Working w/ Data in Products

Max Payson & Thomas Fair

2020 ESRI DEVELOPER SUMMIT | Palm Springs, CA

Purpose

Tips & tricks for working with user data across multiple tenants

Disclaimers

Focus will be on…

Vector data & feature services (they're the most common)

High-level workflows (check out other sessions to dive in!)

Agenda

Overview

Creating services

Provisioning services

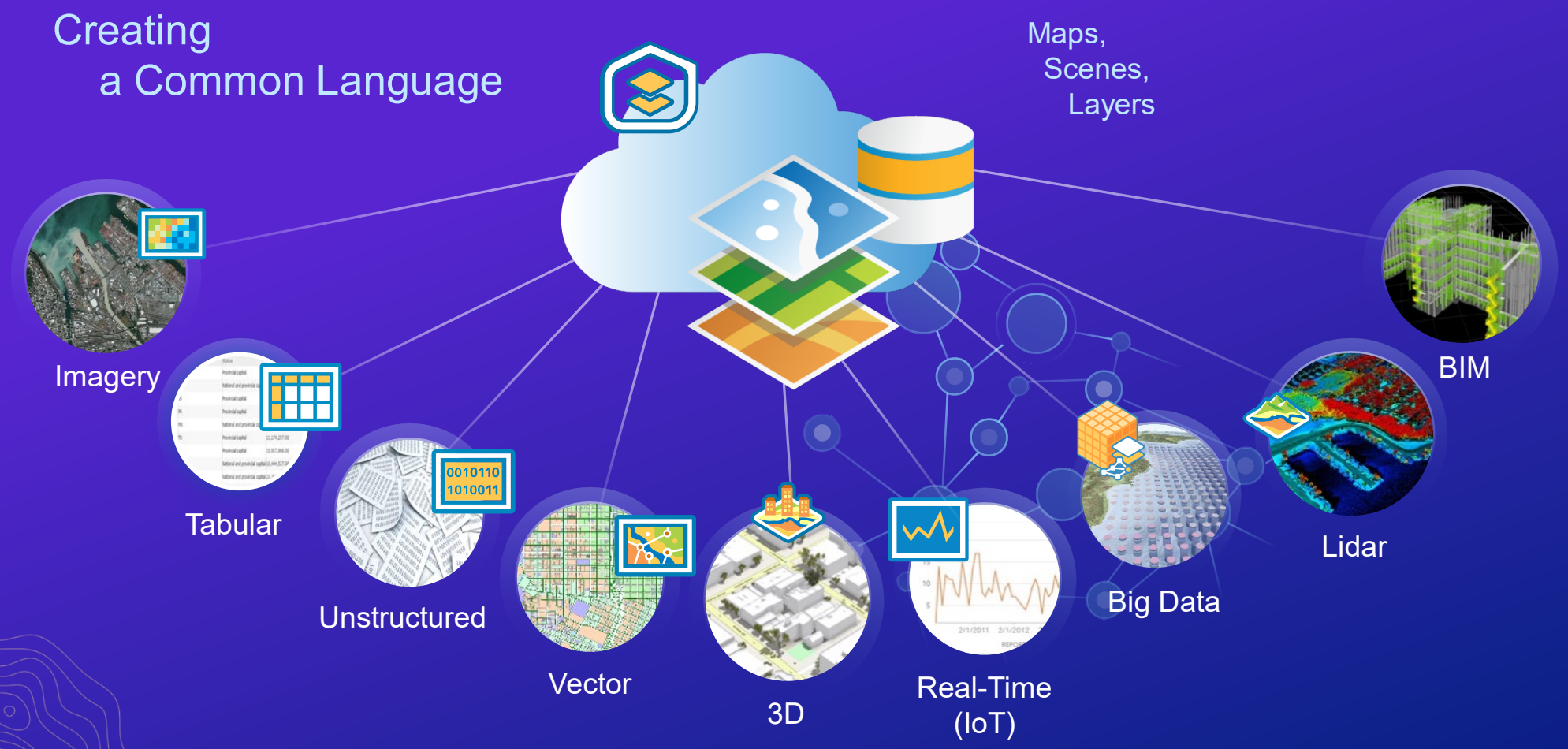
Accessing services

Overview

Data in ArcGIS

ArcGIS Data

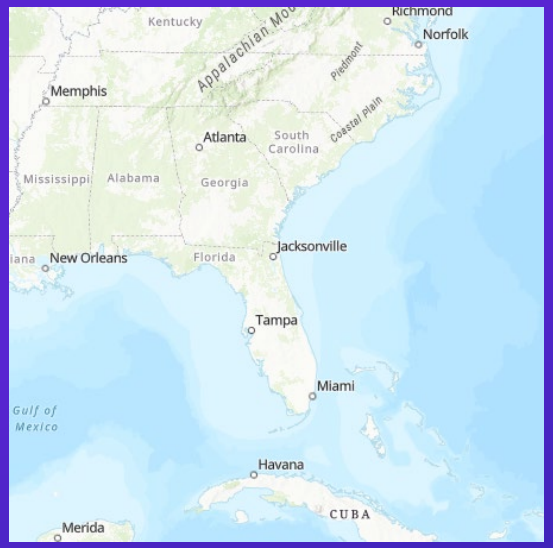
Store, manage, and integrate all types of data



Layers

Layer Type	REST API Service	Hosted In		Cached
		ArcGIS Online	ArcGIS Enterprise	
Feature Layer	Feature Service	X	X	
Tile Layer	Map Service	X	X	X
Vector Tile Layer	Vector Tile Service	X	X	X
Dynamic Map Layer	Map Service		X	
Image Layer	Image Ser		X	
Scene Layer	Scene Service	X	X	X

Tile Layer



Display basemaps and other complex datasets that change infrequently

Represented as images

Tiles can be kept in sync with feature layers

Vector Tile Layer

Display basemaps and other complex datasets that change infrequently

Represented as vectors

Dynamic Map Layer

Display complex data sets that change frequently or need complex rendering requirements

Represented as images

Image Layer

Render, analyze, and dynamically interact with imagery data

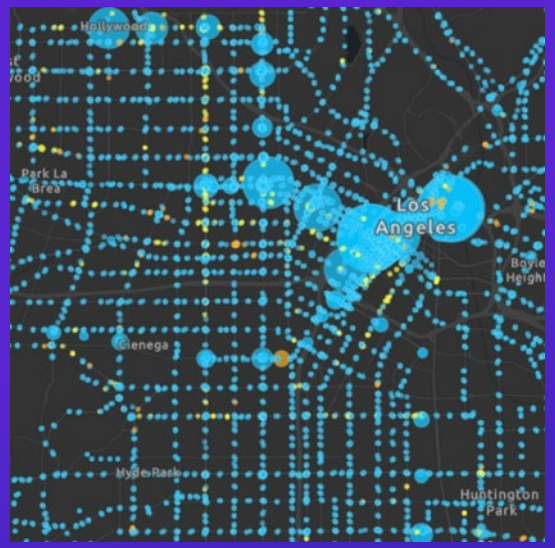
Represented as images

Scene Layer

Display and render 3D datasets

Represented by I3S spec

Feature Layer



Query, render, and edit vector geographic information

Represented as JSON

Web Maps


```
{
    "operationalLayers": [
        {
            "id": "Boundary_1356",
            "layerType": "ArcGISFeatureLayer",
            "url": "https://services3.arcgis.com/GVgbJbqm8hXASVYi/arcgis/rest/services/Boundary/FeatureServer/0",
            "visibility": true,
            "opacity": 1,
            "title": "Boundary",
            "itemId": "30cf3b4c11c5408d87f255ba4e9aa611",
            "disablePopup": true
        },
        {
            "id": "Parks_and_Open_Space_2992",
            "layerType": "ArcGISFeatureLayer",
            "url": "https://services3.arcgis.com/GVgbJbqm8hXASVYi/arcgis/rest/services/Parks_and_Open_Space/FeatureS
            "visibility": true,
            "opacity": 0.51,
            "title": "Parks and Open Space",
            "itemId": "f2ea5d874dad427294641d2d45097c0e",
            "layerDefinition": {
                "drawingInfo": {
                    "renderer": {
                        "visualVariables": [
                            {
                                "type": "sizeInfo",
                                "target": "outline",
                                "expression": "view.scale",
                                "valueExpression": "$view.scale",
                                "stops": [
                                    {
                                        "size": 1.5,
                                        "value": 9501
                                    },
                                    /* ... */
                                ]
                            }
                        ],
                        "type": "uniqueValue",
                        "field1": "TYPE",
```


