Open Framework for Integrated Data Operations (OpenFIDO)

CEC EPC-17-047
CPR Meeting #2 - 10 November 2022

David P. Chassin, PI
SLAC National Accelerator Laboratory
Menlo Park, California

SLAC National Accelerator Laboratory is operated by Stanford University for the US Department of Energy under Contract DE-AC02-76SF00515

CPR Meeting Agenda

Welcome/Introductions

Project goals and objectives

- Updates on major project tasks
- Results
  - Testing and Validation
  - Developer and User Training
- Next steps
- Q&A/Discussion

Administrative Review

- Project schedule and deliverable status
- Budget status
  - CEC funds
  - Matching funds
- Questions
- CPR determination

Project Update

Project goals and objectives

The Problem

- Need to exchange system data & models
- For plans, permits, & compliance
- Relating to distribution energy resources

Target Audience

- Utility planners
- Consumer and advocates
- Hardware and software vendors
- Energy/climate regulators & policy-makers

Product Delivery

- Commercial partner for long-term support

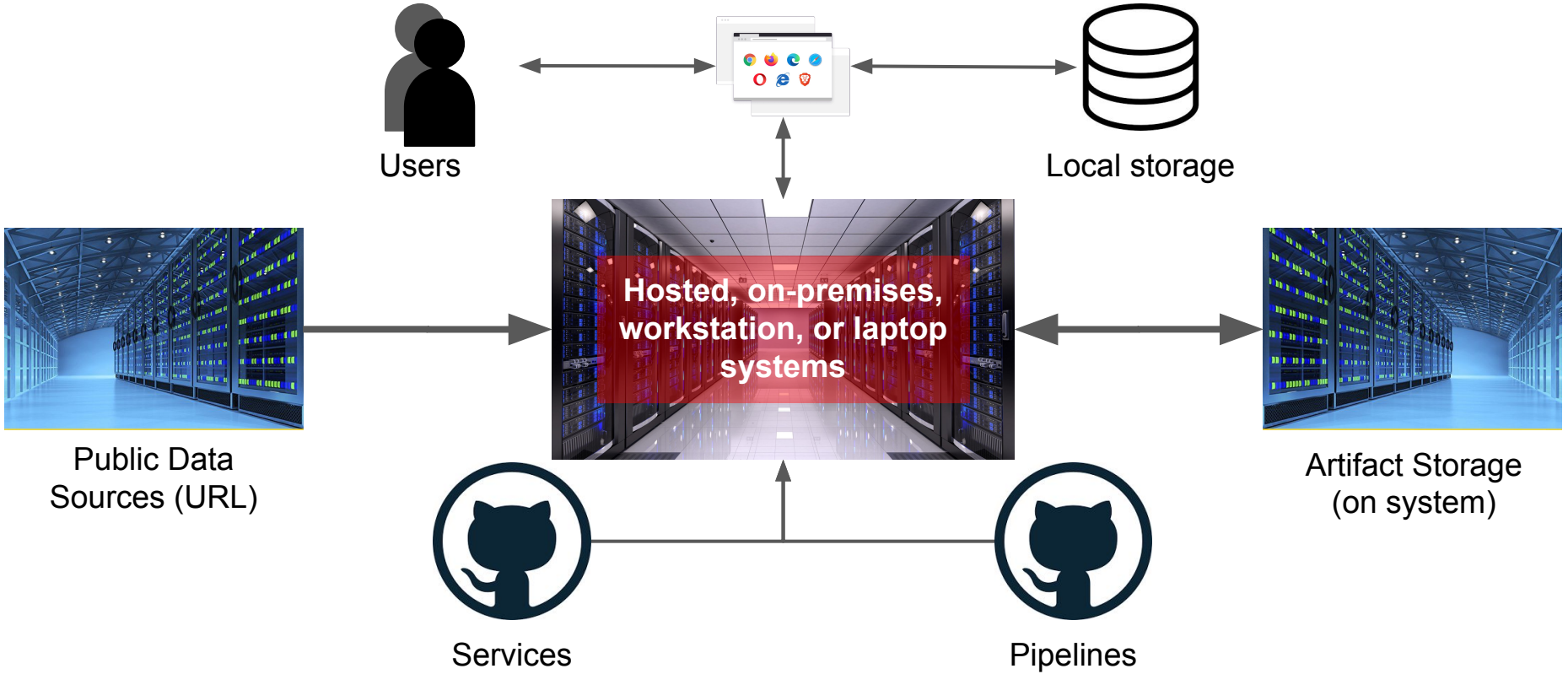
Goals

- Data interchange, synthesis, analysis tool
- Cloud-enabled scalable delivery system
- Focus on DER integration challenges

Objectives

- Modern software infrastructure
- Open-source system
- Collaborations with CEC/DOE projects
  - HiPAS GridLAB-D (CEC-17-046)
  - Hitachi GLOW (CEC-17-043)
  - GRIP (DOE GMLC climate resilience)

OpenFIDO Technical Approach: Platform architecture



OpenFIDO Technical Approach: System Components

Application services on GitHub

Authentication: access control to servers/data

Application: artifact access and management

Workflow: managers pipeline workers

Client: provides user experience

➢ See https://app.openfido.org/

All code is Python/React

All application code is open source

Pipelines on GitHub

Distribution public @ OpenFIDO org on GitHub

➢ See https://source.openfido.org/

Pipelines are versioned for high reproducibility

- Contributions are public w/open review
- Pipelines are coded in various languages (e.g., Python, C/C++)

