

CHAP-DEVOPS

The chap CLI, without Python

```
uvx --from chap-core chap becomes chap-devops chap
```

For people who use the chap CLI already

What this deck covers

1. What changes for you, and what does not
2. How `chap-devops chap` runs chap: the container, the mounts, the user
3. Which image and which chap-core version it runs
4. Models: GitHub, directories, URLs, marketplace ids and built-in models
5. Models that run in docker, and an MLproject as a chapkit service
6. Your files: what is visible, what is written, what chap-devops reports
7. Every message that stops a run, and what to do
8. A full example: three models, compared
9. Troubleshooting

What you do today

```
uv tool install chap-core --python 3.13
chap eval --model-name https://github.com/dhis2-chap/minimalist_example_r \
  --dataset-csv laos_subset.csv --output-file r.nc
chap plot-backtest r.nc --output-file r.html
```

Before the first evaluation, you need:

- Python 3.13 and uv
- R, renv and INLA, for the R models
- Docker, for the `docker_env` models
- the same versions on every machine of your team

What you do with chap-devops

```
chap-devops chap eval --model-name https://github.com/dhis2-chap/minimalist_example_r \
  --dataset-csv laos_subset.csv --output-file r.nc
chap-devops chap plot-backtest r.nc --output-file r.html
```

- **The same commands, the same options.** Each argument after `chap` goes to chap unchanged.
- You need **Docker and nothing more**. chap runs in chap-core's own image.
- R, INLA and uv are in the image already.

One change of words

Today	With chap-devops
<code>chap eval ...</code>	<code>chap-devops chap eval ...</code>
<code>chap plot-backtest ...</code>	<code>chap-devops chap plot-backtest ...</code>
<code>chap export-metrics ...</code>	<code>chap-devops chap export-metrics ...</code>
<code>chap validate data.csv</code>	<code>chap-devops chap validate data.csv</code>
<code>chap --help</code>	<code>chap-devops chap</code>
<code>chap eval --help</code>	<code>chap-devops chap eval --help</code>
<code>uvx --from chap-core==2.5.0 chap ...</code>	<code>chap-devops chap --tag v2.5.0 ...</code>

The options of chap-devops itself

They come **between** `chap` and chap's own command:

Option	What it does
<code>--tag TAG</code>	The chap-core version to run
<code>--image core</code> or <code>--image worker</code>	The image to run in
<code>--group NAME</code>	The <code>chap-devops run</code> group whose models chap can reach
<code>--stop</code>	Stop the model that this run started, after the run
<code>--timeout SECONDS</code>	How long to wait for a model that chap-devops starts (300)
<code>--docker</code>	Give the container the docker socket

For example: `chap-devops chap --tag master --docker eval ...`. `chap-devops chap -help`

shows these options, and `chap-devops chap` alone shows chap's own help.

Under the hood: one `docker run`

```
docker run --rm -i [-t] --init --platform linux/amd64
  --user <your uid>:<your gid>
  -v <cwd>:<cwd> -w <cwd>
  -v <data>/chap:<data>/chap
  -e HOME=/tmp -e CHAP_RUNS_DIR=<data>/chap/runs ...
  [--network <deployment>_default]
  --label com.winterop.chap-devops.kind=cli
  ghcr.io/dhis2-chap/chap-worker:v2.5.0 chap eval ...
```

- `--rm`: the container goes when chap stops. `--init` passes Ctrl-C on.
- `--platform linux/amd64`: the chap-core images are amd64. On Apple silicon, they run under emulation.
- `-t` only when your terminal is a terminal, so a pipe works too.

The same path, inside and outside

`-v <cwd>:<cwd> -w <cwd>` mounts your directory **at the same path**:

- `--dataset-csv laos_subset.csv` reads your file, and finds the `.geojson` beside it, as before.
- `--output-file r.nc` writes into your directory.
- An absolute path in the output means the same file on both sides.

`--user <uid>:<gid>` runs chap as you, so the files belong to you, not to root.