2D map that you can create, style, and share between apps

Represented by JSON object and defined by the Web Map specification

Contains configuration settings for the map extent, layers, styles, pop-ups…

Web Scenes


```
{
    "operationalLayers": [
        {
            "id": "14a3b7e8769-layer5",
            "opacity": 1,
            "title": "Topographic Map 3D",
            "url": "https://tiles.arcgis.com/tiles/P3ePLMYs2RVChkJx/arcgis/rest/services/Portland_v1/MapServer",
            "visibility": true,
            "layerType": "ArcGISTiledMapServiceLayer",
            "itemId": "77618c5b716a4417a2e72d75ccc62f30"
        },
        {
            "id": "159085c4486-layer-4",
            "showLegend": true,
            "opacity": 1,
            "disablePopup": false,
            "title": "Buildings",
            "url": "https://tiles.arcgis.com/tiles/P3ePLMYs2RVChkJx/arcgis/rest/services/Buildings_Portland/SceneSer
            "visibility": true,
            "layerType": "ArcGISSceneServiceLayer",
            "itemId": "d96440302be847f6aa6e6ac0012c53b0",
            "layerDefinition": {
                "elevationInfo": {
                    "mode": "absoluteHeight"
                },
                "drawingInfo": {
                    "renderer": {
                        "authoringInfo": {},
                        "type": "simple",
                        "label": "",
                        "symbol": {
                            "type": "MeshSymbol3D",
                            "symbolLayers": [
                                {
                                    "material": {
                                        "color": [
                                            255,
                                            255,
                                            255
                                        ]
```


Like 2D but for 3D data (urban and natural environments, buildings…)

