OpenDSS–HELICS Co-Simulation Test Bed

Cyber Security and Critical Infrastructure Protection (CCIP) RC
Kadir Has University

October 24, 2025

Executive Summary

This guide shows you how to wrap the validated **IEEE-13 feeder** inside a *HELICS* federate, launch a two-federate cyber–physical co-simulation, and package everything so that newcomers can reproduce results with a single `docker compose up`. After completing the sprint you will be able to

- start the broker, OpenDSS federate and dummy controller with one command;
- verify that voltages match stand-alone OpenDSS to within 0.1 %;
- push updates to GitHub where CI re-runs the same verification automatically.

1 Directory Layout of the Repository

```
root/
  docker/
     Dockerfile          # builds self-contained image
     entrypoint.sh       # starts broker and federates
     docker-compose.yml  # orchestration
  src/
     dss_federate.py     # Python wrapper around OpenDSS
     dummy_ctrl.py       # subscribes to voltage, publishes set-point
  feeders/
     IEEE13.dss          # & supporting .csv, .dat files
  helics/
     broker.json         # federation topology & timing
  notebooks/
     validate.ipynb      # asserts max|V|  0.1 %
  .github/workflows/ci.yml
```

2 Prerequisites

P1 **Docker** ($\geq$ 24.x) with the `docker compose` plugin.

P2 Git & GitHub account with access to the organisation.

P3 Optional: local Python 3.11 with `helics` and `dss_python` wheels for native debugging.

3 Step-by-Step Guide (Each step ends with a *Verification* block)

3.1 1. Clone the Repository

```
$ git clone git@github.com:CCIP-Research/openDSS-helics-testbed.git
$ cd openDSS-helics-testbed
```

Verification 1 Run

```
$ ls feeders/IEEE13.dss
```

and confirm the feeder master file is listed.

3.2 2. Build the Docker Image

```
$ docker compose build          # ~4 min  on first run
```

Verification 2 Final lines should read:

```
Successfully tagged ccip/opendss-helics:latest
```

Exit code must be 0.

3.3 3. Launch the Federation

```
$ docker compose up             # broker     dss_federate     dummy_ctrl
```

Verification 3 Look for:

```
helics_broker  | Created broker with 2 cores.
dss_federate   | Entered executing mode.
dummy_ctrl     | Subscribed to Vhead.
...            | t=60.0 Vpu=1.04
sim_exit       | Simulation completed, exit code 0.
```

3.4 4. Validate Electrical Results

```
$ docker compose exec dss_federate \
    jupyter nbconvert --to notebook --execute notebooks/validate.ipynb
```

Verification 4 Notebook ends with

```
Assertion passed: max| V | = 0.0007 < 0.001
```

3.5 5. Shut Down

```
$ docker compose down           # clean exit
```

4 Inside `dss_federate.py`

```
import helics as h
import dss

# 0 -- load circuit
FEEDER_PATH = "../feeders/IEEE13.dss"

dss.engine.start("batch", FEEDER_PATH, 0)

# 1 -- broker + publications
sim_time = 0
fed = h.helicsCreateValueFederateFromConfig("../helics/broker.json")
volt_pub = h.helicsFederateGetPublication(fed, "Vhead")
set_sub  = h.helicsFederateGetSubscription(fed, "Pset")

# 2 -- execution mode
h.helicsFederateEnterExecutingMode(fed)

# 3 -- time loop
while sim_time < 60:
    dss.solution.solve()
    vpu = dss.circuit.Buses("sourcebus").puVmagAngle[0] / 110.0
    h.helicsPublicationPublishDouble(volt_pub, vpu)
    if h.helicsInputIsUpdated(set_sub):
        p_set = h.helicsInputGetDouble(set_sub)
        dss.loads.first(); dss.loads.kW(p_set)
    sim_time = h.helicsFederateRequestTime(fed, sim_time + 1)

# 4 -- finish
h.helicsFederateFinalize(fed)
```

5 Git & GitHub Workflow

G1 Create a feature branch: `git checkout -b feature/my_wrapper`.

G2 Commit logically (one task per commit) and push.

G3 Open a Pull Request (PR) against `main` using the template.

G4 GitHub Actions will build the Docker image and run `validate.ipynb`. PRs must turn green before review.

G5 After two approvals (peer + PI) the PR may be merged.

6 Continuous Integration (CI)

The workflow `ci.yml` mirrors Steps 2–4:

- **Build**: cache and build Docker image.
- **Run**: `docker compose up -abort-on-container-exit`.
- **Validate**: execute notebook and store artefacts.

