F5 NGINX

15 Essential Metrics for NGINX

December 2024

Dave McAllister

NGINX
Part of F5

NGINX and Monitoring

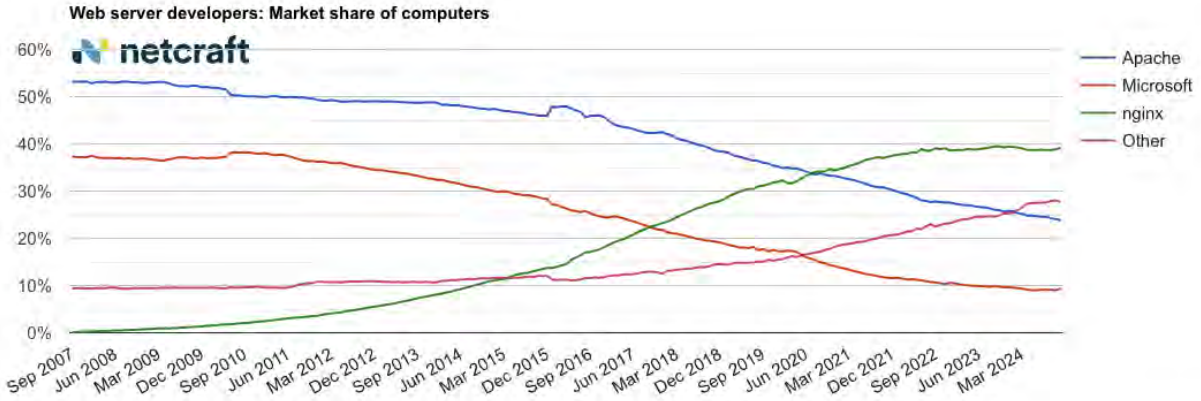
One of the most popular web servers

- Reverse Proxy
- Mail Proxy
- Load Balancing

Many things impact performance of NGINX

- Increase in requests
- Higher disk IO or network IO
- CPU or memory
- Application errors

Monitoring is essential



Developer	October 2024	Percent	November 2024	Percent	Change
nginx	5,053,891	38.87%	5,132,851	39.14%	0.27
Apache	3,131,957	24.09%	3,118,996	23.78%	-0.30
Microsoft	1,170,825	9.00%	1,221,517	9.31%	0.31

November 2024 Web Server Survey | Netcraft

Collecting the data

- Access NGINX Open Source metrics via the stub_status module
- NGINX Plus adds an API endpoint to query specific metrics
  - Plus has a lot of metrics
- Both offer the ability to scrape logs
- While your choice for visualization is wide open, Plus does offer some built-in dashboards

- Add the following to your nginx.conf file

```
Location /nginx_status {
    stub_status;
    allow 127.0.0.1;
    deny  all;
}
```


THE most essential metrics to track depends on the specific goals and needs

Aligning Metrics with Concerns

Request Metrics: Monitoring HTTP requests to the server.

Response Metrics: Monitoring server responses to client requests.

Connection Metrics: Tracking and analyzing server connections.

•**Performance Metrics:** Measuring server efficiency and speed.

Resource Utilization Metrics: Monitoring CPU and memory usage.

Cache Metrics: Monitoring caching mechanisms.

SSL/TLS Metrics: Monitoring SSL/TLS performance.

Security Metrics: Enhancing server security.

Active Connections

This metric indicates the number of currently active connections to the server.

- Helps understand the load on your server
- Do you need to scale your resources
- Do you optimize your configurations to handle traffic more efficiently

- **Variable:**
  - $connections_active (NGINX Open Source)
  - nginx_connections_active (NGINX Plus)
-

Request Rate

The request rate is the number of requests handled per second.

- Helps you understand traffic patterns
- Ensures your server can handle peak loads effectively.
  Sudden spikes in request rates can indicate traffic surges or potential attacks.

- **Variable:**
  - $request_time

Response Time

Response time metrics indicate how long it takes for the server to process requests and for upstream servers to respond.

- helps ensure that applications are performing well
- helps identify bottlenecks in your infrastructure.

