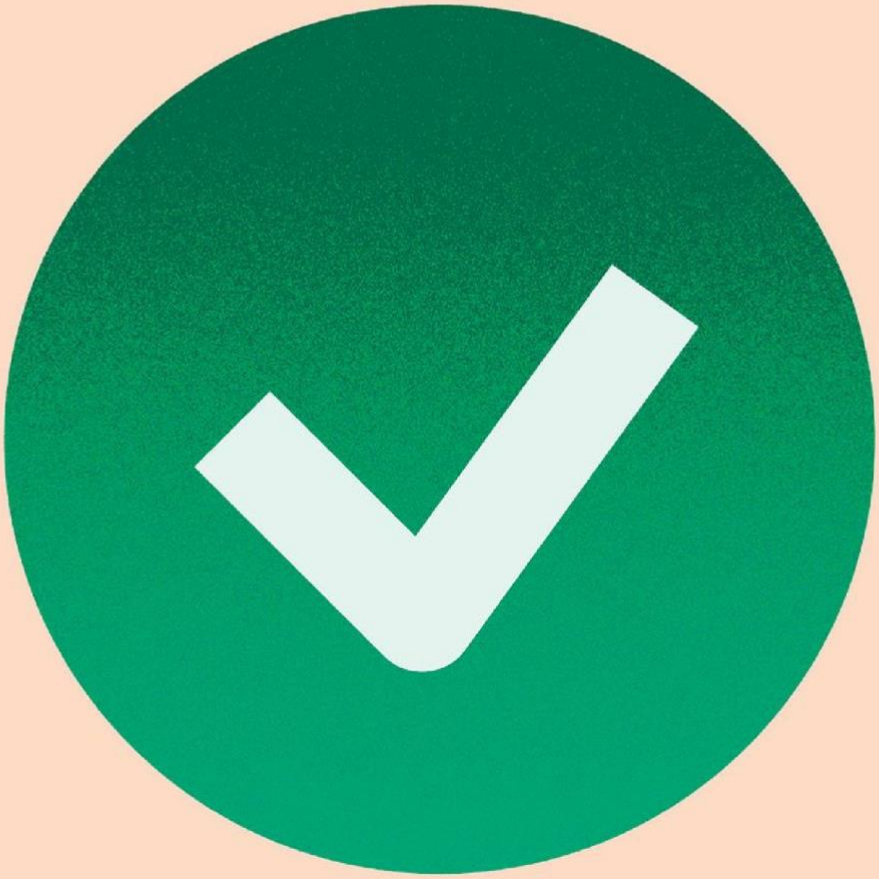
Queue·it

High availability



Queue·it

Table of Contents

1. Introduction

The Queue-it High Availability white paper outlines how our infrastructure is built for high performance during massive load and ensures reliable uptime and continuous operation across all AWS availability zones.

This document covers the strategies we use to ensure high availability and throughput, including load balancing, auto- and pre-scaling, customer segments, throttling, and caching.

You'll discover how we've incorporated fault tolerance into our platform, including microservices architecture, data durability, security groups, and fire drill days. Finally, we'll review our fast recovery and backup strategy, including health checks, data redundancy, and incident management processes.

This white paper is intended for IT security architects, system architects, and solution architects with knowledge of SaaS models and AWS components.

2. High availability and throughput

AWS gives its customers services to deploy high availability resilient IT architecture. AWS has designed its systems to tolerate system or hardware failures with minimal customer impact.

Queue-it leverages AWS high availability capabilities by running Queue-it instances from multiple Amazon AWS data centers in different regions globally.

2.1 Multiple Availability Zone architecture

Queue-it deploys its solution on multiple AWS data centers to increase system availability and continue serving customers even in the event of a total failure on some datacenters.

AWS regions are designed with multiple Availability Zones (AZs). Each AZ group contains one or more discrete data centers. Each Availability Zone is designed as an independent failure zone, physically separated within a typical metropolitan region from other AZs.

Queue-it is deployed in regions with at least 3 Availability Zones. The following regions and Availability Zone are used:

- EU West (Ireland)
    - eu-west-1a
    - eu-west-1b
    - eu-west-1c
- US West (Oregon)
    - us-west-2a
    - us-west-2b
    - us-west-2c
- Asia Pacific North East (Tokyo)
    - ap-northeast-1b
    - ap-northeast-1c
    - ap-northeast-1d
- South America (São Paulo)
    - Sa-east-1
    -

This deployment allows Queue-it to be available even in the unlikely event of two availability zones being unavailable at the same time.

