How to: Create a Password-Protected Wordpress Staging Environment Using Softaculous And Cloudflare Access

Daniel Rosehill
Jan 24 · 8 min read

What's better than a well-functioning Wordpress site, you might be wondering?

A well-functioning Wordpress site *with* a functional staging environment to push changes from!

I used to think that cloud-hosted staging environments were something that only developers with access to AWS tools and EC2 instances could take advantage of.

Wrong!

In the following tutorial I'm going to demonstrate how to quickly and easily set up a Wordpress staging environment using only a run-of-the-mill shared hosting plan and Cloudflare for security.

Prerequisites:

- You have a Cloudflare account.
- You have added your domain to Cloudflare.

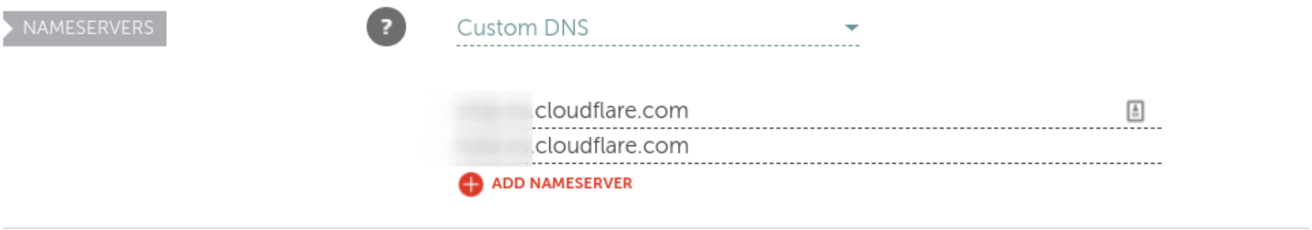
Step One: Point Nameservers to Cloudflare

This one goes without saying.

To use Cloudflare (a Content Delivery Network that does *so much more* than that) you need to set up an account and point over your nameservers.

Cloudflare assigns different nameservers to different users. You will find them in Cloudflare itself and by email. They'll be in the format x.cloudflare.com and you'll need two credentials.

This is a DNS change, so it could take as long as 48 hours to propagate.



Custom DNS

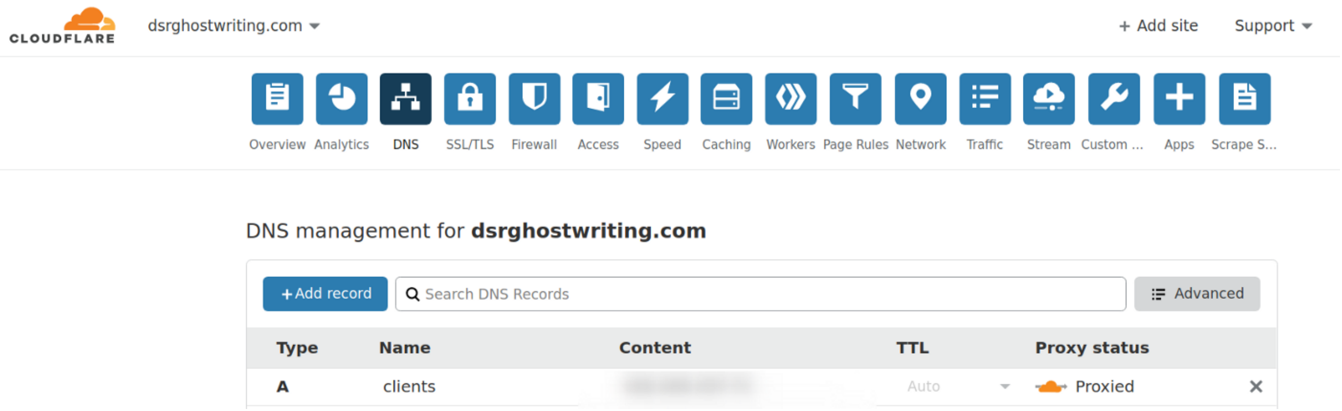
[redacted]cloudflare.com

[redacted]cloudflare.com

Swapping over nameservers to Cloudflare

. . .

Step Two: Set Up DNS Records on Cloudflare

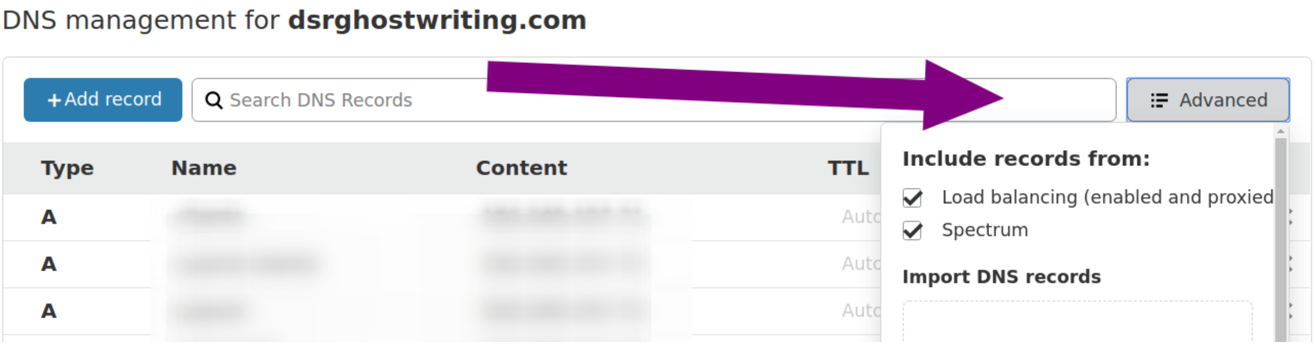


DNS management for **dsrghostwriting.com**

Type	Name	Content	TTL	Proxy status
A	clients	[redacted]	Auto	Proxied

Next, you'll want to copy over your zone files.

If you can export a BIND-formatted list of DNS records from your host, then click the 'Advanced' button in order to pull down a drag and drop import tool.



