# **Kubernetes NGINX Ingress: 10 Useful Configuration Options**

An ingress is a Kubernetes object that provides routing rules that are used for managing external access to the services in a cluster. Ingress makes it easy to define routing rules, paths, name-based virtual hosting, domains or subdomains, and tons of other functionalities for dynamically accessing your applications. But how do you keep track of all these rules, and what happens when you modify an ingress resource or add a new rule? That’s where an ingress controller comes in.

An ingress controller is an implementation of ingress that is tasked with constantly evaluating all the rules defined in your cluster, managing all redirections, and determining where to direct traffic based on the rules defined in the ingress resource. Ingress controllers do not come installed in a Kubernetes cluster by default, so you have to set one up yourself. There are several [implementations of ingress controllers](https://loft.sh/blog/advanced-guide-to-kubernetes-ingress-controllers/?utm_medium=reader&utm_source=other&utm_campaign=blog_kubernetes-nginx-ingress-10-useful-configuration-options) available today; you can check them out [here](https://kubernetes.io/docs/concepts/services-networking/ingress-controllers/).

In this article, you will learn about NGINX ingress controllers and ten useful configuration options you can add to make your application more dynamic.

[**#**](https://loft.sh/blog/kubernetes-nginx-ingress-10-useful-configuration-options/#evaluating-configuration-options)**Evaluating Configuration Options**

There are several types of configuration options that can be applied on Kubernetes’s ingress resources via annotations or ConfigMaps. This list was compiled by researching frequently used options in production systems, gathering detailed online information on ingress controllers, and considering several issues users face on [Stack Overflow](https://stackoverflow.com/) when setting up and configuring ingress resources and controllers. While this article focuses on only ten configuration options, you can review the other options [here](https://kubernetes.github.io/ingress-nginx/user-guide/nginx-configuration/annotations/).

[**#**](https://loft.sh/blog/kubernetes-nginx-ingress-10-useful-configuration-options/#ingress-controller-configuration-categories)**Ingress Controller Configuration Categories**

The NGINX ingress controller has additional configuration options that can be customized and configured to create a more dynamic application. Basically, this can be done in two ways:

* **Annotations**: this option can be used if you want a specific configuration for a particular ingress rule.
* **ConfigMap**: this option can be used when you need to set global configurations for the NGINX ingress controller.

**Note**: annotations take precedence over a ConfigMap.

[**#**](https://loft.sh/blog/kubernetes-nginx-ingress-10-useful-configuration-options/#useful-configurations)**Useful Configurations**

Now, consider the following configuration options for use in your application.

[**#**](https://loft.sh/blog/kubernetes-nginx-ingress-10-useful-configuration-options/#www-redirects)**WWW Redirects**

There are a variety of scenarios where you may want to redirect from www.domain.com to domain.com or vice versa. To enable this feature, add the following annotation:

nginx.ingress.kubernetes.io/from-to-www-redirect: "true"

The resource would look like this:

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: minimal-ingress

annotations:

nginx.ingress.kubernetes.io/from-to-www-redirect: "true"

spec:

rules:

- host: domain.com

http:

paths:

- backend:

serviceName: service-www-redirect

servicePort: 4000

When using a www-redirect, note the following:

* If a new ingress is created at some point with the host equal to the one above, the annotation will be omitted.
* For HTTPS to HTTPS redirects, it’s mandatory to define the SSL Certificate in the Secret and add the TLS section of the ingress.

Read more about www-redirects [here](https://kubernetes.github.io/ingress-nginx/user-guide/nginx-configuration/annotations/#redirect-fromto-www).

[**#**](https://loft.sh/blog/kubernetes-nginx-ingress-10-useful-configuration-options/#ssl-redirects)**SSL Redirects**

SSL redirects are useful for redirecting traffic from HTTP to HTTPS. If this rule is added in the TLS section of the ingress definition, the NGINX controller will redirect (301) to HTTPS. You can disable this rule by adding the following:

nginx.ingress.kubernetes.io/ssl-redirect: "false"

On the other hand, you can enforce a redirect to HTTPS even when no TLS certificate is available in the case of SSL off-loading. This can be done by using the following:

nginx.ingress.kubernetes.io/force-ssl-redirect: "true"

You can also preserve the trailing slash in the URI with ssl-redirect. Here’s an example:

apiVersion: networking.k8s.io/v1beta1

kind: Ingress

metadata:

name: test-ingress

annotations:

nginx.ingress.kubernetes.io/rewrite-target: /

nginx.ingress.kubernetes.io/force-ssl-redirect: "true"

nginx.ingress.kubernetes.io/ssl-redirect: "true"

nginx.ingress.kubernetes.io/preserve-trailing-slash: "true"

spec:

rules:

