# add python to git path

$ ***echo 'export PATH="$PATH:/c/Python27"' > .bashrc***

# Create environment variable for correct distribution  
export CLOUD\_SDK\_REPO="cloud-sdk-$(lsb\_release -c -s)"  
  
# Add the Cloud SDK distribution URI as a package source  
echo "deb http://packages.cloud.google.com/apt $CLOUD\_SDK\_REPO main" | sudo tee -a /etc/apt/sources.list.d/google-cloud-sdk.list  
  
# Import the Google Cloud Platform public key  
curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -  
  
# Update the package list and install the Cloud SDK  
sudo apt-get update && sudo apt-get install google-cloud-sdk

gcloud init

#install additional libraries required by google-cloud-sdk  
sudo apt-get install google-cloud-sdk-app-engine-python google-cloud-sdk-app-engine-python-extrasls

# add google-cloud-sdk to the path  
export PATH="/usr/lib/google-cloud-sdk/bin:$PATH"

<https://cloud.google.com/appengine/docs/standard/python/sockets/ssl_support>

## Specifying the SSL library

If you want to use native Python SSL, you must enable it by specifying ssl for the librariesconfiguration in your application's app.yaml. You should use the latest library version, which is currently [version 2.7.11](https://docs.python.org/release/2.7.11/library/ssl.html). This version supports TLS versions 1.0, 1.1, and 1.2 and corresponds to the SSL versions from Python 2.7.11 and onwards:

libraries:  
- name: ssl  
  version: latest

Note that version 2.7, which supports TLS v1.0 only and corresponds to the SSL version from Python 2.7.8, is deprecated.

# Installing dependencies

Install dependencies using pip:

pip install -t lib -r requirements.txt

The -t lib flag copies the libraries into a lib folder, which is uploaded to App Engine during deployment. See [Using pip requirements.txt with copied libraries](https://cloud.google.com/appengine/docs/standard/python/tools/using-libraries-python-27#pip_requirements) for more information on copying third-party libraries.

The -r requirements.txt flag tells pip to install everything from a requirements.txt file.

# Set current project

gcloud config set project *projectid*

gcloud config set project

SEND FEEDBACK

* [App Engine](https://cloud.google.com/appengine/)

* [Documentation](https://cloud.google.com/appengine/docs/)

* [Python](https://cloud.google.com/appengine/docs/python/)

* [Standard Environment](https://cloud.google.com/appengine/docs/standard/python/)

# **Using third-party libraries**

You can use third-party libraries that are pure Python code with no C extensions, by copying the library into your application directory. If the third-party library is already built-in, [bundled with the runtime](https://cloud.google.com/appengine/docs/standard/python/tools/built-in-libraries-27), you can use the library without copying it into your app.

Third party libraries must be implemented as pure Python code with no C extensions. If copied to your application directory, they count towards file quotas because the library is uploaded to App Engine along with your application code.

### **Copying a third-party library**

To use a third-party library that is not on the list of built-in libraries [bundled with the runtime](https://cloud.google.com/appengine/docs/standard/python/tools/built-in-libraries-27):

1. Create a directory to store your third-party libraries, such as lib/.

mkdir lib

1. Use [pip](https://pypi.python.org/pypi/pip) (version 6 or later) with the -t <directory> flag to copy the libraries into the folder you created in the previous step. For example:

pip install -t lib/ <library\_name>

[Using Homebrew Python on Mac OS X?](https://cloud.google.com/appengine/docs/standard/python/tools/using-libraries-python-27#vendoring)

1. Create a file named appengine\_config.py in the same folder as your app.yaml file.
2. Edit the appengine\_config.py file and provide your library directory to the vendor.add() method.

# appengine\_config.py  
from google.appengine.ext import vendor  
  
# Add any libraries install in the "lib" folder.  
vendor.add('lib')

The appengine\_config.py file above assumes that the current working directory is where the lib folder is located. In some cases, such as unit tests, the current working directory can be different. To avoid errors, you can explicitly pass in the full path to the lib folder using:

vendor.add(os.path.join(os.path.dirname(os.path.realpath(\_\_file\_\_)), 'lib'))

#### **Using pip requirements files with copied libraries**

pip can read a list of libraries to install from a file, known as a requirements file. Requirements files make it easy to set up a new development environment for your app, and upgrade to new versions of libraries.

A requirements file is a text file with one line per library, listing the package name and optionally the version for the package (defaults to latest):

Flask==0.10  
Markdown==2.5.2  
google-api-python-client

To install the libraries from a requirements file, use the -r flag in addition to the -t lib flag:

pip install -t lib **-r requirements.txt**

## Using a built-in third-party library bundled with the runtime

If the third-party library is on the list of [built-in libraries bundled with the App Engine Python runtime](https://cloud.google.com/appengine/docs/standard/python/tools/built-in-libraries-27), you only have to specify it under the [libraries](https://cloud.google.com/appengine/docs/standard/python/config/appref#libraries) directive in the app.yaml, for example:

libraries:  
- name: PIL  
  version: "1.1.7"  
- name: webob  
  version: "1.1.1"

App Engine automatically provides the requested libraries during deployment.

