




# Ascent: Flyweight In Situ Visualization and Analysis for HPC Simulations

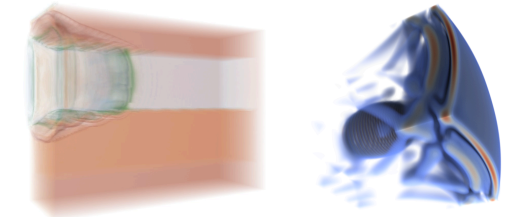
SC20 Tutorial

Matt Larsen (LLNL), Cyrus Harrison (LLNL), Hank Childs (Univ of Oregon)

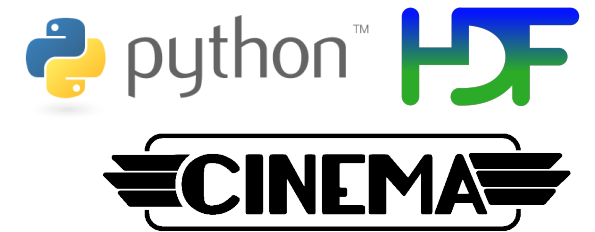


# Ascent is an easy to use flyweight in situ visualization and analysis library for HPC simulations

- **Easy to use in-memory visualization and analysis**
  - Use cases: *Making Pictures*, *Transforming Data*, and *Capturing Data*
  - Young effort, yet already supports most common visualization operations
  - Provides a simple infrastructure to integrate custom analysis
  - Provides C++, C, Python, and Fortran APIs
- **Uses a flyweight design targeted at next-generation HPC platforms**
  - Efficient distributed-memory (MPI) and many-core (CUDA or OpenMP) execution
    - Demonstrated scaling: In situ filtering and ray tracing across **16,384 GPUs** on LLNL's Sierra Cluster
  - Has lower memory requirements than current tools
  - Requires less dependencies than current tools (ex: no OpenGL)
    - Builds with  Spack <https://spack.io/>



Visualizations created using Ascent



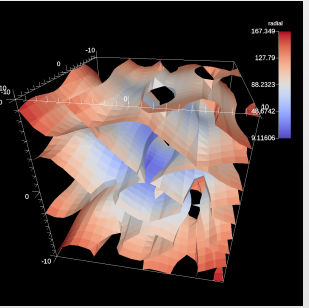
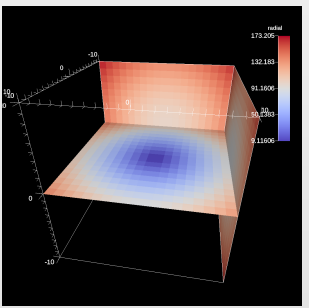
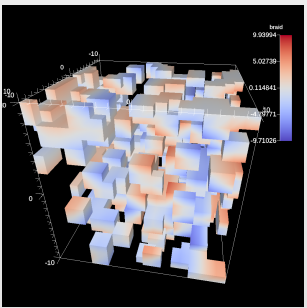
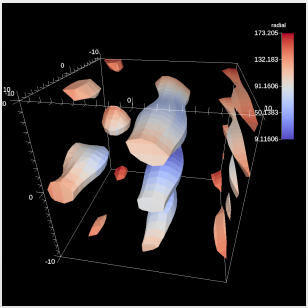
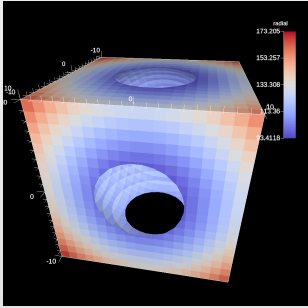
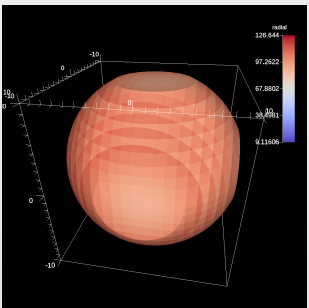
Extracts supported by Ascent

<http://ascent-dav.org>

<https://github.com/Alpine-DAV/ascent>

Website and GitHub Repo

# Ascent is ready for common visualization use cases

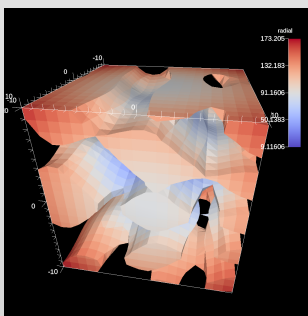
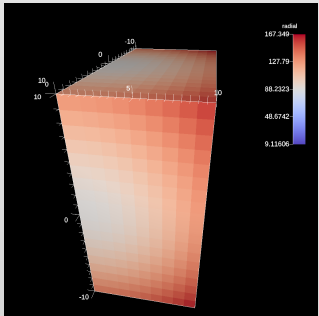


