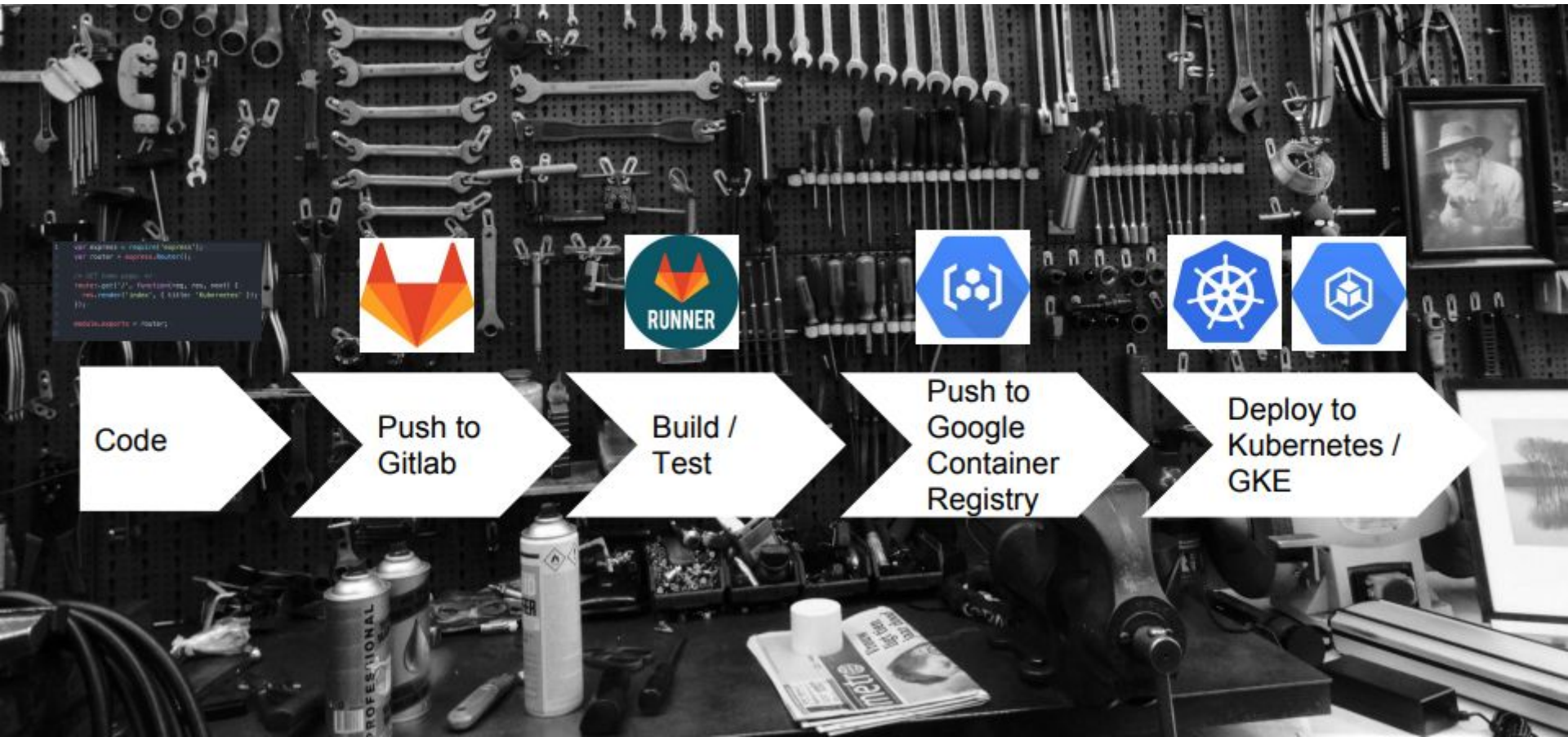
