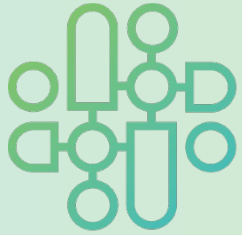


SC23
Denver, CO | i am hpc.

In Situ Analysis and Visualization with Ascent and ParaView Catalyst

[Ascent Hands-on]

SC23 Tutorial
Monday November 13th, 2023

Cyrus Harrison, Lawrence Livermore National Laboratory (LLNL)

Nicole Marsaglia, Lawrence Livermore National Laboratory (LLNL)

SC23
Denver, CO | i am hpc.

Ascent Hands-on Session

SC23 Tutorial

Monday November 13th, 2023

Cyrus Harrison (LLNL)
Nicole Marsaglia (LLNL)

LLNL-PRES-856413
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

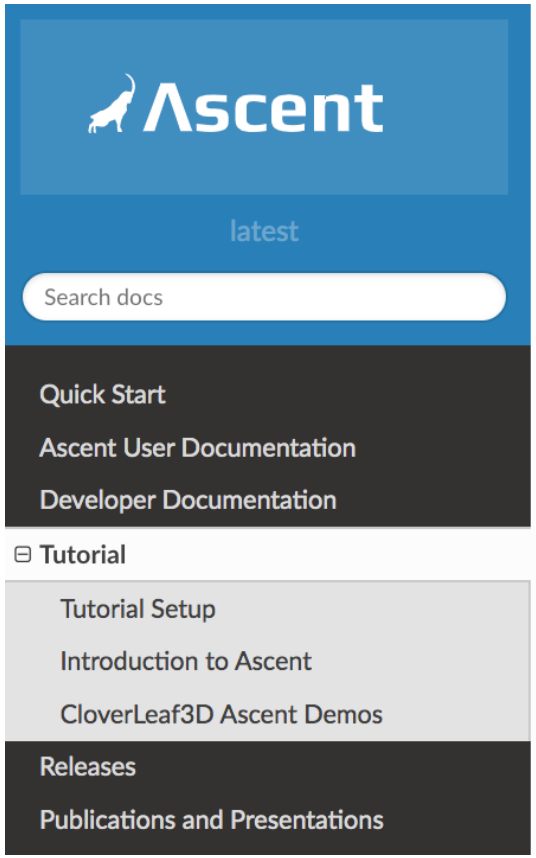
Lawrence Livermore National Laboratory

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:

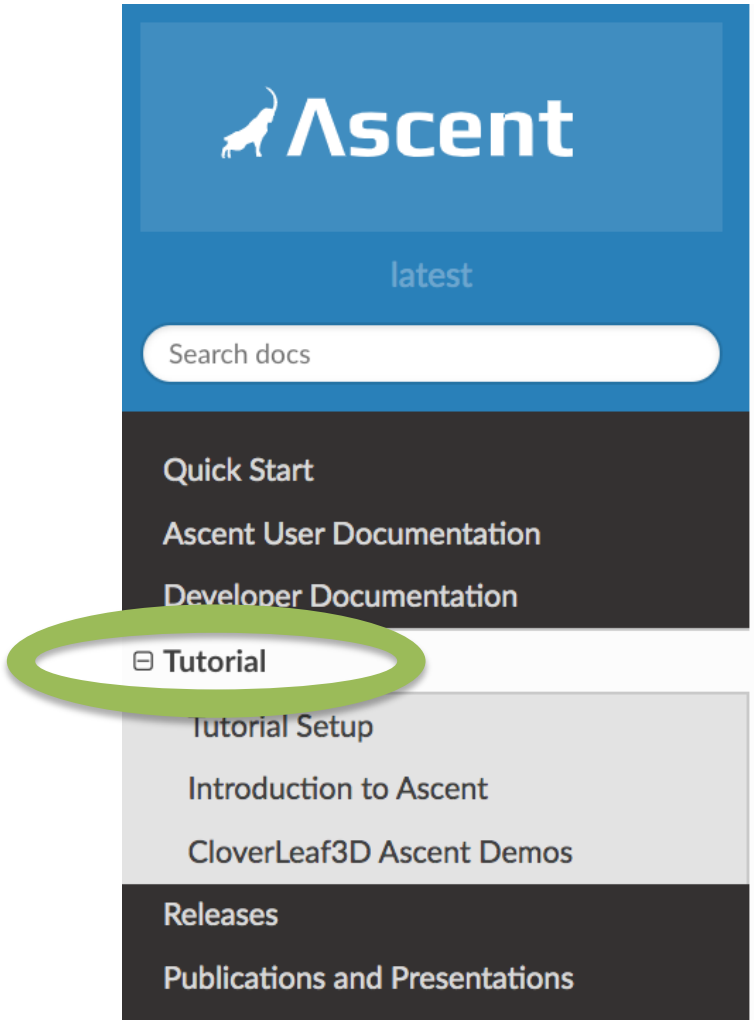
- Formating mesh data for Ascent
- Using Conduit and Ascent's Conduit-based API
- Using and combining Ascent's core building blocks: Scenes, Pipelines, Extracts, Queries, and Triggers
- Using Ascent with the Cloverleaf3D example integration

