Migrations: a first PHP migration, step by step

Source file: doc/eng-us/guides/migrations-tutorial.md Last changed 2026-11-04
https://doc.exponential.earth/learninminutes/exphelp/viewer/guides/migrations-tutorial.md

In this tutorial you write two PHP migrations for a site: one creates a table `exp_news_note` with a first row, the other publishes a folder "News" below the content root and notes it in the table. You check them, look at the dry run and the simulation, run them, read the record, take one back with its `down()`, and see what happens when a migration fails. At the end you write a YAML migration with references, a loop and a condition. It takes about 25 minutes.

The output below is from a run on SQLite on 31 October 2026, on an installation with the clean data; run ids, times and object ids differ on yours. Run every command from the installation root as the user the web server runs as (as root, add `--allow-root-user`).

Guides · Migrations: the hub · Migrations · Next: PHP migrations

Before you start

- Exponential 6.0.15 with the record tables: `./console exp:upgrade:status` shows `6.0.15-migration-tables` as `done` (if it is `pending`, run `./console exp:upgrade --apply` first).
- A development installation. A run takes a caption (a backup) and commits to the site's git repository; on your first try add `--no-commit`, as below, so nothing reaches a remote.

1. Generate the first migration

Shell

```
./console exp:migration:generate --skeleton news_note_table --description="A table for news notes, with a first note"
```

Without `--apply` the command shows the file it would write:

```
Exponential migrations: generate, dry run (--apply writes the file)
File migrations/20261031120000_news_note_table.php
Migration 20261031120000_news_note_table, class \Site\Migrations\NewsNoteTable20261031120000

    <?php
    // description: A table for news notes, with a first note
    namespace Site\Migrations;
    ...
PASS
```

Write it:

Shell

```
./console exp:migration:generate --skeleton news_note_table --description="A table for news notes, with a first note" --apply
```

```
Exponential migrations: generate
Wrote migrations/20261031120000_news_note_table.php
Migration 20261031120000_news_note_table, class \Site\Migrations\NewsNoteTable20261031120000
Next: fill in plan() and up(), then ./console exp:migration:check and ./console exp:migration:dry
-run
PASS
```

The file name starts with the time of now; the class is named after the name and the time, in the namespace `Site\Migrations`, so no two migrations declare the same class. An existing file is never replaced.

2. Fill it in

Replace the body of the class with the change. `plan()` lists what `up()` would do without changing anything (the dry run shows it); `up()` does it and lists what it did; `down()` undoes it.

PHP

```php
<?php
// description: A table for news notes, with a first note
// author: Site team
namespace Site\Migrations;

final class NewsNoteTable20261031120000 extends \expMigrationPhp
{
    const TABLE = 'exp_news_note';

    /** @return array the table in the format of share/db_schema.dba */
    public static function schema()
    {
        return array( self::TABLE => array(
            'name' => self::TABLE,
            'fields' => array(
                'id' => array( 'length' => 11, 'type' => 'int', 'not_null' => '1', 'default' => 0
 ),
                'title' => array( 'length' => 150, 'type' => 'varchar', 'not_null' => '1', 'defau
lt' => '' ),
            ),
            'indexes' => array( 'PRIMARY' => array( 'type' => 'primary', 'fields' => array( 'id'
) ) ),
        ) );
    }

    public function plan( $context )
    {
        return array( 'create the table ' . self::TABLE, 'write the first note' );
    }

    public function up( $context )
    {
        $messages = array();
        if ( !\expKernelTableSchema::install( array( self::TABLE ), $context->db(), self::schema(
), $messages ) )
            throw new \RuntimeException( 'the table could not be created: ' . implode( '; ', $mes
sages ) );
        $context->db()->query( 'INSERT INTO ' . self::TABLE . " ( id, title ) VALUES ( 1, 'First
note' )" );
        $context->setReference( 'news_note_table', self::TABLE );
        return array( 'created ' . self::TABLE, 'wrote the note 1' );
    }
```

```
    public function down( $context )
    {
        $context->db()->query( 'DROP TABLE ' . self::TABLE );
        \expKernelTableSchema::reset();
        return array( 'dropped ' . self::TABLE );
    }
}
```

`expKernelTableSchema::install()` creates the table on every engine from a definition in the format of `share/db_schema.dba` (SQL on MySQL/MariaDB, PostgreSQL, SQLite and Oracle, a collection on MongoDB). Everything goes through `$context->db()`, so the run's transaction is the one used.

