

AWS SOLUTION ARCHITECT

Client report

Author
Niall CREECH

October 10, 2019

Contents

1	Analysis	2
1.1	Overview	2
1.2	Troubleshooting	3
1.2.1	The load balancers healthcheck is unhealthy	3
1.2.2	The load balancer cannot route to subnet	4
1.2.3	The load balancer cannot be accessed from external addresses	6
1.2.4	Instances cannot be accessed from the load balancer	6
1.2.5	Success!	7
1.3	Summary	8
2	Enhancement	9
2.1	Overview	9
2.2	Pillars	10
2.2.1	Operational	10
2.2.2	Security	10
2.2.3	Reliability	10
2.2.4	Performance	11
2.2.5	Cost	11
2.3	Summary	11
3	Expansion	12
3.1	Overview	12
3.2	Pillars	13
3.2.1	Operational	13
3.2.2	Security	13
3.2.3	Reliability	13
3.2.4	Performance	14
3.2.5	Cost	14
3.3	Summary	14

1 | Analysis

1.1 Overview

The current implementation cannot be accessed from external networks. Here we will go through the current setup, look at the issues that are preventing successful access by customers, and capture the minimum changes to return the site to working health.

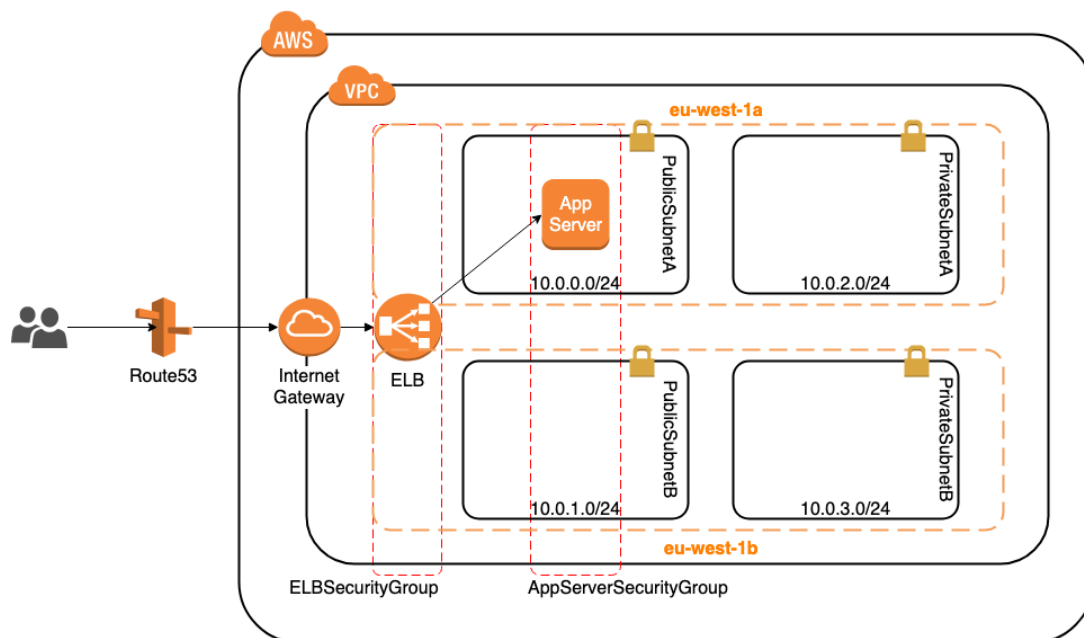


Figure 1.1: Architecture diagram of the current site

Summary of the setup of the initial solution (Figure 1.1)

- (A) The networking VPC has CIDR 10.0.0.0/16 with 2 public subnets 10.0.0.0/24 and 10.0.1.0/24 in availability zones eu-west-1a and eu-west-1b respectively, as well as 2 private subnets 10.0.2.0/24, 10.0.3.0/24 in eu-west-1a and eu-west-1b.
- (B) The VPC has an internet gateway and a classic load balancer (ELB)
- (C) NACL are setup up to allow all traffic between subnets and out to external addresses.
- (D) There are 2 security groups, one for the single EC2 application instance, and one for the ELB.
- (E) DNS is enabled and the instance has a public IP address.

1.2 Troubleshooting

1.2.1 The load balancers healthcheck is unhealthy

The EC2 instance runs a user-data script that installs and starts the Apache webserver `httpd` on port 80. The Elastic Load Balancer (ELB) correctly forwards ports from from 80 to instance port 80. However, the load balancers healthcheck looks to verify instance health on port 443 through a TCP connection, and not on port 80 where the process is actually running. The simplest method to resolve this is to change the ELB healthcheck port from 443 to 80. See <https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/elb-troubleshooting.html>.

