“epiphyte”

Joint effort:

Matt Elliott
Kieran Evans
Doug Fein
Jason Kastner
Kacper Kowalik
Dmitry Mishin
David Rogers
Ian Taylor
Nathan Tolbert
Venkat Yekkirala
Matt Turk

exploratory

open

participatory

exploratory

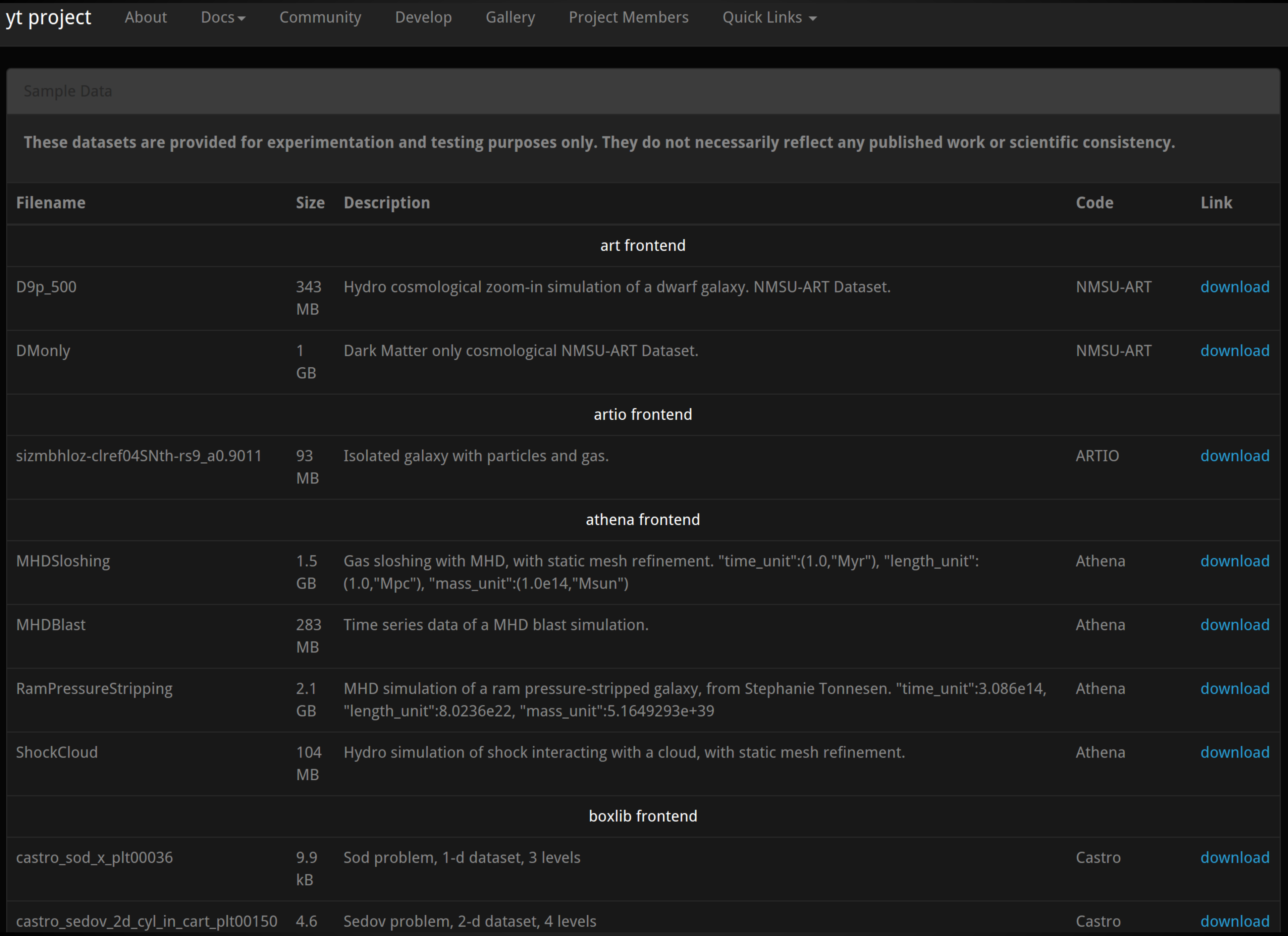
openatory

participatory

We share data at yt-project.org, and usually this means giving access via ssh keys, which is moderately annoying.

Then, this data gets downloaded and looked at locally.

We wanted to avoid all the annoying bits of this by making it easy to upload, and easy to analyze the data.



Sample Data

These datasets are provided for experimentation and testing purposes only. They do not necessarily reflect any published work or scientific consistency.