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 five top-level functions

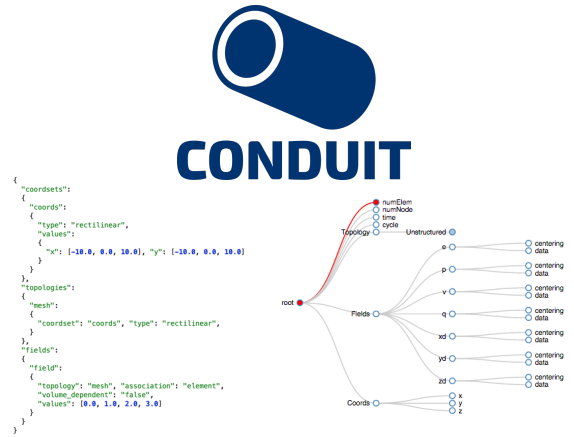
- **open() / close()**
  - Initialize and finalize an Ascent instance
- **publish()**
  - Pass your simulation data to Ascent
- **execute()**
  - Tell Ascent what to do
- **info()**
  - Ask for details about Ascent's last operation

```
//
// Run Ascent
//

Ascent ascent;
ascent.open();

ascent.publish(data);
ascent.execute(actions);
ascent.info(details);

ascent.close();
```

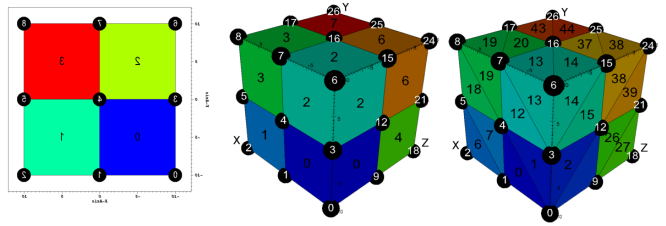
The *publish(), execute(),* and *info()* methods take Conduit trees as an argument.
What is a Conduit tree?

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

```
#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);
```

Instrument your "main" loop or similar function with access to evolving simulation state

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

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

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

Create an Ascent instance and set it up

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

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

```
//
// 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";
```