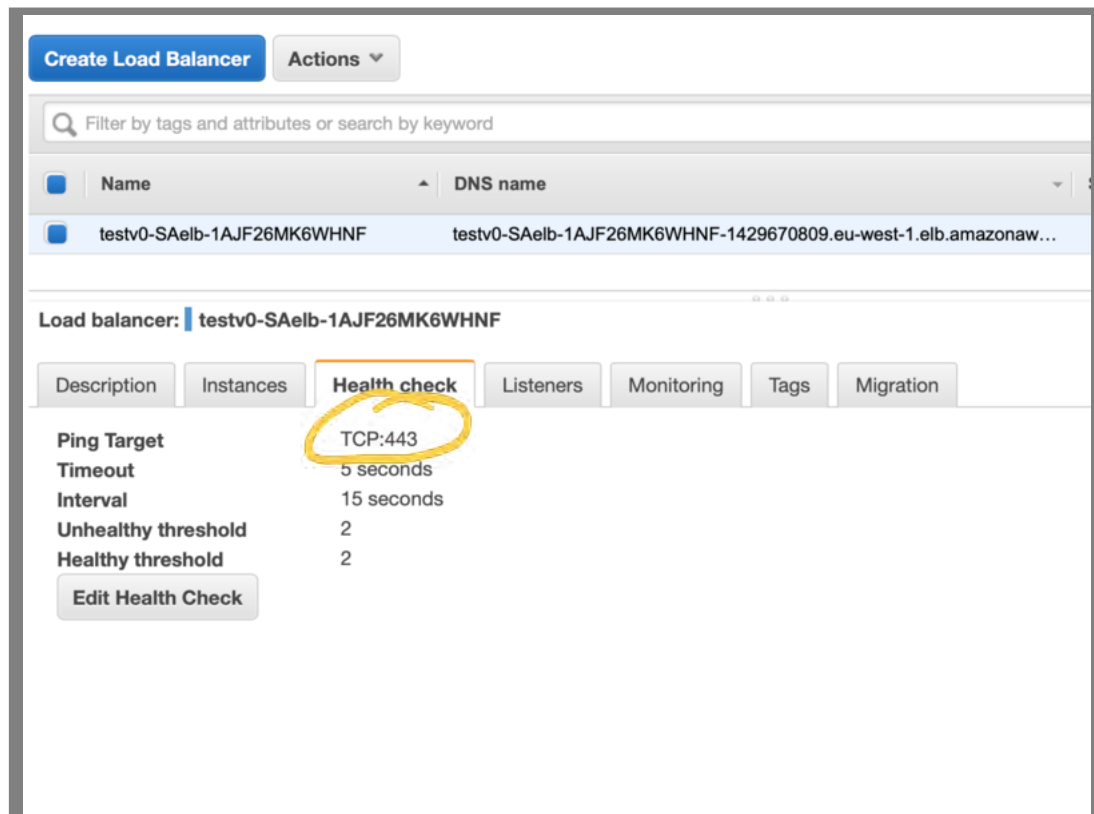
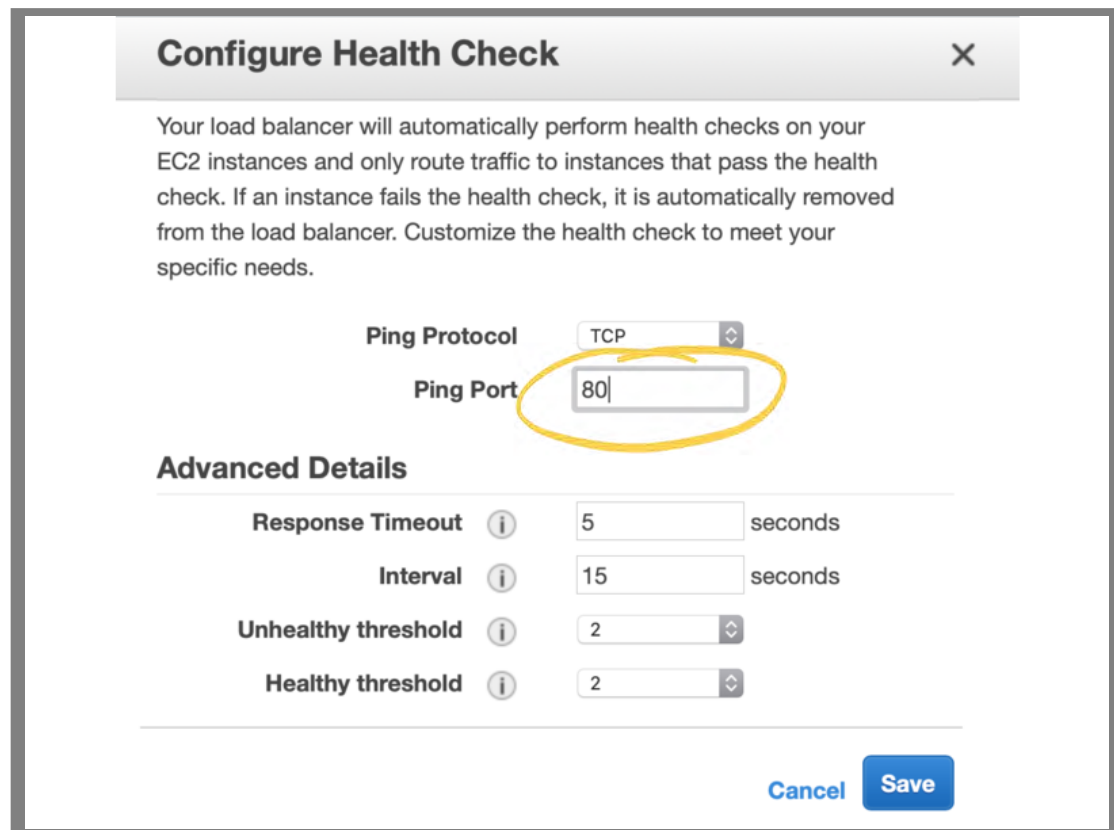


Figure 1.2: In the AWS console, navigate to Services → EC2 → Load Balancers and select the 'Health check' tab.



Configure Health Check ✕

Your load balancer will automatically perform health checks on your EC2 instances and only route traffic to instances that pass the health check. If an instance fails the health check, it is automatically removed from the load balancer. Customize the health check to meet your specific needs.

Ping Protocol TCP

Ping Port 80

Advanced Details

Response Timeout ⓘ	5	seconds
Interval ⓘ	15	seconds
Unhealthy threshold ⓘ	2	
Healthy threshold ⓘ	2	

[Cancel](#) [Save](#)

Figure 1.3: Change the ping port from 443 to 80. The load balancer will now check the instance health by connecting through TCP on the httpd servers default HTTP port.

1.2.2 The load balancer cannot route to subnet

The load balancer can only route to instances with IP addresses in `PublicSubnetB`. However, the instance currently running the application is in `PublicSubnetA` so the ELB cannot pass traffic to it. By adding `PublicSubnetA` to the load balancers available subnets the instance can have traffic routed through to it.

