• **Main goals**
  - Provide access to data, data integration & mapping information/scripts
  - Physics-/network- & data-/ML-based solar analytics
  - Enable reproducible, robust, replicable and generalizable R&D in simulation and emulation of solar system integration
  - Encourage/enforce dataset format/I/O standardization

• **National Labs collaboration**
  - ANL, NREL, ORNL, PNNL

• **Main deliverables**
  - Generic and integrated datasets from four power systems operation technologies
    - smart inverters/distributed energy resources management systems (DERMS)
    - phasor measurement units (PMUs) and data concentrators (DCs)
    - smart meters/advanced metering infrastructure (AMI)
    - new sensor technologies
  - Benchmark analytics (physics and ML based algorithms)
    - Distribution state estimation
    - Optimization functions
    - Transients' analysis
    - Data interfaces (CIM, OpenDSS, Ditto, more in the future)
Challenges in Data Integration, Modeling & Simulations

### Analysis

<table>
<thead>
<tr>
<th>Data</th>
<th>Same</th>
<th>Different</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reproducible</td>
<td>Replicable</td>
<td></td>
</tr>
<tr>
<td>Robust</td>
<td>Generalisable</td>
<td></td>
</tr>
</tbody>
</table>

Open Energy Data Initiative (OEDI) – Data Lakes

1. The incoming flow represents data—structured and unstructured—from high-value sources ranging from energy, laboratory, and resource data, analytic tools, use cases, scientific reports, and more.

2. The reservoir of water allows for a confluence of data, making it accessible for analysis in new ways.

3. Cloud vendors enable sustained access to data hosting solutions on a variety of public platforms.

4. The outflow of water is the analyzed energy data, filtered and streamlined for greater access and analysis of data, accelerating new insights and innovation.
Traditional Data Stack
- DMS: SCADA & RTUs & Load Forecasts
- OMS & CIS & IVR: Outages
- GIS: Equipment & Network Connectivity

Smart Grid Data Stack
- AMI/MDM/Smart Meters
- DRM: Demand Response
- Line Monitoring: Fault Location
- DER: Solar, Storage & EV

New Technologies Data Stack
- Point on Wave
- µ-PMUs
- Line Monitoring: Fault Location

Use-case Data Compilation
- Bad Data Filtering
- Imputation
- Synchronization
- Mapping

Solar Integration Analytics
- DMS: SCADA & Monitoring
- Short-term Load Forecasting
- Load Allocation & Voltage Calculation
- Fault Location
- Service Restoration & Switching

- Short-term Load Forecasting
- Load Allocation & Voltage Calculation
- Fault Location
- Service Restoration & Switching
- Optimal Power Flow
- Unit Commitment

- Planning
- Load Growth Analysis
- Renewables Impact
- Electric Vehicles
- Demand Response Allocation
Conceptual Diagram [Note: Use-case based (In addition to Lake)]

Setup
- Model Library: e.g. IEEE Smart-DS
- Timeseries Library
- Algorithm Library

Run
- OEDI SI API
- Algorithm Federate
- Grid Model
- PowerFlow Federate
- HELICS Broker
- HELICS Filter to make AMI/SCADA/etc.
- Simulation Coordination
- Simulation Monitor
- Dockers

Analyze
- Jupyter Notebooks (interactive analysis)
- Data Archive
- User download
- High-level summaries

User Selection
- Simulation Coordination
OEDI SI API: CIMHub converts models to OpenDSS & ATP

**CIMHub Docker-free Option:**
- Java JDK 11+
- download Blazegraph & CIMHub JAR files
- pip install cimhub

https://github.com/GRIDAPPSD/CIMHub/tree/feature/SET0
Feeder Conversion
Comprehensive Feeder Metric Computations
Graph Theory Network Analysis
Feeder Modification:
  - Modify loads
  - Add solar
  - Set controls of components
  - Add/remove electrical components
  - And more!

OEDI SI API: Distribution Transformation Tool (DiTTo)
Early Version Tester

• Expectations
  • Test the current environment
  • Discuss what they can bring from their projects to OEDI – SI
    • Data
      – Public
      – Private (Confidential)
    • Models
    • Algorithms
  • Become founding members in the OEDI – SI Users Group
• Each testers will be assigned an OEDI – SI SME
  • A follow-up meeting will be setup with each tester
Setting Up Docker Environment

- Docker container
  - Standard unit of software that packages up code and all its dependencies
  - Replicate results of a workflow on different systems without having to load all the different dependencies that each component may require

Install Docker on Ubuntu

To install docker on a Linux environment, first update the Linux package manager and install dependencies to allow the package manager to install them over HTTPS:

```bash
$ sudo apt-get update
$ sudo apt-get install ca-certificates curl gnupg lsb-release
```

Then add Docker's GPG keys and set up the repository with the commands:

```bash
$ sudo mkdir -p /etc/apt/keyrings
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
$ sudo mkdir -p /etc/apt/keyrings
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
```

Finally, update the package index and install the latest version of Docker:

```bash
$ sudo apt-get update
$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin
```

Install Docker Desktop on Windows

- Download the installation .exe
  [https://docs.docker.com/desktop/install/windows-install/](https://docs.docker.com/desktop/install/windows-install/)
- Run the installer
- Follow the instructions on the installation wizard
- Detailed instructions in the User Guide
Using NREL’s DSSE container as an example

- Load Docker Image
  
  $ docker load < example.tar.gz

- Connect Volume

  - Linux
    
    $ mkdir outputs_build
    
    $ docker volume create --name gadal_output --opt type=none --opt device=/path/to/folder/outputs_build --opt o=bind
  
  - Windows
    
    $ docker volume create --name gadal_output --opt type=none --opt device=/c/path/to/folder/outputs_build --opt o=bind

- Run the Container

  $ docker run --rm --mount source=gadal_output,target=/simulation/outputs gadal-example:0.0.0

Please refer to the README in each container for detailed instructions.
Components

- OpenDSS feeder simulation
- Basic measurement federate
- Recording federates
- Distribution system state estimator
- Plotting script to visualize the results
DSSE Use-case Example

1. Run Docker
2. Input Data for 24hr at int=15 min
3. Start PF Fed
4. Run OpenDSS
5. Start Sensor Fed
6. Create Measurements
7. Run State Estimation
8. Start Algorithm Fed
9. Start Recorder Fed
10. Data in JSON
11. Plot Calculate Metrics
12. Output in csv format
13. Simulation Monitor
14. N/A

Local Workstation
HELICS Message Bus
Power Flow Fed
Sensor Fed
Algorithm Fed
Docker Container

Office of Energy Efficiency & Renewable Energy
Solar Energy Technologies Office