

GKE tutorial: Get started with Google Kubernetes Engine

Discover how easy it is to create a Kubernetes cluster, add a service, configure autoscaling, and tap other great features of GKE

By Ian Pointer

InfoWorld |

JUL 30, 2018 3:00 AM PDT

Kubernetes will save us all. If only we could work out how to install and maintain it. For the uninitiated, Kubernetes (also known as K8s to your local neighborhood hipster developer) is an open-source platform for running and orchestrating container-based applications and services. These are most often deployed in Docker containers, but other container runtimes—such as Containerd and Rkt—are supported.

Google has accumulated a great deal of knowledge about running containers in their operations over the past decade and a half. Kubernetes represents the third generation of container management systems at Google, after Borg and Omega, and has emerged as the principal container platform these past few years, pushing past other offerings such as Mesos and Docker's Swarm. For the enterprise, Kubernetes offers something close to the Holy Grail: "What if OpenStack, but it actually works?"

ADVERTISEMENT

However, many people who dive feet-first into Kubernetes find themselves just a little overwhelmed. It's a system of many moving parts that can be somewhat difficult to install, maintain, and operate. But if you're running on the cloud anyhow, why not get your cloud provider to run Kubernetes for you, so you can get on with the more important business of running your applications and workloads?

With Google Kubernetes Engine (GKE), a service that has been running since 2015, your Kubernetes master servers are managed and maintained by Google Site Reliability Engineers. You don't have to install, upgrade, or even pay for them! Plus, your Kubernetes

clusters can be spread across zones within a region for high availability, and of course they integrate with the required enterprise features like Cloud IAM (Identity Access Management) and VPCs (Virtual Private Clouds) for security purposes.



Create a Kubernetes cluster

Let's dive in with an example, shall we? We'll use GKE to bring up a new cluster and run Google's hello-app Docker container as an example. First, you'll need a Google Cloud Platform account, plus the gcloud command utility. You'll also need a copy of kubectl on your system to interact with the cluster we're creating. While this little tutorial will incur charges, if you've just created a new Google Cloud account, our activity will happily fit within your initial free credit, providing you delete the cluster afterwards.

We'll authorize gcloud to work with our account (it will open a webpage to gain credentials) and create a new project in us-central1-a (adjust for the zone that's closest to you).

```
gcloud auth login
gcloud projects create kubernetes-test
gcloud config set compute/zone us-central1-a
```

Having done that, let's go crazy and create a Kubernetes cluster:

[FREE report! Learn how leading CIOs are maximizing the utility of data collected through multiple channels. Download now!]

```
gcloud containers clusters create myfirstcluster
```

And that's it. Goodnight, everybody!

Okay, apparently you want a little more than that. However, if you've ever been involved in the process of setting up a Kubernetes cluster from scratch, you know that an awful lot of things have gone on behind the scenes in that one line, saving you hours, perhaps days of time. And the cluster is up and ready in moments.

By default, gcloud creates a three-node cluster, with each node being an n1-standard-1 instance (you can alter the instance selection with the `--machine-type` parameter). Maybe not what you would use for most production workloads, but perfect for a little test cluster.

ADVERTISEMENT

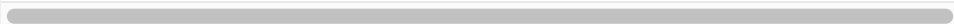
Register credentials to use kubectl

Next, we'll make gcloud register the credentials for the cluster in order to use kubectl, and then get some information about the cluster:

```
gcloud container clusters get-credentials myfirstcluster
kubectl cluster-info
```

This should give you something like this:

```
Kubernetes master is running at https://35.192.176.225
GLBCDefaultBackend is running at https://35.192.176.225/api/v1/namespaces/kube-syst
Heapster is running at https://35.192.176.225/api/v1/namespaces/kube-system/service
KubeDNS is running at https://35.192.176.225/api/v1/namespaces/kube-system/services
kubernetes-dashboard is running at https://35.192.176.225/api/v1/namespaces/kube-sy
```



Add a service to our Kubernetes cluster

We have a Kubernetes cluster running, but it's not doing much at the moment. We can change that by adding a new service to the cluster. We're going to use Google's hello-app container as a quick example. With this next command, we add the service hello-world with the 1.0 version of hello-app on port 8080:

```
kubectl run hello-world --image gcr.io/google-samples/hello-app:1.0 --port 8080
```

`kubectl get deployments` should then show you that hello-world is running. But we would like to talk to it, so let's expose it to the outside world with a load balancer:

```
kubectl expose deployment hello-world --type "LoadBalancer"
```

We can get the load balancer IP by using:

```
kubectl get service hello-server
```

You may have to wait a minute or so for the load balancer to come up. Be aware that this load balancer is a standard Google load balancer that will incur charges on your Google account, though again, for this short example, those charges will amount to pennies at most. Once the external IP is visible, you can go to the address in your browser or use curl:

```
curl 35.184.69.149:8080  
Hello, world!  
Version: 1.0.0
```

Create an autoscaler for our Kubernetes service

We can then set some scaling policies. How about running between 10 and 20 copies in the cluster, scaling up on CPU utilization of 50 percent? Easy!

```
kubectl autoscale deployment hello-world --max 20 --min 10 --cpu-percent 50
```

You can run `kubectl get pod` to see all of the various pods that GKE is spinning up to satisfy the new autoscaling policy. (A *pod* in Kubernetes is a group of one or more containers. It is the smallest unit of compute that the system deals with.) Within seconds, you'll see all 10 pods running happily.

Finally, let's do something quite cool (at least if you've spent time in the ops trenches). GKE provides support for node auto-repair, allowing Kubernetes to respond to issues with nodes in the cluster. Let's switch this on for our test cluster:

```
gcloud container node-pools update default-pool --enable-autorepair
```

Next, run `kubectl get nodes`. Everything should look nice and happy, with “Ready” statuses everywhere. Let’s change that. Go into the Google Cloud Dashboard and do something that most ops people dread: Randomly delete a VM from the cluster. Or throw caution to the wind! Delete two, or even all three!

ADVERTISEMENT

After you delete as many VMs as you like, keep running `kubectl get nodes` and `kubectl get pods`. You should see that any nodes you deleted have been marked as “NotReady,” and some pods may have gone offline (because they were running on that node). However, if you keep watching, you’ll see the cluster bounce back. First the nodes will return (you can verify this in the dashboard), and then the pods will come back as well.

GKE features and advantages

Aside from node auto-repair, running your Kubernetes cluster in GKE has additional benefits. Node auto-upgrade allows your nodes to be automatically upgraded, so you know you’re always running the most secure and stable version of Kubernetes recommended by the GKE team without having to do it yourself.

ADVERTISEMENT

By default, every node in your cluster runs Container-Optimized OS, Google’s own custom, slimmed-down, and hardened version of Linux. While you can use Ubuntu instead, running Container-Optimized OS will keep you on a secure footing maintained by Google itself.

Another useful feature is the ability to leverage Google’s vast networking capabilities. With minimal effort, you can set up Kubernetes clusters across the world and configure load balancing so that when users access one of your services, they are directed to the cluster that is geographically closest to them.

ADVERTISEMENT

GKE also integrates with Google’s Stackdriver offering, allowing logging and monitoring of your clusters to be handled with ease. A great feature of Stackdriver is that it is platform agnostic, so if you have Kubernetes clusters on-premises, or on AWS, then you can integrate those with Stackdriver too. Stackdriver lets you drill down into services, workloads, pods, and even down to the container level to gain insights on metrics like latency, failure rates, and anything else important to the condition of your enterprise.

Finally, as you might expect, GKE integrates seamlessly with other Google Cloud Platform services, so if you want to create a pool of nodes running on pre-emptible VMs with access to high-end GPUs, just go for it!

ADVERTISEMENT

The front-runner in managed Kubernetes

Given Kubernetes' Google lineage and GKE's three-year head start on Azure Kubernetes Service or Amazon EKS, it's not entirely surprising that GKE is ahead of the other major managed Kubernetes offerings in terms of support for Kubernetes features. (What *is* perhaps surprising is that Azure's Kubernetes support is arguably ahead of Amazon's, but that's a story for another time.)

One of the great promises of Kubernetes is that it brings us a giant step closer to a world where you can lift your workload from one cloud provider to another with a minimum of effort. For now, though, running on GKE is one of the easiest ways to hop on the Kubernetes train.

ADVERTISEMENT

Ian Pointer is a senior big data and deep learning architect, working with Apache Spark and PyTorch. He has more than 15 years of development and operations experience.

Follow  

Copyright © 2018 IDG Communications, Inc.

How to choose a low-code development platform

SPONSORED LINKS

Get started with AWS AppConfig

Every second counts when it comes to mitigating cyberattacks and resolving network performance issues. NETSCOUT Visibility Without Borders keeps you one step ahead.

CIS Webinar: Effective Implementation of the CIS Benchmarks & CIS Controls