Introduction to

DRUPAL 8 MIGRATIONS

DrupalCon 2019
Wednesday 4/10
11:45 -12:15
Room: 6C | Level 6

CHROMATIC

CLARE MING

Senior Developer at Chromatic

@bodhicitta

https://github.com/cjming


CHROMATIC
EPIC
https://chromatichq.com
@chromatichq

Outside

Parents

SHAPE

Casper

martha stewart

FamilyCircle

INTUIT

THEATERMANIA

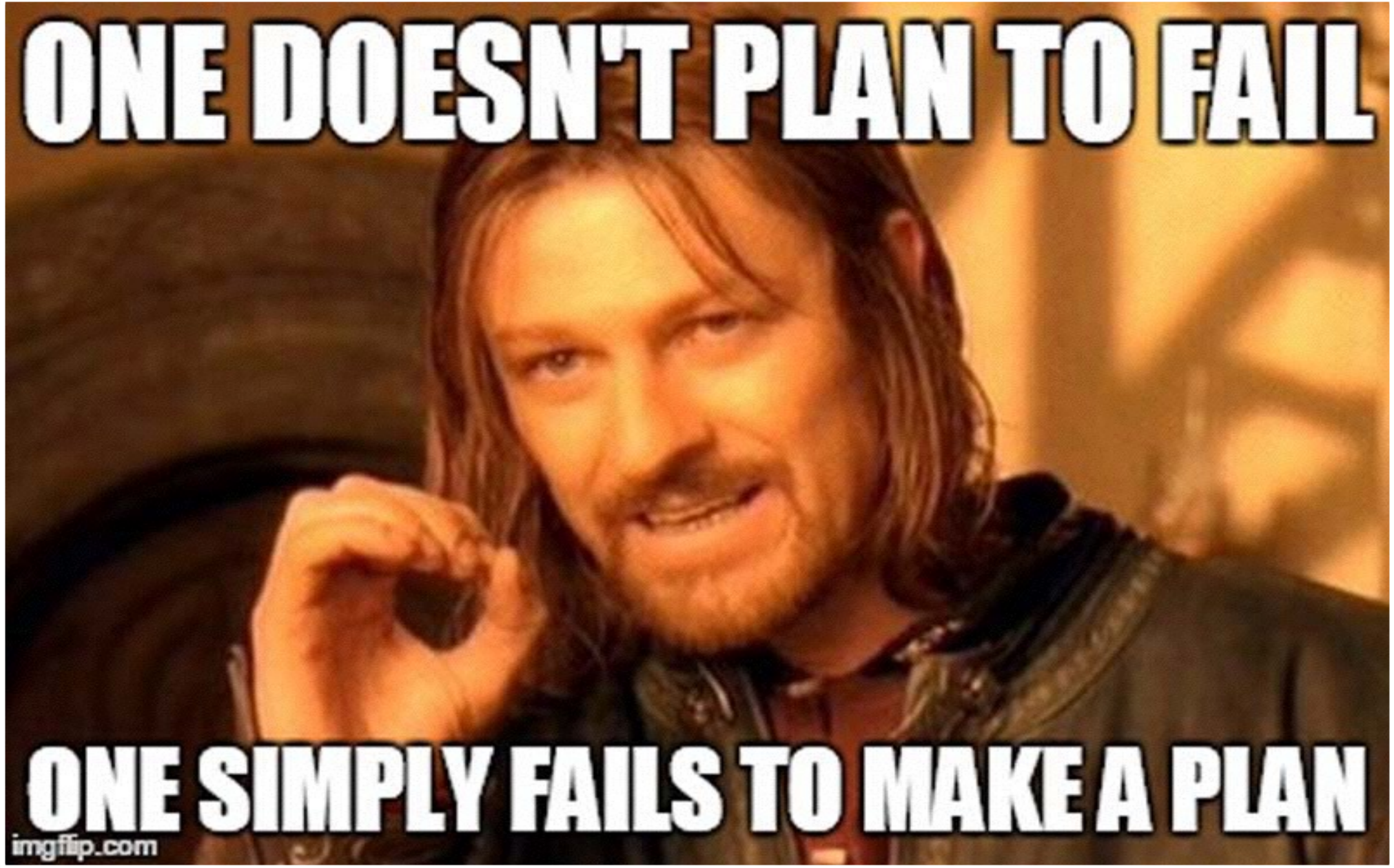

CHROMATIC
https://chromatichq.com
@chromatichq

PREREQUISITES

- Familiarity with Drupal 8
- PHP | OOP
- DRUSH
- Custom Module Development
- Configuration Management

MIGRATING to DRUPAL 8

- Preparation & Planning ***
- Migrate API + Modules
- Migration Plugins - YAML
- ETL - source, process, destination
- Running Migrations - DRUSH
- Debugging / Troubleshooting


ONE DOESN'T PLAN TO FAIL
ONE SIMPLY FAILS TO MAKE A PLAN
imgflip.com

DECISIONS

- Content inventory/site audit
- Content/features to keep/discard
- Refactor information architecture
- Rebrand | Redesign

PREPARE A MIGRATION PLAN

Preparing for a migration is essential.

Migrations involve:

- preparing and analyzing source data
- building a new site to house the migrated data
- LOTS of testing

A successful migration requires careful planning and consideration of many details.

MIGRATE API

- The following modules ship with core:
  - migrate
  - migrate_drupal
  - migrate_drupal_multilingual
  - migrate_drupal_ui
- https://www.drupal.org/docs/8/api/migrate-api
- https://www.drupal.org/docs/8/upgrade

MIGRATE MODULES

- Migrate Plus
- Migrate Tools
- Migrate Upgrade
- Migrate Source CSV

Example Code:

https://github.com/cjming/D8migrations

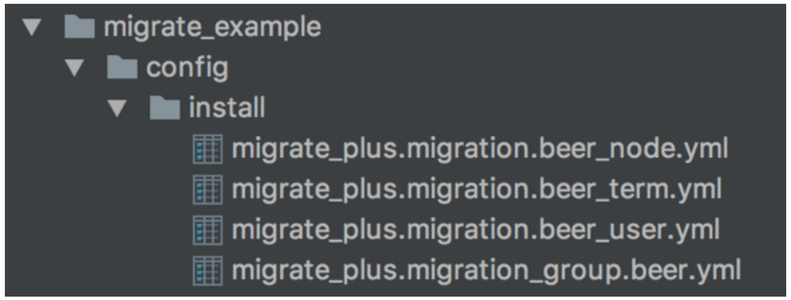
Migrate Plus: migrate_example

https://www.drupal.org/project/migrate_plus

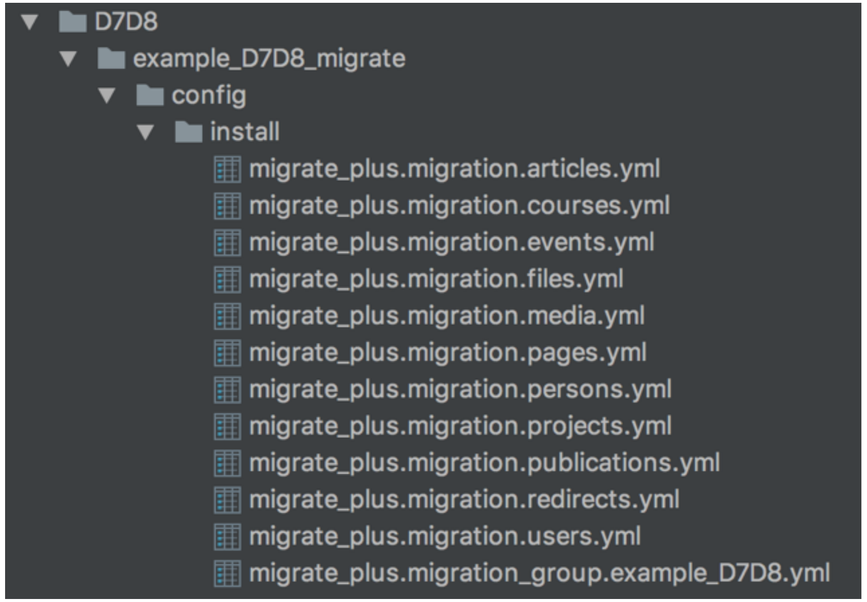
MIGRATIONS 101

- Migration Plugins - YAML
- ETL - Extract, Transform, Load
- Source Plugins (SQL, CSV, JSON, XML, Drupal 6/7, custom, etc)
- Process Plugins (core, contrib, custom)
- Destination Plugins (core, contrib, custom)

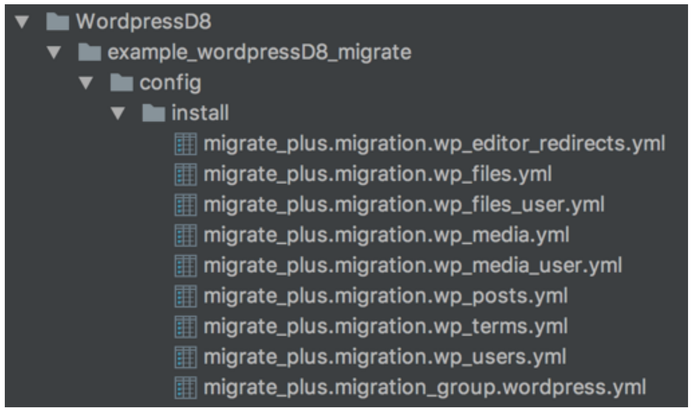
Migrate Plus



D7 to D8



Wordpress to D8



custom migrate module > config > install

SOURCE PLUGINS

- Source plugins make sense of your data - how to extract from the source, define fields, join tables, etc.
- The source plugin used is determined by the migration definition; each migration has a single source plugin.
- Extracts data from a source as rows with properties with each row representing an object to be imported
- Write or customize existing available source plugins
- https://www.drupal.org/docs/8/api/migrate-api/migrate-source-plugins - SQL, CSV, XML, JSON, SOAP

