NGINX

Using NGINX Plus to Load Balance Oracle E-Business Suite

Deployment Guide for NGINX Plus and EBS 12

Version 1

Revision History

- Version 1 (November 2015) – Initial version (NGINX Plus Release 7)

Table of Contents

This guide explains how to deploy NGINX Plus to load balance traffic across a pool of Oracle EBS 12 servers.

You can deploy EBS and NGINX Plus on premise, in a private cloud, or in public clouds including Amazon Elastic Compute Cloud (EC2), Google Cloud Platform, and Microsoft Azure. This guide covers the different installation types, and provides complete instructions for customizing both NGINX Plus and EBS as required.

About NGINX Plus and Oracle EBS

NGINX Plus is the commercially supported version of the open source NGINX software. NGINX Plus is a complete application delivery platform, extending the power of NGINX with a host of enterprise-ready capabilities that are instrumental to building web applications at scale:

- Full-featured HTTP and TCP load balancing
- High-performance reverse proxy
- Caching and offload of dynamic and static content
- Adaptive streaming to deliver audio and video to any device
- Application-aware health checks and high availability
- Advanced activity monitoring available via a dashboard or API
- Management and real-time configuration changes with DevOps-friendly tools

Oracle E-Business Suite (EBS) is a comprehensive suite of integrated, global business applications that enable organizations to make better decisions, reduce costs, and increase performance. Its cross-industry capabilities include enterprise resource planning, customer relationship management, and supply chain planning.

About Sample Values and Copying of Text

- `company.com` is used as a sample organization name (mostly in key names and DNS entries). Replace it with your organization's name.
- Many NGINX Plus configuration blocks in this guide list two sample EBS servers with IP addresses `172.31.0.146` and `172.31.11.210`. Replace these addresses with the IP addresses of your EBS servers. Include a line in the configuration block for each server if you have more or fewer than two. In contrast, port numbers are obligatory values except where noted.
- For readability reasons, some commands appear on multiple lines. If you want to copy and paste them into a terminal window, we recommend that you first copy them into a text editor, where you can substitute the object names that are appropriate for your deployment and remove any extraneous formatting characters that your PDF viewer might insert.
- The configuration examples in the step-by-step instructions include hyperlinks to the NGINX reference documentation, for easy access to more information about the directives. (If a directive appears multiple times in a section, only the first occurrence is hyperlinked.) We recommend that you do not copy hyperlinked text (or any other text) from a PDF file into your configuration files, because it might include unwanted link text and does not include whitespace and other formatting that makes the configuration easy to read. For more information, see Creating and Modifying Configuration Files on page 13.

Prerequisites and System Requirements

- Oracle EBS 12.2, installed and configured according to Oracle best practices.
- Linux system to host NGINX Plus (in on-premise and private-cloud deployments). To avoid potential conflicts with other applications, we recommend that you install NGINX Plus on a fresh system. For the list of Linux distributions supported by NGINX Plus, see NGINX Plus Technical Specifications.
- NGINX Plus R6 and later.

The instructions assume you have basic Linux system administration skills, including the following. Full instructions are not provided for these tasks.

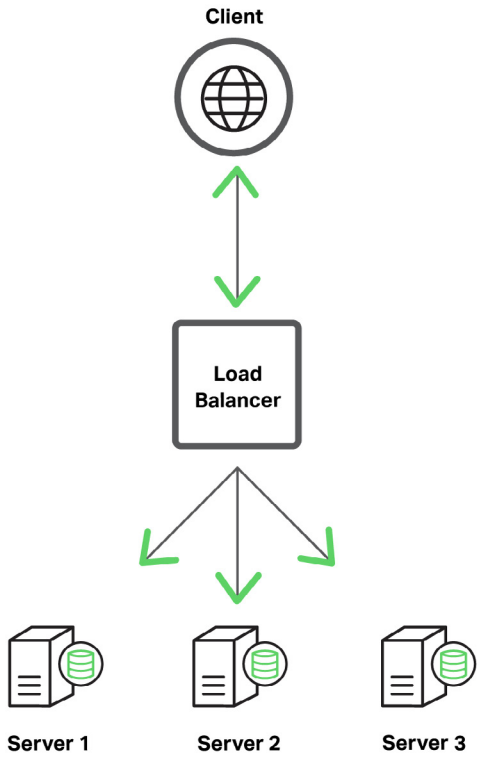
- Installing Linux software from vendor-supplied packages
- Editing configuration files
- Copying files between a central administrative system and Linux servers
- Running basic commands to start and stop services
- Reading log files

Similarly, the instructions assume you have the support of the team that manages your Oracle deployment. Their tasks include the following:

- Modifying Oracle configurations to configure a Web Entry Point
- Verifying the configuration

Architectural Design

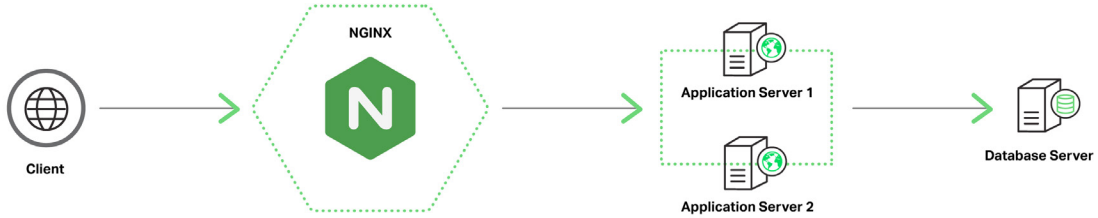
This figure represents a typical load-balancing architecture:



A load balancer performs the following tasks:

- Terminate SSL connections and decrypt SSL traffic
- Select backend servers based on a load-balancing method and health checks
- Forward HTTP requests to selected backend servers
- Provide session persistence
- Provide logging and monitoring capabilities

Oracle EBS has application tiers and a database tier. A load balancer is used in front of application tiers in order to provide higher performance, availability, security, and traffic management for the application servers.



Configuring Firewalls

For improved security, the NGINX Plus load balancer might be located in a DMZ. This might complicate and delay the installation process because of a required firewall configuration.

