

CHAP-DEVOPS

Deploying Chap with one binary

Chap, its models, OCS and DHIS2, with Docker

What chap-devops is, what it deploys, and how it works

The problem

To run Chap, you need many parts that work together:

- **chap-core**: the API, a worker, PostgreSQL and Valkey
- **Forecasting models**: each one is a container with its own image
- **Open Climate Service (OCS)**: climate data, with an S3 store
- **DHIS2**: with the Modeling App and the Climate App

Each part has its own compose file, its own ports and its own settings. You must keep the versions in step, and you must connect the parts.

What chap-devops is

- **One static binary.** You need Docker and Compose 2.24.4 or newer. You do not need Python, uv or a checkout of chap-core.
- **It writes a deployment directory:** compose files, `.env` and `.chap-devops/`.
- **It runs the deployment** with `docker compose`, and tells you what it did and what to do next.
- **Linux and macOS.** On Windows, use WSL with the Linux binary.

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

The first deployment: three commands

```
chap-devops init mychap --models chapkit_simple_multistep_model  
cd mychap  
chap-devops up --wait
```

```
chap-core up http://localhost:8700 2.5.0 auth: off
```

MODEL	STATE	REACH	LAST PING
chapkit-simple-multistep-model	registered	via chap-core	7s ago

```
1 model registered
```

`chap-devops status` says "up" only when chap-core answered as chap-core, not when compose returned.

What a deployment can hold

Component	What it is	Turn it on
<code>chap-core</code>	Chap itself: the API, its worker, Valkey and PostgreSQL	on by default
<code>models</code>	marketplace models, or your own	<code>--models ID,ID , chap-devops models enable</code>
<code>ocs</code>	Open Climate Service: climate data over STAC and openEO	<code>--with ocs</code>
<code>s3</code>	an S3-compatible object store for OCS	<code>--with ocs,s3</code>
<code>dhis2</code>	a demo or development DHIS2 with its own database	<code>--with dhis2</code>

The shapes of a deployment

`--with` adds components to chap-core. `--only` names the whole set.

```
chap-devops init mychap --with ocs,s3,dhis2      # chap-core and everything
chap-devops init climate --only ocs,s3           # OCS alone, no chap-core
chap-devops init dhis --only dhis2               # a DHIS2 alone
chap-devops init models --only none --models ID,ID # model services alone
```

- A deployment without chap-core is a normal shape, not a broken one.
- `chap-devops components enable` and `disable` change the shape later.

The deployment directory

Intent in `.chap-devops/`, YAML:

- `project.yaml`: the pins, ports and compose project name
- `models.yaml`: the enabled models
- `components.yaml`: the components
- `models-manual.yaml`: models you added

Artifacts, rendered from it:

- `compose.yaml`, `compose.chap-devops.yaml`
- `compose.ocs.yaml`, `compose.s3.yaml`, `compose.dhis2.yaml`
- `compose.marketplace.yaml`: one `include:` line per model
- `compose.<service_id>.yaml`: one overlay per model

You change the intent with commands. The artifacts follow.

Your files, and the files chap-devops owns

File	Who owns it
<code>.env</code>	Written once by <code>init</code> . Only pins, the two auth secrets and component settings are added later. The database password is never rewritten.
<code>ocs/climate-service.yaml</code>	Yours. chap-devops makes it once, and changes only <code>read_only</code> and <code>plugins_dir</code> .
<code>dhis2/dhis.conf</code>	Yours, on the same terms. DHIS2 does not start without it.
every <code>compose*.yaml</code>	chap-devops's. They carry the line "Generated by chap-devops from <code>.chap-devops/</code> ".

chap-devops sync

```
unchanged    compose.yml
unchanged    compose.chap-devops.yml
unchanged    compose.chapkit-simple-multistep-model.yml
unchanged    compose.marketplace.yml
in sync: 0 written, 4 unchanged, 0 removed
```

- `chap-devops up` runs `sync` before `docker compose up`.
- The rendering is deterministic: a second sync writes nothing.
- `chap-devops sync --check` writes nothing and exits non-zero on drift, for CI.
- Sync removes only the overlays that it wrote itself.

chap-core's own compose file, byte for byte

- `compose.yml` is chap-core's `compose.ghcr.yml` at the pinned tag, as upstream publishes it. Upstream stays the source of truth.
- chap-devops makes its changes in `compose.chap-devops.yml`, a separate `-f` file:
 - the compose project `name:`
 - the API's host port, with `ports: !override`
 - `SERVICEKIT_REGISTRATION_KEY` for the `chap` service
 - the chap-devops labels on every service
- `!override` is why chap-devops needs Compose 2.24.4 or newer.