Define the source database connection

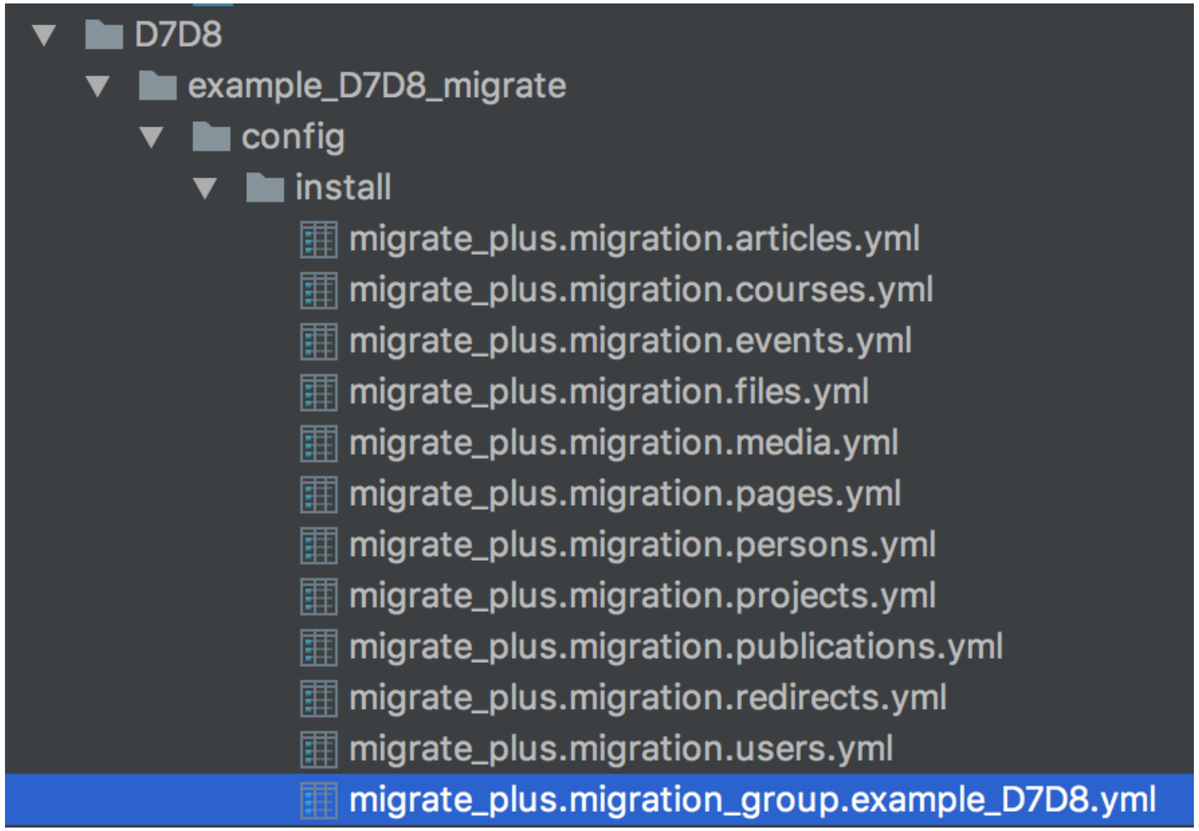
```php
/**
 * Database settings.
 */
$databases['default']['default'] = array(
  'database' => 'D8',
  'username' => '***',
  'password' => '***',
  'prefix' => '',
  'host' => 'localhost',
  'port' => '3306',
  'namespace' => 'Drupal\\Core\\Database\\Driver\\mysql',
  'driver' => 'mysql',
);

// Source db for D7D8 migrations.
$databases['example_D7']['default'] = array(
  'database' => 'D7',
  'username' => '***',
  'password' => '***',
  'prefix' => '',
  'host' => 'localhost',
  'port' => '3306',
  'namespace' => 'Drupal\\Core\\Database\\Driver\\mysql',
  'driver' => 'mysql',
);
```

Note this key - it will be used in your migration YML file

^ Inside your settings.php or settings.local.php file

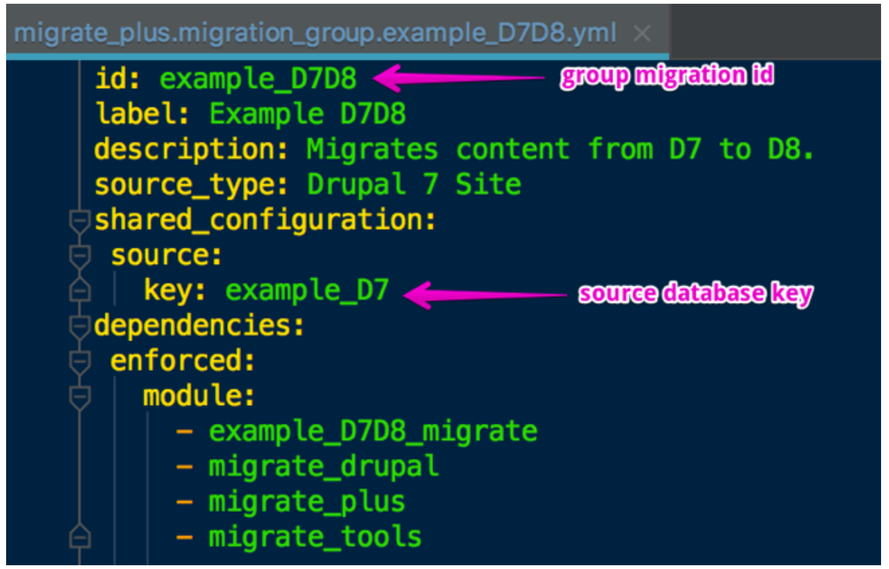
The 'key' configuration option defines the source database connection

File structure for migration YML files:



^ from D7D8 migrations example

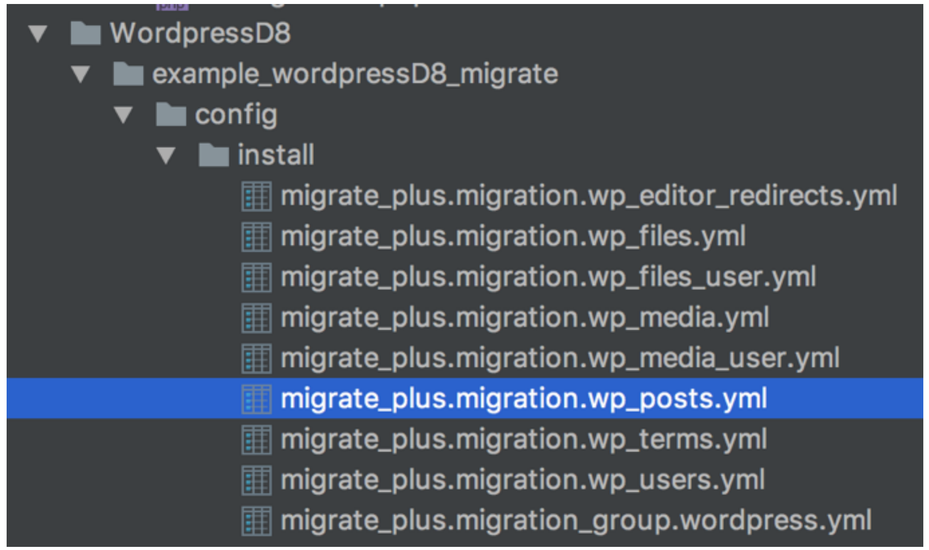
migrate_plus.migration_group.example_D7D8.yml



Note the key can be defined under shared_configuration

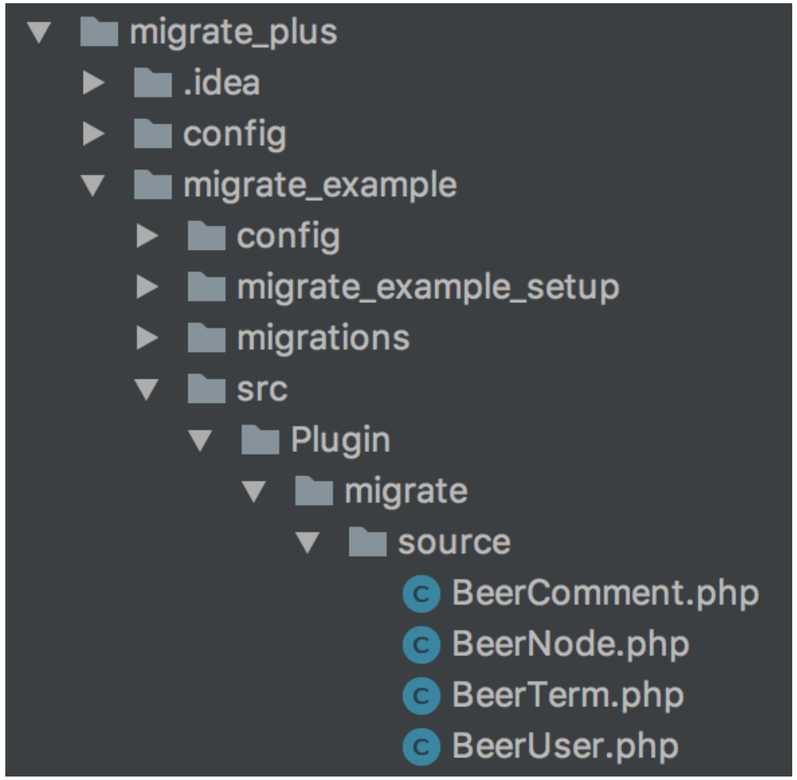
The source database key can be defined in specific migration YML files

```php
/**
 * Database settings.
 */
$databases['default']['default'] = array(
  'database' => 'D8',
  'username' => '***',
  'password' => '***',
  'prefix' => '',
  'host' => 'localhost',
  'port' => '3306',
  'namespace' => 'Drupal\\Core\\Database\\Driver\\mysql',
  'driver' => 'mysql',
);

// Source db for Wordpress to D8 migrations.
$databases['wordpress']['default'] = array(
  'database' => 'wordpress_legacy',
  'username' => '***',
  'password' => '***',
  'prefix' => '',
  'host' => 'localhost',
  'port' => '3306',
  'namespace' => 'Drupal\\Core\\Database\\Driver\\mysql',
  'driver' => 'mysql',
);
```

