

# Automating Research at Scale



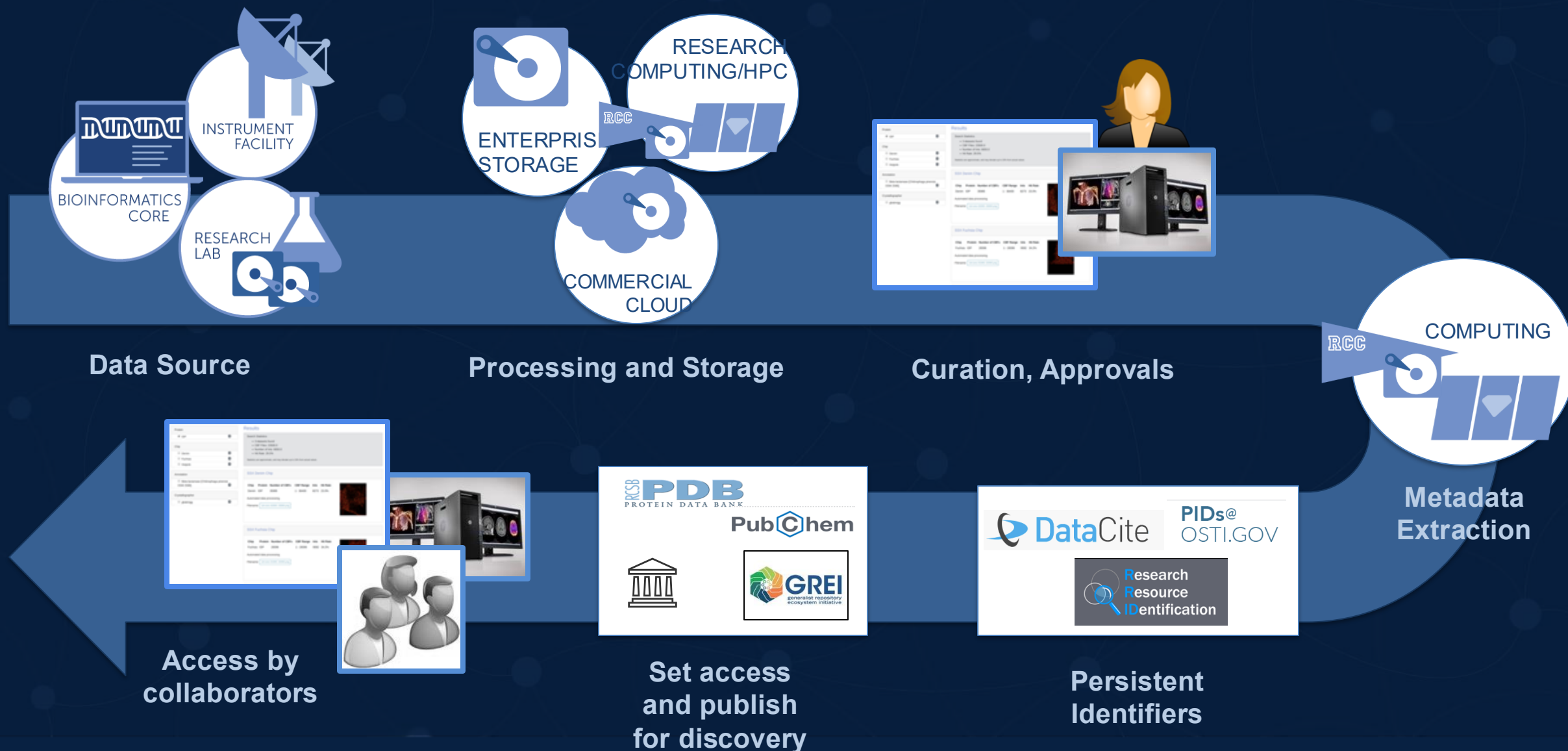
Rachana Ananthakrishnan, Lev Gorenstein,  
Greg Nawrocki, Vas Vasiliadis

[bit.ly/pearc25-globus](https://bit.ly/pearc25-globus)



