INL/CON-25-84339-Revision-0

Creating Apptainer Workflows with Docker-Compose-like Utilities

May 2025

Brandon S Biggs

Changing the World's Energy Future

INL/CON-25-84339-Revision-0

Creating Apptainer Workflows with Docker-Compose-like Utilities

Brandon S Biggs

May 2025

Idaho National Laboratory
Idaho Falls, Idaho 83415

http://www.inl.gov

Prepared for the
U.S. Department of Energy
Under DOE Idaho Operations Office
Contract DE-AC07-05ID14517

May 7, 2025

Brandon Biggs
Title

Creating Apptainer Workflows with Docker-Compose-like Utilities

Battelle Energy Alliance manages INL for the
U.S. Department of Energy's Office of Nuclear Energy

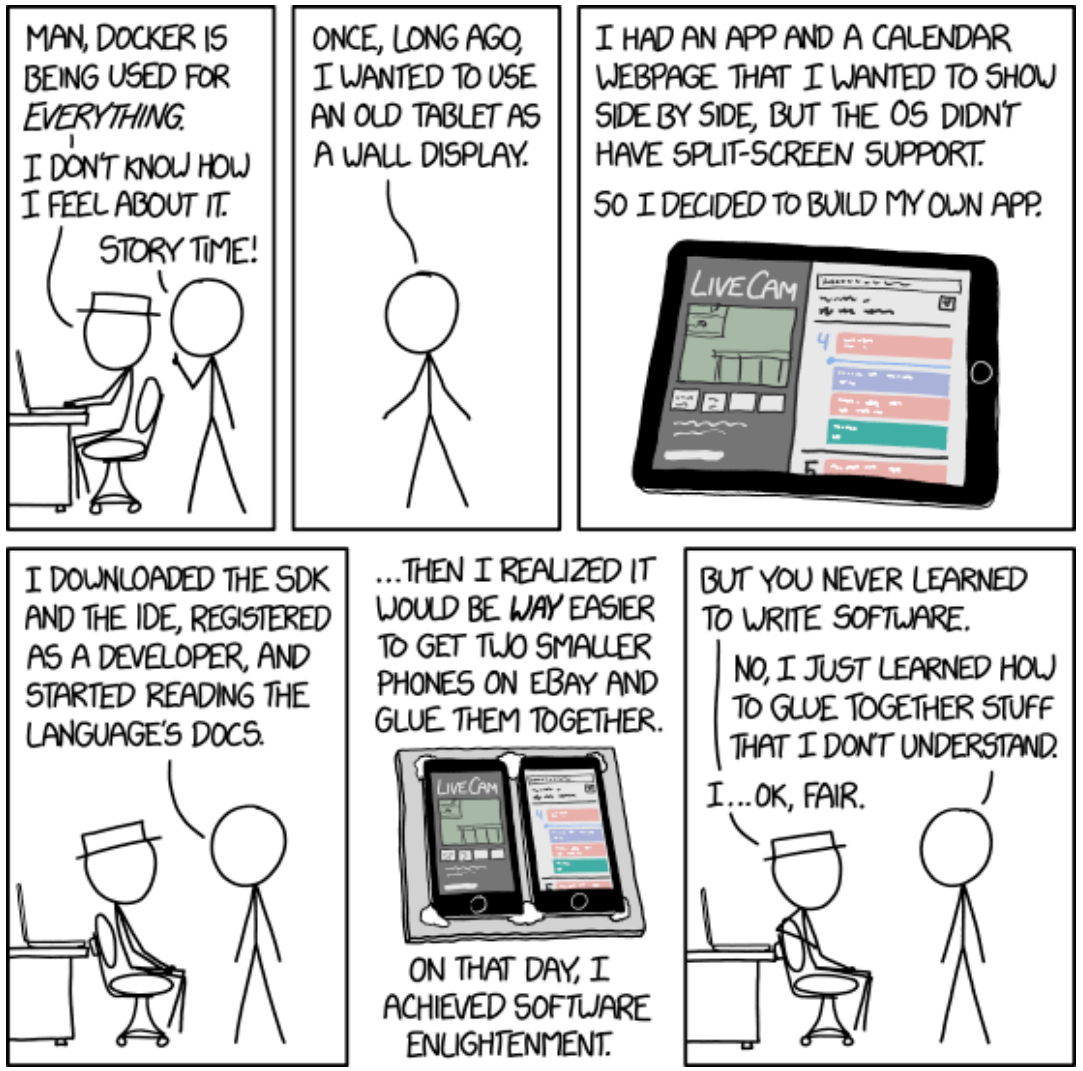
Idaho National Laboratory

Roadmap

- Motivation For Docker-Compose like Utilities
- Available Options
- A Potential Solution
- Example
- Benefits

