Kubernetes Ingress Survey

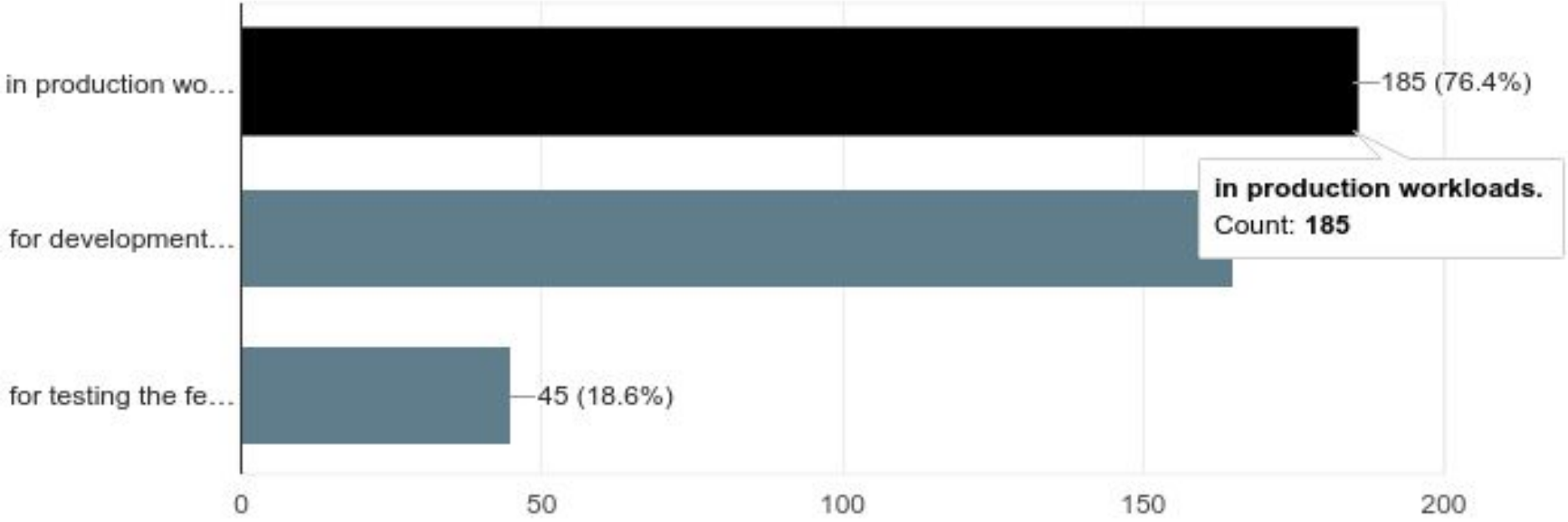
Feb 2018

Data

- N: 248
- Time period: 2 weeks
- Sample: e-mail to sig-network, k8s-dev; twitter.

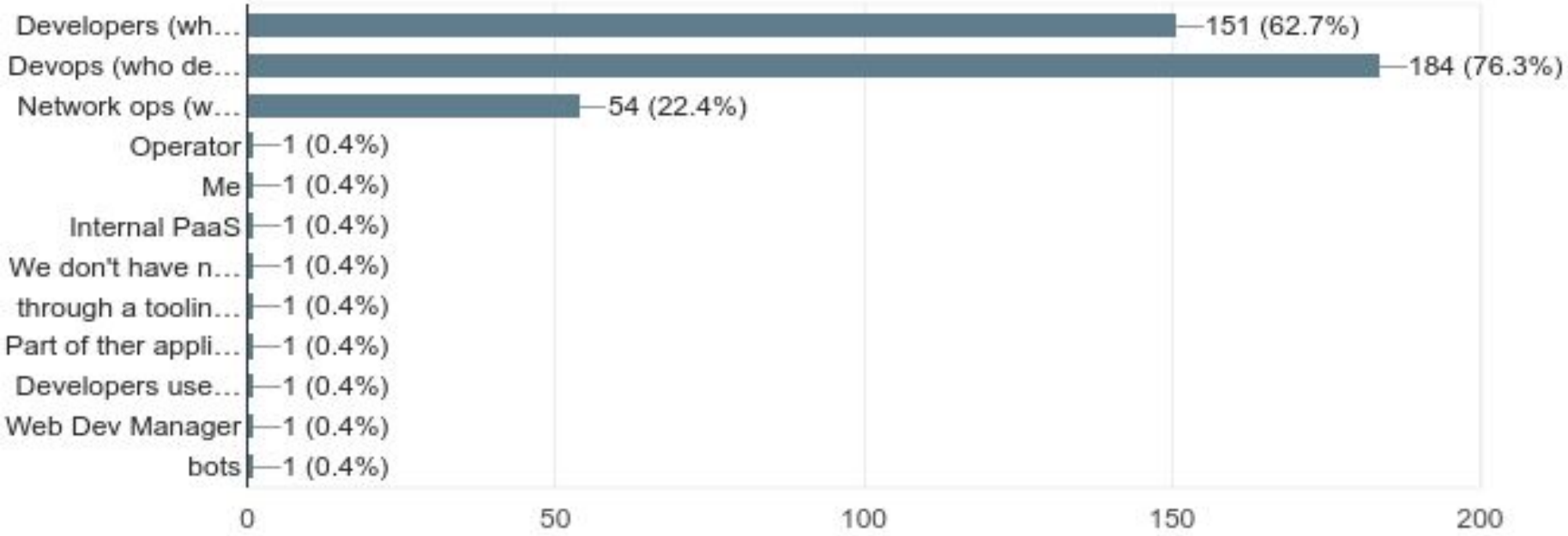
I currently use Ingress (check all that apply)...

