

COMP 3015

Week 10: Using Git for deployments, serving PHP applications (using Nginx), intro to DNS





Week 10 Topics

- Git
 - Dev workflow review
 - How we can use it for deployments
- Serving PHP applications (using Nginx)
- Domain Name System (DNS) Intro
 - How do we link a domain name to an IP address
- Additional Deployment Topics
 - Blue-Green



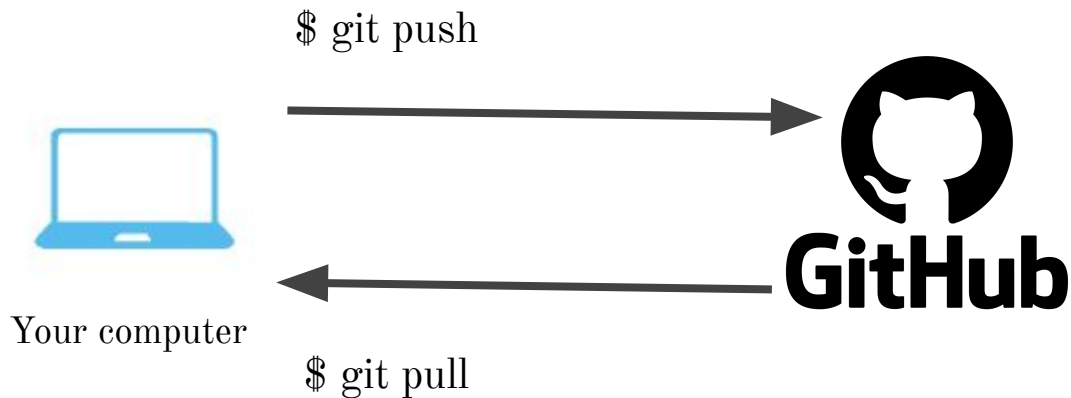
Git Workflow

1. Clone an existing repository, or create a new repository
2. Create a new branch: **\$ git checkout -b <branch name>**
3. Make changes to the code
4. Add the changes to the Git “staging area”: **\$ git add .**
5. Commit the changes: **\$ git commit -m “<descriptive commit message>”**
6. Push the branch to the remote repository: **\$ git push origin <branch name>**



Pushing changes to a remote repository

Your computer authenticates with GitHub using a username/password, or preferably, key based authentication:





How Git can help us deploy to a server

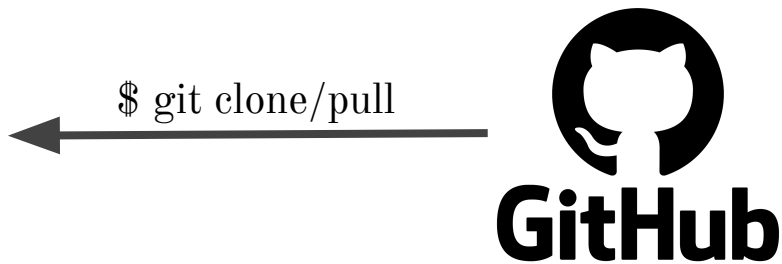
- Our server can authenticate with GitHub
- First, run **\$ git clone <repo info>** on the server
- Run **\$ git pull origin master** when we want to deploy to our server (run it from the web app server!)
- Our server will have whatever was on the master branch



Deployments using Git



Your web server





Deployments using GitHub + Deploy Keys

Steps:

1. Configure Nginx to pass PHP files to php-fpm
2. Configure the Nginx root directory (where Nginx should serve files from)
3. Generate a keypair on the application server
4. Upload the public key to GitHub

Serving PHP applications



Serving PHP Applications

There are three parts we need to understand:

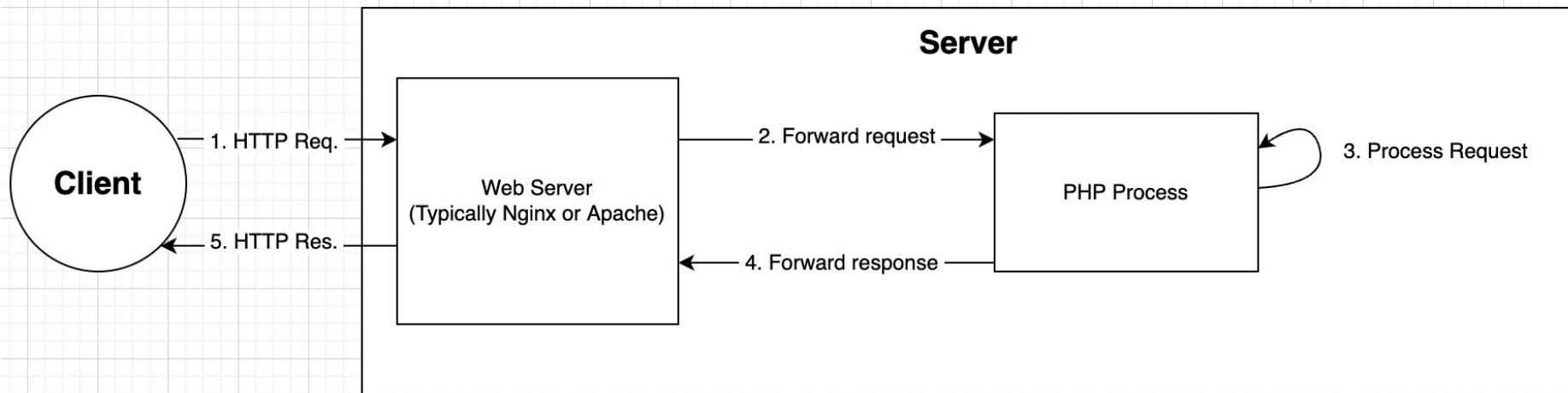
1. What Nginx is
2. What PHP-FPM is
3. How Nginx interacts with PHP-FPM

Nginx





Recall from week 1, our client-server model





Nginx

The main configuration file Nginx uses is **`/etc/nginx/nginx.conf`**

This configuration file can load additional configuration files (which are typically also in the **`/etc/nginx`** directory).



Nginx

- Nginx has one master process and multiple worker processes
- The master process reads and evaluates config files, and manages the worker child processes
- Worker processes fulfill requests



How many connections can we handle?

- We can define the number of [worker_processes](#) that the master Nginx process will manage.
 - In general, setting this to the number of available CPU cores your server has is a good start
- We can define the maximum number of simultaneous connections that a worker process can have open (to clients that have sent requests, other servers, etc.)
 - See [worker_connections](#)
- The maximum number of clients Nginx can handle concurrently is:

Max concurrent connections = [worker_processes](#) * [worker_connections](#)



Concurrency and Parallelism

Concurrency:

- Two or more tasks can start and be run in overlapping time periods
 - Think of it as an attribute of a program/system. When the program runs it may or may not be run in parallel
 - Processes are interleaved



time →

Parallelism:

- Tasks run at the same time (eg. on a multi-core processor)
 - eg. with two CPU cores:



time →



Concurrency and Parallelism

Two lines for one ATM machine: this is concurrency

Two lines and two ATM machines: this is parallelism



Nginx details

Max connections = worker_processes * worker_connections

```
✕ root@ubuntu-s-1vcpu-512mb-10gb-nyc1-01: /etc/nginx (ssh)
user www-data;
worker_processes auto;
pid /run/nginx.pid;
include /etc/nginx/modules-enabled/*.conf;

events {
    use epoll;
    worker_connections 768;
    # multi_accept on;
}
```

[worker_connections](#)

[worker_processes](#)

“auto” will use the number of available CPU cores



Nginx Documentation

- https://nginx.org/en/docs/beginners_guide.html
- https://nginx.org/en/docs/nginx_core_module.html

PHP-FPM





PHP-FPM

PHP-FPM: PHP-FastCGI Process Manager

“FPM (FastCGI Process Manager) is a primary PHP FastCGI implementation containing some features (mostly) useful for heavy-loaded sites.”

FastCGI? → Fast Common Gateway Interface

CGI? → Common Gateway Interface



What is PHP-FPM? (cont.)

Summary:

Web servers (e.g. Nginx) are programs, that we pass other programs to (e.g. our PHP applications), and somehow the web server executes those programs.

We configure our web servers to pass our applications to a running PHP-FPM process, and that process takes care of executing our application.



Passing files to PHP-FPM

Within an Nginx config such as: **/etc/nginx/sites-enabled/default**

~ is for case-sensitive matching:

https://nginx.org/en/docs/http/nginx_http_core_module.html#location

```
# pass PHP scripts to FastCGI server
#
location ~ \.php$ {
    include snippets/fastcgi-php.conf;
    fastcgi_pass unix:/run/php/php8.1-fpm.sock;
}
```



Configuring the root directory, server name

Within an Nginx config such as: **/etc/nginx/sites-enabled/default**

```
root /var/www/html/c3015.cfenn.com;  
  
# Add index.php to the list if you are using PHP  
index index.php index.html index.htm index.nginx-debian.html;  
  
server_name c3015.cfenn.com;
```



