

This tutorial shows you how to run JanusGraph on Google Cloud (GCP). [JanusGraph](http://janusgraph.org/) (<http://janusgraph.org/>) is a graph database that supports working with large amounts of data. Graph databases help you to discover insights by modeling your data entities and the relationships between them. In graph terminology, entities are known as *nodes* or *vertices* and relationships are known as *edges*. Both vertices and edges can have associated properties.

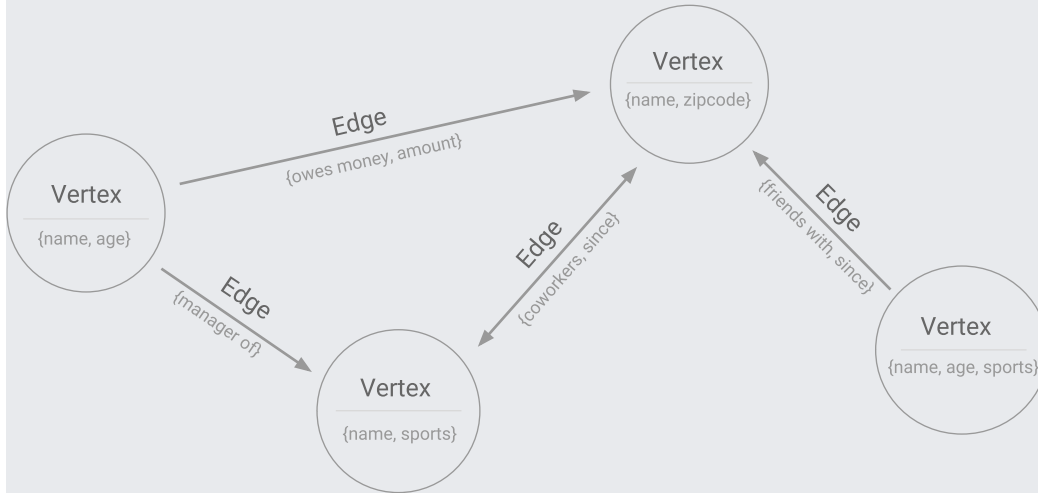


Figure 1. Example of a property graph

Graph databases help you model a variety of domains and activities:

- Social networks
- Fraud analysis
- Physical networks

When creating graph databases, you sometimes create millions or even billions of vertices and edges. When you use JanusGraph with [Bigtable](https://cloud.google.com/bigtable/) ([/bigtable/](https://cloud.google.com/bigtable/)) as the underlying storage layer, you can both execute fast queries and scale your storage layer independently for the size and throughput that you need. Use this tutorial to deploy a scalable JanusGraph infrastructure with Bigtable, which you can then use to traverse the relationships that exist in any graph database.

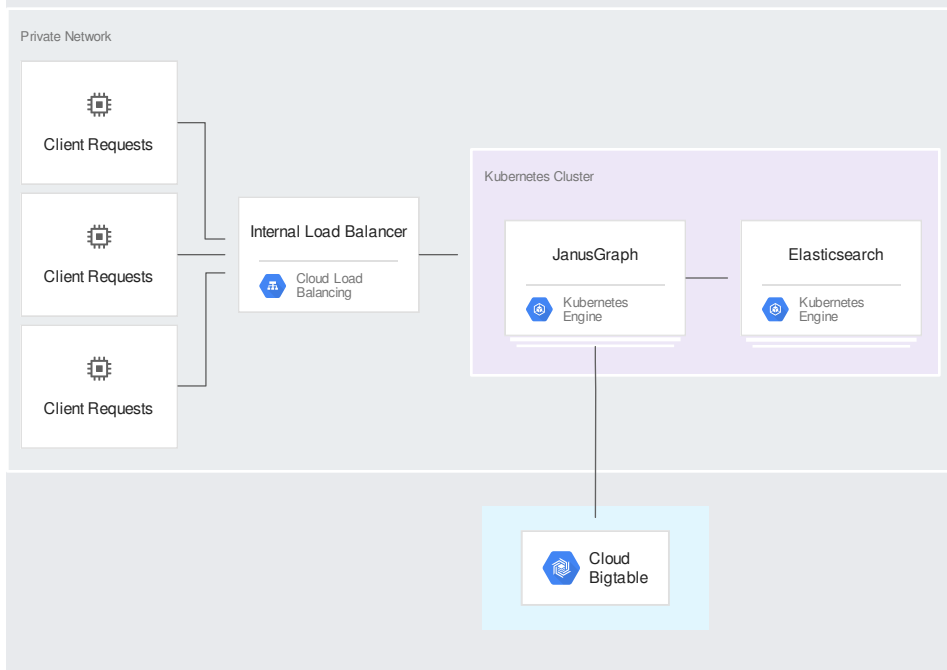


Figure 2. JanusGraph deployment with Bigtable on GKE

- Create a [Bigtable instance](#) (/bigtable/docs/creating-compute-instance).
- Create a [GKE](#) (/kubernetes-engine/) cluster.
- Install [Helm](https://helm.sh/) (https://helm.sh/), a Kubernetes package manager.
- Use a Helm chart to deploy JanusGraph and [Elasticsearch](https://www.elastic.co/) (https://www.elastic.co/).
- Use [Gremlin](http://tinkerpop.apache.org/gremlin.html) (http://tinkerpop.apache.org/gremlin.html) and connect to JanusGraph.
- Load and then query data.

