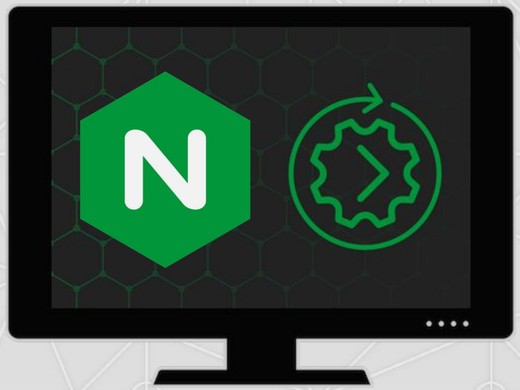
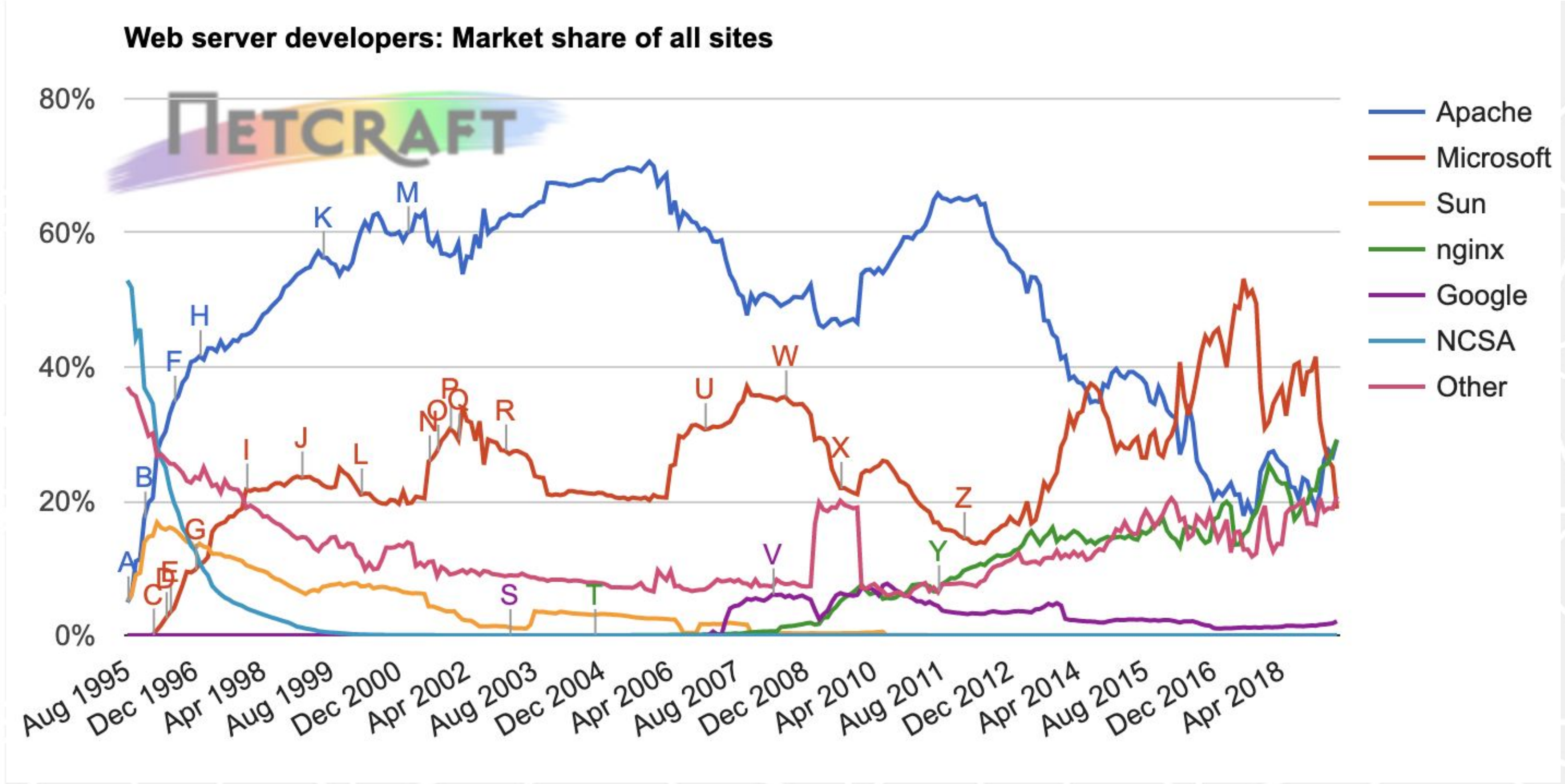
NGINX: Basics and Best Practices

Agenda



Introducing Nginx



NGINX is the **most used web server** on the internet

Source: w3techs, May 2019

About NGINX, Inc.

- Founded in 2011, NGINX Plus first released in 2013
- Offices in SF, London, Cork, Moscow, Singapore, Japan, Sydney, and Moscow
- 1,500+ commercial customers
- 200+ employees
- **Acquired by F5 Networks in May 2019**

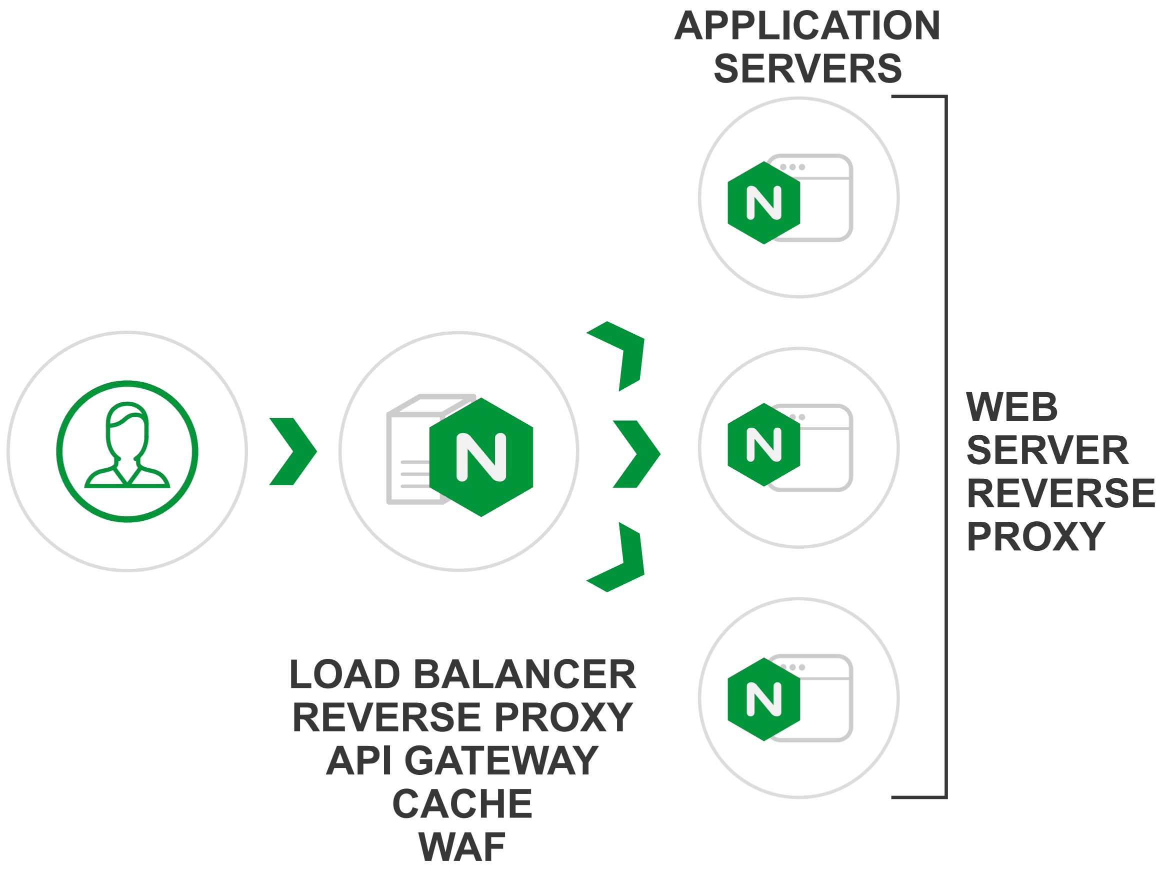
What is NGINX?

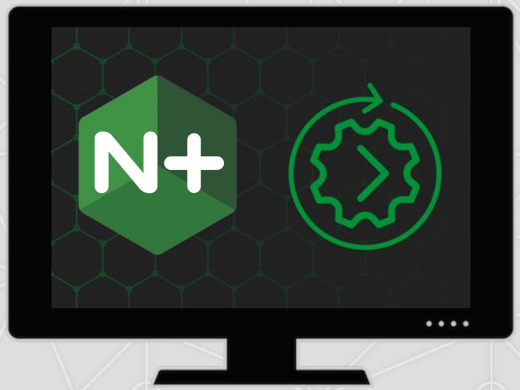
NGINX

- Basic load balancer
- Reverse Proxy and Web Server
- Content Cache
- SSL termination
- Rate limiting
- Basic authentication

NGINX PLUS

- Active health checks
- Session persistence
- DNS service discovery integration
- Cache-purging API
- JWT authentication and OpenID Connect
- Live Activity monitoring (100+ real time metrics)
- Dynamic Modules
- API for Dynamic reconfiguration, Cache-purge, key-value store
- High Availability, Cluster State sync ……much more.