Iso-Volume

Threshold

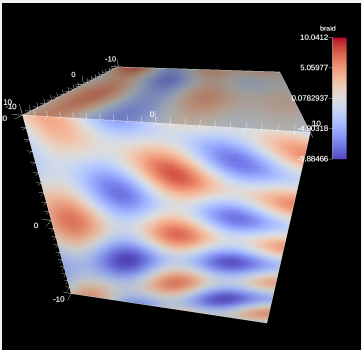
Slice

Contour

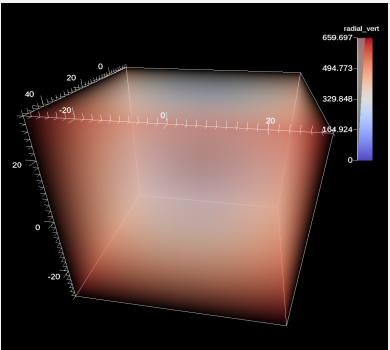


Clips

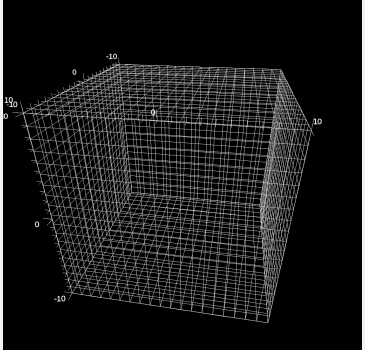
## Rendering



Pseudocolor



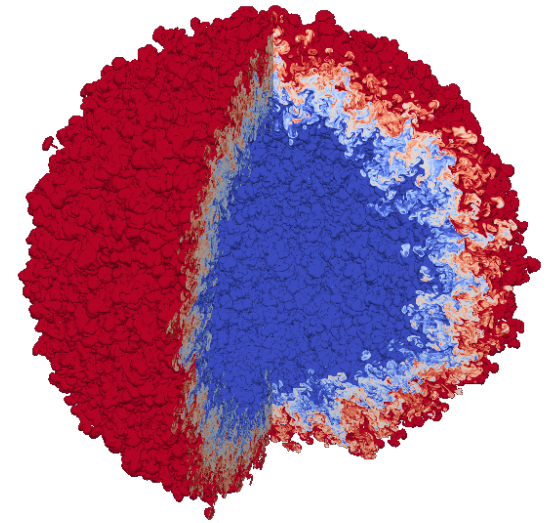
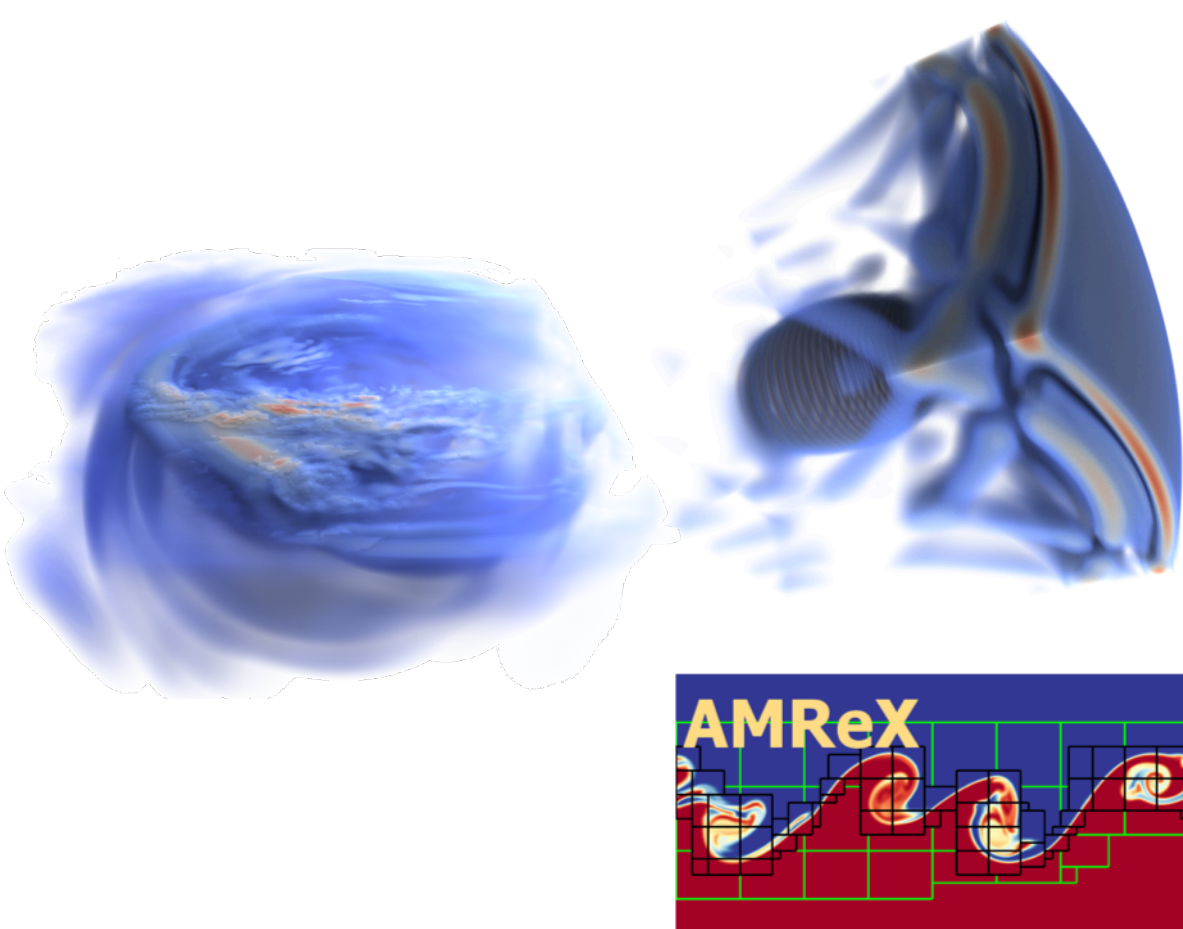
Volume



