SciShield℠

Unleash innovation without compromise.

Getting Stuff Done:

Working with Queues in Drupal

DrupalCamp Asheville 2024

Erich Beyrent

Senior Software Engineer at SciShield

Drupal: https://www.drupal.org/u/ebeyrent

GitHub: https://github.com/ebeyrent

LinkedIn: https://www.linkedin.com/in/erichbeyrent

SciShield℠

Agenda

- What is the Queue API?

SciShield℠

Agenda

- What is the Queue API?
- When to use queues

SciShield℠

Agenda

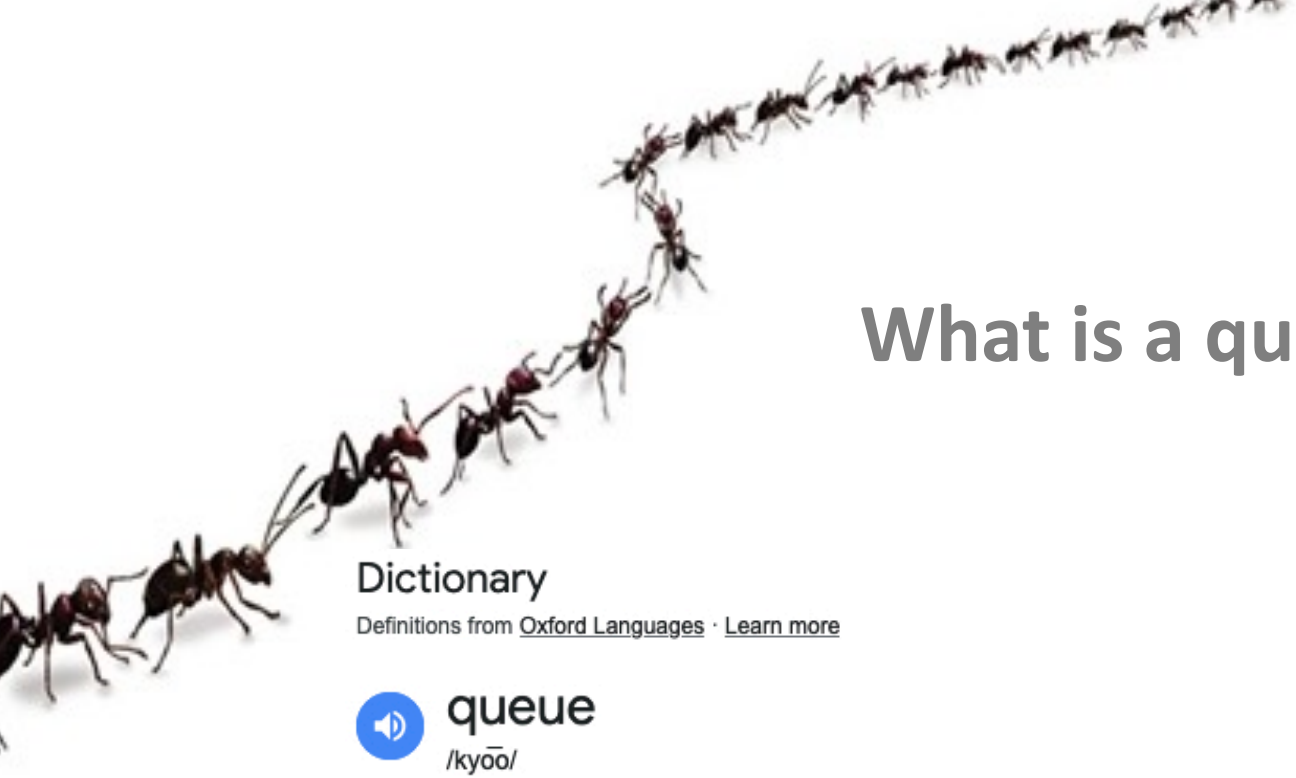
- What is the Queue API?
- When to use queues
- Types of database queues in Drupal

Agenda

- What is the Queue API?
- When to use queues
- Types of database queues in Drupal
- Queue workers

Agenda

- What is the Queue API?
- When to use queues
- Types of database queues in Drupal
- Queue workers
- Automated testing of queues and queue workers



What is a queue?