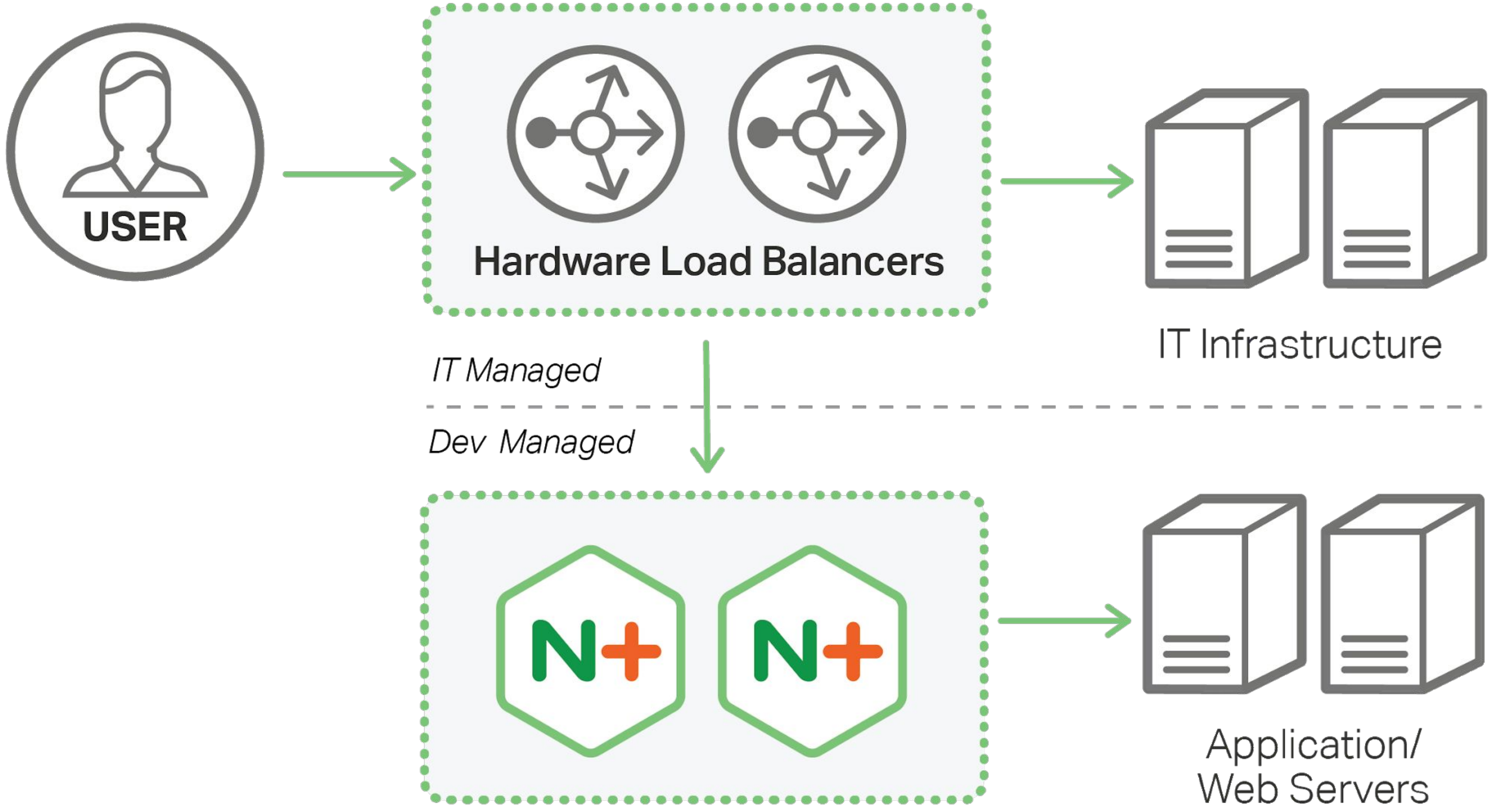




ADC Augment and Modernization

What's happening now

Traditional Application Infrastructure are being augmented



ADC Augment - key use cases

Key use cases

ADC Augment

Enhancing existing app environments

ADC for Multi-Cloud

Scale and Secure Apps across multi-cloud

API Management

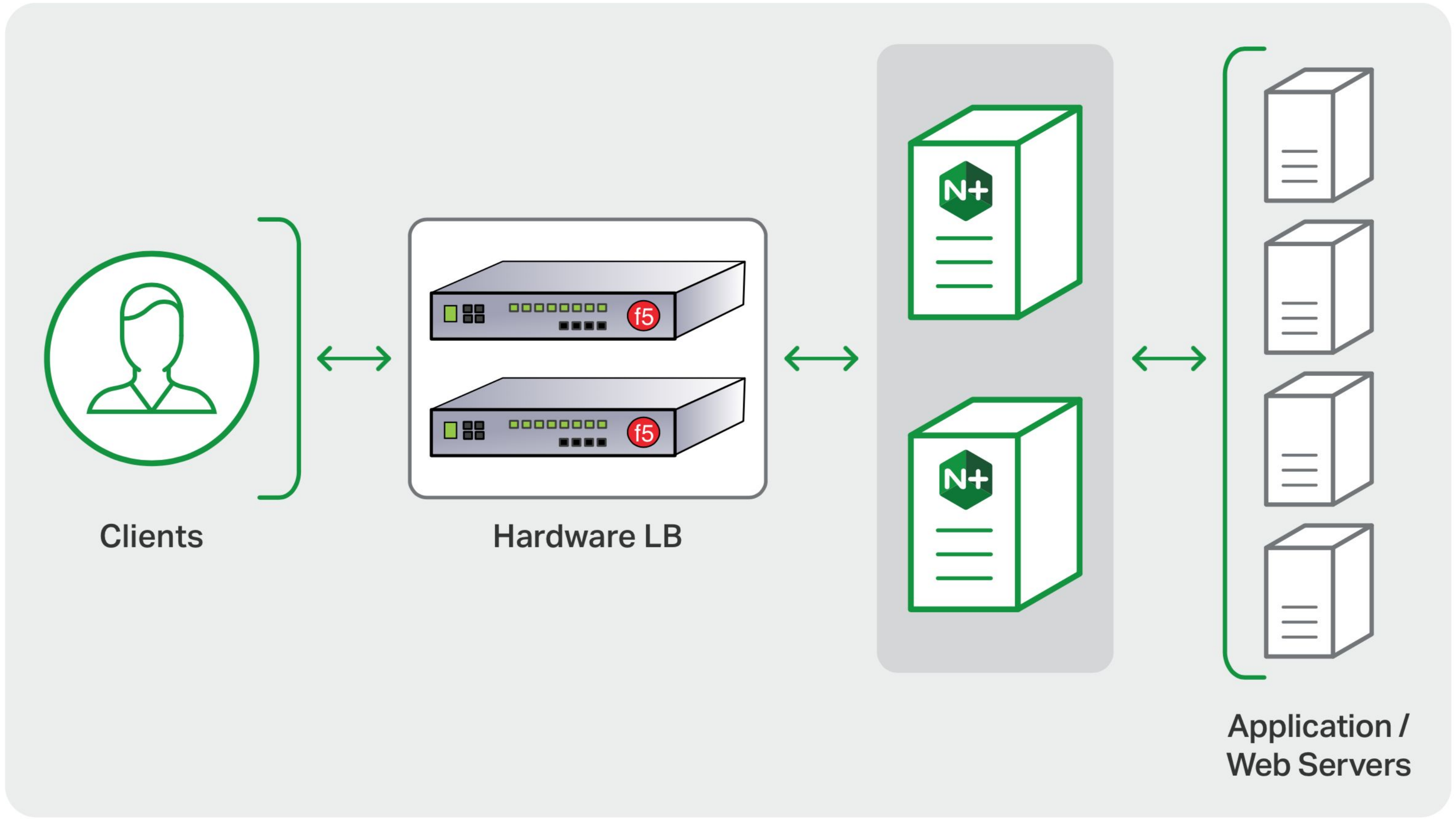
End-to-end API lifecycle services

Kubernetes Integration

Flexible and scalable app services

1. Augment Traditional Load Balancers

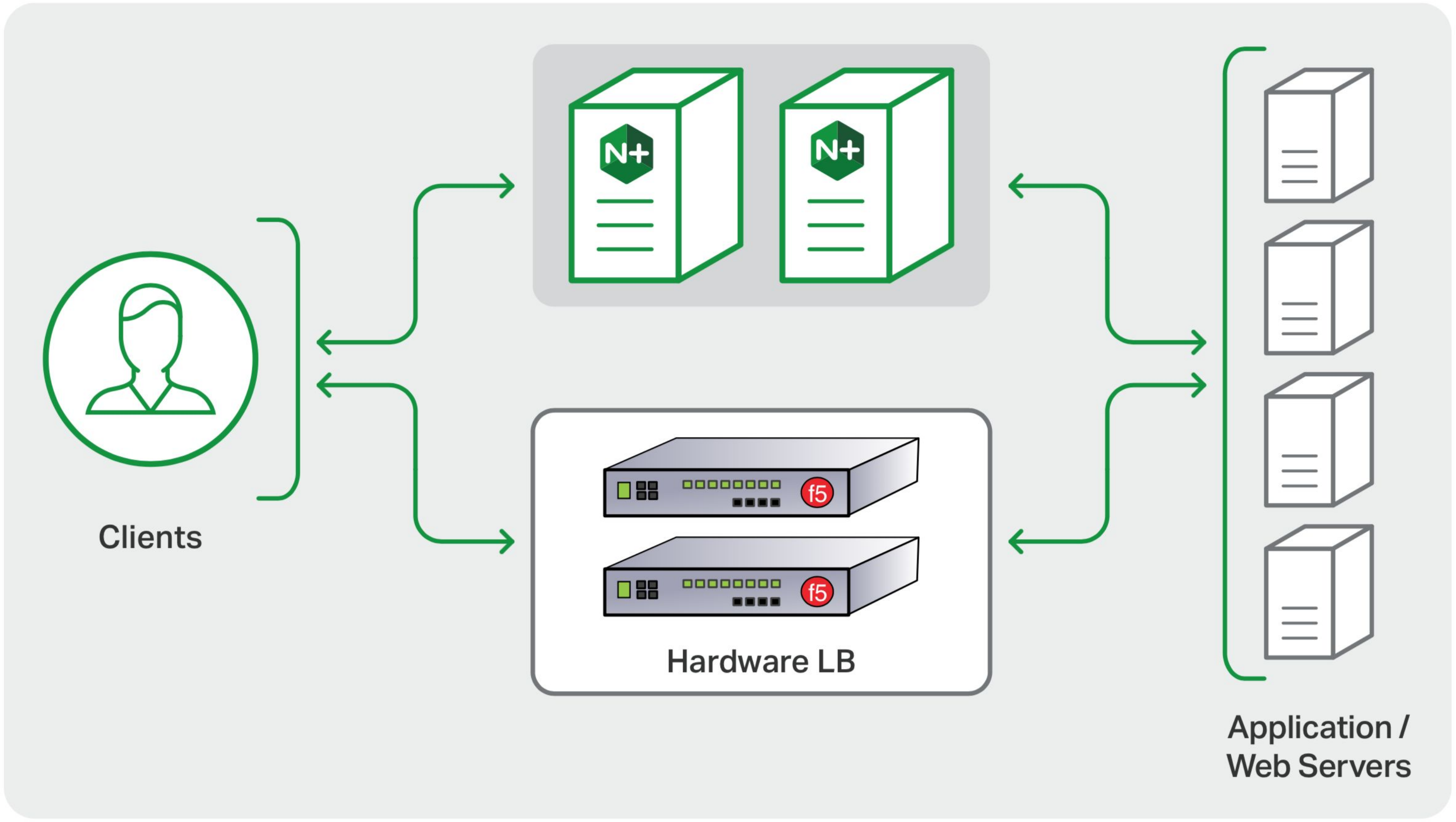
Traditional Application Infrastructure are being augmented



