How to limit incoming traffic with nginx

Managing incoming traffic is one of the essential features to ensure the security and performance of a web server. nginx, with its efficient and flexible architecture, offers various methods to limit and control traffic. In this article, we will explore how to use nginx to limit incoming traffic, thereby improving the security and performance of your server.

nginx is a very popular web server and reverse proxy, known for its ability to handle a large number of simultaneous connections with low resource usage. In addition to its basic features, nginx offers advanced traffic control tools, which can be used to prevent DDoS attacks, reduce server load, and improve user experience.

One of the main methods to limit traffic is to control the connection speed. This can be done using the nginx `ngx_http_limit_conn_module` module.

Add the following configuration in the `http` block of the `nginx.conf` file:

```
http {
  limit_conn_zone $binary_remote_addr zone=addr:10m;
  server {
    location / {
      limit_conn addr 10;
      # Other directives
    }
  }
}
```

In this configuration:

- `limit_conn_zone $binary_remote_addr zone=addr:10m;` defines a 10 MB shared zone to track connections.
- `limit_conn addr 10;` limits the number of simultaneous connections per IP address to 10.

In addition to limiting the number of connections, you can also control the amount of data each client can download in a given period of time. This is possible using the `ngx_http_limit_req_module` module.

Add the following configuration in the `http` block of the `nginx.conf` file:

```
http {
  limit_req_zone $binary_remote_addr
zone=req_limit:10m rate=1r/s;
  server {
    location / {
      limit_req zone=req_limit burst=5 nodelay;
      # Other directives
    }
  }
}
```

In this configuration:

- `limit_req_zone $binary_remote_addr zone=req_limit:10m rate=1r/s;` creates a 10 MB zone to track requests and sets a limit of 1 request per second per IP address.
- `limit_req zone=req_limit burst=5 nodelay;` allows a "burst" of 5 requests over the configured limit without delay.

To block suspicious or known malicious IP addresses, you can use the `ngx_http_access_module` module.

Add the following configuration in the `server` or `location` block of your site's configuration file:

```
server {
  location / {
    deny 192.168.1.1;
    allow all;
    # Other directives
  }
}
```

In this configuration:

- `deny 192.168.1.1;` blocks the specified IP address.
- `allow all;` allows all other IP addresses.

If you are running an API, limiting the rate of requests can prevent resource abuse. Configure rate limiting in your server's configuration file:

```
http {
  limit_req_zone $binary_remote_addr
zone=api_limit:10m rate=10r/m;
  server {
    location /api/ {
      limit_req zone=api_limit burst=20 nodelay;
      # Other directives
    }
```

```
    }
}
```

In this configuration:

- `limit_req_zone $binary_remote_addr zone=api_limit:10m rate=10r/m;` sets a limit of 10 requests per minute per IP address.
- `limit_req zone=api_limit burst=20 nodelay;` allows a "burst" of 20 requests over the configured limit without delay.

Conclusion

Limiting incoming traffic with nginx is an effective method to improve the security and performance of your server. Using the built-in nginx modules, you can easily configure connection limits, bandwidth limits, and malicious IP blocking. Remember to monitor traffic regularly and adjust configurations based on the specific needs of your environment.