Represented by JSON objects defined by the Web Scene Specification

Contains configuration settings for the camera height / angle, layers, styles…

Items

Apps

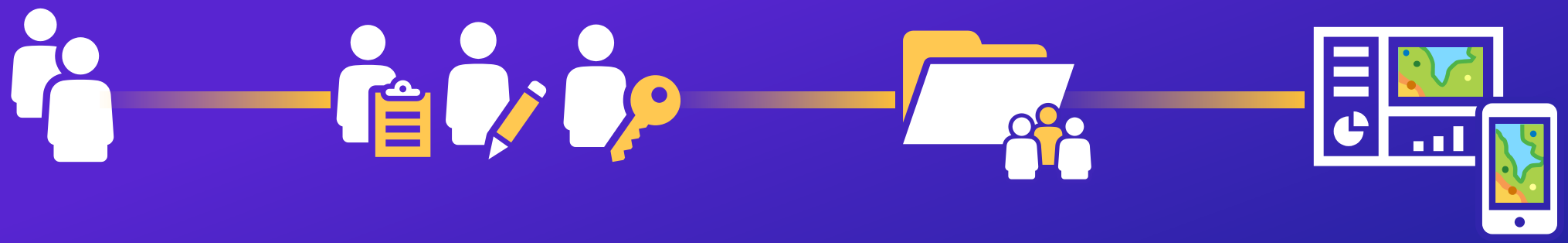
Maps & scenes

Layers

Object storage

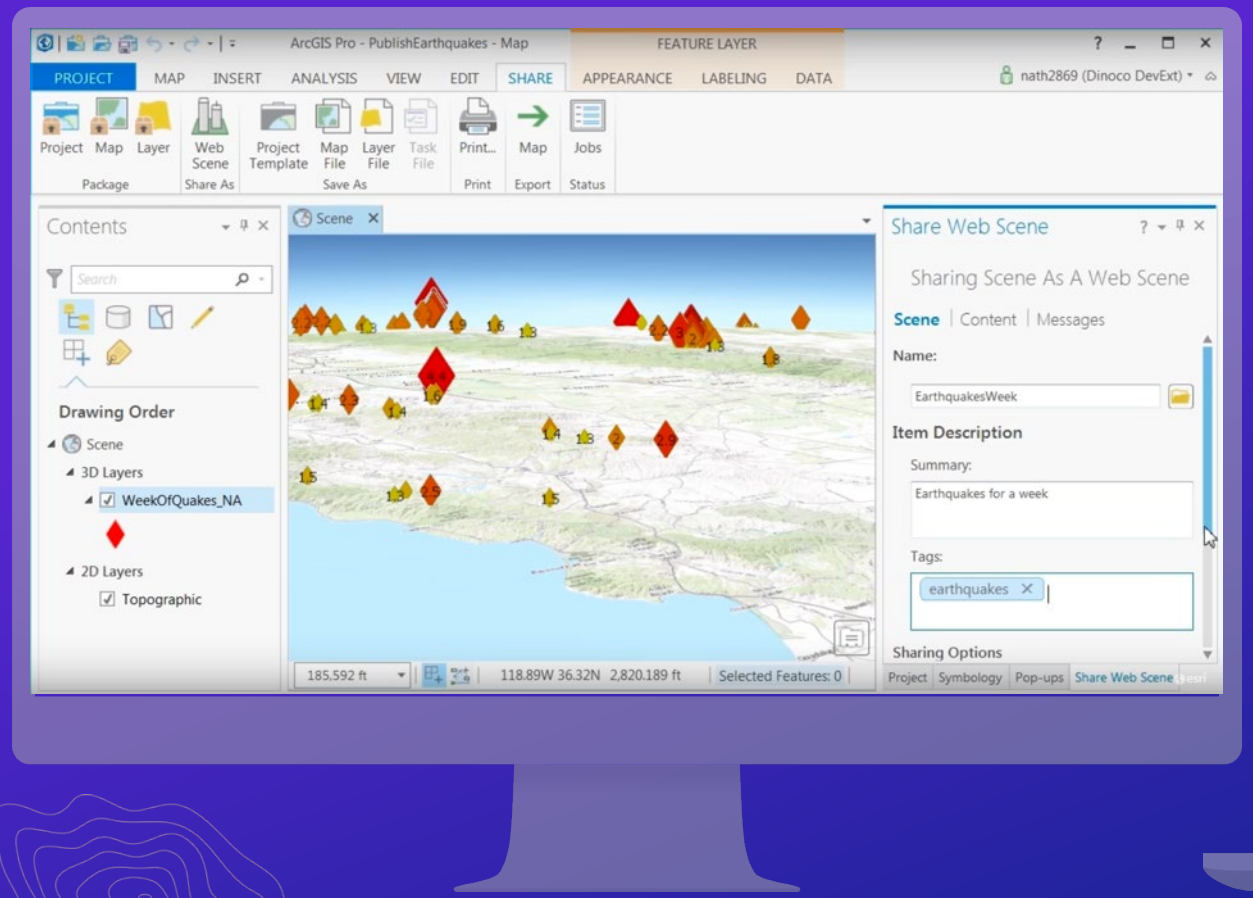
Services / tools

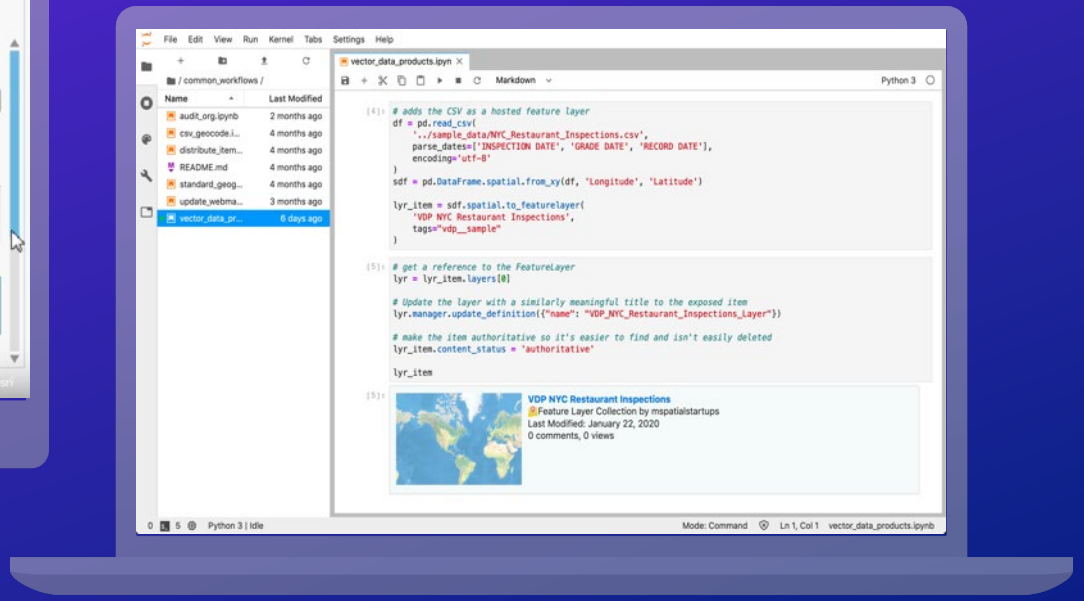
Security Model



Users **Roles & permissions** **Groups** **Shared items**

Authoring


ArcGIS Pro - PublishEarthquakes - Map
FEATURE LAYER
PROJECT
MAP
INSERT
ANALYSIS
VIEW
EDIT
SHARE
APPEARANCE
LABELING
DATA
nath2869 (Dinoco DevExt)
Project
Map
Layer
Web Scene
Project Template
Map File
Layer File
Task File
Print...
Map
Jobs
Package
Share As
Save As
Print
Export
Status
Contents
Search
Drawing Order
Scene
3D Layers
WeekOfQuakes_NA
2D Layers
Topographic
Scene
185.592 ft
118.89W 36.32N 2,820.189 ft
Selected Features: 0
Share Web Scene
Sharing Scene As A Web Scene
Scene
Content
Messages
Name:
EarthquakesWeek
Item Description
Summary:
Earthquakes for a week
Tags:
earthquakes
Sharing Options
Project
Symbology
Pop-ups
Share Web Scene


File
Edit
View
Run
Kernel
Tabs
Settings
Help
vector_data_products.ipyn
Markdown
Python 3
/ common_workflows /
Name
Last Modified
audit_org.ipynb
2 months ago
csv_geocode.i...
4 months ago
distribute_item...
4 months ago
README.md
4 months ago
standard_geog...
4 months ago
update_webma...
3 months ago
vector_data_pr...
6 days ago
# adds the CSV as a hosted feature layer
df = pd.read_csv(
'../sample_data/NYC_Restaurant_Inspections.csv',
parse_dates=['INSPECTION DATE', 'GRADE DATE', 'RECORD DATE'],
encoding='utf-8'
)
sdf = pd.DataFrame.spatial.from_xy(df, 'Longitude', 'Latitude')
lyr_item = sdf.spatial.to_featurelayer(
'VDP NYC Restaurant Inspections',
tags="vdp__sample"
)
# get a reference to the FeatureLayer
lyr = lyr_item.layers[0]
# Update the layer with a similarly meaningful title to the exposed item
lyr.manager.update_definition({"name": "VDP_NYC_Restaurant_Inspections_Layer"})
# make the item authoritative so it's easier to find and isn't easily deleted
lyr_item.content_status = 'authoritative'
lyr_item
VDP NYC Restaurant Inspections
Feature Layer Collection by mspatialstartups
Last Modified: January 22, 2020
0 comments, 0 views
Python 3 | Idle
Mode: Command
Ln 1, Col 1
vector_data_products.ipynb

Creating Services

Common Challenges

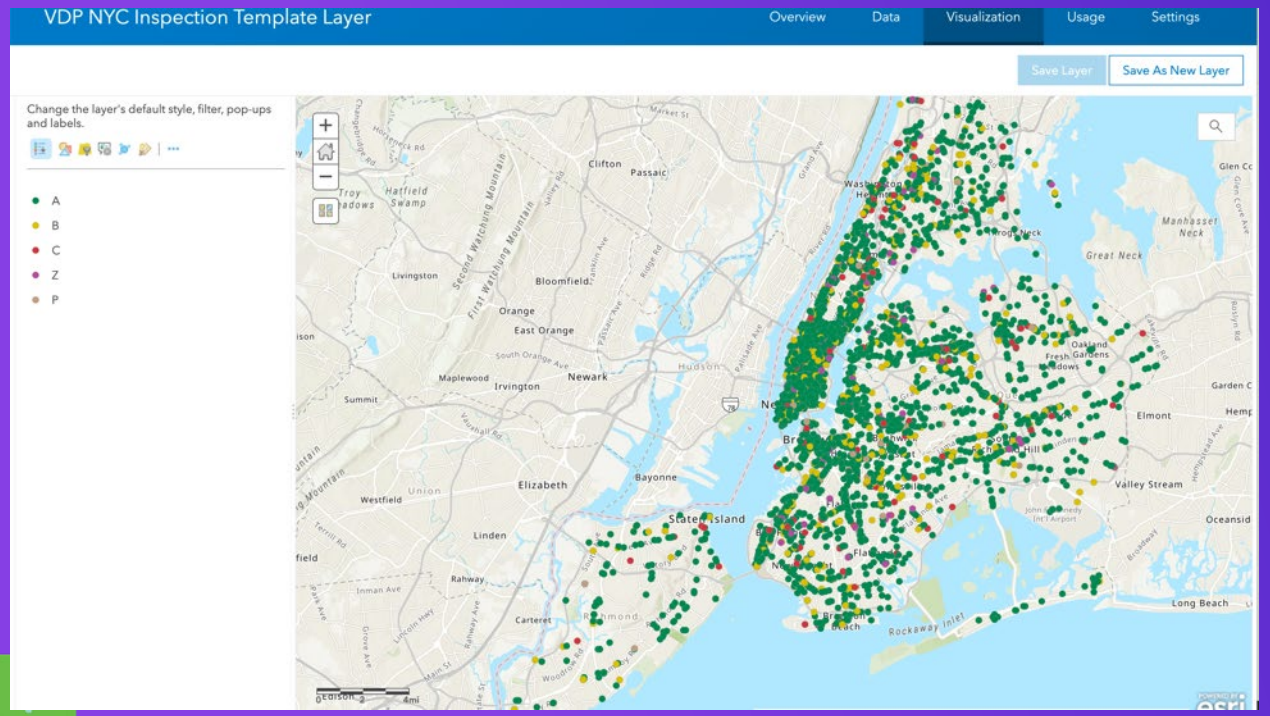
Working with many user owned datasets

Items should have a consistent experience

Automation!