Motivation

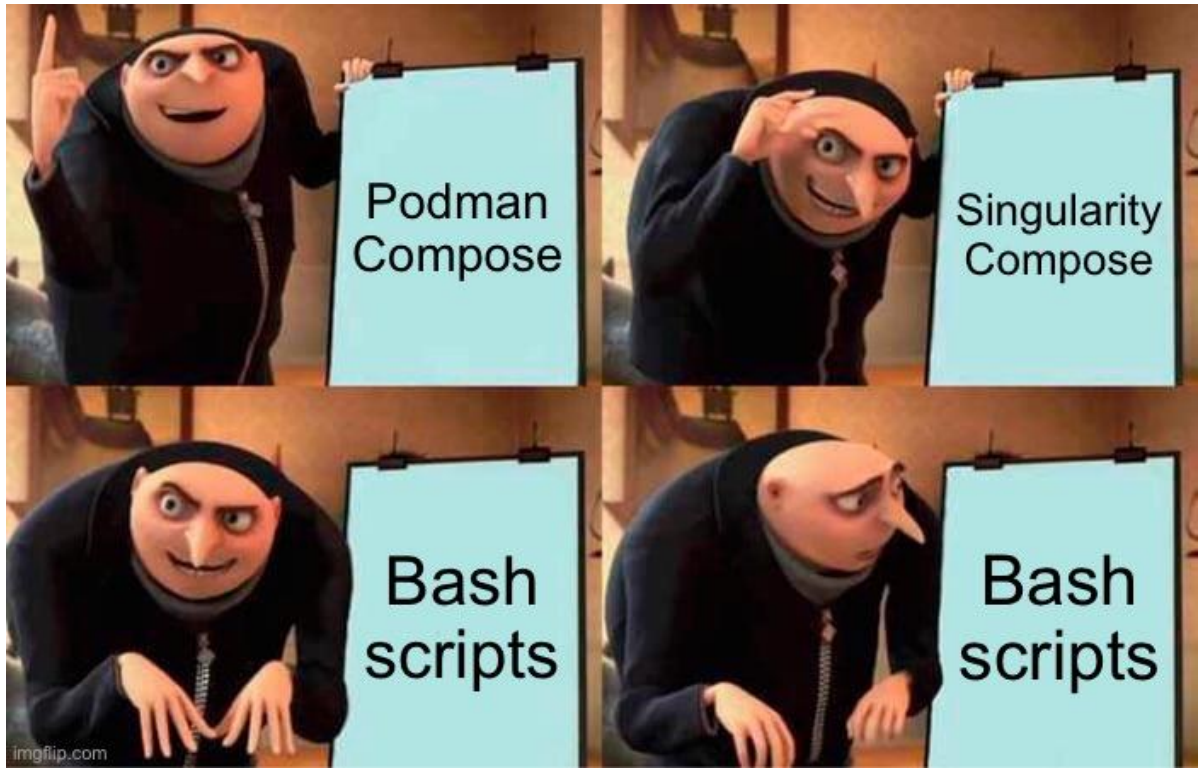
- Users wanted to run software workflows on HPC systems. They asked for Docker/Docker Compose but it isn't allowed on our HPC systems.
- We could tell them to do it without Docker but…
  - Software is complex.
  - Containers are helpful, but combining containers can be tricky too.
  - Our users are mostly engineers. Many don't care how to get app installed and working with other apps.
  - Reproducibility and shareability is important. A series of bash, python, make, perl, etc scripts can be hard to share with less experienced users.
- So, what are our options?



