**How To Get Anaconda To Work With An ArcGIS Python Installation**

Authors: Stephen Palka, Mac Friedrichs, and Claudia Young

Last update: Oct 9, 2018

[**Set up a conda multi-user environment**](#_2d74zj3wgu6z) **1**

[**Steps to use anaconda (Miniconda) with ArcGIS**](#_j1i0zjuwrmja) **1**

[References](#_2u0sa0bo5cph) 3

[**Installing the conda Package Manager**](#_16l6e9xwwf18) **4**

[Windows 7](#_xzgl5i25mxoa) 4

[Authenticating to Earth Engine](#_rocql3ojbg4r) 7

[Using Earth Engine Python API with Jupyter Notebooks](#_z5c55ab7odr3) 8

[**Configuring App Engine to use the conda Environment\***](#_rexkal8g5c4r) **9**

[**Troubleshooting**](#_q1hlziy5cpu4) **11**

# Set up a conda multi-user environment

The steps below allows to create and access a conda environment out of the user folder location:

1. To create a multi-user environment from Anaconda command prompt, example using Miniconda 32-bit to create an ArcGIS 10.3.1 python 32-bit environment:  
   >conda create -p D:\conda\envs\arc1031x32 python=2.7.8 numpy=1.7.1 matplotlib=1.3.0 pyparsing xlrd xlwt console\_shortcut
2. To access the multi-user environment created, add the path to the folder containing the environment(s) to the user's conda configuration file C:\Users\<User>\.condarc. Like this:

envs\_dirs:  
 - D:\conda\envs\

# Steps to use anaconda (Miniconda) with ArcGIS

1. Set conda environment variable depending upon if you want to create a 32 bit and/or 64 bit Python environment. If you are using ArcGIS Pro (10.4.x or later), this uses a 64 bit Anaconda installation so new environments created will default to 64 bit Python. For ArcGIS 10.3.x users, the conda installation needs to be the 32-bit. For this document, the steps are based on a system with ArcGIS Pro and Anaconda 64-bit.
   1. set CONDA\_FORCE\_32BIT=1 or set CONDA\_FORCE\_32BIT= (reference # 3)
2. Create the Anaconda environment. It is important to know which versions of certain modules are installed or the environment you create will not with the ArcGIS Python installation.
   1. Follow step A from reference 1 to find relevant versions of important modules.
   2. For ArcGIS 10.3.1, to create the env under user folder, the command would be:   
      >conda create -n arc1031x32 python=2.7.8 numpy=1.7.1 matplotlib=1.3.0 pyparsing xlrd xlwt console\_shortcut
   3. For ArcGIS 10.3.1x64,to create the env under user folder, the command would be:   
      >conda create -n arc1031x64 python=2.7.8 numpy=1.7.1 matplotlib=1.3.0 pandas pyparsing xlrd xlwt console\_shortcut
   4. Additional examples for other versions of ArcGIS can be found under Step B of the first link. If the environment needs to be created as a multi-user environment, use the previous section command to replace -n for -p with the full path.
3. Pin dependencies so the version of important modules cannot be changed by installed other modules. You should pin the versions of the modules you specified in the previous step.
   1. Follow the “Pinning Dependencies” instructions under Step D of reference 1 (recommended). Example for ArcGIS 10.3.1 32-bit conda environment:  
      > activate arc1041  
      >(echo python ==2.7.8 & echo numpy ==1.7.1 & echo matplotlib ==1.3.0 & echo scipy ==0.13.2) > %CONDA\_PREFIX%\conda-meta\pinned  
      >type %CONDA\_PREFIX%\conda-meta\pinned  
      python ==2.7.8  
      numpy ==1.7.1  
      matplotlib ==1.3.0  
      scipy ==0.13.2

Also, do this step for the ArcGIS 10.3.1 64-bit conda environment.

