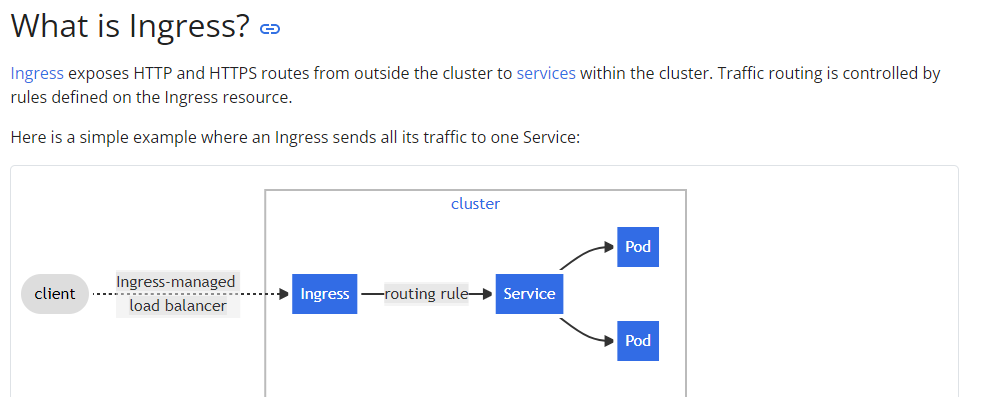
What is Ingress



An Ingress may be configured to give Services externally-reachable URLs, load balance traffic, terminate SSL / TLS, and offer name-based virtual hosting. An [Ingress controller](https://kubernetes.io/docs/concepts/services-networking/ingress-controllers) is responsible for fulfilling the Ingress, usually with a load balancer, though it may also configure your edge router or additional frontends to help handle the traffic.

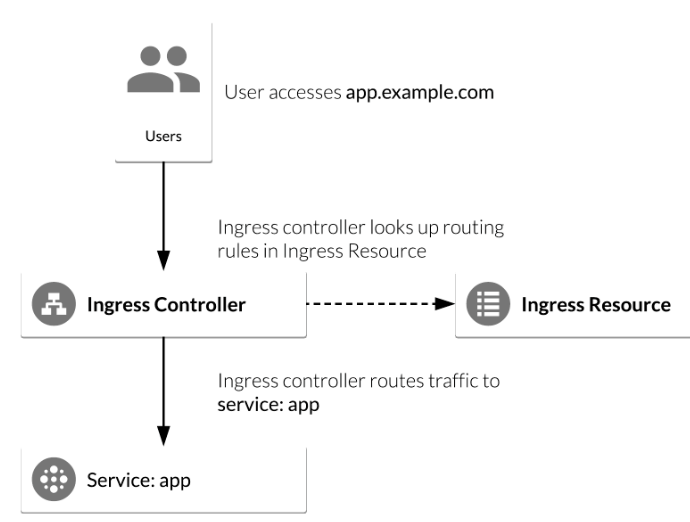
An Ingress does not expose arbitrary ports or protocols. Exposing services other than HTTP and HTTPS to the internet typically uses a service of type [Service.Type=NodePort](https://kubernetes.io/docs/concepts/services-networking/service/" \l "type-nodeport) or [Service.Type=LoadBalancer](https://kubernetes.io/docs/concepts/services-networking/service/" \l "loadbalancer)

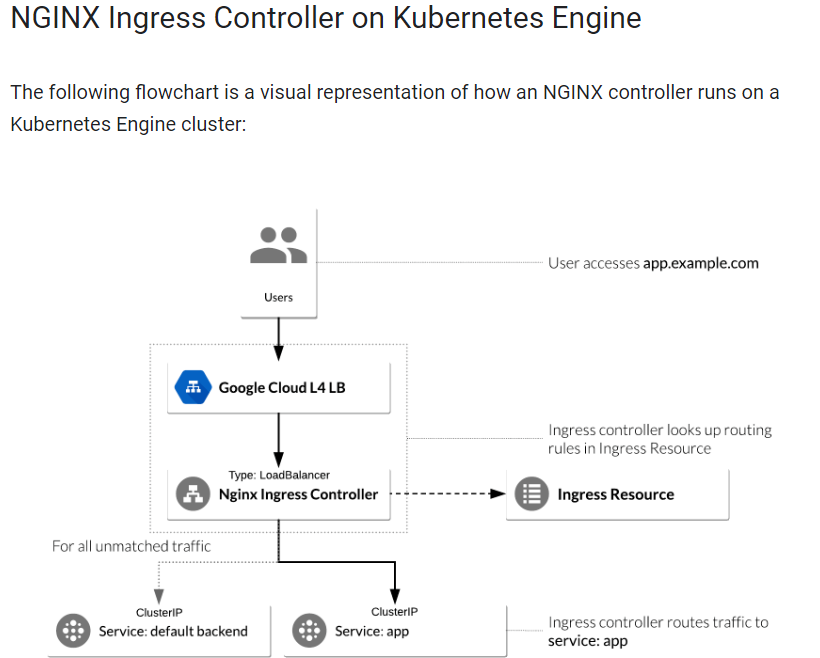
# **NGINX Ingress Controller on Google Kubernetes Engine**

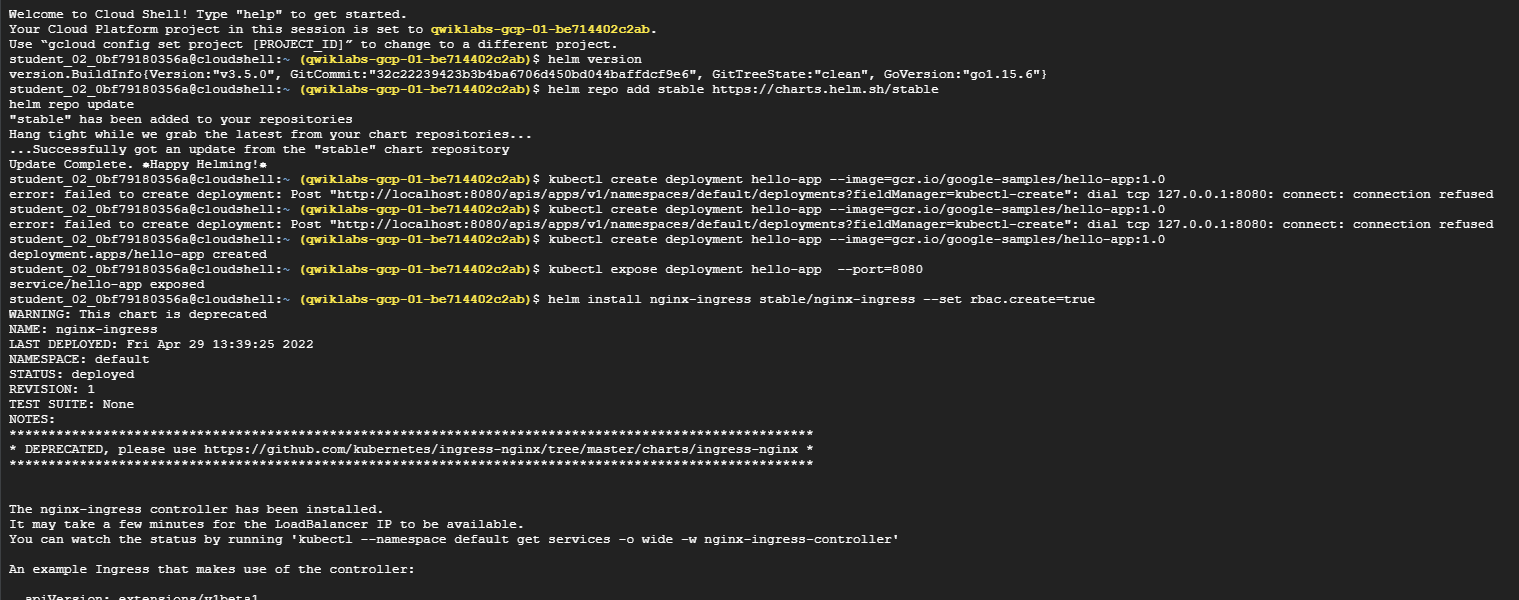
In Kubernetes, [Ingress](https://kubernetes.io/docs/concepts/services-networking/ingress/) allows external users and client applications access to HTTP services.

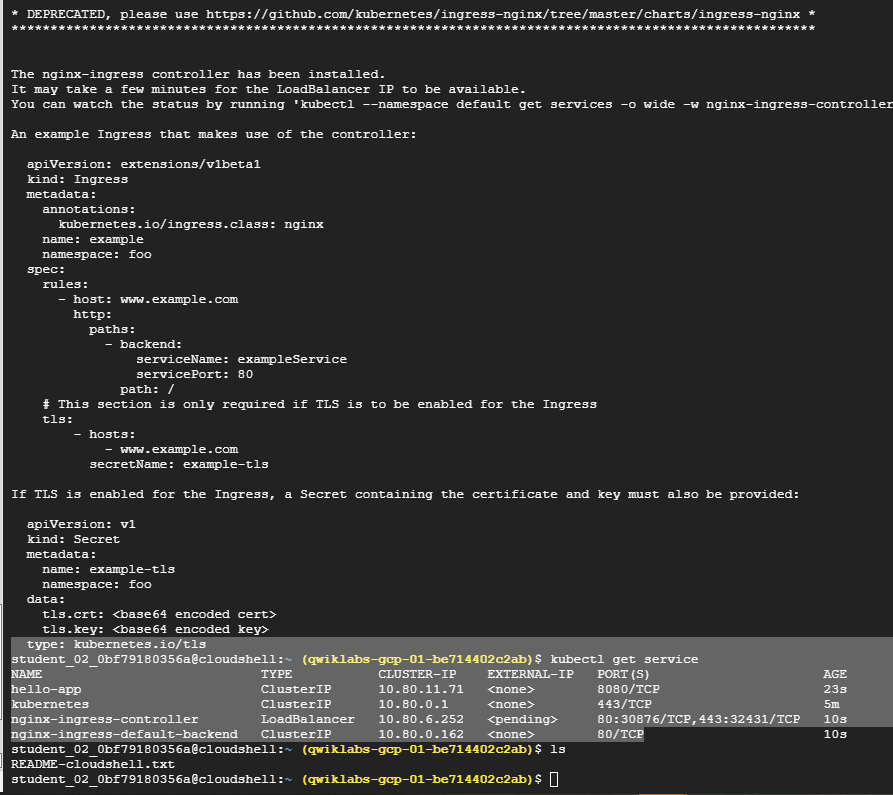
Ingress consists of two components: an *Ingress Resource* and an *Ingress Controller*:

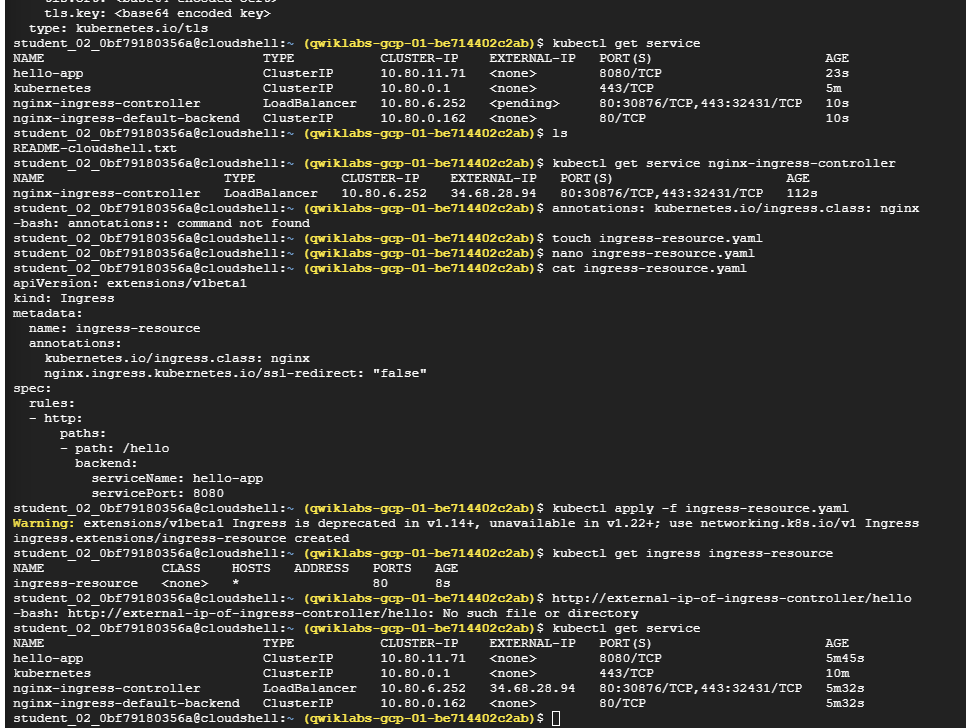
* **Ingress Resource** is a collection of rules for the inbound traffic to reach Services. These are Layer 7 (L7) rules that allow hostnames (and optionally paths) to be directed to specific Services in Kubernetes.
* **Ingress Controller** acts upon the rules set by the Ingress Resource, typically via an HTTP or L7 load balancer. It is vital that both pieces are properly configured so that traffic can be routed from an outside client to a Kubernetes Service.

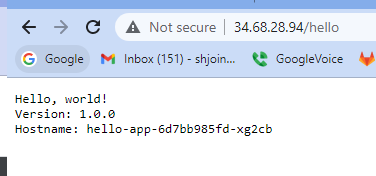


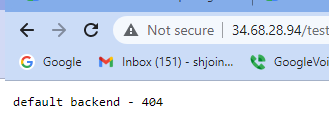












## Overview

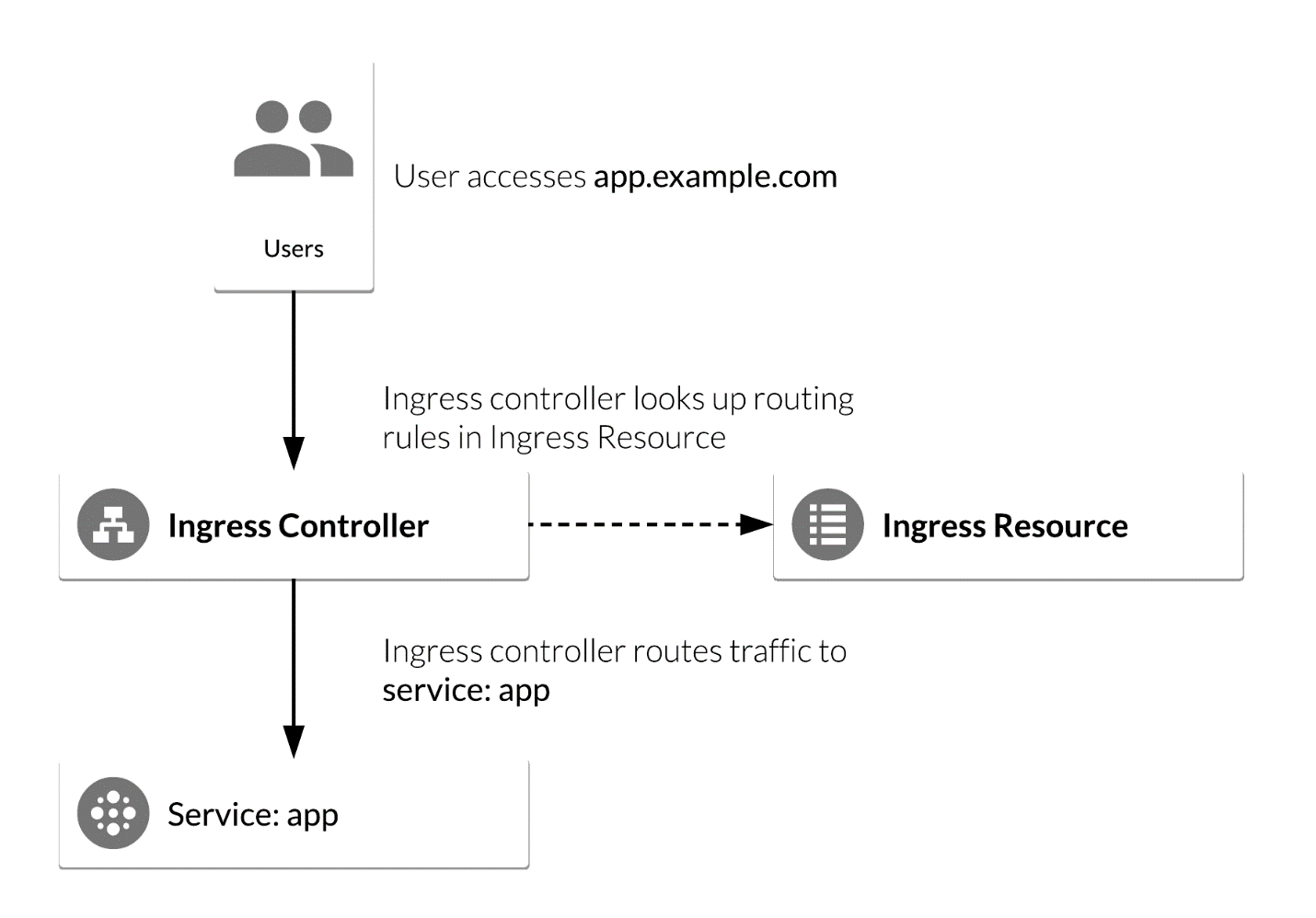
In Kubernetes, [Ingress](https://kubernetes.io/docs/concepts/services-networking/ingress/) allows external users and client applications access to HTTP services. Ingress consists of two components: an Ingress Resource and an Ingress Controller:

* **Ingress Resource** is a collection of rules for the inbound traffic to reach Services. These are Layer 7 (L7) rules that allow hostnames (and optionally paths) to be directed to specific Services in Kubernetes.
* **Ingress Controller** acts upon the rules set by the Ingress Resource, typically via an HTTP or L7 load balancer. It is vital that both pieces are properly configured so that traffic can be routed from an outside client to a Kubernetes Service.

NGINX—a high performance web server—is a popular choice for an Ingress Controller because of its robustness and the many features it boasts. For example, it supports:

* **Websockets**, which allows you to load balance Websocket applications.
* **SSL Services**, which allows you to load balance HTTPS applications.
* **Rewrites**, which allows you to rewrite the URI of a request before sending it to the application.
* **Session Persistence** (NGINX Plus only), which guarantees that all the requests from the same client are always passed to the same backend container.
* **JWTs** (NGINX Plus only), which allows NGINX Plus to authenticate requests by validating JSON Web Tokens (JWTs).

The following diagram illustrates the basic flow of an Ingress Controller in Google Cloud and gives you a rough idea of what you'll be building:



## Objectives

In this lab, you will configure a Kubernetes deployment with an Ingress Resource. You will use NGINX as an Ingress Controller, which you will use to route and load balance traffic from external clients to the deployment. More specifically, you will:

* Deploy a simple Kubernetes web application.
* Deploy an NGINX Ingress Controller using a stable Helm Chart.
* Deploy an Ingress Resource for the application that uses NGINX Ingress as the controller.
* Test NGINX Ingress functionality by accessing the Google Cloud L4 (TCP/UDP) Load Balancer frontend IP and ensure it can access the web application.

## Prerequisites

This is an **advanced level** lab. Experience with Kubernetes and/or containerized applications is suggested. Familiarity with NGINX and Helm is recommended, but not required. If you are looking to get up to speed in these services, be sure to check out the following labs:

* [Kubernetes Engine: Qwik Start](https://google.qwiklabs.com/catalog_lab/911)
* [Managing Deployments Using Kubernetes Engine](https://google.qwiklabs.com/catalog_lab/572)
* [Distributed Load Testing Using Kubernetes](https://google.qwiklabs.com/catalog_lab/936)
* [Helm Package Manager](https://google.qwiklabs.com/catalog_lab/958)

Once you're ready, scroll down to get your lab environment set up.

## Setup

#### Before you click the Start Lab button

Read these instructions. Labs are timed and you cannot pause them. The timer, which starts when you click **Start Lab**, shows how long Google Cloud resources will be made available to you.

This hands-on lab lets you do the lab activities yourself in a real cloud environment, not in a simulation or demo environment. It does so by giving you new, temporary credentials that you use to sign in and access Google Cloud for the duration of the lab.

#### What you need

To complete this lab, you need:

* Access to a standard internet browser (Chrome browser recommended).
* Time to complete the lab.

**Note:** If you already have your own personal Google Cloud account or project, do not use it for this lab.

**Note:** If you are using a Chrome OS device, open an Incognito window to run this lab.

#### How to start your lab and sign in to the Google Cloud Console

1. Click the **Start Lab** button. If you need to pay for the lab, a pop-up opens for you to select your payment method. On the left is a panel populated with the temporary credentials that you must use for this lab.



1. Copy the username, and then click **Open Google Console**. The lab spins up resources, and then opens another tab that shows the **Sign in** page.



**Tip:** Open the tabs in separate windows, side-by-side.

If you see the **Choose an account** page, click **Use Another Account**. 

1. In the **Sign in** page, paste the username that you copied from the left panel. Then copy and paste the password.

**Important:** You must use the credentials from the left panel. Do not use your Google Cloud Training credentials. If you have your own Google Cloud account, do not use it for this lab (avoids incurring charges).

1. Click through the subsequent pages:
   * Accept the terms and conditions.
   * Do not add recovery options or two-factor authentication (because this is a temporary account).
   * Do not sign up for free trials.

After a few moments, the Cloud Console opens in this tab.

**Note:** You can view the menu with a list of Google Cloud Products and Services by clicking the **Navigation menu** at the top-left. 

### **Activate Cloud Shell**

Cloud Shell is a virtual machine that is loaded with development tools. It offers a persistent 5GB home directory and runs on the Google Cloud. Cloud Shell provides command-line access to your Google Cloud resources.

In the Cloud Console, in the top right toolbar, click the **Activate Cloud Shell** button.



Click **Continue**.



It takes a few moments to provision and connect to the environment. When you are connected, you are already authenticated, and the project is set to your PROJECT\_ID. For example:



gcloud is the command-line tool for Google Cloud. It comes pre-installed on Cloud Shell and supports tab-completion.

You can list the active account name with this command:

gcloud auth list

Copied!

content\_copy

(Output)

ACTIVE: \*

ACCOUNT: student-01-xxxxxxxxxxxx@qwiklabs.net

To set the active account, run:

$ gcloud config set account `ACCOUNT`

You can list the project ID with this command:

gcloud config list project

Copied!

content\_copy

(Output)

[core]

project = <project\_ID>

(Example output)

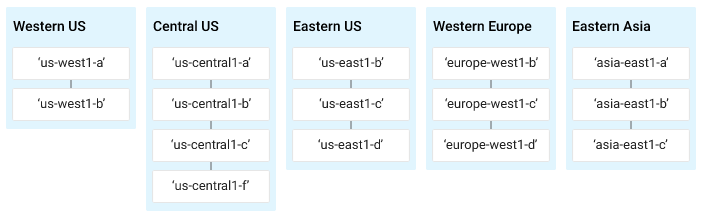
[core]

project = qwiklabs-gcp-44776a13dea667a6

For full documentation of gcloud see the [gcloud command-line tool overview](https://cloud.google.com/sdk/gcloud" \t "_blank).

### **Understanding Regions and Zones**

Certain Compute Engine resources live in regions or zones. A region is a specific geographical location where you can run your resources. Each region has one or more zones. For example, the us-central1 region denotes a region in the Central United States that has zones us-central1-a, us-central1-b, us-central1-c, and us-central1-f.



Resources that live in a zone are referred to as zonal resources. Virtual machine Instances and persistent disks live in a zone. To attach a persistent disk to a virtual machine instance, both resources must be in the same zone. Similarly, if you want to assign a static IP address to an instance, the instance must be in the same region as the static IP.

Learn more about regions and zones and see a complete list in [Regions & Zones documentation](https://cloud.google.com/compute/docs/regions-zones/).

## Set a zone

Before creating a Kubernetes cluster, you'll have to set a default computing zone for our Google Cloud project. Run the following command to see a [list of Google Cloud zones](https://cloud.google.com/compute/docs/regions-zones/):

gcloud compute zones list

Copied!

content\_copy

Now run the following command to set your zone (in this case to us-central1-a):

gcloud config set compute/zone us-central1-a

Copied!

content\_copy

## Create a Kubernetes cluster

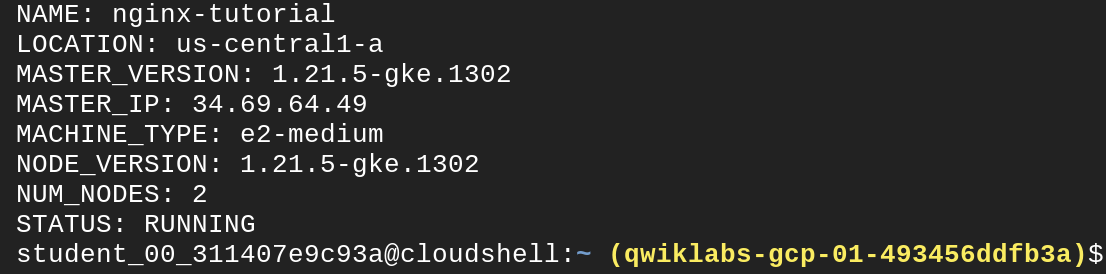
Now that our zone is configured, deploy a Kubernetes Engine cluster. Run the following command to create a cluster named nginx-tutorial that's made up of two nodes (or worker machines):

gcloud container clusters create nginx-tutorial --num-nodes 2

Copied!

content\_copy

It will take a few minutes for this command to complete. Continue when you get a similar output in Cloud Shell:



### **Test Completed Task**

Click **Check my progress** to verify your performed task. If you have successfully created Kubernetes cluster, you will see an assessment score.

Create a Kubernetes cluster

Check my progress

## Install Helm

Now that you have your Kubernetes cluster up and running, install [Helm](https://helm.sh/). Helm is a tool that streamlines Kubernetes application installation and management. You can think of it like apt, yum, or homebrew for Kubernetes. Helm Charts are maintained by the Kubernetes community.

Run helm version in Cloud Shell to check which version you are using and also ensure that Helm is installed:

helm version

Copied!

content\_copy

Add the chart repository and ensure the chart list is up to date:

helm repo add stable https://charts.helm.sh/stable

helm repo update

Copied!

content\_copy

## Deploy an application in Kubernetes Engine

Now that you have Helm configured, deploy a simple web-based application from the Google Cloud Repository. This application will be used as the backend for the Ingress.

From the Cloud Shell, run the following command:

kubectl create deployment hello-app --image=gcr.io/google-samples/hello-app:1.0

Copied!

content\_copy

Your output should resemble the following:

deployment.apps/hello-app created

### **Test Completed Task**

Click **Check my progress** to verify your performed task. If you have successfully deployed an application in Kubernetes Engine, you will see an assessment score.

Deploy an application in Kubernetes Engine

Check my progress

Now expose the hello-app Deployment as a Service by running the following command:

kubectl expose deployment hello-app --port=8080

Copied!

content\_copy

Your output should resemble the following:

service/hello-app exposed

### **Test Completed Task**

Click **Check my progress** to verify your performed task. If you have successfully exposed the created deployment as a service, you will see an assessment score.

Expose the created deployment as a service

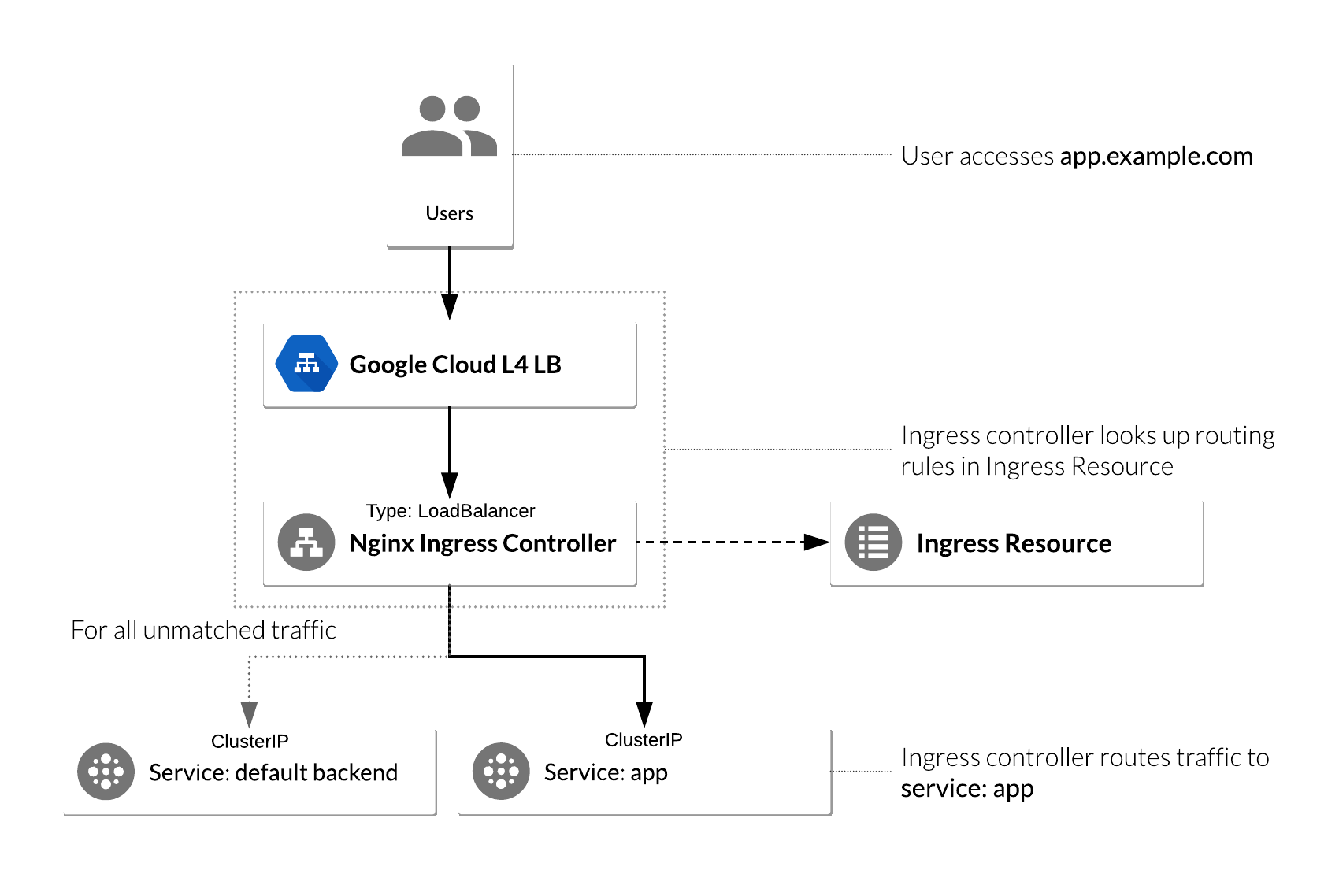
Check my progress

## Deploying the NGINX Ingress Controller via Helm

The Kubernetes platform gives administrators flexibility when it comes to Ingress Controllers—you can integrate your own rather than having to work with your provider's built-in offering. The NGINX controller must be exposed for external access. This is done using Service type: LoadBalancer on the NGINX controller service. On Kubernetes Engine, this creates a Google Cloud Network (TCP/IP) Load Balancer with NGINX controller Service as a backend. Google Cloud also creates the appropriate firewall rules within the Service's VPC to allow web HTTP(S) traffic to the load balancer frontend IP address.

### **NGINX Ingress Controller on Kubernetes Engine**

The following flowchart is a visual representation of how an NGINX controller runs on a Kubernetes Engine cluster:



### **Deploy NGINX Ingress Controller**

Now that you have the bigger picture in mind, go ahead and deploy the NGINX Ingress Controller. Run the following command to do so:

helm install nginx-ingress stable/nginx-ingress --set rbac.create=true

Copied!

content\_copy

Please ignore if any deprecation warning.

kubectl get service

Copied!

content\_copy

Note the service, nginx-ingress-default-backend. The default backend is a Service which handles all URL paths and hosts the NGINX controller. The default backend exposes two URLs:

* /healthz that returns 200
* / that returns 404

Wait a few moments while the Google Cloud L4 Load Balancer gets deployed. Confirm that the nginx-ingress-controller Service has been deployed and that you have an external IP address associated with the service by running the following command:

kubectl get service nginx-ingress-controller

Copied!

content\_copy

You receive a similar output:

NAME TYPE CLUSTER-IP EXTERNAL-IP

nginx-ingress-controller LoadBalancer 10.7.248.226 35.226.162.176

### **Test Completed Task**

Click **Check my progress** to verify your performed task. If you have successfully deployed the NGINX Ingress Controller via Helm, you will see an assessment score.

Deploy the NGINX Ingress Controller via Helm

Check my progress

## Configure Ingress Resource to use NGINX Ingress Controller

An Ingress Resource object is a collection of L7 rules for routing inbound traffic to Kubernetes Services. Multiple rules can be defined in one Ingress Resource or they can be split up into multiple Ingress Resource manifests. The Ingress Resource also determines which controller to utilize to serve traffic. This can be set with an annotation, kubernetes.io/ingress.class, in the metadata section of the Ingress Resource. For the NGINX controller, you will use the nginx value as shown below:

annotations: kubernetes.io/ingress.class: nginx

Copied!

content\_copy

On Kubernetes Engine, if no annotation is defined under the metadata section, the Ingress Resource uses the Google Cloud GCLB L7 load balancer to serve traffic. This method can also be forced by setting the annotation's value to gce, like below:

annotations: kubernetes.io/ingress.class: gce

Copied!

content\_copy

Let's create a simple Ingress Resource YAML file which uses the NGINX Ingress Controller and has one path rule defined by typing the following commands:

touch ingress-resource.yaml

nano ingress-resource.yaml

Copied!

content\_copy

Add the following content in ingress-resource.yaml file:

apiVersion: extensions/v1beta1

kind: Ingress

metadata:

name: ingress-resource

annotations:

kubernetes.io/ingress.class: nginx

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

spec:

rules:

- http:

paths:

- path: /hello

backend:

serviceName: hello-app

servicePort: 8080

Copied!

content\_copy

then press **Ctrl-X**, then press **Y**, then press **Enter** to save the file.

The kind: Ingress dictates it is an Ingress Resource object. This Ingress Resource defines an inbound L7 rule for path /hello to service hello-app on port 8080.

Run the following command to apply those rules to your Kubernetes application:

kubectl apply -f ingress-resource.yaml

Copied!

content\_copy

Verify that Ingress Resource has been created:

kubectl get ingress ingress-resource

Copied!

content\_copy

**Note:** The IP address for the Ingress Resource will not be defined right away. Wait a few moments for the ADDRESS field to get populated.

Your output should resemble the following:

NAME HOSTS ADDRESS PORTS AGE

ingress-resource \* 80

### **Test Ingress and default backend**

You should now be able to access the web application by going to the EXTERNAL-IP/hello address of the **NGINX ingress controller** (found by running kubectl get service nginx-ingress-controller).

Open a new tab and go to the following, replacing the external-ip-of-ingress-controller with the external IP address of the NGINX ingress controller:

http://external-ip-of-ingress-controller/hello

Copied!

content\_copy

Your page should look similar to the following:



To check if the default-backend service is working properly, access any path (other than the path /hello defined in the Ingress Resource) and ensure you receive a 404 message. For example:

http://external-ip-of-ingress-controller/test

Copied!

content\_copy

Your page should look similar to the following:



## Congratulations!

Great work! In this lab you deployed a Kubernetes cluster with an NGINX Ingress Controller. You now have the experience and know-how to use Ingress Controllers in your own Kubernetes applications.



### **Finish Your Quest**

This self-paced lab is part of the [Kubernetes Solutions](https://google.qwiklabs.com/quests/45) Quest. A Quest is a series of related labs that form a learning path. Completing this Quest earns you the badge above, to recognize your achievement. You can make your badge (or badges) public and link to them in your online resume or social media account. Enroll in a Quest and get immediate completion credit if you've taken this lab. [See other available Qwiklabs Quests](https://google.qwiklabs.com/catalog).

### **Take Your Next Lab**

Check out these labs:

* [Continuous Delivery with Jenkins in Kubernetes Engine](https://google.qwiklabs.com/catalog_lab/984)
* [Distributed Load Testing Using Kubernetes](https://google.qwiklabs.com/catalog_lab/936)

### **Next Steps / Learn More**

* [Helm documentation](https://v2.helm.sh/docs/)
* [Kubernetes Ingress](https://kubernetes.io/docs/concepts/services-networking/ingress/)

### **Google Cloud Training & Certification**

...helps you make the most of Google Cloud technologies. [Our classes](https://cloud.google.com/training/courses) include technical skills and best practices to help you get up to speed quickly and continue your learning journey. We offer fundamental to advanced level training, with on-demand, live, and virtual options to suit your busy schedule. [Certifications](https://cloud.google.com/certification/) help you validate and prove your skill and expertise in Google Cloud technologies.

##### Manual Last Updated March 31, 2022

##### Lab Last Tested March 31, 2022

Copyright 2022 Google LLC All rights reserved. Google and the Google logo are trademarks of Google LLC. All other company and product names may be trademarks of the respective companies with which they are associated.