https://xkcd.com/1988/ - Containers

Available Options

- A lot of work has been done to simplify access to scientific computing applications/workflows.
  - Podman Compose [1]
  - Singularity Compose [2]
  - Fuzzball [3]
  - Bash scripts??
  - And many more



[1] https://github.com/containers/podman-compose
[2] https://github.com/singularityhub/singularity-compose
[3] https://ciq.com/products/fuzzball/

A Potential Solution - Process Compose

Pros

- Flexible - Built for regular processes, so it can take advantage of HPC nuances and containers
- Familiar - Similar yml syntax to Docker Compose
- Portable - Just text config files and a single Go binary, easy to version control
- Secure - Root not required
- Simple - Allows software developers to handle dependencies while users have less to figure out

Cons

- Wasn't built solely for containers or HPC
- Not as mature or as concise as Docker Compose
- Not a one-to-one swap
- Doesn't abstract away container or bash nuances (also a pro?)

https://f1bonacc1.github.io/process-compose/

Example – Data Science Stack

- TimescaleDB [1] for time series data
- Jupyter Notebook [2]
- Streamlit [3] (Python based web frontend)

[1] https://www.timescale.com
[2] https://jupyter.org
[3] https://streamlit.io

Example – Docker Compose

```
version: '3.8'

services:
timescaledb:
image:
timescale/timescaledb:latest-
pg14
container_name: timescaledb
ports:
- "5432:5432"
environment:
POSTGRES_USER: postgres
POSTGRES_PASSWORD: postgres
POSTGRES_DB: gdp_db
volumes:
- timescale-
data:/var/lib/postgresql/data
```

```
jupyter:
image: jupyter/scipy-
notebook
container_name: jupyter
ports:
- "8888:8888"
volumes:
-
./notebooks:/home/user_name
environment:
JUPYTER_ENABLE_LAB: "yes"
command: >
start-notebook.sh
```

```
streamlit:
image: python:3.9-slim
container_name: streamlit
ports:
- "8501:8501"
volumes:
- ./app:/app
working_dir: /app
command: >
bash -c "pip install –r
requirements.txt \
&& streamlit run app.py"
```

https://medium.com/@tzhaonj/how-to-use-docker-compose-to-quickly-spin-up-a-full-blown-data-science-stack-3a690bedf702

Example – Process Compose

File: **process-compose.yml**

```yaml
version: "0.5"
vars:
  CONTAINER_RUNTIME: apptainer

processes:
  # Do the initial setup steps like cloning the repo and building the container
  make-data-dirs:
    command: "mkdir -p timescale-data notebooks"
  make-postgres-dirs:
    command: "mkdir -p postgres-run"
  clone-repos:
    command: "git clone https://github.com/streamlit/roadmap.git || git -C roadmap pull"
  build_def:
    command: "echo -e 'Bootstrap: docker\nFrom: python:3.13.3\n\n%post\npip install streamlit' > streamlit.def"
  build_sif:
    command: "[ ! -f 'streamlit.sif' ] && {{ .CONTAINER_RUNTIME }} build streamlit.sif streamlit.def"
    depends_on:
        build_def:
          condition: process_completed_successfully

```

File: **process-compose.yml**

```yaml

  # Start the services
  timescaledb:
    command: "{{ .CONTAINER_RUNTIME }} run docker://timescale/timescaledb:latest-pg14"
    environment:
      - APPTAINER_BINDPATH=timescale-data:/var/lib/postgresql/data,postgres-run:/var/run/postgresql
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=postgres
      - POSTGRES_DB=gdp_db
    depends_on:
      make-data-dirs:
        condition: process_completed_successfully
      make-postgres-dirs:
        condition: process_completed_successfully

  jupyter:
    command: "{{ .CONTAINER_RUNTIME }} run docker://jupyter/scipy-notebook start-notebook.sh"
    environment:
      - APPTAINER_BINDPATH=notebooks:/notebooks
      - JUPYTER_ENABLE_LAB=yes
    depends_on:
      make-data-dirs:
        condition: process_completed_successfully

  streamlit:
    command: "{{ .CONTAINER_RUNTIME }} exec streamlit.sif bash -c 'cd /app && streamlit run streamlit_app.py'"
    environment:
      - APPTAINER_BINDPATH=roadmap:/app
    depends_on:
      clone-repos:
        condition: process_completed
      build_sif:
        condition: process_completed_successfully

