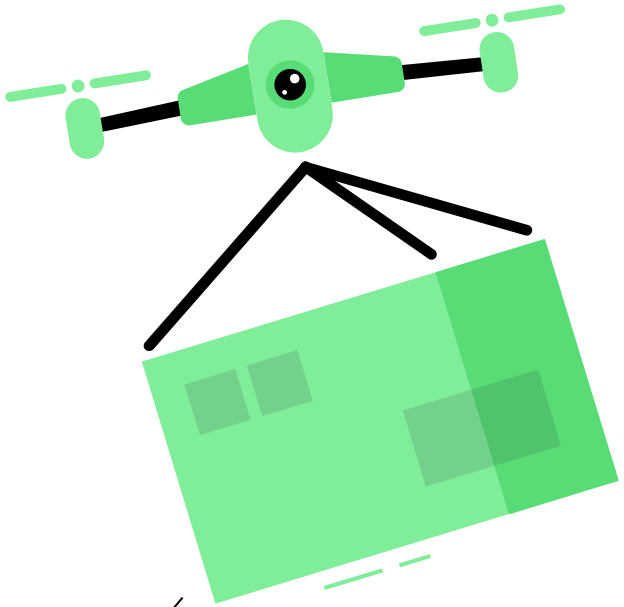
NGINX

Web Server

Content

1. Nginx Introduction
2. Nginx Components
3. Nginx Use Cases
4. Nginx Demo

Nginx Introduction



What is Nginx ?

- Nginx, pronounced like “engine-ex”, is an open-source web server that, since its initial success as a web server, is now also used as a reverse proxy, HTTP cache, and load balancer.

- Some high-profile companies using Nginx include Autodesk, Atlassian, Intuit, T-Mobile, GitLab, DuckDuckGo, Microsoft, IBM, Google, Adobe, Salesforce, VMWare, Xerox, LinkedIn, Cisco, Facebook, Target, Citrix Systems, Twitter, Apple, Intel, and many more (source).

History

- Igor Sysoev originally wrote NGINX to solve problem of the difficulty that existing web servers experienced in handling large numbers (the 10K) of concurrent connections. With its event-driven, asynchronous architecture.

- NGINX revolutionized how servers operate in high-performance contexts and became the fastest web server available.

History - Cont.

- After open sourcing the project in 2004 and watching its use grow exponentially, Sysoev co-founded NGINX, Inc. to support continued development of NGINX and to market NGINX Plus as a commercial product with additional features designed for enterprise customers.
- Today, NGINX and NGINX Plus can handle hundreds of thousands of concurrent connections, and power more of the million busiest sites on the Web than any other server.

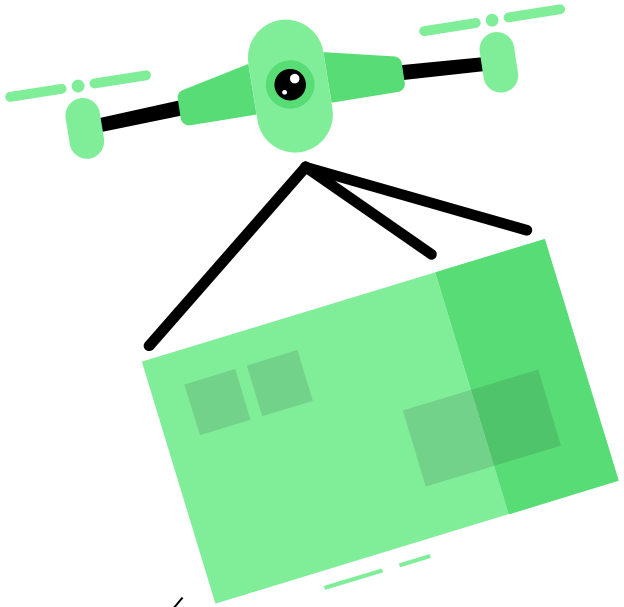
NGINX as a Web Server

- The goal behind NGINX was to create the fastest web server around, and maintaining that excellence is still a central goal of the project.