- Easiest way to introduce NGINX into your network
- Hardware layer 4 load balancer to NGINX
- Can start small with one application being behind NGINX and then expand

2. NGINX Alongside Hardware ADCs

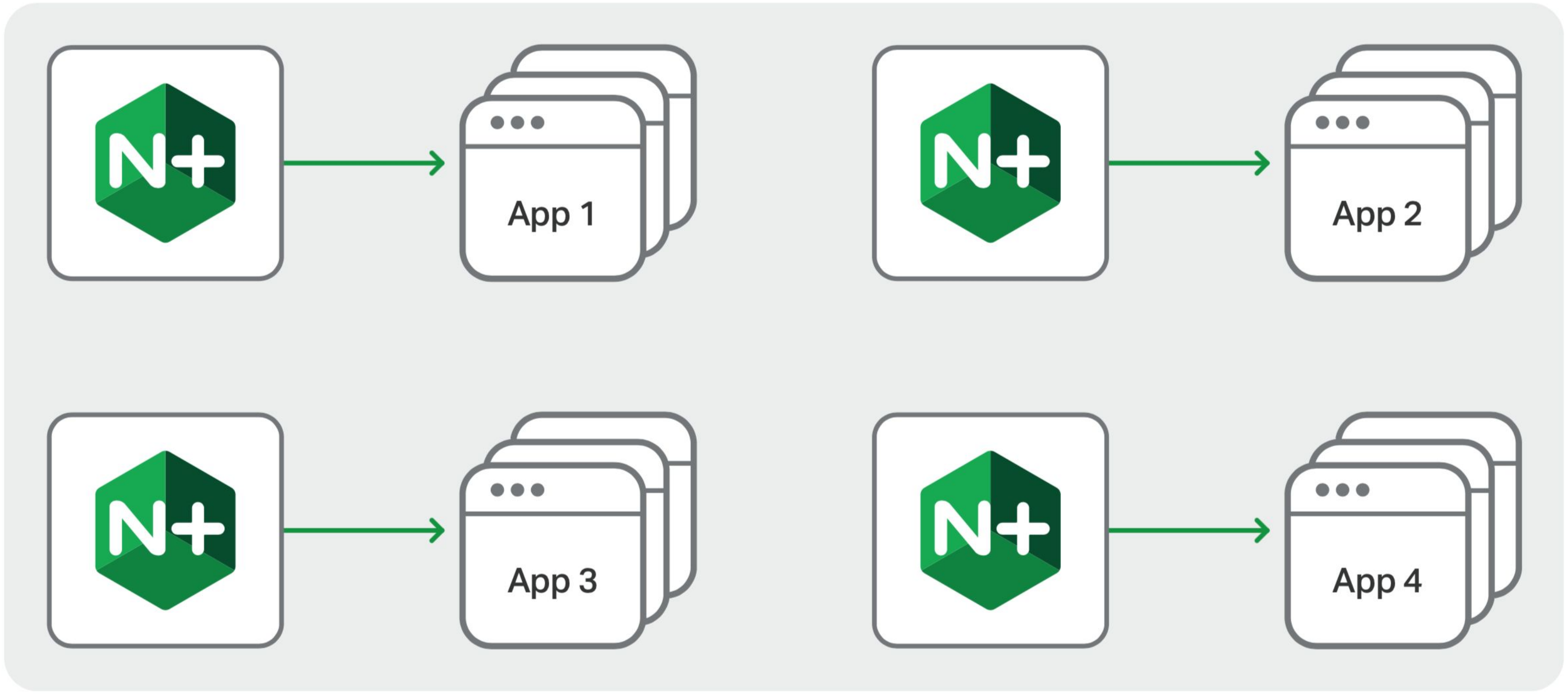
Offload or Migrate new application workloads



- Parallel NGINX deployment
- Good architecture if adopting public cloud while still keeping private datacenter
- Can also start small with one application being behind NGINX and then expand

3. Micro Load Balancers/Gateways

Legacy Hardware ADC replace to a application centric architecture

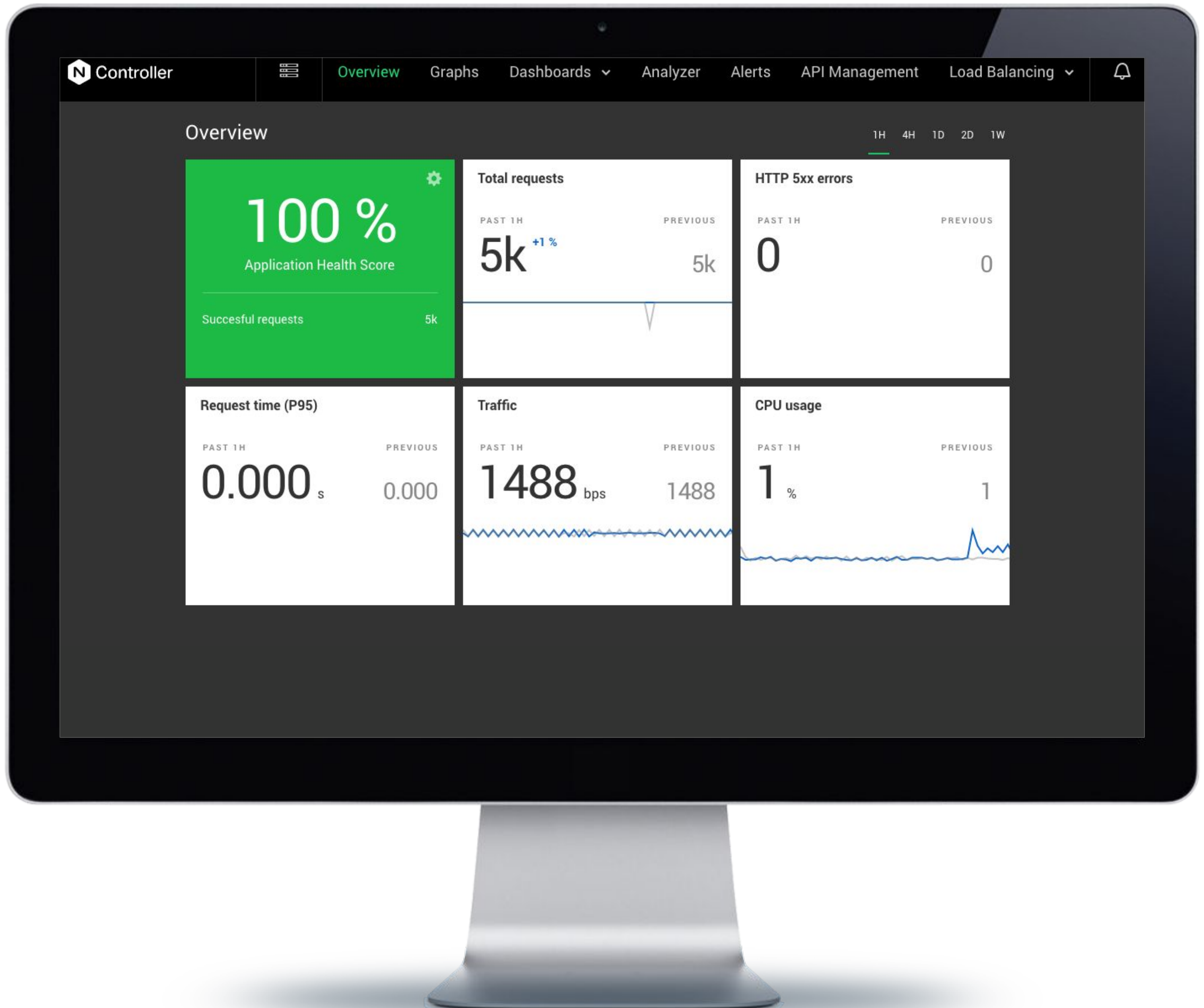


- Load balancer per application
- Load balancer per customer for SaaS providers
- Configuration stored along with application in GitHub
- Fully portable

What is the NGINX Controller?

