Alembic Database migrations

START HERE

`alembic` **`init`** *`<dir>`*
Create migration environment

`alembic` **`revision --autogenerate -m "`***`<msg>`***`"`**
Diff models, write script

`alembic` **`upgrade head`**
Apply all migrations

`alembic` **`downgrade -1`**
Undo last migration

`alembic` **`current`**
Where is the database?

`alembic` **`history`**
List all revisions

`alembic` **`heads`**
Show newest revisions

`alembic` **`stamp head`**
Mark as up to date

Models target_metadata → `revision --autogenerate` → **Script** versions/*.py → `upgrade` → **Database** alembic_version table

CONCEPTS

Always read an autogenerated script before running it: it misses some changes and can be wrong.

The database stores only the current revision id, in the alembic_version table.

Revisions form a chain (down_revision); two scripts with the same parent create branches.

SETUP

`alembic` **`init`** *`<dir>`*
Create scripts dir and alembic.ini

`alembic` **`init -t async`** *`<dir>`*
Init with the async template

`alembic` **`init -t pyproject`** *`<dir>`*
Init with config in pyproject.toml

`alembic` **`list_templates`**
List init templates

`alembic` **`-c`** *`<file>`* *`<cmd>`*
Use another config file

`alembic` **`-n`** *`<name>`* *`<cmd>`*
Use another ini section

`alembic` **`-x`** *`<key>`*`=`*`<value>`* *`<cmd>`*
Pass an argument to env.py

SETUP (cont.)

`alembic` **`--raiseerr`** *`<cmd>`*
Show full stack trace on error

Set sqlalchemy.url in alembic.ini or override it in env.py, and set target_metadata there to your models' MetaData.

CREATE REVISIONS

`alembic` **`revision -m "`***`<msg>`***`"`**
New empty migration script

`alembic` **`revision --autogenerate -m "`***`<msg>`***`"`**
Script from model vs. database diff

`alembic` **`revision --rev-id`** *`<id>`* **`-m "`***`<msg>`***`"`**
Choose the revision id yourself

`alembic` **`revision --head`** *`<rev>`* **`-m "`***`<msg>`***`"`**
Base the new script on a given head

`alembic` **`revision --branch-label`** *`<name>`* **`-m "`***`<msg>`***`"`**
Start a labelled branch

`alembic` **`revision --depends-on`** *`<rev>`* **`-m "`***`<msg>`***`"`**
Require another revision first

CREATE REVISIONS (cont.)

`alembic` **`check`**
Fail if models differ from scripts
Exits non-zero when autogenerate would produce changes: handy in CI.

`alembic` **`edit`** *`<rev>`*
Open a script in $EDITOR

RUN MIGRATIONS

`alembic` **`upgrade head`**
Migrate to the newest revision

`alembic` **`upgrade +1`**
Apply the next migration

`alembic` **`upgrade`** *`<rev>`*
Migrate up to a revision
A unique prefix of the id is enough.

`alembic` **`upgrade`** *`<rev>`*`+`*`<n>`*
Move n steps up from a revision

`alembic` **`downgrade -1`**
Undo the last migration

`alembic` **`downgrade`** *`<rev>`*
Roll back to a revision

`alembic` **`downgrade base`**
Roll back everything

`alembic` **`upgrade`** *`<from>`*`:`*`<to>`* **`--sql`**
Print SQL instead of running it
Offline mode needs the start revision as <from>:<to>; env.py must support it.

INSPECT

`alembic` **`current`**
Show the database's revision

`alembic` **`current -v`**
Current revision with details

`alembic` **`current --check-heads`**
Fail if database is not at head

`alembic` **`history`**
List revisions, newest first

`alembic` **`history -v`**
History with full details

`alembic` **`history -i`**
Mark the current revision

`alembic` **`heads`**
Show newest revisions

`alembic` **`branches`**
Show branch points

`alembic` **`show`** *`<rev>`*
Show one revision

BRANCHES AND REPAIR

`alembic` **`merge heads -m "`***`<msg>`***`"`**
Merge all heads into one
Needed when 'Multiple head revisions' blocks upgrade head.

`alembic` **`stamp head`**
Mark database as up to date
Records the revision without running any migration.

`alembic` **`stamp`** *`<rev>`*
Mark database at a revision

INSIDE A SCRIPT

`op.create_table("`*`<t>`***`", sa.Column(...))`**
Create a table

`op.drop_table("`*`<t>`***`")`**
Drop a table

`op.add_column("`*`<t>`***`", sa.Column(...))`**
Add a column

`op.drop_column("`*`<t>`***`", "`***`<col>`***`")`**
Drop a column

`op.alter_column("`*`<t>`***`", "`***`<col>`***`", ...)`**
Change type, null or name

`op.create_index("`*`<ix>`***`", "`***`<t>`***`", ["`***`<col>`***`"])`**
Create an index

`op.create_foreign_key(...)`
Add a foreign key

`op.execute("`*`<sql>`***`")`**
Run raw SQL

`op.batch_alter_table("`*`<t>`***`")`**
Batch mode for SQLite changes
Used as a context manager; recreates the table where ALTER is unsupported.

Write downgrade() too, or downgrade will fail; a bare pass leaves the change in place.

CONFIG

`compare_type=True`
Detect column type changes (env.py)
Passed to context.configure(); not on by default in older versions.

Autogenerate does not detect table or column renames (it sees drop plus add); edit the script by hand to use rename ops.

RECIPES

First migration for existing models

```
alembic init migrations
# edit alembic.ini url and
env.py target_metadata
alembic revision --autogenerate
-m "initial"
alembic upgrade head
```

Adopt an existing database

```
alembic revision --autogenerate
-m "baseline"
# check the script, then skip
it on this database
alembic stamp head
```

Fix multiple heads

```
alembic heads
alembic merge heads -m "merge"
alembic upgrade head
```

Review SQL before production

```
alembic upgrade head --sql >
migrate.sql
alembic upgrade <from>:<to> --
sql
```