Package ‘plumber’

June 5, 2018

Encoding UTF-8

Type Package

Title An API Generator for R

Version 0.4.6

License MIT + file LICENSE

BugReports `https://github.com/trestletech/plumber/issues`

URL `https://www.rplumber.io` (site)

`https://github.com/trestletech/plumber` (dev)

Description Gives the ability to automatically generate and serve an HTTP API from R functions using the annotations in the R documentation around your functions.

Depends R (>= 3.0.0)

Imports R6 (>= 2.0.0), stringi (>= 0.3.0), jsonlite (>= 0.9.16), httpuv (>= 1.2.3), crayon

LazyData TRUE

ByteCompile TRUE

Suggests testthat (>= 0.11.0), XML, rmarkdown, PKI, base64enc, htmlwidgets, visNetwork, analogsea

Collate 'content-types.R' 'cookie-parser.R' 'parse-globals.R' 'images.R' 'parse-block.R' 'globals.R' 'serializer-json.R' 'shared-secret-filter.R' 'post-body.R' 'query-string.R' 'plumber.R' 'default-handlers.R' 'digital-ocean.R' 'find-port.R' 'includes.R' 'new-rstudio-project.R' 'paths.R' 'plumber-static.R' 'plumber-step.R' 'response.R' 'serializer-content-type.R' 'serializer-html.R' 'serializer-htmlwidget.R' 'serializer-xml.R' 'serializer.R' 'session-cookie.R' 'swagger.R'

RoxygenNote 6.0.1

NeedsCompilation no

Author Trestle Technology, LLC [aut],
Jeff Allen [cre],
Frans van Dunné [ctb],
Sebastiaan Vandewoude [ctb],
SmartBear Software [ctb, cph] (swagger-ui)

Maintainer Jeff Allen `<cran@trestletech.com>`

Repository CRAN

Date/Publication 2018-06-05 04:43:03 UTC

R topics documented:

`addSerializer` *Add a Serializer*

Description

A serializer is responsible for translating a generated R value into output that a remote user can understand. For instance, the `serializer_json` serializes R objects into JSON before returning them to the user. The list of available serializers in plumber is global.

Usage

```
addSerializer(name, serializer)
```

Arguments

`name`	The name of the serializer (character string)
`serializer`	The serializer to be added.

`do_configure_https` *Add HTTPS to a plumber Droplet*

Description

Adds TLS/SSL (HTTPS) to a droplet created using `do_provision()`.

Usage

```
do_configure_https(droplet, domain, email, termsOfService = FALSE,
  force = FALSE)
```

Arguments

`droplet`	The droplet on which to act. See `analogsea::droplet()`.
`domain`	The domain name associated with this instance. Used to obtain a TLS/SSL certificate.
`email`	Your email address; given only to letsencrypt when requesting a certificate to enable them to contact you about issues with renewal or security.
`termsOfService`	Set to `TRUE` to agree to the letsencrypt subscriber agreement. At the time of writing, the current version is available here. Must be set to true to obtain a certificate through letsencrypt.
`force`	If `FALSE`, will abort if it believes that the given domain name is not yet pointing at the appropriate IP address for this droplet. If `TRUE`, will ignore this check and attempt to proceed regardless.

Details

In order to get a TLS/SSL certificate, you need to point a domain name to the IP address associated with your droplet. If you don't already have a domain name, you can register one here. Point a (sub)domain to the IP address associated with your plumber droplet before calling this function. These changes may take a few minutes or hours to propagate around the Internet, but once complete you can then execute this function with the given domain to be granted a TLS/SSL certificate for that domain.

Obtains a free TLS/SSL certificate from letsencrypt and installs it in nginx. It also configures nginx to route all unencrypted HTTP traffic (port 80) to HTTPS. Your TLS certificate will be automatically renewed and deployed. It also opens port 443 in the firewall to allow incoming HTTPS traffic.

Historically, HTTPS certificates required payment in advance. If you appreciate this service, consider donating to the letsencryptproject.

`do_deploy_api` *Deploy or Update an API*

Description

Deploys an API from your local machine to make it available on the remote plumber server.

Usage

```
do_deploy_api(droplet, path, localPath, port, forward = FALSE,
  swagger = FALSE, preflight)
```

Arguments

`droplet`	The droplet on which to act. It's expected that this droplet was provisioned using `do_provision()`. See `analogsea::droplet()` to obtain a reference to a running droplet.
`path`	The remote path/name of the application
`localPath`	The local path to the API that you want to deploy. The entire directory referenced will be deployed, and the `plumber.R` file inside of that directory will be used as the root plumber file. The directory MUST contain a `plumber.R` file.
`port`	The internal port on which this service should run. This will not be user visible, but must be unique and point to a port that is available on your server. If unsure, try a number around `8000`.
`forward`	If `TRUE`, will setup requests targeting the root URL on the server to point to this application. See the `do_forward()` function for more details.
`swagger`	If `TRUE`, will enable the Swagger interface for the remotely deployed API. By default, the interface is disabled.
`preflight`	R commands to run after `plumb()`ing the `plumber.R` file, but before `run()`ing the plumber service. This is an opportunity to e.g. add new filters. If you need to specify multiple commands, they should be semi-colon-delimited.

