DRUPAL CAMP PA 2017

MIGRATION BEST PRACTICES

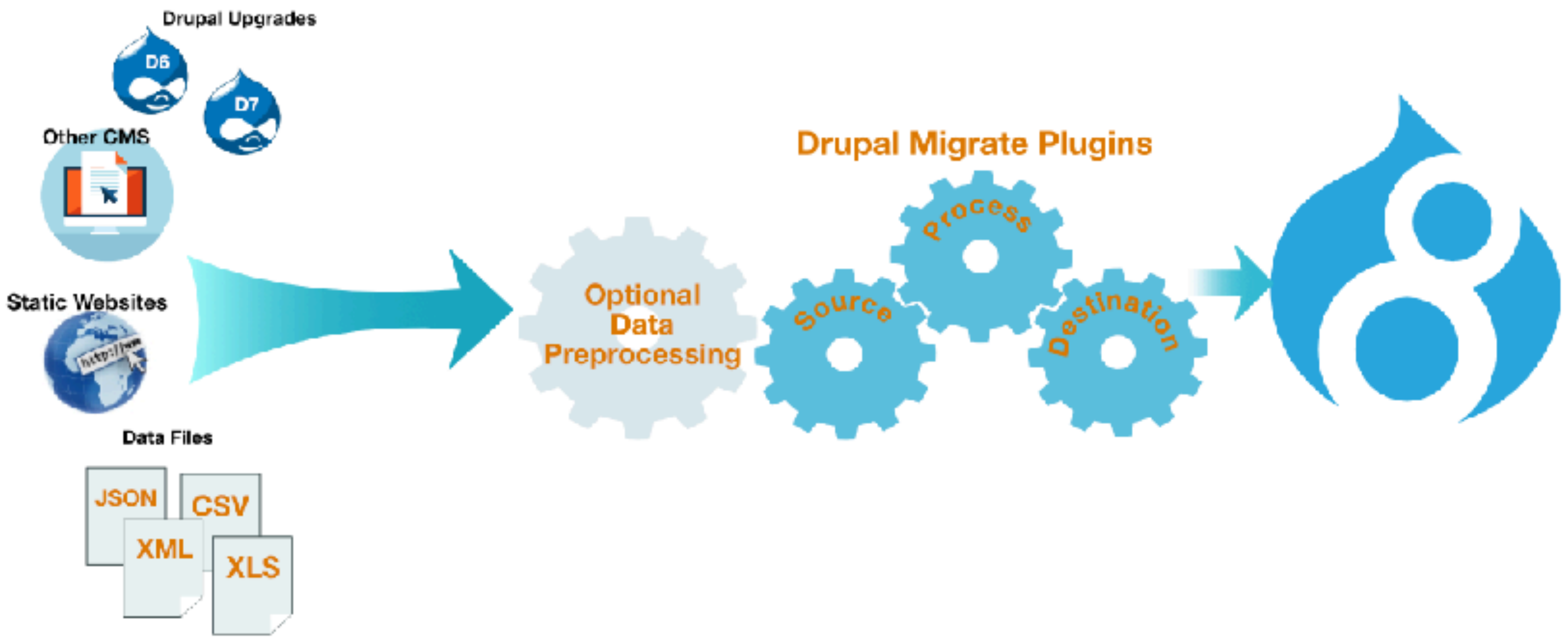
STEPHANIE BRIDGES

DRUPAL DEVELOPER
ACQUIA, INC.

steph.bridges@acquia.com

WHAT IS MIGRATION?

Migration is the process of importing content from an external source.



AUTOMATED MIGRATION METHODOLOGY

ANALYSIS

- Identify sources and targets for content
- Review and document the legacy data
- Define initial business and technical rules

AUTOMATED MIGRATION METHODOLOGY

MAPPING

- Define mapping rules from legacy data to target content types
- Identify exceptions, trouble spots, and data that can’t be automatically migrated

AUTOMATED MIGRATION METHODOLOGY

DEVELOPMENT

- Iterative process, closely coupled with content type development
- Build, import
- Identify exceptions
- Fix
- Repeat

AUTOMATED MIGRATION METHODOLOGY

LAUNCH

- Prior to launch, run full import and do QA
- Going forward, import only delta
- Set old site to ‘read only’
- QA and launch new site

AUTOMATED MIGRATION METHODOLOGY

IMPORTANT CONSIDERATIONS

- How many sites will be migrated?
- How similar are the sites?
- How much data will be migrated?
- Are there other sources of data (external integrations)?
- What are we moving the content from and what are we moving to?
- Can we get samples of the data?
- Can we get access to the data?
- How structured and consistent is the data being migrated?
- If we are migrating from HTML, how consistent is the tagging?

- What kind of data are we moving?
  - content
  - images/video
  - html
  - users
  - taxonomy
  - meta data (for example, OG metatags)
  - redirects
- Is the site multi-lingual?