All public pipeline code is open source

Allows private GitHub organization and projects

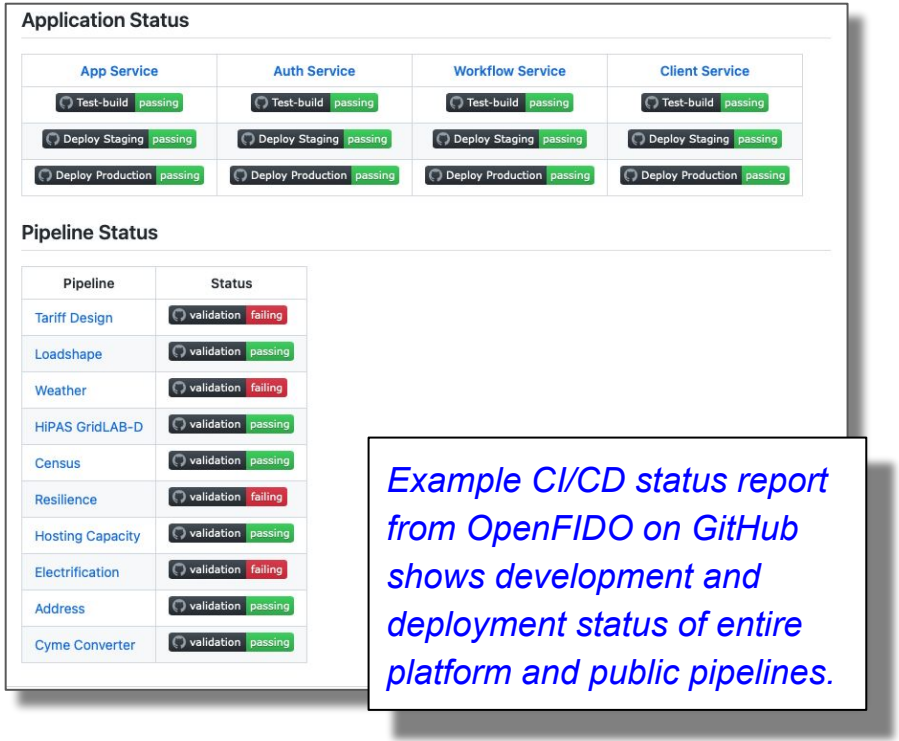
OpenFIDO Technical Approach: CI/CD

Continuous Integration

- Deliver apps from multiple developers/teams
- Introduces automation in development stages
- Solution to problem of integrating new code

Continuous Delivery

- Implements a pipeline of delivery/update tasks
- OpenFIDO uses GitHub "DevOps" methods



Example CI/CD status report from OpenFIDO on GitHub shows development and deployment status of entire platform and public pipelines.

Updates on major project tasks

Task 2 - Requirements Analysis (Done)

- Data exchange requirements and assessment presentation (Done)

Task 2.1 - Analyze existing code (Done)

Task 2.2 - Implementation plan (Done)

Task 2.3 - Validation test plan (Done)

- Data exchange implementation and validation plan presentation (Done)

Task 3 - Implementation (Done)

- CPR Report #1 (Done)

Task 4 - Testing and Validation (Done)

- Testing and Validation Presentation (Done)

Task 5 - Finalize Production (Pending)

- Developer and User Training Documentation (Done)
- Final product release report (Draft)
- CPR Report #2 (Draft)

Task 6 - Evaluation of Project Benefits (Pending)

- Kick-off Meeting Benefits Questionnaire (Done)
- Mid-term Benefits Questionnaire (Canceled)
- Final Meeting Benefits Questionnaire (Pending)

Update on major project tasks (continued)

Task 7 - Technology Transfer Activities

- Updated progress slides (On request)
- CAM Site Visit Schedule (N/A)
- Initial Fact Sheet (Done)
- Final Fact Sheet (Draft)
- Final Presentation Materials (Draft)
- Technology Transfer Plan (Done)
- Technology Transfer Report (Draft)

Notes:

- The draft final report was delivered in the old format more comprehensive.
- In preparing the new shorter final report, several sections were removed.
- These sections will be delivered as updates to the original deliverables from which they were derived.
- Major elements of the project approach, results, and technology transfer sections will be added to the intermediate deliverables when they are finalized.

Key Findings

Requirements

Open Source Software

- Users want open-source software
- Utilities wary of open-source support

Deployment Flexibility

- Utility migration to cloud is going slow
- On-premise servers still preferred
- Local workstation/laptop still desired