Mesh



# We are working to integrate and deploy Ascent with HPC simulation codes (ECP and beyond)



## Ascent in situ rendering at Scale:

Visualization of an idealized Inertial Confinement Fusion (ICF) simulation of Rayleigh-Taylor instability with two fluids mixing in a spherical geometry.



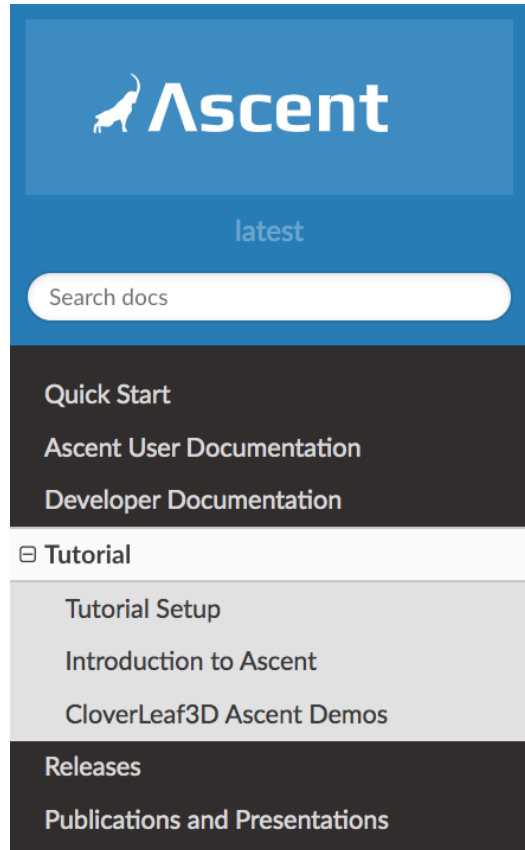
# Today we will teach you about Ascent's API and capabilities

---

## You will learn:

- How to use Conduit, the foundation of Ascent's API
- How to get your simulation data into Ascent
- How to tell Ascent what pictures to render and what analysis to execute

# Ascent tutorial examples are outlined in our documentation and included ready to run in Ascent installs



[Docs](#) » [Tutorial](#)

[Edit on GitHub](#)

## Tutorial

This tutorial introduces how to use Ascent, including basics about:

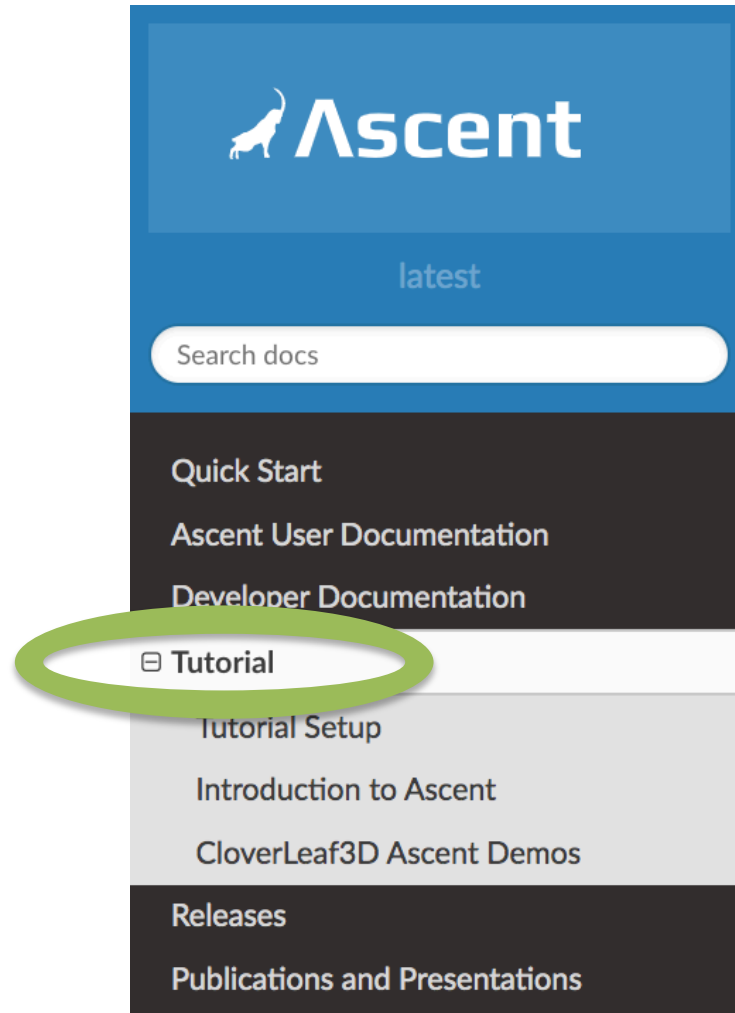
- Formating mesh data for Ascent
- Using Conduit and Ascent's Conduit-based API
- Using and combining Ascent's core building blocks: Scenes, Pipelines, Extracts, Queries, and Triggers
- Using Ascent with the Cloverleaf3D example integration

Ascent installs include standalone C++, Python, and Python-based Jupyter notebook examples for this tutorial. You can find the tutorial source code and notebooks in your Ascent install directory under `examples/ascent/tutorial/ascent_intro/` and the Cloverleaf3D demo files under `examples/ascent/tutorial/cloverleaf_demos/`.

<http://ascent-dav.org>

# Ascent tutorial examples are outlined in our documentation and included ready to run in Ascent installs

- <http://ascent-dav.org>
- Click on “Tutorial”





# Ascent's interface provides four top-level functions

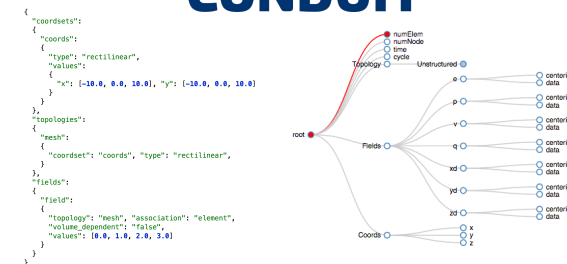
- **open() / close()**
  - Initialize and finalize an Ascent instance
- **publish()**
  - Pass your simulation data to Ascent
- **execute()**
  - Tell Ascent what to do

```
//  
// Run Ascent  
//  
  
Ascent ascent;  
ascent.open();  
ascent.publish(data);  
ascent.execute(actions);  
ascent.close();
```

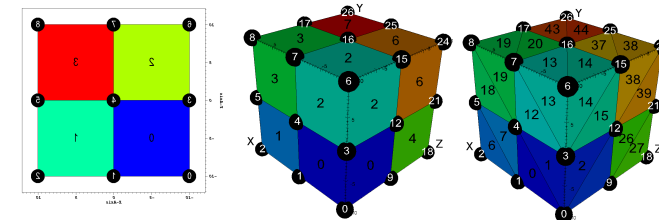
The *publish()* and *execute()* methods take a Conduit tree as an argument.  
What is a Conduit tree?