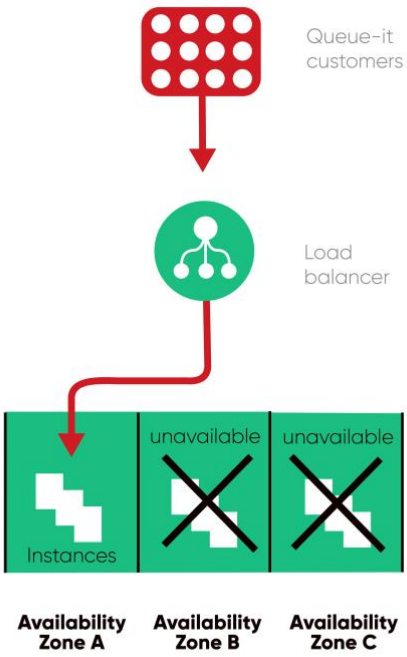


Figure 1 : Availability Zones

2.2 Application load balancers

AWS Application Load Balancers (ALBs) are region-based services and can distribute the load to virtual servers in Amazon's Elastic Compute Cloud (EC2 instances) located in any of the AZs within its region.

Queue-it uses AWS ALBs to distribute the load to servers on different AZs and automatically deregister any unhealthy server instances.

The ALB uses Queue-it health check API to monitor the health of QueueFront instances deployed on EC2. It dynamically registers and deregisters new instances, directing traffic to healthy instances.

2.3 Cell-based architecture / customer segments

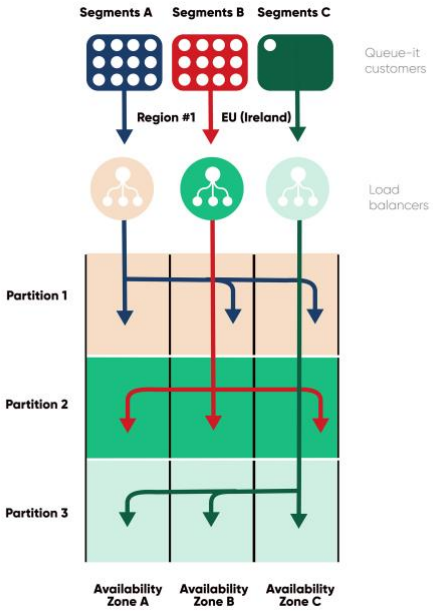
Queue-it is designed to isolate potential attacks on a specific customer waiting room from impacting other customers. Each Queue-it customer has at least one dedicated subdomain used to serve their waiting room pages (customerid.queue-it.net where “customerid” is the GO Queue-it Platform account the customer owns).

Queue-it groups customers in at least three segments per region based on their subdomains and has three independent partitions continuously running QueueFront runtime. Each customer segment is mapped to a partition with a dedicated load balancer entry point handling traffic for those customers and distributing this load on corresponding partition’s EC2 instances (Web servers running QueueFront).

All these partitions are deployed in the three Availability Zones as shown in the following diagram. This segmentation prevents one customer segment traffic from impacting customers in other segments.

For example, traffic for customers in segment A in Ireland is managed by Load balancer Par-EU-West-1-A and underlying Par-EU-West-1-A partition QueueFront EC2 instances. Traffic for customers in segment B is managed by Par-EU-West-1-B load balancer and underlying EC2 instances. So, customers in segment B are not impacted even if all partition A EC2 instances (on all three availability zones) are down.

Figure 2: Queue-it customer segments, load balancers, deployment partitions, and availability zones



In the case of a DDoS attack on a given customer segment (which could include a single customer having a high visibility sale or registration, aka event), the Queue-it operations team can quickly point this segment to a partition without a QueueFront stack deployed.

The following figure shows how a single customer having a DDoS attack is isolated on one segment of its own and mapped to a partition without QueueFront stack.