Result Reproducibility

- Must retrieve/reproduce old results

Implementation

Deployment Platform

- Python and React preferred
- Python language stability concerns

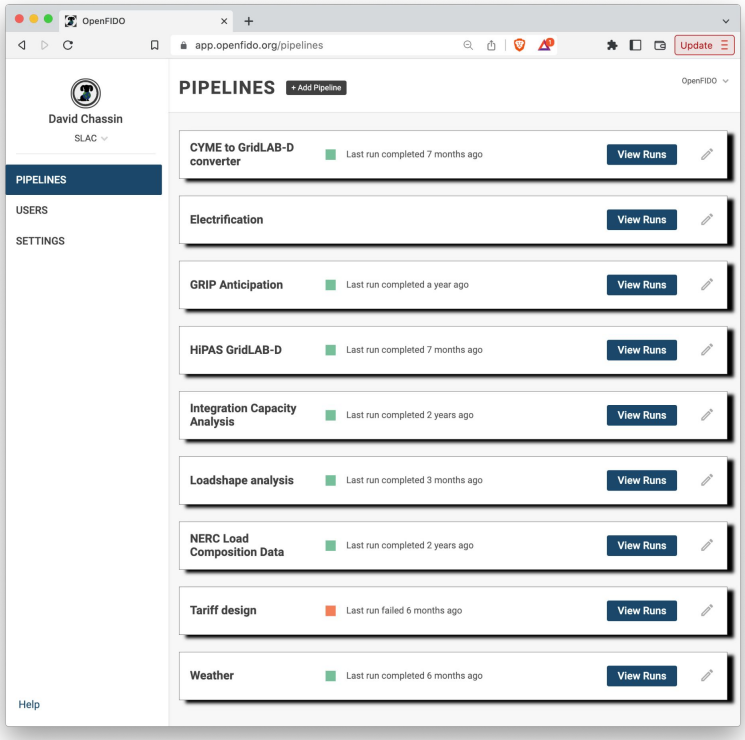
CI/CD Modernization

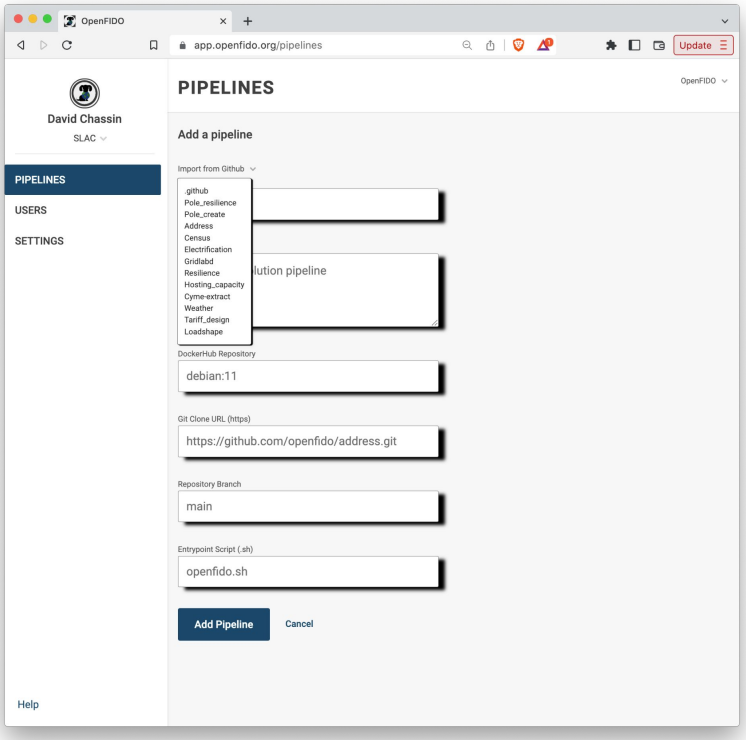
- Older CI/CD tools unstable
- GitHub is preferred

Commercializability

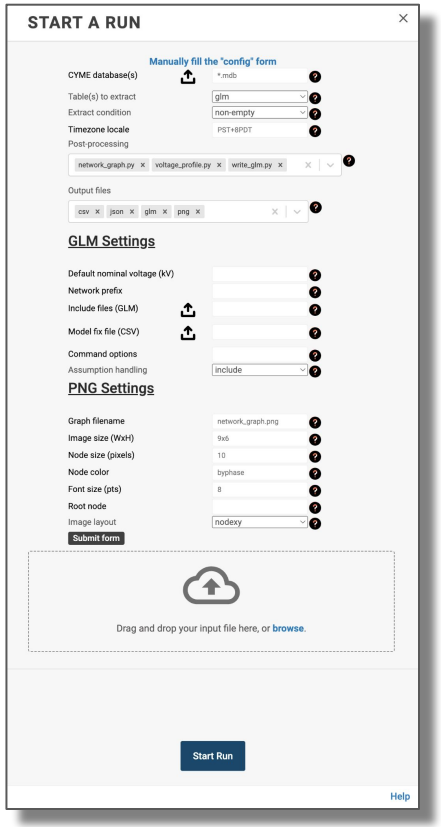
- Limited open-source licenses (no GPL)
- LF Energy application pending