DNS management for **dsrghostwriting.com**

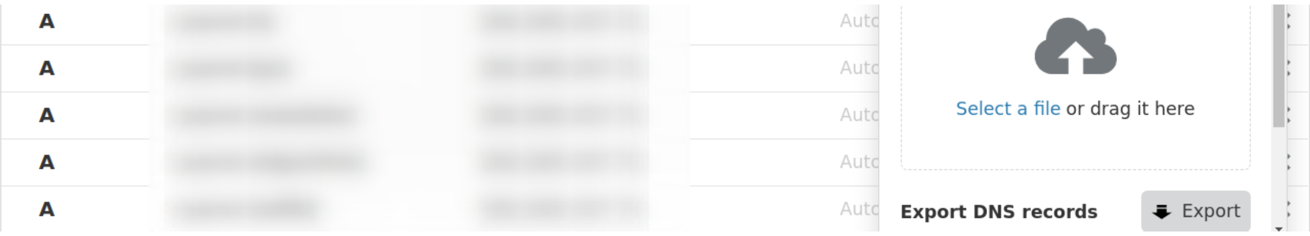
Type	Name	Content	TTL
A	[redacted]	[redacted]	Auto
A	[redacted]	[redacted]	Auto
A	[redacted]	[redacted]	Auto

Include records from:

☑ Load balancing (enabled and proxied

☑ Spectrum

Import DNS records

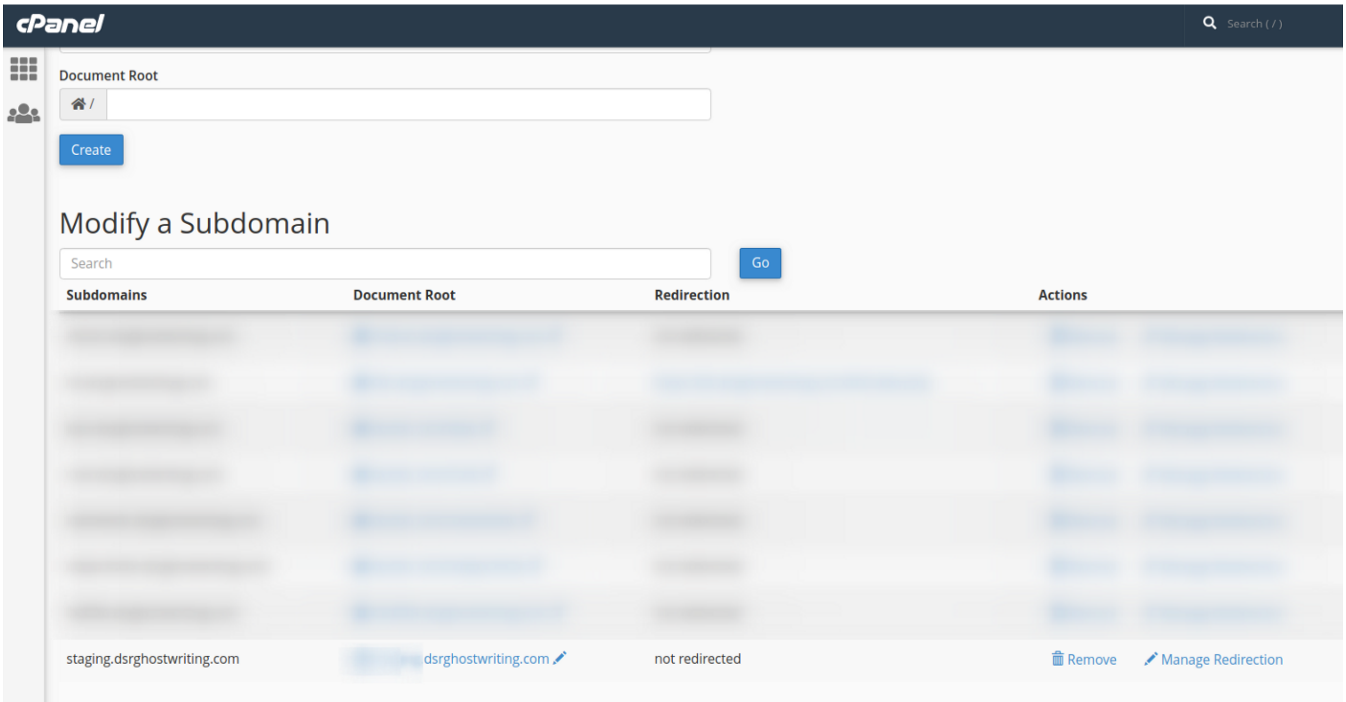


Step Three: Create Subdomain

Your staging environment should be a **subdomain** of your site — not a directory.

Create this in Cpanel and give the subdomain a document root that makes it obvious that it's a staging environment.

I went for staging.mydomain.com.



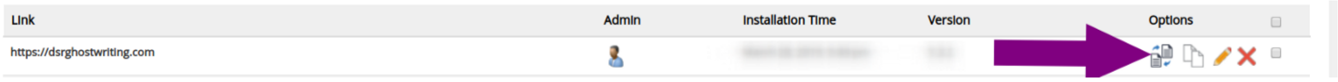
Modify a Subdomain

Subdomains	Document Root	Redirection	Actions
[redacted]	[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]	[redacted]
staging.dsrghostwriting.com	[redacted]dsrghostwriting.com	not redirected	Remove Manage Redirection

Step Four: Create Staging Environment in Softaculous

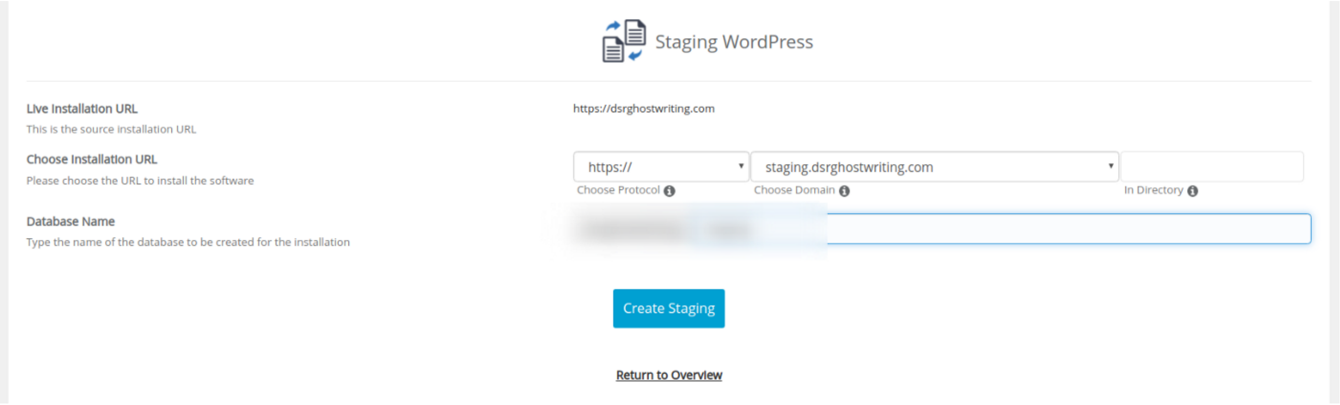
Next, you'll need to install Wordpress on your staging environment with Softaculous and mark it as that.

