## Create and Manage Cloud Resources

* how to do the following: Write gcloud commands and use Cloud Shell, create and deploy virtual machines in Compute Engine, run containerized applications on Google Kubernetes Engine, and configure network and HTTP load balancers.
* A Tour of Google Cloud Hands-on Labs
  + In this first hands-on lab you will access the Google Cloud Platform Console and use these basic Google Cloud features: Projects, Resources, IAM Users, Roles, Permissions, and APIs.
* Creating a Virtual Machine
  + In this hands-on lab, you’ll learn how to create a Google Compute Engine virtual machine and understand zones, regions, and machine types. To preview, watch the short video Create a Virtual Machine, GCP Essentials.
* 1.0 Compute Engine: Qwik Start - Windows
  + Google Compute Engine lets you create and run virtual machines on Google infrastructure. In this lab you create a Windows Server instance in the Google Compute Engine and access it with RDP. Watch a short preview, Launch a Windows Server Instance, GCP Essentials.
* 2.0 Getting Started with Cloud Shell and gcloud
  + In this hands-on lab you will learn how to connect to computing resources hosted on Google Cloud Platform via the web. You will also learn how to use Cloud Shell and the Cloud SDK gcloud command. For a preview, watch the short video Get Started with Cloud Shell, GCP Essentials.
* 3.0 Kubernetes Engine: Qwik Start
  + Google Kubernetes Engine provides a managed environment for deploying, managing, and scaling your containerized applications using Google infrastructure. This hands-on lab shows you how deploy a containerized application with Kubernetes Engine. Watch the short video Manage Containerized Apps with Kubernetes Engine.
* 4.0 Set Up Network and HTTP Load Balancers
  + In this hands-on lab, you'll learn how setup both network load balancers and HTTP load balancers for your application running in Compute Engine virtual machines.
* 5.0 Create and Manage Cloud Resources: Challenge Lab
  + This challenge lab tests your skills and knowledge from the labs in the Create and Manage Cloud Resources quest. You should be familiar with the content of the labs before attempting this lab.

## 1.0 Compute Engine: Qwik Start - Windows

* Overview
  + Compute Engine lets you create and run VMs on Google infrastructure. Compute Engine offers scale, performance, and value that allows you to easily launch large compute clusters on Google's infrastructure.
  + You can run your Windows apps on Compute Engine and take advantage of many benefits available to VM instances, such as reliable storage options, the speed of the Google network, and Autoscaling.
  + learn how to launch a Windows Server instance in Compute Engine and use RDP to connect to it.
* Create a virtual machine instance
  + on the Navigation menu , click Compute Engine > VM instances, and then click Create Instance.
  + In the Machine configuration section, for Series select N1.
  + In the Boot disk section, click Change to begin configuring your boot disk.
  + Under OS select Windows Server and under Version select Windows Server 2012 R2 Datacenter, and then click Select. Leave all other settings as their defaults. Click Create to create the instance.
* In the Cloud Console, in the top right toolbar, click the Activate Cloud Shell button.
  + You can list the active account name with this command: gcloud auth list
  + You can list the project ID with this command: gcloud config list project
* Remote Desktop (RDP) into the Windows Server
  + Test the status of Windows Startup
    - After a short time, the Windows Server instance will be provisioned and listed on the VM Instances page with a green status icon Green Status Icon.
    - However the server instance may not yet be ready to accept RDP connections, as it takes a while for all the OS components to initialize.
    - To see whether the server instance is ready for an RDP connection, run the following command at your Cloud Shell terminal command line:
      * gcloud compute instances get-serial-port-output instance-1
      * If prompted, type n and press Enter.
    - Repeat the command until you see the following in the command output, which tells you that the OS components have initialized and the Windows Server is ready to accept your RDP connection (attempt in the next step).
      * ------------------------------------------------------------
      * Instance setup finished. instance-1 is ready to use.
      * ------------------------------------------------------------
  + RDP into the Windows Server
    - To set a password for logging into the RDP, run the following command in Cloud Shell terminal and replace [instance] with the VM Instance that you have created and set [username] as admin.