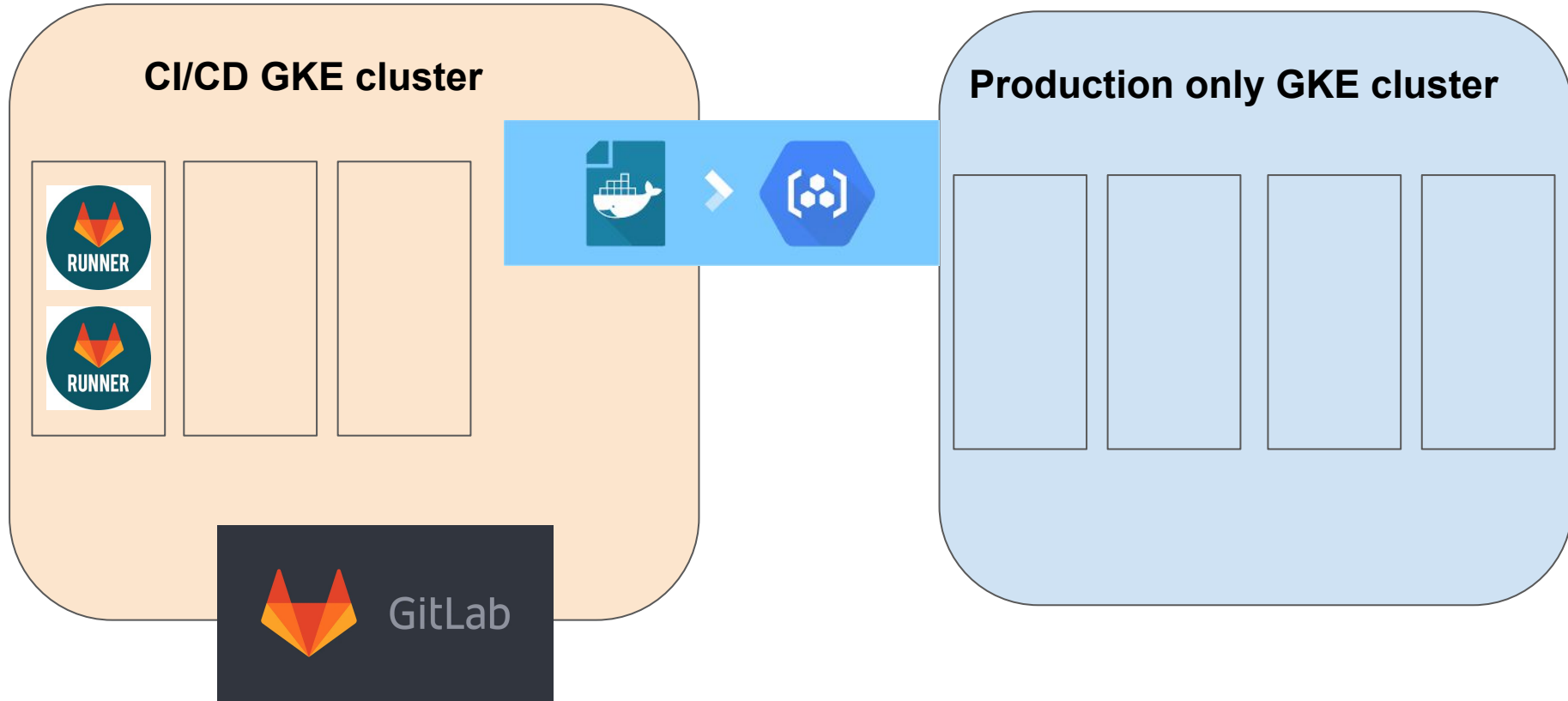


Microservices CI/CD proposal

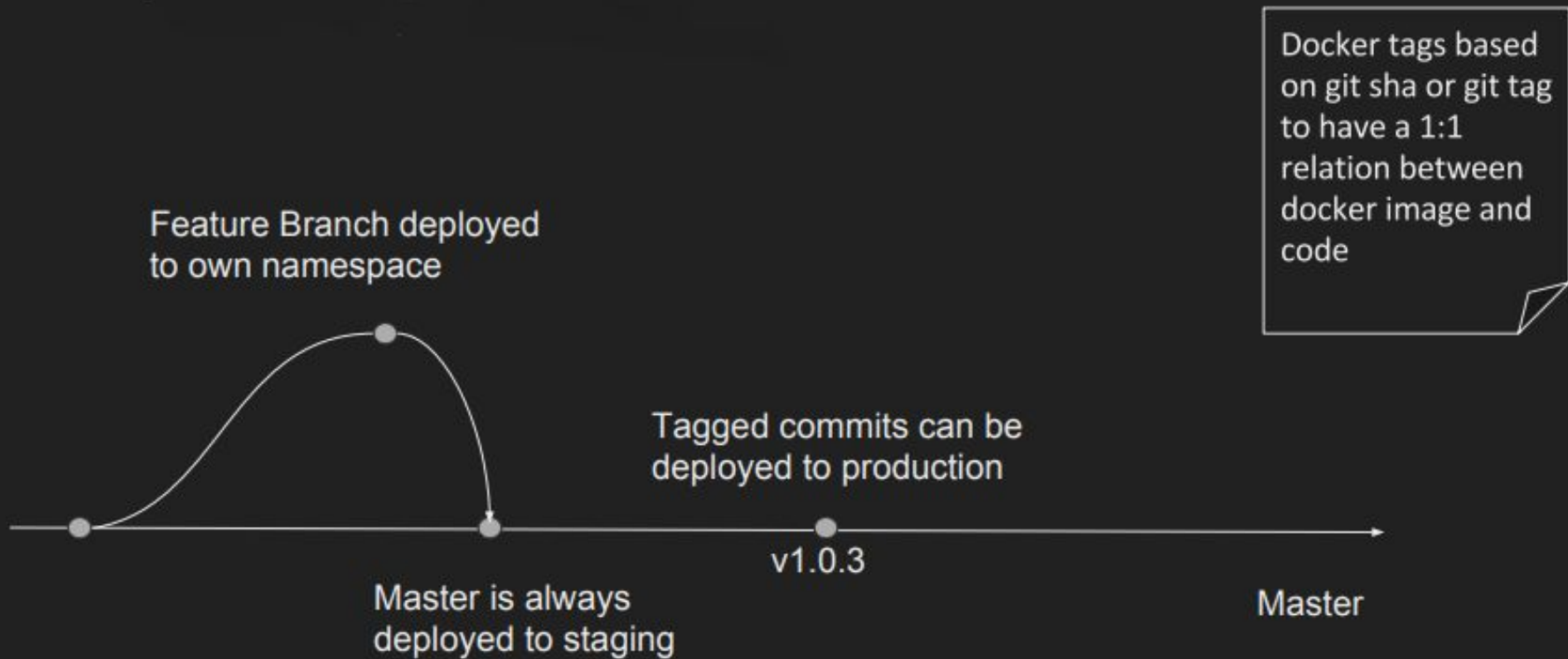
Use the same basic Tools



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 we are creating a new one).
- Make sure the GitLab-managed cluster is checked since we'd like to use AutoDevOps.

Enter the details for your Kubernetes cluster

Please make sure that your Google account meets the following requirements:

- Your account must have [access to Google Kubernetes Engine](#)
- Make sure your account [meets the requirements](#) to create Kubernetes clusters
- This account must have permissions to create a Kubernetes cluster in the [Google Kubernetes Engine project](#) specified below

Read our [help page](#) on Kubernetes cluster integration.

[Select a different Google account](#)

Kubernetes cluster name

Environment scope

Choose which of your environments will use this cluster.

Google Cloud Platform project

The project you/your team has created.

To use a new project, first create one on [Google Cloud Platform](#).

Zone

Pick a zone that is close to where other services are hosted.

[Learn more about zones](#)

Number of nodes

Machine type

[Learn more about machine types](#) and [pricing](#).

☒ RBAC-enabled cluster

Enable this setting if using role-based access control (RBAC). This option will allow you to install applications on RBAC clusters. [More information](#)

☒ GitLab-managed cluster

Allow GitLab to manage namespace and service accounts for this cluster. [More information](#)

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.

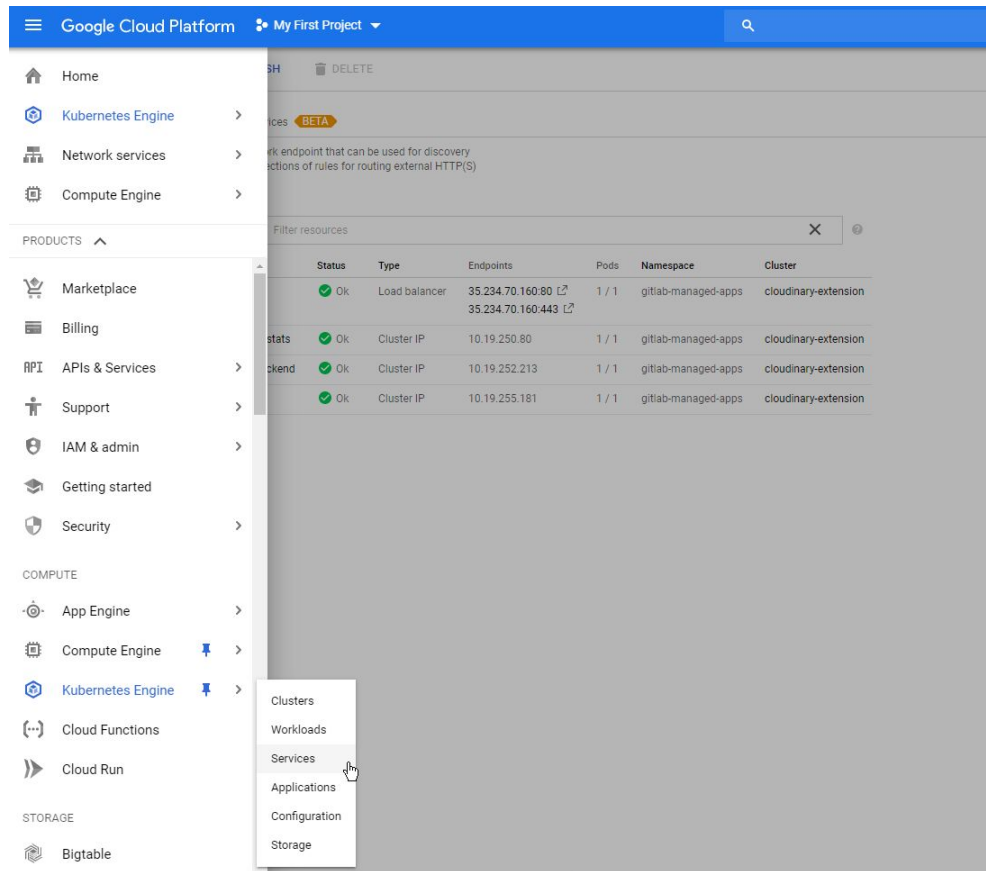
Applications

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 endpoint. Ingress Endpoint <input type="text" value="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	Installed
	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. Issuer Email <input type="text" value="ylu@vistaprint.com"/> Issuers represent a certificate authority. You must provide an email address for your Issuer. More information	Installed
	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:





The screenshot shows the Google Cloud Platform console interface. The left sidebar contains a navigation menu with categories like PRODUCTS, COMPUTE, and STORAGE. Under the 'Kubernetes Engine' category, a sub-menu is open, highlighting 'Services'. The main content area displays a table of installed services for the 'cloudinary-extension' cluster.

Status	Type	Endpoints	Pods	Namespace	Cluster
Ok	Load balancer	35.234.70.160.80 35.234.70.160.443	1 / 1	gitlab-managed-apps	cloudinary-extension
Ok	Cluster IP	10.19.250.80	1 / 1	gitlab-managed-apps	cloudinary-extension
Ok	Cluster IP	10.19.252.213	1 / 1	gitlab-managed-apps	cloudinary-extension
Ok	Cluster IP	10.19.255.181	1 / 1	gitlab-managed-apps	cloudinary-extension

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  

<input type="checkbox"/> Name ^	Status	Type	Endpoints	Pods	Namespace	Cluster
<input type="checkbox"/> ingress-nginx-ingress-controller	 Ok	Load balancer	35.234.70.160:80  35.234.70.160:443 	1 / 1	gitlab-managed-apps	cloudinary-extension
<input type="checkbox"/> ingress-nginx-ingress-controller-stats	 Ok	Cluster IP	10.19.250.80	1 / 1	gitlab-managed-apps	cloudinary-extension
<input type="checkbox"/> ingress-nginx-ingress-default-backend	 Ok	Cluster IP	10.19.252.213	1 / 1	gitlab-managed-apps	cloudinary-extension
<input type="checkbox"/> 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".



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 endpoint.

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](#)

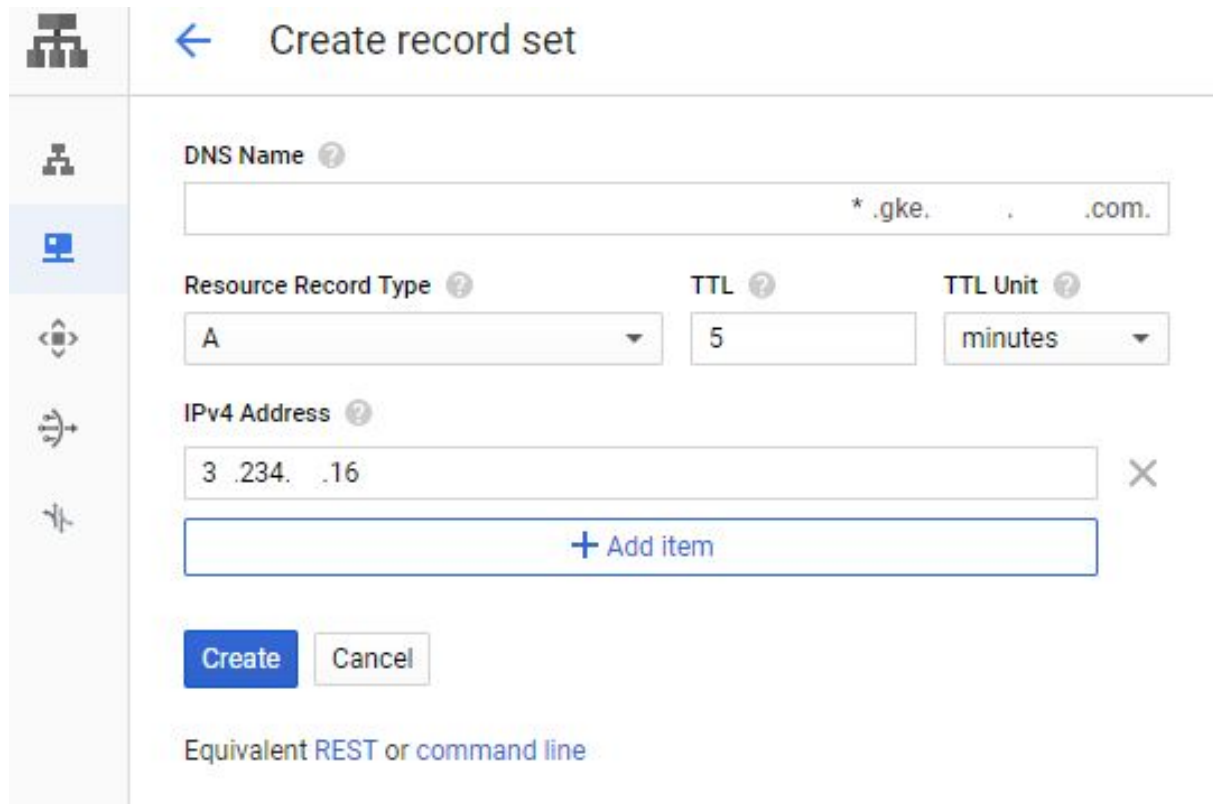
The screenshot shows the Google Cloud Platform navigation menu. The 'PRODUCTS' section is expanded, and a red arrow points to the 'Cloud DNS' option. Other visible options include Home, Kubernetes Engine, Network services, Compute Engine, Memorystore, Filestore, VPC network, Network services, Hybrid Connectivity, Network Service Tiers, and Network Security.

PRODUCTS
Home
Kubernetes Engine
Network services
Compute Engine
Memorystore
Filestore
VPC network
Network services
Hybrid Connectivity
Network Service Tiers
Network Security

Cloud DNS
Load balancing
Cloud DNS
Cloud CDN
Cloud NAT
Traffic Director

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.



The screenshot shows the 'Create record set' interface in the Google Cloud DNS console. On the left is a vertical sidebar with icons for DNS zones, record sets, DNSSEC, and other DNS-related features. The main area is titled 'Create record set' with a back arrow. It contains the following fields and controls:

- DNS Name**: A text input field with a help icon. The value entered is `*.gke. .com.`.
- Resource Record Type**: A dropdown menu with a help icon, currently set to `A`.
- TTL**: A text input field with a help icon, containing the value `5`.
- TTL Unit**: A dropdown menu with a help icon, currently set to `minutes`.
- IPv4 Address**: A text input field with a help icon, containing the value `3 .234. .16`. To the right of the field is a close icon (X).
- + Add item**: A button to add additional IPv4 addresses.
- Create** and **Cancel**: Two buttons at the bottom of the form.
- Equivalent REST or command line**: A link at the bottom of the page.

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

<input type="checkbox"/> 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	
<input type="checkbox"/> *.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>

C

Contentful Cloudinary Extension

Project

Repository

Issues0

Merge Requests0

CI / CD

Operations

Registry

Packages

Wiki

Snippets

Settings

General

Members

Integrations

Repository

CI / CD

Operations

Pages

Audit Events

Runners

Register and see your runners for this project.

Expand

Variables

Collapse

Environment variables are applied to environments via the runner. They can be protected by only exposing them to protected branches or tags. Additionally, they will be masked by default so they are hidden in job logs, though they must match certain regex requirements to do so. You can use environment variables for passwords, secret keys, or whatever you want. You may also add variables that are made available to the running application by prepending the variable key with `K8S_SECRET_`. [More information](#)

Type	Key	Value	State	Masked	Scope	
Variable	AWS_ACCESS_KEY	*****	Protected	<input type="checkbox"/>	Masked	<input checked="" type="checkbox"/>
Variable	AWS_SECRET_KEY	*****	Protected	<input type="checkbox"/>	Masked	<input type="checkbox"/>
Variable	CODE_QUALITY_I	*****	Protected	<input type="checkbox"/>	Masked	<input type="checkbox"/>
Variable	DEPENDENCY_SC	*****	Protected	<input type="checkbox"/>	Masked	<input type="checkbox"/>
Variable	LICENSE_MANAG	*****	Protected	<input type="checkbox"/>	Masked	<input type="checkbox"/>
Variable	POSTGRES_ENAB	*****	Protected	<input type="checkbox"/>	Masked	<input type="checkbox"/>
Variable	SAST_DISABLED	*****	Protected	<input type="checkbox"/>	Masked	<input type="checkbox"/>
Variable	TEST_DISABLED	*****	Protected	<input type="checkbox"/>	Masked	<input type="checkbox"/>
Variable	Input variable key	Input variable	Protected	<input type="checkbox"/>	Masked	<input checked="" type="checkbox"/>

Save variables

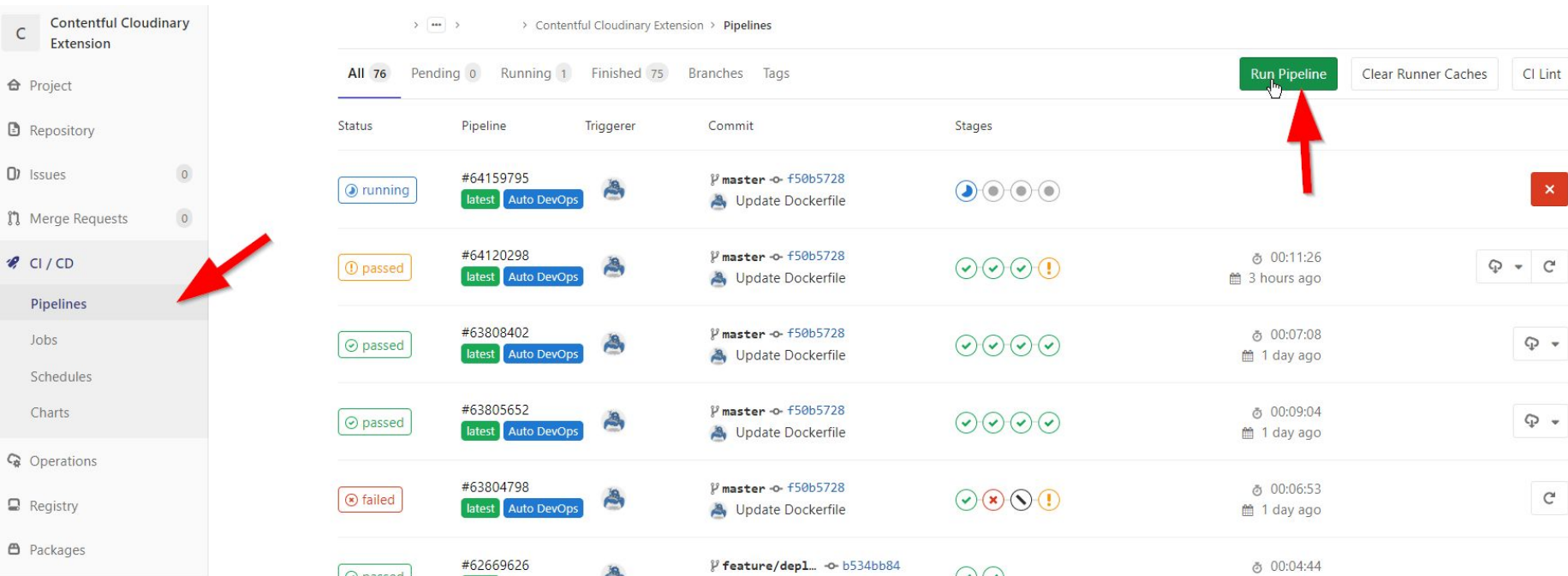
Reveal values

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.



The screenshot displays the GitLab CI/CD interface for the 'Contentful Cloudinary Extension' project. The left sidebar contains the navigation menu, with 'CI / CD' and 'Pipelines' highlighted. A red arrow points to 'Pipelines'. The main area shows a table of pipeline runs. A 'Run Pipeline' button is highlighted in the top right with a red arrow.

Contentful Cloudinary Extension

> Contentful Cloudinary Extension > Pipelines

All 76 Pending 0 Running 1 Finished 75 Branches Tags

Run Pipeline Clear Runner Caches CI Lint

Status	Pipeline	Triggerer	Commit	Stages	
running	#64159795 latest Auto DevOps		master -> f50b5728 Update Dockerfile		
passed	#64120298 latest Auto DevOps		master -> f50b5728 Update Dockerfile		00:11:26 3 hours ago
passed	#63808402 latest Auto DevOps		master -> f50b5728 Update Dockerfile		00:07:08 1 day ago
passed	#63805652 latest Auto DevOps		master -> f50b5728 Update Dockerfile		00:09:04 1 day ago
failed	#63804798 latest Auto DevOps		master -> f50b5728 Update Dockerfile		00:06:53 1 day ago
passed	#62669626		feature/depl... -> b534bb84		00:04:44