242 responses



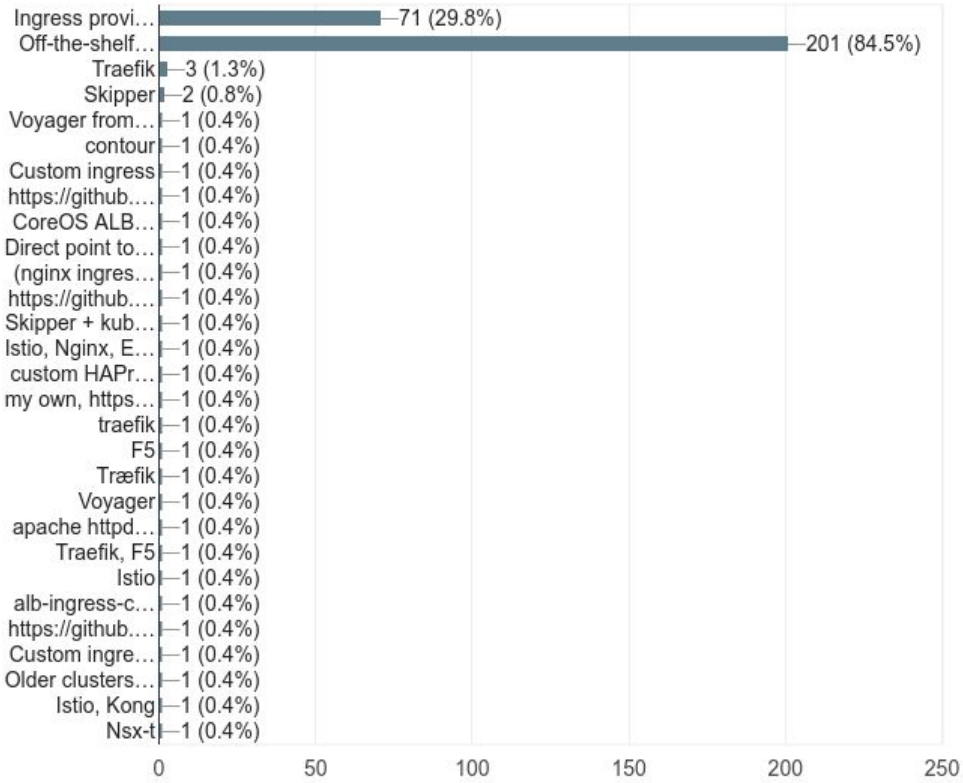
Who authors and manages Ingress definitions in your organization? (Check all that apply.)

241 responses



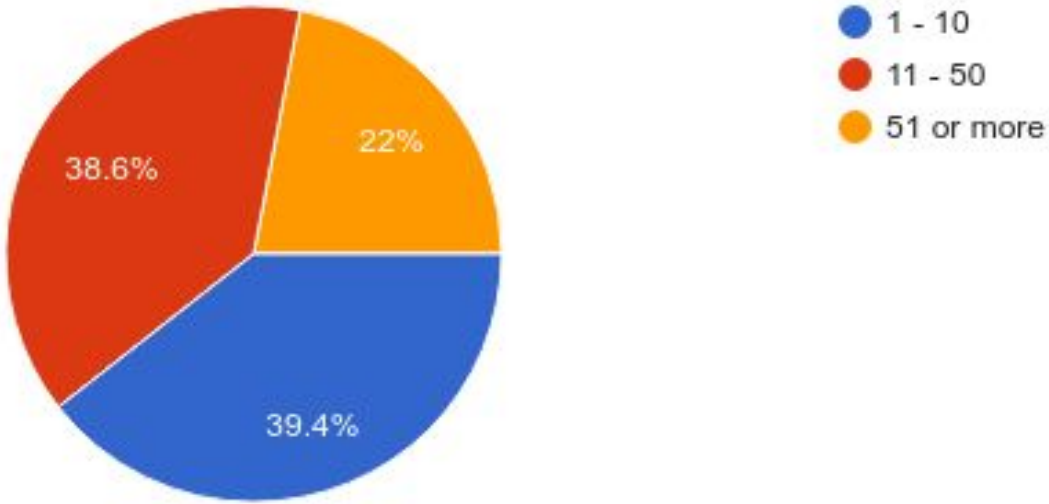
What Ingress providers do you use? (Check all that apply.)