1. Configure anaconda to see arcpy and vice versa:
   1. Using the python script usercustomize.py from EGIS documentation (reference 1)
   2. Set \*.pth files for each python version (32-bit or 64-bit):
      1. Navigate to the applicable ArcGIS site-package directory.
         1. 32-bit path will look something like: C:\Python27\ArcGIS10.3\Lib\site-packages
         2. 64-bit path will look something like: C:\Python27\ArcGISx6410.3\Lib\site-packages  
            In each directory there should be a file either named Desktop10.3.pth (32-bit) or DTBGGP64.pth (64-bit).
      2. Copy the corresponding file to the same directory in your corresponding Anaconda environment.
         1. 32-bit path will look something like: C:\Users\<User>\AppData\Local\conda\conda\envs\arc1031\Lib\site-packages (user-defined env) or D:\conda\envs\arc1031x32 (multi-user env)
      3. To configure arcpy to see an anaconda environment, create zconda.pth file and place it in the ArcGIS Python site-package directory (e.g. for 64-bit, C:\Python27\ArcGISx6410.3\Lib\site-packages).
         1. If needed, create a file name zconda with the extension .pth under the corresponding ArcGIS Python site-package directory.
         2. In the file exists, copy the path to the corresponding Anaconda environment python site-package directory.
2. Test using the *Testing and Troubleshooting* section from the EGIS documentation (reference 1) to be sure everything is working properly.

## References

1. <https://my.usgs.gov/confluence/display/EGIS/Using+Anaconda+modules+from+the+ESRI+python+environment> (Follow through step 4)
2. <https://gis.stackexchange.com/questions/119503/getting-arcpy-to-work-with-anaconda>(Follow response from WxCZar)
3. <https://stackoverflow.com/questions/33709391/using-multiple-python-engines-32bit-64bit-and-2-7-3-5> (Follow response from Mike Müller)
4. ArcGIS python 32-bit and 64-bit, <https://community.esri.com/groups/technical-support/blog/2013/07/29/64-bit-vs-32-bit-python-explained/>

Installing the Earth Engine Python API using the conda packaging manager [WINDOWS 7 GUIDE ONLY, BUT SHOULD WORK ON WINDOWS 10]

The Earth Engine Python API is distributed as a Python package on the [Python Package Index (PyPI)](https://pypi.python.org/pypi/earthengine-api)and the [source code is available on Github](https://github.com/google/earthengine-api). The following instructions give an overview of installing the Google Earth Engine Python API. To use the Earth Engine Python API you'll need to [install the client library and its dependencies](https://developers.google.com/earth-engine/python_install_manual#installing-the-client-library) on your computer and then [set up authentication credentials](https://developers.google.com/earth-engine/python_install_manual#setting-up-authentication-credentials).

Table of Contents:

Installing the conda Package Manager

Windows

Creating a conda Environment for the Earth Engine API

Authenticating to Earth Engine

Adding IPython Notebooks and SciPy to the conda Environment

Configuring App Engine to use the conda Environment

Troubleshooting

# **Installing the conda Package Manager**

[**conda**](http://conda.pydata.org/docs/intro.html) is a cross-platform application for installing software packages. Although it is not specific to installing Python packages, it is being adopted by the scientific Python community for it's ability to handle Python packages that have dependencies on shared libraries. conda is open source, and is available as part of the [Anaconda](http://docs.continuum.io/anaconda/index.html) Python distribution, managed by Continuum Analytics.

The easiest way to install conda, is to download and install the Anaconda Python Distribution available. Binary installers are available for various operating systems, and specific instructions/notes for each operating system are listed below.

If you don't want to install the full Anaconda distribution (which is about 350MB), there are alternative methods of installing conda: you can install the minimal conda installation (a.k.a. [miniconda](http://conda.pydata.org/miniconda.html)), install the [conda package from PyPI](https://pypi.python.org/pypi/conda), or to download and install from the [source code](https://github.com/conda/conda).

## **Windows 7**

1. Go to<http://continuum.io/downloads> and download the Python 2.7 Graphical Installer for your platform (32 bit or 64 bit).

2. Execute the Graphical Installer package, and follow the instructions.

3. When asked where it should be installed, choose “Just me”. By default, Anaconda will be installed in C:\Users\<username>\AppData\Local\Continuum\Anaconda.

