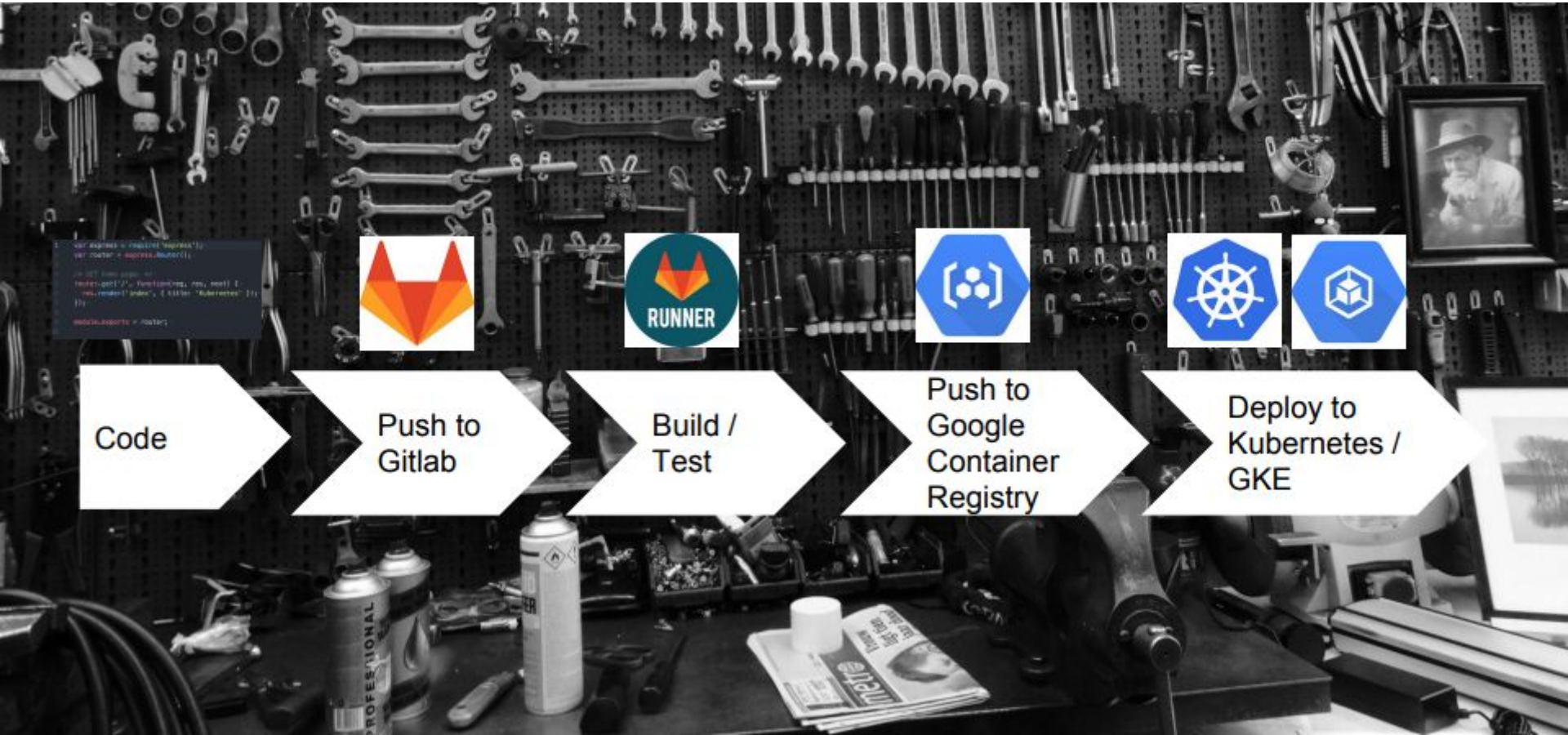
