

To map a domain name to IP address of instance and host django and node application using same server with nginx.

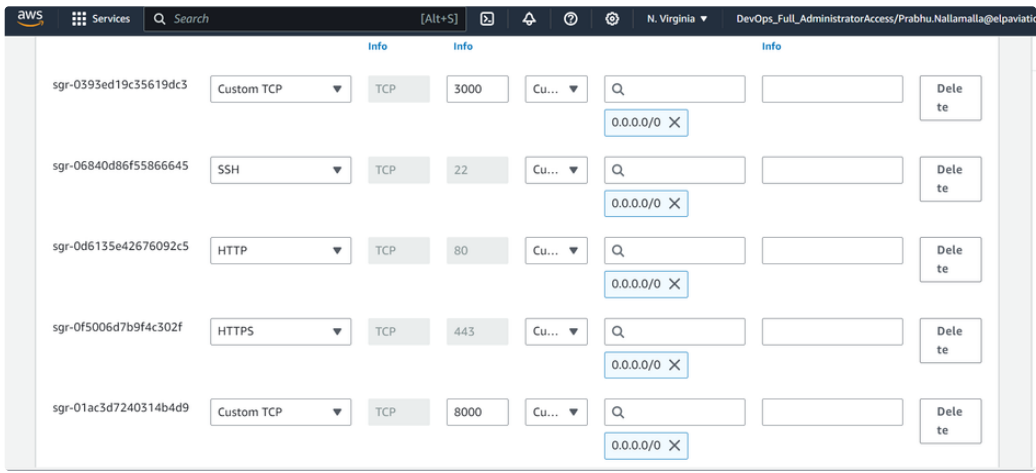
To map a domain name to IP address of instance and host django and node application using same server with nginx:

- Create a DNS record of A which is used to map ip address into domain name. So that we can use domain name in the browser instead of IP address.
- From the following i have a domain hemanth.tech, i have added an A record with do. Now it will be do.hemanth.tech.

Type ▲	Name ▲	Content	Proxy status	TTL	Actions
A	do	54.196.67.36	 DNS only	Auto	Edit ▶

To host Django Application using gunicorn & nginx in Production:

- Make sure that we have enabled the inbound rules of ssh, http, https, 8000, 3000.



1. Install Python and Nginx:

```
1 sudo su
2 sudo apt update
3 sudo apt upgrade
4 sudo apt install python3-pip python3-dev nginx
```

- python3-pip: This is the Python package installer for Python 3. The [pip tool is used to install Python packages from the Python Package Index \(PyPI\)](#). python3-pip is the package that provides the pip tool for Python 3.
- python3-dev: This package contains [files needed for building Python extensions. It includes header files and a static library for Python, which are necessary when building or installing certain Python packages that require compilation.](#)
- nginx: This is a high-performance web server that is widely used for serving static content, reverse proxying, load balancing, and more. In this context, nginx is one of the packages being installed using the apt package manager.