- NGINX consistently beats Apache and other servers in benchmarks measuring web server performance. Since the original release of NGINX.

- NGINX has grown along with it and now supports all the components of the modern Web, including WebSocket, HTTP/2, gRPC, and streaming of multiple video formats (HDS, HLS, RTMP, and others).

Nginx Components



Installing Nginx

- **Debian and Ubuntu**
  sudo apt-get update
  sudo apt-get install nginx

- **RedHat / CentOS**
  sudo yum install epel-release
  sudo yum update
  sudo yum install nginx

Main configuration file

- The Main configuration file /etc/nginx/nginx.conf
- You can check the syntax of it before restarting the server by running: nginx -t
- You can get more info about nginx by : nginx -h

Log files

- The log files are located in /var/log/nginx

- Under this directory we will find two files

access.log & error.log

Nginx configuration context

1. The Main Context
2. The Event Context
3. The HTTP Context
4. The Server Context
5. The Location Context
6. The Upstream Context

The Main Context

- The main context is placed at the beginning of the core Nginx configuration file. The directives for this context cannot be inherited in any other context and therefore can't be overridden.

- The main context is used to configure details that affect the entire application on a basic level. Some common details that are configured in the main context are the user and group to run the worker processes and the file to save the main process ID.

```
# The main context is here,
outside any other contexts

. . .

context {

    . . .

}
```

The Event Context

- The events context sets global options for connection processing. The events context is contained within the main context. There can be only one event context defined within Nginx configuration.

- Nginx uses an event-based connection processing model, so the directive defined within this context determines how worker processes should handle connections.

```
# main context

events {

    # events context
    . . .

}
```

The HTTP Context

- When configuring Nginx as a web server or reverse proxy, the “http” context will hold the majority of the configuration. This context will contain all of the directives and other contexts necessary to define how the program will handle HTTP or HTTPS connections.

- The http context is a sibling of the events context, so they should be listed side-by-side, rather than nested.

```
# main context

events {
    # events context

    . . .

}

http {
    # http context

    . . .

}
```

The Server Context

The “server” context is declared within the “http” context. This is our first example of nested, bracketed contexts. It is also the first context that allows for multiple declarations.

There can be multiple server contexts inside the HTTP context. The directives inside the server context handle the processing of requests for resources associated with a particular domain or IP address.

```
# main context

http {

    # http context

    server {

        # first server context

    }

    server {

        # second server context

    }

}
```

The Location Context

Location contexts define directives to handle the request of the client. When any request for resource arrives at Nginx, it will try to match the URI (Uniform Resource Identifier) to one of the locations and handle it accordingly.

```
# main context

server {

    # server context

    location /match/criteria {

        # first location context

    }
```

