Deploy Kubernetes Load Balancer Service with Terraform

GSP233

Google Cloud Self-Paced Labs

Overview

In Terraform, a Provider is the logical abstraction of an upstream API. This lab will show you how to setup a Kubernetes cluster and deploy Load Balancer type NGINX service on it.

Objectives

In this lab, you will learn how to:

- Deploy a Kubernetes cluster along with a service using Terraform

Prerequisites

For this lab, you should have experience with the following:

- Familiarity with Kubernetes Services
- Familiarity with `kubectl` CLI.

Setup and Requirements

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 Qwiklabs 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

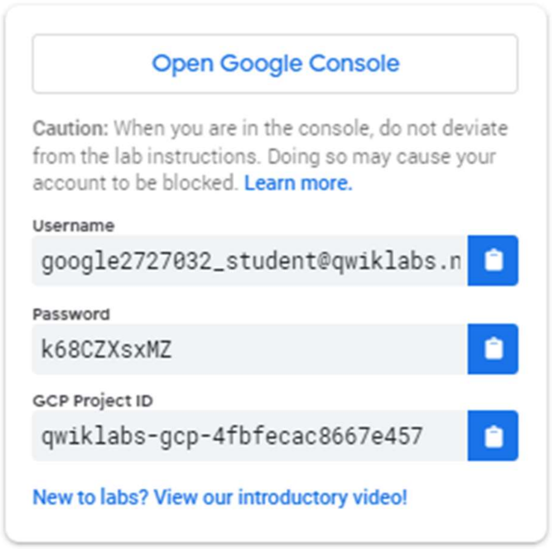
To complete this lab, you need:

- 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 Pixelbook, 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.



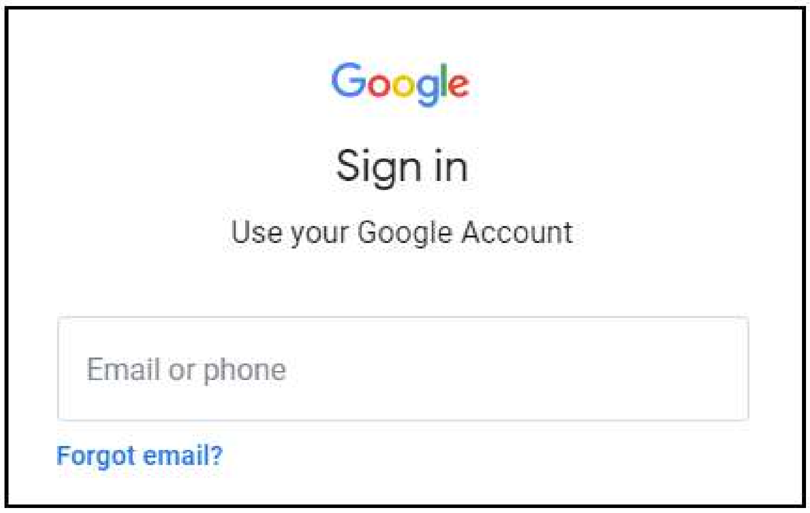
Caution: When you are in the console, do not deviate from the lab instructions. Doing so may cause your account to be blocked. Learn more.

Username
google2727032_student@qwiklabs.n

Password
k68CZXsxMZ

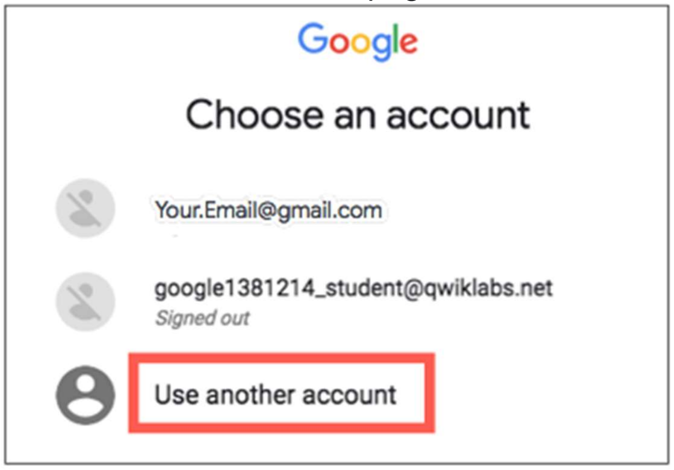
GCP Project ID
qwiklabs-gcp-4fbfecac8667e457

2. 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**.



3. In the **Sign in** page, paste the username that you copied from the Connection Details panel. Then copy and paste the password.

