jwilder / docker-gen

Generate files from docker container meta-data

#docker #go

339 commits | 3 branches | 26 releases | 43 contributors | MIT

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

jwilder Merge pull request #292 from HaraldNordgren/go_versions … Latest commit 4edc190 28 days ago

cmd/docker-gen	Fix help and README.md for notify-sighup	2 years ago
examples	Documentation: fix signal number/name mismatch	3 years ago
templates	Deal with networks (supports new overlay network)	3 years ago
.dockerignore	Optimize Docker image	4 years ago
.gitignore	Remove accidentally committed release files	5 years ago
.travis.yml	Bump Go versions	a month ago
Dockerfile	MAINTAINER is deprecated, using LABEL now	a year ago
GLOCKFILE	Update go-dockerclient	a year ago
LICENSE	Add license	5 years ago
Makefile	Add builds for arm64	10 months ago
README.md	feat: Add "whereNot" function	a year ago
config.go	Merge pull request #130 from AXA-GROUP-SOLUTIONS/include_stopped	3 years ago
context.go	Addressed formatting issue.	10 months ago
context_test.go	Fix ECS container ID parsing	10 months ago
docker_client.go	Add server info	3 years ago
docker_client_test.go	Restructure files	3 years ago
example.conf	Add service registration w/ etcd example	5 years ago
generator.go	Fix error message capitalization	3 years ago
generator_test.go	Add -wait parameter to debounce watched events	3 years ago
reflect.go	Fix error message capitalization	3 years ago
reflect_test.go	Restructure files	3 years ago
template.go	feat: Add "whereNot" function	a year ago
template_test.go	Fix linter failures	a month ago
utils.go	Restructure files	3 years ago
utils_test.go	Fix linter failures	a month ago

README.md

docker-gen

latest 0.7.3 | build passing | license MIT

`docker-gen` is a file generator that renders templates using docker container meta-data.

It can be used to generate various kinds of files for:

- **Centralized logging** - fluentd, logstash or other centralized logging tools that tail the containers JSON log file or files within the container.
- **Log Rotation** - logrotate files to rotate container JSON log files
- **Reverse Proxy Configs** - nginx, haproxy, etc. reverse proxy configs to route requests from the host to containers
- **Service Discovery** - Scripts (python, bash, etc..) to register containers within etcd, hipache, etc..

===

Installation

There are three common ways to run docker-gen:

- on the host
- bundled in a container with another application
- separate standalone containers

Host Install

Linux/OSX binaries for release 0.7.3

- amd64
- i386
- alpine-linux

Download the version you need, untar, and install to your PATH.

```
$ wget https://github.com/jwilder/docker-gen/releases/download/0.7.3/docker-gen-linux-amd64-0.7.3.tar.gz
$ tar xvzf docker-gen-linux-amd64-0.7.3.tar.gz
$ ./docker-gen
```

Bundled Container Install

Docker-gen can be bundled inside of a container along-side applications.

jwilder/nginx-proxy trusted build is an example of running docker-gen within a container along-side nginx. jwilder/docker-register is an example of running docker-gen within a container to do service registration with etcd.

Separate Container Install

It can also be run as two separate containers using the jwilder/docker-gen image, together with virtually any other image.

This is how you could run the official nginx image and have docker-gen generate a reverse proxy config in the same way that `nginx-proxy` works. You may want to do this to prevent having the docker socket bound to a publicly exposed container service.

Start nginx with a shared volume:

```
$ docker run -d -p 80:80 --name nginx -v /tmp/nginx:/etc/nginx/conf.d -t nginx
```

Fetch the template and start the docker-gen container with the shared volume:

```
$ mkdir -p /tmp/templates && cd /tmp/templates
$ curl -o nginx.tmpl https://raw.githubusercontent.com/jwilder/docker-gen/master/templates/nginx.tmpl
$ docker run -d --name nginx-gen --volumes-from nginx \
   -v /var/run/docker.sock:/tmp/docker.sock:ro \
   -v /tmp/templates:/etc/docker-gen/templates \
   -t jwilder/docker-gen -notify-sighup nginx -watch -only-exposed /etc/docker-gen/templates/nginx.tmpl
/etc/nginx/conf.d/default.conf
```