Dictionary

Definitions from Oxford Languages · Learn more

queue

/kyo͞o/

noun

noun: **queue**; plural noun: **queues**

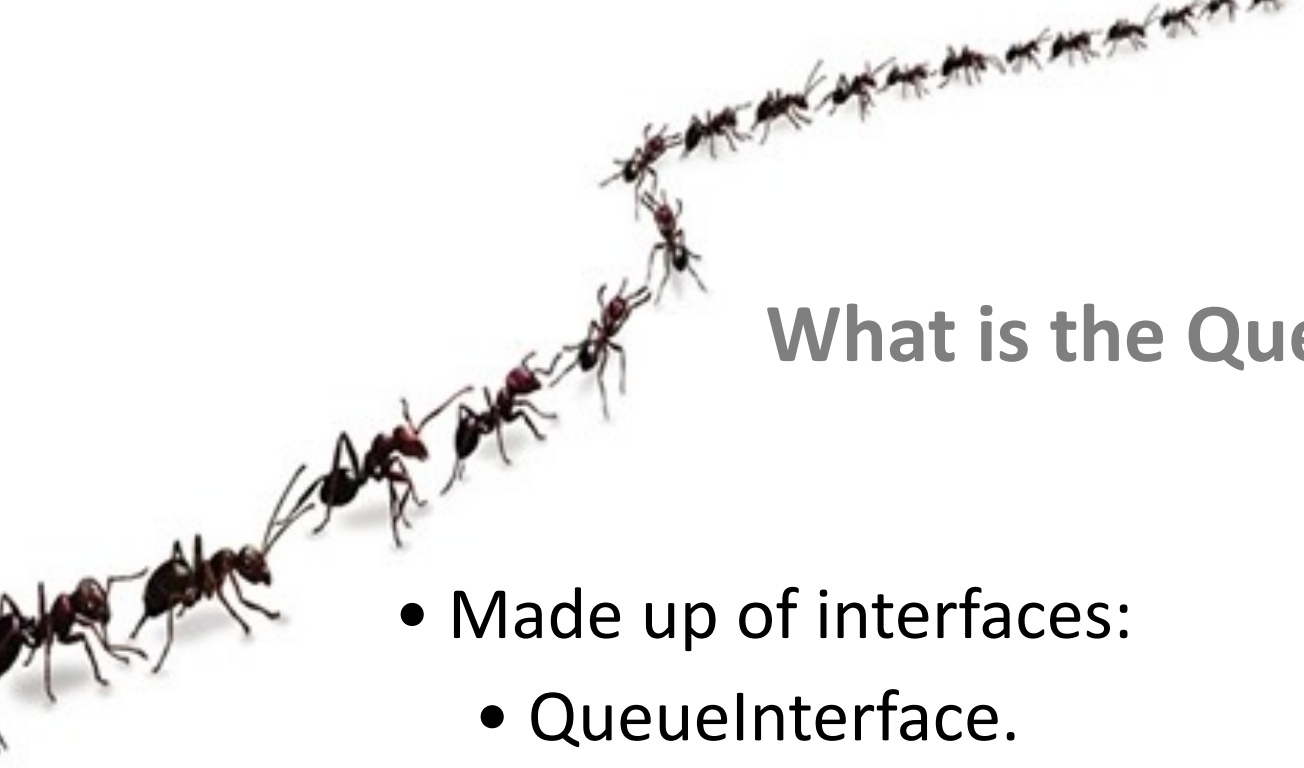
1. BRITISH
   a line or sequence of people or vehicles awaiting their turn to be attended to or to proceed.

   Similar: line | row | column | file | chain | string | stream | procession

2. COMPUTING
   a list of data items, commands, etc., stored so as to be retrievable in a definite order, usually the order of insertion.

