# **Multi-Tenant System for Sales and Distribution**

**Amazon Web Service**

Amazon is a IAAS infrastructure as a service that provides virtualized cloud computing resources over the internet.

To use amazon relational database we need AWS elastic Beanstalk.

AWS Elastic Beanstalk provides support for [running Amazon Relational Database Service Amazon RDS](https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/using-features.managing.db.html)

To decouple your database instance from your environment, you can run a database instance in Amazon RDS and configure your application to connect to it on launch. This enables you to connect multiple environments to a database, terminate an environment without affecting the database

**AWS Elastic Beanstlk environment configuration for RDBS**

Here are few steps you can follow to configure environment properties for AWS relational database

1. Open the [Elastic Beanstalk console](https://console.aws.amazon.com/elasticbeanstalk).
2. Navigate to the [management page](https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/environments-console.html) for your environment.
3. Choose Configuration.
4. On the Software configuration card, choose Modify.
5. In the Environment properties section, define the variables that your application reads to construct a connection string. For compatibility with environments that have an integrated RDS DB instance, use the following.
   * **RDS\_HOSTNAME** – The hostname of the DB instance.

Amazon RDS console label – **Endpoint** (this is the hostname)

* + **RDS\_PORT** – The port on which the DB instance accepts connections. The default value varies among DB engines.

Amazon RDS console label – **Port**

* + **RDS\_DB\_NAME** – The database name, **ebdb**.

Amazon RDS console label – **DB Name**

* + **RDS\_USERNAME** – The user name that you configured for your database.

Amazon RDS console label – **Username**

* + **RDS\_PASSWORD** – The password that you configured for your database.


        Environment Properties section with RDS properties added
      

# Adding an Amazon RDS DB Instance to your Node.js Application Environment

Downloading driver

{

"name": "my-app",

"version": "0.0.1",

"private": true,

"dependencies": {

"ejs": "latest",

"aws-sdk": "latest",

"express": "latest",

"body-parser": "latest",

**"mssql": "latest"**

},

"scripts": {

"start": "node app.js"

}

}

**Common Driver Packages for Node.js**

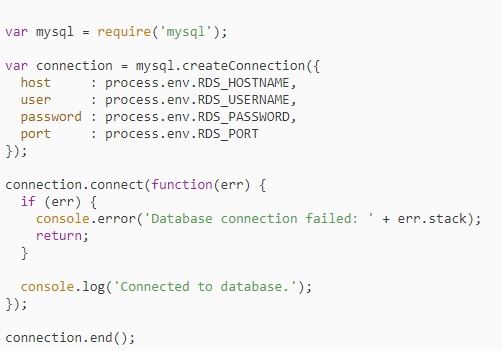
* **MySQL** – mysql
* **PostgreSQL** – pg
* **SQL Server** – mssql
* **Oracle** – oracle or oracledb

The Oracle package and version depend on the Node.js version you're using:

* + **Node.js 6.x, 8.x** – Use the latest version of oracledb.
  + **Node.js 4.x** – Use the oracledb version 2.2.0.
  + **Node.js 5.x, 7.x** – Use the latest version of oracle. The oracledb package doesn't support these Node.js versions.

## Connecting to a Database

Elastic Beanstalk provides connection information for attached DB instances in environment properties. Use os.environ['*VARIABLE*'] to read the properties and configure a database connection.



# Adding an Amazon RDS DB Instance to Your .NET Application Environment

## Downloading a Driver

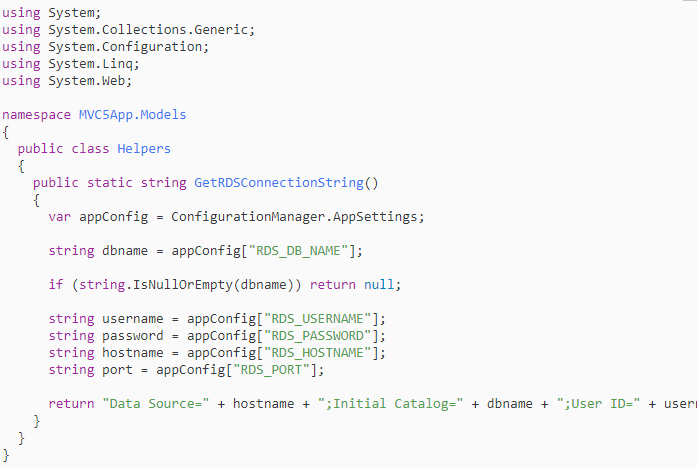
Download and install the EntityFramework package and a database driver for your development environment with NuGet.