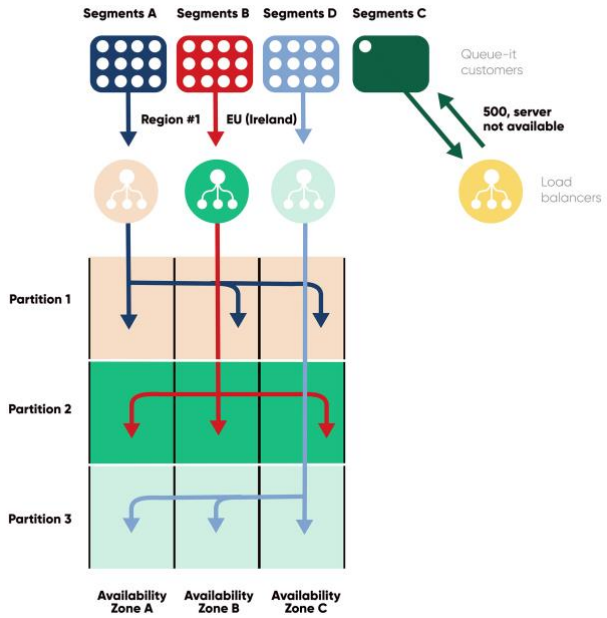


Figure 3 Sending traffic from one segment to an empty partition

Queue-it operations team has built tools to easily move a customer to a segment or map a segment of customers to a given partition shown below.

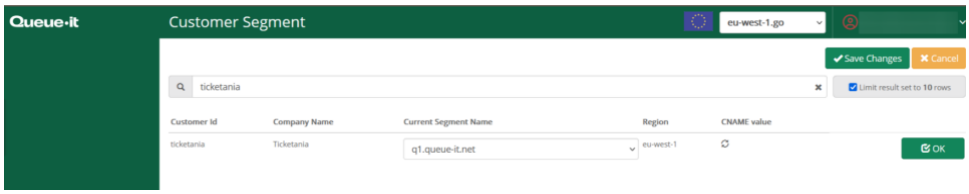


Figure 4: Mapping a customer to a segment

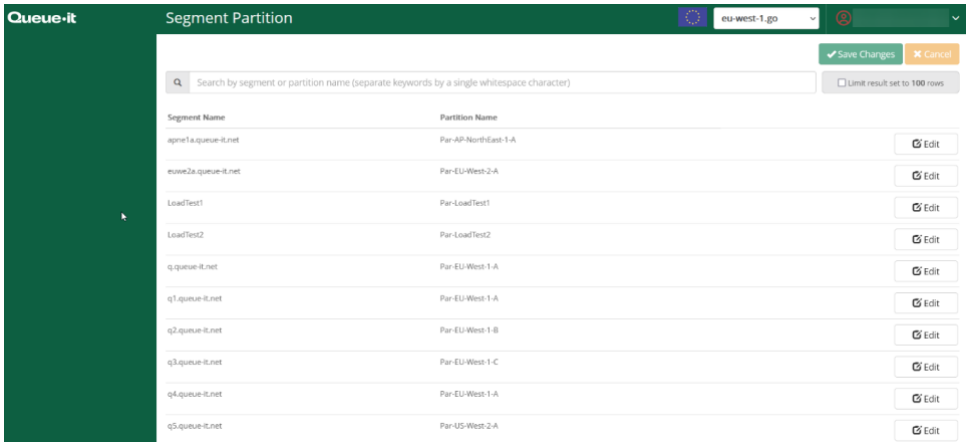


Figure 5: Mapping a customer segment to a partition

This cell-based architecture also allows Queue-it to plan and execute the deployment of new QueueFront releases in a controlled manner and reduce the risk of impacting all customers in a given region. New releases are deployed and evaluated on a single partition (impacting one customer segment) and only after successful testing the deployment is applied to other partitions.

2.4 Autoscaling and pre-scaling

Queue-it uses AWS scaling groups to scale up / down resources (like EC2 instances running QueueFront). Queue-it uses an aggressive auto-scaling strategy to automatically spawn multiple new EC2 instances to serve requests from unplanned traffic peaks. Autoscaling will be triggered based on multiple metrics of running EC2 instances. It is configured to over-provision new instances, prioritizing availability of our system over cost.