Suggested Workflow

1. Author “template” services in UI
2. Create new services
3. Copy template properties



Author templates

Quick demo

Layer Properties

Item properties

(description, tags, license…)


```
{
  "id": "b71c61e14425426984ec8a1a94bb77f4",
  "owner": "mpayson_startups",
  "orgId": "q7zPNeKmTWeh7Aor",
  "created": 1573506895000,
  "isOrgItem": true,
  "modified": 1576031962000,
  "guid": null,
  "name": null,
  "title": "VDP NYC Inspection Template Layer",
  "type": "Feature Service",
  "typeKeywords": […
  ],
  "description": "<div>A very detailed, riveting descriptio
  "tags": [
    "vdp",
    "nyc",
    "inspection",
    "template",
    "map"
  ],
  "snippet": "Template representation of NYC restaurant ins
```


https://<content-url>/items/<itemId>?f=json

Item data

(popups, clustering, filters…)


```
{
  "layers": [
    {
      "id": 0,
      "popupInfo": {
        "title": "NYC Restaurant Inspections",
        "fieldInfos": [
          {
            "fieldName": "FID",
            "label": "FID",
            "isEditable": false,
            "tooltip": "",
            "visible": false,
            "stringFieldOption": "textbox"
          },
          {
            "fieldName": "CAMIS",
            "label": "CAMIS",
            "isEditable": true,
            "tooltip": "",
            "visible": true,
            "stringFieldOption": "textbox",
```


https://<content-url>/items/<itemId>/data?f=json

Service properties

(visible range, editing…)


```
  "advancedQueryCapabilities": {…
  },
  "useStandardizedQueries": false,
  "geometryType": "esriGeometryPoint",
  "minScale": 1155582,
  "maxScale": 0,
  "extent": {
    "xmin": -8264796.2973010931,
    "ymin": 4940321.7979350407,
    "xmax": -8204437.0359609667,
    "ymax": 4999145.4596300479,
    "spatialReference": {
      "wkid": 102100,
      "latestWkid": 3857
    }
  },
  "drawingInfo": {
    "renderer": {
      "type": "uniqueValue",
      "field1": "GRADE",
      "defaultSymbol": {
        "color": [
```

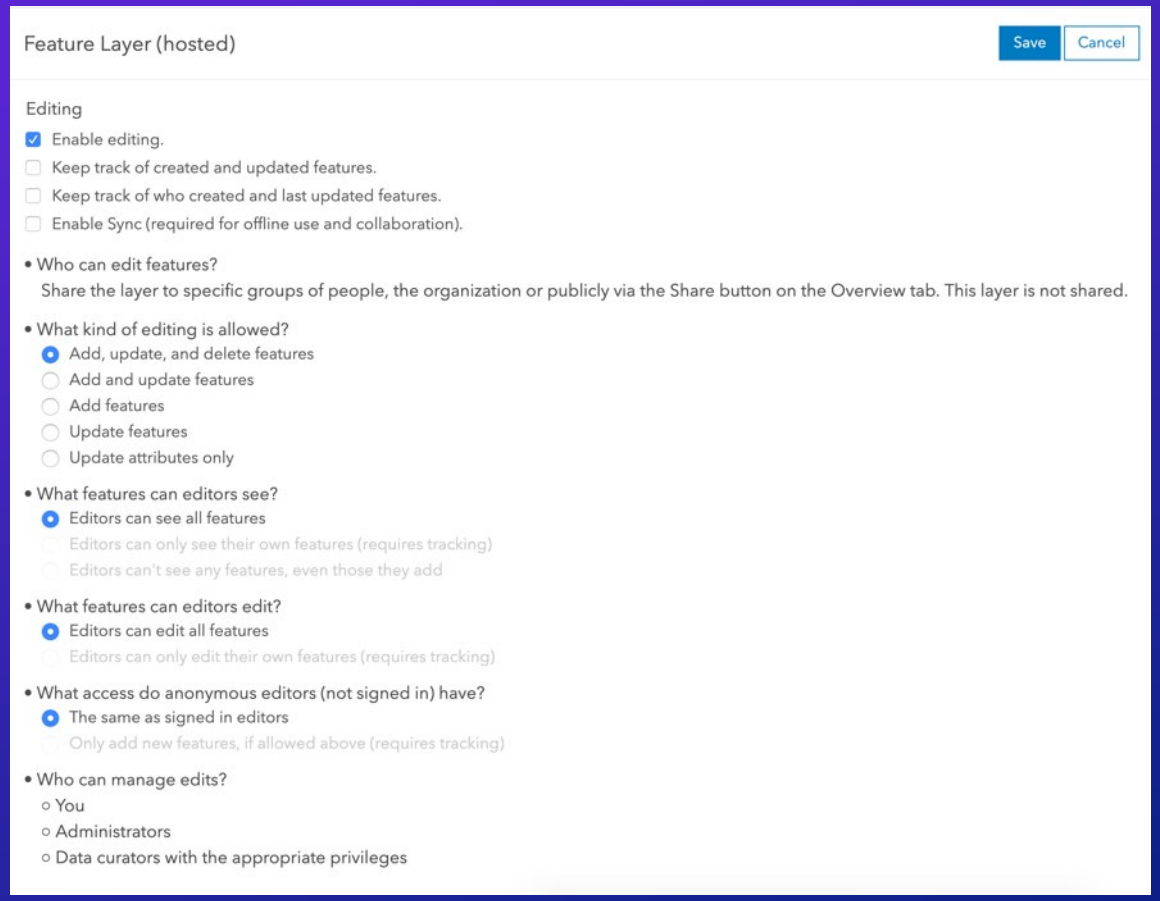

https://<catalog-url>/<service-name>/FeatureServer/<id>?f=json

Editing Properties

What kind of editing

Permissions & tracking

Who can edit via sharing



https://<adminservicecatalog-url>/services/<serviceName>/FeatureServer/updateDefinition

Map & App Properties

Item properties

(description, tags, license…)


```
{
  "item": {
    "id": "15284c7826b048568817bda952972e17",
    "owner": "mpayson_startups",
    "created": 1575304815000,
    "isOrgItem": true,
    "modified": 1575304983000,
    "guid": null,
    "name": null,
    "title": "VDP NYC Inspection Template Web App",
    "type": "Web Mapping Application",
    "typeKeywords": […
    ],
    "description": null,
    "tags": [
      "vdp",
      "nyc",
      "inspection",
      "template",
      "map"
    ],
    "snippet": null,
```


https://<content-url>/items/<itemId>?f=json

Item data

(settings, UI configuration…)


```
{
  "theme": {
    "name": "FoldableTheme",
    "styles": [
      "default",
      "black",
      "blue",
      "cyan",
      "green",
      "purple",
      "red",
      "yellow"
    ],
    "version": "2.14",
    "sharedTheme": {
      "isPortalSupport": true,
      "useHeader": false,
      "useLogo": false
    }
  },
  "portalUrl": "http://startups.maps.arcgis.com",
  "appId": "",
```


https://<content-url>/items/<itemId>/data?f=json

Create Service

(1) Item information **{...}**

(2.a) Create a new service

https://<usercontent-url>/createService & https://<adminservicecatalog-url>/services/<serviceName>/FeatureServer/addToDefinition

(2.b) Publish data file

https://<usercontent-url>/addItem & https://<usercontent-url>/publish

(3) Copy properties

https://<usercontent-url>/items/<itemId>/update & https://<adminservicecatalog-url>/services/<serviceName>/FeatureServer/updateDefinition

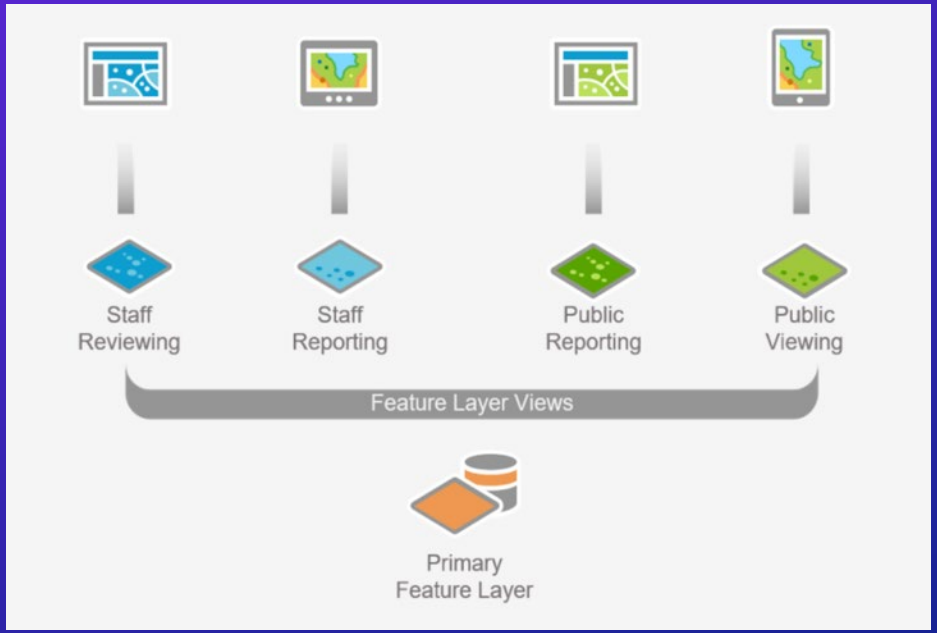
Use the Python API, Solution.js, or other wrappers (services not fully documented)