Please review the network configuration requirements below and make appropriate changes to the firewalls before proceeding with the configuration.

Purpose	Port	Source	Destination
Admin access, file transfer	22	Administrative network	NGINX Plus load balancer
Installation and update of NGINX Plus software	443	NGINX Plus load balancer	https://plus-pkgs.nginx.com
HTTP to HTTPS redirects	80	Any	NGINX Plus
Production HTTPS traffic	443	Any	NGINX Plus
Access to backend application	8000*	NGINX Plus	Backend application servers
Access to load-balanced application from application servers	443	Backend application servers	NGINX Plus load balancer

* Replace port 8000 with the application port if needed

Installing NGINX Plus

You can install NGINX Plus on premise, in a private cloud, or in a public cloud such as the Amazon Elastic Compute Cloud (EC2), the Google Cloud Platform, or Microsoft Azure. See the instructions for your installation type:

- On premise or private cloud – Instructions for your Linux operating system at the NGINX Plus Customer Portal
- Amazon Elastic Compute Cloud (EC2) – Installing NGINX Plus on Amazon EC2
- Google Compute Cloud – Installing NGINX Plus on the Google Cloud Platform
- Microsoft Azure – Installing NGINX Plus on Microsoft Azure

After installing NGINX Plus in any environment, you need to install an SSL certificate on the NGINX Plus server so that it can participate in encrypted communication with EBS clients.

Configuring an SSL Certificate for NGINX Plus

Install an SSL certificate on the NGINX Plus server. There are several ways to obtain the required certificate, including the following. For your convenience, step-by-step instructions are provided for the second and third options.

- If you already have an SSL certificate for NGINX Plus installed on another UNIX or Linux system (including systems running Apache HTTP Server or NGINX), copy it to the **/etc/nginx/ssl/** directory on the NGINX Plus server.
- If you need to request a new certificate from a certificate authority (CA) or your organization's security group, see Creating a Certificate Request with the OpenSSL Command below.
- If you already have an SSL certificate on a Windows system, see Exporting and Converting an SSL Certificate from an IIS Server on page 10.

CREATING A CERTIFICATE REQUEST WITH THE OPENSSL COMMAND

1. Log in as the root user on a machine that has the **openssl** software installed.
2. Create a private key to be packaged in the certificate.

```
root# openssl genrsa -out ~/company.com.key 2048
```

3. Create a backup of the key file in a secure location. If you lose the key, the certificate becomes unusable.

```
root# cp ~/company.com.key secure-dir/company.com.key.backup
```

4. Create a Certificate Signing Request (CSR) file.

```
root# openssl req -new -sha256 -key ~/company.com.key \
      -out ~/company.com.csr
```

5. Request a certificate from a CA or your internal security group, providing the CSR file (**company.com.csr**). As a reminder, never share private keys (**.key** files) directly with third parties.

   The certificate needs to be PEM format rather than in the Windows-compatible PFX format. If you request the certificate from a CA website yourself, choose NGINX or Apache (if available) when asked to select the server platform for which to generate the certificate. If you receive a certificate in PFX format, see the instructions in Exporting and Converting an SSL Certificate from an IIS Server below.

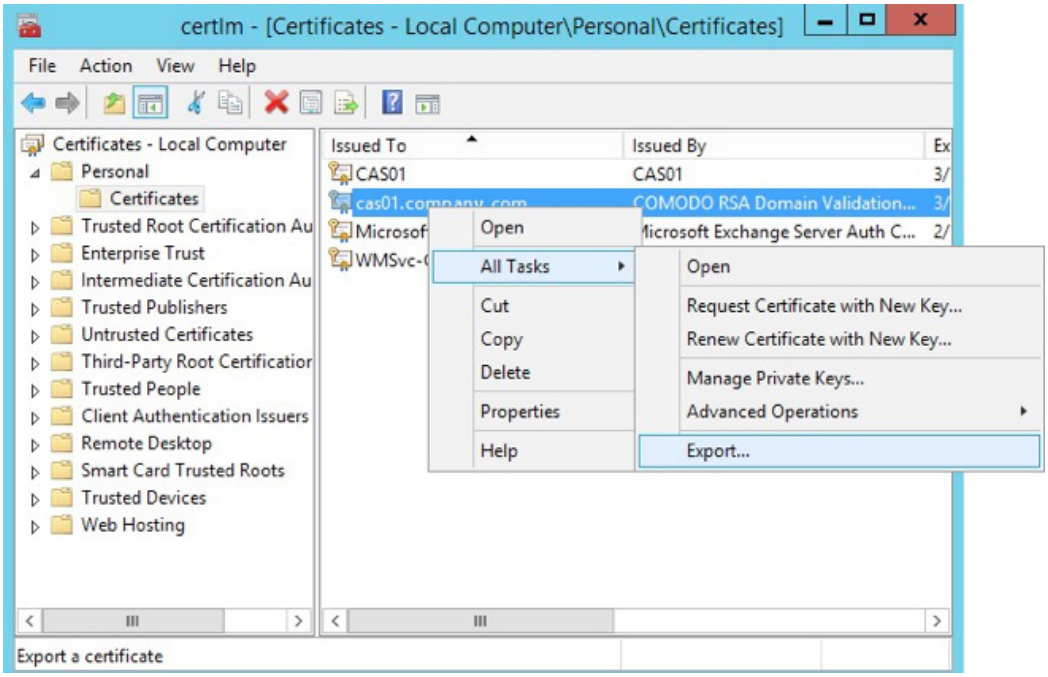
6. Copy or move the certificate file and associated key files to the **/etc/nginx/ssl/** directory on the NGINX Plus server.

EXPORTING AND CONVERTING AN SSL CERTIFICATE FROM AN IIS SERVER

On Windows systems, SSL certificates are packaged in a Public-Key Cryptography Standards (PKCS) archive file with extension **.pfx**. You need to export the **.pfx** file and convert the contents to the Linux compatible PEM format.

Working in the Microsoft Management Console, perform the following steps:

1. Open the **Certificates** snap-in.
2. In the left-hand navigation pane, click the **Certificates** folder in the logical store for the certificate you want to export (in the following figure, it is **Personal > Certificates**).
3. In the main pane, right-click the certificate to be exported (in the following figure, it is **cas01.company.com.**
4. On the menu that pops up, select **All Tasks**, then click **Export**.