The caches: the second run is faster

`<data>` is chap-devops's data directory: `~/.local/share/chap-devops`, or `$CHAP_DEVOPS_DATA_DIR`. chap-devops sets:

Variable	Value	What it keeps
<code>CHAP_RUNS_DIR</code>	<code><data>/chap/runs</code>	one directory for each model
<code>UV_CACHE_DIR</code>	<code><data>/chap/cache/uv</code>	Python packages
<code>UV_PYTHON_INSTALL_DIR</code>	<code><data>/chap/cache/python</code>	Python versions
<code>RENV_PATHS_ROOT</code>	<code><data>/chap/cache/renv</code>	R packages
<code>RENV_CONFIG_SANDBOX_ENABLED</code>	<code>FALSE</code>	no read-only sandbox
<code>MPLCONFIGDIR</code>	<code><data>/chap/cache/matplotlib</code>	the font cache

With the sandbox off, `rm -rf ~/.local/share/chap-devops/chap` deletes all of it.

Which image

<code>--model-name</code>	Image	Size on disk
a GitHub repository, a model directory	<code>chap-worker</code>	about 12 GB
a chapkit URL, a model id	the image on this machine; else <code>chap-core</code>	
<code>builtin:NAME</code>	the image on this machine; else <code>chap-core</code>	
no model (<code>plot-backtest</code> , <code>export-metrics</code>)	<code>chap-core</code>	about 2.4 GB
anything, in a deployment	<code>chap-worker</code>	already there

- The worker image has uv, R and INLA. chap-core's own worker runs models in it.
- `--image core` or `--image worker` overrides the choice.
- `--image core` with a GitHub model gets a warning first: an R model fails there with `Rscript: not found`.

Which chap-core version

1. `--tag TAG` , when you give it.
2. In a deployment: **the deployment's tag**, so the CLI and the server are the same version.
3. Outside a deployment: **the newest chap-core release**, from GitHub.
4. With `--offline`: the newest image of that kind on this machine.

If GitHub does not answer, chap-devops uses the newest local image, and says so:

```
warning: the newest chap-core release could not be read; using v2.3.1, the
newest image on this machine
```

The lines before the run

```
running `chap eval` in ghcr.io/dhis2-chap/chap-worker:v2.5.0; the first run pulls it, about 12 GB  
files: chap reads and writes in /home/me/work  
model: auto_arima_chapkit at http://auto-arima-chapkit:8000 answers
```

- The image and its tag, and the size when it is not local yet.
- The directories that chap can see.
- The model, when chap-devops checked that it answers.

A run that only asks for help (`chap-devops chap, --help, --version`) gets only the first line.

Models: five kinds

<code>--model-name</code>	What happens
<code>https://github.com/...</code>	chap clones it, and runs it in the worker image
<code>./my_model</code> (a directory)	chap runs it from your directory
<code>http://...</code> (a chapkit service)	chap-devops checks that it answers, then chap calls it
<code>auto_arima_chapkit</code> (an id)	chap-devops starts the model, then chap calls it
<code>builtin:seasonal_median</code>	chap runs it itself; chap-devops starts nothing

A word is a model id only when no file or directory of that name is in the current directory. A directory of the same name wins, as it does for chap.

A model id: chap-devops starts the model

Outside a deployment, chap-devops starts the model in a `chap-devops run` group:

```
starting auto-arma-chapkit (auto_arma_chapkit in ~/.local/share/chap-devops/run/default)
running `chap eval` in ghcr.io/dhis2-chap/chap-core:v2.5.0
model: auto_arma_chapkit at http://auto-arma-chapkit:8000 answers
chap finished; it wrote arma.nc
auto_arma_chapkit keeps running for the next run; `chap-devops stop auto_arma_chapkit`
stops it, and `--stop` stops it after a run
```

The container joins the group's network, where the model has its service name. chap-devops gives chap that URL in place of the id.

A model id in a deployment: no `chap-devops up`

```
cd ~/mychap && mkdir -p eval && cd eval  
chap-devops chap eval --model-name auto_arima_chapkit \  
  --dataset-csv laos_subset.csv --output-file arima.nc
```

starting auto-arima-chapkit for this run, without chap-core

- chap-devops starts **only that model**, with `--no-deps`: chap-core, postgres and redis stay down.
- For this start, the model gets no registration settings, so it does not wait for a chap-core that does not run.
- A later `chap-devops up` recreates the model, and it registers as usual.

Keep it running, or stop it

Situation	After the run
chap-devops started the model	It keeps running, so the next run starts at once
chap-devops started it, with <code>--stop</code>	chap-devops stops it
the model ran before	chap-devops never stops it

```
auto_arima_chapkit ran before this run, so --stop left it running
```

`--timeout SECONDS` (300) is how long chap-devops waits for a model that it starts. A first start pulls the model's image (1 to 7 GB), and chap-devops shows that pull.

