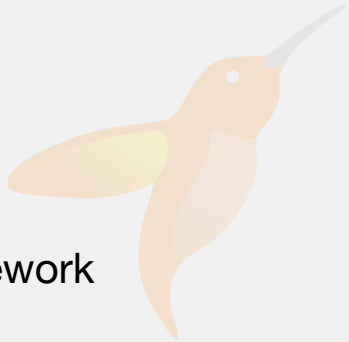
Documentation Language: Swift

Framework

JobsPostgres

Postgres implementation for Hummingbird jobs framework



Overview

JobsPostgres provides a Hummingbird Jobs Queue driver using PostgresNIO and the PostgresMigrations library.

Setup

The Postgres job queue driver uses `PostgresClient` from `PostgresNIO` and `DatabaseMigrations` from the PostgresMigrations library to perform the database migrations needed for the driver.

The Postgres job queue configuration includes two values.

- `pollTime`: This is the amount of time between the last time the queue was empty and the next time the driver starts looking for pending jobs.
- `queueName`: Name of queue used to differentiate itself from other queues.

```
import JobsPostgres
import PostgresNIO
import ServiceLifecycle

let postgresClient = PostgresClient(...)
let postgresMigrations = DatabaseMigrations()
let jobQueue = JobQueue(
    .postgres(
        client: postgresClient,
```

```
        migrations: postgresMigrations,
        configuration: .init(
            pollTime: .milliseconds(50),
            queueName: "MyJobQueue"
        ),
        logger: logger
    ),
    numWorkers: 4,
    logger: logger
)
```

The easiest way to ensure the migrations are run is to use the `DatabaseMigrationService` and add that as a `Service` to your `ServiceGroup`. The job queue service will not run until the migrations have been run in either `dryRun` mode or for real.

```
let migrationService = DatabaseMigrationService(
    client: postgresClient,
    migrations: postgresMigrations,
    logger: logger,
    dryRun: false
)
let serviceGroup = ServiceGroup(
    configuration: .init(
        services: [postgresClient, migrationService, jobQueue],
        gracefulShutdownSignals: [.sigterm, .sigint],
        logger: jobQueue.queue.logger
    )
)
try await serviceGroup.run()
```

Additional Features

There are features specific to the Postgres Job Queue implementation. Some of these are available in other queues and others not.

Push Options

When pushing a job to the queue there are a couple of options you can provide.

Delaying jobs

As with all queue drivers you can add a delay before a job is processed. The job will sit in the pending queue and will not be available for processing until time has passed its delay until time.

```
// Add TestJob to the queue, but don't process it for 2 minutes
try await jobQueue.push(TestJob(), options: .init(delayUntil: .now + 120))
```

Job Priority

The postgres queue allows you to give a job a priority. Jobs with higher priorities are run before jobs with lower priorities. There are five priorities `.lowest`, `.lower`, `.normal`, `.higher` and `.highest`.

```
// Add BackgroundJob to the queue. It will only get processed if there are n
// with a higher priority on the queue.
try await jobQueue.push(BackgroundJob(), options: .init(priority: .lowest))
```

Cancellation

The `PostgresJobQueue` conforms to protocol `CancellableJobQueue`. This requires support for cancelling jobs that are in the pending queue. It adds one new function `cancel(jobID:)`. If you supply this function with the `JobID` returned by `push(_:options:)` it will remove it from the pending queue.

```
// Add TestJob to the queue and immediately cancel it
let jobID = try await jobQueue.push(TestJob(), options: .init(delayUntil: .n
try await jobQueue.cancel(jobID: jobID)
```

Pause and Resume

The `PostgresJobQueue` conforms to protocol `ResumableJobQueue`. This requires support for pausing and resuming jobs that are in the pending queue. It adds two new functions `pause(jobID:)` and `resume(jobID:)`. If you supply these function with the `JobID` returned by `push(_:options:)` you can remove from the pending queue and add them back in at a later date.

```
// Add TestJob to the queue and immediately remove it and then add it back t
let jobID = try await jobQueue.push(TestJob(), options: .init(delayUntil: .n
try await jobQueue.pause(jobID: jobID)
try await jobQueue.resume(jobID: jobID)
```

Topics

Job Queue

`class PostgresJobQueue`

Postgres Job queue implementation

See Also

Related Documentation

- Jobs

  Offload work your server would be doing to another server.

- JobsRedis

  Redis implementation for Hummingbird jobs framework

- Hummingbird

  Lightweight, modern, flexible server framework written in Swift.

- HummingbirdPostgres

  Working with Postgres databases.

Reference

JobsRedis

Redis implementation for Hummingbird jobs framework