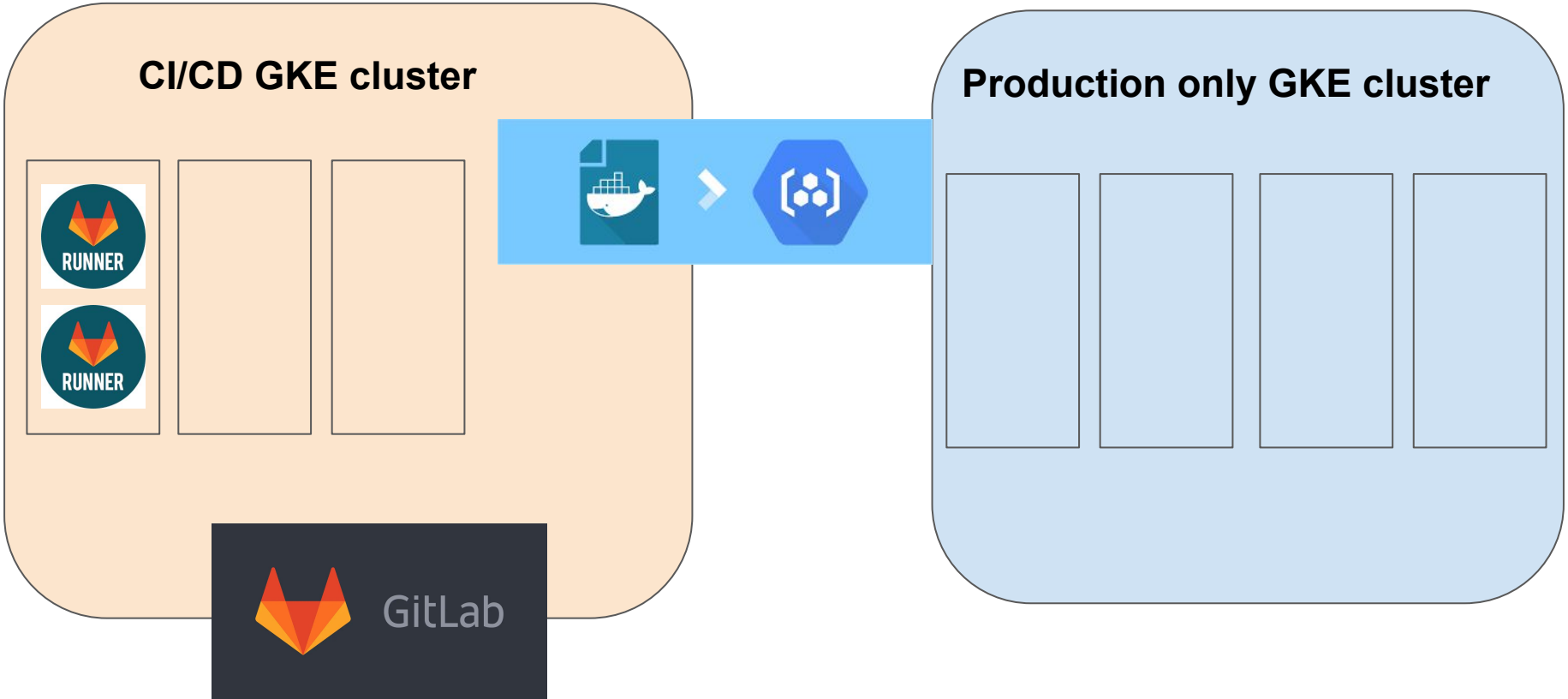
Microservices CI/CD proposal