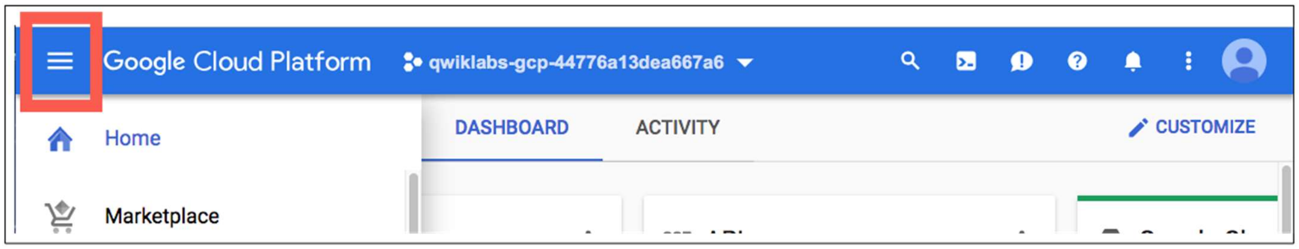
   Important: You must use the credentials from the Connection Details panel. Do not use your Qwiklabs credentials. If you have your own Google Cloud account, do not use it for this lab (avoids incurring charges).

4. 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.

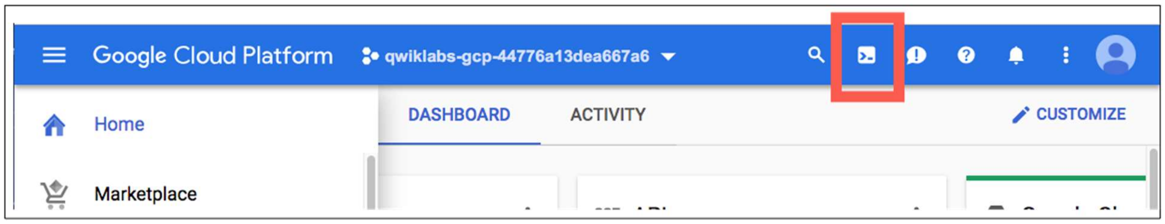
Note: You can view the menu with a list of Google Cloud Products and Services by clicking the **Navigation menu** at the top-
left.



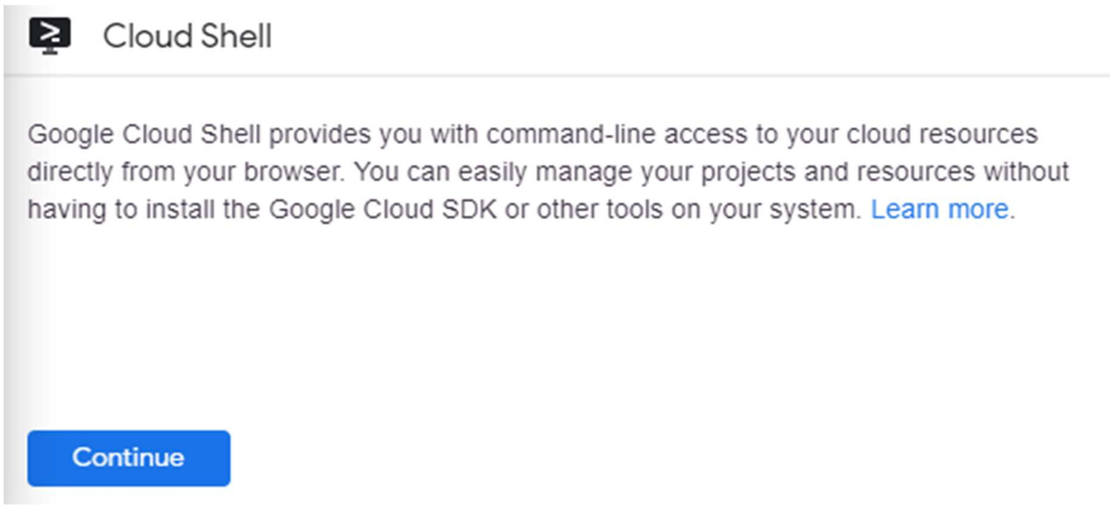
Activate Cloud Shell

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.



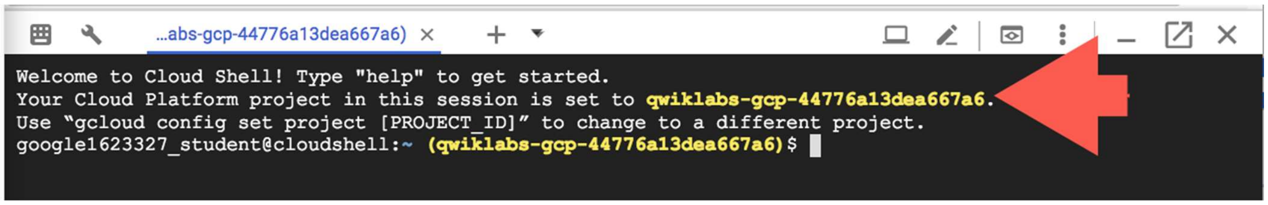
Click **Continue**.



