**Develop your Google Cloud Network: Challenge Lab Solution**

[[](https://medium.com/@itxcrusher?source=post_page-----ec847a46a19a--------------------------------)](https://medium.com/@itxcrusher?source=post_page-----ec847a46a19a--------------------------------)

[Muhammad Hassaan Javed](https://medium.com/@itxcrusher?source=post_page-----ec847a46a19a--------------------------------)

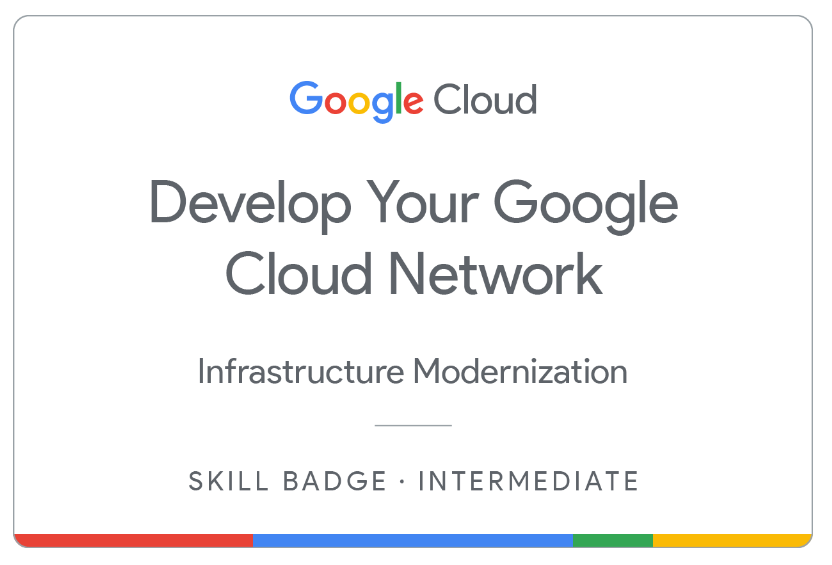
·

[Follow](https://medium.com/m/signin?actionUrl=https%3A%2F%2Fmedium.com%2F_%2Fsubscribe%2Fuser%2Ff869e52969e8&operation=register&redirect=https%3A%2F%2Fmedium.com%2F%40itxcrusher%2Fdevelop-your-google-cloud-network-challenge-lab-solution-ec847a46a19a&user=Muhammad+Hassaan+Javed&userId=f869e52969e8&source=post_page-f869e52969e8----ec847a46a19a---------------------post_header-----------)

6 min read

·

Jun 2, 2024



Develop your Google Cloud Network Badge

**Introduction**

Welcome to the “Develop Your Google Cloud Network” challenge lab. This lab tests your proficiency in Google Cloud by giving you a real-world scenario with tasks to complete. Unlike guided labs, here you rely on your skills and knowledge to complete the tasks. An automated scoring system will provide feedback on your progress.

To excel, you need to complete all tasks within the given timeframe. This lab is recommended for those who have completed the “Develop Your Google Cloud Network” skill badge. Are you ready for the challenge?

**Setup Instructions**

Before starting, ensure you:

* Use an Incognito or private browser window to avoid conflicts with personal Google Cloud accounts.
* Have access to a standard internet browser (Chrome recommended).
* Have sufficient time as the lab is timed and cannot be paused once started.

**Challenge Scenario**

As a cloud engineer at Jooli Inc., you are tasked with helping the Griffin team set up their environment. They need a development and production VPC, a bastion host, a Cloud SQL instance, and a Kubernetes cluster for a WordPress environment. You will also set up monitoring and grant access to an additional engineer.

**Task Overview**

1. **Create Development VPC**
2. **Create Production VPC**
3. **Create Bastion Host**
4. **Create and Configure Cloud SQL Instance**
5. **Create Kubernetes Cluster**
6. **Prepare the Kubernetes Cluster**
7. **Create a WordPress Deployment**
8. **Enable Monitoring**
9. **Provide Access for an Additional Engineer**

**Task 0: Initial Steps**

**Access GCP Console and Terminal**

Start the Lab by pressing the Start Lab button  
Copy username and password from the sidebar  
Press the Open Google Cloud Console button  
Enter username and password to sign in  
Press I understand button  
Check Agreement and press Agree and Continue  
Press the activate cloud shell button on the top right corner  
In the cloud shell, click continue, and authorize if a popup appears

**Export the following variables**

Make sure to replace [content] with the values of your lab!

export REGION=[your\_lab\_region]  
export ZONE=[your\_lab\_zone]  
export ADDITIONAL\_ENGINEER\_EMAIL=[your\_lab\_username2]

**Task 01: Create development VPC**

gcloud compute networks create griffin-dev-vpc --subnet-mode=custom  
  
gcloud compute networks subnets create griffin-dev-wp \  
 --network=griffin-dev-vpc \  
 --range=192.168.16.0/20 \  
 --region=$REGION  
  
gcloud compute networks subnets create griffin-dev-mgmt \  
 --network=griffin-dev-vpc \  
 --range=192.168.32.0/20 \  
 --region=$REGION

**Alternative Approach: Using Google Cloud Console:**

* Navigate to the **VPC network**.
* Click **Create VPC network**.
* Name it griffin-dev-vpc.
* Add subnets griffin-dev-wp (192.168.16.0/20) and griffin-dev-mgmt (192.168.32.0/20).
* Click **Create**.

**Task 02: Create production VPC**

gcloud compute networks create griffin-prod-vpc --subnet-mode=custom  
  
gcloud compute networks subnets create griffin-prod-wp \  
 --network=griffin-prod-vpc \  
 --range=192.168.48.0/20 \  
 --region=$REGION  
  
gcloud compute networks subnets create griffin-prod-mgmt \  
 --network=griffin-prod-vpc \  
 --range=192.168.64.0/20 \  
 --region=$REGION

**Alternative Approach: Using Google Cloud Console:**

* Navigate to the **VPC network**.
* Click **Create VPC network**.
* Name it griffin-prod-vpc.
* Add subnets griffin-prod-wp (192.168.48.0/20) and griffin-prod-mgmt (192.168.64.0/20).
* Click **Create**.

**Task 3: Create a bastion host**

**Bastion host:**

gcloud compute instances create griffin-bastion \  
 --machine-type=e2-medium \  
 --zone=$ZONE \  
 --tags=bastion \  
 --network-interface=subnet=griffin-dev-mgmt \  
 --network-interface=subnet=griffin-prod-mgmt \  
 --metadata=startup-script='#! /bin/bash  
 sudo apt-get update  
 sudo apt-get install -yq git htop  
 ' \  
 --scopes=cloud-platform \  
 --image-family=debian-10 \  
 --image-project=debian-cloud

**Alternative Approach: Using Google Cloud Console:**

* Navigate to **VM instances**.
* Click **Create instance**.
* Name it griffin-bastion.
* Select e2-medium as the machine type.
* Under **Networking**, add two network interfaces:
* The first interface connected to griffin-dev-mgmt
* The second interface connected to griffin-prod-mgmt
* Allow SSH connections.
* Click **Create**.

**Firewall rules allowing TCP traffic on port 22:**

gcloud compute firewall-rules create griffin-dev-allow-ssh \  
 --network=griffin-dev-vpc \  
 --allow=tcp:22 \  
 --source-ranges=0.0.0.0/0 \  
 --target-tags=bastion \  
 --description="Allow SSH access to bastion host"

gcloud compute firewall-rules create griffin-prod-allow-ssh \  
 --network=griffin-prod-vpc \  
 --allow=tcp:22 \  
 --source-ranges=0.0.0.0/0 \  
 --target-tags=bastion \  
 --description="Allow SSH access to bastion host in production"

**Alternative Approach: Using Google Cloud Console:**

* Navigate to **Firewall**.
* Click **Create a Firewall rule**.
* Name it griffin-dev-allow-ssh, set the network to griffin-dev-vpc, and allow tcp:22 from 0.0.0.0/0.
* Repeat for griffin-prod-allow-ssh in griffin-prod-vpc.

**Task 04: Create and Configure Cloud SQL Instance**

gcloud sql instances create griffin-dev-db \  
 --database-version=MYSQL\_5\_7 \  
 --tier=db-n1-standard-1 \  
 --region=$REGION  
  
gcloud sql databases create wordpress --instance=griffin-dev-db  
  
gcloud sql users create wp\_user --host=% --instance=griffin-dev-db --password=password123  
  
gcloud sql connect griffin-dev-db --user=root << EOF  
CREATE DATABASE wordpress;  
CREATE USER 'wp\_user'@'%' IDENTIFIED BY 'stormwind\_rules';  
GRANT ALL PRIVILEGES ON wordpress.\* TO 'wp\_user'@'%';  
FLUSH PRIVILEGES;  
EOF

Enter password: password123 (You can change it in the above command)

**Alternative Approach: Using Google Cloud Console:**

* Navigate to **SQL**.
* Click **Create instance**.
* Select **MySQL** and configure the instance with the name griffin-dev-db.
* Choose the appropriate region and settings.
* Create the wordpress database and wp\_user user with necessary privileges using the SQL command interface.

**Task 05: Create Kubernetes cluster**

gcloud container clusters create griffin-dev \  
 --zone=$ZONE \  
 --num-nodes=2 \  
 --machine-type=e2-standard-4 \  
 --network=griffin-dev-vpc \  
 --subnetwork=griffin-dev-wp

**Alternative Approach: Using Google Cloud Console:**

* Navigate to **Kubernetes Engine**.
* Click **Create Cluster**.
* Select **Standard Cluster**.
* Name it griffin-dev.
* Set the **Node Pools**:
* Machine type: e2-standard-4
* Number of nodes: 2
* Set the **Network** to griffin-dev-vpc and **Subnetwork** to griffin-dev-wp.
* Click **Create**.

**Task 06: Prepare the Kubernetes cluster**

**Copy configuration files:**

gsutil cp -r gs://cloud-training/gsp321/wp-k8s .

**Change to the directory and list the files:**

cd wp-k8s  
ls

**Update the wp-env.yaml file**

nano wp-env.yaml

Update the values of username to wp\_user and password to stormwind\_rules

**Setup secrets and volumes**

gcloud iam service-accounts keys create key.json \  
 --iam-account=cloud-sql-proxy@$GOOGLE\_CLOUD\_PROJECT.iam.gserviceaccount.com  
kubectl create secret generic cloudsql-instance-credentials \  
 --from-file key.json

**Alternative Approach: Using Google Cloud Console:**

* **Copying Files:**
* Navigate to **Cloud Shell** and use the gsutil command to copy files.
* **Editing Files:**
* Use **Cloud Shell Editor** or other preferred text editor to modify wp-env.yaml.

**Task 07: Create WordPress deployment**

**Retrieve and copy the connection name:**

gcloud sql instances describe griffin-dev-db --format='value(connectionName)'

**Edit the wp-deployment.yaml file**

nano wp-deployment.yaml

Find and replace the placeholder YOUR\_SQL\_INSTANCE with the instance connection name retrieved in the previous step. Now Save the changes and exit the text editor.

**Verify the changes:**

cat wp-deployment.yaml

**Deploy WordPress:**

kubectl apply -f wp-env.yaml  
kubectl apply -f wp-deployment.yaml  
kubectl apply -f wp-service.yaml

**Alternative Approach: Using Google Cloud Console:**

* **Editing Files:**
* Use **Cloud Shell Editor** to modify and verify the wp-deployment.yaml.
* **Deploying WordPress:**
* Use **Cloud Shell** to run the kubectl commands for deploying WordPress.

**Task 08: Enable Monitoring**

**List the services in the Kubernetes cluster:**

kubectl get services

Look for the wordpress service of type LoadBalancer. The EXTERNAL-IP column will contain the IP address you use as the WordPress site URL.

export WORDPRESS\_SITE\_URL=[EXTERNAL\_IP]

**Create uptime check**

gcloud monitoring uptime create griffin-dev-wp-uptime-check \  
 --display-name="Griffin Dev WP Uptime Check" \  
 --resource-labels=host=$WORDPRESS\_EXTERNAL\_IP

**Alternative for Uptime Check: Using the Google Cloud Console**

If the CLI approach continues to present issues, you can create an uptime check through the Google Cloud Console:

1. Go to the Google Cloud Console.
2. Navigate to **Monitoring**.
3. Select **Uptime Checks** from the menu.
4. Click on **Create Uptime Check**.
5. Fill in the required details:

* **Title**: Griffin Dev WP Uptime Check
* **Resource Type**: URL
* **Hostname**: $WORDPRESS\_SITE\_URL (use the actual external IP address, e.g., 34.48.95.59)
* **Path**: /
* **Port**: 80
* **Check Frequency**: 5 minutes
* **Timeout**: 10 seconds

Save the uptime check. This should successfully create an uptime check for your WordPress site.

**Task 09: Provide access for an additional engineer**

gcloud projects add-iam-policy-binding $GOOGLE\_CLOUD\_PROJECT \  
 --member="user:$ADDITIONAL\_ENGINEER\_EMAIL" \  
 --role="roles/editor"

**Alternative Approach: Using Google Cloud Console:**

* Navigate to **IAM & Admin**.
* Click **Add** to add a new member.
* Enter the email of the additional engineer.
* Assign the **Editor** role.
* Click **Save**.

**Conclusion**

Congratulations on completing the lab! You have successfully set up a development environment using Google Cloud, including VPCs, a bastion host, a Cloud SQL instance, and a Kubernetes cluster. You also enabled monitoring and granted access to an additional engineer. Keep exploring Google Cloud to enhance your skills further.