Cloud Lab 1: Taking a simple Go App through Docker and Kubernetes to Google Cloud

Created by Lomax, Ken, last modified about 3 hours ago

Patriots,

as part of " Pimping our skills - keeping up with the cloud" we are starting a series of **Cloud Labs.**

Cloud Labs will be intense ~2 hour labs, where we focus on a cloud-related technology or journey in much more detail than our Lunch Talks allow. They will feature examples and hands-on tasks, to learn and practice common cloud-related stories. Some will be quite basic, and some much more advanced. The goal is to provide successful initial journeys for engineers in the cloud, and to refer you to excellent relevant material to help you go further. They will be bite-sized examples, to introduce you to these technologies without overwhelming you in information and noise.

The first Cloud Lab will be **Taking a simple Go App through Docker and Kubernetes to Google Cloud**.

The Agenda is still being adjusted, and will include things like:

Create a project on Google Cloud

- **Go** - Building a simple Go Micro Service
- **Docker** - Packaging it in Docker
- **Kubernetes** - Wrapping in Kubernetes
- **Google Cloud** - Taking to the Cloud
- **Update** - Doing a rolling update in the Cloud

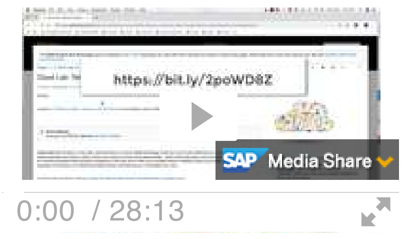
Come along with your laptops and brains, and see how all the pieces fit together. And if you want to lead a Cloud Lab let me know 🙂

Merci

Ken

This CloudLab is Tested on a Mac

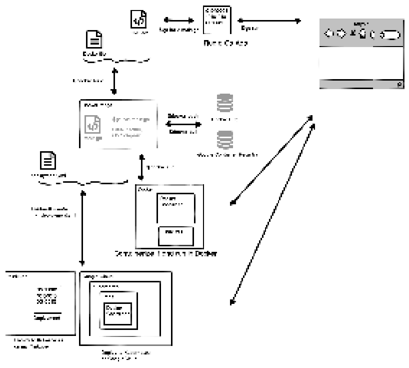
We want to keep this Cloud Labs correct. Please report any problems you have to ken.lomax@sap.com"



Initiatives to Keep up with the Cloud to include:

- Lunch Talks - Presentations over Pizzas
- **Cloud Labs** - More in-depth than Lunch Talks
- VideoQ&A -Watch one of the many excellent youtubes out there, and then discuss
- Dojos - Hands-on Workshops with rotating pairs in the driving seat
- Workshops
- Hackathons

Journey

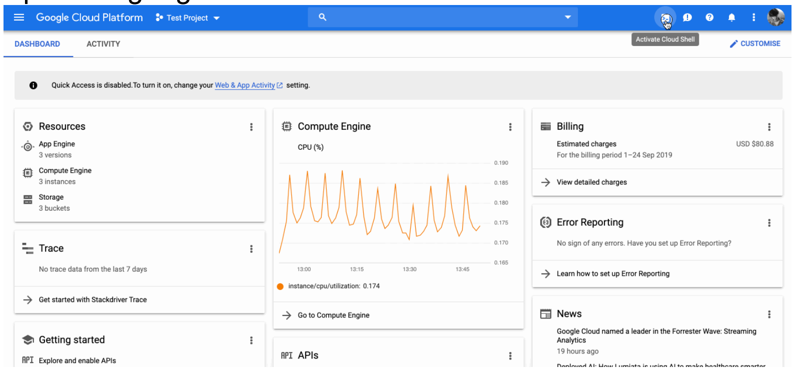


Prerequisites	Tour	Learn more here..

I want to write a simple Go nano service to see how it works

If on **Windows..** you can follow these steps by using the google cloud shell..

- Create a (free) account at cloud.google.com
- Create a new project in this account called cloudlab1
- Open the google cloud shell



Google Cloud Platform · Test Project

Activate Cloud Shell

DASHBOARD ACTIVITY CUSTOMISE

Quick Access is disabled. To turn it on, change your Web & App Activity setting.

Resources: App Engine 3 versions; Compute Engine 3 instances; Storage 3 buckets

Compute Engine: CPU (%) — instance/cpu/utilization: 0.174 — Go to Compute Engine

Billing: Estimated charges USD $80.88 For the billing period 1–24 Sep 2019 — View detailed charges

Error Reporting: No sign of any errors. Have you set up Error Reporting? — Learn how to set up Error Reporting

Trace: No trace data from the last 7 days — Get started with Stackdriver Trace

News: Google Cloud named a leader in the Forrester Wave: Streaming Analytics — 19 hours ago

Getting started: Explore and enable APIs

APIs

- You can now work in the google cloud shell.