THE UNIVERSITY OF  
CHICAGO

# Data processing pipeline pattern





# Requirements for such pipelines

- **Reliable, near-real time data access**
- **Uniform policy for data access, based on local policy**
- **Delegation of data access management to PI**
- **Ability to compute on data across storage classes**
- **Applying best practices with data processing pipeline**
- **Support for data organization to facilitate FAIR data**
- **...**

**Requires automation to scale**



# An example of what this looks like in practice...





# What's going on behind the curtain?

**Globus  
Flows**



**Move**



Transfer  
raw files

**Compute**



Launch  
analysis job

**Move**



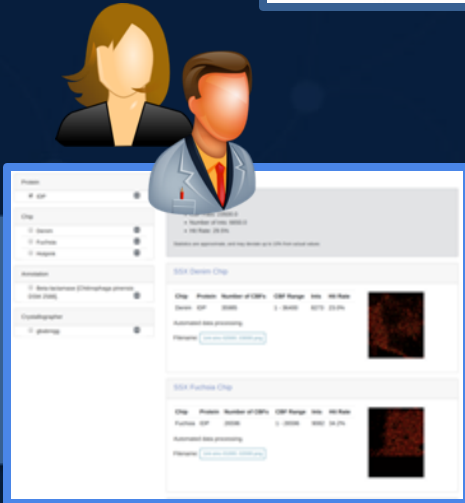
Transfer  
results

**Share**



Set access  
controls

**FAIR data,  
by default**



**Clean up**



Delete  
temp data

**Ingest**



Restricted  
metadata

**Ingest**



Open  
metadata

**Compute**



Extract  
metadata



# Our instrument research environment

## Sharing Repository



## Storage Endpoint (share)

set permissions

Transfer Service

globus

ingest metadata



Search Service

discover and access data



Compute Resource

Compute Endpoint

Storage Endpoint (scratch)

Registered compute function

```
def process_img  
    import os  
    import glob  
    from PIL import Image  
    files = (f for f in os.listdir('data') if f.endswith('.jpg'))  
    if not os.path.exists('results'):  
        os.makedirs('results')  
    for file in files:  
        image = Image.open(file)  
        # Generate thumbnail  
        image.thumbnail((128, 128))  
        # Save thumbnail  
        image.save('results/' + file)
```



Compute Service

invoke image processing function

Instrument Capture Machine (your laptop)