      * gcloud compute reset-windows-password [instance] --zone us-central1-a --user [username]
      * If asked Would you like to set or reset the password for [admin] (Y/n)?, enter Y.
    - There are different ways to connect to your server through RDP, depending on whether you are on Windows or not:
    - Click on RDP. specify Windows username admin and password from the Shell output.
* Test your understanding
  + We can create a Windows instance in Google Cloud by changing its \_\_\_\_ in the VM instance console.
    - Boot disk to Windows image
  + Which command is used to check whether the server is ready for an RDP connection?
    - gcloud compute instances get-serial-port-output

## 2.0 Getting Started with Cloud Shell and gcloud

* Overview
  + Cloud Shell provides you with command-line access to computing resources hosted on Google Cloud. Cloud Shell is a Debian-based virtual machine with a persistent 5-GB home directory, which makes it easy for you to manage your Google Cloud projects and resources. The gcloud command-line tool and other utilities you need are pre-installed in Cloud Shell, which allows you to get up and running quickly.
  + how to connect to computing resources hosted on Google Cloud via Cloud Shell with the gcloud tool.
  + What you'll do
    - Practice using gcloud commands.
    - Connect to compute services hosted on Google Cloud.
* 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
  + You can list the project ID with this command: gcloud config list project
* Task 1: Configure your environment
  + Understanding regions and zones
    - Certain Google 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. The following image shows zones in their respective regions:
    - Graphical user interface

      Description automatically generated
    - Resources that live in a zone are referred to as zonal resources. Virtual machine instances and persistent disks live in a zone. If you want 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 address.
    - To see what your default region and zone settings are, run the following commands:
      * gcloud config get-value compute/zone
      * gcloud config get-value compute/region
      * If the google-compute-default-region or google-compute-default-zone responses are (unset), that means no default zone or region is set.
  + Identify your default region and zone
    - Copy your project ID to your clipboard or text editor. The project ID is listed in 2 places:
    - In the GC Console, Click Navigation menu -> Home > Dashboard - > Project info.
    - On the Qwiklabs tab near your username and password.
    - In Cloud Shell, run the following gcloud command, replacing <your\_project\_ID> with the project ID you copied: gcloud compute project-info describe --project <your\_project\_ID>
      * Find the default zone and region metadata values in the output. You'll use the zone (google-compute-default-zone) from the output later in this lab.
      * If the google-compute-default-region and google-compute-default-zone keys and values are missing from the output, no default zone or region is set.
  + Set environment variables
    - Environment variables define your environment and help save time when you write scripts that contain APIs or executables.
    - Create an environment variable to store your Project ID, replacing <your\_project\_ID> with the value for name from the gcloud compute project-info describe command you ran earlier:
      * export PROJECT\_ID=<your\_project\_ID>
    - Create an environment variable to store your Zone, replacing <your\_zone> with the value for zone from the gcloud compute project-info describe command you ran earlier:
      * export ZONE=<your\_zone>
    - To verify that your variables were set properly, run the following commands:
      * echo $PROJECT\_ID
      * echo $ZONE
    - If the variables were set correctly, the echo commands will output your Project ID and Zone.
  + Create a virtual machine with the gcloud tool
    - Use the gcloud tool to create a new virtual machine (VM) instance.
    - To create your VM, run the following command:
      * gcloud compute instances create gcelab2 --machine-type n1-standard-2 --zone $ZONE
      * Command details
        + gcloud compute allows you to manage your Compute Engine resources in a format that's simpler than the Compute Engine API.
        + instances create creates a new instance.
        + gcelab2 is the name of the VM.
        + The --machine-type flag specifies the machine type as n1-standard-2.
        + The --zone flag specifies where the VM is created.
        + If you omit the --zone flag, the gcloud tool can infer your desired zone based on your default properties. Other required instance settings, such as machine type and image, are set to default values if not specified in the create command.
    - To open help for the create command, run the following command:
      * gcloud compute instances create --help
    - Note: Press ENTER or the spacebar to scroll through the help content. To exit the content, type Q.