Centralized Monitoring and management

- Alerting
- API management
- Load balancer management
- Configuration analysis
- Customizable dashboards
- Monitoring



Overview

100 %
Application Health Score
Succesful requests 5k

Total requests
PAST 1H 5k +1 %
PREVIOUS 5k

HTTP 5xx errors
PAST 1H 0
PREVIOUS 0

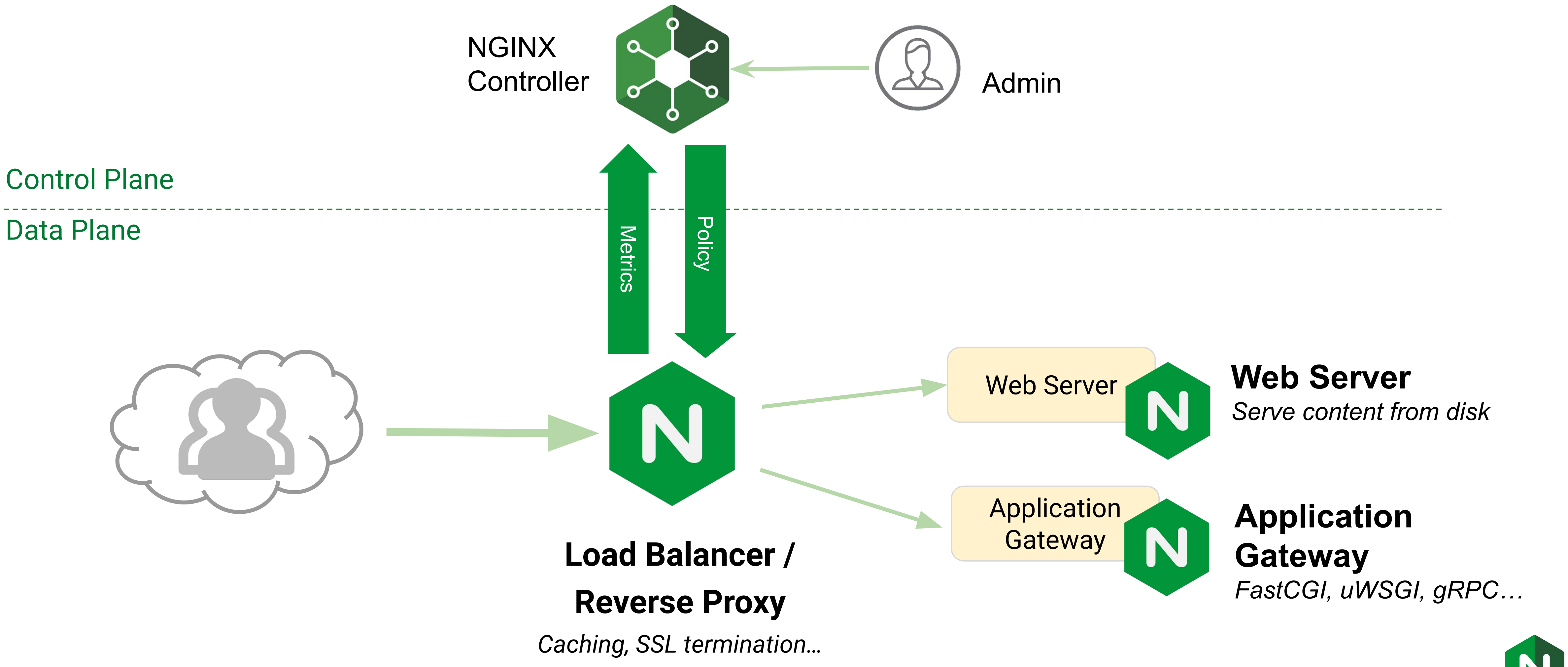
Request time (P95)
PAST 1H 0.000 s
PREVIOUS 0.000

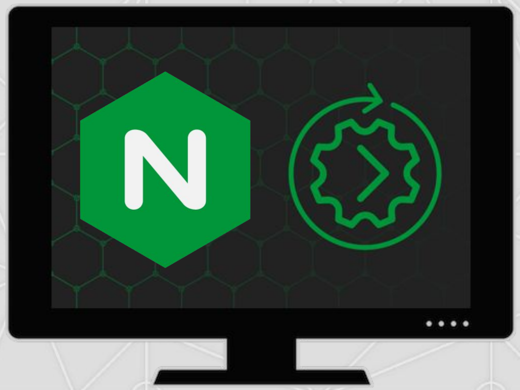
Traffic
PAST 1H 1488 bps
PREVIOUS 1488

CPU usage
PAST 1H 1 %
PREVIOUS 1

What is the NGINX Controller?

Nginx and the Nginx Controller





Installing NGINX

Nginx Installation Options

- **Official NGINX repo**
  - **Mainline (recommended)** - **Actively developed; new minor releases made every 4-6 weeks with new features and enhancements.**
  - **Stable** - Updated only when critical issues or security vulnerabilities need to be fixed.
  - **NGINX PLUS -** receives all new features, once they have been tested and proven in NGINX mainline. Additional enterprise-specific features are included in NGINX Plus.
- **OS vendor and other 3rd party repos**
  - Not as frequently updated; e.g. Debian Jessie (8.9) has NGINX 1.6.2
  - Typically built off NGINX mainline branch, sometimes with 3rd party mods
- **Compile from source**
  - Most difficult.-Download the latest version of the NGINX source code, configure, build and install it. You will have the option of building various Nginx module

NGINX Installation: Debian/Ubuntu

Create `/etc/apt/sources.list.d/nginx.list` with the following contents:

```
deb http://nginx.org/packages/mainline/OS/ CODENAME nginx

deb-src http://nginx.org/packages/mainline/OS/ CODENAME nginx
```

- OS – `ubuntu` or `debian` depending on your distro
- CODENAME:
  - `- jessie or stretch for debian`
  - `- trusty, xenial, artful, or bionic for ubuntu`

```
$ wget http://nginx.org/keys/nginx_signing.key
$ apt-key add nginx_signing.key
$ apt-get update
$ apt-get install –y nginx
$ /etc/init.d/nginx start
```

NGINX Installation: CentOS/Red Hat

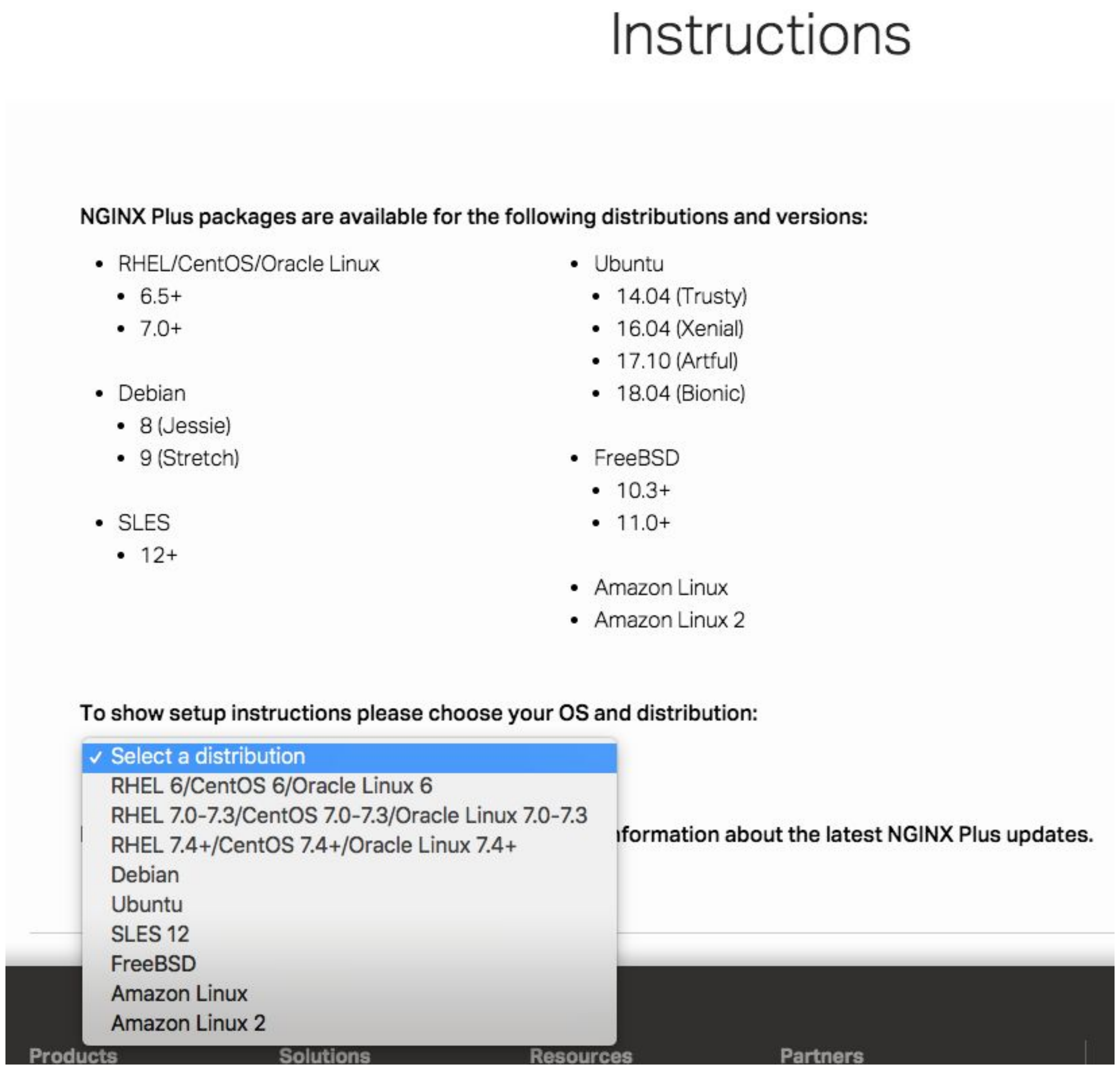
Create `/etc/yum.repos.d/nginx.repo` with the following contents:

```
[nginx]
name=nginx repo
baseurl=http://nginx.org/packages/mainline/OS/OSRELEASE/$basearch/
gpgcheck=0
enabled=1
```

- OS -- `rhel` or `centos` depending on your distro
- OSRELEASE -- `6` or `7` for 6.x or 7.x versions, respectively

```
$ yum –y install nginx
$ systemctl enable nginx
$ systemctl start nginx
$ firewall-cmd --permanent --zone=public --add-port=80/tcp
$ firewall-cmd --reload
```