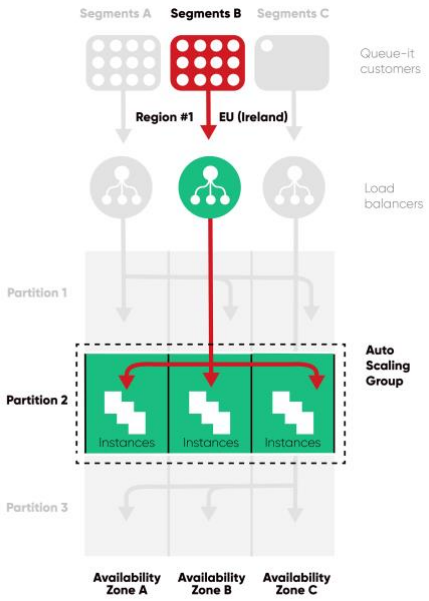


Figure 6: Autoscaling groups

In addition to auto-scaling, Queue-it works in tight collaboration with our customers to pre-scale (manually scaling horizontally and vertically) ahead of planned events with expected peaks. This allows us to manage peaks without the need to trigger autoscaling, which could take 2-5 minutes to have all new EC2 instances up and running. Queue-it has performed a load test with 1 million users joining the queue per minute for 2 subsequent minutes and the infrastructure scaled to this load resulting in 0 503 responses from ALBs and 2 million inflows correctly registered on the test waiting room.

2.5 Throttling (AWS WAF and Queue-it request analysis)

All Queue-it waiting room access is protected by AWS WAF (Web Application Firewall) and AWS Shield Advanced. Shield Advanced safeguards against both network level as well as well-known application-level DDoS attacks.

Queue-it has implemented waiting room specific WAF Web ACL rules to counter and block HTTP layer attacks specific to waiting room logic, for example how many requests per minute can occur for a single Queue ID, regardless of IP used etc.

For special events with very high bot activity such as product drops, Queue-it's SRE team uses additional tailored AWS WAF Web ACL rules to counter these bots and throttle down the number of requests to the given waiting room based on different parameters. This throttling helps mitigate DDoS attacks and increases the overall availability of our systems in addition to assuring fair access to resources.

For requests allowed by both Shield and WAF Web ACL rules, QueueFront servers apply tailored request analysis and can redirect suspicious requests to a challenge page. This occurs when certain parameters are triggered, such as an unusually high number of Queue IDs requested by a single IP address in a short period of time.

Request analysis helps mitigate the risk of bots and automated scripts which could impact an entire customer's segment resources by requiring them to solve a challenge before entering a waiting room and getting a Queue ID.

2.6 CloudFront and caching

Queue-it uses Amazon's CloudFront CDN to cache static assets like images. When a customer customizes the waiting room theme, all static assets will automatically be deployed to CloudFront. When end-users download the pages, these assets will be served from the closest CloudFront Edge server. This will increase the speed and throughput of overall waiting room page load and increase availability by preventing QueueFront infrastructure from needing to serve static assets, reducing the number of requests managed by QueueFront considerably.

QueueFront also has a tailored cache service layer to reduce requests to databases (PostgreSQL and DynamoDB). In addition to the custom cache, PostgreSQL is set up with a writer and reader instance, to account for the read patterns for some tables and offload the writer instance from these requests, increasing overall availability and throughput.

3. Fault tolerance

3.1 Micro services architecture and circuit breaker design pattern

Queue-it's virtual waiting room solution is composed of multiple micro services designed to reduce the impact of one component / micro service not responding (limiting the radius blast). If a micro service is faulty and not responding in time, this careful design for failure assures that only a set of limited services are impacted (the ones calling it directly), which avoids cascading errors to upstream services.

The circuit breaker design pattern is used to stop calling a faulty service after several retries to avoid impacting this service further. It cuts requests to the faulty service and serves cached data. It opens back the circuit to make API calls only when there's a higher chance that the service can respond correctly and within the expected times.

3.2 Retry, timeout, and exponential backoff

In addition to the circuit breaker pattern, Queue-it software uses different strategies to reduce the impact of a faulty component and provide a better chance for recovery. To increase the likelihood of self-healing or issue resolution before the next API call occurs, we implement backoff strategies by reducing the frequency of calls to underlying services.

As an example, this is done on waiting room page status API calls to get specific Queue ID information (expected waiting time, number of Queue IDs ahead, etc.) and this API responds with the next time it should be called by the waiting room JavaScript.

The Queue page will also gracefully reduce functionality by showing the latest correct API response (cached data) if this API is not responding within the set timeout.