**Common Entity Framework Database Providers for .NET**

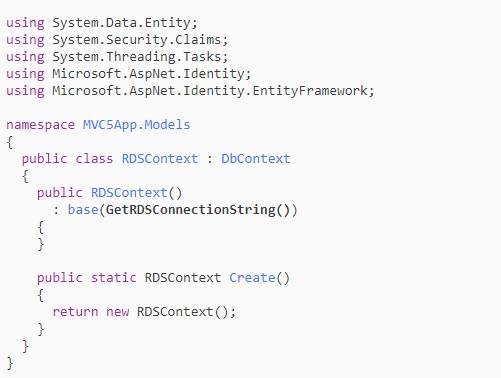
* **SQL Server** – Microsoft.EntityFrameworkCore.SqlServer
* **MySQL** – Pomelo.EntityFrameworkCore.MySql
* **PostgreSQL** – Npgsql.EntityFrameworkCore.PostgreSQL

## Connecting to a Database

Elastic Beanstalk provides connection information for attached DB instances in environment properties. Use ConfigurationManager.AppSettings to read the properties and configure a database connection.



Use the connection string to initialize your database context.

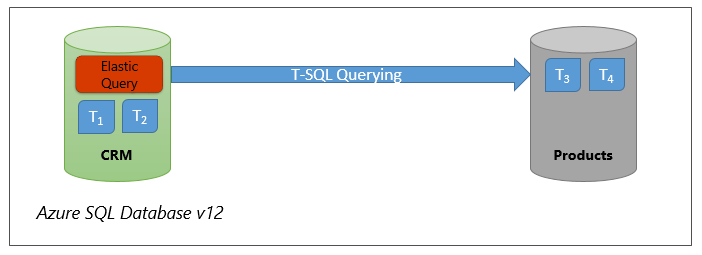


Cloud Computing is when you access computing services – like server, storage, networking, software – over the internet from a provider like Microsoft Azure, Amazon Web Service, Google Cloud Platform. Cloud Computing platforms, like Azure tend to be less expensive and more secure, reliable and flexible.

**Microsoft Azure**

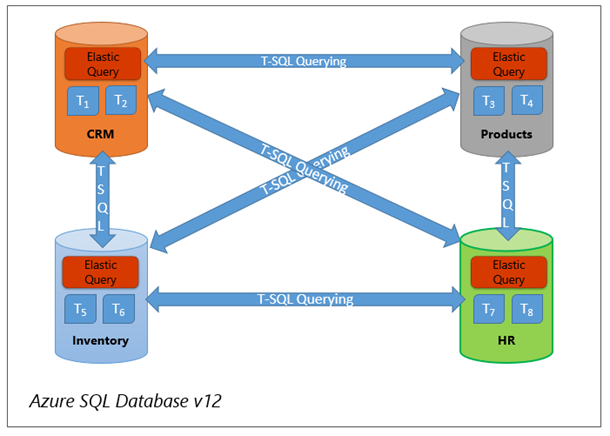
**Microsoft Azure** is a [cloud computing](https://en.wikipedia.org/wiki/Cloud_computing) service created by [Microsoft](https://en.wikipedia.org/wiki/Microsoft) for building, testing, deploying, and managing applications and services through Microsoft-managed [data centers](https://en.wikipedia.org/wiki/Data_center). It provides [software as a service (SaaS)](https://en.wikipedia.org/wiki/Software_as_a_service), [platform as a service (PaaS)](https://en.wikipedia.org/wiki/Platform_as_a_service) and [infrastructure as a service (IaaS)](https://en.wikipedia.org/wiki/Infrastructure_as_a_service) and supports many different [programming languages](https://en.wikipedia.org/wiki/Programming_language), tools and frameworks, including both Microsoft-specific and third-party software and systems.

Most notably, elastic database query now supports querying across databases in Azure SQL Database. This makes possible common cross-database querying tasks like selecting from a remote table into a local table.



Elastic database query now provides access to tables in remote Azure SQL Databases through a simple extension in the DDL for external data sources and external tables. You can define an external data source that, for instance, provides access to a remote database which stores reference data shared among all databases of your data tier. You can also easily copy the contents of tables from a remote database to another using INSERT INTO... SELECT statement.

It also allows for richer remote database querying topologies like the one illustrated in the following figure where a number of databases need access to each other’s tables.