# Conduit provides intuitive APIs for in-memory data description and exchange

- **Provides an intuitive API for in-memory data description**
  - Enables *human-friendly* hierarchical data organization
  - Can describe in-memory arrays without copying
  - Provides C++, C, Python, and Fortran APIs
- **Provides common conventions for exchanging complex data**
  - Shared conventions for passing complex data (e.g. *Simulation Meshes*) enable modular interfaces across software libraries and simulation applications
- **Provides easy to use I/O interfaces for moving and storing data**
  - Enables use cases like binary checkpoint restart
  - Supports moving complex data with MPI (serialization)



Hierarchical in-memory data description



Conventions for sharing in-memory mesh data

<http://software.llnl.gov/conduit>  
<http://github.com/llnl/conduit>

Website and GitHub Repo

# Ascent uses Conduit to provide a flexible and extendable API

- Conduit underpins Ascent's support for C++, C, Python, and Fortran interfaces
- Conduit also enables using YAML to specify Ascent actions
- Conduit's zero-copy features help couple existing simulation data structures
- Conduit Blueprint provides a standard for how to present simulation meshes

Learning Ascent equates to learning how to construct and pass Conduit trees that encode your data and your expectations.



# To start, let's look at the Ascent “First Light” Example in C++

- [https://ascent.readthedocs.io/en/latest/Tutorial Intro First Light.html](https://ascent.readthedocs.io/en/latest/Tutorial%20Intro%20First%20Light.html)

```
#include <iostream>

#include "ascent.hpp"
#include "conduit_blueprint.hpp"

using namespace ascent;
using namespace conduit;

int main(int argc, char **argv)
{
    // echo info about how ascent was configured
    std::cout << ascent::about() << std::endl;

    // create conduit node with an example mesh using
    // conduit blueprint's braid function
    // ref: https://llnl-conduit.readthedocs.io/en/latest/blueprint_mesh.html#braid

    // things to explore:
    //  changing the mesh resolution

    Node mesh;
    conduit::blueprint::mesh::examples::braid("hexs",
                                              50,
                                              50,
                                              50,
                                              mesh);
```

This code generates an example mesh

# To start, let's look at the Ascent “First Light” Example in C++

- [https://ascent.readthedocs.io/en/latest/Tutorial Intro First Light.html](https://ascent.readthedocs.io/en/latest/Tutorial%20Intro%20First%20Light.html)

```
// create an Ascent instance  
Ascent a;
```

```
// open ascent  
a.open();
```

```
// publish mesh data to ascent  
a.publish(mesh);
```

Create an Ascent instance and set it up

Now Ascent has access to our mesh data

# To start, let's look at the Ascent “First Light” Example in C++

- [https://ascent.readthedocs.io/en/latest/Tutorial Intro First Light.html](https://ascent.readthedocs.io/en/latest/Tutorial%20Intro%20First%20Light.html)

```
//  
// Ascent's interface accepts "actions"  
// that to tell Ascent what to execute  
//  
Node actions;  
Node &add_act = actions.append();  
add_act["action"] = "add_scenes";  
  
// Create an action that tells Ascent to:  
// add a scene (s1) with one plot (p1)  
// that will render a pseudocolor of  
// the mesh field `braid`  
Node & scenes = add_act["scenes"];  
  
// things to explore:  
// changing plot type (mesh)  
// changing field name (for this dataset: radial)  
scenes["s1/plots/p1/type"] = "pseudocolor";  
scenes["s1/plots/p1/field"] = "braid";  
// set the output file name (ascent will add ".png")  
scenes["s1/image_name"] = "out_first_light_render_3d";
```

```
// view our full actions tree  
std::cout << actions.to_yaml() << std::endl;
```

Create a tree that describes the actions we want Ascent to do

```
-  
  action: "add_scenes"  
  scenes:  
    s1:  
      plots:  
        p1:  
          type: "pseudocolor"  
          field: "braid"  
          image_name: "out_first_light_render_3d"
```