238 responses



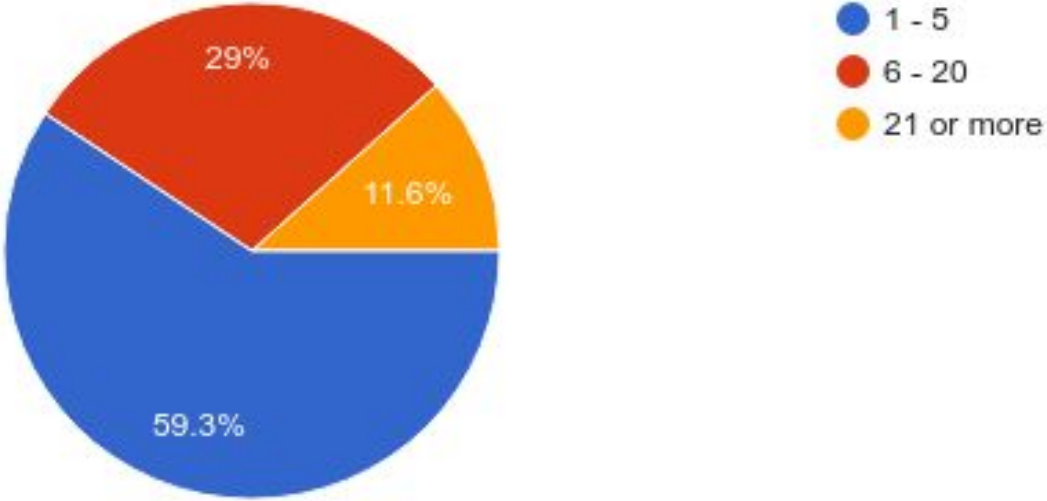
How many Ingress objects per cluster?

241 responses



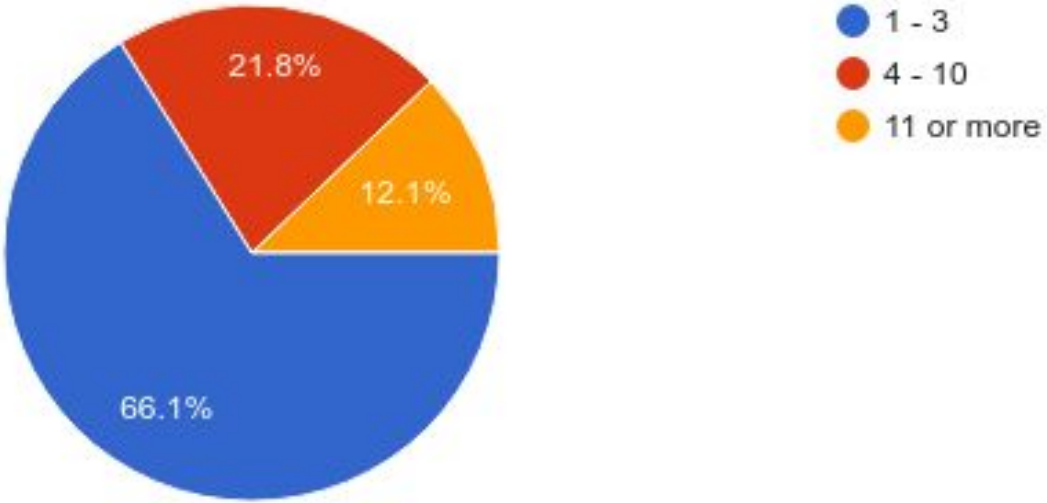
How many paths in the Ingress definition? (Across all `hosts` sections.)

241 responses



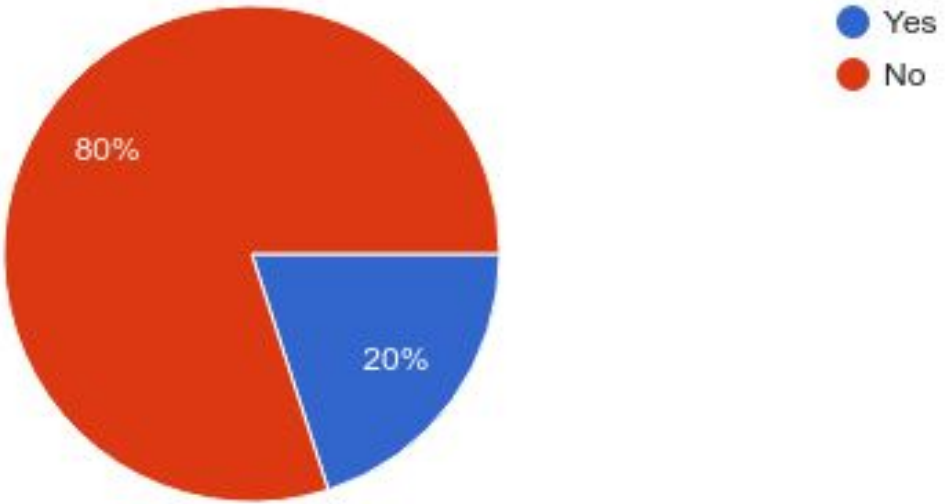
How many different Services referenced by a single Ingress?

