Cloning Apps and Items with ArcGIS API for Python

John Yaist

Raul Jimenez Ortega

Topics

- Cloning Content Scenario
- Demo – Cloning Hosted Feature Layers and Webmaps
- Discussion – clone_items() function and key parameters
- Demo – Cloning Dashboards and Web Mapping Applications

Staging

Production

Demo:

Cloning Feature Layers and Webmaps

Accessing the function

Content Manager

ContentManager

- **Accessed from the *content* property of a *GIS* object**

```
gis = GIS(profile="your_ent_admin_profile")
content_manager = gis.content
```

```
type(content_manager)
```

```
arcgis.gis.ContentManager
```

- ***clone_items()* is a method on the ContentManager**

```
clone_result = gis.content.clone_items(items=[item_list]
                                       search_existing_items=True)
```

https://developers.arcgis.com/python/api-reference/arcgis.gis.toc.html#arcgis.gis.ContentManager.clone_items

clone_items()

clone_items(*items, folder=None, item_extent=None, use_org_basemap=False, copy_data=True, search_existing_items=True, item_mapping=None, group_mapping=None, owner=None*)

Clone content to the GIS by creating new items.

Cloning an item will create a copy of the item and for certain item types a copy of the item dependencies in the GIS.

For example a web application created using Web AppBuilder or a Configurable App Template which is built from a web map that references one or more hosted feature layers. This function will clone all of these items to the GIS and swizzle the paths in the web map and web application to point to the new layers.

This creates an exact copy of the application, map, and layers in the GIS.

Key Parameters

- ***search_existing_items***

  default value = True

- ***item_mapping***

  optional dictionary

search_existing_items

```
clone_result = gis.content.clone_items(items=[item_list]
                                       search_existing_items=True)
```

- uses the item *typeKeyword* property to label result
  - cloned items have a typeKeyword formatted like below added:

    source-<itemId>

  - example: *Feature Layer* item id e9541fd18bdc439dbf84ef7e929475ac.

    cloned item typeKeyword: source-e9541fd18bdc439dbf84ef7e929475ac

item_mapping

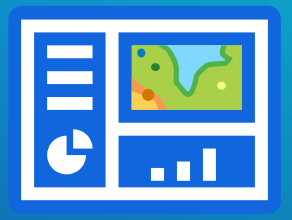
```
clone_result = gis.content.clone_items(items=[item_list]
                                       item_mapping={source.id:target.id}
```

- **uses an existing item in the *target* GIS**
  - **useful for items comprised of items:**
    - **Web Mapping Applications**
    - **ArcGIS Dashboards**

- ***clone_items()***
  - **does not automatically clone all items comprising more complex item types**

Demo:

Cloning ArcGIS Dashboards and Web Mapping Applications



Conclusion

- *search_existing_items=True* – prevents duplication of previously cloned items

- Webmaps – Feature Layers are cloned by default
- Feature Layers – data is cloned by default

- cloning Web Mapping Applications and Dashboards
  - use item_mapping to control which items serve particular roles

