Goal

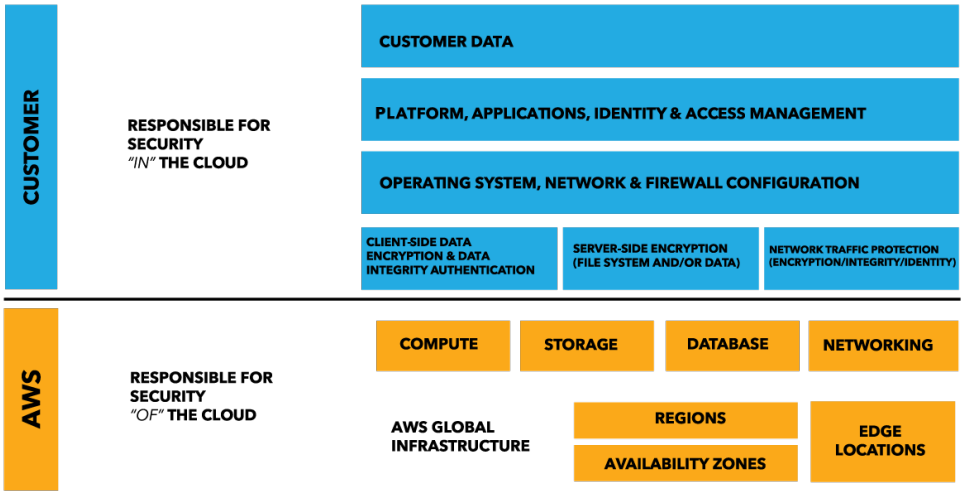
Our goal is to achieve a highly available and scalable MyBB Deployment on AWS.
The solution have to meet the following requirements:

Fully Automated

By creating a Cloudformation Template we can automate the creation of AWS Resources required to deploy our application

Security

Based on the Responsibility Sharing Model from AWS, It is our responsibility to secure our created resources, network, application, platform and data



We focused here securing our resources and network, this will be explained later in the architecture design

Highly Available

We want to achieve maximum availability and avoid down times with our solution, this depends on our architecture and network.

Scalability & Self Healing

At pick time we must be able to handle traffics spikes, this must be an automated process that doesn't require any human intervention.