Filename	Size	Description	Code	Link
		art frontend		
D9p_500	343 MB	Hydro cosmological zoom-in simulation of a dwarf galaxy. NMSU-ART Dataset.	NMSU-ART	download
DMonly	1 GB	Dark Matter only cosmological NMSU-ART Dataset.	NMSU-ART	download
		artio frontend		
sizmbhloz-clref04SNth-rs9_a0.9011	93 MB	Isolated galaxy with particles and gas.	ARTIO	download
		athena frontend		
MHDSloshing	1.5 GB	Gas sloshing with MHD, with static mesh refinement. "time_unit":(1.0,"Myr"), "length_unit":(1.0,"Mpc"), "mass_unit":(1.0e14,"Msun")	Athena	download
MHDBlast	283 MB	Time series data of a MHD blast simulation.	Athena	download
RamPressureStripping	2.1 GB	MHD simulation of a ram pressure-stripped galaxy, from Stephanie Tonnesen. "time_unit":3.086e14, "length_unit":8.0236e22, "mass_unit":5.1649293e+39	Athena	download
ShockCloud	104 MB	Hydro simulation of shock interacting with a cloud, with static mesh refinement.	Athena	download
		boxlib frontend		
castro_sod_x_plt00036	9.9 kB	Sod problem, 1-d dataset, 3 levels	Castro	download
castro_sedov_2d_cyl_in_cart_plt00150	4.6	Sedov problem, 2-d dataset, 4 levels	Castro	download

It’s awkwardly sized: not so huge, but not emailable. And it’s bigger than Dropbox wants to give out for free. We get data from lots of people from different resources and computational access.

postulate:

for this to be worthwhile, people have to use it.

postulate:

for this to be worthwhile, people have to use it.

for people to use it, it must have zero barriers to entry.

postulate:

for this to be worthwhile, people have to use it.

for people to use it, it must have zero barriers to entry.

for people to love it, it must make hard things easy.

postulate:

for this to be worthwhile, people have to use it.

for people to use it, it must have zero barriers to entry.

for people to love it, it must make hard things easy.

we have to go where people already are.

motivation / desires

Data

motivation / desires

submit
analyze
curate
annotate
link

Use Case 1:

Large data is stored at NDS-Epiphyte, citable, and we want to make analysis accessible in a non-interactive fashion.

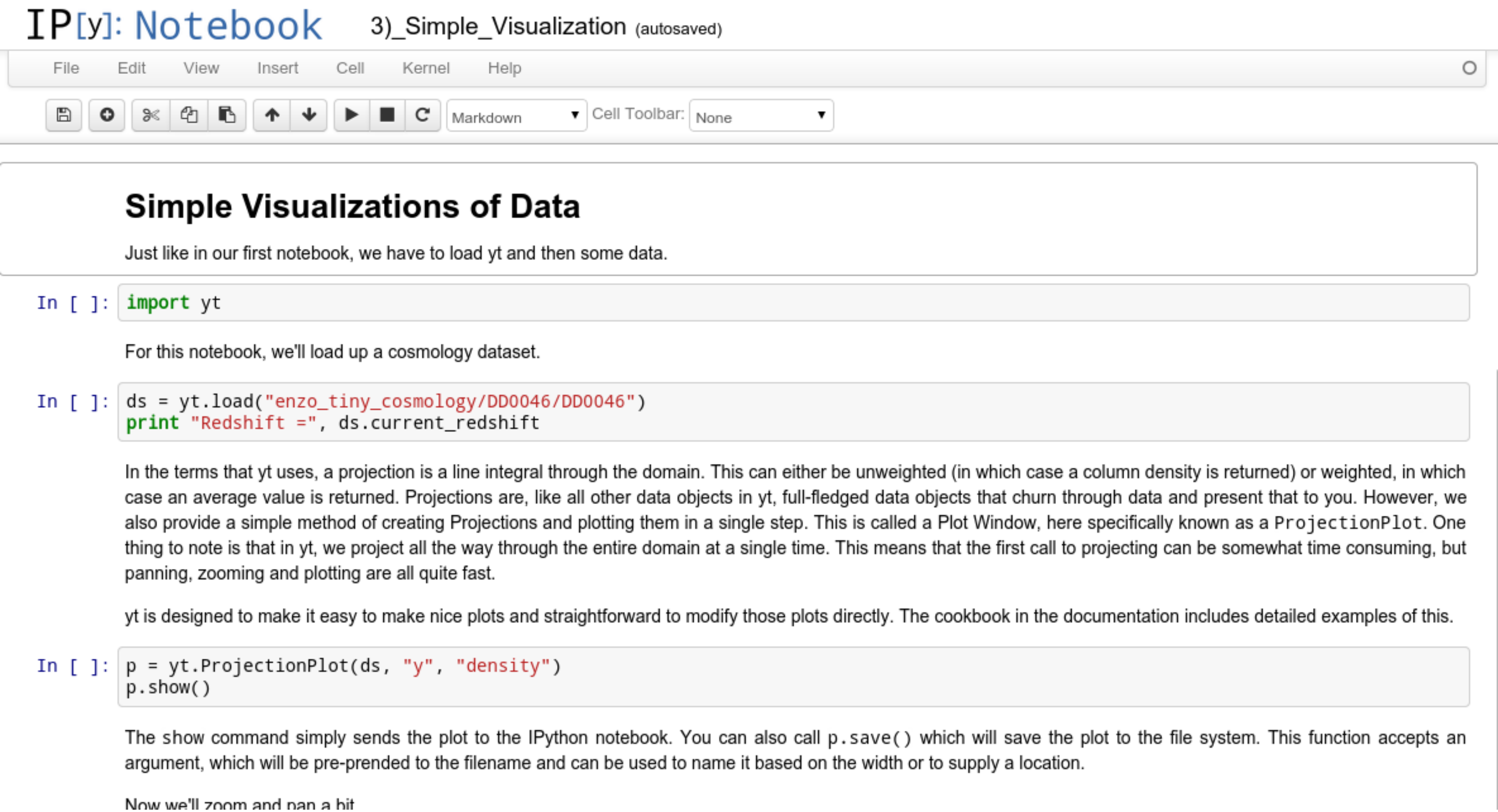
(submit a script, get a plot)

```
$ epiphyte run my_script.py
Running…
Job id: 329314821

$ eiphyte get_results 329314821
```