4. Go to your Start Menu > All Programs > Anaconda folder, and start the “Anaconda Command Prompt”.

5. List the system path

echo %PATH%

If the system path does not start with a path that includes the Anaconda installation location, see the troubleshooting section below.

https://github.com/ContinuumIO/anaconda-issues/issues/41

[Useful CONDA cheat sheet](https://conda.io/docs/_downloads/conda-cheatsheet.pdf)

**Creating a conda Environment for the Earth Engine API**

This section illustrates how to build a conda environment that allows you to run Python scripts against the Earth Engine API. Note that these instructions were written using conda version 3.5.5. and pip version 9.0.1 [Updated June 2018: The Earth Engine Python API works with Python 2.7, but it is recommended to install and use Python 3 if not already doing so]

\*\*Before starting, make sure [conda-forge channel](https://conda-forge.org/#about) is connected (if not already):

conda config --add channels conda-forge

1. Start by creating a new conda environment (let’s call it “ee-python”, but it can be named anything). Specify which version of Python you want to use (2.7 or 3). In this example we install Python 2.7 into our new environment.

conda create -n ee-python python=2.7

2. Next, activate this environment:

activate ee-python

Don't do this now, but once you are done using a conda environment, you can deactivate the environment by executing the following code (this will not delete the environment):

deactivate

**NOTE:** The following conda packages can also be installed all at once as:

conda install oauth2client simplejson pip

**NOTE:** (EROS users) If conda install throws SSL routine errors, try setting SSL check to

*False* like [this suggestion](https://collaboration.usgs.gov/wg/erosdc/EROS%20IT/Lists/TechWiki/DispForm.aspx?ID=397), but be sure to set it back to *True* when done. Or try

connecting to the **Colorado VPN** first.

Or they can be installed separately as:

3. Install oauth2client. The Earth Engine API uses this for authenticating clients.

conda install oauth2client

4. Install SimpleJSON.

conda install -y simplejson

5. Check to see if PIP is already installed by looking at its version. (pip should already come installed with Python).

pip --version

5a. If not already installed, Install PIP into the ee-python conda environment, so that it can be used to install Python packages that are not available as conda packages, but are available in the Python Package Index ([PyPI](https://pypi.python.org/)).

conda install -y pip

6. Next install the Earth Engine Python API package **using pip** ***(recommended method)***.

pip install earthengine-api

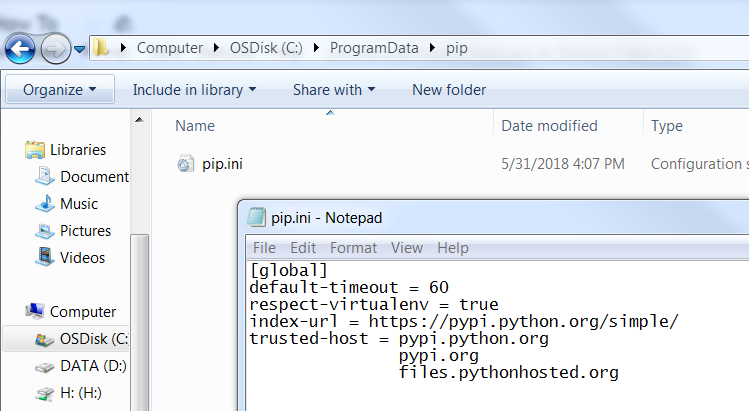
**NOTE:** (EROS users) If unsuccessful and SSL Errors appear, first try

pip install earthengine-api -i <https://pypi.org/simple/>

But may require a permanent pip fix solution per:

<https://stackoverflow.com/a/29751768> which requires admin

privileges to modify the existing pip.ini config file to this:



**\*\***If pip still doesn’t work, could instead try to use

**Option 2:** using conda

conda install earthengine-api(installs from [conda forge](https://github.com/conda-forge/earthengine-api-feedstock)**)**

or

**Option 3:** using easy\_install instead of pip or conda

easy\_install earthengine-api

7. Test that you can successfully import the Earth Engine API package, by running:

python -c "import ee; print ee.\_\_version\_\_"

**Note for Windows users:** *Make sure that you execute this command in a directory that does not contains python.exe (such as the default Anaconda Command Prompt) , so that the python command correctly executes the python executable in the conda environment and has access to the Python packages that you have installed.*

## **Authenticating to Earth Engine**

To setup your connection to the Earth Engine servers, follow the instructions in the Setting Up Authentication Credentials section of the Developer's Guide found [HERE](https://developers.google.com/earth-engine/python_install_manual#setting-up-authentication-credentials):

**Note:** Be sure to have a browser that enables cookies when performing the authentication steps here. (Chrome should work, IE admin settings might not?)