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"
```

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

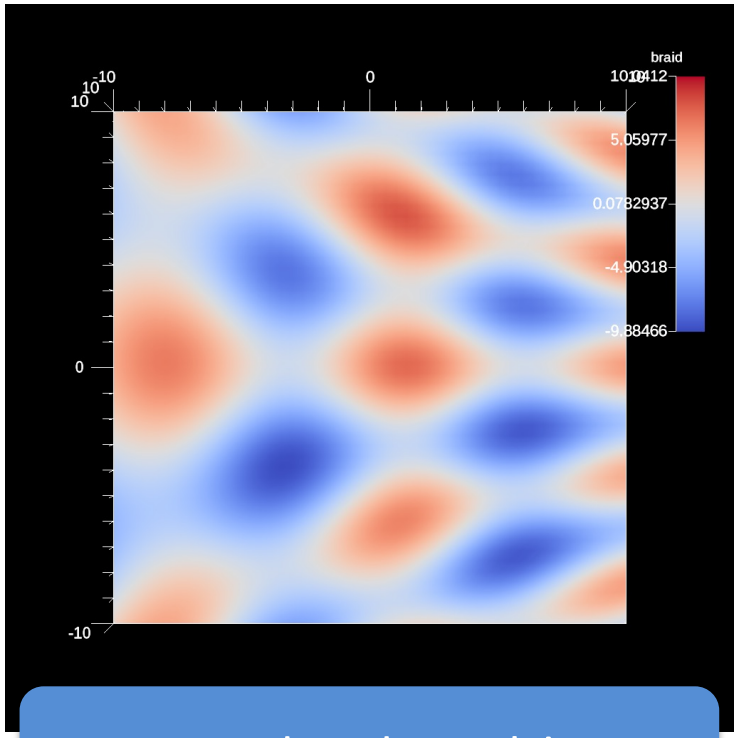
Equivalent YAML Description

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

- https://ascent.readthedocs.io/en/latest/Tutorial_Intro_First_Light.html

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

Tell Ascent to execute these actions



Rendered Result!

Ascent's interface provides five composable building blocks

Scenes
(Render Pictures)

Pipelines
(Transform Data)

Extracts
(Capture Data)

Queries
(Ask Questions)

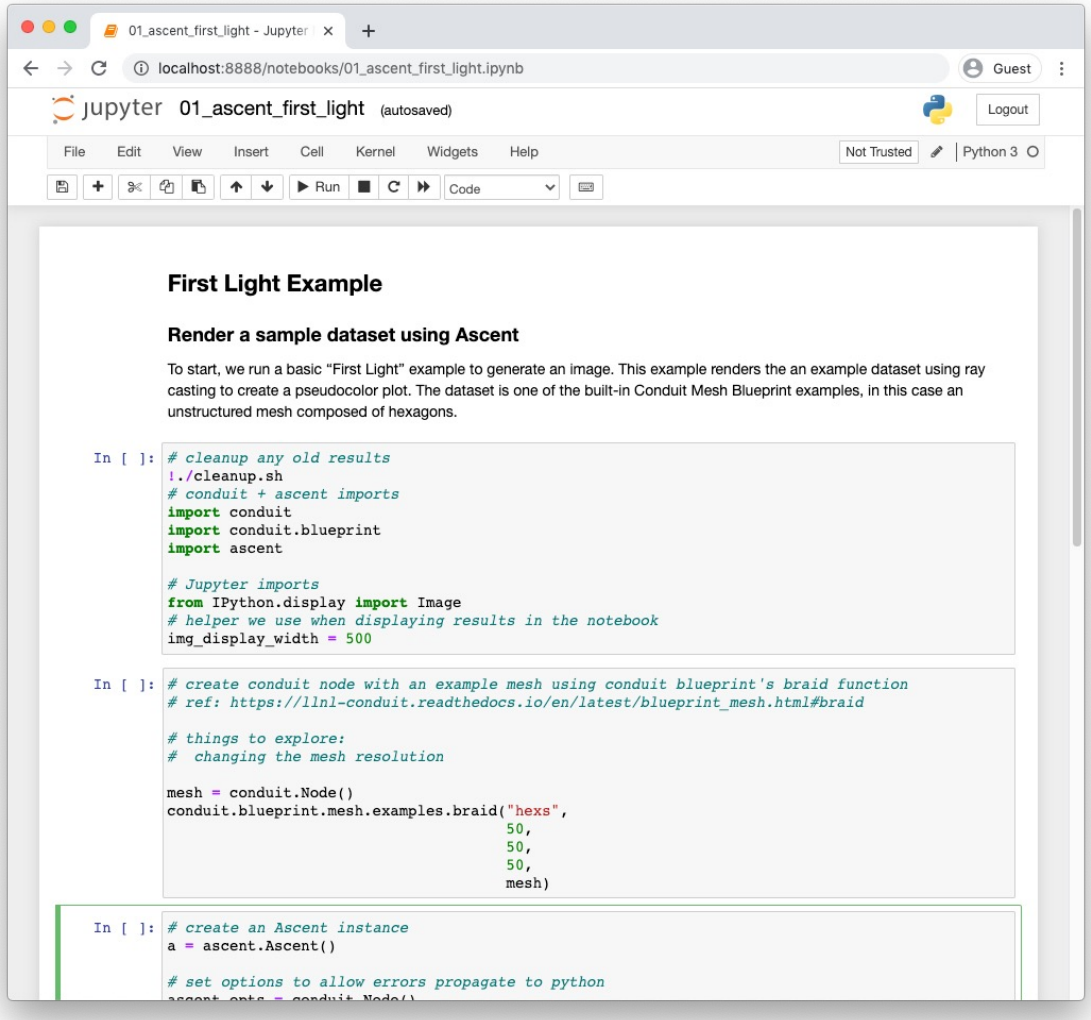
Triggers
(Adapt Actions)

The tutorial provides examples for all of these.

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