### **Using built-in bundled libraries with the local development server**

Many of the built-in libraries provided by the runtime are automatically available to the local development server. However, the following built-in libraries must be installed locally before you can use them with the local development server:

* [lxml](https://pypi.python.org/pypi/lxml)
* [matplotlib](https://pypi.python.org/pypi/matplotlib)
* [mysqldb](https://pypi.python.org/pypi/MySQL-python/)
* [numpy](https://pypi.python.org/pypi/numpy)
* [PIL](https://pypi.python.org/pypi/pillow)
* [crcmod](https://pypi.python.org/pypi/crcmod)
* [pycrypto](https://pypi.python.org/pypi/pycrypto)
* [grpcio](https://pypi.org/project/grpcio)
* [protobuf](https://pypi.org/project/protobuf)

You can use the [pip](https://pypi.python.org/pypi/pip) command to install all of these packages from the [Python package index (PyPI)](https://pypi.python.org/).

sudo pip install lxml==2.3.5

Depending on your platform, you might need to install build support tools and Python sources to install these libraries.

* On Linux, the package manager can provide these prerequisites and can often provide a pre-built version of the library.
* On Windows, installers for pre-built versions are usually available. \*On OS X, the Xcode Command Line Tools are required to build some packages.

The development server uses the package version you have installed locally regardless of the version specified in app.yaml. If you want, set up a [virtualenv](https://virtualenv.pypa.io/en/latest/) for your project to provide the exact package version. Note that the virtualenv is only used for these binary packages locally and will not be made available to your application once deployed. To add additional third-party libraries, use the method described in [Installing a library](https://cloud.google.com/appengine/docs/standard/python/tools/using-libraries-python-27#installing_a_library).

#### **Using Django in the local development server**

**Warning:** Support for Django versions 1.2 and 1.3 is deprecated and will be removed. See the [Django 1.2, 1.3 Turndown](https://cloud.google.com/appengine/docs/deprecations/django)document for details and timetable.

[Django](http://www.djangoproject.com/) is a full-featured web application framework for Python. It provides a full stack of interchangeable components, including dispatch, views, middleware, and templating components, and many others.

The Django data modeling interface is not compatible with the App Engine datastore. You can use the App Engine data modeling libraries ([db](https://cloud.google.com/appengine/docs/standard/python/datastore/) or [ndb](https://cloud.google.com/appengine/docs/standard/python/ndb/)) in your Django applications. However, third-party Django applications that use the Django data modeling interface, most notably Django's Admin application, might not directly work with App Engine.

The Datastore modeling library (DB) is the default. To use Django with the [NDB storage API](https://cloud.google.com/appengine/docs/standard/python/ndb/) instead, add'google.appengine.ext.ndb.django\_middleware.NdbDjangoMiddleware', to the MIDDLEWARE\_CLASSES entry in your Django settings.py file. It's a good idea to insert it in front of any other middleware classes, since some other middleware might make datastore calls and those won't be handled properly if that middleware is invoked before this middleware. You can learn more about Django middleware in the [project documentation](http://docs.djangoproject.com/en/dev/topics/http/middleware/).

To enable Django in your app, specify the WSGI application and Django library in app.yaml:

...  
handlers:  
- url: /.\*  
  script: main.app  # a WSGI application in the main module's global scope  
  
libraries:  
- name: django  
  version: "1.4"

The DJANGO\_SETTINGS\_MODULE environment variable must be set to the name of your Django settings module, typically 'settings', before packages are imported.

If your Django settings module is something other than settings.py, set the DJANGO\_SETTINGS\_MODULE environment variable accordingly either in your app.yaml file:

env\_variables:  
  DJANGO\_SETTINGS\_MODULE: 'myapp.settings'

Or in your Python code:

import os  
# specify the name of your settings module  
os.environ['DJANGO\_SETTINGS\_MODULE'] = 'myapp.settings'  
  
import django.core.handlers.wsgi  
app = django.core.handlers.wsgi.WSGIHandler()

#### **Using matplotlib in the local development server**

**Note:** The experimental release of matplotlib is not supported on the development server. You can still add **matplotlib** to the **libraries** list, but it will raise an **ImportError** exception when imported.