A chapkit URL: checked before chap starts

chap-devops asks the URL for `/api/v1/info`, from a container on the same network. If nothing answers, chap is not started:

```
error: the model server at http://nothing:8000 is not running (no answer on
/api/v1/info), so chap was not started; `nothing` is not a model of this
deployment or group. Give a model id instead (`--model-name
auto_arima_chapkit`) and chap-devops starts it, or start one with `chap-devops run ID`;
the `models:` line lists the URLs that answer
```

For a server elsewhere, the message ends with "check that the server runs and that this machine can reach it".

Not localhost

`chap-devops ps` shows a model at `http://localhost:5001`. That URL works on your machine. In the container, `localhost` is the container itself:

```
error: `http://localhost:5001` is this machine, and in the container
`localhost` is the container itself; use `--model-name
http://auto-arma-chapkit:8000`, the same model on the network
```

- When `chap-devops` knows which model has that port, the message names its service URL.
- For a service on this machine, use `http://host.docker.internal:PORT`. In the container, that name is this machine.

Models that run in docker

A model with `docker_env` in its `MLproject` starts a container of its own.
chap-devops reads the `MLproject` first (from GitHub, or from your directory):

```
error: `https://github.com/dhis2-chap/chap_auto_ewars` runs in docker  
(`docker_env` in its MLproject), and the container has no docker socket;  
run it again with `chap-devops chap --docker ...`
```

```
chap-devops chap --docker eval --model-name https://github.com/dhis2-chap/chap_auto_ewars \  
--dataset-csv laos_subset.csv --output-file ewars.nc
```

What `--docker` does

- It mounts `/var/run/docker.sock` in the container. Docker Desktop, Colima and OrbStack all answer that path.
- It adds the socket's group to the container user, as the container sees it (on Docker Desktop, group 0).
- The run directories are at the same path on both sides, so the paths that chap gives docker are real paths on your machine.

```
warning: --docker gives the container control of docker on this machine
```

The socket is full control of docker. So `--docker` is never the default.

Built-in models

From chap-core v2.5.0, chap has baselines of its own. Give them with the `builtin:` prefix:

```
chap-devops chap eval --model-name builtin:seasonal_median \  
  --dataset-csv laos_subset.csv --output-file seasonal_median.nc
```

- The names are `persistence`, `global_median` and `seasonal_median`.
- chap runs them itself, so chap-devops starts no model, and either image works.
- An older tag does not have them, and chap-devops refuses them there.

An MLproject as a chapkit service

From chap-core v2.5.0, `--run-config.as-chapkit` starts a model directory or a GitHub repository as a chapkit service in the container:

```
chap-devops chap eval --model-name ./my_model --run-config.as-chapkit \  
  --dataset-csv laos_subset.csv --output-file my_model.nc
```

- The `MLproject` can use `uv_env`, `renv_env` or no environment.
- chap refuses `docker_env`, `conda_env` and `python_env` in this mode. So chap-devops does not ask for `--docker`.
- An `MLproject` from `chapkit mlproject export` needs this mode.

Your files: what chap-devops does for you

- **Output directories:** chap-devops makes the directory of each `--output-file` that does not exist. chap does not, and it fails with a traceback.
- **Paths outside the directory:** `../out/r.nc`, `/data/r.nc` or `--output-file=/data/r.nc`. chap-devops mounts the nearest existing directory above the path, at the same path.
- **--out-file** (`chap simulate`): chap-devops makes its directory too.
- **The deployment directory:** if you run `chap-devops chap` there, chap-devops tells you that a subdirectory (`mkdir eval && cd eval`) keeps chap's files apart from the compose files.

