ArcGIS Pro SDK for .NET Understanding Feature Services, a Guide for Developers

Charlie Macleod

Russell Brennan

2019 ESRI DEVELOPER SUMMIT
Palm Springs, CA

Agenda

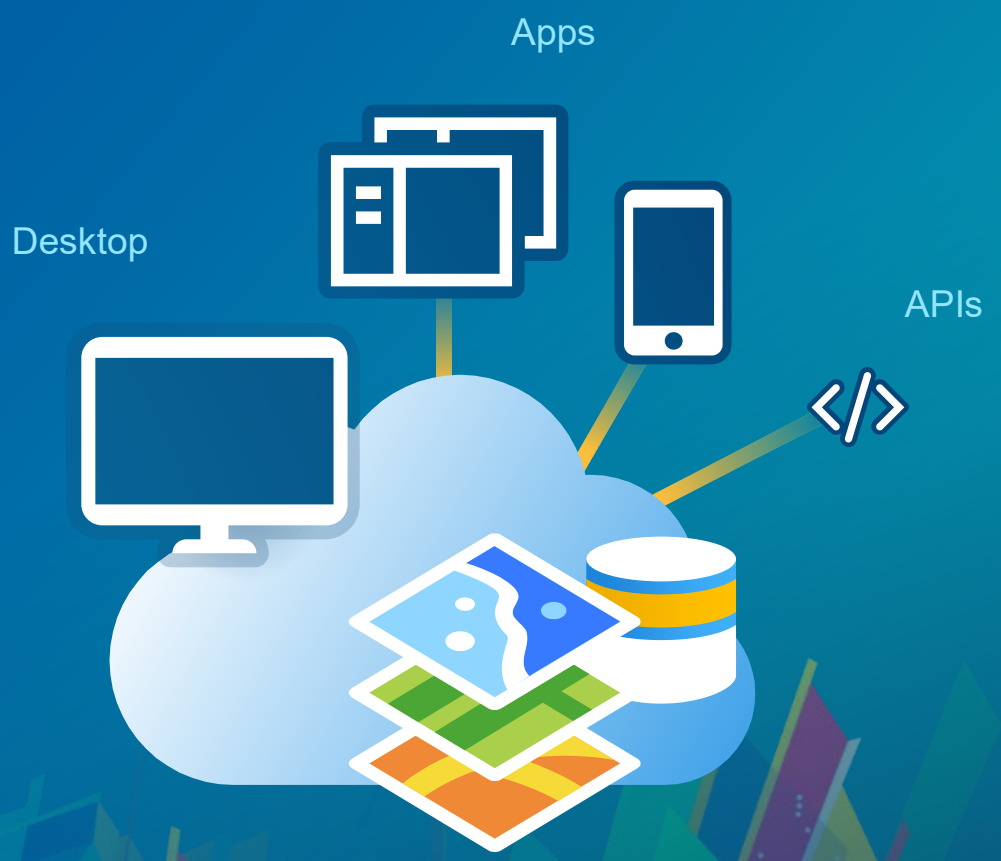
- Feature Service Overview
- Architecture and Data Storage
- Branch Versioning
- Editing Feature Services
- Tips and tricks

- Please silence your cell phones

Feature Service Overview

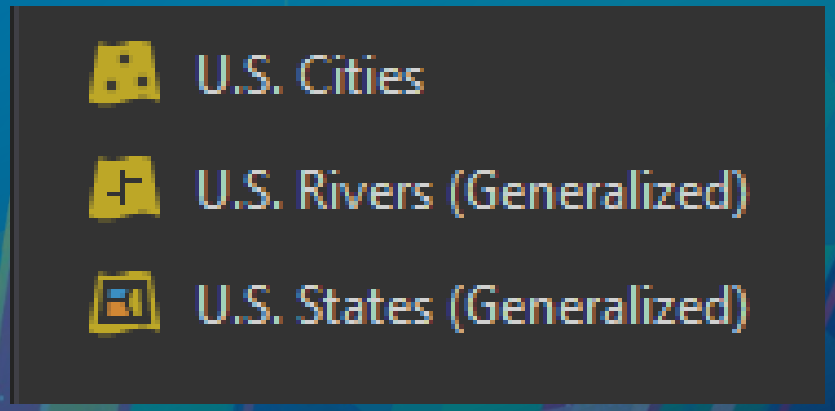
Feature services

- Platform feature access
- Access via
  - Web Browser
  - Desktop
  - Apps
- Branch versioning
- Distributed/disconnected editing via Sync
- Utility Networks
- Parcel Fabrics



Feature services

- Contain layers and tables
- Allows query and edit of feature geometry and attributes.
- Queries and edits are made via REST requests
- Naming:
  - web feature layers
  - sometimes just web layers
  - 1 web feature layer can have many sub layers



What types of data are supported?

- Data structures supported by all feature services
  - Points
  - Lines
  - Polygons
  - Relationship classes
  - Attachments…
- Some feature services support more complex data
  - Utility Network, Annotation, Dimensions (10.7)
  - Parcel Fabric (coming soon)
  - Future ?

What else is included?

- Layers from a feature service contain more than just the row data
  - Renderer, symbols, etc.
  - Editing templates
  - Field visibility
  - Definition queries
  - Other layer properties…

Feature Service REST API

- **Operations on Service:**

Supported Operations: Query QueryDomains Apply Edits Create Replica Synchronize Replica Unregister Replica

- **Operations on Layer:**

Supported Operations: Query Apply Edits Add Features Update Features Delete Features Validate SQL Generate Renderer Return Updates

- **Service JSON (service resource)**

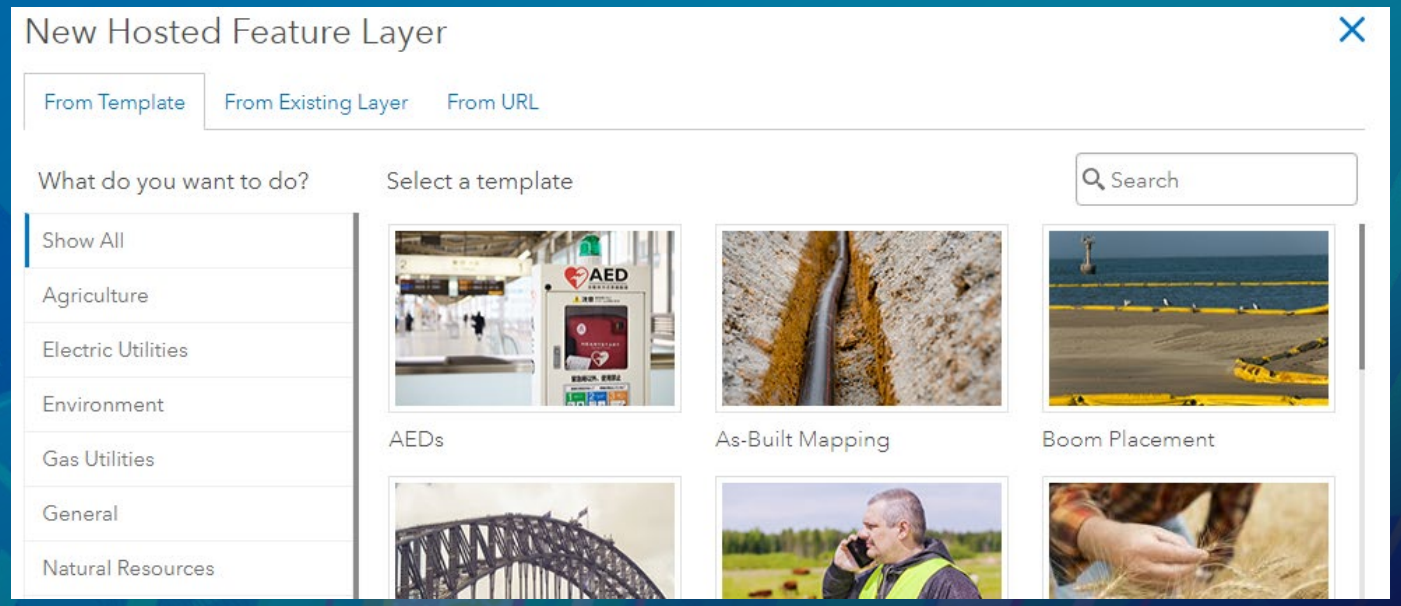
https://< server >/server/rest/services/TrueCurveTest/FeatureServer?f=pjson