- **Variable:**
  - $request_time, $upstream_response_time (NGINX Open Source)
  - nginx_http_request_time, nginx_http_upstream_response_time (NGINX Plus)

Error Rates

- Monitors error rates (e.g., 4xx and 5xx status codes)
  - Crucial for identifying issues with applications or server configurations.
  - High error rates can indicate problems such as misconfigurations, application bugs, or denial→of→service attacks.
- **Variable:**
  - $status

CPU and Memory Usage

- Understand your NGINX server's resource consumption
  - Can indicate the need for resource optimization or hardware upgrades.
- **Tool/Variable:**
  - System monitoring tools (e.g., top, htop, or vmstat for NGINX Open Source),
  - nginx_process_cpu and nginx_process_mem (NGINX Plus)

SSL Handshake Time

Ensures that secure connections are established quickly and efficiently.

- Long handshake times can indicate issues with SSL configurations
- or the need for hardware improvements.

- **Variable:**
  - $ssl_handshake_time (NGINX Open Source)
  - nginx_ssl_handshake_time (NGINX Plus)

Throughput

Measures the total amount of data sent to clients.

- Monitoring throughput helps understand the data flow
- Ensure that the server can handle the required bandwidth.

- **Variable:**
  - $bytes_sent (NGINX Open Source),
  - nginx_http_bytes_sent (NGINX Plus)

Blocked Requests

Helps identify and respond to security threats.

- High numbers of blocked requests can indicate ongoing attacks or unauthorized access attempts.

- **Variable:**
  - $status (NGINX Open Source),
    - nginx_security_blocked (NGINX Plus)

Cache Hit Ratio (if using NGINX as a reverse proxy)

- **M**easures the effectiveness of your caching strategy.
  - A high cache hit ratio indicates that many requests are being served from the cache
  - This reduces the load on upstream servers and improves response times.
- **Variable:**
  - $upstream_cache_status (NGINX Open Source),
  - nginx_cache_hit (NGINX Plus)

So What?

Scenario 1
Monitoring Active Connections → Spike

Baseline	Mean
Active Connections	500
Request Rate	100 r/s
Response Time	200 ms
Error Rate	1%

Sudden Spike	
Active Connections	1500
Request Rate	300 r/s
Response Time	300 ms
Error Rate	1.5%

Normal Operation	IQR
Active Connections	450→550
Request Rate	95→105 r/s
Response Time	180→220 ms
Error Rate	0.8→1.2%

- Traffic surge? Marketing?
- DDOS Attack? Crosscheck 404 and 403 errors
- Resource Limitations? Load balancing reconfig or added resource?

Scenario 2
Monitoring Active Connections → Gradual Increase

Baseline	Mean
Active Connections	500
Request Rate	100 r/s
Response Time	200 ms
Error Rate	1%

Normal Operation	IQR
Active Connections	450→550
Request Rate	95→105 r/s
Response Time	180→220 ms
Error Rate	0.8→1.2%

Active Connections	500→700→900→100
Request Rate	100→120→150→170 r/s
Response Time	200→220→245→260 ms
Error Rate	1.0→1.2→1.3→1.6%

- Growth in Users? Planning and scaling resourses
- Performance Degradation? Resource saturation?
- Infrastructure scaling? Vertical or Horizontal?

Scenario 3
Monitoring Active Connections with High Variability

Baseline	Mean
Active Connections	500
Request Rate	100 r/s
Response Time	200 ms
Error Rate	1%

High Variability	
Active Connections	400→ 800→ 300→ 900→ 500
Request Rate	90→ 150→ 80→ 170→ 100 r/s
Response Time	190→ 250→ 180→ 270→ 200 ms
Error Rate	1% → 1.4% → 0.9% → 1.6% → 1%

Normal Operation	IQR
Active Connections	450→550
Request Rate	95→105 r/s
Response Time	180→220 ms
Error Rate	0.8→1.2%

- Intermittent Traffic? Periodic Events? Batch work?
- Load Balancing issues? Configuration challenges?
- Application bottlenecks? Resources outside of the apps?

Scenario 1
Monitoring Blocked Requests → Spike

Baseline	Mean
Active Connections	500
Request Rate	100 r/s
Response Time	200 ms
Error Rate	1%
Blocked Requests	5/m

Normal Operation	IQR
Active Connections	450→550
Request Rate	95→105 r/s
Response Time	180→220 ms
Error Rate	0.8→1.2%
Blocked Requests	4-6/m

Sudden Spike	
Active Connections	600
Request Rate	110 r/s
Response Time	210 ms
Error Rate	1.5%
Blocked Requests	50/m

- Security threat?
- Firewall or WAF configuration? Recent changes?
- False Positives? Aggressive security rules? Configuration errors?