This tutorial uses the following billable components of Google Cloud:

- Compute Engine, which is used by GKE
- Bigtable

To generate a cost estimate based on your projected usage, use the [pricing calculator](#) (/products/calculator). New Google Cloud users might be eligible for a [free trial](#) (/free-trial).

1. [Sign in](https://accounts.google.com/Login) (https://accounts.google.com/Login) to your Google Account.
If you don't already have one, [sign up for a new account](https://accounts.google.com/SignUp) (https://accounts.google.com/SignUp).
2. In the Cloud Console, on the project selector page, select or create a Cloud project.

★ **Note:** If you don't plan to keep the resources that you create in this procedure, create a project instead of selecting an existing project. After you finish these steps, you can delete the project, removing all resources associated with the project.

[Go to the project selector page](https://console.cloud.google.com/projectselector2/home/dashboard) (https://console.cloud.google.com/projectselector2/home/dashboard)

3. Make sure that billing is enabled for your Google Cloud project. [Learn how to confirm billing is enabled for your project](#) (/billing/docs/how-to/modify-project).
4. Enable the Bigtable, Bigtable Admin, Compute Engine, and GKE APIs.

[Enable the APIs](https://console.cloud.google.com/flows/enableapi?apiid=bigtable,bigtableadmin.googleapis.com,compute.googleapis.com,container.goog) (https://console.cloud.google.com/flows/enableapi?apiid=bigtable,bigtableadmin.googleapis.com,compute.googleapis.com,container.goog)

When you finish this tutorial, you can avoid continued billing by deleting the resources that you created. See [Cleaning up](#) (#cleaning_up) for more detail.

In this tutorial, you use [Cloud Shell](#) (/shell/docs/overview) to enter commands. Cloud Shell gives you access to the command line in the Cloud Console and includes Cloud SDK and other tools that you need to develop in GCP. Cloud Shell appears as a window at the bottom of the Cloud Console. It can take several minutes to initialize, but the window appears immediately.

1. Activate Cloud Shell.

[ACTIVATE Cloud Shell](https://console.cloud.google.com/?cloudshell=true) (https://console.cloud.google.com/?cloudshell=true)

2. In Cloud Shell, set the default Compute Engine zone to the zone where you are going to create your Bigtable cluster and GKE cluster. This tutorial uses `us-central1-f`.
3. Create a GKE cluster to deploy JanusGraph:
4. [Install Helm](https://docs.helm.sh/using_helm/#installing-helm) (https://docs.helm.sh/using_helm/#installing-helm) in your Cloud Shell environment:

For the JanusGraph storage backend, this tutorial uses Bigtable, which can scale rapidly to meet your needs. This tutorial uses a single-node development cluster, which is both economical and sufficient for the tutorial. You can start your projects with a development cluster and then move to a larger production cluster when you are ready to work with production data. The Bigtable documentation includes detailed discussion about performance and scaling to help you pick a cluster size for your own work.

Create your Bigtable instance by following these steps.

1. In the Cloud Console, go to the **Create Instance** page:

[GO TO THE CREATE INSTANCE PAGE](https://console.cloud.google.com/bigtable/create-instance) (https://console.cloud.google.com/bigtable/create-instance)

2. In the **Instance name** box, enter a name for your instance. You can use `janusgraph` or another lowercase name of your choosing. The page automatically sets **Instance ID** and **Cluster ID** after you enter your instance name.
3. Set **Instance type** to **Development**.
4. Under **Zone**, select `us-central1-f` or the zone where you created your GKE cluster earlier.
5. Click **Create** to create the instance.

Make note of the instance ID, because you will use it in an upcoming step.

You use Helm to deploy applications to your Kubernetes cluster. After creating your cluster, configure Helm to work with the cluster.

1. Paste the following commands into Cloud Shell:

You might see the output `Waiting for tiller install...` a few times, but when it stops, you can use Helm with your Kubernetes cluster.

In addition to using Bigtable as its storage backend, JanusGraph will use Elasticsearch as the [indexing backend](https://docs.janusgraph.org/index-backend/elasticsearch/) (<https://docs.janusgraph.org/index-backend/elasticsearch/>).

