**Как запустить сайт с помощью Gunicorn: Django, Flask, что угодно**

Рано или поздно вы приходите к тому, что надо запустить проект на сервере. Этот туториал покажет как превратить любой Python-скрипт в сайт. Также будут примеры запуска проектов, сделанных на Django и Flask.

Цель туториала — сделать сайт, который узнает ваш IP-адрес:



**Что надо знать**

Конечно же, без определённых знаний и навыков запустить сайт на сервере не получится. Вот небольшой список того, что вы должны знать и уметь:

* [Знаете что такое веб-сервер](https://dvmn.org/encyclopedia/web-server/web-server/)
* Знаете что такое HTTP запрос/ответ
* [Умеете работать с вёрсткой](https://dvmn.org/encyclopedia/frontend/how-to-write-html/)
* [Умеете работать с Systemd](https://dvmn.org/encyclopedia/deploy/systemd/)
* Умеете подключаться к серверу по ssh
* [Запускаете команды из консоли](https://dvmn.org/encyclopedia/tutorial/tutorial_cli/)

**Системные требования**

* ОС Ubuntu 20.04 LTS
* Python3

**1. Запустите веб-сервер**

Без веб-сервера не работает ни один сайт в интернете. Каждый раз, когда вы открываете страницу сайта ваш браузер связывается с сервером и запрашивает необходимые для работы ресурсы: HTML, картинки, шрифты и прочее. Веб-сервер отвечает за то, чтобы найти эти файлы и переслать их браузеру.

Давайте начнём с самого простого сайта, который просто показывает текст Here be dragons. Сайт будет реализован в виде Python-скрипта. Чтобы превратить скрипт в сайт нужен веб-сервер. А чтобы веб-сервер понял, как правильно запустить скрипт, существует стандарт [WSGI](https://www.python.org/dev/peps/pep-0333/).

WSGI — это стандарт взаимодействия между Python-скриптом и веб-сервером

Стандарт WSGI требует, чтобы в скрипте была особенная функция. Она принимает на вход два аргумента - словарь с данными HTTP-запроса и обработчик запроса. Когда веб-сервер снова получит от браузера HTTP-запрос, то он найдёт эту функцию и запустит.

Стандарт WSGI поддерживает много разных веб-серверов. Один из них [Gunicorn](https://gunicorn.org/" \t "_blank). Он относительно быстр, легко настраивается и работает со многими веб-фреймворками. Написан Gunicorn на Python и поставляется в виде обычной библиотеки.

Gunicorn — это веб-сервер с поддержкой стандарта WSGI

Что-ж, пора начинать. Сперва установите Gunicorn:

**#** pip install gunicorn

Теперь создайте ту самую функцию для обработки HTTP-запросов. Позже её запустит Gunicorn. Создайте файл server.py и положите в него код:

**def** process\_http\_request(environ, start\_response):

status = '200 OK'

response\_headers = [

('Content-type', 'text/plain; charset=utf-8'),

]

start\_response(status, response\_headers)

text = 'Here be dragons'.encode('utf-8')

**return** [text]

Веб-сервер Gunicorn сам запустит функцию process\_http\_request. Согласно стандарту WSGI она обязана принимать два аргумента environ и start\_response.

environ — это словарь с данными об HTTP запросе. Запрос от браузер сначала прилетает к веб-серверу Gunicorn, тот упаковывает полезную информацию в словарь environ и передаёт его в функцию process\_http\_request.

start\_response — это функция, которую даёт Gunicorn. Она нужна, чтобы отправить в браузер первую самую важную часть – статус HTTP ответа и заголовки. В качестве аргументов она принимает статус 200 OK и заголовок Content-type: text/plain; charset=utf-8;. Gunicorn упакует их и отправит браузеру.

Функция process\_http\_request возвращает список строк с текстом Here be dragons. Это то, что браузер получит в теле HTTP ответа. Перед отправкой программа кодирует текст в utf-8, так как в интернете можно передавать только закодированный текст.