Use Case 2:

Data is stored at NDS-Epiphyte, and we make available interactive analysis and visualization of that data.

(IPython notebooks, RStudio, …)



Simple Visualizations of Data

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

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

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

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

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

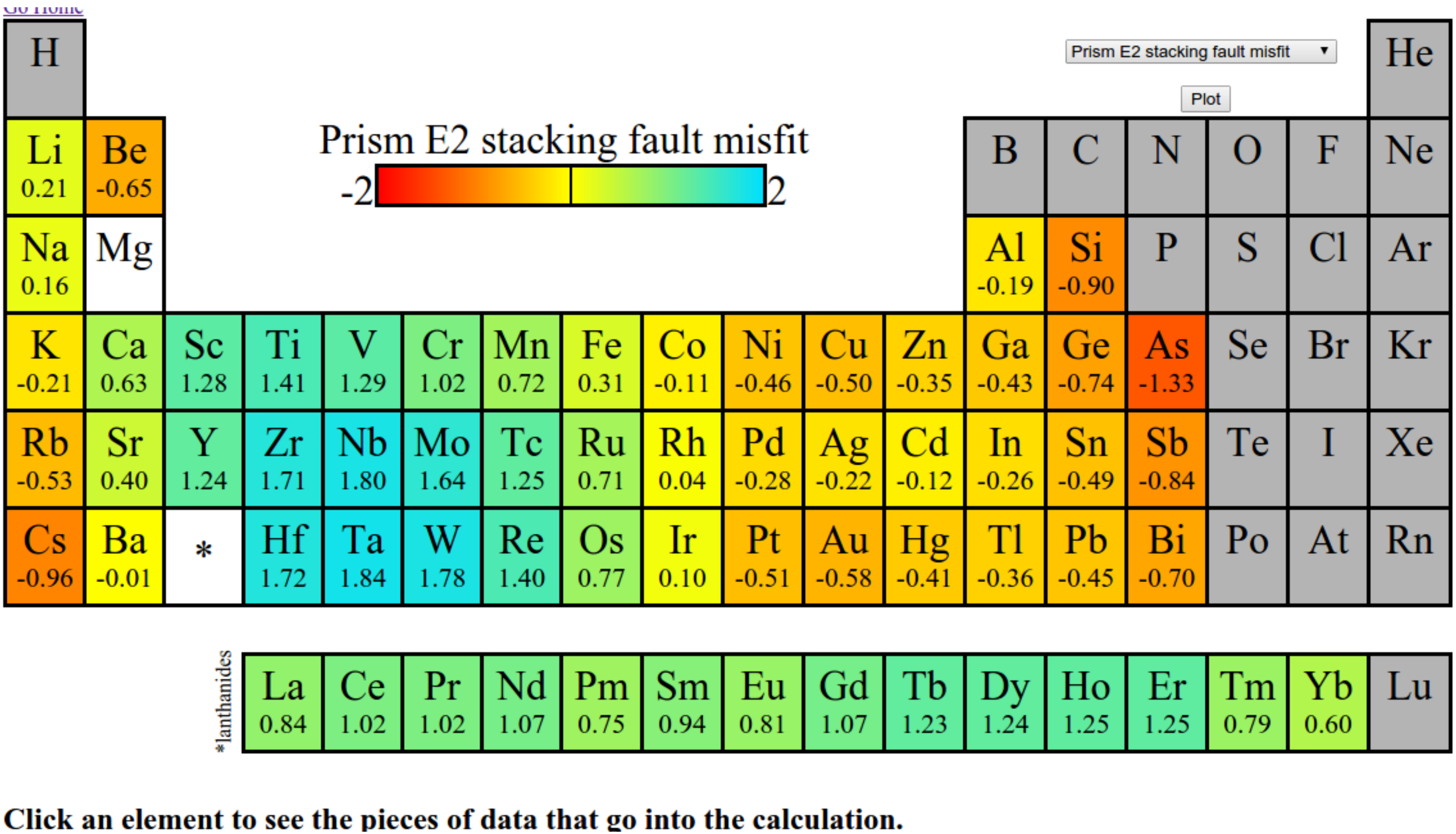
The `show` command simply sends the plot to the IPython notebook. You can also call `p.save()` which will save the plot to the file system. This function accepts an argument, which will be pre-prended to the filename and can be used to name it based on the width or to supply a location.