5. In the Certificate Export Wizard window that pops up, click **Yes, export the private key**. (This option appears only if the private key is marked as exportable and you have access to it.)

6. If prompted for a password (used to encrypt the **.pfx** file before export), type it in the **Password** and **Confirm** fields. (Remember the password, as you need to provide it when importing the bundle to NGINX Plus.)

7. Click **Next**.

8. In **File name** field, type the filename and path to the location for storing the exported file (certificate and private key). Click **Next**, then **Finish**.

9. Copy the **.pfx** file to the NGINX Plus server.

Working on the NGINX Plus server (which must have the **openssl** software installed), perform the following steps:

10. Log in as the root user.

11. Extract the private key file from the **.pfx** file. You are prompted first for the password protecting the **.pfx** file (see Step 6 above), then for a new password used to encrypt the private key file being created (**company.com.key.encrypted** in the following sample command).

```
root# openssl pkcs12 -in exported-certs.pfx —nocerts \
      -out company.com.key.encrypted
```

12. Decrypt the key file. At the prompt, type the password you created in the previous step for the private key file.

```
root# openssl rsa -in company.com.key.encrypted \
      —out company.com.key
```

13. Extract the certificate file.

```
root# openssl pkcs12 -in exported-cert.pfx -clcerts \
      -nokeys -out company.com.crt
```

14. Copy or move the certificate files and key files to the **/etc/nginx/ssl/** directory.

Configuring Oracle EBS

For Oracle applications to work with a load balancer, you need to configure a Web Entry Point. For full instructions, please refer to the Oracle documentation on configuring Web Entry points, *Using Load-Balancers with Oracle E-Business Suite Release 12.2* (MOS Doc ID 1375686.1).

Use the AutoConfig Context Editor to set the configuration values in the applications context file on application servers.

Here are examples of appropriate values:

- Load Balancer Entry Point – `store.company.com`
- Application Server 1 – `apps-tier1.company.com`
- Application Server 2 – `apps-tier2.company.com`
- Web Entry protocol – `https`
- Application Tier Web Protocol – `http`
- Application Tier Web Port – `8000`
- Active Web Port – `443`

Configuring NGINX Plus for Oracle EBS

Complete the instructions in this section to configure NGINX Plus to load balance EBS servers.

Creating and Modifying Configuration Files

To reduce errors, this guide has you copy directives from files provided by NGINX, Inc. into your configuration files, instead of using a text editor to type in the directives yourself. Then you go through the sections in this guide (starting with Configuring Global Settings on page 15) to learn how to modify the directives as required for your deployment.

There is one file for a single Web Entry Point and one file for multiple Web Entry Points. If you are installing and configuring NGINX Plus on a fresh Linux system and using it only to load balance EBS traffic, you can use the provided file as your main NGINX Plus configuration file, which by convention is called **/etc/nginx/nginx.conf.**

We recommend, however, that instead of a single configuration file you use the scheme that is set up automatically when you install an NGINX Plus package, especially if you already have an existing NGINX Plus deployment or plan to expand your use of NGINX Plus to other purposes in future. In the conventional scheme, the main configuration file is still called **/etc/nginx/nginx.conf**, but instead of including all directives in it you create separate configuration files for different functions and store the files in the **/etc/nginx/conf.d** directory. You then use the `include` directive in the main file to reference the separate files in the appropriate contexts.

To download the complete configuration file for a single or multiple Web Entry Points from the NGINX website, run the appropriate command:

```
root# curl https://nginx.com/resource/conf/ \
      oracle-single-entry-point.conf > /etc/nginx/conf.d/
```

or

```
root# curl https://nginx.com/resource/conf/ \
      oracle-multiple-entry-point.conf > /etc/nginx/conf.d/
```

(You can also download the URL via a browser and copy the text into the indicated file.)

To set up the conventional configuration scheme, add an `http` configuration block in the main **nginx.conf** file, if it does not already exist. (The standard placement is below any global directives; see with Configuring Global Settings on page 15.) Add this include directive:

```
http {
    include conf.d/oracle-(single|multiple)-entry-point.conf;
}
```

You can also use wildcard notation to reference all files that pertain to a certain function or traffic type in the appropriate context block. For example, if you name all HTTP configuration files *function*-**http.conf**, this is an appropriate `include` directive:

```
http {
    include conf.d/*-http.conf;

}
```

For reference purposes, the full configuration files are also provided in this document:

- Full Configuration for a Single Web Entry Point on page 26
- Full Configuration for Multiple Web Entry Points on page 29

We recommend, however, that you do not copy text directly from this document. PDF does not use the same mechanisms for positioning text (such as line breaks and white space) as text editors do. In text copied from a PDF into an editor, lines run together and indenting of child statements in configuration blocks is missing or inconsistent. The absence of formatting does not present a problem for NGINX Plus, because (like many compilers) it ignores white space during parsing, relying solely on semicolons and curly braces as delimiters. The absence of white space does, however, make it more difficult for humans to interpret the configuration and modify it without making mistakes.

ABOUT RELOADING UPDATED CONFIGURATION

We recommend that each time you complete a set of updates to the configuration, you run the `nginx -t` command to test the configuration file for syntactic validity.

```
root# nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

To tell NGINX Plus to start using the new configuration, run one of the following commands:

```
root# nginx -s reload
```

or

```
root# service nginx reload
```

Configuring Global Settings

Verify that the main nginx.conf file includes the following global directives, adding them as necessary.

```
user nginx;
worker_processes auto;
error_log /var/log/nginx/error.log info;
pid /var/run/nginx.pid;

events {
   worker_connections 1024;
}
# If using the standard configuration scheme, this is the usual
# location for the 'http' block with 'include' directives
# that refer to files in the conf.d directory.
```

Configuring the Virtual Server for HTTPS Traffic

In the `http` context, define a virtual server for HTTPS traffic.