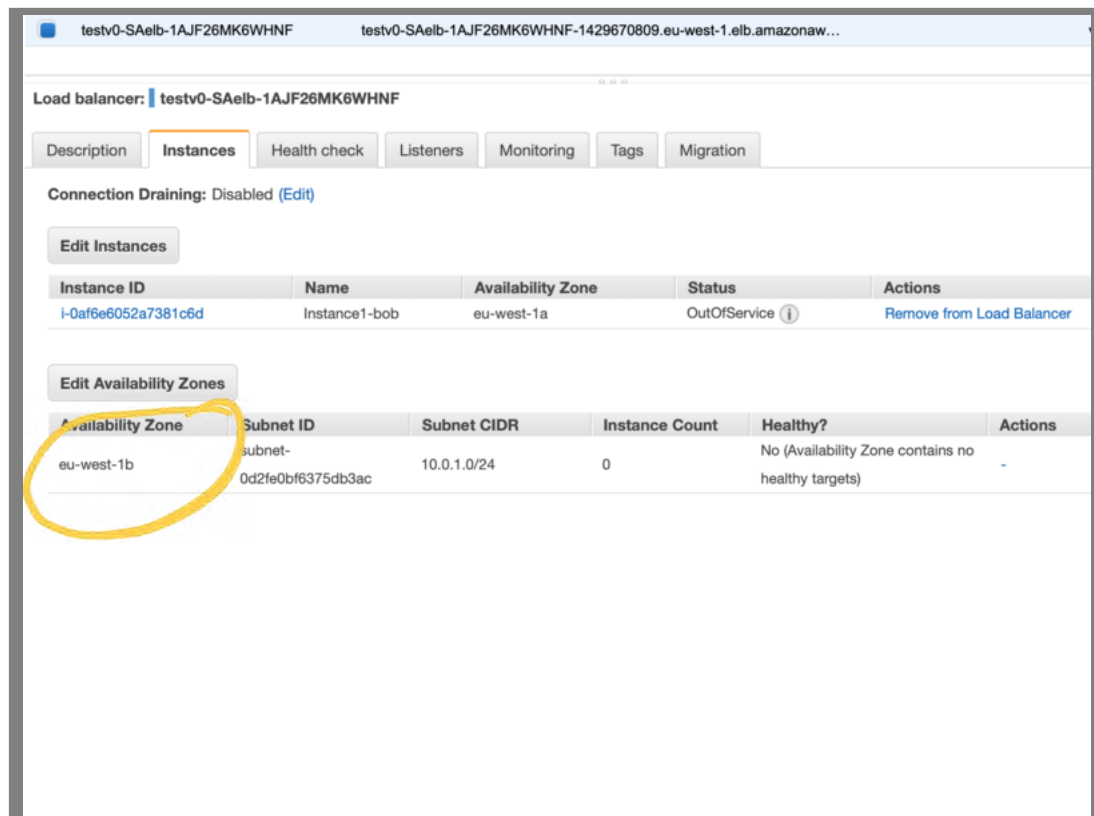


Figure 1.4: In the Load Balancers configuration page, select the load balancer and then the 'Instances' tab. Note that only the eu-west-1b zone is available to the load balancer, which has no instances

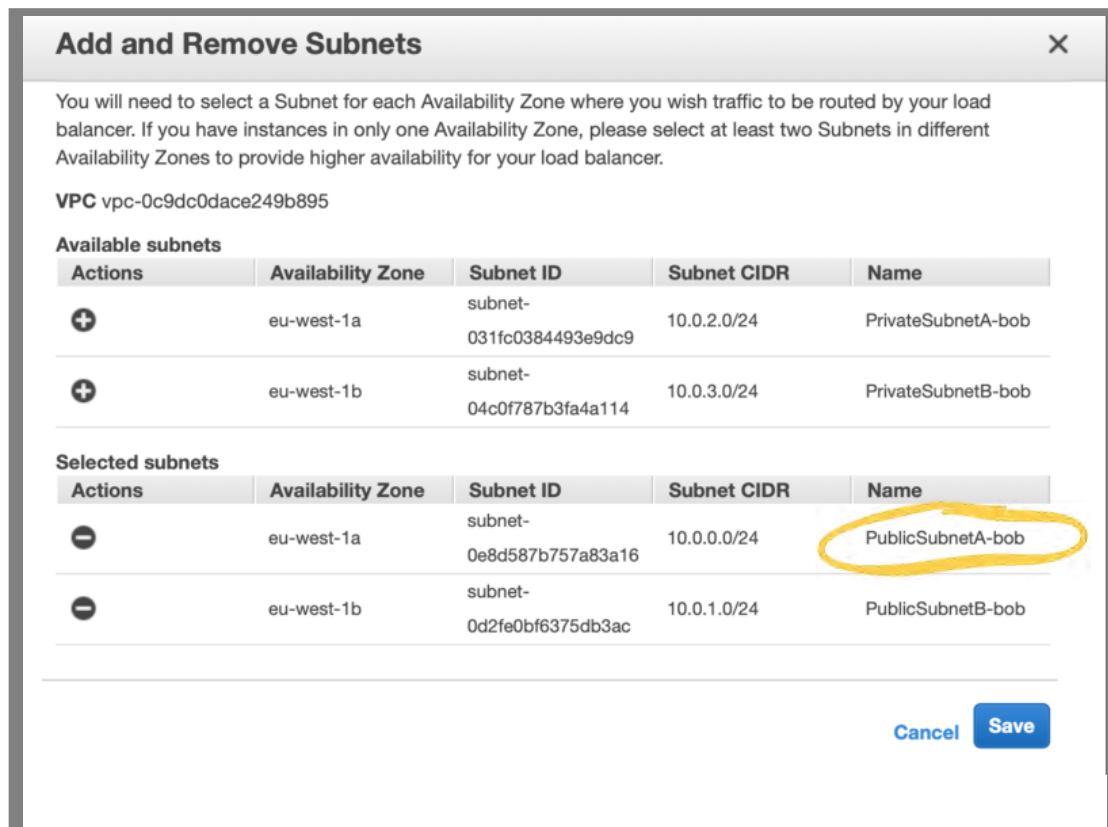


Figure 1.5: Add PublicSubnetA to the load balancers available subnets. This means that the load balancer can now direct traffic to the instance in eu-west-1a

1.2.3 The load balancer cannot be accessed from external addresses

There are no inbound security group rules for the group ELBSecurityGroup, which means that the ELB will not route any traffic to the instance. As we want the site to be accessible by everyone, we allow all addresses, by using the CIDR range 0.0.0.0/0, to access the load balancer on port 80.

