lambdaric(3) 0.1 | Tcl Runtime Support for AWS Lambda

Cyan Ogilvie

LAMBDARIC

Tcl runtime for AWS Lambda

SYNOPSIS

package require lambdaric ?0.1?

lambdaric *setupScript eventLambda*
lambdaric::apig_event *event*
lambdaric::ns_compat::ns_conn *op ?arg ...?*
lambdaric::ns_compat::ns_set *op set ?arg ...?*
lambdaric::ns_compat::ns_respond ?**-status** *httpCode*? ?**-type** *contentType*? ?**-string|-binary** *body*?

DESCRIPTION

This package provides the framework for implementing AWS Lambda functions in Tcl. It also provides optional helpers for handling events from API Gateway with event version 1 or 2 (also covers Application Load Balancers and Lambda Function URLs), exposing the HTTP request context in shims compatible with NaviServer.

Since time is money on AWS Lambda, this package goes to some lengths to be as efficient as possible in execution time and memory usage. It should be very feasible to do real work in a 128 MiB function with low single digit millisecond billed time.

COMMANDS

lambdaric *setupScript eventLambda* Connect to the AWS Lambda Runtime API and start handling events. The script *setupScript* is run first (with errors reported to the runtime API), and then if that completed without errors, it enters a loop handling invocation events. Each event invokes the lambda *eventLambda*, which must accept two arguments: **event** and

context, which are the JSON documents from the runtime API describing the event.

lambdaric::apig_event ***event*** Parses the JSON *event* as an API Gateway event of either version 1 or 2 and sets up the HTTP request context accordingly. The HTTP request context can be accessed through the ns_compat shims.

lambdaric::ns_compat::* Compatibility shims that implement a subset of the NaviServer Tcl API for interacting with the HTTP request state. See the NaviServer documentation for the commands of the same name for details.

EXAMPLES

Implement a hello world HTTP event handler, which would work with API Gateway, an Application Load Balancer or CloudFront (via a Lambda Function URL origin):

```
package require lambdaric
namespace import lambdaric::ns_compat::*

lambdaric {
    # Nothing to set up
} {{event context} {
    lambdaric::apig_event [encoding convertfrom utf-8 $event]
    ns_respond -type text/plain -string "hello from lambda"
}}
```

BUILDING

This package has no external dependencies other than Tcl.

Currently Tcl 8.7 is required, but if needed polyfills could be built to support 8.6.

From a Release Tarball

Download and extract the release, then build in the standard TEA way:

```
wget https://github.com/cyanogilvie/tcl-lambdaric/releases/download/v0.1/lambdaric0.1.tar.gz
tar xf lambdaric0.1.tar.gz
cd lambdaric0.1
./configure
make
sudo make install
```

From the Git Sources

Fetch the code and submodules recursively, then build in the standard autoconf / TEA way:

```
git clone --recurse-submodules https://github.com/cyanogilvie/tcl-lambdaric
cd tcl-lambdaric
autoconf
./configure
make
sudo make install
```

In a Docker Build

Build from a specified release version, avoiding layer pollution and only adding the installed package without documentation to the image, and strip debug symbols, minimising image size:

```
WORKDIR /tmp/tcl-lambdaric
RUN wget https://github.com/cyanogilvie/tcl-lambdaric/releases/download/v0.1/lambdaric0.1.ta
    ./configure; make test install-binaries install-libraries && \
    strip /usr/local/lib/liblambdaric*.so && \
    cd .. && rm -rf tcl-lambdaric
```

For any of the build methods you may need to pass `--with-tcl /path/to/tcl/lib` to `configure` if your Tcl install is somewhere nonstandard.

AVAILABLE IN

The most recent release of this package is available by default in the `alpine-tcl` container image: docker.io/cyanogilvie/alpine-tcl and the `cftcl` Tcl runtime snap: https://github.com/cyanogilvie/cftcl.

SOURCE CODE

This package's source code is available at https://github.com/cyanogilvie/tcl-lambdaric. Please create issues there for any bugs discovered.

LICENSE