- **Layer JSON (layer resource)**

https://< server >/server/rest/services/<service name>/FeatureServer/0?f=pjson

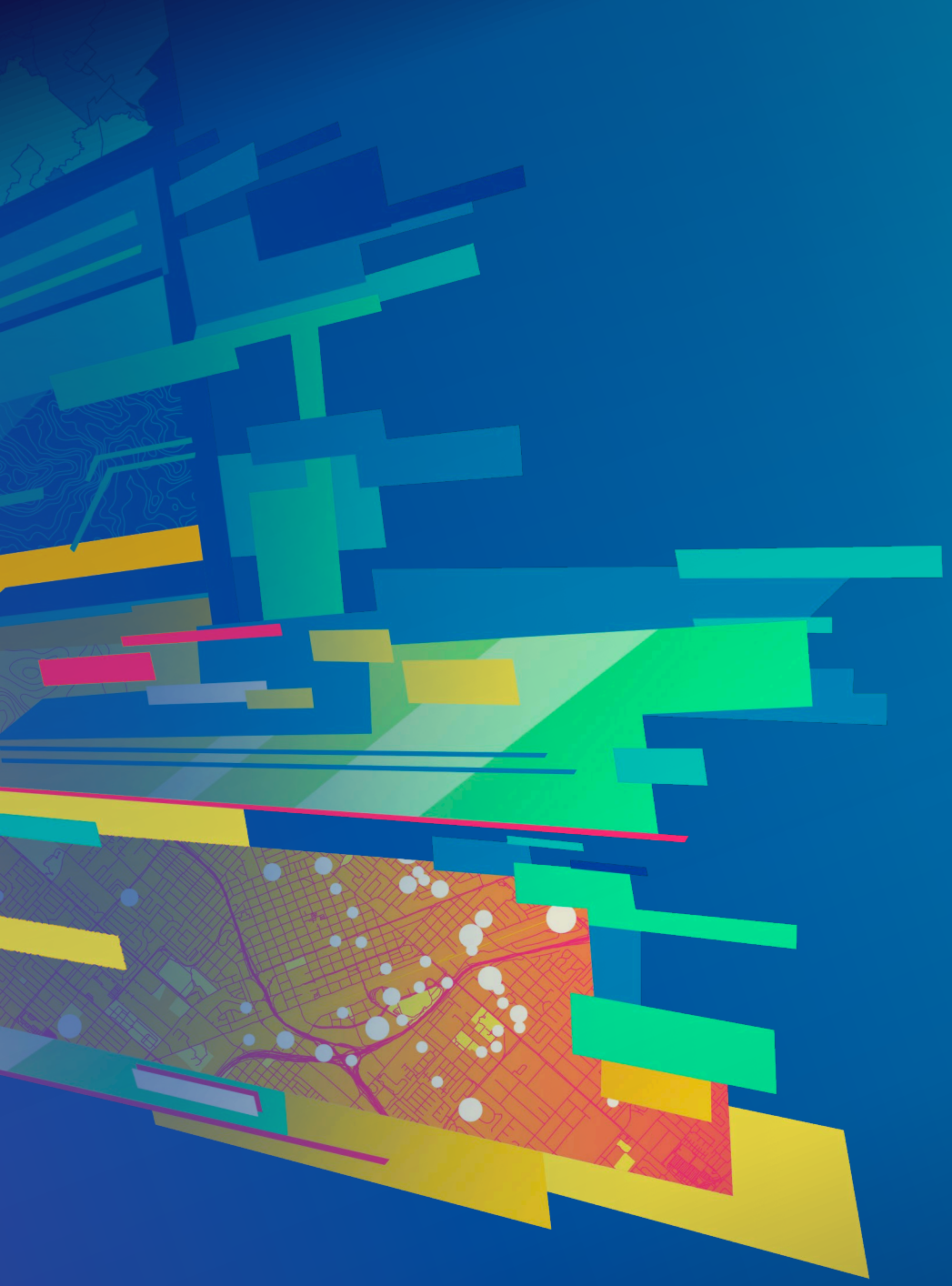
How do you create a feature service?

- Publishing from Pro or ArcMap
- REST
  - Upload SD files & making requests to publish
- Python API
- In Portals you can also:
  - Upload a file geodatabase
  - Upload an SD file
  - Create from another service
  - Create from a template



Using feature services in ArcGIS Pro

- Mostly seamless user experience
- Developers write code as though it were any other data source
  - Supports objects you would expect in a geodatabase data source
    - cursors, feature classes, dataset definitions, connection properties, etc.
- *In general*, code written for any data store will work for feature services



Demo

Basics - Creating connections

Creating Feature Service Layers in the Pro SDK

- ArcGIS.Desktop.Mapping:
  - `LayerFactory CreateLayer()` overloads*
    - Provide the service URI
  - `CreateLayer()` returns:
    - A GroupLayer if the end-point URI points to the service
      - Contains a feature layer added for each layer defined on the service
      - Adds any stand-alone tables for each table defined on the service
    - A FeatureLayer if the end-point URI includes a layer id

*`StandaloneTableFactory CreateTable()` for non-spatial tables

Creating Feature Service Layers in the Pro SDK

- **LayerFactory example:**

```
var hosted = @"https://<server1>/server/rest/services/Hosted/Foo/FeatureServer";
var by_ref = @"https://<server1>/server/rest/services/Bar/FeatureServer";
var portal = @"https://<server2>/portal/home/item.html?id=GUID HERE";

await QueuedTask.Run(() =>  {
  var groupLyr1 = LayerFactory.Instance.CreateLayer(
                          new Uri(hosted, UriKind.Absolute), MapView.Active.Map);

  var groupLyr2 = LayerFactory.Instance.CreateLayer(
                          new Uri(by_ref, UriKind.Absolute), MapView.Active.Map);

  var groupLyr3 = LayerFactory.Instance.CreateLayer(
                      new Uri(portal, UriKind.Absolute), MapView.Active.Map, 0);
```

Creating Feature Service Layers in the Pro SDK

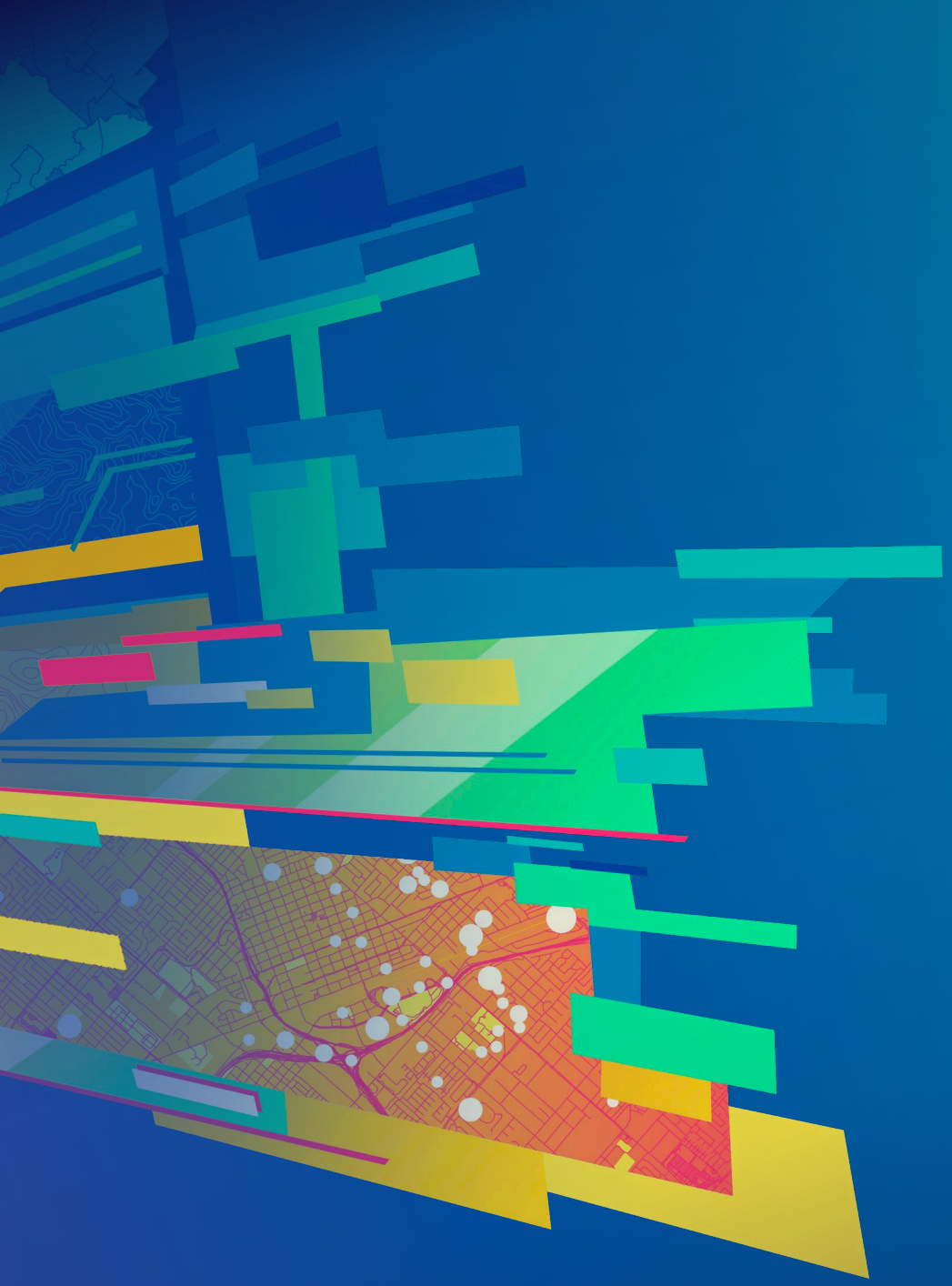
- **ArcGIS.Core.Data**
  - Connect to the `Geodatabase` via a `ServiceConnectionProperties` and the URI
    - Retrieve the relevant feature class(es) or table(es)
    - Use as the data source to LayerFactory or StandAloneTableFactory

```
//using ArcGIS.Core.Data

var url = @"https://<server1>/server/rest/services/Foo/FeatureServer";
await QueuedTask.Run(() =>  {

  var svc_props = new ServiceConnectionProperties(new Uri(url, UriKind.Absolute));
  var fs_db = new Geodatabase(svc_props);
  var dataset_name = ...;
  var fc = fs_db.OpenDataset<FeatureClass>(dataset_name);

  var featlayer = LayerFactory.Instance.CreateFeatureLayer(fc, MapView.Active.Map, 0);
```

