

CHAP-DEVOPS

Chap behind the Modeling App

A DHIS2, the apps and Chap, connected with one command

For people who use DHIS2 and the Modeling App

What you see, and what runs behind it

In DHIS2:

- the Modeling App: evaluate a model, make a forecast
- the Climate App: climate data for your org units
- analytics: the data that the apps read

Behind it:

- **chap-core**: runs the evaluations and forecasts
- **models**: one container each (Simple Multistep, Auto-ARIMA, ...)
- **OCS**: climate data, if you want it

The Modeling App reaches chap-core through a **route** in DHIS2: DHIS2 sends each request on to chap-core. chap-core never calls DHIS2.

Three things must be correct

Without it	What the Modeling App does
the <code>chap</code> route	sends you to "get started"; it cannot see Chap
the analytics tables	has no data to send, and says nothing about why
the apps	there is no Chap screen in DHIS2

Before chap-devops, you did each of these by hand: a route JSON, the App Hub, an analytics run.

A DHIS2 with Chap, from nothing

```
chap-devops init mychap --with dhis2 --models chapkit_simple_multistep_model
cd mychap
chap-devops up
chap-devops dhis2 connect
```

- `--with dhis2` adds a DHIS2, with the Laos climate demo (DHIS2 2.42).
- `chap-devops dhis2 connect` makes the route, installs the apps and generates analytics.

Open <http://localhost:8780> and log in as `admin` / `district`.

The first `chap-devops up` downloads about 25 GB of images: chap-core, the model and DHIS2. Do it on a fast network.

What --with dhis2 adds

Service	What it is
dhis2	DHIS2 itself, from <code>dhis2/core</code> , on port 8780
dhis2-db	its own PostgreSQL. Never published on the host.
dhis2-dump	a one-shot that prepares the demo dump, and exits
dhis2-prep	a one-shot on every start: <code>ANALYZE</code> , and it resets stale jobs

- Its own database, apart from chap-core's: the two never share data.
- Three volumes: `dhis2_home` (apps and files), `dhis2_db` , and `dhis2_dump` (a download cache).

The first start takes minutes

DHIS2 migrates its schema before it answers. The demo database is restored before that.

Shape	About
A native image, empty database	40 seconds
Under emulation (an amd64 image on arm64)	8 to 15 minutes
With the demo database	the above, plus the restore

- `chap-devops status` shows each service. `chap-devops logs dhis2` shows the migration.
- `chap-devops dhis2` commands wait for DHIS2's API, up to 20 minutes by default.

Memory

- DHIS2 needs about **4 to 5 GB** for the analytics populate phase.
- Below that, the JVM is killed part-way through analytics. The run never finishes, and the log shows no Java error.
- `chap-devops doctor` warns below 6 GB for docker, and fails below 4 GB.
- On Docker Desktop, raise the VM's memory: Settings, Resources.
- With OCS on 8 GB or less: set `CLIMATE_SERVICE_MAX_CONCURRENT_JOBS=1` in `.env`, then run `chap-devops restart ocs`.

```
# DHIS2_JAVA_TOOL_OPTIONS=-Xms2g -Xmx4g -XX:+UseG1GC
```

That line in `.env` sets the heap. Uncomment it and edit the whole set.

Why `chap-devops up` does not connect

`chap-devops up` is a thin wrapper around `docker compose up`. Connecting is the wrong work for it:

- it needs **DHIS2 credentials**, which are not `chap-devops`'s to invent;
- it uses the **network**, the App Hub, and `chap-devops up` never does;
- DHIS2's API is **not ready when `up` returns**.

So `chap-devops up` and `chap-devops status` say it until a connect is recorded:

```
chap-devops has not connected this DHIS2 to Chap; run `chap-devops dhis2 connect` once DHIS2 answers
```

chap-devops dh1s2 connect , step by step

1. **The route.** It creates the chap route, or repoints one that points elsewhere. Then it sends a request through it, to prove that chap-core answers.
2. **The apps.** It installs the Modeling App and the Climate App from the App Hub, at the newest version that this DHIS2 can run.
3. **Analytics.** It generates the analytics tables, and waits for the run.