Cloud Shell

Google Cloud Shell provides you with command-line access to your cloud resources directly from your browser. You can easily manage your projects and resources without having to install the Google Cloud SDK or other tools on your system. Learn more.

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:



```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to qwiklabs-gcp-44776a13dea667a6.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
google1623327_student@cloudshell:~ (qwiklabs-gcp-44776a13dea667a6)$
```

`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
```

(Output)

```
Credentialed accounts:
 - <myaccount>@<mydomain>.com (active)
```

(Example output)

```
Credentialed accounts:
 - google1623327_student@qwiklabs.net
```

You can list the project ID with this command:

```
gcloud config list project
```

(Output)

```
[core]
project = <project_ID>
```

(Example output)

```
[core]
project = qwiklabs-gcp-44776a13dea667a6
```

For full documentation of `gcloud` see the gcloud command-line tool overview.

Kubernetes Services

A service is a grouping of pods that are running on the cluster. Services are "cheap" and you can have many services within the cluster. Kubernetes services can efficiently power a microservice architecture.

Services provide important features that are standardized across the cluster: load-balancing, service discovery between applications, and features to support zero-downtime application deployments.

Each service has a pod label query which defines the pods which will process data for the service. This label query frequently matches pods created by one or more replication controllers. Powerful routing scenarios are possible by updating a service's label query via the Kubernetes API with deployment software.

Why Terraform?

While you could use `kubectl` or similar CLI-based tools mapped to API calls to manage all Kubernetes resources described in YAML files, orchestration with Terraform presents a few benefits:

- You can use the same configuration language to provision the Kubernetes infrastructure and to deploy applications into it.
- **Drift detection** - `terraform plan` will always present you the difference between reality at a given time and config you intend to apply.
- **Full lifecycle management** - Terraform doesn't just initially create resources, but offers a single command to create, update, and delete tracked resources without needing to inspect the API to identify those resources.
- **Synchronous feedback** - While asynchronous behavior is often useful, sometimes it's counter-productive as the job of identifying operation result (failures or details of created resource) is left to the user. e.g. you don't have IP/hostname of load balancer until it has finished provisioning, hence you can't create any DNS record pointing to it.
- Graph of relationships - Terraform understands relationships between resources which may help in scheduling - e.g. Terraform won't try to create a service in a Kubernetes cluster until the cluster exists.

Clone the sample code

1. In Cloud Shell, start by cloning the sample code:

```
gsutil -m cp -r gs://spls/gsp233/* .
```

2. Navigate to the `tf-gke-k8s-service-lb` directory:

```
cd tf-gke-k8s-service-lb
```

Understand the code

1. Review the contents of the `main.tf` file:

```
cat main.tf
```

Example Output (do not copy):

```
...

variable "region" {
  default = "us-west1"
}

variable "location" {
  default = "us-west1-b"
}

variable "network_name" {
  default = "tf-gke-k8s"
}

provider "google" {
  region = var.region
}

resource "google_compute_network" "default" {
  name                    = var.network_name
  auto_create_subnetworks = false
}

resource "google_compute_subnetwork" "default" {
  name                     = var.network_name
  ip_cidr_range            = "10.127.0.0/20"
  network                  = google_compute_network.default.self_link
  region                   = var.region
  private_ip_google_access = true
}
...
```

- Variables are defined for `region`, `zone`, and `network_name`. These will be used to create the Kubernetes cluster
- The Google Cloud provider will let us create resources in this project
- There are several resources defined to create the appropriate network and cluster
- At the end, there are some outputs which you'll see after running `terraform apply`

2. Review the contents of the `k8s.tf` file:

```
cat k8s.tf
```

Example Output (do not copy):