===

Usage

```
$ docker-gen
Usage: docker-gen [options] template [dest]

Generate files from docker container meta-data

Options:
  -config value
      config files with template directives. Config files will be merged if this option is specified multiple
times. (default [])
  -endpoint string
```

```
        docker api endpoint (tcp|unix://..). Default unix:///var/run/docker.sock
    -interval int
        notify command interval (secs)
    -keep-blank-lines
        keep blank lines in the output file
    -notify restart xyz
        run command after template is regenerated (e.g restart xyz)
    -notify-output
        log the output(stdout/stderr) of notify command
    -notify-sighup container-ID
        send HUP signal to container.  Equivalent to 'docker kill -s HUP container-ID'
    -only-exposed
        only include containers with exposed ports
    -only-published
        only include containers with published ports (implies -only-exposed)
    -include-stopped
        include stopped containers
    -tlscacert string
        path to TLS CA certificate file (default "/Users/jason/.docker/machine/machines/default/ca.pem")
    -tlscert string
        path to TLS client certificate file (default "/Users/jason/.docker/machine/machines/default/cert.pem")
    -tlskey string
        path to TLS client key file (default "/Users/jason/.docker/machine/machines/default/key.pem")
    -tlsverify
        verify docker daemon's TLS certicate (default true)
    -version
        show version
    -watch
        watch for container changes
    -wait
        minimum (and/or maximum) duration to wait after each container change before triggering

  Arguments:
    template - path to a template to generate
    dest - path to a write the template. If not specfied, STDOUT is used

  Environment Variables:
    DOCKER_HOST - default value for -endpoint
    DOCKER_CERT_PATH - directory path containing key.pem, cert.pm and ca.pem
    DOCKER_TLS_VERIFY - enable client TLS verification]
```

If no `<dest>` file is specified, the output is sent to stdout. Mainly useful for debugging.

Configuration file

Using the -config flag from above you can tell docker-gen to use the specified config file instead of command-line options. Multiple templates can be defined and they will be executed in the order that they appear in the config file.

An example configuration file, **docker-gen.cfg** can be found in the examples folder.

Configuration File Syntax

```
[[config]]
Starts a configuration section

dest = "path/to/a/file"
path to a write the template. If not specfied, STDOUT is used

notifycmd = "/etc/init.d/foo reload"
run command after template is regenerated (e.g restart xyz)

onlyexposed = true
only include containers with exposed ports

template = "/path/to/a/template/file.tmpl"
path to a template to generate

watch = true
watch for container changes

wait = "500ms:2s"
debounce changes with a min:max duration. Only applicable if watch = true


[config.NotifyContainers]
Starts a notify container section
```

```
containername = 1
container name followed by the signal to send

container_id = 1
or the container id can be used followed by the signal to send
```

Putting it all together here is an example configuration file.

```
[[config]]
template = "/etc/nginx/nginx.conf.tmpl"
dest = "/etc/nginx/sites-available/default"
onlyexposed = true
notifycmd = "/etc/init.d/nginx reload"

[[config]]
template = "/etc/logrotate.conf.tmpl"
dest = "/etc/logrotate.d/docker"
watch = true

[[config]]
template = "/etc/docker-gen/templates/nginx.tmpl"
dest = "/etc/nginx/conf.d/default.conf"
watch = true
wait = "500ms:2s"

[config.NotifyContainers]
nginx = 1  # 1 is a signal number to be sent; here SIGHUP
e75a60548dc9 = 1  # a key can be either container name (nginx) or ID
```

===

Templating

The templates used by docker-gen are written using the Go text/template language. In addition to the built-in functions supplied by Go, docker-gen provides a number of additional functions to make it simpler (or possible) to generate your desired output.

Emit Structure

Within the templates, the object emitted by docker-gen will be a structure consisting of following Go structs:

```
type RuntimeContainer struct {
    ID           string
    Addresses    []Address
    Networks     []Network
    Gateway      string
    Name         string
    Hostname     string
    Image        DockerImage
    Env          map[string]string
    Volumes      map[string]Volume
    Node         SwarmNode
    Labels       map[string]string
    IP           string
    IP6LinkLocal string
    IP6Global    string
    Mounts       []Mount
    State        State
}

type Address struct {
    IP           string
    IP6LinkLocal string
    IP6Global    string
    Port         string
    HostPort     string
    Proto        string
    HostIP       string
}

type Network struct {
    IP                  string
    Name                string
```

```
        Gateway             string
        EndpointID          string
        IPv6Gateway         string
        GlobalIPv6Address   string
        MacAddress          string
        GlobalIPv6PrefixLen int
        IPPrefixLen         int
    }

    type DockerImage struct {
        Registry   string
        Repository string
        Tag        string
    }

    type Mount struct {
      Name        string
      Source      string
      Destination string
      Driver      string
      Mode        string
      RW          bool
    }

    type Volume struct {
        Path      string
        HostPath  string
        ReadWrite bool
    }

    type SwarmNode struct {
        ID      string
        Name    string
        Address Address
    }

    type State struct {
      Running bool
    }

    // Accessible from the root in templates as .Docker
    type Docker struct {
        Name                 string
        NumContainers        int
        NumImages            int
        Version              string
        ApiVersion           string
        GoVersion            string
        OperatingSystem      string
        Architecture         string
        CurrentContainerID   string
    }

    // Host environment variables accessible from root in templates as .Env
```