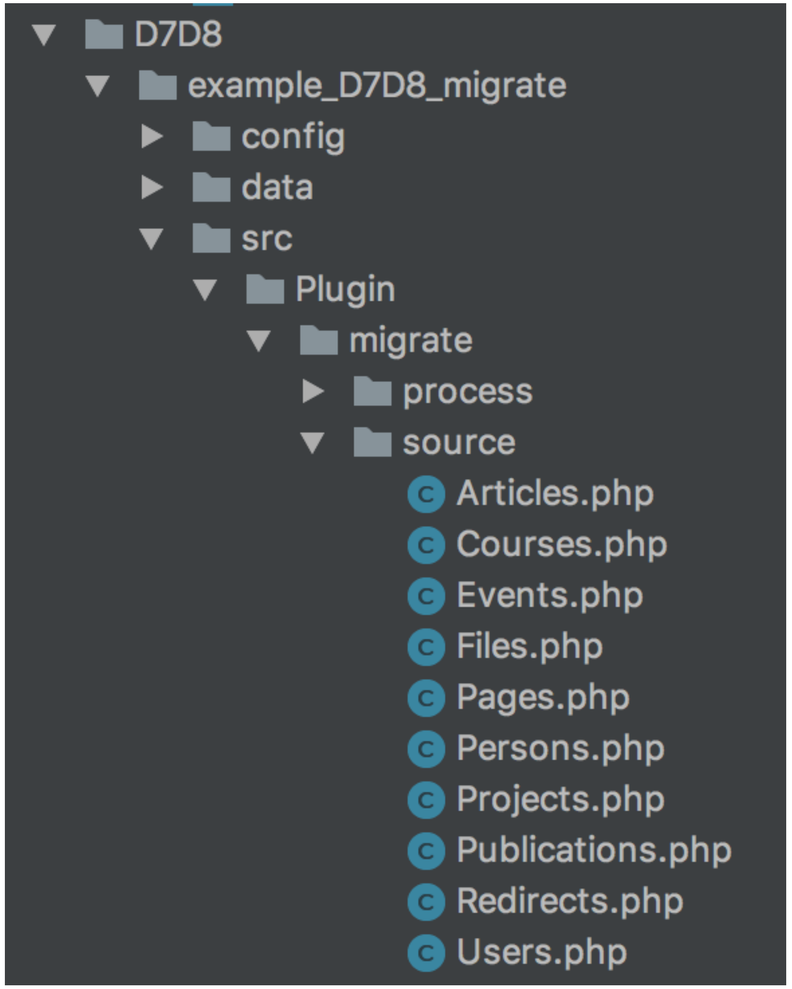


From Wordpress to D8 migrations example

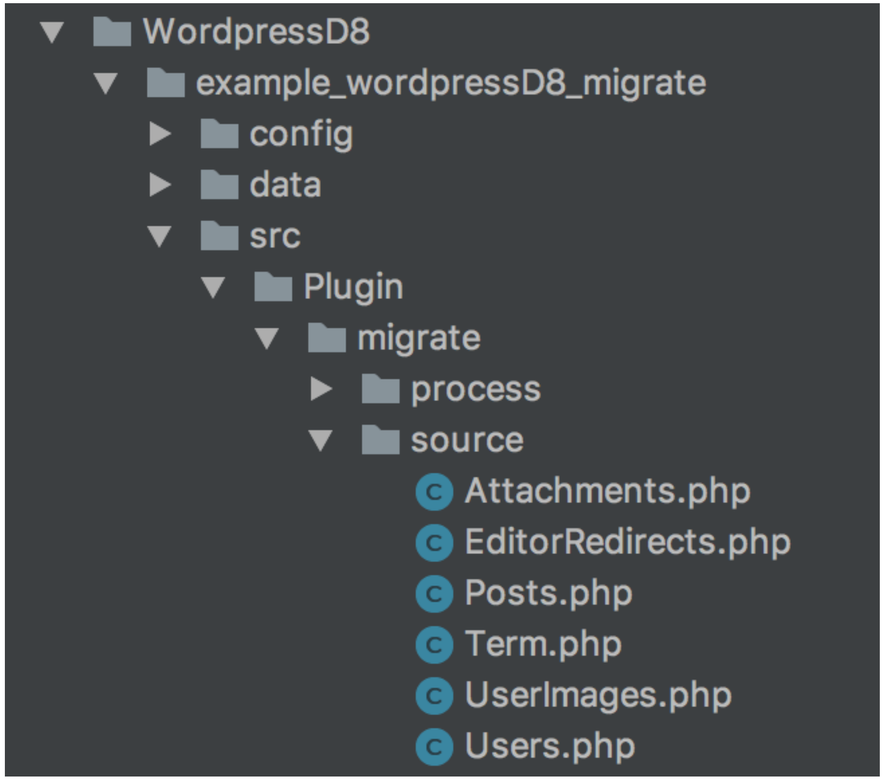
Migrate Plus



D7 to D8



Wordpress to D8



custom migrate module > src > Plugin > migrate > source

SOURCE PLUGINS - SQL

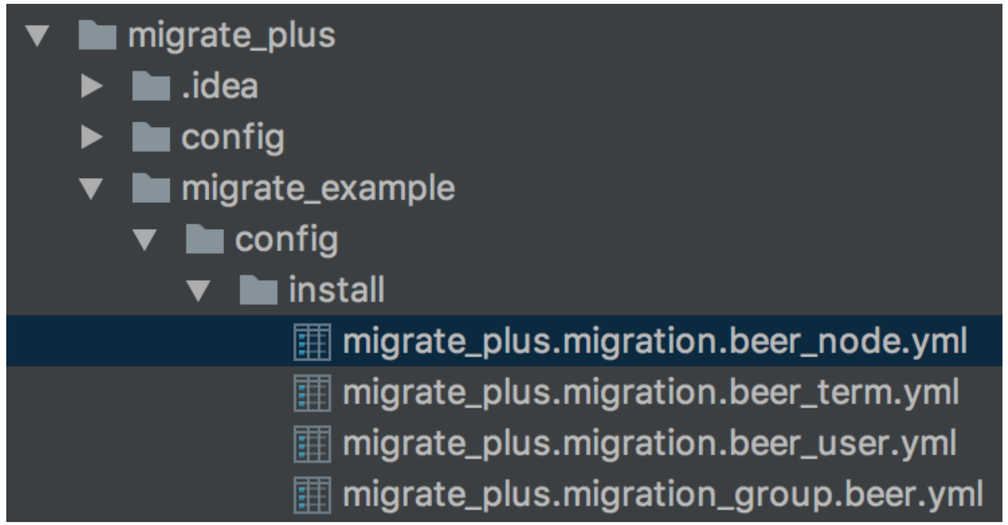
The SQL source plugin must implement three methods:

- query(): Returns the query that selects the data from the source database.
- fields(): Returns available fields on the source.
- getIds(): Defines the source fields uniquely identifying a source row.

Migrating data from a SQL source:
https://www.drupal.org/docs/8/api/migrate-api/migrate-source-plugins/migrating-data-from-a-sql-source

Migrate Plus: migrate_example

config > install > migrate_plus.migration.beer_node.yml




```yaml
# Migration configuration for beer content.
id: beer_node
label: Beers of the world
migration_group: beer
migration_tags:
  - example
source:
  plugin: beer_node
destination:
  plugin: entity:node
process:
  # Hardcode the destination node type (bund
  type:
    plugin: default_value
    default_value: migrate_example_beer
  title: name
  nid: bid
  uid:
    plugin: migration_lookup
    migration: beer_user
    source: aid
  sticky:
    plugin: default_value
    default_value: 0
  field_migrate_example_country: countries
  field_migrate_example_beer_style:
    plugin: migration_lookup
    migration: beer_term
    source: terms
```

source plugin id for beer content (nodes)

```php
<?php

namespace Drupal\migrate_example\Plugin\migrate\source;

use Drupal\migrate\Plugin\migrate\source\SqlBase;
use Drupal\migrate\Row;

/**
 * Source plugin for beer content.
 *
 * @MigrateSource(
 *   id = "beer_node"
 * )
 */
class BeerNode extends SqlBase {

  /**
   * {@inheritdoc}
   */
  public function query() {
    // An important point to note is that your query *must* return a single row
    // for each item to be imported. Here we might be tempted to add a join to
    // migrate_example_beer_topic_node in our query, to pull in the
    // relationships to our categories. Doing this would cause the query to
    // return multiple rows for a given node, once per related value, thus
    // processing the same node multiple times, each time with only one of the
    // multiple values that should be imported. To avoid that, we simply query
    // the base node data here, and pull in the relationships in prepareRow()
    // below.
    $fields = [
      'bid',
      'name',
      'body',
      'excerpt',
      'aid',
      'countries',
      'image',
      'image_alt',
      'image_title',
      'image_description',
    ];
    $query = $this->select( table: 'migrate_example_beer_node', alias: 'b')
      ->fields( table_alias: 'b', $fields);
    return $query;
  }
```

migrate source id

```php
  /**
   * {@inheritdoc}
   */
  public function fields() {
    $fields = [
      'bid' => $this->t( string: 'Beer ID'),
      'name' => $this->t( string: 'Name of beer'),
      'body' => $this->t( string: 'Full description of the beer'),
      'excerpt' => $this->t( string: 'Abstract for this beer'),
      'aid' => $this->t( string: 'Account ID of the author'),
      'countries' => $this->t( string: 'Countries of origin. Multiple values, delimited by pipe'),
      'image' => $this->t( string: 'Image path'),
      'image_alt' => $this->t( string: 'Image ALT'),
      'image_title' => $this->t( string: 'Image title'),
      'image_description' => $this->t( string: 'Image description'),
      // Note that this field is not part of the query above - it is populated
      // by prepareRow() below. You should document all source properties that
      // are available for mapping after prepareRow() is called.
      'terms' => $this->t( string: 'Applicable styles'),
    ];

    return $fields;
  }

  /**
   * {@inheritdoc}
   */
  public function getIds() {
    return [
      'bid' => [
        'type' => 'integer',
        'alias' => 'b',
      ],
    ];
  }
```

