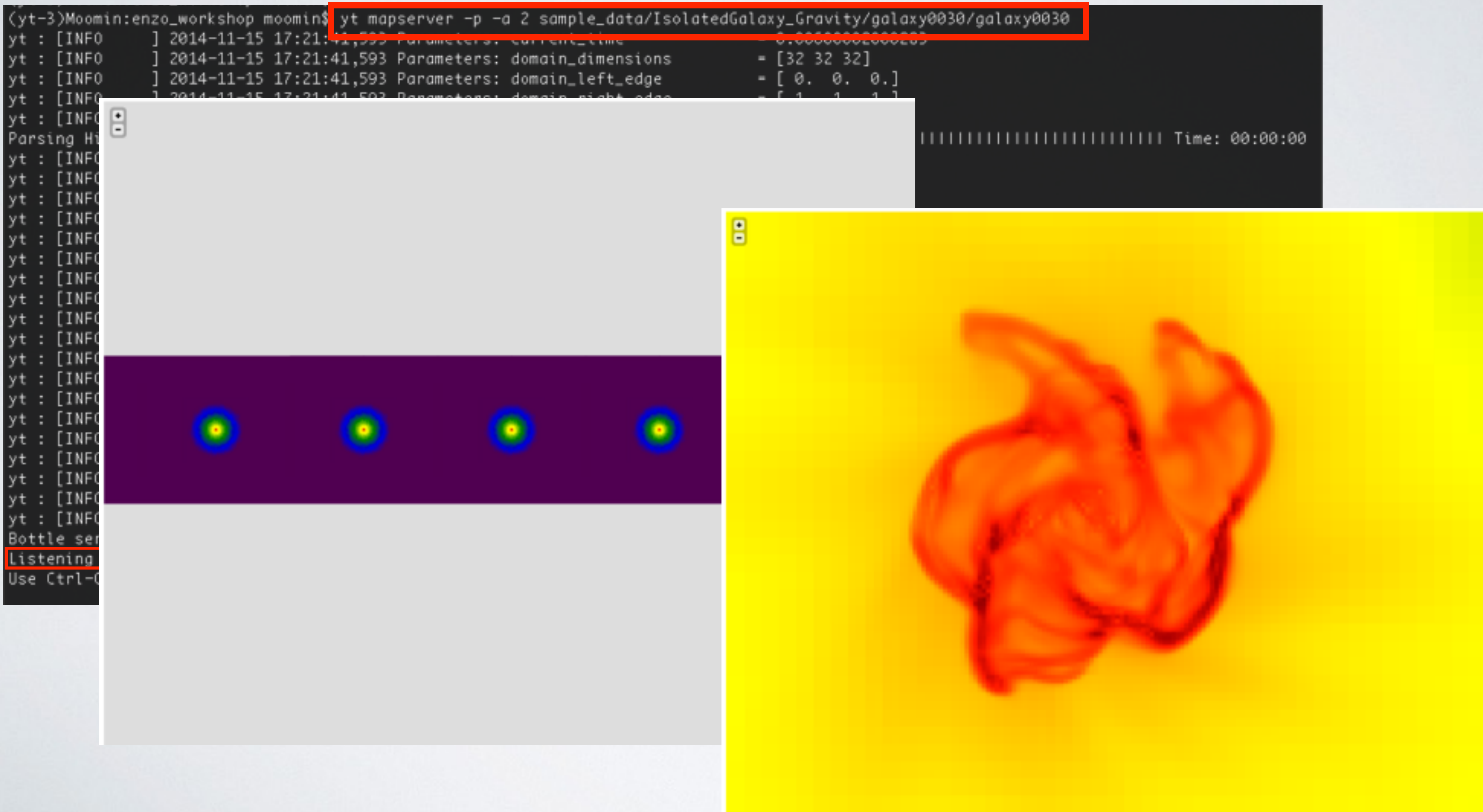


```
(yt-3)Moomin:enzo_workshop moomin$ yt mapserver -p -a 2 sample_data/IsolatedGalaxy_Gravity/galaxy0030/galaxy0030
yt : [INFO     ] 2014-11-15 17:21:41,593 Parameters: current_time              = 0.00600000200283
yt : [INFO     ] 2014-11-15 17:21:41,593 Parameters: domain_dimensions         = [32 32 32]
yt : [INFO     ] 2014-11-15 17:21:41,593 Parameters: domain_left_edge          = [ 0.  0.  0.]
yt : [INFO     ] 2014-11-15 17:21:41,593 Parameters: domain_right_edge         = [ 1.  1.  1.]
yt : [INFO     ] 2014-11-15 17:21:41,594 Parameters: cosmological_simulation   = 0.0
Parsing Hierarchy 100% |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| Time: 00:00:00
yt : [INFO     ] 2014-11-15 17:21:41,642 Gathering a field list (this may take a moment.)
yt : [INFO     ] 2014-11-15 17:21:41,930 Loading field plugins.
yt : [INFO     ] 2014-11-15 17:21:41,930 Loaded angular_momentum (8 new fields)
yt : [INFO     ] 2014-11-15 17:21:41,930 Loaded astro (14 new fields)
yt : [INFO     ] 2014-11-15 17:21:41,930 Loaded cosmology (20 new fields)
yt : [INFO     ] 2014-11-15 17:21:41,931 Loaded fluid (56 new fields)
yt : [INFO     ] 2014-11-15 17:21:41,931 Loaded fluid_vector (88 new fields)
yt : [INFO     ] 2014-11-15 17:21:41,931 Loaded geometric (102 new fields)
yt : [INFO     ] 2014-11-15 17:21:41,931 Loaded local (102 new fields)
yt : [INFO     ] 2014-11-15 17:21:41,932 Loaded magnetic_field (108 new fields)
yt : [INFO     ] 2014-11-15 17:21:41,933 Loaded species (126 new fields)
yt : [INFO     ] 2014-11-15 17:21:42,867 Projection completed
yt : [INFO     ] 2014-11-15 17:21:42,875 xlim = 0.000000 1.000000
yt : [INFO     ] 2014-11-15 17:21:42,875 ylim = 0.000000 1.000000
yt : [INFO     ] 2014-11-15 17:21:42,876 Making a fixed resolution buffer of (('gas', 'density')) 800 by 800
yt : [INFO     ] 2014-11-15 17:21:42,884 xlim = 0.000000 1.000000
yt : [INFO     ] 2014-11-15 17:21:42,884 ylim = 0.000000 1.000000
yt : [INFO     ] 2014-11-15 17:21:42,885 Making a fixed resolution buffer of (('gas', 'density')) 800 by 800
yt : [INFO     ] 2014-11-15 17:21:42,892 Making a fixed resolution buffer of (('gas', 'density')) 800 by 800
Bottle server starting up (using RocketServer())...
Listening on http://127.0.0.1:8080/
Use Ctrl-C to quit.
```

Command line **yt**

Mapserver

```
(yt)> yt mapserver -p -a 2 sample_data/IsolatedGravity/galaxy0030/galaxy0030
```



```
(yt-3)Moomin:enzo_workshop moomin$ yt mapserver -p -a 2 sample_data/IsolatedGalaxy_Gravity/galaxy0030/galaxy0030
yt : [INFO     ] 2014-11-15 17:21:
yt : [INFO     ] 2014-11-15 17:21:41,593 Parameters: domain_dimensions         = [32 32 32]
yt : [INFO     ] 2014-11-15 17:21:41,593 Parameters: domain_left_edge          = [ 0.  0.  0.]
yt : [INFO
yt : [INFO                                                                           |||||||||||||||||||||||||| Time: 00:00:00
Parsing Hi
yt : [INFO
yt : [INFO
yt : [INFO
yt : [INFO
yt : [INFO
yt : [INFO
yt : [INFO
yt : [INFO
yt : [INFO
yt : [INFO
yt : [INFO
yt : [INFO
yt : [INFO
yt : [INFO
yt : [INFO
yt : [INFO
yt : [INFO
yt : [INFO
yt : [INFO
Bottle ser
Listening
Use Ctrl-C
```

iPython notebook

Command line is quick

but hard to save

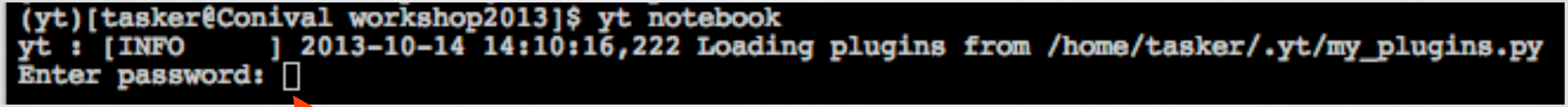
and share

Let's try iPython notebook:

yt in your web browser

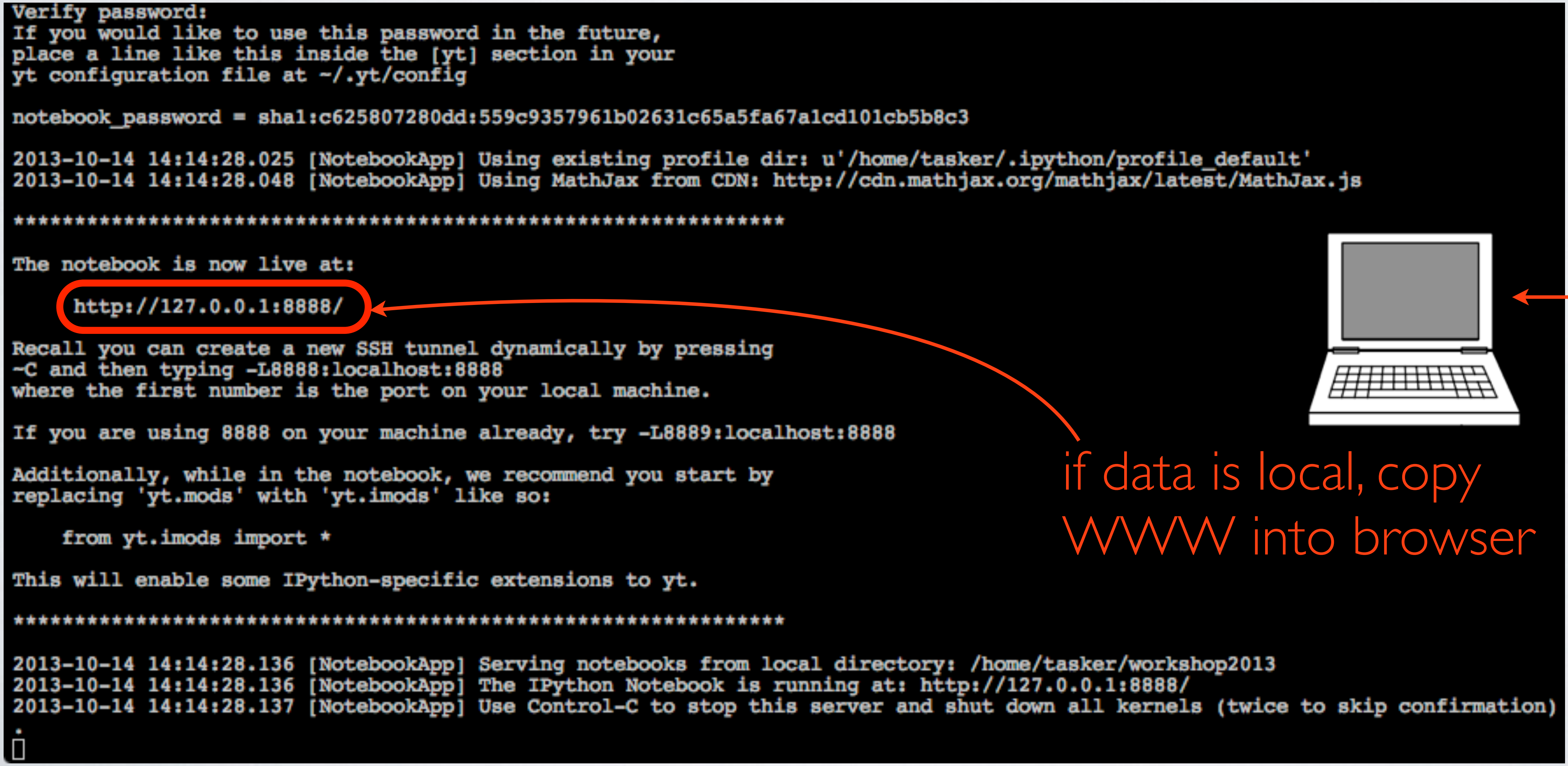
iPython notebook

```
(yt)> yt notebook
```



```
(yt)[tasker@Conival workshop2013]$ yt notebook
yt : [INFO      ] 2013-10-14 14:10:16,222 Loading plugins from /home/tasker/.yt/my_plugins.py
Enter password: ▯
```

