

Home > DevOps

Deploying a WordPress site to the AWS Cloud like a pro: our guide for painless maintenance using Docker and AWS Managed Services.

20 August 2021 - 7 min. read



WordPress is the easiest way to manage and create content. Its flexibility is loved by authors: with a couple of plugins you can do everything from hosting a cute kittens photo gallery to hosting an e-commerce site.

Let's face it: seen from the IT guy point of view, WordPress is a technical nightmare. When someone has to deal with it the horror begins: scalability is challenging, installation isn't scripted and a LAMP stack is not always easy to maintain.

In this article we'll give you some technical hints and examples to ease your relationship with WordPress in a cloud environment based on AWS.

We'll try to use as many AWS managed services as we can to be able to offload boring and dangerous tasks.

Database

We want our database to be highly available and scalable, so obviously we'll use Amazon Aurora with MySQL compatibility.

With Amazon Aurora we don't have to worry about space usage because the underlying storage can automatically scale when needed. In addition, with point-intime recovery you can restore your data with the granularity of a second. You can also use Multi-AZ configurations with read replicas that can take over in case of an Availability Zone failure.

AWS will automatically assign a reader endpoint and a writer endpoint (with a DNS record) when you create an Amazon Aurora Cluster instance.

In case of a failure (or maintenance) the read replica will be automatically promoted to become the writer and the endpoint will be automatically updated: simply configure your wp-config.php file with the writer endpoint and AWS will do all the work for you. If you know that your database will be stressed by a lot of read traffic you can add up to 15 read replicas to a single Aurora Cluster but you'll need to tell WordPress to use them. In this case, you can take advantage of the hyperdb plugin: with hyperdb you can define as many database read replicas as you want in your configuration file and use them. The example configuration file is well documented; here's an example configuration:

```
wpdb->add_database(array(
```

```
'host' => mydbcluster.cluster-123456789012.us-east-1.rds.amazonaws.c
om,
 'user' => DB USER,
 'password' => DB_PASSWORD,
 'name' => DB NAME,
));
$wpdb->add database(array(
 'host'
                => mydbcluster.cluster-ro-123456789012.us-east-1.rds.
amazonaws.com,
 'user' => DB USER,
 'password' => DB PASSWORD,
 'name' => DB NAME,
               => 0,
 'write'
 'read' \Rightarrow 1,
));
```

Instead, if you are starting small and don't know how and when your site will need to scale, Aurora Serverless is the best choice for you: it has the ability to scale the

compute layer and to "pause" if your website isn't accessed, so you can also save money !

Compute

We want to take advantage of the elasticity of the cloud, so using EC2 and Autoscaling Groups can be the natural choice. Also, we can go further and use Docker containers with an ECS Fargate cluster with a little bit of application refactoring.

Using containers reduces maintenance activities for operating systems updates and makes scaling more easily. As a bonus point we can also automate the deployment workflow using pipelines.

WordPress offers a pre-built container image with a vanilla installation:

We can start building our docker image adapting it to our needs, installing plugins automatically using wp-cli: in our example we'll install the wordpress-seo plugin (we'll assume that you already have a working wp-config.php file)

Please note that this is only an example. We suggest you to tailor and customize your Dockerfile to your needs: we always recommend that you know what software you are using and running, especially in containerized solutions.

Example Dockerfile:

```
FROM wordpress
```

```
COPY wp-config.php /usr/src/wordpress/
```

RUN curl -O https://raw.githubusercontent.com/wp-cli/builds/gh-pages /phar/wp-cli.phar && chmod +x wp-cli.phar && mv wp-cli.phar /usr/loca l/bin/wp

```
WORKDIR /usr/src/wordpress/
```

RUN wp --allow-root core update RUN wp --allow-root plugin install wordpress-seo

WORKDIR /var/www/html

We just automated our installation, making it maintainable and ready for testing in different environments.

Simply run:

```
docker build . -t myawesomewordpresscontainer
```

and you'll have a ready-to-go container to deploy. You can use a docker-compose.yml file to test on your local pc, run it in a development environment or in production using ECS.

Creating a serverless execution environment on AWS ECS is pretty simple:

On AWS Console go to ECS -> Create Cluster, select "Networking Only" to define a Fargate Cluster:



Give it a name:

Configure cluster	
Cluster name*	wordpress-ecs
Networking	
Create a new VPC for your cluster to use. as Fargate tasks.	A VPC is an isolated portion of the AWS Cloud populated by AWS objects, such
Create VPC	Create a new VPC for this cluster
Tags	
Key	Value
Add key	Add value
CloudWatch Container Insights	
CloudWatch Container Insights is a monite collects, aggregates, and summarizes cor such as container restart failures to help y CloudWatch Container Insights	vring and troubleshooting solution for containerized applications and microservices. It npute utilization such as CPU, memory, disk, and network; and diagnostic information ou isolate issues with your clusters and resolve them quickly. (27 Learn more Enable Container Insights
*Required	Cancel Previous Create