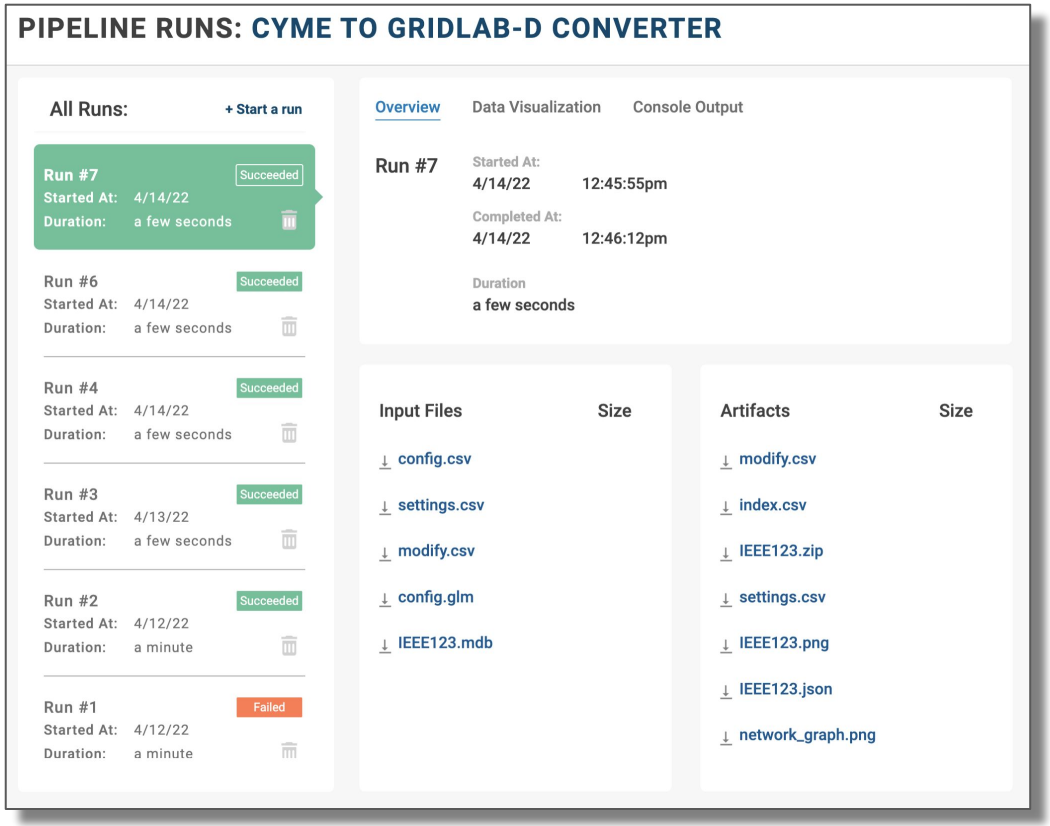
OpenFIDO Pipeline Examples



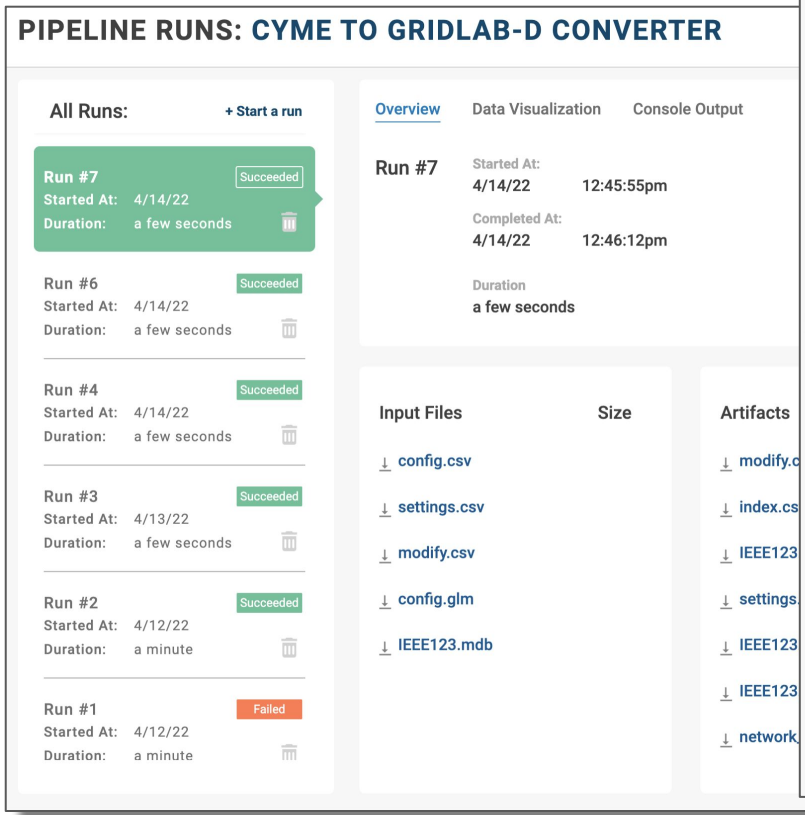


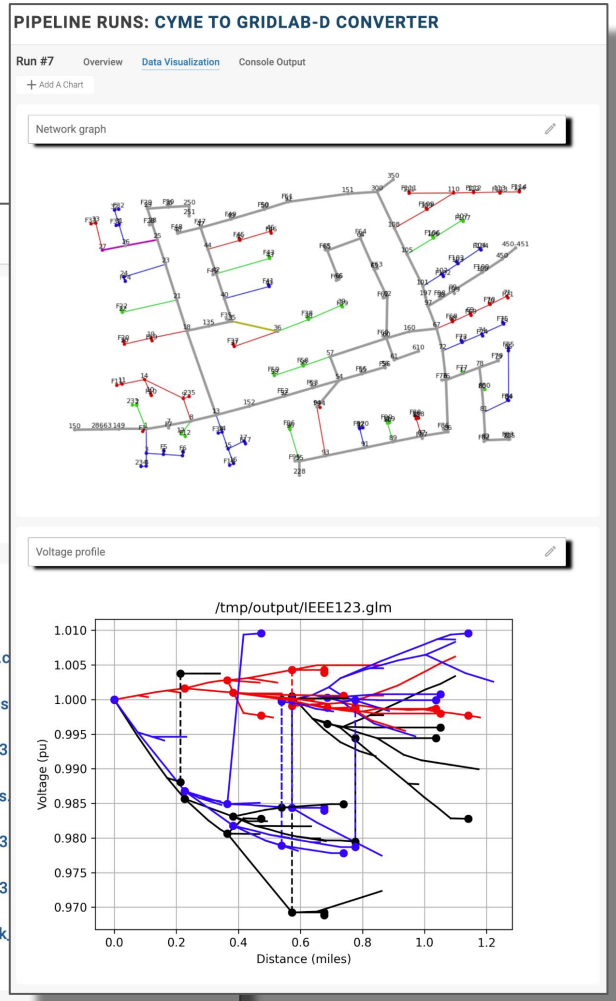
Pipeline Example 1: Cyme Converter



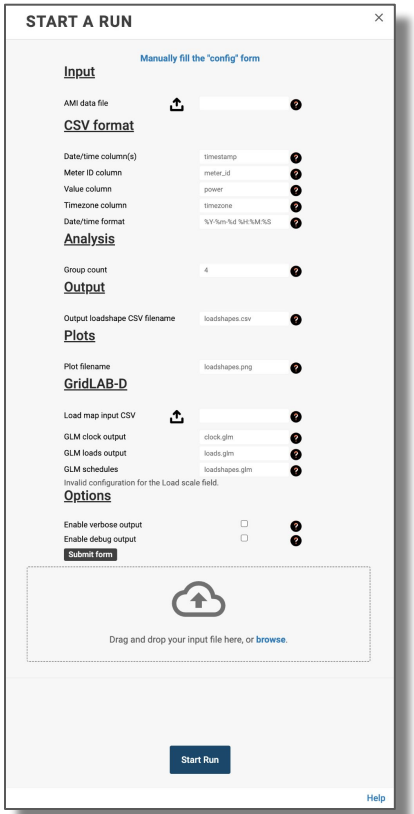


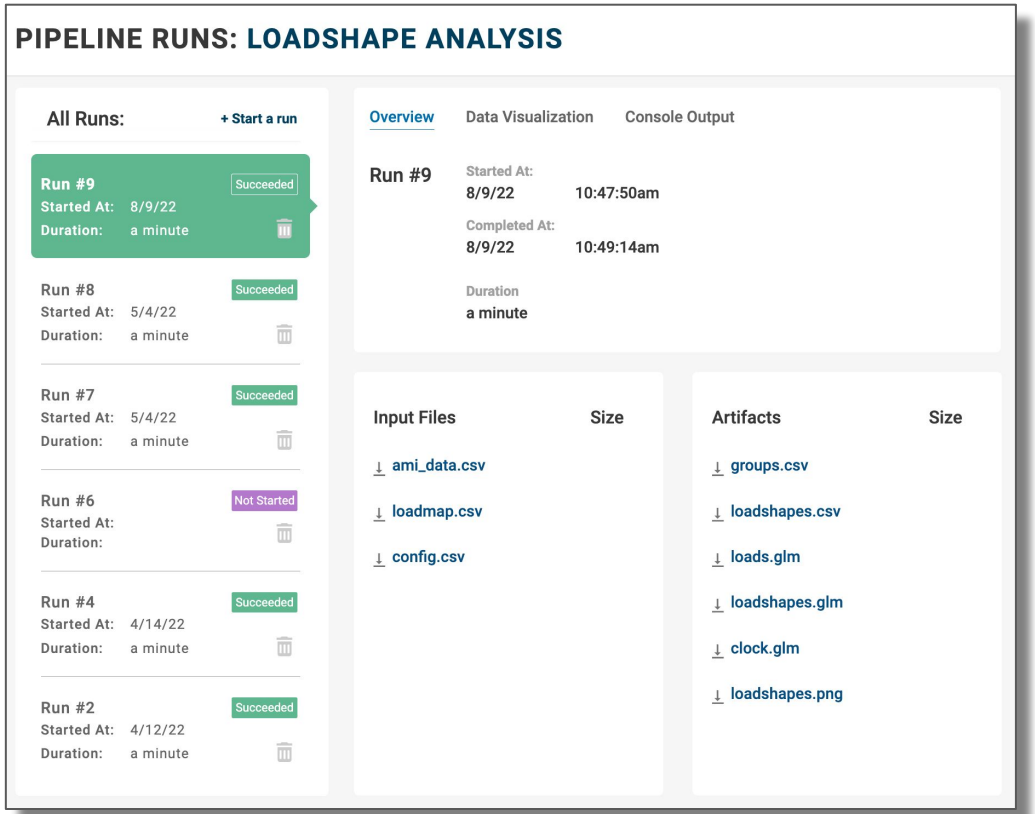
Pipeline Example 1: Cyme Converter



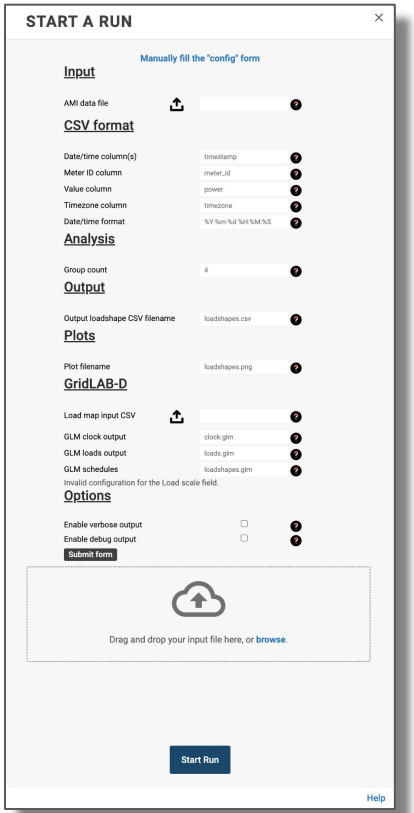


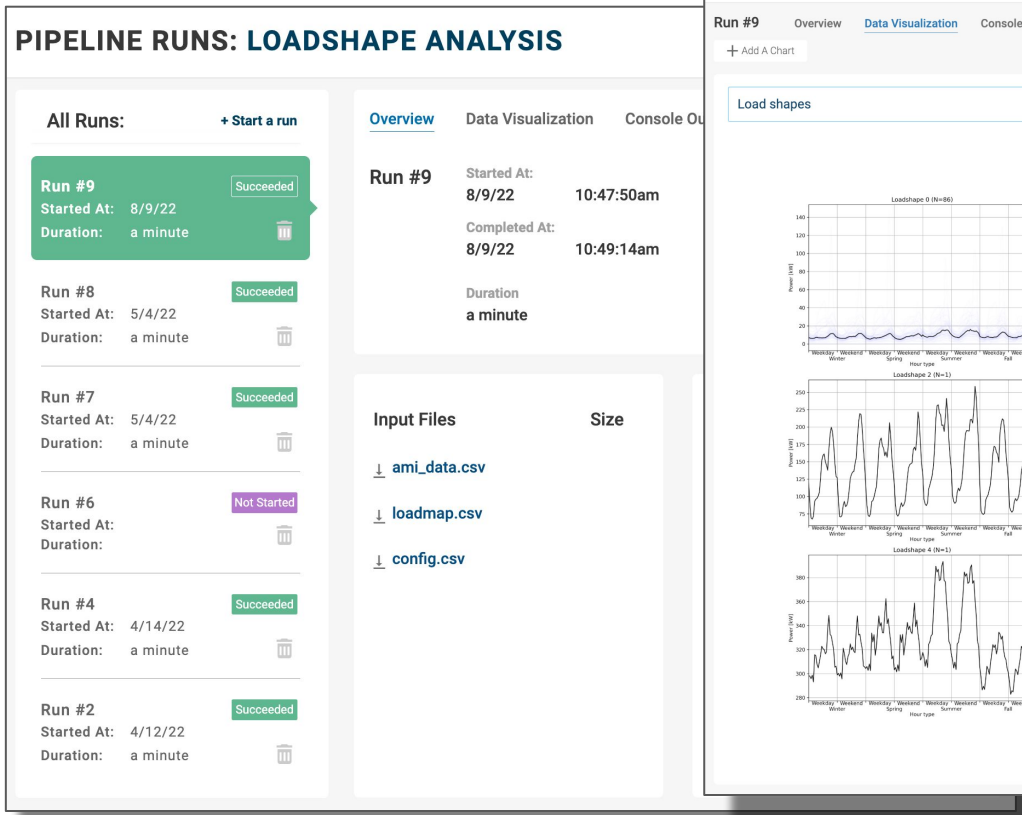
Pipeline Example 2: Loadshape analysis