2. Creating a Python Virtual Environment:

```
1 sudo pip3 install virtualenv
```

- This will install a **virtual environment package in python**. Let's create a project directory to host our Django application and create a virtual environment inside that directory.
- **virtualenv env creates the virtual environment from the virtualenv package we have installed.**

```

root@ip-172-31-37-247:/home/ubuntu# mkdir ~/taskdir
root@ip-172-31-37-247:/home/ubuntu# cd ~/taskdir
root@ip-172-31-37-247:~/taskdir# virtualenv env
created virtual environment CPython3.10.12.final.0-64 in 1033ms
  creator CPython3Posix(dest=/root/taskdir/env, clear=False, no_vcs_ignore=False
, global=False)
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle
, via=copy, app_data_dir=/root/.local/share/virtualenv)
    added seed packages: pip==23.3.1, setuptools==69.0.2, wheel==0.42.0
  activators BashActivator,CShellActivator,FishActivator,NushellActivator,PowerS
hellActivator,PythonActivator
root@ip-172-31-37-247:~/taskdir# source env/bin/activate
(env) root@ip-172-31-37-247:~/taskdir# pip install django gunicorn
Collecting django
  Downloading Django-5.0-py3-none-any.whl.metadata (4.1 kB)
Collecting gunicorn
  Downloading gunicorn-21.2.0-py3-none-any.whl.metadata (4.1 kB)
Collecting asgiref>=3.7.0 (from django)
  Downloading asgiref-3.7.2-py3-none-any.whl.metadata (9.2 kB)
Collecting sqlparse>=0.3.1 (from django)
  Downloading sqlparse-0.4.4-py3-none-any.whl (41 kB)
----- 41.2/41.2 kB 776.6 kB/s eta 0:00:00
Collecting packaging (from gunicorn)
  Downloading packaging-23.2-py3-none-any.whl.metadata (3.2 kB)
Collecting typing_extensions>=4 (from asgiref>=3.7.0->django)
  Downloading typing_extensions-4.9.0-py3-none-any.whl.metadata (3.0 kB)
Downloaded Django-5.0-py3-none-any.whl (8.1 MB)
----- 8.1/8.1 MB 42.5 MB/s eta 0:00:00
Downloaded gunicorn-21.2.0-py3-none-any.whl (80 kB)
----- 80.2/80.2 kB 8.4 MB/s eta 0:00:00
Downloaded asgiref-3.7.2-py3-none-any.whl (24 kB)
Downloaded packaging-23.2-py3-none-any.whl (53 kB)
----- 53.0/53.0 kB 5.6 MB/s eta 0:00:00
Downloaded typing_extensions-4.9.0-py3-none-any.whl (32 kB)
Installing collected packages: typing-extensions, sqlparse, packaging, gunicorn,
asgiref, django

```

```

1 pwd
2 mkdir ~/taskdir
3 cd ~/taskdir
4 virtualenv env
5 source env/bin/activate

```

The command `cd ~/taskdir` is used to change the current working directory to a directory named "taskdir" located within the user's home directory.

3. Installing Django and gunicorn:

- This installs Django and gunicorn in our virtual environment.
- **Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.**

```

1 pip install django gunicorn

```

- **Gunicorn is a WSGI compliant web server for Python Applications that receive requests sent to the Web Server from a Client and forwards them onto the Python applications or Web Frameworks (such as Flask or Django) in order to run the appropriate application code for the request.**

4. Setting up our Django project:

- This command **creates a project named taskutils in ~/taskdir, and creates manage.py file.**
- `django-admin` command is used as command line utility for managing django projects. It provides various common tasks such as **creating projects, running development servers and creating db tables.**
- `startproject` is the sub command to create the project.

```

1 django-admin startproject taskutils ~/taskdir

```

```

django-admin startproject taskutils ~/taskdir
(env) root@ip-172-31-37-247:~/taskdir# django-admin startproject taskutils ~/taskdir
(env) root@ip-172-31-37-247:~/taskdir# ~/taskutils manage.py makemigrations
bash: /root/taskutils: No such file or directory
(env) root@ip-172-31-37-247:~/taskdir# ~/taskdir manage.py makemigrations
bash: /root/taskdir: Is a directory
(env) root@ip-172-31-37-247:~/taskdir# ~/taskdir/manage.py makemigrations
No changes detected
(env) root@ip-172-31-37-247:~/taskdir# ~/taskdir/manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK

```

Add your IP address or domain to the ALLOWED_HOSTS variable in settings.py.

```

1 ~/taskdir/manage.py makemigrations
2 ~/taskdir/manage.py migrate
3 sudo ufw allow 8000
4 ~/taskdir/manage.py runserver
5 ~/taskdir/manage.py runserver 0.0.0.0:8000

```

- **manage.py makemigrations:** in django it creates the files that are need to migrate, This command is used to create new database migration files based on the changes you've made to your models.

