Web Services and Applications Deployment Guide

Load balancing

4/5/2026

Contents

Load balancing

Web Services supports any third-party load balancer that supports sticky sessions. You should configure session affinity (sticky sessions) based on `JSESSIONID`. After your load balancer is set up, you can use the following URL for health checks: `http://<Web_Services_Host>:<Web_Services_Monitoring_Port>/actuator/health/readiness`.

where,

- `Web_Services_Host` - is the host name where GWS application is hosted. In case you want to set a specific hostname, specify it in the **address** option in the **management** section of the **Application.yaml** file.
  For example:

  ```
  management:
    server:
      address: <address>
  ```

- `Web_Services_Monitoring_port` - is the value configured in the **port** option in the **management** section of the **Application.yaml** file.

  ```
  management:
    server:
      port: <port>
  ```

HTTPS Configuration for monitoring endpoint

If you want to enable TLS for the monitoring endpoint, the following additional configurations are needed.

For example:

```
management:
  server:
    port: <port>
    ssl:
      enabled: true
      key-store: <keystore path>
      key-store-password: <password>
      key-alias: <alias>
      key-password: <password>
  endpoints:
    web:
      exposure:
        include: 'health,prometheus,reindexdetails'
```

Important TLS Considerations

- Do not use IP address literals (for example, 192.168.1.215 or fe80::3831:400c:dc07:7c40) in your monitoring endpoint URL when TLS is enabled.
- SNI (Server Name Indication) requires that all hostnames used contain at least one dot (.). Avoid short or local hostnames (like localhost or myhost); use a fully qualified domain name (FQDN) such as

`monitoring.mycompany.com`.

- Ensure that your SSL certificate's Common Name (CN) or Subject Alternative Names (SANs) match the hostname used to access the service.
- The certificate specified in the key store must be properly configured, trusted, and should not be expired.

If you are configuring your solution to use Hypertext Transfer Protocol over Secure Socket Layer, you do not need to set up HTTPS between your load balancer and Web Services.

> **Important**
> Web Services and Applications does not currently support web sockets.

Load balancing GWS Service Platform

- Use a load balancer to distribute requests across multiple GWS Service Platform instances. Load balancing improves availability, scalability, and performance by ensuring that no single instance receives all incoming traffic.
- If a GWS Service Platform instance experiences downtime, loadbalancer marks that instance as unhealthy. Loadbalancers will not route requests to the unhealthy instance until the service restarts and completes one full cache update cycle successfully.

Health Check Configuration

- Configure **active health checks** to continuously verify the health of GWS Service Platform instances before routing traffic. Active health checks send periodic requests, such as to the **/ready** endpoint, to detect problems early.
- Active health checks are essential for maintaining high availability and faster recovery. They remove unhealthy instances immediately and automatically restore them when they recover, ensuring uninterrupted service.
- If only passive health checks are used, issues are detected only after a client request fails, which may result in failed transactions, delayed responses, and routing requests to unhealthy instances.
- NGINX Plus supports both active and passive health checks. The open-source NGINX supports only passive health checks.
- `Sample NGINX Plus Configuration:`

```
health_check interval=1s fails=1 passes=1 uri=/ready match=ready_check mandatory;

# — interval=1s  → NGINX sends a health check request every 1000 milliseconds (1
second)
# — fails=1      → If the server fails *1 consecutive check*, it is marked as
UNHEALTHY
# — passes=1     → If the server passes *1 consecutive check*, it is marked as
```

```
HEALTHY again
# − uri=/ready   → NGINX calls the /ready endpoint on each backend server to test
availability
# − match=ready_check → Uses the criteria defined in the "match ready_check" block
to decide health
# − mandatory → Health check must be valid; if misconfigured, NGINX will not start/
reload

match ready_check {
     status 200;  #The server is considered HEALTHY only if it returns HTTP 200 (OK)
 }

upstream backend_service {
        server <host>:<port> max_fails=0;
        server <host>:<port> max_fails=0;
    }


# − upstream backend_service  → Defines a load-balancing group of backend servers
# − server <host>:<port>      → Adds a backend server to the upstream pool
# − max_fails=0               → Disables passive health checks, If max_fails not set
Passive HC enabled by default (max_fails=1, fail_timeout=10s).
```