For example, this is a JSON version of an emitted RuntimeContainer struct:

```
{
   "ID":"71e9768075836eb38557adcfc71a207386a0c597dbeda240cf905df79b18cebf",
   "Addresses":[
      {
         "IP":"172.17.0.4",
         "Port":"22",
         "Proto":"tcp",
         "HostIP":"192.168.10.24",
         "HostPort":"2222"
      }
   ],
   "Gateway":"172.17.42.1",
   "Node": {
       "ID":"I2VY:P7PF:TZD5:PGWB:QTI7:QDSP:C5UD:DYKR:XKKK:TRG2:M2BL:DFUN",
       "Name":"docker-test",
       "Address": {
           "IP":"192.168.10.24"
       }
   },
   "Labels": {
       "operatingsystem":"Ubuntu 14.04.2 LTS",
       "storagedriver":"devicemapper",
       "anything_foo":"something_bar"
   },
   "IP":"172.17.0.4",
   "Name":"docker_register",
   "Hostname":"71e976807583",
   "Image":{
      "Registry":"jwilder",
      "Repository":"docker-register"
   },
   "Env":{
      "ETCD_HOST":"172.17.42.1:4001",
      "PATH":"/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin",
      "DOCKER_HOST":"unix:///var/run/docker.sock",
      "HOST_IP":"172.17.42.1"
   },
   "Volumes":{
      "/mnt":{
         "Path":"/mnt",
         "HostPath":"/Users/joebob/tmp",
         "ReadWrite":true
      }
   }
}
```

Functions

- *`closest $array $value`* : Returns the longest matching substring in `$array` that matches `$value`
- *`coalesce ...`* : Returns the first non-nil argument.
- *`contains $map $key`* : Returns `true` if `$map` contains `$key` . Takes maps from `string` to `string` .
- *`dict $key $value ...`* : Creates a map from a list of pairs. Each `$key` value must be a `string` , but the `$value` can be any type (or `nil`). Useful for passing more than one value as a pipeline context to subtemplates.
- *`dir $path`* : Returns an array of filenames in the specified `$path` .
- *`exists $path`* : Returns `true` if `$path` refers to an existing file or directory. Takes a string.
- *`first $array`* : Returns the first value of an array or nil if the arry is nil or empty.
- *`groupBy $containers $fieldPath`* : Groups an array of `RuntimeContainer` instances based on the values of a field path expression `$fieldPath` . A field path expression is a dot-delimited list of map keys or struct member names specifying the path from container to a nested value, which must be a string. Returns a map from the value of the field path expression to an array of containers having that value. Containers that do not have a value for the field path in question are omitted.
- *`groupByKeys $containers $fieldPath`* : Returns the same as `groupBy` but only returns the keys of the map.
- *`groupByMulti $containers $fieldPath $sep`* : Like `groupBy` , but the string value specified by `$fieldPath` is first split by `$sep` into a list of strings. A container whose `$fieldPath` value contains a list of strings will show up in the map output under each of those strings.
- *`groupByLabel $containers $label`* : Returns the same as `groupBy` but grouping by the given label's value.
- *`hasPrefix $prefix $string`* : Returns whether `$prefix` is a prefix of `$string` .
- *`hasSuffix $suffix $string`* : Returns whether `$suffix` is a suffix of `$string` .
- *`intersect $slice1 $slice2`* : Returns the strings that exist in both string slices.
- *`json $value`* : Returns the JSON representation of `$value` as a `string` .
- *`keys $map`* : Returns the keys from `$map` . If `$map` is `nil` , a `nil` is returned. If `$map` is not a `map` , an error will be thrown.
- *`last $array`* : Returns the last value of an array.
- *`parseBool $string`* : parseBool returns the boolean value represented by the string. It accepts 1, t, T, TRUE, true, True, 0, f, F, FALSE, false, False. Any other value returns an error. Alias for `strconv.ParseBool`
- *`replace $string $old $new $count`* : Replaces up to `$count` occurences of `$old` with `$new` in `$string` . Alias for `strings.Replace`
- *`sha1 $string`* : Returns the hexadecimal representation of the SHA1 hash of `$string` .
- *`split $string $sep`* : Splits `$string` into a slice of substrings delimited by `$sep` . Alias for `strings.Split`
- *`splitN $string $sep $count`* : Splits `$string` into a slice of substrings delimited by `$sep` , with number of substrings returned determined by `$count` . Alias for `strings.SplitN`
- *`trimPrefix $prefix $string`* : If `$prefix` is a prefix of `$string` , return `$string` with `$prefix` trimmed from the beginning. Otherwise, return `$string` unchanged.
- *`trimSuffix $suffix $string`* : If `$suffix` is a suffix of `$string` , return `$string` with `$suffix` trimmed from the end. Otherwise, return `$string` unchanged.
- *`trim $string`* : Removes whitespace from both sides of `$string` .