What chap-devops reports after the run

```
chap finished; it wrote arima.nc  
`chap-devops chap plot-backtest arima.nc --output-file arima.html` plots it
```

- chap-devops looks at the mounted directories before and after the run, and names the files that are new or changed.
- It also names the files that the arguments name, wherever they are.
- For a `.nc` file, it gives the command that plots it.
- When chap fails, chap-devops exits with chap's own status:

```
error: chap exited with status 1; its own message is above
```

Every message that stops a run

chap-devops says	What to do
<code>--json</code> does not apply	Drop <code>--json</code> : the output is chap's own
<code>`./x`</code> is not a directory here	Give a directory, a GitHub URL or a model id
is not a model of the deployment	<code>chap-devops models enable ID</code> , then run again
there is no marketplace model	<code>chap-devops models search WORD</code> finds the id
there is no <code>`chap-devops run`</code> group called	<code>chap-devops ps</code> lists the groups
no directory above it exists to mount	Make the directory first
<code>--offline</code> needs a ... image on this machine	Run once without <code>--offline</code> , or give <code>--tag</code>
<code>`chap plot-dataset`</code> shows its plot in a browser	Use <code>plot-backtest</code> , which writes a file

chap's commands in a container

Command	In <code>chap-devops</code> <code>chap</code>
<code>eval</code>	yes, with every kind of model
<code>plot-backtest</code>	yes, it writes a file
<code>export-metrics</code>	yes
<code>validate</code>	yes
<code>test</code>	yes
<code>simulate</code>	yes, from v2.5.0: it writes a CSV with <code>--out-file</code>
<code>plot-dataset</code>	no : it opens a browser, and a container has none

chap-devops refuses `plot-dataset` before the run and names `plot-backtest`.

A full example: three models, compared (1)

Get the data, and check it:

```
mkdir -p ~/compare && cd ~/compare  
curl -fsS0 https://raw.githubusercontent.com/dhis2-chap/chap-core/master/example_data/laos_subset.csv  
curl -fsS0 https://raw.githubusercontent.com/dhis2-chap/chap-core/master/example_data/laos_subset.geojson  
chap-devops chap validate laos_subset.csv
```

```
Validation passed: no issues found.
```

A full example: three models, compared (2)

```
chap-devops chap eval --model-name https://github.com/dhis2-chap/minimalist_example_r \
  --dataset-csv laos_subset.csv --output-file out/r.nc \
  --backtest-params.n-splits 3 --backtest-params.n-periods 3
chap-devops chap eval --model-name auto_arima_chapkit \
  --dataset-csv laos_subset.csv --output-file out/arima.nc \
  --backtest-params.n-splits 3 --backtest-params.n-periods 3
chap-devops chap --stop eval --model-name chapkit_simple_multistep_model \
  --model-configuration-yaml multistep.yaml \
  --dataset-csv laos_subset.csv --output-file out/multistep.nc \
  --backtest-params.n-splits 3 --backtest-params.n-periods 3
```

chap-devops makes `out/` with the first run. The third run stops its model after. The data has no humidity, so `multistep.yaml` gives Simple Multistep only `additional_continuous_covariates: [rainfall, mean_temperature]`.

A full example: three models, compared (3)

```
chap-devops chap export-metrics out/r.nc out/arma.nc out/multistep.nc \
  --output-file out/comparison.csv
chap-devops chap plot-backtest out/arma.nc --output-file out/arma.html
chap-devops stop auto_arma_chapkit
```

- `comparison.csv` has one row for each model: `crps`, `mae`, `rmse`, `coverage_10_90` and more. Lower is better for the errors.
- `chap` can print `RuntimeWarning` lines about `log1p`. They do not stop the run.

Troubleshooting

You see	Cause and fix
<code>Cannot reach the Docker daemon. The model ... requires Docker</code>	A <code>docker_env</code> model ran without the socket. Add <code>--docker</code> .
<code>Rscript: not found</code>	An R model in the <code>chap-core</code> image. Drop <code>--image core</code> .
<code>FileNotFoundError: [Errno 2]</code>	An output directory that chap-devops could not make. Make it, then run again.
<code>did not answer on http://...:8000 within 300s</code>	A slow first start. Give a longer <code>--timeout</code> .
<code>docker run exited with status 125</code>	The image did not pull. Check the tag, or run <code>chap-devops doctor</code> .

Each message is in the book's Troubleshooting chapter, with more detail.

Start now

```
curl -fsSL https://raw.githubusercontent.com/winterop-com/chap-devops/main/install.sh | sh
chap-devops doctor
chap-devops chap
```

- The full guide, from install to a comparison of two models:
<https://winterop-com.github.io/chap-devops/use-cases/evaluate-with-chap-cli.html>
- The chapter about `chap-devops chap`:
<https://winterop-com.github.io/chap-devops/chap-cli.html>
- A model in the MLproject format: `chap-devops chapkit mlproject run DIR` serves it with no build. See the **MLproject models** deck.