Curassow Documentation

Release 0.5.0

Kyle Fuller

Apr 07, 2017

Contents

Curassow is a Swift Nest HTTP Server. It uses the pre-fork worker model and it's similar to Python's Gunicorn and Ruby's Unicorn.

It exposes a Nest-compatible interface for your application, allowing you to use Curassow with any Nest compatible web frameworks of your choice.

CHAPTER 1

Quick Start

To use Curassow, you will need to install it via the Swift Package Manager, you can add it to the list of dependencies in your *Package.swift*:

```
import PackageDescription


let package = Package(
  name: "HelloWorld",
  dependencies: [
    .Package(url: "https://github.com/kylef/Curassow.git", majorVersion: 0, minor: 6),
  ]
)
```

Afterwards you can place your web application implementation in *Sources* and add the runner inside *main.swift* which exposes a command line tool to run your web application:

```
import Curassow
import Inquiline


serve { request in
  return Response(.ok, contentType: "text/plain", body: "Hello World")
}
```

Then build and run your application:

```
$ swift build --configuration release
$ ./.build/release/HelloWorld
```

Check out the Hello World example application.

CHAPTER 2

Contents

Configuration

Curassow provides you with an API, and a command line interface for configuration.

Server Socket

bind

The socket to bind to. This may either be in the form `HOST:PORT` or `unix:PATH` where any valid IP is acceptable for `HOST`.

```
$ curassow --bind 127.0.0.1:9000
[INFO] Listening at http://127.0.0.1:9000 (940)
```

```
$ curassow --bind unix:/tmp/helloworld.sock
[INFO] Listening at unix:/tmp/helloworld.sock (940)
```

Worker Processes

workers

The number of worker processes for handling requests.

```
$ curassow --workers 3
[INFO] Listening at http://0.0.0.0:8000 (940)
[INFO] Booting worker process with pid: 941
[INFO] Booting worker process with pid: 942
[INFO] Booting worker process with pid: 943
```

By default, the value of the environment variable `WEB_CONCURRENCY` will be used. If the environment variable is not set, `1` will be the default.

worker-type

The type of worker to use. This defaults to `sync`. Currently the only supported value is `sync`, there may be an async and gcd worker in the future.

```
$ curassow --worker-type sync
```

timeout

By default, Curassow will kill and restart workers after 30 seconds if it hasn't responded to the master process.

```
$ curassow --timeout 30
```

You can set the timeout to `0` to disable worker timeout handling.

Server

daemon

Daemonize the Curassow process. Detaches the server from the controlling terminal and enters the background.

```
$ curassow --daemon
```

Signal Handling

The master and worker processes will respond to the following signals as documented.

Master Process

- `QUIT` and `INT` - Quickly shutdown.
- `TERM` - Gracefully shutdown, this will wait for the workers to finish their current requests and gracefully timeout.
- `TTIN` - Increases the worker count by one.
- `TTOU` - Decreases the worker count by one.

Example

TTIN and TTOU signals can be sent to the master to increase or decrease the number of workers.

To increase the worker count by one, where $PID is the PID of the master process.

```
$ kill -TTIN $PID
```

To decrease the worker count by one:

```
$ kill -TTOU $PID
```

Worker Process

You may send signals directly to a worker process.

- `QUIT` and `INT` - Quickly shutdown.
- `TERM` - Gracefully shutdown.

Curassow Architecture

This document outlines the architecture and design of the Curassow HTTP server.

Curassow uses the pre-fork worker model, which means that HTTP requests are handled independently inside child worker processes.

Arbiter

The arbiter is the master process that manages the children worker processes. It has a simple loop that listens for signals sent to the master process and handles these signals.

Signals

The arbiter will watch for system signals and perform actions when it receives them.

SIGQUIT and SIGINT

The quit and interrupt signals can be used to quickly shutdown Curassow.

SIGTERM

The termination signal can be used to gracefully shutdown Curassow. The arbiter will wait for worker processes to finish handling their current requests or gracefully timeout.

TTIN

Increment the amount of worker processes by one.

TTOU

Decrement the amount of worker processes by one.

Worker

The worker process is responsible for handling HTTP requests.