```
# In the "http { }" block
server {
   listen 443 ssl;
   ssl_certificate /etc/nginx/ssl/server.crt;
   ssl_certificate_key /etc/nginx/ssl/server.key;
   ssl_protocols TLSv1.2;
}
```

This server listens on every IP address. If needed, you can restrict listening to one or more IP addresses (IPv4 or IPv6). For example, with this `listen` directive the server listens on address 10.210.15.20 and port 443:

```
listen 10.210.15.20:443 ssl;
```

Setting the Default MIME Type

In case the EBS server does not specify the MIME type of the data it is sending to the client (in the `Content-Type` field of the response header), define the default MIME type as `text/html`. Include these directives in the `http` context:

```
# In the “http { }” block
include /etc/nginx/mime.types;
default_type text/html;
```

Configuring Load Balancing

To configure load balancing, create an upstream server group with all of your Oracle app servers, and add directives to the existing `server` block to proxy all traffic to the new upstream group.

1. In the `http` context, add an `upstream` block to create an upstream server group called **oracle** (each upstream group name in the configuration must be unique).

   ```
   # In the "http { }" block
   upstream oracle {
       zone oracle 64k;
       server 172.31.11.210:8000 max_fails=0;
       server 172.31.0.146:8000 max_fails=0;
   }
   ```

   The `zone` directive creates a 64-KB shared memory zone, also called **oracle**, for storing configuration and runtime state information about the group that is shared among worker processes.

   Add a `server` directive for each of your Oracle app servers. You can identify servers by IP address or hostnames. If using hostnames, make sure that the operating system on the NGINX Plus server can resolve them. The sample block has two servers.

   NGINX Plus supports two different kinds of application health checks, active and passive. We recommend configuring active health checks (see Configuring Active Health Checks on page 17) and disabling passive health checks by including the `max_fails=0` parameter to each `server` directive.

2. In the `server` block that you created in Configuring the Virtual Server for HTTPS Traffic on page 15, add a `location` block that proxies all traffic to the upstream group.

   ```
   # In the "server { }" block
   location / {
      proxy_pass http://oracle;
      proxy_set_header Host $host;
   }
   ```

Configuring Session Persistence

EBS applications require session persistence. Without it, you will experience unexpected session logouts almost immediately after logging in. Oracle supports three methods for session persistence: active cookie, passive cookie, and IP-based.

For simplicity, configure active-cookie session persistence with the NGINX Plus "sticky cookie" method. NGINX Plus adds a cookie called **ngxcookie** to every new user session, recording a hash of the backend server that was selected for the first request from the user. The cookie expires when the browser restarts.

Add the `sticky` directive to the `upstream` block you created in Configuring Load Balancing above, so the complete block looks like this:

```
# In the "http { }" block
upstream oracle {
    zone oracle 64k;
    server 172.31.11.210:8000 max_fails=0;
    server 172.31.0.146:8000 max_fails=0;
    sticky cookie ngxcookie;
}
```

Configuring HTTP Redirects

As configured in Configuring the Virtual Server for HTTPS Traffic on page 15, NGINX Plus accepts only HTTPS traffic on port 443. In case your users forget and use plain HTTP (start URLs with **http**:// instead of **https**://), create a redirect that returns HTTP response code `302` and the rewritten URL for every HTTP request. Opening port 80 does not decrease security, because the requests to this port don't result in connections to your backend servers.

Add a new `server` block in the `http` context:

```
# In the "http { }" block
server {
    listen 80;
    status_zone oracle-http-redirect;
    return 302 https://$http_host$request_uri;
}
```

Configuring Active Health Checks

The open source NGINX software performs basic checks on responses from upstream servers, retrying failed requests where possible. NGINX Plus adds out-of-band application health checks (also known as *synthetic transactions*). The related Slow-start feature gradually ramps up traffic to servers in the load-balanced group as they recover from a failure, allowing them to "warm up" without being overwhelmed.

These features enable NGINX Plus to detect and work around a much wider variety of problems and have the potential to significantly improve the availability of your Oracle applications.

We will use an active health check to verify that the Oracle application returns the `X-ORACLE-DMS-ECID` header. If not, the health check fails and NGINX Plus doesn't send requests to the failed server.

Add a new `match` block inside the `http` block to test for the `X-ORACLE-DMS-ECID` header.

```
# In the "http { }" block
match oracleok {
    status 200-399;
    header X-ORACLE-DMS-ECID;
}
```

Add a new `location` block for the health check, in the `server` block you created in Configuring the Virtual Server for HTTPS Traffic on page 15:

```
# In the "server { }" block
location @health_check {
    internal;
    proxy_connect_timeout 3s;
    proxy_read_timeout 3s;
    proxy_pass http://oracle;
    proxy_set_header Host "oracle.nginxlab.net";
    health_check match=oracleok interval=4s
                 uri=/OA_HTML/AppsLocalLogin.jsp;
}
```

Note that the `location` block is in the `server` block, but the `match` block is in the `http` block.

Configuring Caching for Application Acceleration

Caching of static objects like the following significantly improves the performance of Oracle EBS:

- Images
- CSS files
- JavaScript files
- Java applets

Before configuring caching, make sure that the NGINX Plus host has adequate free disk space and disk performance. SSDs are preferred for their superior performance, but standard spinning media can be used.

1. Create a directory for cached files:

```
root@nginx # mkdir /var/oracle-cache
root@nginx # chown nginx /var/oracle-cache
```

2. In the main `http` context, define the path to the cache, its maximum size (here, 500 MB), and the name (**cache_oracle**) and maximum size (50 MB) of the shared memory zone used for storing cache keys. Adjust the size values as appropriate for the amount of free disk space on the NGINX Plus host.

```
# In the "http { }" block
proxy_cache_path /var/oracle-cache max_size=500m
keys_zone=cache_oracle:50m;
```

3. In the `server` block you created in Configuring the Virtual Server for HTTPS Traffic on page 15, enable caching by defining the name of the shared memory zone for the cache (**cache_oracle**). The any parameter to the `proxy_cache_valid` directive specifies that all responses are cached and expire after one hour.