- http:

paths:

- path: /testpath

backend:

serviceName: test

servicePort: 80

[**#**](https://loft.sh/blog/kubernetes-nginx-ingress-10-useful-configuration-options/#timeout-settings)**Timeout Settings**

There are also several timeout settings you can configure in the ingress resource using annotations:

* proxy-connect-timeout: this defines the timeout for establishing a connection with a proxied server. The default value is 60 seconds, and the timeout typically cannot exceed 75 seconds. Check [here](https://nginx.org/en/docs/http/ngx_http_proxy_module.html" \l "proxy_connect_timeout) for more information.
* proxy-send-timeout: this will set a timeout for transmitting a request to the proxied server. The timeout is set only between two successive write operations, not for the transmission of the whole request. According to [NGINX ingress documentation](https://nginx.org/en/docs/http/ngx_http_proxy_module.html#proxy_send_timeout), if the proxied server does not receive anything within this time, the connection is closed.

For example, you can add timeout settings to your ingress resource, like the following:

apiVersion: networking.k8s.io/v1beta1

kind: Ingress

metadata:

name: cafe-ingress-with-annotations

annotations:

nginx.org/proxy-connect-timeout: "30s"

nginx.org/proxy-read-timeout: "20s"

spec:

rules:

- host: cafe.example.com

http:

paths:

- path: /tea

backend:

serviceName: tea-svc

servicePort: 80

- path: /coffee

backend:

serviceName: coffee-svc

servicePort: 80

Read more about the various timeout settings available [here](https://docs.nginx.com/nginx-ingress-controller/configuration/ingress-resources/advanced-configuration-with-annotations/).

[**#**](https://loft.sh/blog/kubernetes-nginx-ingress-10-useful-configuration-options/#cors)**CORS**

You can also enable cross-origin resource sharing (CORS) in an ingress rule. This allows you to control the methods, headers, origins of requests, and other elements that are allowed to make requests to your cluster. There are several options that can also be activated when CORS is enabled on the ingress resource; for example, the origin of request, the exposed headers, and so forth. To activate CORS on ingress, add the enable-cors annotation on the ingress. Here’s an example:

nginx.ingress.kubernetes.io/enable-cors: "true"

There are other annotations you can use to control the CORS behavior:

nginx.ingress.kubernetes.io/cors-allow-methods # controls method accepted

nginx.ingress.kubernetes.io/cors-allow-headers # controls allowed headers

Here’s an example:

apiVersion: networking.k8s.io/v1beta1

kind: Ingress

metadata:

name: test-ingress

annotations:

nginx.ingress.kubernetes.io/enable-cors: "true"

nginx.ingress.kubernetes.io/cors-allow-methods: "PUT, GET, POST, OPTIONS"

nginx.ingress.kubernetes.io/cors-allow-headers: "X-Forwarded-For, X-app123-XPTO"

nginx.ingress.kubernetes.io/cors-expose-headers: "\*, X-CustomResponseHeader"

nginx.ingress.kubernetes.io/cors-max-age: 600

nginx.ingress.kubernetes.io/cors-allow-credentials: "false"

spec:

rules:

- http:

paths:

- path: /testpath

backend:

serviceName: test

servicePort: 80

You can read more about how to control the CORS functionality [here](https://kubernetes.github.io/ingress-nginx/user-guide/nginx-configuration/annotations/#enable-cors).

[**#**](https://loft.sh/blog/kubernetes-nginx-ingress-10-useful-configuration-options/#rate-limiting)**Rate Limiting**

Rate-limiting annotations are useful for defining limits on connections and transmission rates. They can be very helpful for mitigating DDoS attacks.

* nginx.ingress.kubernetes.io/limit-connections: this defines the number of concurrent connections allowed from an IP address.
* nginx.ingress.kubernetes.io/limit-rps: *rps* stands for “request per second”, and it is used to define the number of connections that may be accepted from an IP per second.

Here’s an example:

apiVersion: networking.k8s.io/v1beta1

kind: Ingress

metadata:

name: test-ingress

annotations:

nginx.ingress.kubernetes.io/limit-rps: "5"

nginx.ingress.kubernetes.io/limit-rpm: "300"

nginx.ingress.kubernetes.io/limit-connections: "10"

spec:

rules:

- http:

paths:

- path: /testpath

backend:

serviceName: test

servicePort: 80

**Note**: when you specify both annotations in a single ingress resource, limit-rps takes precedence. Read more about rate-limiting ingress resources [here](https://kubernetes.github.io/ingress-nginx/user-guide/nginx-configuration/annotations/#rate-limiting).

