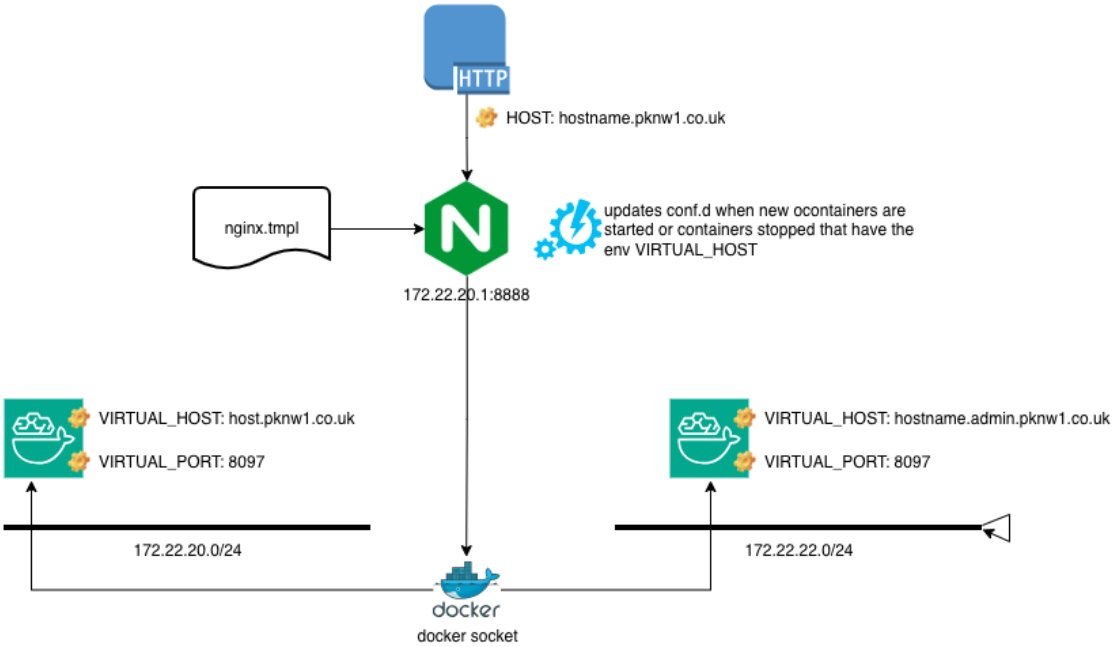
jwilder/nginx-proxy

 Nginx Reverse Proxy	

- ☐ Monitors all containers via /var/run/.docker.sock
- ☐ if ENV var VIRTUAL_HOST, adds a new reverse proxy instance
- ☐ if ENV var VIRTUAL_PORT, uses the port else uses 80
- ☐ HTTP request received
- ☐ if HOST is recognised, proxy to the server

Typically a request will be received by Nginx Proxy Manager and if no specific named host is setup, will pass the request throigh to 172.22.20.1:8888 - the nginx proxy - which will either accept or reject the request based on hostname




```
name: internal_proxy_core
services:
  internal_proxy:
    image: docker.io/nginxproxy/nginx-proxy:latest
    container_name: internal_proxy
    hostname: internal_proxy
    volumes:
      - /tmp:/tmp
      - ./nginx.tmpl:/app/nginx.tmpl
      - ./custom_opts.conf:/etc/nginx/conf.d/custom.conf
      - ./404.conf:/etc/nginx/snippets/error.conf
      - ./errors:/var/www/errors
      - ./errors:/usr/share/nginx/html
      - /var/run/docker.sock:/tmp/docker.sock:ro
      - /var/run/fcgiwrap.socket:/var/run/fcgiwrap.socket
      - /var/run/php-fpm.sock:/var/run/php-fpm.sock
    ports:
      - 172.22.20.1:8888:80
      - 172.22.22.1:8888:80
    dns:
```

```
      - 127.0.0.11
      - 172.22.20.1
      - 8.8.8.8
    networks:
      proxy:
      admin:
    environment:
      - PUID=20210§
      - PGID=20210
      - DISABLE_ACCESS_LOGS=true
      - TRUST_DOWNSTREAM_PROXY=true
networks:
  admin:
    external: true

  proxy:
    external: true
```

custom nginx.tmpl

```
# nginx-proxy{{ if $.Env.NGINX_PROXY_VERSION }} version : {{ $.Env.NGINX_PROXY_VERSION
}}{{ end }}

{{- /*
     * Global values. Values are stored in this map rather than in individual
     * global variables so that the values can be easily passed to embedded
     * templates (Go templates cannot access variables outside of their own
     * scope) and displayed in the debug endpoint output.
     */}}
{{- $globals := dict }}
{{- $_ := set $globals "containers" $ }}
{{- $_ := set $globals "Env" $.Env }}
{{- $_ := set $globals "Docker" $.Docker }}
{{- $_ := set $globals "CurrentContainer" (where $globals.containers "ID"
$globals.Docker.CurrentContainerID | first) }}

{{- $config := dict }}
{{- $_ := set $config "nginx_proxy_version" $.Env.NGINX_PROXY_VERSION }}
{{- $_ := set $config "default_cert_ok" (and (exists "/etc/nginx/certs/default.crt")
```

```
(exists "/etc/nginx/certs/default.key")) }}
{{- $_ := set $config "external_http_port" ($globals.Env.HTTP_PORT | default "80") }}
{{- $_ := set $config "external_https_port" ($globals.Env.HTTPS_PORT | default "443") }}
{{- $_ := set $config "sha1_upstream_name" ($globals.Env.SHA1_UPSTREAM_NAME | default
"false" | parseBool) }}
{{- $_ := set $config "default_root_response" ($globals.Env.DEFAULT_ROOT | default "404")
}}
{{- $_ := set $config "trust_default_cert" ($globals.Env.TRUST_DEFAULT_CERT | default
"true") }}
{{- $_ := set $config "trust_downstream_proxy" ($globals.Env.TRUST_DOWNSTREAM_PROXY |
default "true" | parseBool) }}
{{- $_ := set $config "enable_access_log" ($globals.Env.DISABLE_ACCESS_LOGS | default
"false" | parseBool | not) }}
{{- $_ := set $config "enable_ipv6" ($globals.Env.ENABLE_IPV6 | default "false" |
parseBool) }}
{{- $_ := set $config "prefer_ipv6_network" ($globals.Env.PREFER_IPV6_NETWORK | default
"false" | parseBool) }}
{{- $_ := set $config "ssl_policy" ($globals.Env.SSL_POLICY | default "Mozilla-
Intermediate") }}
{{- $_ := set $config "enable_debug_endpoint" ($globals.Env.DEBUG_ENDPOINT | default
"false") }}
{{- $_ := set $config "hsts" ($globals.Env.HSTS | default "max-age=31536000") }}
{{- $_ := set $config "acme_http_challenge" ($globals.Env.ACME_HTTP_CHALLENGE_LOCATION |
default "true") }}
{{- $_ := set $config "enable_http2" ($globals.Env.ENABLE_HTTP2 | default "true") }}
{{- $_ := set $config "enable_http3" ($globals.Env.ENABLE_HTTP3 | default "false") }}
{{- $_ := set $config "enable_http_on_missing_cert"
($globals.Env.ENABLE_HTTP_ON_MISSING_CERT | default "true") }}
{{- $_ := set $config "https_method" ($globals.Env.HTTPS_METHOD | default "redirect") }}
{{- $_ := set $config "non_get_redirect" ($globals.Env.NON_GET_REDIRECT | default "301")
}}
{{- $_ := set $config "default_host" $globals.Env.DEFAULT_HOST }}
{{- $_ := set $config "resolvers" $globals.Env.RESOLVERS }}
{{- /* LOG_JSON is a shorthand that sets logging defaults to JSON format */}}
{{- $_ := set $config "enable_json_logs" ($globals.Env.LOG_JSON | default "false" |
parseBool) }}
{{- $_ := set $config "log_format" $globals.Env.LOG_FORMAT }}
{{- $_ := set $config "log_format_escape" $globals.Env.LOG_FORMAT_ESCAPE }}
```

```
{{- $_ := set $globals "config" $config }}

{{- $_ := set $globals "vhosts" (dict) }}
{{- $_ := set $globals "networks" (dict) }}
# Networks available to the container running docker-gen (which are assumed to
# match the networks available to the container running nginx):
{{- /*
     * Note: $globals.CurrentContainer may be nil in some circumstances due to
     * <https://github.com/nginx-proxy/docker-gen/issues/458>.  For more context
     * see <https://github.com/nginx-proxy/nginx-proxy/issues/2189>.