Feature Layer Views

Support different workflows or users

Settings inherited and separate

Usage, updating, editing, & management



create_service.ipynb | share_service.ipynb

Code | Python 3

Define the template item and get its properties

[4]:
```
t_item = Item(gis, 'b71c61e14425426984ec8a1a94bb77f4')

# item-level properties
t_item_props = dict(t_item)
dirpath = tempfile.mkdtemp()
thumbnail_path = t_item.download_thumbnail(dirpath)

# item data-level properties
t_item_data_props = t_item.get_data()

# service-level properties
t_lyr_props = dict(t_item.layers[0].properties)
```

[5]:
```
t_item.title
```

[5]:
```
'VDP NYC Inspection Template Layer'
```

Create services

An empty service

[ ]:
```
res = gis.content.clone_items([t_item], folder='@epcdemos', copy_data=False)
empty_lyr_item = res[0]
empty_lyr_item
```

Create services

Quick demo

Update Tips

Data

Apply edits: transactional edits

Overwrite: ETL full dataset

Append: large updates or ETL

Schema

Update definition: add fields, set domains, etc

Item properties

Update the template then copy to services

Provisioning Services

Common Challenges

Give users access to their data

Understand who administers services

Understand who pays for services

Three Options

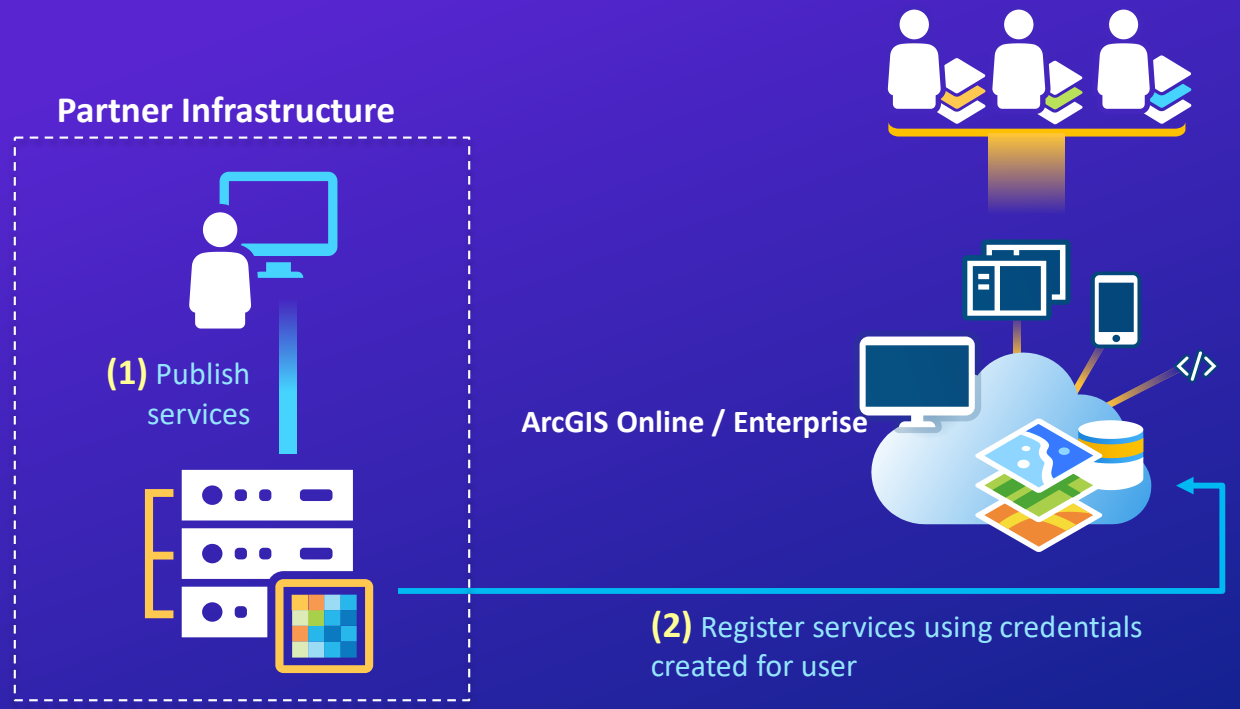
- Register service in user’s GIS
- Share service via group
- Publish data directly to user’s GIS