**Note for EROS users (as of: 4/18/18):** Authenticating behind the EROS WAF to the Google Servers sometimes bumps into *SSL Handshake Errors* (per the DOI Security Cert). This is the same known issue with authenticating to the Google Cloud Platform (i.e. gsutil, Cloud SDK, etc.). Helpdesk has/had an open ticket regarding this issue, but it might not affect everyone equally. Depending on what set of ports CITT opens up, you *might* need to connect to the **Colorado VPN** before authenticating with Earth Engine, for now...

1. Authenticate to the Earth Engine servers by using the following command [(instructions here)](https://developers.google.com/earth-engine/command_line#authenticate):

earthengine authenticate

**Set up your GEE Authentication credentials following** [**these instructions**](https://developers.google.com/earth-engine/python_install_manual#setting-up-authentication-credentials).

1. The Earth Engine APIs use the [OAuth 2.0 protocol](https://oauth.net/2/) for authenticating clients. In order to authenticate, you will need to first setup a credentials file on your computer that authorizes access to Earth Engine on behalf of your Google account.

python -c "import ee; ee.Initialize()"

***[Optionally]* Test the GEE API Command Line tools**

1. Test the functionality on the command line from [these instructions](https://developers.google.com/earth-engine/command_line#ls):

Check the contents of your GEE user folder (Assets).

earthengine ls

**If no errors, then you are good to go!**

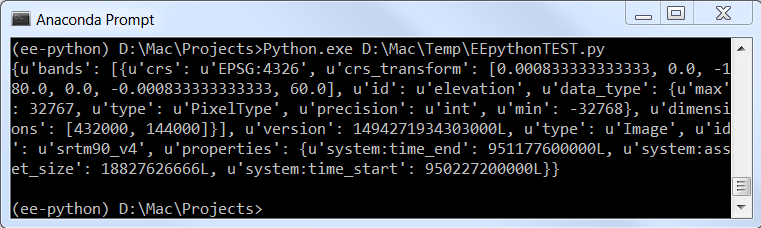
***[Optionally]* Run a test Python script for Earth Engine**

1. Create a small EEpythonTESTscript.py file containing the example test script from *Testing the Installation* [instructions here](https://developers.google.com/earth-engine/python_install_manual#testing-the-installation).

2. Use the default *Anaconda Command Prompt* in your new environment (activate ee-python),so that the python command correctly executes the python executable in the conda environment and has access to the Python packages that you have installed.

Python.exe D:Path\To\Test\Script\EEpythonTEST.py

If successful, it should look something like this. A print out of the metadata for an image:



You are now able to use the Earth Engine Python API.

Additional Developer Resources here:

* [How does this different Client-Server programming model work in Earth Engine?](https://developers.google.com/earth-engine/concepts_overview)
* [Debugging Guide - JS tips can apply to Python since the API calls are identical](https://developers.google.com/earth-engine/debugging)
* [How to find help and where to look first](https://developers.google.com/earth-engine/help)
* [Command Line Tool](https://developers.google.com/earth-engine/command_line)
* [Familiarize with the data structures and API methods](https://developers.google.com/earth-engine/getstarted#earth-engine-data-structures)
* [GEEUS17](https://events.withgoogle.com/google-earth-engine-user-summit-2017/breakout-sessions/#content) and [GEEUS18](https://sites.google.com/earthoutreach.org/eeus2018/agenda/session-descriptions) User Summit content. Including [this presentation](https://docs.google.com/presentation/d/e/2PACX-1vTwUIOkdjLisiSS2dUkpFVge3GsxJVOEgxk4QsTgtIxjU3Y2FEMXpwpFJVfK19ew14VQKddOVNSzlpT/pub?slide=id.p) by Tyler Erickson of the Earth Engine team.

