Real-Life Python Geoprocessing in the Cloud

Jason Wise, GISP
GIS Technical Lead
Terracon

Northeast Arc User Group (NEARC) Fall Conference
19 October 2016
North Falmouth, Massachusetts
Why cloud geoprocessing?

Geoprocessing: Enables you to do cool stuff with ArcGIS.

Python geoprocessing: Enables further customization.

ArcGIS for Server (AGS): Publishes tools for your coworkers or customers.

AGS in the Cloud: Eliminates hardware concerns; improves availability.
Real-life examples

Case study: Simple document management system.

• Big litigation project
• Coworkers needed to review many scanned documents.
• People in different offices needed to view documents and enter basic information.
• We needed a document management system, not a map.
Real-life examples

Case study: Simple document management system.

Why am I talking about this?

user → Web interface (HTML/JavaScript) → database

scanned documents (image files)
How do you provide a database to a Web application?
May as well use AGS!

Real-life examples

Case study: Simple document management system.
Real-life examples

Case study: Simple document management system.

Use a geoprocessing service with two tools.
• GetScans
• UpdateAttribute

user → Web interface (HTML/JavaScript) → database

scanned documents (image files)
Real-life examples

Case study: Simple document management system.

Interesting parts:

- AGS geoprocessing service provides convenient database access.
- AGS can be useful even when there’s no map.
Real-life examples

Case study: Automatic data collection.

- Driving thousands of piles for solar farms.
- Continuously logging x, y, z, and hammer status.

![Graph showing Pile Height (inches) over Time of Day (UTC)]
Real-life examples

Case study: Automatic data collection.

• Need to process data daily and produce maps & reports.
Real-life examples

Case study: Automatic data collection.

Interesting parts:
• This time, there’s a map but no geoprocessing service.
• A nightly scheduled task runs Python geoprocessing code.
• Results are immediately available through an existing map service.
Real-life examples

Case study: Processing sites for multiple clients.

- Store national datasets for soils, elevation, etc.
- Upload project geometry to geoprocessing service.
- Process data in area of interest.
- Automatically generate MXDs and publish map services.
- Cache tiles on staging server.
- Republish to production servers.
Real-life examples

Case study: Processing sites for multiple clients.

Interesting parts:

• We use geoprocessing services to publish map services.
• Tools use many input & output datasets, so they must be configurable.
• Tools are complex and frequently updated, so they must be modular.
Easy publishing

Examples typically involve publishing a simple tool on a local network.
Real-life publishing

You can quickly run into problems.
Problem: importing code

```python
import arcpy
import os
from myclass1 import MyClass1
```
Problem: importing code

Solutions:

• Don’t import code (bad solution).
• Make your own Python library (more on this later).
Problem: data paths

Scenario:

- Your publishing machine doesn't have access to the server's data.
- In your code, you specify the path to the data on the server.

```
inputpath = r'E:\Temp\Alignment.shp'
outputpath = r'E:\Temp\Alignment2.shp'
```
Problem: data paths

First try:
Your tool won’t run locally, because the data don’t exist here.

Solutions:
• Replicate the data path on your local machine.
• Write the tool so it will exit gracefully with invalid data.

Spoiler: These won’t work.
Problem: data paths

Second try:

You run the tool and publish it...
Problem: data paths

Second try:

…but it doesn’t exactly run on the server.

What’s this path?
ArcGIS has modified your code.

Problem: data paths

Second try:

```python
# Esri start of added imports
import sys, os, arcpy
# Esri end of added imports

# Esri start of added variables
g_ESRI_variable_1 = os.path.join(arcpy.env.packageWorkspace,u'..\cd\temp1\Alignment.shp')
g_ESRI_variable_2 = u'\scratchFolder\Alignment2.shp'
# Esri end of added variables

import arcpy
import os.path

class Toolbox(object):
```
Problem: data paths

Second try:

ArcGIS has modified your code.

def execute(self, parameters, messages):
    inputpath = g_ESRI_variable_1
    outputpath = g_ESRI_variable_2
    arcpy.AddMessage('File paths: %s, %s' % (inputpath, outputpath))
    arcpy.management.CopyFeatures(inputpath, outputpath)
    arcpy.AddMessage('Done.')
    return
Problem: data paths

Third try:

Read a data path from a text file.
Added bonus: you can tailor the file to different machines.
In other words, you’re making a configuration file.

![config.txt file with path C:\Temp\Alignment.shp]
Unfortunately, it doesn’t work. ArcGIS uploads the local config file to the server.
Problem: data paths

Third try:

```
# Esri start of added imports
import sys, os, arcpy

# Esri end of added imports

# Esri start of added variables
g_ESRI_variable_1 = os.path.join(arcpy.env.packageWorkspace, u'..\..\cd\temp\config.txt')
g_ESRI_variable_2 = u'scratchFolder\Alignment2.shp'

# Esri end of added variables

import arcpy
import os.path

class Toolbox(object):
    def __init__(self):
        return
def execute(self, parameters, messages):
    configpath = g_ESRI_variable_1
    configfile = open(configpath)
inpath = configfile.read()
arcpy.AddMessage('Config file: %s' % configpath)
arcpy.AddMessage('Input path: %s' % inpath)
outputpath = g_ESRI_variable_2
arcpy.AddMessage('File paths: %s, %s' % (inpath, outputpath))
arcpy.management.CopyFeatures(inpath, outputpath)
arcpy.AddMessage('Done.')
return
```
Problem: data paths