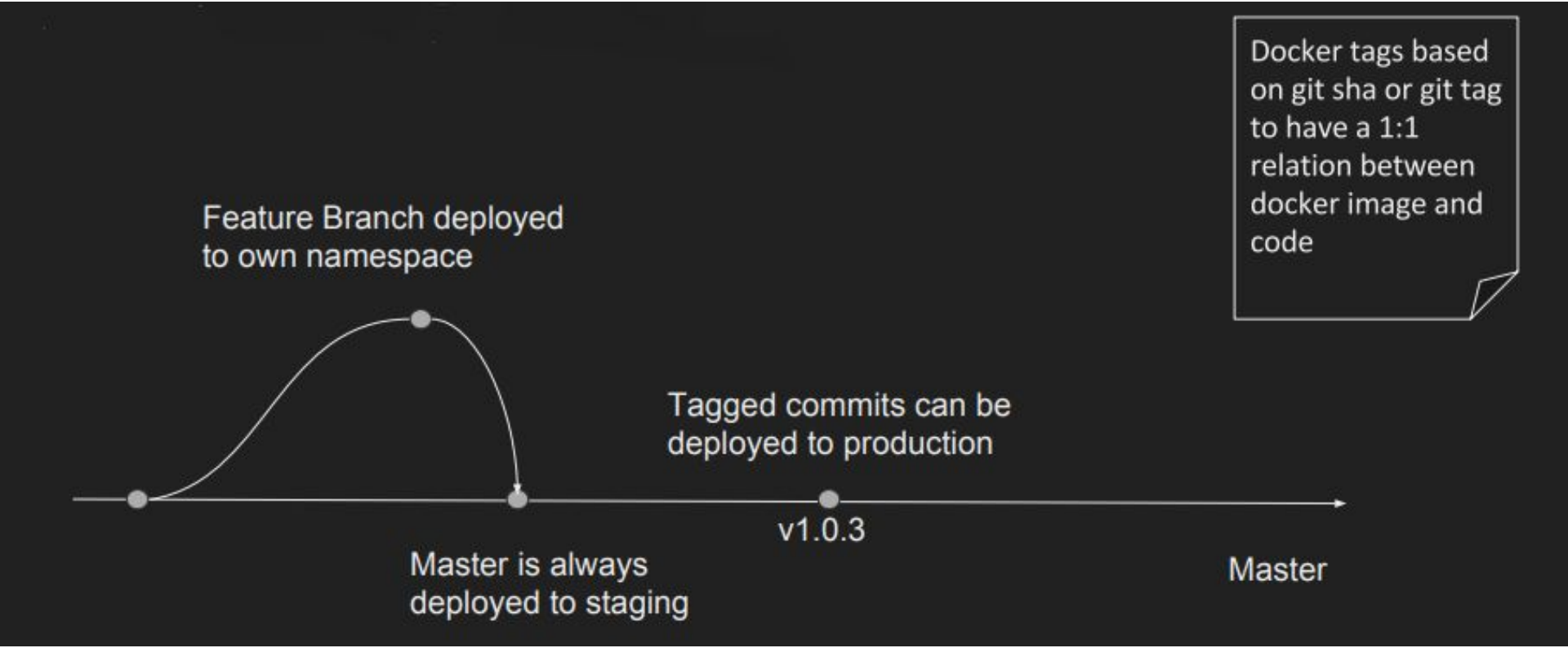
Use the same basic Tools



CI/CD and Production GKE clusters



CI/CD Cluster: Branching/Deployment per project



How to: GitLab AutoDevOps + GKE

Prerequisite:

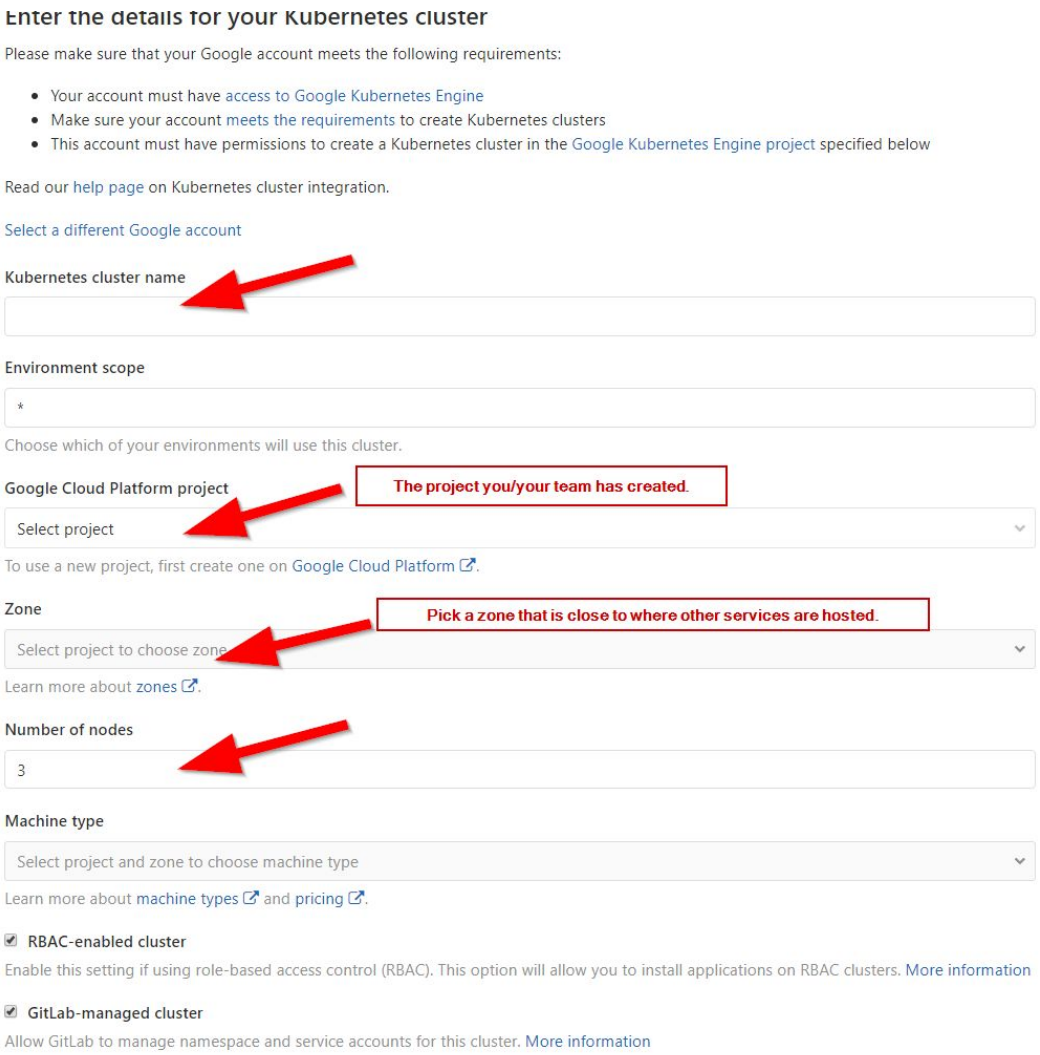
- **Have an account that has access to the GCP your team is going to use.**
- **Have a project in GCP. Create one if you don't have one already, this can be done easily via GCP console.**
- **A Dockerfile which exposes to a specific port (example the port 5000)**

Step 1: configure the cluster

- **Go to your repository, click on Operations > Kubernetes. Log in with your GCP account.**

- **Fill out the form, which contains basic information that is needed to create a cluster (assuming e are creating a new one).**

- **Make sure the GitLab-managed cluster is checked wince we'd like to use AutoDevOps.**



Step 2: configure the cluster

After the cluster has been created, you can configure the cluster. Fill out the base domain, which you/your team owns and has a wildcard DNS configured.

cloudinary-extension

Integration status

Enable or disable GitLab's connection to your Kubernetes cluster.

Environment scope

*

Choose which of your environments will use this cluster.

Base domain

gke.merch. .com

Specifying a domain will allow you to use Auto Review Apps and Auto Deploy stages for Auto DevOps. The domain should have a wildcard DNS configured matching the domain. Alternatively `35.234.70.160.nip.io` can be used instead of a custom domain. More information.

Save changes

Step 2: configure the cluster

Install some applications. Commonly used ones are: Ingress, cert-manager and GitLab Runner.

Applications

Choose which applications to install on your Kubernetes cluster. Helm Tiller is required to install any of the following applications. More information

Helm Tiller
Helm streamlines installing and managing Kubernetes applications. Tiller runs inside of your Kubernetes Cluster, and manages releases of your charts.
Installed

Ingress
Ingress gives you a way to route requests to services based on the request host or path, centralizing a number of services into a single entrypoint.
Installed

Ingress Endpoint

35.234.70.160

Point a wildcard DNS to this generated endpoint in order to access your application after it has been deployed. More information

Cert-Manager
Cert-Manager is a native Kubernetes certificate management controller that helps with issuing certificates. Installing Cert-Manager on your cluster will issue a certificate by Let's Encrypt and ensure that certificates are valid and up-to-date.
Installed