NGINX Plus Installation



Instructions

NGINX Plus packages are available for the following distributions and versions:

- RHEL/CentOS/Oracle Linux
  - 6.5+
  - 7.0+
- Debian
  - 8 (Jessie)
  - 9 (Stretch)
- SLES
  - 12+
- Ubuntu
  - 14.04 (Trusty)
  - 16.04 (Xenial)
  - 17.10 (Artful)
  - 18.04 (Bionic)
- FreeBSD
  - 10.3+
  - 11.0+
- Amazon Linux
- Amazon Linux 2

To show setup instructions please choose your OS and distribution:

✓ Select a distribution
RHEL 6/CentOS 6/Oracle Linux 6
RHEL 7.0-7.3/CentOS 7.0-7.3/Oracle Linux 7.0-7.3
RHEL 7.4+/CentOS 7.4+/Oracle Linux 7.4+
Debian
Ubuntu
SLES 12
FreeBSD
Amazon Linux
Amazon Linux 2

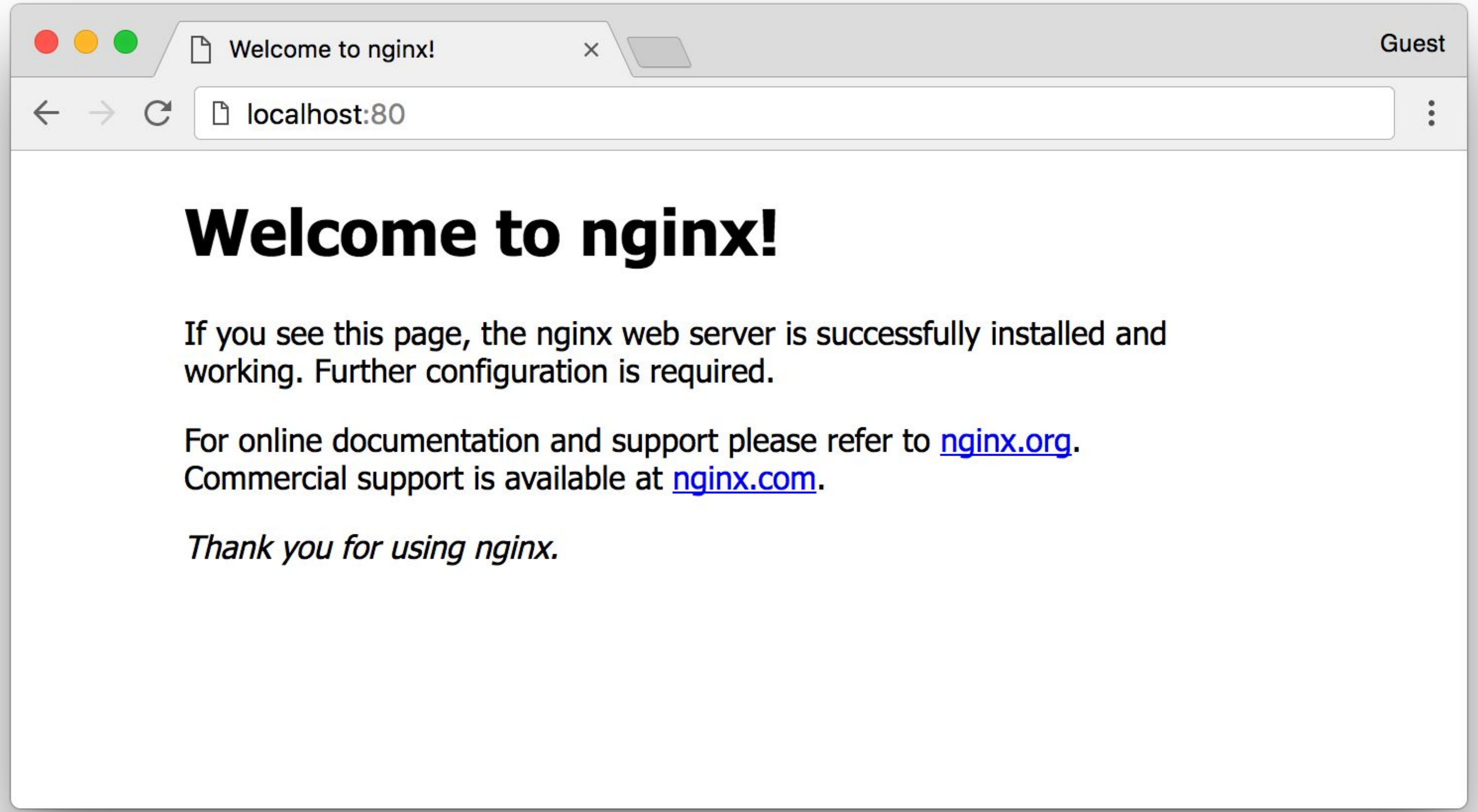
ıformation about the latest NGINX Plus updates.

- Visit `cs.nginx.com/repo_setup`
- Select OS from drop down list
- Instructions similar to OSS installation
- Mostly just using a different repo and installing client certificate

Verifying Installation

```
$ nginx -v

nginx version: nginx version: nginx/1.15.7 (nginx-plus-r17)



$ ps -ef | grep nginx

root       1088      1  0 19:59 ?        00:00:00 nginx: master process /usr/sbin/nginx -c
/etc/nginx/nginx.conf

nginx      1092   1088  0 19:59 ?        00:00:00 nginx: worker process
```

Verifying Installation


Welcome to nginx!

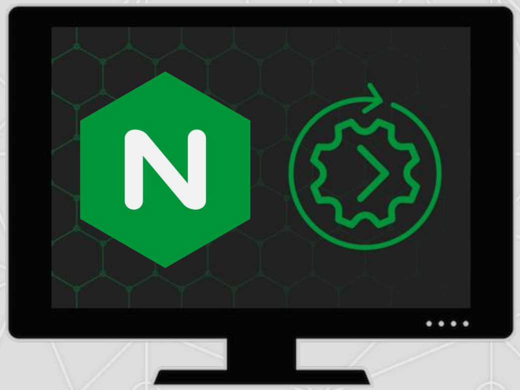
localhost:80

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.



Essential files, commands and directories

Key NGINX Commands

`nginx -h`	Shows all command line options
`nginx -t`	Configuration syntax check
`nginx -T`	Displays full, concatenated configuration
`nginx -V`	Shows version and build details
`nginx –s reload`	Gracefully reload NGINX processes

```
$ sudo nginx –t && sudo nginx –s reload
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

```
$ sudo nginx -T > nginx_support_mm-dd-yy.txt
```

Key System Commands

`ps aux | grep nginx`	To check running processes
`ps -ef --forest | grep nginx`	To check running processes (Show Process Hierarchy in Forest Format)

`service nginx status`
`systemctl status nginx`
	Show Nginx Status

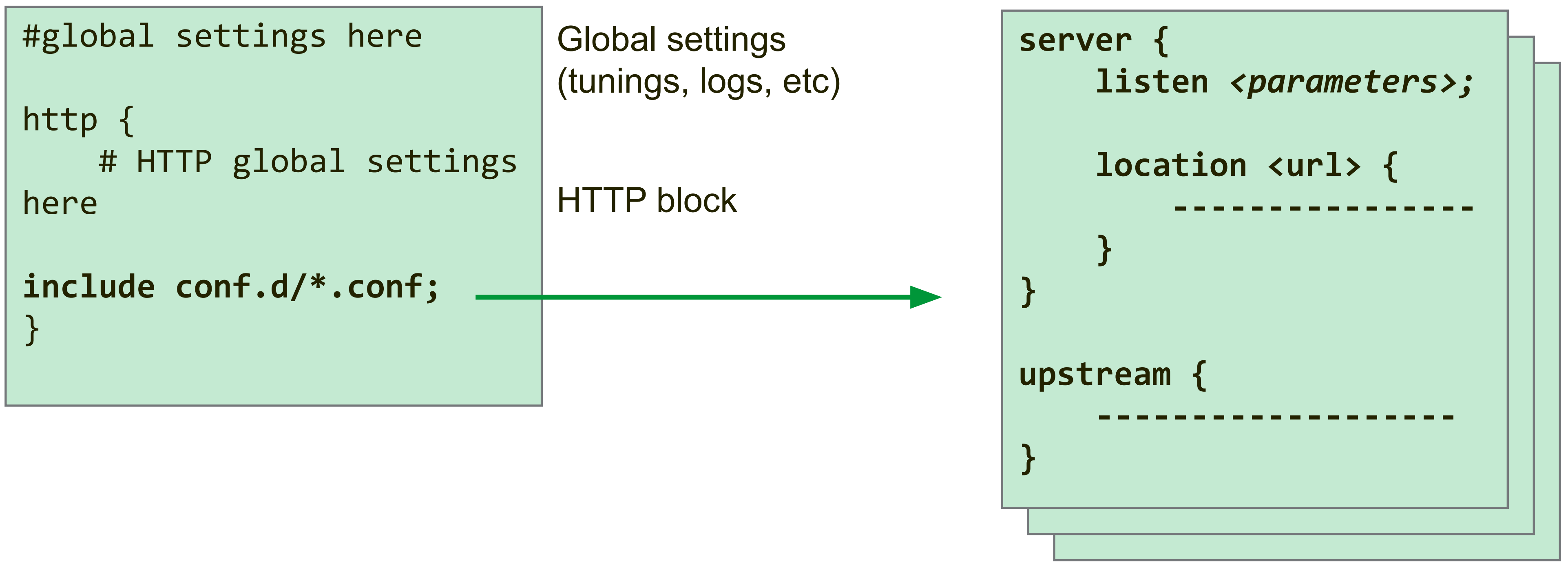
`netstat -tulpn`	Information and statistics about protocols in use and current TCP/IP network connections.
`sudo lsof -i -P -n`	Check the listening ports and applications on linux

