

Distributed Load Testing Using Kubernetes

GSP182



Google Cloud Self-Paced Labs

Overview

In this lab you will learn how to use Kubernetes Engine to deploy a distributed load testing framework. The framework uses multiple containers to create load testing traffic for a simple REST-based API. Although this solution tests a simple web application, the same pattern can be used to create more complex load testing scenarios such as gaming or Internet-of-Things (IoT) applications. This solution discusses the general architecture of a container-based load testing framework.

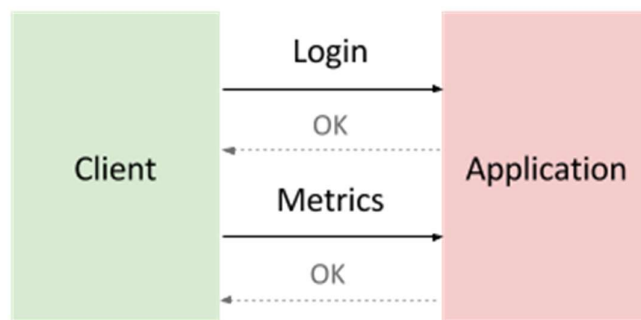
System under test

For this lab the system under test is a small web application deployed to Google App Engine. The application exposes basic REST-style endpoints to capture incoming HTTP POST requests (incoming data is not persisted).

Example workloads

The application that you'll deploy is modeled after the backend service component found in many Internet-of-Things (IoT) deployments. Devices first register with the service and then begin reporting metrics or sensor readings, while also periodically re-registering with the service.

Common backend service component interaction looks like



this:

To model this interaction, you'll use `Locust`, a distributed, Python-based load testing tool that is capable of distributing requests across multiple target paths. For example, Locust can distribute requests to the `/login` and `/metrics` target paths.

The workload is based on the interaction described above and is modeled as a set of Tasks in Locust. To approximate real-world clients, each Locust task is weighted. For example, registration happens once per thousand total client requests.