`do_forward` *Forward Root Requests to an API*

Description

Forward Root Requests to an API

Usage

```
do_forward(droplet, path)
```

Arguments

`droplet`	The droplet on which to act. It's expected that this droplet was provisioned using `do_provision()`.
`path`	The path to which root requests should be forwarded

`do_provision` *Provision a DigitalOcean plumber server*

Description

Create (if required), install the necessary prerequisites, and deploy a sample plumber application on a DigitalOcean virtual machine. You may sign up for a Digital Ocean account here. This command is idempotent, so feel free to run it on a single server multiple times.

Usage

```
do_provision(droplet, unstable = FALSE, example = TRUE, ...)
```

Arguments

`droplet`	The DigitalOcean droplet that you want to provision (see `analogsea::droplet()`). If empty, a new DigitalOcean server will be created.
`unstable`	If `FALSE`, will install plumber from CRAN. If `TRUE`, will install the unstable version of plumber from GitHub.
`example`	If `TRUE`, will deploy an example API named `hello` to the server on port 8000.
`...`	Arguments passed into the `analogsea::droplet_create()` function.

Details

Provisions a Ubuntu 16.04-x64 droplet with the following customizations:

- A recent version of R installed
- plumber installed globally in the system library
- An example plumber API deployed at `/var/plumber`
- A systemd definition for the above plumber API which will ensure that the plumber API is started on machine boot and respawned if the R process ever crashes. On the server you can use commands like `systemctl restart plumber` to manage your API, or `journalctl -u plumber` to see the logs associated with your plumber process.
- The 'nginx“ web server installed to route web traffic from port 80 (HTTP) to your plumber process.
- `ufw` installed as a firewall to restrict access on the server. By default it only allows incoming traffic on port 22 (SSH) and port 80 (HTTP).
- A 4GB swap file is created to ensure that machines with little RAM (the default) are able to get through the necessary R package compilations.

do_remove_api *Remove an API from the server*

Description

Removes all services and routing rules associated with a particular service. Optionally purges the associated API directory from disk.

Usage

```
do_remove_api(droplet, path, delete = FALSE)
```

Arguments

droplet	The droplet on which to act. It's expected that this droplet was provisioned using `do_provision()`. See `analogsea::droplet()` to obtain a reference to a running droplet.
path	The path/name of the plumber service
delete	If `TRUE`, will also delete the associated directory (`/var/plumber/whatever`) from the server.

do_remove_forward *Remove the forwarding rule*

Description

Removes the forwarding rule from the root path on the server. The server will no longer forward requests for `/` to an application.

Usage

```
do_remove_forward(droplet)
```

Arguments

droplet	The droplet on which to act. It's expected that this droplet was provisioned using `do_provision()`. See `analogsea::droplet()` to obtain a reference to a running droplet.

`forward` *Forward Request to The Next Handler*

Description

This function is used when a filter is done processing a request and wishes to pass control off to the next handler in the chain. If this is not called by a filter, the assumption is that the filter fully handled the request itself and no other filters or endpoints should be evaluated for this request.

Usage

```
forward()
```

`include_file` *Send File Contents as Response*

Description

Returns the file at the given path as the response.

Usage

```
include_file(file, res, content_type)

include_html(file, res)

include_md(file, res, format = NULL)

include_rmd(file, res, format = NULL)
```

Arguments

`file`	The path to the file to return
`res`	The response object into which we'll write
`content_type`	If provided, the given value will be sent as the `Content-type` header in the response.
`format`	Passed as the `output_format` to `rmarkdown::render`

Details

`include_html` will merely return the file with the proper `content_type` for HTML. `include_md` and `include_rmd` will process the given markdown file through `rmarkdown::render` and return the resultant HTML as a response.

plumb *Plumber Router*

Description

Routers are the core request handler in plumber. A router is responsible for taking an incoming request, submitting it through the appropriate filters and eventually to a corresponding endpoint, if one is found.

Usage

```
plumb(file, dir = ".")

plumber
```

Arguments

`file`	The file to parse as the plumber router definition
`dir`	The directory containing the `plumber.R` file to parse as the plumber router definition. Alternatively, if an `entrypoint.R` file is found, it will take precedence and be responsible for returning a runnable Plumber router.

Format

An object of class `R6ClassGenerator` of length 24.

Details

See `http://www.rplumber.io/docs/programmatic/` for additional details on the methods available on this object.

PlumberEndpoint *Plumber Endpoint*

Description

Defines a terminal handler in a PLumber router.

Usage

```
PlumberEndpoint
```

Format

An object of class `R6ClassGenerator` of length 24.

`PlumberStatic` *Static file router*

Description

Creates a router that is backed by a directory of files on disk.

Usage

```
PlumberStatic
```

Format

An object of class `R6ClassGenerator` of length 24.

`serializer_json` *Plumber Serializers*

Description

Serializers are used in Plumber to transform the R object produced by a filter/endpoint into an HTTP response that can be returned to the client. See here for more details on Plumber serializers and how to customize their behavior.

Usage

```
serializer_json()

serializer_unboxed_json()

serializer_content_type(type)

serializer_html()

serializer_htmlwidget()
```

Arguments

`type`	The value to provide for the `Content-Type` HTTP header.

`sessionCookie` *Store session data in encrypted cookies.*

Description

Store session data in encrypted cookies.

Usage

```
sessionCookie(key, name = "plumber", ...)
```

Arguments

`key`	The secret key to use. This must be consistent across all sessions where you want to save/restore encrypted cookies. It should be a long and complex character string to bolster security.
`name`	The name of the cookie in the user's browser.
`...`	Arguments passed on to the `response$setCookie` call to, for instance, set the cookie's expiration.

Index