Issuer Email

ylu@vistaprint.com

Issuers represent a certificate authority. You must provide an email address for your Issuer. More information

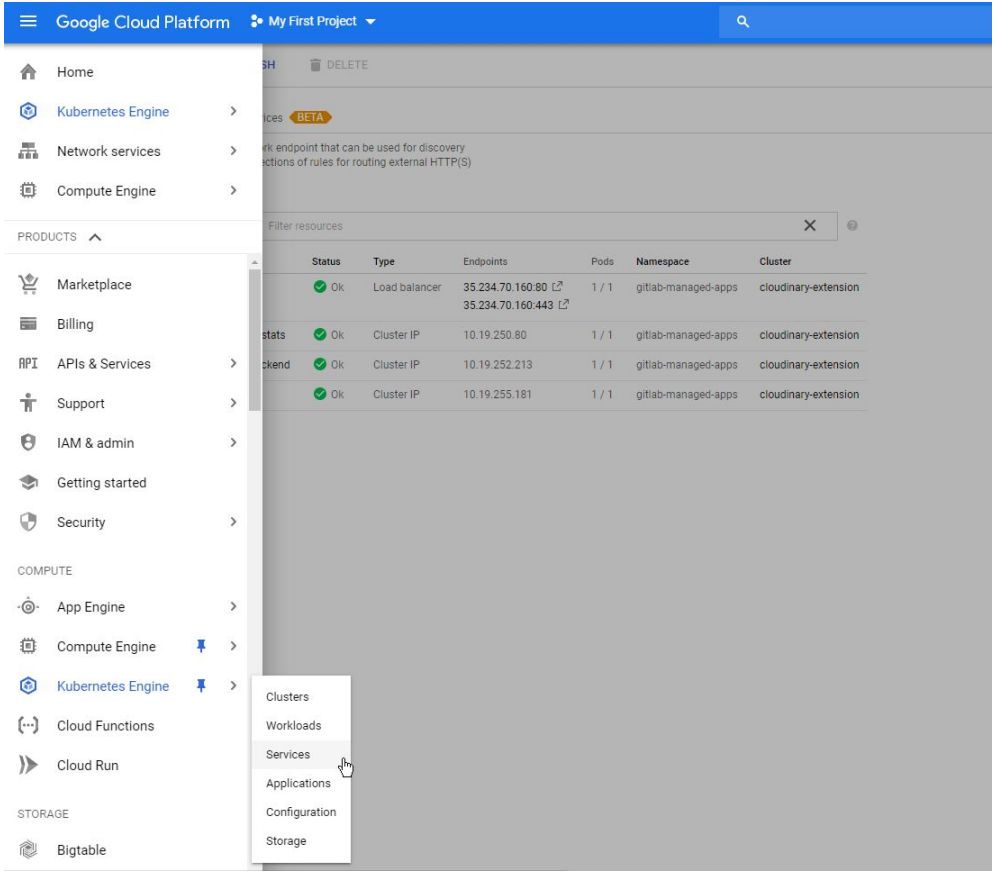
Prometheus
Prometheus is an open-source monitoring system with GitLab Integration to monitor deployed applications.
Install

GitLab Runner
GitLab Runner connects to the repository and executes CI/CD jobs, pushing results back and deploying applications to production.
Installed

JupyterHub
JupyterHub, a multi-user Hub, spawns, manages, and proxies multiple instances of the single-user Jupyter notebook server. JupyterHub can be used to serve notebooks to a class of students, a corporate data science group, or a scientific research group.
Install

Step 2: configure the cluster

In the console, go to the cluster services, and you will see those apps installed:



Step 2: configure the cluster

Services REFRESH DELETE

Kubernetes services Brokered services BETA

Services are sets of Pods with a network endpoint that can be used for discovery and load balancing. Ingresses are collections of rules for routing external HTTP(S) traffic to Services.