```

```
./process-compose -f process-compose.yml
```

```
Version:    v1.46.0                                          Process Compose 🔥
Hostname:   [redacted]
Processes:  3/8
RAM | CPU:  88.4 MiB | 0.1%

● PID(P) NAME(N)           NAMESPACE(C)  STATUS(S)   AGE(A)  HEALTH(H)  MEM(M)     CPU(U)  RESTARTS(R)  EXIT CODE(E)
● 54330  build_def         default       Completed   -       -          -          -       -            0
● 54341  build_sif         default       Completed   42s     -          -          -       -            0
● 54329  clone-repos       default       Completed   0s      -          -          -       -            0
● 54345  jupyter           default       Running     9m9s    -          30.3 MiB   0.0%    -            -
● 54328  make-data-dirs    default       Completed   0s      -          -          -       -            0
● 54327  make-postgres-dirs default       Completed   -       -          -          -       -            0
● 57454  streamlit         default       Running     8m26s   -          28.8 MiB   0.0%    -            -
● 54346  timescaledb       default       Running     9m9s    -          29.3 MiB   0.0%    -            -

                                        jupyter
demo
[I 2025-04-10 10:16:58.719 ServerApp] Jupyter Server 2.8.0 is running at:
[I 2025-04-10 10:16:58.719 ServerApp] http://h100.hpc.inl.gov:8888/lab?
token=5abd37cdb9f985c43b152474246e1eab65d0b48dfda8d2e3
[I 2025-04-10 10:16:58.719 ServerApp]     http://127.0.0.1:8888/lab?
```

Benefits – Why would I use this over Bash, Python, or <insert tool>?

- Low barrier to entry
- Full stack in 3 commands with no root
  - Curl binary, curl process-compose.yml, ./process-compose –f yml
- A lot of similarities to tools that already exist
- Portable – All setup code can be in a single file

Questions?

Idaho National Laboratory

Battelle Energy Alliance manages INL for the U.S. Department of Energy's Office of Nuclear Energy. INL is the nation's center for nuclear energy research and development, and also performs research in each of DOE's strategic goal areas: energy, national security, science and the environment.

Process-compose.yml

```
version: "0.5"
vars:
  CONTAINER_RUNTIME: apptainer

processes:
  # Do the initial setup steps like cloning the repo and building the container
  make-data-dirs:
    command: "mkdir -p timescale-data notebooks"
  make-postgres-dirs:
    command: "mkdir -p postgres-run"
  clone-repos:
    command: "git clone https://github.com/streamlit/roadmap.git || git -C roadmap pull"
  build_def:
    command: "echo -e 'Bootstrap: docker\nFrom: python:3.13.3\n\n%post\npip install streamlit' > streamlit.def"
  build_sif:
    command: "[ ! -f 'streamlit.sif' ] && {{ .CONTAINER_RUNTIME }} build streamlit.sif streamlit.def"
    depends_on:
       build_def:
         condition: process_completed_successfully

  # Start the services
  timescaledb:
    command: "{{ .CONTAINER_RUNTIME }} run docker://timescale/timescaledb:latest-pg14"
    environment:
      - APPTAINER_BINDPATH=timescale-data:/var/lib/postgresql/data,postgres-run:/var/run/postgresql
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=postgres
      - POSTGRES_DB=gdp_db
    depends_on:
      make-data-dirs:
        condition: process_completed_successfully
      make-postgres-dirs:
        condition: process_completed_successfully

  jupyter:
    command: "{{ .CONTAINER_RUNTIME }} run docker://jupyter/scipy-notebook start-notebook.sh"
    environment:
      - APPTAINER_BINDPATH=notebooks:/notebooks
      - JUPYTER_ENABLE_LAB=yes
    depends_on:
      make-data-dirs:
        condition: process_completed_successfully

  streamlit:
    command: "{{ .CONTAINER_RUNTIME }} exec streamlit.sif bash -c 'cd /app && streamlit run streamlit_app.py'"
    environment:
      - APPTAINER_BINDPATH=roadmap:/app
    depends_on:
      clone-repos:
        condition: process_completed
      build_sif:
        condition: process_completed_successfully
```