3. ARCHAIC
   a braid of hair worn at the back.

SciShield℠

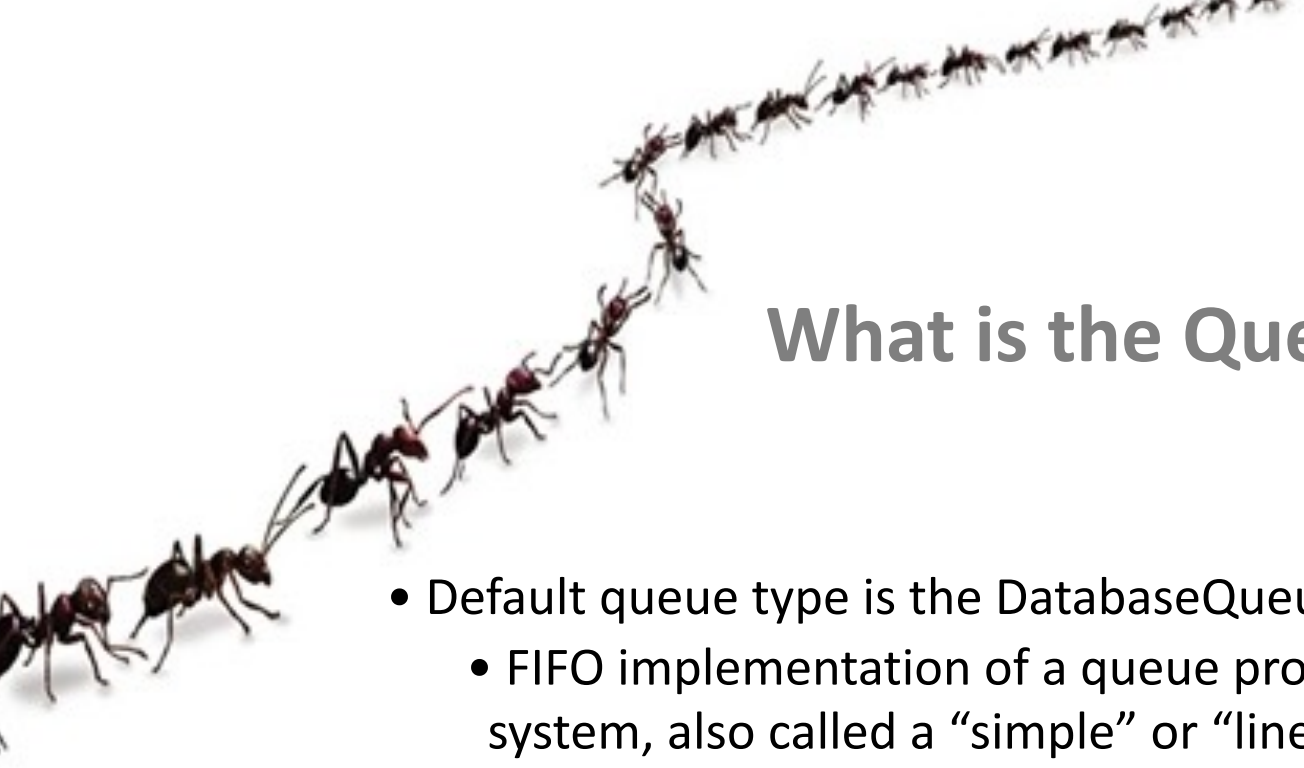


What is the Queue API?

- Made up of interfaces:
  - QueueInterface.
    - DelayableQueueInterface
    - ReliableQueueInterface
  - QueueWorkerInterface

SciShield℠

```php
<?php

namespace Drupal\Core\Queue;

/**
 * Interface for a queue.
 */
interface QueueInterface {

  public function createItem($data);

  public function numberOfItems();

  public function claimItem($lease_time = 3600);

  public function deleteItem($item);

  public function releaseItem($item);

  public function createQueue();

  public function deleteQueue();

}
```



What is the Queue API?

- Default queue type is the DatabaseQueue
  - FIFO implementation of a queue processing system, also called a “simple” or “linear” queue

- Options
  - Run via cron with time limits (i.e. process certain amount of items within a given period of time)
  - Queues can be run directly via drush

SciShield℠

Types of Drupal queues

- Reliable
  - Database: preserves the order of messages and guarantees each item will be processed at least once.
- Non-Reliable
  - Amazon SQS, In-memory: Data might be lost if a crash occurs, but can deal with significantly more writes and might be worth the risk depending on the volume of data to be processed.

