Nginx uWSGI configuration

Note:

<https://nginx.org/en/docs/>

<https://nginx.org/en/docs/beginners_guide.html>

<https://nginx.org/en/docs/http/configuring_https_servers.html>

<https://uwsgi-docs.readthedocs.io/en/latest/Systemd.html>.

## Nginx configuration file syntax

Configuration files are made from line *directives*:

Directive::space\*<name>(space+<parameter>)\*[{directive}\*];

I.e. a directive is a line of text starting with a name followed by space delimited parameters optionally followed by a curly brace list of nested directives. All directives must be terminated by a closing brace or a semicolon. If the semicolon is missing, nginx will fail to start.

Line breaks may occur within braces.

Any line may be terminated with a comment of the form:

… # comment

A *directive* with the optional braces present is also called a *block* or a *context*.

Some directives can not be redefined, so the first definition wins. In others it appears it can be redefined, so the last wins. It seems that a good strategy would to be not to redefined directives.

## Nginx /etc/nginx.conf

See also, <https://www.digitalocean.com/community/tutorials/understanding-the-nginx-configuration-file-structure-and-configuration-contexts>

/etc/nginx.conf first sets some high level variables such as the name of the user it will run as, then includes all /etc/nginx/modules-enabled/\*.conf files, and then it gives three *core* directives/blocks/contexts, namely ‘events’, ‘http’, and ‘mail’. These are also called *main* contexts. As of the time of this writing, the /etc/nginx/modules-enabled directory is empty, and the mail directive is currently commented out. Note that ‘modules-enabled’ differs from ‘sites-enabled’. These are different directories.

The http directive first assigns values to general http parameters such as the mime-types, character encoding, ssl protocols, and the location of the logs. Then at the bottom all /etc/nginx/sites-enabled/\* files and then all /etc/nginx/conf.d/\*.conf files are included into the http directive.

It is not necessarily the case that a variable assigned to earlier in the scanning order may be changed later in the scanning order.

## Nginx /etc/nginx/sites-enabled/\* files.

As noted above, a ‘server block’ is just a configuration line in a configuration file. ‘server blocks’ tell nginx where to find website code.

Configuration files are given the same name as the domain they are configuring. This is required for the SSL scripts. (see “nginx install journal reasoning-technology-server-1/Adding an SSL certificate”).

Configuration files are placed in

/etc/nginx/sites-available/<domain-name>

They are activated by symbolically linking them into

/etc/nginx/sites-enabled

## Nginx control

Do not run nginx directly, as shown in the ‘Beginners Guide’. Instead use systemd:

# systemctl enable|disable|start|restart|status|stop nginx

The enable command causes the service to start a boot time.

## Example conf files

Of course nginx communicates with the internet via http protocol on ip port 80. Each time a request comes into nginx via the internet, it matches the domain name on that request against a the value of the server name directive, which is found in a server block. Once a match is found, the configuration in the corresponding server block is used. (I don’t know what happens if more than one server block has the same value for its server\_name directive, and I hope to never need to have to know ;-)

This file forwards all traffic to reasoning technology on port 80 back to the https site.

http\_redirect.conf:

# send all port 80 requests permanently to the https listeners

server {

listen 80;

listen [::]:80;

server\_name \*.reasoningtechnology.com reasoningtechnology.com;

return 301 https://$host/$request\_uri;

}

With this blanket redirect, there is no need for individual redirect server blocks in other .conf files.

This file configures our current simple html-css-js site:

Reasoning\_technology\_html\_css\_js.conf:

server {

server\_name reasoningtechnology.com www.reasoningtechnology.com;

charset utf-8;

listen [::]:443 ssl ipv6only=on;

listen 443 ssl;

ssl\_certificate /etc/letsencrypt/live/reasoningtechnology.com-0001/fullchain.pem;

ssl\_certificate\_key /etc/letsencrypt/live/reasoningtechnology.com-0001/privkey.pem;

include /etc/letsencrypt/options-ssl-nginx.conf;

ssl\_dhparam /etc/letsencrypt/ssl-dhparams.pem;

access\_log /var/log/nginx/html\_css\_js\_reasoningtechnology.com\_access.log;

error\_log /var/log/nginx/html\_css\_js\_reasoningtechnology.com\_error.log;

root /var/www/html/reasoning-technology\_html-css-js\_release;

index index.html index.htm index.nginx-debian.html;

location / {

try\_files $uri $uri/ =404;

}

}

Certbot turns out pretty easy to run when the site is already up. See the SSL doc.

This file configures uwsgi both for unix uwsgi communication with nginx, and for ip http communication:

Customer\_gateway.conf:

upstream wsgi\_server\_location{

server unix://home/nginx\_customer\_gateway\_mediary/socket;

}

server {

server\_name cg.reasoningtechnology.com;

charset utf-8;

listen [::]:443 ssl;

listen 443 ssl;

ssl\_certificate /etc/letsencrypt/live/reasoningtechnology.com-0001/fullchain.pem;

ssl\_certificate\_key /etc/letsencrypt/live/reasoningtechnology.com-0001/privkey.pem;

include /etc/letsencrypt/options-ssl-nginx.conf;

ssl\_dhparam /etc/letsencrypt/ssl-dhparams.pem;

access\_log /var/log/nginx/cg.reasoningtechnology.com\_access.log;

error\_log /var/log/nginx/cg.reasoningtechnology.com\_error.log;

client\_max\_body\_size 75M;

location / {

uwsgi\_pass wsgi\_server\_location;

include uwsgi\_params;

}

}

server {

server\_name cgd.reasoningtechnology.com;

charset utf-8;

listen 443 ssl; # managed by Certbot

ssl\_certificate /etc/letsencrypt/live/reasoningtechnology.com-0001/fullchain.pem; # managed by Certbot

ssl\_certificate\_key /etc/letsencrypt/live/reasoningtechnology.com-0001/privkey.pem; # managed by Certbot

include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot

ssl\_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

access\_log /var/log/nginx/cgd.reasoningtechnology.com\_access.log;

error\_log /var/log/nginx/cgd.reasoningtechnology.com\_error.log;

location / {

proxy\_pass http://localhost:8000/;

proxy\_http\_version 1.1;

proxy\_set\_header Upgrade $http\_upgrade;

proxy\_set\_header Connection "Upgrade";

}

}