

CHAP-DEVOPS

Running models

`chap-devops run`: one command starts a model and gives you its URL

No deployment directory, no compose file, no chap-core

Why `chap-devops run`

A forecasting model is a service: a container with an HTTP API. Often you want only that service:

- to call the model's API from a notebook or a script
- to evaluate it with `chap-devops chap eval`
- to try a model before you add it to a deployment
- to run two or three models side by side and compare them

`chap-devops init` and `chap-devops up` make a full deployment. `chap-devops run` makes nothing that you must keep.

The first model

```
chap-devops -v run chapkit_simple_multistep_model
```

```
starting chapkit-simple-multistep-model  
  (chapkit_simple_multistep_model in /home/me/.local/share/chap-devops/run/default)  
running chapkit_simple_multistep_model on http://localhost:5001 (answered in 4s)  
hint: `chap-devops stop chapkit_simple_multistep_model` stops it  
hint: `chap-devops -C /home/me/.local/share/chap-devops/run/default  
      logs chapkit-simple-multistep-model` shows its log
```

- The command returns when the model answers its own `/health`.
- So the URL works when the line is printed. The model keeps running.

In the foreground: **-a**

-a (also **--attach** or **--foreground**, as on **chap-devops up**) stays in the foreground with the model's log. Ctrl-C stops the model:

```
following the log of chapkit-simple-multistep-model; Ctrl-C stops it
...
stopped chapkit_simple_multistep_model; its data stays,
and `chap-devops run chapkit_simple_multistep_model` starts it again with it
```

- The model goes out of its group, as with **chap-devops stop**. Its data stays.
- **--rm**: Ctrl-C removes the data volume too.
- A model that ran before the command keeps running. Ctrl-C stops only the log.

Ctrl-C while the model starts

Ctrl-C during the pull or the wait takes back out what the start put in:

```
error: stopped by Ctrl-C; chapkit_simple_multistep_model is taken back out of  
/home/me/.local/share/chap-devops/run/default
```

- The model that this run enabled goes back out of the group, as after a failed start. No container is left.
- The exit code is 130, as a shell gives for Ctrl-C.

What the model can be

MODEL	What <code>chap-devops run</code> does
a marketplace id: <code>chapkit_simple_multistep_model</code>	enables it at the catalogue's stable version
a GitHub repository the marketplace lists	the same: the catalogue's reviewed pin
any other GitHub repository URL	adds it: the newest published <code>sha-</code> build of the default branch
a ghcr image with a tag or digest	pins exactly that image
a local image: <code>name:tag</code>	runs the image from your image store, never pulled

A word with no `:`, `/` or `@` is a marketplace id, never a local image.

Four ways to start a model

```
chap-devops run chapkit_simple_multistep_model
chap-devops run https://github.com/chap-models/chapkit_ghr_model
chap-devops run ghcr.io/my-org/chapkit_dengue_model:sha-1eb8cf1
chap-devops run my-dengue-model:dev
```

- A first run downloads the model's image: 1 GB for a Python model, 6 to 7 GB for an R model.
- A second `chap-devops run` of the same model starts it as it is. It does not move its version or its port.

An MLproject directory: `chap-devops chapkit mlproject` `run`

A model in the MLproject format is a directory, not an image. Serve it with no build:

```
chap-devops chapkit mlproject run rain-model
```

```
serving /home/me/rain-model with `chapkit mlproject run`  
  in ghcr.io/dhis2-chap/chapkit-r-tidyverse-cli:latest (...)  
starting it on http://localhost:5001; the logs follow, and Ctrl-C stops it  
the MLproject service answers at http://localhost:5001
```

- It runs in the foreground, with the logs. Ctrl-C stops it.
- See the **MLproject models** deck.

Where the model runs: groups

Outside a deployment, `chap-devops run` keeps its models in a **group**:

- a deployment that chap-devops owns, in `~/.local/share/chap-devops/run/<group>/`
- with no chap-core and no component: only model services
- the first `chap-devops run` into a group creates it
- `--group NAME` selects a group; the default group is `default`

Each group is a compose project of its own. Its containers, volumes and network carry its name, so a stop in one group touches nothing else.