Create a Docker Repository for the image: Select "Amazon ECR -> Repositories

General settings	
/isibility settings Info	
 Private Access is managed by IAM and repository policy permission 	ons.
 Public Publicly visible and accessible for image pulls. 	
Repository name rovide a concise name. A developer should be able to identify	y the repository contents by the name.
kr.ecr.eu-west-1.amazonaws.con	n/ wordpress-repo
4 out of 256 characters maximum (2 minimum). The name m hyphens, underscores, and forward slashes.	sust start with a letter and can only contain lowercase letters, numbers,
nable tag immutability to prevent image tags from being over	any either by subsequent image pushes using the same tag. Disable tag
mmutability to allow image tags to be overwritten. Disabled	e written by subsequent iningle pusites using the same tag, bisable tag
mmutability to allow image tags to be overwritten. Disabled Once a repository is created, the visibility sett	ewritten by subsequent image pusies training the same tag. Disaute tag
mmutability to allow image tags to be overwritten. D Disabled Once a repository is created, the visibility sett mage scan settings	ing of the repository can't be changed.
 Disabled Once a repository is created, the visibility sett a repository is created, the visibility sett mage scan settings can on push nable scan on push to have each image automatically scanne nanually started to get scan results. Disabled 	ing of the repository can't be changed.
Immutability to allow image tags to be overwritten. Disabled Once a repository is created, the visibility sett mage scan settings can on push rable scan on push to have each image automatically scanne Disabled Encryption settings	ing of the repository can't be changed.
mmutability to allow image tags to be overwritten. Disabled Once a repository is created, the visibility sett mage scan settings can on push inable scan on push to have each image automatically scanne namually started to get scan results. Disabled Encryption settings (MS encryption Ou can use AWS Key Management Service (KMS) to encrypt in etings. Disabled	Ing of the repository can't be changed.

Click on the just created ECR repository by clicking on "View push commands" you'll get ready-to-go instructions to build and upload the container to the repository:

ECR > Repositories > wordpress-repo		
wordpress-repo		View push commands

Once image upload is done you can add a task definition for the ECS cluster, he task definition will define how to run our container in the ECS cluster:

Click on "task-definitions" on the AWS Console sidebar and select "Create new Task Definition":



Give the task definition a name and assign the resources.

Since we are developing a small demo we'll select 0.25 CPU units and 0.5 GB of memory (keep also in mind that it is always better to scale horizontally having multiple small containers).

Click on "Add Container" to add the container definition, so we can specify everything for the Docker execution Environment

Container name*	wordpress-container		
lmage*	dkr.ecr.eu	-west-1. <u>amazonaws</u> .com/ <u>wordpress</u> .	repo:latest
Private repository authentication*	0		0
Memory Limits (MiB)	Soft limit 128		0
	Add Hard limit Define hard and/or soft memory the 'memory' and 'memoryRess ECS recommends 300-500 MiB	limits in MiB for your container. Hard and s ervation' parameters, respectively, in task d as a starting point for web applications.	oft limits correspond to efinitions.
Port mappings			0
	80	tcp -	8

Don't forget to map port 80!

After you create the task definition you can define a service to run the container:

eploy Info		
Environment		
Existing cluster Select an existing cluster. To create a new cluster, go	to Clusters.	
wordpress-ecs		
Compute configuration (advanced)		
Deployment configuration		
Application type Info Specify what type of application you want to run.		
• Service Launch a group of tasks handling a long-running computing work that can be stopped and restarted. For example, a web application.	 Task Launch a standalone task that runs and terminates. For example, a batch job. 	
Task definition Select an existing task definition. To create a new tas pecify revision manually Manually input revision instead of choosing from task definition family.	k definition, go to Task definitions. the 100 most recent revisions for the selected	
Family	Revision	
wordpress-task 🔻	1 (LATEST)	
Service name Assign a unique name for this service.		
wordpress-service		
Desired tasks Specify the number of tasks to launch.		
2	\$	
Deployment options		

As you can see we already choose to deploy 2 tasks, so our wordpress installation will be highly available and load balanced.

We're not showing you security groups and balancing options because they are common configuration tasks on AWS (need help ? Write to us!).

Storage

Since our container is stateless (by definition, here you can find more the details about differences between stateful and stateless services:

https://www.proud2becloud.com/stateful-vs-stateless-the-good-the-bad-and-theugly/) we need to solve a final problem: static assets and persistence.

Our service of choice is Amazon EFS, a shared and distributed file system. We can store assets that are in the wp-content/uploads directory by simply adding an EFS filesystem it in the task definition and giving it a name:

	Add volume			3
	Name	wp-assets		0
	Volume type	EFS	•	0
	File system ID	· · · ·	C	0
		Create an Amazon Elastic File System i the Amazon EFS console .	in	
	Access point ID		C	0
		Create an access point for your file system in the Amazon EFS console .	em	
	Root directory	1		0
	Encryption in transit	Enable transit encryption		0
	EFS IAM authorization	Enable IAM authorization		0
	 Advanced configuration 			
	*Required			Cancel Add
And then define the m	nount in the co	ntainer:		

STORAGE AND LOGGING			
Read only root file system			0
Mount points		wp-assets *	0
		/var/www.html/wp-content/uploads/	
			0
	0	Add mount point	

Don't forget to add an AWS Backup job for the EFS share!

