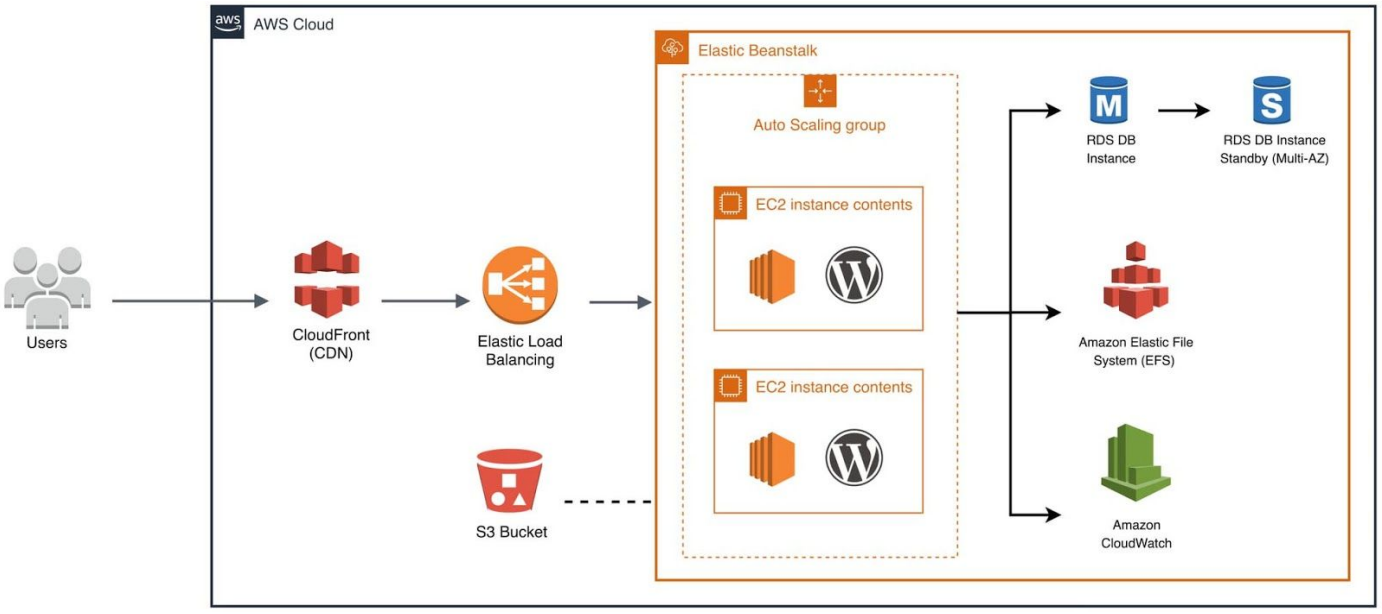
Cloud-1



For deploying a high-availability WordPress website i used the **AWS Elastic Beanstalk** service:

AWS Elastic Beanstalk is an easy-to-use service for deploying and scaling web applications and services developed with Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker on familiar servers such as Apache, Nginx, Passenger, and IIS.

In my case i chose to use the Elastic Beanstalk CLI, and i created a new application with an environment of type (load balancing, auto scaling) with these information:

Name: wordpress-cloud
Region: US East (N. Virginia) - us-east-1
Platform: PHP - PHP 7.4 running on 64bit Amazon Linux 2
Application code: Sample application

Elastic Beanstalk created resources:

- **EC2 instance** – An Amazon Elastic Compute Cloud (Amazon EC2) virtual machine configured to run web apps on the platform that you choose.
  Each platform runs a specific set of software, configuration files, and scripts to support a specific language version, framework, web container, or combination of these. Most

platforms use either Apache or NGINX as a reverse proxy that sits in front of your web app, forwards requests to it, serves static assets, and generates access and error logs.

- **Instance security group** – An Amazon EC2 security group configured to allow inbound traffic on port 80. This resource lets HTTP traffic from the load balancer reach the EC2 instance running your web app. By default, traffic isn't allowed on other ports.