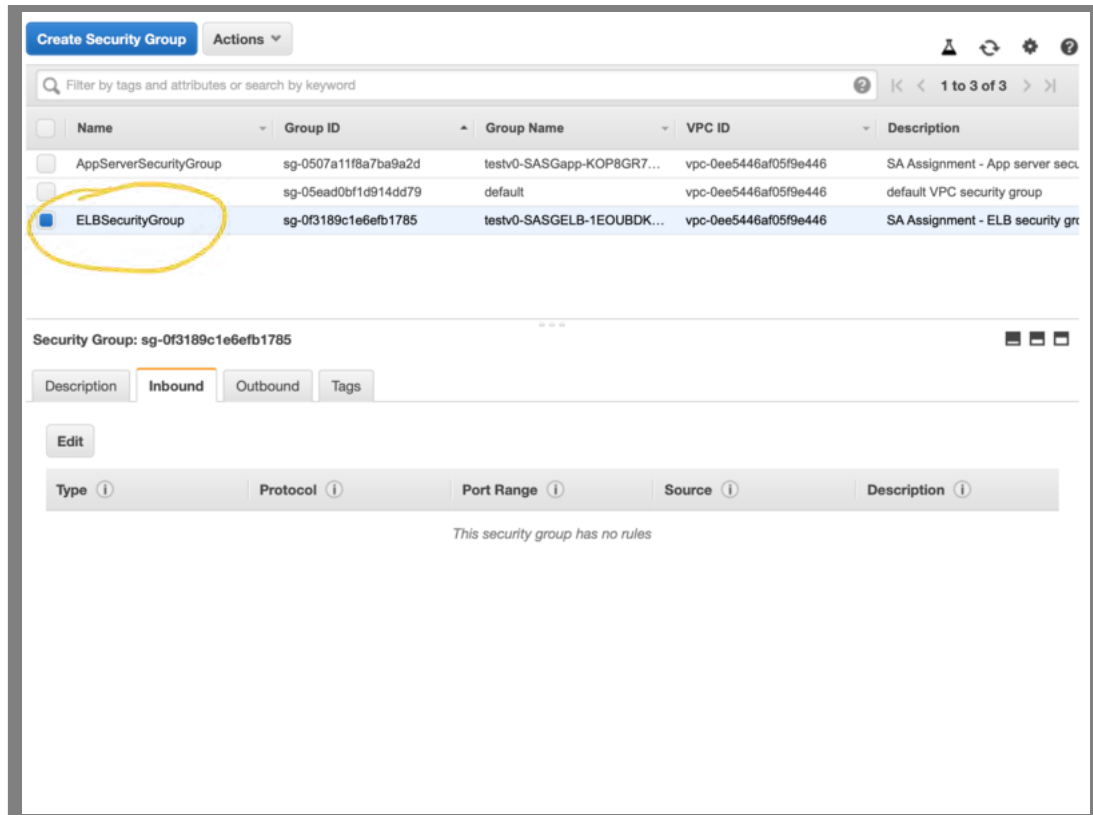


Figure 1.6: Navigate to Services → EC2 → Security Groups. Select the ELBSecurityGroup group and the 'Inbound' tab. By default, no inbound rules are set.

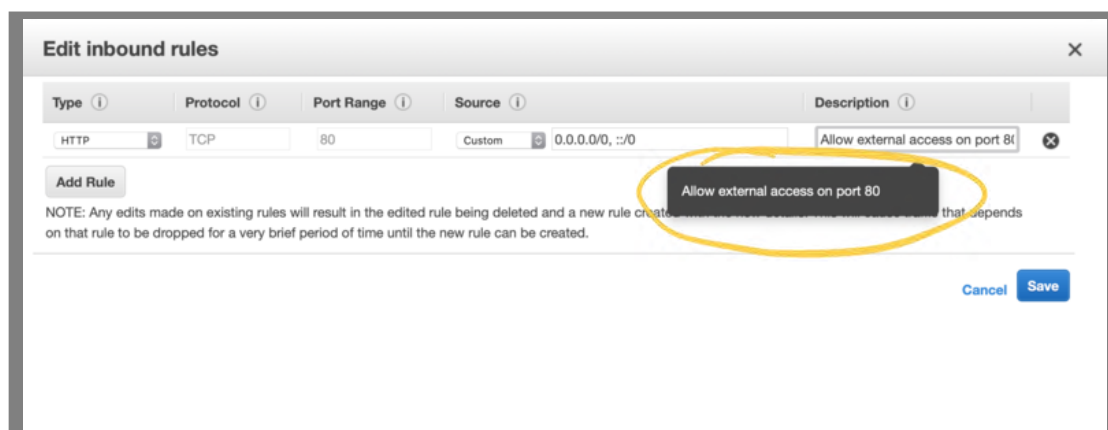


Figure 1.7: Edit the group and add an inbound rule to allow HTTP access from everywhere, 0.0.0.0/0. Users will now be able to reach the load balancer from the internet and other external networks.

1.2.4 Instances cannot be accessed from the load balancer

The current security group for the EC2 instances does not contain any inbound rules. This means that traffic from the load balancer is blocked. We can change the security group to allow traffic from the

ELB to the instance by adding a rule to allow TCP traffic to port 80. This also allows the ELBs TCP-based healthcheck to pass.