239 responses



Do you have different Ingresses pointing to the same service?

240 responses



If yes, what is your use case?

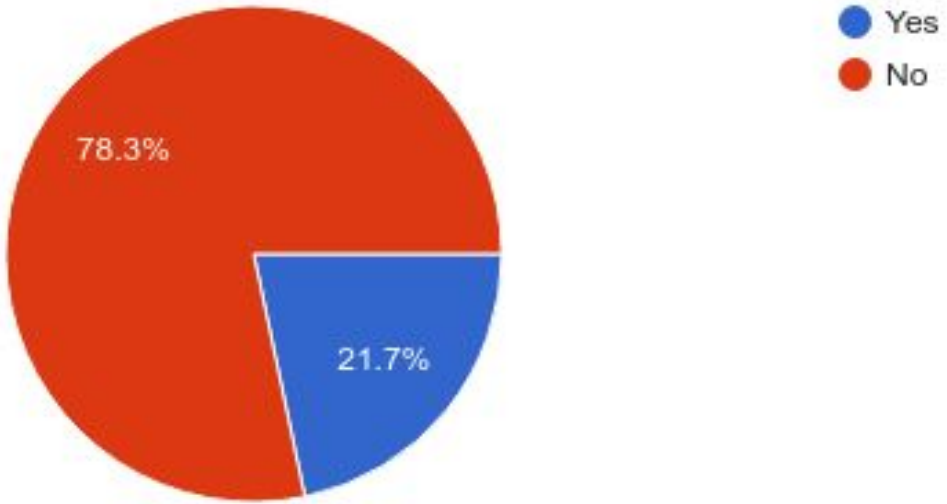
A/B testing, one for setting the cookie and the other to check the cookie
Access control, ext ingress controller requires oauth2 auth
Access while Main Ingress is in maintenance
act as a gateway to the micro services.
API and webfront ui
API versioning
api.site.com and tenant.site.com/api/ pointing to API service
Application running services on multiple ports
Automatic rendering of routes
backwards compatibility with an older url structure
Blue/Green deployments
Canary ab testing
client certificates for all except a specific CIDR range
Complex path and host based routing
cross-namespace ingress
Different host header for clients
for letsencrypt cert validation
Google cloud load balancer for public traffic, nginx for internal VPN traffic
I expect we will have this, as we may have ingress controllers for serving external facing requests, and another one for serving internal only requests. Part of the issue is the path to the ingress controller, and separation might be better for management.
Ingress for external and internal access; One for external with SSL termination and one for internal without ssl
Ingresses that uses Whitelist, and Ingresses that use CA Certificate auth, with different hosts but pointing to the same Service
Internal to VPC is a different Ingress than Ingress from External
kube-lego
Kube-lego
kube-lego
Legacy
legacy :(
Mainly for migration of services to new hostnames.
migrating from a complicated custom ingress to a much simpler version
Multi DNS or certificate
Multiple DNS names
multiple domains, same service, cloudflare warp tunnel
Multiple websites with the same backend

If yes, what is your use case?

Non SSL and testing new ingress configs
Oauth callbacks across k8s namespace by path
public vs. private traffic routing
Regex support in nginx ingress, Allowing some entries to have regex and other to not
Services with multiple ports
Some our clients have special urls, with different SSL certs, that still not the same underlying product.
Tcp forwarding?
To be exposed on different ingress controller
To secure some endpoints on the services, white list IP, password..
With and without SSL, different yaml owners

Do you have different paths within an Ingress pointing to the same service?