Scenario 2
Monitoring Blocked Requests → Gradual Increase

Baseline	Mean
Active Connections	500
Request Rate	100 r/s
Response Time	200 ms
Error Rate	1%
Blocked Requests	5/m

Normal Operation	IQR
Active Connections	450→550
Request Rate	95→105 r/s
Response Time	180→220 ms
Error Rate	0.8→1.2%
Blocked Requests	4-6/m

Sudden Spike	
Active Connections	500→ 550→ 600
Request Rate	100→ 105→ 110 r/s
Response Time	200→ 210→ 220 ms
Error Rate	1% → 1.2% → 1.4%
Blocked Requests	5→ 10→ 20

- Target for attackers? Up the security?
- Security rules adjustment? Real users or not?
- Resource impact? Are the blocks affecting performance?

Scenario
Monitoring Blocked Requests → High Variability

Baseline	Mean
Active Connections	500
Request Rate	100 r/s
Response Time	200 ms
Error Rate	1%
Blocked Requests	5/m

Normal Operation	IQR
Active Connections	450→550
Request Rate	95→105 r/s
Response Time	180→220 ms
Error Rate	0.8→1.2%
Blocked Requests	4-6/m

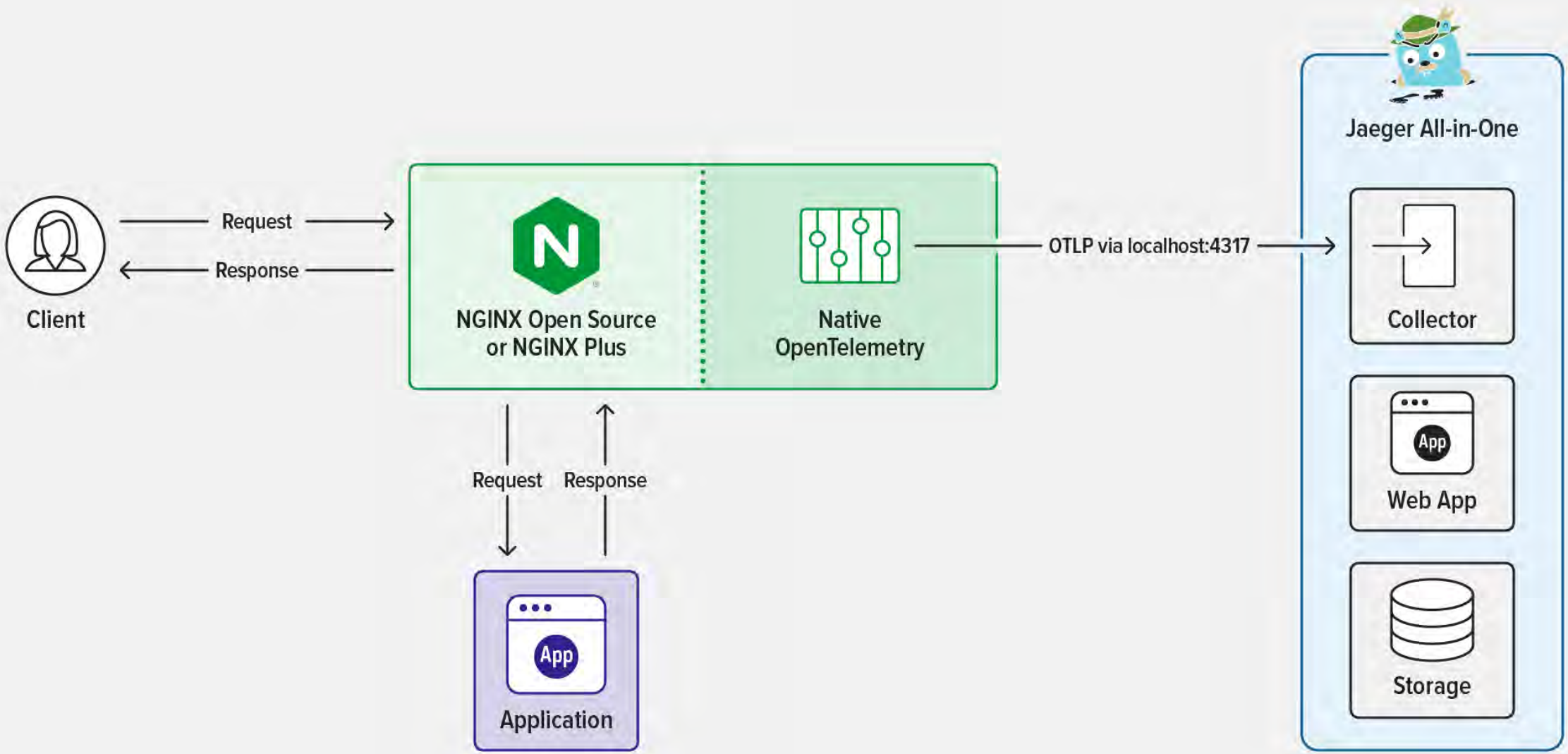
Sudden Spike	
Active Connections	450→ 700→ 500→ 800→ 450
Request Rate	95→ 120→ 100→ 130→ 95 r/s
Response Time	180→ 230→ 200→ 250→ 180 ms
Error Rate	1%→ 1.5%→ 1.2%→ 1.8%→ 1%
Blocked Requests	5 → 30 → 10 → 40 → 5