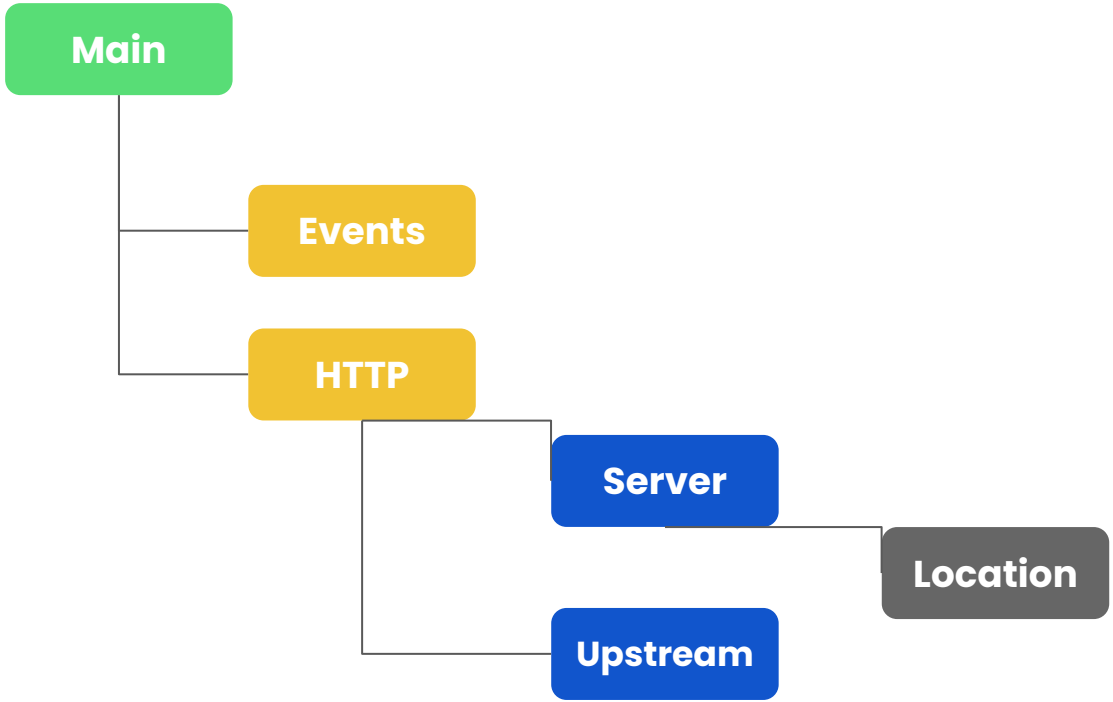
The Upstream Context

- The upstream context is used to configure and define an upstream server. This context is allowed to define a pool of backend servers that Nginx can proxy the used when request. This context is usually we are configuring proxies of various types.

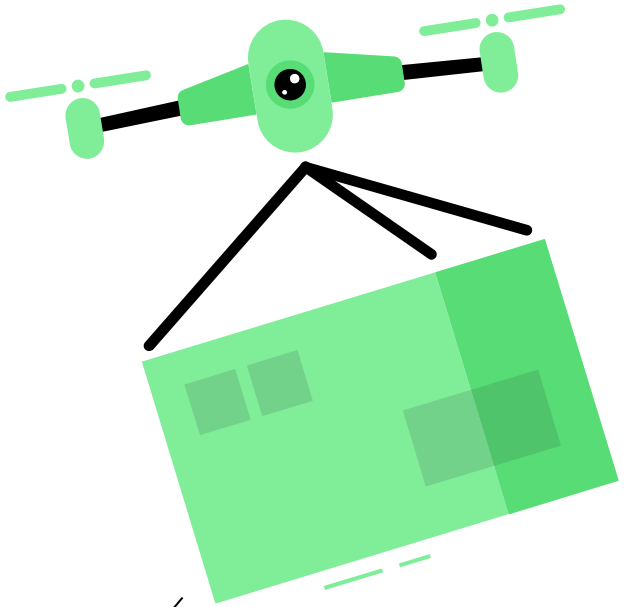
- Upstream context enables Nginx to perform load balancing while proxying the request. This context is defined inside the HTTP context and outside any server context.

```
# main context

http {

    # http context

    upstream upstream_name {
        server proxy_server1;
        server proxy_server2;
    }

    server {

        # server context
    }

}
```