If on a **Mac..**

- This cloudlab assumes you have installed GoLang.
- Once you have done that, open a Terminal for the following steps.

Write a web server in Go that listens on port 8080 and returns "Hello world!" and the Host name.

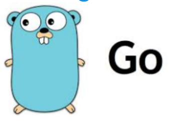
```
mkdir -p cloudlab1
cd cloudlab1
go mod init cloudlab1
```

Create the file cloudlab1.go in this folder:

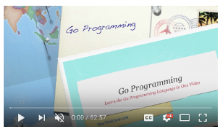
cloudlab1.go

```
package main // Go expects entry point to be a f

import (
        "fmt" // Imports will cause an error if
        "log"
        "net/http"
        "os"
)

func main() {
        // use PORT environment variable, or def
        port := "8080"
        if fromEnv := os.Getenv("PORT"); fromEnv
```

Golang Home



Golang in 45 minutes by Derek Banas



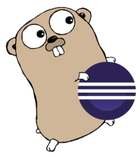
Golang founder Rob Pike on why ... successful

Pages ... with the cloud"

```
                port = fromEnv
        }

        // register hello function to handle all
        server := http.NewServeMux()
        server.HandleFunc("/", hello)

        // start the web server on port and acce
        log.Printf("Server listening on port %s"
        err := http.ListenAndServe(":"+port, ser
        log.Fatal(err)
}

// hello responds to the request with a plain-te
func hello(w http.ResponseWriter, r *http.Reques
        log.Printf("Serving request: %s", r.URL.
        host, _ := os.Hostname()
        fmt.Fprintf(w, "Hello, world!\n")
        fmt.Fprintf(w, "Hostname: %s\n", host)
}
```

Go Http Package Documentation

Goclipse



Compile and run the application.

> go run cloudlab1.go

In a second terminal call this webservice:

> curl localhost:8080

Open localhost:8080 and confirm you can access your service.

Note that the hostname is the name of your local machine.

I want to compile my Go App to execute on any Linux machine

Compile your Go application to Linux

Blog on Go Compilation for Scratch

By default a compiled Go program will link to dynamic libraries at run time. This is ok if you compile and then run your Go application on one (type of) machine. But if you want to compile your Go on a Mac, and then run it on Linux, you need to compile your Go application as one static binary, with all dependent libraries included. There is a good explanation of this here. To do this run:

> CGO_ENABLED=0 GOOS=linux go build -a -installsuffix cgo -o cloudlab1 .

This gives you a main program that will execute (only) on a (virtual, or real) Linux machine which is what we require when using Docker in the next steps.

> go run cloudlab1

Open localhost:8080 and confirm you can access your service.

Note again that the hostname is the name of your local machine.

I want to containerize my Go App, so that it can run on any machine that has Docker on it

If you are running on your local computer, you need to have Installed Docker. If you are running directly in

Containerize your Go App

Docker Docs

Create a Dockerfile alongside main.go describing how to build a Docker Image containing your Go App.

Dockerfile

GCP, this is already done for you.

```
# Which Docker image shall we based ours upon?
FROM golang
# Do we want to copy over any files ?
ADD . app/
# Set the working directory
WORKDIR app/
# What command should the running Docker Contair
CMD go run cloudlab1.go
```

Build the Docker Image from the Dockerfile. The -t flag allows you to specify a name and optional tag (v1). Use your Docker Hub account name, in this example "kenlomax" (<yourDockerHubName>/cloudlab1:v1):

```
docker build -t kenlomax/cloudlab1:v1 .
```

This Docker Image will contain everything that it needs to execute on top of a Linux kernel - libs, bins, executables

List your image with

```
docker images -a
```

A running instance of a Docker Image is called a Docker Container.

Run a Docker Container in detached mode

```
docker run -d -p 8080:8080 kenlomax/cloudlab1:v1
```

Hit it with:

```
curl localhost:8080
```

Note ..

- your **Linux executable** "cloudlab1" is running on your (non Linux) computer. This is possible because Docker is executing within a virtual Linux machine.
- the hostname is **NOT** the name of your computer, but instead is the Linux VM in which the Docker Container is running.

Find the id of your running docker container

```
docker ps
```

Log into your docker container and look around

```
docker exec -it <docker id> bash
ls -la
```

To stop a Docker Container

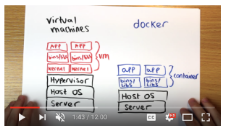
```
docker ps -a
docker stop <the container ID that's running>
```

ⓘ **Aside**
On Linux systems, Docker directly leverages the kernel of the host system, and file system mounts are native.

On Windows and Mac, it's slightly different. These operating systems do not provide a Linux Kernel, so Docker starts a virtual machine with a small Linux installed and runs Docker containers in there. More..