Equivalent YAML Description

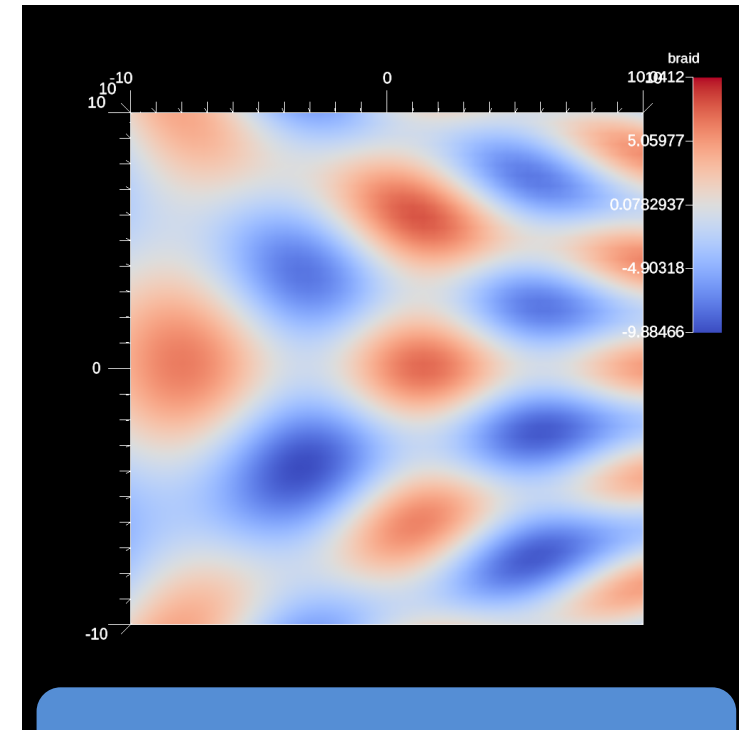


# To start, let's look at the Ascent “First Light” Example in C++

- [https://ascent.readthedocs.io/en/latest/Tutorial Intro First Light.html](https://ascent.readthedocs.io/en/latest/Tutorial%20Intro%20First%20Light.html)

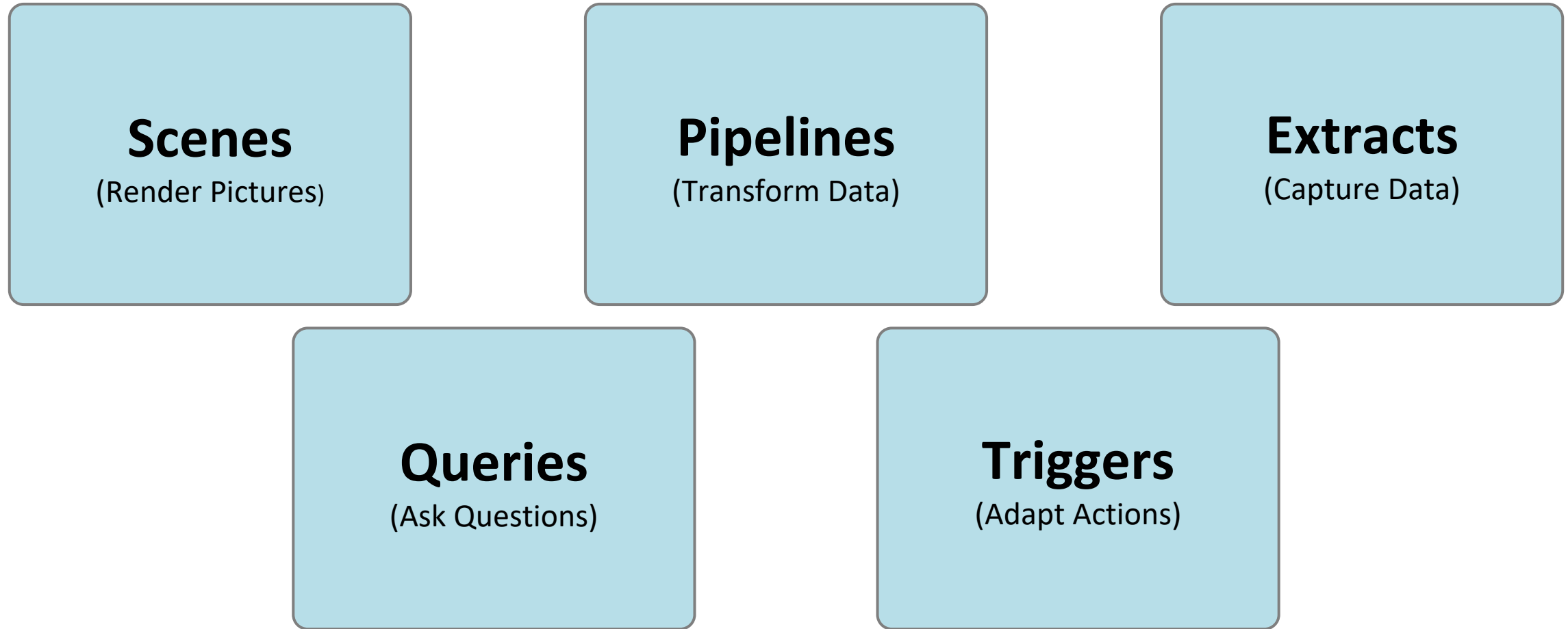
```
// execute the actions  
a.execute(actions);
```