[**#**](https://loft.sh/blog/kubernetes-nginx-ingress-10-useful-configuration-options/#custom-max-body-size)**Custom Max Body Size**

This rule can be used to set the maximum size of the body in a request. If the body exceeds the maximum size set, NGINX will return a 413 error to the client.

The body size can be configured by using the following:

nginx.ingress.kubernetes.io/proxy-body-size: 8m

Here’s an example:

apiVersion: networking.k8s.io/v1beta1

kind: Ingress

metadata:

name: test-ingress

annotations:

nginx.ingress.kubernetes.io/proxy-body-size: 8m

spec:

rules:

- http:

paths:

- path: /testpath

backend:

serviceName: test

[**#**](https://loft.sh/blog/kubernetes-nginx-ingress-10-useful-configuration-options/#whitelist-source-range)**Whitelist Source Range**

It is also possible to whitelist the allowed IPs from the client. This means you can configure an ingress resource to only allow requests from a particular IP address. This feature can prevent unknown or unauthorized requests from reaching your cluster.

To specify whitelist source range, use the annotation below:

nginx.ingress.kubernetes.io/whitelist-source-range

Here’s an example:

apiVersion: networking.k8s.io/v1beta1

kind: Ingress

metadata:

name: test-ingress

annotations:

ingress.kubernetes.io/whitelist-source-range: "10.0.0.0/24,172.10.0.1"

spec:

rules:

- http:

paths:

- path: /testpath

backend:

serviceName: test

**Note**: you can run into an issue where the whitelisted IP can’t access the resource. In this case, you may need to enable externalTrafficPolicy in your service definition. Review [this answer on Stack Overflow](https://stackoverflow.com/questions/47990836/kubernetes-whitelist-source-range-blocks-instead-of-whitelist-ip) for more information.

[**#**](https://loft.sh/blog/kubernetes-nginx-ingress-10-useful-configuration-options/#default-backend)**Default Backend**

The default backend is used for handling an unknown request or a request that is not mapped to any path or host in the ingress resource. If you don’t define this service and enter a path not mapped, it returns an HTTP 404 (page not found) error. To fix this issue, create a service and map it to the default backend. This can be used to show custom 404 pages and error messages.

To configure this rule, add an annotation:

nginx.ingress.kubernetes.io/default-backend: <svc name>

Here’s an example:

apiVersion: networking.k8s.io/v1beta1

kind: Ingress

metadata:

name: test-ingress

annotations:

nginx.ingress.kubernetes.io/default-backend: <svc name>

spec:

rules:

- http:

paths:

- path: /testpath

backend:

serviceName: test

[**#**](https://loft.sh/blog/kubernetes-nginx-ingress-10-useful-configuration-options/#enable-access-log)**Enable Access Log**

Access logs allow you to view information about client requests. NGINX writes the logs in a file once the request has been processed. They are enabled by default in NGINX, but they may need to be disabled for a given ingress. To do this, use this annotation:

nginx.ingress.kubernetes.io/enable-access-log: "false"

Here’s an example:

apiVersion: networking.k8s.io/v1beta1

kind: Ingress

metadata:

name: test-ingress

annotations:

nginx.ingress.kubernetes.io/enable-access-log: "false"

spec:

rules:

- http:

paths:

- path: /testpath

backend:

serviceName: test

[**#**](https://loft.sh/blog/kubernetes-nginx-ingress-10-useful-configuration-options/#backend-protocol)**Backend Protocol**

You can use the backend protocol to specify how NGINX should communicate with the backend service. Valid values include HTTP, HTTPS, GRPC, GRPCS, AJP, and FCGI. By default, NGINX uses HTTP.

Here’s an example:

nginx.ingress.kubernetes.io/backend-protocol: "HTTPS"

[**#**](https://loft.sh/blog/kubernetes-nginx-ingress-10-useful-configuration-options/#conclusion)**Conclusion**

The NGINX ingress controller makes it easy to configure the rules and set up a more dynamic application for handling requests and responses from the client. This makes NGINX a great choice for ingress controllers with the available number of configurations and settings that can be applied to your ingress resource. In this article, you have learned what an ingress is, what the role of an ingress controller is, and how you can configure your ingress rules to be more dynamic. Learn more about configuring ingress resources [here](https://kubernetes.github.io/ingress-nginx/user-guide/nginx-configuration/annotations/).

*Photo by*[*Denys Nevozhai*](https://unsplash.com/@dnevozhai?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText)*on [Unsplash](https://unsplash.com/s/photos/traffic?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText)*