7 Quick Reference Table

Step	Command	Expected Log Snippet	Pass Criterion
Clone	`git clone`	Repo tree visible	Files match layout
Build	`docker compose build`	`Successfully tagged`	Exit 0
Run	`docker compose up`	`Entered executing mode`	60 steps, exit 0
Validate	nbconvert	`Assertion passed`	$\max \lvert\Delta V\rvert < 0.001$

8 Troubleshooting Cheatsheet

Container exits instantly
Check `docker compose logs dss_federate`. Path errors to feeder files are common.

Broker out of sync
Ensure `coreInitString` in `broker.json` matches `-federates=2`. Increase when adding more federates.

Validation fails
Confirm `dummy_ctrl.py` does not override set-points in this sprint and that the baseline CSV exists.

Federate hangs at `Initializing`
Broker port mismatch; verify port 22400 is free or change it consistently across all configs.

9 FAQ

Q: Notebook cannot find `voltages.csv`.
Run the simulation once first—`dss_federate.py` writes the log only after completion.

Q: How do I add a third federate?
Increase `-federates` in `broker.json` and copy the pattern used for `dummy_ctrl.py`.

10 Next Milestones

M1 Integrate HELICS `delay` filter on the set-point channel and re-run validation.

M2 Replace `dummy_ctrl.py` with an ns-3-based controller federate.

M3 Enable PCAP logging for forthcoming ML/FL experiments.

A Reference File Templates

A.1 docker/Dockerfile

```
FROM python:3.11-slim

# Install dependencies
RUN pip install --no-cache-dir helics==3.* dss_python==0.13.* \
    notebook nbconvert jupyterlab

# Create working directory
WORKDIR /app
COPY . /app

ENTRYPOINT ["bash", "docker/entrypoint.sh"]
```

A.2 docker/entrypoint.sh

```
#!/usr/bin/env bash
set -e

# start HELICS broker
autobroker --federates=2 --loglevel=4 &
BROKER_PID=$!

# slight delay to ensure broker ready
sleep 1

# launch federates
python src/dss_federate.py &
python src/dummy_ctrl.py &

wait $BROKER_PID
```

A.3 docker-compose.yml

```
version: "3.8"
services:
  testbed:
    build: ./docker
    volumes:
      - .:/app
    tty: true
```

A.4 src/dummy_ctrl.py

```
import helics as h

FED = h.helicsCreateCombinationFederateFromConfig("../helics/broker.json")
sub = h.helicsFederateGetSubscription(FED, "Vhead")
pub = h.helicsFederateGetPublication(FED, "Pset")

h.helicsFederateEnterExecutingMode(FED)

sim_time = 0
while sim_time < 60:
    # If voltage update received, decide simple control action
    if h.helicsInputIsUpdated(sub):
        v_pu = h.helicsInputGetDouble(sub)
        # simple  d e a d band  controller
        p_cmd = 100 if v_pu < 0.98 else 0
        h.helicsPublicationPublishDouble(pub, p_cmd)
    sim_time = h.helicsFederateRequestTime(FED, sim_time + 1)

h.helicsFederateFinalize(FED)
```

A.5 helics/broker.json

```
{
  "name": "ccip_testbed_broker",
  "cores": 1,
  "coreType": "zmq",
  "coreInitString": "--federates=2",
  "timeDelta": 1.0,
  "publications": [
    {"name": "Vhead", "type": "double"}
  ],
  "subscriptions": [
    {"name": "Pset", "type": "double"}
  ]
}
```

A.6 .github/workflows/ci.yml

```
name: CI
on:
  push:
    branches: [ main ]
  pull_request:

jobs:
  build-test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - name: Build & run testbed
        run: |
          docker compose -f docker/docker-compose.yml build
          docker compose -f docker/docker-compose.yml up --abort-on-
             container-exit --exit-code-from testbed
```

A.7 notebooks/validate.ipynb

Below is a minimal Python cell layout (exported as JSON in the actual file):

```
import pandas as pd
import numpy as np

baseline = pd.read_csv("comparison/baseline.csv")
sim      = pd.read_csv("logs/voltages.csv")

max_delta = np.max(np.abs(baseline['V'] - sim['V']))
assert max_delta < 1e-3, f"Voltage␣deviation␣too␣high:␣{max_delta:.4f}␣pu"
print("Assertion␣passed:␣max| V |␣=", max_delta)
```