Tell Ascent to execute these actions



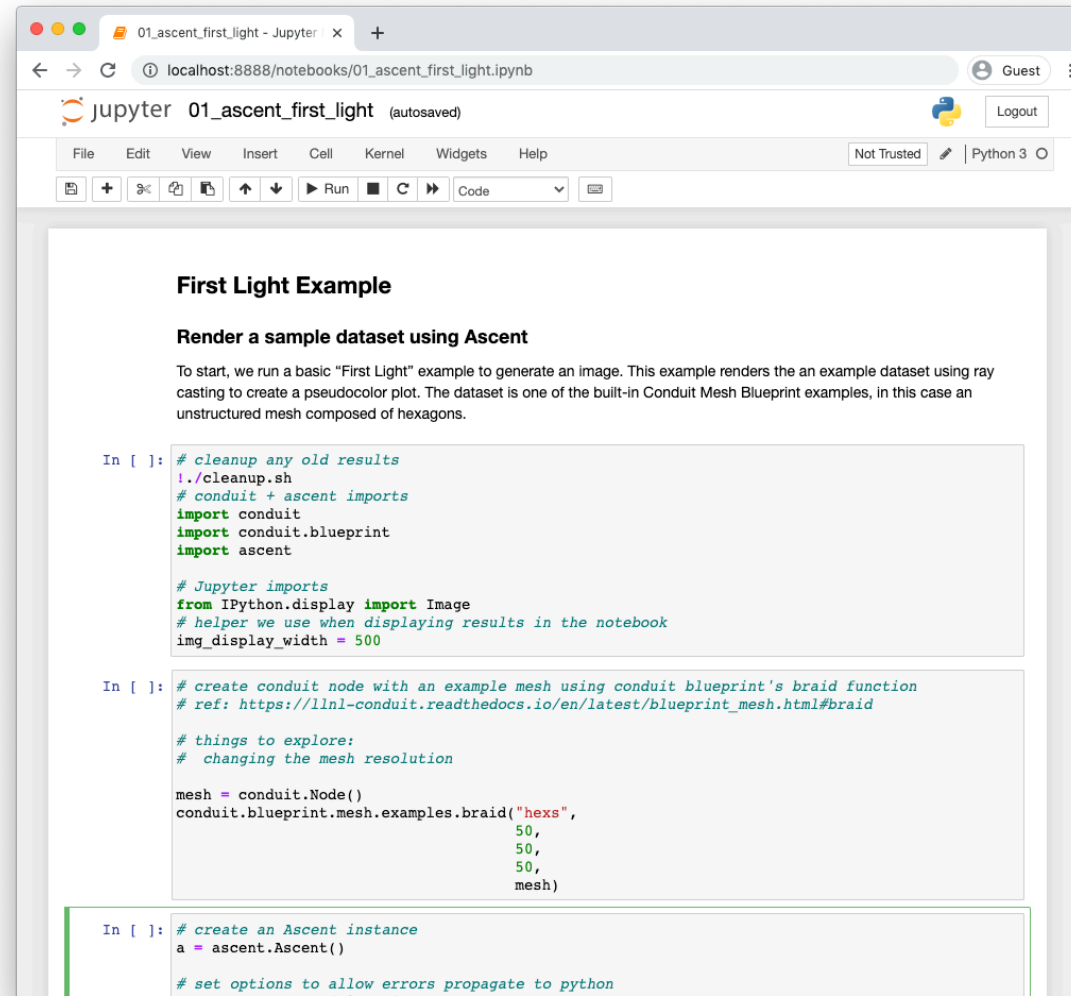
Rendered Result!

# Ascent's interface provides five composable building blocks



The tutorial provides examples for all of these.