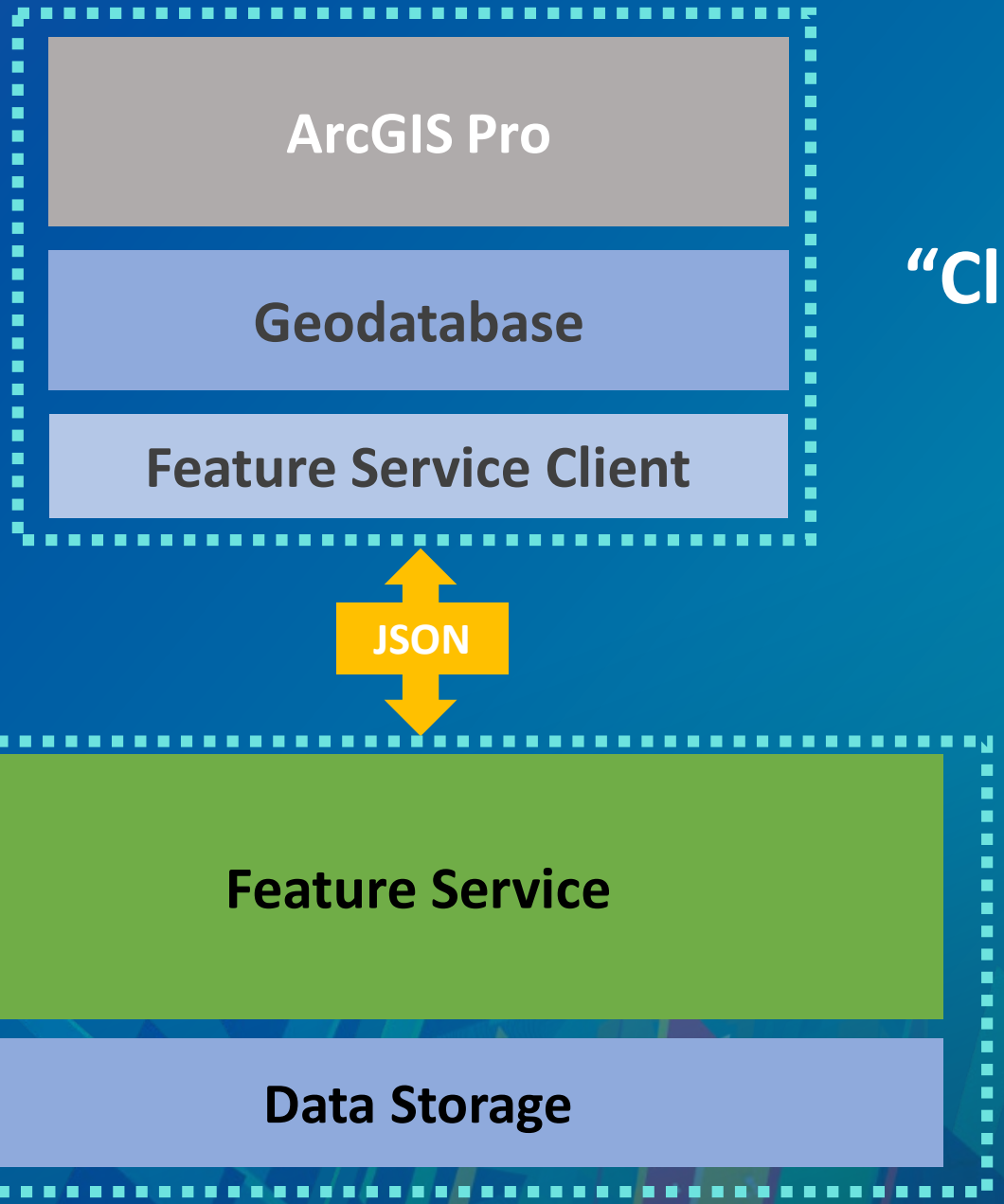


Demo

Basics - Creating connections

Architecture and Data Storage

Services
Architecture

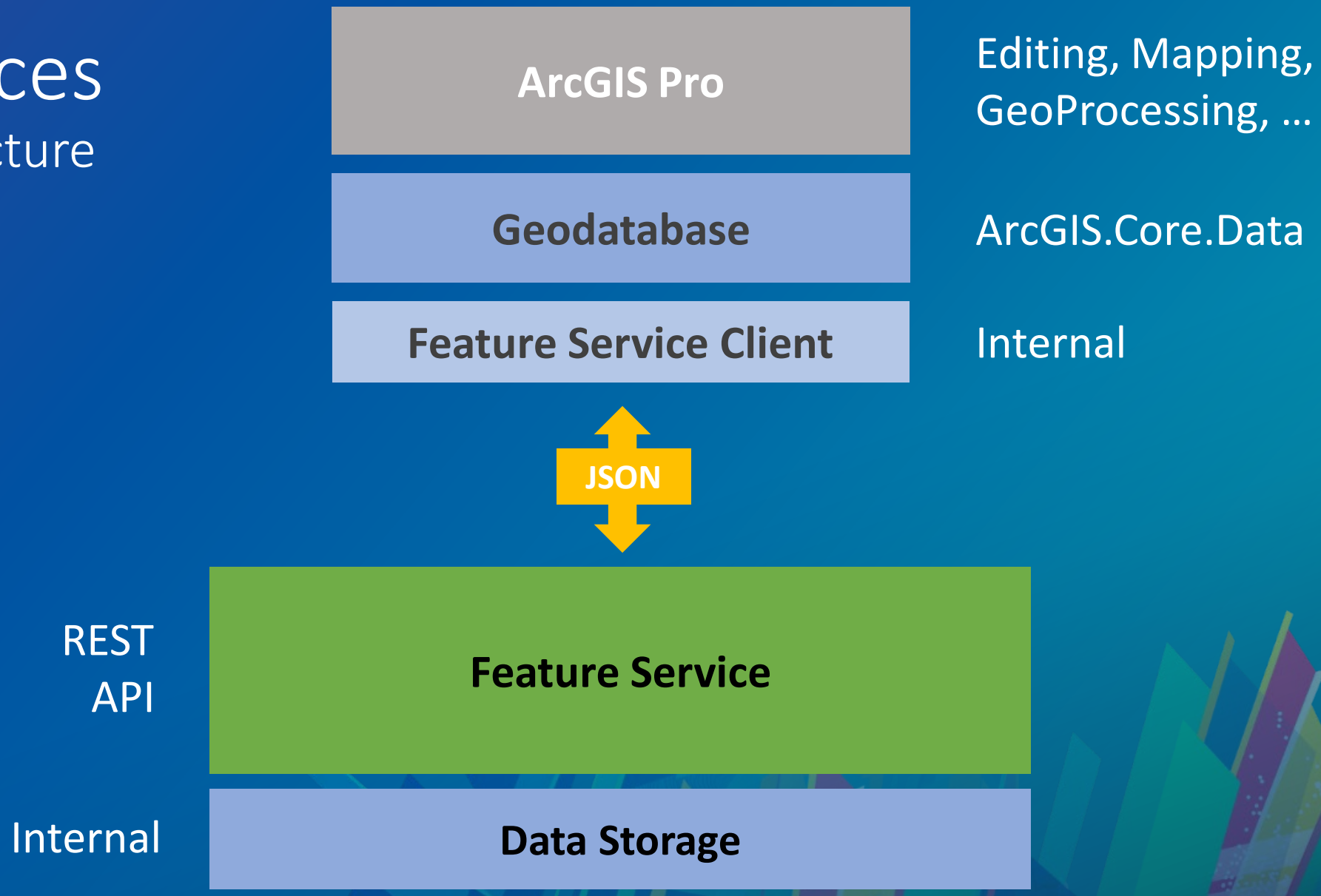


"Client"