```
# In the "server { }" block
proxy_cache cache_oracle;
proxy_cache_valid any 1h;
```

You can track cache usage using the following methods:

- Statistics from the NGINX Plus Status module, displayed on the built-in dashboard, or fed to a custom or third-party reporting tool
- The NGINX Plus access log, when the log format includes the `$upstream_cache_status` variable

For detailed configuration instructions, see the next section.

Configuring Advanced Logging and Monitoring

NGINX Plus provides multiple ways to monitor your Oracle EBS installation, providing data about unavailable servers, failed health checks, response code statistics, and performance. In addition to its built-in tools, NGINX Plus easily integrates into enterprise monitoring systems through industry-standard protocols.

CONFIGURING LOGGING WITH A CUSTOM MESSAGE FORMAT

You can customize the format of messages written to the NGINX Plus access log to include more application-specific information. Most system variables can be included in log messages. The predefined NGINX Plus **combined** format includes the following variables:

- `$body_bytes_sent` – Number of bytes in the body of the response sent to the client
- `$http_user_agent` – User-Agent header in the client request
- `$http_referer` – Referer header in the client request
- `$remote_addr` – Client IP address
- `$remote_user` – Username provided for HTTP basic authentication
- `$request` – Full original request line
- `$status` – Response status code
- `$time_local` – local time in the Common Log Format

For the complete list of NGINX Plus variables, see http://nginx.org/en/docs/varindex.html.

To make troubleshooting of our load-balancing deployment easier, let's add the $upstream_addr variable (the address of the actual server generating the response) to the variables from the combined format.

Add the following lines in the http context to enable access logging to **/var/log/nginx/access.log** and to define the message format:

```
# In the "http { }" block
access_log /var/log/nginx/access.log main;
log_format main '$remote_addr - $remote_user [$time_local] "$request"
                $status $body_bytes_sent "$http_referer"
                "$http_user_agent" $upstream_addr';
```

To disable logging completely, for a virtual server, or for a location, include the following directive in the `http`, `server`, or `location` block:

```
access_log off;
```

Note that the message format for error logs is predefined and cannot be changed.

CONFIGURING LOGGING WITH SYSLOG

The syslog utility is a widely used standard for message logging. It is used in the backbone of many monitoring and log-aggregation solutions.

You can configure NGINX Plus to direct both error logs and access logs to syslog servers. Here we configure logging to a syslog server listening on the IP address 192.168.1.1 and default UDP port 514.

To configure the error log, add the following line in the main context, the `http` context, or a `server` or `location` block:

```
# In the main, "http { }", "server { }", or "location { }" block
error_log syslog:server=192.168.1.1 info;
```

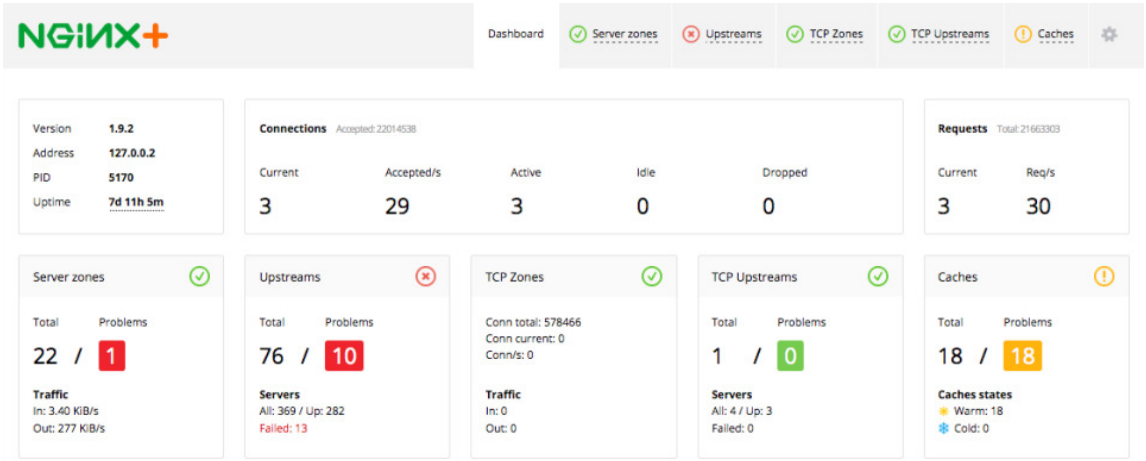
To configure the access log using the predefined combined format, add the following line in the `http` context (it appears on multiple lines here solely for formatting reasons):

```
access_log
syslog:server=192.168.1.1,facility=local7,tag=oracle,severity=info
          combined;
```

You can include multiple `error_log` and `access_log` directives in the same context. Messages are sent to every syslog server and file.

CONFIGURING LIVE ACTIVITY MONITORING

NGINX Plus includes a live activity monitoring interface that provides key load and performance metrics in real time. Statistics are reported through a RESTful JSON interface, making it very easy to feed the data to a custom or third-party monitoring tool. These instructions deploy the dashboard that is built into NGINX Plus.



The quickest way to configure the built-in dashboard is to download the sample configuration file from the NGINX, Inc. website:

```
root# curl https://nginx.com/resource/conf/status.conf > \
          /etc/nginx/conf.d/
```

Then include the file in the `http` context in the main configuration file (**nginx.conf**):

```
# In the "http { }" block in nginx.conf
include conf.d/status.conf;
```

Comments in the **status.conf** file explain which directives you must customize for your deployment. In particular, the default settings in the sample configuration file allow anyone on any network to access the dashboard. **We strongly recommend that you use one or more of the following methods to restrict access to the dashboard:**

- IP address-based access control lists (ACLs). In the sample configuration file, uncomment the `allow` and `deny` directives, and substitute the address of your administrative network for `10.0.0.0/8`. Only users on the specified network can access the status page.

  ```
  allow 10.0.0.0/8;
  deny all;
  ```

- HTTP basic authentication. In the sample configuration file, uncomment the `auth_basic` and `auth_ basic_user_file` directives and add user entries to the **/etc/nginx/users** file (for example, by using an htpasswd generator).