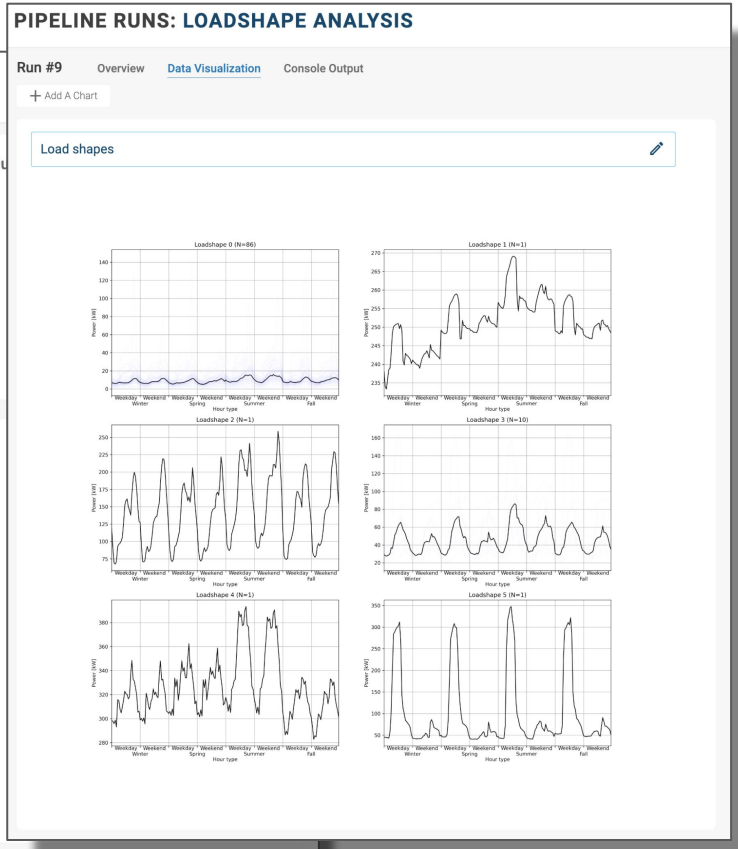




Pipeline Example 2: Loadshape analysis







Results from testing and validation: Inputs and Output

Pipeline inputs

Tariff design: OpenEI, GLM model files

Loadshape: AMI data, SCADA data

Weather: location, year

HiPAS GridLAB-D: GLM model files

Census: location, year

Resilience: GLM model files, time range, location

Hosting Capacity: GLM model files, load models

Electrification: GLM model files, load forecast

Address: location

Cyme converter: Cyme MDB files

Output artifacts

CSV data files

PNG charts/graph images

GLM model files

Results from testing and validation: Quality and Coverage

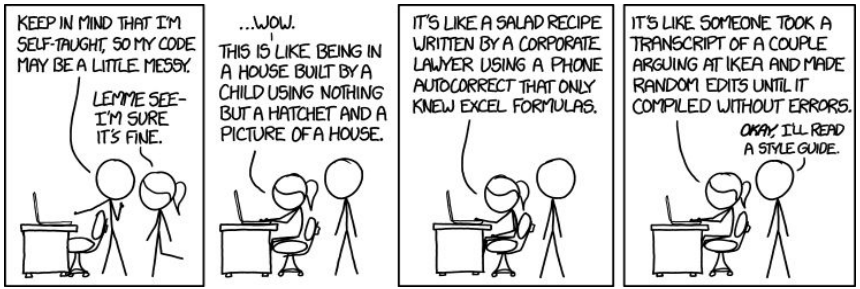
Lint code quality score

Lint score calculation

$$100.0 - 100 \times \frac{5 \times Frequency\ of\ convention\ errors}{Number\ of\ statements}$$