Monitor script

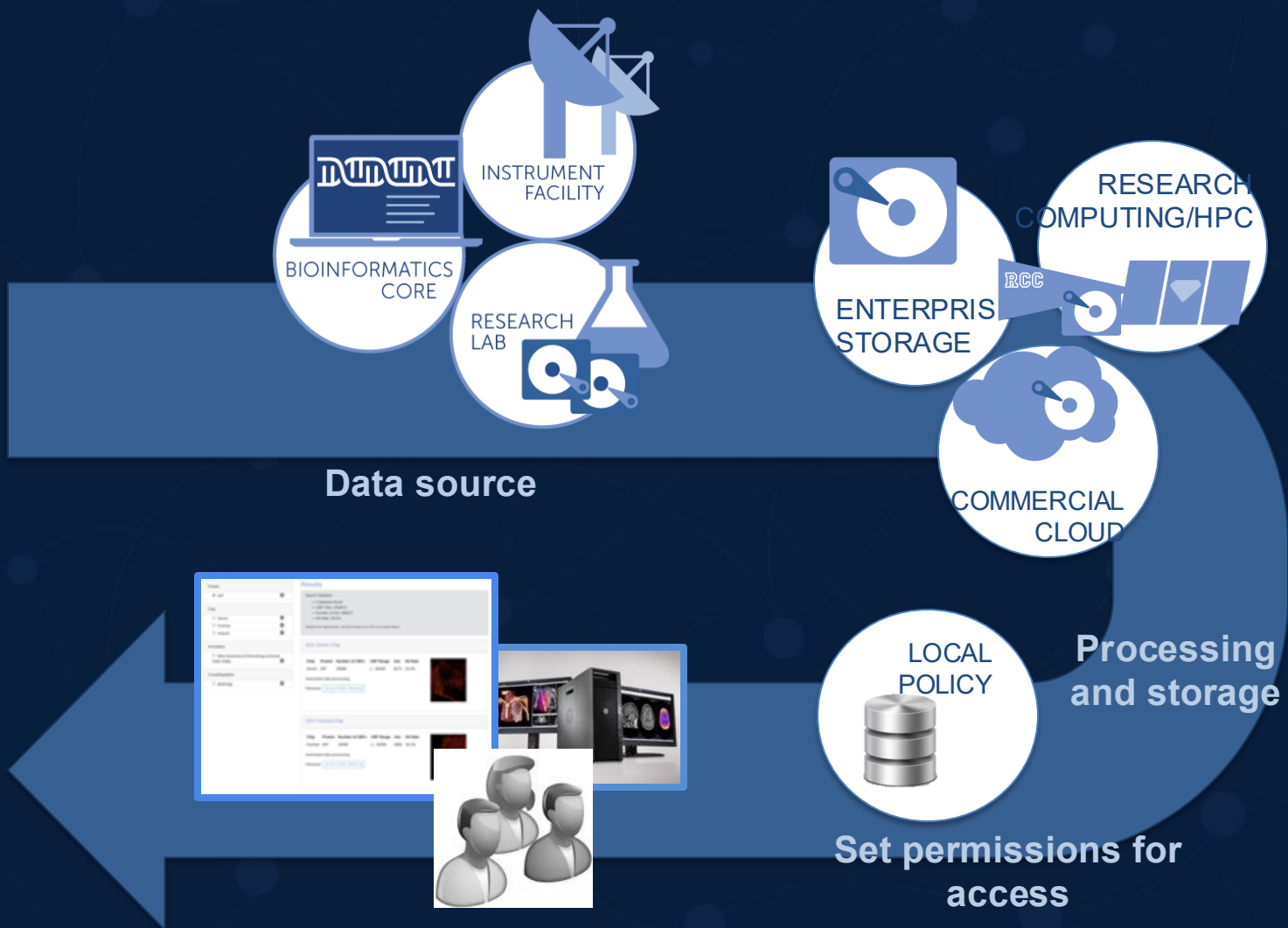
trigger flow run

Storage Endpoint (using GCP)

transfer control



# Data processing pipeline pattern



1. Credentials for automation
2. Preparing instrument for automation
3. Configuring data collections for automated data transfer and sharing
4. Configuring the computation environment
5. Creating and deploying flows for automation
6. Creating trigger scripts

Access by

collaborators



# Creating credentials for automation



# Managing service accounts/app credentials

- Service Account = Application Identity/Credential:  
**app\_client\_uuid@clients.auth.globus.org**
- These are confidential apps with client ID and secret
- Ensure application is on a secure device
- Set up policy for rotation of secret
- Assign project admins to manage the registration



# Registering a service account

- **Web app: Settings → Developers**  
**[app.globus.org/settings/developers](https://app.globus.org/settings/developers)**

Register an...

App



Register a service account or application credential  
for automation



Applications that authenticate and act as the application itself. These applications are used for automation and as service or community accounts, and do NOT act on behalf of other users.



# Get app credentials at [app.globus.org/settings/developers](https://app.globus.org/settings/developers)

Application Registration

App Registration

Project Name: Client Type Tests

App Name\*:

A human-friendly name to help you keep track of this app.

Native App: ☐ Will be used by a native application

Redirects:

One URL per line -- must be HTTPS

Required Identity: ☐ Require that the user has linked an identity from a specific Identity Provider, but does not require that the user authenticate with that identity each time.

Pre-select Identity Provider: ☐ Pre-select a specific Identity Provider on login page

Use effective identity (ID token + userinfo): ☒ Identity returned in userinfo is the primary identity unless a required identity is selected above. If unchecked, the identity returned in userinfo is the most recently authenticated identity. (This should usually be checked for most applications.)

Privacy Policy URL:

Terms & Conditions URL:

This secret will only be shown once, please copy and store it securely. If lost, Globus will not be able to recover it, and you will need to register a new secret.