When you make changes to your Django models (e.g., adding a new model, adding a field to an existing model, etc.), **Django doesn't automatically update the database schema. Instead, it generates a migration file that describes the changes to be made to the database.**

- This command analyzes the current state of your models and compares it to the previous state recorded in the previous migrations. It then creates new migration files in the `migrations` directory of each app, specifying the changes needed to bring the database schema up to date.
- **manage.py migrate:** This command is used to apply the changes specified in the migration files to the actual database. It synchronizes the database schema with the current state of your models. Running migrate looks at the migrations directory of each app and applies any pending migrations to the database. It updates the schema and creates or modifies database tables as necessary.

Note that you should run makemigrations whenever you make changes to your models, and then run migrate to apply those changes to the database. These commands ensure that your database schema stays in sync with your Django models.

- `python manage.py runserver` - used to simply start a development server, here the host is 127.0.0.1:8000, we need to change it to 0.0.0.0:8000
- It indicating that you are not allowed to access the django application without adding your IP address to allowed hosts.

```

root@ip-172-31-37-247: ~/taskdir
"""
Django settings for taskutils project.

Generated by 'django-admin startproject' using Django 5.0.

For more information on this file, see
https://docs.djangoproject.com/en/5.0/topics/settings/

Or the full list of settings and their values, see
https://docs.djangoproject.com/en/5.0/ref/settings/
"""

from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/5.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-ic&#+2ka_lvdph%jt4+*-*_*qi&qkd@$&b_&y1cj^oeq$1a'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = ['do.hemanth.tech']

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]

```

```

1 vim taskutils/settings.py
2 add do.hemanth.tech to allowed hosts
3 ~/taskdir/manage.py runserver 0.0.0.0:8000

```

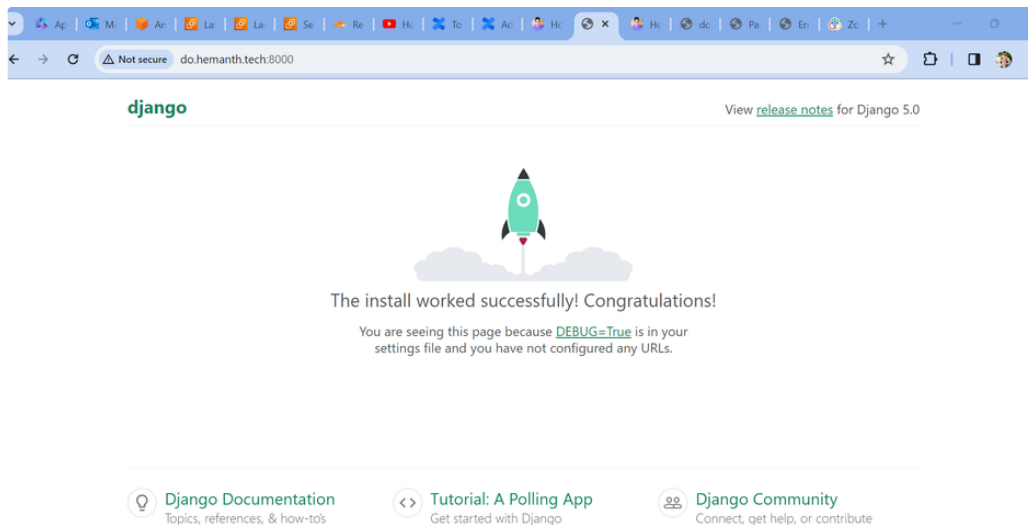
```

Applying sessions.0001_initial... OK
(env) root@ip-172-31-37-247:~/taskdir# sudo ufw allow 8000
Rules updated
Rules updated (v6)
(env) root@ip-172-31-37-247:~/taskdir# ~/taskdir/manage.py runserver 0.0.0.0:8000

Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
December 14, 2023 - 18:58:02
Django version 5.0, using settings 'taskutils.settings'
Starting development server at http://0.0.0.0:8000/
Quit the server with CONTROL-C.

```



Configuring gunicorn:

Lets test gunicorn's ability to serve our application. And reload the application page it will be same.