- Intermittent attack? Malicious actors? Automated scripts?
- Load Balancer or Proxy issues? Inconsistent load caused by blocks?
- Application misconfigurations? Sporadically exploited vulnerabilities?

OpenTelemetry Module for NGINX

nginxinc/nginx-otel (github.com)

- Enables NGINX to send to an Otel collector
  - Fully supports W3C trace context
  - Supports OTLP and gRPC protocols
- Performance
  - Current community modules reduce request processing by ~50%
  - Native module ~10%
- Setup and config are inline with the NGINX application configuration
- Allows dynamic control of trace parameters using cookies, tokens and/or variables
- Prebuilt packages are available, including RedHat and derivatives, Debian, Ubuntu and derivatives


Client
Request
Response
NGINX Open Source
or NGINX Plus
Native
OpenTelemetry
OTLP via localhost:4317
Jaeger All-in-One
Collector
App
Web App
Storage
Request
Response
App
Application

Example Configuration

```
load_module modules/ngx_otel_module.so;

events {
}

http {

    otel_exporter {
        endpoint localhost:4317;
    }

    server {
        listen 127.0.0.1:8080;

        location / {
            otel_trace         on;
            otel_trace_context inject;

            proxy_pass http://backend;
        }
    }
}
```

- Otel_exporter – specifies Otel data export parameters
  - Endpoint – address to accept telemetry data
  - Interval – interval (max) between exports (5s)
  - Batch_size – max spans sent in one batch per worker (512)
  - Batch_count – number of pending batches per worker (4)
- Otel_trace – enables or disables tracing
  - Can be a variable
- Otel_trace_context – propagation directives
  - Extract | inject | propagate | ignore

Prometheus and NGINX

NGINX Prometheus Exporter is an open-source exporter that translates NGINX metrics into a format that Prometheus can scrape.

- It works by reading the NGINX status module or the Plus API.
- https://github.com/nginxinc/nginx-prometheus-exporter

- NGINX OSS
- NGINX Plus
- NGINX Ingress Controller
- NGINX Gateway Fabric

Name	Type	Description
nginx_connections_accepted	Counter	Accepted client connections.
nginx_connections_active	Gauge	Active client connections.
nginx_connections_handled	Counter	Handled client connections.
nginx_connections_reading	Gauge	Connections where NGINX is reading the request header.
nginx_connections_waiting	Gauge	Idle client connections.
nginx_connections_writing	Gauge	Connections where NGINX is writing the response back to the client.
nginx_http_requests_total	Counter	Total http requests.

Concluding

- Key Metrics for NGINX
- Scenarios and Analysis
- Hardware vs. Software Concerns
- Security and Performance Optimization
- It's not just the metrics

"The most effective debugging tool is still careful thought, coupled with judiciously placed print statements."

- Brian Kernighan *Unix for Beginners* 1979

Thanks!

Docs

Slides on GitHub

Linkedin: in/davemc

Repos

NGINX Prometheus Exporter

OTel Module for NGINX