3. A second migration that needs the first

Write `migrations/20261031120100_news_folder.php` by hand, or generate it as above:

PHP

```php
<?php
// description: The folder News below the content root
// requires: 20261031120000_news_note_table
namespace Site\Migrations;

final class NewsFolder20261031120100 extends \expMigrationPhp
{
    const REMOTE_ID = 'site-news-folder';

    public function plan( $context )
    {
        return array( 'publish the folder News below node 2' );
    }

    public function up( $context )
    {
        $root = \eZContentObjectTreeNode::fetch( 2 );
        if ( !$root instanceof \eZContentObjectTreeNode )
            throw new \RuntimeException( 'there is no content root (node 2)' );
        $object = \eZContentFunctions::createAndPublishObject( array(
            'parent_node_id' => 2,
            'language' => $root->attribute( 'object' )->attribute( 'initial_language_code' ),
            'class_identifier' => 'folder',
            'creator_id' => \eZUser::currentUserID(),
            'remote_id' => self::REMOTE_ID,
            'attributes' => array( 'name' => 'News' ),
        ), false );
        if ( !$object instanceof \eZContentObject )
            throw new \RuntimeException( 'the folder News could not be published' );
        $id = (int)$object->attribute( 'id' );
        $context->db()->query( 'INSERT INTO ' . $context->reference( 'news_note_table', 'exp_news
_note' )
                               . " ( id, title ) VALUES ( 2, 'folder $id' )" );
        return array( "published the folder News (object $id)" );
    }

    public function down( $context )
    {
        $object = \eZContentObject::fetchByRemoteID( self::REMOTE_ID );
        if ( $object instanceof \eZContentObject )
            $object->purge();
        $context->db()->query( 'DELETE FROM exp_news_note WHERE id = 2' );
        return array( 'removed the folder News and its note' );
```

```
    }
}
```

The header line `requires:` makes it run after the first migration, whatever the timestamps say. The folder is created as the run's user (`eZUser::currentUserID()`): the signed-in user, or `[RunSettings] RunAsUser` on the console. The remote id lets `down()` find the folder on every installation, where its object id differs.

4. Check

Shell

```
./console exp:migration:check
```

```
Exponential migrations: check
  site                    migrations

  ok    20261031120000_news_note_table              php  migrations/20261031120000_news_note_tab
le.php
  ok    20261031120100_news_folder                  php  migrations/20261031120100_news_folder.p
hp

2 migration(s) checked
PASS
```

The check needs no database and runs nothing; put it in CI. A wrong header key, a requirement that does not exist, a cycle, a PHP syntax error or a second class are each an error with exit 1.

5. Status and dry run

Shell

```
./console exp:migration:status
```

```
Exponential migrations: status
Run 20261031-193019-c2dff4, environment development, database sqlite (transactions: full)

  pending     20261031120000_news_note_table              site              A table for news no
tes, with a first note
  pending     20261031120100_news_folder                  site              The folder News bel
ow the content root

Summary: 2 migration(s): 0 done, 2 pending, 0 not needed, 0 failed, 0 unknown; 2 would-apply
PASS
```

Shell

```
./console exp:migration:migrate
```

```
Exponential migrations: dry run, nothing is changed (--apply runs the migrations)
Run 20261031-193019-1a60c4, environment development, database sqlite (transactions: full)

  would-apply 20261031120000_news_note_table               site               A table for news no
tes, with a first note
              would create the table exp_news_note
              would write the first note
  would-apply 20261031120100_news_folder                   site               The folder News bel
ow the content root
              would publish the folder News below node 2

Captions: pre would-take, post would-take
Summary: 2 migration(s): 0 done, 2 pending, 0 not needed, 0 failed, 0 unknown; 2 would-apply
PASS
```

The dry run also checks the code trees: when `settings/`, `extension/` or `design/` have changes that are not committed, it warns that a run with `--apply` would be refused.