Протокол HTTP позволяет отвечать браузеру не сразу, а порциями. Случается, что веб-сервер раздаёт не только мелкие фрагменты тексты, но и огромные видеофайлы. За один раз передать такой объём невозможно, поэтому HTTP-ответ разбивается на части. А раз стандарт WSGI обязан быть универсальным, то он всегда требует от функций списка или другого итерируемого объекта с бинарными строками внутри. В данном случае используется список с одним единственным элементом – закодированной строкой 'Here be dragons'.

В скрипте сейчас есть всё необходимое, чтобы увидеть текст Here be dragons в браузере. Осталось запустить веб-сервер Gunicorn. Запустите его командой:

**$** gunicorn -b 82.148.28.32:80 server:process\_http\_request

Ключ -bили --bind означает “привязка” к определённому IP-адресу и порту.

82.148.28.32 — это IP-адрес сервера на котором хотите запустить Gunicorn. Тем же способом можно запустить веб-сервер локально на адресе 127.0.0.1, либо на всех сетевых интерфейсах сразу через 0.0.0.0.

80 — номер порта, на котором по умолчанию работает протокол HTTP . Если привязать Gunicorn к другому порту, то придётся вручную указывать порт в адресной строке браузера: http://82.148.28.32:8080.

server — название вашего скрипта без .py, а process\_http\_request — название функции. Так вы укажете Gunicorn, где искать функцию, которая обработает запрос браузера.

После запуска Gunicorn сообщит в консоль, что он готов и ждёт входящих запросов от браузера:

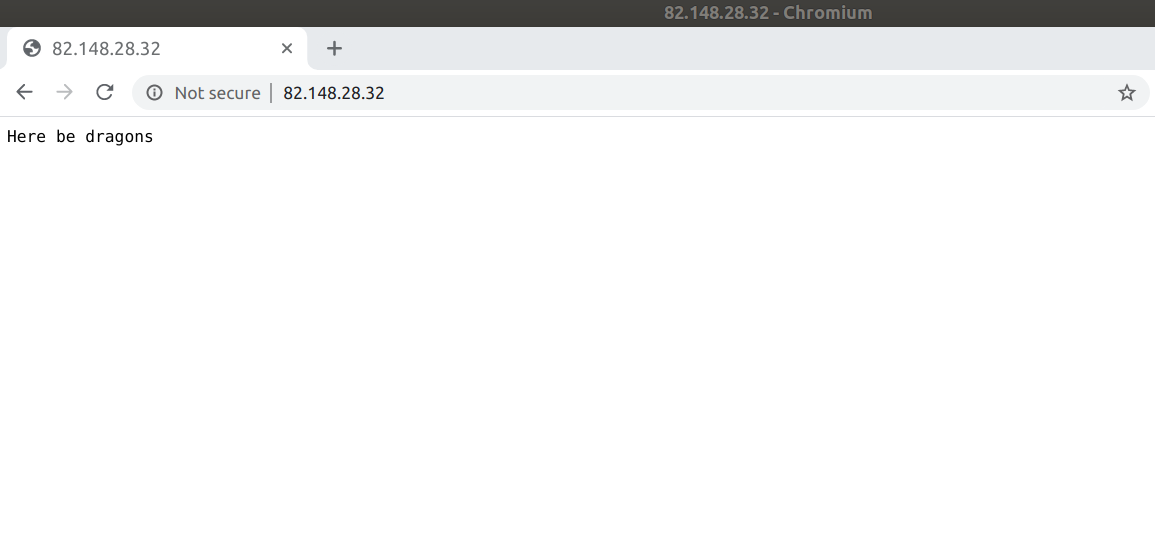
[2020-08-06 11:32:08 +0400] [13234] [INFO] Starting gunicorn 20.0.4

[2020-08-06 11:32:08 +0400] [13234] [INFO] Listening at: http://82.148.28.32:80 (13234)