In this section, you use a [Helm chart](https://helm.sh/docs/topics/charts/) (<https://helm.sh/docs/topics/charts/>) to deploy [JanusGraph](https://hub.helm.sh/charts/stable/janusgraph) (<https://hub.helm.sh/charts/stable/janusgraph>) and [Elasticsearch](https://hub.helm.sh/charts/elastic/elasticsearch) (<https://hub.helm.sh/charts/elastic/elasticsearch>) to your Kubernetes cluster. When you install the JanusGraph chart, Elasticsearch is included as a dependency, which simplifies the process.

1. In Cloud Shell, set an environment variable to hold the value of the Bigtable instance ID that you noted earlier. Replace `[YOUR_INSTANCE_ID]` with the instance ID you specified earlier.

For example, if you used the default suggestion of `janusgraph` for your Bigtable instance ID, you would run:

2. Create a `values.yaml` file, which supplies Helm with the specific configuration to use when installing JanusGraph:

3. Deploy the JanusGraph Helm chart by using the `values.yaml` file that you created:

The install waits until all of the resources are ready before it completes. This process might take several minutes.

When the `helm install` command finishes, it outputs a `NOTES` section that describes a getting started experience. From Cloud Shell, follow the steps that the `NOTES` section outlines to test if your JanusGraph environment is working.

1. Set an environment variable with the name of a Kubernetes pod that is running JanusGraph:
2. Connect to the pod and run the Gremlin shell:
3. In the Gremlin console, connect to the Apache TinkerPop server:

The output looks similar to the following:

4. Run the following Gremlin commands to create two vertices and an edge:

The output looks similar to the following:

5. Issue a Gremlin query to see what the label is for the vertex that follows an edge out from the vertex with the label `hello`:

The query syntax is explained in the next section. For now, you see the word "world" as the output from the query:

Now that you have deployed JanusGraph and can connect to it by using Gremlin, you can begin loading and querying your own data. To help demonstrate what that process looks like, load the sample dataset that comes bundled with JanusGraph: the [Graph of the Gods](https://docs.janusgraph.org/#getting-started) (<https://docs.janusgraph.org/#getting-started>), which depicts mythological deities and their location properties.

1. While you are still in the Gremlin shell from the last section, enter the following command:

When the command completes, it returns `null`:

2. With the sample graph loaded, you can issue graph traversal queries. For example, to find all brothers of Jupiter, enter the following query:

You can break down this query by looking at the steps that it traverses:

Traversal step	Explanation
<code>g.V()</code>	Start with the collection of vertices.
<code>has('name', 'jupiter')</code>	Find one that has the property <code>name</code> with the value of <code>jupiter</code> .
<code>out('brother')</code>	From there, follow any edges that are labeled <code>brother</code> .
<code>values('name')</code>	For the vertices where those edges lead, get the <code>name</code> property.

Here's the output of the query:

To get more familiar with the traversal queries that are possible on this Graph of the Gods dataset, try out other sample queries from the [JanusGraph docs](https://docs.janusgraph.org/#global-graph-indices) (<https://docs.janusgraph.org/#global-graph-indices>).

To avoid incurring charges to your Google Cloud Platform account for the resources used in this tutorial:




Caution: Deleting a project has the following effects:

- **Everything in the project is deleted.** If you used an existing project for this tutorial, when you delete it, you also delete any other work you've done in the project.
- **Custom project IDs are lost.** When you created this project, you might have created a custom project ID that you want to use in the future. To preserve the URLs that use the project ID, such as an `appspot.com` URL, delete selected resources inside the project instead of deleting the whole project.

1. In the Cloud Console, go to the **Manage resources** page.

[Go to the Manage resources page \(https://console.cloud.google.com/iam-admin/projects\)](https://console.cloud.google.com/iam-admin/projects)

2. In the project list, select the project you want to delete and click **Delete** .

3. In the dialog, type the project ID, and then click **Shut down** to delete the project.

- Read more about [JanusGraph](http://janusgraph.org/) (<http://janusgraph.org/>) and [graph databases](https://wikipedia.org/wiki/Graph_database) (https://wikipedia.org/wiki/Graph_database).
- Learn about [Apache TinkerPop](http://tinkerpop.apache.org/) (<http://tinkerpop.apache.org/>), and explore the [Gremlin](http://tinkerpop.apache.org/gremlin.html) (<http://tinkerpop.apache.org/gremlin.html>) graph traversal language.
- Dive deeper into graph uses cases by deploying an [example JanusGraph application](/blog/products/gcp/developing-a-janusgraph-backed-service-on-google-cloud-platform) (</blog/products/gcp/developing-a-janusgraph-backed-service-on-google-cloud-platform>).
- Try out other Google Cloud features for yourself. Have a look at our [tutorials](/docs/tutorials) (</docs/tutorials>).