  ```
  auth_basic on;
  auth_basic_user_file /etc/nginx/users;
  ```

- Client certificates, which are part of a complete configuration of SSL or TLS. For more information, see NGINX SSL Termination in the NGINX Plus Admin Guide and the documentation for the HTTP SSL module.
- Firewall. Configure your firewall to disallow outside access to the port for the dashboard (**8080** in the sample configuration file).

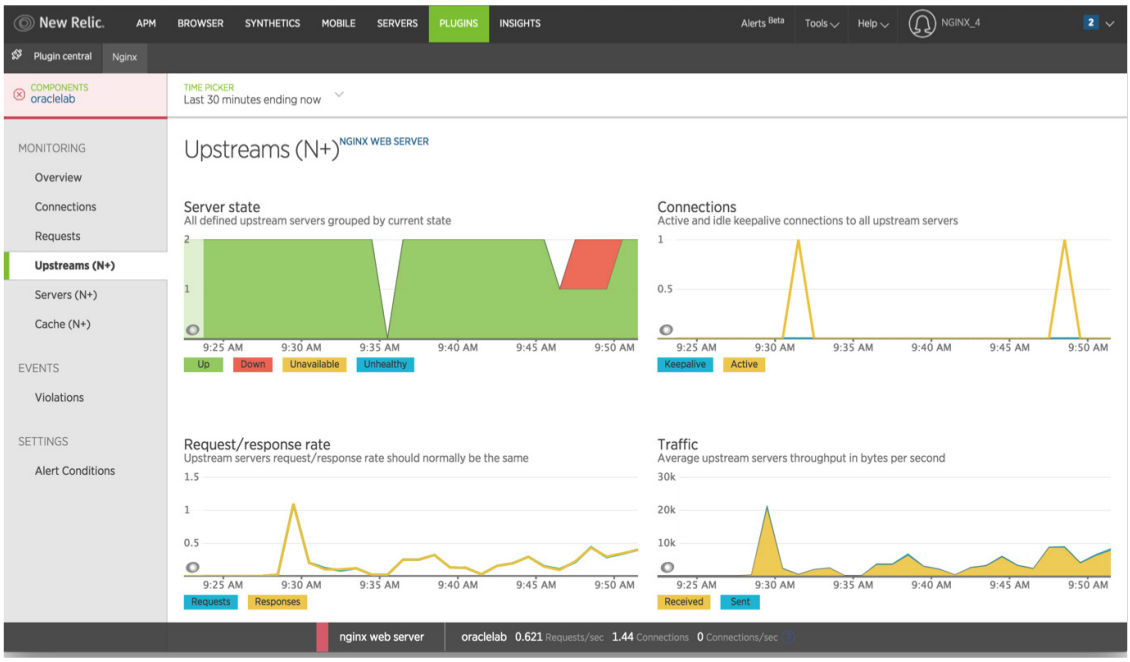
When you reload NGINX Plus with the new configuration, the NGINX Plus dashboard is available immediately at **http://*nginx-server-address*:8080**.

For detailed information about live activity monitoring, see the following documentation:

- Live Activity Monitoring of NGINX Plus (at nginx.com)
- Status module (at nginx.org)

MONITORING WITH THIRD-PARTY TOOLS

The Status module provides all metrics in JSON format, so you can feed them to many monitoring systems. Here we describe how to configure the NGINX plug-in for New Relic. We assume that you already have a New Relic account.



1. Configure the NGINX open source repository in order to download the agent. Use the instructions on the nginx.org website but don't install the **nginx** package since the **nginx-plus** package is already installed: http://nginx.org/en/linux_packages.html.
2. Install the **nginx-nr-agent** package using standard package management tools (`yum`, `apt-get`).
3. Insert your New Relic license key and NGINX Plus status URL into the configuration file (**/etc/nginx-nr-agent/nginx-nr-agent.ini**).
4. Run the following command to start the agent in daemon mode.

```
root@nginx # service nginx-nr-agent start
```

For more detailed installation and configuration instructions, see the **README.txt** file bundled with the **nginx-nr-agent** package. For generic installation instructions, see http://newrelic.com/plugins/nginx-inc/13.

Configuring Backup Servers for Disaster Recovery

If you have backup EBS servers, either at the same physical location as your regular servers or at a disaster recovery site, you can include them in the configuration so that EBS continues to work even if all the primary EBS servers go down.

To configure backup servers, add them to the `upstream` block you created in Configuring Load Balancing on page 16 and include the backup parameter. NGINX Plus does not forward traffic to them unless the primary servers all go down.

```
# In the "upstream { }" block
server 10.180.71.10:8000 max_fails=0 backup;
server 10.180.71.215:8000 max_fails=0 backup;
```

You can then use a DNS-based global load-balancing solution to secure against site-level failures.

Configuring NGINX Plus for High Availability

To increase the reliability of your EBS deployment even more, configure NGINX Plus for high availability (HA).

CONFIGURING HIGH AVAILABILITY IN AN ON-PREMISE DEPLOYMENT

NGINX Plus Release 6 (R6) and later includes a solution for fast and easy configuration of NGINX Plus in an active-passive high-availability (HA) setup, based on software from the keepalived open source project.

The keepalived solution has three components: the `keepalived` daemon, an implementation of the Virtual Router Redundancy Protocol (VRRP) to manage virtual routers (virtual IP addresses), and a health-check facility to determine whether a service (in this case, NGINX Plus) is up and operational. If a service on a node fails the configured number of health checks, `keepalived` reassigns the virtual IP address from the master (active) node to the backup (passive) node.

VRRP ensures that there is a master node at all times. The backup node listens for VRRP advertisement packets from the master node. If it does not receive an advertisement packet for a period longer than three times the configured advertisement interval, the backup node takes over as master and assigns the configured virtual IP addresses to itself.

Run the **nginx-ha-setup** script (available in the **nginx-ha-keepalived** package, which must be installed in addition to the base NGINX Plus package) on both nodes as the root user. The script configures a high-availability NGINX Plus environment with an active-passive pair of nodes acting as master and backup. It prompts for the following data:

- IP address of the local and remote nodes (one of which will be configured as a master, the other as a backup)
- One free IP address to be used as the cluster endpoint's (floating) virtual IP address

The configuration of the `keepalived` daemon is recorded in the file **/etc/keepalived/keepalived.conf**. The configuration blocks in the file control notification settings, the virtual IP addresses to manage, and the health checks to use to test the services that rely on virtual IP addresses. The following is the configuration file created by the **nginx-ha-setup** script on a CentOS 7 machine. Note that this is not an NGINX Plus configuration file, so the syntax is different (semicolons are not used to delimit directives, for example).