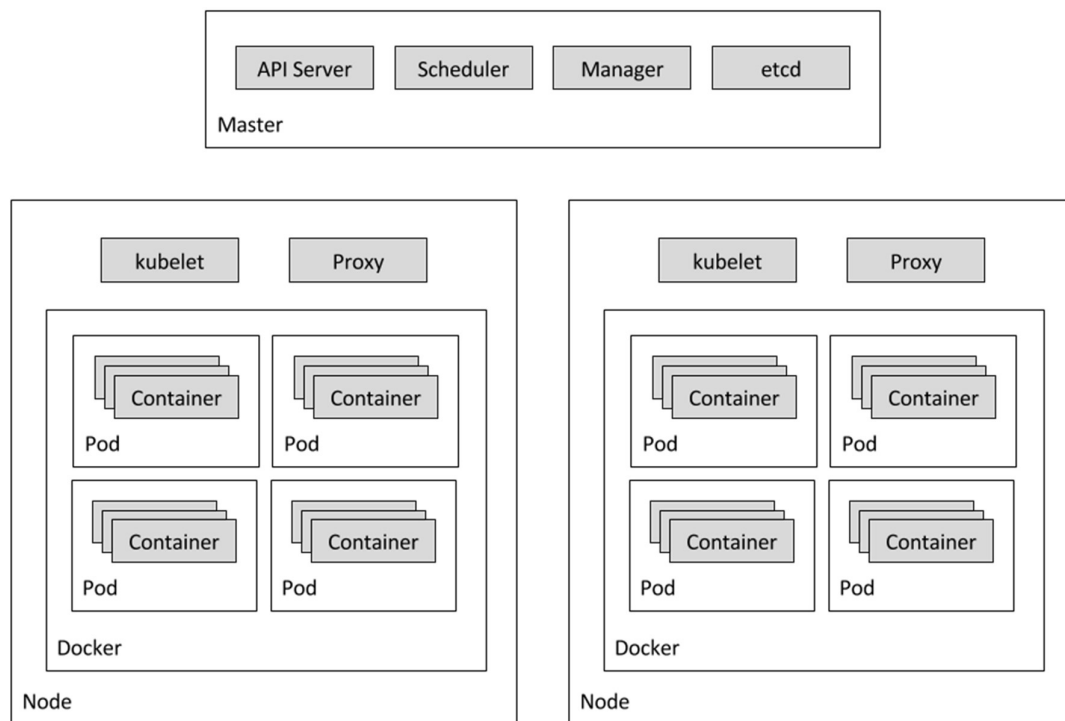
Container-based computing

- The Locust container image is a Docker image that contains the Locust software.
- A `container cluster` consists of at least one cluster master and multiple worker machines called nodes. These master and node machines run the Kubernetes cluster orchestration system. For more information about clusters, see the [Kubernetes Engine documentation](#)
- A `pod` is one or more containers deployed together on one host, and the smallest compute unit that can be defined, deployed, and managed. Some pods contain only a single container. For example, in this lab, each of the Locust containers runs in its own pod.
- A `Deployment controller` provides declarative updates for Pods and ReplicaSets. This lab has two deployments: one for `locust-master` and other for `locust-worker`.
- Services

A particular pod can disappear for a variety of reasons, including node failure or intentional node disruption for updates or maintenance. This means that the IP address of a pod does not provide a reliable interface for that pod. A more reliable approach would use an abstract representation of that interface that never changes, even if the underlying pod disappears and is replaced by a new pod with a different IP address. A Kubernetes Engine `service` provides this type of abstract interface by defining a logical set of pods and a policy for accessing them.

In this lab there are several services that represent pods or sets of pods. For example, there is a service for the DNS server pod, another service for the Locust master pod, and a service that represents all 10 Locust worker pods.

The following diagram shows the contents of the master and worker nodes:



What you'll do

- Create a system under test i.e. a small web application deployed to Google App Engine.
- Use Kubernetes Engine to deploy a distributed load testing framework.
- Create load testing traffic for a simple REST-based API.

Prerequisites

- Familiarity with App Engine and Kubernetes Engine Google Cloud services.
- Familiarity with standard Linux text editors such as Vim, Emacs or Nano

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 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:

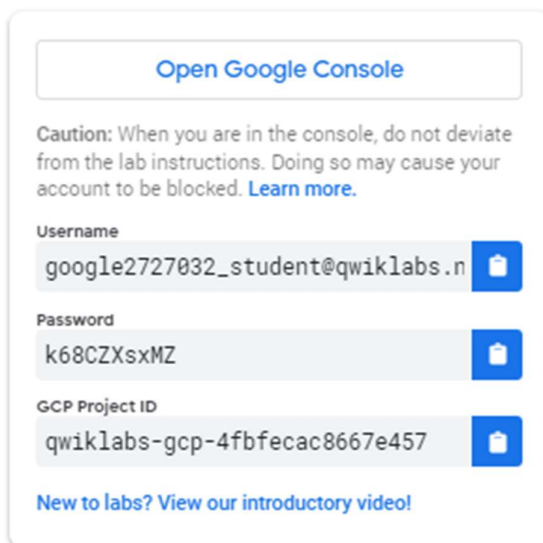
- Access to a standard internet browser (Chrome browser recommended).
- Time to complete the lab.

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

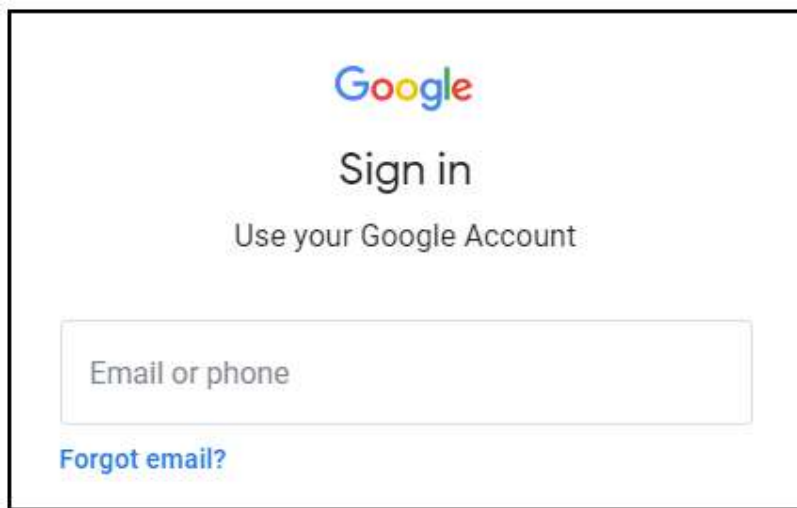
1. Click the **Start Lab** button. If you need to pay for the lab, a pop-up opens for you to select your payment method. On the left is a panel populated with the temporary credentials that you must use for this lab.



