**LOADING BALANCING CONTAINERS:**

STEP-1:NGNIX PROXY

which can dynamically discovery and update its load balance configuration when new containers are loaded. Thankfully this has already been created and is called ngnix proxy

Nginx-proxy accepts HTTP requests and proxies the request to the appropriate container based on the request Hostname. This is transparent to the user with happens without any additional performance overhead.

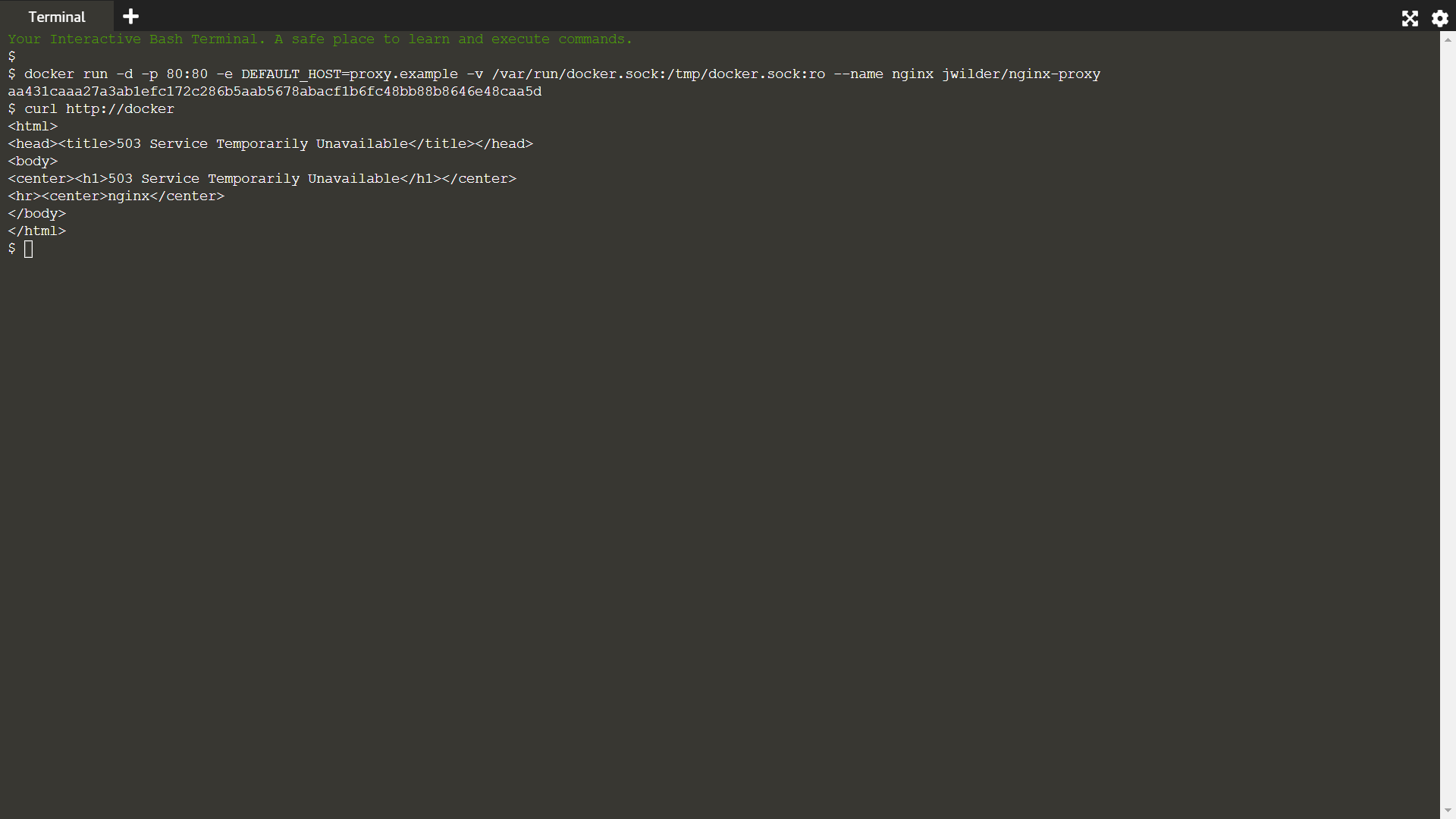
PROPERTIES:

There are three keys properties required to be configured when launching the proxy container.