Now we'll zoom and pan a bit

Use Case 3:

Individual research communities store collections of data, in front of which they want to place compute-heavy webapps.

(Queries, correlations, …)



Prism E2 stacking fault misfit

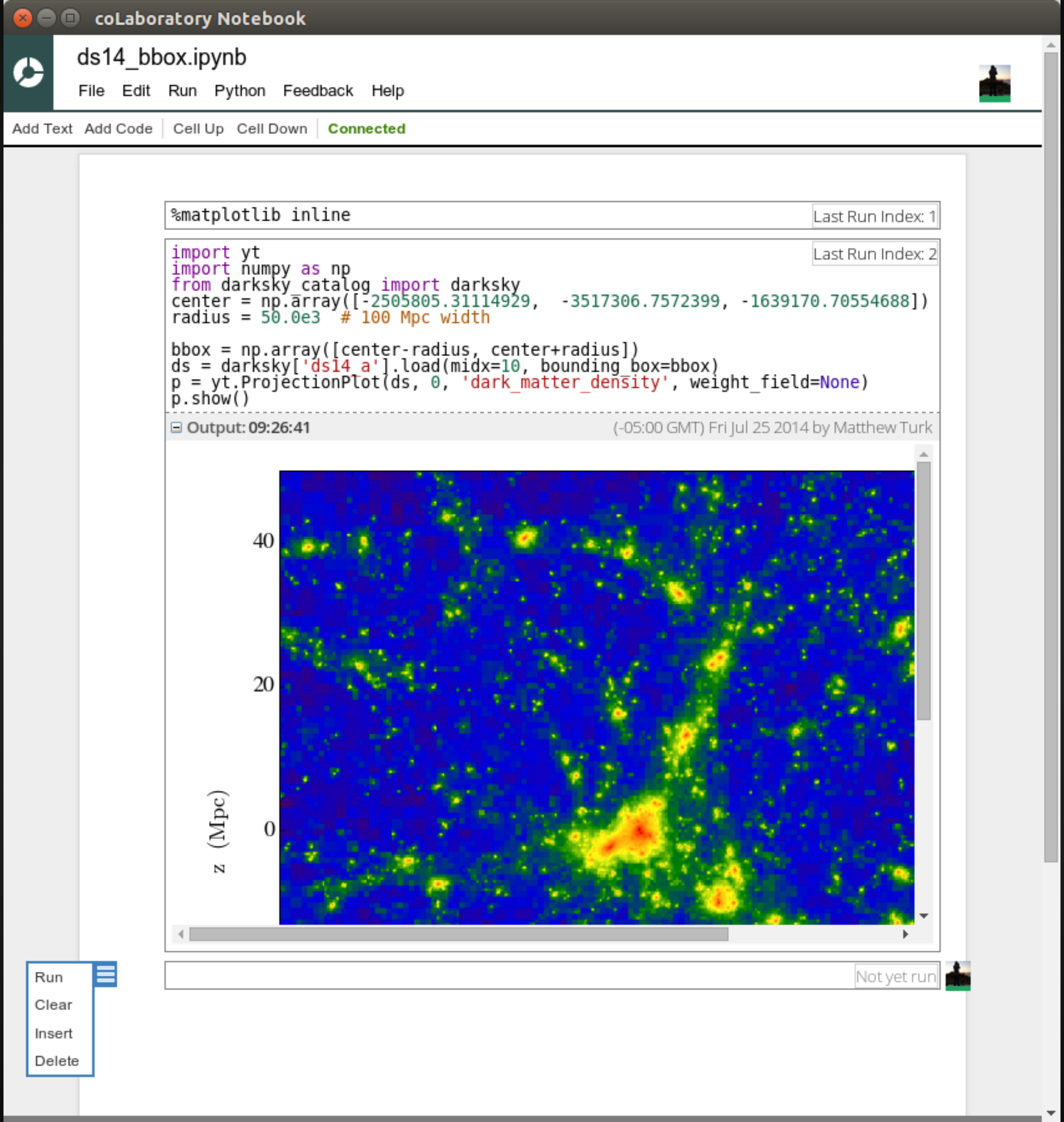
H																	He
Li 0.21	Be -0.65											B	C	N	O	F	Ne
Na 0.16	Mg											Al -0.19	Si -0.90	P	S	Cl	Ar
K -0.21	Ca 0.63	Sc 1.28	Ti 1.41	V 1.29	Cr 1.02	Mn 0.72	Fe 0.31	Co -0.11	Ni -0.46	Cu -0.50	Zn -0.35	Ga -0.43	Ge -0.74	As -1.33	Se	Br	Kr
Rb -0.53	Sr 0.40	Y 1.24	Zr 1.71	Nb 1.80	Mo 1.64	Tc 1.25	Ru 0.71	Rh 0.04	Pd -0.28	Ag -0.22	Cd -0.12	In -0.26	Sn -0.49	Sb -0.84	Te	I	Xe
Cs -0.96	Ba -0.01	*	Hf 1.72	Ta 1.84	W 1.78	Re 1.40	Os 0.77	Ir 0.10	Pt -0.51	Au -0.58	Hg -0.41	Tl -0.36	Pb -0.45	Bi -0.70	Po	At	Rn
		*lanthanides	La 0.84	Ce 1.02	Pr 1.02	Nd 1.07	Pm 0.75	Sm 0.94	Eu 0.81	Gd 1.07	Tb 1.23	Dy 1.24	Ho 1.25	Er 1.25	Tm 0.79	Yb 0.60	Lu