MIGRATE IN DRUPAL 8

MIGRATION COMPONENTS IN DRUPAL 8

- Drupal 8 migration API provides services for migrating data from one place to another (generally, importing into Drupal entities)
- The core migrate module provides a general purpose framework which can be used to build migrations
- The core Migrate Drupal module provides an upgrade path from Drupal 6 or 7 to Drupal 8
- Migration components are implemented as plugins
- Contrib modules provide additional functionality, including command line tools, additional source/process/destination plugins, plugin types, and API extensions

ANATOMY OF A MIGRATION

OVERVIEW

- Migrations are defined as configuration entities
- There are three parts to a migration configuration
- Individual components are plugins
- The source plugin provides the data as rows
- Each row is handed to a set of process plugins which transform and map the data to destination properties
- The destination plugin saves the data to the entity

ANATOMY OF A MIGRATION

CONFIGURATION ENTITY

- The configuration is defined using YAML
- A configuration entity must contain four keys
  - id - string which identifies the migration
  - source - associative array which contains the plugin name and any configuration details
  - process - defines how the source data properties are to be mapped to the destination
  - destination - defines the destination entity

ANATOMY OF A MIGRATION

SOURCE PLUGIN

- plugin is the only required key
- track_changes can be used to allow importing changed rows in addition to new ones
- other keys will define settings such as filename, database credentials, etc

Basic Example

```
source:
  plugin: plugin_name
  track_changes: TRUE
```

CSV Source

```
source:
  plugin: csv
  header_row_count: 1
  path: path/to/data.csv
  keys:
    - id
  column_names:
    -
      id: ID
    -
      parent_id: ParentId
    -
      name: Name
    -
      description: Description
```

ANATOMY OF A MIGRATION

PROCESS PLUGINS

- Process plugins are used to map each field in the source to its corresponding destination entity property
- Plugins can also transform the data in addition to mapping
- Process plugins can be chained, with the data returned by a plugin passed to the next in the chain

```
process:
  type:
    plugin: default_value
    default_value: article
  uid:
    plugin: default_value
    default_value: 1
  title: title
  ‘body/value’: body
  ‘body/summary’: teaser
  ‘body/format’: rich_html
  field_tags:
    -
      plugin: skip_on_empty
      method: process
      source: tags
    -
      plugin: explode
      delimiter: ‘,’
    -
      plugin: migration_lookup
      migration: article_terms
```

ANATOMY OF A MIGRATION

PROCESS PLUGINS

- get
- default_value
- callback
- concat
- explode
- extract
- flatten
- format_date
- machine_name
- migration_lookup
- static_map
- skip_on_empty (row or process)
- skip_row_if_not_set
- iterator
- machine_name
- flatten
- entity_lookup*
- entity_generate*
- file_blob*
- merge*
- skip_on_value*

ANATOMY OF A MIGRATION

DESTINATION PLUGIN

- Destination has a mandatory plugin key
- Generally the value for this is `entity:entity_type`
- Additional keys can be used to specify the default bundle or whether this is a translation

```
destination:
 plugin: entity:node
```

MIGRATE API EVENTS

- Migrate implements events using an event subscriber
- You create a service which responds to the event(s)
    - MigrateEvents::PRE_IMPORT
    - MigrateEvents::POST_IMPORT
    - MigrateEvents::PRE_ROLLBACK
    - MigrateEvents::POST_ROLLBACK
    - MigrateEvents::MAP_SAVE
    - MigrateEvents::MAP_DELETE
    - MigrateEvents::PRE_ROW_SAVE
    - MigrateEvents::POST_ROW_SAVE
    - MigrateEvents::PRE_ROW_DELETE
    - MigrateEvents::POST_ROW_DELETE
    - MigrateEvents::PREPARE_ROW*

MIGRATE EXAMPLE

Migration Group

```
id: beer
label: Beer Imports
description: A few simple beer-related imports, to demonstrate how to implement migrations.
source_type: Custom tables
shared_configuration:
  source:
    key: default

dependencies:
  enforced:
    module:
      - migrate_example
```

migrate_plus.migration_group.beer.yml

```yaml
id: beer_term
label: Migrate style categories from the source database to taxonomy terms
migration_group: beer
source:
  plugin: beer_term

destination:
  plugin: entity:taxonomy_term

process:
  name: style
  description: details
  vid:
    plugin: default_value
    default_value: migrate_example_beer_styles
  tid:
    plugin: migration_lookup
    migration: beer_term
    source: style_parent

migration_dependencies: {}
dependencies:
  enforced:
    module:
      - migrate_example
```

migrate_plus.migration.beer_term.yml

```
id: beer_user
label: Beer Drinkers of the world
migration_group: beer
source:
  plugin: beer_user
destination:
  plugin: entity:user
process:
  pass: password
  mail: email
  init: email
  status: status
  roles:
    plugin: default_value
    default_value: 2
  name:
    plugin: dedupe_entity
    source: username
    entity_type: user
    field: name
    postfix: _
  created:
    plugin: callback
    source: registered
    callable: strtotime
  changed: '@created'
  access: '@created'
  login: '@created'
  field_migrate_example_gender:
    plugin: static_map
    source: sex
    map:
      0: Male
      1: Female
    bypass: true
  field_migrate_example_favbeers:
    plugin: migration_lookup
    source: beers
    migration: beer_node

migration_dependencies: {}
```

migrate_plus.migration.beer_user.yml

```yaml
id: beer_node
label: Beers of the world
migration_group: beer
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
```

migrate_plus.migration.beer_node.yml

CREATING MIGRATIONS IN DRUPAL 8

CREATING A MIGRATION

BEFORE YOU WRITE ANY CODE

- Make sure you have completed the analysis of the source content and you understand how to retrieve the data you will need
- Decide how you are going to access the source content — e.g. directly via SQL, exported CSV files, XML/JSON data, either via static files or an HTTP endpoint on the legacy site
- Have defined your content model for your new Drupal site and have set up your content entities (nodes, taxonomy, paragraph items, media bundles, etc.)

CREATING YOUR FIRST MIGRATION

THINGS TO CONSIDER

- What is the configuration management strategy for your site?
  - How will you update configuration when you make changes or add new YAML files?
- What is your data source?
  - Can you use an existing source plugin, or will you need to write your own (any SQL source will require a custom source plugin)
- Will you need to do any processing of your source data during mapping that cannot be accomplished using existing process plugins?
  - Recommend using process plugins when you need to transform data during mapping

MIGRATION CONSIDERATIONS

MIGRATING INTO PARAGRAPHS

- Create a migration configuration for your paragraphs items
- Destination plugin is 'entity_reference_revisions:paragraph'
- The parent node migration must have the paragraph migration as a migration dependency
- Paragraphs items are referenced by their entity_id **and** revision_id
- Process plugin in node migration will need to provide both values

Referencing a paragraph item in a node:

Assuming a node field named “field_paragraphs” and a source row property named paragraph_items which contains the source ID(s) from the “paragraph_migration” migration.

```
field_paragraphs:
  -
    plugin: explode
    delimiter: ','
    source: paragraph_items
  -
    plugin: migration_lookup
    migration: paragraph_migration
  -
    plugin: skip_on_empty
    method: process
  -
    plugin: iterator
    process:
      target_id: '0'
      target_revision_id: '1'
```

MIGRATION CONSIDERATIONS

MIGRATING MULTI-LINGUAL CONTENT

- Migrate source data for non-default language must contain reference to associated default language content
- Create migrations which migrate all content in the site's default language
- Create separate migrations which migrate all other languages
    - Each translation migration must depend on the associated default language migration
    - Migration must map the entity id of the migrated entity to the id of the default language entity
    - The destination plugin must include the "translations" property

NODE TRANSLATIONS

```
process:
  nid:
    plugin: migration_lookup
    migration: article
    source: parent_id
```

```
destination:
  plugin: ‘entity:node'
  translations: true
```

PARAGRAPH ITEMS TRANSLATIONS

```
process:
  parent:
    plugin: migration_lookup
    migration: paragraph_migration
    source: master_id
  id: '@parent/0'
  revision_id: '@parent/1'


destination:
  plugin: 'entity_reference_revisions:paragraph'
  translations: true
```

CUSTOM PROCESS PLUGIN

- Creating a custom process plugin is a straightforward process
- Create your plugin class in a custom module in **module_name/src/Plugin/migrate/process**
- Class should extend **ProcessPluginBase**
- Class must implement the **transform** method
- Annotate your plugin in the class DocBlock

```
/**
 * Creates a custom ECK component.
 *
 * @MigrateProcessPlugin(
 *   id = "eck_component"
 * )
 */
```

PROCESS PLUGIN EXAMPLE

```
<?php

namespace Drupal\migrate_demo_content\Plugin\migrate\process;

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

/**
 * Decode HTML entities for use in unformatted text fields.
 *
 * @MigrateProcessPlugin(
 *   id = "html_entity_decode"
 * )
 */
class HtmlEntityDecode extends ProcessPluginBase {

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

}
```

LINKS

Migate API:

https://www.drupal.org/docs/8/api/migrate-api

Migrating from CSV sources:

https://www.drupal.org/docs/8/modules/migrate-source-csv/using-the-migrate-source-csv-plugin

https://www.mtech-llc.com/blog/ada-hernandez/how-migrate-images-drupal-8-using-csv-source

https://evolvingweb.ca/blog/drupal-8-migration-migrating-basic-data-part-1

Migrating multi-lingual:

https://evolvingweb.ca/blog/migrate-translations-csv-json-or-xml-drupal-8

QUESTIONS?