* The first is binding the container to port 80 on the host using -p 80:80. This ensures all HTTP requests are handled by the proxy.
* Mounting file works in the same way as directories using -v /var/run/docker.sock:/tmp/docker.sock:ro. We specify :ro to restrict access to read-only.
* Finally, we can set an optional \_-e DEFAULTHOST=<domain>. If a request comes in and doesn't make any specified hosts, then this is the container where the request will be handled.
* Use the command below to launch nginx-proxy.
* docker run -d -p 80:80 -e DEFAULT\_HOST=proxy.example -v /var/run/docker.sock:/tmp/docker.sock:ro --name nginx jwilder/nginx-proxy
* Because we're using a DEFAULT\_HOST, any requests which come in will be directed to the container that has been assigned the HOST proxy.example.

REQUEST:

* You can make a request to the web server using curl http://docker. As we have no containers, it will return a 503 error.



STEP-2:SINGLE HOST

Nginx-proxy is now listening to events which Docker raises on start / stop. A sample website called katacoda/docker-http-server has been created which returns the machine name its running on. This allows us to test that our proxy is working as expected. Internally its a PHP and Apache2 application listening on port 80.

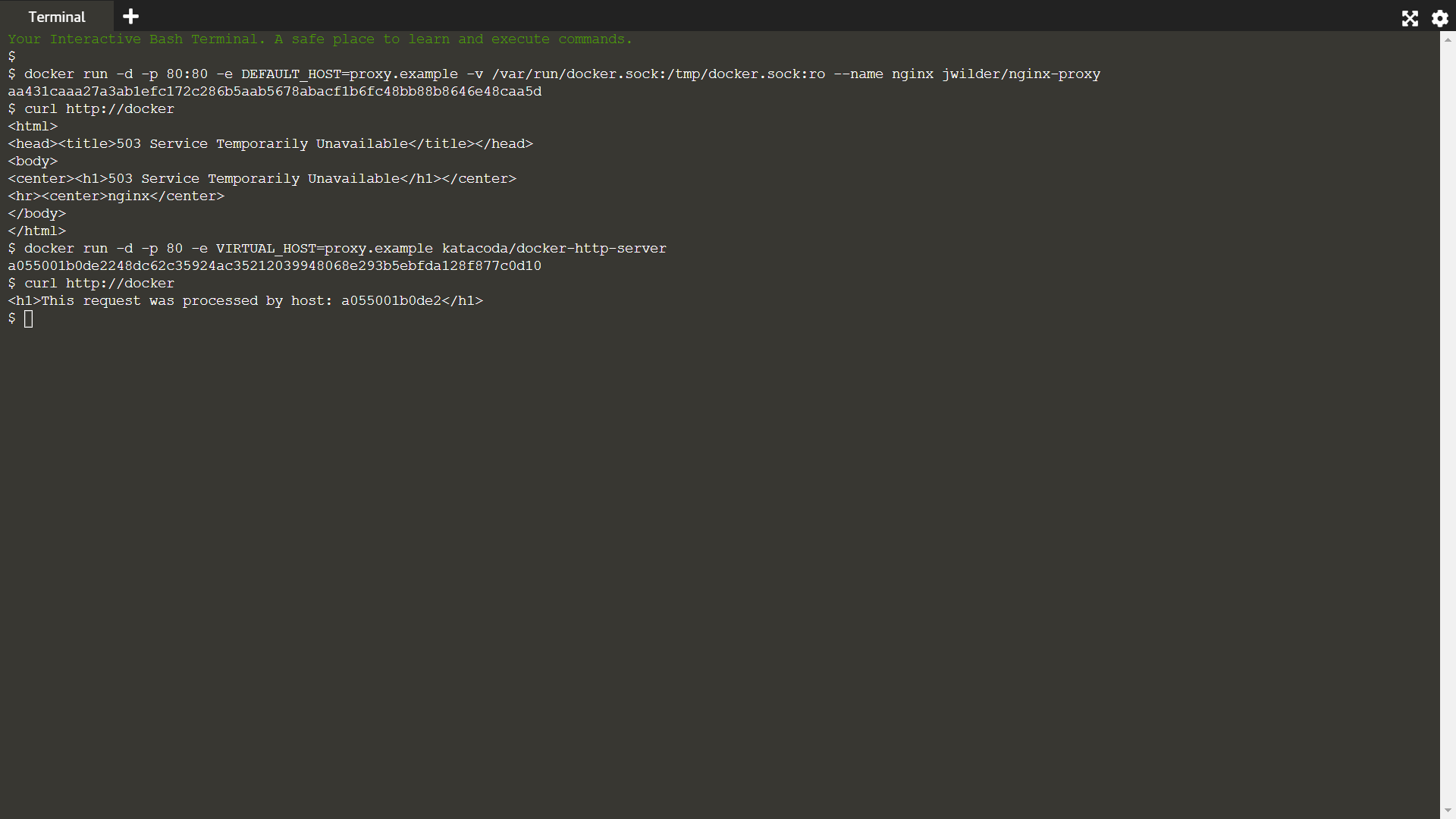
SINGLE CONTAINER:

In this scenario we'll set our HOST to match our DEFAULT\_HOST so it will accept all requests.

docker run -d -p 80 -e VIRTUAL\_HOST=proxy.example katacoda/docker-http-server

TESTING:

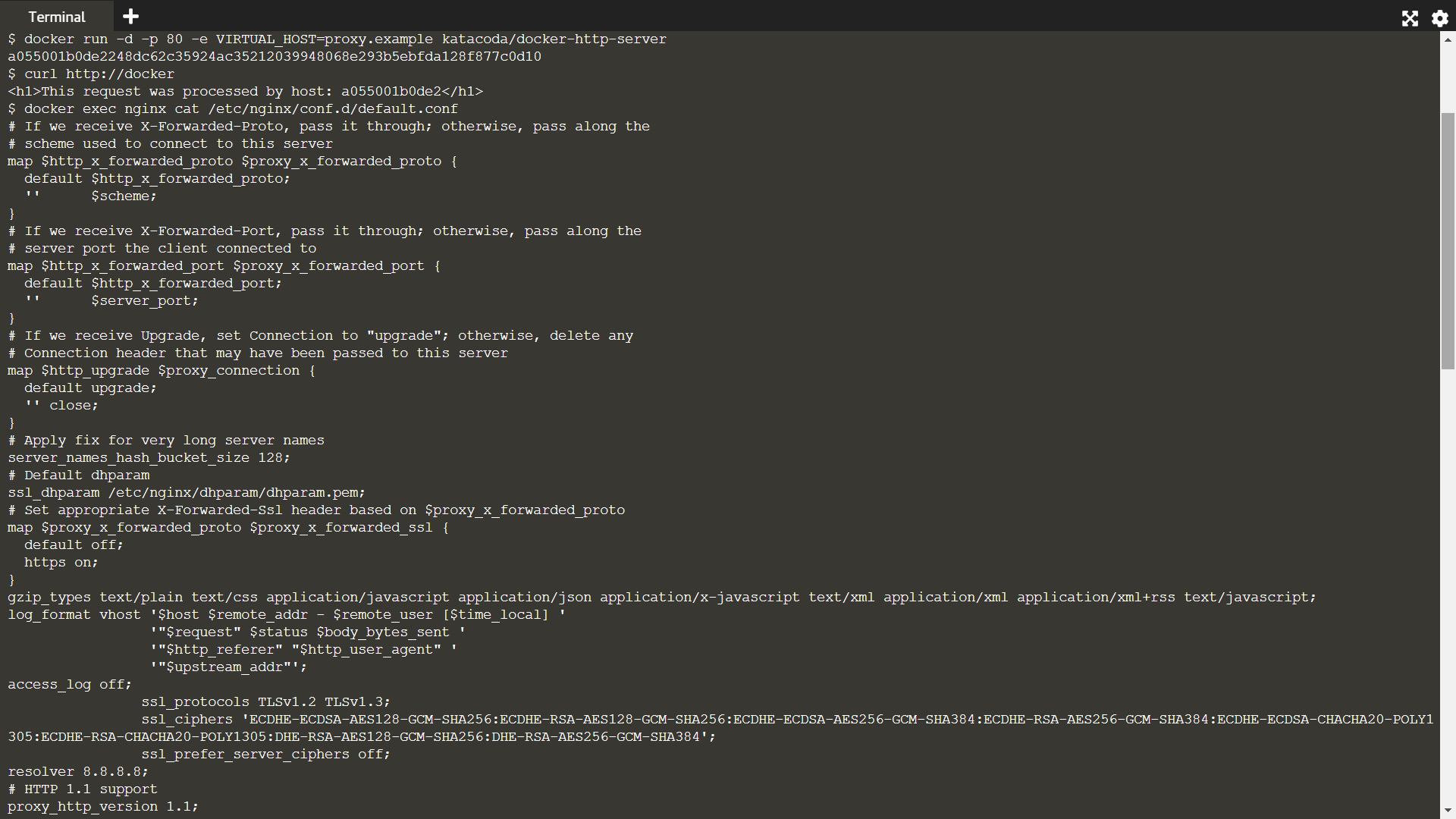
Sometimes it takes a few seconds for NGINX to reload but if we execute a request to our proxy using curl http://docker then the request will be handled by our container