```
# Path to executable path
$ /usr/sbin/nginx

# Default Log Path
$ /var/log/nginx
```

Key Files and Directories

- **`/etc/nginx/`**   `# Where all NGINX configuration is stored`
- **`/etc/nginx/nginx.conf`** `# Top-level NGINX configuration, should not require much modification`
- **`/etc/nginx/conf.d/*.conf`** `# Where your HTTP/S configuration for virtual servers and upstreams goes, e.g.` `www.example.com.conf`
- **`/etc/nginx/stream.d/*.conf`** `# Where your TCP/UDP Streams for virtual servers and upstreams goes, e.g.` `DNS_53.conf`
- **`/var/log/nginx/access.log`** `# Details about requests and responses`
- **`/var/log/nginx/error.log`** `# Details about NGINX errors`

Key Files and Directories



`/etc/nginx/`

`nginx.conf`

```
#global settings here

http {
    # HTTP global settings
here

include conf.d/*.conf;
}
```

Global settings (tunings, logs, etc)

HTTP block

`/etc/nginx/conf.d/`

`example.com.conf`

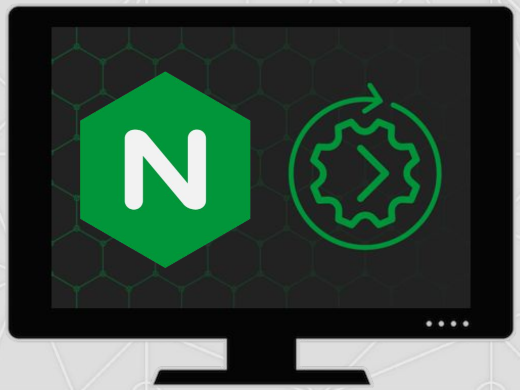
```
server {
    listen <parameters>;

    location <url> {
        ----------------
    }
}

upstream {
    -------------------
}
```

Listen for requests

Rules to handle each request

Optional: upstreams configurations in same file,

`something.com.conf.disabled` Not loaded



Basic configurations

Simple Virtual Server

```
server {
    listen       80 default_server;
    server_name  www.example.com;

   # ...
}
```

- `server` defines the context for a virtual server
- `listen` specifies IP/port NGINX should listen on. No IP means bind to all IPs on system
- `server_name` specifies hostname of virtual server

Basic Web Server Configuration

```
server {
    listen       80 default_server;
    server_name  www.example.com;

    location / {
        root   /usr/share/nginx/html;
        index  index.html index.htm;

    }
}
```

- `root` specifies directory where files are stored
- `index` defines files that will be used as an index

- `www.example.com maps to /usr/share/nginx/html/index.html (then index.htm)`
- `www.example.com/i/file.txt -> /usr/share/nginx/html/i/file.txt`

Multiplexing Multiple Sites on One IP

```
# www.example.com.conf
server {
    listen       80 default_server;
    server_name  www.example.com;
    # ...
}
# www.example2.com.conf
server {
    listen       80;
    server_name  www.example2.com;
    # ...
}
# www.example3.com.conf
server {
    listen       80;
    server_name  www.example3.com;
    # ...
}
```

- NGINX can multiplex a single IP/port using the `Host:` header.
- `default_server` defines the virtual server to use if `Host` header is empty. It is best practice to have a `default_server`.

Basic SSL Configuration

```
server {
    listen       80 default_server;
    server_name  www.example.com;
    return 301 https://$server_name$request_uri;
}
server {
    listen 443 ssl default_server;
    server_name  www.example.com;
    ssl_certificate cert.crt;
    ssl_certificate_key cert.key;
    ssl_ciphers HIGH;

    location / {
        root   /usr/share/nginx/html;
        index  index.html index.htm;
    }
}
```

- Force all traffic to SSL is good for security, customer trust and SEO
- Use Let's Encrypt to get free SSL certificates
- Use Mozilla SSL Configuration Generator to generate recommended nginx SSL configurations: `https://mozilla.github.io/server-side-tls/ssl-config-generator/`

Basic HTTP/2 Configuration

```
server {
    listen 443 ssl http2 default_server;
    server_name  www.example.com;

    ssl_certificate cert.crt;
    ssl_certificate_key cert.key;
}
```

- HTTP/2 improves performance with little to no backend changes
- Add `http2` parameter to `listen` directive of existing SSL-enabled virtual server. HTTP/2 is only supported with SSL in all browsers.
- NGINX only does HTTP/2 client side, server side is still HTTP/1.1. gRPC is a special case.
- Note: HTTP/2 requires OpenSSL 1.0.2 or later to work properly

Basic Reverse Proxy Configuration

```
server {
    location ~ ^(.+\.php)(.*)$ {
        fastcgi_split_path_info ^(.+\.php)(.*)$;

        # fastcgi_pass 127.0.0.1:9000;
        fastcgi_pass unix:/var/run/php7.0-fpm.sock;

        fastcgi_index index.php;
        include fastcgi_params;
    }
}
```

- Requires PHP FPM:
  `apt-get install -y php7.0-fpm`
- Can also use PHP 5
- Similar directives available for uWSGI and SCGI.
- Additional PHP FPM configuration may be required

Basic Load Balancing Configuration

```
upstream my_upstream {
    server server1.example.com:80;
    server server2.example.com:80;
    least_conn;
}
server {
    location / {
        proxy_set_header Host $host;
        proxy_pass http://my_upstream;
    }
}
```

- `upstream` defines the load balancing pool
- Default load balancing algorithm is round robin. Others available:
  - `least_conn` selects server with least amount of active connections
  - `least_time` factors in connection count and server response time. Available in NGINX Plus only.
- `proxy_pass` links virtual server to `upstream`
- By default NGINX rewrites `Host` header to name and port of proxied server. `proxy_set_header` overrides and passes through original client `Host` header.

Layer 7 Request Routing

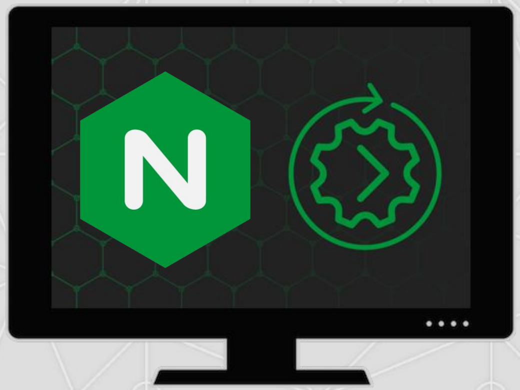
```
server {
    # ...

    location /service1 {
        proxy_pass http://upstream1;
    }

    location /service2 {
        proxy_pass http://upstream2;
    }

    location /service3 {
        proxy_pass http://upstream3;
    }
}
```

- `location` blocks are used to do Layer 7 routing based on URL
- Regex matching can also be used in `location` blocks

Basic Caching Configuration

```
proxy_cache_path /path/to/cache levels=1:2
                 keys_zone=my_cache:10m max_size=10g
                 inactive=60m use_temp_path=off;

server {
    location / {
        proxy_cache my_cache;
        # proxy_cache_valid 5m;
        proxy_set_header Host $host;
        proxy_pass http://my_upstream;
    }
}
```

- `proxy_cache_path` defines the parameters of the cache.
- `keys_zone` defines the size of memory to store cache keys in. A 1 MB zone can store data for about 8,000 keys.
- `max_size` sets upper limit of cache size. Optional.
- `inactive` defines how long an object can stay in cache without being accessed. Default is 10 m.
- `proxy_cache` enables caching for the context it is in



Advanced configurations

Modifications to main nginx.conf

```
user  nginx;
worker_processes auto;

# ...

http {
    # ...

    keepalive_timeout 300s;
    keepalive_requests 100000;
}
```

- Set in main nginx.conf file
- Default value for worker_processes varies on system and installation source
- `auto` means to create one worker process per core. This is recommended for most deployments.
- `keepalive_timeout` controls how long to keep idle connections to clients open. Default: 75s
- `keeplive_requests` Max requests on a single client connection before its closed.Default: 100
- `keepalive_*` can also be set per virtual server

HTTP/1.1 Keepalive to Upstreams

```
upstream my_upstream {
    server server1.example.com;
    keepalive 32;
}
server {
    location / {
        proxy_set_header Host $host;
        proxy_http_version 1.1;
        proxy_set_header Connection "";

        proxy_pass http://my_upstream;
    }
}
```

- `keepalive` enables TCP connection cache
- By default NGINX uses HTTP/1.0 with `Connection: Close`
- `proxy_http_version` upgrades connection to HTTP/1.1
- `proxy_set_header` enables keepalive by clearing `Connection: Close` HTTP header

SSL Session Caching

```
server {
    listen 443 ssl default_server;
    server_name  www.example.com;

    ssl_certificate cert.crt;
    ssl_certificate_key cert.key;

    ssl_session_cache   shared:SSL:10m;
    ssl_session_timeout 10m;
}
```

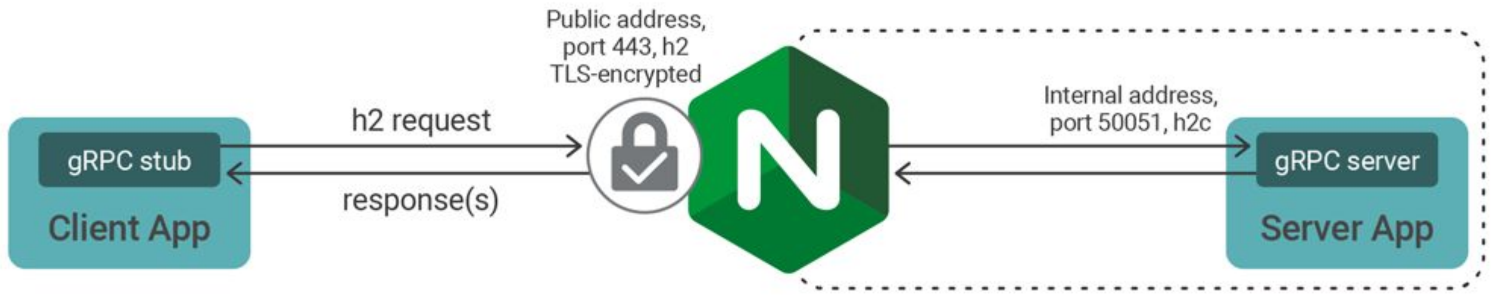
- Improves SSL/TLS performance
- 1 MB session cache can store about 4,000 sessions
- Cache shared across all NGINX workers

Advanced Caching Configuration

