Alembic Database migrations

CONCEPTS

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

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

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

SETUP

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

> 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` **`check`**
Fail if models differ from scripts

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

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

RUN MIGRATIONS (cont.)

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

INSPECT

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

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

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

BRANCHES AND REPAIR

`alembic` **`merge heads -m`** "*`<msg>`*"
Merge all heads into one

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

INSIDE A SCRIPT

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

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

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

> 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)

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