189 responses

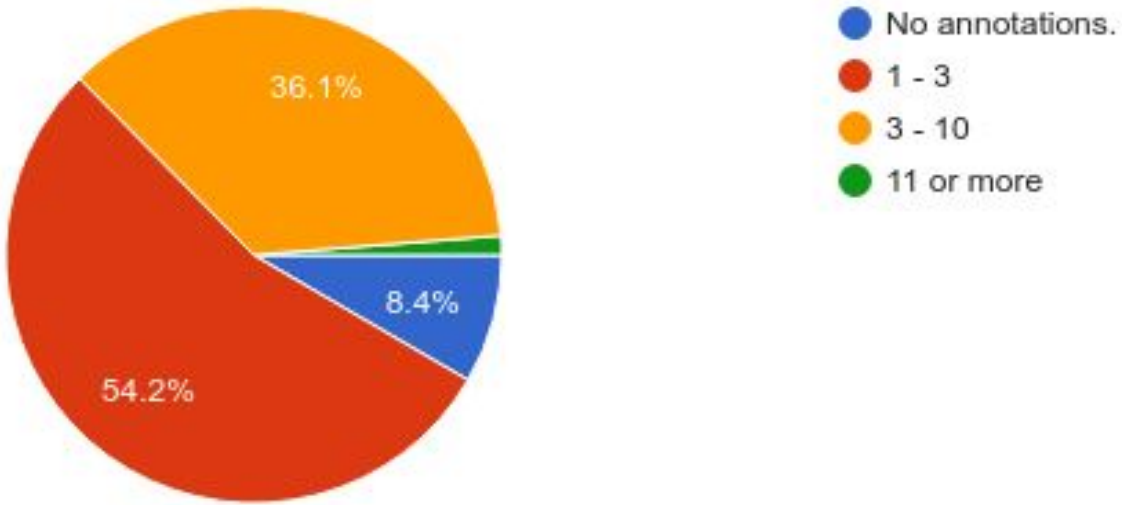


If yes, what is your use case? 30 responses

For example if you point to external name service you can slowly migrate paths to the old service
just generic services, but this question sounds really strange to me. How should it be?
multiple names for the same end point
It is easier to manage multiple narrow use cases than have to carve out exceptions
Different service ports
Different ACLs
Two urls different apps pointing to the same container. Just to bring backward compat.
Wild cards do not work
legacy :(
Multiple domains, like www sub-domain
Mostly because of out legacy system where lots of code bits were reused across domains
Old mytaxi app versions sometimes use different paths for a service (camelcase path vs. lowercase path)
backwards-compatibility
We keep different paths temporarily to keep compatibility working with our customers.
public vs non piblic api
Migrating legacy stuff
To secure some endpoints on the services, white list IP, password..
Legacy application with bad webpath config
Multiple vhost names
Path prefix commodity
versioning
API
Not different paths, per-se, different hosts.
Legacy supporting wrong domains for redirects
DNS with no wildcard
Backward compatibility
Legacy paths being re routed to new service
Service Mock Proxy
Our ingresses reflects our exposed api, one service might provide a number of paths, for instance, some management api calls are separated from the rest of the api paths
Two services were combined into one, and it was easier to do this than updating all the dependent applications

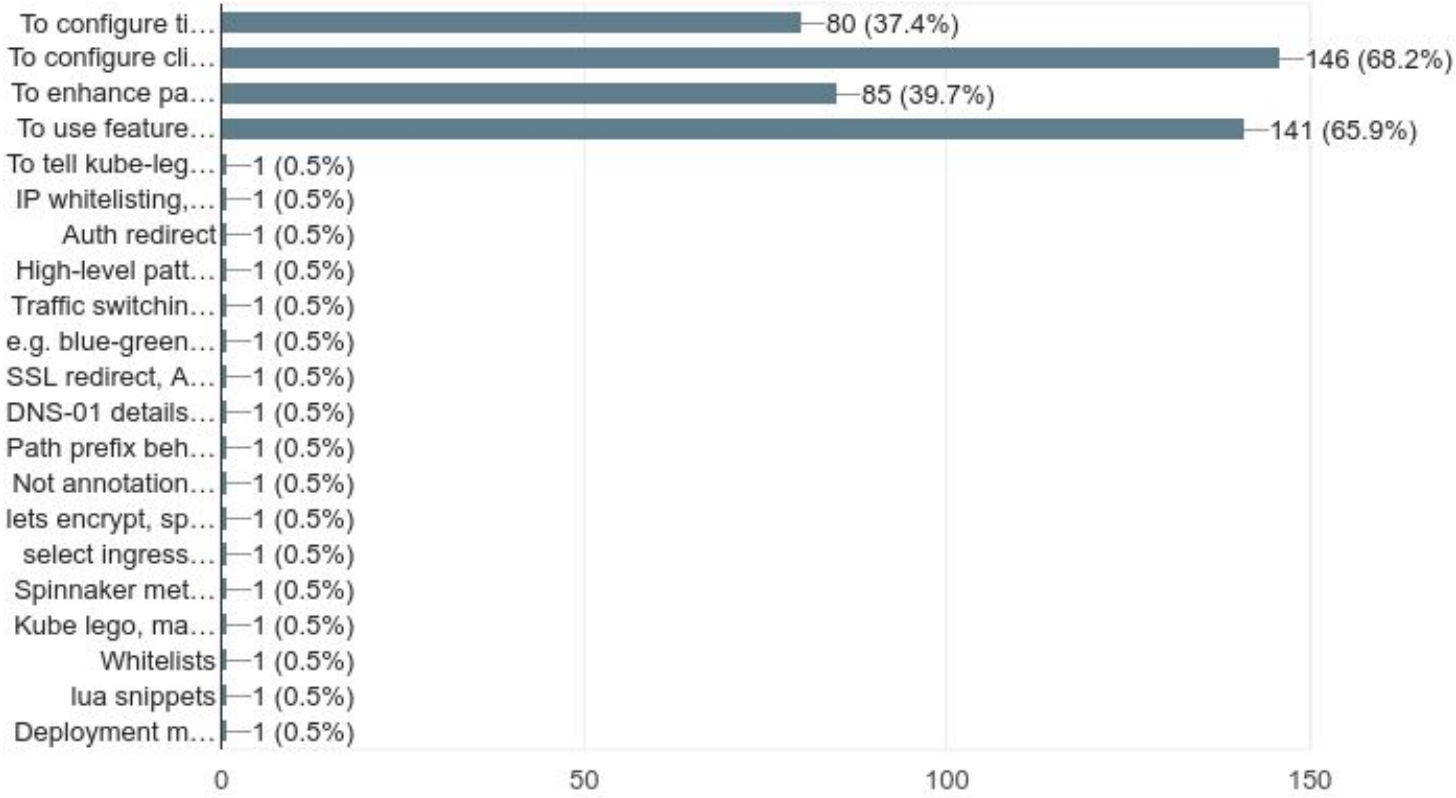
How many annotations in your Ingress object? (Not including status annotations populated by the controller).