any password OK

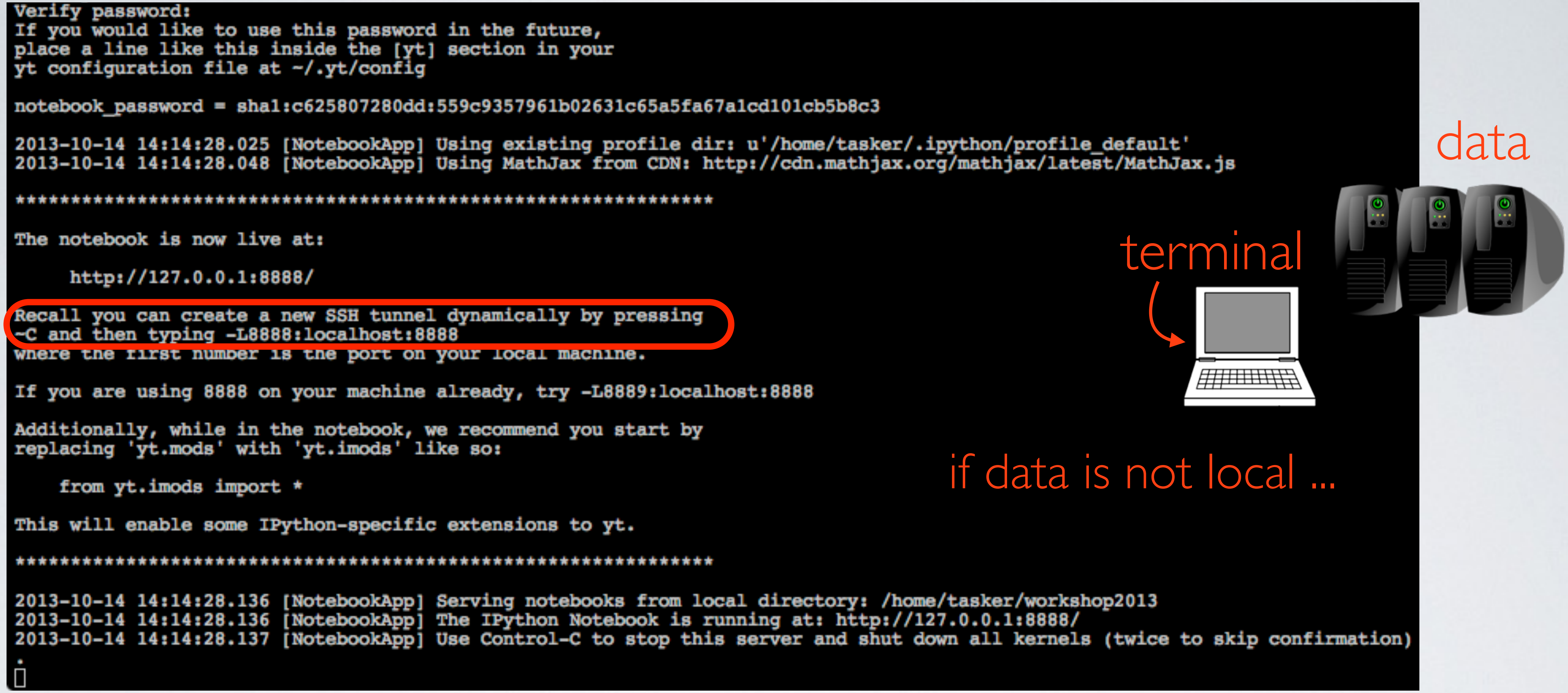


```
Verify password:
If you would like to use this password in the future,
place a line like this inside the [yt] section in your
yt configuration file at ~/.yt/config

notebook_password = sha1:c625807280dd:559c9357961b02631c65a5fa67a1cd101cb5b8c3

2013-10-14 14:14:28.025 [NotebookApp] Using existing profile dir: u'/home/tasker/.ipython/profile_default'
2013-10-14 14:14:28.048 [NotebookApp] Using MathJax from CDN: http://cdn.mathjax.org/mathjax/latest/MathJax.js

**********************************************************************

The notebook is now live at:

     http://127.0.0.1:8888/

Recall you can create a new SSH tunnel dynamically by pressing
~C and then typing -L8888:localhost:8888
where the first number is the port on your local machine.

If you are using 8888 on your machine already, try -L8889:localhost:8888

Additionally, while in the notebook, we recommend you start by
replacing 'yt.mods' with 'yt.imods' like so:

    from yt.imods import *

This will enable some IPython-specific extensions to yt.

**********************************************************************

2013-10-14 14:14:28.136 [NotebookApp] Serving notebooks from local directory: /home/tasker/workshop2013
2013-10-14 14:14:28.136 [NotebookApp] The IPython Notebook is running at: http://127.0.0.1:8888/
2013-10-14 14:14:28.137 [NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation)
.
▯
```

data

if data is local, copy WWW into browser

iPython notebook



```
Verify password:
If you would like to use this password in the future,
place a line like this inside the [yt] section in your
yt configuration file at ~/.yt/config

notebook_password = sha1:c625807280dd:559c9357961b02631c65a5fa67a1cd101cb5b8c3

2013-10-14 14:14:28.025 [NotebookApp] Using existing profile dir: u'/home/tasker/.ipython/profile_default'
2013-10-14 14:14:28.048 [NotebookApp] Using MathJax from CDN: http://cdn.mathjax.org/mathjax/latest/MathJax.js

******************************************************************

The notebook is now live at:

     http://127.0.0.1:8888/

Recall you can create a new SSH tunnel dynamically by pressing
~C and then typing -L8888:localhost:8888
where the first number is the port on your local machine.

If you are using 8888 on your machine already, try -L8889:localhost:8888

Additionally, while in the notebook, we recommend you start by
replacing 'yt.mods' with 'yt.imods' like so:

    from yt.imods import *

This will enable some IPython-specific extensions to yt.

******************************************************************

2013-10-14 14:14:28.136 [NotebookApp] Serving notebooks from local directory: /home/tasker/workshop2013
2013-10-14 14:14:28.136 [NotebookApp] The IPython Notebook is running at: http://127.0.0.1:8888/
2013-10-14 14:14:28.137 [NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation)
```

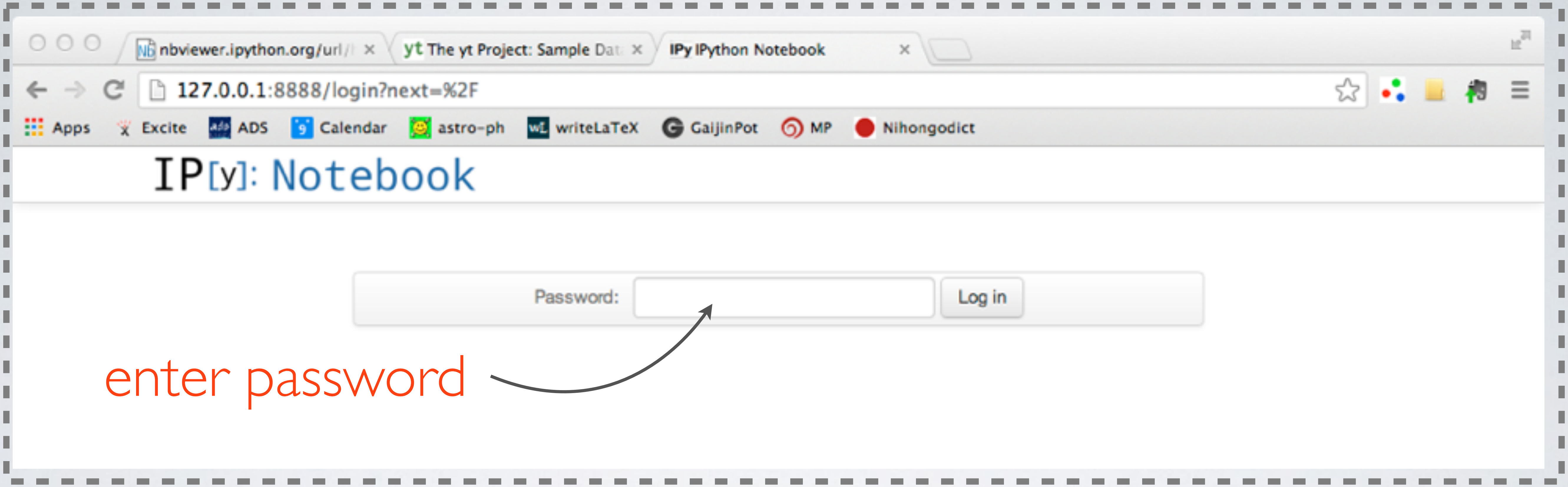
if data is not local ...

```
(yt)> ~C
ssh > -L88XX:localhost:88XX
```

Then go to:

`http://127.0.0.1:8888/` in web browser

iPython notebook

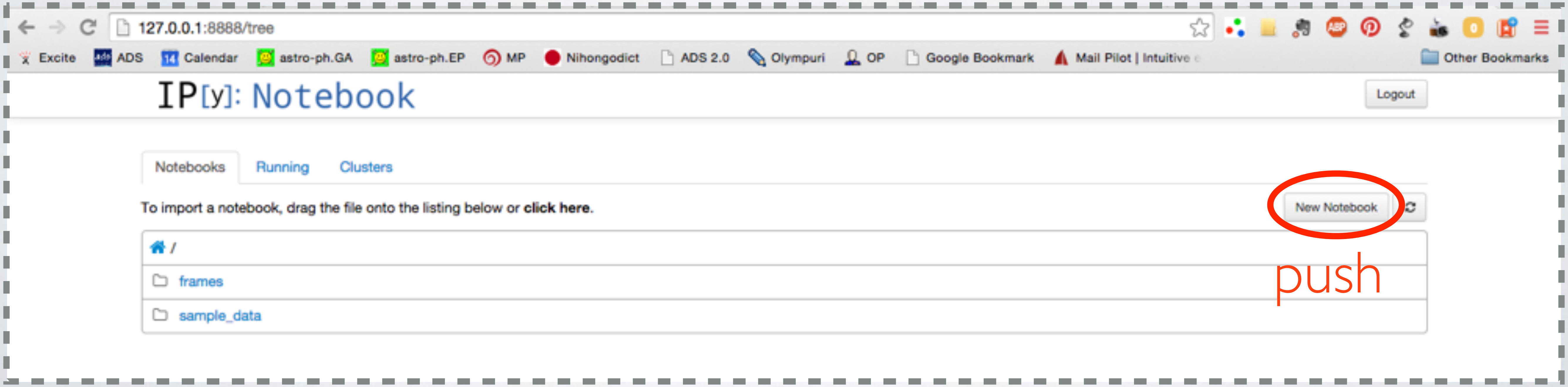


127.0.0.1:8888/login?next=%2F

IP[y]: Notebook

Password:

enter password



127.0.0.1:8888/tree

IP[y]: Notebook

To import a notebook, drag the file onto the listing below or **click here**.

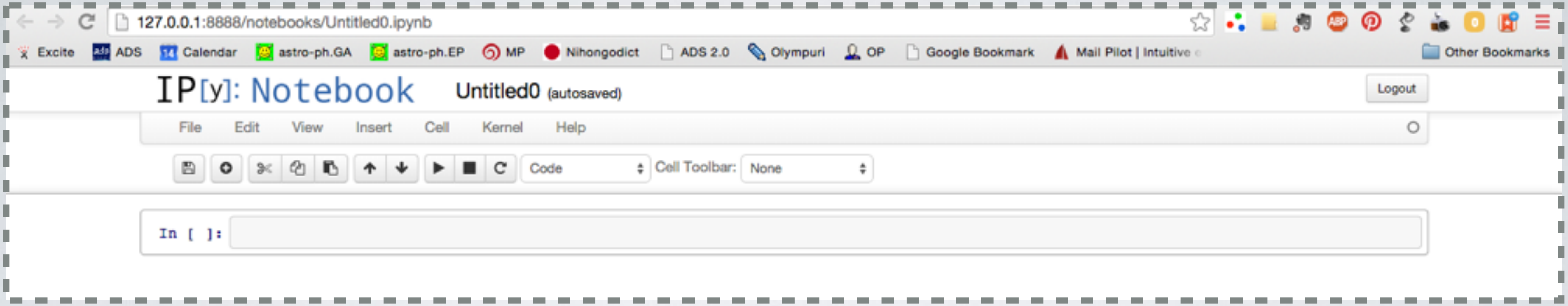
/

frames

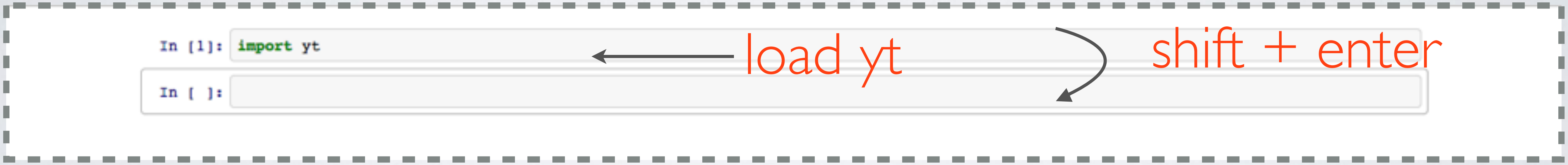
sample_data

push

iPython notebook



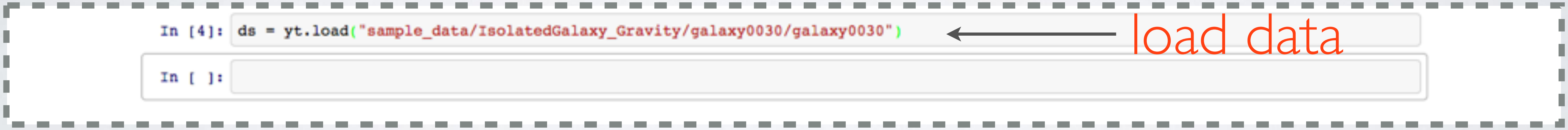
```
In [ ]:
```



```
In [1]: import yt
In [ ]:
```

load yt

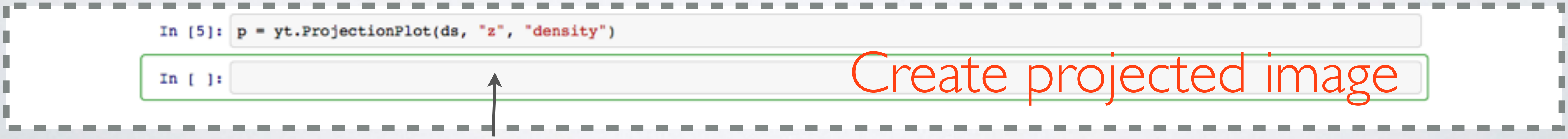
shift + enter



```
In [4]: ds = yt.load("sample_data/IsolatedGalaxy_Gravity/galaxy0030/galaxy0030")
In [ ]:
```

load data

field

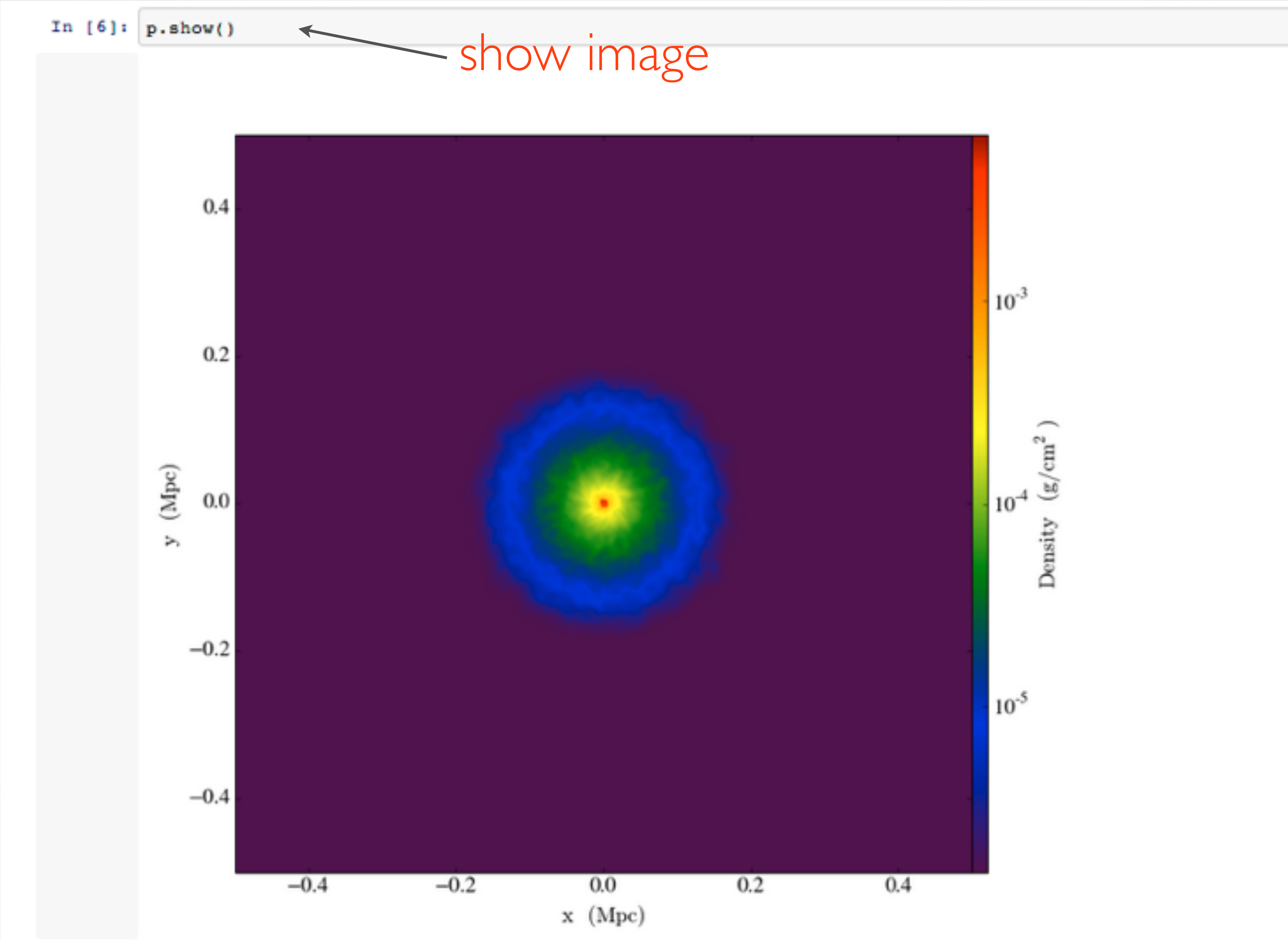


```
In [5]: p = yt.ProjectionPlot(ds, "z", "density")
In [ ]:
```

Create projected image

axis

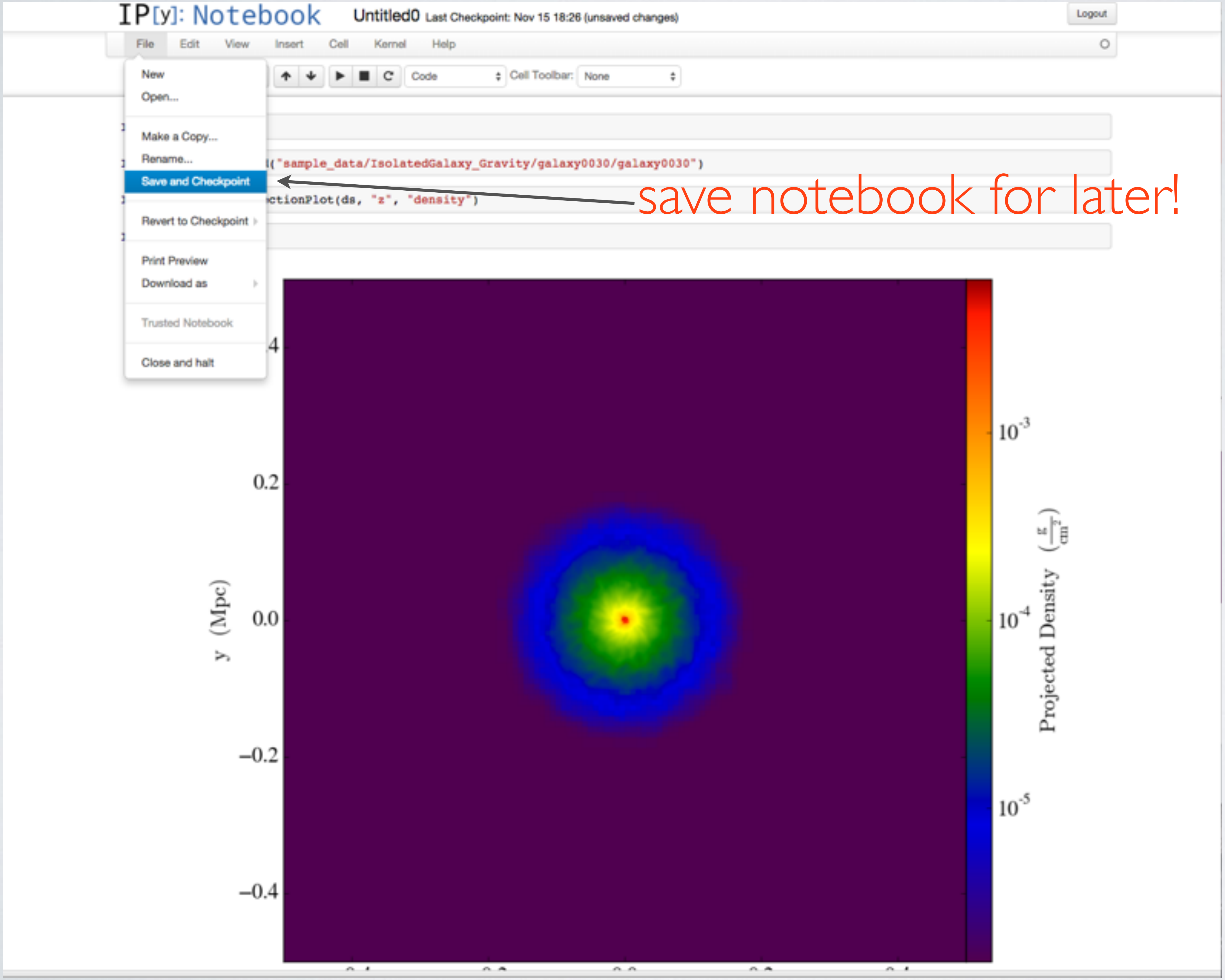
iPython notebook



```
In [6]: p.show()
```

show image

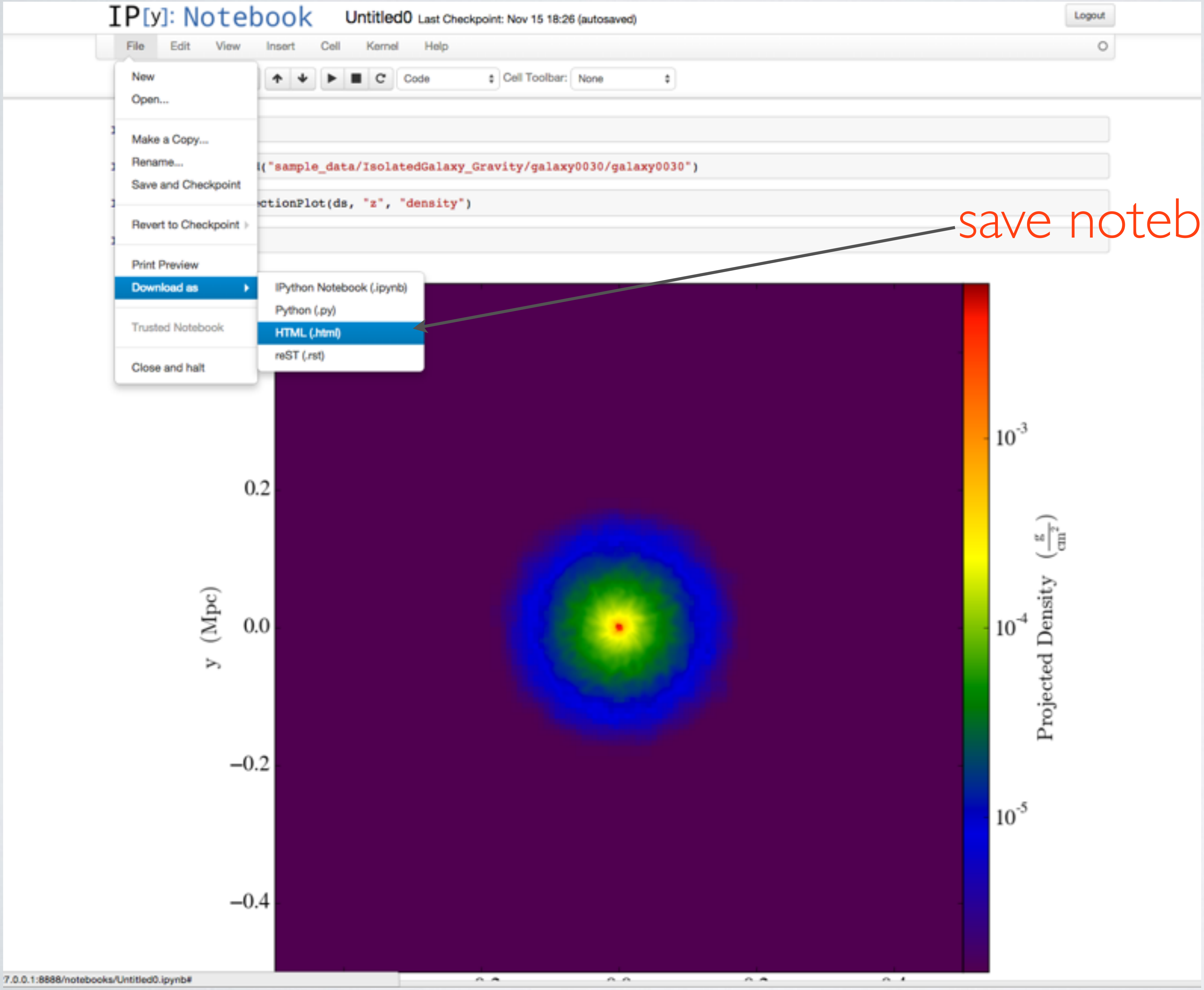
iPython notebook



```
l("sample_data/IsolatedGalaxy_Gravity/galaxy0030/galaxy0030")
ctionPlot(ds, "z", "density")
```

save notebook for later!

iPython notebook



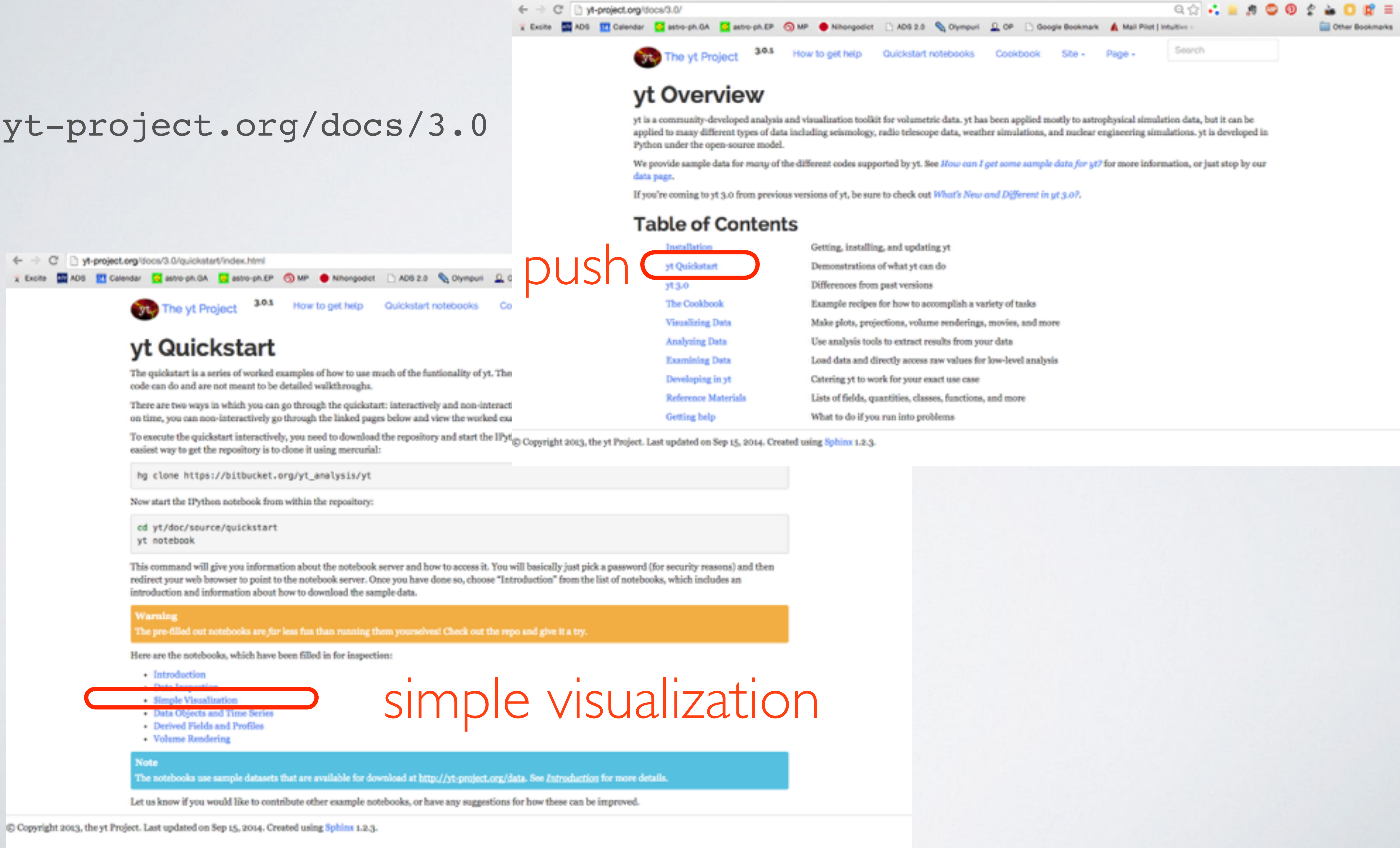
```
("sample_data/IsolatedGalaxy_Gravity/galaxy0030/galaxy0030")
ctionPlot(ds, "z", "density")
```

save notebook for later!

iPython notebook

Let's try a bigger example

`yt-project.org/docs/3.0`



yt Overview

yt is a community-developed analysis and visualization toolkit for volumetric data. yt has been applied mostly to astrophysical simulation data, but it can be applied to many different types of data including seismology, radio telescope data, weather simulations, and nuclear engineering simulations. yt is developed in Python under the open-source model.

We provide sample data for *many* of the different codes supported by yt. See *How can I get some sample data for yt?* for more information, or just stop by our data page.

If you're coming to yt 3.0 from previous versions of yt, be sure to check out *What's New and Different in yt 3.0?*.

Table of Contents

push

yt Quickstart

The quickstart is a series of worked examples of how to use much of the funtionality of yt. The
code can do and are not meant to be detailed walkthroughs.

There are two ways in which you can go through the quickstart: interactively and non-interact
on time, you can non-interactively go through the linked pages below and view the worked exa

To execute the quickstart interactively, you need to download the repository and start the IPyt
easiest way to get the repository is to clone it using mercurial:

```
hg clone https://bitbucket.org/yt_analysis/yt
```

Now start the IPython notebook from within the repository:

```
cd yt/doc/source/quickstart
yt notebook
```

This command will give you information about the notebook server and how to access it. You will basically just pick a password (for security reasons) and then redirect your web browser to point to the notebook server. Once you have done so, choose "Introduction" from the list of notebooks, which includes an introduction and information about how to download the sample data.

Warning
The pre-filled out notebooks are *far* less fun than running them yourselves! Check out the repo and give it a try.

Here are the notebooks, which have been filled in for inspection:

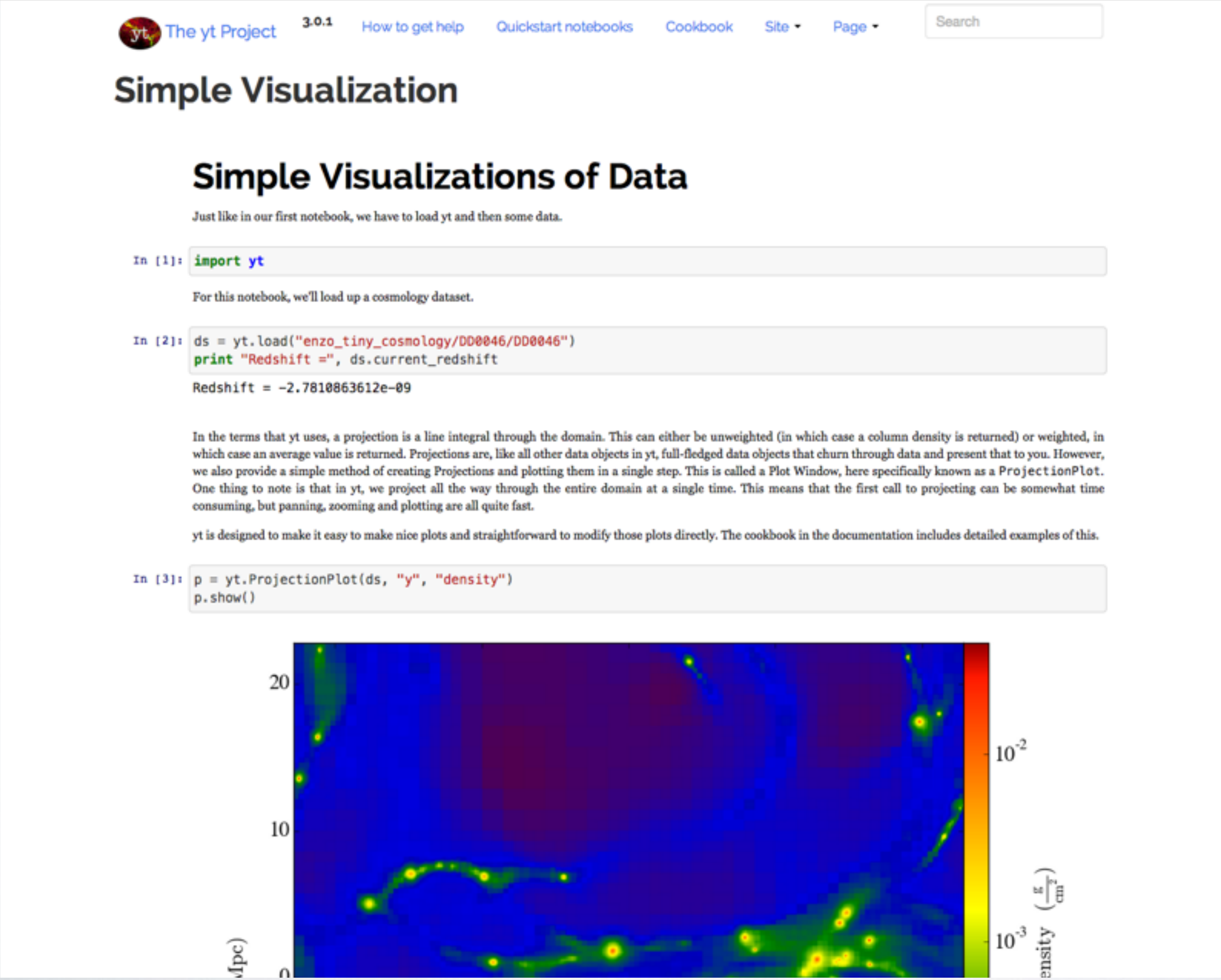
- Introduction
- Data Inspection
- Simple Visualization
- Data Objects and Time Series
- Derived Fields and Profiles
- Volume Rendering

Note
The notebooks use sample datasets that are available for download at http://yt-project.org/data. See *Introduction* for more details.

Let us know if you would like to contribute other example notebooks, or have any suggestions for how these can be improved.

simple visualization

iPython notebook



Simple Visualization

Simple Visualizations of Data

Just like in our first notebook, we have to load yt and then some data.

```
In [1]: import yt
```

For this notebook, we'll load up a cosmology dataset.

```
In [2]: ds = yt.load("enzo_tiny_cosmology/DD0046/DD0046")
        print "Redshift =", ds.current_redshift

Redshift = -2.7810863612e-09
```

In the terms that yt uses, a projection is a line integral through the domain. This can either be unweighted (in which case a column density is returned) or weighted, in which case an average value is returned. Projections are, like all other data objects in yt, full-fledged data objects that churn through data and present that to you. However, we also provide a simple method of creating Projections and plotting them in a single step. This is called a Plot Window, here specifically known as a `ProjectionPlot`. One thing to note is that in yt, we project all the way through the entire domain at a single time. This means that the first call to projecting can be somewhat time consuming, but panning, zooming and plotting are all quite fast.

yt is designed to make it easy to make nice plots and straightforward to modify those plots directly. The cookbook in the documentation includes detailed examples of this.

```
In [3]: p = yt.ProjectionPlot(ds, "y", "density")
        p.show()
```

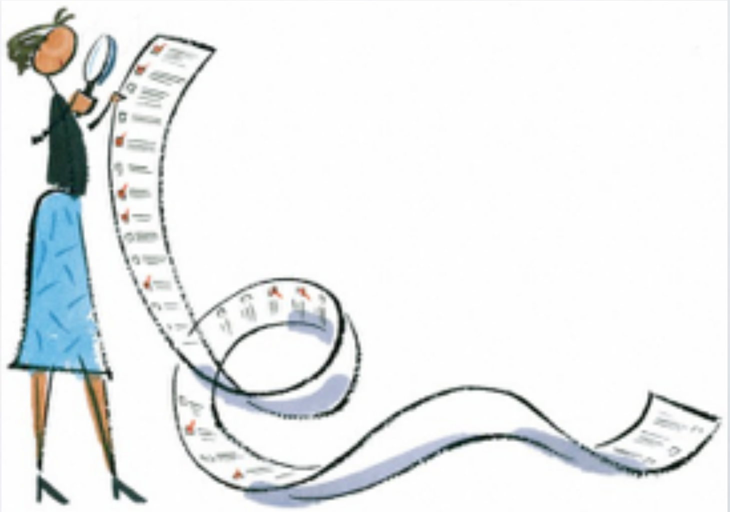
do

Scripts

Command line & iPython notebook are great for quick analysis...

But what if you want repeat the same commands 100s of times?

or write a longer programme?

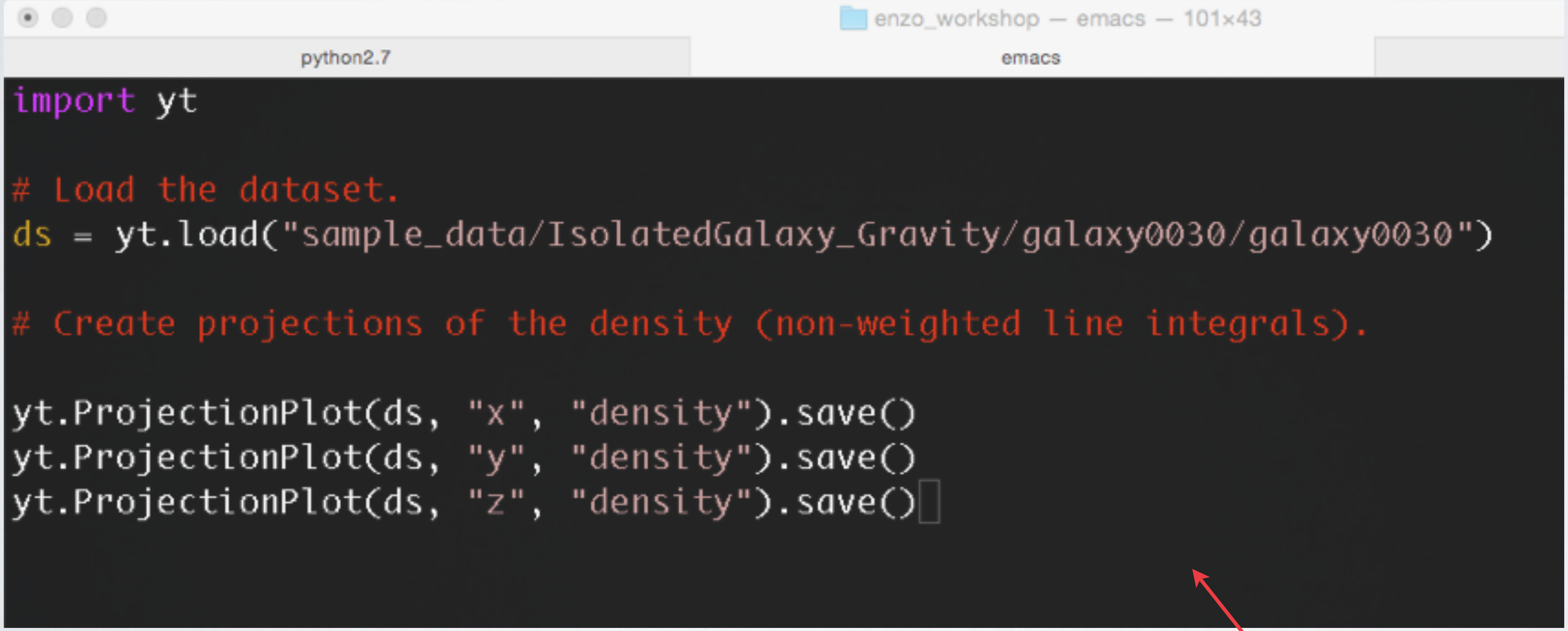


Scripts

Let's write a script:

```
(yt) > emacs -nw simple_projection.py
```

any text editor, e.g. emacs, vim ....



```python
import yt

# Load the dataset.
ds = yt.load("sample_data/IsolatedGalaxy_Gravity/galaxy0030/galaxy0030")

# Create projections of the density (non-weighted line integrals).

yt.ProjectionPlot(ds, "x", "density").save()
yt.ProjectionPlot(ds, "y", "density").save()
yt.ProjectionPlot(ds, "z", "density").save()
```