The screenshot shows a sign-in panel with the following elements:

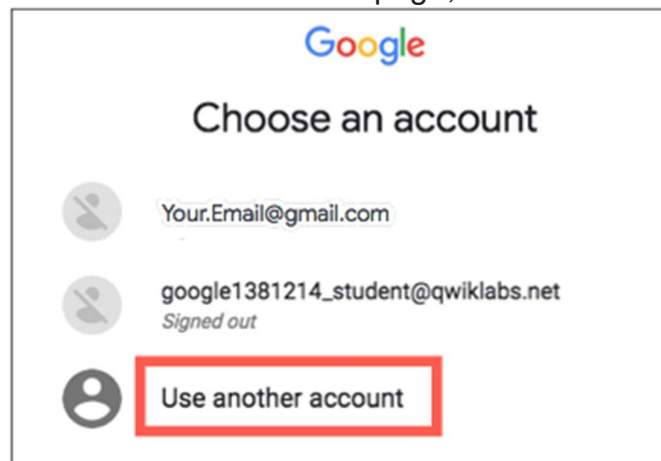
- A button at the top labeled "Open Google Console".
- A caution message: "Caution: When you are in the console, do not deviate from the lab instructions. Doing so may cause your account to be blocked. [Learn more.](#)"
- Three input fields, each with a copy icon to its right:
 - Username:** google2727032_student@qwiklabs.n
 - Password:** k68CZXsxMZ
 - GCP Project ID:** qwiklabs-gcp-4fbfecac8667e457
- A link at the bottom: "New to labs? [View our introductory video!](#)"

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).

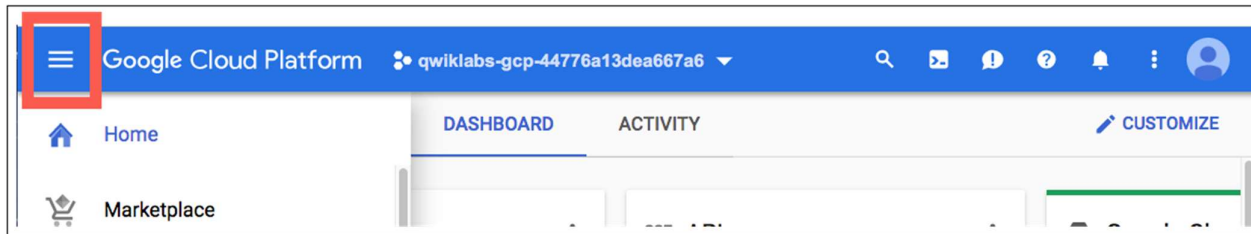
4. Click through the subsequent pages:

- 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-

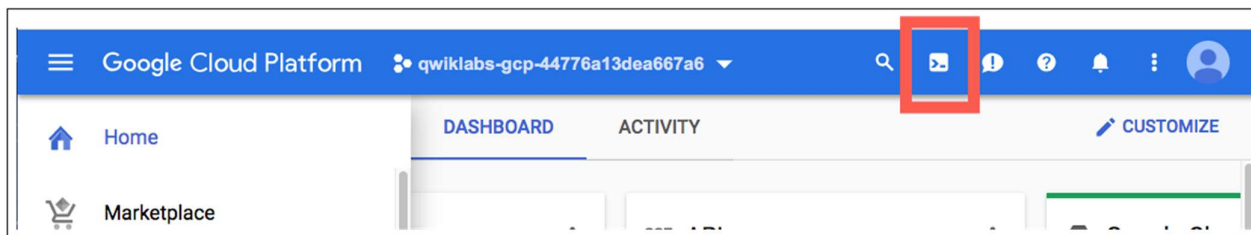
left.