"Server"

Enterprise, Online, ArcGIS Server

Services
Architecture

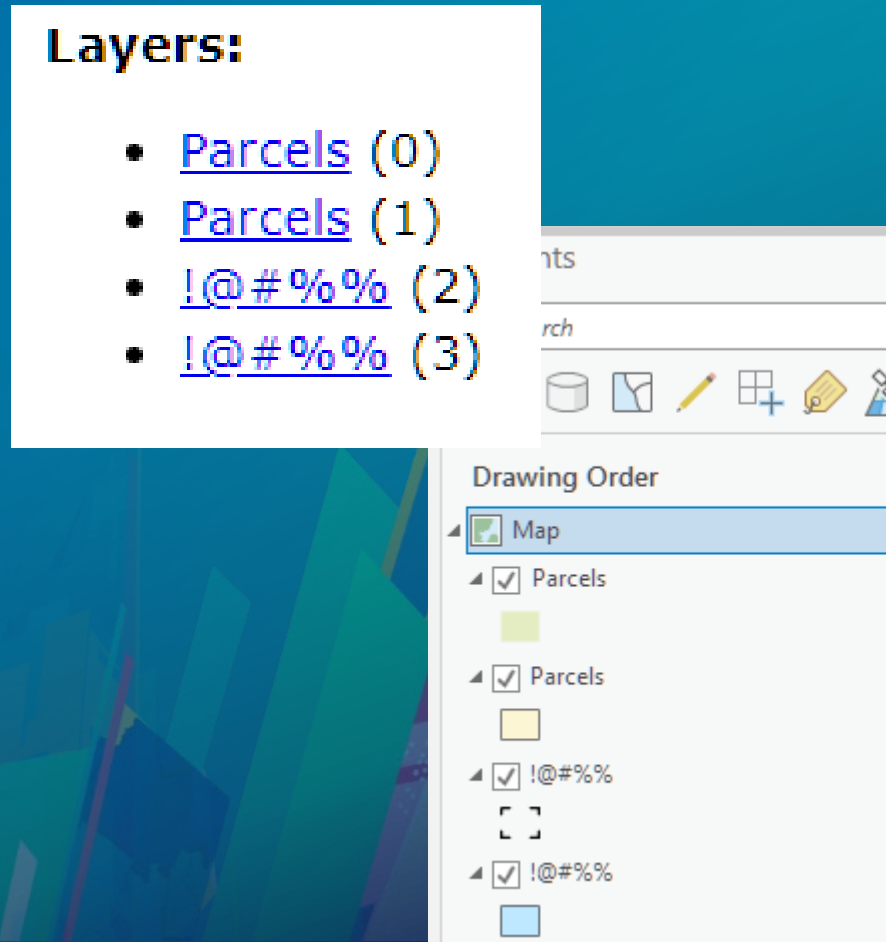


Common translations by the feature service client

Pro SDK API	REST API
Search Cursor	FeatureService Query
Insert or update cursor	FeatureService ApplyEdits
EditOperation.Create > Execute	FeatureService ApplyEdits
Open feature class	FeatureService layer resource
Open datastore	FeatureService server resource, relationships resource, queryDomains
… many more	

Technical details – feature class and layer naming

- Service has a layer which has an ID and a name
- Name is not unique for the service
- Internally we need the name to be:
  - Parse-able for SQL queries
  - Unique for the datastore
- Feature class name is L + layer ID + cleaned up name
  - Allows internal SQL parsing
  - Eg: L1Parcels, L217Highways, …
- Service layer name becomes the class alias name
  - Most of the time you should see layer name



Technical details – functionality unique to Pro

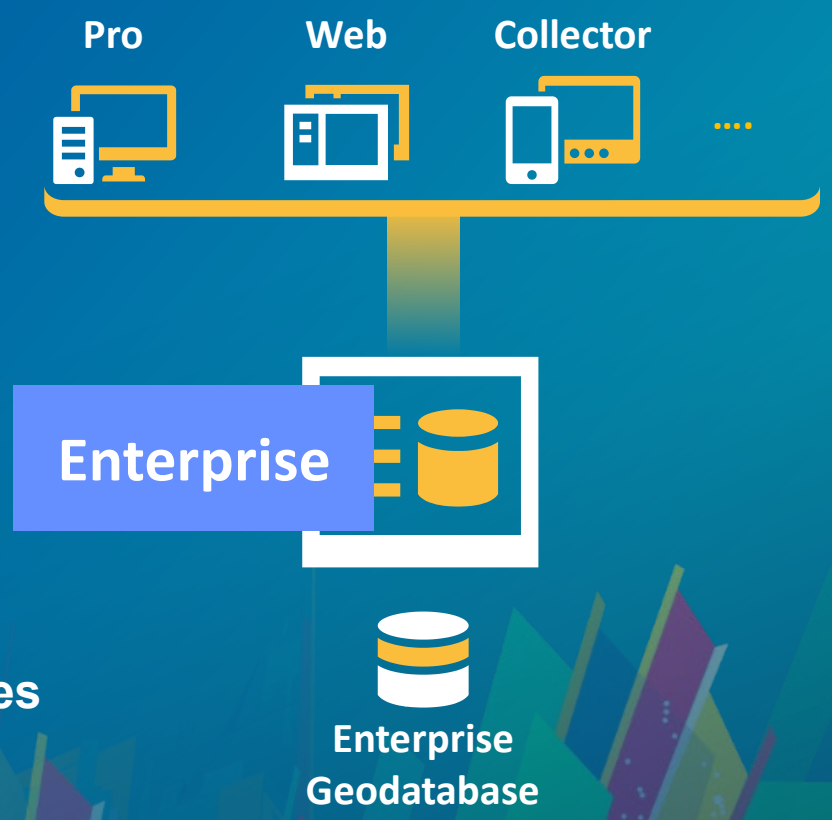
- Caching
  - Local copy of the data used for editing and querying
  - Improves query performance
  - Write through – updated as edits happen locally
- Pagination
  - Gets records a page at a time
  - All queries will return full result sets

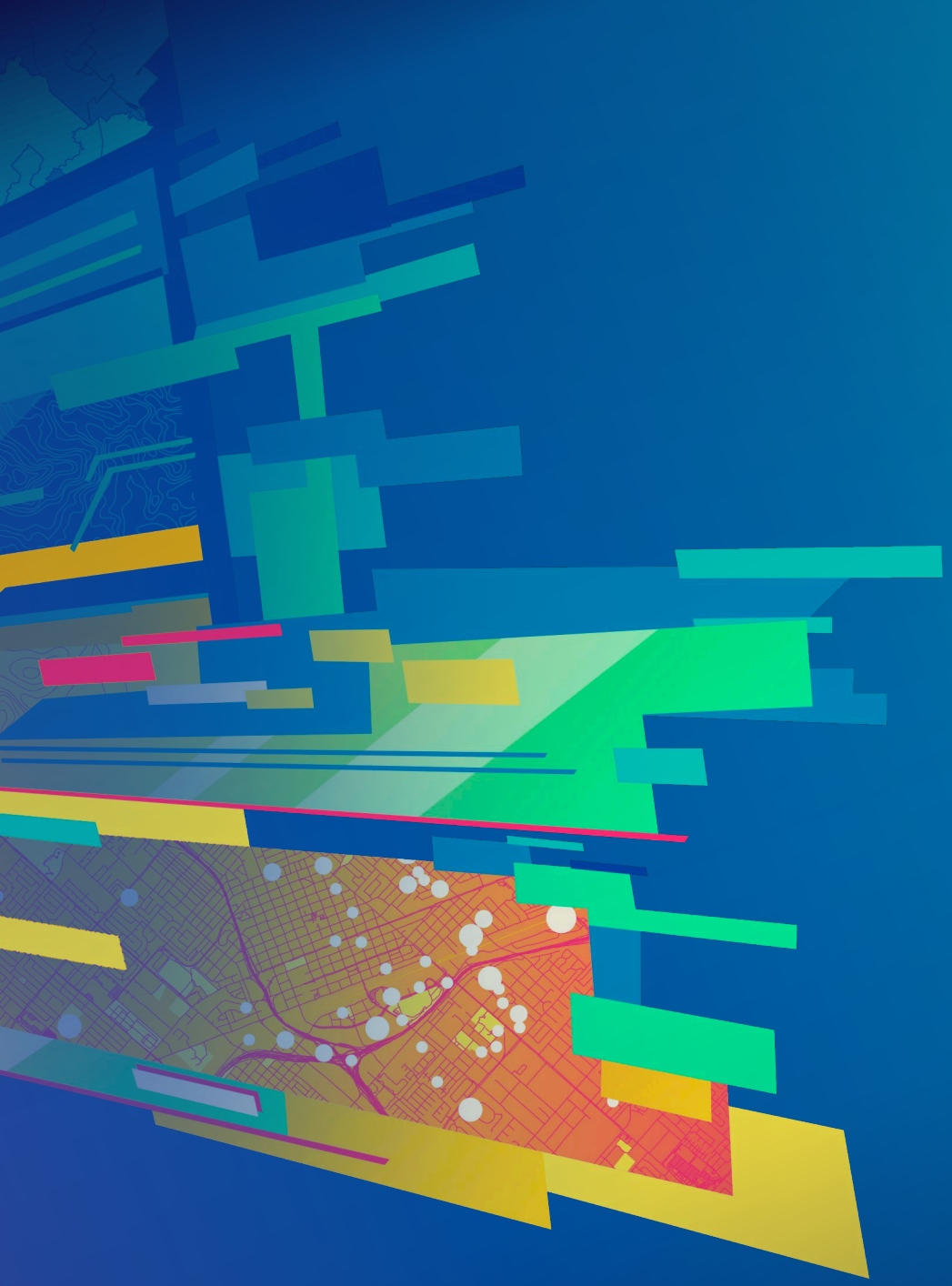
Data storage options - hosted

Data storage options – by reference

- Feature Services referencing an Enterprise Geodatabase
  - ArcGIS Enterprise (federated)
  - ArcGIS Server (unfederated / stand alone)