```
vrrp_script chk_nginx_service {
    script "/usr/libexec/keepalived/nginx-ha-check"
    interval 3
    weight 50
}

vrrp_instance VI_1 {
    interface eth0
    state BACKUP
    priority 101
    virtual_router_id 51
    advert_int 1
    unicast_src_ip 192.168.100.100
    unicast_peer {
        192.168.100.101
    }
    authentication {
        auth_type PASS
        auth_pass f8f0e5114cbe031a3e1e622daf18f82a
    }
    virtual_ipaddress {
        192.168.100.150
    }
    track_script {
         chk_nginx_service
    }
    notify "/usr/libexec/keepalived/nginx-ha-notify"
}
```

For detailed documentation, see High Availability Support for NGINX Plus in the NGINX Plus Admin Guide.

CONFIGURING HIGH AVAILABILITY IN A PUBLIC CLOUD DEPLOYMENT

Most public cloud systems have integrated tools for ensuring high availability of load-balancer instances. NGINX Plus is available in three cloud environments, which provide the indicated solutions for high availability of NGINX Plus instances:

- Amazon EC2 – Elastic Load Balancing
- Google Cloud Platform – Google Compute Engine HTTP load balancing
- Microsoft Azure – Azure Traffic Manager with Azure Load Balancer

Please refer to the specific documentation provided by your cloud vendor.

We recommend that you use the integrated cloud tools as simple high-availability solutions and let NGINX Plus perform more sophisticated operations:

- Security
- SSL termination
- Advanced request routing
- Health checks
- Session persistence
- Monitoring
- Caching

Configuring Multiple Web Entry Points

The preceding sections of this document describe how to configure NGINX Plus load balancing for a single Web Entry Point.

In some cases you will need to configure multiple Web Entry Points through the same load balancer, for reasons like the following:

- Access from your internal network vs. externally available servers
- Access by different groups of users (employees, partners, customers)
- Access with different networking requirements (for example, a multihop DMZ configuration)

If multiple Web Entry Points are required, then for each Web Entry Point you need to:

- Add a new `upstream` block for each set of app servers
- Add a new `server` block for each load balancer entry point
- Ensure that each shared memory zone has a unique name
- Include the `server_name` directive in every `server` block
- Change the listeners from any IP to specific IP addresses, if needed
- Provide additional SSL certificate files if not using UCC or wildcard certificates

For a sample configuration, see Full Configuration for Multiple Web Entry Points on page 29.

Full Configuration Files

For your convenience, the configuration files in this section include all directives discussed in this guide. It is intended for reference. As explained in About Sample Values and Copying of Text on page 5, we recommend that you do not copy text from this PDF file into configuration files, because it might include unwanted link text and does not include whitespace and other formatting that makes the configuration easy to read. Instead, download the appropriate file from the NGINX, Inc. website as described in Creating and Modifying Configuration Files on page 13.

Please note that this configuration file contains sample values that you need to change for your deployment.

Full Configuration for a Single Web Entry Point

```
user nginx;
worker_processes auto;
error_log /var/log/nginx/error.log info;
pid /var/run/nginx.pid;

events {
    worker_connections 1024;
}

http {
    include /etc/nginx/mime.types;
    default_type text/html;

    proxy_cache_path /var/oracle-cache keys_zone=cache_oracle:50m
                     max_size=500m;

    # Custom logging configuration
    log_format main '$remote_addr - $remote_user [$time_local]
                    "$request" $status $body_bytes_sent "$http_referer"
                    "$http_user_agent" $upstream_addr';
    access_log /var/log/nginx/access.log main;

    upstream oracle {
        zone oracle 64k;

        # Production servers
        server 172.31.11.210:8000 max_fails=0;
        server 172.31.0.146:8000 max_fails=0;
```

```
    # Disaster recovery servers
    server 172.33.111.210:8000 max_fails=0 backup;
    server 172.33.100.146:8000 max_fails=0 backup;

    # Session persistence
    sticky cookie ngxcookie;
}

server {
    listen 80;
    status_zone oracle-http-redirect;
    return 302 https://$http_host$request_uri;
}

server {
    listen 443 ssl;
    ssl_certificate /etc/nginx/ssl/server.crt;
    ssl_certificate_key /etc/nginx/ssl/server.key;
    ssl_protocols TLSv1.2;
    status_zone oracle-ssl;
    proxy_cache cache_oracle;

    location / {
        proxy_pass http://oracle;
        proxy_set_header Host $host;
        proxy_cache_valid any 1h;
    }

    location @health_check {
        internal;
        proxy_connect_timeout 3s;
        proxy_read_timeout 3s;
        proxy_pass http://oracle;
        proxy_set_header Host "oracle.company.com";
        health_check match=oracleok interval=4s
                     uri=/OA_HTML/AppsLocalLogin.jsp;
    }
}

match oracleok {
    status 200-399;
    header X-ORACLE-DMS-ECID;
}
```

```
    server {
        # Status zone allows the status page to display statistics for
        # the entire server block.
        # Enable it for every "server {}" block in other configuration
        # files.
        status_zone status-page;
        # If NGINX Plus is listening on multiple IP addresses, uncomment
        # this directive to restrict the status page to a single IP
        # address.
        #listen 10.2.3.4:8080;

        # Status page is enabled on port 8080 by default.
        listen 8080;

        # HTTP basic authentication is enabled by default.
        # You can add users with any htpasswd generator.
        # Command-line and other online tools are very easy to find.
        # You can also reuse the htpasswd file from an Apache HTTP
        # server installation.
        #auth_basic on;
        #auth_basic_user_file /etc/nginx/users;

        # Limit access to the status page to users on admin networks
        # only. Uncomment the "allow" directive and change the network
        # address.
        #allow 10.0.0.0/8;
        deny all;

        # NGINX provides a sample HTML status page for easy dashboard
        # view.
        root /usr/share/nginx/html;
        location = /status.html { }