Types of Drupal database queues

- Priority
  - This type of queue was implemented in the D7 Priority Queue module
  - A D9 port of this module can be found at https://github.com/JulienD/priority_queue
- Unique
  - https://www.drupal.org/project/queue_unique
  - Provides a queue that only accepts unique items
- Parallel
  - Special type of queue where multiple queue workers can be running concurrently *

When to use Queues

Common Issue:

- Bulk-updating lots of records in the database

Techniques:

- Write a custom cron job to update the data
- Write a script that can be run via drush
- Write a form, controller, and batch process to update the data

When to use Queues

Common Issue:

- Work needs to be done in response to a user interaction
- Examples:
  - Create new entities from submitted data
  - Update related entities from submitted data
  - Transform submitted data or add data fetched from external data store

- Not scalable
- Not fault-tolerant
- Solutions can take lots of code
- Forking can be dangerous in production

Image source: https://annemoss.com/2021/10/23/second-act-masterclass-6-key-takeaways-from-100-episodes/

SciShield℠

Queue Workers

- Queue Workers are plugins
- Implements the QueueWorkerInterface

```php
<?php

namespace Drupal\Core\Queue;

use Drupal\Component\Plugin\PluginInspectionInterface;

/**
 * Defines an interface for a QueueWorker plugin.
 */
interface QueueWorkerInterface extends PluginInspectionInterface {

  /**
   * Works on a single queue item.
   */
  public function processItem($data);

}
```

Queue Workers

- Sole job is to processes a queue item and handle failures
  - Throw **\Drupal\Core\Queue\RequeueException**
    - Processing is not yet finished. This will allow another process to claim the item immediately.
  - Throw **\Exception**
    - Throw when a problem is encountered. The API will automatically log the exception and requeue the item to be processed later
  - Throw **\Drupal\Core\Queue\SuspendQueueException**
    - Throw when a problem is encountered that will prevent all other queue workers from processing the queue. Behaves like throwing a standard exception, except queue workers won't process the queue for the rest of the cron run.

Queue Workers

- QueueWorker instances process, by default, one job per worker

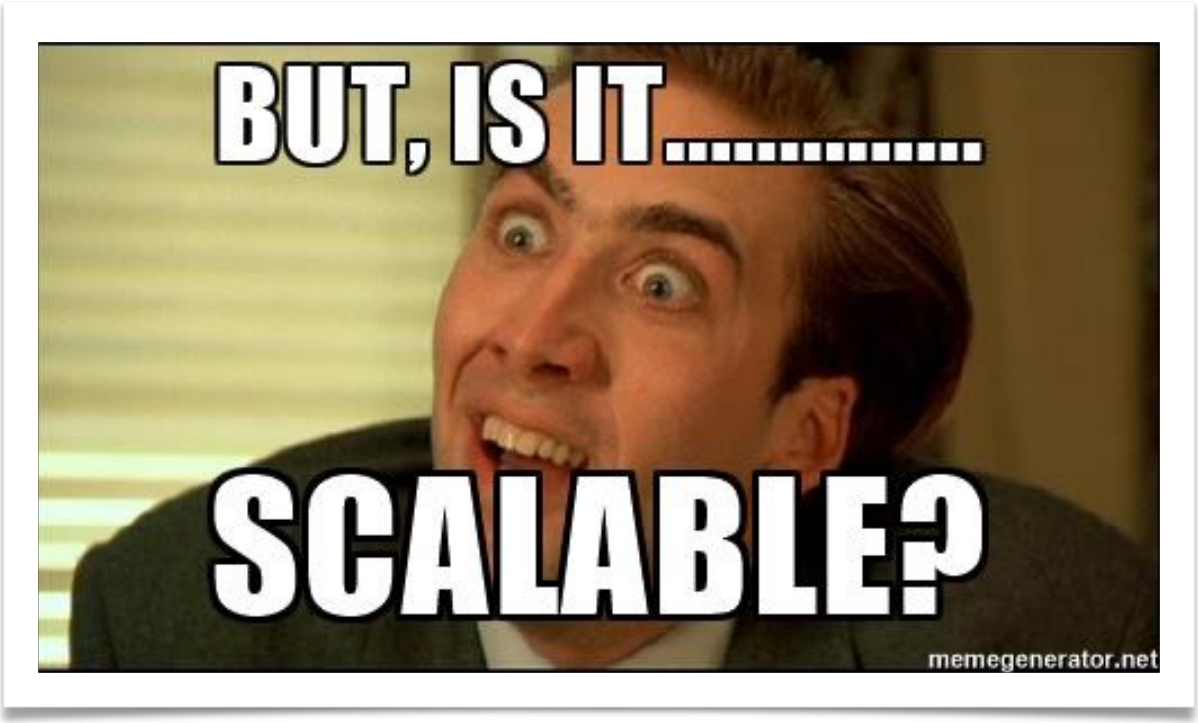


Image source: https://www.linkedin.com/pulse/buzz-off-9-buzzwords-tech-industry-needs-eliminate-from-deighan/

SciShield℠

Yes, they scale!