  + Explore gcloud commands
    - The gcloud tool offers simple usage guidelines that are available by adding the -h flag (for help) onto the end of any gcloud command.
    - Run the following command: gcloud -h
    - You can access more verbose help by appending the --help flag onto a command or running the gcloud help command. Run the following command: gcloud config --help
      * gcloud help config
      * The results of the gcloud config --help and gcloud help config commands are equivalent. Both return long, detailed help.
      * gcloud Global Flags govern the behavior of commands on a per-invocation level. Flags override any values set in SDK properties.
    - View the list of configurations in your environment: gcloud config list
    - To see all properties and their settings: gcloud config list --all
    - List your components: gcloud components list
      * This command displays the gcloud components that are ready for you to use in this lab.
* Task 2: Install a new component
  + Next, you'll install a gcloud component that makes working in the gcloud tool easier.
  + Auto-complete mode
    - gcloud interactive has auto prompting for commands and flags and displays inline help snippets in the lower section of the pane as the command is typed.
    - You can use dropdown menus to auto-complete static information, such as command and sub-command names, flag names, and enumerated flag values.
    - Install the beta components: sudo apt-get install google-cloud-sdk
    - Enable the gcloud interactive mode: gcloud beta interactive
      * When using the interactive mode, press TAB to complete file path and resource arguments. If a dropdown menu appears, press TAB to move through the list, and press the spacebar to select your choice.
    - To try this feature, start typing the following command, and use auto-complete to replace <your\_vm> with an existing VM in your project: gcloud compute instances describe <your\_vm>
      * A list of commands is displayed below the Cloud Shell pane. Pressing F2 toggles the active help section to ON or OFF.
    - To exit from the interactive mode, run the following command: exit
* Task 3: Connect to your VM instance with SSH
  + gcloud compute makes connecting to your instances easy. The gcloud compute ssh command provides a wrapper around SSH, which takes care of authentication and the mapping of instance names to IP addresses. To connect to your VM with SSH, run the following command:
    - gcloud compute ssh gcelab2 --zone $ZONE
      * Do you want to continue? (Y/n) To continue, type Y.
    - To leave the passphrase empty, press ENTER twice.
    - You don't need to do anything here, so to disconnect from SSH and exit the remote shell, run the following command: exit
* Task 4: Use the Home directory
  + Now try out your Home directory. The contents of your Cloud Shell Home directory persist across projects between all Cloud Shell sessions, even after the virtual machine is terminated and restarted.
  + Change your current working directory: cd $HOME
  + Open your .bashrc configuration file by using the vi text editor: vi ./.bashrc
    - The editor opens and displays the contents of the file.
  + To exit the editor, press ESC, then type :wq, and then press Enter.
  + Test your understanding
    - Three basic ways to interact with Google Cloud services and resources are:
      * Command-line interface, Cloud Console, Client libraries

## 3.0 Kubernetes Engine: Qwik Start

* Overview
  + Google Kubernetes Engine (GKE) provides a managed environment for deploying, managing, and scaling your containerized applications using Google infrastructure. The Kubernetes Engine environment consists of multiple machines (specifically Compute Engine instances) grouped to form a container cluster. In this lab, you get hands-on practice with container creation and application deployment with GKE.
* Cluster orchestration with Google Kubernetes Engine
  + Google Kubernetes Engine (GKE) clusters are powered by the Kubernetes open source cluster management system. Kubernetes provides the mechanisms through which you interact with your container cluster. You use Kubernetes commands and resources to deploy and manage your applications, perform administrative tasks, set policies, and monitor the health of your deployed workloads.
  + Kubernetes draws on the same design principles that run popular Google services and provides the same benefits: automatic management, monitoring and liveness probes for application containers, automatic scaling, rolling updates, and more. When you run your applications on a container cluster, you're using technology based on Google's 10+ years of experience with running production workloads in containers.