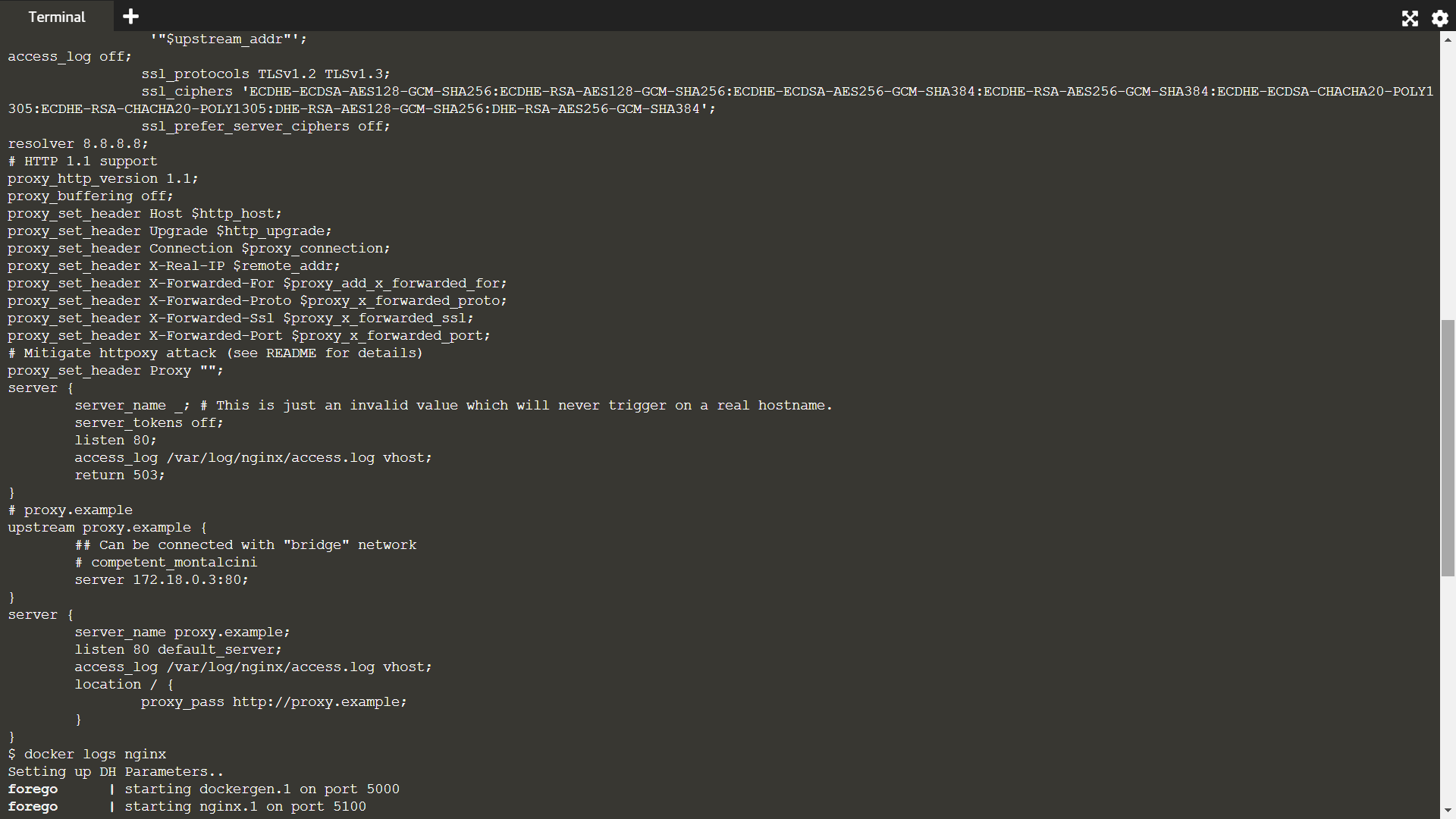


STEP-4:Genersted NGNIX Configuration:

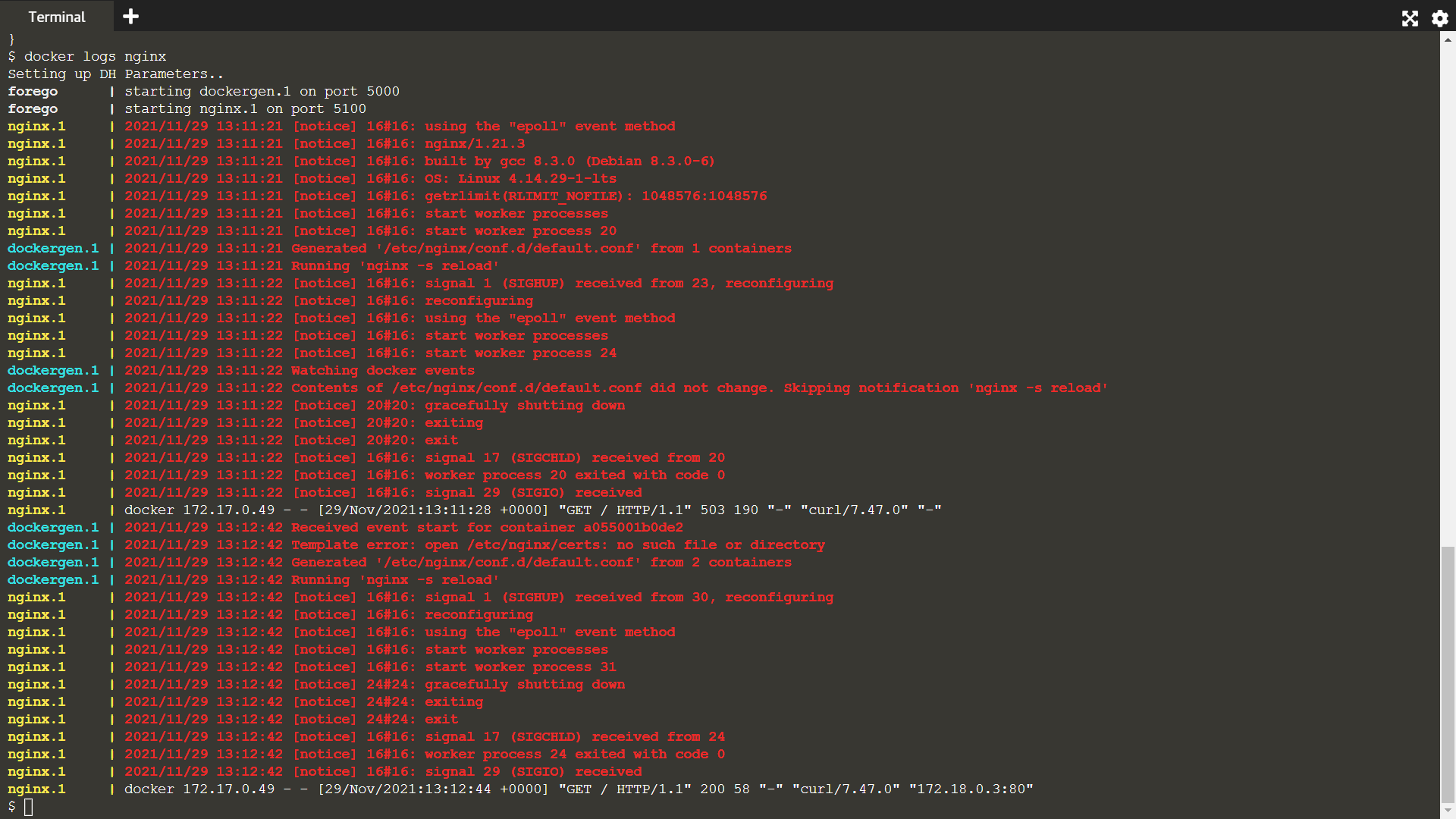
While nginx-proxy automatically creates and configures NGINX for us, if you're interested in what the final configuration looks like then you can output the complete config file with docker exec as shown below.

docker exec nginx cat /etc/nginx/conf.d/default.conf





Additional information about when it reloads configuration can be found in the logs using docker logs nginx



.