Third try:
Problem: data paths

Fourth try:

Conceal your paths from ArcGIS.

```python
def execute(self, parameters, messages):
    configchunks = ['C:', 'Temp', 'config.txt']
    configpath = os.sep.join(configchunks)
    configfile = open(configpath)
    inputpath = configfile.read()
    arcpy.AddMessage('Config file: %s' % configpath)
    arcpy.AddMessage('Input path: %s' % inputpath)
    outputpath = g_ESRI_variable_1
    arcpy.AddMessage('File paths: %s, %s' % (inputpath, outputpath))
    arcpy.management.CopyFeatures(inputpath, outputpath)
    arcpy.AddMessage('Done.')
return
```
Problem: data paths

My favorite solution:
Make your own Python library!
Put your code in a folder in site-packages

or...

Problem: data paths
Problem: data paths

...use a .pth file.
Problem: data paths

Import your code.

```python
from os import path
from terrapy.toolbox.managertool import ManagerTool
from terrapy.publish.publishingmanager import PublishingManager
```
Problem: data paths

One file contains a hard-coded path to a configuration file.

```
1. path = 'C:\GIS\GIS003_ConfigFiles\REGCConfig.xml'
```
The config file contains the paths to your data...

```xml
<terraform>
  <log>
    <logfolder>C:\Temp</logfolder>
    <logfile>Terracon.log</logfile>
  </log>
  <workspaces default="TileGDB">
    <workspace name="TempFolder" path="C:\Temp"/>
    <workspace name="ProjectBaseFolder" path="C:\GIS"/>
    <workspace name="MapTemplates" path="C:\GIS\GIS004_TerraPy\Maps"/>
    <workspace name="TestInputs" path="C:\GIS\GIS004_TerraPy\TerraData\Inputs.gdb"/>
    <workspace name="ConnectionFiles" path="C:\Users\javis6\AppData\Roaming\ESRI\Desktop10.3\ArcCatalog"/>
    <workspace name="SDM" path="C:\GIS\GIS004_TerraPy\TerraData\PretendSDM.gdb"/>
    <workspace name="TileMasterFolder" path="C:\GIS\GIS004_TerraPy\TerraData\Landslides"/>
    <workspace name="TileGDB" path="C:\GIS\GIS004_TerraPy\TerraData\Landslides\Landslides.gdb"/>
    <workspace name="NED" path="H:\NGTOC_NED_Ops\Output\distribution\13"/>
    <workspace name="Geometry" path="C:\GIS\GIS004_TerraPy\TerraData\ProjectSpecific.gdb"/>
    <workspace name="Lookup" path="C:\GIS\GIS004_TerraPy\TerraData\Config001\Lookup.gdb"/>
    <workspace name="Geology" path="E:\J1149197_CPG_GeohazardAssessmentModel_USI\U_TerraData\gdb\Geology_Units.gdb"/>
    <workspace name="Rover" path="C:\GIS\J1149328_Rover2\TerraData\Rover.gdb"/>
    <workspace name="TempMD80" path="C:\GIS\TempMD80\TerraData\REGC.gdb"/>
    <workspace name="PileData" path="C:\GIS\PT7156045_PDM\TerraData\PileData.gdb"/>
    <workspace name="Measurements" path="C:\GIS\GIS004_TerraPy\TerraData\MeasureConfig.gdb"/>
    <workspace name="FourBrothersInput" path="C:\GIS\PT7156045_PDM\TerraData\GDrive\FourBrothers"/>
  </workspaces>
</terraform>
```
Problem: data paths

...among other things.
Bonus!

Your library can be reusable and modular (unlike uploaded code).
Problem: republishing

During development, you’ll publish your tools multiple times. This is a pain.
Problem: republishing

Solution: Make a tool that publishes other tools.

def publish_tools(self, toolboxpath, toolnames, server, webfolder, servicename):
    toolboxmodule = self.import_toolbox(toolboxpath)
    if toolboxmodule is None:
        return False

    resultlist = []
    for toolname in toolnames:
        result = self.get_tool_result(toolboxmodule, toolname)
        resultlist.append(result)

    sdpwd = path.join(self.tempfolder, 'ToolDefinition.sd')
    sddraft = path.join(self.tempfolder, 'Draft.sd')
    arcpy.CreateGPSDDraft(resultlist, sddraft, servicename, copy_data_to_server=False, folder_name=webfolder, summary='Says ConfigTest.', showMessages='Info')

    return self.publish_sd(sdpwd, sddraft, server)
Problem: running locally

Reasons you might not want to run your tool on the publishing machine:

• Lack of data
• Long running time
Problem: running locally

Solution:
Detect null input (or some other condition) and run in “publishing mode.”

```python
def run_manager(self):
    projectname = self.toolparamdict['projectname']
    webfolder = self.toolparamdict['webfolder']

    if projectname is None and webfolder is None:
        self.add_message('No inputs provided. Running in publishing mode.')
        return True

    return self.manager.convert_old_regc(webfolder)
```
Final tips

• Be sure to set the service’s message level to “info.”
• If possible, install ArcGIS Desktop on the server.
  • Note that you’ll have two Python installations and two .pth files (not counting ArcGIS Pro).
• Even without Desktop, you can use arcpy on the server (if you have AGS).