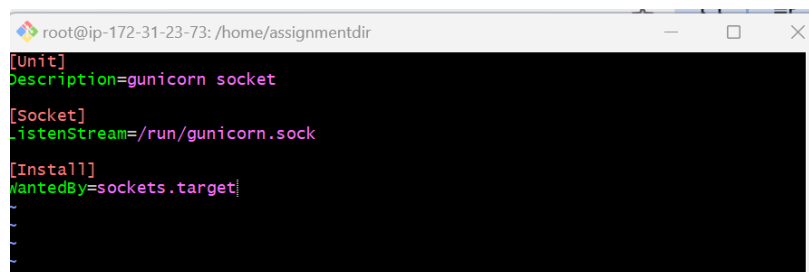
```
(env) root@ip-172-31-23-73:/home/assignmentdir# gunicorn --bind 0.0.0.0:8000 taskutils.wsgi
[2023-12-08 17:50:39 +0000] [3592] [INFO] Starting gunicorn 21.2.0
[2023-12-08 17:50:39 +0000] [3592] [INFO] Listening at: http://0.0.0.0:8000 (3592)
[2023-12-08 17:50:39 +0000] [3592] [INFO] Using worker: sync
[2023-12-08 17:50:39 +0000] [3593] [INFO] Booting worker with pid: 3593
[2023-12-08 17:50:55 +0000] [3592] [INFO] Handling signal: winch
```

- We want our app to automatically start, and serve as a service.
- Deactivate the virtualenvironment by executing the command below. Let's create a system socket file for gunicorn now. Next, we will create a service file for gunicorn.
- Deactivate the virtual environment after running Gunicorn because Gunicorn is going to be managed as a system service (systemd).
System services typically run in the system's default environment, not within a virtual environment. So, deactivating the virtual environment is a good practice to ensure consistency and avoid potential issues.
- After deactivating the virtual environment, you proceed to configure Gunicorn as a systemd service and set up Nginx as a reverse proxy. These steps involve system-level configurations, and it's standard to perform them outside the virtual environment.
- This file defines a systemd socket unit for Gunicorn. **Sockets are a way for systemd to listen on a network socket and activate a service when incoming traffic is detected.**

Purpose:

- It sets up a socket that listens for incoming connections on a specified address and port.
- When a connection is established, it activates the corresponding gunicorn.service unit.
- Lets now start and enable the gunicorn socket.
- This file defines a systemd service unit for Gunicorn. Service units are used to specify how a service should be started, stopped, and managed by systemd.
- It sets up the configuration for running Gunicorn as a service.
- Specifies the working directory, user, group, and other parameters.

```
1 gunicorn --bind 0.0.0.0:8000 taskutils.wsgi
2 deactivate
3 sudo vim /etc/systemd/system/gunicorn.socket
4
5 [Unit]
6 Description=gunicorn socket
7
8 [Socket]
9 ListenStream=/run/gunicorn.sock
10
11 [Install]
12 WantedBy=sockets.target
```



```
root@ip-172-31-23-73: /home/assignmentdir
[Unit]
Description=gunicorn socket

[Socket]
ListenStream=/run/gunicorn.sock

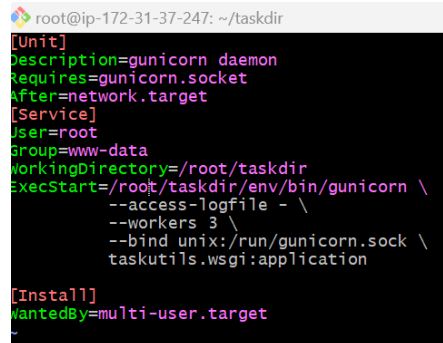
[Install]
WantedBy=sockets.target
```

```
1 sudo vim /etc/systemd/system/gunicorn.service
2 [Unit]
3 Description=gunicorn daemon
4 Requires=gunicorn.socket
```

```

5 After=network.target
6 [Service]
7 User=root
8 Group=www-data
9 WorkingDirectory=/root/taskdir
10 ExecStart=/root/taskdir/env/bin/gunicorn \
11     --access-logfile - \
12     --workers 3 \
13     --bind unix:/run/gunicorn.sock \
14     taskutils.wsgi:application
15
16 [Install]
17 WantedBy=multi-user.target