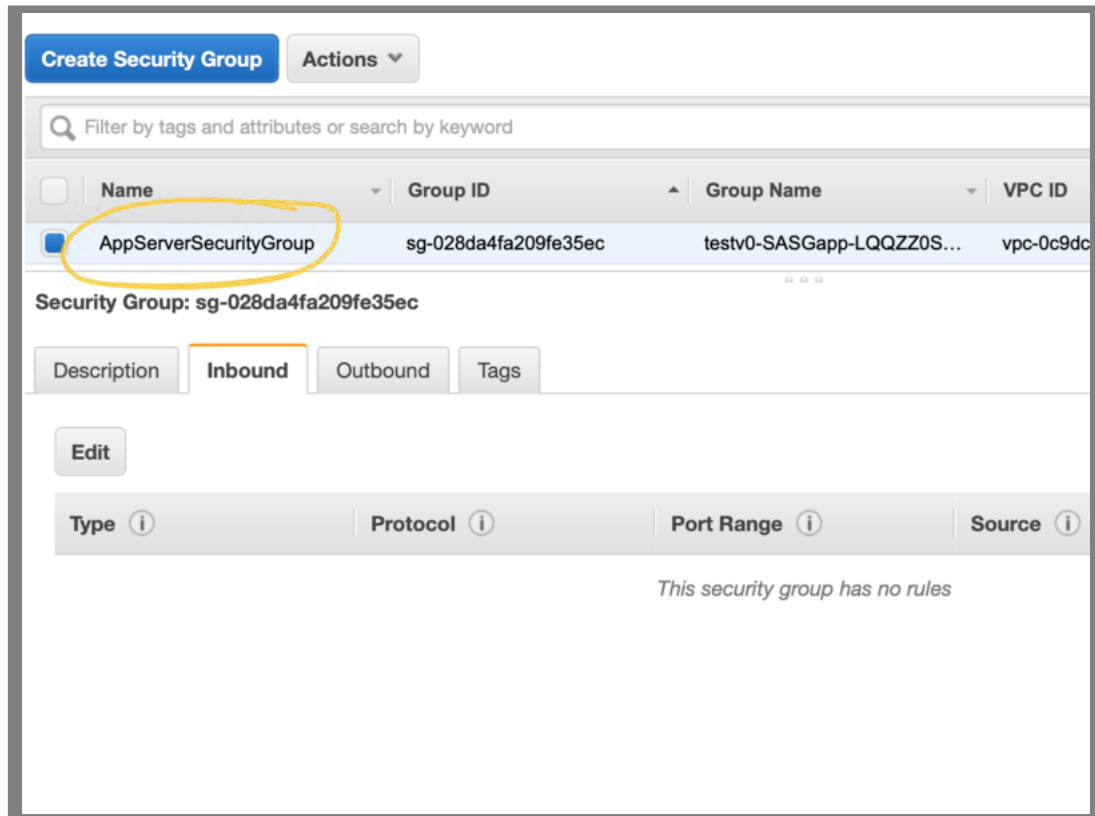


Figure 1.8: Still in the 'Security Groups' section, select the AppServerSecurityGroup group and the 'Inbound' tab. As we can see, there are no rules set.

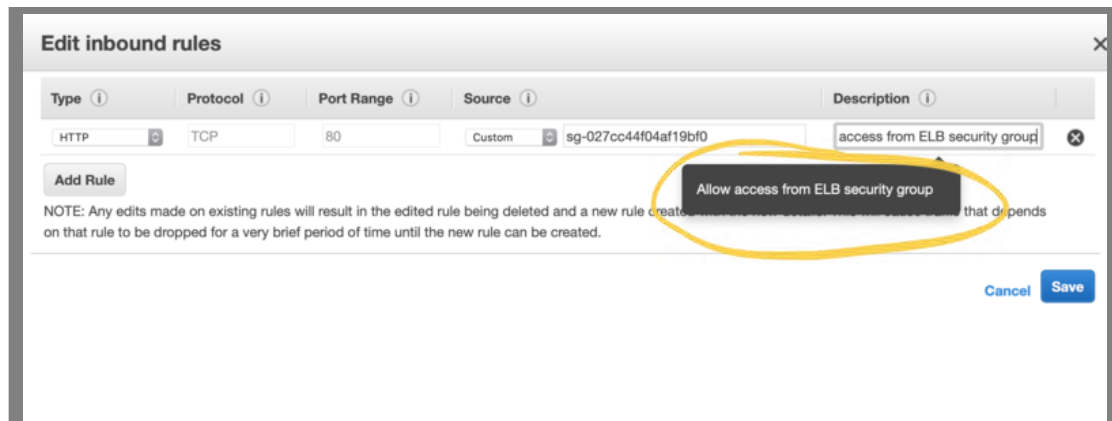


Figure 1.9: Edit the group and add an inbound rule to allow HTTP access from the ELBSecurityGroup. The security group Id needed for the 'Source' value can be found by typing 'sg' then choosing from one of the values in the popup menu.

1.2.5 Success!

The DNS name for the site can be found on the 'Description' tab of the 'Load balancers' configuration page when the load balancer is selected.

Website: `http://<load-balancer-dns-name>/demo.html`

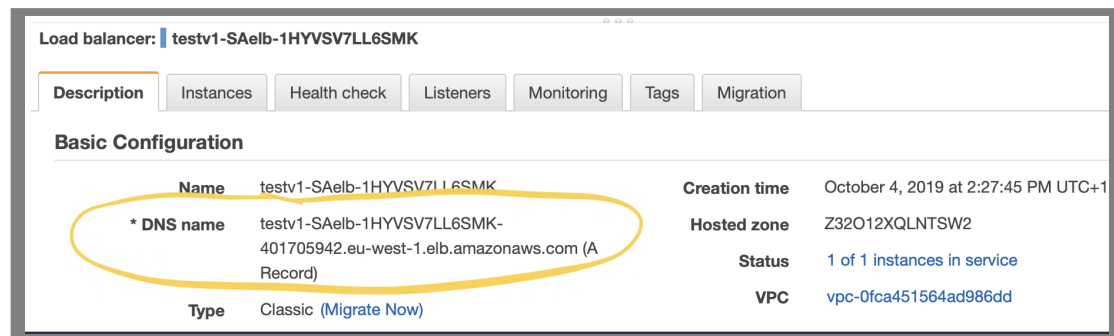


Figure 1.10: The DNS name of the load balancer can be found in the load balancers description tab