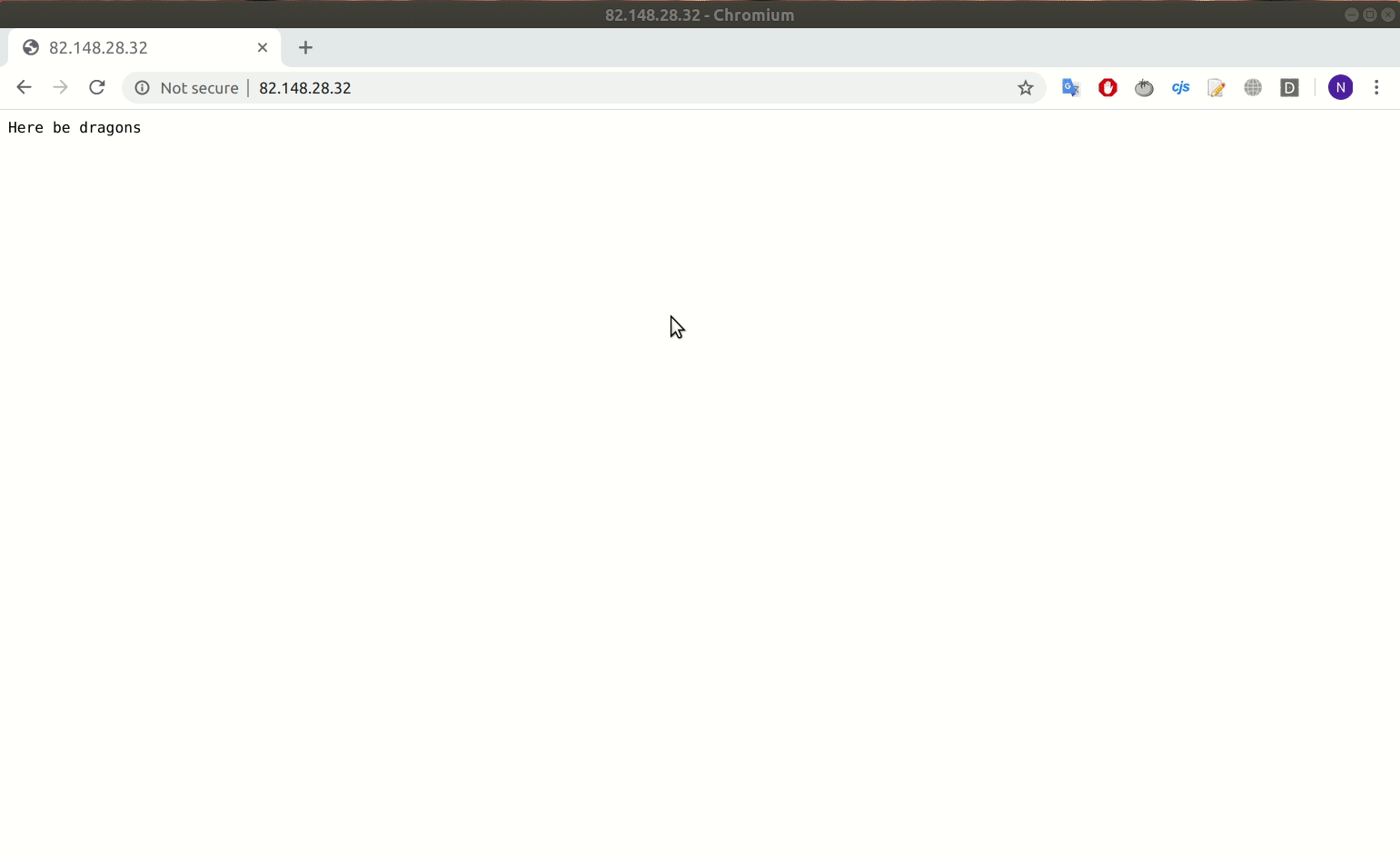
[2020-08-06 11:32:08 +0400] [13234] [INFO] Using worker: sync

[2020-08-06 11:32:08 +0400] [13237] [INFO] Booting worker with pid: 13237

Давайте проверим работу Gunicorn. Для этого откройте сайт и в адресной строке браузера введите IP-адрес сервера http://82.148.28.32. Вы увидите текст Here be dragons:



Скрипт отправляет не только текст, но ещё статус и заголовки. Их можно проверить через инструменты разработчика браузера:



В Status Code вы видите статус 200 OK и заголовок Content-type, в котором обычный текст text/plain.

Отлично, вы теперь умеете запускать веб-сервер Gunicorn!

**2. Сделайте сайт**

У вас есть заготовка сайта *Here be dragons*. Превратим её в сайт, что определяет IP-адрес пользователя.