     */}}
{{- if $globals.CurrentContainer }}
    {{- range sortObjectsByKeysAsc $globals.CurrentContainer.Networks "Name" }}
        {{- $_ := set $globals.networks .Name . }}
#     {{ .Name }}
    {{- else }}
#     (none)
    {{- end }}
{{- else }}
# /!\ WARNING: Failed to find the Docker container running docker-gen.  All
#              upstream (backend) application containers will appear to be
#              unreachable.  Try removing the -only-exposed and -only-published
#              arguments to docker-gen if you pass either of those.  See
#              <https://github.com/nginx-proxy/docker-gen/issues/458>.
{{- end }}

{{- /*
     * Template used as a function to get a container's IP address.  This
     * template only outputs debug comments; the IP address is "returned" by
     * storing the value in the provided dot dict.
     *
     * The provided dot dict is expected to have the following entries:
     *   - "globals": Global values.
     *   - "container": The container's RuntimeContainer struct.
     *
     * The return value will be added to the dot dict with key "ip".
     */}}
{{- define "container_ip" }}
    {{- $ipv4 := "" }}
```

```
{{- $ipv6 := "" }}
#     networks:
{{- range sortObjectsByKeysAsc $.container.Networks "Name" }}
    {{- /*
         * TODO: Only ignore the "ingress" network for Swarm tasks (in case
         * the user is not using Swarm mode and names a network "ingress").
         */}}
    {{- if eq .Name "ingress" }}
#         {{ .Name }} (ignored)
        {{- continue }}
    {{- end }}
    {{- if eq .Name "host" }}
        {{- /* Handle containers in host nework mode */}}
        {{- if (index $.globals.networks "host") }}
#         both container and proxy are in host network mode, using localhost IP
            {{- $ipv4 = "127.0.0.1" }}
            {{- continue }}
        {{- end }}
        {{- range sortObjectsByKeysAsc $.globals.CurrentContainer.Networks "Name" }}
            {{- if and . .Gateway (not .Internal) }}
#         container is in host network mode, using {{ .Name }} gateway IP
                {{- $ipv4 = .Gateway }}
                {{- break }}
            {{- end }}
        {{- end }}
        {{- if $ipv4 }}
            {{- continue }}
        {{- end }}
    {{- end }}
    {{- if and (not (index $.globals.networks .Name)) (not $.globals.networks.host) }}
#         {{ .Name }} (unreachable)
        {{- continue }}
    {{- end }}
    {{- /*
         * Do not emit multiple `server` directives for this container if it
         * is reachable over multiple networks or multiple IP stacks. This avoids
         * accidentally inflating the effective round-robin weight of a server due
         * to the redundant upstream addresses that nginx sees as belonging to
         * distinct servers.
```

```
             */}}
        {{- if or $ipv4 $ipv6 }}
    #         {{ .Name }} (ignored; reachable but redundant)
            {{- continue }}
        {{- end }}
    #         {{ .Name }} (reachable)
        {{- if and . .IP }}
            {{- $ipv4 = .IP }}
        {{- end }}
        {{- if and . .GlobalIPv6Address }}
            {{- $ipv6 = .GlobalIPv6Address }}
        {{- end }}
        {{- if and (empty $ipv4) (empty $ipv6) }}
    #             /!\ No IPv4 or IPv6 for this network!
        {{- end }}
    {{- else }}
    #         (none)
    {{- end }}
    {{ if and $ipv6 $.globals.config.prefer_ipv6_network }}
    #     IPv4 address: {{ if $ipv4 }}{{ $ipv4 }} (ignored; reachable but IPv6 prefered){{
else }}(none usable){{ end }}
    #     IPv6 address: {{ $ipv6 }}
        {{- $_ := set $ "ip" (printf "[%s]" $ipv6) }}
    {{- else }}
    #     IPv4 address: {{ if $ipv4 }}{{ $ipv4 }}{{ else }}(none usable){{ end }}
    #     IPv6 address: {{ if $ipv6 }}{{ $ipv6 }}{{ if $ipv4 }} (ignored; reachable but
IPv4 prefered){{ end }}{{ else }}(none usable){{ end }}
        {{- if $ipv4 }}
            {{- $_ := set $ "ip" $ipv4 }}
        {{- else if $ipv6}}
            {{- $_ := set $ "ip" (printf "[%s]" $ipv6) }}
        {{- end }}
    {{- end }}
{{- end }}

{{- /*
     * Template used as a function to get the port of the server in the given
     * container.  This template only outputs debug comments; the port is
     * "returned" by storing the value in the provided dot dict.
```

```
     *
     * The provided dot dict is expected to have the following entries:
     *   - "container": The container's RuntimeContainer struct.
     *
     * The return value will be added to the dot dict with key "port".
     */}}
{{- define "container_port" }}
    {{- /* If only 1 port exposed, use that as a default, else 80. */}}
    #     exposed ports (first ten):{{ range $index, $address := (sortObjectsByKeysAsc
$.container.Addresses "Port") }}{{ if lt $index 10 }} {{ $address.Port }}/{{
$address.Proto }}{{ end }}{{ else }} (none){{ end }}
    {{- $default_port := when (eq (len $.container.Addresses) 1) (first
$.container.Addresses).Port "80" }}
    #     default port: {{ $default_port }}
    {{- $port := eq $.port "default" | ternary $default_port $.port }}
    #     using port: {{ $port }}
    {{- $addr_obj := where $.container.Addresses "Port" $port | first }}
    {{- if and $addr_obj $addr_obj.HostPort }}
    #         /!\ WARNING: Virtual port published on host.  Clients
    #                      might be able to bypass nginx-proxy and
    #                      access the container's server directly.
    {{- end }}
    {{- $_ := set $ "port" $port }}
{{- end }}

{{- define "ssl_policy" }}
    {{- if eq .ssl_policy "Mozilla-Modern" }}
    ssl_protocols TLSv1.3;
    {{- /*
         * This ssl_ciphers directive is not used but necessary to get TLSv1.3 only.