```
proxy_cache_path /path/to/cache levels=1:2
                 keys_zone=my_cache:10m max_size=10g
                 inactive=60m use_temp_path=off;

server {
    location / {
        proxy_cache my_cache;
        proxy_cache_lock on;
        proxy_cache_revalidate on;
        proxy_cache_use_stale error timeout updating
                 http_500 http_502 http_503 http_504;
        proxy_cache_background_update on;

        proxy_set_header Host $host;
        proxy_pass http://my_upstream;
    }
}
```

- `proxy_cache_lock` instructs NGINX to only send one request to the upstream when there are multiple cache misses for the same file.
- `proxy_cache_revalidate` instructs NGINX to use `If-Modified-Since` when refreshing cache.
- `proxy_cache_use_stale` instructs NGINX to serve stale content instead of an error.
- `proxy_cache_background_update` instructs NGINX to do all cache updates in the background. Combined with `proxy_cache_use_stale updating`, stale content will be served.

gRPC Proxying with SSL Termination


Public address, port 443, h2 TLS-encrypted
h2 request
response(s)
gRPC stub
Client App
Internal address, port 50051, h2c
gRPC server
Server App

```
server {
    listen 443 ssl http2;
     ssl_certificate     server.crt;
     ssl_certificate_key server.key;

    location / {
        grpc_pass grpc://localhost:50051;
    }
}
```

- Configure SSL and HTTP/2 as usual
- Go sample application needs to modified to point to NGINX IP Address and port.

Active Health Checks

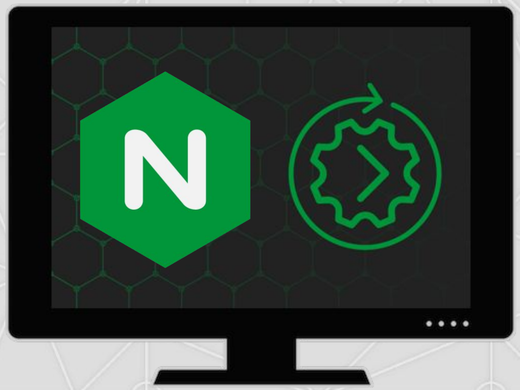
```
upstream my_upstream {
    zone my_upstream 64k;
    server server1.example.com slow_start=30s;
    server server2.example.com slow_start=30s;

}
server {
    # ...
    location @health {
        internal;
        health_check interval=5s uri=/test
            match=statusok;
        proxy_set_header HOST www.example.com;
        proxy_pass http://my_upstream;
    }

match statusok {
    # Used for /test.php health check
    status 200;
    header Content-Type = text/html;
    body ~ "i’m is alive";
}
```

- Polls `/test` every 5 seconds
- If response is not `200`, server marked as failed
- If response body does not contain “I’m alive”, server marked as failed
- Recovered/new servers will slowly ramp up traffic over 30 seconds
- Exclusive to NGINX Plus

Sticky Cookie Session Persistence

```
upstream my_upstream {
    server server1.example.com;
    server server2.example.com;

    sticky cookie name expires=1h
        domain=.example.com path=/;
}
```

- NGINX will insert a cookie using the specified *name*
- `expires` defines how long the cookie is valid for. The default is for the cookie to expire at the end of the browser session.
- `domain` specifies the domain the cookie is valid for. If not specified, domain field of cookie is left blank
- `path` specifies the path the cookie is set for. If not specified, path field of cookie is left blank
- Exclusive to NGINX Plus



Monitoring and Logging

NGINX Access Logs

```
access_log  /var/log/nginx/access.log;
```

```
192.168.179.1 - - [15/May/2017:16:36:25 -0700] "GET / HTTP/1.1" 200 612 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X
10_12_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.36" "-"
192.168.179.1 - - [15/May/2017:16:36:26 -0700] "GET /favicon.ico HTTP/1.1" 404 571 "http://fmemon-redhat.local/"
"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110
Safari/537.36" "-"
192.168.179.1 - - [15/May/2017:16:36:31 -0700] "GET /basic_status HTTP/1.1" 200 100 "-" "Mozilla/5.0 (Macintosh; Intel
Mac OS X 10_12_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.36" "-"
```

- Enabled by default. Can be disabled with the `access_log off` directive.
- Nginx uses the **combined log format** (also used by Apache) and includes IP address, date, request , referrer, user agent, etc. You can add additional NGINX variables, e.g. timing and a Log format configurable with the `log_format` directive
- Can enable access logs at a virtual server scope

NGINX Error Logs

```
error_log  /var/log/nginx/error.log [level];
```

```
2018/03/22 11:29:08 [error] 12696#12696: upstream timed out (110: Connection timed out) while connecting to upstream,
health check "" of peer 10.70.88.24:8832 in upstream "Dev.InternalApi"
2018/03/22 11:29:23 [error] 12696#12696: upstream timed out (110: Connection timed out) while connecting to upstream,
health check "" of peer 10.70.88.15:8832 in upstream "Dev.InternalApi"
2018/03/23 15:25:35 [error] 19997#0: *1 open() "/var/www/nginx-default/phpmy-admin/scripts/setup.php" failed (2: No such
file or directory), client: 80.154.42.54, server: localhost, request: "GET /phpmy-admin/scripts/setup.php HTTP/1.1",
host: "www.example.com"
```

- Enabled by default. Can be disabled with the `error_log off` directive.
- Can enable access logs at a virtual server scope

Error Log Levels

```
error_log  /var/log/nginx/error.log [level];
```

debug	Detailed Trace
info	General Info
notice	Something Normal
warn	Something Strange
error	Unsuccessful
crit	Important Issue(s)
alert	Fix Now!
emerg	Unusable

Extra examples

```
log_format simple escape=json
 '{"timestamp":"$time_iso8601","client":"$remote_addr","uri":"$uri","status":"$status"}';

server {
    server_name  www.example.com;
    access_log /var/log/nginx/example.log simple;
    error_log syslog:server=192.168.1.1 debug;
}

server {
    server_name  www.example2.com;
     map $status $condition {
        ~^[23] 0;
        default 1;
    }
    access_log /var/log/nginx/example2.log simple custom if=$condition;
    error_log  /var/log/nginx/example2_error.log info;
}
```

Example log parsing commands:

`tail -f 10 error.log`	Tail error logs (last 10 lines)
`tail -f 10 access.log | grep 127.0.0.1`	Tail and grep (filter) access logs
`cat access.log | cut -d '"' -f3 | cut -d ' ' -f2 | sort | uniq -c | sort -rn`	Sort access by Response Codes
`awk '($9 ~ /404/)' access.log | awk '{print $7}' | sort | uniq -c | sort -rn`	Which links are broken (HTTP 404)?
`awk -F\" '{print $2}' access.log | awk '{print $2}' | sort | uniq -c | sort -r`	What are my most requested links?

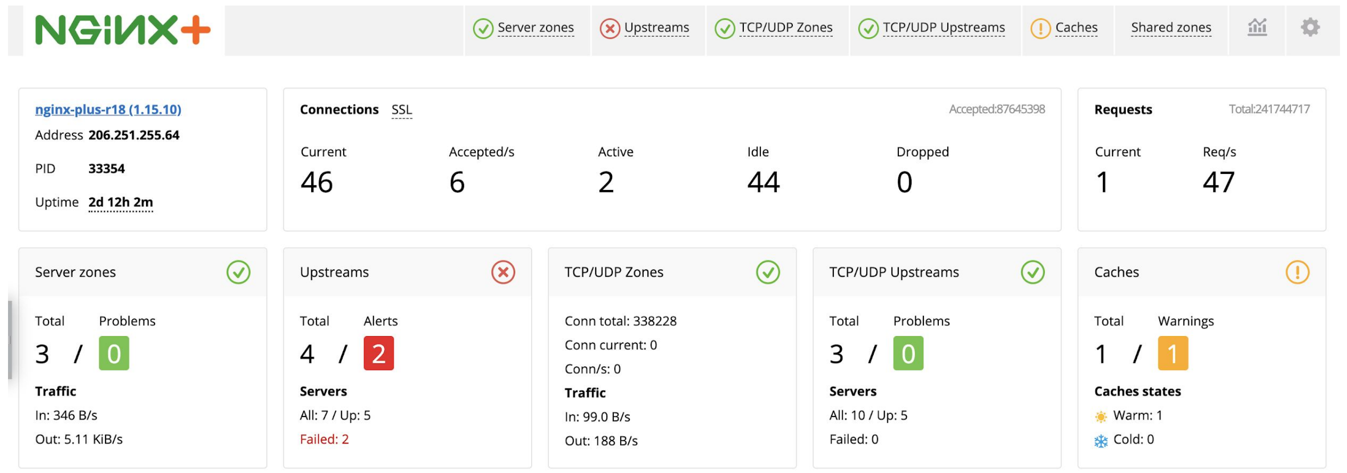
NGINX Stub Status Module