Migrate Plus: migrate_example

src > Plugin > migrate > source > BeerNode.php

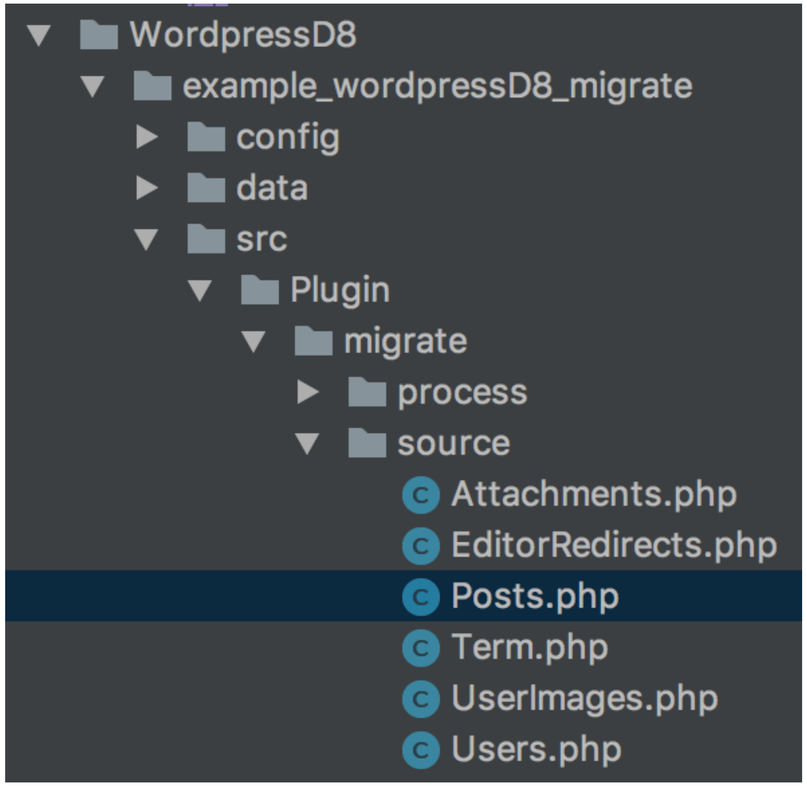
Source properties can be added in the source plugin by implementing the prepareRow() method:

```php
  /**
   * {@inheritdoc}
   */
  public function prepareRow(Row $row) {
    // As explained above, we need to pull the style relationships into our
    // source row here, as an array of 'style' values (the unique ID for
    // the beer_term migration).
    $terms = $this->select( table: 'migrate_example_beer_topic_node',  alias: 'bt')
      ->fields( table_alias: 'bt', ['style'])
      ->condition( field: 'bid', $row->getSourceProperty( property: 'bid'))
      ->execute()
      ->fetchCol();
    $row->setSourceProperty( property: 'terms', $terms);

    // As we did for favorite beers in the user migration, we need to explode
    // the multi-value country names.
    if ($value = $row->getSourceProperty( property: 'countries')) {
      $row->setSourceProperty( property: 'countries', explode( delimiter: '|', $value));
    }
    return parent::prepareRow($row);
  }

}
```

Migrate Plus: migrate_example

src > Plugin > migrate > source > BeerNode.php

Source properties can be added in the source plugin by the prepareRow() method:



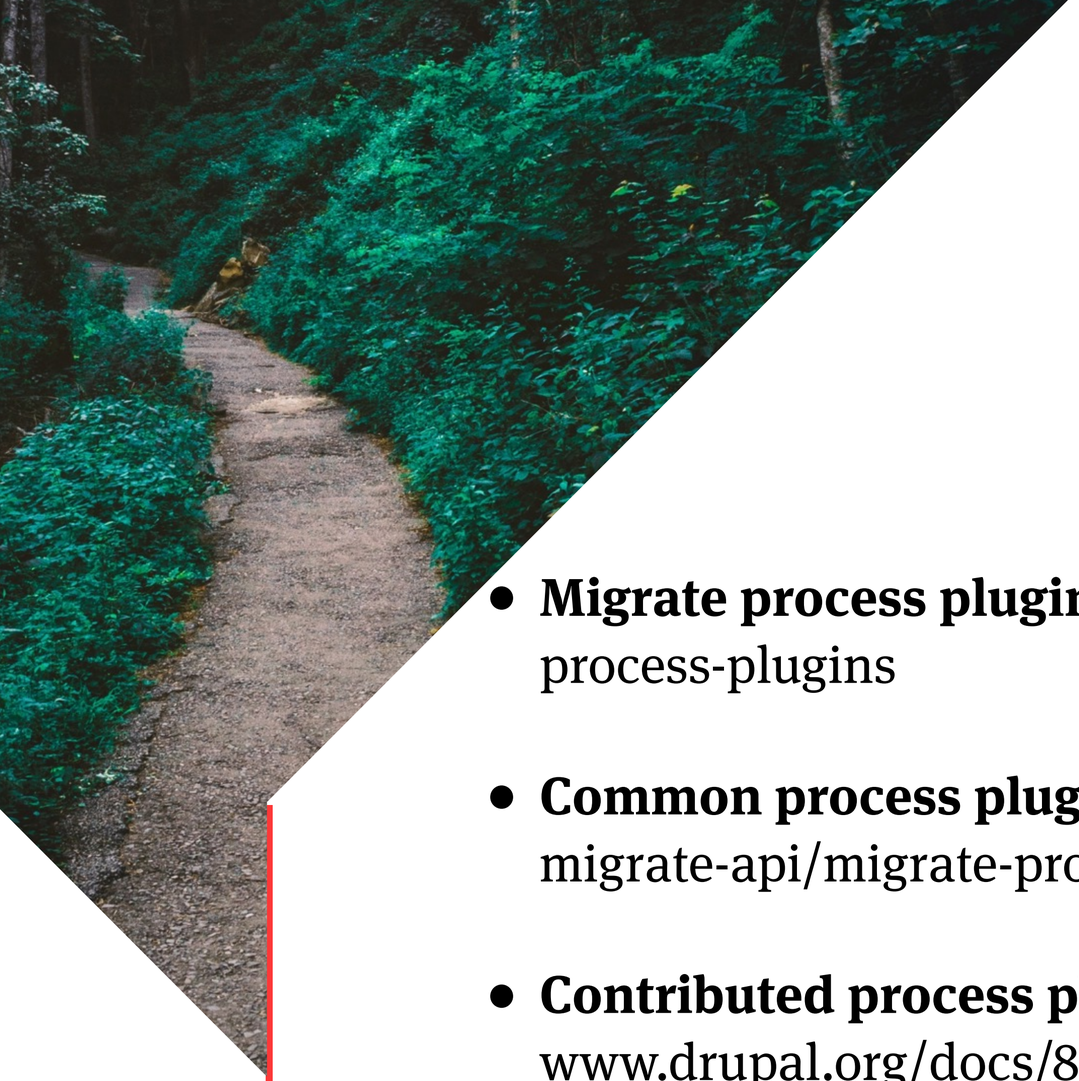

```php
public function prepareRow(Row $row) {
  // Set tag terms.
  $query = $this->select('wp_term_relationships', 'tr');
  $query->fields('tr', ['term_taxonomy_id']);
  $query->join('wp_term_taxonomy', 'tt', 'tr.term_taxonomy_id = tt.term_id');
  $query->condition('tr.object_id', $row->getSourceProperty('id'));
  $query->condition('tt.taxonomy', Term::VOCABULARIES, 'IN');
  $query->orderBy('term_order');
  $row->setSourceProperty('tags', $query->execute()->fetchCol());
  // Store the post attachment ID.
  $query = $this->select('wp_posts', 'p');
  $query->fields('p', ['id']);
  $query->condition('p.post_parent', $row->getSourceProperty('id'));
  $query->orderBy('p.id');
  $query->range(0, 1);
  $row->setSourceProperty('post_attachment', current($query->execute()->fetchCol()));
  // Set the author text.
  $this->setAuthor($row);
  // Set metatags.
  $this->setMetatags($row);
  return parent::prepareRow($row);
}
```

^ from Wordpress to D8 example
src > Plugin > migrate > source > Posts.php

Migrate Source CSV

- plugin: csv => The source plugin ID we want to use.
- path: public://csv/example.csv => Path to where the csv is located.
- header_row_count: 1 => How many lines to skip at the beginning of the CSV because they are considered the header row(s).
- keys => A yml array of column names that make the row unique, so you don't re-import the same thing multiple times.
- column_names => Numeric indexed yml array of column names.

```yaml
id: alpha_files
migration_group: example_customD8_alpha
migration_tags:
  - CustomD8 Alpha
label: 'Import Alpha Files'

source:
  plugin: csv
  path: ./modules/custom/example_customD8_migrate_alpha/data/example_customD8_migrate_alpha.csv
  header_row_count: 1
  enclosure: '"'
  keys:
    - filepath
  column_names:
    0:
      url: 'URL Path'
    1:
      filepath: 'File Path'
    2:
      dcrpath: 'Local Path to XML'
    3:
      last-modified: 'Modified Date'
    5:
      keep?: 'Migrate Yes or No'
  constants:
    file_source_uri: 'http://www.alpha.com'

process:
  filter1:
    plugin: check_for_docs
    source: filepath
  filter2:
    plugin: alpha_check_migrate
    source: keep?
  destination_folder:
    plugin: alpha_destination
    source: filepath
  fid:
    plugin: alpha_file_import
    source: url
  filename:
    plugin: file_name
    source: filepath
```