Register Service

User controls access

Developer controls admin & pays for service

Works with any service type



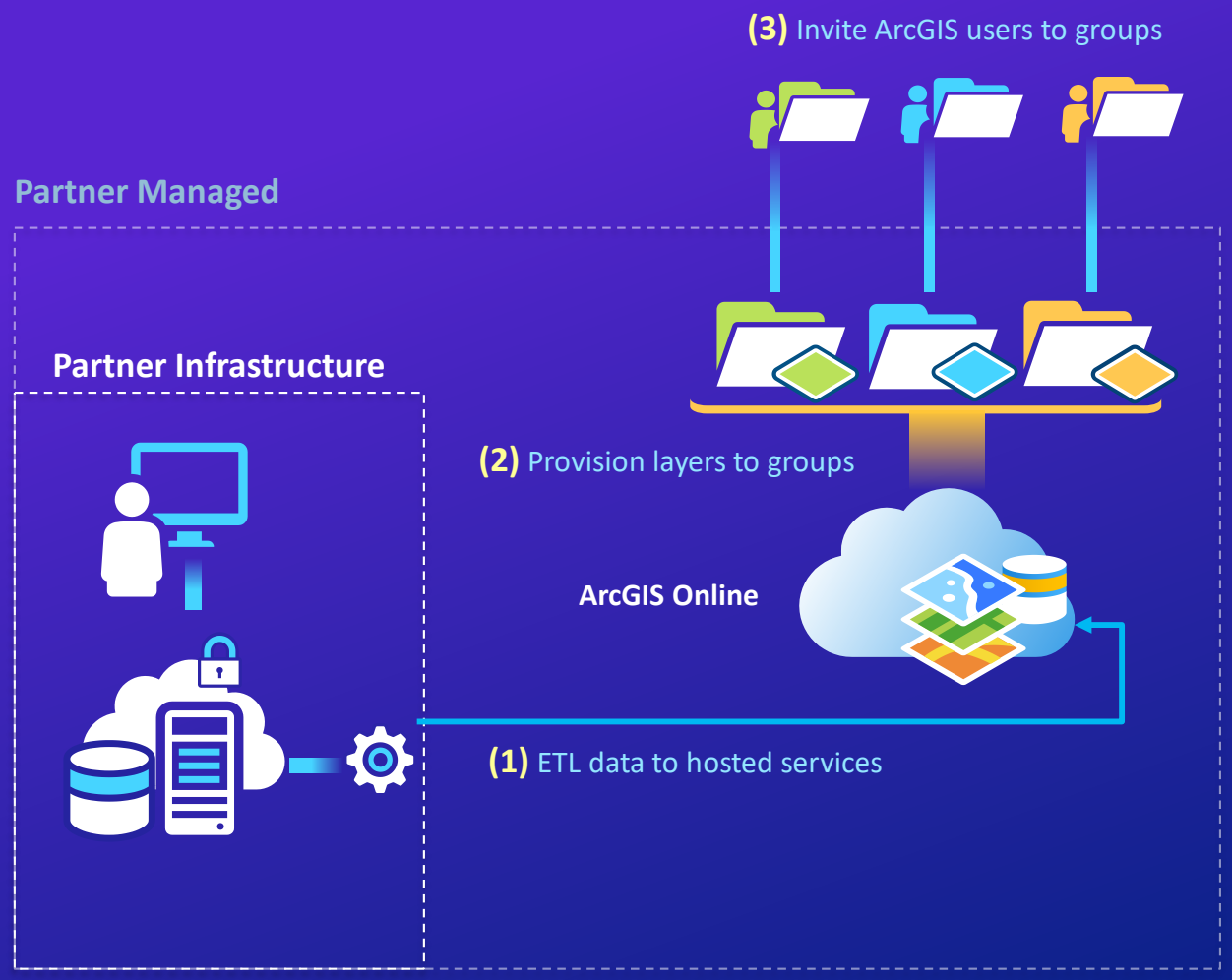
*For partners

Share Service

Developer controls access & admin

Developer pays for service

Works with hosted or registered service in Online



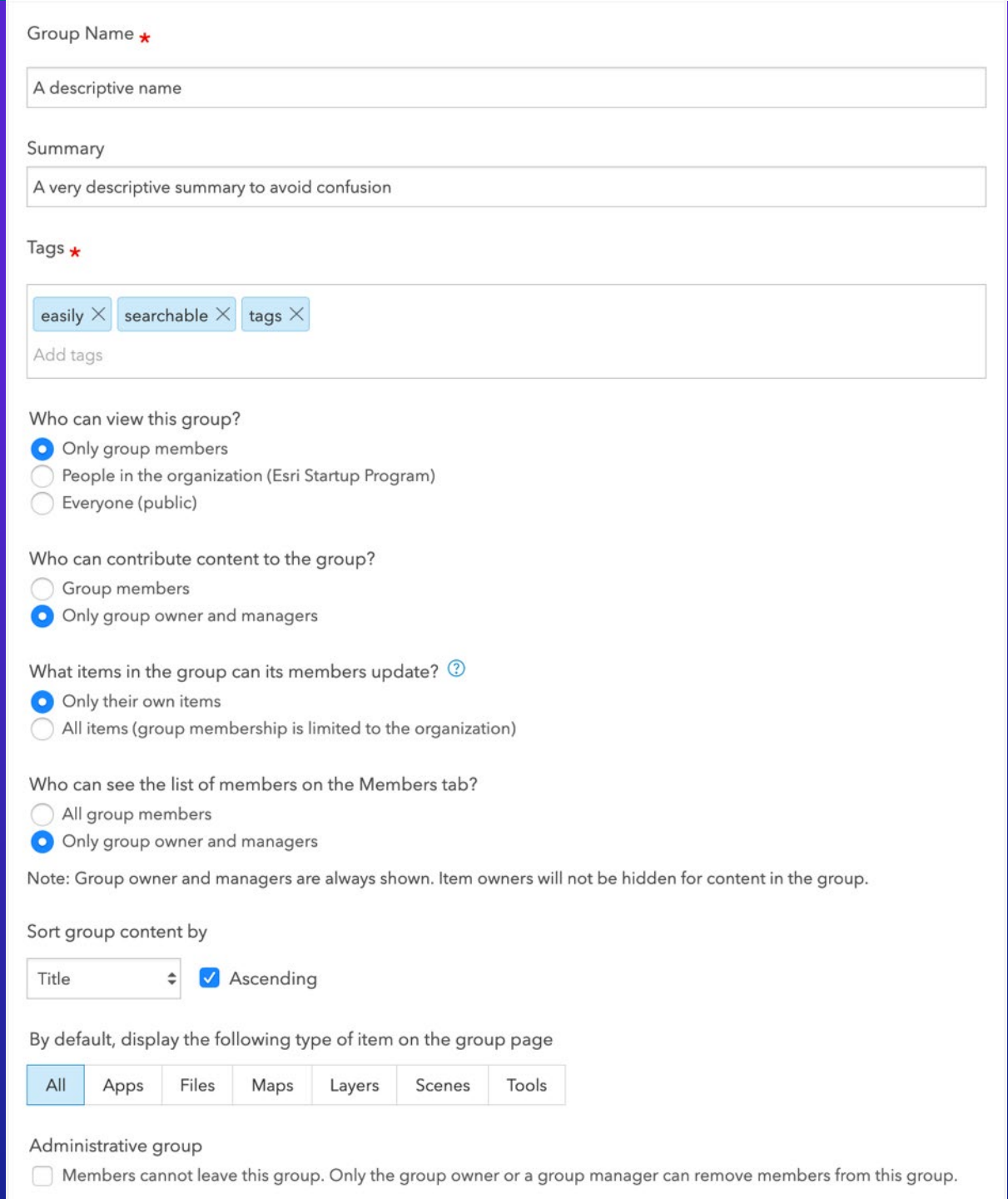
*For partners

Tips for Groups

Keep them private and limit possible user actions

Give searchable names & tags

Great for developer-managed users

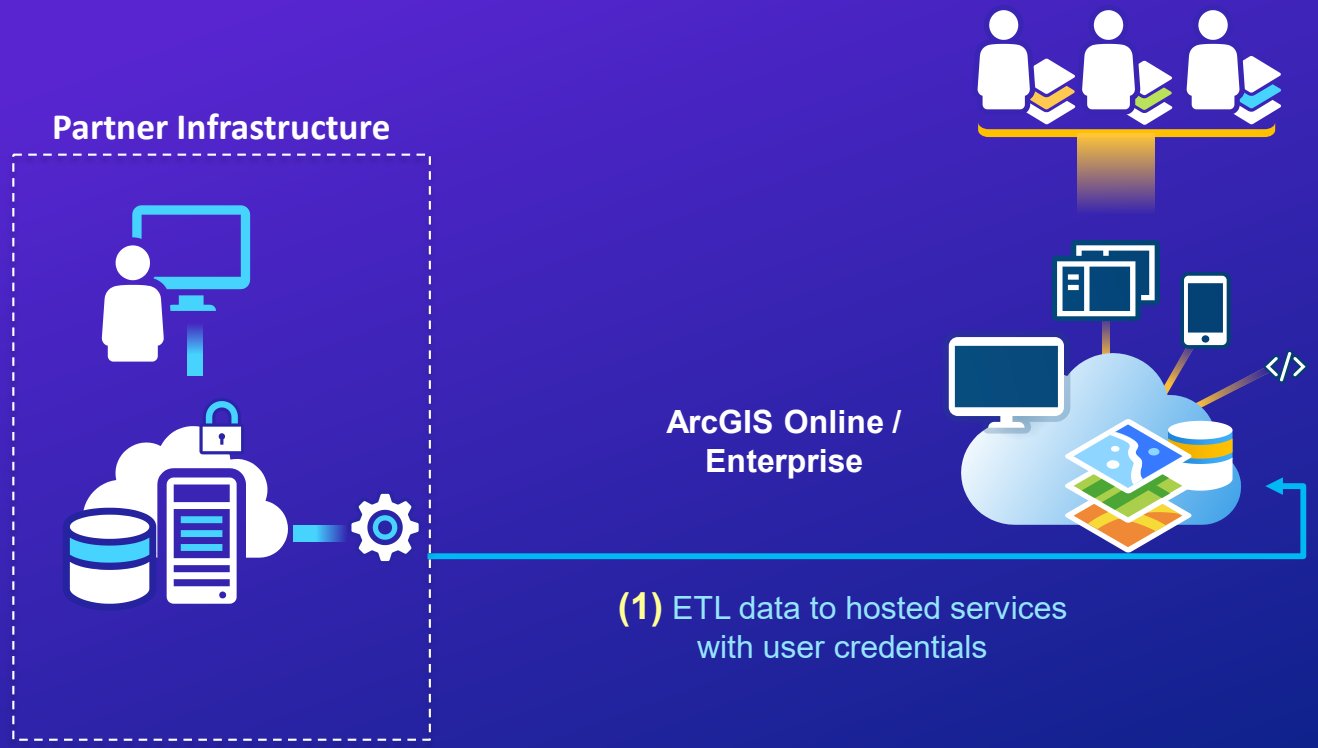


Publish Service Directly

User controls access & admin

User pays for service

Works with any service supported by user infrastructure



create_service.ipynb | share_service.ipynb

Code | Python 3

Share via group

```
[ ]: # create an empty item from before
t_item = Item(gis, 'b71c61e14425426984ec8a1a94bb77f4')

res = gis.content.clone_items([t_item], folder='@epcdemos', copy_data=False)
empty_lyr_item = res[0]
empty_lyr_item
```

```
[ ]: # create a group
group_schema = {
    "title": "EPC 2020 Demo Group",
    "tags": "test, group, poc, epc",
    "description": "A group to showcase sharing data",
    "access": 'private',
    "is_invitation_only": True,
    "users_update_items": False
}
group = gis.groups.create_from_dict(group_schema)

# add users via their ArcGIS Online user name
# users will receive a notification to accept the invite
usernames=['mpayson_ourcommunity']
group.invite_users(usernames)

# share the outputs to the group
gis.content.share_items([empty_lyr_item], groups=[group])
```