238 responses



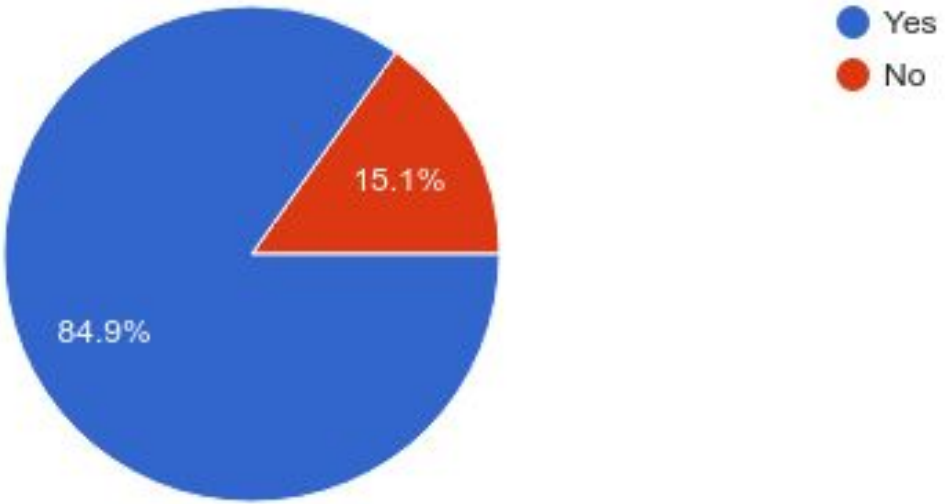
What are your annotations used for? (Check all that apply.)

214 responses



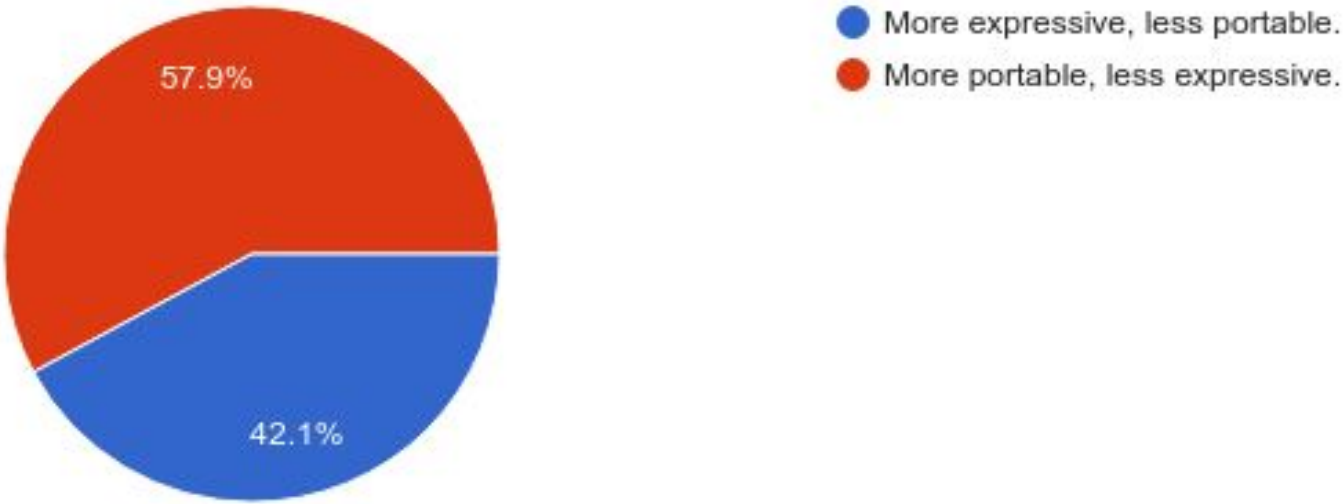
Do you expect the Ingress spec (without annotations) to be transparently portable across environments?