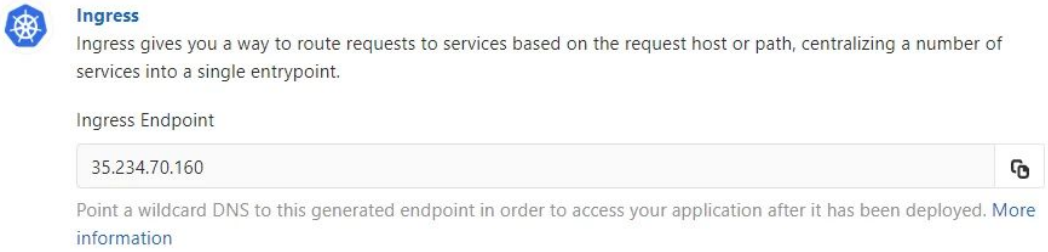
Is system object : False Filter resources

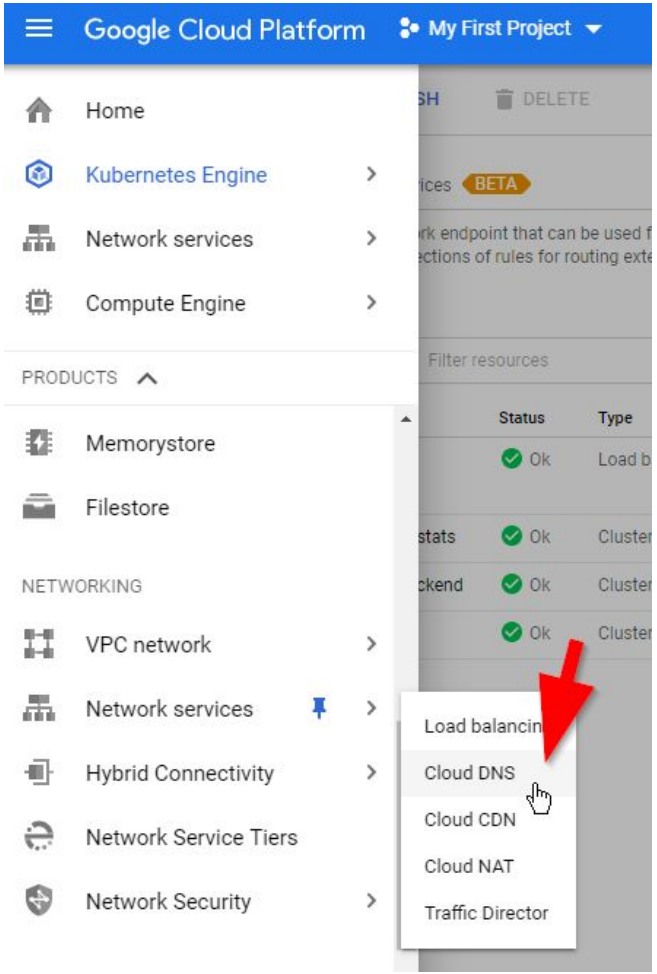
Name	Status	Type	Endpoints	Pods	Namespace	Cluster
ingress-nginx-ingress-controller	Ok	Load balancer	35.234.70.160:80 35.234.70.160:443	1 / 1	gitlab-managed-apps	cloudinary-extension
ingress-nginx-ingress-controller-stats	Ok	Cluster IP	10.19.250.80	1 / 1	gitlab-managed-apps	cloudinary-extension
ingress-nginx-ingress-default-backend	Ok	Cluster IP	10.19.252.213	1 / 1	gitlab-managed-apps	cloudinary-extension
tiller-deploy	Ok	Cluster IP	10.19.255.181	1 / 1	gitlab-managed-apps	cloudinary-extension

Step 3: Configure wildcard DNS to ingress endpoint

- In the navigation panel, go to Cloud DNS. This is the AWS Route53 equivalent in GCP.