- Only implementation to support:
  - Versioning, utility networks, true curves, annotation, dimensions, attribute rules, contingent attribute values





Demo

General Info

Branch Versioning

Branch versioning

- New versioning model in Enterprise geodatabase
- Only for by reference services not available for hosted services
- Only editable via feature service
- Timestamp based
- Only versioning model for newer datasets
  - Parcel fabric
  - Utility network
- Same basic workflows as before (edit, reconcile, post)
- Same types of versions (private, protected, public)

More about versions and permissions

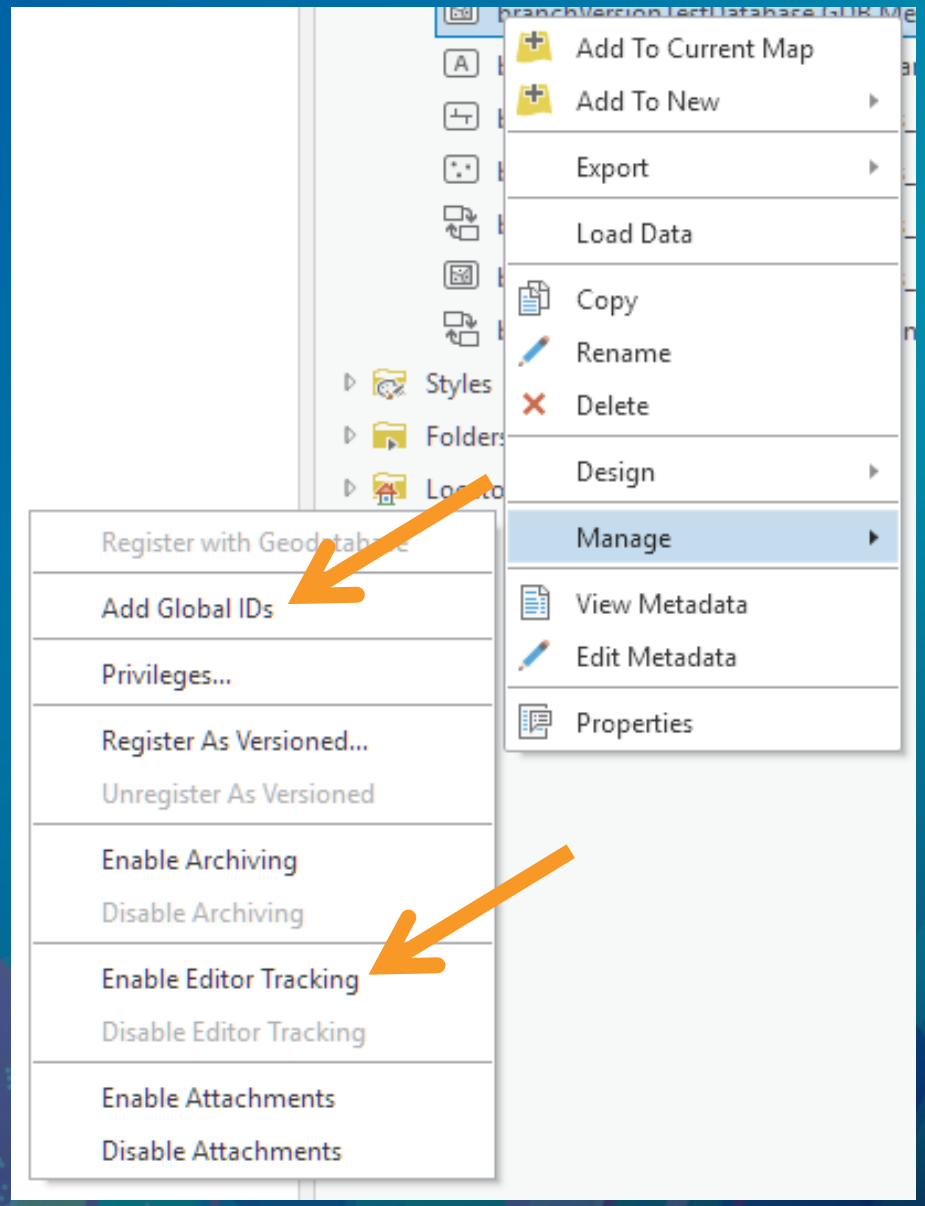
- Services can be secured or unsecured
  - Secured – shared within your Organization or within your ArcGIS Server
  - Unsecured – shared with everyone
- Feature service and version access based on the Portal user not DBMS user
  - Due to version access it is recommended to use secured services
  - Editor tracking comes from portal user
- Esri suggests using a protected version for the Default version
  - Other users can create versions from it
  - Allows QA by admin before data is posted

Preparing data for branch versioning

- Create tables, set up schema, load data
  - Add global id’s
  - Enable editor tracking
  - Relationships cannot be OID based
- Make a branch version connection in Pro
- Register as branch versioned
- Publish to Enterprise ‘by reference’ feature service
- Enable version management capability

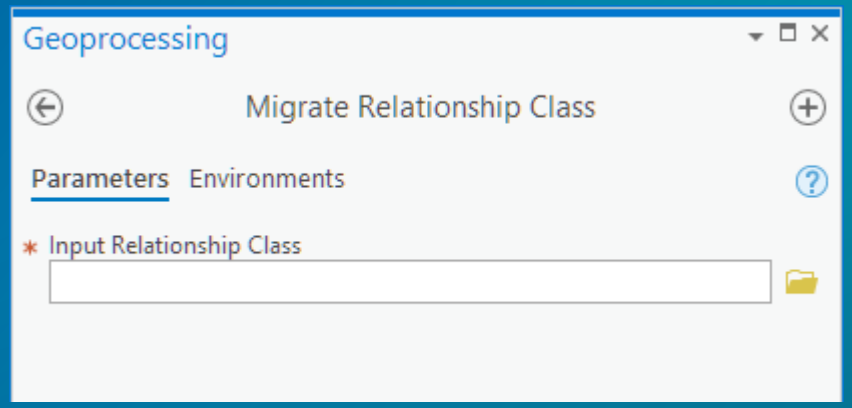
Preparing data for branch versioning

- Create tables, set up schema, load data
  - Add global id's
  - Enable editor tracking
  - Relationships cannot be OID based
- Make a branch version connection in Pro
- Register as branch versioned
- Publish to Enterprise 'by reference' feature service
- Enable version management capability