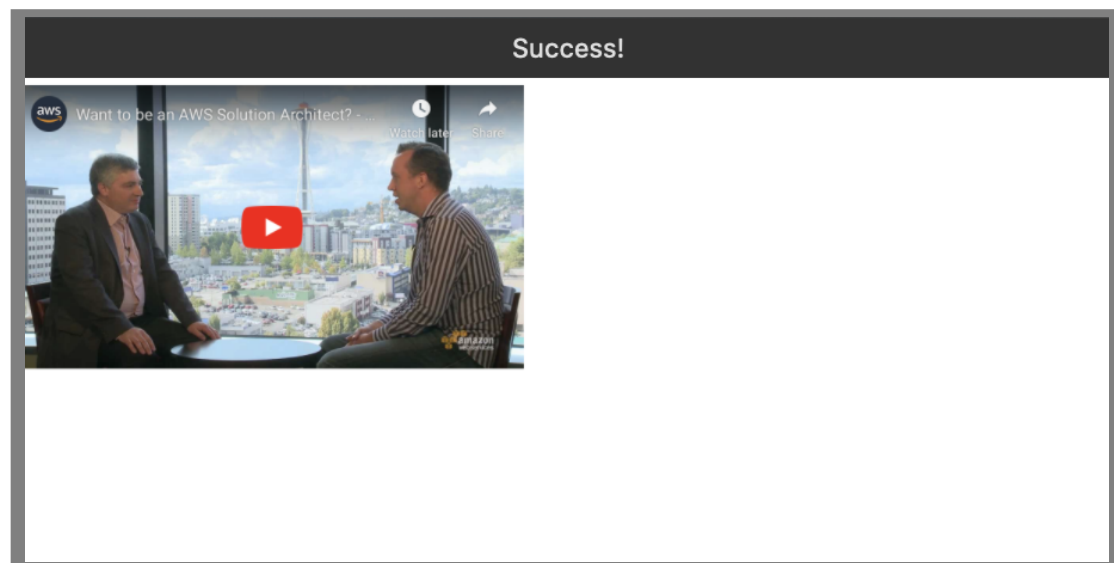


Figure 1.11: The instances are now reachable to customers on the internet

1.3 Summary

This walkthrough covers the issues that are blocking customers accessing the site. Following the step-by-step instructions will ensure that the site is reachable. However, these represent the minimal changes needed and not necessarily all those that would be recommended for a production site. These changes should also ideally be implemented through updating the clients infrastructure-as-code such as CloudFormation or Terraform as applicable. Next we will look at some other changes that will greatly improve the quality of the infrastructure.

2 | Enhancement

2.1 Overview

There are a number of immediate improvements we can make to the initial solution that will bring benefits in reliability, security, and observability. These improvements are small and iterative but have a large impact on the production performance.

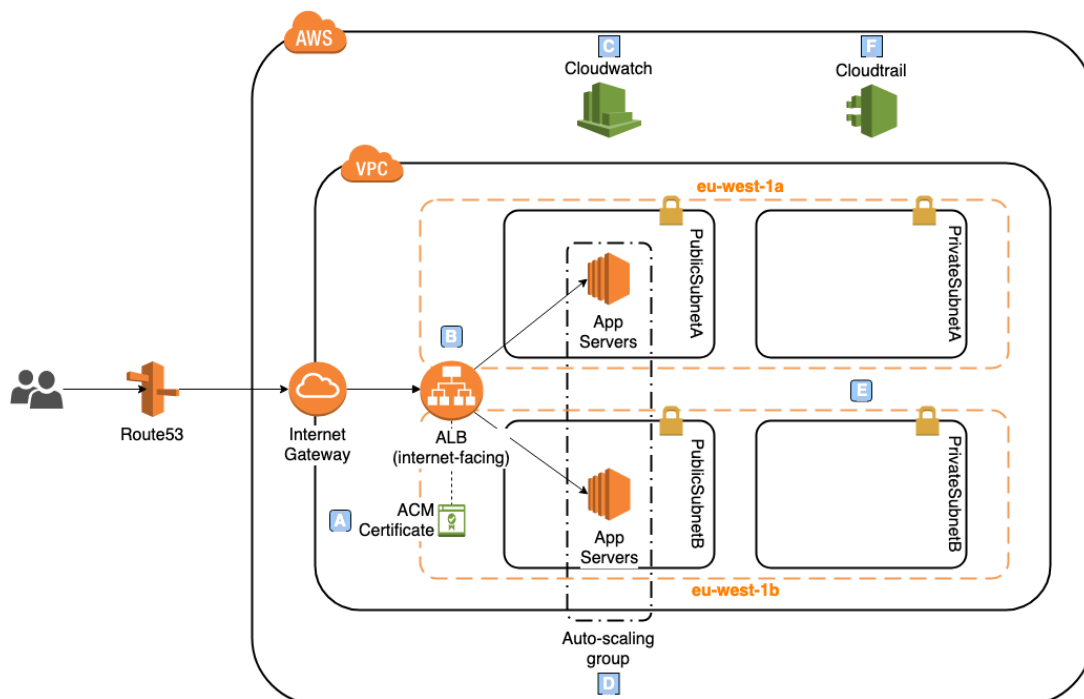


Figure 2.1: Architecture diagram of an improved solution.

Changes and enhancements to the initial solution (Figure 2.1)

- (A) Addition of ACM certificate to load balancer to allow HTTPS to be enabled [5].
- (B) Moving to Application Load Balancer(ALB) for HTTP->HTTPS redirection and layer 7 routing and WAF filtering [12].
- (C) Enable detailed instance and load balancer metrics, shipping to Cloudwatch with dashboard. Logs and OS-level metrics through `collectd` and Cloudwatch agent installation on the applications AMI [1].
- (D) Instances placed in Auto-scaling group (ASG) for automatic management and healing of instances using load balancer HTTP:80 healthcheck [2].
- (E) Instances are replicated into multiple availability zones.
- (F) Cloudtrail is enabled in the AWS account to log all AWS API access [7].