```
provider "kubernetes" {
  version = "~> 1.10.0"
  host    = google_container_cluster.default.endpoint
  token   = data.google_client_config.current.access_token
  client_certificate = base64decode(
    google_container_cluster.default.master_auth[0].client_certificate,
  )
  client_key = base64decode(google_container_cluster.default.master_auth[0].client_key)
  cluster_ca_certificate = base64decode(
    google_container_cluster.default.master_auth[0].cluster_ca_certificate,
  )
}

resource "kubernetes_namespace" "staging" {
  metadata {
    name = "staging"
  }
}

resource "google_compute_address" "default" {
  name   = var.network_name
  region = var.region
}

resource "kubernetes_service" "nginx" {
  metadata {
    namespace = kubernetes_namespace.staging.metadata[0].name
    name      = "nginx"
  }

  spec {
    selector = {
      run = "nginx"
    }

    session_affinity = "ClientIP"

    port {
      protocol    = "TCP"
      port        = 80
      target_port = 80
    }

    type             = "LoadBalancer"
    load_balancer_ip = google_compute_address.default.address
  }
}

resource "kubernetes_replication_controller" "nginx" {
  metadata {
    name      = "nginx"
    namespace = kubernetes_namespace.staging.metadata[0].name
    labels = {
      run = "nginx"
    }
  }

  spec {
    selector = {
      run = "nginx"
    }

    template {
      container {
        image = "nginx:latest"
        name  = "nginx"

        resources {
          limits {
            cpu    = "0.5"
            memory = "512Mi"
          }

          requests {
            cpu    = "250m"
            memory = "50Mi"
          }
        }
      }
    }
  }
}

output "load-balancer-ip" {
  value = google_compute_address.default.address
}
```

- The script configures a Kubernetes provider with Terraform and creates the service, namespace and a replication_controller resource.
- The script returns an `nginx` service IP as a output.

Initialize and install dependencies

The `terraform init` command is used to initialize a working directory containing the Terraform configuration files.

This command performs several different initialization steps in order to prepare a working directory for use and is always safe to run multiple times, to bring the working directory up to date with changes in the configuration:

1. Run `terraform init`:

```
terraform init
```

Example Output (do not copy):

```
...
* provider.google: version = "~> 3.8.0"
* provider.kubernetes: version = "~> 1.10.0"

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running `terraform plan` to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

2. Run the `terraform apply` command, which is used to apply the changes required to reach the desired state of the configuration.

```
terraform apply
```

3. Review Terraform's actions and inspect the resources which will be created.

4. When ready, type **yes** to begin Terraform actions.

On completion, you should see similar output:

Example Output (do not copy):

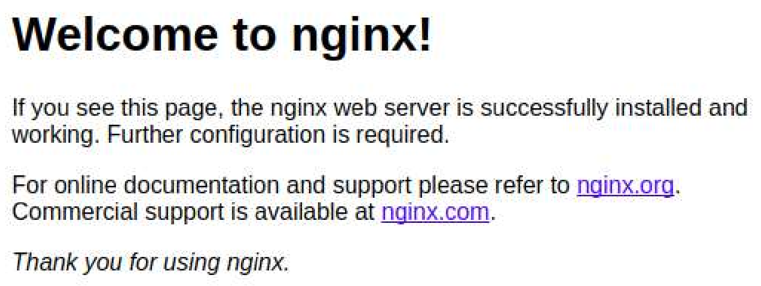
```
Apply complete! Resources: 7 added, 0 changed, 0 destroyed.

Outputs:

cluster_name = tf-gke-k8s
cluster_region = us-west1
cluster_zone = us-west1-b
load-balancer-ip = 35.233.177.223
network = https://www.googleapis.com/compute/beta/projects/qwiklabs-gcp-
5438ad3a5e852e4a/global/networks/tf-gke-k8s
subnetwork_name = tf-gke-k8s
```

Verify resources created by Terraform

1. In the console, navigate to **Navigation menu** > **Kubernetes Engine**.
2. Click on `tf-gke-k8s` cluster and check its configuration.
3. In the left panel, click **Services & Ingress** and check the `nginx` service status.
4. Click the **Endpoints** IP address to open the `Welcome to nginx!` page in a new browser tab.



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Click *Check my progress* to verify your performed task. If you have successfully deployed infrastructure with Terraform, you will see an assessment score.

Congratulations!

In this lab, you used Terraform to initialize, plan, and deploy a Kubernetes cluster along with a service.

Finish Your Quest

This self-paced lab is part of the Managing Cloud Infrastructure with Terraform and DevOps Essentials Quests. 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 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.

Take Your Next Lab

Continue your Quest with HTTPS Content-Based Load balancer with Terraform, or check out these suggestions:

- Using a NAT Gateway with Kubernetes Engine
- using Vault on Compute Engine for Secret Management

Next Steps / Learn More

- See how others are Terraform in the Community

Google Cloud Training & Certification

...helps you make the most of Google Cloud technologies. Our classes 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 help you validate and prove your skill and expertise in Google Cloud technologies.

Manual Last Updated January 15, 2021

Lab Last Tested January 15, 2021

Copyright 2021 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.