save this

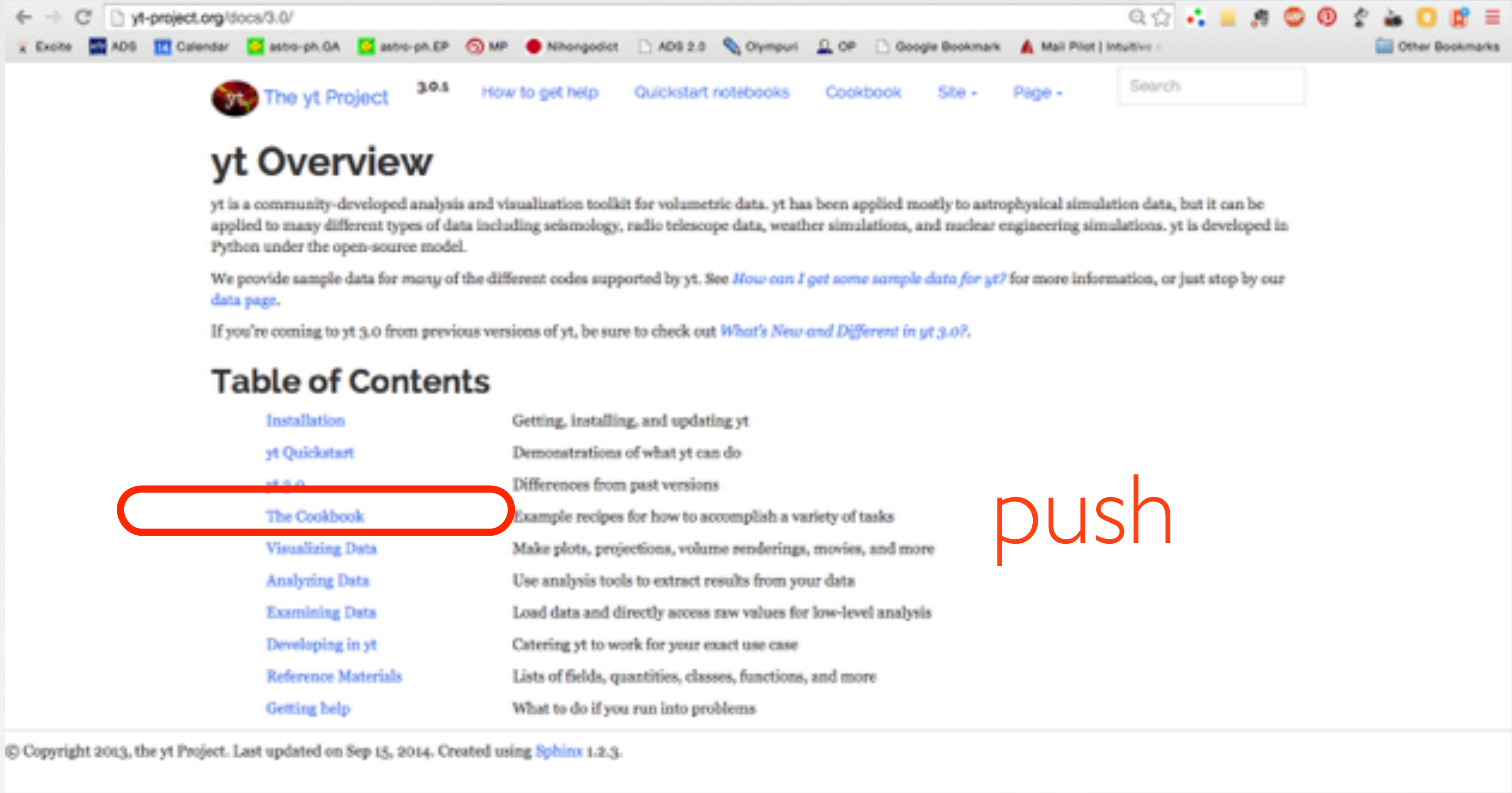
Scripts

```
(yt)> iyt

==================
| Welcome to yt! |
==================


In [1]: execfile(“simple_projection.py”)

In [2]: exit()

ls

yt upload_image galaxy0030_Projection_z_Density.png
```

Scripts

What about other plots?

`yt-project.org/docs/3.0`



yt Overview

yt is a community-developed analysis and visualization toolkit for volumetric data. yt has been applied mostly to astrophysical simulation data, but it can be applied to many different types of data including seismology, radio telescope data, weather simulations, and nuclear engineering simulations. yt is developed in Python under the open-source model.

We provide sample data for *many* of the different codes supported by yt. See *How can I get some sample data for yt?* for more information, or just stop by our data page.

If you're coming to yt 3.0 from previous versions of yt, be sure to check out *What's New and Different in yt 3.0?*.

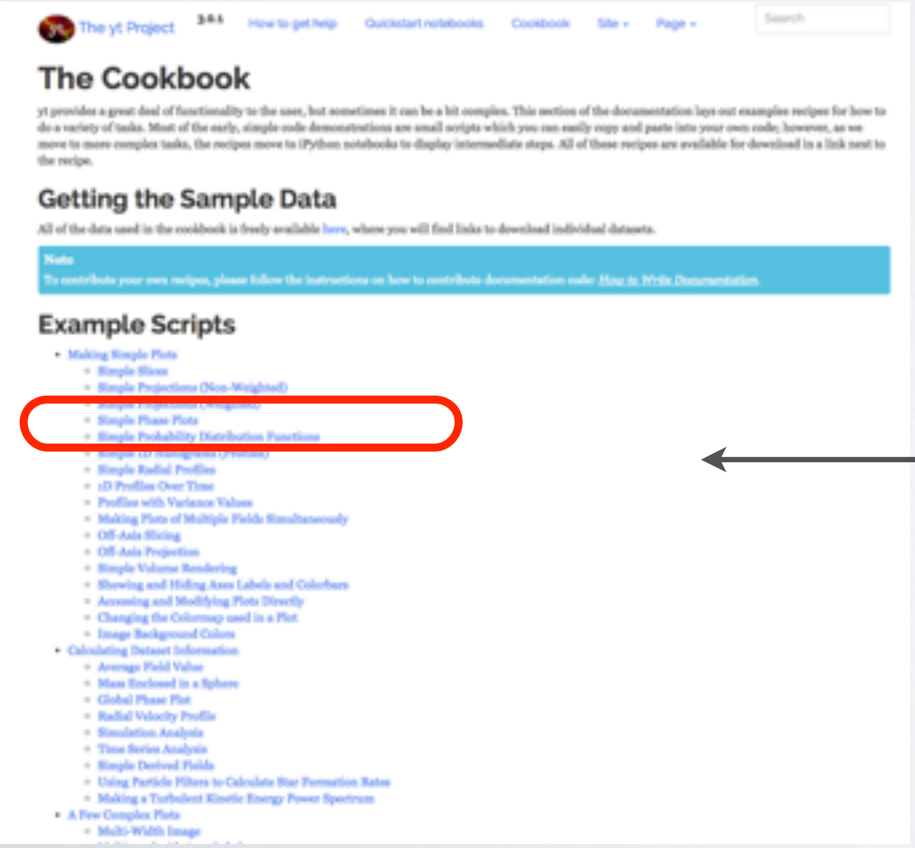
Table of Contents

Installation	Getting, installing, and updating yt
yt Quickstart	Demonstrations of what yt can do
yt 3.0	Differences from past versions
The Cookbook	Example recipes for how to accomplish a variety of tasks
Visualizing Data	Make plots, projections, volume renderings, movies, and more
Analyzing Data	Use analysis tools to extract results from your data
Examining Data	Load data and directly access raw values for low-level analysis
Developing in yt	Catering yt to work for your exact use case
Reference Materials	Lists of fields, quantities, classes, functions, and more
Getting help	What to do if you run into problems

© Copyright 2013, the yt Project. Last updated on Sep 15, 2014. Created using Sphinx 1.2.3.

push

'The Cookbook'



The Cookbook

yt provides a great deal of functionality to the user, but sometimes it can be a bit complex. This section of the documentation lays out examples recipes for how to do a variety of tasks. Most of the early, simple code demonstrations are small scripts which you can easily copy and paste into your own code; however, as we move to more complex tasks, the recipes move to IPython notebooks to display intermediate steps. All of these recipes are available for download in a link next to the recipe.

Getting the Sample Data

All of the data used in the cookbook is freely available here, where you will find links to download individual datasets.

Note
To contribute your own recipes, please follow the instructions on how to contribute documentation code: *How to Write Documentation*.

Example Scripts

- Making Simple Plots
  - Simple Slices
  - Simple Projections (Non-Weighted)
  - Simple Projections (Weighted)
  - Simple Phase Plots
  - Simple Probability Distribution Functions
  - Simple 1D Histograms (Profiles)
  - Simple Radial Profiles
  - 1D Profiles Over Time
  - Profiles with Variance Values
  - Making Plots of Multiple Fields Simultaneously
  - Off-Axis Slicing
  - Off-Axis Projection
  - Simple Volume Rendering
  - Showing and Hiding Axes Labels and Colorbars
  - Accessing and Modifying Plots Directly
  - Changing the Colormap used in a Plot
  - Image Background Colors
- Calculating Dataset Information
  - Average Field Value
  - Mass Enclosed in a Sphere
  - Global Phase Plot
  - Radial Velocity Profile
  - Simulation Analysis
  - Time Series Analysis
  - Simple Derived Fields
  - Using Particle Filters to Calculate Star Formation Rates
  - Making a Turbulent Kinetic Energy Power Spectrum
- A Few Complex Plots
  - Multi-Width Image

← Many examples!

Let's try 'Simple Phase Plots'

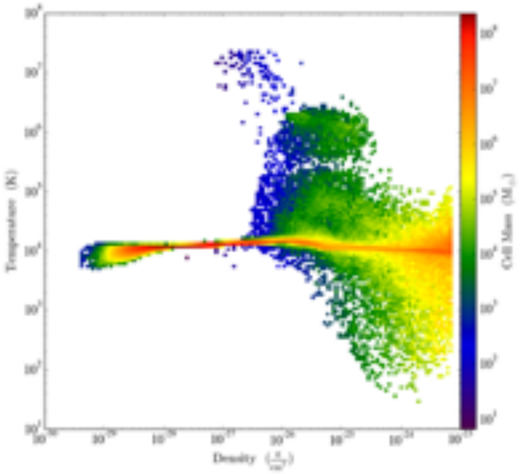
Scripts

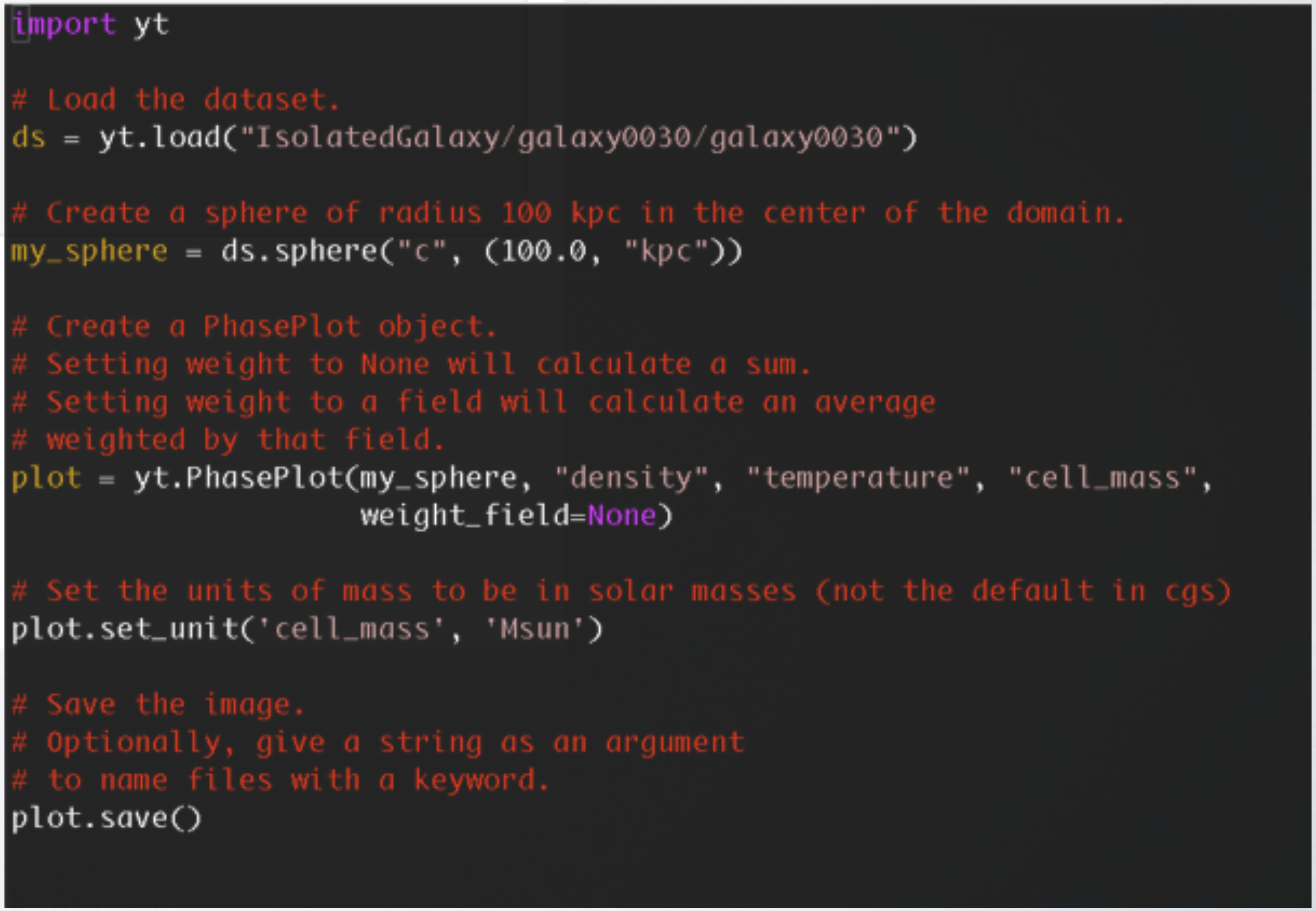
Simple Phase Plots

This demonstrates how to make a phase plot. Phase plots can be thought of as two-dimensional histograms, where the value is either the weighted-average or the total accumulation in a cell. See *2D Phase Plots* for more information.

