NGINX

Frontend серверы

Первое звено на серверной стороне, которое и начинает обработку запроса. Как правило, это легкие и быстрые веб-сервера, практически не занимающиеся вычислениями.
Программное обеспечение фронтенда принимает запрос, далее, если может, то сразу отвечает на него, иначе проксирует запрос к бэкенду.
На Front End обычно ложатся задачи не требующие никаких бизнес-вычислений.

Задачи Frontend серверов

Отдача статики

Под статикой подразумевается JavaScript / CSS / HTML / Fonts – данные для рендеринга web-страниц.
Так Frontend может отдавать картинки, видео и другой статический объемный контент.

```
<html lang="en">
<head>
   <title>Main Page</title>
</head>
<body>
   <h1 class="center">Hello, world</h1>
</body>
</html>
```

Задачи Frontend серверов

Балансировка

При приходе запроса на Frontend он передает данные на некоторый из N Backend, прописанных у него в настройках. Обычно Backend для обработки запроса ищется по алгоритму Round-Robin, но есть и умные Frontend, учитывающие текущую загрузку Backend.

```
http {
    upstream backend {
        server restful-1:8080;
        server restful-2:8080;
    }
}
```

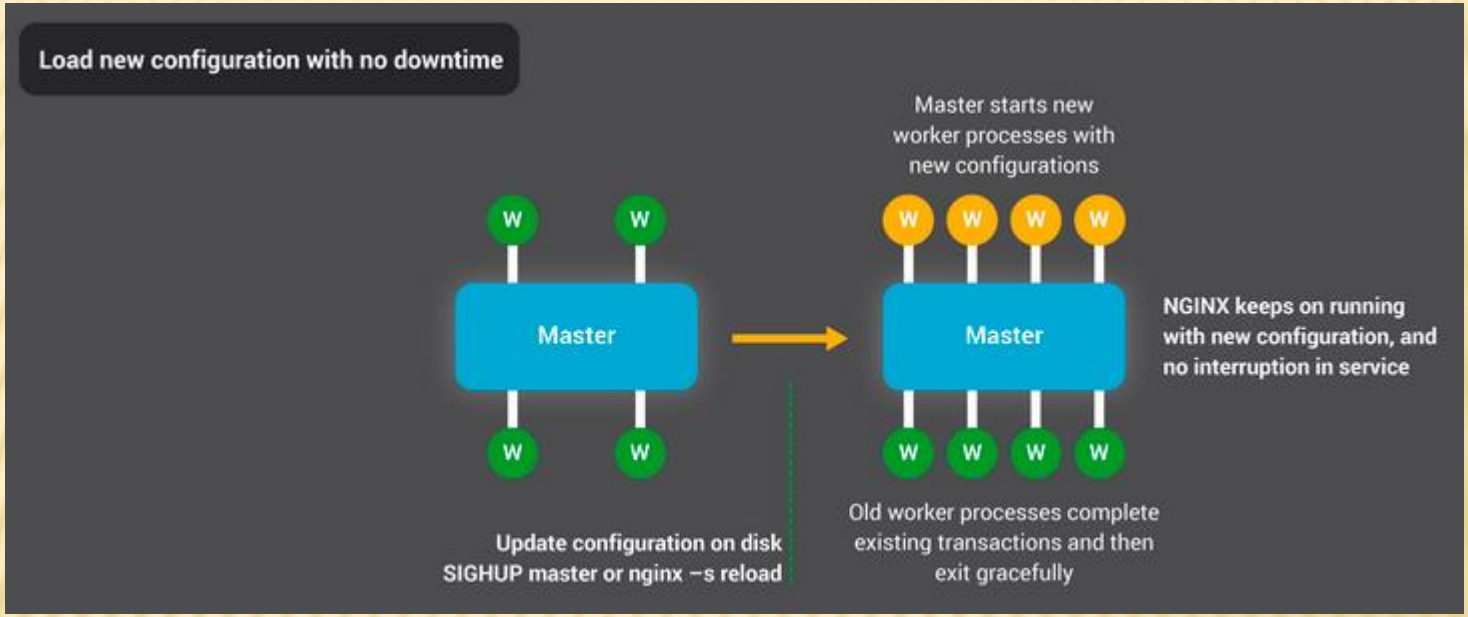
Задачи Frontend серверов

Балансировка

При приходе запроса на Frontend он передает данные на некоторый из N Backend, прописанных у него в настройках. Обычно Backend для обработки запроса ищется по алгоритму Round-Robin, но есть и умные Frontend, учитывающие текущую загрузку Backend.

```
http {
    upstream backend {
        server restful-1:8080;
        server restful-2:8080;
    }
}
```

NGINX



У nginx есть один главный и несколько рабочих процессов. Основная задача главного процесса — чтение и проверка конфигурации и управление рабочими процессами (главный процесс запускается от имени суперпользователя выполняет такие операции, как чтение конфигурации и открытие портов).
Рабочие процессы выполняют фактическую обработку запросов.

NGINX

nginx -s <signal> stop/quit/reload/reopen

- stop - принудительное завершение;
- quit - воркеры завершают обслуживание текущих задач и завершаются;
- reload - перезагрузка конфигурации;
- reopen - переоткрытие конфигурационного файла.

При перезагрузке конфигурации (reload) master-процесс валидирует ее, после этого посылает команду рабочим процессам завершиться. Рабочие процессы в свою очередь заканчивают принимать новые соединения и завершают обслуживание текущих, после чего завершаются.

NGINX

Конфигурационный файл делится на блоки:

```
{
    events {}
    http {
        # Блок server должен различаться по портам (listen) и имени сервера (server_name).
        server {
            listen      80
            server_name *.example.com
            # Блок location задаёт regexp префикса, который сравнивается с URI из запроса.
            # Для подходящих запросов добавлением URI к пути, указанному в директиве root,
            # получается путь к запрашиваемому файлу в локальной файловой системе.
            # Если есть совпадение с несколькими блоками location, nginx выбирает блок
с самым длинным префиксом.
            location / {
                root            /var/www
            }
        }
    }
}
```

NGINX

Конфигурационный файл делится на блоки: *балансировка*

```
upstream backend {
    server localhost:8081
    server localhost:8082
}
location / {
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_pass http://backend;
}
```

upstream описывает группу серверов для балансировки. По умолчанию соединения распределяются по серверам циклически (в режиме Round-Robin) с учётом весов серверов. Если при попытке работы с сервером происходит ошибка, то соединение передаётся следующему серверу, и так далее до тех пор, пока не будут опробованы все работающие серверы. Если связь с серверами не удалась, соединение будет закрыто.
server задаёт адрес в виде доменного имени или ip-адреса и другие параметры сервера.

Мастабирование Frontend серверов

Для масштабирования Frontend обычно используется DNS-балансировка. Т.е. запрос приходит на hostname, а DNS в свою очередь резолвит этот hostname в IP-адрес одного из Frontend серверов.

Для обеспечения отказоустойчивости берутся два сервера - основной и дублирующий, на каждом имеется два сетевых интерфейса. Внешние интерфейсы имеют одинаковые IP-адреса, но на дублирующем Frontend он выключен. Вторым сетевым интерфейсом они смотрят друг на друга. В случае, если основной сервер выходит из строя, дублирующий поднимает внешний IP-адрес и принимает обработку запросов на себя.