To do this click on the icon that looks like two pieces of paper being exchanged (the first one to the left) next to the site you want to create a staging environment for:



Link	Admin	Installation Time	Version	Options
https://dsrghostwriting.com		[redacted]	[redacted]	

You'll want to choose the subdomain you just created as the installation URL:



Staging WordPress

Live Installation URL
This is the source installation URL
https://dsrghostwriting.com

Choose Installation URL
Please choose the URL to install the software
https:// (Choose Protocol) staging.dsrghostwriting.com (Choose Domain) In Directory

Database Name
Type the name of the database to be created for the installation
[redacted]

Softaculous will duplicate your current site on to your new staging environment and show you a confirmation messages.

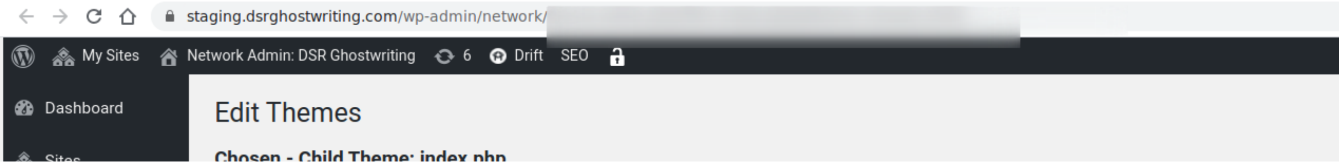
. . .

Step Five: Build Our Your Next Version(s) in Your Staging Environment

From here on in, you should work on your site in the staging environment.

(Pedantic note: calling it a staging environment is something of a misnomer, because in professional development setups the staging server is actually the environment *after* the development server where raw development is QA'd before being pushed to production. But let's stick with Softaculous's terminology.)

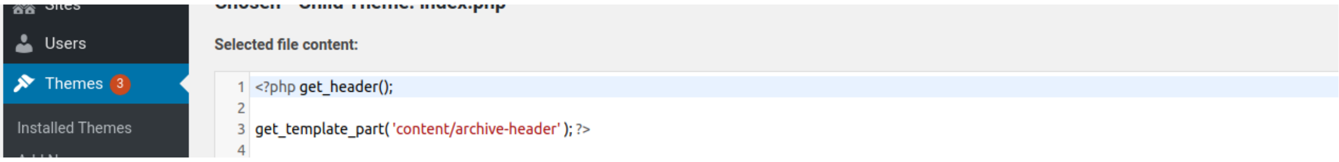
The beauty of the staging environment is that Softaculous will copy your Wordpress instance *exactly* between the two instances (or servers) while automatically rewriting every staging.yoursite resource to yoursite.



staging.dsrghostwriting.com/wp-admin/network/[redacted]

Edit Themes

Chosen - Child Theme: index.php



Chosen Child Theme: index.php

Selected file content:

```
<?php get_header();

get_template_part( 'content/archive-header' ); ?>
```

That means the Wordpress directory *and* its MySQL database with all its tables.

Which means that:

- All **plugin and theme** additions, changes, and deletions and
- All **content** additions, changes, and deletions

Will be replicated whenever you push to live.

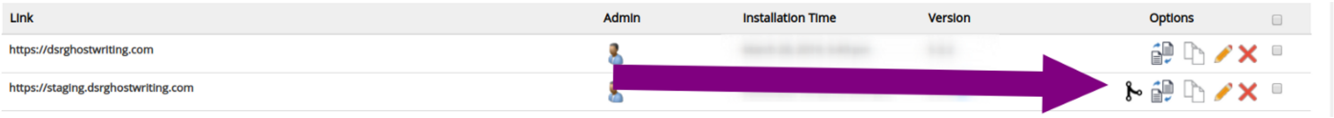
Additionally all updates — to the Wordpress core, to themes, and to plugins — will update.

So your involvement with your production site can cease with clicking the 'Push to Live' button .

This means, of course, that ***you should never directly edit your production site*** or your work will be overwritten the next time you push to live. (But if you do, you can just overwrite staging with them!)

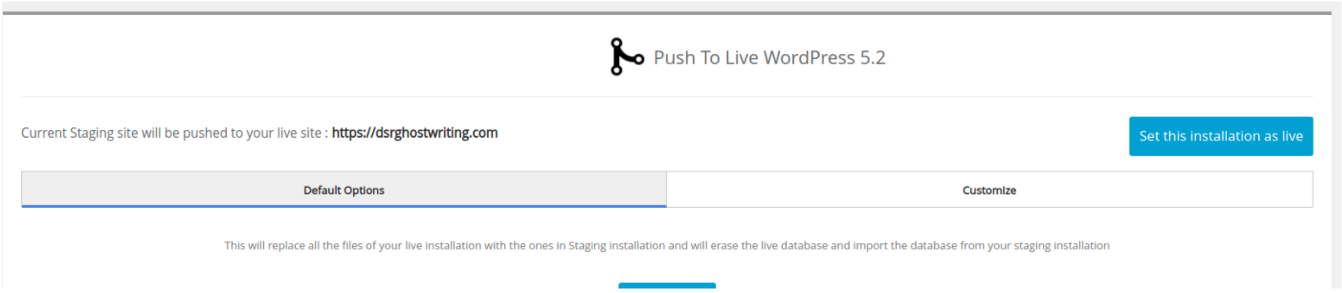
Now you can take your sweet time about fine-tweak-ing and QA-ing your site until it is pixel and functionality perfect. No more rushed sloppy work!

When you're done and ready to push out your latest batch of changes, simply click on the path icon:



Link	Admin	Installation Time	Version	Options	□
https://dsrghostwriting.com		[redacted]	[redacted]		□
https://staging.dsrghostwriting.com		[redacted]	[redacted]		□

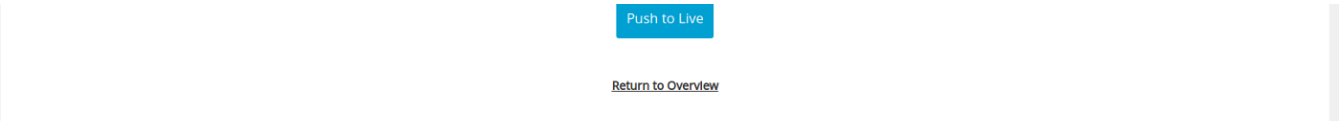
And then click 'Push to Live'



Push To Live WordPress 5.2

Current Staging site will be pushed to your live site : **https://dsrghostwriting.com**

This will replace all the files of your live installation with the ones in Staging installation and will erase the live database and import the database from your staging installation



Push to Live

Return to Overview

Softaculous will first take a backup of your production environment to help you restore it if sometimes goes amiss. Then it will propagate the files and MySQL changes.