# For the remainder of the tutorial, we will run the Ascent Tutorial examples using Jupyter Notebooks



The screenshot shows a Jupyter Notebook interface in a web browser. The browser tab is titled '01\_ascent\_first\_light - Jupyter'. The address bar shows 'localhost:8888/notebooks/01\_ascent\_first\_light.ipynb'. The Jupyter interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations and execution. The notebook content is titled 'First Light Example' and includes a sub-header 'Render a sample dataset using Ascent'. Below the sub-header is a paragraph explaining the goal: 'To start, we run a basic "First Light" example to generate an image. This example renders the an example dataset using ray casting to create a pseudocolor plot. The dataset is one of the built-in Conduit Mesh Blueprint examples, in this case an unstructured mesh composed of hexagons.' The notebook contains three code cells. The first cell is a setup block with comments and imports. The second cell creates a Conduit mesh using the 'braid' function. The third cell creates an Ascent instance and sets options.

```
In [ ]: # cleanup any old results
        !./cleanup.sh
        # conduit + ascent imports
        import conduit
        import conduit.blueprint
        import ascent

        # Jupyter imports
        from IPython.display import Image
        # helper we use when displaying results in the notebook
        img_display_width = 500

In [ ]: # create conduit node with an example mesh using conduit blueprint's braid function
        # ref: https://llnl-conduit.readthedocs.io/en/latest/blueprint_mesh.html#braid

        # things to explore:
        # changing the mesh resolution

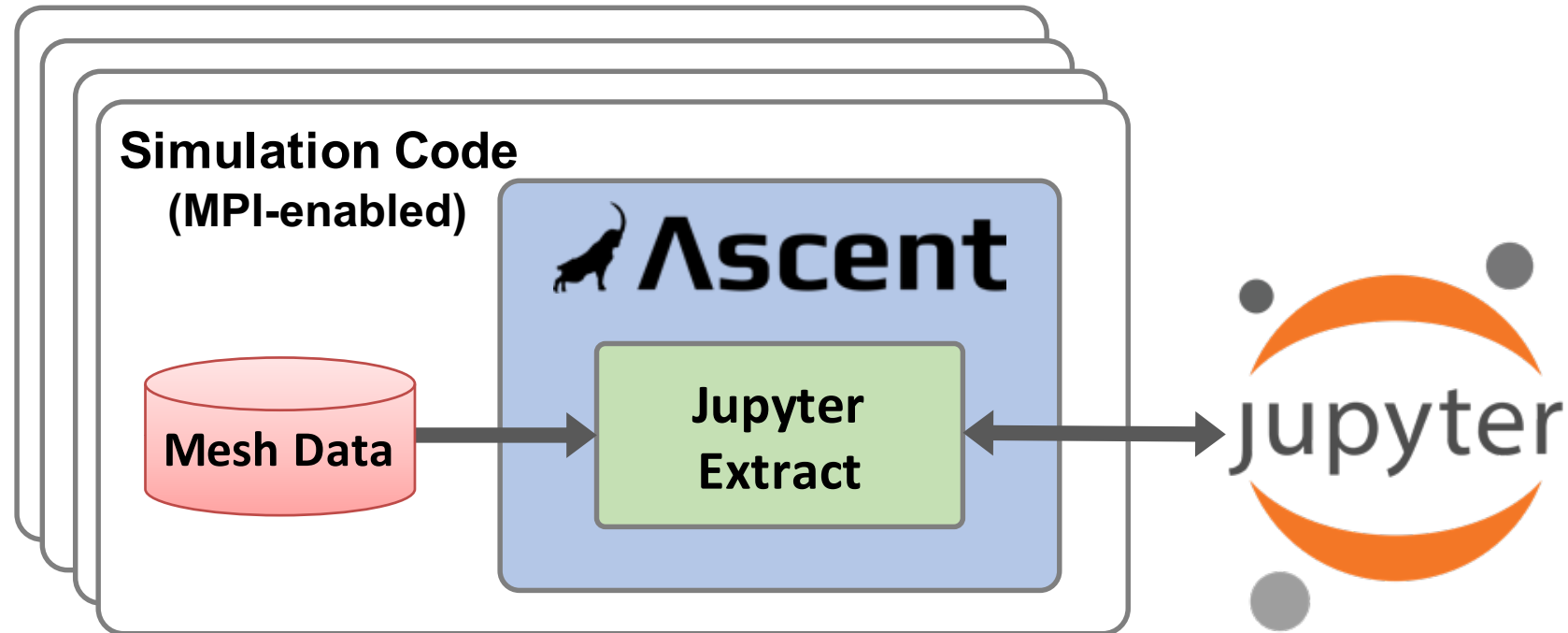
        mesh = conduit.Node()
        conduit.blueprint.mesh.examples.braid("hexs",
                                              50,
                                              50,
                                              50,
                                              mesh)

In [ ]: # create an Ascent instance
        a = ascent.Ascent()

        # set options to allow errors propagate to python
        ascent.opts = conduit.Node()
```



# Ascent's Jupyter Extract provides a path to connect your simulation to a Jupyter Notebook



With the *Jupyter Extract*, users of any simulation code with Ascent integrated can run Jupyter Notebooks and use Python to interact with in-memory data.

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344.  
Lawrence Livermore National Security, LLC

This research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy's Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation's exascale computing imperative.



**Disclaimer**

This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.