- *`when $condition $trueValue $falseValue`* : Returns the `$trueValue` when the `$condition` is `true` and the `$falseValue` otherwise
- *`where $items $fieldPath $value`* : Filters an array or slice based on the values of a field path expression `$fieldPath` . A field path expression is a dot-delimited list of map keys or struct member names specifying the path from container to a nested value. Returns an array of items having that value.
- *`whereNot $items $fieldPath $value`* : Filters an array or slice based on the values of a field path expression `$fieldPath` . A field path expression is a dot-delimited list of map keys or struct member names specifying the path from container to a nested value. Returns an array of items **not** having that value.
- *`whereExist $items $fieldPath`* : Like `where` , but returns only items where `$fieldPath` exists (is not nil).
- *`whereNotExist $items $fieldPath`* : Like `where` , but returns only items where `$fieldPath` does not exist (is nil).
- *`whereAny $items $fieldPath $sep $values`* : Like `where` , but the string value specified by `$fieldPath` is first split by `$sep` into a list of strings. The comparison value is a string slice with possible matches. Returns items which OR intersect these values.
- *`whereAll $items $fieldPath $sep $values`* : Like `whereAny` , except all `$values` must exist in the `$fieldPath` .
- *`whereLabelExists $containers $label`* : Filters a slice of containers based on the existence of the label `$label` .
- *`whereLabelDoesNotExist $containers $label`* : Filters a slice of containers based on the non-existence of the label `$label` .
- *`whereLabelValueMatches $containers $label $pattern`* : Filters a slice of containers based on the existence of the label `$label` with values matching the regular expression `$pattern` .

===

Examples

- Automated Nginx Reverse Proxy for Docker
- Docker Log Management With Fluentd
- Docker Service Discovery Using Etcd and Haproxy

NGINX Reverse Proxy Config

jwilder/nginx-proxy trusted build.

Start nginx-proxy:

```
$ docker run -d -p 80:80 -v /var/run/docker.sock:/tmp/docker.sock -t jwilder/nginx-proxy
```

Then start containers with a VIRTUAL_HOST env variable:

```
$ docker run -e VIRTUAL_HOST=foo.bar.com -t ...
```

If you wanted to run docker-gen directly on the host, you could do it with:

```
$ docker-gen -only-published -watch -notify "/etc/init.d/nginx reload" templates/nginx.tmpl /etc/nginx/sites-
enabled/default
```

Fluentd Log Management

This template generate a fluentd.conf file used by fluentd. It would then ship log files off the host.

```
$ docker-gen -watch -notify "restart fluentd" templates/fluentd.tmpl /etc/fluent/fluent.conf
```

Service Discovery in Etcd

This template is an example of generating a script that is then executed. This template generates a python script that is then executed which register containers in Etcd using its HTTP API.

```
$ docker-gen -notify "/bin/bash /tmp/etcd.sh" -interval 10 templates/etcd.tmpl /tmp/etcd.sh
```

Development

This project uses glock for managing 3rd party dependencies. You'll need to install glock into your workspace before hacking on docker-gen.

```
$ git clone <your fork>
$ cd <your fork>
$ make get-deps
$ make
```

TODO

- Add event status for handling start and stop events differently

License

MIT