Using PostgreSQL on Google App Engine

Introduction

PostgreSQL is a powerful open source object-relational database system. It has more than 15 years of active development and a proven architecture that has earned it a strong reputation for reliability, data integrity, and correctness. It is fully ACID compliant, has full support for foreign keys, joins, views, triggers, and stored procedures (in multiple languages).

In this article, we show you how to install PostgreSQL using Cloud Launcher and access the database instance from App Engine.

Installing PostgreSQL Using Cloud Launcher

1. Open the Cloud Launcher for PostgreSQL, then click **Launch on Google Cloud Platform**.

2. In the Google Developers Console, select a project (or create a new project), then click **Continue**.

3. On the VM Instances page, specify settings for the PostgreSQL virtual machine.

    - Machine type, Boot disk, and Zone (geographic zone) cannot be changed after the VM has been created, so you should specify settings that meet your resource requirements.

    - You can specify firewall rules and project API access. These settings can be edited after the VM has been created, and are discussed in more detail in the next section.

    - You can also add additional disks after the VM has been created. For better performance and data safety, you can use this first disk (the boot disk) as the PostgreSQL database engine, and then set up the data storage on a separate persistent disk.

4. Click **Create**. A progress indicator appears, and a confirmation message is displayed when the PostgreSQL instance has been successfully deployed.

    - Click **Manage the VM Instance** to view usage and edit settings.

- Click **SSH** to open an SSH client in a new browser window. To use another SSH client, click the menu icon next to the **SSH** button, then select **Use another SSH client** to display standard SSH connection information.

Configuring the PostgreSQL VM Access Settings

About App Engine Networks

All App Engine projects include a default network. If you do not explicitly assign your instances to a network, all instances added to the project are automatically assigned to this default network. An instance can be assigned to only one network.

A network performs the same function that a router does: it defines the network range and gateway IP address, handles communication between instances, and serves as a gateway between instances and other networks. A network is constrained to a single project; it cannot span projects. Any communication between instances in different networks, even within the same project, must be through external IP addresses. In the API, a network is represented by the Network resource.

The PostgreSQL Compute Engine instance belongs to the private network of the project specified when the instance is created. App Engine Managed VMs are also Compute Engine instances, and can therefore belong to the same project and private network as a PostgreSQL Compute Engine instance. Applications running on standard App Engine instances will be on separate networks, and therefore would need to make requests to the PostgreSQL database over the public internet.

To configure access settings:

1. Select **VM instances** in the Google Developer Console, then select the PostgreSQL instance.

2. Click **Edit** at the top of the page.

3. Update the PostgreSQL instance settings, then click **Save** at the bottom of the page to save your changes.

Access Settings

- Firewall -- The default firewall rules do not allow HTTP or HTTPS connections to your PostgreSQL instance. You can use these check boxes to add a rule that allows HTTP or HTTPS access. Note that an instance must have an external IP address before it can receive traffic from outside its network. By default all incoming traffic from outside a network is blocked.

- Project access -- Select this check box to allow API access to all Google Cloud services in the same project. For more information, see Connecting to Other Google Cloud Platform Services

Using Python to Connect to PostgreSQL

You can either of the following PostgreSQL packages to connect to a PostgreSQL instance using Python:

- psycopg2 -- this package contains the `psycopg2` module.
- PyGreSQL -- this package contains the `pgdb` module.

Set Up an Isolated Python Environment and Install a PostgreSQL Package

1. On the VM Instances page, click **SSH** to open an SSH client for the applicable App Engine instance.

2. If you don't have virtualenv, you can install it using pip.

   ```
   sudo pip install virtualenv
   ```

3. Run the following commands to create an isolated Python environment.

   ```
   virtualenv env
   source env/bin/activate
   ```

4. Install the package.

   - Use the following command to install the `psycopg2` package:

     ```
     pip install psycopg2
     ```

   - Use the following command to install the `PyGreSQL` package:

     ```
     pip install pygresql
     ```

Connecting to PostgreSQL Using psycopg2

The following example will connect to the PostgreSQL database and print a connection error if one occurs.

```
#!/usr/bin/python
import psycopg2

try:
    conn = psycopg2.connect("dbname='db_name' user='db_user'
host='localhost' password='db_password'")
except:
    print "Cannot connect to the database"
```

Connecting to PostgreSQL Using pgdb

The following example will connect to the PostgreSQL database and print a connection error if one occurs.

```
#!/usr/bin/python
import pg

try:
    conn = pg.connect("dbname='db_name' user='db_user'
host='localhost' password='db_password'")
except:
    print "Cannot connect to the database"
```

Best practices

For better performance and data safety, install the database engine on the boot disk, and then set up the data storage on a separate persistent disk. To learn how to add a disk for your database, see How to Set Up a New Persistent Disk for PostgreSQL Data.

This tutorial provided you with a basic look at a one-machine, single-disk installation of PostgreSQL. In a production environment, it's a good idea to employ strategies for high availability, scalability, archiving, backup, load balancing, and disaster recovery. To learn more about disaster recovery planning, see How to Design a Disaster Recover Plan. For information about setting up high availability, see How to Set Up PostgreSQL for High Availability and Replication with Hot Standby.

App Engine instances are autoscaling, but this single-disk PostgreSQL Compute Engine instance is not. However, you can use managed instance groups to set up autoscaling for a PostgreSQL Compute Engine instance. Autoscaling is a feature of managed instance groups. A managed instance group is a pool of homogeneous instances, created from a common instance

template. An autoscaler adds or remove instances from a managed instance group. Although Compute Engine has both managed and unmanaged instance groups, only managed instance groups can be used with autoscaler.