238 responses



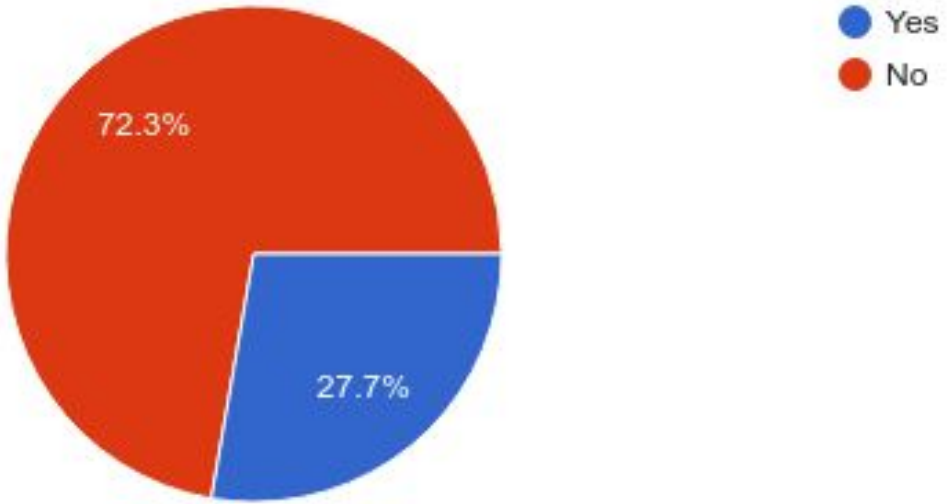
Would you prefer a less portable spec that is more expressive or a less expressive but more portable spec?

190 responses



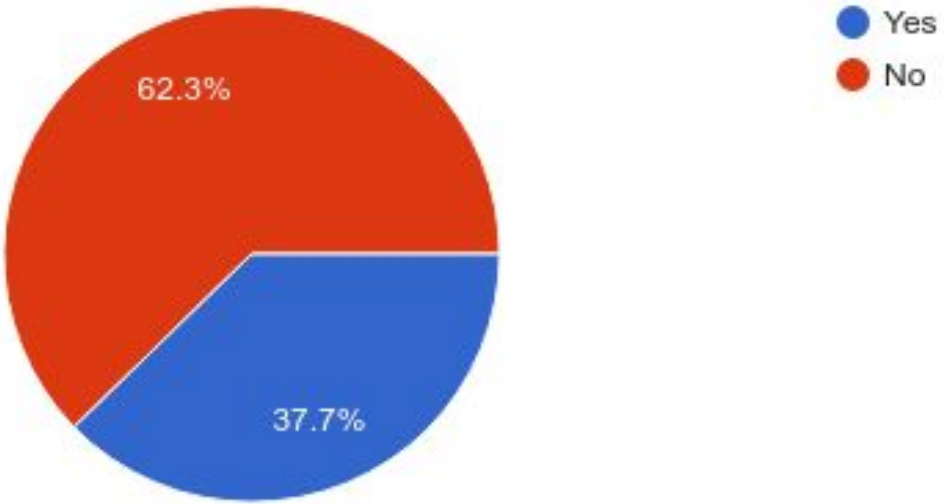
Have you migrated an Ingress configuration between providers?

238 responses



Is the vanilla Ingress (no annotations) expressive enough for your application?

236 responses



If you could add one feature/change one thing about the current Ingress specification, what would that be? (90 responses)

Na
Remove regular expressions from the spec and inline the Secret in the Ingress rules at runtime (informers to know about all the secrets in a namespace)
I'm quite happy with it at the moment - we use the contrib nginx Ingress on AWS with kube-lego to get certs.
Automatic full encryption to the pod including the service.
- support for load balancer protocols per backend; round robbin, session sticky, etc - support for service backends per route.
List of hosts in the same ingress for maybe configuring a Public and an internal hostname without redeclaring twice the path rules
L4 support
Zero-downtime deployment on bare metal
Nesting, or use multiple at ones. Example nginx auth using oauth but still use istio
Standardize on annotation usage both in syntax and semantics across Ingress providers (aka: make annotations truly portable)
Improved header routing options (more than host header)
Handle the case with zero pods/endpoints, redirect to a backup service and scale the deployment from zero to 1 or more on the first packet
Ingress should support bare TCP/UDP/etc protocols, not just HTTP/HTTPS.
Ingress claims. Right now you have one ingress controller to Rule The Cluster (or one per namespace..). Security is hard this way.
It would be super to have traffic weights on the backend definition within ingress.
Traffic weights for backend definitions
since it seems to target http, it should be fully compliant with http, but also should not contain regex specific rules
Less annotations, more standard fields

If you could add one feature/change one thing about the current Ingress specification, what would that be? (90 responses)

Allow ingress controller specific config in `spec` rather than just annotations. For example in GCE controller, you can specify static IP annotation, but only one. We need a v4 and a v6 static IP so have to manually configure that after the LB is created.
Add regexp support in host and paths
use tls secrets from other namespaces
More advanced routing, specifically with regard to SSL redirection and general removal of annotations, which largely seem like a hack.
- Cluster ingress as means of testing and validation of configuration,
- Ingress that are Anycast IP aware
- Provider redundant ingress approach
TCP & UDP Services
It meets all my needs

Visualization of paths per ingress, visualization of ingress per service.
Add SSL passthrough / TCP forwarding as a native option instead of it being an annotation (disregard if this is the case and the nginx-ingress just doesn't support it yet.)
Provide a way to compute the target service dynamically from the route
A single load balancer to serve all ingresses is not sufficient for a large cluster in public cloud.
We need a mechanism to flexibly configure our ingress cloud-provider L7 LB - there are many API configurations that can not expressed by annotations.
I am a public cloud provider and we sells our cloud provider LB attached to Ingress. There are many kinds of Qos level of our LBs. The "ingress-class" annotation can not satisfy our need since it's just a simple string. We need a way to configure all these Qos parameters.