Score must greater than 90% to pass validation

No industry standard exists for lint score



Code testing coverage

Evaluates the fraction of tested statements

Relative to the total number of statements

Must be greater than 99% to pass validation

Google test coverage standards[1]:

- 60% is acceptable
- 75% is commendable
- 90% is exemplary

Gains are logarithmic (diminishing returns)

[1] https://testing.googleblog.com/2020/08/code-coverage-best-practices.html

Results from testing and validation: Approach

Methodology

- CI/CD-based testing/validation (GitHub)
  - Verify programmatic soundness of code
  - Coding style/conventions requirements
  - Verify code correctness
  - Verify testing coverage
- Integration testing
  - Verify that code change requests are ok
  - Verify staged system is ready to deploy
- Pipelines
  - Originate from other projects (e.g., HiPAS)
  - Tested separately and independently
  - Pipeline validation not in OpenFIDO scope

Implementation

- Phase 1 (code submittal)
  - *eslint* (code style/conventions)
  - *black* (code correctness)
  - *test* or *pytest* (unit test/validation)
  - *jest-puppeteer* (integrated test/validation)
    - Authentication
    - Account management
    - Organization management
    - Pipeline creation/running
    - Housekeeping
- Phase 2 (code review)
  - Staged deployment

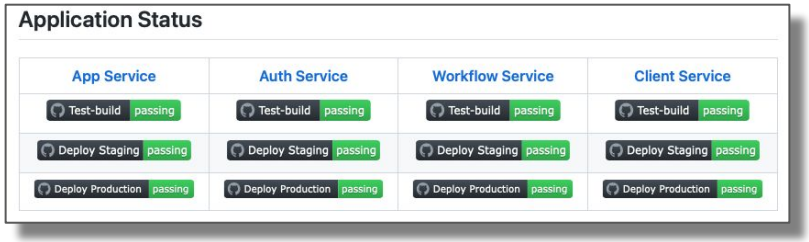
Results from testing and validation: Platform Status

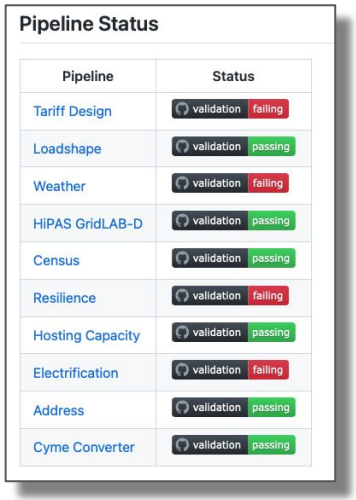
GitHub "Actions" CI/CD status reports

- Applications and pipelines status in real-time

Table A.1 - GitHub Actions

Component	Test	Result	Disposition
App	211 unit tests	>97% coverage	100% tests passing. See Table A.2
	style	100/100 ok	
	lint	>90% score	See Table A.3
Auth	136 unit tests	>92% coverage	100% tests passing. See Table A.4
	style	100% ok	
	lint	>90% score	See Table A.5
Workflow	160 unit tests	>99% coverage	100% tests passing. See Table A.6
	style	100% ok	
	lint	>90% score	See Table A.7
Client	jest-puppeteer	No issues	





Example real-time CI/CD application and pipeline status reports from GitHub.

Results from testing and validation: CI/CD Status

Table A.2 - App Missing Coverage

File location	Missed Statements	Proposed Remedy
app/pipelines/services.py	18/254 uncovered	Add additional tests for uncovered statements.
app/workflows/routes.py	4/134 uncovered	Add additional tests for uncovered statements.
app/workflows/services.py	7/279 uncovered	Add additional tests for uncovered statements.

Table A.3 - App Lint Issues

Test Result	Issue Description	Proposed Remedy
92.6% lint score	Code Style and Formatting	Run the *lint* command, and review each potential style or formatting issue listed against the linter's recommended fix.

Table A.4 - Auth Missing Coverage

File location	Missed Statements	Proposed Remedy
app/auth.py	32/260 uncovered	Add additional tests for uncovered statements.
app/mail.py	13/52 uncovered	Add additional tests for uncovered statements.
app/org.py	14/221 uncovered	Add additional tests for uncovered statements.
app/services.py	3/211 uncovered	Add additional tests for uncovered statements.

Table A.5 - Auth Lint Issues

Test Result	Issue Description	Proposed Remedy
90.4% lint score	Code Style and Formatting	Run the *lint* command, and review each potential style or formatting issue listed against the linter's recommended fix.

Table A.6 - Workflow Missing Coverage

File location	Missed Statements	Proposed Remedy
app/workflows/queries.py	2/52 uncovered	Add additional tests for uncovered statements.

Table A.7 - Workflow Lint Issues

Test Result	Issue Description	Proposed Remedy
92.2% lint score	Code Style and Formatting	Run the *lint* command, and review each potential style or formatting issue listed against the linter's recommended fix.

Results from testing and validation: Pipeline Status

Pipelines status

- HiPAS GridLAB-D master/develop issues
- Once resolved CD status will update ok

Known use-case pipeline issues

- *Resilience*: pole failure accuracy uncertain, some vegetation data missing
- *Tariff design*: slow
- *Electrification*: network losses missing
- *ICA*: speed-up not meeting goals

Table B.1: Pipeline testing/validation summary

Pipeline Name	Validation status	Cause/Action
Tariff design	Failing	HiPAS GridLAB-D template download failed
Loadshape	Passing	
Weather	Failing	HiPAS NSRBD weather download failed
GridLAB-D	Passed	
Census	Passing	
Resilience	Failing	HiPAS GridLAB-D template download failed
Hosting capacity	Passed	
Electrification	Failing	HiPAS GridLAB-D template download failed
Address	Passing	
Cyme Extract	Passing	