6. Simulate

On PostgreSQL and SQLite the migrations can run for real in a transaction that is rolled back at the end:

Shell

```
./console exp:migration:dry-run --simulate --diff
```

```
Exponential migrations: simulation, the database is rolled back afterwards
Run 20261031-193020-599bb5, environment development, database sqlite (transactions: full), as adm
in

  simulated   20261031120000_news_note_table               site               A table for news no
tes, with a first note
              would create the table exp_news_note
              would write the first note
              simulation passed in 7 ms
              created exp_news_note
              wrote the note 1
              + table exp_news_note (2 columns, 1 row)
  simulated   20261031120100_news_folder                   site               The folder News bel
ow the content root
              would publish the folder News below node 2
              simulation passed in 359 ms
              published the folder News (object 57)
              ~ table exp_news_note: rows 1 -> 2
              ~ table ezcontentobject: rows 18 -> 19
              ~ table ezcontentobject_attribute: rows 85 -> 90
              ~ table ezcontentobject_tree: rows 19 -> 20
              ...

Summary: 2 migration(s): 0 done, 2 pending, 0 not needed, 0 failed, 0 unknown; 2 simulated
WARN simulated in a transaction that was rolled back: the database is unchanged, but what the mig
rations wrote outside it (files, caches, the search index, mail) is not undone
PASS
```

(The diff lists every table that changed; a few lines are left out here.) On MySQL/MariaDB, Oracle and MongoDB the simulation is not run: a schema change is committed at once there, so the command lists the plan with a warning.

7. Run

Shell

```
./console exp:migration:migrate --apply --no-commit
```

On the installation of this tutorial the captions were switched off as well (`--no-caption`), so the output shows only the run:

```
Exponential migrations: run
Run 20261031-193020-1356d6, environment development, database sqlite (transactions: full), as adm
in

  applied     20261031120000_news_note_table               site               A table for news no
tes, with a first note
              created exp_news_note
              wrote the note 1
  applied     20261031120100_news_folder                   site               The folder News bel
ow the content root
              published the folder News (object 57)

Repository: no commit, push disabled
Summary: 2 migration(s): 2 done, 0 pending, 0 not needed, 0 failed, 0 unknown; 2 applied
WARN no caption before the run (--no-caption or [CaptionSettings] Before=disabled): a failure can
not be restored automatically
PASS
```

With the captions on (the default), the run says which it took:

```
Captions: pre 20261031-193126, post 20261031-193127
```

The run wrote its log, `migrations/runs/20261031-193020-1356d6.log`:

```
2026-11-01T02:30:20Z info run 20261031-193020-1356d6 by admin on sqlite in development: 2 to appl
y
2026-11-01T02:30:20Z info applied 20261031120000_news_note_table in 3 ms: created exp_news_note |
 wrote the note 1
2026-11-01T02:30:20Z info applied 20261031120100_news_folder in 21 ms: published the folder News
(object 57)
2026-11-01T02:30:20Z info run 20261031-193020-1356d6 done: 2 applied, 0 failed
2026-11-01T02:30:20Z info the record of the run is not committed ([RepositorySettings] Commit=dis
abled or --no-commit)
```

and the status file `migrations/status/development.json` (schema `exp-migration-status/1`) with both migrations as `done`. Without `--no-commit` both files are committed, tagged `migrations/development/<run-id>` and pushed (The migrations repository).

8. The record

Shell

```
./console exp:migration:status
```

```
Exponential migrations: status
Run 20261031-193020-ec6ba1, environment development, database sqlite (transactions: full)

  done        20261031120000_news_note_table               site               A table for news no
tes, with a first note
              recorded done by run 20261031-193020-1356d6 on 2026-10-31T19:30:20-07:00
  done        20261031120100_news_folder                   site               The folder News bel
ow the content root
              recorded done by run 20261031-193020-1356d6 on 2026-10-31T19:30:20-07:00

The latest runs, newest first:
  20261031-193020-1356d6   done        2026-10-31T19:30:20-07:00  cli admin, push disabled
Summary: 2 migration(s): 2 done, 0 pending, 0 not needed, 0 failed, 0 unknown
PASS
```

A second `exp:migration:migrate` now lists both as `done` and does nothing.

9. A changed file

Edit `20261031120100_news_folder.php` after it ran (here: a comment added at the end) and run again:

```
  done        20261031120000_news_note_table               site               A table for news no
tes, with a first note
  refused     20261031120100_news_folder                   site               The folder News bel
ow the content root
              CHANGED: the file is not the one applied (checksum 75375541e9e9 -> c82e1066cc76)

Summary: 2 migration(s): 2 done, 0 pending, 0 not needed, 0 failed, 0 unknown; 1 refused
FAIL: 20261031120100_news_folder changed after it was applied (checksum 75375541e9e9 -> c82e1066c
c76); a run with --force --reason=<text> accepts the change
FAIL (exit 1)
```

A migration that ran is never run again, so a changed file would mean installations that differ. When the change does not matter, accept it with a reason; it is recorded and audited:

Shell

```
./console exp:migration:migrate --apply --force --reason="a comment only"
```

```
  accepted    20261031120100_news_folder                   site               The folder News bel
ow the content root
              the change of the file was accepted: a comment only
```

A real change belongs in a new migration.