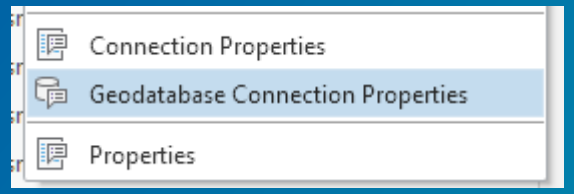
Preparing data for branch versioning

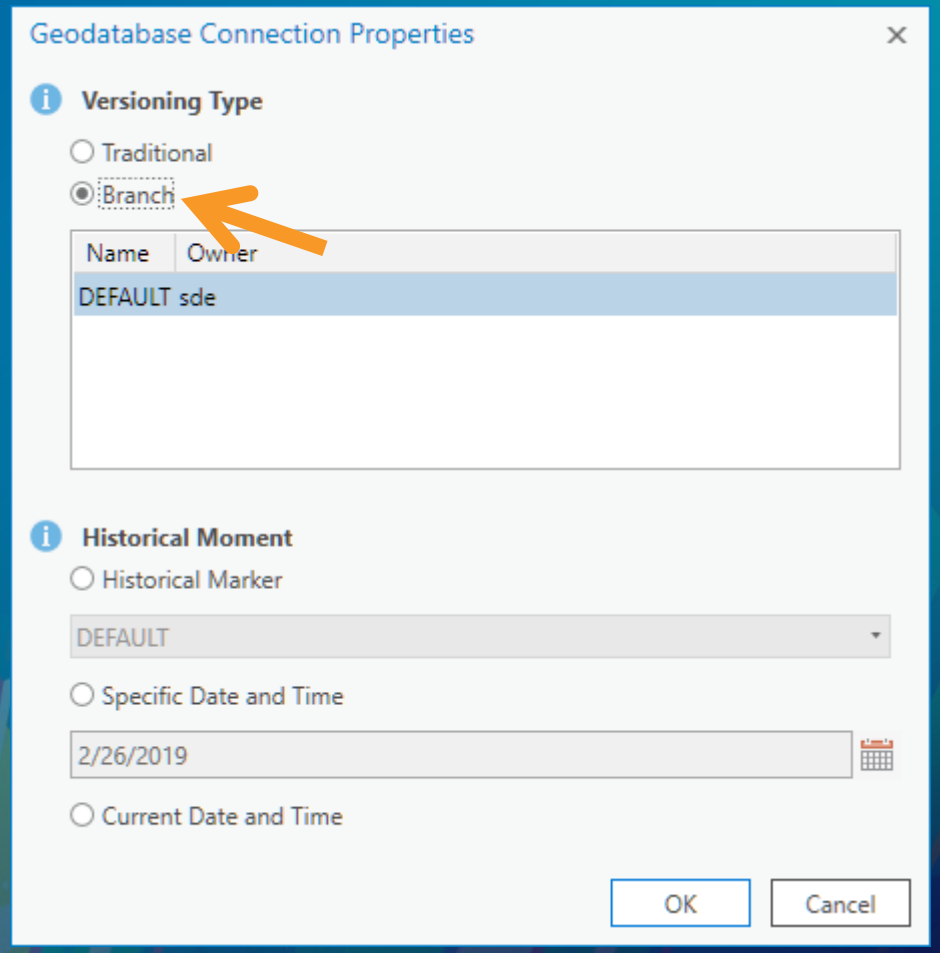
- Create tables, set up schema, load data
  - Add global id's
  - Enable editor tracking
  - Relationships cannot be OID based
- Make a branch version connection in Pro
- Register as branch versioned
- Publish to Enterprise 'by reference' feature service
- Enable version management capability



Preparing data for branch versioning

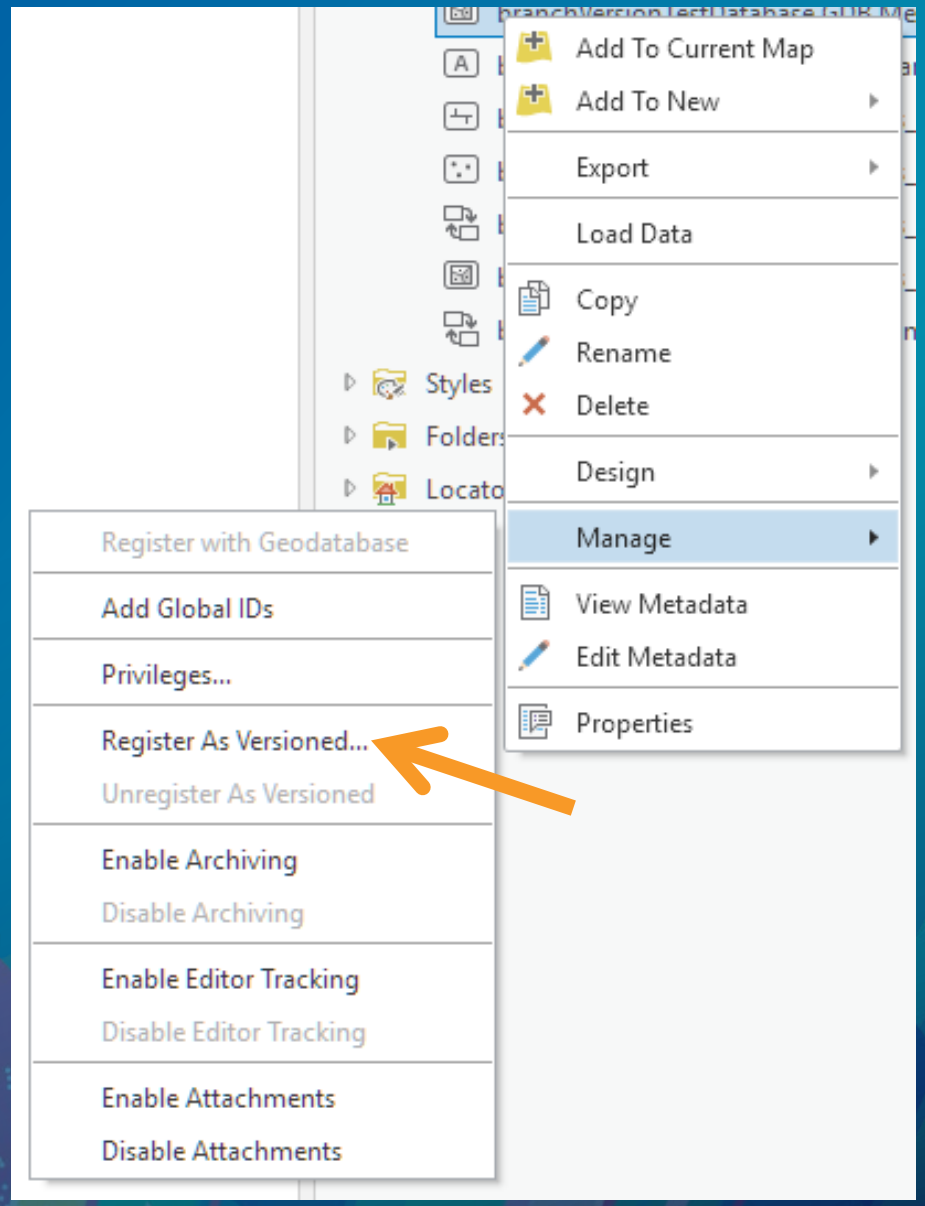
- Create tables, set up schema, load data
  - Add global id's
  - Enable editor tracking
  - Relationships cannot be OID based
- Make a branch version connection in Pro
- Register as branch versioned
- Publish to Enterprise 'by reference' feature service
- Enable version management capability





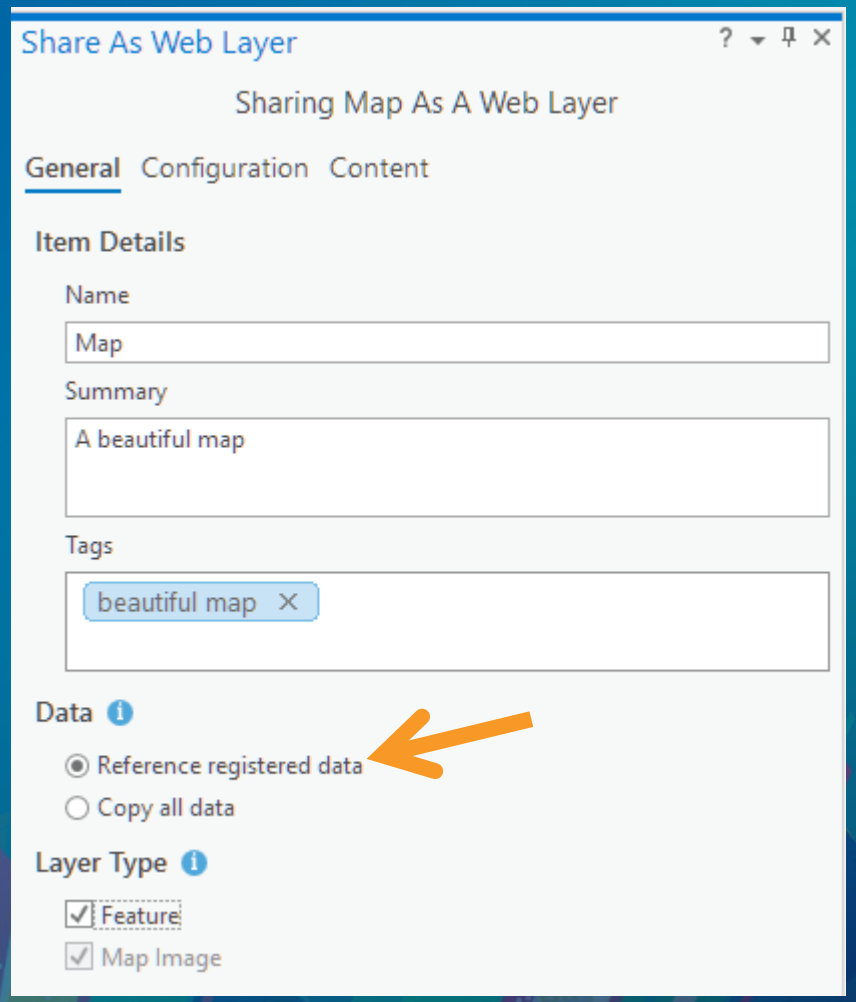
Preparing data for branch versioning

- Create tables, set up schema, load data
  - Add global id's
  - Enable editor tracking
  - Relationships cannot be OID based
- Make a branch version connection in Pro
- Register as branch versioned
- Publish to Enterprise 'by reference' feature service
- Enable version management capability