        # Standard HTTP features are fully supported with the status
        # page. This directive provides a redirect from "/" to
        # "/status.html"
        location = / {
            return 301 /status.html;
        }

        # Main status location. HTTP features like authentication,
        # access control, header changes, and logging are fully
        # supported.
        location /status {
            status;
            status_format json;
            access_log off;
        }
    }
}
```

Full Configuration for Multiple Web Entry Points

This configuration is for two Web Entry Points with the following settings:

- First Web Entry Point
  - Domain name – **oracle-one.company.com**
  - SSL certificate and key – **server_one.crt** and **server_one.key**
  - Status zone – **oracle-ssl-one**
  - Cache zone – **cache_oracle_one**
  - Upstream name – **oracle_one**
  - EBS servers – 172.31.11.210, 172.31.0.146
  - Backup (DR) EBS servers – 172.33.111.210, 172.33.100.146

- Second Web Entry Point
  - Domain name – **oracle-two.company.com**
  - SSL certificate and key – **server_two.crt and server_two.key**
  - Status zone – **oracle-ssl-two**
  - Cache zone – **cache_oracle_two**
  - Upstream name – **oracle_two**
  - EBS servers – 172.31.11.211, 172.31.0.147
  - Backup (DR) EBS servers – 172.33.111.211, 172.33.100.147

```
user nginx;
worker_processes auto;
error_log /var/log/nginx/error.log info;
pid /var/run/nginx.pid;

events {
    worker_connections 1024;
}

http {
    include /etc/nginx/mime.types;
    default_type text/html;

    proxy_cache_path /var/oracle-cache-one
                     keys_zone=cache_oracle_one:50m max_size=500m;
    proxy_cache_path /var/oracle-cache-two
                     keys_zone=cache_oracle_two:50m max_size=500m;
```

```
# Custom logging configuration
log_format main '$remote_addr - $remote_user [$time_local]
                "$request" $status $body_bytes_sent "$http_referer"
                "$http_user_agent" $upstream_addr';
access_log /var/log/nginx/access.log main;

upstream oracle_one {
    zone oracle_one 64k;

    # Production servers
    server 172.31.11.210:8000 max_fails=0;
    server 172.31.0.146:8000 max_fails=0;

    # Disaster recovery servers
    server 172.33.111.210:8000 max_fails=0 backup;
    server 172.33.100.146:8000 max_fails=0 backup;

    # Session persistence
    sticky cookie ngxcookie;
}


upstream oracle_two {
    zone oracle_two 64k;

    # Production servers
    server 172.31.11.211:8000 max_fails=0;
    server 172.31.0.147:8000 max_fails=0;

    # Disaster recovery servers
    server 172.33.111.211:8000 max_fails=0 backup;
    server 172.33.100.147:8000 max_fails=0 backup;

    # Session persistence
    sticky cookie ngxcookie;
}

server {
    listen 80;
    status_zone oracle-http-redirect;
    return 302 https://$http_host$request_uri;
}
```

```
server {
    listen 192.168.210.10:443 ssl;
    server_name oracle-one.company.com;
    ssl_certificate /etc/nginx/ssl/server_one.crt;
    ssl_certificate_key /etc/nginx/ssl/server_one.key;
    ssl_protocols TLSv1.2;
    status_zone oracle-ssl-one;
    proxy_cache cache_oracle_one;

    location / {
        proxy_pass http://oracle_one;
        proxy_set_header Host $host;
        proxy_cache_valid any 1h;
    }

    location @health_check {
        internal;
        proxy_connect_timeout 3s;
        proxy_read_timeout 3s;
        proxy_pass http://oracle_one;
        proxy_set_header Host "oracle-one.company.com";
        health_check match=oracleok interval=4s
                     uri=/OA_HTML/AppsLocalLogin.jsp;
    }
}

server {
    listen 192.168.210.11:443 ssl;
    server_name oracle-two.company.com;
    ssl_certificate /etc/nginx/ssl/server_two.crt;
    ssl_certificate_key /etc/nginx/ssl/server_two.key;
    ssl_protocols TLSv1.2;
    status_zone oracle-ssl;
    proxy_cache cache_oracle;

    location / {
        proxy_pass http://oracle_two;
        proxy_set_header Host $host;
        proxy_cache_valid any 1h;
    }
```

```
    location @health_check {
        internal;
        proxy_connect_timeout 3s;
        proxy_read_timeout 3s;
        proxy_pass http://oracle_two;
        proxy_set_header Host "oracle-two.company.com";
        health_check match=oracleok interval=4s
                     uri=/OA_HTML/AppsLocalLogin.jsp;
    }
}

match oracleok {
    status 200-399;
    header X-ORACLE-DMS-ECID;
}

server {
    # Status zone allows the status page to display statistics for
    # the entire server block. Enable it for every "server {}" block
    # in other configuration files.
    status_zone status-page;

    # If NGINX Plus is listening on multiple IP addresses, uncomment
    # this directive to restrict the status page to a single IP
    # address.
    #listen 10.2.3.4:8080;

    # Status page is enabled on port 8080 by default.
    listen 8080;

    # HTTP basic authentication is enabled by default.
    # You can add users with any htpasswd generator.
    # Command-line and other online tools are very easy to find.
    # You can also reuse the htpasswd file from an Apache HTTP
    # server installation.
    #auth_basic on;
    #auth_basic_user_file /etc/nginx/users;

    # Limit access to the status page to users on admin networks
    # only. Uncomment the "allow" directive and change the network
    # address.
    #allow 10.0.0.0/8;
    deny all;

    # NGINX provides a sample HTML status page for easy dashboard
    # view.
    root /usr/share/nginx/html;
    location = /status.html { }
```

```
        # Standard HTTP features are fully supported with the status
        # page.
        # This directive provides a redirect from "/" to "/status.html"
        location = / {
            return 301 /status.html;
        }
        # Main status location. HTTP features like authentication,
        # access control, header changes, and logging are fully
        # supported.
        location /status {
            status;
            status_format json;
            access_log off;
        }
    }
}
```

For more information,

visit nginx.com, or
send us an email at
nginx-inquiries@nginx.com

NGINX