The extensible APIs that you can use include .NET, .NET Core, Node.js, Java, Python, and MongoDB. If you’re using .NET, you can use DocumentDB API, MongoDB API, Graph API, or Table API. Java and Node.js will only work with the Document DB APIs, MongoDB APIs, and Graph APIs.

We’re excited to share that we can now use the global cloud database [MongoDB Atlas](https://www.mongodb.com/cloud/atlas/azure-mongodb) for free on Microsoft Azure. The newly available free tier on Azure is known as the M0, and grants users 512 MB of storage which is ideal for learning MongoDB, prototyping, and early development.

Cloud based Multitenant for Sales and Distribution

Node js Platform as a service

* Node.js-related software for your operating system:
  + **MacOS**, install Homebrew and Node.js, then install the ODBC driver and SQLCMD. See [Step 1.2 and 1.3](https://www.microsoft.com/sql-server/developer-get-started/node/mac/).
  + **Ubuntu**, install Node.js, then install the ODBC driver and SQLCMD. See [Step 1.2 and 1.3](https://www.microsoft.com/sql-server/developer-get-started/node/ubuntu/).
  + **Windows**, install Chocolatey and Node.js, then install the ODBC driver and SQLCMD. See [Step 1.2 and 1.3](https://www.microsoft.com/sql-server/developer-get-started/node/windows/).

**Get SQL server connection information**

Get the connection information you need to connect to the Azure SQL database. You'll need the fully qualified server name or host name, database name, and login information for the upcoming procedures.

1. Sign in to the [Azure portal](https://portal.azure.com/).
2. Navigate to the **SQL databases** or **SQL managed instances** page.
3. On the **Overview** page, review the fully qualified server name next to **Server name** for a single database or the fully qualified server name next to **Host** for a managed instance. To copy the server name or host name, hover over it and select the **Copy** icon.

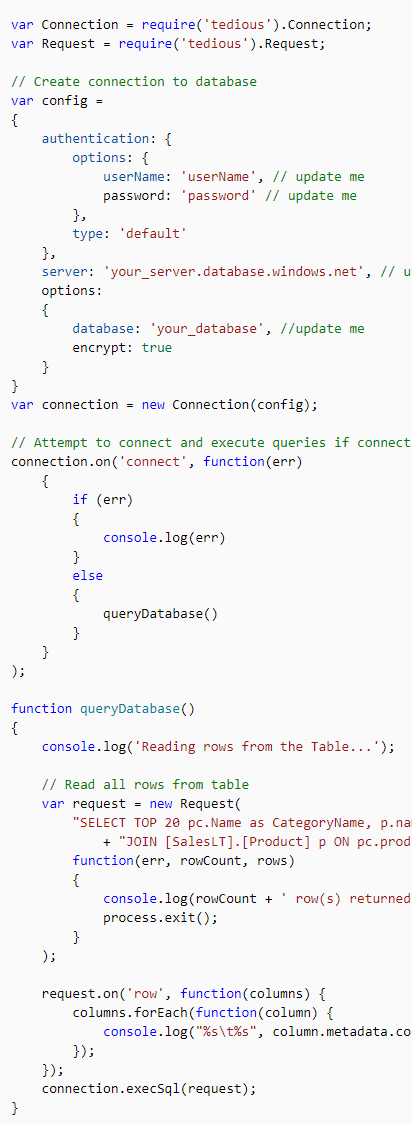
**Create the project**

Open a command prompt and create a folder named *sqltest*. Navigate to the folder you created and run the following command:

1. npm init -y
2. npm install tedious@5.0.3
3. npm install async@2.6.2

**Add code to query database**

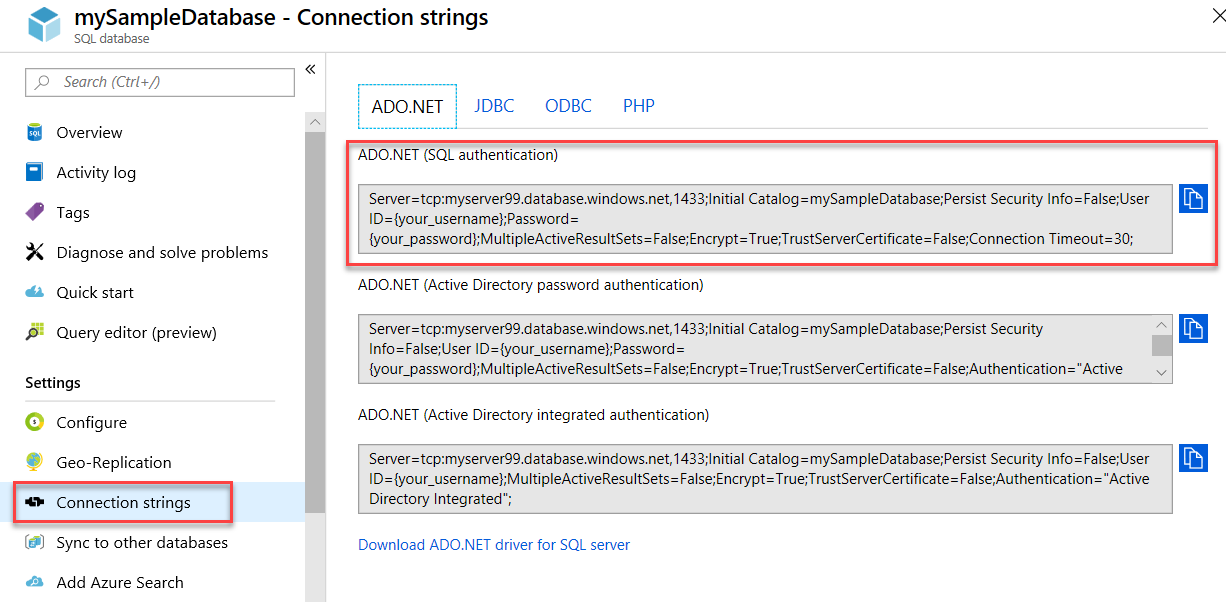
1. In your favorite text editor, create a new file, *sqltest.js*.
2. Replace its contents with the following code. Then add the appropriate values for your server, database, user, and password.



node sqltest.js

## Get ADO.NET connection information (optional)

1. Navigate to the **mySampleDatabase** page and, under **Settings**, select **Connection strings**.
2. Review the complete **ADO.NET** connection string.



1. Open a command prompt and create a folder named **sqltest**. Navigate to this folder and run this command.

cmdCopy

dotnet new console

This command creates new app project files, including an initial C# code file (**Program.cs**), an XML configuration file (**sqltest.csproj**), and needed binaries.

1. In a text editor, open **sqltest.csproj** and paste the following XML between the <Project>tags. This XML adds System.Data.SqlClient as a dependency.

XMLCopy

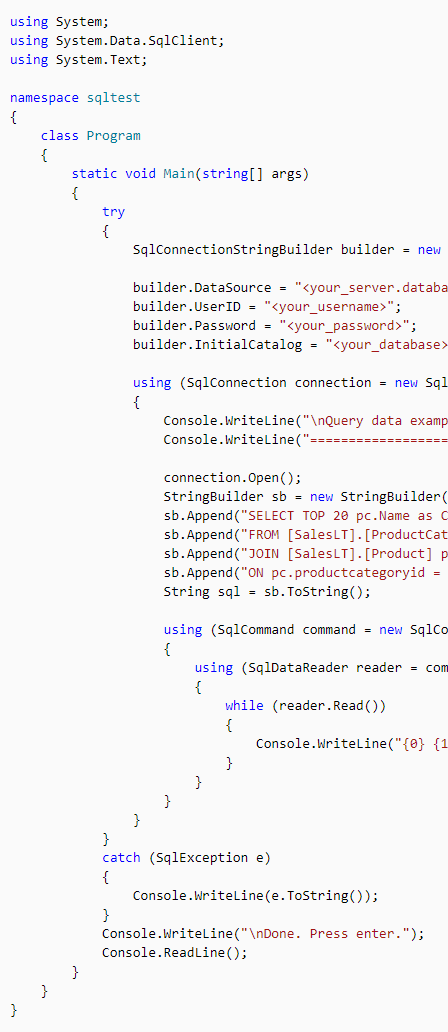
<ItemGroup>

<PackageReference Include="System.Data.SqlClient" Version="4.6.0" />

</ItemGroup>

**Insert code to query SQL database**

1. In a text editor, open **Program.cs**.
2. Replace the contents with the following code and add the appropriate values for your server, database, username, and password.



**Google Cloud Platform**

Google Cloud Platform, offered by Google, is a suite of cloud computing services that runs on the same infrastructure that Google uses internally for its end-user products, such as Google Search and YouTube.