Click an element to see the pieces of data that go into the calculation.

Use Case 4:

Interactive, collaborative development of analysis and visualization of data, deployed on NDS-Epiphyte resources.

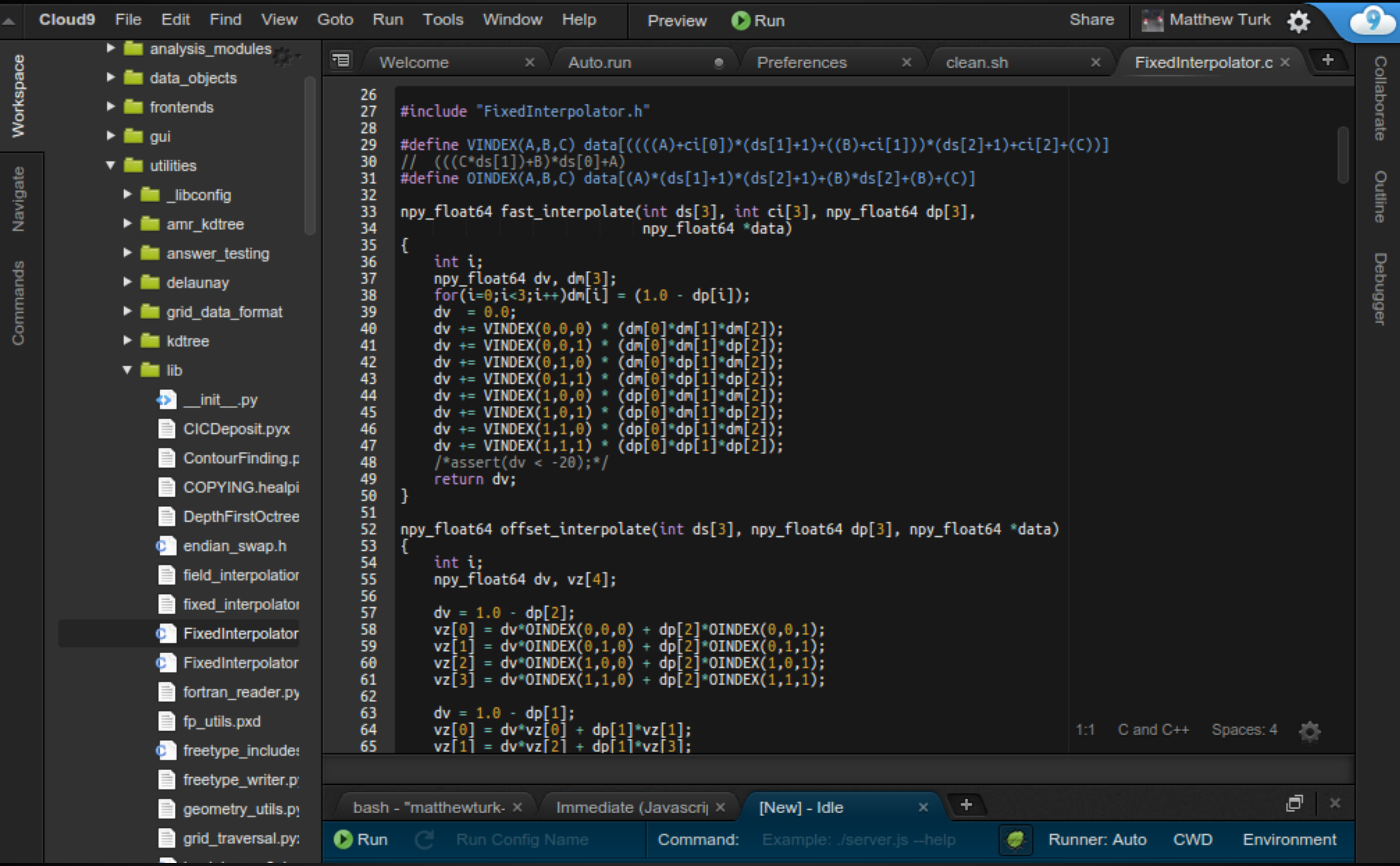
(coLaboratory, Cloud9 IDE, ...)



```
%matplotlib inline
```