Monitoring & Auditing

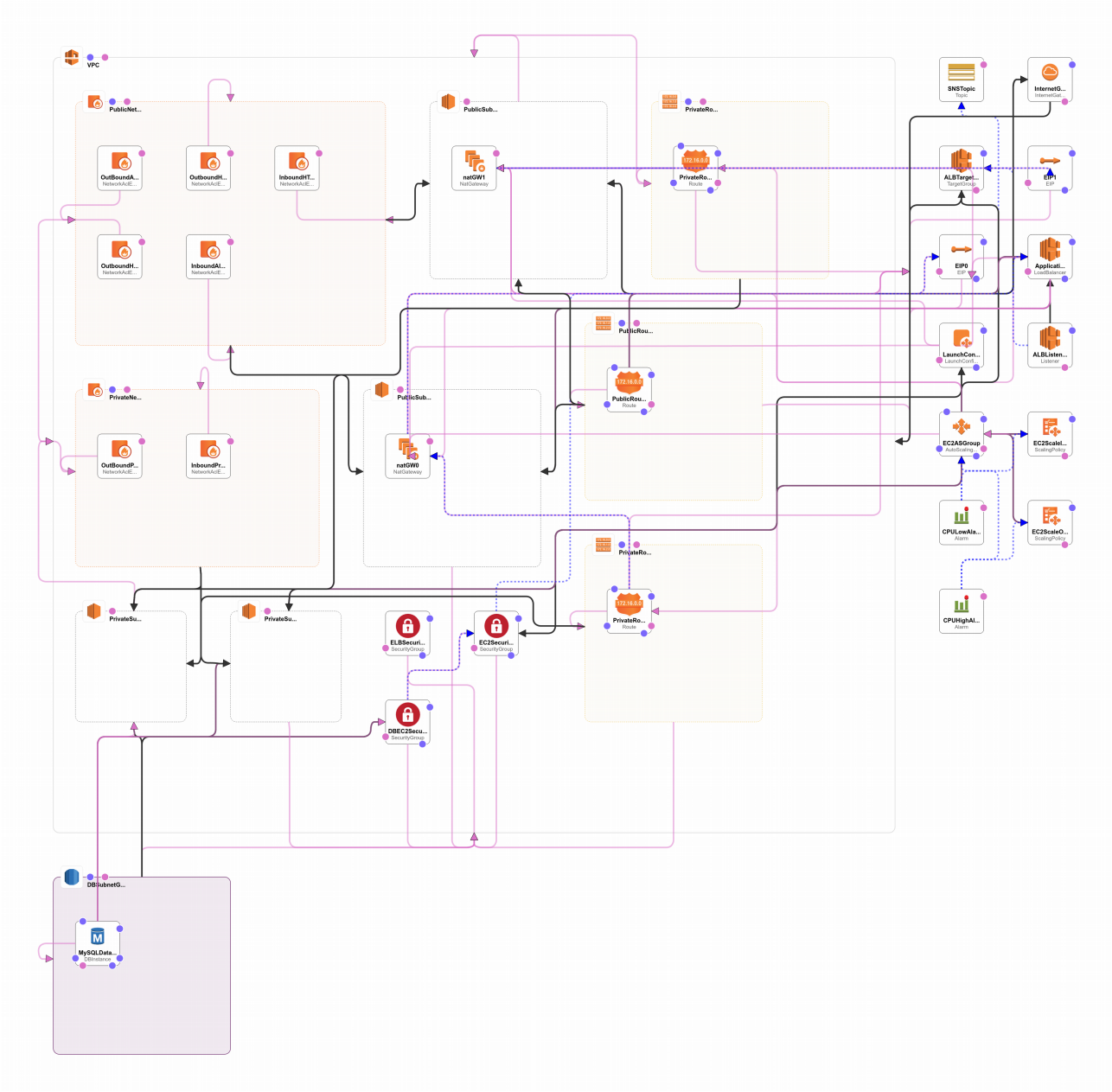
We want to be sure that our system is healthy and working fine, thus, we need to use some monitoring tools (mainly CloudWatch and Route 53) combined with SNS to notify us whenever something critical occurs.

Architecture

Our architecture consists of the following resources
- VPC with 4 Subnets
- Minimum of 2 EC2 Instances scaled automatically with EC2 AutoScaling
- Application Elastic Load Balancer
- Multi-AZ RDS MySQL Cluster

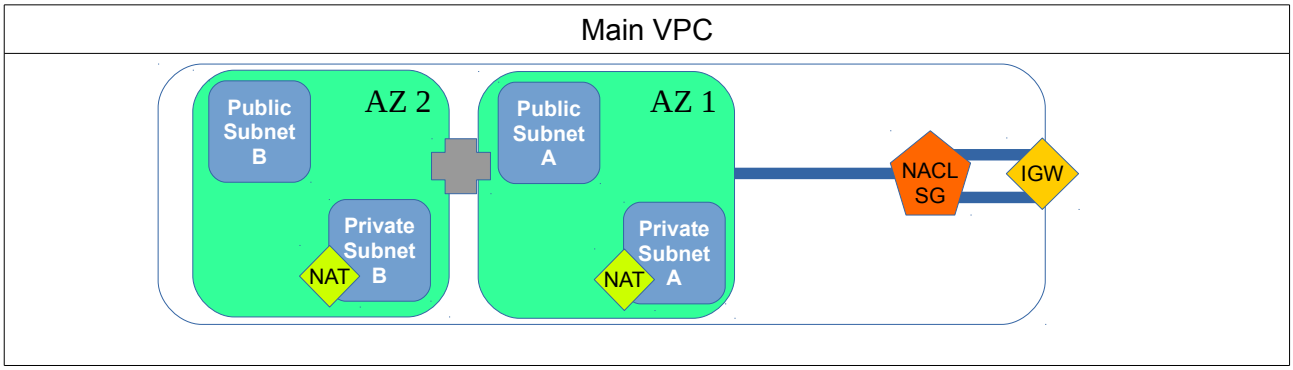
All resources must be launched or created within the VPC.