Добавьте в скрипт server.py HTML разметку:

HTML = """

<!doctype html>

<html lang="ru">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/css/bootstrap.min.css" integrity="sha384-B0vP5xmATw1+K9KRQjQERJvTumQW0nPEzvF6L/Z6nronJ3oUOFUFpCjEUQouq2+l" crossorigin="anonymous">

<title>Узнать IP адрес</title>

</head>

<body class="bg-info">

<div class="container">

<h1 class="text-white text-center mx-auto p-2 mt-4 mb-3">Ваш IP-адрес</h1>

<div class="bg-light card-block mx-auto text-center">

<h1 class="display-3 p-2">0.0.0.0</h1>

</div>

</div>

</body>

</html>

"""

**def** process\_http\_request(environ, start\_response):

status = '200 OK'

response\_headers = [

('Content-type', 'text/html; charset=utf-8'),

]

start\_response(status, response\_headers)

html\_as\_bytes = HTML.encode('utf-8')

**return** [html\_as\_bytes]

Так как вместо Here be dragons функция теперь возвращает HTML, то Content-type изменился на text/html — вы сообщаете браузеру, что это разметка HTML.

Протестируйте обновлённый скрипт. Остановите Gunicorn командой Ctrl-C в консоли и запустите снова. В браузере обновите страницу сайта:



Сайт работает, IP-адрес он показывает не настоящий: 0.0.0.0. Это лишь временная заглушка.

А откуда взять IP-адрес? Функция process\_http\_request принимает всего лишь два аргумента: словарь environ и функцию start\_response. Данные об HTTP запросе environ получены от вашего браузера, значит и IP-адрес стоит искать там.

Содержимое словаря environ можно вывести в консоль и посмотреть что там лежит. Для этого добавьте в вашу функцию одну отладочную строчку кода перед return:

**def** process\_http\_request(environ, start\_response):

...

print(environ)

**return** [html\_as\_bytes]

Перезапустите Gunicorn и обновите страницу в браузере. В консоли вы увидите подобный вывод:

[2020-08-06 10:30:02 +0000] [58594] [INFO] Starting gunicorn 20.0.4

[2020-08-06 10:30:02 +0000] [58594] [INFO] Listening at: http://0.0.0.0:80 (58594)

[2020-08-06 10:30:02 +0000] [58594] [INFO] Using worker: sync

[2020-08-06 10:30:02 +0000] [58596] [INFO] Booting worker with pid: 58596

{'wsgi.errors': <gunicorn.http.wsgi.WSGIErrorsWrapper object at 0x7f08d1b93df0>, 'wsgi.version': (1, 0), 'wsgi.multithread': False, 'wsgi.multiprocess': False, 'wsgi.run\_once': False, 'wsgi.file\_wrapper': <class 'gunicorn.http.wsgi.FileWrapper'>, 'wsgi.input\_terminated': True, 'SERVER\_SOFTWARE': 'gunicorn/20.0.4', 'wsgi.input': <gunicorn.http.body.Body object at 0x7f08d1b93ee0>, 'gunicorn.socket': <socket.socket fd=9, family=AddressFamily.AF\_INET, type=SocketKind.SOCK\_STREAM, proto=0, laddr=('82.148.28.32', 80), raddr=('79.141.160.49', 57897)>, 'REQUEST\_METHOD': 'GET', 'QUERY\_STRING': '', 'RAW\_URI': '/', 'SERVER\_PROTOCOL': 'HTTP/1.1', 'HTTP\_HOST': '82.148.28.32', 'HTTP\_CONNECTION': 'keep-alive', 'HTTP\_CACHE\_CONTROL': 'max-age=0', 'HTTP\_DNT': '1', 'HTTP\_UPGRADE\_INSECURE\_REQUESTS': '1', 'HTTP\_USER\_AGENT': 'Mozilla/5.0 (X11; Linux x86\_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.105 Safari/537.36', 'HTTP\_ACCEPT': 'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,\*/\*;q=0.8,application/signed-exchange;v=b3;q=0.9', 'HTTP\_ACCEPT\_ENCODING': 'gzip, deflate', 'HTTP\_ACCEPT\_LANGUAGE': 'ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7', 'wsgi.url\_scheme': 'http', 'REMOTE\_ADDR': '79.141.160.49', 'REMOTE\_PORT': '57897', 'SERVER\_NAME': '0.0.0.0', 'SERVER\_PORT': '80', 'PATH\_INFO': '/', 'SCRIPT\_NAME': ''}

IP-адрес вашего компьютера находится в значении по ключу REMOTE\_ADDR: 79.141.160.49. Чтобы проверить свой IP можете обратиться к уже готовому сервису в интернете — [2ip.ru](https://2ip.ru/).