2.2 Pillars

2.2.1 Operational

Monitoring and logging on the existing instance can be increased by installing and enabling the cloudwatch agent on the application instances. The agent can then be set up to ship os-level metrics and application logs to Cloudwatch. Detailed metrics can also be enabled on the load balancer, auto-scaling group, and instances (C).

2.2.2 Security

By moving from a classic load balancer (ELB) to the newer layer 7 Application Load Balancer (ALB), there are gains not only in performance, but much more flexibility in routing to different targets, alongside the ability to use HTTP header information (B). By adding an automatically renewing ACM certificate we enable HTTPS support for the service. Termination of TLS on the load balancer means that the application servers themselves experience no additional load, so larger and more expensive instances are not required. We can also enable HTTP->HTTPS redirection on the load balancer, further enhancing security, and again, offloading that work from the application code.

Future placement of the AWS WAF service in front of the ALB mean that granular controls in mitigating attacks can be added at minimal extra cost. NACL are quite open currently and could be more restrictive, the private NACL allows all external traffic to pass for example. We can increase security by allowing only access from public subnets, or additionally from private endpoints, NAT gateways etc.

In addition to load balancer enhancements, the business should look at methods of hardening the EC2 instances, such as pre-baking AMI's, or moving them into private subnets and adding the NAT service to allow internet access. In this case adding the SSM agent to instances and using session manager to allow IAM users SSH access would be preferable to installing publically facing bastion servers.

Cloudtrail is enabled across all accounts and regions so that access to the AWS API can be controlled, alerted, and audited (F).

2.2.3 Reliability

In the initial solution there is a single instance with no automatic recovery in the event of failure. With the application server running in only one availability zone, in the event of network disruption, the entire service would become unavailable. By replicating EC2 instances into more than one availability zone, and enabling cross-zone load-balancing, we greatly increase the reliability of the service health (E).

Placing the instances into an auto-scaling group not only helps to ensure we have spread the application servers across availability zones, but also allows the ASG to replace failing servers with new ones based on the load balancers health check, giving the service an element of auto-healing (D).

2.2.4 Performance

With the addition of the auto-scaling group, we can add in Cloudwatch alarms to control the number of instances running based on current spare capacity. For example, we could set a CPU percentage or network utilisation target, ensuring that the ASG increased or decreased the number of instances as required to maintain that usage level.

Note, the current solution uses bursting t2.micro instances with CPU credit limits. This is low cost, but performance is not suitable for sustained traffic levels where exhausting CPU credits will cause degradation of availability. The instance type will need to be right-sized given realistic production traffic patterns, and a non-bursting instance type may be preferable.

2.2.5 Cost

The auto-scaling functionality should mean that service costs are kept minimal. Further choices should be made as the service is evaluated in production, where reserving some baseline instances or combining capacity with spot purchases as part of an EC2 fleet may be applicable as the service evolves.

2.3 Summary

The changes we've covered here will provide a robust and scalable system. Depending on the clients balance between change, cost, and performance characteristics, it would also be recommended to look at some more changes such as,

- Moving the applications into private subnets and using the public NAT instances to allow them outbound access to the internet.
- Adding a custom zone and domain name routing to the load balancer would be a requirement of a live system to give the service a friendly name.

We have also assumed here that the applications are stateless, and that no data is stored in a database as yet. The system is designed so that adding an RDS database such as Aurora MySQL/Postgres will be simple. If the applications develop to become stateful then additional options such as an S3 bucket or EFS filesystem to preserve state for instances as they are terminated or created. Volume snapshots may be needed depending on the needs of the application and use of EBS volumes.

3 | Expansion

3.1 Overview

Building on the initial solution but with the use of a number of additional AWS services, we can make the site scale globally and meet the performance demands of a high capacity business.