```
server {
    location /basic_status {
        stub_status;
    }
}
```

```
$ curl http://127.0.0.1/basic_status
Active connections: 1
server accepts handled requests
 7 7 7
Reading: 0 Writing: 1 Waiting: 0
```

- Provides aggregated NGINX statistics
- Restrict access so it's not publicly visible

NGINX Plus Extended Status Module



nginx-plus-r18 (1.15.10)
Address 206.251.255.64
PID 33354
Uptime 2d 12h 2m

Connections SSL Accepted:87645398
Current 46 Accepted/s 6 Active 2 Idle 44 Dropped 0

Requests Total:241744717
Current 1 Req/s 47

Server zones
Total 3 / Problems 0
Traffic
In: 346 B/s
Out: 5.11 KiB/s

Upstreams
Total 4 / Alerts 2
Servers
All: 7 / Up: 5
Failed: 2

TCP/UDP Zones
Conn total: 338228
Conn current: 0
Conn/s: 0
Traffic
In: 99.0 B/s
Out: 188 B/s

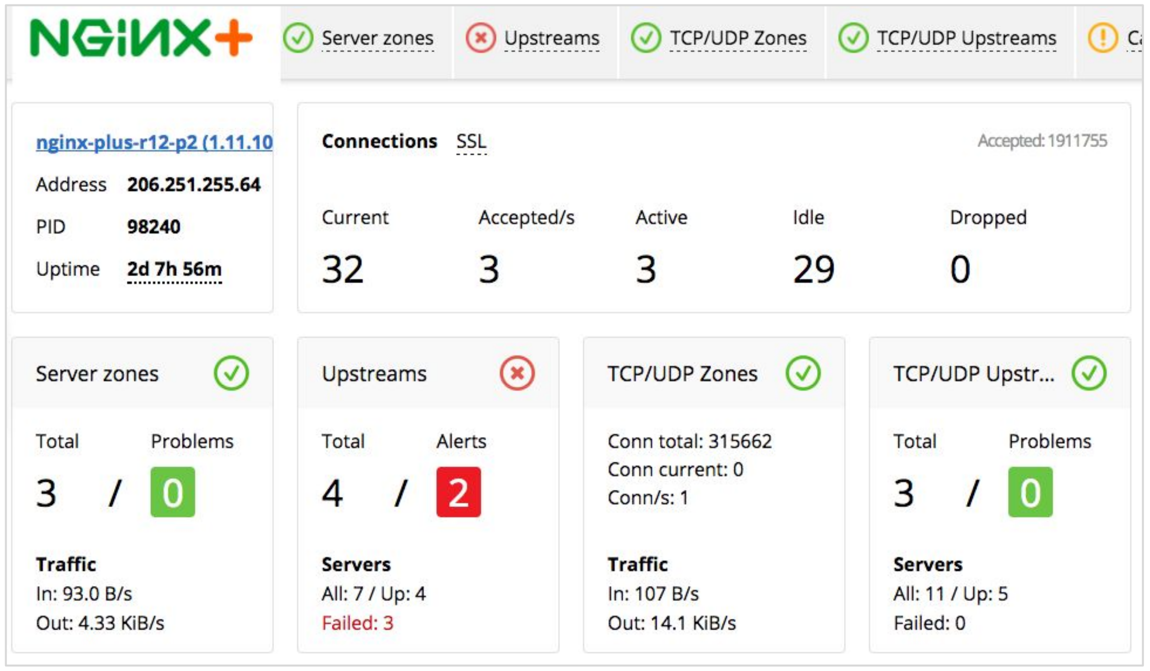
TCP/UDP Upstreams
Total 3 / Problems 0
Servers
All: 10 / Up: 5
Failed: 0

Caches
Total 1 / Warnings 1
Caches states
Warm: 1
Cold: 0

```
upstream my_upstream {
    #...
    zone my_upstream 64k;
}

server {
    #...
    status_zone my_virtual_server;
    }
```

- Provides detailed NGINX Plus statistics
- Over 100+ additional metrics
- Monitoring GUI also available; see **demo.nginx.com**
- Exclusive to NGINX Plus

NGINX Plus Dashboard



nginx-plus-r12-p2 (1.11.10
Address 206.251.255.64
PID 98240
Uptime 2d 7h 56m

Connections SSL Accepted: 1911755

Current	Accepted/s	Active	Idle	Dropped
32	3	3	29	0

Server zones
Total Problems
3 / 0
Traffic
In: 93.0 B/s
Out: 4.33 KiB/s

Upstreams
Total Alerts
4 / 2
Servers
All: 7 / Up: 4
Failed: 3

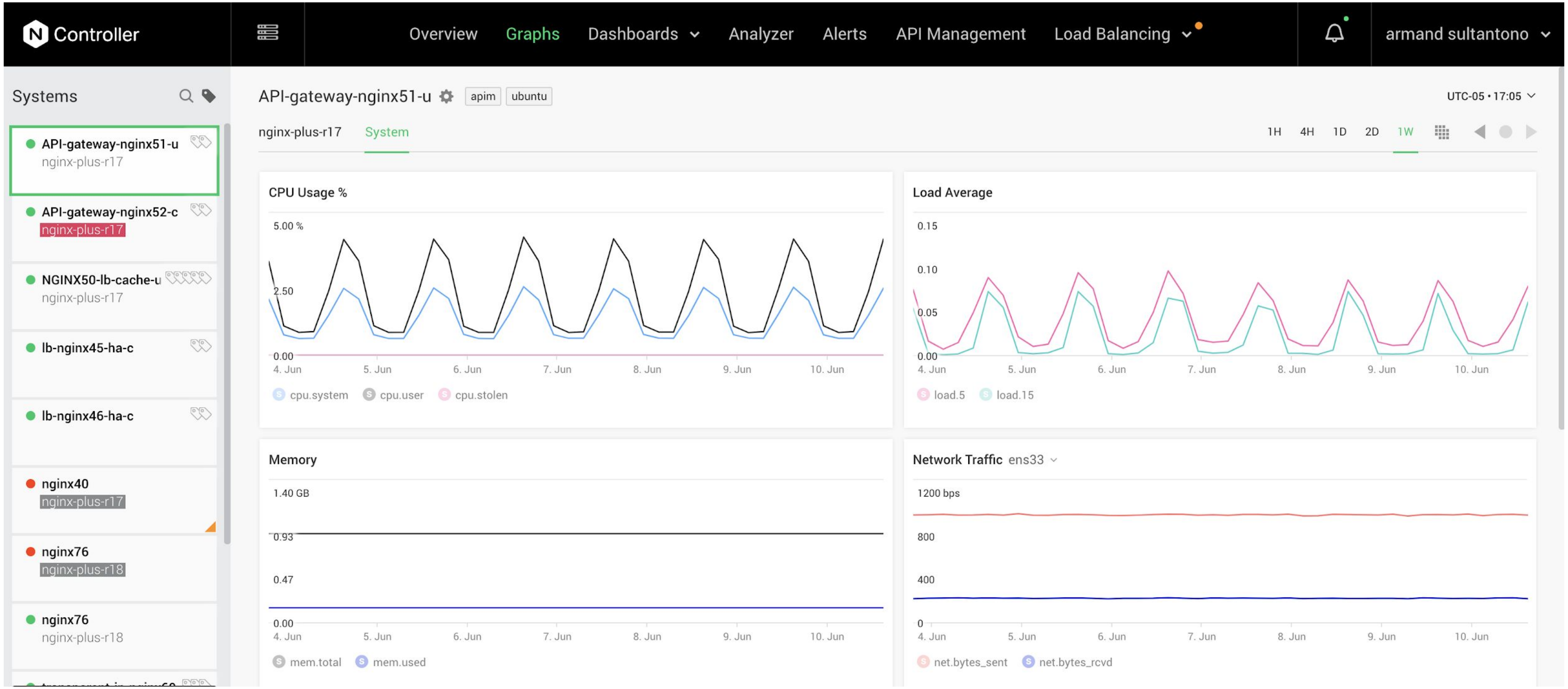
TCP/UDP Zones
Conn total: 315662
Conn current: 0
Conn/s: 1
Traffic
In: 107 B/s
Out: 14.1 KiB/s

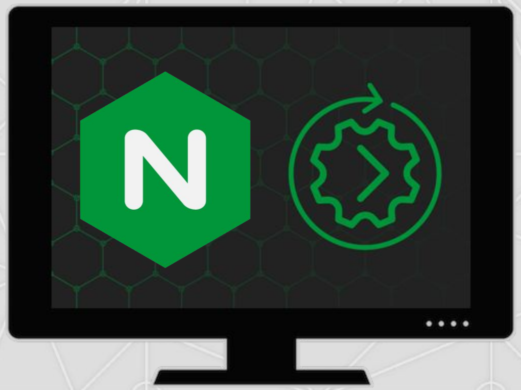
TCP/UDP Upstr...
Total Problems
3 / 0
Servers
All: 11 / Up: 5
Failed: 0

```
"nginx_build": "nginx-plus-r12-p2",
"nginx_version": "1.11.10",
"pid": 98240,
"ppid": 50622,
"processes": {
    "respawned": 0
},
"requests": {
    "current": 1,
    "total": 9915307
},
"server_zones": {
    "hg.nginx.org": {
        "discarded": 9150,
        "processing": 0,
        "received": 146131844,
        "requests": 597471,
        "responses": {
            "1xx": 0,
            "2xx": 561986,
            "3xx": 12839,
            "4xx": 7081,
            "5xx": 6415,
            "total": 588321
        },
        "sent": 14036626711
    },
```

- Over 100 metrics additional real time metrics
- Per virtual server and per backend server statistics
- JSON output to export to your favorite monitoring tool
- See **demo.nginx.com** for live demo

NGINX Controller





Summary

Summary

- It is recommended to use the NGINX mainline branch for most deployments
- All configuration should go into separate files in /etc/nginx/conf.d/*.conf
- Forcing all traffic to SSL improves security and improves search rankings
- Keepalive connections improve performance by reusing TCP connections
- SSL session caching and HTTP/2 improve SSL performance
- NGINX status module and logging capability provide visibility
- NGINX Plus is recommended for all production, load balancing, API gateway deployments
- NGINX Controller enables you to manage the entire lifecycle of NGINX from monitoring to configuration from single pane of glass

Try NGINX Plus for free at **nginx.com/free-trial-request**

NGINX Conf 2019

Sep. 9–12, 2019 | Seattle, WA

The official event for all things NGINX

REGISTER NOW:

www.nginx.com/nginxconf/2019

Q & A

Try NGINX Plus free for 30 days: nginx.com/free-trial-request