```



```

root@ip-172-31-37-247: ~/taskdir
[Unit]
Description=gunicorn daemon
Requires=gunicorn.socket
After=network.target
[Service]
User=root
Group=www-data
WorkingDirectory=/root/taskdir
ExecStart=/root/taskdir/env/bin/gunicorn \
    --access-logfile - \
    --workers 3 \
    --bind unix:/run/gunicorn.sock \
    taskutils.wsgi:application
[Install]
WantedBy=multi-user.target

```

```

1 sudo systemctl start gunicorn.socket
2 sudo systemctl enable gunicorn.socket

```

To deploy a node.js application in production:

- Node.js is an open-source, cross-platform JavaScript runtime environment that allows developers to execute JavaScript code server-side. Traditionally, JavaScript was mainly associated with web browsers and client-side development. However, Node.js extends JavaScript's capabilities to the server, enabling the development of scalable and efficient server-side applications.
- Install Node.js and npm.
- npm is the default package manager for Node.js. It is a command-line tool that helps developers manage project dependencies, install libraries, and share code with others.
- Both npm and Yarn serve the same purpose: they are tools for managing and installing dependencies in a Node.js project.

```

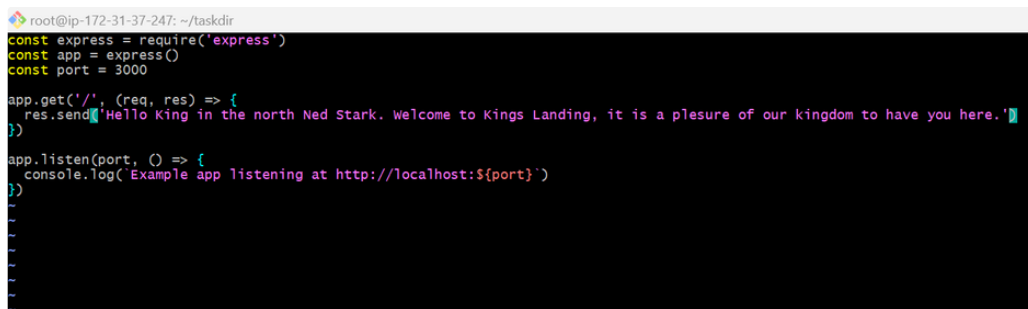
1 sudo apt-get install -y nodejs
2 apt install npm
3 node --version && npm --version

```

- Create a sample node.js file using the following command.
- **const express = require('express')** - This line imports the Express.js framework, which simplifies the process of creating web servers and handling HTTP requests in Node.js.
- **const app = express()** - This line creates an instance of the Express application. The `app` object is used to configure routes and define how the server should respond to different HTTP requests.
- **app.get('/', (req, res) => {
 res.send('Hello King in the north Ned Stark. Welcome to Kings Landing, it is a pleasure for our kingdom to have you here.')
})** – This code defines a route for the root URL ("/). When a user makes a GET request to the root URL, the server responds with the specified message using the `res.send()` method.
- **const port = 3000**
- **app.listen(port, () => {
 console.log(Example app listening at <http://localhost:\${port}>)**

`})` – The script specifies that the server will listen on port 3000. The `app.listen()` method starts the server, and the callback function logs a message to the console indicating that the server is running and accessible at `http://localhost:3000/`.

```
1 sudo vi nodeapp.js
2 const express = require('express')
3 const app = express()
4 const port = 3000
5
6 app.get('/', (req, res) => {
7   res.send('Hello King in the north Ned Stark. Welcome to Kings Landing, it is a plesure of our kingdom to have
8 })
9
10 app.listen(port, () => {
11   console.log(`Example app listening at http://localhost:${port}`)
12 })
13
```