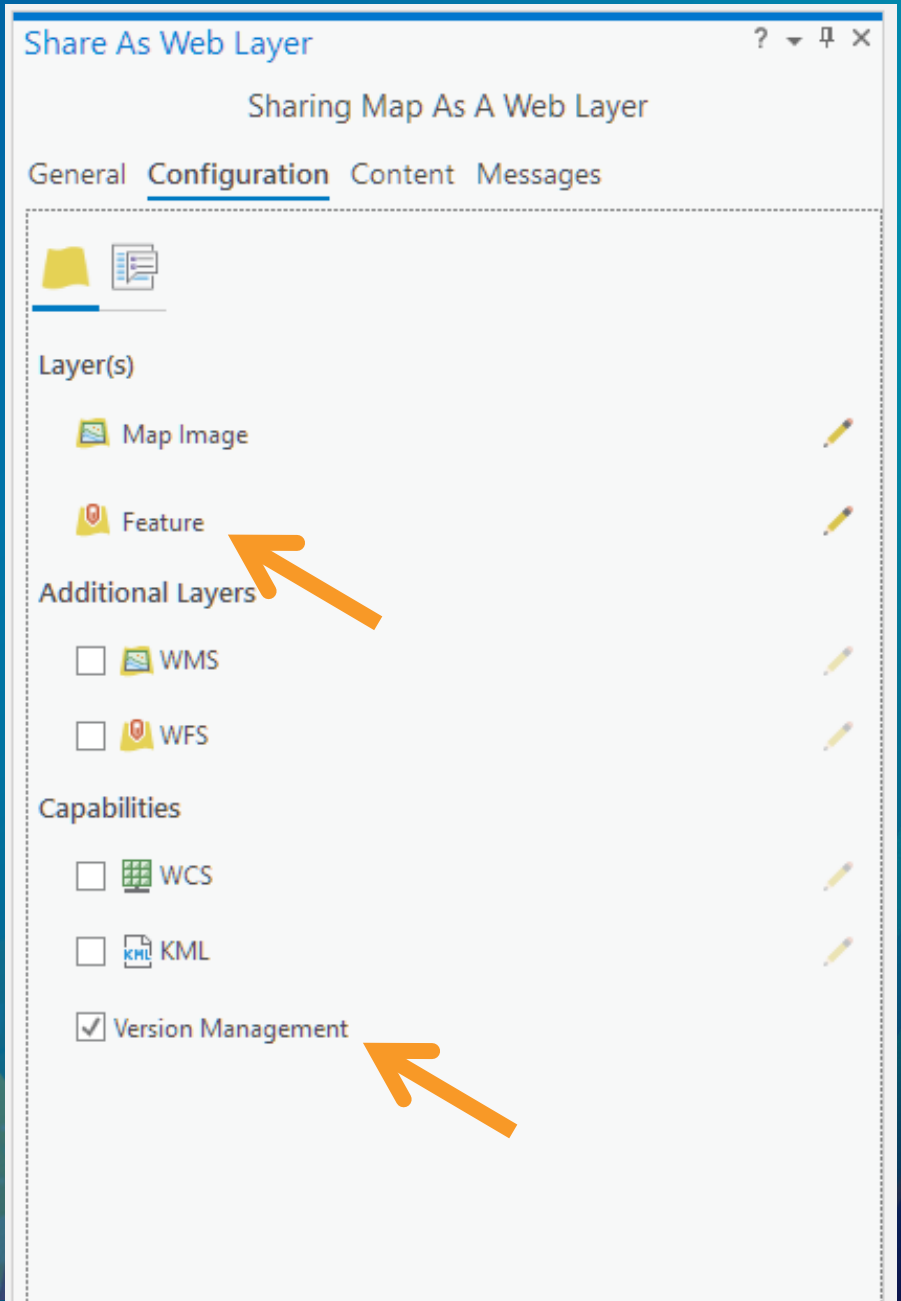
Preparing data for branch versioning

- Create tables, set up schema, load data
  - Add global id's
  - Enable editor tracking
  - Relationships cannot be OID based
- Make a branch version connection in Pro
- Register as branch versioned
- Publish to Enterprise 'by reference' feature service
- Enable version management capability



Preparing data for branch versioning

- Create tables, set up schema, load data
  - Add global id's
  - Enable editor tracking
  - Relationships cannot be OID based
- Make a branch version connection in Pro
- Register as branch versioned
- Publish to Enterprise 'by reference' feature service
- Enable version management capability (VMS)



Editing feature services

Feature service editing permissions

- Permissions
  - Query, Insert, Update, Delete
  - Service level, not on the individual layer
- Ownership based access control
  - Unique to feature services
  - Row level security based on editor tracking

Transaction model

- **Feature Service only**
  No VMS

- Applies to all hosted services
  - Short transactions, last in wins
- Applies to ‘by reference’ services if VMS is not present
  - Even when data is versioned
- No edit sessions

- **Feature Service with**
  Version Management (VMS)

- Long transaction editing behavior for Pro
  - Create versions
  - Edit in version
  - Reconcile
  - Review conflicts
  - Post
  - Delete version

VMS = version management server

Determining Editing Behavior

Check underlying `GeodatabaseType` and `RegistrationType` of the feature class:

```
((Geodatabase)featLayer.GetFeatureClass().GetDatastore()).GetGeodatabaseType();
 featLayer.GetFeatureClass()?.GetRegistrationType();
```

GeodatabaseType	RegistrationType	Version	EditOperationType	Example
Service	NonVersioned	N/A	SHORT	Hosted or By Reference
Service	Versioned	Default	LONG*	Branch Versioned
Service	Versioned	Named	LONG	Branch Versioned

*no undo/redo

Edit Behavior

- Summary of characteristics by Long and Short type

GeodatabaseType	RegistrationType	CancelEdit	Group Edits	Undo/Redo	Save/Discard
Service	NonVersioned	YES*	NO	NO	NO
Service	Versioned (Default)	YES	YES	NO	NO
Service	Versioned (Named)	YES	YES	YES	YES

LONG SHORT

*2.2 and earlier - Create cannot be canceled

Locking model – named version

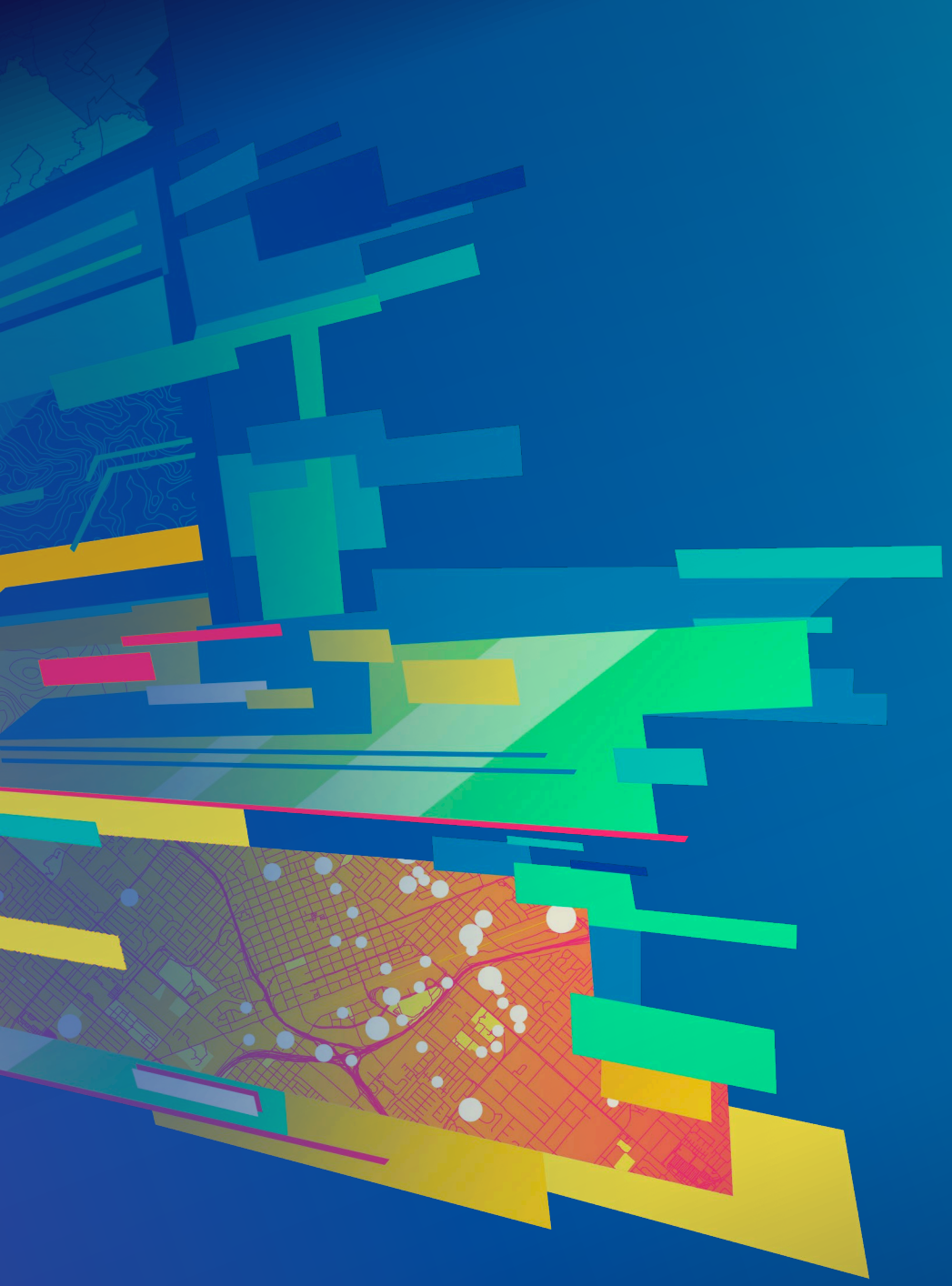
- Branch versioned editing is single editor, multiple viewer model
  - One edit session at a time per version
- Connecting to version takes a shared lock
- Editing in a version takes an exclusive lock
  - Starts edit session
  - Cannot take exclusive when a second shared lock exists.
- Save or discard releases exclusive lock
  - Stops edit session
- Releasing the data store releases shared lock.