Generate a keypair on your **server** using ssh-keygen

```
root@ubuntu-s-1vcpu-512mb-10gb-nyc1-01:~# ssh-keygen -t ed25519 -C "christian_fenn@bcit.ca"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/root/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_ed25519
Your public key has been saved in /root/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:blBt5Cf5ruvdSgEn+UAP/fLMHNBqnN0oTe1x+SFVwb0 christian_fenn@bcit.ca
The key's randomart image is:
+--[ED25519 256]--+
|      +. ...=|
|      =o .o.|
|      . @.+O. o|
|      . . XX.*E.|
|      . S  =+O.+o|
|      o ....* .|
|      o  o      |
|      .  +  .    |
|      .+.o..    |
+-----[SHA256]-----+
root@ubuntu-s-1vcpu-512mb-10gb-nyc1-01:~#
```





Adding the public key to GitHub

```
root@ubuntu-s-1vcpu-512mb-10gb-nyc1-01:~# cat /root/.ssh/id_ed25519.pub  
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIEw7gRMhEg13YTUNJPsvXxskVlaPCUQ7WH8jE2C6Mpb christian_fenn@bcit.ca  
root@ubuntu-s-1vcpu-512mb-10gb-nyc1-01:~# █
```

Add the public key (note the .pub extension) to GitHub:

Deploy keys

Add deploy key



SSH

c3015.cfenn.com deploy key
SHA256:b1Bt5Cf5ruvdSgEn+UAP/fLMHNBqnN0oTe1x+SFVwb0
Added on Nov 16, 2022 by @ChristianFenn
Never used — Read-only

Delete



Deploying: \$ git pull origin master

```
root@ubuntu-s-1vcpu-512mb-10gb-nyc1-01:/var/www/html/c3015.cfenn.com# git pull origin master
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 2 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 879 bytes | 879.00 KiB/s, done.
From github.com:ChristianFenn/c3015.cfenn.com
 * branch                master      -> FETCH_HEAD
    e358e0b..987d2d9      master      -> origin/master
Updating e358e0b..987d2d9
Fast-forward
 index.php | 15 ++++++++-----
 1 file changed, 12 insertions(+), 3 deletions(-)
root@ubuntu-s-1vcpu-512mb-10gb-nyc1-01:/var/www/html/c3015.cfenn.com#
```


Note that you'll need to clone the repository first.

\$ git clone <repo info>




Alternatives to Nginx / PHP-FPM?

FrankenPHP: <https://frankenphp.dev/> - this uses an application server called [Caddy](#)



Domain Name System (DNS)

Intro





Domain Name System (DNS) Introduction

DNS translates domain names to IP addresses.

As a web app developer you need to know about DNS records.



DNS Introduction (cont.)

DNS works with different types of records

- “A” record: points to the IPv4 address of a server
- “AAAA” record: points to the IPv6 address of a server
- “CNAME” record: points one domain to another domain
 - Commonly used for adding support for the www “prefix”
- “NS” record: name server record, which contains the authoritative name servers



DNS Introduction ()

DNS record changes do not occur immediately!