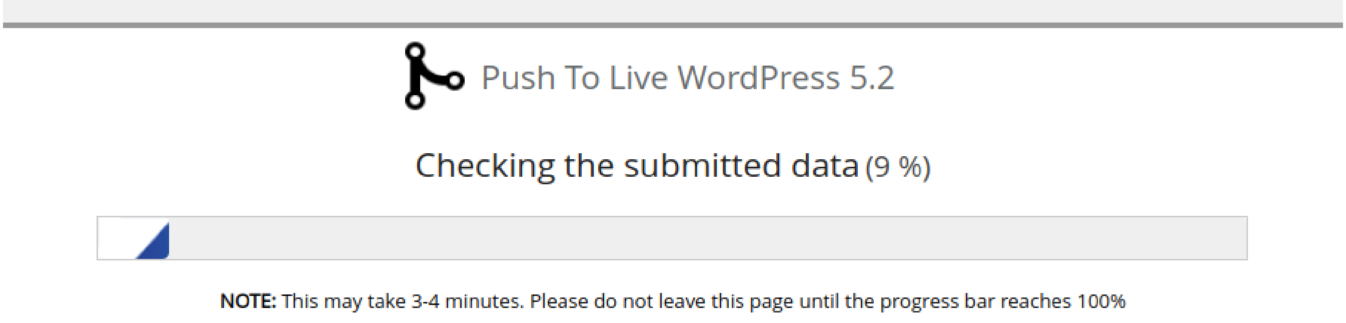
. . .

Push to Live Process

Keep the tab open and make sure that the push to live process runs through all its steps. This can take about four minutes, depending on the size of the site being pushed.

The stages you should see Softaculous go through are:

Step 1: Validation



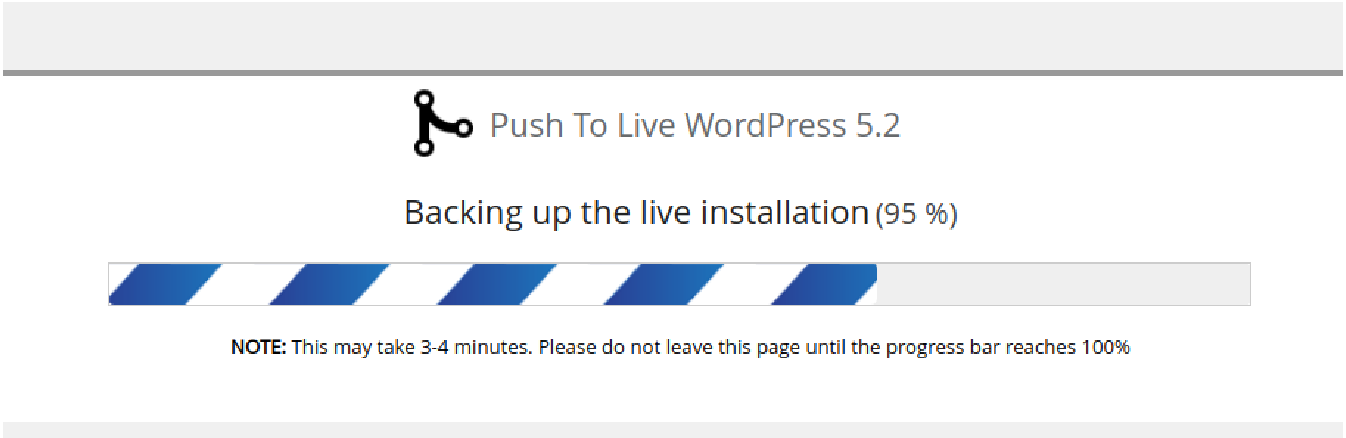
Push To Live WordPress 5.2

Checking the submitted data (9 %)

NOTE: This may take 3-4 minutes. Please do not leave this page until the progress bar reaches 100%

Step 2: Backup:

(I explain later why you might want to set up a cron job to automatically clear these from your server)

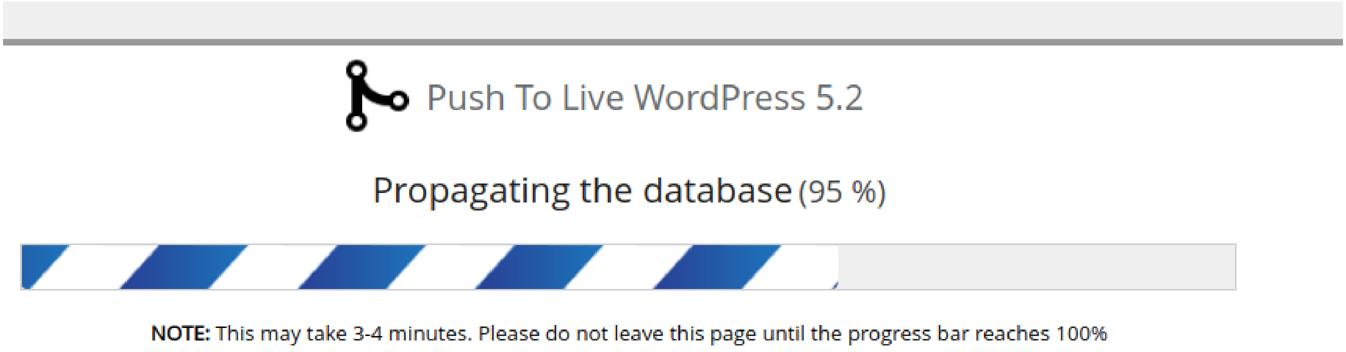


Push To Live WordPress 5.2

Backing up the live installation (95 %)

NOTE: This may take 3-4 minutes. Please do not leave this page until the progress bar reaches 100%