Overview



VPC

Our VPC contents 4 Subnets spreads on 2 Availability Zones
- 2 Public Subnets
- 2 Private Subnets



The VPC is connected to the Internet via Internet Gateway (IGW), however private subnets are configured to not accept incoming traffics but they are able to send traffic to the Internet via a NAT Gateway. This imported the we must try to isolate our resources and make only things public that safe to access by everyone and not critical to us.

We also used Network ACL and Security Group to filter traffic and restrict access to certain ports This should satisfy our security requirements

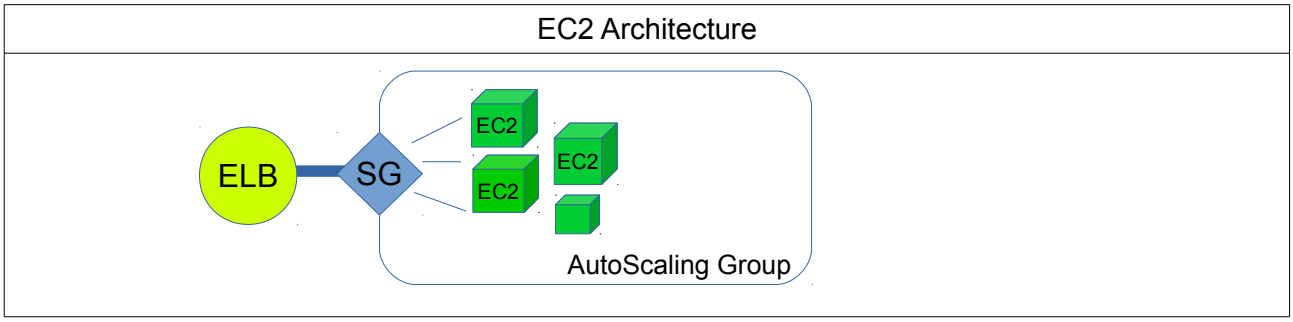
Application Architecture

EC2

In order to make our system highly available and scalable we decided to use EC2 with AutoScaling and Application Elastic Load Balancer.

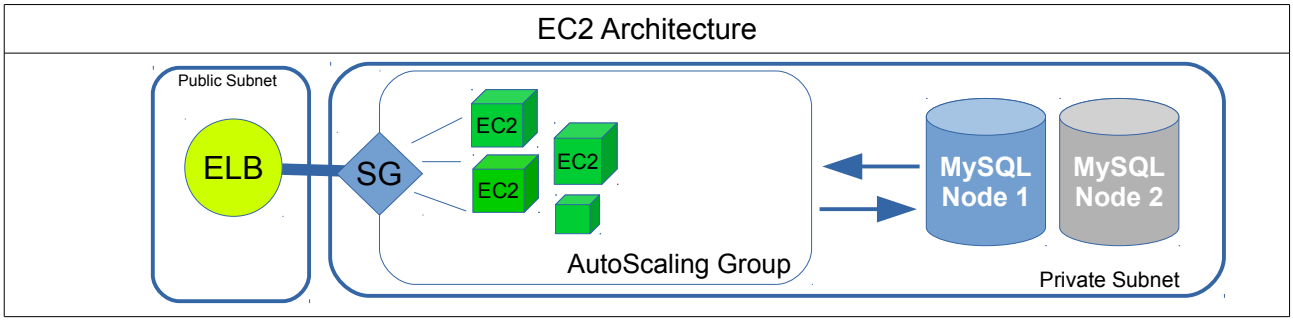
We setup an AutoScaling Group with the following configurations with a minimum of 2 instances, a minimum of 1 instance per Availability Zone.
Then we attached it to an Application Elastic Load Balancer. ELB is a highly available and scalable managed Load Balancer. The Auto Scaling Group will launch instances in the private subnet, we made sure that only the ELB can forward **only HTTP traffic** to the instances thanks the security group. The ELB will spread the traffics evenly on the instances in both availability zones. We also enabled cookies sickness due to the kind of our application (MyBB)

Thanks to EC2 AutoScaling and ELB our system is self-healing, whenever an instance is no more healthy the ELB will stop sending traffic to it and AutoScaling will replace it and deploy our application again on a healthy instance automatically.