Results of testing and validation: Cyme Converter

National Grid Load Forecast (LGF) Study

- 15 year load growth projection
- Analysis is updated annually
- 2021 LGF done w/DOE GridLAB-D
- 2022 LGF done w/HiPAS GridLAB-D
- Converted ~2000 Cyme feeders
- Used "Cyme Extract" pipeline
- Included weather and solar PV

Results of National Grid LGF Study

- Generated HiPAS GridLAB-D models
- 97.5% success unsupervised conversion
- 2.5% required manual intervention
  - Cyme network model errors
  - Cyme load model errors
  - Cyme-GridLAB model mismatches
- Validated based on energy consumption relative to 2021 within load growth

Results from testing and validation: Lessons Learned

Requirements

- Data exchange needs evolve quickly
- Data formats change without warning
- Vendors can be uncooperative/resistant
- Need standard approach to validation
- Need legal framework for data sharing

Implementation

- Dependencies across tools is challenging
- Python changes/updates frequent issue
- Utilities need rapid pipeline deployment
- Utilities need time to validate tools
- Security compliance not part of CI/CD

Contribution Acknowledgments

Southern California Edison

Anthony James (Resilience)
Frank Gonzales (Resilience)
Stacie Bartholow (Cyber-security)

National Grid

Pedram Jahangiri (Load forecast baseline)
Balaji Doraibabu (Load forecast study)
Sayonsom Chando (Load model validation)

Hitachi America Laboratories

Yanzhu Yu (ICA, model validation)
Joseph Chongfuongprinya (Cyme model)
Natsushiko Futamara (AWS performance)

Gridworks

Matthew Tisdale (Benefits analysis)

US Department of Energy SULI Program

Johnson Hsiung (Electrification)
Jewel Newman (Tariff Design)
Michelle Huang (Electrification)
Jorge Higuera (Electrification)
Veronika Lubeck (Load modeling)
Wonseok Choi (ICA)

Presence Product Group

Kevin Rohling (UI/UX requirements/validation)

Contribution Acknowledgments (continue)

SLAC National Accelerator Laboratory

Alyona Teyber (technical manager)

Mitchell Victoriano (software engineer)

Duncan Ragsdale (software engineer)

Anna Peery (software engineer)

Jimmy Leu (software engineer)

Derin Serbetcioglu (former software engineer)

Jonathan Goncalvez (former software engineer)

Stanford University

Xiaochu Wang (Postdoc)

Fuhong Xie (former postdoc)

Lily Buechler (PhD student)

Marie-Louise Arlt (former visiting PhD student)

Sheila Naby (RA)

Kamran Tehranchi (RA)

Mohammed Nijad (RA)

Sara Borchers (RA, now at Tesla)

Adhithya Antonysamy (RA, now at Tesla)

Developer and user training

Developer Documentation

- OpenFIDO orientation (done)
- Application service (done)
- Workflow service (done)
- Client service (done)
- Authentication service (done)

Training Videos

- General orientation video (done): https://youtu.be/WLyI6nimr40
- Developer knowledge transfer (done): https://www.youtube.com/watch?v=JJ3FsPxy-Q4
- Pipeline orientation (pending)
- Youtube channel (in progress)

User Documentation

- Tariff design (done)
- Loadshape (done)
- Weather (done)
- GridLAB-D simulation (done)
- Census (done)
- Resilience (done)
- Hosting capacity (done)
- Electrification (done)
- Address (done)
- Cyme extract (done)

Next steps

Task 5 - Finalize Production

- Developer and User Training Docs
- Final product release report
- CPR Report #2

Task 6 - Evaluation of Project Benefits

- Final Meeting Benefits Questionnaire

Task 7 - Technology Transfer Activities

- Final Fact Sheet
- Final Presentation Materials
- Final Technology Transfer Report

Commercialization/Tech Transfer

- Linux Foundation Energy
- SCE technical support
  - DOE-funded under GRIP
- Outreach
  - PG&E (climate resilience/PSPS)
  - NRECA (grid resilience)
  - National Grid (load forecasting)

Questions / Discussion

Contacts:
David P. Chassin (PI) - dchassin@slac.stanford.edu
Duncan Ragsdale - duncanr@slac.stanford.edu

Administrative Review

Project schedule and status of deliverables

Original schedule

Project Start	10/1/2018
Project End	12/31/2022
Elapsed Time	96%

No-cost extension

Project Start	1/1/2023
Project End	6/30/2023
Elapsed Time	0%

Note: Project End date does not include final three-month closure period.

Deliverables Status (by task)

1.	Final report	In progress
2.	Data exchange requirements and assessment	Done
	Data exchange implementation and validation plan	Done
3.	CPR Report #1	Done
4.	Testing and validation results	Done
5.	Developer and User Training Documentation	Done
	Final product release report	In progress
	CPR Report #2	Pending
6.	Kick-off Meeting Benefits Questionnaire	Done
	Mid-term Benefits Questionnaire	Canceled
	Final Meeting Benefits Questionnaire	Pending
7.	Final Fact Sheet	Pending
	Final Presentation Materials	Pending
	Final Technology Transfer Report	Pending

Budget status

Financial status *(as of 10/1/2022)*

Budget	**1,000,000**	**(100%)**
Labor + M&S	957,487	(96%)
Commitments	7,251	(1%)
Unspent	35,262	(4%)
Invoiced	**1,000,000**	**(100%)**

Matched Funding

Organization	*Committed*	*Received*
National Grid	**30,000**	**42,009 (140%)**

Questions / Discussion

Contacts:
David P. Chassin (PI) - dchassin@slac.stanford.edu
Steve Chao (Finance) - scwchao@slac.stanford.edu