* Kubernetes on Google Cloud
  + When you run a GKE cluster, you also gain the benefit of advanced cluster management features that Google Cloud provides. These include:
    - Load balancing for Compute Engine instances
    - Node pools to designate subsets of nodes within a cluster for additional flexibility
    - Automatic scaling of your cluster's node instance count
    - Automatic upgrades for your cluster's node software
    - Node auto-repair to maintain node health and availability
    - Logging and Monitoring with Cloud Monitoring for visibility into your cluster
* Task 1: Set a default compute zone gcloud config set compute/zone us-central1-a
* Task 2: Create a GKE cluster
  + A cluster consists of at least one cluster master machine and multiple worker machines called nodes. Nodes are Compute Engine virtual machine (VM) instances that run the Kubernetes processes necessary to make them part of the cluster.
  + To create a cluster, run the following command, replacing [CLUSTER-NAME] with the name you choose for the cluster (for example:my-cluster). gcloud container clusters create [CLUSTER-NAME]
  + You can ignore any warnings in the output. It might take several minutes to finish creating the cluster.
* Task 3: Get authentication credentials for the cluster
  + After creating your cluster, you need authentication credentials to interact with it.
  + To authenticate the cluster, run the following command, replacing [CLUSTER-NAME] with the name of your cluster: gcloud container clusters get-credentials [CLUSTER-NAME]
* Task 4: Deploy an application to the cluster
  + You can now deploy a containerized application to the cluster. For this lab, you'll run hello-app in your cluster.
  + GKE uses Kubernetes objects to create and manage your cluster's resources. Kubernetes provides the Deployment object for deploying stateless applications like web servers. Service objects define rules and load balancing for accessing your application from the internet.
  + To create a new Deployment hello-server from the hello-app container image, run the following kubectl create command: kubectl create deployment hello-server --image=gcr.io/google-samples/hello-app:1.0
    - This Kubernetes command creates a Deployment object that represents hello-server. In this case, --image specifies a container image to deploy. The command pulls the example image from a Container Registry bucket. gcr.io/google-samples/hello-app:1.0 indicates the specific image version to pull. If a version is not specified, the latest version is used.
  + To create a Kubernetes Service, which is a Kubernetes resource that lets you expose your application to external traffic, run the following kubectl expose command:
    - kubectl expose deployment hello-server --type=LoadBalancer --port 8080
      * In this command:
      * --port specifies the port that the container exposes.
      * type="LoadBalancer" creates a Compute Engine load balancer for your container.
    - To inspect the hello-server Service, run kubectl get: kubectl get service
    - To view the application from your web browser, open a new tab and enter the following address, replacing [EXTERNAL IP] with the EXTERNAL-IP for hello-server. http://[EXTERNAL-IP]:8080
* Task 5: Deleting the cluster gcloud container clusters delete [CLUSTER-NAME]
  + When prompted, type Y to confirm.

## 4.0 Set Up Network and HTTP Load Balancers

* Overview
  + learn the differences between a network load balancer and an HTTP load balancer and how to set them up for your applications running on Compute Engine virtual machines (VMs).
  + There are several ways you can load balance on Google Cloud. This lab takes you through the set up of the following load balancers:
    - Network Load Balancer
    - HTTP(s) Load Balancer
* What you'll do
  + Set up a network load balancer. Set up an HTTP load balancer.
  + Get hands-on experience learning the differences between network load balancers and HTTP load balancers.
* Task 1: Set the default region and zone for all resources
  + In Cloud Shell, set the default zone: gcloud config set compute/zone us-central1-a
  + Set the default region: gcloud config set compute/region us-central1
* Task 2: Create multiple web server instances
  + For this load balancing scenario, create three Compute Engine VM instances and install Apache on them, then add a firewall rule that allows HTTP traffic to reach the instances.
  + Create three new virtual machines in your default zone and give them all the same tag. The code provided sets the zone to us-central1-a. Setting the tags field lets you reference these instances all at once, such as with a firewall rule. These commands also install Apache on each instance and give each instance a unique home page.
    - gcloud compute instances create www1 \
    - --image-family debian-9 \
    - --image-project debian-cloud \
    - --zone us-central1-a \
    - --tags network-lb-tag \
    - --metadata startup-script="#! /bin/bash
    - sudo apt-get update
    - sudo apt-get install apache2 -y
    - sudo service apache2 restart
    - echo '<!doctype html><html><body><h1>www1</h1></body></html>' | tee /var/www/html/index.html"
    - gcloud compute instances create www2 \
    - --image-family debian-9 \
    - --image-project debian-cloud \
    - --zone us-central1-a \
    - --tags network-lb-tag \
    - --metadata startup-script="#! /bin/bash
    - sudo apt-get update
    - sudo apt-get install apache2 -y
    - sudo service apache2 restart
    - echo '<!doctype html><html><body><h1>www2</h1></body></html>' | tee /var/www/html/index.html"
    - gcloud compute instances create www3 \
    - --image-family debian-9 \
    - --image-project debian-cloud \
    - --zone us-central1-a \
    - --tags network-lb-tag \
    - --metadata startup-script="#! /bin/bash
    - sudo apt-get update
    - sudo apt-get install apache2 -y
    - sudo service apache2 restart
    - echo '<!doctype html><html><body><h1>www3</h1></body></html>' | tee /var/www/html/index.html"
  + Create a firewall rule to allow external traffic to the VM instances:
    - gcloud compute firewall-rules create www-firewall-network-lb \
    - --target-tags network-lb-tag --allow tcp:80
  + Now you need to get the external IP addresses of your instances and verify that they are running.
  + Run the following to list your instances. You'll see their IP addresses in the EXTERNAL\_IP column:
    - gcloud compute instances list
  + Verify that each instance is running with curl, replacing [IP\_ADDRESS] with the IP address for each of your VMs:
    - curl http://[IP\_ADDRESS]
* Task 3: Configure the load balancing service
  + When you configure the load balancing service, your virtual machine instances will receive packets that are destined for the static external IP address you configure. Instances made with a Compute Engine image are automatically configured to handle this IP address.
  + Create a static external IP address for your load balancer:
    - gcloud compute addresses create network-lb-ip-1 \
    - --region us-central1
  + Add a legacy HTTP health check resource:
    - gcloud compute http-health-checks create basic-check
  + Add a target pool in the same region as your instances. Run the following to create the target pool and use the health check, which is required for the service to function:
    - gcloud compute target-pools create www-pool \
    - --region us-central1 --http-health-check basic-check
  + Add the instances to the pool:
    - gcloud compute target-pools add-instances www-pool \
    - --instances www1,www2,www3
  + Add a forwarding rule:
    - gcloud compute forwarding-rules create www-rule \
    - --region us-central1 \
    - --ports 80 \
    - --address network-lb-ip-1 \
    - --target-pool www-pool
* Task 4: Sending traffic to your instances
  + Now that the load balancing service is configured, you can start sending traffic to the forwarding rule and watch the traffic be dispersed to different instances.
  + Enter the following command to view the external IP address of the www-rule forwarding rule used by the load balancer: gcloud compute forwarding-rules describe www-rule --region us-central1
  + Use curl command to access the external IP address, replacing IP\_ADDRESS with an external IP address from the previous command:
    - while true; do curl -m1 IP\_ADDRESS; done
  + The response from the curl command alternates randomly among the three instances. If your response is initially unsuccessful, wait approximately 30 seconds for the configuration to be fully loaded and for your instances to be marked healthy before trying again.
  + Use Ctrl + c to stop running the command.
* Task 5: Create an HTTP load balancer
  + HTTP(S) Load Balancing is implemented on Google Front End (GFE). GFEs are distributed globally and operate together using Google's global network and control plane. You can configure URL rules to route some URLs to one set of instances and route other URLs to other instances. Requests are always routed to the instance group that is closest to the user, if that group has enough capacity and is appropriate for the request. If the closest group does not have enough capacity, the request is sent to the closest group that does have capacity.
  + To set up a load balancer with a Compute Engine backend, your VMs need to be in an instance group. The managed instance group provides VMs running the backend servers of an external HTTP load balancer. For this lab, backends serve their own hostnames.
  + First, create the load balancer template:
    - gcloud compute instance-templates create lb-backend-template \
    - --region=us-central1 \
    - --network=default \
    - --subnet=default \
    - --tags=allow-health-check \
    - --image-family=debian-9 \
    - --image-project=debian-cloud \
    - --metadata=startup-script='#! /bin/bash
    - apt-get update
    - apt-get install apache2 -y
    - a2ensite default-ssl
    - a2enmod ssl
    - vm\_hostname="$(curl -H "Metadata-Flavor:Google" \
    - http://169.254.169.254/computeMetadata/v1/instance/name)"
    - echo "Page served from: $vm\_hostname" | \
    - tee /var/www/html/index.html
    - systemctl restart apache2'
  + Create a managed instance group based on the template:
    - gcloud compute instance-groups managed create lb-backend-group \
    - --template=lb-backend-template --size=2 --zone=us-central1-a
  + Create the fw-allow-health-check firewall rule. This is an ingress rule that allows traffic from the Google Cloud health checking systems (130.211.0.0/22 and 35.191.0.0/16). This lab uses the target tag allow-health-check to identify the VMs.
    - gcloud compute firewall-rules create fw-allow-health-check \
    - --network=default \
    - --action=allow \
    - --direction=ingress \
    - --source-ranges=130.211.0.0/22,35.191.0.0/16 \
    - --target-tags=allow-health-check \
    - --rules=tcp:80
  + Now that the instances are up and running, set up a global static external IP address that your customers use to reach your load balancer.
    - gcloud compute addresses create lb-ipv4-1 \
    - --ip-version=IPV4 \
    - --global
  + Note the IPv4 address that was reserved:
    - gcloud compute addresses describe lb-ipv4-1 \
    - --format="get(address)" \
    - --global
  + Create a health check for the load balancer:
    - gcloud compute health-checks create http http-basic-check \
    - --port 80
  + Create a backend service:
    - gcloud compute backend-services create web-backend-service \
    - --protocol=HTTP \
    - --port-name=http \
    - --health-checks=http-basic-check \
    - --global
  + Add your instance group as the backend to the backend service:
    - gcloud compute backend-services add-backend web-backend-service \
    - --instance-group=lb-backend-group \
    - --instance-group-zone=us-central1-a \
    - --global
  + Create a URL map to route the incoming requests to the default backend service:
    - gcloud compute url-maps create web-map-http \
    - --default-service web-backend-service
  + Create a target HTTP proxy to route requests to your URL map:
    - gcloud compute target-http-proxies create http-lb-proxy \
    - --url-map web-map-http
  + Create a global forwarding rule to route incoming requests to the proxy:
    - gcloud compute forwarding-rules create http-content-rule \
    - --address=lb-ipv4-1\
    - --global \
    - --target-http-proxy=http-lb-proxy \
    - --ports=80
* Task 6: Testing traffic sent to your instances
  + In the Cloud Console, from the Navigation menu, go to Network services > Load balancing.
  + Click on the load balancer that you just created (web-map-http).
  + In the Backend section, click on the name of the backend and confirm that the VMs are Healthy. If they are not healthy, wait a few moments and try reloading the page.
  + When the VMs are healthy, test the load balancer using a web browser, going to http://IP\_ADDRESS/, replacing IP\_ADDRESS with the load balancer's IP address.
  + This may take three to five minutes. If you do not connect, wait a minute, and then reload the browser.
  + Your browser should render a page with content showing the name of the instance that served the page, along with its zone (for example, Page served from: lb-backend-group-xxxx).

## 5.0 Create and Manage Cloud Resources: Challenge Lab

* Topics tested:
  + Create an instance
  + Create a 3-node Kubernetes cluster and run a simple service
  + Create an HTTP(s) load balancer in front of two web servers
* Challenge scenario
  + You have started a new role as a Junior Cloud Engineer for Jooli, Inc. You are expected to help manage the infrastructure at Jooli. Common tasks include provisioning resources for projects.
  + You are expected to have the skills and knowledge for these tasks, so step-by-step guides are not provided. Some Jooli, Inc. standards you should follow:
    - Create all resources in the default region or zone, unless otherwise directed.
    - Naming normally uses the format team-resource; for example, an instance could be named nucleus-webserver1.
    - Allocate cost-effective resource sizes. Projects are monitored, and excessive resource use will result in the containing project's termination (and possibly yours), so plan carefully. This is the guidance the monitoring team is willing to share: unless directed, use f1-micro for small Linux VMs, and use n1-standard-1 for Windows or other applications, such as Kubernetes nodes.
* Task 1. Create a project jumphost instance
  + You will use this instance to perform maintenance for the project. Requirements:
    - Name the instance Instance name (nucleus-jumphost-989).
    - Use an f1-micro machine type.
    - Use the default image type (Debian Linux).
  + gcloud compute instances create nucleus-jumphost-989 --machine-type f1-micro --zone us-east1-b
* Task 2. Create a Kubernetes service cluster
  + The team is building an application that will use a service running on Kubernetes. You need to:
    - Create a cluster (in the us-east1-b zone) to host the service.
    - Use the Docker container hello-app (gcr.io/google-samples/hello-app:2.0) as a place holder; the team will replace the container with their own work later.
    - Expose the app on port App port number 8082.
  + gcloud config set compute/zone us-east1-b
  + gcloud config set compute/region us-east1
  + gcloud container clusters create mak-clu
  + gcloud container clusters get-credentials mak-clu
  + kubectl create deployment hello-app --image=gcr.io/google-samples/hello-app:2.0
  + kubectl expose deployment hello-app --type=LoadBalancer --port 8082
* Task 3. Set up an HTTP load balancer
  + You will serve the site via nginx web servers, but you want to ensure that the environment is fault-tolerant. Create an HTTP load balancer with a managed instance group of 2 nginx web servers. Use the following code to configure the web servers; the team will replace this with their own configuration later.
    - cat << EOF > startup.sh
    - #! /bin/bash
    - apt-get update
    - apt-get install -y nginx
    - service nginx start
    - sed -i -- 's/nginx/Google Cloud Platform - '"\$HOSTNAME"'/' /var/www/html/index.nginx-debian.html
    - EOF
  + You need to:
    - Create an instance template.
    - Create a target pool.
    - Create a managed instance group.
    - Create a firewall rule named as Firewall rule to allow traffic (80/tcp).
    - Create a health check.
    - Create a backend service, and attach the managed instance group with named port (http:80).
    - Create a URL map, and target the HTTP proxy to route requests to your URL map.
    - Create a forwarding rule.
  + cat << EOF > startup.sh
  + #! /bin/bash
  + apt-get update
  + apt-get install -y nginx
  + service nginx start
  + sed -i -- 's/nginx/Google Cloud Platform - '"\$HOSTNAME"'/' /var/www/html/index.nginx-debian.html
  + EOF
  + gcloud compute instance-templates create nginx-template --region=us-east1 --network=default \
  + --subnet=default --tags=allow-health-check --image-family=debian-9 --image-project=debian-cloud --metadata-from-file startup-script=startup.sh
  + gcloud compute http-health-checks create basic-check
  + gcloud compute target-pools create nginx-pool --region us-east1 --http-health-check basic-check
  + gcloud compute instance-groups managed create nginx-group --template=nginx-template --size=2 --zone=us-east1-b
  + gcloud compute firewall-rules create accept-tcp-rule-911 --network=default --action=allow --direction=ingress --source-ranges=130.211.0.0/22,35.191.0.0/16 --target-tags=allow-health-check --rules=tcp:80
  + gcloud compute health-checks create http http-basic-check --port 80
  + gcloud compute backend-services create nginx-backend-service --protocol=HTTP --port-name=http --health-checks=http-basic-check --global
  + gcloud compute backend-services add-backend nginx-backend-service --instance-group=nginx-group --instance-group-zone=us-east1-b --global
  + gcloud compute url-maps create web-map-http --default-service nginx-backend-service
  + gcloud compute target-http-proxies create http-lb-proxy --url-map web-map-http
  + gcloud compute forwarding-rules create http-content-rule --address=lb-ipv4-1 --global --target-http-proxy=http-lb-proxy --ports=80
  + gcloud compute instance-groups managed set-named-ports nginx-group --named-ports http:80 --zone=us-east1-b