Sharing services

Share vs publish directly

Accessing Existing Services

Common Challenges

Finding data specific to your application

Discovering what data users have

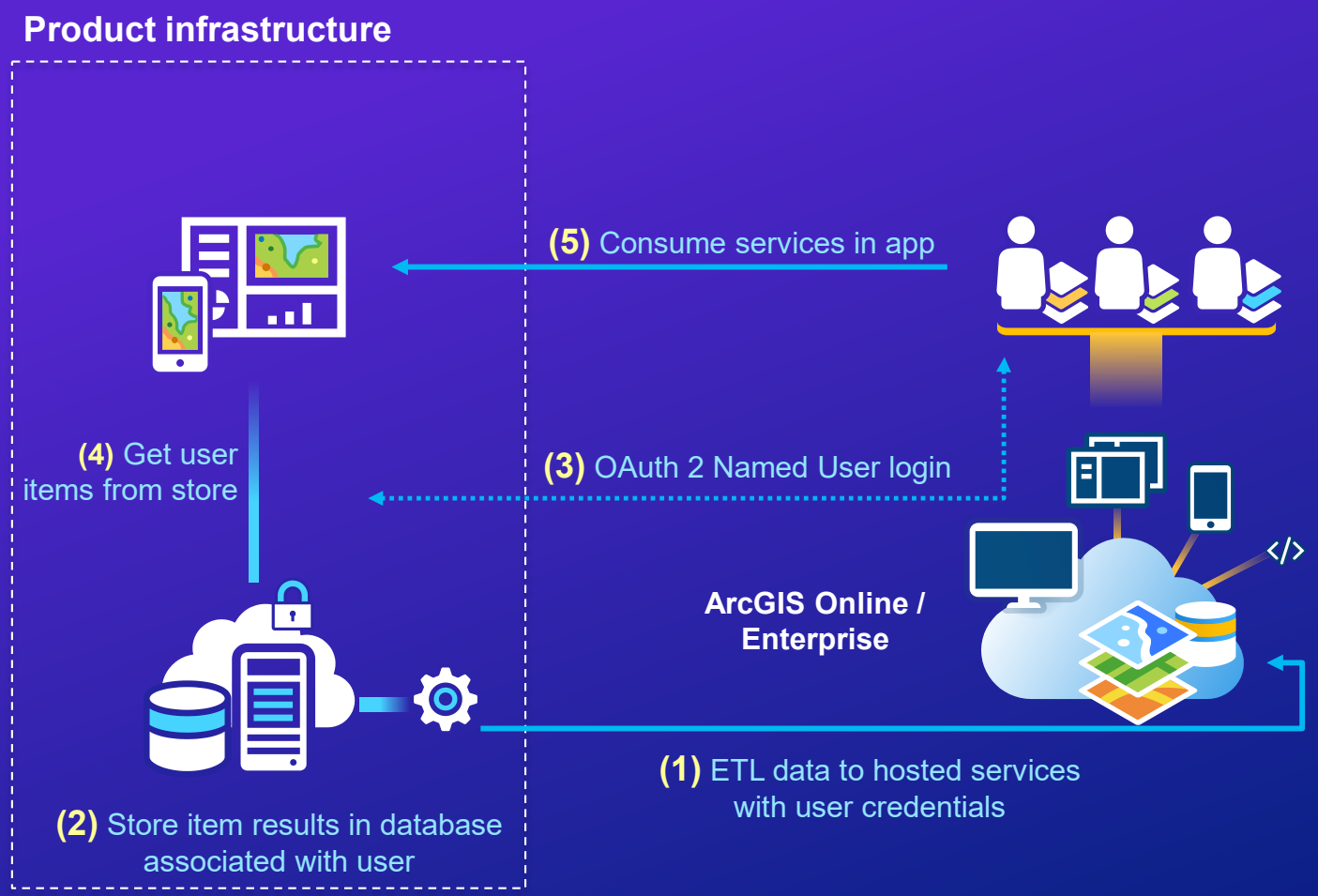
Standardizing data models

Two Options

- Store item info in user store
- Query for items user can access

Store Item Info

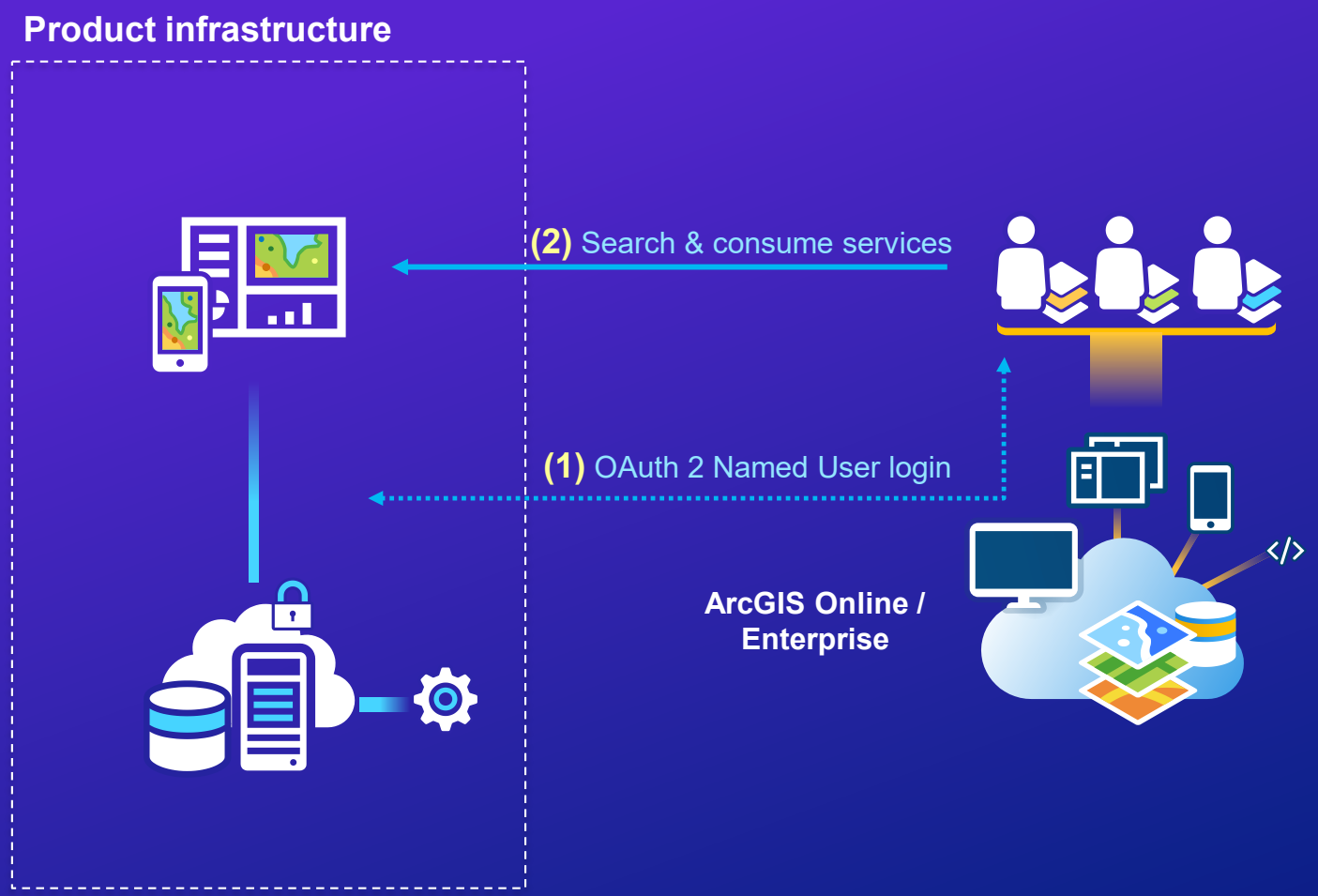
Requires a backend & integrated auth