10. Take one migration back

Shell

```
./console exp:migration:rollback 20261031120100_news_folder
```

```
Exponential migrations: rollback with down(), dry run (--apply undoes them)
Run 20261031-193104-4d6423, environment development, database sqlite (transactions: full)

  would-undo  20261031120100_news_folder                  site              The folder News bel
ow the content root

Summary: 1 migration(s): 1 done, 0 pending, 0 not needed, 0 failed, 0 unknown; 1 would-undo
PASS
```

Shell

```
./console exp:migration:rollback 20261031120100_news_folder --apply
```

```
Exponential migrations: rollback with down()
Run 20261031-193104-c8d367, environment development, database sqlite (transactions: full), as adm
in

  undone      20261031120100_news_folder                  site              The folder News bel
ow the content root
              removed the folder News and its note

Summary: 1 migration(s): 0 done, 1 pending, 0 not needed, 0 failed, 0 unknown; 1 undone
PASS
```

The record of the migration is removed, so it is pending again; the next `exp:migration:migrate --apply` runs it.

11. A failure

Add a migration that writes a row and then fails:

PHP

```php
<?php
// description: Writes a note, then fails
namespace Site\Migrations;

final class FailsAfterChange extends \expMigrationPhp
{
    public function up( $context )
    {
        $context->db()->query( "INSERT INTO exp_news_note ( id, title ) VALUES ( 3, 'half done' )
" );
        throw new \RuntimeException( 'the import file is missing' );
    }
}
```

and run with the captions on (the default):

```
Exponential migrations: run
Run 20261031-193127-c293f8, environment development, database sqlite (transactions: full), as adm
in

  done        20261031120000_news_note_table               site               A table for news no
tes, with a first note
  done        20261031120100_news_folder                   site               The folder News bel
ow the content root
  failed      20261031120300_fails_after_change            site               Writes a note, then
 fails
              the import file is missing

Captions: pre 20261031-193128, failure 20261031-193128-2
Restore: restored (caption 20261031-193128)
Repository: no commit, push disabled
Summary: 3 migration(s): 2 done, 0 pending, 0 not needed, 1 failed, 0 unknown
WARN restored the caption 20261031-193128 taken before the run (database, storage, settings); the
 failed state is kept in the caption 20261031-193128-2
FAIL (exit 1)
```

The migration's transaction undid its row; then the state was kept in a caption of kind `migration-failure` and the caption taken before the run was restored. Fix the migration (or delete the file, since it never ran) and run again. Rollback and captions explains each part, and Troubleshooting what to do with every message.

12. A YAML migration

The same life cycle works for a migration written as YAML steps. The output below was taken on 1 November 2026 on a development installation where it was the only migration. Generate one:

Shell

```
./console exp:migration:generate --skeleton news_languages --type=yml --description="The language
s of the news, checked before the import" --apply
```

```
Exponential migrations: generate
Wrote migrations/20261101094239_news_languages.yml
Migration 20261101094239_news_languages
Next: write the steps, then ./console exp:migration:check and ./console exp:migration:dry-run
PASS
```

The skeleton has the header and `steps: []`, with an example in comments. Replace it with:

YAML

```yaml
# description: The languages of the news, checked before the import
# author: Site team
steps:
    - type: reference
      set:
          news_languages: [ eng-US, ger-DE ]
          news_parent: "${idmap.node.news}"
    - foreach: "${ref.news_languages}"
      as: language
      steps:
          - type: assert
            that: { value: "${loop.language}", matches: "/^[a-z]{3}-[A-Z]{2}$/" }
            message: "${loop.language} is not a language code"
    - type: assert
      if: { environment: production }
      that: { value: "${ref.news_parent}", not_equals: 2 }
      message: production names its own news node in its id map
```

`${idmap.node.news}` reads the id map of the environment, `migrations/idmap/development.json` here:

JSON

```json
{
    "schema": "exp-migration-idmap/1",
    "ids": {
        "node": {
            "news": 2
        }
    }
}
```