Secret:



# Preparing your instrument for data automation



# Install Globus Connect

- **Acquisition machines mostly run Windows**
- **→ Globus Connect Personal...**
  - ...installed as a local user account (assuming PI login)
  - ...running as a service account (best practice, policy driven)
  - ...outbound connections only → easier to get approval!
- **Endpoint must have access to instrument data**
  1. GCP runs on instrument storage (same system)
  2. Endpoint host mounts instrument storage (SMB, NFS, other)





# Creating the landing zone(s) for instrument data



# 1. Understand the processing model

- **Scenario 1: Data moves from instrument to “adjacent” cluster for post-processing; post-processed data moved to institutional storage**
- **Scenario 2: Data transferred to researcher-owned storage in RCC or other compute facility**
- **Scenario 3: Data held in facility-owned storage; PI initiates transfer to compute-accessible storage**
  - More common in outsourced providers, national facilities



## 2. Prepare the landing zone (“scratch” area)

- **Who owns the landing zone?**
  - Facility: You set it up once and configure for each job
  - Researcher: Researcher configures; may use a Flow to set create directories, set permissions, etc.
- **Collections and permissions**
  - Landing zone must be *writable* by whomever requests the transfer: facility operator, researcher, or ...service account
  - Guest collection provides the most flexibility



# Configuring collections for automation



# Configure a Globus Guest Collection

- **Automation → Guest collection**
  - The alternative is a lot more work!
- **Considerations:**
  - How is the instrument output organized?
  - Who will be requesting (initiating) transfers?
- **Best practice: Use (another) service account with flow run permissions; grant 'R' access to guest collection**





# Guest collection use pattern 1

- **Create a guest collection at top level directory**
  - Done by a user who has a local account
- **For each experiment/project/modality**
  - Create a folder
  - Set permissions for PI/collaborators to read data from the folder
- **Can automate permission management by using local policy store**



# Guest collection use pattern 2

- **Create a guest collection for each experiment, project, modality**
  - Grant Access Manager role to PI for managing permissions
  - Set permissions for collaborators to read data from the folder
- **Can automate role and permission management by using local policy store**



# Create the guest collection

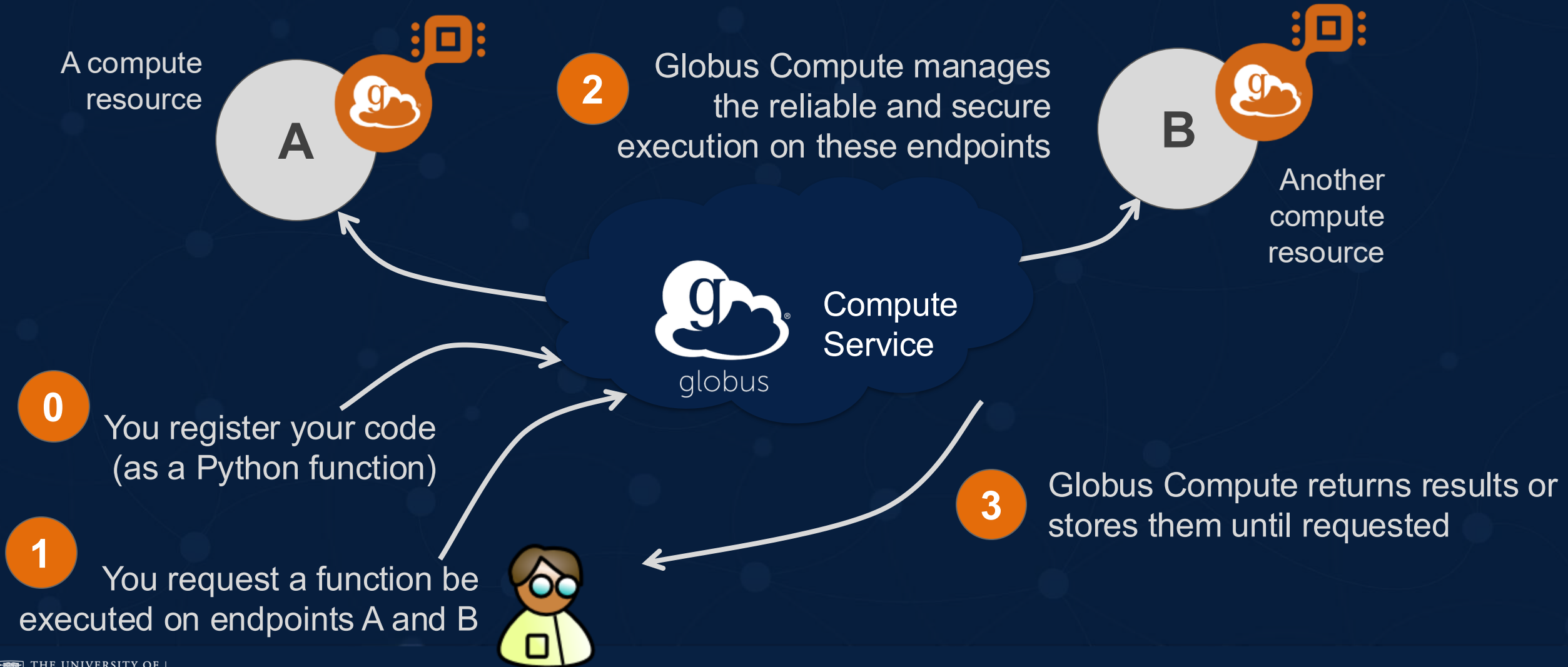
- 1. Create a guest collection**  
— Pay attention to root path
- 2. Create a Group to help you manage sharing**
- 3. Grant the service account/group permissions on the guest collection**
- 4. Make service account an Access Manager on the guest collection**