Ports

- **One published port** for chap-core and its models: the API, **8700** by default, `--api-port N` otherwise.
- chap-core reaches each model on the compose network at `http://<service_id>:8000`. A model needs no host port.
- A model can get a port of its own, from **5001 to 5999**:

```
chap-devops models expose chapkit-simple-multistep-model # lowest free port, 5001 up
chap-devops models expose chapkit-simple-multistep-model --bind 127.0.0.1
```

- The components: DHIS2 on **8780**, OCS on **8790**, S3 on none.
- `--port-base` moves the model range. `none` keeps a component off the host.

Authentication

```
chap-devops init mychap --models chapkit_simple_multistep_model --api-token # at creation
chap-devops auth enable # later
chap-devops auth show --reveal # read the token
```

- `CHAP_API_TOKEN` protects the whole API, except `/health`, `/health/ready` and `/system/info`.
- `SERVICEKIT_REGISTRATION_KEY` is a second secret. The models send it when they register.
- `chap-devops` writes both to `.env`, and the overlays pass them on.
- Neither change restarts anything: `chap-devops up` gives the new `.env` to the containers.

Pins: everything is pinned

- **Models:** an image tag, a `sha-<commit>` build for marketplace images.
- **chap-core:** a release tag. `init` resolves `latest` to the release it points at today, for example `v2.5.0`.
- **DHIS2:** a minor line, `2.42`, so a patch release comes with a pull.
- **OCS and S3:** moving tags (`main`, `latest`), because they publish no releases. Set `OCS_IMAGE_TAG` in `.env` to pin one.

Only `chap-devops models enable` and `chap-devops update` move a model pin.

up, update, restart: one job each

Command	What it does
<code>chap-devops up</code>	Starts what is on disk. Changes no pin.
<code>chap-devops update</code>	Moves the pins to what upstream publishes now, and pulls. Touches no container.
<code>chap-devops restart</code>	Recreates the services whose image or configuration changed.

```
chap-devops update --dry-run      # what would move, and nothing changes
chap-devops update --list-tags    # where chap-core can go
chap-devops update --chap-tag master
```

The closing line of `chap-devops update`

Each run ends with one of these lines:

```
already up to date
updated 1 model pin; restart needed: chap, worker (run `chap-devops restart`)
updated chap-core v2.4.0 -> v2.5.0; Chap is not running, the new versions start with `chap-devops up`
updated 1 model pin; nothing needs a restart
pulled a new image for ocs; restart needed: ocs (run `chap-devops restart`)
```

A move to an older chap-core asks first: an older schema against a migrated database. `chap-devops backup create` comes before it.

chap-devops status

```
chap-core    up    http://localhost:8700    2.5.0    auth: on
ocs          up    http://localhost:8790    3 datasets    212.0 MB data
```

MODEL	STATE	REACH	LAST PING
chapkit-simple-multistep-model	registered	port 5001	12s ago
chapkit-rwanda-malaria-bym-model	running, not registered	via chap-core	-
auto-arma-chapkit	not running	via chap-core	-

- `up, token rejected` is chap-core up with another token, not down.
- Each row that needs work gets a hint with the command.

chap-devops doctor

One pass over the machine, then the deployment:

- **The machine:** the docker CLI and daemon, the Compose version, the architecture, disk, memory for DHIS2, the hosts Chap pulls from, the GitHub limit, and the chap-devops version.
- **The deployment:** the compose project name, the files, sync drift, `.env`, the volumes, every host port, the pins, each model image on ghcr, and whether Chap answers.

Each line is `ok`, `warn`, `fail` or `skip`. A line that is not `ok` names the fix. The exit code is non-zero only for a `fail`.

Jobs: what chap-core ran

ID	TYPE	NAME	STATUS	STARTED	DURATION
e3e9f243...	create_backtest	eval-chapkit-rwanda-malaria-bym-model	FAILURE	30m ago	10s
501d5b5d...	create_backtest	eval-chapkit-ghr-model	SUCCESS	33m ago	2m 9s

- A failed job says why in `chap-devops jobs logs ID`, and nowhere else.
- `chap-devops jobs show`, `cancel` and `delete` take an id, or its first characters.
- `--status FAILURE`, `--type create_backtest` and `--limit N` filter the list.

Jobs of one backtest, from chap-core v2.5.0

One backtest also starts one `run_baseline` job for each built-in model, named `baseline/<model>`. So one backtest is four jobs:

ID	TYPE	NAME	STATUS	STARTED	DURATION
002a1f4d...	run_baseline	baseline/seasonal_median	SUCCESS	17s ago	4s
11bee580...	run_baseline	baseline/persistence	SUCCESS	17s ago	4s
12fbf146...	create_backtest	chap-devops-test-t-mlproject-r-20261015-161535	SUCCESS	17s ago	8s
58fcabe2...	run_baseline	baseline/global_median	SUCCESS	17s ago	4s
4202aea8...	create_dataset	chap-devops-test-t-mlproject-r-20261015-161535	SUCCESS	20s ago	2s

5 jobs: 5 done

This is `chap-devops jobs` after one `chap-devops models test --backtest` on `chap-core master`.

chap-devops api: one authenticated request

```
chap-devops api GET /v2/services
chap-devops api POST /v1/analytics/make-dataset --data @dataset.json
chap-devops api GET /v1/jobs/59159ad4-893b-4bbf-a855-70a38c3fb787
```

- The base URL and the `Authorization: Bearer` header of this deployment are filled in.
- JSON comes back pretty-printed. A JSON string prints as its text.
- Exit code **0** for a 2xx, **1** for a 4xx or 5xx, **2** when chap-core does not answer.