The check reads the YAML and checks every step against its step type, without a database. A file with mistakes (a misspelled type, an unknown operator, a loop variable that no loop sets, a reference name that starts with a digit) shows each with its place:

```
Exponential migrations: check
  site                    migrations

  ok    20261101094239_news_languages               yml  migrations/20261101094239_news_language
s.yml
  FAIL  20261101094500_news_typo                    yml  migrations/20261101094500_news_typo.yml
        steps[0]: the unknown step type refrence (registered: assert, reference; an extension add
s one in migration.ini [StepSettings] StepTypeList)
        steps[1] (assert).that: value takes exactly one operator (equals, not_equals, in, not_in,
 gt, gte, lt, lte, matches, empty), not bigger
        steps[2].steps[0]: ${loop.language} names no variable of a loop around it (item, key, ind
ex)
        steps[2].steps[0] (reference): set.1st: a reference name is letters, digits and _, not st
arting with a digit

2 migration(s) checked
...
FAIL (exit 1)
```

With the broken file gone, the dry run shows the plan, with the loop unrolled and the step for production skipped:

Shell

```
./console exp:migration:migrate
```

```
Exponential migrations: dry run, nothing is changed (--apply runs the migrations)
Run 20261101-094515-4b20db, environment development, database sqlite (transactions: full)

  would-apply 20261101094239_news_languages               site              The languages of th
e news, checked before the import
              would set the reference news_languages to ["eng-US","ger-DE"]
              would set the reference news_parent to 2
              would run for each of 2 item(s), as language:
                would check that {"value":"eng-US","matches":"/^[a-z]{3}-[A-Z]{2}$/"}
                would check that {"value":"ger-DE","matches":"/^[a-z]{3}-[A-Z]{2}$/"}
              would skip steps[2] (assert): the condition is false

Captions: pre would-take, post would-take
Summary: 1 migration(s): 0 done, 1 pending, 0 not needed, 0 failed, 0 unknown; 1 would-apply
PASS
```

and the run does it:

Shell

```
./console exp:migration:migrate --apply --no-commit --no-caption
```

```
Exponential migrations: run
Run 20261101-094534-57f2cf, environment development, database sqlite (transactions: full), as adm
in

  applied     20261101094239_news_languages               site              The languages of th
e news, checked before the import
              set the reference news_languages
              set the reference news_parent
              checked that {"value":"eng-US","matches":"/^[a-z]{3}-[A-Z]{2}$/"}
              checked that {"value":"ger-DE","matches":"/^[a-z]{3}-[A-Z]{2}$/"}
              skipped steps[2] (assert): the condition is false

Repository: no commit, push disabled
Summary: 1 migration(s): 1 done, 0 pending, 0 not needed, 0 failed, 0 unknown; 1 applied
WARN no caption before the run (--no-caption or [CaptionSettings] Before=disabled): a failure can
not be restored automatically
PASS
```

The record, the run log, the checksum guard, the captions and the commit work as for the PHP migrations above. An assert that is false fails the migration like an exception in `up()`, and its transaction is rolled back. This migration has a loop and no `down:`, so `exp:migration:rollback 20261101094239_news_languages` refuses it: add a `down:` list to undo a YAML migration by itself, or roll the run back to its caption.

Besides `reference` and `assert`, the kernel's step types `content`, `location` and `content_version` create and change objects, their translations, locations and versions (YAML migration steps); the steps for classes and roles are added to `migration.ini` `[StepSettings] StepTypeList` as they come, and an extension registers its own there. The file format lists every key, placeholder, condition and loop.

What you did

- generated a migration and filled in `plan()`, `up()` and `down()`;
- ordered two migrations with `requires:` and passed a value from one to the next with `setReference()`;
- checked them without a database, looked at the dry run and the simulation;
- ran them, read the record and the run log, met the checksum guard;
- took one back with `down()` and saw a failure restored from its caption;
- wrote a YAML migration with references, a loop, a condition and the id map, and checked, planned and ran it.

Next: PHP migrations for everything a migration class can do.

Related pages

- Migrations: the hub, All recipes, Frequently asked questions
- The guide, PHP migrations, Rollback and captions, The migrations repository
- In the administration, Commands, Settings, Policies, views and events
- The file format, The feature
- When a migration fails, Troubleshooting messages
- Migrations versus migrating a site

Exponential Copyright © 1998-2026 Exponential Foundation and 7x
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".
This documentation is free documentation, licensed under the GNU Free Documentation License 1.3 or later:
https://www.gnu.org/licenses/fdl-1.3.html.