Two groups side by side

```
chap-devops run chapkit_simple_multistep_model # group default
chap-devops run auto_arima_chapkit --group trial
chap-devops run chapkit_ghr_model --group trial
chap-devops ps
```

GROUP	ID	SERVICE	STATE	URL
default	chapkit_simple_multistep_model	chapkit-simple-multistep-model	up	http://localhost:5001
trial	auto_arima_chapkit	auto-arima-chapkit	up	http://localhost:5002
trial	chapkit_ghr_model	chapkit-ghr-model	not running	http://localhost:5003

hint: groups live in /home/me/.local/share/chap-devops/run

Ports

- `--port auto`, the default, takes the lowest port in **5001-5999** that nothing on the machine holds and that no group gave to a model.
- So two groups never collide, also with models that are stopped.
- `--port 5050` asks for one port. A port that is in use is refused, with the fix.

Two `chap-devops run` commands at the same time wait for each other, and get different ports.

Only this machine

- A group publishes every model port on **127.0.0.1**.
- So a model that `chap-devops run` started is reachable from this machine, not from the network. A model has no login.
- `--bind 0.0.0.0` publishes one model on every address of the machine.

```
chap-devops run chapkit_simple_multistep_model --bind 0.0.0.0
```

Do that only on a network that you trust.

The states of a model

`chap-devops ps` gives each model the state that `chap-devops status` gives:

State	What it means
<code>up</code>	its <code>/health</code> answers
<code>running, not answering</code>	the container runs, and <code>/health</code> does not answer yet, or never will
<code>not running</code>	the container is gone

`chap-devops ps --group trial` lists one group.

Waiting, and when a start fails

- `chap-devops run` waits up to **300 seconds** for the model to answer.
`--timeout SECONDS` changes that. A first R model can take some minutes.
- `--no-wait` returns as soon as the container started.
- When the container does not start, chap-devops says what compose said:

```
error: chapkit-simple-multistep-model did not start (Error response from daemon:
pull access denied ...); fix that, then
`chap-devops run chapkit_simple_multistep_model` tries again
```

- chap-devops then takes the model out again, so `chap-devops ps` does not list it.

Stopping

```
chap-devops stop chapkit_simple_multistep_model # in whichever group has it
chap-devops stop --group trial                  # every model in one group
chap-devops stop --all                          # every model in every group
```

- The model's data volume stays. `chap-devops run ID` starts it again.
- When two groups have the same model, `chap-devops stop` asks for `--group`.

Stopping, and removing the data

```
chap-devops stop chapkit_simple_multistep_model --purge
chap-devops stop --group trial --purge
chap-devops stop --all --purge
```

- `--purge` also removes the data volume.
- A group that has no model left goes too: its network, its directory and every volume of it.
- A volume that docker will not remove keeps its group. Run the same command again when the volume is free.

Your own chap-core: `--chap-core`

A group has no chap-core, so its models register nowhere. Give it yours:

```
chap-devops run -a --rm --chap-core http://localhost:8000 \  
https://github.com/chap-models/chapkit_ghr_model
```

```
running chapkit_ghr_model on http://localhost:5001 (answered in 2s)  
registered with http://localhost:8000
```

- Every model of the group registers there. `--group NAME` keeps them apart.
- chap-devops finds out whether chap-core is a container or a process, and where it must call the models back. `--models-host` sets it.

A group is a deployment

Every other chap-devops command works on a group with `-C`:

```
chap-devops -C ~/.local/share/chap-devops/run/default logs chapkit-simple-multistep-model  
chap-devops -C ~/.local/share/chap-devops/run/default status  
chap-devops -C ~/.local/share/chap-devops/run/default models test chapkit_simple_multistep_model
```

The messages from a group name their commands in that way, so you can copy them.

In a deployment of your own

- In a deployment directory (or with `-C DIR`), `chap-devops run`, `chap-devops ps` and `chap-devops stop` work on that deployment.
- `--group` is then refused: the deployment is the group.
- `chap-devops run` publishes the model as the deployment does.
- With chap-core in the deployment, `chap-devops models configs` lists what chap-core can run.