**Google Cloud Platform Database Service Cloud SQL**

Cloud SQL is a fully-managed database service that makes it easy to set up, maintain, manage, and administer your relational databases on Google Cloud Platform.

You can use Cloud SQL with either [MySQL](https://cloud.google.com/sql/docs/mysql/) or [PostgreSQL](https://cloud.google.com/sql/docs/postgres/). Click the card below for more information.

Not sure what storage option is right for you? Check out the [Choosing a Storage Option](https://cloud.google.com/storage-options/) guide.

### **Features**

* Fully managed MySQL Community Edition databases in the cloud.
* Second Generation instances support MySQL 5.6 or 5.7, and provide up to 416 GB of RAM and 10 TB data storage, with the option to automatically increase the storage size as needed.
* First Generation instances support MySQL 5.5 or 5.6, and provide up to 16 GB of RAM and 500 GB data storage.

**Note:**Second Generation is replacing First Generation; support for First Generation instances ends January 30, 2020. To upgrade a First Generation instance to Second Generation, see [Upgrading a First Generation Instance to Second Generation](https://cloud.google.com/sql/docs/mysql/upgrade-2nd-gen).

* Create and manage instances in the [Google Cloud Platform Console](https://console.cloud.google.com/).
* Instances available in US, EU, or Asia.
* Customer data encrypted on Google’s internal networks and in database tables, temporary files, and backups.
* Support for secure external connections with the Cloud SQL Proxy or with the SSL/TLS protocol.
* Support for private IP (private services access).
* Data replication between multiple zones with automatic failover.
* Import and export databases using mysqldump, or import and export CSV files.
* Support for MySQL wire protocol and standard MySQL connectors.
* Automated and on-demand backups, and point-in-time recovery.
* Instance cloning.
* Integration with Stackdriver logging and monitoring.
* [ISO/IEC 27001](http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=54534) compliant.

### **Supported languages**

You can use Cloud SQL for MySQL with App Engine applications that are written in Java, Python, PHP, Node.js, Go, and Ruby. You can also use Cloud SQL for MySQL with external applications using the standard MySQL protocol.

### **How you can connect to Cloud SQL for MySQL instances**

You can connect to a Cloud SQL instance for MySQL from:

* A mysql client. [Learn more](https://cloud.google.com/sql/docs/mysql/connect-admin-ip).
* Third-party tools like SQL Workbench or Toad for MySQL. [Learn more](https://cloud.google.com/sql/docs/mysql/admin-tools).
* External applications. [Learn more](https://cloud.google.com/sql/docs/mysql/connect-external-app).
* App Engine applications. [Learn more](https://cloud.google.com/sql/docs/mysql/connect-app-engine).
* Applications running on Compute Engine. [Learn more](https://cloud.google.com/sql/docs/mysql/connect-compute-engine).
* Applications running on Google Kubernetes Engine. [Learn more](https://cloud.google.com/sql/docs/mysql/connect-kubernetes-engine).
* Cloud Functions. [Learn more](https://cloud.google.com/sql/docs/mysql/connect-cloud-functions).
* Google Apps Script scripts [Learn more](https://developers.google.com/apps-script/jdbc).

Connecting to Cloud SQL by using Private Google access is not supported. Private services access is supported. For more information, see [Private Access Options for Services](https://cloud.google.com/vpc/docs/private-access-options).

### **Differences between Cloud SQL and standard MySQL functionality**

In general, the MySQL functionality provided by a Cloud SQL instance is the same as the functionality provided by a locally-hosted MySQL instance. However, there are a few differences between a standard MySQL instance and a Cloud SQL for MySQL instance.