         * see https://serverfault.com/questions/1023766/nginx-with-only-tls1-3-cipher-
suites
         */}}
    ssl_ciphers ECDHE-ECDSA-AES256-GCM-SHA384;
    ssl_conf_command Ciphersuites
TLS_AES_128_GCM_SHA256:TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256;
    ssl_prefer_server_ciphers off;
    {{- else if eq .ssl_policy "Mozilla-Intermediate" }}
    ssl_protocols TLSv1.2 TLSv1.3;
```

```
    ssl_ciphers 'ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:DHE-RSA-CHACHA20-POLY1305';
    ssl_prefer_server_ciphers off;
    {{- else if eq .ssl_policy "Mozilla-Old" }}
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2 TLSv1.3;
    ssl_ciphers 'ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:DHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES256-SHA:ECDHE-RSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:AES128-GCM-SHA256:AES256-GCM-SHA384:AES128-SHA256:AES256-SHA256:AES128-SHA:AES256-SHA:DES-CBC3-SHA:@SECLEVEL=0';
    ssl_prefer_server_ciphers on;
    {{- else if eq .ssl_policy "AWS-TLS13-1-3-2021-06" }}
    ssl_protocols TLSv1.3;
    {{- /*
         * This ssl_ciphers directive is not used but necessary to get TLSv1.3 only.
         * see https://serverfault.com/questions/1023766/nginx-with-only-tls1-3-cipher-suites
         */}}
    ssl_ciphers ECDHE-ECDSA-AES256-GCM-SHA384;
    ssl_conf_command Ciphersuites TLS_AES_128_GCM_SHA256:TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256;
    ssl_prefer_server_ciphers on;
    {{- else if eq .ssl_policy "AWS-TLS13-1-2-2021-06" }}
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers 'ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384';
    ssl_prefer_server_ciphers on;
    {{- else if eq .ssl_policy "AWS-TLS13-1-2-Res-2021-06" }}
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers 'ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384';
    ssl_prefer_server_ciphers on;
    {{- else if eq .ssl_policy "AWS-TLS13-1-2-Ext1-2021-06" }}
```

```
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers 'ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:AES128-GCM-SHA256:AES128-SHA256:AES256-GCM-SHA384:AES256-SHA256';
    ssl_prefer_server_ciphers on;
    {{- else if eq .ssl_policy "AWS-TLS13-1-2-Ext2-2021-06" }}
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers 'ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:AES128-GCM-SHA256:AES128-SHA256:AES128-SHA:AES256-GCM-SHA384:AES256-SHA256:AES256-SHA';
    ssl_prefer_server_ciphers on;
    {{- else if eq .ssl_policy "AWS-TLS13-1-1-2021-06" }}
    ssl_protocols TLSv1.1 TLSv1.2 TLSv1.3;
    ssl_ciphers 'ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:AES128-GCM-SHA256:AES128-SHA256:AES128-SHA:AES256-GCM-SHA384:AES256-SHA256:AES256-SHA:@SECLEVEL=0';
    ssl_prefer_server_ciphers on;
    {{- else if eq .ssl_policy "AWS-TLS13-1-0-2021-06" }}
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2 TLSv1.3;
    ssl_ciphers 'ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:AES128-GCM-SHA256:AES128-SHA256:AES128-SHA:AES256-GCM-SHA384:AES256-SHA256:AES256-SHA:@SECLEVEL=0';
    ssl_prefer_server_ciphers on;
    {{- else if eq .ssl_policy "AWS-FS-1-2-Res-2020-10" }}
    ssl_protocols TLSv1.2;
    ssl_ciphers 'ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384';
    ssl_prefer_server_ciphers on;
    {{- else if eq .ssl_policy "AWS-FS-1-2-Res-2019-08" }}
    ssl_protocols TLSv1.2;
    ssl_ciphers 'ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-
```

```
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384';
    ssl_prefer_server_ciphers on;
    {{- else if eq .ssl_policy "AWS-FS-1-2-2019-08" }}
    ssl_protocols TLSv1.2;
    ssl_ciphers 'ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES128-SHA:ECDHE-
ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-
AES256-SHA384:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA';
    ssl_prefer_server_ciphers on;
    {{- else if eq .ssl_policy "AWS-FS-1-1-2019-08" }}
    ssl_protocols TLSv1.1 TLSv1.2;
    ssl_ciphers 'ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES128-SHA:ECDHE-
ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-
AES256-SHA384:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:@SECLEVEL=0';
    ssl_prefer_server_ciphers on;
    {{- else if eq .ssl_policy "AWS-FS-2018-06" }}
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    ssl_ciphers 'ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES128-SHA:ECDHE-
ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-
AES256-SHA384:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:@SECLEVEL=0';
    ssl_prefer_server_ciphers on;
    {{- else if eq .ssl_policy "AWS-TLS-1-2-Ext-2018-06" }}
    ssl_protocols TLSv1.2;
    ssl_ciphers 'ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES128-SHA:ECDHE-
ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-
AES256-SHA384:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:AES128-GCM-SHA256:AES128-
SHA256:AES128-SHA:AES256-GCM-SHA384:AES256-SHA256:AES256-SHA';
    ssl_prefer_server_ciphers on;
    {{- else if eq .ssl_policy "AWS-TLS-1-2-2017-01" }}
    ssl_protocols TLSv1.2;
    ssl_ciphers 'ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:AES128-GCM-SHA256:AES128-
SHA256:AES256-GCM-SHA384:AES256-SHA256';
    ssl_prefer_server_ciphers on;
    {{- else if eq .ssl_policy "AWS-TLS-1-1-2017-01" }}
```

```
    ssl_protocols TLSv1.1 TLSv1.2;
    ssl_ciphers 'ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:AES128-GCM-SHA256:AES128-SHA256:AES128-SHA:AES256-GCM-SHA384:AES256-SHA256:AES256-SHA:@SECLEVEL=0';
    ssl_prefer_server_ciphers on;
    {{- else if eq .ssl_policy "AWS-2016-08" }}
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    ssl_ciphers 'ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:AES128-GCM-SHA256:AES128-SHA256:AES128-SHA:AES256-GCM-SHA384:AES256-SHA256:AES256-SHA:@SECLEVEL=0';
    ssl_prefer_server_ciphers on;
    {{- else if eq .ssl_policy "AWS-2015-05" }}
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    ssl_ciphers 'ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:AES128-GCM-SHA256:AES128-SHA256:AES128-SHA:AES256-GCM-SHA384:AES256-SHA256:AES256-SHA:DES-CBC3-SHA:@SECLEVEL=0';
    ssl_prefer_server_ciphers on;
    {{- else if eq .ssl_policy "AWS-2015-03" }}
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    ssl_ciphers 'ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES128-SHA:DHE-RSA-AES128-SHA:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:AES128-GCM-SHA256:AES128-SHA256:AES128-SHA:AES256-GCM-SHA384:AES256-SHA256:AES256-SHA:DHE-DSS-AES128-SHA:DES-CBC3-SHA:@SECLEVEL=0';
    ssl_prefer_server_ciphers on;
    {{- else if eq .ssl_policy "AWS-2015-02" }}
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    ssl_ciphers 'ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES128-SHA:DHE-RSA-AES128-SHA:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:AES128-GCM-SHA256:AES128-SHA256:AES128-SHA:AES256-GCM-SHA384:AES256-SHA256:AES256-SHA:DHE-DSS-AES128-
```

```
SHA:@SECLEVEL=0';
    ssl_prefer_server_ciphers on;
    {{- end }}
{{- end }}

{{- define "location" }}
    {{- $vpath := .VPath }}
    {{- $override := printf "/etc/nginx/vhost.d/%s_%s_location_override" .Host (sha1
.Path) }}
    {{- if and (eq .Path "/") (not (exists $override)) }}
        {{- $override = printf "/etc/nginx/vhost.d/%s_location_override" .Host }}
    {{- end }}
    {{- if exists $override }}
    include {{ $override }};
    {{- else }}
        {{- $keepalive := $vpath.keepalive }}
    location {{ .Path }} {
        {{- if eq $vpath.network_tag "internal" }}
        # Only allow traffic from internal clients
        include /etc/nginx/network_internal.conf;
        {{- end }}

        {{ $proto := $vpath.proto }}
        {{ $upstream := $vpath.upstream }}
        {{ $dest := $vpath.dest }}
        {{- if eq $proto "uwsgi" }}
        include uwsgi_params;
        uwsgi_pass {{ trim $proto }}://{{ trim $upstream }};
        {{- else if eq $proto "fastcgi" }}
            {{- if (exists "/etc/nginx/fastcgi.conf") }}
        include fastcgi.conf;
            {{- else if (exists "/etc/nginx/fastcgi_params") }}
        include fastcgi_params;
            {{- else }}
        # neither /etc/nginx/fastcgi.conf nor /etc/nginx/fastcgi_params found, fastcgi
won't work
            {{- end }}
        root {{ trim .VhostRoot }};
        fastcgi_pass {{ trim $upstream }};
```

```
            {{- if ne $keepalive "disabled" }}
        fastcgi_keep_conn on;
            {{- end }}
        {{- else if eq $proto "grpc" }}
        grpc_pass {{ trim $proto }}://{{ trim $upstream }};
        {{- else if eq $proto "grpcs" }}
        grpc_pass {{ trim $proto }}://{{ trim $upstream }};
        {{- else }}
        proxy_pass {{ trim $proto }}://{{ trim $upstream }}{{ trim $dest }};
        set $upstream_keepalive {{ if ne $keepalive "disabled" }}true{{ else }}false{{ end
}};
        {{- end }}

        {{- if (exists (printf "/etc/nginx/htpasswd/%s_%s" .Host (sha1 .Path) )) }}
        auth_basic "Restricted {{ .Host }}{{ .Path }}";
        auth_basic_user_file {{ (printf "/etc/nginx/htpasswd/%s_%s" .Host (sha1 .Path))
}};
        {{- else if (exists (printf "/etc/nginx/htpasswd/%s" .Host)) }}
        auth_basic "Restricted {{ .HostIsRegexp | ternary "access" .Host }}";
        auth_basic_user_file {{ (printf "/etc/nginx/htpasswd/%s" .Host) }};
        {{- end }}

        {{- if (exists (printf "/etc/nginx/vhost.d/%s_%s_location" .Host (sha1 .Path) ))
}}
        include {{ printf "/etc/nginx/vhost.d/%s_%s_location" .Host (sha1 .Path) }};
        {{- else if (exists (printf "/etc/nginx/vhost.d/%s_location" .Host)) }}
        include {{ printf "/etc/nginx/vhost.d/%s_location" .Host}};
        {{- else if (exists "/etc/nginx/vhost.d/default_location") }}
        include /etc/nginx/vhost.d/default_location;
        {{- end }}
    }
    {{- end }}
{{- end }}

{{- define "upstream" }}
    {{- $path := .Path }}
    {{- $vpath := .VPath }}
upstream {{ $vpath.upstream }} {
    {{- $servers := 0 }}
```

```
    {{- $loadbalance := $vpath.loadbalance }}
    {{- if $loadbalance }}
    # From the container's loadbalance label:
    {{ $loadbalance }}
    {{- end }}
    {{- range $port, $containers := $vpath.ports }}
        {{- range $container := $containers }}
    # Container: {{ $container.Name }}
            {{- $args := dict "globals" $.globals "container" $container }}
            {{- template "container_ip" $args }}
            {{- $ip := $args.ip }}
            {{- $args = dict "container" $container "path" $path "port" $port }}
            {{- template "container_port" $args }}
            {{- if $ip }}
                {{- $servers = add1 $servers }}
    server {{ $ip }}:{{ $args.port }};
            {{- end }}
        {{- end }}
    {{- end }}
    {{- /* nginx-proxy/nginx-proxy#1105 */}}
    {{- if lt $servers 1 }}
    # Fallback entry
    server 127.0.0.1 down;
    {{- end }}
    {{- $keepalive := $vpath.keepalive }}
    {{- if and (ne $keepalive "disabled") (gt $servers 0) }}
        {{- if eq $keepalive "auto" }}
    keepalive {{ mul $servers 2 }};
        {{- else }}
    keepalive {{ $keepalive }};
        {{- end }}
    {{- end }}
}
{{- end }}

{{- /* debug "endpoint" location template */}}
{{- define "debug_location" }}
    {{- $debug_paths := dict }}
    {{- range $path, $vpath := .VHost.paths }}
```

```
        {{- $tmp_ports := dict }}
        {{- range $port, $containers := $vpath.ports }}
            {{- $tmp_containers := list }}
            {{- range $container := $containers }}
                {{- $tmp_containers = dict "Name" $container.Name | append $tmp_containers
}}
            {{- end }}
            {{- $_ := set $tmp_ports $port $tmp_containers }}
        {{- end }}
        {{- $debug_vpath := deepCopy $vpath | merge (dict "ports" $tmp_ports) }}
        {{- $_ := set $debug_paths $path $debug_vpath }}
    {{- end }}

    {{- $debug_vhost := deepCopy .VHost }}
    {{- /* If it's a regexp, do not render the Hostname to the response to avoid rendering
config breaking characters */}}
    {{- $_ := set $debug_vhost "hostname" (.VHost.is_regexp | ternary "Hostname is a
regexp and unsafe to include in the debug response." .Hostname) }}
    {{- $_ := set $debug_vhost "paths" $debug_paths }}

    {{- $debug_response := dict
        "global" .GlobalConfig
        "request" (dict
            "host" "$host"
            "https" "$https"
            "http2" "$http2"
            "http3" "$http3"
            "ssl_cipher" "$ssl_cipher"
            "ssl_protocol" "$ssl_protocol"
        )
        "vhost" $debug_vhost
    }}

    {{- /*
         * The maximum line length in an nginx config is 4096 characters.
         * If we're nearing this limit (with headroom for the rest
         * of the directive), strip vhost.paths from the response.
         */}}
    {{- if gt (toJson $debug_response | len) 4000 }}
```

```
        {{- $_ := unset $debug_vhost "paths" }}
        {{- $_ := set $debug_response "warning" "Virtual paths configuration for this
hostname is too large and has been stripped from response." }}
    {{- end }}

    location  /nginx-proxy-debug {
        default_type application/json;
        return 200 '{{ toJson $debug_response }}{{ "\\n" }}';
    }
{{- end }}

{{- define "access_log" }}
    {{- when .Enable "access_log /var/log/nginx/access.log vhost;" "" }}
{{- end }}

# If we receive X-Forwarded-Proto, pass it through; otherwise, pass along the
# scheme used to connect to this server
map $http_x_forwarded_proto $proxy_x_forwarded_proto {
    default {{ if $globals.config.trust_downstream_proxy }}$http_x_forwarded_proto{{ else
}}$scheme{{ end }};
    '' $scheme;
}

map $http_x_forwarded_host $proxy_x_forwarded_host {
    default {{ if $globals.config.trust_downstream_proxy }}$http_x_forwarded_host{{ else
}}$host{{ end }};
    '' $host;
}

# If we receive X-Forwarded-Port, pass it through; otherwise, pass along the
# server port the client connected to
map $http_x_forwarded_port $proxy_x_forwarded_port {
    default {{ if $globals.config.trust_downstream_proxy }}$http_x_forwarded_port{{ else
}}$server_port{{ end }};
    '' $server_port;
}

# Include the port in the Host header sent to the container if it is non-standard
map $server_port $host_port {
```

```
    default :$server_port;
    80 '';
    443 '';
}

# If the request from the downstream client has an "Upgrade:" header (set to any
# non-empty value), pass "Connection: upgrade" to the upstream (backend) server.
# Otherwise, the value for the "Connection" header depends on whether the user
# has enabled keepalive to the upstream server.
map $http_upgrade $proxy_connection {
    default upgrade;
    '' $proxy_connection_noupgrade;
}
map $upstream_keepalive $proxy_connection_noupgrade {
    # Preserve nginx's default behavior (send "Connection: close").
    default close;
    # Use an empty string to cancel nginx's default behavior.
    true '';
}
# Abuse the map directive (see <https://stackoverflow.com/q/14433309>) to ensure
# that $upstream_keepalive is always defined.  This is necessary because:
#   - The $proxy_connection variable is indirectly derived from
#     $upstream_keepalive, so $upstream_keepalive must be defined whenever
#     $proxy_connection is resolved.
#   - The $proxy_connection variable is used in a proxy_set_header directive in
#     the http block, so it is always fully resolved for every request -- even
#     those where proxy_pass is not used (e.g., unknown virtual host).
map "" $upstream_keepalive {
    # The value here should not matter because it should always be overridden in
    # a location block (see the "location" template) for all requests where the
    # value actually matters.
    default false;
}

# Apply fix for very long server names
server_names_hash_bucket_size 128;

# Default dhparam
{{- if (exists "/etc/nginx/dhparam/dhparam.pem") }}
```

```
ssl_dhparam /etc/nginx/dhparam/dhparam.pem;
{{- end }}

# Set appropriate X-Forwarded-Ssl header based on $proxy_x_forwarded_proto
map $proxy_x_forwarded_proto $proxy_x_forwarded_ssl {
    default off;
    https on;
}

gzip_types text/plain text/css application/javascript application/json application/x-
javascript text/xml application/xml application/xml+rss text/javascript;


{{- /* See https://nginx.org/en/docs/http/ngx_http_log_module.html#log_format for details
and variables
     * LOG_FORMAT_ESCAPE sets the escape part of the log format
     * LOG_FORMAT        sets the log format
     */}}
{{- $logEscape := $globals.config.log_format_escape | default "default" | printf
"escape=%s" }}
{{- $logFormat := $globals.config.log_format | default `$host $remote_addr - $remote_user
[$time_local] "$request" $status $body_bytes_sent "$http_referer" "$http_user_agent"
"$upstream_addr"` }}

{{- if $globals.config.enable_json_logs }}
# JSON Logging enabled (via LOG_JSON env variable)
    {{- $logEscape = $globals.config.log_format_escape | default "json" | printf
"escape=%s" }}
    {{- $logFormat = $globals.config.log_format | default
`{"time_local":"$time_iso8601","client_ip":"$http_x_forwarded_for","remote_addr":"$remote_
addr","request":"$request","status":"$status","body_bytes_sent":"$body_bytes_sent","reques
t_time":"$request_time","upstream_response_time":"$upstream_response_time","upstream_addr"
:"$upstream_addr","http_referrer":"$http_referer","http_user_agent":"$http_user_agent","re
quest_id":"$request_id"}` }}
{{- end }}

log_format vhost {{ $logEscape }} '{{ $logFormat }}';

access_log off;
```

```
{{- /* Lower the SSL policy of the http context
    * if at least one vhost use a TLSv1 or TLSv1.1 policy
    * so TLSv1 and TLSv1.1 can be enabled on those vhosts
    */}}
{{- $httpContextSslPolicy := $globals.config.ssl_policy }}
{{- $inUseSslPolicies := groupByKeys $globals.containers "Env.SSL_POLICY" }}
{{- range $tls1Policy := list "AWS-TLS13-1-1-2021-06" "AWS-TLS13-1-0-2021-06" "AWS-FS-1-1-
2019-08" "AWS-FS-2018-06" "AWS-TLS-1-1-2017-01" "AWS-2016-08" "AWS-2015-05" "AWS-2015-03"
"AWS-2015-02" "Mozilla-Old" }}
    {{- if has $tls1Policy $inUseSslPolicies }}
# Using Mozilla-Old SSL policy on the http context to allow TLSv1 and TLSv1.1
        {{- $httpContextSslPolicy = "Mozilla-Old" }}
        {{- break }}
    {{- end }}
{{- end }}

{{- template "ssl_policy" (dict "ssl_policy" $httpContextSslPolicy) }}
error_log /dev/stderr;

{{- if $globals.config.resolvers }}
resolver {{ $globals.config.resolvers }};
{{- end }}

{{- if (exists "/etc/nginx/proxy.conf") }}
include /etc/nginx/proxy.conf;
{{- else }}
# HTTP 1.1 support
proxy_http_version 1.1;
proxy_set_header Host $host$host_port;
proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection $proxy_connection;
proxy_set_header X-Real-IP $remote_addr;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header X-Forwarded-Host $proxy_x_forwarded_host;
proxy_set_header X-Forwarded-Proto $proxy_x_forwarded_proto;
proxy_set_header X-Forwarded-Ssl $proxy_x_forwarded_ssl;
proxy_set_header X-Forwarded-Port $proxy_x_forwarded_port;
proxy_set_header X-Original-URI $request_uri;
```

```
# Mitigate httpoxy attack (see README for details)
proxy_set_header Proxy "";
{{- end }}

{{- /* Precompute and store some information about vhost that use VIRTUAL_HOST_MULTIPORTS.
*/}}
{{- range $vhosts_yaml, $containers := groupBy $globals.containers
"Env.VIRTUAL_HOST_MULTIPORTS" }}
    {{- /* Print a warning in the config if VIRTUAL_HOST_MULTIPORTS can't be parsed. */}}
    {{- $parsedVhosts := fromYaml $vhosts_yaml }}
    {{- if (empty $parsedVhosts) }}
        {{- $containerNames := list }}
        {{- range $container := $containers }}
            {{- $containerNames = append $containerNames $container.Name }}
        {{- end }}
# /!\ WARNING: the VIRTUAL_HOST_MULTIPORTS environment variable used for {{ len
$containerNames | plural "this container" "those containers" }} is not a valid YAML
string:
# {{ $containerNames | join ", " }}
        {{- continue }}
    {{- end }}

    {{- range $hostname, $vhost := $parsedVhosts }}
        {{- $vhost_data := get $globals.vhosts $hostname | default (dict) }}
        {{- $paths := $vhost_data.paths | default (dict) }}

        {{- if (empty $vhost) }}
            {{ $vhost = dict "/" (dict) }}
        {{- end }}

        {{- range $path, $vpath := $vhost }}
            {{- if (empty $vpath) }}
                {{- $vpath = dict
                    "dest" ""
                    "port" "default"
                    "proto" "http"
                }}
            {{- end }}
```

```
            {{- $dest := $vpath.dest | default "" }}
            {{- $port := $vpath.port | default "default" | toString }}
            {{- $proto := $vpath.proto | default "http" }}

            {{- $path_data := get $paths $path | default (dict) }}
            {{- $path_ports := $path_data.ports | default (dict) }}
            {{- $path_port_containers := get $path_ports $port | default (list) | concat
$containers }}
            {{- $_ := set $path_ports $port $path_port_containers }}
            {{- $_ := set $path_data "ports" $path_ports }}

            {{- if (not (hasKey $path_data "dest")) }}
                {{- $_ := set $path_data "dest" $dest }}
            {{- end }}

            {{- if (not (hasKey $path_data "proto")) }}
                {{- $_ := set $path_data "proto" $proto }}
            {{- end }}

            {{- $_ := set $paths $path $path_data }}
        {{- end }}
        {{- $_ := set $vhost_data "paths" $paths }}
        {{- $_ := set $globals.vhosts $hostname $vhost_data }}
    {{- end }}
{{- end }}

{{- /* Precompute and store some information about vhost that use VIRTUAL_HOST. */}}
{{- range $hostname, $containers := groupByMulti $globals.containers "Env.VIRTUAL_HOST"
"," }}
    {{- /* Ignore containers with VIRTUAL_HOST set to the empty string. */}}
    {{- $hostname = trim $hostname }}
    {{- if not $hostname }}
        {{- continue }}
    {{- end }}

    {{- /* Drop containers with both VIRTUAL_HOST and VIRTUAL_HOST_MULTIPORTS set
         * (VIRTUAL_HOST_MULTIPORTS takes precedence thanks to the previous loop).
         */}}
```

```
    {{- range $_, $containers_to_drop := groupBy $containers "Env.VIRTUAL_HOST_MULTIPORTS"
}}
        {{- range $container := $containers_to_drop }}
            {{- $containers = without $containers $container }}
        {{- end }}
    {{- end }}
    {{- if (eq (len $containers) 0) }}
        {{- continue }}
    {{- end }}

    {{- $vhost_data := get $globals.vhosts $hostname | default (dict) }}
    {{- $paths := $vhost_data.paths | default (dict) }}

    {{- $tmp_paths := groupByWithDefault $containers "Env.VIRTUAL_PATH" "/" }}

    {{- range $path, $containers := $tmp_paths }}
        {{- $dest := groupByKeys $containers "Env.VIRTUAL_DEST" | first | default "" }}
        {{- $proto := groupByKeys $containers "Env.VIRTUAL_PROTO" | first | default "http"
| trim }}

        {{- $path_data := get $paths $path | default (dict) }}
        {{- $path_ports := $path_data.ports | default (dict) }}
        {{- range $port, $containers := groupByWithDefault $containers "Env.VIRTUAL_PORT"
"default" }}
            {{- $path_port_containers := get $path_ports $port | default (list) | concat
$containers }}
            {{- $_ := set $path_ports $port $path_port_containers }}
        {{- end }}
        {{- $_ := set $path_data "ports" $path_ports }}

        {{- if (not (hasKey $path_data "dest")) }}
            {{- $_ := set $path_data "dest" $dest }}
        {{- end }}

        {{- if (not (hasKey $path_data "proto")) }}
            {{- $_ := set $path_data "proto" $proto }}
        {{- end }}

        {{- $_ := set $paths $path $path_data }}
```

```
    {{- end }}
    {{- $_ := set $vhost_data "paths" $paths }}
    {{- $_ := set $globals.vhosts $hostname $vhost_data }}
{{- end }}

{{- /* Loop over $globals.vhosts and update it with the remaining informations about each
vhost. */}}
{{- range $hostname, $vhost_data := $globals.vhosts }}
    {{- $is_regexp := hasPrefix "~" $hostname }}
    {{- $upstream_name := or $is_regexp $globals.config.sha1_upstream_name | ternary (sha1
$hostname) $hostname }}

    {{- $vhost_containers := list }}

    {{- range $path, $vpath_data := $vhost_data.paths }}
        {{- $vpath_containers := list }}
        {{- range $port, $vport_containers := $vpath_data.ports }}
            {{ $vpath_containers = concat $vpath_containers $vport_containers }}
        {{- end }}

        {{- /* Get the NETWORK_ACCESS defined by containers w/ the same vhost, falling
back to "external". */}}
        {{- $network_tag := groupByKeys $vpath_containers "Env.NETWORK_ACCESS" | first |
default "external" }}

        {{- $loadbalance := groupByLabel $vpath_containers "com.github.nginx-proxy.nginx-
proxy.loadbalance" | keys | first }}
        {{- $keepalive := groupByLabel $vpath_containers "com.github.nginx-proxy.nginx-
proxy.keepalive" | keys | first | default "auto" }}

        {{- $upstream := $upstream_name }}
        {{- if (not (eq $path "/")) }}
            {{- $sum := sha1 $path }}
            {{- $upstream = printf "%s-%s" $upstream $sum }}
        {{- end }}

        {{- $_ := set $vpath_data "network_tag" $network_tag }}
        {{- $_ := set $vpath_data "upstream" $upstream }}
        {{- $_ := set $vpath_data "loadbalance" $loadbalance }}
```

```
        {{- $_ := set $vpath_data "keepalive" $keepalive }}
        {{- $_ := set $vhost_data.paths $path $vpath_data }}

        {{ $vhost_containers = concat $vhost_containers $vpath_containers }}
    {{- end }}

    {{- $userIdentifiedCert := groupByKeys $vhost_containers "Env.CERT_NAME" | first }}

    {{- $vhostCert := "" }}
    {{- if exists (printf "/etc/nginx/certs/%s.crt" $hostname) }}
        {{- $vhostCert = $hostname }}
    {{- end }}

    {{- $parentVhostCert := "" }}
    {{- if gt ($hostname | sprigSplit "." | len) 2 }}
        {{- $parentHostname := ($hostname | sprigSplitn "." 2)._1 }}
        {{- if exists (printf "/etc/nginx/certs/%s.crt" $parentHostname) }}
            {{- $parentVhostCert = $parentHostname }}
        {{- end }}
    {{- end }}

    {{- $trust_default_cert := groupByLabel $vhost_containers "com.github.nginx-
proxy.nginx-proxy.trust-default-cert" | keys | first | default
$globals.config.trust_default_cert | parseBool }}
    {{- $defaultCert := and $trust_default_cert $globals.config.default_cert_ok | ternary
"default" "" }}

    {{- $cert := or $userIdentifiedCert $vhostCert $parentVhostCert $defaultCert }}
    {{- $cert_ok := and (ne $cert "") (exists (printf "/etc/nginx/certs/%s.crt" $cert))
(exists (printf "/etc/nginx/certs/%s.key" $cert)) }}

    {{- $enable_debug_endpoint := groupByLabel $vhost_containers "com.github.nginx-
proxy.nginx-proxy.debug-endpoint" | keys | first | default
$globals.config.enable_debug_endpoint | parseBool }}
    {{- $default := eq $globals.config.default_host $hostname }}
    {{- $https_method := groupByKeys $vhost_containers "Env.HTTPS_METHOD" | first |
default $globals.config.https_method }}
    {{- $enable_http_on_missing_cert := groupByKeys $vhost_containers
"Env.ENABLE_HTTP_ON_MISSING_CERT" | first | default
```

```
$globals.config.enable_http_on_missing_cert | parseBool }}
    {{- /* When no trusted certs (default and/or vhost) are present we want to ensure that HTTP is enabled; hence switching from 'nohttp' or 'redirect' to 'noredirect' */}}
    {{- $https_method_disable_http := list "nohttp" "redirect" | has $https_method }}
    {{- if and $https_method_disable_http (not $cert_ok) $enable_http_on_missing_cert }}
        {{- $https_method = "noredirect" }}
    {{- end }}
    {{- $non_get_redirect := groupByLabel $vhost_containers "com.github.nginx-proxy.nginx-proxy.non-get-redirect" | keys | first | default $globals.config.non_get_redirect }}

    {{- $http2_enabled := groupByLabel $vhost_containers "com.github.nginx-proxy.nginx-proxy.http2.enable" | keys | first | default $globals.config.enable_http2 | parseBool }}
    {{- $http3_enabled := groupByLabel $vhost_containers "com.github.nginx-proxy.nginx-proxy.http3.enable" | keys | first | default $globals.config.enable_http3 | parseBool }}

    {{- $acme_http_challenge := groupByKeys $vhost_containers "Env.ACME_HTTP_CHALLENGE_LOCATION" | first | default $globals.config.acme_http_challenge }}
    {{- $acme_http_challenge_legacy := eq $acme_http_challenge "legacy" }}
    {{- $acme_http_challenge_enabled := false }}
    {{- if (not $acme_http_challenge_legacy) }}
        {{- $acme_http_challenge_enabled = parseBool $acme_http_challenge }}
    {{- end }}

    {{- /* Get the SERVER_TOKENS defined by containers w/ the same vhost, falling back to "". */}}
    {{- $server_tokens := groupByKeys $vhost_containers "Env.SERVER_TOKENS" | first | default "" | trim }}

    {{- /* Get the SSL_POLICY defined by containers w/ the same vhost, falling back to empty string (use default). */}}
    {{- $ssl_policy := groupByKeys $vhost_containers "Env.SSL_POLICY" | first | default "" }}

    {{- /* Get the HSTS defined by containers w/ the same vhost, falling back to "max-age=31536000". */}}
    {{- $hsts := groupByKeys $vhost_containers "Env.HSTS" | first | default $globals.config.hsts }}
```

```
    {{- /* Get the VIRTUAL_ROOT By containers w/ use fastcgi root */}}
    {{- $vhost_root := groupByKeys $vhost_containers "Env.VIRTUAL_ROOT" | first | default
"/var/www/public" }}

    {{- $vhost_data = merge $vhost_data (dict
        "cert" $cert
        "cert_ok" $cert_ok
        "enable_debug_endpoint" $enable_debug_endpoint
        "default" $default
        "hsts" $hsts
        "https_method" $https_method
        "non_get_redirect" $non_get_redirect
        "http2_enabled" $http2_enabled
        "http3_enabled" $http3_enabled
        "is_regexp" $is_regexp
        "acme_http_challenge_legacy" $acme_http_challenge_legacy
        "acme_http_challenge_enabled" $acme_http_challenge_enabled
        "server_tokens" $server_tokens
        "ssl_policy" $ssl_policy
        "trust_default_cert" $trust_default_cert
        "upstream_name" $upstream_name
        "vhost_root" $vhost_root
    ) }}
    {{- $_ := set $globals.vhosts $hostname $vhost_data }}
{{- end }}


{{- /*
     * If needed, create a catch-all fallback server to send an error code to
     * clients that request something from an unknown vhost.
     *
     * This server must appear first in the generated config because nginx uses
     * the first `server` directive to handle requests that don't match any of
     * the other `server` directives.  An alternative approach would be to add
     * the `default_server` option to the `listen` directives inside this
     * `server`, but some users inject a custom `server` directive that uses
     * `default_server`.  Using `default_server` here would cause nginx to fail
     * to start for those users.  See
     * <https://github.com/nginx-proxy/nginx-proxy/issues/2212>.
```

```
     */}}
{{- block "fallback_server" $globals }}
    {{- $globals := . }}
    {{- $http_exists := false }}
    {{- $https_exists := false }}
    {{- $default_http_exists := false }}
    {{- $default_https_exists := false }}
    {{- $http3_enabled := false }}
    {{- range $vhost := $globals.vhosts }}
        {{- $http := ne $vhost.https_method "nohttp" }}
        {{- $https := ne $vhost.https_method "nohttps" }}
        {{- $http_exists = or $http_exists $http }}
        {{- $https_exists = or $https_exists $https }}
        {{- $default_http_exists = or $default_http_exists (and $http $vhost.default) }}
        {{- $default_https_exists = or $default_https_exists (and $https $vhost.default)
}}
        {{- $http3_enabled = or $http3_enabled $vhost.http3_enabled }}
    {{- end }}
    {{- $fallback_http := not $default_http_exists }}
    {{- $fallback_https := not $default_https_exists }}
    {{- /*
         * If there are no vhosts at all, create fallbacks for both plain http
         * and https so that clients get something more useful than a connection
         * refused error.
         */}}
    {{- if and (not $http_exists) (not $https_exists) }}
        {{- $fallback_https = true }}
    {{- end }}
    {{- if or $fallback_http $fallback_https }}
server {
    server_name _; # This is just an invalid value which will never trigger on a real
hostname.
    server_tokens off;
    access_log off;

{{ template "access_log" (dict "Enable" $globals.config.enable_access_log) }}
    http2 on;

    client_max_body_size 4096M;
```

```
        {{- if $fallback_http }}
    listen {{ $globals.config.external_http_port }}; {{- /* Do not add `default_server` (see comment above). */}}
                {{- if $globals.config.enable_ipv6 }}
    listen [::]:{{ $globals.config.external_http_port }}; {{- /* Do not add `default_server` (see comment above). */}}
                {{- end }}
        {{- end }}
        {{- if $fallback_https }}
    listen {{ $globals.config.external_https_port }} ssl; {{- /* Do not add `default_server` (see comment above). */}}
            {{- if $globals.config.enable_ipv6 }}
    listen [::]:{{ $globals.config.external_https_port }} ssl; {{- /* Do not add `default_server` (see comment above). */}}
            {{- end }}
            {{- if $http3_enabled }}
    http3 on;
    listen {{ $globals.config.external_https_port }} quic reuseport; {{- /* Do not add `default_server` (see comment above). */}}
                {{- if $globals.config.enable_ipv6 }}
    listen [::]:{{ $globals.config.external_https_port }} quic reuseport; {{- /* Do not add `default_server` (see comment above). */}}
                {{- end }}
            {{- end }}
    ssl_session_cache shared:SSL:50m;
    ssl_session_tickets off;
        {{- end }}
        {{- if $globals.config.default_cert_ok }}
    ssl_certificate /etc/nginx/certs/default.crt;
    ssl_certificate_key /etc/nginx/certs/default.key;
        {{- else }}
    # No default certificate found, so reject SSL handshake;
    ssl_reject_handshake on;
        {{- end }}

    #location / {
    #    root /usr/share/nginx/html;
    #    internal;
    #}
```

```
        #{{- if (exists "/usr/share/nginx/html/50x.html") }}
    error_page 400 404 500 502 503 504 /50x.html;
    location /50x.html {
        root /usr/share/nginx/html;
        internal;
    }
       # {{- end }}
    location ^~ / {
        return 503;
    }
}
    {{- end }}
{{- end }}

{{- range $hostname, $vhost := $globals.vhosts }}
    {{- $default_server := when $vhost.default "default_server" "" }}

    {{- range $path, $vpath := $vhost.paths }}
# {{ $hostname }}{{ $path }}
        {{ template "upstream" (dict "globals" $globals "Path" $path "VPath" $vpath) }}
    {{- end }}

    {{- if (eq $vhost.https_method "redirect") }}
server {
    server_name {{ $hostname }};
        {{- if $vhost.server_tokens }}
    server_tokens {{ $vhost.server_tokens }};
        {{- end }}
    {{ template "access_log" (dict "Enable" $globals.config.enable_access_log) }}
    access_log off;
    listen {{ $globals.config.external_http_port }} {{ $default_server }};
        {{- if $globals.config.enable_ipv6 }}
    listen [::]:{{ $globals.config.external_http_port }} {{ $default_server }};
        {{- end }}
    include /etc/nginx/snippets/error.conf

        {{- if (or $vhost.acme_http_challenge_legacy $vhost.acme_http_challenge_enabled)
}}
```

```
    # Do not HTTPS redirect Let's Encrypt ACME challenge
    location ^~ /.well-known/acme-challenge/ {
        auth_basic off;
        auth_request off;
        allow all;
        root /usr/share/nginx/html;
        try_files $uri =404;
        break;
    }
        {{- end }}

        {{- if $vhost.enable_debug_endpoint }}
            {{ template "debug_location" (dict "GlobalConfig" $globals.config "Hostname"
$hostname "VHost" $vhost) }}
        {{- end }}

    location / {
        {{- $redirect_uri := "https://$host$request_uri" }}
        {{- if ne $globals.config.external_https_port "443" }}
            {{- $redirect_uri = printf "https://$host:%s$request_uri"
$globals.config.external_https_port }}
        {{- end}}
        if ($request_method ~ (OPTIONS|POST|PUT|PATCH|DELETE)) {
            return {{ $vhost.non_get_redirect }} {{ $redirect_uri }};
        }
        return 301 {{ $redirect_uri }};
    }
}
    {{- end }}

server {
    {{- if $vhost.is_regexp }}
        {{- if or
            (printf "/etc/nginx/vhost.d/%s" $hostname | exists)
            (printf "/etc/nginx/vhost.d/%s_location" $hostname | exists)
            (printf "/etc/nginx/vhost.d/%s_location_override" $hostname | exists)
            (printf "/etc/nginx/htpasswd/%s" $hostname | exists)
        }}
    # https://github.com/nginx-proxy/nginx-proxy/issues/2529#issuecomment-2437609249
```

```
    # Support for vhost config file(s) named like a regexp ({{ $hostname }}) has been
removed from nginx-proxy.
    # Please name your vhost config file(s) with the sha1 of the regexp instead ({{
$hostname }} -> {{ sha1 $hostname }}) :
    # - /etc/nginx/vhost.d/{{ sha1 $hostname }}
    # - /etc/nginx/vhost.d/{{ sha1 $hostname }}_location
    # - /etc/nginx/vhost.d/{{ sha1 $hostname }}_location_override
    # - /etc/nginx/htpasswd/{{ sha1 $hostname }}
        {{- end }}
    {{- end }}

    server_name {{ $hostname }};
    {{- if $vhost.server_tokens }}
    server_tokens {{ $vhost.server_tokens }};
    {{- end }}
    {{ template "access_log" (dict "Enable" $globals.config.enable_access_log) }}
    {{- if $vhost.http2_enabled }}
    http2 on;
    {{- end }}
    {{- if or (eq $vhost.https_method "nohttps") (eq $vhost.https_method "noredirect") }}
    listen {{ $globals.config.external_http_port }} {{ $default_server }};
        {{- if $globals.config.enable_ipv6 }}
    listen [::]:{{ $globals.config.external_http_port }} {{ $default_server }};
        {{- end }}

        {{- if (and (eq $vhost.https_method "noredirect")
$vhost.acme_http_challenge_enabled) }}
    location /.well-known/acme-challenge/ {
        auth_basic off;
        allow all;
        root /usr/share/nginx/html;
        try_files $uri =404;
        break;
    }
        {{- end }}
    {{- end }}
    {{- if ne $vhost.https_method "nohttps" }}
    listen {{ $globals.config.external_https_port }} ssl {{ $default_server }};
        {{- if $globals.config.enable_ipv6 }}
```

```
    listen [::]:{{ $globals.config.external_https_port }} ssl {{ $default_server }};
        {{- end }}

        {{- if $vhost.http3_enabled }}
    http3 on;
    add_header alt-svc 'h3=":{{ $globals.config.external_https_port }}"; ma=86400;';
    listen {{ $globals.config.external_https_port }} quic {{ $default_server }};
            {{- if $globals.config.enable_ipv6 }}
    listen [::]:{{ $globals.config.external_https_port }} quic {{ $default_server }};
            {{- end }}
        {{- end }}

        {{- if $vhost.cert_ok }}
            {{- template "ssl_policy" (dict "ssl_policy" $vhost.ssl_policy) }}

    ssl_session_timeout 5m;
    ssl_session_cache shared:SSL:50m;
    ssl_session_tickets off;

    ssl_certificate /etc/nginx/certs/{{ (printf "%s.crt" $vhost.cert) }};
    ssl_certificate_key /etc/nginx/certs/{{ (printf "%s.key" $vhost.cert) }};

            {{- if (exists (printf "/etc/nginx/certs/%s.dhparam.pem" $vhost.cert)) }}
    ssl_dhparam {{ printf "/etc/nginx/certs/%s.dhparam.pem" $vhost.cert }};
            {{- end }}

            {{- if (exists (printf "/etc/nginx/certs/%s.chain.pem" $vhost.cert)) }}
    ssl_stapling on;
    ssl_stapling_verify on;
    ssl_trusted_certificate {{ printf "/etc/nginx/certs/%s.chain.pem" $vhost.cert }};
            {{- end }}

            {{- if (not (or (eq $vhost.https_method "noredirect") (eq $vhost.hsts "off")))
}}
    set $sts_header "";
    if ($https) {
        set $sts_header "{{ trim $vhost.hsts }}";
    }
    add_header Strict-Transport-Security $sts_header always;
```

```
            {{- end }}
        {{- else if not $vhost.trust_default_cert | and $globals.config.default_cert_ok }}
    # No certificate found for this vhost, and the default certificate isn't trusted, so
reject SSL handshake.
    ssl_reject_handshake on;
        {{- else }}
    # No certificate for this vhost nor default certificate found, so reject SSL
handshake.
    ssl_reject_handshake on;
        {{- end }}
    {{- end }}

    {{- $vhostFileName :=  $vhost.is_regexp | ternary (sha1 $hostname) $hostname }}

    {{- if (exists (printf "/etc/nginx/vhost.d/%s" $vhostFileName)) }}
    include {{ printf "/etc/nginx/vhost.d/%s" $vhostFileName }};
    {{- else if (exists "/etc/nginx/vhost.d/default") }}
    include /etc/nginx/vhost.d/default;
    {{- end }}

    {{- if $vhost.enable_debug_endpoint }}
        {{ template "debug_location" (dict "GlobalConfig" $globals.config "Hostname"
$hostname "VHost" $vhost) }}
    {{- end }}

    {{- range $path, $vpath := $vhost.paths }}
        {{- template "location" (dict
            "Path" $path
            "Host" $vhostFileName
            "HostIsRegexp" $vhost.is_regexp
            "VhostRoot" $vhost.vhost_root
            "VPath" $vpath
        ) }}
    {{- end }}

    {{- if and (not (contains $vhost.paths "/")) (ne $globals.config.default_root_response
"none")}}
    location / {
        return {{ $globals.config.default_root_response }};
```

```
    }
    {{- end }}
}
{{- end }}
```

Revision #3
Created 2026-02-14 21:04:25 CET by pknw1
Updated 2026-02-14 23:33:03 CET by pknw1