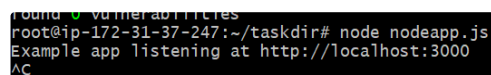
```
root@ip-172-31-37-247: ~/taskdir
const express = require('express')
const app = express()
const port = 3000

app.get('/', (req, res) => {
  res.send('Hello King in the north Ned Stark. Welcome to Kings Landing, it is a plesure of our kingdom to have you here.')
})

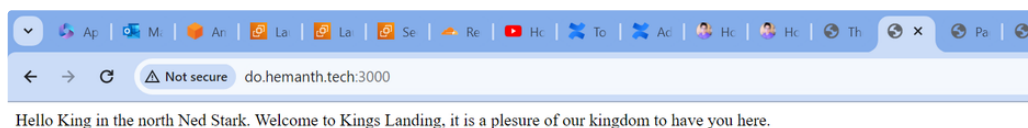
app.listen(port, () => {
  console.log(`Example app listening at http://localhost:${port}`)
})
```

- First we need to install express which is a frame work of node.js to run the application.
- **Express.js is a minimal and flexible web application framework for Node.js. It provides a robust set of features for building web and mobile applications.**
- Key Features are Middleware support, routing, templating engines, and a vibrant ecosystem of extensions.
- The command is used to execute/run a Node.js script named "nodeapp.js" using the Node.js runtime.
- We should now be able to see the webpage with a content when you visit `do.hemanth.tech:3000`.

```
1 npm install express
2 node nodeapp.js
```



```
root@ip-172-31-37-247:~/taskdir# node nodeapp.js
Example app listening at http://localhost:3000
^C
```



do.hemanth.tech:3000

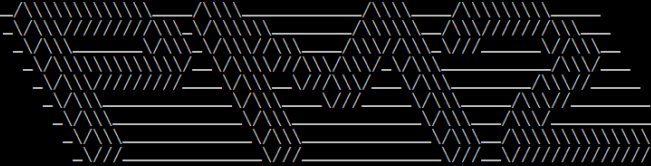
Hello King in the north Ned Stark. Welcome to Kings Landing, it is a plesure of our kingdom to have you here.

- Let's install and use pm2 as a process manager. Install pm2 using the commands below:
- The command `sudo npm i pm2 -g` is used to install the pm2 package globally using the Node Package Manager (npm).
- **PM2 is the name of the package being installed. pm2 is a process manager for Node.js applications. It allows you to manage and deploy Node.js applications, providing features such as process clustering, load balancing, and automatic restarts.**

```
1 sudo npm i pm2 -g
2 pm2 start nodeapp.js
```

```
root@ip-172-31-37-247:~/taskdir# pm2 start nodeapp.js
```

```
-----
```



Runtime Edition

PM2 is a Production Process Manager for Node.js applications
with a built-in Load Balancer.

Start and Daemonize any application:
`$ pm2 start app.js`

Load Balance 4 instances of api.js:
`$ pm2 start api.js -i 4`

Monitor in production:
`$ pm2 monitor`

Make pm2 auto-boot at server restart:
`$ pm2 startup`

To go further checkout:
<http://pm2.io/>

```
-----
```

```
[PM2] Spawning PM2 daemon with pm2_home=/root/.pm2
[PM2] PM2 Successfully daemonized
[PM2] Starting /root/taskdir/nodeapp.js in fork_mode (1 instance)
[PM2] Done.
```

id	name	mode	u	status	cpu	memory
0	nodeapp	Fork	0	online	0%	19.4mb

•

```
1 sudo nano /etc/nginx/sites-available/host /etc/nginx/si
2 server {
3     listen 80;
4     server_name do.hemanth.tech;
5
6     location /django/ {
7         proxy_pass http://unix:/run/gunicorn.sock;
8         include proxy_params;
9     }
10
11     location /static_django/ {
12         alias /root/taskdir/static/;
13     }
14
15     location /node/ {
16         proxy_pass http://localhost:3000;
17         proxy_http_version 1.1;
18         proxy_set_header Upgrade $http_upgrade;
19         proxy_set_header Connection 'upgrade';
20         proxy_set_header Host $host;
21         proxy_cache_bypass $http_upgrade;
22     }
23 }
```