Step 3: Deployment / Propagation

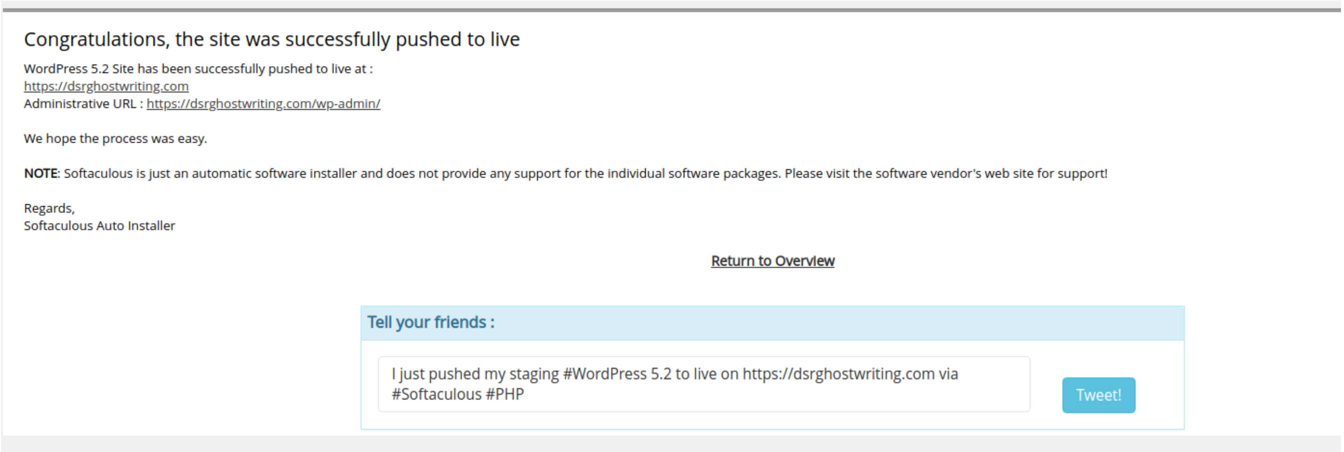


Push To Live WordPress 5.2

Propagating the database (95 %)

NOTE: This may take 3-4 minutes. Please do not leave this page until the progress bar reaches 100%

Step 4: Confirmation



Congratulations, the site was successfully pushed to live

WordPress 5.2 Site has been successfully pushed to live at :
https://dsrghostwriting.com
Administrative URL : https://dsrghostwriting.com/wp-admin/

We hope the process was easy.

NOTE: Softaculous is just an automatic software installer and does not provide any support for the individual software packages. Please visit the software vendor's web site for support!

Regards,
Softaculous Auto Installer

Return to Overview

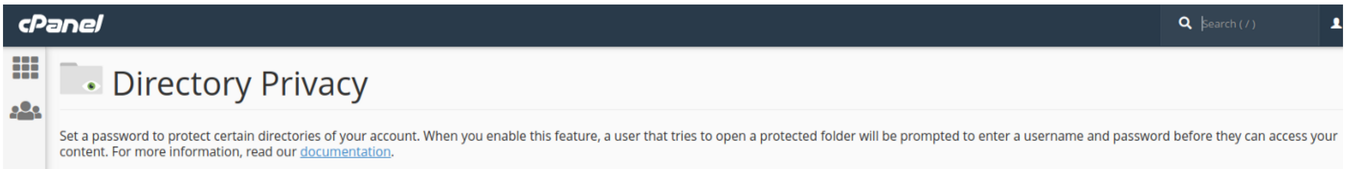
Tell your friends :

I just pushed my staging #WordPress 5.2 to live on https://dsrghostwriting.com via #Softaculous #PHP

Step Six: Using Cloudflare to Password Protect The Staging Environment

Ideally, cloud-hosted staging environments should not be exposed to the public or indexed by search engines.

It’s where you’ll be doing your raw work and making things break, after all.



Directory Privacy

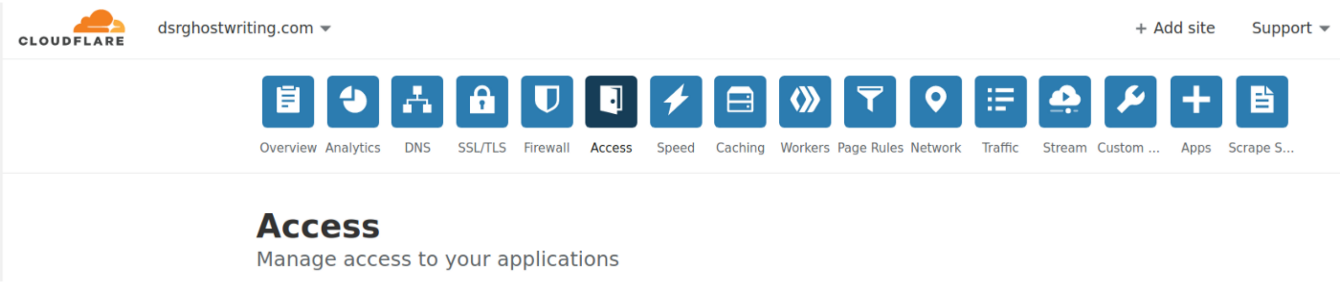
Set a password to protect certain directories of your account. When you enable this feature, a user that tries to open a protected folder will be prompted to enter a username and password before they can access your content. For more information, read our documentation.

The problem with using the Directory Privacy tool built into cPanel and Softaculous’s staging methodology is that it works by automatically regenerating the domain’s .htaccess file …. which of course gets carried over to production when you Push to Live — meaning that your public-facing Wordpress installation will be password protected.

Likewise for a robots.txt file. If you disable spiders on staging … you'll risk production vanishing from Google too.

Caging the subdomain at with Cloudflare is a more elegant solution to secure and hide your staging environment from the public and prevent it from indexing on Google.

To do this, set up Cloudflare Access:

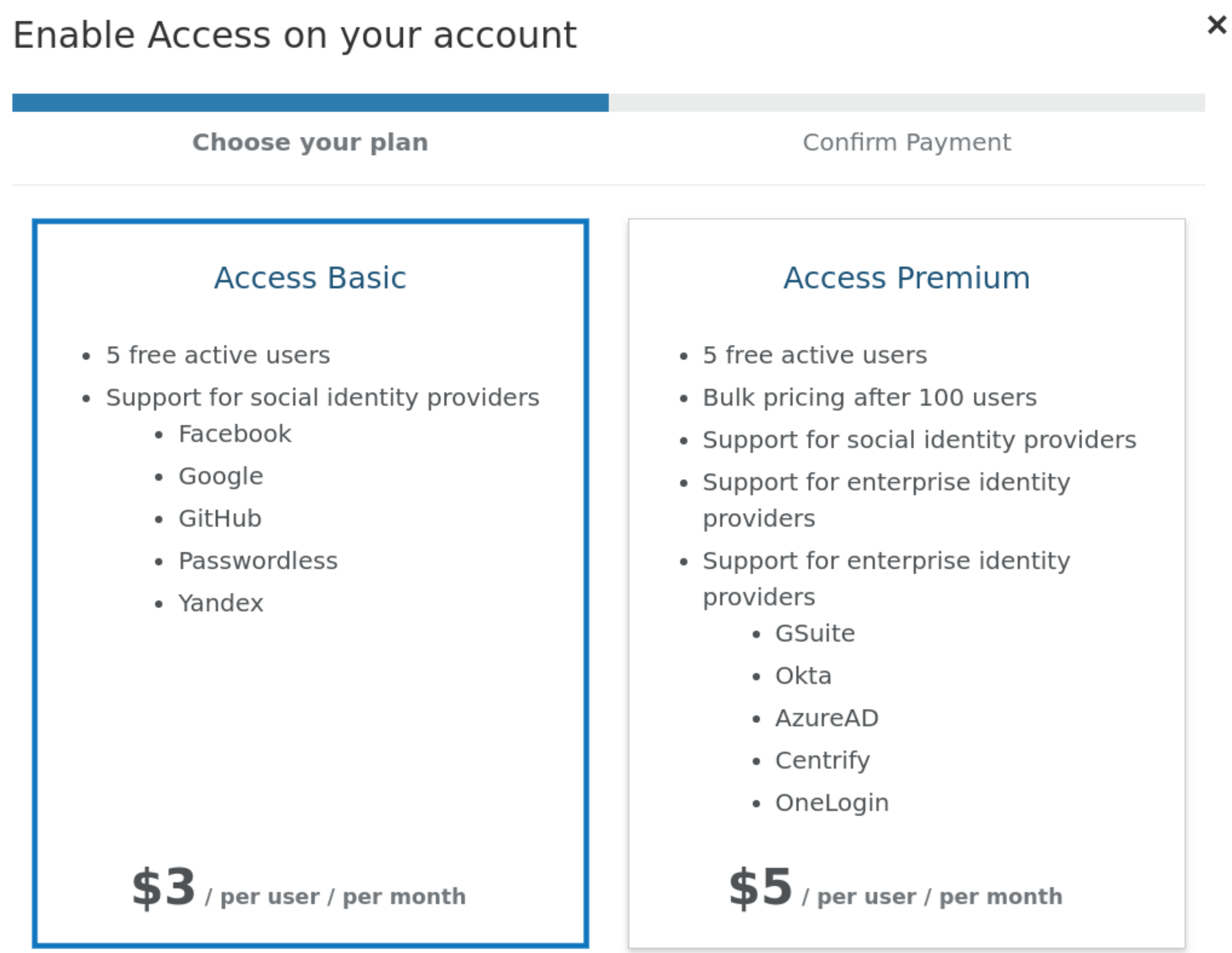


Access

Manage access to your applications

(At the time of writing) Access is free for up to five users per month.

But even if they change their pricing mode, the basic protection tier is only $3/user/month. If you're doing a lot of development work on your website, it's well worth it, in my opinion.



Enable Access on your account

Choose your plan — Confirm Payment

Access Basic

- 5 free active users
- Support for social identity providers
  - Facebook
  - Google
  - GitHub
  - Passwordless
  - Yandex

$3 / per user / per month

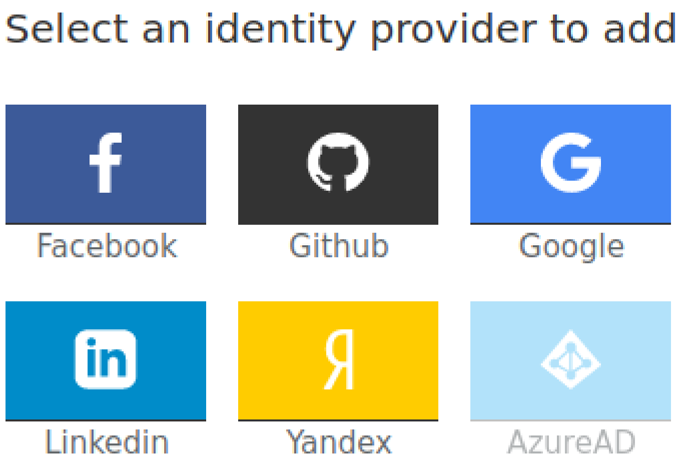
Access Premium

- 5 free active users
- Bulk pricing after 100 users
- Support for social identity providers
- Support for enterprise identity providers
- Support for enterprise identity providers
  - GSuite
  - Okta
  - AzureAD
  - Centrify
  - OneLogin

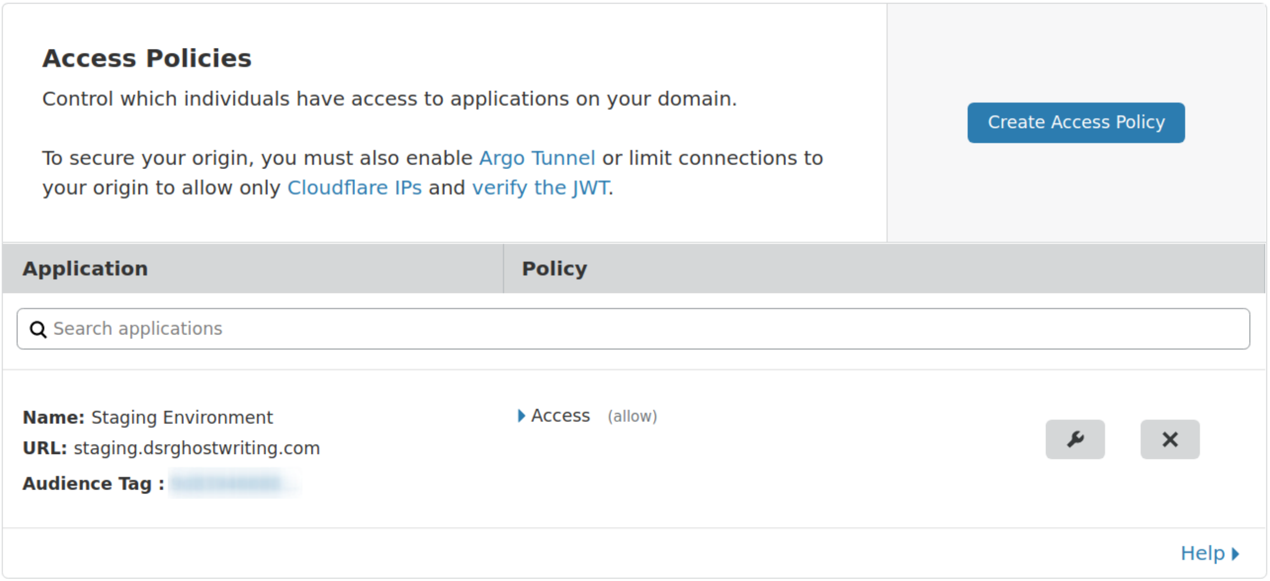
$5 / per user / per month

The easiest authentication method to set up is One-Time Pin (OTP). This means that every time you want to log in you'll have to enter a pin which gets sent to your email.

But you can also configure Single Sign On (SSO) through Facebook, Google, Github, and other providers.



Next, you'll want to create an **access policy** for your staging environment.



Access Policies

Control which individuals have access to applications on your domain.