**You can stop here!**

Otherwise…The following contains additional information for using GEE with Jupyter.

## **Using Earth Engine Python API with Jupyter Notebooks**

Further training/reading:

* + Review slide deck: [The Earth Engine Python Environment](http://g.co/earth/eeus2018-python)
  + Get familiar with Git. <https://software-carpentry.org/lessons/> lesson "Version Control with Git"
  + Get familiar with JupyterLab by reading the docs (in particular, the Getting Started and User Guide sections) <http://jupyterlab.readthedocs.io>
  + Optional background info:
    - JupyterHub <http://zero-to-jupyterhub.readthedocs.io> <http://jupyterhub.readthedocs.io/en/latest/>
    - Docker Containers: <https://docs.docker.com/get-started/>

While I don't foresee anyone on the Earth Engine team at Google building a GDAL + GEE image any time soon, there shouldn't be a barrier to anyone else in the Earth Engine community building such an image, if they have knowledge of [how Docker works](https://docs.docker.com/get-started/). GDAL is just one possible tool for working with GEE-generated data, and there are many other useful pre- and post- processing software tools that may be useful as well.

Mac TODO:

* Summarize Code Editor JavaScript functionality and purpose, differences compared to Python API → insert comparison graphic.

1. Overview of JupyterHub background info → Best for collaboration and prototyping environment prior to any production-scale engineering.

* JupyterCon 2018
* Jupyter Widgets
* Ipyleaflet visualization similar to code editor

2. Using Jupyter Notebooks for the GEE Python API

* Example steps cloning community ipython notebooks (link below)
* Steps for running .ipynb scripts and visualizing results.

3. Further Reading: Kubernetes

There already exists a small collection of Docker images that contain the Earth Engine Python API in the gee-community GitHub organization:

<https://github.com/gee-community/ee-jupyter-contrib>

In particular, you may be interested in using this Docker configuration as a model:

<https://github.com/gee-community/ee-jupyter-contrib/tree/master/docker/jupyter-scipy-notebook-ee>

(Note that I have found the Jupyter Project's UI (JupyterLab) to often be more useful than Google's Datalab UI, because the Jupyter Project's UI allows you to utilize [Jupyter Widgets](http://jupyter.org/widgets).)

* Ipyleaflet
* [folium](https://github.com/python-visualization/folium)

# **Configuring App Engine to use the conda Environment\***

**\*Have not tested this App Engine configuration at EROS.**

1. Install Google App Engine for Python, by following the instructions [here](https://developers.google.com/appengine/docs/python/gettingstartedpython27/introduction).

2. Switch to the temporary directory:

cd ~/temp

3. Clone the Earth Engine API repository:

git clone<https://code.google.com/p/earthengine-api/>

4. Move to the directory for the App Engine sample project.

cd earthengine-api/python/examples/AppEngine/hello\_world

5. Update the application name in app.yaml either manually, or by executing the following commands. The application name is arbitrary, and will later be part of the URL that provides access to your app.

export APPENGINE\_APP\_ID=test-app

mv app.yaml app.yaml.bak

sed "s/<your-appid-here>/$APPENGINE\_APP\_ID/g" app.yaml.bak > app.yaml

6. Add a link to your private key file.

ln -s $HOME/.keys/privatekey.pem privatekey.pem

7. Update the EE\_ACCOUNT variable in config.py to match your service account email either manually, or by executing the following commands:

mv config.py config.py.bak

*sed "s/EE\_ACCOUNT = 'your-service-account-id@developer.gserviceaccount.com'/EE\_ACCOUNT = '$MY\_SERVICE\_ACCOUNT'/g" config.py.bak > config.py*

