**Manage Kubernetes in Google Cloud: Challenge Lab**

experimentLabschedule1 hour 30 minutesuniversal\_currency\_alt5 Creditsshow\_chartIntermediate

Rate Lab

infoThis lab may incorporate AI tools to support your learning.

**GSP510**



**Introduction**

In a challenge lab you’re given a scenario and a set of tasks. Instead of following step-by-step instructions, you will use the skills learned from the labs in the course to figure out how to complete the tasks on your own! An automated scoring system (shown on this page) will provide feedback on whether you have completed your tasks correctly.

When you take a challenge lab, you will not be taught new Google Cloud concepts. You are expected to extend your learned skills, like changing default values and reading and researching error messages to fix your own mistakes.

To score 100% you must successfully complete all tasks within the time period!

This lab is recommended for students who have enrolled in the [Manage Kubernetes in Google Cloud](https://www.cloudskillsboost.google/authoring/course_templates/783) skill badge. Are you ready for the challenge?

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.

To complete this lab, you need:

* Access to a standard internet browser (Chrome browser recommended).

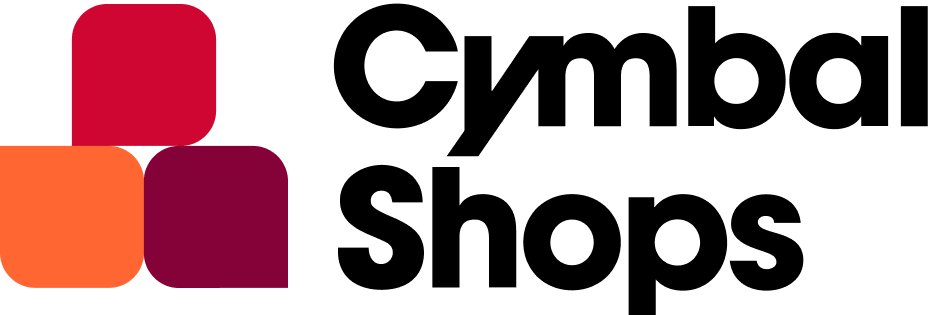
**Note:** Use an Incognito or private browser window to run this lab. This prevents any conflicts between your personal account and the Student account, which may cause extra charges incurred to your personal account.

* Time to complete the lab---remember, once you start, you cannot pause a lab.

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

**Challenge scenario**

You were onboarded at Cymbal Shops just a few months ago. You have spent a lot of time working with containers in Docker and Artifact Registry and have learned the ropes of managing new and existing deployments on GKE. You've had practice updating manifests as well as scaling, monitoring, and debugging applications running on your clusters.



Your team would like you to start managing their Kubernetes deployments to ensure smooth rollouts and application updates to their new e-commerce website. Before you switch to this new role, the development team would like you to demonstrate your new skills. As part of this demonstration, they have a list of tasks they would like to see you do in an allotted period of time in a sandbox environment.

Your challenge

As part of the sandbox environment, your developers have created an Artifact Registry repository named demo-repo that has some code with a basic example application that you will deploy onto a cluster.

**Note:** This image created in the repo is a containerized version of the code you will be downloading from the spls/gsp510/hello-app bucket later in the lab.

Your tasks will include the following:

* Creating a GKE cluster based on a set of configurations provided.
* Enabling Managed Prometheus on the cluster for metrics monitoring.
* Deploying a Kubernetes manifest onto the cluster, and debugging the errors.
* Creating a logs-based metric and alerting policy for the errors on the Kubernetes cluster.
* Fixing the manifest errors, containerizing your application code and pushing it to Artifact Registry using Docker.
* Exposing a service for your application on the cluster and verifying your updates.

**Task 1. Create a GKE cluster**

The new Cymbal Shops e-commerce website will run natively on GKE and the team would like to see your experience working and setting up clusters. In this section, you will need to create a Kubernetes cluster based on a set of configurations provided to run your demo applications on.

1. Create a GKE cluster named hello-world-ewk0 with the following configuration:

|  |  |
| --- | --- |
| **Setting** | **Value** |
| **Zone** | us-central1-c |
| **Release channel** | **Regular** |
| **Cluster version** | 1.27.8 *or newer* |
| **Cluster autoscaler** | **Enabled** |
| **Number of nodes** | **3** |
| **Minimum nodes** | **2** |
| **Maximum nodes** | **6** |

Click **Check my progress** to verify the objective.

Assessment Completed!

Create a GKE cluster

Check my progress

*Assessment Completed!*

**Task 2. Enable Managed Prometheus on the GKE cluster**

As part of the e-commerce website strategy, Cymbal Shops would like to start using Managed Prometheus for metrics and monitoring on the cluster to ensure a good experience for their customers. In this section, you will enable Managed Prometheus on the cluster for metric monitoring and create a namespace to deploy a sample Prometheus application and pod monitoring on.

1. Enable the Prometheus managed collection on the GKE cluster.
2. Create a namespace on the cluster named gmp-8uj9.
3. Download a sample Prometheus app:

gsutil cp gs://spls/gsp510/prometheus-app.yaml .