Each step checks first, and does nothing when it is correct already. A second run repoints nothing and reinstalls nothing, and says so.

The `chap` route

- A **Route** is a row in DHIS2 with `code: "chap"`. DHIS2 serves it under `/api/routes/chap/run/` and passes each request to chap-core.
- The target is `http://chap:8000/**`: the compose service name, never `localhost`. The `/**` makes DHIS2 pass every path under it.
- The route needs the `F_CHAP_MODELING_APP` authority.
- With an API token, the route carries `Authorization: Bearer`. DHIS2 stores it and never lists it back.

The route is repointed, not created

A seeded demo database **already has a chap route**, and it points at someone else's Chap. So chap-devops compares, and rewrites:

```
repointed the `chap` route at http://chap:8000/**  
it pointed at http://158.39.75.126/stable/**  
verified chap-core answered through it: healthy
```

It leaves the route alone only when the URL is this chap-core, the route is not disabled, and it has the **F_CHAP_MODELING_APP** authority.

When the route does not answer

The route is proved with `GET /api/routes/chap/run/health`, the same path as the app uses:

```
warning: the `chap` route is in place but nothing answered through it:  
HTTP 502 Bad Gateway; run `chap-devops status` to see whether chap-core is up
```

DHIS2 42 and later allow only the targets in

`route.remote_servers_allowed`. chap-devops writes `http://*,https://*` into `dhis2/dhis.conf`, so the route to `http://chap:8000` is allowed.

Analytics

The Modeling App and the Climate App read DHIS2's `analytics_*` tables. The tables exist only after an analytics run.

```
chap-devops dhis2 analytics          # start the run, and wait for it
chap-devops dhis2 analytics --no-wait # start it, and return
```

- A run that goes already is watched, not queued behind: DHIS2 runs one at a time.
- **No `lastYears`**. On the Laos demo, `lastYears=8` wrote zero rows and reported success. chap-devops never sends it.

Is analytics done? What `show` can say

On a seeded database, DHIS2's "last analytics" time comes from the dump, not from this deployment. So `chap-devops dhisis2 show` labels it:

Row	What chap-devops checked
<code>never run</code>	DHIS2 records no run at all
<code>(a run finished on this deployment)</code>	a run finished since DHIS2 started
<code>(unconfirmed on a seeded database)</code>	the time can be the dump's

When it is unconfirmed, `chap-devops dhisis2 analytics` settles it. A second run is safe.

The apps, from the App Hub

App	What it is for
Modeling App	the Chap user interface in DHIS2
DHIS2 Climate App	imports climate data through Google Earth Engine: the Modeling App's covariates

- DHIS2 downloads the app itself, from the App Hub. chap-devops only asks.
- chap-devops selects the newest version that this DHIS2 can run.
- An app at that version is left alone. An app at another version is moved.

The demo database: the seed

<code>--dhis2-seed</code>	What happens
<code>default</code>	the demo dump for the pinned line: the Laos climate demo on 2.42
<code>none</code>	an empty database, migrated from nothing
a URL	that dump, downloaded
a path	that file, in the deployment directory

The seed applies **once**, when the database is created. A later `chap-devops up` does not restore it again.

Which DHIS2 version

```
chap-devops init mychap --with dhis2 --dhis2-tag 2.41
chap-devops init mychap --with dhis2 --dhis2-tag 2.43 --dhis2-seed none
chap-devops init mychap --with dhis2 --dhis2-image dhis2/core-dev --dhis2-tag master
chap-devops components enable dhis2 --tag 2.41          # later
```

- The tag is a minor line (2.42), so a patch release comes with a pull.
- `dhis2/core-dev` with `master` is the next, unreleased DHIS2. It has no demo database, so it starts empty.