- Define a queue item with multiple operations or entities
- Use multiple linux cron tasks to process the same queue at same time or staggered
- Use systemd or supervisord to keep workers running
- “Dynamic” queues*

SciShield℠

Anatomy of a queue implementation

Create a custom module:

- Queue Worker
- Code to populate the queue

Image source: https://upload.wikimedia.org/wikipedia/commons/thumb/f/fb/Vesalius-copy.jpg/700px-Vesalius-copy.jpg

```php
<?php

namespace Drupal\my_module\Plugin\QueueWorker;

/**
 * Queue worker to process entities.
 *
 * @QueueWorker(
 *   id = “process_entity_queue_worker”,
 *   title = @Translation("Handles entity updates.”),
 *   cron = {"time" = 60},
 * )
 */
class MyQueueWorker extends QueueWorkerBase implements ContainerFactoryPluginInterface {

 /**
  * {@inheritdoc}
  */
 public function processItem($item) {
  // Queue item processing logic goes here.
 }
}
```

Anatomy of a queue implementation

Create a custom module:

- Queue Worker
- Code to populate the queue

Image source: https://upload.wikimedia.org/wikipedia/commons/thumb/f/fb/Vesalius-copy.jpg/700px-Vesalius-copy.jpg

```php
<?php

/**
 * Implements hook_entity_update().
 */
function my_module_entity_update(EntityInterface $entity) {
  if ($entity->getEntityTypeId() === ‘my_entity_type’) {
    $queue = \Drupal::service('queue')->get('process_entity_queue_worker');
    $item = new \stdClass();
    $item->nid = $entity->id();
   $queue->createItem($item);
  }
}
```

```php
<?php

/**
 * Implements hook_ENTITY_TYPE_update().
 */
function my_module_my_entity_type_update(EntityInterface $entity) {
  $queue = \Drupal::service('queue')->get('process_entity_queue_worker');
  $item = new \stdClass();
  $item->nid = $entity->id();
 $queue->createItem($item);
}
```

```php
<?php

/**
 * {@inheritdoc}
 */
public function postSave(EntityStorageInterface $storage, $update = TRUE) {
  $queue = \Drupal::service('queue')->get('process_entity_queue_worker');
  $item = new \stdClass();
  $item->nid = $this->id();
 $queue->createItem($item);
}
```

```php
<?php
/**
 * @var \Drupal\Core\Queue\QueueFactory
 */
protected QueueFactory $queueFactory;

public function queueEntities() {
  $query = $this->entityTypeManager->getStorage('my_entity_type')->getQuery();
  $query->accessCheck(FALSE);
  // Add conditions….
  $ids = $query->execute();

  $queue = $this->queueFactory->get('process_entity_queue_worker');
  foreach ($ids as $entity_id) {
    $item = new \stdClass();
    $item->nid = $entity_id;
   $queue->createItem($item);
  }
}
```

```php
<?php
/**
 * @var \Drupal\Core\Queue\QueueFactory
 */
protected QueueFactory $queueFactory;

public function queueEntities() {
  $query = $this->entityTypeManager->getStorage('my_entity_type')->getQuery();
  $query->accessCheck(FALSE);
  // Add conditions….
  $ids = $query->execute();
  $batches = array_chunk($ids, 50);
  $queue = $this->queueFactory->get('process_entity_queue_worker');
  foreach ($batches as $entity_ids) {
    $item = new \stdClass();
    $item->nids = $entity_ids;
    $queue->createItem($item);
  }
}
```

Image source: http://wonderoutside.org/wp-content/uploads/2019/03/DSC_0020.jpg

Other modules to investigate

Advanced Queue
https://www.drupal.org/project/advancedqueue

Queue UI
https://www.drupal.org/project/queue_ui

Queue Mail
https://www.drupal.org/project/queue_mail

Queue Order
https://www.drupal.org/project/queue_order

Queue Unique
https://www.drupal.org/project/queue_unique

More advanced queue implementations

Drupal Queues with RabbitMQ

https://github.com/robiningelbrecht/drupal-amqp-rabbitmq

SciShield℠

SciShield℠

Unleash innovation without compromise.

Questions?