So a tool that only wants "start this model and give me its URL" has one code path, with or without `chap-devops init`.

Watching everything: chap-devops top

```
chap-devops top    3 deployments  7/9 running  at 14:02:41 (every 2s)
|▼ mychap  init          5/6 running
|  └─ chap              running (healthy)  1.20%   612MiB / 15.6GiB
|    └─ auto-arma-chapkit exited
|▼ run/default  run      1/1 running
|  └─ chapkit-ghr-model  running          0.50%   230MiB / 15.6GiB
|▶ run/trial    run      1/2 running
j/k move  enter fold  l logs  s stop model  r refresh  q quit
```

Every deployment and every group on the machine, live. `l` shows a log, and `s` stops the model under the cursor.

Evaluate a model that runs

A model that `chap-devops run` started is on its group's network. `chap-devops chap` joins that network:

```
chap-devops chap eval --model-name auto_arima_chapkit \
  --dataset-csv laos_subset.csv --output-file arima.nc
```

- `chap` reaches the model at `http://auto-arima-chapkit:8000`.
- Do not give `chap` the `http://localhost:5001` URL. In `chap`'s container, `localhost` is that container.

Or let `chap-devops chap` start it

Give `chap-devops chap` a marketplace id of a model that does not run. `chap-devops` starts it in a group, as `chap-devops run` does:

```
starting auto-arma-chapkit (auto_arma_chapkit in /home/me/.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
```

Call the model's API yourself

The URL from `chap-devops run` is the model's own API:

```
curl -s http://localhost:5001/health  
curl -s http://localhost:5001/api/v1/info
```

- `/health` answers when the model is ready.
- `/api/v1/info` says what the model is, and which data it needs.
- The model's API documentation is at `/docs`: open <http://localhost:5001/docs> in a browser.

For tools: `--json`

```
chap-devops --json run chapkit_simple_multistep_model
```

```
{
  "bind": "127.0.0.1",
  "enabled": true,
  "group": "default",
  "id": "chapkit_simple_multistep_model",
  "messages": [...],
  "ok": true,
  "port": 5001,
  "project_dir": "/home/me/.local/share/chap-devops/run/default",
  "service_id": "chapkit-simple-multistep-model",
  "url": "http://localhost:5001",
  "wait": {"ready": true, "waited_s": 4, "models": [...]}
}
```

`--json` for `ps` and `stop`

- `chap-devops ps --json: {"models": [...]}`, with the same fields for each model, plus `state`.
- `chap-devops stop --json`: the models it `stopped`, the groups it `removed`, and the volumes that went with them, `removed_volumes`.
- A failure: `{"ok": false, "error": ..., "hint": ...}`.
- docker's own progress, such as a pull, goes to `stderr`. `stdout` is the one JSON document.

Where the files are

```
~/.local/share/chap-devops/  
  run/  
    default/          the default group: a chap-devops deployment  
      .chap-devops/    its state: project.yaml, models.yaml  
      compose.*.yaml  its compose files, rendered by chap-devops  
    trial/            the group `trial`  
    chap/             the caches of `chap-devops chap`
```

`$CHAP_DEVOPS_DATA_DIR`, else `$XDG_DATA_HOME/chap-devops`, else `~/.local/share/chap-devops`.

Summary

You want	Command
a model and its URL	<code>chap-devops run ID</code>
a model in the foreground, with its log	<code>chap-devops run ID -a</code>
the models that run, or a live view	<code>chap-devops ps</code> , or <code>chap-devops top</code>
its log	<code>chap-devops -C ~/.local/share/chap-devops/run/default logs SERVICE</code>
to evaluate it	<code>chap-devops chap eval --model-name ID ...</code>
an MLproject directory as a service	<code>chap-devops chapkit mlproject run DIR</code>
to register it with your chap-core	<code>chap-devops run ID --chap-core URL</code>
to stop it	<code>chap-devops stop ID</code> , or Ctrl-C with <code>-a</code>
to remove it and its data	<code>chap-devops stop ID --purge</code>