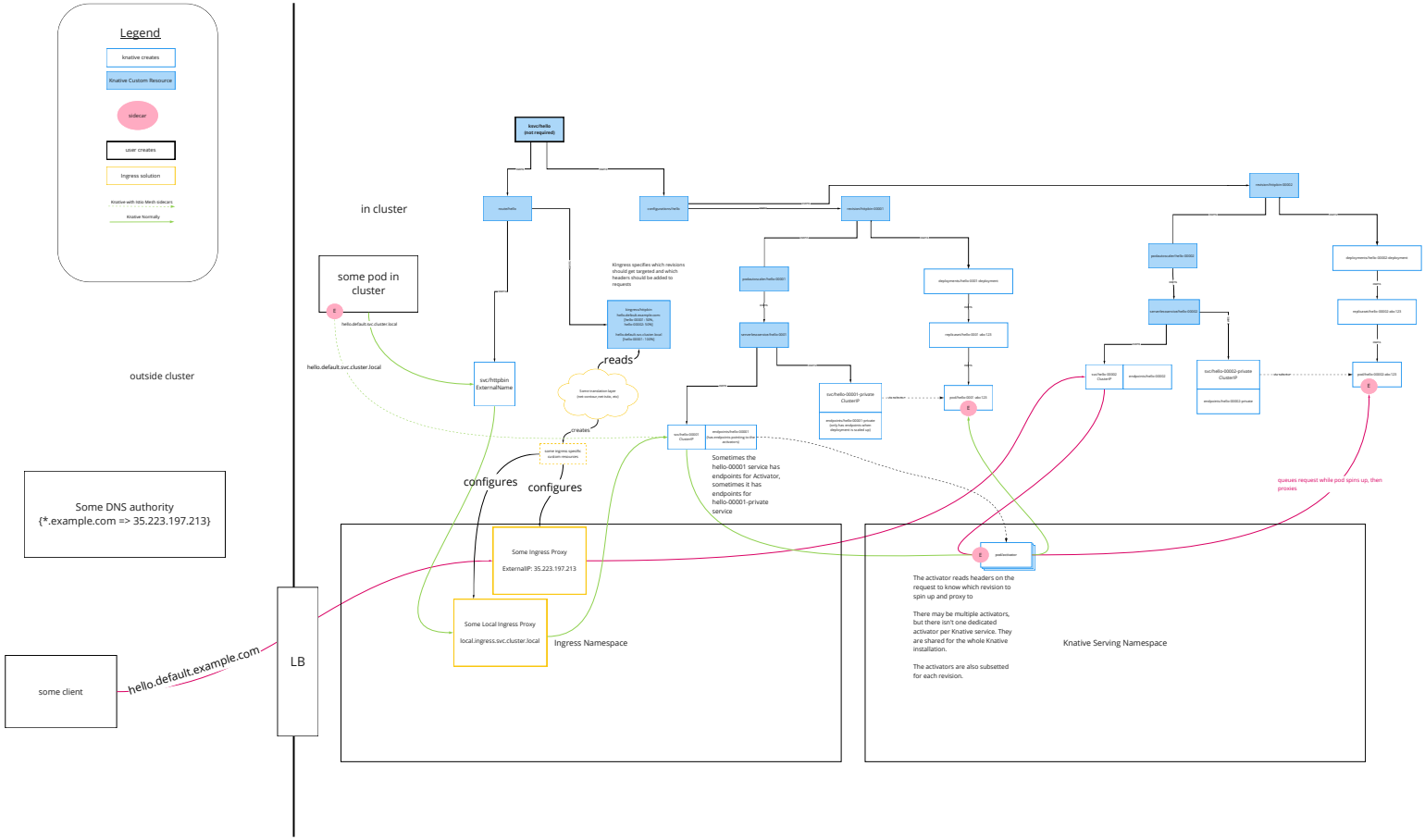
Legend
knative creates
Knative Custom Resource
sidecar
user creates
Ingress solution
Knative with Istio Mesh sidecars
Knative Normally
ksvc/hello (not required)
route/hello
configuration/hello
revision/hbpbin-00001
revision/hbpbin-00002
in cluster
some pod in cluster
hello.default.svc.cluster.local
outside cluster
Kingress specifies which revisions should get targeted and which headers should be added to requests
kingress/hbpbin
hello.default.example.com
hello-00001: 50%
hello-00002: 50%
hello.default.svc.cluster.local:
hello-00001: 100%
podautoscaler/hello-00001
deployments/hello-0001-deployment
serverless-service/hello-00001
replicaset/hello-0001-abc123
pod/hello-0001-abc123
svc/hello-00001-private
ClusterIP
endpoints/hello-00001-private
(only has endpoints when deployment is scaled up)
podautoscaler/hello-00002
deployments/hello-00002-deployment
serverless-service/hello-00002
replicaset/hello-00002-abc123
svc/hello-00002
ClusterIP
endpoints/hello-00002
svc/hello-00002-private
ClusterIP
endpoints/hello-00002-private
pod/hello-00002-abc123
svc/hello-00001
ClusterIP
endpoints/hello-00001
(Sometimes activator points to this activator)
reads
Some translation layer
(net-contour, net-istio, etc)
creates
some ingress-specific custom resources
svc/httpbin
ExternalName
configures
configures
Some Ingress Proxy
ExternalIP: 35.223.197.213
Some Local Ingress Proxy
local.ingress.svc.cluster.local
Ingress Namespace
Sometimes the hello-00001 service has endpoints for Activator, sometimes it has endpoints for hello-00001-private service
queues request while pod spins up, then proxies
podactivator
The activator reads headers on the request to know which revision to spin up and proxy to
There may be multiple activators, but there isn't one dedicated activator per Knative service. They are shared for the whole Knative installation.
The activators are also subsetted for each revision.
Knative Serving Namespace
Some DNS authority
{*.example.com => 35.223.197.213}
some client
hello.default.example.com
LB