[Matplotlib](http://matplotlib.org/) is a plotting library that produces graphs and figures in a variety of image formats. On App Engine, the interactive modes of matplotlib are not supported, and a number of other features are also unavailable. This means you cannot use [pyplot.show()](http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.show) as many matplotlib tutorials suggest. Instead, you should use [pyplot.savefig()](http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.savefig) to write image data to the output stream, a [cStringIO.StringIO](https://docs.python.org/2/library/stringio.html) instance, or the Google Cloud Storage using the [Cloud Storage Client Library](https://cloud.google.com/appengine/docs/standard/python/googlecloudstorageclient/app-engine-cloud-storage-sample).

Matplotlib allows [extensive customization](http://matplotlib.org/users/customizing.html#customizing-matplotlib) through the use of the matplotlibrc configuration file, which should be placed in the application's top-level directory. Alternatively, you can set the MATPLOTLIBRC environment variable to a path relative to your application's directory.

The default [backend](http://matplotlib.org/faq/usage_faq.html#what-is-a-backend) is AGG, which allows writing files of all supported formats: PNG (the default format), RAW, PS, PDF, SVG and SVGZ. If you make the PIL library available by adding PIL to the libraries section of app.yaml, then the AGG backend will automatically support writing JPEG and TIFF image formats as well.

Matplotlib comes with a number of fonts which are automatically available. You can use custom fonts by uploading them in TTF format along with your application, and setting the TTFPATH environment variable to the path where they are located, relative to your application's directory. For more information, see the [app.yaml reference](https://cloud.google.com/appengine/docs/standard/python/config/appref#handlers_environment_variables).

A number of matplotlib features are not supported on App Engine. In particular:

* There is no ~/.matplotlib directory. However, there are alternative locations to place the matplotlibrcconfiguration file, as described above.
* Interactive backends and GUI elements are not supported.
* The EMF, Cairo and GDK backends are not supported.
* There is no caching, and therefore a number of mechanisms will re-calculate or re-download data that would normally be cached. Specific caching mechanisms that have been disabled include font data calculated by[matplotlib.font\_manager.FontManager.findfont](http://matplotlib.org/api/font_manager_api.html#matplotlib.font_manager.FontManager.findfont), sample data downloaded by[matplotlib.cbook.get\_sample\_data](http://matplotlib.org/api/cbook_api.html#matplotlib.cbook.get_sample_data) and financial data downloaded by[matplotlib.finance.fetch\_historical\_yahoo](http://matplotlib.org/api/finance_api.html#matplotlib.finance.fetch_historical_yahoo).
  + Because there is no caching, it is not possible to call [matplotlib.cbook.get\_sample\_data](http://matplotlib.org/api/cbook\_api.html#matplotlib.cbook.get\_sample\_data) with asfileobj=False unless examples.download is set to False.
* All features that invoke external commands have been disabled.
  + Use of fontconfig has been disabled. Fonts are found through the mechanism described above.
  + Use of LaTeX for text rendering is not supported. Setting text.usetex to True will not work.
  + Use of an external PostScript distiller program is not supported. Setting ps.usedistiller to ghostscriptor xpdf will not work.
  + Use of an external video encoding program is not supported. The [matplotlib.animation.Animation.save](http://matplotlib.org/api/animation_api.html#matplotlib.animation.Animation.save)method will not work, and therefore, the [matplotlib.animation](http://matplotlib.org/api/animation_api.html) package is not useful.
  + The [matplotlib.cbook.report\_memory](http://matplotlib.org/api/cbook_api.html#matplotlib.cbook.report_memory) function and [matplotlib.cbook.MemoryMonitor](http://matplotlib.org/api/cbook_api.html#matplotlib.cbook.MemoryMonitor) class are not supported.
* The matplotlib.test function has been disabled.

**Note:** The **pylab** and **matplotlib.pyplot** modules are stateful and not thread safe. If you use them on App Engine, you must set **threadsafe: false** in **app.yaml**, and be aware that the plotter state will be preserved between requests on the same instance. For example, you will need to call [**pyplot.clf()**](http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.clf) at the beginning of each request to ensure that previous plots are not visible. It is recommended that you use the thread-safe object-oriented API instead of the stateful pyplot API.

## What's next

* Learn more about the [App Engine Python runtime environment](https://cloud.google.com/appengine/docs/standard/python/runtime).
* Review the [built-in third-party libraries reference](https://cloud.google.com/appengine/docs/standard/python/tools/built-in-libraries-27).

*Was this page helpful? Let us know how we did:*

SEND FEEDBACK

*Except as otherwise noted, the content of this page is licensed under the*[*Creative Commons Attribution 3.0 License*](https://creativecommons.org/licenses/by/3.0/)*, and code samples are licensed under the*[*Apache 2.0 License*](https://www.apache.org/licenses/LICENSE-2.0)*. For details, see our*[*Site Policies*](https://developers.google.com/terms/site-policies)*. Java is a registered trademark of Oracle and/or its affiliates.*

*Last updated April 17, 2018.*