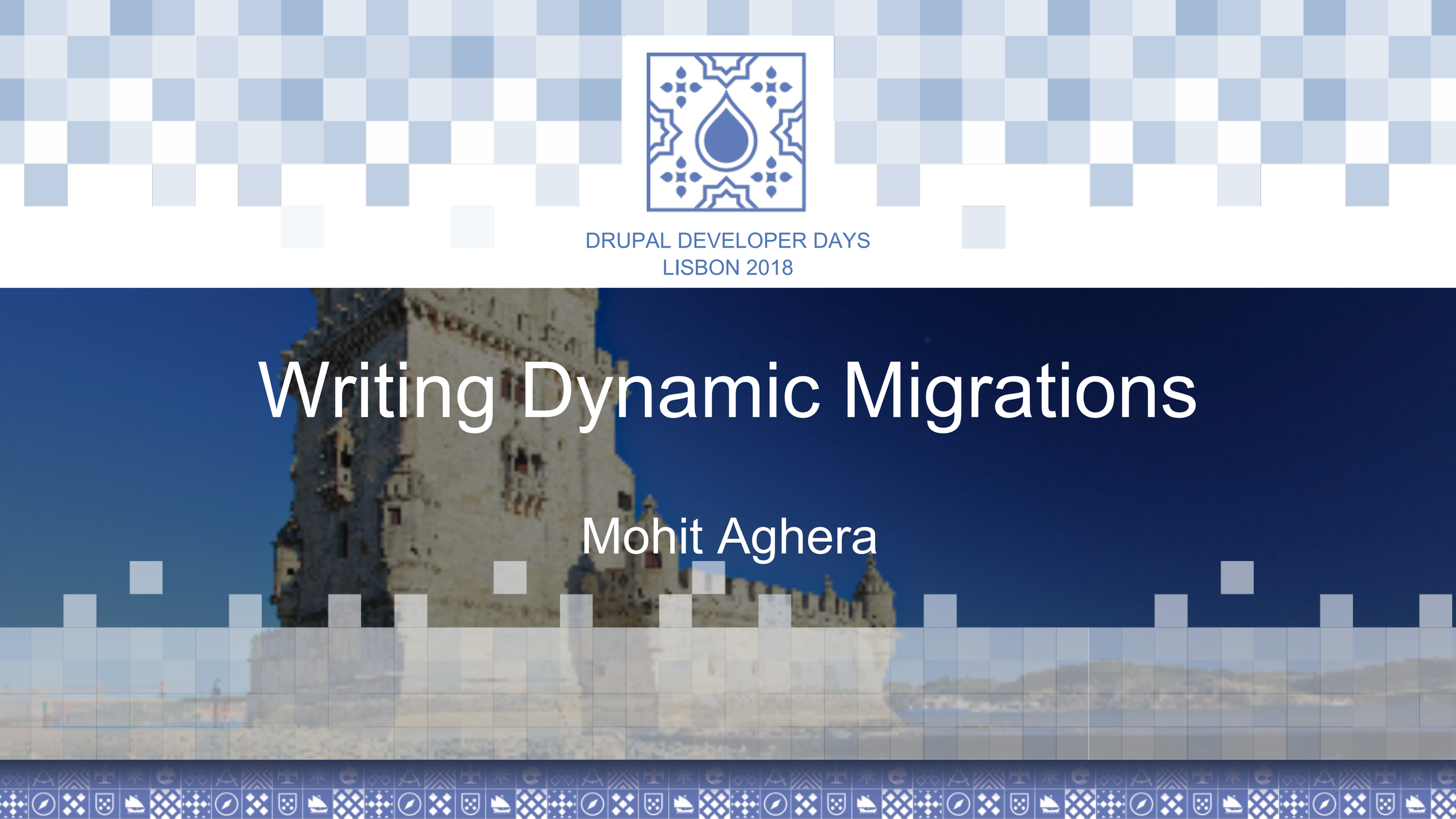


DRUPAL DEVELOPER DAYS
LISBON 2018

Writing Dynamic Migrations

Mohit Aghera

Diamond Sponsor

Platinum Sponsors

Gold Sponsors

Special Thanks to..

About Me

Mohit Aghera

- Back-end developer @Axelerant
- Drupal.org: mohit_aghera
- Twitter: @mohit_rocks

Dynamic Migrations

- What are dynamic migrations ?
- Why do we need it ?

Quick Recap !!!

Migrations : A Brief Introduction

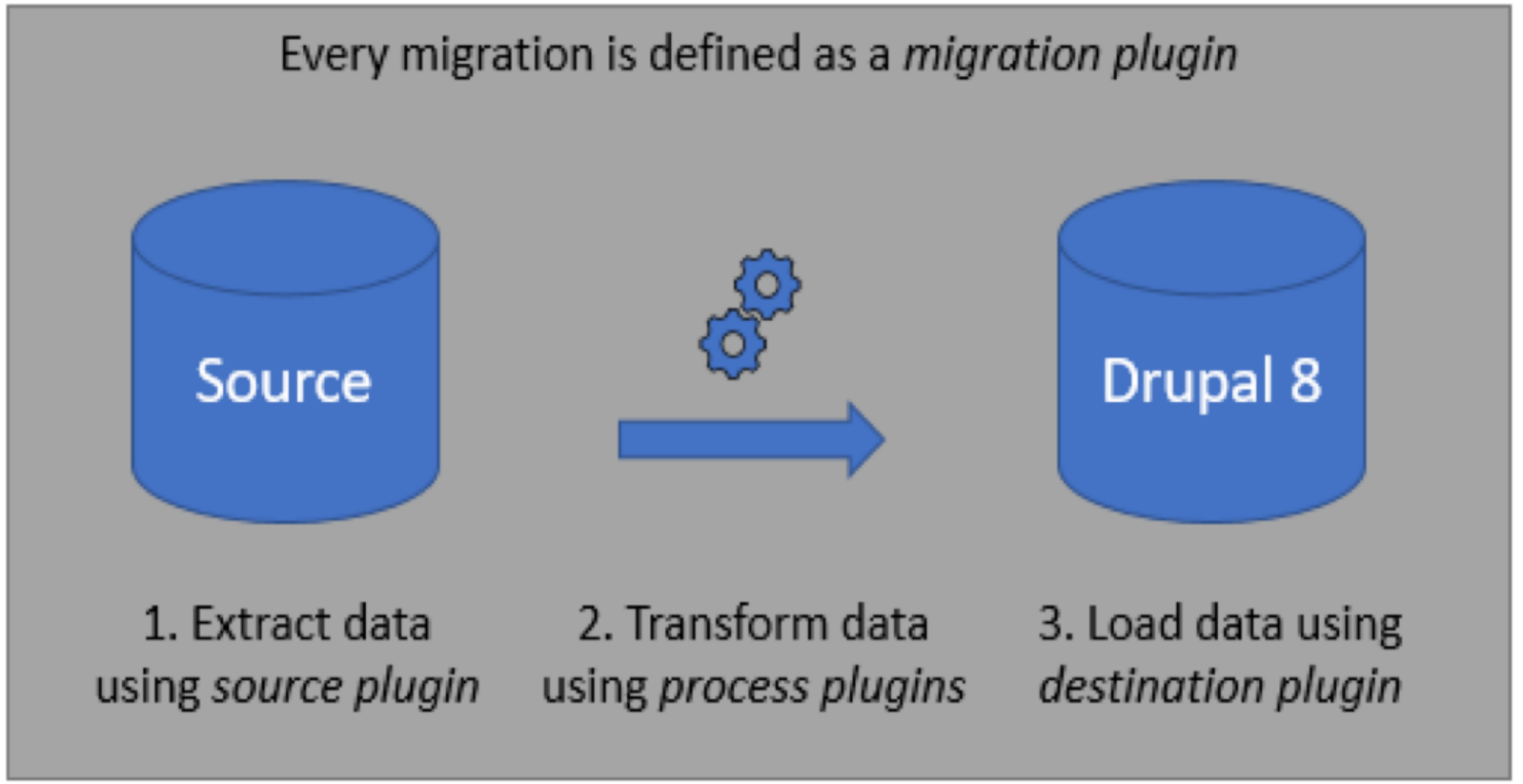
What Migrate API Provides:

- Migrate Data from source to destination
- Keeps track of record of migrated data
- Provides framework for writing migrations

Migration Plugins

- Source Plugins
- Process Plugin
- Destination Plugins
- And many more...

Migration Flow



Understanding Derivatives

- Drupal 7 has “_info()” hooks
- Wonder how do we generate multiple blocks with single block plugin ?
- Drupal 8 has “Derivatives”
- Allows to generate instances dynamically

How Derivative Works?

- Plugin manager uses ”Discovery” mechanism
- DerivativeDiscoveryDecorator” decorator class

Define Along with Plugins

```
/**
 * Provides a generic Menu block.
 *
 * @Block(
 *   id = "system_menu_block",
 *   admin_label = @Translation("Menu"),
 *   category = @Translation("Menus"),
 *   deriver = "Drupal\system\Plugin\Derivative\SystemMenuBlock",
 *   forms = {
 *     "settings_tray" = "\Drupal\system\Form\SystemMenuOffCanvasForm",
 *   },
 * )
 */
```

Deriver Implementation

SystemMenuBlock Deriver implementation:

```php
/**
 * {@inheritdoc}
 */
public function getDerivativeDefinitions($base_plugin_definition) {
  foreach ($this->menuStorage->loadMultiple() as $menu => $entity) {
    $this->derivatives[$menu] = $base_plugin_definition;
    $this->derivatives[$menu]['admin_label'] = $entity->label();
    $this->derivatives[$menu]['config_dependencies']['config'] =
[$entity->getConfigDependencyName()];
  }
  return $this->derivatives;
}
```

Derivatives Examples

Notable Examples:

- SystemMenuBlock
- BlockContentBlock
- LanguageBlock

And many more..

Possible approaches for Migration

- Configuration Entities
- Migration Templates

Migration as Configuration Entities

- Similar to any configurations
- Resides in [module]/config directory
- Typically names are given like “migrate.migration.node_page.yml”

Migration as Configuration Entities

Advantages:

- Easy to write and run
- No need to write specific logic to run unlike migration templates

Migration as Configuration Entities

Disadvantages:

- Not much flexible
- Dynamic migration generation is not possible
- Can’t generate based on user inputs

Migration as Templates

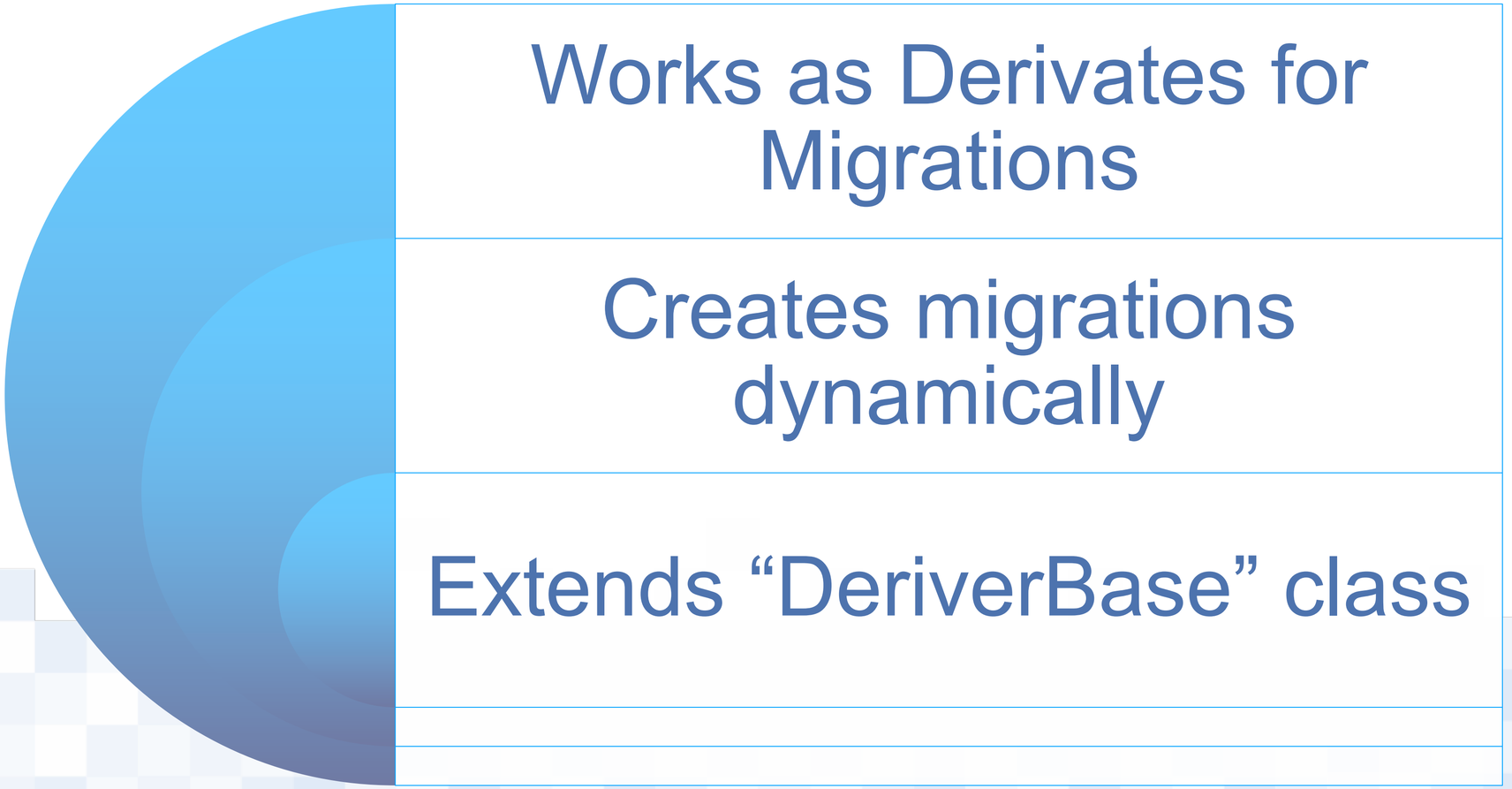
- Resides in [module]/migration_templates directory
- Names could match the migration id
- Typically names are given like “node_page.yml”

Migration as Templates

Advantages:

- Much more flexible than configuration entities for each variation
- Easy to change runtime
- Ability to generate dynamic migration using reusable templates

Derivers for Migrations



Writing Derivers

Implement *getDerivativeDefinitions()* and write your logic.

Using Derivers

Specify in migrations template file

```
id: node_product_translation
label: "Product Content"
migration_tags:
  - content_import
deriver: Drupal\content_import\Plugin\migrate\ProductNodeDeriver
```

Running Migrations

```
$migration = \Drupal::service( id: 'plugin.manager.migration')
->createInstance('node_product_translation:' . $language);

$executable = new MigrateExecutable($migration, new MigrateMessage());

$migration_status = $executable->import();
```

Use cases

- Migrating large amount of sites with similar architecture but different languages in each site (our use case, https://github.com/mohit-rocks/drupal-days)
- Core: Migrating D7 to D8
- Core: Migrating Workbench Moderation to Content Moderation (https://www.drupal.org/project/wbm2cm)
- And many more...

DRUPAL DEVELOPER DAYS
LISBON 2018

Thank you !!!