To secure your origin, you must also enable Argo Tunnel or limit connections to your origin to allow only Cloudflare IPs and verify the JWT.

Create Access Policy

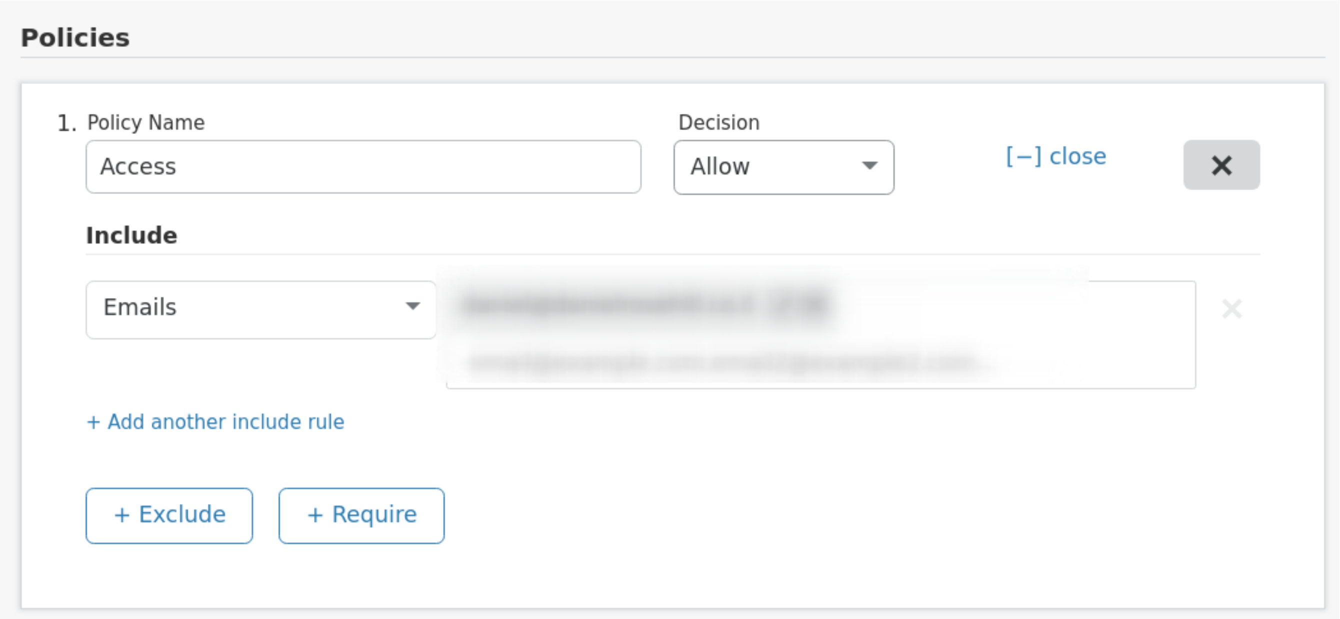
Application	Policy
Name: Staging Environment **URL:** staging.dsrghostwriting.com **Audience Tag :** [redacted]	▸ Access (allow)

Help ▸

If you're using the staging environment a lot, **then it makes sense to extend the default session duration from 24 hours to something like the maximum** … which is a month.

The policy I set up is the most simple one for authenticating by email OTP.

Set that up like this:



Policies

1. Policy Name: Access

Decision: Allow

Include

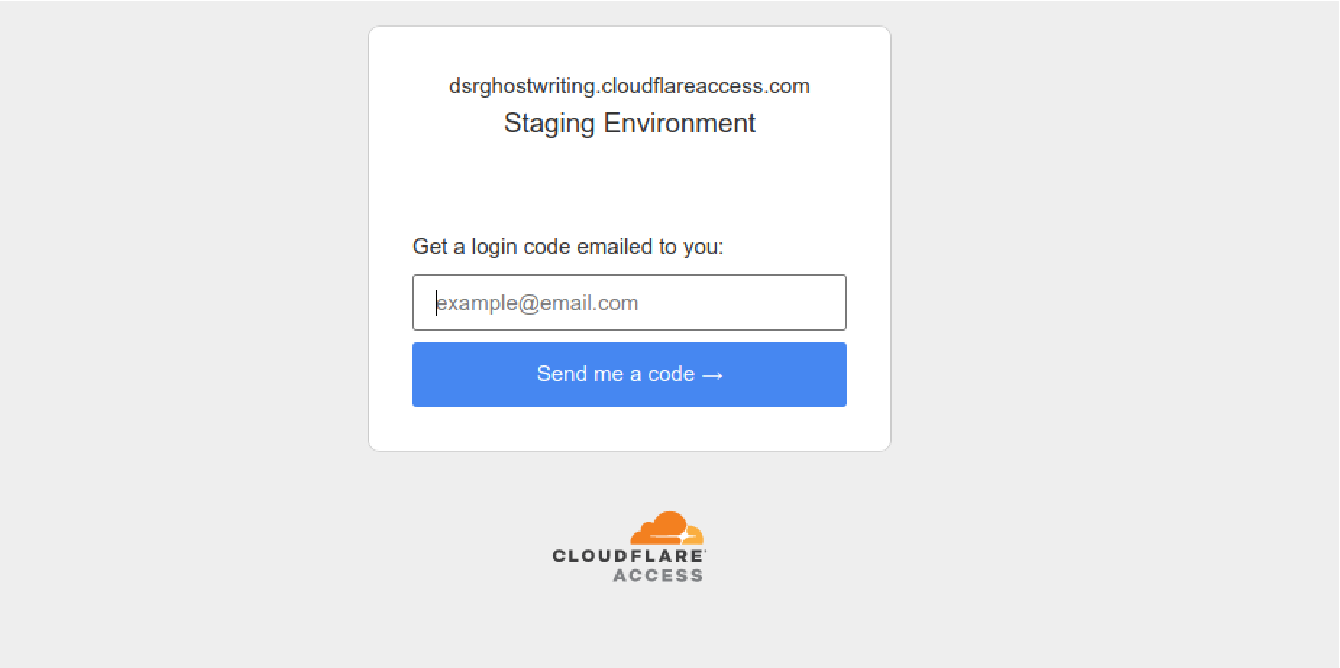
Emails: [redacted]

You can input a range of emails which you wish to be able to authenticate by OTP.

Save the policy to deploy it.

Step Seven: Log in to Staging!