NOTE:

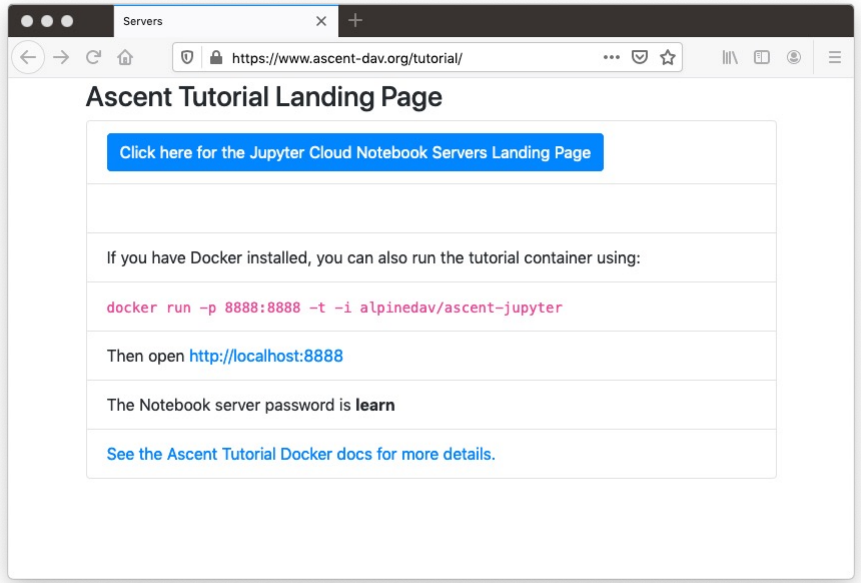
- VPNs or firewalls may block access to general AWS IP addresses and ports
- You may need to disconnect from VPN or request a firewall exemption
- LLNL attendees, you can use the EOR process: https://cspservices.llnl.gov/eor/



You can run our tutorial examples using cloud hosted Jupyter Lab servers

Start here:

https://www.ascent-dav.org/tutorial/



Thanks!

Ascent Resources:

- Github: https://github.com/alpine-dav/ascent
- Docs: http://ascent-dav.org/
- Tutorial Landing Page: https://www.ascent-dav.org/tutorial/

Contact Info:

Cyrus Harrison: cyrush@llnl.gov
Nicole Marsaglia: marsaglia1@llnl.gov

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.