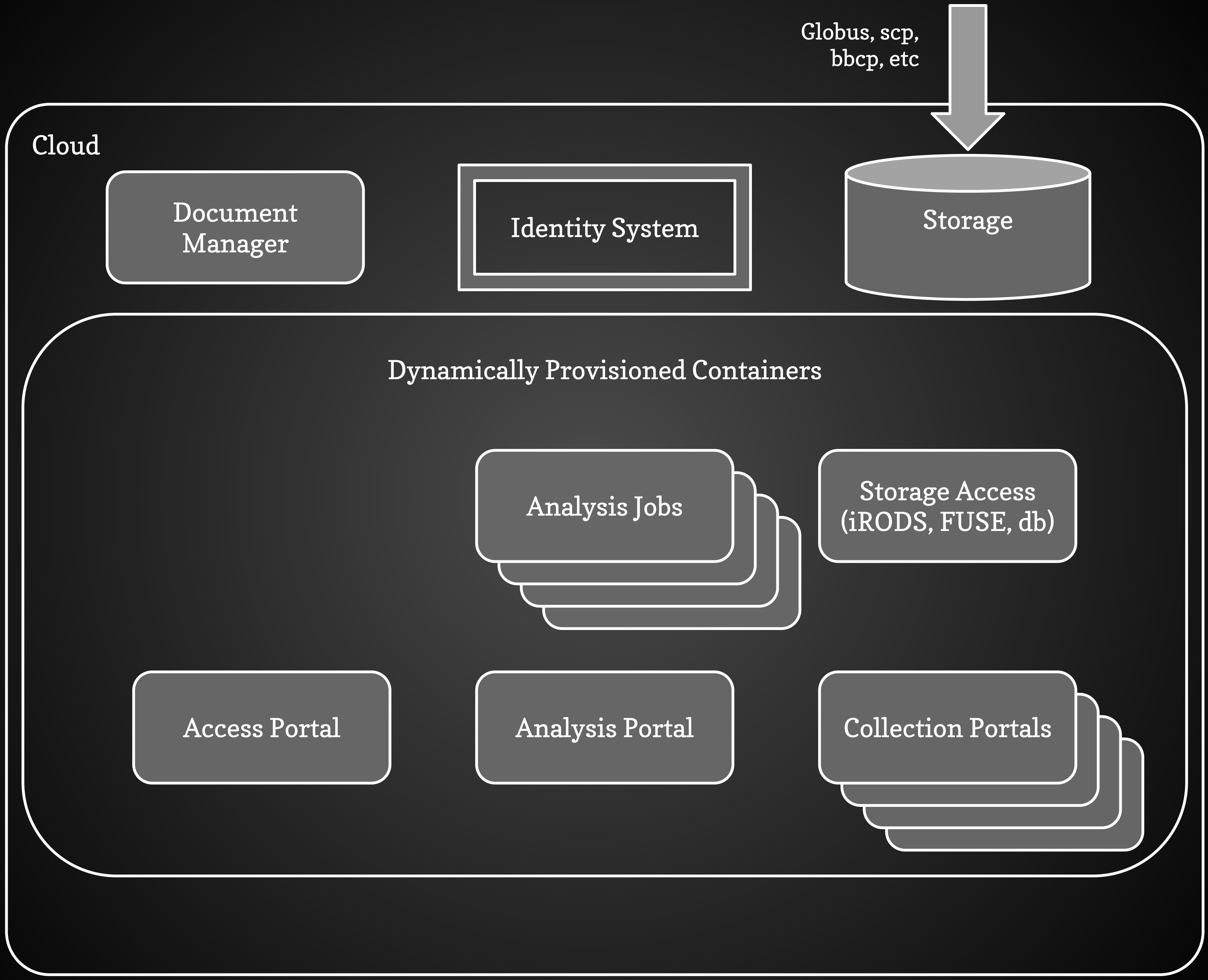
```
import yt
import numpy as np
from darksky_catalog import darksky
center = np.array([-2505805.31114929,  -3517306.7572399, -1639170.70554688])
radius = 50.0e3  # 100 Mpc width

bbox = np.array([center-radius, center+radius])
ds = darksky['ds14_a'].load(midx=10, bounding_box=bbox)
p = yt.ProjectionPlot(ds, 0, 'dark_matter_density', weight_field=None)
p.show()
```