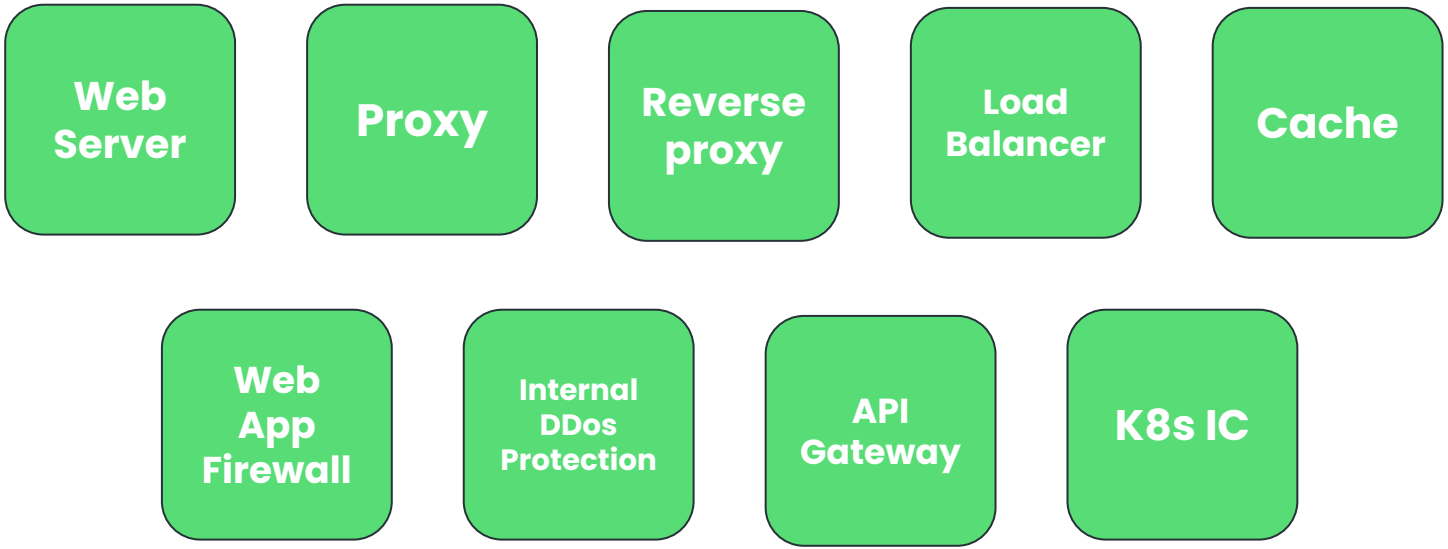
Configuration Contexts



Nginx Use Cases

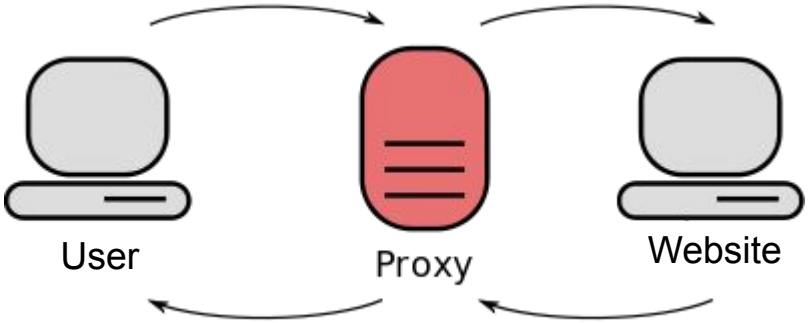


Nginx Use cases



Forward Proxy

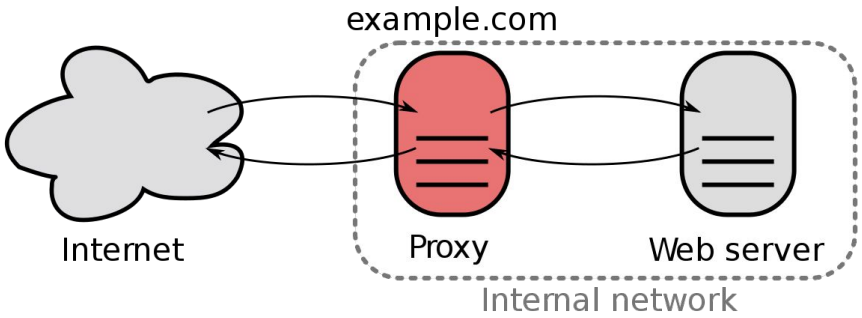
A forward proxy is what most people call ‘a proxy’. You send a connection request to it, and the forward proxy retrieves data from the internet. It usually lets clients on an otherwise firewall-restricted network to access the internet.