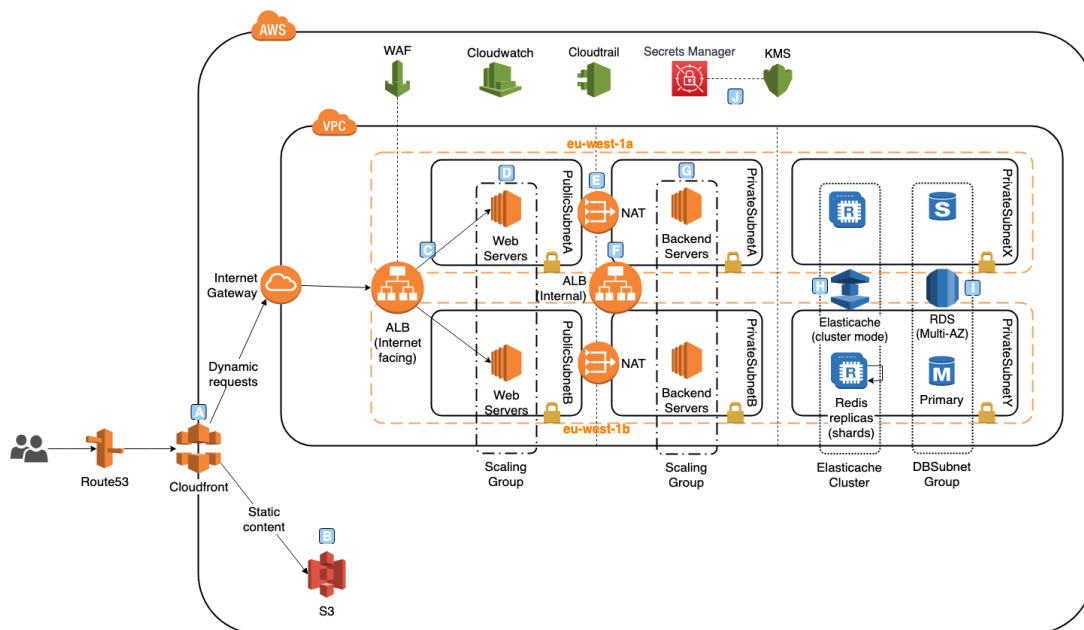


Figure 3.1: Large scalable production solution

Global scale improvements

- (A) Cloudfront provides cached content from edge locations around the world [6].
- (B) A secured S3 bucket provides a highly scalable source of static content
- (C) Active content is sourced through a public application load balancer (ALB).
- (D) WAF is configured with rules in front of the ALB [9, 10, 11]
- (E) Applications requiring public internet addresses are installed on EC2 instances controlled by an auto-scaling group (ASG) in public subnets distributed across more than one availability zone.
- (F) The NAT gateway service provides address translation to allow outbound internet access for the private subnets.
- (G) Applications that do not require public internet addresses are installed on EC2 instances inside an ASG in private subnets distributed across more than one availability zone.
- (H) ElastiCache (redis) provides in-memory caching for the database backend. Using multi-node with clustering helps performance, reliability, and scalability [3].
- (I) RDS in multi-AZ mode gives resilient failover from primary to secondary, as well as < 5 min point in time recovery [4].
- (J) Secrets manager works with the Key Management Service (KMS) to allow instances with the right IAM roles to access their relevant credentials [8].

3.2 Pillars

3.2.1 Operational

The addition of Cloudfront and WAF metrics means that observability of the customers usage and experiences with the site overall are covered in more depth. Additional origins can be added in, meaning that new infrastructure can be added into the main site easily.

3.2.2 Security

By moving as much of the application into non-accessible private subnets, the exposure of running services to attack is greatly reduced. In addition, strong security group and NACL rules can protect the independent application and data components. There may be options depending on the behaviour of the services moving forward to restrict access to running services further and integrate private subnets with VPC endpoint services to prevent internal traffic traversing networks outside of AWS.

Cloudfront can also be set to filter requests, allow only certain request types, filter cookies, etc. Meaning there the site can be restricted to the smallest set of requests that it needs to be functional for the end-users.

Although bastion servers are an option, access to private servers could be added through the SSM session manager and IAM user access. This not only removes the need to run publicly exposed bastion servers, but also means that access can be logged and audited through Cloudtrail.

Secrets access is kept secure through AWS Secrets manager, allowing services to have granular access to encrypted data during runtime. There is also the option of regularly rotating credentials such as database passwords.

3.2.3 Reliability

The caching provided by Cloudfront means that the site can remain functional if degraded, for users even in the event of the servers becoming unavailable. Finely controlled fallback to a static version of the site or other sources through the use of origin groups.

The use of ElastiCache Redis in multi-node with clustering enabled means that data is sharded and those shards replicated across AZs, adding resilience.

3.2.4 Performance

The use of Cloudfront for static assets such as images, videos, and Javascript, means that the performance and network transfer overhead is moved from the EC2 instances onto globally distributed edge locations. This gives greater responsiveness for users and higher availability assurances. With the ElastiCache clustering, we can have multiple sources of each data shard cached from the database, improving throughput.

3.2.5 Cost

Moving assets to edge locations reduces bandwidth costs for moving data through the network, as well as reducing the needed EC2 compute power to run the service. Although we add cost through having ElastiCache in multi-nodes setup, we constrain this by clustering so that we can horizontally scale our cache, rather than buying more expensive instance types. The cost-performance trade-off will need to be balanced dependent on the customer need and requirements.

3.3 Summary

This possible configuration builds upon the initial basic implementation and incorporates features that support a globally scalable customer service. The infrastructure costs are kept minimal, with costs scaling efficiently with traffic increase. The use of CDN, database caching, and auto-scaling mean that customer response times and availability are maintained even with high rates of customer engagement, even given a variety of traffic across multimedia, static, and dynamic requests.

Bibliography

- [1] Amazon Web Services. Amazon cloudwatch logs. <http://docs.aws.amazon.com/AmazonCloudWatchLogs/latest/APIReference/cwl-api.pdf>, 2019.
- [2] Amazon Web Services. Amazon ec2 auto scaling. <http://docs.aws.amazon.com/autoscaling/ec2/APIReference/as-api.pdf>, 2019.
- [3] Amazon Web Services. Amazon elasticache for redis. <http://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/redis-ug.pdf>, 2019.
- [4] Amazon Web Services. Amazon relational database service. <http://docs.aws.amazon.com/AmazonRDS/latest/APIReference/rds-api.pdf>, 2019.
- [5] Amazon Web Services. Aws certificate manager. <http://docs.aws.amazon.com/acm/latest/APIReference/acm-apiref.pdf>, 2019.
- [6] Amazon Web Services. Aws cloudformation. <http://docs.aws.amazon.com/AWSCloudFormation/latest/APIReference/cfn-api.pdf>, 2019.
- [7] Amazon Web Services. Aws cloudtrail. <http://docs.aws.amazon.com/awscloudtrail/latest/APIReference/awscloudtrail-api.pdf>, 2019.
- [8] Amazon Web Services. Aws secrets manager. <http://docs.aws.amazon.com/secretsmanager/latest/userguide/secretsmanager-userguide.pdf>, 2019.
- [9] Amazon Web Services. Aws shield advanced. <http://docs.aws.amazon.com/waf/latest/DDOSAPIReference/AWS-Shield.pdf>, 2019.
- [10] Amazon Web Services. Aws waf. <http://docs.aws.amazon.com/waf/latest/APIReference/waf-apiref.pdf>, 2019.
- [11] Amazon Web Services. Aws waf, aws firewall manager, and aws shield advanced. <http://docs.aws.amazon.com/waf/latest/developerguide/waf-dg.pdf>, 2019.
- [12] Amazon Web Services. Elastic load balancing. <http://docs.aws.amazon.com/elasticloadbalancing/2012-06-01/APIReference/elb-api.pdf>, 2019.