Search for Items

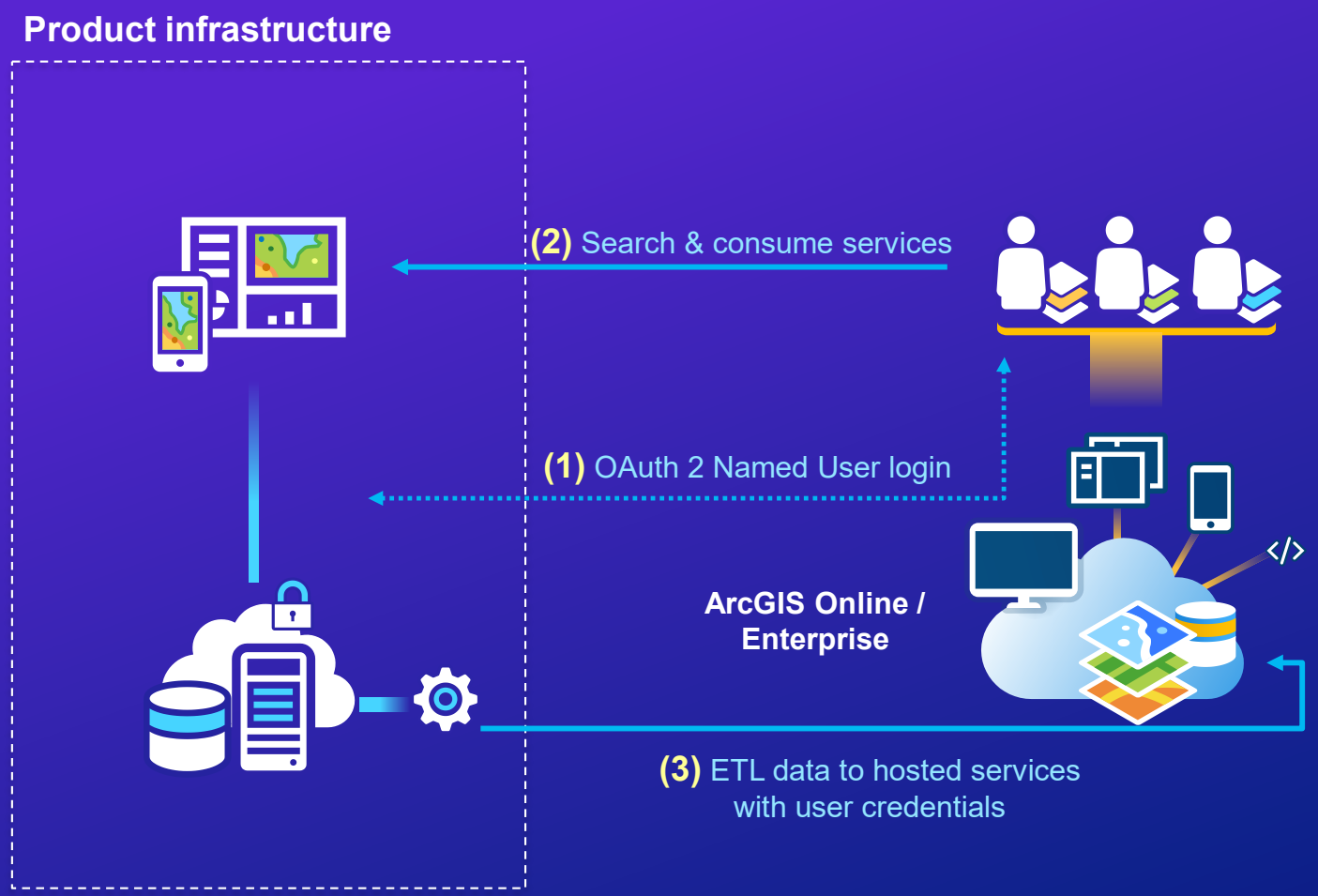
Query responses specific to user

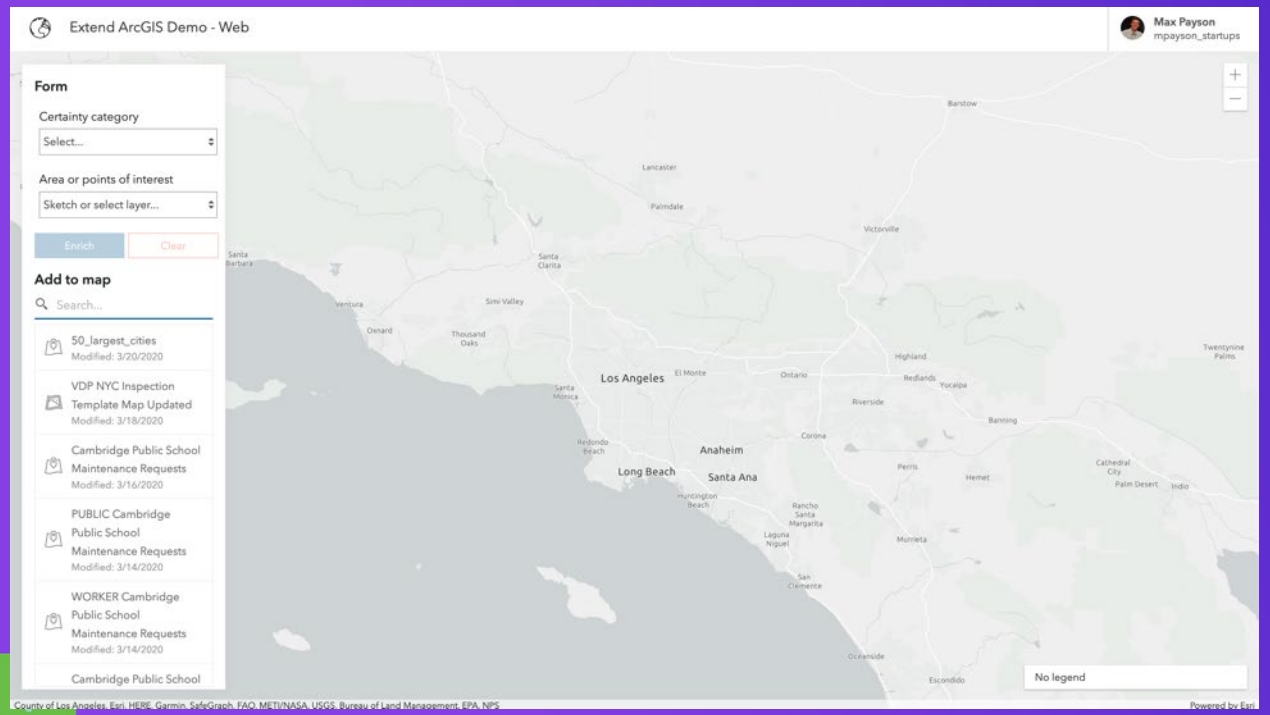
Can build UI to select items and map to schema



Search for Items

Use identifiable metadata (eg titles, tags)





Search API

In practice

Thank you!

Code? https://github.com/mpayson/presentations
Questions? mpayson@esri.com, tfair@esri.com