Copied!

content\_copy

1. Update the <todo> sections (lines 35-38) with the following configuration.
   * **containers.image**: nilebox/prometheus-example-app:latest
   * **containers.name:** prometheus-test
   * **ports.name**: metrics
2. Deploy the application onto the gmp-8uj9 namespace on your GKE cluster.
3. Download the pod-monitoring.yaml file:

gsutil cp gs://spls/gsp510/pod-monitoring.yaml .

Copied!

content\_copy

1. Update the <todo> sections (lines 18-24) with the following configuration:
   * **metadata.name**: prometheus-test
   * **labels.app.kubernetes.io/name**: prometheus-test
   * **matchLabels.app**: prometheus-test
   * **endpoints.interval**: 50s
2. Apply the pod monitoring resource onto the gmp-8uj9 namespace on your GKE cluster.

Click **Check my progress** to verify the objective.

Assessment Completed!

Enable Managed Prometheus on the GKE cluster

Check my progress

*Assessment Completed!*

**Task 3. Deploy an application onto the GKE cluster**

The development team at Cymbal Shops will be consistently releasing new application code to the cluster that you will have to successfully deploy into production. In this section, you will deploy a Kubernetes manifest onto the cluster and inspect the issue.

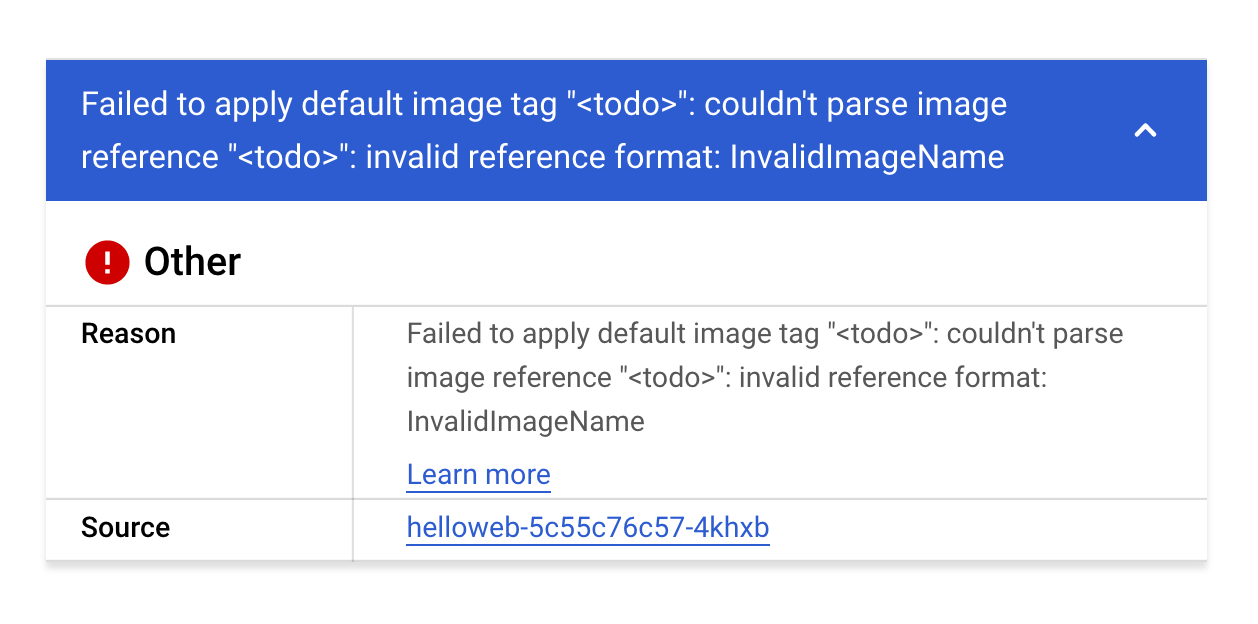
1. Download the demo deployment manifest files:

gsutil cp -r gs://spls/gsp510/hello-app/ .

Copied!

content\_copy

1. Create a deployment onto the gmp-8uj9 namespace on your GKE cluster from the helloweb-deployment.yaml manifest file. It is located in the hello-app/manifests folder.
2. Verify you have created the deployment, and navigate to the **helloweb** deployment details page. You should see the following error:



This error seems to stem from an invalid image name in the manifest that you just deployed. Before you fix the image name, you will create a logs-based metric and alerting policy so that your team can be notified if this happens again in the future.

Click **Check my progress** to verify the objective.

Assessment Completed!

Deploy an application onto the GKE cluster

Check my progress

*Assessment Completed!*

**Task 4. Create a logs-based metric and alerting policy**

Cymbal Shops would like to set up some logs based metrics and alerting policies to aggregate the number of errors and warnings in their Kubernetes pods and set up an alerting mechanism for their clusters when a certain amount of errors are above a specific threshold. In this section, you will demonstrate your knowledge on building these metrics and alerting policies for the team.

Create a logs-based metric

1. In the Logs Explorer, create a query that exposes warnings/errors you saw in the previous section on the cluster.

**Hint:** your query should have just one *Resource Type* and one *Severity* selected.

If the query is correct, upon running you should see the following errors show up in the logs:

Error: InvalidImageName

Failed to apply default image tag "<todo>": couldn't parse image reference "<todo>": invalid reference format

1. Create a logs-based metric from this query. For **Metric type**, use **Counter** and for the **Log Metric Name** use pod-image-errors.

Create an alerting policy

1. Create an Alerting Policy based on the logs-based metric you just created. Use the following details to configure your policy:
   * **Rolling Window**: 10 min
   * **Rolling window function**: Count
   * **Time series aggregation**: Sum
   * **Condition type**: Threshold
   * **Alert trigger**: Any time series violates
   * **Threshold position**: Above threshold
   * **Threshold value**: 0
   * **Use notification channel**: ***Disable***
   * **Alert policy name**: Pod Error Alert

Click **Check my progress** to verify the objective.

Assessment Completed!

Create a logs-based metric and alerting policy

Check my progress

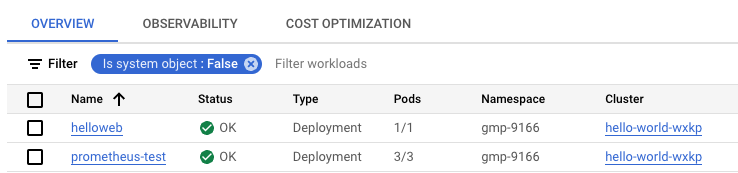
*Assessment Completed!*

**Task 5. Update and re-deploy your app**

The development team would like to see you demonstrate your knowledge on deleting and updating deployments on the cluster in case of an error. In this section, you will update a Kubernetes manifest with a correct image reference, delete a deployment, and deploy the updated application onto the cluster.

1. Replace the <todo> in the image section in the helloweb-deployment.yaml deployment manifest with the following image:
   * us-docker.pkg.dev/google-samples/containers/gke/hello-app:1.0
2. **Delete** the **helloweb** deployment from your cluster.
3. Deploy the updated helloweb-deployment.yaml manifest onto your cluster on the gmp-8uj9 namespace.

You should verify that it has deployed correctly with no errors. Your Kubernetes Workloads page should resemble the following:



Click **Check my progress** to verify the objective.

Assessment Completed!

Update and re-deploy your app

Check my progress

*Assessment Completed!*

**Task 6. Containerize your code and deploy it onto the cluster**

Lastly, as part of the Cymbal Shops e-commerce strategy, the application team will be providing you code that you will need to containerize and store in a registry, and then update the cluster with the newest version of that code.

In this section, you will containerize your application code, update an image in Artifact Registry, and set that to the image on your cluster. Your team has a repository in Artifact Registry named demo-repo that contains a containerized version of the hello-app sample app in Docker. You will update the code for the build locally, then push a new version to the repository.

1. In the hello-app directory, update the **main.go** file to use Version: 2.0.0 on line 49.
2. Use the hello-app/Dockerfile to create a Docker image with the v2 tag.

**Note:** you should follow the Artifact Registry naming conventions as detailed [here](https://cloud.google.com/artifact-registry/docs/docker/names#containers).

1. Push the newly built Docker image to your repository in Artifact Registry using the v2 tag.
2. Set the image on your **helloweb** deployment to reflect the v2 image you pushed to Artifact Registry.
3. Expose the **helloweb** deployment to a LoadBalancer service named helloweb-service-thhn on port 8080, and set the target port of the container to the one specified in the Dockerfile.
4. Navigate to the external load balancer IP address of the helloweb-service-thhn service, and you should see the following text returned by the service:

Hello, world!

Version: 2.0.0

Hostname: helloweb-6fc7476576-cvv5f

**Note:** It may take a few minutes for the webpage to load.

Click **Check my progress** to verify the objective.

Assessment Completed!

Containerize your code and deploy it onto the cluster

Check my progress

*Assessment Completed!*

**Congratulations!**

Congratulations! In this lab you verified your skills on creating a GKE cluster based on a set of configurations and enabling the Prometheus managed collection. You also troubleshooted errors on a cluster and created a logs-based metric and alerting policy, and then updated Kubernetes manifests to fix the issue. Lastly, you containerized your application code and pushed a new build to a repository in Artifact Registry using Docker, updated your cluster to reflect the new image, and exposed a service on the cluster. You are ready to start managing Kubernetes in your own environments!

Earn your next skill badge

This self-paced lab is part of the [Manage Kubernetes in Google Cloud](https://www.cloudskillsboost.google/quests/257) skill badge quest. Completing this skill badge quest earns you the badge above, to recognize your achievement. Share your badge on your resume and social platforms, and announce your accomplishment using #GoogleCloudBadge.

This skill badge quest is part of Google's Cloud [Hybrid and Multi-Cloud Cloud Architect](https://cloud.google.com/training/kubernetes-anthos-hybridcloud) learning path. If you have already completed the other skill badge quests in this learning path, search the Google Cloud Skills Boost catalog for 20+ other skill badge quests in which you can enroll.

Google Cloud training and 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 Aug 11, 2023**

**Lab Last Tested Aug 11, 2023**

Copyright 2024 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.