Остался последний рывок — отобразить IP-адрес на сайте. Допишите скрипт и перезапустите Gunicorn. Получится что-то подобное:

HTML = """

<!doctype html>

<html lang="ru">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/css/bootstrap.min.css" integrity="sha384-B0vP5xmATw1+K9KRQjQERJvTumQW0nPEzvF6L/Z6nronJ3oUOFUFpCjEUQouq2+l" crossorigin="anonymous">

<title>Узнать IP адрес</title>

</head>

<body class="bg-info">

<div class="container">

<h1 class="text-white text-center mx-auto p-2 mt-4 mb-3">Ваш IP-адрес</h1>

<div class="bg-light card-block mx-auto text-center">

<h1 class="display-3 p-2">{ip\_address}</h1>

</div>

</div>

</body>

</html>

"""

**def** process\_http\_request(environ, start\_response):

status = '200 OK'

response\_headers = [

('Content-type', 'text/html; charset=utf-8'),

]

start\_response(status, response\_headers)

html = HTML.format(ip\_address=environ["REMOTE\_ADDR"])

html\_as\_bytes = html.encode('utf-8')

**return** [html\_as\_bytes]

Ура! Теперь у вас работает полноценный сайт. Сайт по определению IP-адреса:



**3. Демонизируйте Gunicorn**

В норме программы, запущенные внутри сеанса ssh будут остановлены вместе с завершением этого сеанса. Стоит пользователю отключиться от сервера и сайт сразу упадёт из-за остановки Gunicorn.

Для решения таких проблем в Linux существуют *демоны* — это программы, которые работают в фоне. Их запускает не пользователь, а операционная система, благодаря чему демон продолжает работать даже когда пользователь уходит с сервера. Чтобы сайт не падал Gunicorn тоже надо сделать демоном.

Systemd — система, которая управляет демонами. С её помощью вы сделаете Gunicorn демоном, а также сможете отслеживать его работу. С Systemd ваш сайт всегда будет онлайн.

Перейдите в папку /etc/systemd/system и создайте файл getip.service с настройками для будущего демона Gunicorn:

[Unit]

Description**=GetIP site**

[Service]

Type**=simple**

WorkingDirectory**=/root/wsgi**

ExecStart**=gunicorn -b 82.148.28.32:80 server:process\_http\_request**

Restart**=always**

[Install]

WantedBy**=multi-user.target**

Замените путь к каталогу WorkingDirectory=/root/wsgi на тот, где лежит ваш скрипт. Также поменяйте IP адрес в настройке ExecStart.

После добавления нового сервиса перенастройте Systemd и запустите сервис:

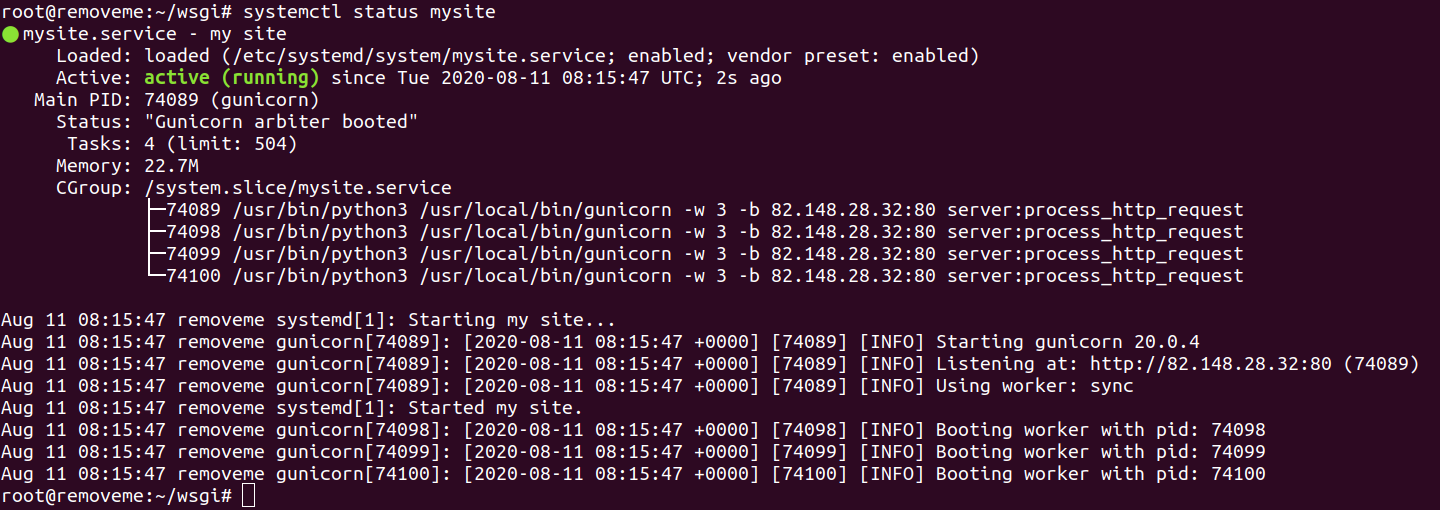
**$** systemctl daemon-reload

**$** systemctl start getip.service

Если при запуске сервиса будет ошибка, то вы не заметите этого. Проверить работу сервиса можно командой:

**#** systemctl status getip

Вот пример вывода статуса сервиса:



Если всё прошло успешно, вы увидите зелёный кружок перед именем сервиса getip.service и зелёный индикатор активности active (running).

Осталось добавить сервис getip в автозапуск при старте сервера:

**#** systemctl enable getip.service

Вы увидите в консоли:

Created symlink /etc/systemd/system/multi-user.target.wants/getip.service → /etc/systemd/system/getip.service.

Теперь ваш сайт будет работать всегда! Даже если вы перезапустите сервер, то Gunicorn включится сам.

«Девман» — авторская методика обучения программированию. Готовим к работе крутых программистов на Python. Станьте программистом, пройдите [курс программирования на Python](https://dvmn.org/).

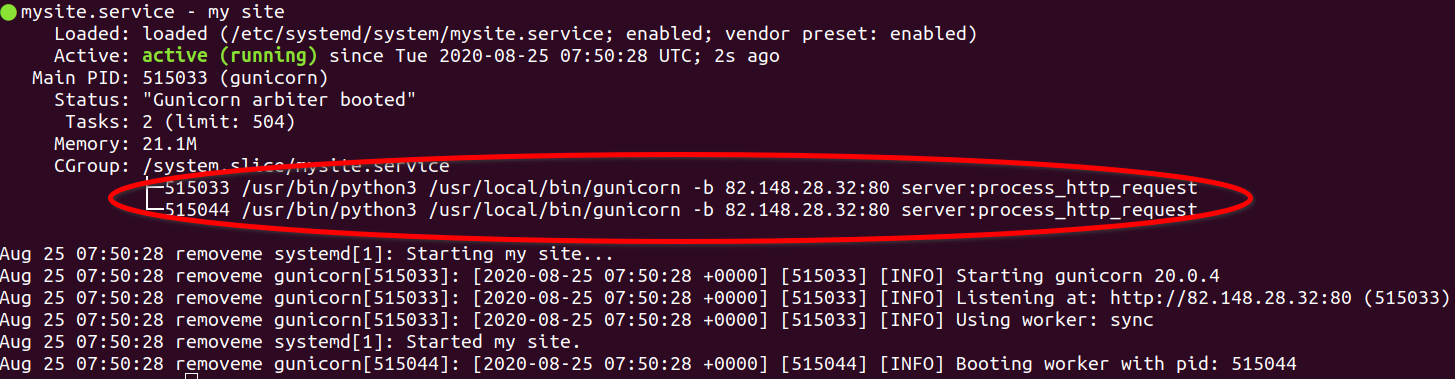
**4. Добавьте воркеров к Gunicorn**

Gunicorn — это серьёзный веб-сервер, рассчитанный на большую нагрузку. Он умеет обрабатывать несколько запросов одновременно благодаря своей архитектуре. У Gunicorn есть главный процесс, который управляет набором рабочих процессов — воркеров. Главный процесс только распределяет запросы от клиентов сайта, а обрабатывают их воркеры.