(simple_phase.py)

```
import yt

# Load the dataset.
ds = yt.load("IsolatedGalaxy/galaxy0030/galaxy0030")

# Create a sphere of radius 100 kpc in the center of the domain.
my_sphere = ds.sphere("c", (100.0, "kpc"))

# Create a PhasePlot object.
# Setting weight to None will calculate a sum.
# Setting weight to a field will calculate an average
# weighted by that field.
plot = yt.PhasePlot(my_sphere, "density", "temperature", "cell_mass",
                    weight_field=None)

# Set the units of mass to be in solar masses (not the default in cgs)
plot.set_unit('cell_mass', 'Msun')

# Save the image.
# Optionally, give a string as an argument
# to name files with a keyword.
plot.save()
```





```
import yt

# Load the dataset.
ds = yt.load("IsolatedGalaxy/galaxy0030/galaxy0030")

# Create a sphere of radius 100 kpc in the center of the domain.
my_sphere = ds.sphere("c", (100.0, "kpc"))

# Create a PhasePlot object.
# Setting weight to None will calculate a sum.
# Setting weight to a field will calculate an average
# weighted by that field.
plot = yt.PhasePlot(my_sphere, "density", "temperature", "cell_mass",
                    weight_field=None)

# Set the units of mass to be in solar masses (not the default in cgs)
plot.set_unit('cell_mass', 'Msun')

# Save the image.
# Optionally, give a string as an argument
# to name files with a keyword.
plot.save()
```

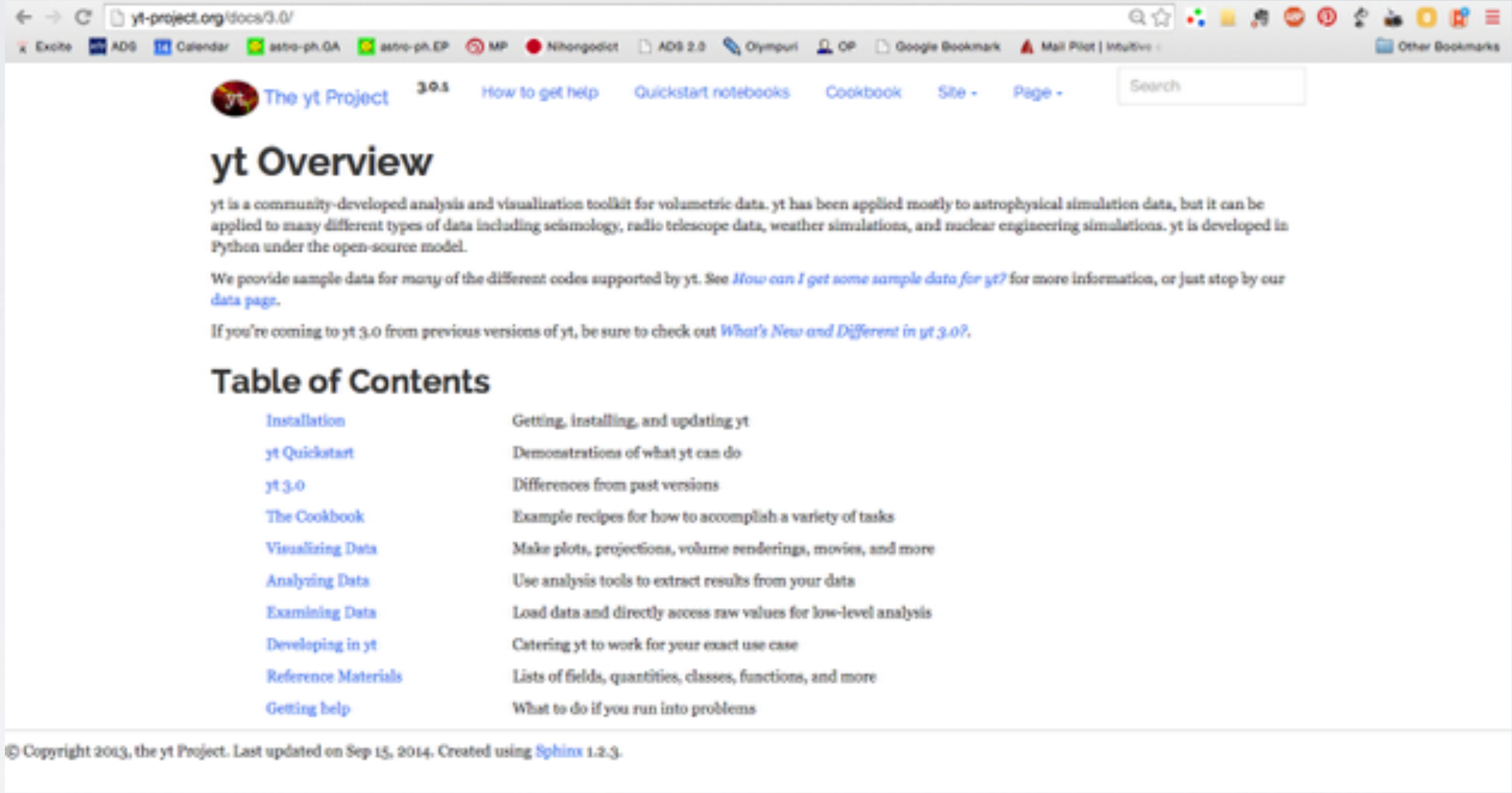
Scripts

```
(yt)> iyt

==================
| Welcome to yt! |
==================


In [1]: execfile(“simple_phase.py”)

In [2]: exit()
```

The docs

Still not enough information?

`yt-project.org/docs/3.0`



yt Overview

yt is a community-developed analysis and visualization toolkit for volumetric data. yt has been applied mostly to astrophysical simulation data, but it can be applied to many different types of data including seismology, radio telescope data, weather simulations, and nuclear engineering simulations. yt is developed in Python under the open-source model.

We provide sample data for *many* of the different codes supported by yt. See *How can I get some sample data for yt?* for more information, or just stop by our data page.

If you're coming to yt 3.0 from previous versions of yt, be sure to check out *What's New and Different in yt 3.0?*.

Table of Contents

Installation	Getting, installing, and updating yt
yt Quickstart	Demonstrations of what yt can do
yt 3.0	Differences from past versions
The Cookbook	Example recipes for how to accomplish a variety of tasks
Visualizing Data	Make plots, projections, volume renderings, movies, and more
Analyzing Data	Use analysis tools to extract results from your data
Examining Data	Load data and directly access raw values for low-level analysis
Developing in yt	Catering yt to work for your exact use case
Reference Materials	Lists of fields, quantities, classes, functions, and more
Getting help	What to do if you run into problems

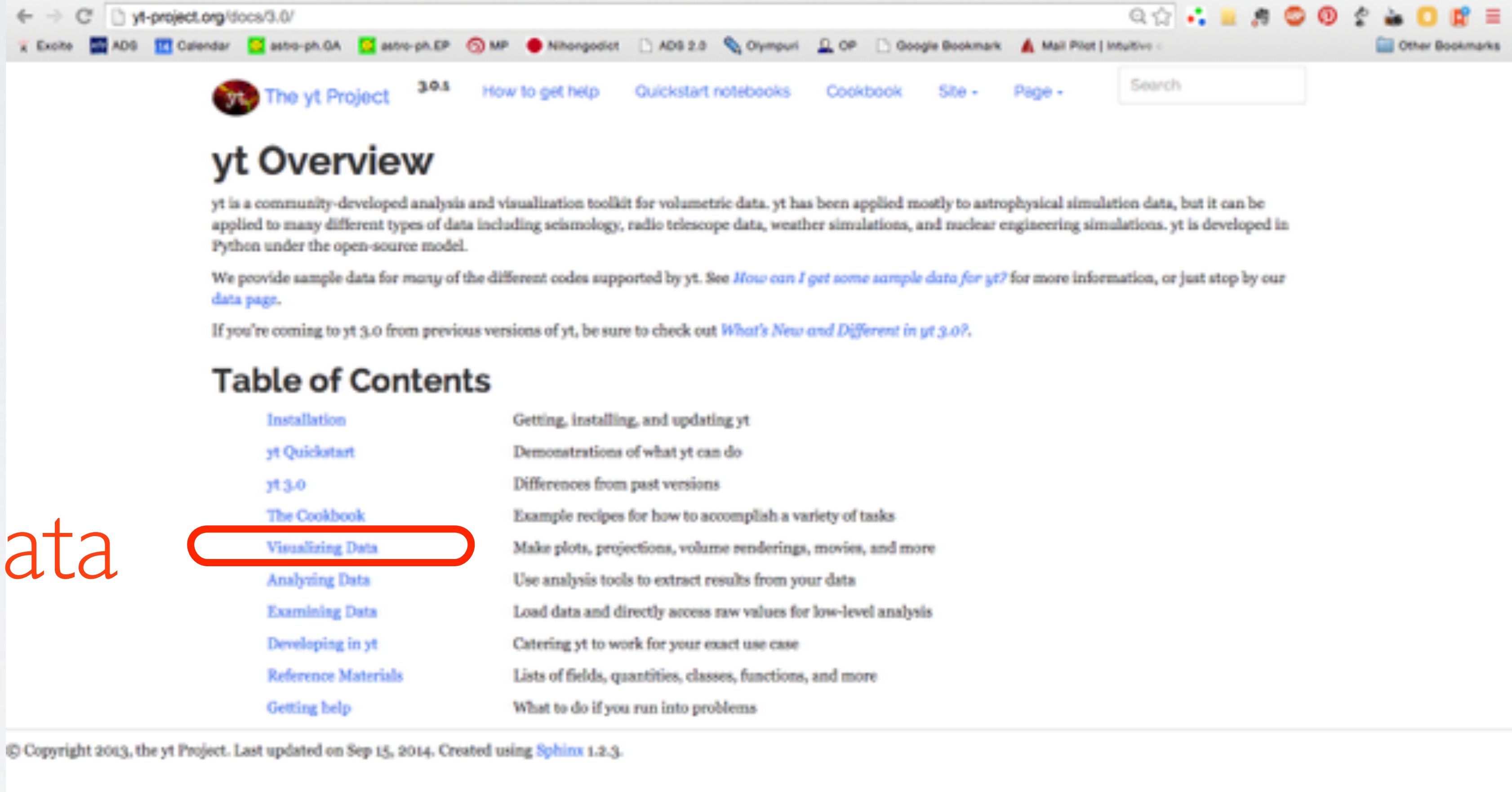
© Copyright 2013, the yt Project. Last updated on Sep 15, 2014. Created using Sphinx 1.2.3.

The documentation (docs) contain much more!

The docs

e.g. let's modify an image

visualizing data



yt Overview

yt is a community-developed analysis and visualization toolkit for volumetric data. yt has been applied mostly to astrophysical simulation data, but it can be applied to many different types of data including seismology, radio telescope data, weather simulations, and nuclear engineering simulations. yt is developed in Python under the open-source model.

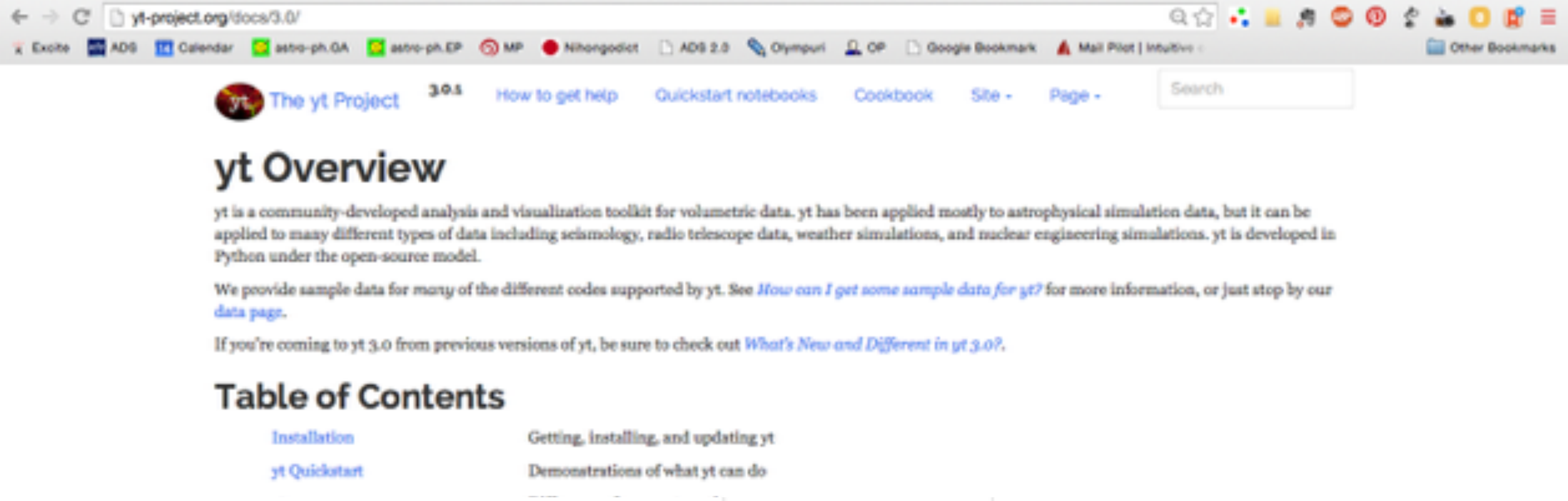
We provide sample data for *many* of the different codes supported by yt. See *How can I get some sample data for yt?* for more information, or just stop by our data page.