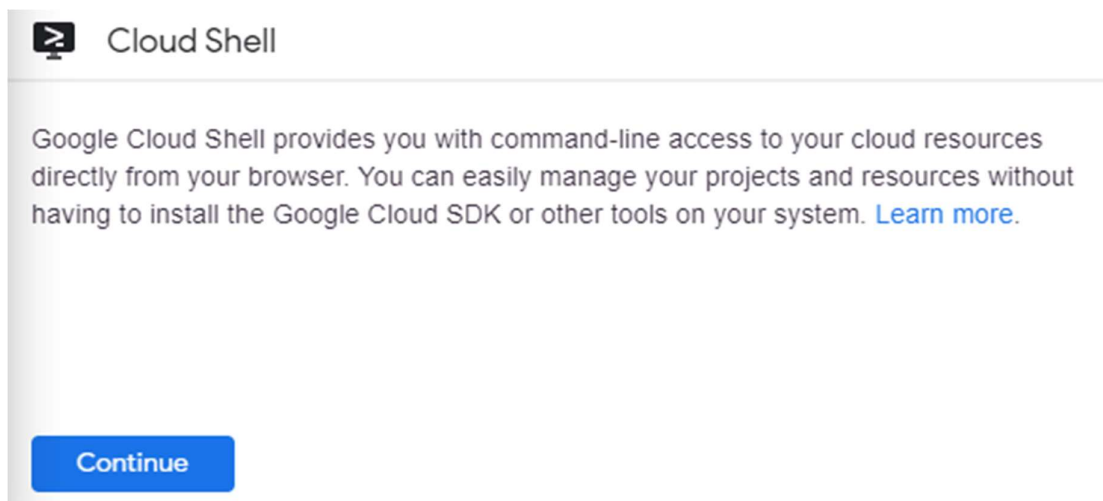
Activate Cloud Shell

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.

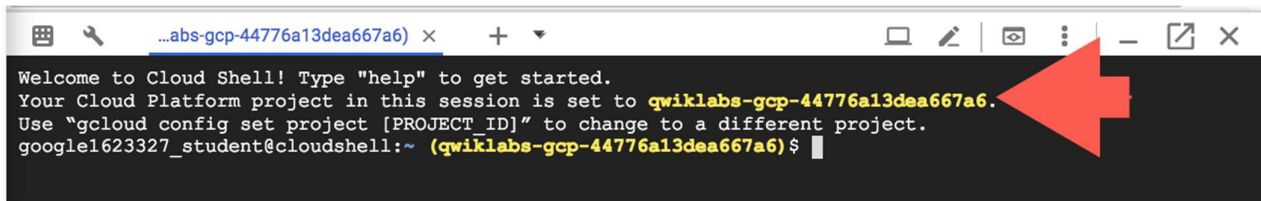
In the Cloud Console, in the top right toolbar, click the **Activate Cloud Shell** button.



Click **Continue**.



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:



```
...abs-gcp-44776a13dea667a6) x + ▾
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](#).

Set project and zone

Define environment variables for the `project id`, `region` and `zone` you want to use for the lab.

```
PROJECT=$(gcloud config get-value project)
REGION=us-central1
ZONE=${REGION}-a
CLUSTER=gke-load-test
TARGET=${PROJECT}.appspot.com
gcloud config set compute/region $REGION
gcloud config set compute/zone $ZONE
```


Get the sample code and build a Docker image for the application

Get the source code from the repository by running:

```
gsutil -m cp -r gs://spl/s/gsp182/distributed-load-testing-using-kubernetes .
```

Move into the directory:

```
cd distributed-load-testing-using-kubernetes/
```

Build docker image and store it in container registry.

```
gcloud builds submit --tag gcr.io/$PROJECT/locust-tasks:latest docker-image/.
```

Example Output:

ID	CREATE_TIME	DURATION	SOURCE
47f1b8f7-0b81-492c-aa3f-19b2b32e515d	xxxxxxx	51S	STATUS
gs://project_id_cloudbuild/source/1554261539.12-a7945015d56748e796c55f17b448e368.tgz			
gcr.io/project_id/locust-tasks (+1 more) SUCCESS			

Click *Check my progress* to verify the objective.

Deploy Web Application

The `sample-webapp` folder contains a simple Google App Engine Python application as the "system under test". To deploy the application to your project use the `gcloud app deploy` command:

```
gcloud app deploy sample-webapp/app.yaml
```

Note: You will need the URL of the deployed sample web application when deploying the `locust-master` and `locust-worker` deployments which is already stored in `TARGET` variable.

Click *Check my progress* to verify the objective.

Deploy Kubernetes Cluster

First create the [Google Kubernetes Engine](#) cluster using the `gcloud` command shown below:

```
gcloud container clusters create $CLUSTER \  
  --zone $ZONE \  
  --num-nodes=5
```

Example Output:

NAME	LOCATION	MASTER_VERSION	MASTER_IP	MACHINE_TYPE
gke-load-test	us-central1-a	1.11.7-gke.12	34.66.156.246	n1-standard-1
gke.12 5		RUNNING		1.11.7-

Click *Check my progress* to verify the objective.

Load testing master

The first component of the deployment is the Locust master, which is the entry point for executing the load testing tasks described above. The Locust master is deployed with a single replica because we need only one master.

The configuration for the master deployment specifies several elements, including the ports that need to be exposed by the container (8089 for web interface, 5557 and 5558 for communicating with workers). This information is later used to configure the Locust workers.

The following snippet contains the configuration for the ports:

```
ports:  
  - name: loc-master-web  
    containerPort: 8089  
    protocol: TCP  
  - name: loc-master-p1  
    containerPort: 5557  
    protocol: TCP  
  - name: loc-master-p2  
    containerPort: 5558  
    protocol: TCP
```

Deploy locust-master

Replace `[TARGET_HOST]` and `[PROJECT_ID]` in `locust-master-controller.yaml` and `locust-worker-controller.yaml` with the deployed endpoint and project-id respectively.

```
sed -i -e "s/[TARGET_HOST]/$TARGET/g" kubernetes-config/locust-master-controller.yaml
sed -i -e "s/[TARGET_HOST]/$TARGET/g" kubernetes-config/locust-worker-controller.yaml
sed -i -e "s/[PROJECT_ID]/$PROJECT/g" kubernetes-config/locust-master-controller.yaml
sed -i -e "s/[PROJECT_ID]/$PROJECT/g" kubernetes-config/locust-worker-controller.yaml
```

Deploy Locust master:

```
kubectl apply -f kubernetes-config/locust-master-controller.yaml
```

To confirm that the `locust-master` pod is created, run the following command:

```
kubectl get pods -l app=locust-master
```

Next, deploy the `locust-master-service`:

```
kubectl apply -f kubernetes-config/locust-master-service.yaml
```

This step will expose the pod with an internal DNS name (`locust-master`) and ports 8089, 5557, and 5558. As part of this step, the `type: LoadBalancer` directive in `locust-master-service.yaml` will tell Google Kubernetes Engine to create a Compute Engine forwarding-rule from a publicly available IP address to the `locust-master` pod.

To view the newly created forwarding-rule, execute the following:

```
kubectl get svc locust-master
```

Example Output:

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)
locust-master	LoadBalancer	10.59.244.88	35.222.161.198	8089:30865/TCP,5557:30707/TCP,5558:31327/TCP
AGE			1m	

Click *Check my progress* to verify the objective.

Load testing workers

The next component of the deployment includes the Locust workers, which execute the load testing tasks described above. The Locust workers are deployed by a single deployment that creates multiple pods. The pods are spread out across the Kubernetes cluster. Each pod uses environment variables to control important configuration information such as the hostname of the system under test and the hostname of the Locust master.

After the Locust workers are deployed, you can return to the Locust master web interface and see that the number of slaves corresponds to the number of deployed workers.

The following snippet contains the deployment configuration for the name, labels, and number of replicas:

```
apiVersion: "apps/v1"
kind: "Deployment"
metadata:
  name: locust-worker
  labels:
    name: locust-worker
spec:
  replicas: 5
  selector:
    matchLabels:
      app: locust-worker
  template:
    metadata:
      labels:
        app: locust-worker
    spec:
  ...
```

Deploy locust-worker

Now deploy `locust-worker-controller`:

```
kubectl apply -f kubernetes-config/locust-worker-controller.yaml
```

The `locust-worker-controller` is set to deploy 5 `locust-worker` pods. To confirm they were deployed run the following:

```
kubectl get pods -l app=locust-worker
```

Scaling up the number of simulated users will require an increase in the number of Locust worker pods. To increase the number of pods deployed by the deployment, Kubernetes offers the ability to resize deployments without redeploying them.

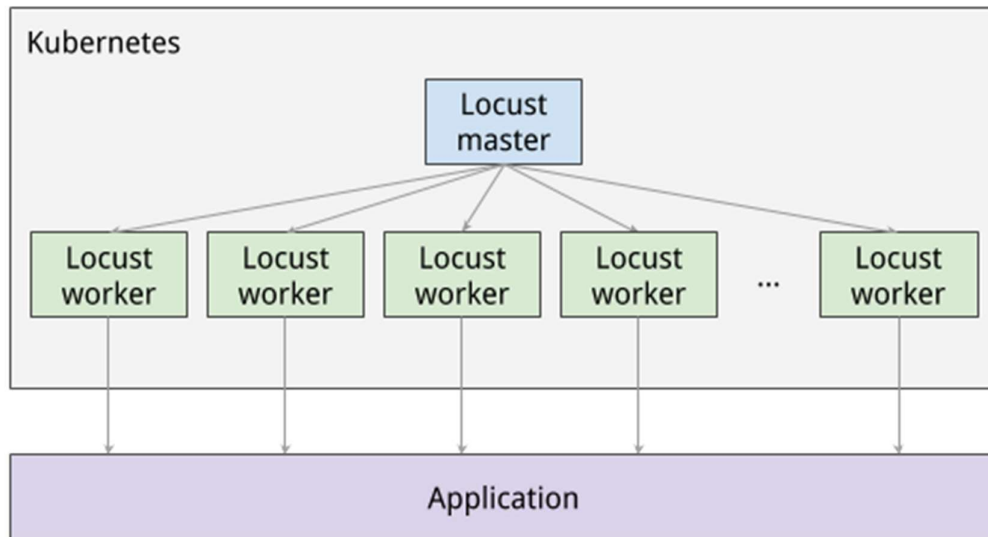
The following command scales the pool of Locust worker pods to 20:

```
kubectl scale deployment/locust-worker --replicas=20
```

To confirm that pods have launched and are ready, get the list of `locust-worker` pods:

```
kubectl get pods -l app=locust-worker
```

The following diagram shows the relationship between the Locust master and the Locust workers:



Click *Check my progress* to verify the objective.

Execute Tests

To execute the Locust tests, get the external IP address by following command:

```
EXTERNAL_IP=$(kubectl get svc locust-master -o yaml | grep ip | awk -F": " '{print $NF}')
```

```
echo http://$EXTERNAL_IP:8089
```

Click the link and navigate to Locust master web interface.

The Locust master web interface enables you to execute the load testing tasks against the system under test, as shown in the following sample image:

Start new Locust swarm

Number of users to simulate

Hatch rate (users spawned/second)

Start swarming

To begin, specify the total number of users to simulate and a rate at which each user should be spawned. Next, click Start swarming to begin the simulation. For example you can specify number of users as 300 and rate as 10.

Click **Start swarming**.

[Statistics](#) [Failures](#) [Exceptions](#)

Type	Name	# requests	# fails	Median	Average	Min	Max	Content Size	# reqs/sec
POST	/login	31	0	12	15	8	73	54	0.2
POST	/metrics	30362	0	12	16	7	529	95	147.9
Total		30393	0	12	16	7	529	94	148.1

[Download request statistics CSV](#)
[Download response time distribution CSV](#)

As time progress and users are spawned, you will see statistics begin to aggregate for simulation metrics, such as the number of requests and requests per second.

To stop the simulation, click **Stop** and the test will terminate. The complete results can be downloaded into a spreadsheet.

Congratulations!



Finish Your Quest

This self-paced lab is part of the [Google Cloud Solutions I: Scaling Your Infrastructure](#) and [Kubernetes Solutions](#) 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 (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](#).

Take Your Next Lab

Continue your Quest with the next lab, or check out these suggestions:

- [Continuous Delivery with Jenkins in Kubernetes Engine](#)
- [Running Dedicated Game Servers in Google Kubernetes Engine](#)

Next Steps / Learn More

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 October 15, 2020

Lab Last Tested October 15, 2020

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.