Проверьте, сколько воркеров сейчас использует Gunicorn. Вдруг он использует сервер не на полную? Количество воркеров можно проверить в статусе демона. Проверьте статус Gunicorn командой:

**#** systemctl status getip

Вы увидите статус Gunicorn:



На скриншоте красным выделена группа процессов. Видно, что запущено 2 процесса: 1 главный и 1 рабочий — воркер.

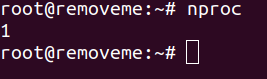
Веб серверу нужен хотя бы один воркер. Но одного воркера мало, чтобы разогнать Gunicorn по максимуму. Это всё из-за того, что воркеру мешают операции ввода/вывода, из за которых Python засыпает и ждёт ответа. Если воркеров будет мало, сервер работает не в полную силу, а если слишком много — тормозит. Число воркеров зависит от ядер процессора. Документация Gunicorn советует придерживаться такой формулы:

N воркеров = Количество ядер x 2 + 1

Количество ядер процессора показывает штатная утилита nproc:

**$** nproc

Вот пример вывода npoc:



Итак, если у процессора 1 ядро, то по формуле получается 3 воркера. Эта опция указывается в настройке демона Gunicorn. Вот новое содержимое файла getip.service:

[Unit]

Description**=GetIP site**

[Service]

Type**=simple**

WorkingDirectory**=/root/wsgi**

ExecStart**=gunicorn -w 3 -b 82.148.28.32:80 server:process\_http\_request**

Restart**=always**

[Install]

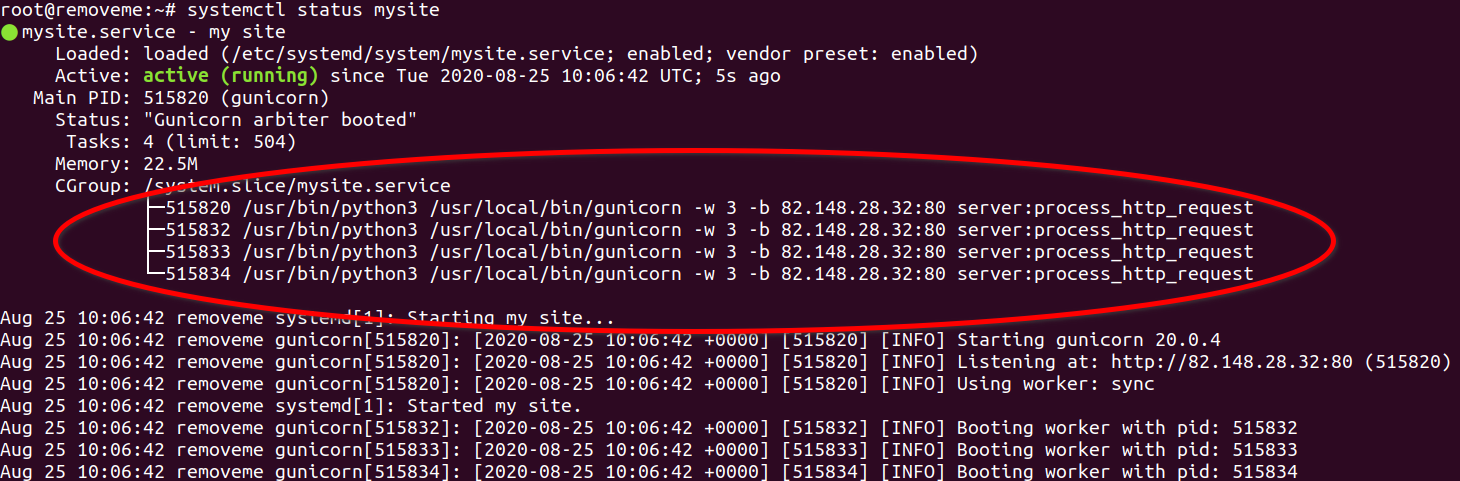
WantedBy**=multi-user.target**

Поменялись только настройки ExecStart. Gunicorn запускается с ещё одним ключом -w или --workers, что как раз и означает количество воркеров.

А теперь проверьте статус Gunicorn:

**#** systemctl status getip

В консоли будет:



Снова обратите внимание на группу процессов. Сейчас запущено 4 процесса: 1 главный и 3 воркера. Теперь Gunicorn будет работать быстрее.

**5. Как запустить Django через Gunicorn**

В Django уже есть встроенный веб-сервер — runserver. Его используют для быстрой отладки во время разработки. Но сами же разработчики Django прямым текстом [пишут о том](https://docs.djangoproject.com/en/2.2/ref/django-admin/#runserver), что использовать в продакшне его не стоит:

DO NOT USE THIS SERVER IN A PRODUCTION SETTING. It has not gone through security audits or performance tests.

Gunicorn быстрее и заботится о безопасности. Его можно использовать в продакшне, и даже ребята из Django советуют это делать.

Django поддерживает работу с WSGI веб-серверами из коробки. Django оборачивает весь проект в одну простую функцию и кладёт её в файлы wsgi.py и asgi.py в папке проекта рядом с settings.py.

Ниже будет пример запуска пустого Django-проекта через Gunicorn. В результате вы увидите в браузере страницу Django “Congratulations!” . Это та самая страница с ракетой.

В репозитории Django уже создала файлы wsgi.py и asgi.py. Который их них запускать? Файл asgi.py создан для асинхронных веб-серверов, а Gunicorn так не умеет. Используйте файл wsgi.py.

Для Django проекта важно из какого каталога будет запущен Gunicorn. Скрипты Python вычисляют пути к другим файлам проекта принимая за точку отсчёта корень проекта. А корнем считается текущий путь – тот каталог, откуда запущены скрипты. Если запустить Gunicorn из другого каталога, то и путь к корню проекта получится неправильный.

Воспользуйтесь утилитой tree, чтобы понять, откуда запускать Gunicorn:

**$** tree

.

├── blog

│ ├── \_\_init\_\_.py

│ ├── asgi.py

│ ├── settings.py

│ ├── urls.py

│ └── wsgi.py

└── manage.py

1 directory, 6 files

Как видно, wsgi.py лежит в папке blog. Gunicorn запускается уровнем выше — оттуда, где лежит manage.py:

**$** gunicorn -b 82.148.28.32:80 blog.wsgi:application

Вот и всё, что требуется чтобы запустить Django-проект через Gunicorn!

**6. Как запустить Flask через Gunicorn**

Фреймворк Flask, как и Django, тоже поддерживает работу с Gunicorn из коробки.

Вот простейший сайт на Flask, который выводит в браузер текст Here be dragons. Содержимое скрипта server.py:

**from** flask **import** Flask, Response

app = Flask(\_\_name\_\_)

**@app**.route("/")

**def** index():

**return** Response("Here be dragons"), 200

**if** \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

Сайт можно запустить в отладочном режиме:

**$** python server.py

Для этого в скрипте есть блок if \_\_name\_\_ == "\_\_main\_\_" со строкой запуска отладочного сервера Flask.

Как и runserver в Django, тот тоже работает в однопоточном режиме и не справится с нагрузкой на сайт. На помощь придёт Gunicorn. Нужна функция соответствующая WSGI-протоколу.

Объект app во Flask-скрипте ведёт себя подобно функции – его можно запустить, передав аргументы:

app(environs, start\_response)

Чтобы запустить сайт Flask через Gunicorn достаточно выполнить команду:

**$** gunicorn -b 82.148.28.32:80 server:app

server — название модуля без .py, app — функция в скрипте.

Как видите, запустить Flask-проект через Gunicorn очень легко.

**Что почитать**

* [**PEP 333** – Python Web Server Gateway Interface](https://lectureswww.readthedocs.io/5.web.server/wsgi.html)
* [Официальный сайт **Gunicorn**](https://gunicorn.org/)