3.3 Durability of data

Queue-it uses AWS S3 service to store and back important data up. AWS S3 service has 99.999999999% durability achieved by redundantly storing the data on different devices on different Availability Zones. Queue-it also uses multi region storage for some important data to increase availability and durability.

3.4 Security groups and principle of least privilege

Queue-it services and components are isolated with security groups / virtual firewalls (AWS VPCs) providing access only to a set of expected authorized clients to reduce the chance of attacking services. Internet-facing services like QueueFront servers exist behind AWS Load balancer with WAF (Web Application Firewall) and underlying backend services. These services are only accessible via VPNs or from within VPCs / internal IP addresses that are effectively protecting them from external attacks.

3.5 Fire drill days and fault tolerance tests

The Queue-it DevOps team conducts fire drill days to simulate various system faults and ensure the backup and restore processes are efficient and accurate.

Fire drills evaluate overall system resilience by simulating faults on various parts of the system and testing the DevOps team's ability to detect the issue, accurately assess its impact and potentially fix it by harnessing and improving on automated healing system properties and / or relevant processes to fix the issue.

4. Fast recovery and backup strategy

4.1 Health checking and continuous monitoring

Queue-it uses an external service to monitor the different Queue segments that are up and running. A real-time status report is publicly available at http://status.queue-it.net/.

Also, the Queue-it's DevOps team has designed a comprehensive alert system based on different system metrics to ensure prompt response to any detected faults.

In addition to external party monitoring and advanced level of internal application monitoring and alerts, hundreds of Queue-it customers spanning different time zones are offered the best system availability checks.

QueueFront infrastructure receives millions of requests to Queue pages 24/7, 365 days a year, where our customers are closely monitoring their events daily. That makes issues become immediately apparent, which gives little time to detect any potential issues.

The Queue-it's support team (in Australia, Denmark, USA) is available 24/7 to help customers and report any incidents to the DevOps team for quick recovery.

4.2 Infrastructure as code (AWS CloudFormation)

Queue-it uses AWS best practices to deploy AWS services using CloudFormation, to automate the creation of infrastructure stack and eliminate the risk of human errors.

This provides Queue-it with tools for fast recovery in case of major incidents. 4.3 Data redundancy

Queue-it uses PostgreSQL and Dynamo DB with managed backups (RDS backup and Dynamo DB multi-AZ read / write). PostgreSQL is backed up for 30 days and its data (waiting room / custom theme data) can be restored to a specific second within the 30-day retention period.

In addition to the backups which allow for a quick recovery within a region, Queue-it also automatically takes snapshots of the database every hour. In case RDS is no longer available on an active region, the snapshots are accessible from another region with RDS PostgreSQL setup in passive mode and can be activated within minutes.

4.4 AMI Amazon Image

Quick setup of QueueFront EC2 instances image / required server software. Queue-it uses Amazon AMI images to quickly boot EC2 instances with the software and configuration needed. This helps reduce recovery time if needed, since DevOps would not have to install needed software individually but just boot an instance based on an image containing all necessary software to deploy QueueFront.

4.5 Incident management process

Queue-it development and operational processes promote continuous improvements. This includes incident management process with a postmortem meeting to learn how to eliminate the root cause of the incident or improve response time, eliminating or at least reducing time for a fast recovery.

5. Appendix

5.1 Queue-it White Papers

Below is the full list of other Queue-it white papers available digitally:

1. API
2. Bots and Abuse Management
3. Custom Theme
4. Health Check
5. High Availability
6. Load Test
7. Mobile App
8. Notifications and Logs
9. QuickStart
10. Security Considerations
11. Technical Integration
12. User Management

Queue-it white papers can also be found on the GO Platform under the “White papers and guides” menu link.

5.2 Change Record

Date	Author	Version	Change Reference
2024-10-15	Muhammad Jamil Chaudhary	12	Full review and adding South America region, Introduction and Advanced Shield detail
2022-11-21	Ismail Taouti	11.3	Added information about the latest load test. Updated information about DB recovery
2022-09-27	Ismail Taouti	11.2	Added coherent diagrams reflecting Queue-it brand style
2022-09-10	Ismail Taouti	11.1	Updated adding further clarifications and diagrams to different sections
2022-08-25	Ismail Taouti	11	Changes to include information about High availability of Queue-it engine
2018-11-24	Andrew Morris	10.1	Conversion to new template