See: <https://www.whatsmydns.net/>



Deployment Strategies

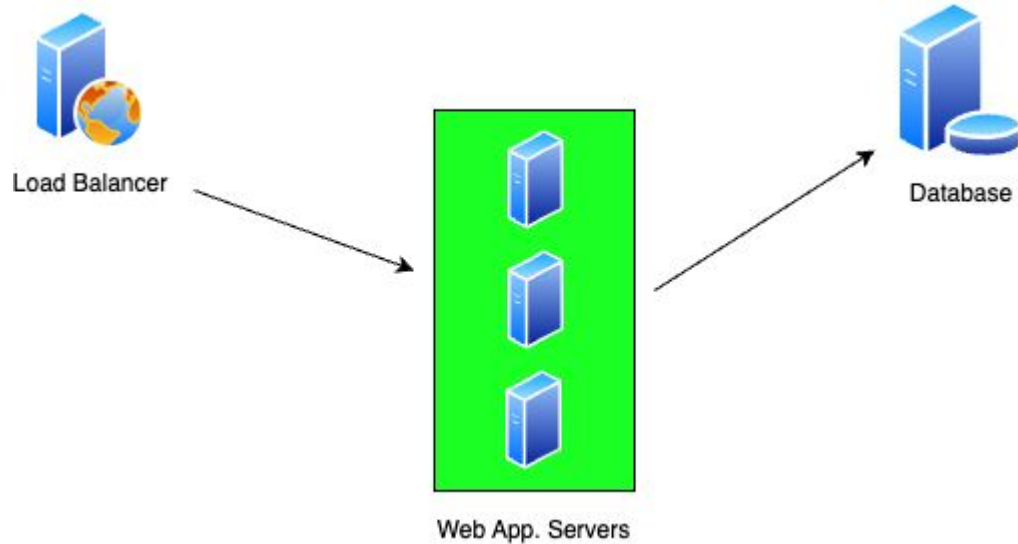


Blue-Green Deployments

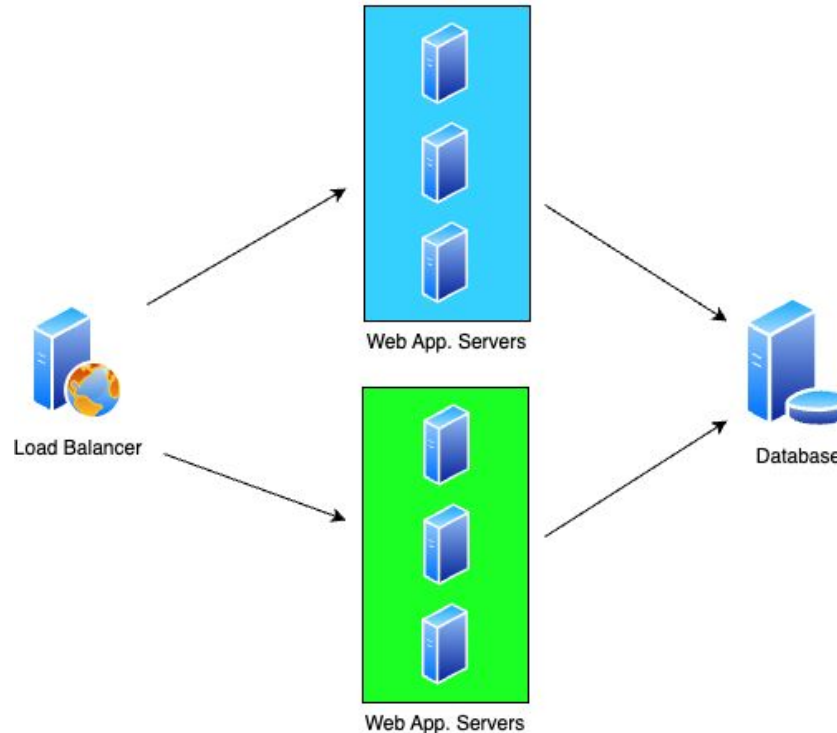
Idea:

1. Run application servers behind a load balancer
2. On deployment, replicate the servers
3. Test the newly provisioned servers are in an OK state post-deployment
4. Modify the load balancer to shift traffic to the newly provisioned servers

Blue-Green Deployments: pre-deployment



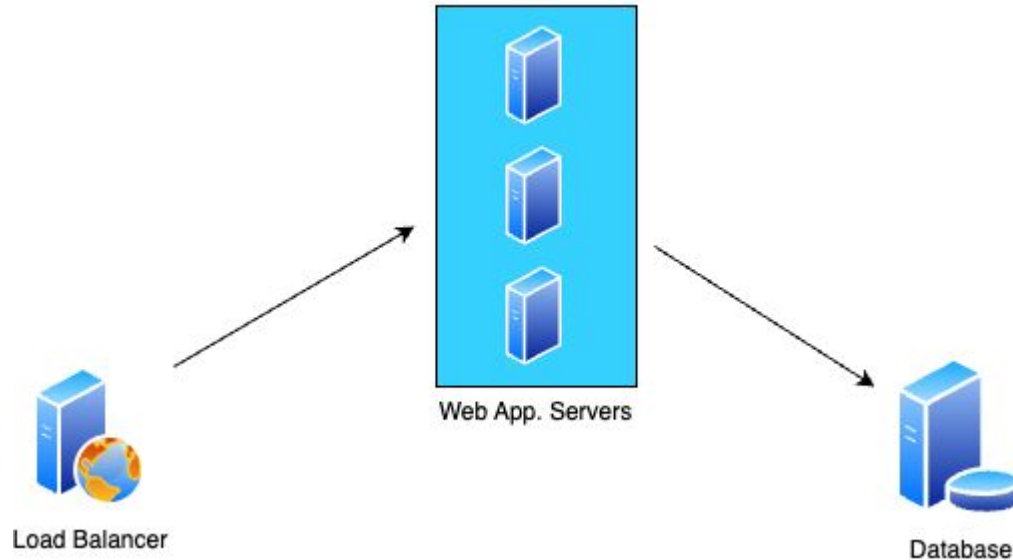
Blue-Green Deployments: deployment initiated



A deployment is initiated and servers are duplicated.

New servers running the updated application are in the blue group.

Blue-Green Deployments: success



If all goes well, 100% of the traffic can be shifted over to the servers running the new code!

No new lab today

Work on assignment #3

Practice final exam is posted