If you could add one feature/change one thing about the current Ingress specification, what would that be? (90 responses)

I am curious to learn if we can standardize all these annotations. If we are using them, why not graduate them. Especailly, I noticed there is a storageClass for PV&PVC which allows users to configure their backend storages, e.g. API parameters, Qos levels etc. Can we do the same for Ingress? We also need a way to configure our Ingress providers. And, a single ingress is not ALWAYS enough for a production cluster.

Expose all underlying provide configuration

Allow which protocol will be used in the upstream. For example to configure envoy with grpc-web.

-

Routing weights

Make ingress more like a complete API gateway solution , with support for Auth, throttling, canary , etc

Dynamic path segment routing with prefix replacement, e.g. it should be easy to configure `/api/<scv>/<resource>` for any set of services (matching `<svc>` segment), and strip URL path prefix leaving just the `/<resource>` segment.

More authentication mechanisms

Better overview with kubectl or dashboard.

regex support in teh nginx ingress.

make ingress class (which controller should handle this?) a first-class attribute

Exponential traffic ramp-up during a deploy.

Better debugging for when the controller does not respond or misdirects a request.

Enrich regex path support.

arbitrary (url)path rewrites

I know it's not what ingresses are for, but defining TCP routing would be very helpful. In TCP mode any specified paths should be ignored, but what I would really want to have is port based routing.

regex/wildcard for domain and path routing

Proxy settings

If you could add one feature/change one thing about the current Ingress specification, what would that be? (90 responses)

After fighting with the annotations, configmaps and other convoluted ways os injecting snippets into an nginx.conf file, I ended up writing a fully custom nginx template. At that point the Ingress resource was basically empty. Its only purpose was now to identity services (not even paths) so that endpoint arrays can be passed to the nginx template Go variables.

In the end I ask myself wether the Ingress spec can even succeed in the face of all the differences between ingress controller options. Perhaps the custom resource approach used on Voyager is the right model after all.

Portability accross clouds can be achieved by simply running the ingress controller inside the cluster -- the cloud-specific portion that remains is a Service[type=LoadBalancer], and that seems reasonable.

Per namespace ingress, that would be great

The ability of setting metadata variables in rules.spec, eg: "-host: myapp.${metadata.namespace}.my.domain"

I think that overall if the architecture of the ingress could change then perhaps it could become portable on prem and off prem in every cloud. Ideally the design would enforce that the ingress provider (nginx etc) would reside on on or more nodes (ideally separate due to cpu or security concerns but i digress) and would expose a node port that would be mapped to a dns provider. This removes most all of the complexities around aws vs gce and has the bonus of working onprem just as well.

More support by default for bare metal deployments. It's too cloud specific

tcp/udp support

List of endpoint pods Like endpoint die service, makes Debugging easier

Host aliases

Detailed metrics

routing rules based on header key/value

If you could add one feature/change one thing about the current Ingress specification, what would that be? (90 responses)

Ingress is a name I have to explain again and again. I do not know a better name. It fits.
TLS mode (HTTPS => HTTP, passthrough, re-encapsultation)
Service mesh integration
Move all annotations into a CRD and provide a way to apply that CRD to Ingress. That way I can consume a 3th party ingress resource as -is and still customize it for my specific environment.
TCP support for ease of Istio integration, consider leveraging Envoy APIs
Mod_rewrite
The ability to control more arbitrary parameters of the underlying glbc, for example; enabling Googles Identity Aware Proxy
Healthchecks
Don't have enough experience and support enough use cases to suggest this yet.
Hook into the load balancer API?
ha
Add chaining for e.g. OAuth proxies, whether they are run by the cloud provider (like GCP's IAP) or in-cluster.
Support more IngressRules (not just HTTP) or be able to force HTTPS inside IngressTLS (or just explicitly disable http?)
Include status from ingress controllers
Tcp ingress rules, sni support
Add TCP proxy support
Secure backend config
Dispatch to different versions of a service using a cookie
I would like some manner of security integrated, like oauth2
Be more opinionated about core features needed in load balancing, and let the third party controllers adapt.

If you could add one feature/change one thing about the current Ingress specification, what would that be? (90 responses)

standardize path expressions. I'd also like to see the community adopt a single really great ingress controller. If that is istio, contour, etc. doesn't matter. Just make it great

ingress should support l4 protocol and routing based on ports
Support tcp.
Make TLS passthrough and TCP/UDP proxying first class parameters.
Being able to set values from ConfigMaps
host regexp matching (multi-tenant web apps with tenants having personalized subdomains)
Multi cluster ingress
Standardisation/clarification around how Ingress Controllers should handle "default backends". i.e. How to handle the case where multiple Ingress Resources exist, each with a different default backend.

Traffic-shifting - add equiv of Istio's RouteRule
nginx ingress - proxy_pass https and ingress pointing to service in another namespace
Regexp hosts (or at least, wildcard prefixed hosts, ie as in https://github.com/kubernetes/kubernetes/issues/41881)