RDS (MySQL Database)

RDS is a fully managed Database Service, with RDS we created our MySQL Multi-AZ Cluster. RDS launches 2 database nodes per availability zone and take care about failover mechanisms and availability. We made sure that our RDS resides in the private subnets and only accessible by the EC2 Instances launched by the AutoScaling.



Monitoring and Notification

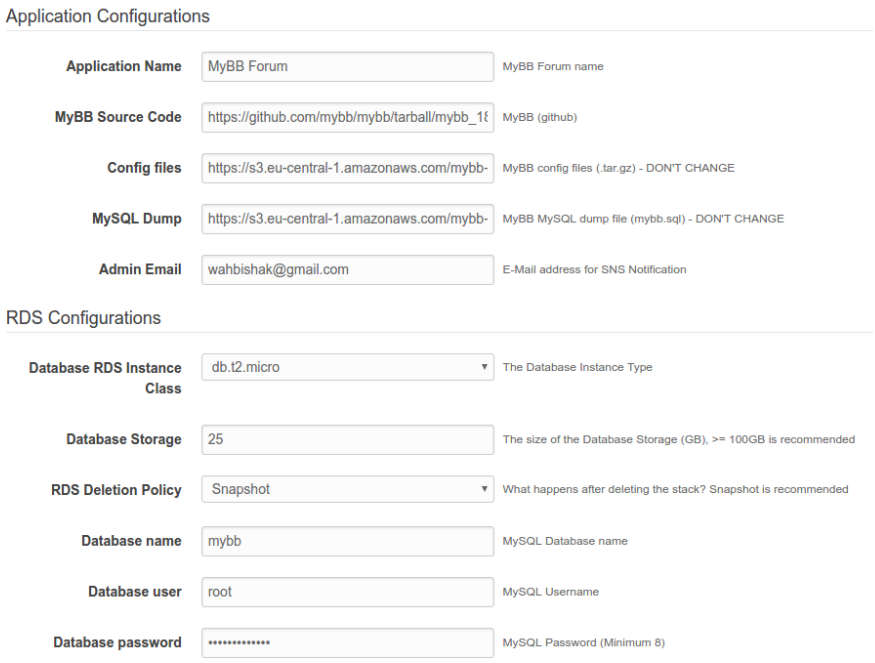
We setup CloudWatch alert to notify us on
- Scaling In (when new instances are created within our the AutoScaling Group
- Scaling errors, this can due lack of capacity within an Availability Zone or reaching our limits that could be increased by a request or due to a broken deployment.

We will be notified by SNS via Email. The user can configure the email on stack creation.

We can also add more monitoring alerts such as ELB Latency, ELB 5XX Counts, CPU Utilization.

Automation

In order to create all the resources we created a CloudFormation template. We can also use versioning tools to keep track on template updates. The Stack is configurable via CloudFormation.



Improvement potential

We could also use CodeDeploy together with CodePipeline to improve application update deployments. CodeDeploy and CodePipeline are considered CI and CD tools that are very recommended to use. However this wasn't required in the task.

MyBB is using memcache, currently we have it hosted locally on the EC2, but it is worth to think to use ElastiCache Memcache, which is a fully managed highly available and scalable Cache service, but this depends on many factors such costs and necessity.

To reduce load on our EC2 instances, we can use Cloudfront to cache images and static contents. Cloudfront will cache static contents and make them available at edge networks around the world.

It is also worth to note that we can use SSL Certificates provided for free by AWS through the AWS Certificate Manager and attach it to ELB or CloudFront.

System Management Tool

As we are going to use DynamoDB it is worth to know that DynamoDB can be used via a JavaScript SDK. This allows us to write our application using JavaScript and host it on S3, which is highly available and scalable by default and we don’t need to care about it. We can use Cloudfront if necessary but it will be worth it anyway as we can use HTTPS (SSL) when we attach our S3 Bucket to Cloudfront.

But we MUST use Web Identity Provider and STS to provide temporary credentials to authenticate to the system to avoid hard-coding aws credentials in order to manage our AWS resources and access data on DynamoDB. We need to configure different Roles and Policies to achieve this goal.

This is known as ***serverless*** architecture.