There are also WordPress plugins that can take advantage of S3. Since we want to keep our demo installation simple, we won't use them for this specific case, but you can, consider them while planning your installation.

Caching

We will not show these steps but, as a general rule of thumb, you can reduce compute cost and make the site more responsive for users by adding a CloudFront distribution in front of your load balancer.

You can also use an application cache for user sessions and database query offloading using Amazon Elasticache for Redis with the redis-cache WordPress plugin.

Deploying a new Elasticache for Redis cluster is a matter of minutes: search for Elasticache on the AWS console and then click "Create":

Cluster engine	Redis In-memory data structure store used as database, cache and message broker. ElastiCache for Redis offers Multi-AZ with Auto-Failover and enhanced robustness. Cluster Mode enabled Memcached High-performance, distributed memory object caching system, intended for use in speeding up dynamic web applications.
Location	
Choose a location	Amazon Cloud Use Amazon's cloud for your ElastiCache instances On-Premises Create your ElastiCache instances on AWS Outposts. You need to create a subnet ID on an Outpost first.
Redis settings	land ekeratariatian ta energidar urban riskt aining Amanan ElectiGenka Dedia ekutara Langura
Name	wordpress-redis-cluster
Description	Elasticache Redis kluster for wordpress article
Engine version compatibility	6.x
Port	6379
Parameter group	default.redis6.x
Node type	cache.t3.micro (0.5 GiB)
Number of replicas	2
Multi-A7	

In the "Advanced Redis settings" create a new subnet group and select at least two subnets in different availability zones to keep the cluster private and highly available, select or create a security group to grant wordpress containers access to the Elasticache cluster.

Subpet group	Create new		-	0
Sublict group	Create new			0
Name	redis-wordpress-subnet-group			0
Description	Description			0
VPC ID	vnc-05a7fc6d		•	0
	.,			
Subnets				0
	Subnet ID 👻	Availability zone 🗸	CIDR Block •	
	subnet-f3ee3694	eu-west-1c	10.100.130.0/24	
	subnet-9b993fd2	eu-west-1a	10.100.128.0/24	
	subnet-1058d849	eu-west-1b	10.100.4.0/24	
	subnet-baed35dd	eu-west-1c	10.100.134.0/24	
	subnet-cc62c485	eu-west-1a	10.100.132.0/24	
	subnet-cc3dd997	eu-west-1b	10.100.133.0/24	
	subnet-f133d7aa	eu-west-1b	10.100.129.0/24	
	subnet-ae7e95cb	eu-west-1c	10.100.2.0/24	
	subnet-1ba7fc73	eu-west-1a	10.100.0.0/24	
	subnet-2a9ca86c	eu-west-1b	10.100.1.0/24	
	subnet-41efa924	eu-west-1c	10.100.5.0/24	
	subnet-60f39d17	eu-west-1a	10.100.3.0/24	
Augilability and allocated	 No sectorement 			
Avanability zones placement	Select zones			0

Click on "Create" and your cluster will be ready in a couple of minutes.

Add in the Dockerfile this line to install the redis-cache plugin

RUN --allow-root plugin install redis-cache

And then configure it to cache user sessions using the the primary endpoint shown in cluster details:



WAF

To prevent and mitigate attacks and common vulnerabilities consider enabling AWS WAF and configure it to use Managed Rules for AWS Firewall ruleset. You can enable AWS WAF on the Application Load Balancer or on the CloudFront Distribution (if you choose to use it)

Maintaining containers: ThePipelines!

Our choice for automating our WordPress container maintenance is the CodeBuild/CodePipeline duo.

simply follow these steps to deploy pipelines: https://www.proud2becloud.com/awsfargate-services-deployment-with-continuous-delivery-pipeline/

You can also rely on a blu/green deployment strategy. Here's how to do it: https://www.proud2becloud.com/how-to-setup-a-continuous-deployment-pipelineon-aws-for-ecs-blue-green-deployments/

Starting a containerized WordPress deployment from scratch isn't hard: with the right managed services you'll offload a lot of tasks like maintaining high availability, keeping the focus on maintaining content.

What's your experience with containerized WordPress deployment? Have you ever faced any challenges?

That's all for today.

Keep reading and see you in 14 days on #Proud2beCloud!



Damiano Giorgi

Ex on-prem systems engineer, lazy and prone to automating boring tasks. In constant search of technological innovations and new exciting things to experience. And that's why I love Cloud Computing! At this moment, the only "hardware" I regularly dedicate myself to is that my bass; if you can't find me in the office or in the band room try at the pub or at some airport, then!

Copyright © 2011-2021 by beSharp srl - P.IVA IT02415160189