Locking model

- Do not connect to the same version as the same user from two different Pro sessions
- Why not?
  - The second session will abort 1st sessions pending edits
- Default version
  - Multiple editor, no locking behavior.

Mixed mode editing

- "Mixed Mode" is the default in Pro
  - Data from multiple datasources can be edited at the same time
    - Contrast with Arcmap – only one workspace can be edited at any one time.
  - TOC may include data with "Short" and "Long" transaction semantics
    - Long transactions go on the undo/redo stack
    - Short transactions are committed immediately
  - Can lead to inconsistent UI experience for Undo/Redo, Save/Discard when both are edited together



Demo

Editing Feature Services

Special considerations

aka Tips and Tricks

Performance

- GetCount -> uses feature service count API
- If possible avoid insert/update for 1000’s of records
  - Bulk editing is generally slow
  - Go to back end data for by ref
  - Go to Append API if possible for hosted
  - RUSSELL to look into flushing in an edit session
- Calculating values -> use calculate field with SQL -> uses REST API

Other considerations

- DDL via GP for hosted services
  - Add field, delete field, add index, delete index are supported for hosted data
  - More to come
- Be careful of definition queries applied to layers when publishing
  - Rows can disappear
- Insert only feature services
  - Rows can disappear
- Read only feature access to map service sub layers is also available

Feature caching

- Turned on by default
- User can disable for NonVersioned (no VMS) services
- Multiple editor scenarios
  - Non versioned: refresh map to see other’s edits
  - Branch default: refresh version to see other’s edits
  - Named version: single editor nobody else can edit
- Available for both map service and feature service

Offline feature services

- Ability added at 2.2
- When offline the layer data store becomes mobile geodatabase
  - Accessible from feature layer not from data store
- No undo/redo (coming soon)

ArcGIS Pro SDK for .NET – Technical Sessions

Date	Time	Session	Location
Thu, Mar 07	9:00 am – 10:00 am	Understanding Feature Services, a Guide for Developers	Primrose C-D
	10:30 am - 11:30 am	An Overview of the Utility Network Management API	San Jacinto
	1:00 pm – 2:00 pm	An Overview of the Geodatabase API	Mesquite G-H
	5:30 pm - 6:30 pm	Advanced Pro Customization with focus on Categories and Custom Settings	Smoketree A-E
Fri, Mar 08	8:30 am – 9:30 am	Advanced Editing with Focus on Edit Operations, Transaction Types, and Events	Mesquite C
	10:00 am - 11:00 am	Demonstrating Pro Extensibility with Add-Ins	Mesquite B

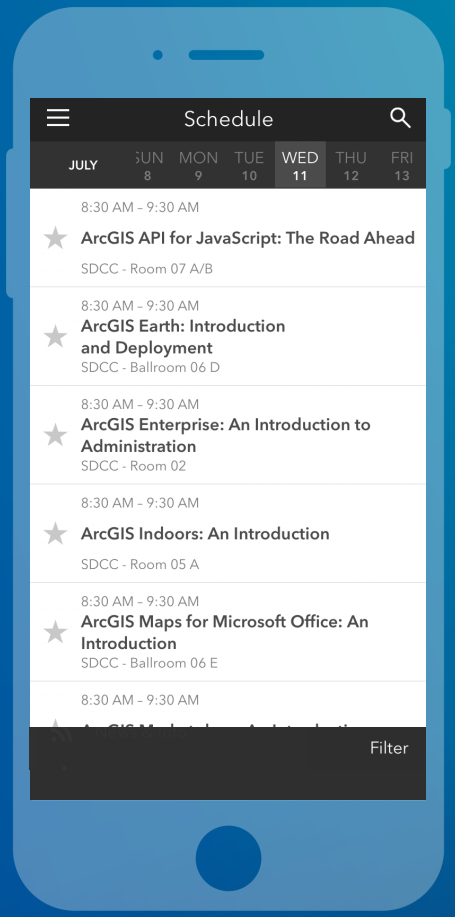
ArcGIS Pro – Road Ahead Session

Date	Time	Session	Location
Thu, Mar 07	4:00 pm – 5:00 pm	ArcGIS Pro: The Road Ahead	Primrose B

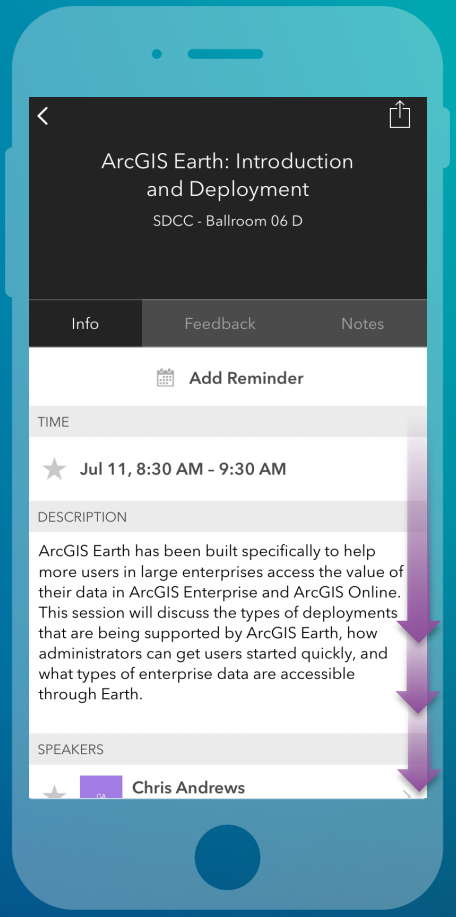
Please Take Our Survey on the App

Download the Esri Events app and find your event

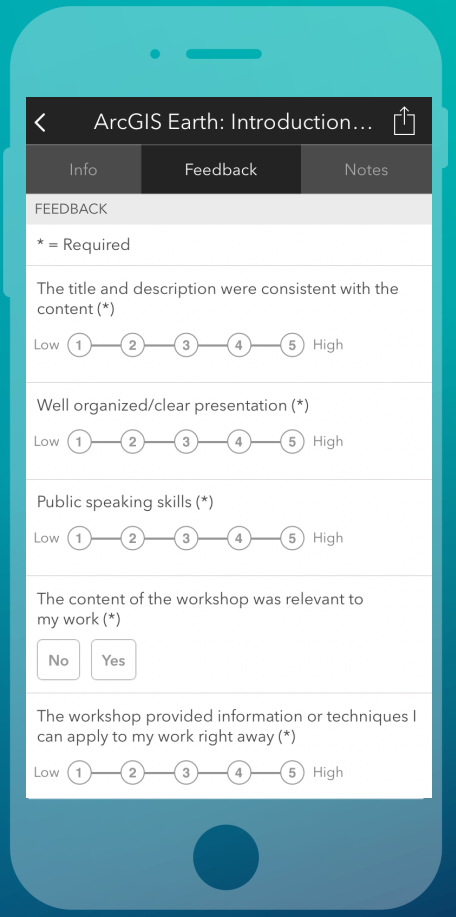
Select the session you attended



Scroll down to find the feedback section



Complete answers and select “Submit”



esri | THE SCIENCE OF WHERE