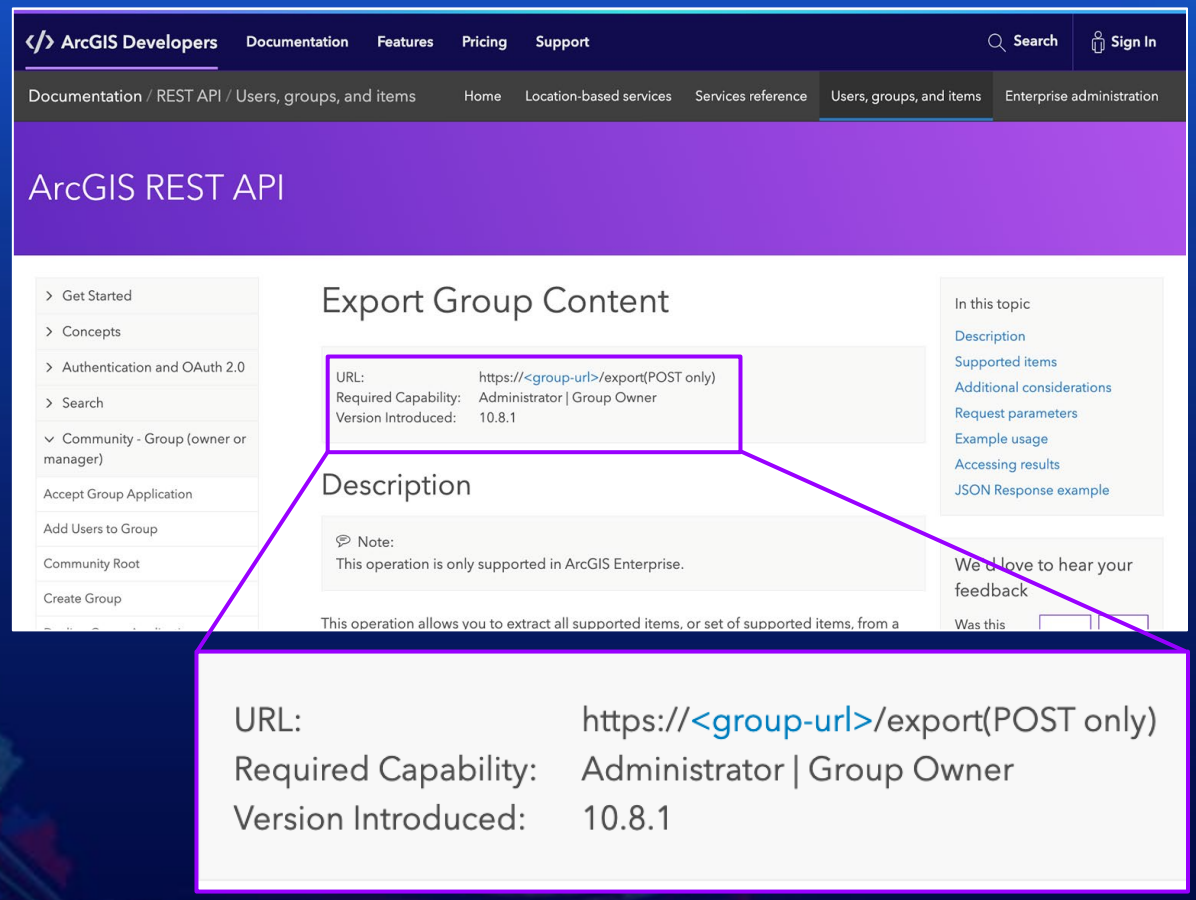
ArcGIS API for Python Cloning Content Guide

- https://developers.arcgis.com/python/guide/cloning-content/

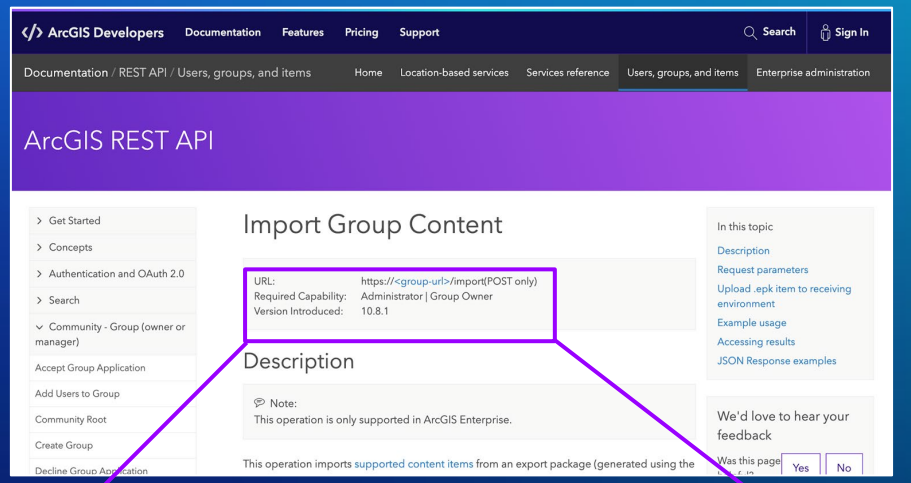
Please share your feedback via the survey available at the link below the presentation.

Group Export and Import APIs

Available on **ArcGIS Enterprise >= 10.8.1** (July 2020)



URL: https://<group-url>/export(POST only)
Required Capability: Administrator | Group Owner
Version Introduced: 10.8.1



URL: https://<group-url>/import(POST only)
Required Capability: Administrator | Group Owner
Version Introduced: 10.8.1

*Simplify the flow of content migration **between groups.***

Group Export and Import

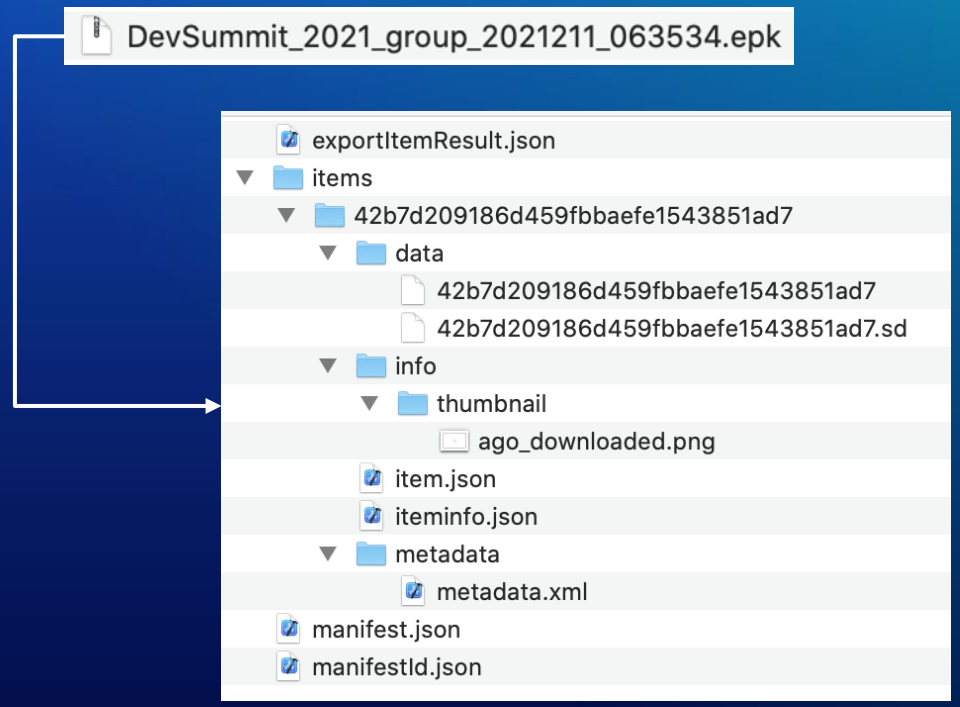
How the operations work

Export:

- **Generate an export package (.epk)** containing all supported items[1] that have been shared **with the group**.
- That epk can be uploaded to a **separate** ArcGIS Enterprise **organization.**

Import:

- Support **content items from an export** into the **selected group**.
- When the items are imported, their item IDs are maintained, and URLs are updated to reflect the new environment
- Before running the operation, the .epk must be added as an item. And after the operations complete, the package will be deleted.



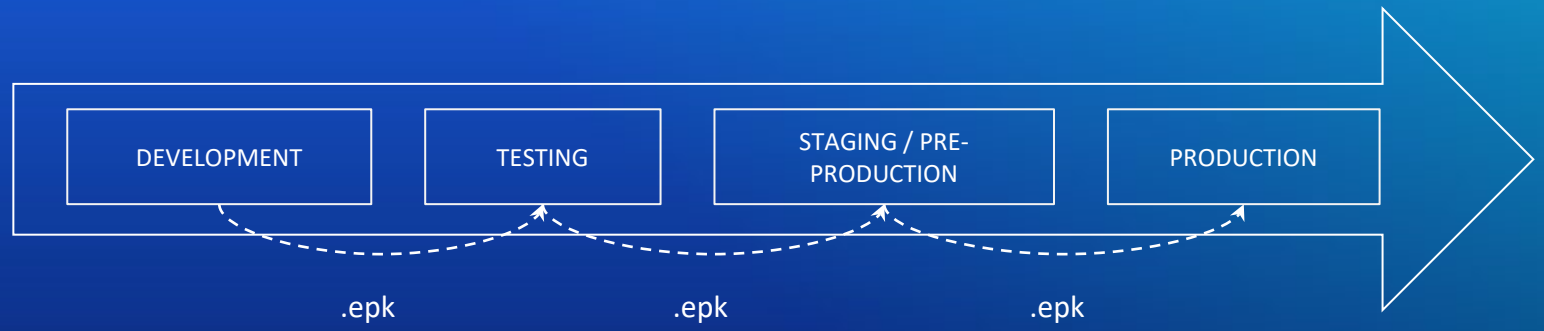
[1] *Supported items types are listed in the docs (+30)*

Group Export and Import

When would you use them? Example scenarios

Scenario 1)

Move content between **deployment environments**



Scenario 2)

Moving content between **disconnected environments**

Scenario 3)

Select content that needs to be backed up and restored or archive (**partial backups**).

Group Export and Import

Some considerations to take into account

- **Considerations** about this version of the APIs (10.8.1)**.**
  - **It is still available:**
    - **Through REST API and ArcGIS for Python API (no Portal UI/UX yet)**
    - **In ArcGIS Enterprise (no in ArcGIS Online at this time)**

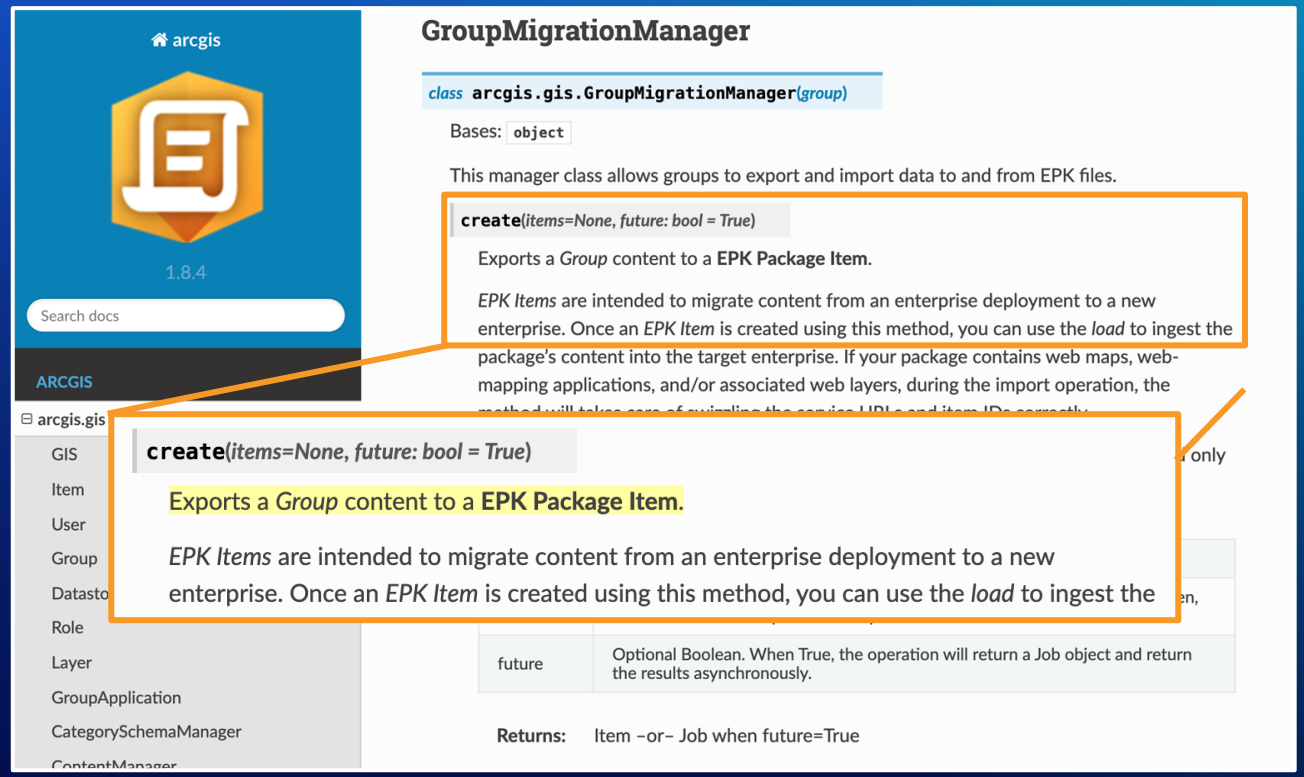
- **Other technical considerations**
  - **Only available to Administrators and Group Owners.**
  - **An .epk has a 5GB file limit.**
  - **For items with dependencies (like web maps), items and dependencies must be shared with the group.**

Group Export and Import

[NEW] GroupMigrationManager class since ArcGIS API for Python 1.8.1

This new class is derived of a Group object and encapsulate the operations:

REST API (operation)	ArcGIS API for Python (method)
/export	create
/import	load



load(*epk_item, item_ids: list = None, overwrite: bool = True, future: bool = True*)

Imports the EPK content into the current *Group*.

Group Export and Import

DEMO TIME!

cloneMigrations – export.py

```python
import os
from arcgis.gis import GIS

print("Connecting to source Portal...")
source = GIS("https://mormont.esri.es/portal", os.environ["MORMONT_USER"], os.environ["MORMONT_PWD"])

print("Getting group from source...")
dev_group = source.groups.search("DevSummit 2021 group")[0]
print(dev_group)
print("Building epk file...")

epk_job = dev_group.migration.create(future=True) # Async call
epk_file = epk_job.result()

print("Downloading export package...")
fp = epk_file.download()
print("Export operation finished:")
print(fp)
```

Exporting a group on 4 lines of code

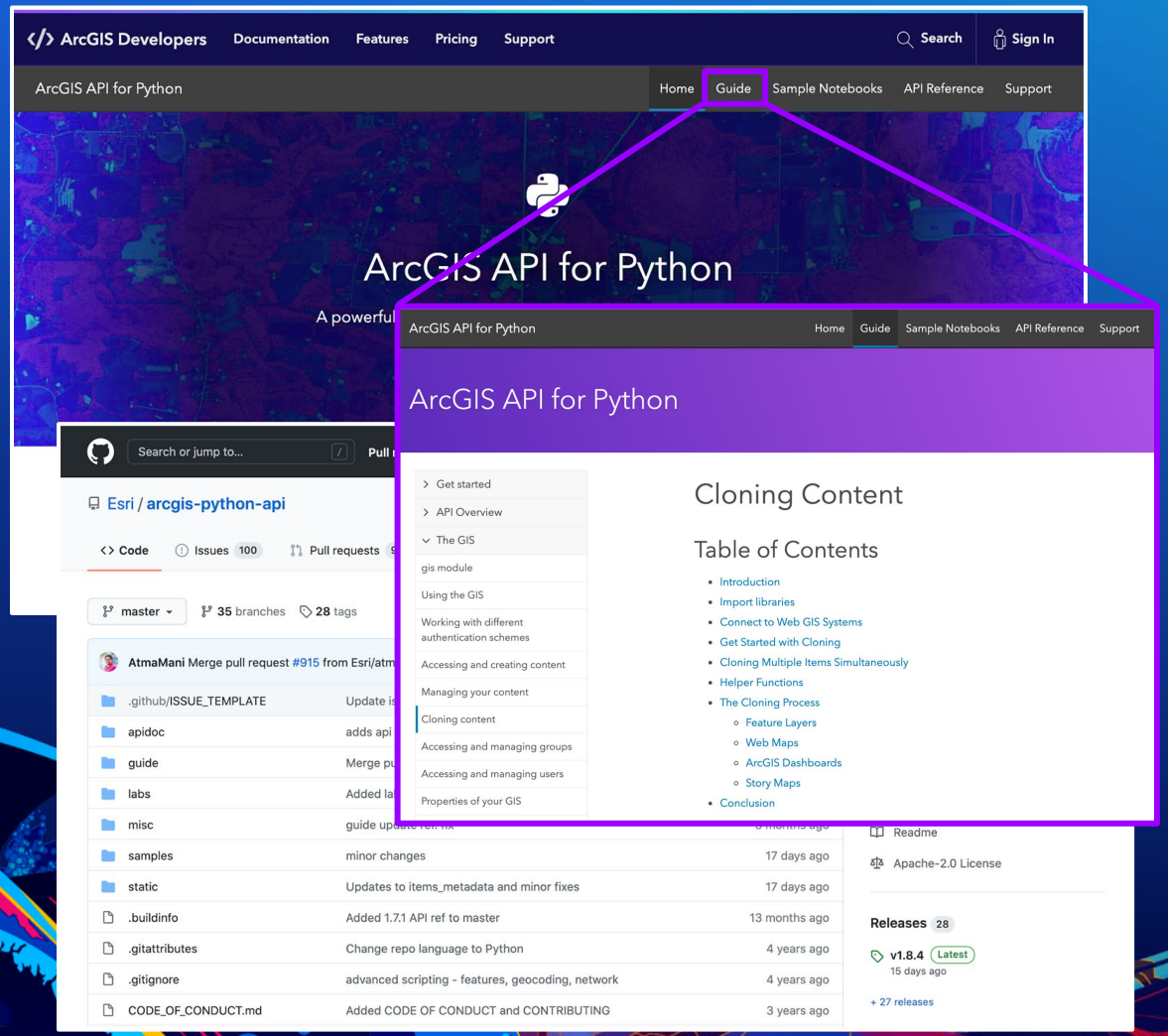
Importing a group on 6 lines of code

cloneMigrations – import.py

```python
import os
from arcgis.gis import GIS
from utils import clean_data

print("Connecting to target Portal...")
target = GIS("https://formacionutility.esri.es/portal", os.environ["UTILITY_USER"], os.environ["UTILITY_PWD"])

print("Cleaning data from previous executions")
clean_data()

print("Adding export package to Target Portal...")
pitem = target.content.add({'title': "DEVtoTEST Migration Package", "tags": ['Migration', 'TEST', 'Cloning'], 'type': 'Export Package'}, data=fp)
print(pitem)

print("Creating target group...")
new_group = target.groups.create(title="DevSummit 2021 backup group", tags="Migration,TEST,Cloning")

print("Sharing Export Package (.epk) to the target group...")
pitem.share(groups=[new_group])

print("Inspecting the epk...")
m = new_group.migration
resp = m.inspect(pitem)
from pprint import pprint
pprint(resp)

print("Executing import operation...")
res = m.load(pitem)
print("Import operation finished")
print(res)
```

ArcGIS API for Python

Keep diving: developers.arcgis.com/python | github.com/Esri/arcgis-python-api | esri.com/arcgis-blog/?s=#&products=api-python



Python for Geographers

On Demand

ESRI TECHNICAL SESSION

Would you like to accent your GIS skills with Python but are not sure where to get started? Learn the basics of Python-speak in the ArcGIS notebooks environment where we'll use packages to make GIS workflows reproducible through code. We will work with spatial data and make relevant maps using open source packages as well as the ArcGIS API for Python. To conclude we'll discuss useful resources as you delve deeper into the universe of Python for Data Science and GIS. Keywords: Computational geography, Introduction, Reproducible science

Manushi Majumdar

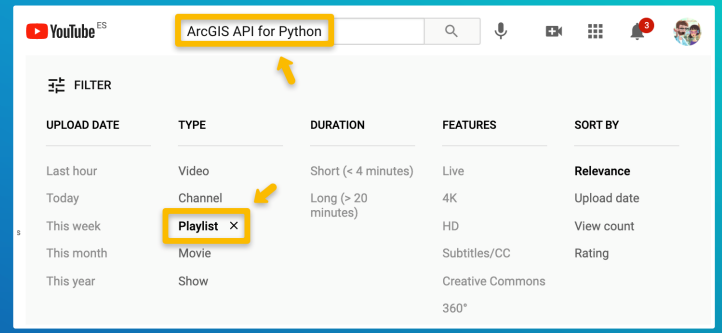
Deploying Apps and Services with ArcGIS API for Python

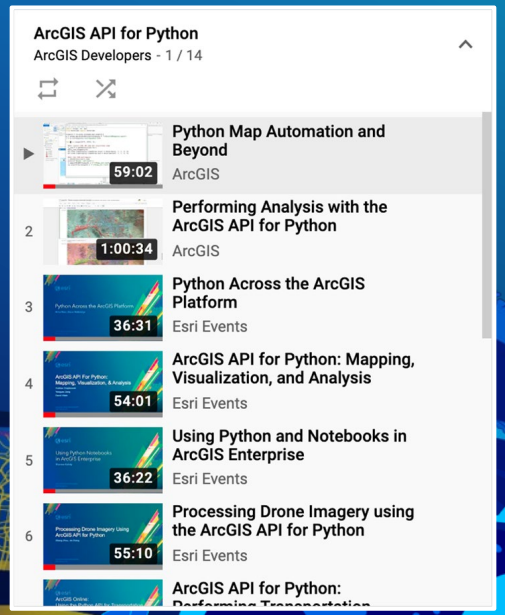
On Demand

ESRI TECHNICAL SESSION

This talk aimed for GIS developers demonstrates two different capabilities of the Python API. First, we show how the apps module of the Python API can be to build elegant GIS web apps such as Dashboards and StoryMaps that can be deployed easily in your GIS. Next we demonstrate how light weight web APIs such as FAAS (AWS Lambda for instance) be built using the API. We finish by demonstrating how even complex applications such as models for deep learning inferencing can be deployed in this fashion. Keywords: App development, Cloud GIS, FaaS, Lambda, GeoLambda

Akhil Negi, Divyansh Jha, Atma Mani





Please share your feedback via the survey available at the link below the presentation.