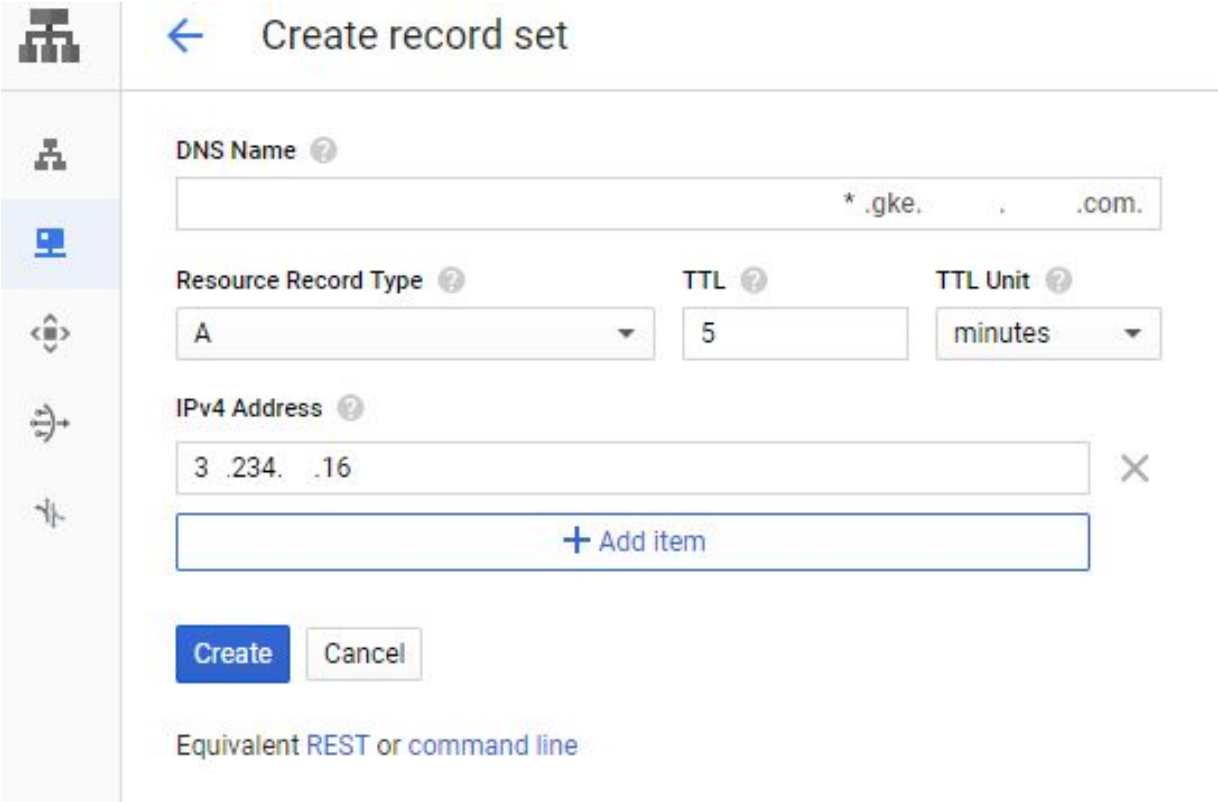
- Copy the IP shown in GitLab Kubernetes Application "Ingress".





Step 3: Configure wildcard DNS to ingress endpoint

- In the zone where your base domain is created in, add a record of type "A", which maps a DNS name to IPV4.



Step 3: Configure wildcard DNS to ingress endpoint

Zone details EDIT ADD RECORD SET DELETE ZONE

services

DNS name: gke. .v .com. Type: Public

DNS peering: Disabled

Record sets

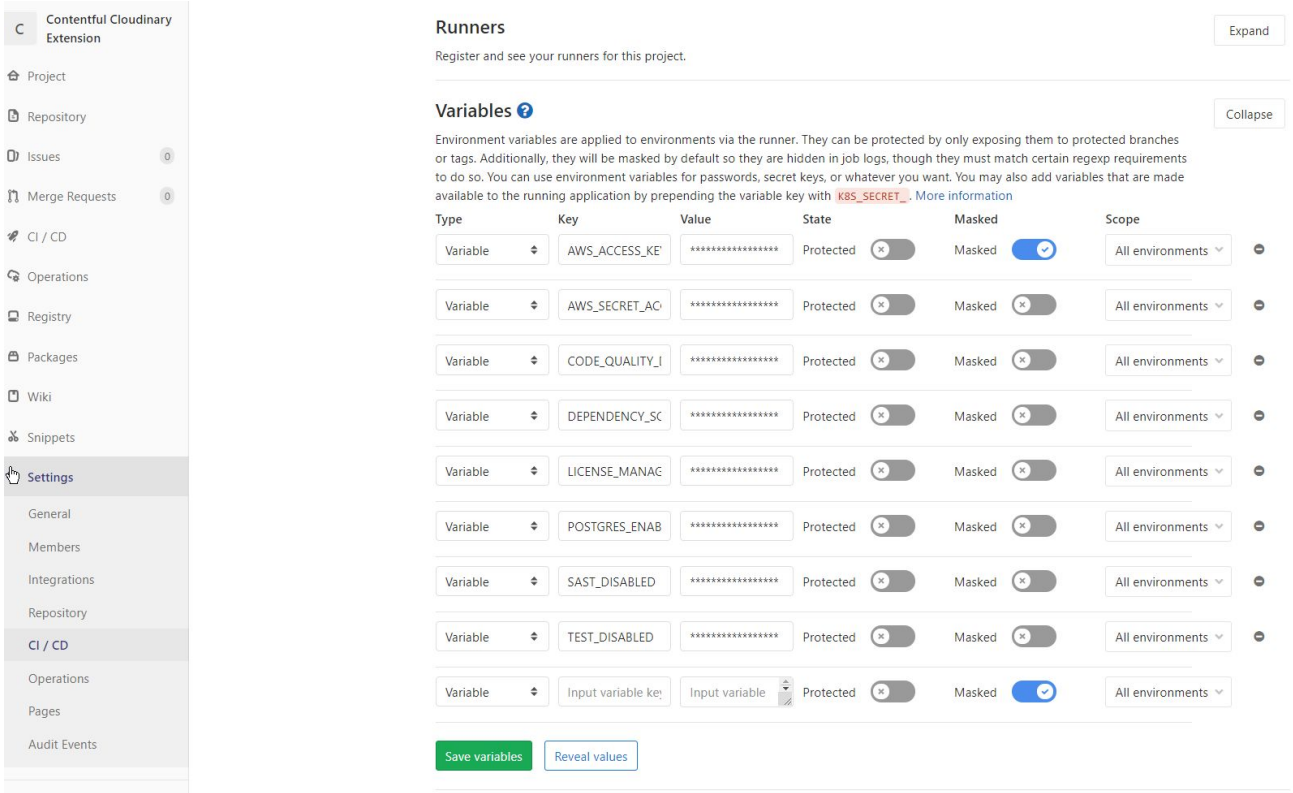
Add record set Delete record sets

Filter record sets

DNS name	Type	TTL (seconds)	Data
gke. .v .com.	NS	21600	ns-cloud-c1.googledomains.com. ns-cloud-c2.googledomains.com. ns-cloud-c3.googledomains.com. ns-cloud-c4.googledomains.com.
gke. .v .com.	SOA	21600	ns-cloud-c1.googledomains.com. cloud-dns-hostmaster.google.com. 1 21600 3600 259200 300
*.gke. .v .com.	A	300	3 .234. .16

Step 4: Configure jobs

GitLab provides you with a .gitlab-ci.yml template in AutoDevOps. You can turn on/off some jobs by setting values in Environment Variables (Settings > CI/CD > Variables). List of configurations you can use: https://docs.gitlab.com/ee/topics/autodevops/index.html#environment-variables



Step 5: Configure alternate DNS name

- By default, the DNS name configured by Auto DevOps makes use of the path to the repo in GitLab, which is usually very long.
- You can add ADDITIONAL_HOSTS to environment variables (https://docs.gitlab.com/ee/topics/autodevops/#environment-variables). It needs to be a fully qualified name, can be a comma separated list of fqdn names.
- If you configured the ADDITIONAL_HOSTS after deploying to kubernetes, you need to delete the Ingress type service in the list of services first for the certs to work on additional hosts.
- The job only creates a new one when there is no existing ingress service. Otherwise, the additional hosts will only be accessible via HTTP, not HTTPS.

Step 6: Run the pipeline

Go to your repo in GitLab. Go to CI/CD on the left panel, and click on "Pipeline". On top right of the page, you will see a button "Run Pipeline". Wait patiently until the job finishes.