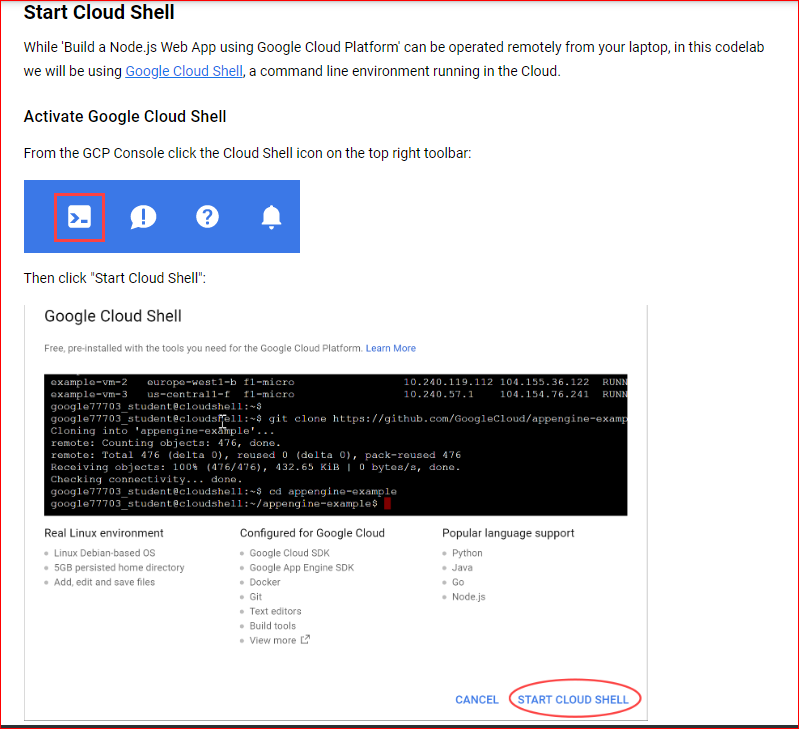
# **Cloud SQL Pricing**

This is normally paid service. But we try to find the free service for students of educators.

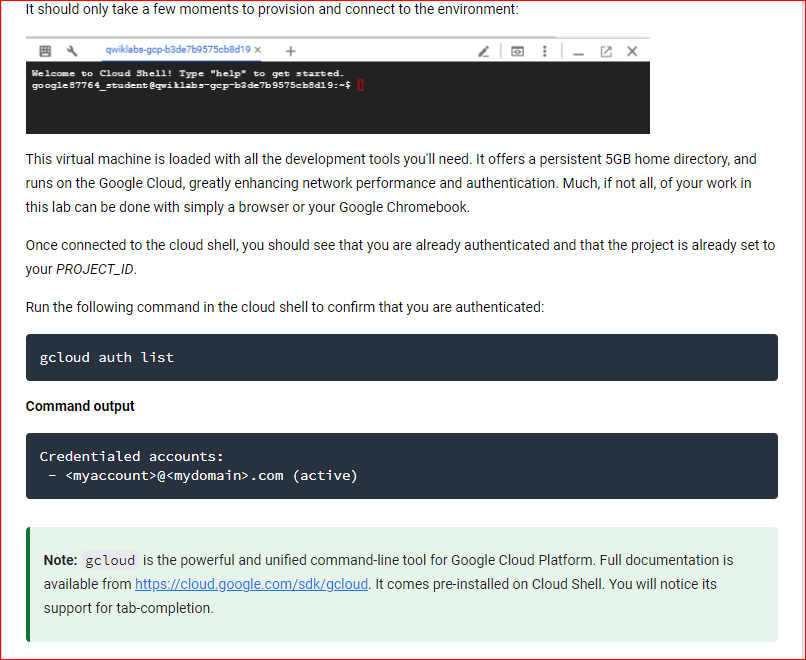
**Google Cloud Platform Database Service configuration for Node.js**

**Step-1:**

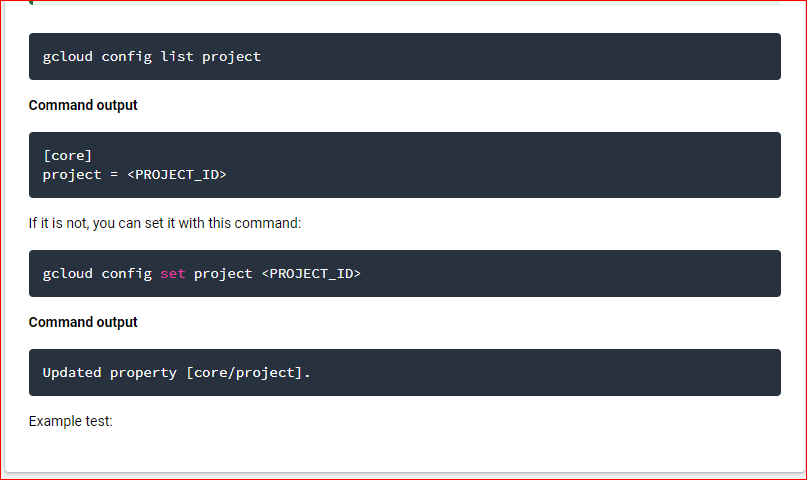
## **Create an App Engine app**



**Step-2:**



**Step-3:**

****