⊟ Output: 09:26:41 (-05:00 GMT) Fri Jul 25 2014 by Matthew Turk



```c

#include "FixedInterpolator.h"

#define VINDEX(A,B,C) data[((((A)+ci[0])*(ds[1]+1)+((B)+ci[1]))*(ds[2]+1)+ci[2]+(C))]
//  (((C*ds[1])+B)*ds[0]+A)
#define OINDEX(A,B,C) data[(A)*(ds[1]+1)*(ds[2]+1)+(B)*ds[2]+(B)+(C)]

npy_float64 fast_interpolate(int ds[3], int ci[3], npy_float64 dp[3],
                             npy_float64 *data)
{
    int i;
    npy_float64 dv, dm[3];
    for(i=0;i<3;i++)dm[i] = (1.0 - dp[i]);
    dv  = 0.0;
    dv += VINDEX(0,0,0) * (dm[0]*dm[1]*dm[2]);
    dv += VINDEX(0,0,1) * (dm[0]*dm[1]*dp[2]);
    dv += VINDEX(0,1,0) * (dm[0]*dp[1]*dm[2]);
    dv += VINDEX(0,1,1) * (dm[0]*dp[1]*dp[2]);
    dv += VINDEX(1,0,0) * (dp[0]*dm[1]*dm[2]);
    dv += VINDEX(1,0,1) * (dp[0]*dm[1]*dp[2]);
    dv += VINDEX(1,1,0) * (dp[0]*dp[1]*dm[2]);
    dv += VINDEX(1,1,1) * (dp[0]*dp[1]*dp[2]);
    /*assert(dv < -20);*/
    return dv;
}

npy_float64 offset_interpolate(int ds[3], npy_float64 dp[3], npy_float64 *data)
{
    int i;
    npy_float64 dv, vz[4];

    dv = 1.0 - dp[2];
    vz[0] = dv*OINDEX(0,0,0) + dp[2]*OINDEX(0,0,1);
    vz[1] = dv*OINDEX(0,1,0) + dp[2]*OINDEX(0,1,1);
    vz[2] = dv*OINDEX(1,0,0) + dp[2]*OINDEX(1,0,1);
    vz[3] = dv*OINDEX(1,1,0) + dp[2]*OINDEX(1,1,1);

    dv = 1.0 - dp[1];
    vz[0] = dv*vz[0] + dp[1]*vz[1];
    vz[1] = dv*vz[2] + dp[1]*vz[3];
```

- Non-interactive
- Interactive
- Outward-facing
- Collaborative


Globus, scp,
bbcp, etc
Cloud
Document Manager
Identity System
Storage
Dynamically Provisioned Containers
Analysis Jobs
Storage Access (iRODS, FUSE, db)
Access Portal
Analysis Portal
Collection Portals

requirements:

swappable components

requirements:

swappable components

open source, open development

requirements:

swappable components

open source, open development

standards for interoperability (OAI-ORE)

Inspired by NDS1, we decided to explore a Platform-as-a-Service for data analysis.

What could we do with a service oriented architecture and sufficient motivation?

Humility.

Humility.

We must evaluate existing technologies, utilize them where appropriate, and remain flexible enough to allow future inclusion and utilization.

Humility.

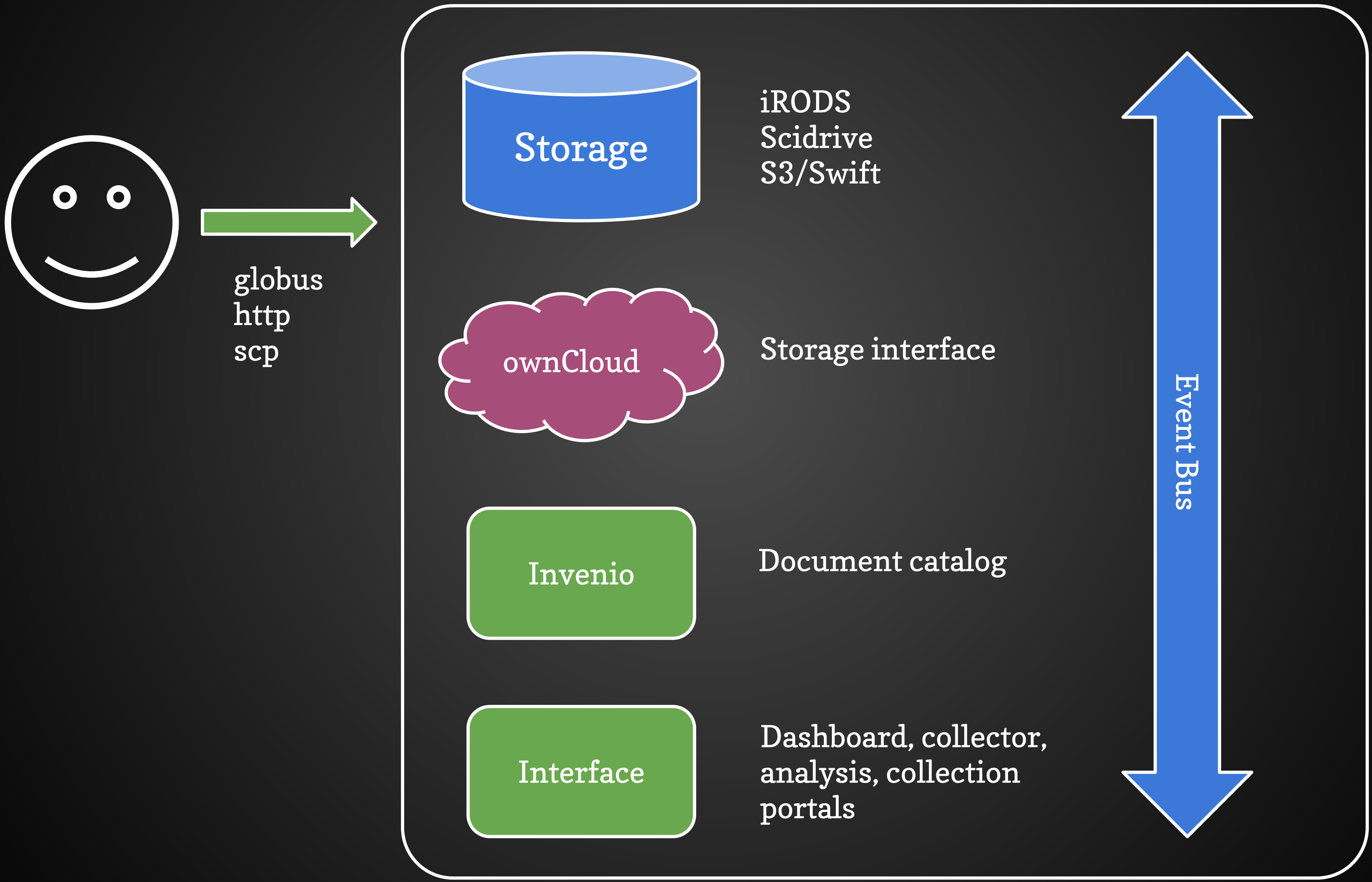
We must evaluate existing technologies, utilize them where appropriate, and remain flexible enough to allow future inclusion and utilization.

Dataverse, Docker, DSpace, Globus Nexus, InCommon, Invenio, iPlant, iRODS, Medici2, mesos, Open Science Framework, ownCloud, ResearchCompendia, SciDrive, SEAD, many many more...

(demo time)

A Near-Term Vision:

Storage and documents are easily federated.



globus
http
scp

Storage

iRODS
Scidrive
S3/Swift

ownCloud

Storage interface

Invenio

Document catalog

Interface

Dashboard, collector, analysis, collection portals

Event Bus

Interface

Dashboard, collector, analysis, collection portals

- Non-interactive
- Interactive
- Outward-facing
- Collaborative

Interested?

discuss@nationaldataservice.org
http://bitbucket.org/nds-org/
#nds-epiphyte / chat.freenode.net