PROCESS PLUGINS

- **Migrate process plugins** - https://www.drupal.org/docs/8/api/migrate-api/migrate-process-plugins
- **Common process plugins provided by core** - https://www.drupal.org/docs/8/api/migrate-api/migrate-process-plugins/list-of-core-migrate-process-plugins
- **Contributed process plugins** (Migrate Plus + Migrate Process Extra) - https://www.drupal.org/docs/8/api/migrate-api/migrate-process-plugins/list-of-process-plugins-provided-by-migrate-plus
- **Custom process plugins** - https://www.drupal.org/docs/8/api/migrate-api/migrate-process/writing-a-process-plugin

```yaml
# Migration configuration for beer content.
id: beer_node
label: Beers of the world
migration_group: beer
migration_tags:
  - example
source:
  plugin: beer_node
destination:
  plugin: entity:node
process:
  type:
    plugin: default_value
    default_value: migrate_example_beer
  title: name
  nid: bid
  uid:
    plugin: migration_lookup
    migration: beer_user
    source: aid
  sticky:
    plugin: default_value
    default_value: 0
  field_migrate_example_country: countries
  field_migrate_example_beer_style:
    plugin: migration_lookup
    migration: beer_term
    source: terms
  'body/value': body
  'body/summary': excerpt

migration_dependencies:
  required:
    - beer_term
    - beer_user
dependencies:
  enforced:
    module:
      - migrate_example
```

field mapping to D8 from source w/ core process plugins

Process Plugins

The process key of a migration defines how the destination should be constructed from the source data.

The value of the process key is an associative array, and each key is a destination property.

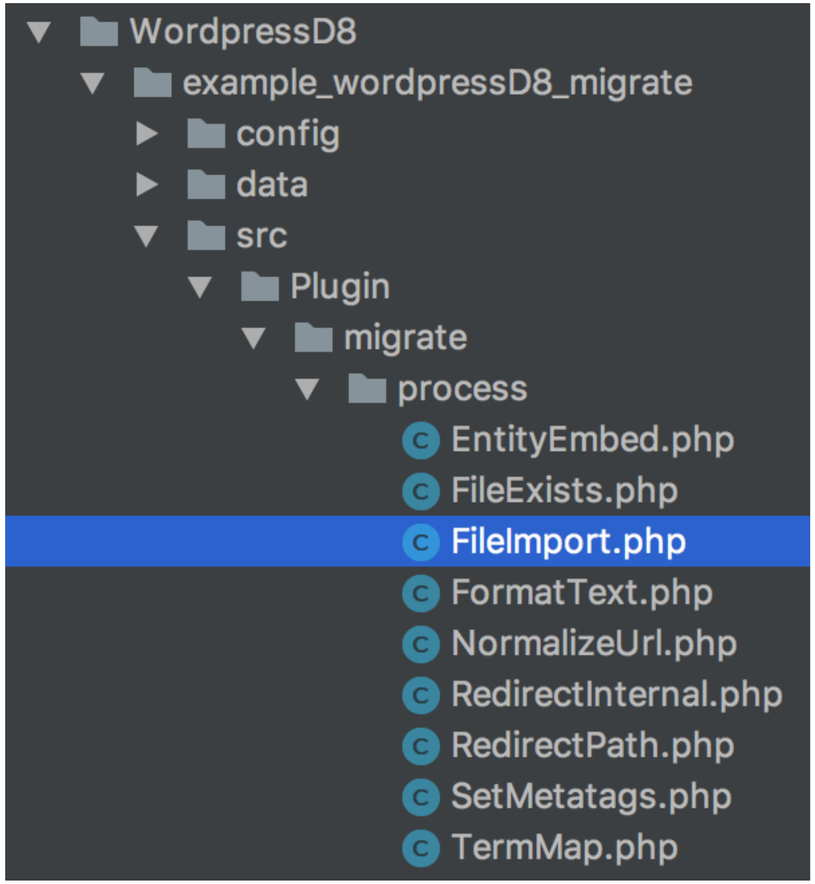
The values associated with each key describe how the destination value is created.

Migrate Plus: migrate_example

config > install > migration_plus.migration.beer_node.yml

Migrate process overview - https://www.drupal.org/node/2129651

Custom Process Plugin

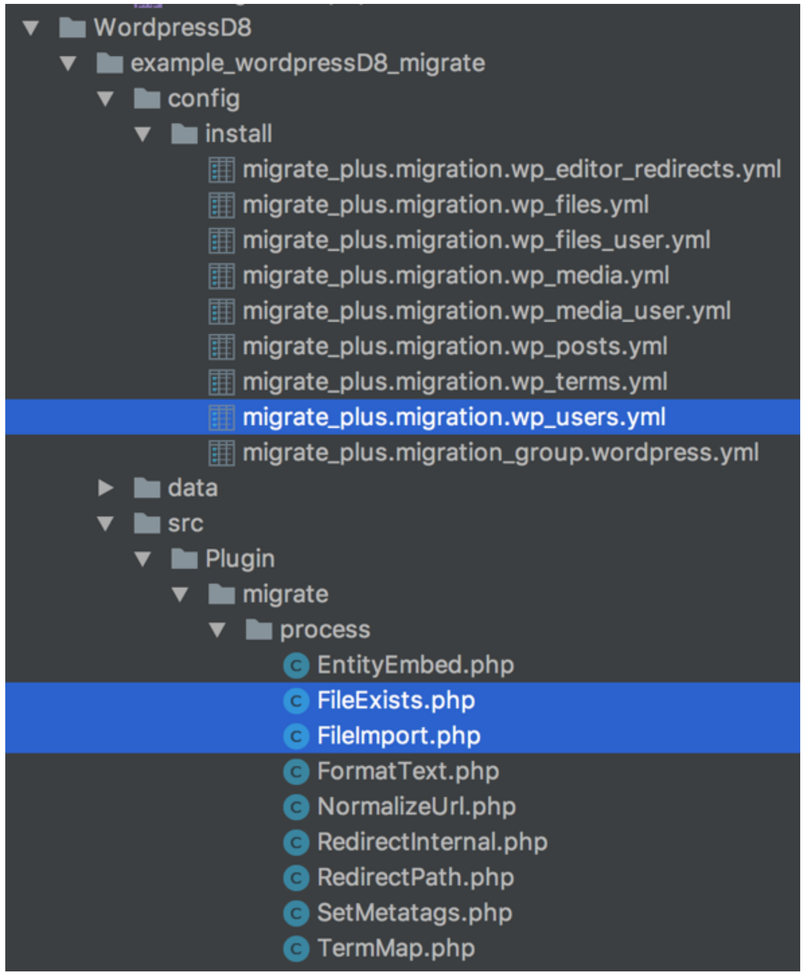



```php
<?php

namespace Drupal\example_wordpressD8_migrate\Plugin\migrate\process;

use Drupal\migrate\MigrateExecutableInterface;
use Drupal\migrate\MigrateSkipProcessException;
use Drupal\migrate\MigrateSkipRowException;
use Drupal\migrate\ProcessPluginBase;
use Drupal\migrate\Row;

/**
 * Import a file as a side-effect of a migration.
 *
 * @MigrateProcessPlugin(
 *   id = "file_import"
 * )
 */
class FileImport extends ProcessPluginBase {

  /**
   * {@inheritdoc}
   */
  public function transform($value, MigrateExecutableInterface $migrate_executable, Row $row, $destination_property) {
    if (empty($value)) {
      // Skip this item if there is no URL.
      throw new MigrateSkipRowException();
    }

    // Save the file, return its ID.
    $file = system_retrieve_file($value, destination: 'public://', managed: TRUE, replace: FILE_EXISTS_REPLACE);
    if (!$file) {
      // Skip this item if saving the file fails.
      throw new MigrateSkipProcessException();
    }
    return $file->id();
  }
}
```

^ from Wordpress to D8 example
src > Plugin > migrate > process > FileImport.php

Core + Custom Process Plugins

wp_users migration uses the file_exists + file_import custom process plugins to transform the source data (user_photo_url) and populate the field_user_photo field



^ from Wordpress to D8 example
src > Plugin > migrate > process > FileExists.php
src > Plugin > migrate > process > FileImport.php

migrate_plus.migration.wp_users.yml

```yaml
id: wp_users
label: Users
migration_tags:
  - Wordpress
migration_group: wordpress
source:
  plugin: users
  key: wordpress
  constants:
    space: ' '
destination:
  plugin: entity:user
process:
  field_user_photo:
    -
      plugin: skip_on_empty
      method: process
      source: user_photo_url
    -
      plugin: file_exists
    -
      plugin: file_import
  password: user_pass
  field_user_display_name:
    plugin: concat
    source:
      - first_name
      - constants/space
      - last_name
```

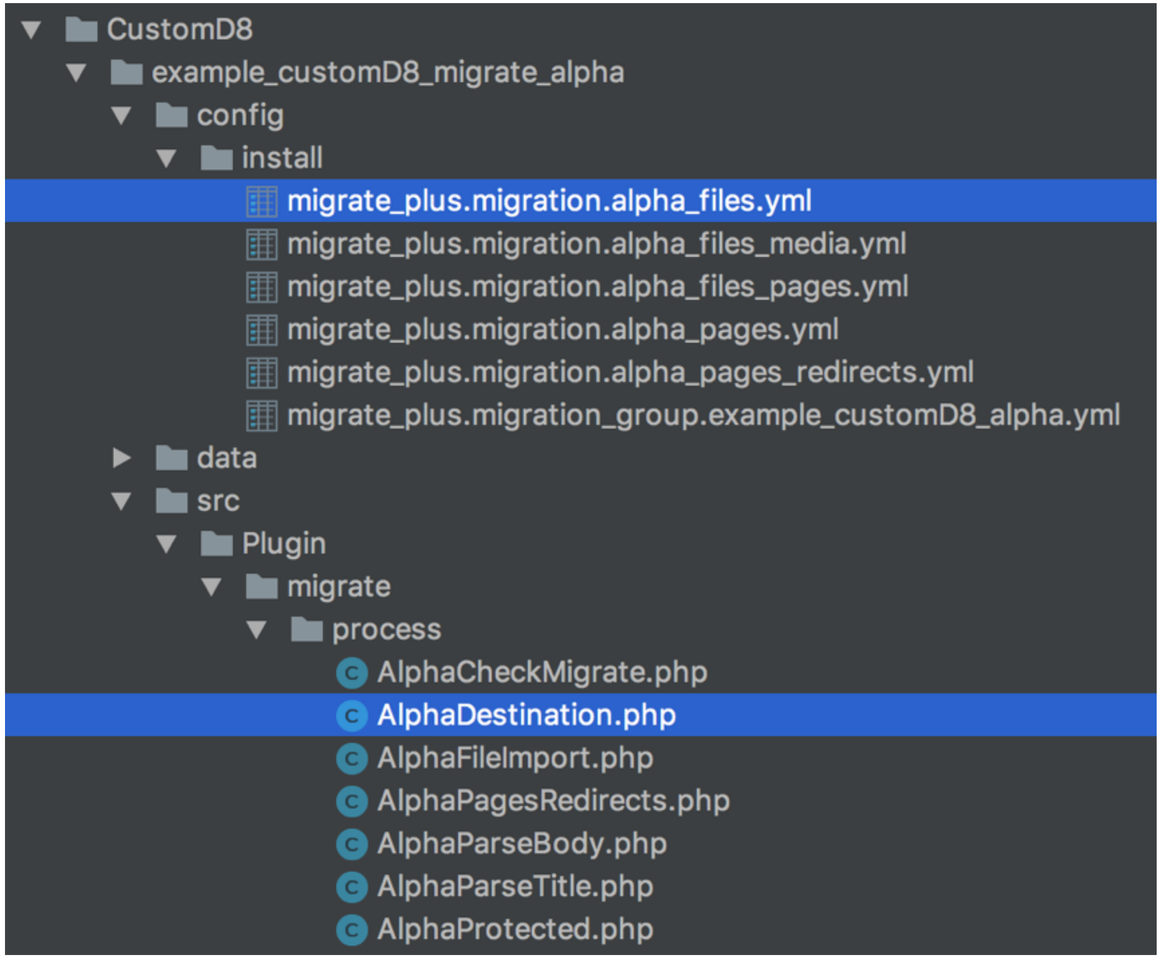
core process plugin

custom process plugins

core process plugin

Core + Custom Process Plugins

Variables can be used in migration configuration.



^ from Custom to D8 Alpha example
src > Plugin > migrate > process > AlphaDestination

migrate_plus.migration.alpha_files.yml

```yaml
process:
  filter1:
    plugin: check_for_docs
    source: filepath
  filter2:
    plugin: alpha_check_migrate
    source: keep?
  destination_folder:
    plugin: alpha_destination
    source: filepath
  fid:
    plugin: alpha_file_import
    source: url
  filename:
    plugin: file_name
    source: filepath
  uri:
    plugin: concat
    delimiter: /
    source:
      - '@destination_folder'
      - '@filename'
  created:
    plugin: callback
    source: last-modified
    callable: strtotime
  changed:
    plugin: callback
    callable: strtotime
  uid:
    plugin: default_value
    default_value: 1
  status:
    plugin: default_value
    default_value: 1

destination:
  plugin: 'entity:file'
```

custom process plugin returns a new destination property called destination_folder

The URI uses a core process plugin concat w/ a delimiter to concatenate the folder with the filename - this is the value of URI

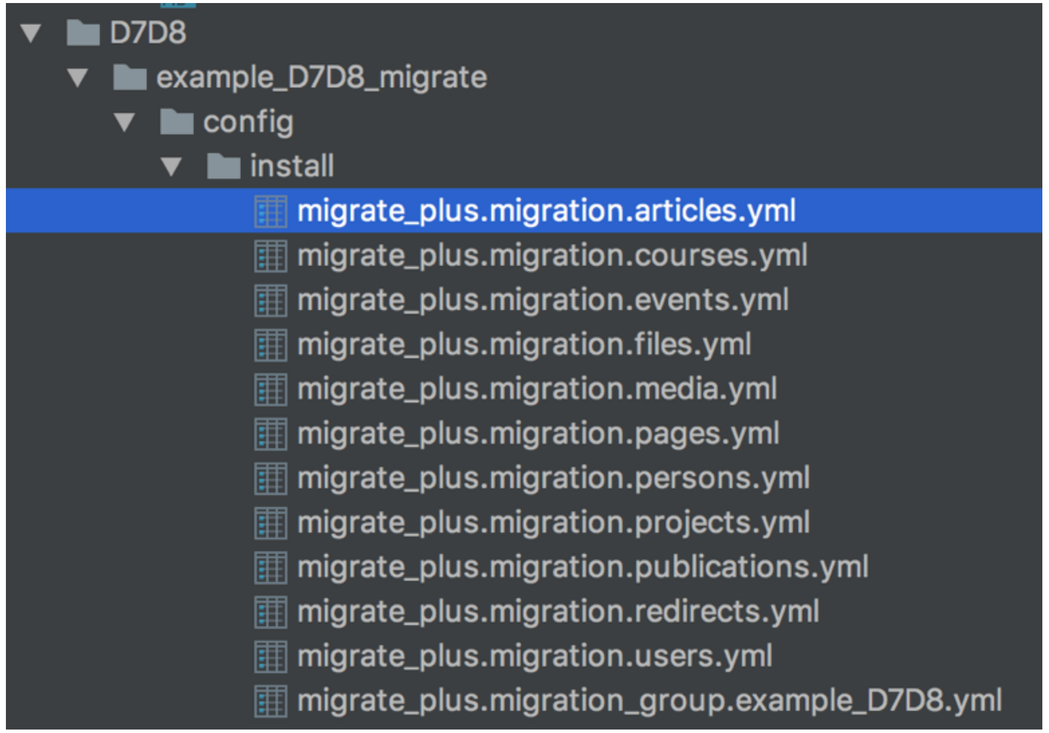
Process Plugins provided by core:

- **array_build** - Builds an array based on the key and value configuration.
- **callback** - Passes the source value to a callback.
- **concat** - Concatenates a set of strings.
- **default_value** - Returns a given default value if the input is empty.
- **download** - Downloads a file from a HTTP(S) remote location into the local file system.
- **entity_exists** - Checks if a given entity exists.
- **explode** - Splits the source string into an array of strings, using a delimiter.
- **extract** - Extracts a value from an array.
- **file_copy** - Copies or moves a local file from one place into another.
- **flatten** - Flattens the source value.
- **format_date** - Converts date/datetime from one format to another.
- **get** - Gets the source value.
- **log** - Logs values without changing them.
- **machine_name** - Creates a machine name.
- **menu_link_parent** - Figures out menu link parent plugin IDs. Additional documentation.
- **migration_lookup** - Looks up the value of a property based on a previous migration.
- **MakeUniqueBase** - Ensures the source value is unique. Abstract base class. See make_unique_entity_field.
- **make_unique_entity_field** - Ensures the source value is made unique against an entity field.
- **route** - Sets the destination route information based on the source link_path.
- **skip_on_empty** - Skips processing the current row when the input value is empty.
- **skip_row_if_not_set** - Skips processing the current row when a source value is not set.
- **static_map** - Changes the source value based on a static lookup map.
- **subset** - Returns a substring of the input value.
- **sub_process** - Runs an array of arrays through its own process pipeline.
- **url_encode** - URL-encodes the input value.

List of core Migrate process plugins - https://www.drupal.org/docs/8/api/migrate-api/migrate-process-plugins/list-of-core-migrate-process-plugins

Core Process Plugins

Examples of the migrate_lookup + sub_process process plugins provided by core.



^ from D7 to D8 example
config > install > migrate_plus.migration.articles.php

migrate_plus.migration.articles.yml

```yaml
field_related_persons:
  plugin: sub_process
  source: field_persons
  process:
    target_id:
      plugin: migration_lookup
      migration: persons
      no_stub: true
      source: nid
field_related_projects:
  plugin: sub_process
  source: field_projects
  process:
    target_id:
      plugin: migration_lookup
      migration: projects
      no_stub: true
      source: nid
field_related_topics:
  plugin: sub_process
  source: field_topics
  process:
    target_id: tid
field_related_publications:
  plugin: sub_process
  source: field_publications
  process:
    target_id:
      plugin: migration_lookup
      migration: publications
      no_stub: true
      source: nid
field_related_courses:
  plugin: sub_process
  source: field_courses
  process:
    target_id:
      plugin: migration_lookup
      migration: courses
      no_stub: true
      source: nid
```

Pass additional needed values to the process plugin

```yaml
process:
  field_metatag_description:
    plugin: parse_xml
    source: dcrpath
    module: example_customD8_migrate_alpha
```

^ from Custom to D8 Alpha example
config > install > migrate_plus.migration.alpha_pages.yml

ParseXml.php

```php
<?php

namespace Drupal\example_customD8_migrate\Plugin\migrate\process;

use ...

/**
 * Load xml and import values into specified fields.
 *
 * @MigrateProcessPlugin(
 *   id = "parse_xml"
 * )
 */
class ParseXml extends ProcessPluginBase {

  /**
   * {@inheritdoc}
   */
  public function transform($value, MigrateExecutableInterface $migrate_executable, Row $row, $destination_property) {

    if (empty($value)) {
      // Skip this item if there is no file path.
      throw new MigrateSkipRowException();
    }
    $matched_field = '';
    // If the module name is N/A, throw error.
    if (empty($module = $this->configuration['module'])) {
      throw new RequirementsException( message: 'No module name supplied.');
    }
    // Set the relative page to the DCR/XML.
    $xml_path_and_name = 'modules/custom/' . $module . '/data/xml/' . $value;
```

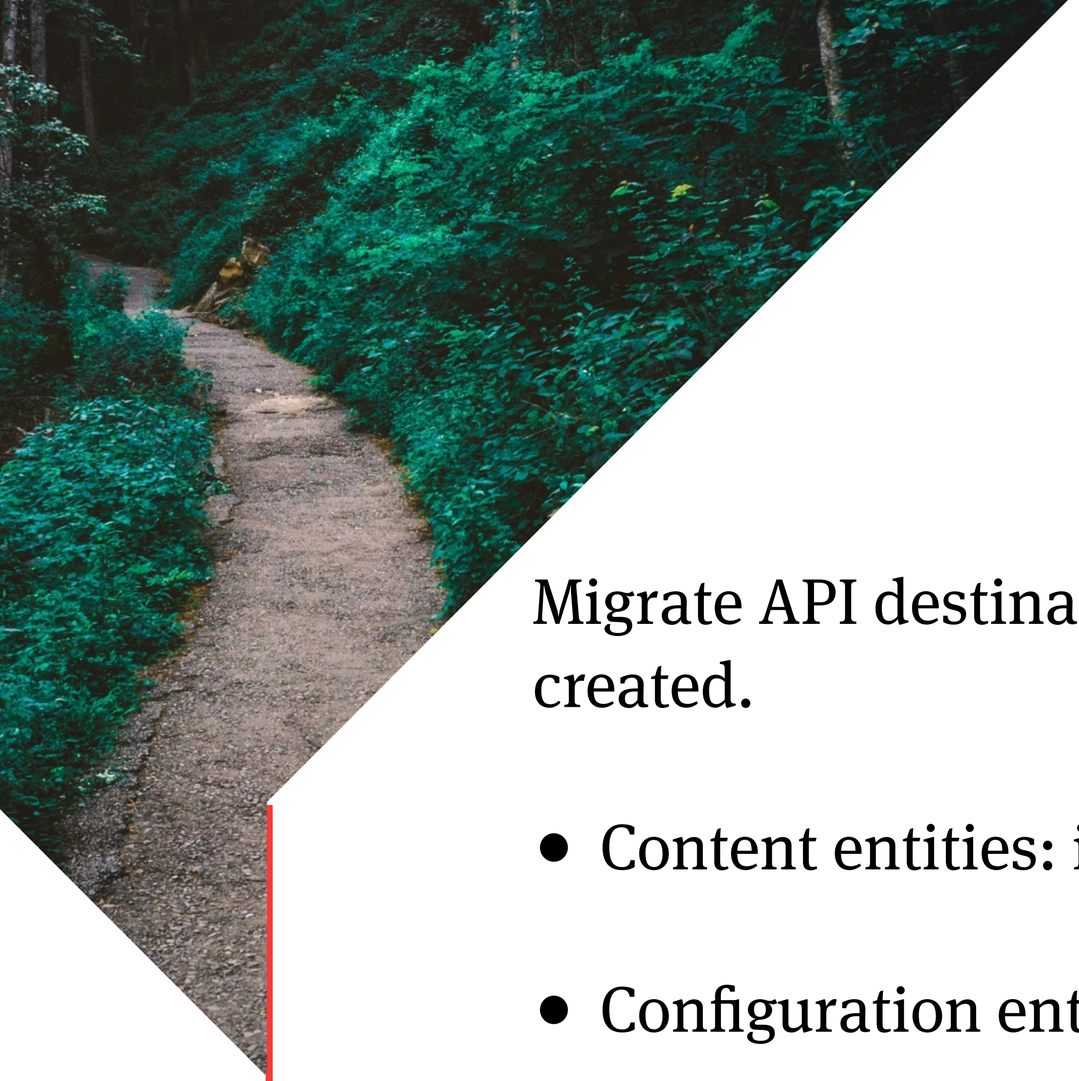
Process plugin id

Passed into the parse_xml process plugin

Process Plugins provided by Migrate Plus:

- **array_pop** - The "extract" plugin in core can extract array values when indexes are already known. This plugin helps extract the last value in an array by performing a "pop" operation.
- **array_shift** - The "extract" plugin in core can extract array values when indexes are already known. This plugin helps extract the first value in an array by performing a "shift" operation.
- **entity_lookup** - Allows you to match source data to existing Drupal 8 entities and return their IDs (primarily for populating entity reference fields).
- **entity_generate** - Extends entity_lookup to actually generate an entity from the source data where one does not already exist.
- **file_blob** - Allows you to create a file (and corresponding file entity) from blob data.
- **merge** - Allows you to merge multiple source arrays into one array.
- **multiple_values** - Converts from a single value to multiple so that the next plugin in the process pipeline treats the values individually instead of an array of values.
- **single_value** - Treats an array of values as a single value.
- **skip_on_value** - Like core's skip_on_empty, but allows you to skip either the row or process upon matching (or not) a specific value.
- **str_replace** - Wrapper around str_replace, str_ireplace and preg_replace.
- **transliteration** - process strings through the transliteration service to remove language decorations and accents. Especially helpful with file names.

List of process plugins provided by Migrate Plus - https://www.drupal.org/docs/8/api/migrate-api/migrate-process-plugins/list-of-process-plugins-provided-by-migrate-plus



DESTINATION PLUGINS

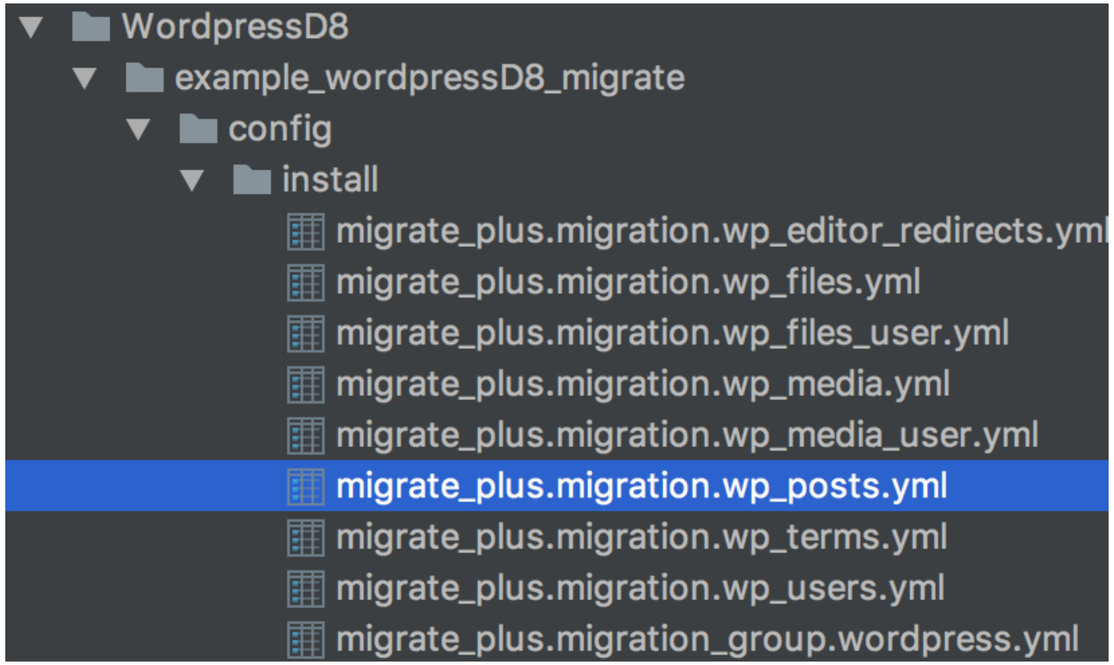
Migrate API destination plugins inform which Drupal entities are created.

- Content entities: i.e. nodes, users, taxonomy terms and files
- Configuration entities: i.e. content type and field definitions

Destination entity type is typically defined as entity:<entity_type>, for example **entity:node**

Migrate destination plugins & examples - https://www.drupal.org/docs/8/api/migrate-api/migrate-destination-plugins-examples

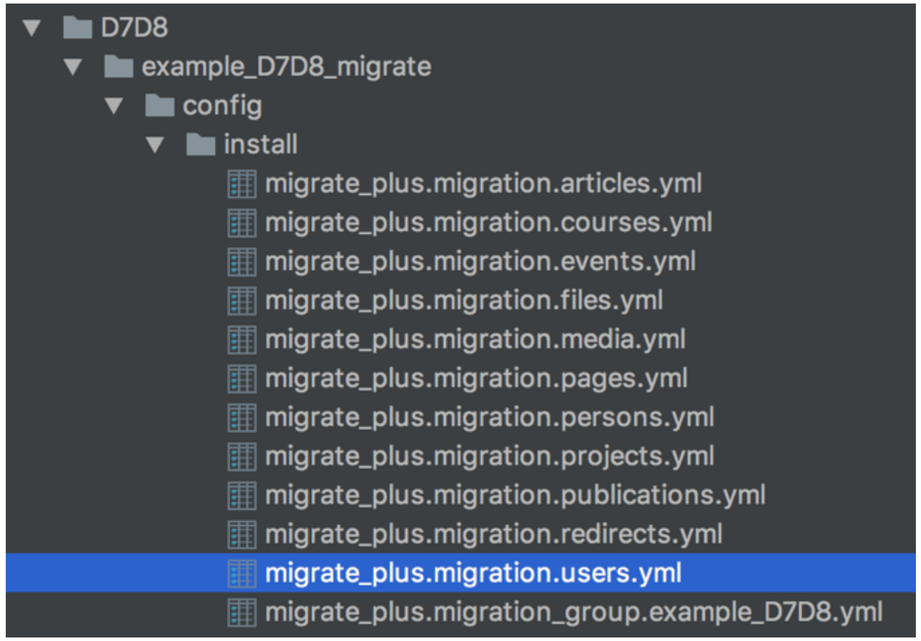
Destination Plugins



migrate_plus.migration.wp_posts.yml

```yaml
id: wp_posts
label: Posts
migration_tags:
  - Wordpress
migration_group: wordpress
source:
  plugin: posts
  key: wordpress
  constants:
    slash: '/'
destination:
  plugin: entity:node
  bundle: post
process:
  type:
    plugin: default_value
    default_value: post
  created:
    plugin: callback
    callable: strtotime
    source: post_date
  changed:
    plugin: callback
    callable: strtotime
    source: post_modified
```

Destination Plugins



migrate_plus.migration.users.yml

```yaml
id: users
label: Users
migration_tags:
  - D7
migration_group: example_D7D8
source:
  plugin: users
destination:
  plugin: entity:user
process:
  name: name
  pass: pass
  mail: mail
  status: status
  created: created
  changed: changed
  access: access
  login: login
  timezone: timezone
  langcode: language
  init: init
  roles:
    plugin: static_map
    source: roles
    map:
      2: authenticated
      3: editor
      4: administrator
```

migrate_plus.migration.alpha_files.yml

```yaml
process:
  filter1:
    plugin: check_for_docs
    source: filepath
  filter2:
    plugin: alpha_check_migrate
    source: keep?
  destination_folder:
    plugin: alpha_destination
    source: filepath
  fid:
    plugin: alpha_file_import
    source: url
  filename:
    plugin: file_name
    source: filepath
  uri:
    plugin: concat
    delimiter: /
    source:
      - '@destination_folder'
      - '@filename'
  created:
    plugin: callback
    source: last-modified
    callable: strtotime
  changed:
    plugin: callback
    callable: strtotime
  uid:
    plugin: default_value
    default_value: 1
  status:
    plugin: default_value
    default_value: 1

destination:
  plugin: 'entity:file'
```

migrate_plus.migration.alpha_files_media.yml

```yaml
id: alpha_files_media
migration_group: example_customD8_alpha
migration_tags:
  - Example CustomD8 Alpha
label: 'Generate Alpha File Media Entities'

source:
  plugin: csv
  path: ./modules/custom/example_customD8_migrate_alpha/data/example_customD8_migrate_alpha.csv
  header_row_count: 1
  enclosure: '"'
  keys:
    - filepath
  column_names:
    0:
      url: 'URL Path'
    1:
      filepath: 'File Path'
    2:
      dcrpath: 'Local Path to XML'
    3:
      last-modified: 'Modified Date'
    5:
      keep?: 'Migrate Yes or No'
    6:
      title: 'Title'
    7:
      description: 'Description'
  constants:
    file_dest_uri: 'public://alpha'

destination:
  plugin: entity:media
process:
  filter1:
    plugin: check_for_docs
    source: filepath
  filter2:
    plugin: alpha_check_migrate
    source: keep?
  bundle:
    plugin: default_value
    default_value: file
  filename:
    plugin: file_name
    source: filepath
```

CONFIGURATION

IMPORT / EXPORT

drush cim --partial
--source=/{pathtoproject}/{webroot}/modules/custom/
{custom_migrate_module}/config/install/ -y

drush cex

RESOURCES for RUNNING MIGRATIONS

- https://www.drupal.org/docs/8/api/migrate-api/executing-migrations
- https://drupalize.me/tutorial/drupal-drupal-migration-drush
- https://drupalize.me/tutorial/run-custom-migrations

DRUSH

Run migrations from command line

Required: Migrate Tools

Migrate Drush Commands

- **migrate-status** - Lists migrations and their status.
- **migrate-import** - Performs import operations.
- **migrate-rollback** - Performs rollback operations.
- **migrate-stop** - Cleanly stops a running operation.
- **migrate-reset-status** - Sets a migration status to Idle if it's gotten stuck.
- **migrate-messages** - Lists any messages associated with a migration import.

https://www.drupal.org/project/migrate_tools

Executing, Rolling back, Flags

```
drush ms (drush migrate-status)

drush migrate-import {MIGRATION_ID}

drush migrate-import files --feedback="500 items”

drush migrate-import files --limit=10

drush migrate-rollback articles
```

drush ms

Group: Import from Drupal 7 (migrate_drupal_7)	Status	Total	Imported	Unprocessed	Last imported
upgrade_block_content_type	Idle	1	1	0	2017-01-11 17:16:11
upgrade_d7_dblog_settings	Idle	1	1	0	2017-01-11 17:16:11
upgrade_d7_filter_settings	Idle	1	1	0	2017-01-11 17:16:11
upgrade_d7_global_theme_settings	Idle	1	1	0	2017-01-11 17:16:11
upgrade_d7_image_settings	Idle	0	1	-1	2017-01-11 17:16:11
upgrade_d7_image_styles	Idle	6	5	0	2017-01-11 17:16:11
upgrade_d7_menu	Idle	8	8	0	2017-01-11 17:16:11
upgrade_d7_node_settings	Idle	1	1	0	2017-01-11 17:16:11
upgrade_d7_search_settings	Idle	0	0	-1	2017-01-11 17:16:11
upgrade_d7_system_authorize	Idle	0	1	-1	2017-01-11 17:16:11
upgrade_d7_system_cron	Idle	0	1	-1	2017-01-11 17:16:11
upgrade_d7_system_date	Idle	1	1	0	2017-01-11 17:16:11
upgrade_d7_system_file	Idle	1	0	0	2017-01-11 17:16:11
upgrade_d7_system_mail	Idle	1	0	0	2017-01-11 17:16:11
upgrade_d7_system_performance	Idle	1	1	0	2017-01-11 17:16:11
upgrade_d7_url_alias	Idle	4144	4144	0	2017-01-13 10:42:44

DEBUGGING & TROUBLESHOOTING

RESOURCES for DEBUGGING

- Xdebug - set breakpoints and inspect values during migrations
- https://www.drupal.org/project/migrate_devel
- https://www.drupal.org/docs/8/api/migrate-api/debugging-migrations
- https://www.mtech-llc.com/blog/lucas-hedding/troubleshooting-drupal-8-migration

“Problems [can] be demystified by using Xdebug and putting breakpoints in two spots in Core's MigrateExecutable.

First is in the ***::import()*** *method where it rewinds the source and then processes it.*

The second place I regularly put a breakpoint is in ***::processRow()****...[If] I know which process plugin is breaking, I might put a breakpoint in it directly.*

For example, the ***sub_process*** *or* ***migration_lookup*** *process plugins tend to be complicated and a common place...to drop a breakpoint.”*

– Lucas Hedding

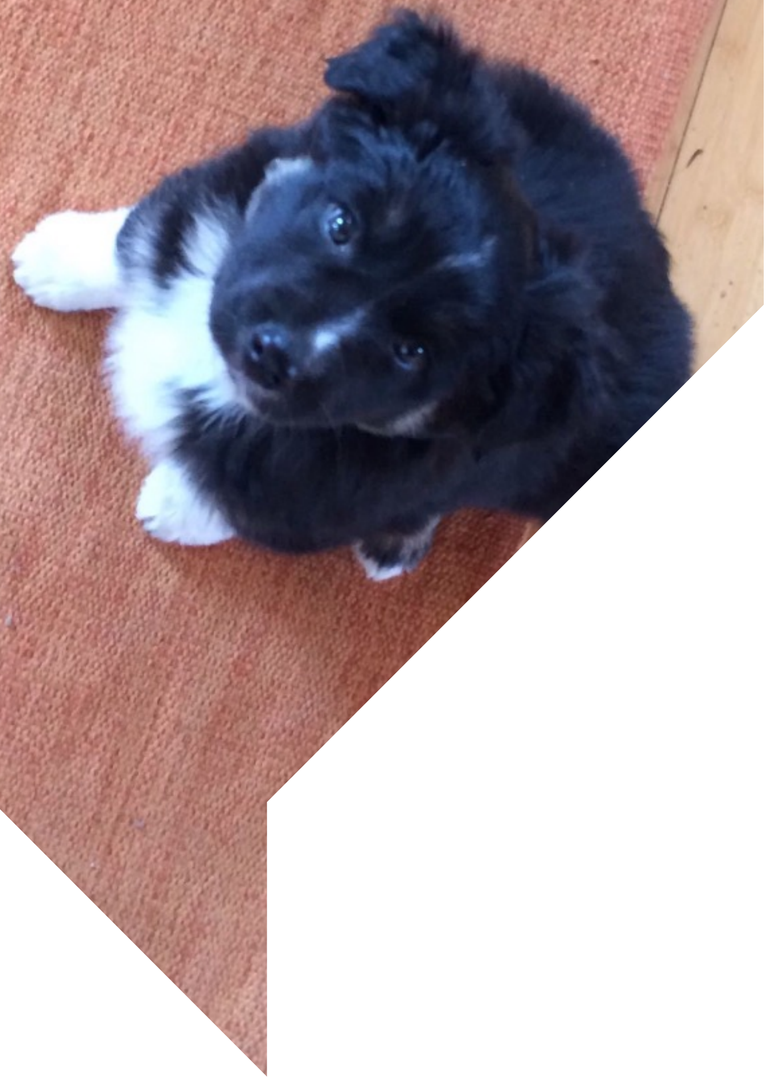
https://www.mtech-llc.com/blog/lucas-hedding/troubleshooting-drupal-8-migration

RESOURCES

- https://www.drupal.org/docs/8/api/migrate-api
- https://drupalize.me/tutorial/introduction-migrations-drupal-8
- https://www.drupal.org/project/migrate_plus
  - migrate_example
  - migrate_example_advanced

ACKNOWLEDGEMENTS

- https://www.drupal.org/docs/8/api/migrate-api
- https://drupalize.me/tutorial/introduction-migrations-drupal-8
- Mike Ryan https://www.drupal.org/u/mikeryan
- Adam Zimmermann https://chromatichq.com/users/adam-zimmermann
- Les Cordell https://www.linkedin.com/in/leslie-cordell-a1621964



THANK YOU!

QUESTIONS?

CHROMATIC | https://chromatichq.com | @chromatichq

Clare Ming | @bodhicitta | https://github.com/cjming