If you're coming to yt 3.0 from previous versions of yt, be sure to check out *What's New and Different in yt 3.0?*.

Table of Contents

The docs

e.g. let's modify an image



yt Overview

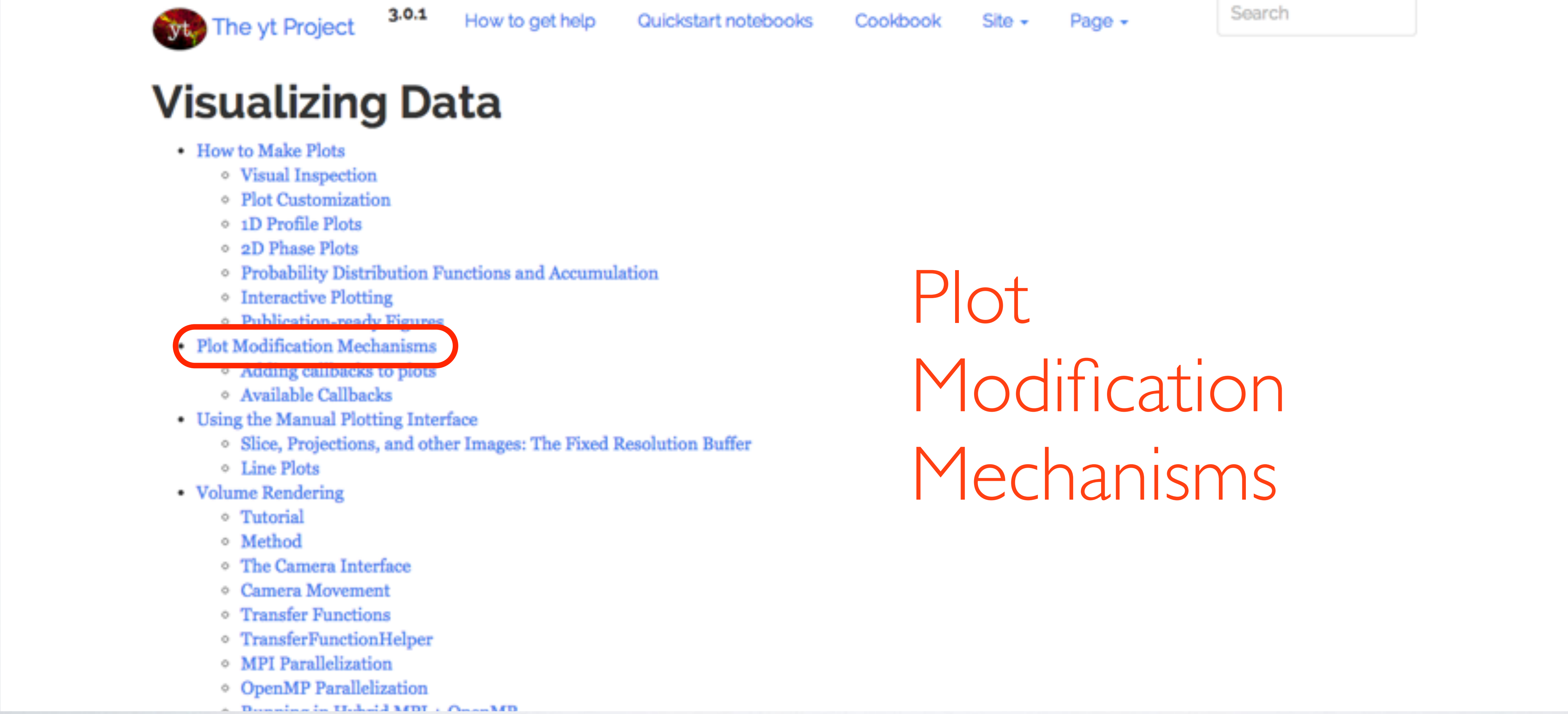
yt is a community-developed analysis and visualization toolkit for volumetric data. yt has been applied mostly to astrophysical simulation data, but it can be applied to many different types of data including seismology, radio telescope data, weather simulations, and nuclear engineering simulations. yt is developed in Python under the open-source model.

We provide sample data for *many* of the different codes supported by yt. See *How can I get some sample data for yt?* for more information, or just stop by our data page.

If you're coming to yt 3.0 from previous versions of yt, be sure to check out *What's New and Different in yt 3.0?*.

Table of Contents

Installation	Getting, installing, and updating yt
yt Quickstart	Demonstrations of what yt can do

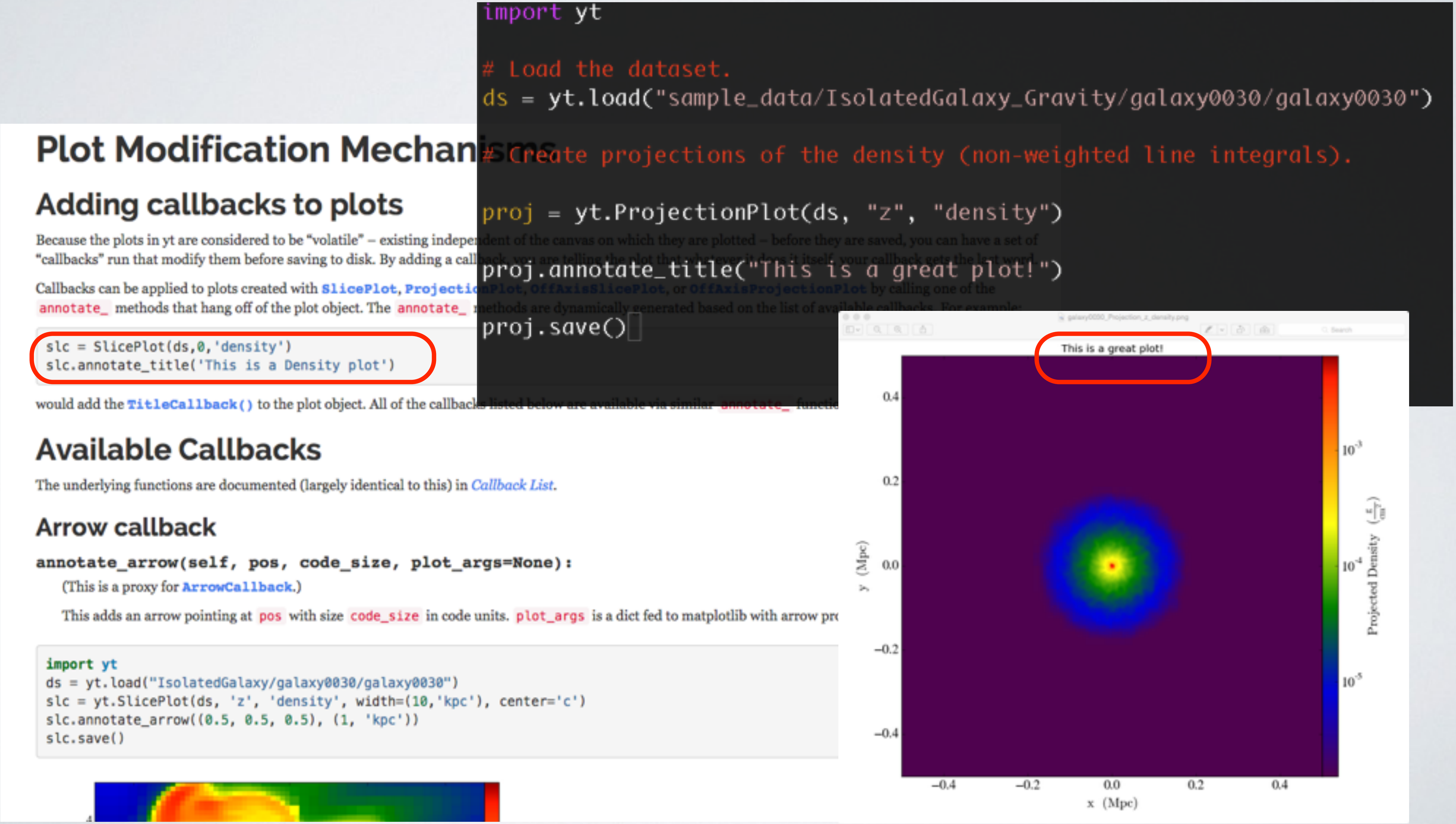


Visualizing Data

Plot Modification Mechanisms

The docs

e.g. let's modify an image



```
import yt

# Load the dataset.
ds = yt.load("sample_data/IsolatedGalaxy_Gravity/galaxy0030/galaxy0030")

# Create projections of the density (non-weighted line integrals).

proj = yt.ProjectionPlot(ds, "z", "density")

proj.annotate_title("This is a great plot!")

proj.save()
```

Plot Modification Mechanisms

Adding callbacks to plots

Because the plots in yt are considered to be "volatile" – existing independent of the canvas on which they are plotted – before they are saved, you can have a set of "callbacks" run that modify them before saving to disk. By adding a callback, you are telling the plot that whatever it does it itself, your callback gets the last word.

Callbacks can be applied to plots created with `SlicePlot`, `ProjectionPlot`, `OffAxisSlicePlot`, or `OffAxisProjectionPlot` by calling one of the `annotate_` methods that hang off of the plot object. The `annotate_` methods are dynamically generated based on the list of available callbacks. For example:

```
slc = SlicePlot(ds,0,'density')
slc.annotate_title('This is a Density plot')
```

would add the `TitleCallback()` to the plot object. All of the callbacks listed below are available via similar `annotate_` functio

Available Callbacks

The underlying functions are documented (largely identical to this) in *Callback List*.

Arrow callback

`annotate_arrow(self, pos, code_size, plot_args=None):`

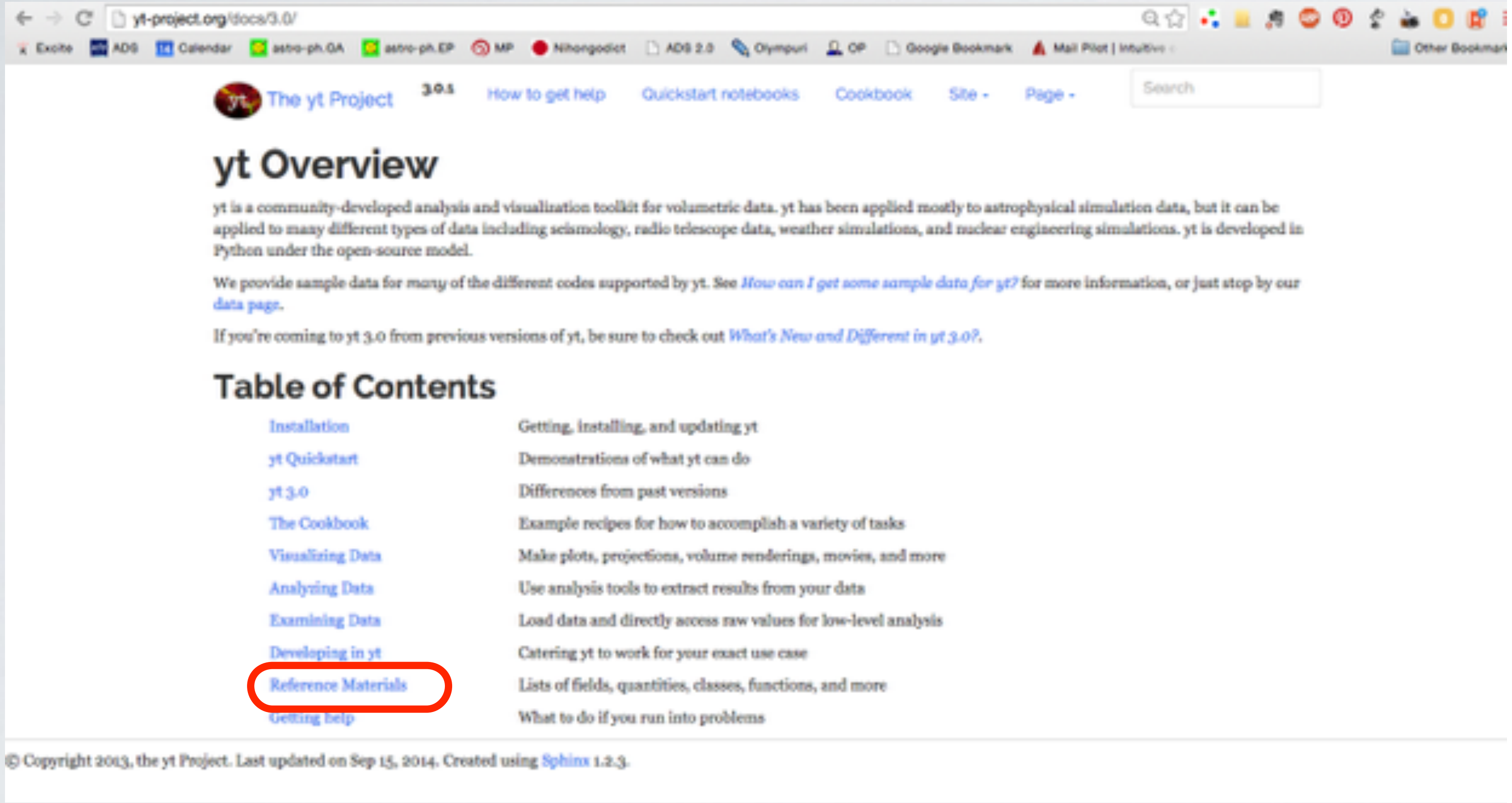
(This is a proxy for `ArrowCallback`.)

This adds an arrow pointing at `pos` with size `code_size` in code units. `plot_args` is a dict fed to matplotlib with arrow pr

```
import yt
ds = yt.load("IsolatedGalaxy/galaxy0030/galaxy0030")
slc = yt.SlicePlot(ds, 'z', 'density', width=(10,'kpc'), center='c')
slc.annotate_arrow((0.5, 0.5, 0.5), (1, 'kpc'))
slc.save()
```

The docs

Also very very useful ...



yt Overview

yt is a community-developed analysis and visualization toolkit for volumetric data. yt has been applied mostly to astrophysical simulation data, but it can be applied to many different types of data including seismology, radio telescope data, weather simulations, and nuclear engineering simulations. yt is developed in Python under the open-source model.

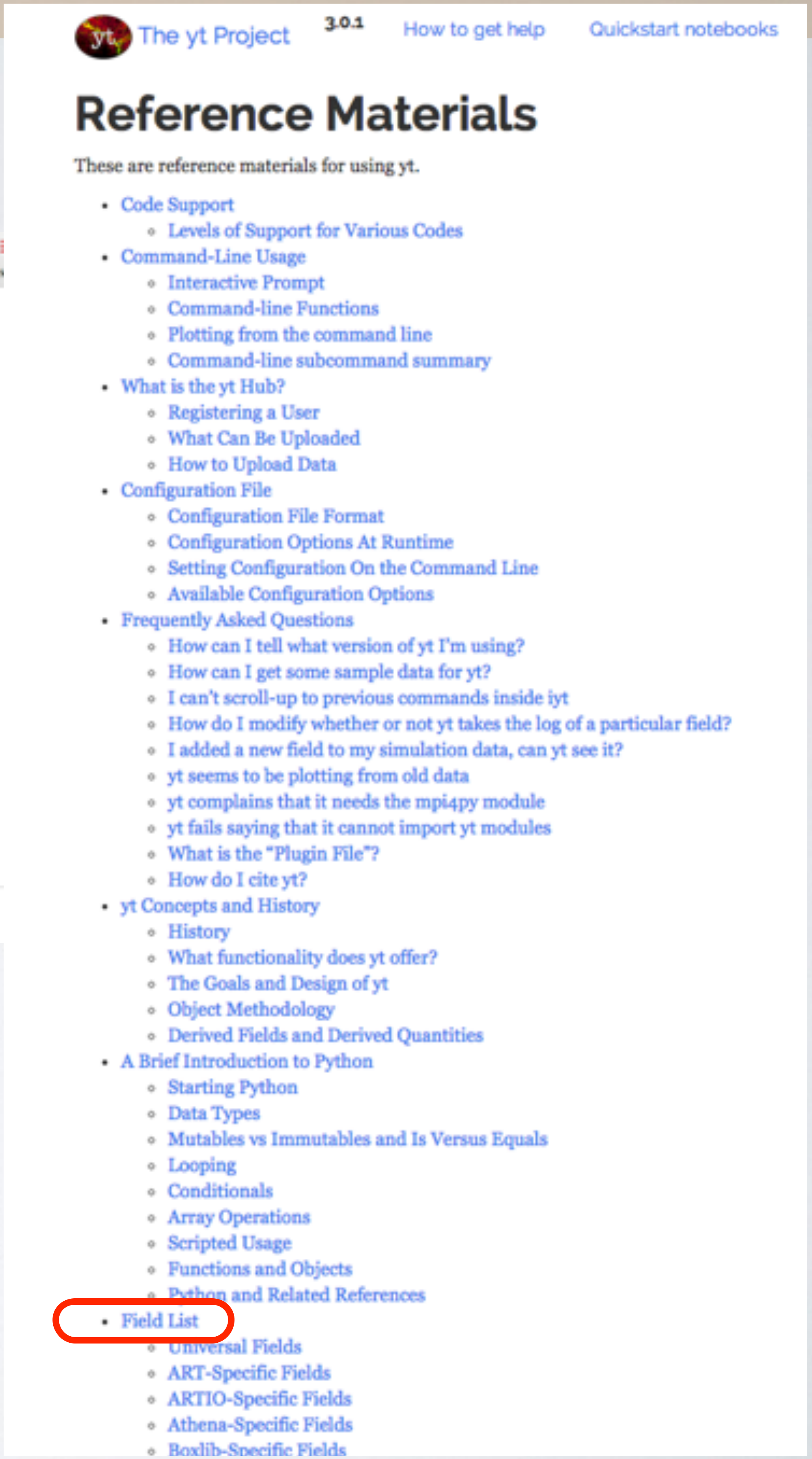
We provide sample data for *many* of the different codes supported by yt. See *How can I get some sample data for yt?* for more information, or just stop by our data page.

If you're coming to yt 3.0 from previous versions of yt, be sure to check out *What's New and Different in yt 3.0?*.

Table of Contents

Installation	Getting, installing, and updating yt
yt Quickstart	Demonstrations of what yt can do
yt 3.0	Differences from past versions
The Cookbook	Example recipes for how to accomplish a variety of tasks
Visualizing Data	Make plots, projections, volume renderings, movies, and more
Analyzing Data	Use analysis tools to extract results from your data
Examining Data	Load data and directly access raw values for low-level analysis
Developing in yt	Catering yt to work for your exact use case
Reference Materials	Lists of fields, quantities, classes, functions, and more
Getting help	What to do if you run into problems

© Copyright 2013, the yt Project. Last updated on Sep 15, 2014. Created using Sphinx 1.2.3.



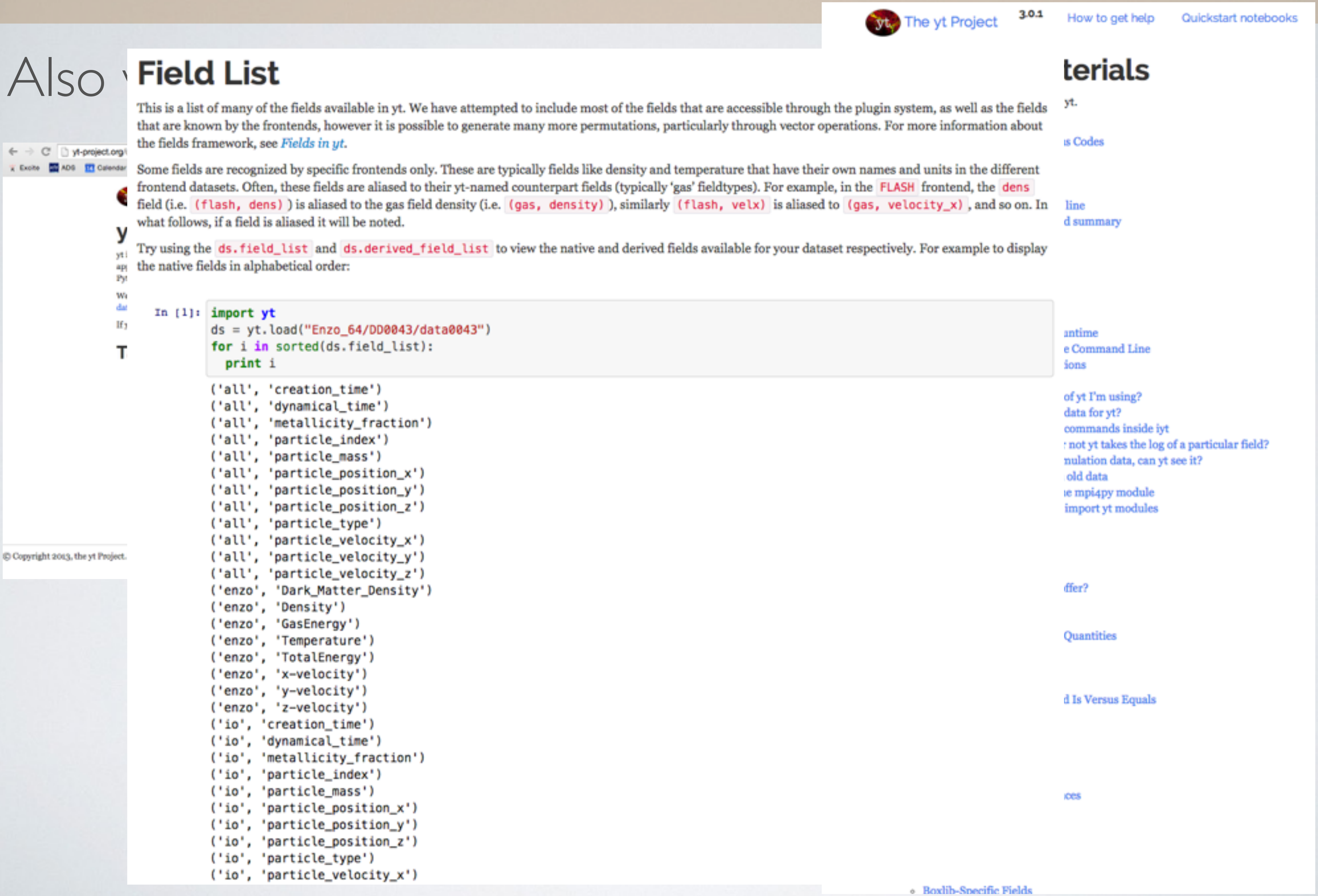
Reference Materials

These are reference materials for using yt.

- Code Support
  - Levels of Support for Various Codes
- Command-Line Usage
  - Interactive Prompt
  - Command-line Functions
  - Plotting from the command line
  - Command-line subcommand summary
- What is the yt Hub?
  - Registering a User
  - What Can Be Uploaded
  - How to Upload Data
- Configuration File
  - Configuration File Format
  - Configuration Options At Runtime
  - Setting Configuration On the Command Line
  - Available Configuration Options
- Frequently Asked Questions
  - How can I tell what version of yt I'm using?
  - How can I get some sample data for yt?
  - I can't scroll-up to previous commands inside iyt
  - How do I modify whether or not yt takes the log of a particular field?
  - I added a new field to my simulation data, can yt see it?
  - yt seems to be plotting from old data
  - yt complains that it needs the mpi4py module
  - yt fails saying that it cannot import yt modules
  - What is the "Plugin File"?
  - How do I cite yt?
- yt Concepts and History
  - History
  - What functionality does yt offer?
  - The Goals and Design of yt
  - Object Methodology
  - Derived Fields and Derived Quantities
- A Brief Introduction to Python
  - Starting Python
  - Data Types
  - Mutables vs Immutables and Is Versus Equals
  - Looping
  - Conditionals
  - Array Operations
  - Scripted Usage
  - Functions and Objects
  - Python and Related References
- Field List
  - Universal Fields
  - ART-Specific Fields
  - ARTIO-Specific Fields
  - Athena-Specific Fields
  - Boxlib-Specific Fields

The docs

Also



yt-project.org

Excite ADS Calendar

© Copyright 2013, the yt Project.

The yt Project 3.0.1 How to get help Quickstart notebooks

terials

yt.

s Codes

line
d summary

antime
e Command Line
ions

of yt I'm using?
data for yt?
commands inside iyt
not yt takes the log of a particular field?
nulation data, can yt see it?
old data
e mpi4py module
import yt modules

ffer?

Quantities

d Is Versus Equals

ces

- Boxlib-Specific Fields

Field List

This is a list of many of the fields available in yt. We have attempted to include most of the fields that are accessible through the plugin system, as well as the fields that are known by the frontends, however it is possible to generate many more permutations, particularly through vector operations. For more information about the fields framework, see *Fields in yt*.

Some fields are recognized by specific frontends only. These are typically fields like density and temperature that have their own names and units in the different frontend datasets. Often, these fields are aliased to their yt-named counterpart fields (typically 'gas' fieldtypes). For example, in the `FLASH` frontend, the `dens` field (i.e. `(flash, dens)`) is aliased to the gas field density (i.e. `(gas, density)`), similarly `(flash, velx)` is aliased to `(gas, velocity_x)`, and so on. In what follows, if a field is aliased it will be noted.

Try using the `ds.field_list` and `ds.derived_field_list` to view the native and derived fields available for your dataset respectively. For example to display the native fields in alphabetical order:

```
In [1]: import yt
        ds = yt.load("Enzo_64/DD0043/data0043")
        for i in sorted(ds.field_list):
          print i
```

```
('all', 'creation_time')
('all', 'dynamical_time')
('all', 'metallicity_fraction')
('all', 'particle_index')
('all', 'particle_mass')
('all', 'particle_position_x')
('all', 'particle_position_y')
('all', 'particle_position_z')
('all', 'particle_type')
('all', 'particle_velocity_x')
('all', 'particle_velocity_y')
('all', 'particle_velocity_z')
('enzo', 'Dark_Matter_Density')
('enzo', 'Density')
('enzo', 'GasEnergy')
('enzo', 'Temperature')
('enzo', 'TotalEnergy')
('enzo', 'x-velocity')
('enzo', 'y-velocity')
('enzo', 'z-velocity')
('io', 'creation_time')
('io', 'dynamical_time')
('io', 'metallicity_fraction')
('io', 'particle_index')
('io', 'particle_mass')
('io', 'particle_position_x')
('io', 'particle_position_y')
('io', 'particle_position_z')
('io', 'particle_type')
('io', 'particle_velocity_x')
```

Summary

- You can run **yt** by.....

the command line: very quick!

with the iPython notebook: easy to save and share
like an online lab book

scripts: great for repeating jobs
best for more complicated programmes

- Practice running examples from the docs

create: A slice
A radial profile