8. Add links to the Python library dependencies.

ln -s $HOME/anaconda/envs/ee-python/lib/python2.7/site-packages/ee ee

ln -s $HOME/anaconda/envs/ee-python/lib/python2.7/site-packages/oauth2client/ oauth2client

ln -s $HOME/anaconda/envs/ee-python/lib/python2.7/site-packages/six.py six.py

ln -s $HOME/anaconda/envs/ee-python/lib/python2.7/site-packages/httplib2 httplib2

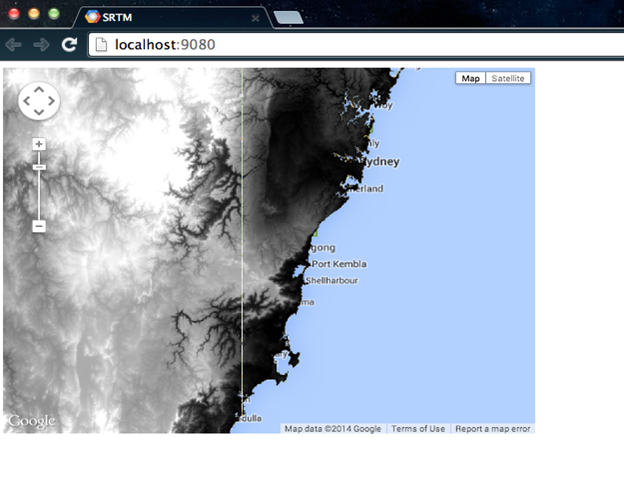
9. Start up the Google App Engine development server.

dev\_appserver.py .

10. [OSX] Start GoogleAppEngineLauncher

Go to Preferences, edit Python path to anaconda/bin/python2.7

11. Open up a browser and navigate to URL where your application is running (typically [http://localhost:8080](http://localhost:8080/)). You should see a map with a DEM[1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] , you browser window should look similar to the following screenshot.



12. When finished developing, deploy the application

appcfg.py update .

13. Browse to the following URL to see the deployed app. [http://APPENGINE\_APP\_ID.appspot.com/](http://appengine_app_id.appspot.com/)

# **Troubleshooting**

**AttributeError: 'module' object has no attribute 'SignedJwtAssertionCredentials'**

This error is usually caused when the application cannot find the Crypto. Make sure that you can run the following import:

from oauth2client import crypt

**NotImplementedError: PKCS12 format is not supported by the PyCrpto library. Try converting to a "PEM"**

This error can occur if the PEM file does not start with "-----BEGIN". Open the PEM file with a text editor, and remove any lines that occur before "-----BEGIN".

**In Windows, opening up the Anaconda Command Prompt does not set the path correctly.**

As of version 2.0.1 of Anaconda, the batch file that sets the system path does not work correctly. This issue has been reported on the [Anaconda issue tracker](https://github.com/ContinuumIO/anaconda-issues/issues/41). The following code can be used to replace the activate batch files (located in anaconda\Scripts\activate.bat).

@echo off

rem +===========================================================================

rem | Initialisation

rem +===========================================================================

for %%i in ("%~dp0..\envs") do (

set ANACONDA\_ENVS=%%~fi

)

if not "%1" == "" (

if not exist "%ANACONDA\_ENVS%\%1\python.exe" (

echo No environment named "%1" exists in %ANACONDA\_ENVS%

goto :eof

)

set ANACONDA\_ENV\_NAME=%1

set ANACONDA=%ANACONDA\_ENVS%\%1

title Anaconda (%ANACONDA\_ENV\_NAME%^)

) else (

set ANACONDA\_ENV\_NAME=

for %%i in ("%~dp0..") do (

set ANACONDA=%%~fi

)

title Anaconda

)

set ANACONDA\_SCRIPTS=%ANACONDA%\Scripts

set PATH=%ANACONDA%;%ANACONDA\_SCRIPTS%;%PATH%

rem echo Setting path to: %PATH%

if not "%ANACONDA\_ENV\_NAME%" == "" (

echo Activating environment %ANACONDA\_ENV\_NAME%...

set PROMPT=[%ANACONDA\_ENV\_NAME%] $P$G

)

rem vi:set ts=8 sw=4 sts=4 tw=0 expandtab: