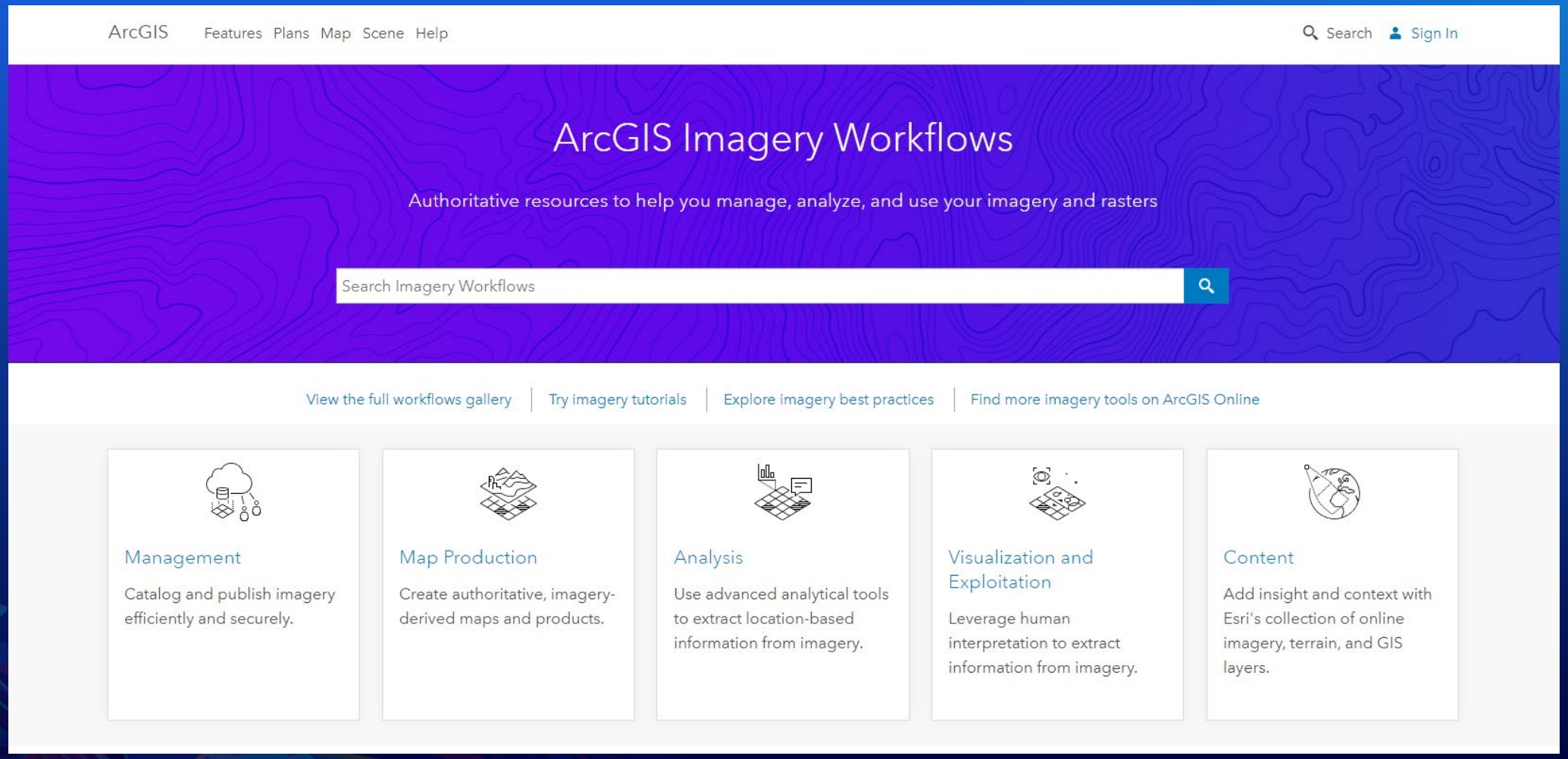
ArcGIS Workflows for Optimizing Image Management and Services in the Cloud

Peter Becker & Randall Rebello

2021 ESRI DEVELOPER SUMMIT

ArcGIS Imagery Workflows



http://www.esriurl.com/ImageryWorkflows

Image Management in ArcGIS

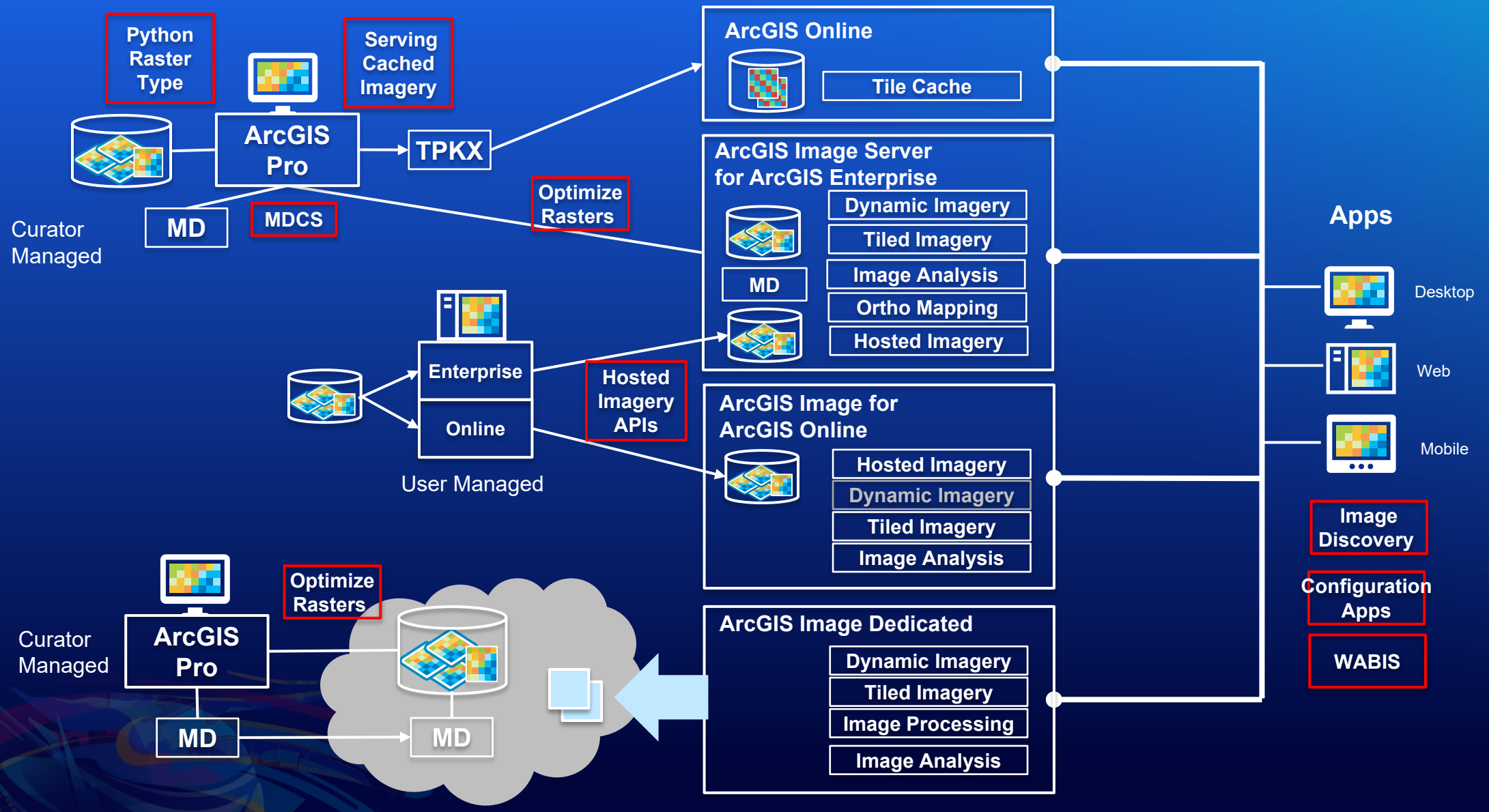


Image Management Tools

Image Management Tools

- Workflows to manage:
  - Preprocessed Orthomosaics
  - Elevation
  - Lidar
  - Multidimensional Data
  - High-Resolution Satellite Imagery
  - Medium-Resolution Imagery
  - Drone Imagery
  - Frame Camera Imagery
  - Scanned Maps
  - Browse Imagery

- All use MDCS

Mosaic Dataset Configuration Scripts (MDCS)

Typical Image Management Workflow

- Create geodatabases
- Create a mosaic dataset
- Add rasters to a mosaic dataset
- Build footprints
- Build Boundary
- Share / Publish

- Many other tools
- Each require different parameters (>150)

MDCS : Mosaic Dataset Configuration Scripts

- Simplifies automation for creating and publishing Mosaic Datasets
- Template for automation
- Documents best practice
- Advantage of using MDCS
  - Parameters described in a XML file
  - Defines tools, the parameters and order in which they are run
  - Easily repeated on multiple datasets
  - Providing Logging of the process
  - Easily configured for multiple projects easily
  - Easy to transfer to others
  - Maintained from one version of ArcGIS to other

MDCS : Mosaic Dataset Configuration Scripts

- XML File
- Defines a set of commands and their required parameters
- Each command correspond to a GP Tool related to creating Mosaic datasets

- Sample list of commands:
  - CM : Create New mosaic dataset
  - AR : Add rasters to a mosaic dataset
  - BF: Build footprints
  - BB: Build Boundary
  - DO: Define Overviews
  - BO: Build Overviews

```
<?xml version="1.0" ?><Application>
  <Name>Scanned Maps</Name>
  <Command>CM+AR+AF+SP+CV+CC+BS+CV1+AI</Command>
```

MDCS : Mosaic Dataset Configuration Scripts

- Example of parameters for a command.
  - Build Footprints
  - In Python :
    - arcpy.management.BuildFootprints(in_mosaic_dataset, {where_clause}, {reset_footprint}, {min_data_value}, {max_data_value}, {approx_num_vertices}, {shrink_distance}, {maintain_edges}, {skip_derived_images}, {update_boundary}, {request_size}, {min_region_size}, {simplification_method}, {edge_tolerance}, {max_sliver_size}, {min_thinness_ratio})
  - In MDCS : Define parameters and values.

```
<BuildFootprint>
    <where_clause/>
    <reset_footprint>GEOMETRY</reset_footprint>
    <max_data_value>255</max_data_value>
    <min_data_value>1</min_data_value>
    <approx_num_vertices>8</approx_num_vertices>
    <shrink_distance>0</shrink_distance>
    <update_boundary>UPDATE_BOUNDARY</update_boundary>
    <skip_derived_images>SKIP_DERIVED_IMAGES</skip_derived_images>
    <maintain_edges>MAINTAIN_EDGES</maintain_edges>
    <request_size>1500</request_size>
    <min_region_size/>
    <simplification_method>ENVELOPE</simplification_method>
</BuildFootprint>
```

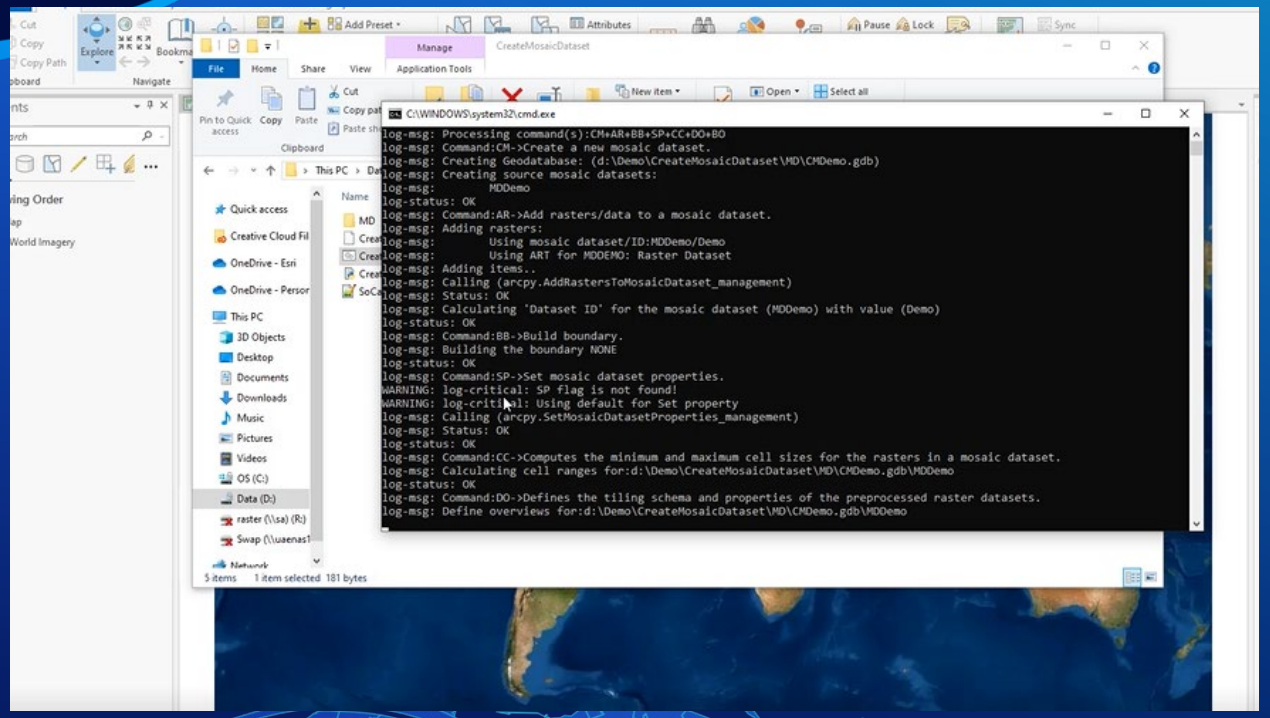
MDCS : Mosaic Dataset Configuration Scripts

- Manage data on ArcGIS Pro, or run on Server
- Publish to ArcGIS Enterprise or ArcGIS Image Dedicated
- Use example workflows and configure

- Workflows to catalog and publish Imagery
  - Managing Drone Imagery
  - Managing Elevation
  - Managing Lidar
  - Managing Multidimensional Data
  - Managing Frame Camera Imagery
  - Managing High-Resolution Satellite Imagery
  - Managing Medium-Resolution Imagery
  - Managing Preprocessed Orthophotos
  - Managing Scanned Maps
  - Managing Browse Imagery

- Publish to ArcGIS Image Server
- Write custom commands and call using MDCS.
- It is freely available on GitHub.
- You can fork / pull / push your own changes or submit an issue.

https://github.com/Esri/mdcs-py



Demo : Creation of Mosaic Dataset using MDCS.

Randall Rebello

Python Raster Types

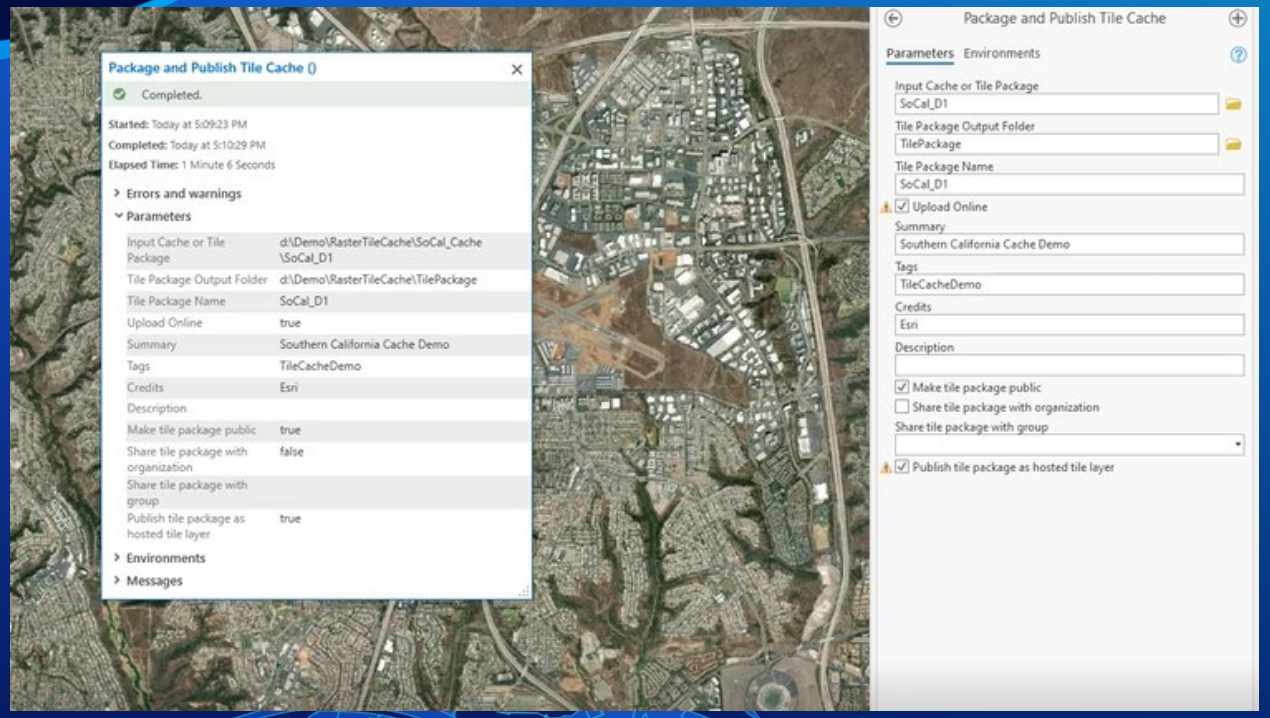
- Python raster type is customizable to your data type
- Can be customized to recognize file types, meta info

- Has a RasterTypeFactory, Builder and Crawler.
- Samples : https://github.com/Esri/raster-types
- Help: https://desktop.arcgis.com/en/arcmap/latest/manage-data/raster-and-images/custom-raster-types-in-python.htm

Serving Cached Imagery

Raster Tile Cache Tools

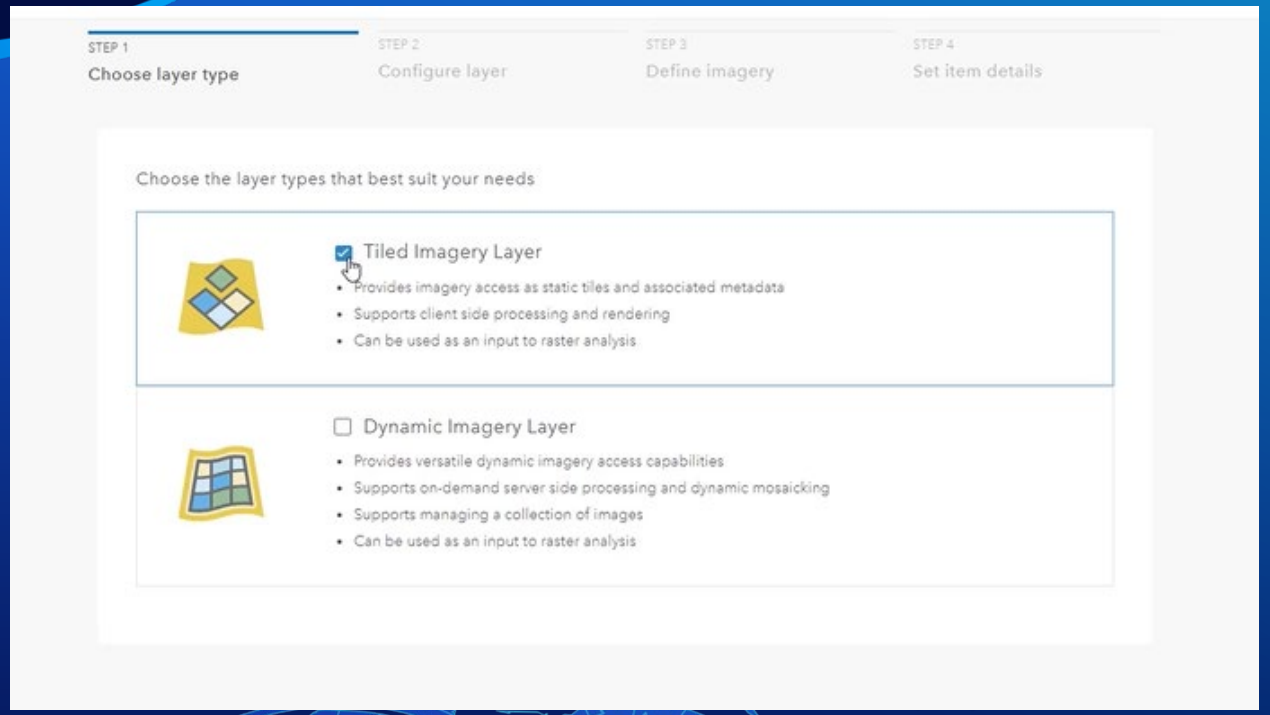
- Custom tools to create and publish TileCache
- Input can be rasters or mosaic dataset
- Primarily used for Static Data
- Easy to define scale for image generation and display
- Append to or edit existing tile cache services
- Publish to:
  - ArcGIS Enterprise
  - ArcGIS Online
  - ArcGIS Image Dedicated
- Scripts open and can be modified



Demo : Serving Cached Imagery

Randall Rebello

Hosted Imagery



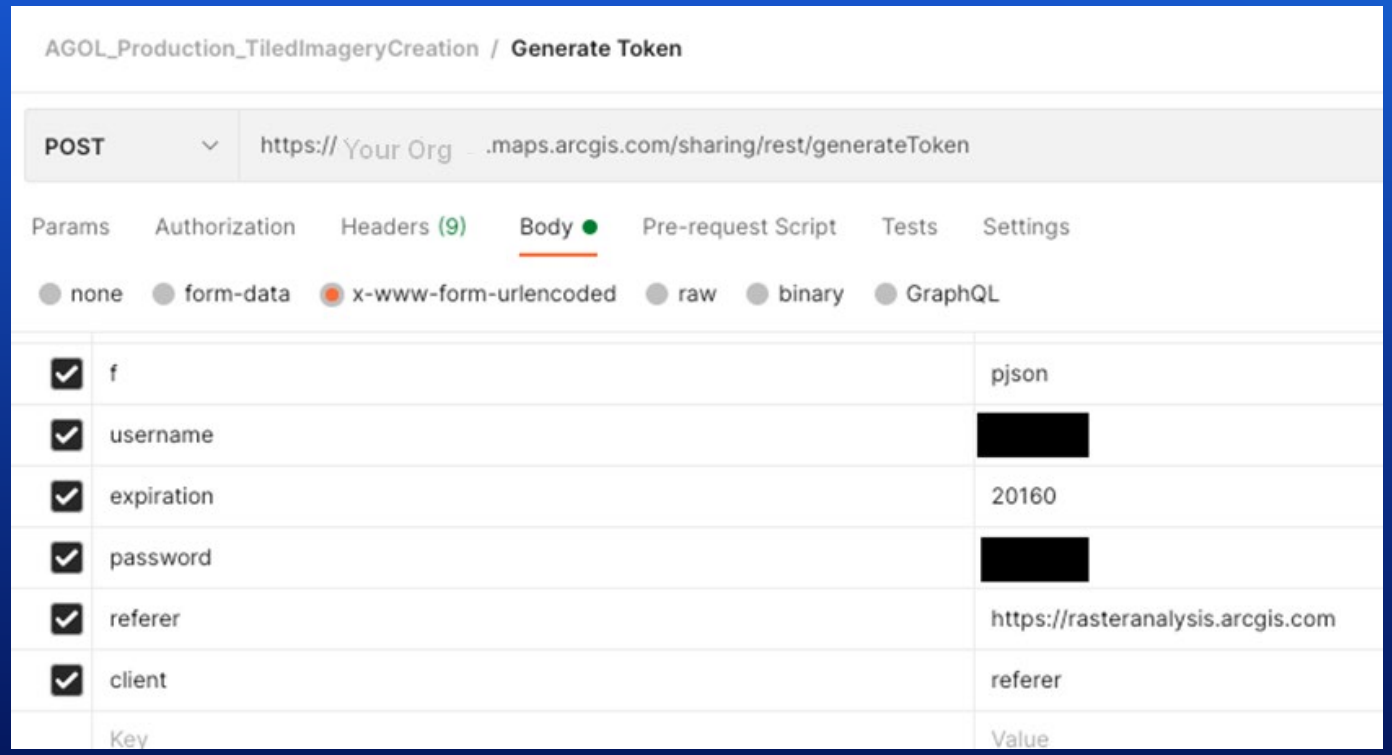
Demo : Hosted Imagery

Randall Rebello

Publishing to Hosted Imagery Programmatically

Publishing to Hosted imagery - Part 1

- Generate a token with your username/password



- The referrer URL should be consistent with your organization's org (e.g. https://rasteranalysis1.arcgis.com is hive 1)

Publishing to Hosted imagery - Part 2

- Generate the SAS connection string to the ArcGIS Online blob storage
- The blocked out the part is your ArcGIS Online username

AGOL_Production_TiledImageryCreation / **Generate Direct Access URL to AGOL blob storage**

POST https://Your Org.maps.arcgis.com/sharing/rest/content/users/[redacted]/generateDirectAccessUrl

Params | Authorization | Headers (9) | Body | Pre-request Script | Tests | Settings

none | form-data | x-www-form-urlencoded | raw | binary | GraphQL

KEY	VALUE
token	[redacted]
f	pjson
storeType	rasterStore
expiration	1440
Key	Value

- With the SAS string, you can choose any Azure blob storage API (eg Azure Python storage API) or app to upload the data

Publishing to Hosted imagery - Part 3a

- Generate a list of URLs of the files and give them as input for the CopyRaster service tool

AGOL_Production_TiledImageryCreation / **Create Tiled Imagery Layer using Copy Raster service**

POST https://rasteranalysis.arcgis.com/03e6LFX6hxm1ywlK/arcgis/rest/services/RasterAnalysisTools/GPServer/CopyRaster/submitJob

Params | Authorization | Headers (9) | Body | Pre-request Script | Tests | Settings

none | form-data | x-www-form-urlencoded | raw | binary | GraphQL

KEY	VALUE
inputRaster	{"uris":["https://arcgisazureimagery.blob.core.windows.net/gur1ywxnyloll4ys-f9aaaa8a793acc41c2f93e5749cceb2e/_images/1613165969502/ned_1.aux","https://arcgisazureimagery.blob.core.windows.net/gur1ywxnyloll4ys-f9aaaa8a793acc41c2f93e5749cceb2e/_images/1613165969502/ned_1.rrd","https://arcgisazureimagery.blob.core.windows.net/gur1ywxnyloll4ys-f9aaaa8a793acc41c2f93e5749cceb2e/_images/1613165969502/ned_1.tfw","https://arcgisazureimagery.blob.core.windows.net/gur1ywxnyloll4ys-f9aaaa8a793acc41c2f93e5749cceb2e/_images/1613165969502/ned_1.tif","https://arcgisazureimagery.blob.core.windows.net/gur1ywxnyloll4ys-f9aaaa8a793acc41c2f93e5749cceb2e/_images/1613165969502/ned_1.tif.aux.xml","https://arcgisazureimagery.blob.core.windows.net/gur1ywxnyloll4ys-f9aaaa8a793acc41c2f93e5749cceb2e/_images/1613165969502/ned_1.tif.xml","https://arcgisazureimagery.blob.core.windows.net/gur1ywxnyloll4ys-f9aaaa8a793acc41c2f93e5749cceb2e/_images/1613165969502/ned_2.aux","https://arcgisazureimagery.blob.core.windows.net/gur1ywxnyloll4ys-f9aaaa8a793acc41c2f93e5749cceb2e/_images/1613165969502/ned_2.rrd","https://arcgisazureimagery.blob.core.windows.net/gur1ywxnyloll4ys-
outputName	
f	
token	
context	
Key	

Response

Note the .OVR| .RRD not required

Publishing to Hosted imagery - Part 3b

- You also need to specify the “TilesOnly” keyword in the outputName parameter.

AGOL_Production_TiledImageryCreation / **Create Tiled Imagery Layer using Copy Raster service**

POST https://rasteranalysis.arcgis.com/03e6LFX6hxm1ywlK/arcgis/rest/services/RasterAnalysisTools/GPServer/CopyRaster/submitJob

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

KEY	VALUE
inputRaster	{"uris":["https://arcgisazureimagery.blob.core.windows.net/gur1ywxn
outputName	{"serviceProperties": { "name" : "singlemosaic_tiledimagery1", "capabilities" : "TilesOnly|Image,Mensuation"}}
f	pjson
token	[redacted]
context	{"outSR":{"wkid":3857},"rasterType":{"rasterTypeName":"Raster Datas
Key	Value

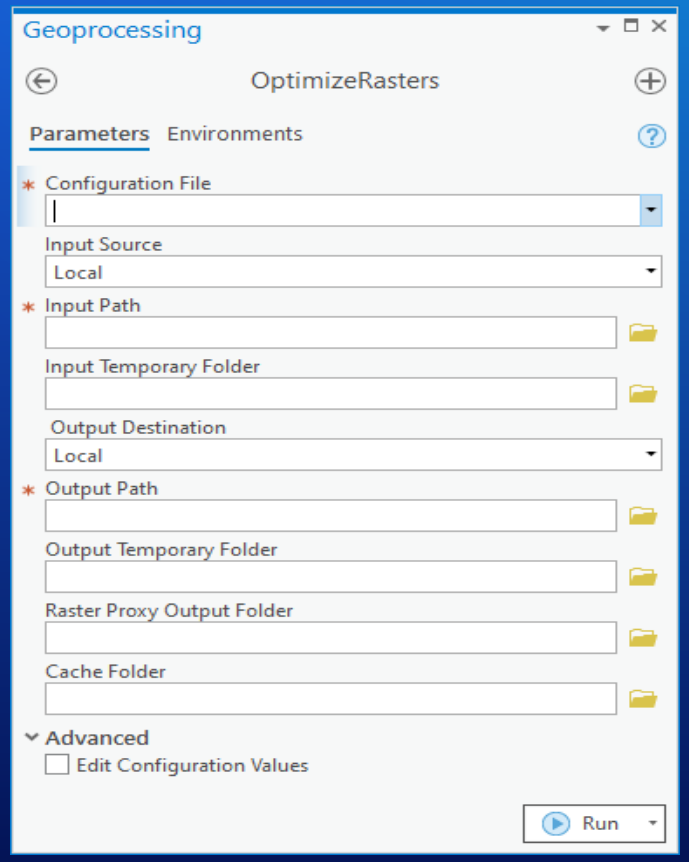
- In this example, the Copy Raster service will create the tiled imagery layer with the name specified and mosaicked the uploaded images to a single tiled imagery layer

Optimize Rasters

Optimize Rasters

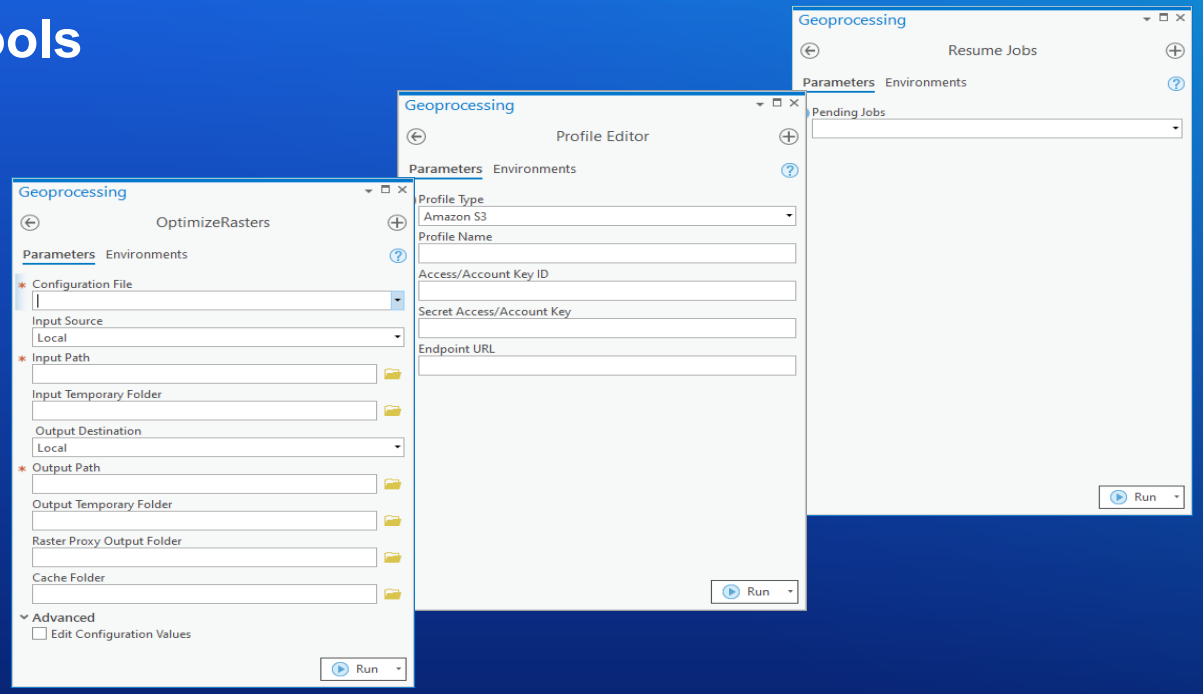
- Python based GPTool
- Converts and compresses raster datasets
  - MRF, Tiled GeoTIFF, COG
- Simplified user interface
- Can be scripted (also runs using Lambda)
- Can transfer data to cloud storage

  (Amazon, Microsoft and Google)
- Works on large datasets and large collections
- Can create Raster Proxies (optimizes access)



Optimize Rasters Cont.

- Optimize Rasters provides three GPTools
  - Optimize Rasters
  - Profile Editor
  - Resume Jobs



  - Also supports command line operations via batch files or python scripts

```
call "c:\Program Files\ArcGIS\Pro\bin\Python\envs\arcgispro-py3\python.exe" c:\Image_Mgmt_Workflows\OptimizeRasters\OptimizeRasters.py
-config=c:\Image_Mgmt_Workflows\OptimizeRasters\Templates\Landsat8_RasterProxy.xml
-input=L8/160/043/LC81600432015109LGN00
-clouddownload=true
-inputbucket=landsat-pds
-clouddownloadtype=amazon
-output=c:\temp\landsatpdsdata\L8\160\043\LC81600432015109LGN00
```

Optimize Rasters Cont.

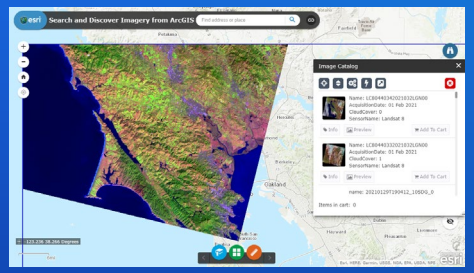
- **Configuration Template File :**
  - Define conversion parameters
  - Define compression parameters
  - Define cloud based storage Input Parameters.
  - Define cloud based storage Output Parameters.
  - ….

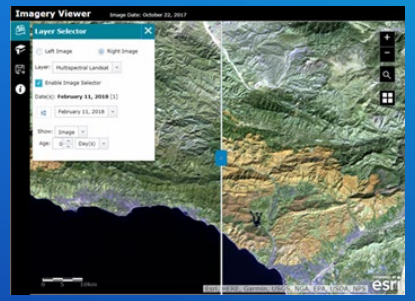
  - Source code is open.
  - Available at : https://github.com/Esri/OptimizeRasters

```xml
<?xml version="1.0" encoding="utf-8"?>
<!--Please note: Paths and extensions defined herein are considered as case-sensitive in OptimizeRasters.py-->
<OptimizeRasters>
  <Description>This template has to be used when you want to optimize and convert your imagery to MRF format with
  JPEG compression</Description>
  <Defaults>
    <!--Acceptable modes are [mrf, mrf_jpeg, tif, tif_lzw, tif_jpeg, rasterproxy. The Following items while still
    supported are no longer encouraged to be used: cachingmrf, clonemrf, splitmrf]-->
    <Mode>mrf</Mode>
    <!--File extensions considered as (Rasters). These files will not be copied from the input path-->
    <RasterFormatFilter>tif,tiff,TIF,TIFF,img,jp2,JP2,IMG,JPG,jpg,sid,SID</RasterFormatFilter>
    <!--File extensions to ignore completely while copying files/data from the input path-->
    <ExcludeFilter>tmp,rrd,idx,lrc,mrf_cache,pjp,ppng,pft,pzp,ovr,aux.xml,aux,tfw,TFW,pjg</ExcludeFilter>
    <!--'true' will scan for (Rasters) in sub-directories. Acceptable values are [true, yes, t, 1, y, false, no,
    f, 0, n]-->
    <IncludeSubdirectories>true</IncludeSubdirectories>
    <!--Compression to use on output (Rasters)-->
    <Compression>JPEG</Compression>
    <!--how the data is stored as Pixel interleave or band interleave. For cloneMRF the value should be the same
    as input (Def: Pixel)-->
    <Interleave></Interleave>
    <!--Compression quality to apply for JPEG compression (Def: 85)-->
    <Quality>95</Quality>
    <!--LERC precision to apply for LERC compression (Def: 0.5 for int data and 0.001 for float data)-->
    <LERCPrecision></LERCPrecision>
    <!--Build pyramids? Acceptable values are [true, yes, t, 1, y, false, no, f, 0, n, only, external]-->
    <BuildPyramids>true</BuildPyramids>
    <!--Pyramid levels to create (Def:  2)-->
    <PyramidFactor></PyramidFactor>
    <!--Pyramid sampling [nearest,avg,average,gauss,cubic,cubicspline,lanczos,average_mp,average_magphase,mode]
    (Def: average)-->
    <!--The avg sampling method is only supported by mrf format, which is similar to average, but is a pure
    averaging of every 2x2 pixels from top left, which is slightly faster -->
    <PyramidSampling>avg</PyramidSampling>
    <!--Pyramid compression [jpeg, lzw, deflate] (Def: jpeg)-->
    <PyramidCompression>LERC</PyramidCompression>
    <!--No data value. If undefined/empty value -a_nodata will not be applied (Def: undefined)-->
    <NoDataValue></NoDataValue>
    <!--Output tile size-->
    <BlockSize>512</BlockSize>
    <!--This needs to be specified when generating caching or clone MRF, the value should be based on the input
    raster pyramid factor (Def: 2)-->
    <Scale></Scale>
    <!--If 'True' raster output extensions will not be renamed to 'mrf'. Acceptable values are [true, yes, t, 1,
    y, false, no, f, 0, n]-->
    <KeepExtension>false</KeepExtension>
    <!--Simultaneous threads to use for parallel processing/instances of gdal_translate/gdal_addo/etc (Def: 10)-->
    <Threads>4</Threads>
    <!--Path to save log/completed job files-->
```

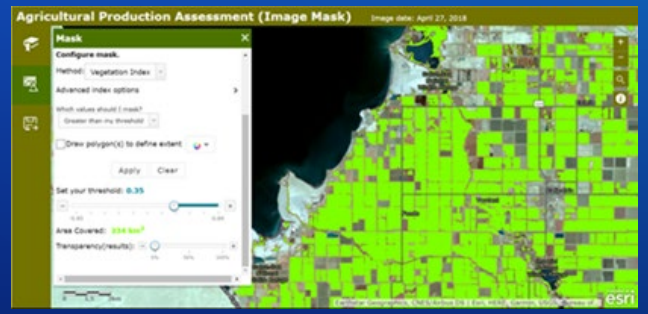
Other Tools in Imagery Workflows

Resources for web app development:



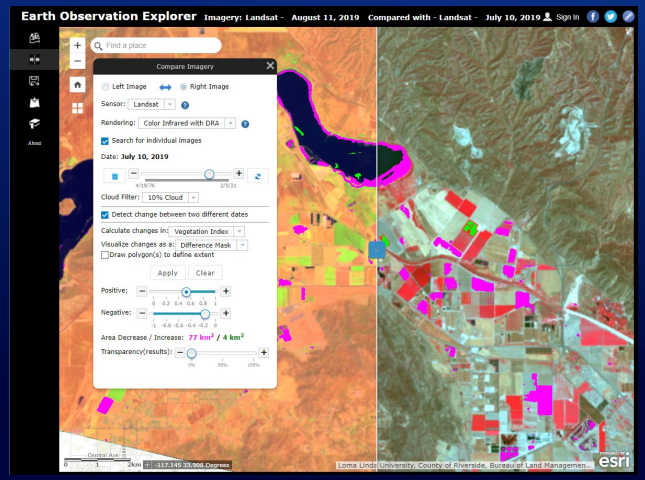


- **Image Services - Catalog widget:**
  https://esriurl.com/ImageDiscoveryDemoApp
- **Web App Builder for Image Services (WABIS)**
  - **Custom widgets to use in Web AppBuilder.**
  - **Built specifically for Image Services** https://esriurl.com/WABIS



- **Configuration Templates**

  Imagery Viewer: https://github.com/Esri/imagery-viewer-template
  - **Image Visit:** https://github.com/Esri/image-visit-template
  - **Image Change detection:** https://github.com/Esri/image-change-detection-template

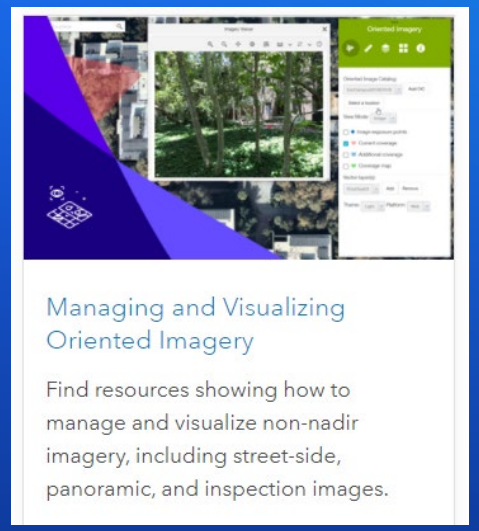


- **Imagery Apps GitHub repo:** https://esriurl.com/ImageryAppsSourceCode
- **Unlock Earth's Secrets page:** https://esriurl.com/UnlockEarthsSecrets
- **Configurable app repos:** https://esriurl.com/ImageryConfigurableAppsBlog

Oriented Imagery

Access imagery at any angle for any location

- **Pick location and find best available imagery**
- **Multiple sources for imagery**
- **Integrate into GIS**
- **Enable remote monitoring and inspection workflows**
- **Measurements** **(When suitable metadata available)**
- **Superposition of vectors & fuse into 3D scenes**



Inspection

Street View

Oblique / Drone & Aerial

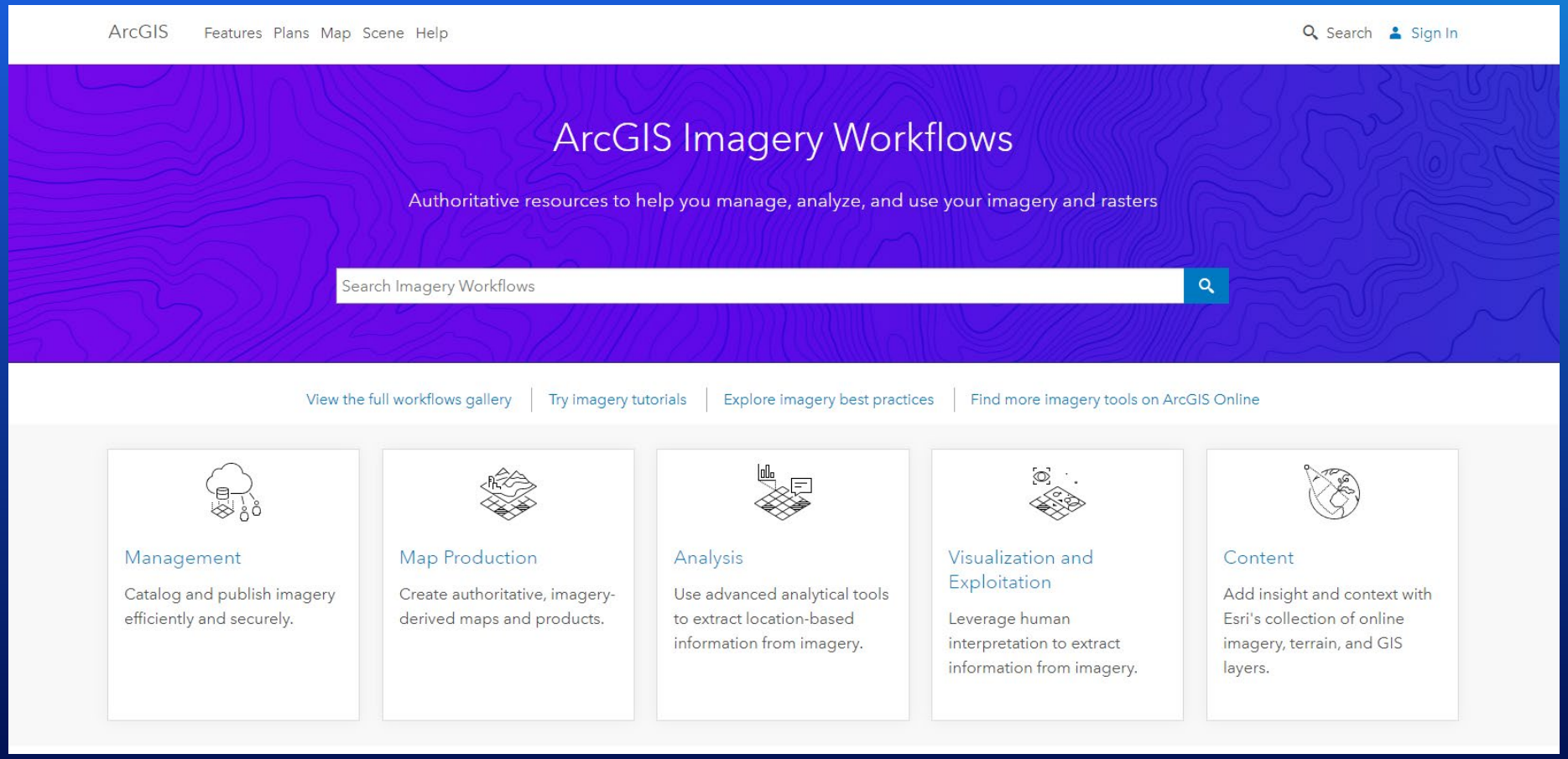
See Demo Theater:
Introduction to Oriented Imagery

- **Python Imagery Types to ingest custom meta data and images**

Panorama

Video

ArcGIS Imagery Workflows



http://www.esriurl.com/ImageryWorkflows

Please provide your feedback for this session by clicking on the session survey link directly below the video.

esri®

THE SCIENCE OF WHERE®

Copyright © 2021 Esri. All rights reserved.