Docker CheatSheet

Good overview of Docker in 12 minutes



Dockerhub

Docker run reference

I want to make my Docker Image available to the world, so anyone can run it

Create an account at hub.docker.com

Push to Docker Hub

Log in to Docker with your Docker ID. This allows you to push and pull images from Docker Hub.

```
docker login
```

Push your image to Docker Hub so others can see it there.

```
docker push kenlomax/cloudlab1:v1
```

`Check you can see it at https://hub.docker.com/`

Note from docker's output that it pushed only the topmost layer - the other layers are available to Docker Hub already.

Now any one with docker running on their machine, can download this image and run it with:

```
docker run -d -p 8080:8080 kenlomax/cloudlab1:v1
```

Aside
Docker runs processes in isolated containers. A container is a process which runs on a host. The host may be local or remote. When an operator executes `docker run`, the container process that runs is isolated in that it has its own file system, its own networking, and its own isolated process tree separate from the host.

Dockerhub

Pushing to Docker hub, from Docker Docs:

I want my App to benefit from Kubernetes' declarative scaling mojo

Wrap in Kubernetes so we can start scaling it

Create a Kubernetes Deployment file

```
vi deployment.yml
```

deployment.yml

```
---
kind: Service
apiVersion: v1
metadata:
  name: cloudlab1service
spec:
  selector:
    app: myapp
  ports:
    - protocol: "TCP"
      # Port accessible inside cluster
      port: 8081
      # Port to forward to inside the pod
      targetPort: 8080
      # Port accessible outside cluster
      nodePort: 30003
  type: LoadBalancer

---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mycloudlab1deployment
spec:
```

Why Kubernetes in 5 minutes.

Kubernetes tutorials

Kubernetes for the Spring Developer by Meghan

Pages

with the cloud"

James Quigley on Introduction to

```
  selector:
    matchLabels:
      app: myapp
  replicas: 2
  template:
    metadata:
      labels:
        app: myapp
    spec:
      containers:
        - name: myapp
          image: kenlomax/cloudlab1:v1 # Replace
          ports:
            - containerPort: 8080
```

Install Minikube

Install kubectl

Run in local Kubernetes*

Minikube is a tool that makes it easy to run Kubernetes locally. Minikube runs a single-node Kubernetes cluster inside a VM on your laptop for users looking to try out Kubernetes or develop with it day-to-day.

Start minikube*

> minikube start

Kubectl is Kubernetes' command-line tool, to deploy and manage applications on Kubernetes from your command line.

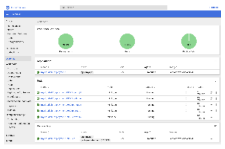
Point your kubectl to your local minikube

> kubectl config use-context minikube

Try querying your minkube using kubectl

> kubectl get pods
>
> kubectl get deployments

Explore the Kubernetes dashboard



> minikube dashboard

Deploy your Kubernetes application to your minikube

> kubectl create -f deployment.yml

Note the output says service and deployment are created - the two parts of your deployment file

The Kubernetes dashboard now shows new elements: 2 new pods, 1 new service and 1 new deployment

> minikube dashboard

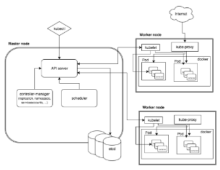
Query the kubernetes status using kubectl

> kubectl get pods
>
> kubectl get services

Microservices, Docker, and Kubernetes

Kubernetes 101: Terms

Kubernets Architecture Overview



Minikube's Tutorial

Kubectl

Kubectl CheetSheet

Minikube sets this

you need to switch

back to it in the future, run:

kubectl config use-context minikube

Access your Kubernetes application

The `--type=LoadBalancer` flag indicates that you want to expose your Service outside of the cluster. On cloud providers that support load balancers, an external IP address would be provisioned to access the Service. On Minikube, the `LoadBalancer` type makes the Service accessible through the `minikube service` command:

```
minikube service cloudlab1service
```

Our deployment says we want 5 pods. Note that Kubernetes will immediately create replacement pods if any of them die.

> Kill a pod:
>
>

Also we can scale up and down. This is very cool 🙂

> Scale up and down:
>
>

I want to run this on Google Cloud as a fully-fledged Kubernetes App, accessible to all

Create an account at cloud.google.com, which is free for a year(!)

A billing account *must* be set on the project in order to deploy the application.

.

Create a project on Google Cloud

Create a project named **cloudlab1** in your cloud.google.com account, and take a note of the **Project ID** that google assigns to it.

ⓘ **Execute the rest of the shell commands in this cloudlab in your google cloud Shell**

Google Cloud

Google Cloud

Deploy to Google Cloud

In your google cloud project, access APIs & Services to enable

- Kubernetes Engine API (used for building and managing container based applications, powered by the open source Kubernetes technology)
- Compute Engine API (Creates and runs virtual machines on Google Cloud Platform.)

Create a container cluster from within your google cloud shell

> gcloud container clusters create cloudlab1 --zone=europe-west1-b --num-nodes=3

Launch your Google Code Editor (with this link or with the pencil icon just above your google shell), to recreate the deployment.yml file you wrote earlier, but this time save them in your google cloud's root directory:

- deployment.yml,

Deploy your application to this Kubernetes cluster, with the following command from within your google cloud shell:

> kubectl apply -f deployment.yml

reached

Kubernetes 110 Your First Deployment

Deploying a stateless application

Google Cloud

Deploying a containerized web app

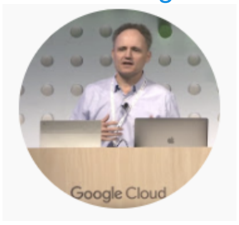
Google Cloud

Google Cloud

```
kubectl get service
```

Ping your service which now available as a Kubernetes Cluster on Google Cloud

Deploying a Web Application on Google App Engine with Manuel Amunategui



Kubernetes Engine

I want to see how I can update my app without bringing it down

Install gcloud sdk

Create a new Version and push a new Docker Image to Google Cloud (rather than Docker Hub)

Back on your laptop, modify cloudlab1.go

```
vi cloudlab1.go
docker build -t kenlomax/cloudlab1:v2 .
docker run -d -p 8080:8080 kenlomax/cloudlab1:v2
docker push kenlomax/cloudlab1:v2
```

Google Cloud SDK

CLOUD SDK

Container Registry Quickstart

Google Cloud

Using Container Registry's Docker Hub Mirror

.

Deploy a new version of your App

Kubernetes Engine's rolling update mechanism ensures that your application remains up and available even as the system replaces instances of your old container image with your new one across all the running replicas.

Apply a rolling update to the existing deployment with an image update:

e.g.

```
kubectl set image deployment mycloudlab1deployment
myapp=kenlomax/cloudlab1:v2
```

Find the EXTERNAL-IP address where your application can be reached

```
kubectl get service
```

Note the change is (soon) visible

Clean up

That's it for now 🙂

To clean up your GCP:

delete.

This will delete your project.

Additional Notes

*Minikube freezing on you on OSX?

If your Minikube freezes on you try

> minikube stop
>
> minikube delete
>
> ps -ef | grep -Ei "vbo[x]|virtualbo[x]" # visually ensure none running
>
> rm -rf ~/.minikube
>
> brew cask install macdown --verbose
>
> brew cask reinstall minikube

11 COMMENTS

Hallinan, Stephen

Can you stream to London ?

Lomax, Ken

Yes Sir. We will make it an online meeting and/or some other channel.

Lomax, Ken

Online Meeting

If you are not in Munich, **you can join Online Here.**

Unknown User (alumni.2b24c)

I want to join the meeting in Munich office. Do I have to sign up somewhere so that you know how many people will come or is it just open door ?

Lomax, Ken

At present open door. Will ping if there are updates

Unknown User (alumni.4bad0)

Fantastic tutorial! Thank you!

Lomax, Ken

> golang

Will change this to extend scratch instead

Hasan, Waseem

FYI, Extending scratch didn't work for me locally. I kept getting this error:

```
Step 1/3 : FROM scratch
 --->
Step 2/3 : ADD main /
ADD failed: stat /var/lib/docker/tmp/docker-builder865421398/main: no such file or directory
```

So I just typed in the content of the Dockerfile from the tutorial video (the one that uses golang) which worked fine.

Lomax, Ken

Hi

if you have your Dockerfile and main, main.go all in the same folder it should work:

```
C02VJ5W3HTD6:cloudlab1 d061192$ pwd
/Users/d061192/go/src/cloudlab1
C02VJ5W3HTD6:cloudlab1 d061192$ ls
Dockerfile main main.go
C02VJ5W3HTD6:cloudlab1 d061192$ docker build -t kenlomax/cloudlab1:v1 .
Sending build context to Docker daemon  6.516MB
Step 1/3 : FROM scratch
 --->
Step 2/3 : ADD main /
 ---> Using cache
 ---> 8740c00df52a
Step 3/3 : CMD ["/main"]
 ---> Using cache
 ---> e7947d264691
Successfully built e7947d264691
```

If the Dockerfile is somewhere else, you will get that error you mentioned.

ALSO, this is not tested on Windows. I suggest using a Google Cloud account to get immediate access to linux, and run the commands in there instead of Windows.

Gerstle, Christoph

Hi,

I am failing to paste "gcloud container clusters create cloudlab1 --num-nodes=3 --zone=europe-west1-b" to the gcloud console.

How is this possible?

Thanks,

Christoph

Gerstle, Christoph

Today it works! Hmm