- **Load balancer** – An Elastic Load Balancing load balancer configured to distribute requests to the instances running your application. A load balancer also eliminates the need to expose your instances directly to the internet.

- **Load balancer security group** – An Amazon EC2 security group configured to allow inbound traffic on port 80. This resource lets HTTP traffic from the internet reach the load balancer. By default, traffic isn't allowed on other ports.

- **Auto Scaling group** – An Auto Scaling group configured to replace an instance if it is terminated or becomes unavailable.

- **Amazon S3 bucket** – A storage location for your source code, logs, and other artifacts that are created when you use Elastic Beanstalk.

- **Amazon CloudWatch alarms** – Two CloudWatch alarms that monitor the load on the instances in your environment and that are triggered if the load is too high or too low. When an alarm is triggered, your Auto Scaling group scales up or down in response.

- **AWS CloudFormation stack** – Elastic Beanstalk uses AWS CloudFormation to launch the resources in your environment and propagate configuration changes. The resources are defined in a template that you can view in the AWS CloudFormation console.

- **Domain name** – A domain name that routes to your web app in the form subdomain.region.elasticbeanstalk.com.

- **RDS instance** – Amazon Relational Database Service (Amazon RDS) makes it easy to set up, operate, and scale a relational database in the cloud. It provides cost-efficient and resizable capacity while automating time-consuming administration tasks such as hardware provisioning, database setup, patching and backups. It frees you to focus on your applications so you can give them the fast performance, high availability, security and compatibility they need.

- **EFS** – Creates an EFS file system and mount points in each Availability Zone/subnet in your VPC. so the website uses Amazon Elastic File System (Amazon EFS) as the shared storage for uploaded files.

Scalability and High availability:

In the Auto Scaling group you can configure how your Scalability and High availability system work, in my case i set:

1. **Metric** (The metric that is monitored to determine if the environment's capacity is too low or too high)**: CPUUtilization**
2. **Statistic** (Choose how the metric is interpreted)**: Average**
3. **Unit: Percent**
4. **Period** (The period between metric evaluations)**: 1 Min**
5. **Breach duration** (The amount of time a metric can exceed a threshold before triggering a scaling operation)**: 1 Min**
6. **Upper threshold: 22 Percent**
7. **Scale up increment: 1 EC2 instances**
8. **Lower threshold: 2 Percent**
9. **Scale down increment: -1 EC2 instances**

And also configure the compute capacity of the environment, in my case i use:

Max: 2 EC2 Instances
Min: 5 EC2 Instances (you can use more)

For the database high availability i chose **Availability: High (Multi-AZ)** that's creates a standby in a different Availability Zone (AZ) to provide data redundancy, eliminate I/O freezes, and minimize latency spikes during system backups.**,** and also I created a read replica for my DB instance.

- **Notification:**

For ELB Environment:
In any action or change that happen in the environment like when the a new incidence created or removed i received an email and also a notification in the environment logs like these screens:

➔ **Added instance:**

2020-12-19 11:33:23 UTC+0100	INFO	Added instance [i-055e0de3fa5ec22c2] to your environment.
2020-12-19 11:16:24 UTC+0100	INFO	Added instance [i-0298929060eb122b1] to your environment.

➔ **Removed instance:**

2020-12-19 11:41:22 UTC+0100	WARN	Environment health has transitioned from Ok to Warning. No data received from 1 out of 3 instances.
2020-12-19 11:41:22 UTC+0100	INFO	Removed instance [i-0298929060eb122b1] from your environment.

For RDS:
I created an Event subscription that notified me when a database changed or stopped or deleted...

Also i made a cloudWatch alarm that triggered when my monthly billing exceeded a specific number, in my case i chose 5$.

The cost to host a high availability website

As you see I used in my architecture **AWS Elastic Beanstalk(ELB),** you pay for AWS resources (e.g. EC2 instances, S3 buckets, RDS and EFS Storage size) you create to store and run your application. You only pay for what you use, as you use it, there are no minimum fees and no upfront commitments.

By: aerragha
Link: http://d3v1krrl8cvgep.cloudfront.net/