Forward only: back up first

DHIS2 migrates its schema **forward only**:

- a **newer** image migrates the database when it starts, and you cannot undo it;
- an **older** image on a migrated database answers its health check, but every API request gives 404.

```
chap-devops backup create
chap-devops components enable dhis2 --tag 2.43
chap-devops up
```

chap-devops warns when a change moves the tag while `dhis2_db` exists.

dhis2/dhis.conf

- DHIS2 does **not start without it**. chap-devops writes it once, and never rewrites it. It is yours.
- Its values come from the environment: the database host, name, user and password, and `DHIS2_ENCRYPTION_PASSWORD`.
- The encryption password is 32 random characters in `.env`, written once. Changing it later makes the stored credentials unreadable.

After an edit:

```
chap-devops restart dhis2
```

`chap-devops restart` sees that the file is newer than the container, and recreates it.

The DHIS2 login that chap-devops uses

The first source with a value wins:

Order	Source
1	DHIS2_API_TOKEN in .env : a personal access token
2	DHIS2_ADMIN_PASSWORD in .env , for DHIS2_ADMIN_USERNAME
3	CHAP_DEVOPS_DHIS2_TOKEN in the environment
4	CHAP_DEVOPS_DHIS2_PASSWORD in the environment
5	seed_password: in .chap-devops/components.yaml , only on a DHIS2 that chap-devops deployed
6	admin / district , only on a DHIS2 that chap-devops deployed

There is **no --password flag**: a password on a command line goes into the shell history.

A DHIS2 that runs elsewhere

```
chap-devops init chap-server --models chapkit_simple_multistep_model --api-token  
cd chap-server && chap-devops up  
chap-devops dhis2 use https://dhis2.example.org --chap-url https://chap.example.org  
chap-devops dhis2 connect
```

- `use` records both URLs in `.chap-devops/components.yaml`.
- Then it asks DHIS2 two things: does `/api/ping` answer, and does it accept the login.
- An external DHIS2 has **no default login**. A personal access token is the natural choice.

What changes with an external DHIS2

- **connect** sets the route, and nothing else. Apps and analytics change a server that chap-devops does not run, so they stay with its admin. **connect** ends with a **skipped:** line for each.
- **Nothing asks docker:** there is no container.
- **The allowlist is that server's:** `route.remote_servers_allowed` in its own `dhis.conf`.
- It is one DHIS2 or the other: **use** refuses while the **dhis2** component is on.

`chap-devops dhis2 show` and `connected_at`

- `chap-devops dhis2 show` asks and reports: where the route points and whether chap-core answers through it, analytics, the two apps, and each missing piece. It changes nothing.
- After a connect that proved the route, chap-devops records `connected_at` in `.chap-devops/components.yaml`.
- `connected_at` only stops the hint in `up` and `status`. It is **not** proof that the route is correct today. `chap-devops dhis2 show` is what asks DHIS2.

Climate data from OCS

```
chap-devops components enable ocs
chap-devops up
chap-devops open ocs
```

- Open Climate Service serves climate data over STAC and openEO, on port 8790.
- WorldPop and CHIRPS3 work with no credentials. ERA5-Land needs one in `.env`.
- `ocs/climate-service.yaml` is the instance's configuration. Edit it: it ships with example values.

OCS: schedules

OCS can sync a dataset again on a schedule. The file only turns the scheduler on:

```
scheduler:  
  enabled: true
```

- Make the schedules in the Schedule tab of a dataset in the OCS web UI.
- Or make them with `POST /schedules/sync`. `chap-devops backup` keeps them.
- An old `dataset_sync:` list stops OCS with `scheduler.dataset_sync` is no longer supported. Remove the list, run `chap-devops restart ocs`, and make the schedules again.

More models

```
chap-devops models list           # the marketplace
chap-devops models enable auto_arima_chapkit # one more model
chap-devops up
```

- A new model registers with chap-core, and the Modeling App lists it.
- `chap-devops status` shows the registration of each model.
- `chap-devops models test --all --backtest` runs a backtest of each model through chap-core, the way the Modeling App does.

