Task Queues with IronWorkers

Posted At : November 3, 2014 12:55 PM | Posted By : Cutter
Related Categories: IronWorker, Development

As most of my readers know, I still love ColdFusion. In 15 years of development there are still very few things that I've found that I can't do quickly and easily with CF. That said, ColdFusion isn't always the right tool for the job. I don't mean that ColdFusion can't do the job, only that it's maybe not the best tool. The UI stuff is one great example. The ColdFusion UI stuff was written so that the most novice of developers can spin something up fast. But, anyone who's had to do something a bit more complex quickly finds the limitations behind it's generated UI code. Another is extremely resource intensive processes. Doing image manipulation on one image occasionally isn't such a big deal, but what happens when you have to process several thousand? As powerful as the language constructs are for manipulating these images, the constant conversion of binaries into Java Image Buffer objects, the increasing back and forth within RAM, it begins to bog your server down to the point of a dead crawl.

This image example is just one of many, and when you're writing enterprise level applications you're going to hit these hurdles, and using ColdFusion in these instances is like using a hammer when you need an Xacto blade. This is when you start looking for better options to these processes, that can interoperate with your current ColdFusion services. In looking for just such an option, I was pointed to **Iron.io**.

Iron.io is a cloud-based set of services that can be run on many of the major clouds. They began with a distributed Message Queueing service (**IronMQ**) built for handling critical messaging needs for distributed cloud applications. Building upon their queueing abilities, they also created **IronWorker**s. IronWorkers are async processing task queues. They allow you to define what your process environment needs to look like, your task script to process, and then you can queue up tasks which can asynchronously run in their own independent container environments. Once queued, IronWorker with run X number of tasks asynchronously (X being dependent on the plan level you choose). Each task runs within it's own sandbox, with it's own independent RAM and processor allocation, so that one running process does not affect another. As tasks complete their sandbox is torn down, the queue continues to spin up the remaining tasks on demand, until the task queue is done.

The ease with which they've developed this service is amazing. Your ".worker" file defines your environment. Each instance is a headles Ubuntu system. You can select from a number of "runtime" setups, allowing you to work in the language that you're most comfortable with (Node, Ruby, PHP, Java, etc), as well as picking a specific "stack" if you want to mix and match the setup a bit. Each instance also already

comes preloaded with several common Linux Packages (ImageMagick, cURL, SoX, etc). Within your ".worker" file, any additional files, folders, etc that you require can also be defined, including .deb packages.

Once your worker is defined, assets gathered, scripts written, etc., you then create your worker from a simple command line call. Once the "build" is complete, your new task service is ready to be called. This can be done via command line or (perhaps more commonly) via an http call. You can even define a webhook for your worker that can kick off your tasks from Git or elsewhere. You can pass variables to your task as part of it's "payload". This is just a simple HTTP Post, passing in name/value pairs that can be used within your process script. The "payload" is limited to 64k in size, so any files you may need on the fly (such as images to process) should be retrieved from within your process, most likely from somewhere like S3. Your process does it's thing, your script sends a command to exit the process (process.exit()) and it's done, spinning up the next task in the queue.

There really is a lot more to it. You can get as simple or as complex as you are capable of writing. IronWorkers are extremely powerful, and scale beautifully. After several weeks working on multiple processes, I can also say that their support is exceptional. They have HipChat channels setup to assist people, and they've been extremely responsive and helpful. They also maintain sample repositories of many common tasks, in a variety of languages, to help you get started while working in whichever environment you're most comfortable with.