Now, whenever you (or anybody) visits your staging environment, they'll be presented with the following page:



dsrghostwriting.cloudflareaccess.com

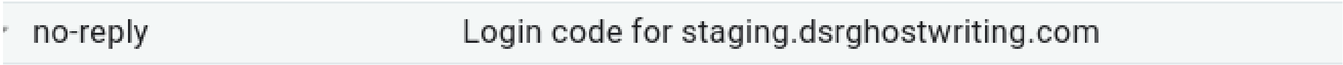
Staging Environment

Get a login code emailed to you:

example@email.com

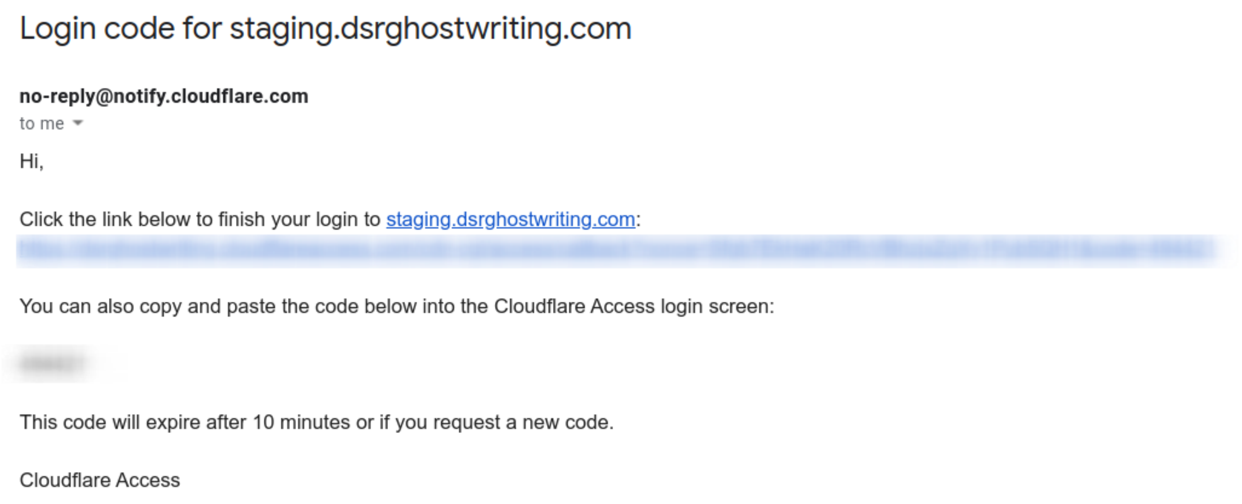
In order to securely log in, simply enter your email address into the box.

You'll get an instantaneous OTP form Cloudflare:



no-reply Login code for staging.dsrghostwriting.com

Click the one time link *or* enter the code to gain access:



Login code for staging.dsrghostwriting.com

no-reply@notify.cloudflare.com
to me

Hi,

Click the link below to finish your login to staging.dsrghostwriting.com:
[redacted]

You can also copy and paste the code below into the Cloudflare Access login screen:

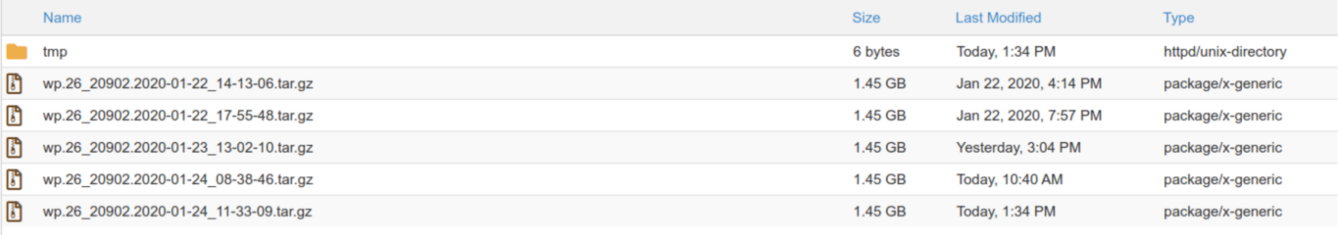
[redacted]

This code will expire after 10 minutes or if you request a new code.

Cloudflare Access

Step Eight (Optional): Add a cron job to clear old Softaculous backups

If you're doing a few pushes to live every day, the automatic backups which Softaculous generates can easily accumulate a few dozen gigabytes-worth of storage:



Name	Size	Last Modified	Type
tmp	6 bytes	Today, 1:34 PM	httpd/unix-directory
wp.26_20902.2020-01-22_14-13-06.tar.gz	1.45 GB	Jan 22, 2020, 4:14 PM	package/x-generic
wp.26_20902.2020-01-22_17-55-48.tar.gz	1.45 GB	Jan 22, 2020, 7:57 PM	package/x-generic
wp.26_20902.2020-01-23_13-02-10.tar.gz	1.45 GB	Yesterday, 3:04 PM	package/x-generic
wp.26_20902.2020-01-24_08-38-46.tar.gz	1.45 GB	Today, 10:40 AM	package/x-generic
wp.26_20902.2020-01-24_11-33-09.tar.gz	1.45 GB	Today, 1:34 PM	package/x-generic

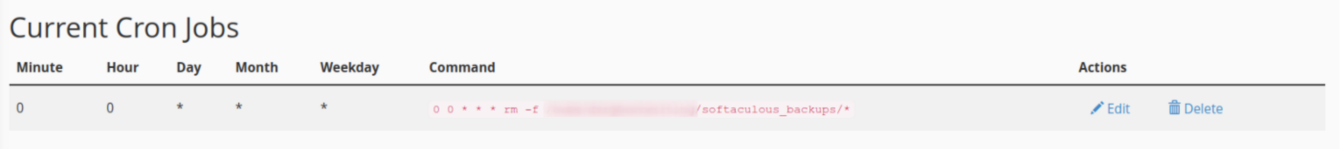
If you're on a storage-confined server, then you might want to automatically clear these daily.

If you have a separate backup solution in place as I do (which should mop up the entire server) then there's really no need to take these backups too. (You *can* disable the functionality entirely, but you'd need root access to your server to do so. If you have a VPS / dedicated server that's fully backed up then why not?)

In most shared hosting installations, the autogenerated Softaculous backups are stored in:

root/domain/softaculous_backups

I set up a once daily job to clear these out:



Current Cron Jobs

Minute	Hour	Day	Month	Weekday	Command	Actions	
0	0	*	*	*	`0 0 * * * rm -f [redacted]/softaculous_backups/*`	Edit	Delete

The cron line:

```
0 0 * * * rm -f /home/X/softaculous_backups/*
```

Where x=your domain document root (assuming you're on a shared host that offers Softaculous).

That's all there is to it. It's a few simple steps but it's made a world of difference for me so far in being able to carefully plan and deploy new versions of my website.

Congratulations — you're now the proud owner of a password-protected Wordpress staging environment in the cloud!

WordPress Development Web Development Staging Web Design