Backup and restore

```
chap-devops backup create                # the databases, the model and component data, the files
chap-devops backup restore chap-devops-backup-mychap-20261004-101500.tar.gz
```

- One `tar.gz`: `manifest.yaml`, the project files, a `pg_dump` of the chap-core database and of the DHIS2 database, and one tar per model volume and per other component volume.
- `tar -tzf` reads it. An admin can restore it by hand.
- `restore` prints what it will overwrite and stop, then asks. `--yes` answers in advance.
- The caches (`logs`, `runs`, `renv`, `uv`) are left out.

Several deployments on one machine

- Each `chap-devops init` directory is a deployment with its own containers, data and compose project name.
- They share only the host ports. Two ways:
 - **Take turns:** `chap-devops up` sees the other deployment on the port, and asks to stop it. Its data is kept. `--replace` answers in advance.
 - **Their own ports:** `--api-port 8701 --port-base 5101`, and `--ocs-port`, `--dhis2-port`.
- `chap-devops cleanup` deletes what deployments whose directory is gone left in docker, after it asks.

Container labels and `chap-devops top`

Every container that chap-devops starts carries labels:

Label	Value
<code>com.winterop.chap-devops.role</code>	<code>chap-core</code> , <code>model</code> , <code>ocs</code> , <code>s3</code> or <code>dhis2</code>
<code>com.winterop.chap-devops.model</code>	the marketplace id of a model
<code>com.winterop.chap-devops.kind</code>	<code>init</code> for a deployment, <code>run</code> for a <code>chap-devops run</code> group, <code>cli</code> for the one-shot container of <code>chap-devops chap</code> and <code>chap-devops chapkit</code>

`chap-devops top` uses them to show every deployment on the machine as one live tree: each service with its state, CPU, memory and URL. `l` shows a log, `s` stops a model.

Offline

- The marketplace comes from the network, then a cache, then a snapshot that is compiled into the binary.
- `--offline` never touches the network. A laptop with no connection still resolves the catalogue.
- `chap-devops update` is the one command that refuses to run offline: a plan from an old catalogue is not a plan.
- `chap-devops doctor --offline` skips the network checks, and says so.

One model, no deployment: `chap-devops run`

```
chap-devops run chapkit_simple_multistep_model
```

```
running chapkit_simple_multistep_model on http://localhost:5001 (answered in 4s)
```

- A marketplace id, a GitHub URL, a ghcr image or a local image.
- No directory to keep: the model runs in a group under chap-devops's data directory.
- `-a` keeps it in the foreground with its log, and Ctrl-C stops the model.
- `chap-devops ps` lists the models, and `chap-devops stop` stops them.

See the **chap-devops run** deck for groups, ports and `chap-devops top`.

The chap CLI, without Python: `chap-devops chap`

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

- chap-core's own `chap` CLI runs in chap-core's image.
- Your directory is mounted, so the files are on your machine.
- A model id starts that model only. No `chap-devops up`, no chap-core.

See the **chap CLI** deck.

The chapkit CLI, without Python: `chap-devops chapkit`

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

```
the MLproject service answers at http://localhost:5001
```

- chapkit's own `chapkit` CLI runs in a container. You need only Docker.
- `mlproject run` serves a model in the MLproject format, with no image to build. Ctrl-C stops it.
- `mlproject migrate` makes it a chapkit service, and `mlproject export` goes back.

See the **MLproject models** deck.

DHIS2 and the Modeling App

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

`chap-devops dhis2 connect` makes the `chap` route from DHIS2 to chap-core, installs the Modeling App and the Climate App, and generates analytics.

`chap-devops dhis2 use URL` points it at a DHIS2 that runs elsewhere.

See the **Modeling App** deck.

Rules chap-devops keeps

- **Every command reports.** It says what it found, what changed and what to do next. No command exits in silence.
- **Messages name the file or the command,** and give the way out on the same line.
- **State is checked, not assumed.** "up" means chap-core answered.
- **Nothing needs the network that does not have to.**
- **.env is written once.** The database password in it outlives everything else.

Where to read more

- The book: <https://winterop-com.github.io/chap-devops/>
- The code: <https://github.com/winterop-com/chap-devops>
- For an AI assistant: the "For AI assistants" chapter tells it how to set up chap-devops for you, command by command.

```
chap-devops --help  
chap-devops doctor
```