What the Modeling App lists

The model list shows **configured models**, not model services:

```
<model name> [<variant name>]
```

- A configured model is a model with a set of option values and covariates.
- One model service can give several entries: Simple Multistep gives two.
- `chap-devops models configs` shows the same list in the terminal, with the option values and covariates of each entry.
- From chap-core v2.5.0, the list also has the built-in baselines:
Persistence, **Global median** and **Seasonal median**.

A forecast back into DHIS2: Import

On a prediction run, **Import** writes the forecast into DHIS2 data elements.

- Five boxes, one per quantile. Each suggests its CHAP ... Quantile ... element of the demo database.
- From Modeling App 7.3.0, **Use alert outputs** is on by default. Then **Clear and import** works only with an **Outbreak indicator**.
- Pick CHAP Dengue Cases (Any) - Weekly Outbreak Alert : type Outbreak , because the box does not suggest it. Or turn **Use alert outputs** off.

Where the entries come from

chap-core 2.4 and later makes no configured model when a model registers.

chap-devops makes them, one for each configuration of the marketplace entry:

```
chap-devops models configs sync
```

```
chapkit_simple_multistep_model: created configured model monthly_climate  
chapkit_simple_multistep_model: created configured model monthly_selfhistory
```

`chap-devops up --wait` and `chap-devops dhis2 connect` do the same step. A model without entries shows in `chap-devops status` as `registered, not configured`.

A variant of your own

```
chap-devops models configs add chapkit_simple_multistep_model \  
  --name short_lags --set n_target_lags=3 --set rf_max_depth=5 \  
  --covariates rainfall
```

```
created configured model short_lags of chapkit_simple_multistep_model
```

- `--name` is the variant name: the Modeling App shows it in brackets.
- `chap-devops` checks each key and each value against the options of the model.
- With no options at a terminal, `add` opens a form: one field for each option, with its default.
- `chap-devops models test ID --backtest --config short_lags` runs a backtest of it.

Change, archive, export

Command	What it does
<code>chap-devops models configs update ID NAME</code>	gives the variant new values
<code>chap-devops models configs archive ID NAME</code>	archives the variant
<code>chap-devops models configs export ID</code>	writes the variants in the marketplace format

- chap-core never deletes a configured model. It **archives** it, and keeps it for the backtests and predictions that use it.
- An update makes a new configured model and archives the old one.
- The Modeling App shows an archived configured model as Archived.

A 401 from the Modeling App

```
{"detail": "Missing or invalid API token"}
```

The route sends a token that chap-core does not accept, or sends none.

```
chap-devops dhis2 connect      # writes this deployment's token into the route
```

Two causes:

- `chap-devops auth rotate` ran, and the route still has the old token;
- `chap-devops auth enable` ran, and `chap-devops up` did not.

Models that do not register after `auth enable`

```
chapkit-simple-multistep-model  running, not registered via chap-core -
```

A protected chap-core refuses a registration without the key. The containers read `.env` only when compose creates them:

```
chap-devops up
chap-devops status
```

`chap-devops auth enable` writes `SERVICEKIT_REGISTRATION_KEY` beside the token, and gives it to every model.

When something does not work

```
chap-devops doctor      # docker, ports, memory, pins, and whether Chap is up
chap-devops status      # chap-core, the models, DHIS2
chap-devops dhis2 show  # the route, the apps, analytics
```

Each line that is wrong names the fix. For example:

```
warn memory 5.0 GB available to docker
keep at least 6.0 GB for docker: DHIS2 alone wants 4.0 GB for the
analytics populate phase, ...
```

Start now

```
curl -fsSL https://raw.githubusercontent.com/dhis2-chap/chap-devops/main/install.sh | sh
chap-devops init mychap --with dhis2 --models chapkit_simple_multistep_model
cd mychap && chap-devops up && chap-devops dhis2 connect
```

The DHIS2 chapter: <https://dhis2-chap.github.io/chap-devops/dhis2.html>