Reverse Proxy

- A reverse proxy server is an intermediate connection point positioned at a network's edge. It receives initial HTTP connection requests.

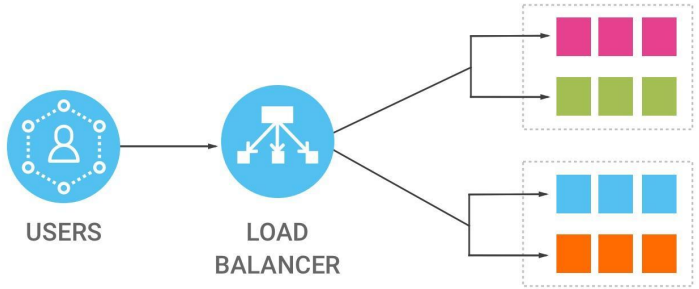
- The reverse proxy serves as a gateway between users and your application origin server. In so doing it handles all policy management and traffic routing.



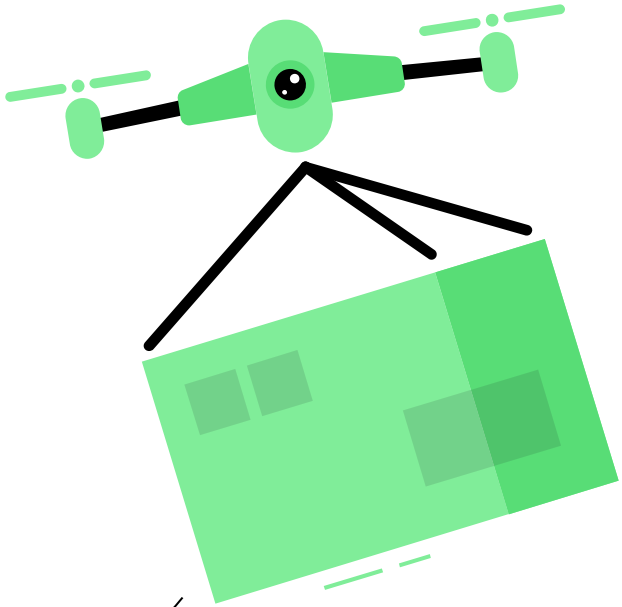
Nginx as LoadBalancer

- Nginx can be configured as a simple yet powerful load balancer to improve your servers resource availability and efficiency.

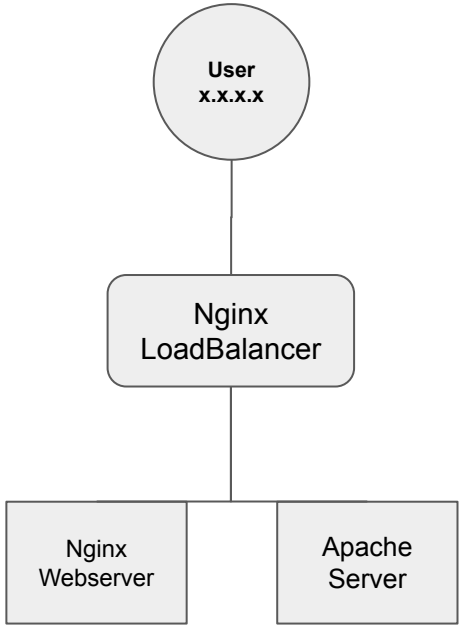
- Nginx acts as a single entry point to a distributed web application working on multiple separate servers.



Nginx Demo



Nginx Demo


User
x.x.x.x
Nginx
LoadBalancer
Nginx
Webserver
Apache
Server

Nginx Lab

- Install Nginx
- Change the default file (index.html) to (yourName.html)
- Make two html files, and change the configuration file to access the first file on port 81, and access the second file on port 82.
- Make a load balancer between 2 servers (like the lecture)
- Tell me the main differences between Apache & Nginx

Thanks!

Do you have any questions?