# Preparing computation environment and post processing code(s)



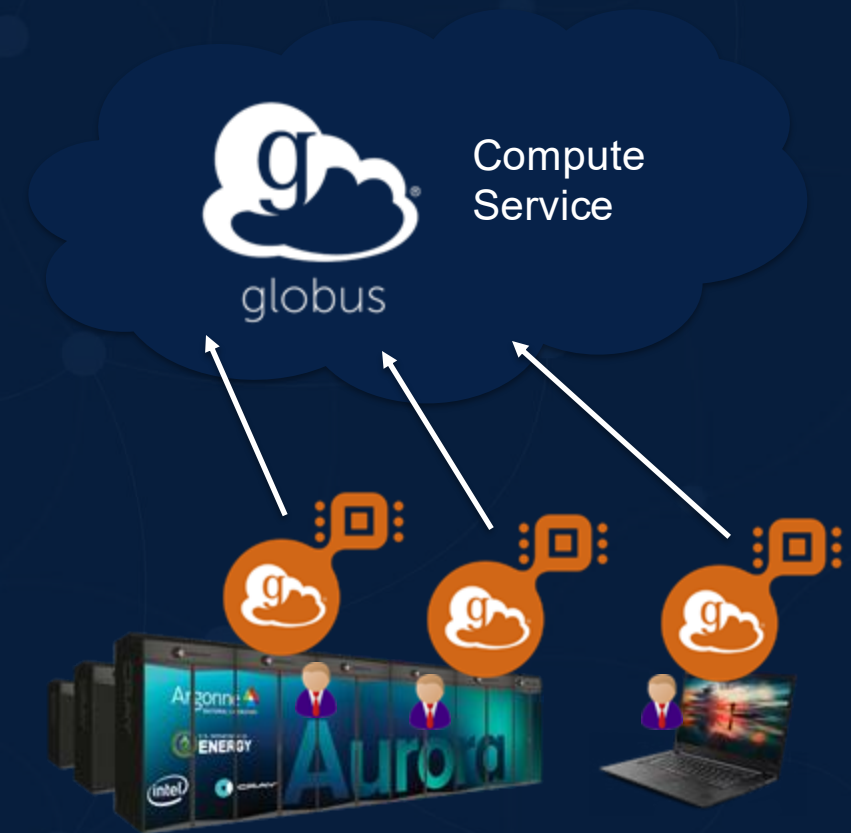
# Globus Compute from the researcher's PoV





 Globus Compute transforms any computing resource into a function serving endpoint

- **Python pip installable agent**
- **Elastic resource provisioning from local, cluster, or cloud system (via Parsl)**
- **Parallel execution using local fork or via common schedulers**
  - Slurm, PBS, LSF, Cobalt, K8s



# Installing a single-user compute endpoint

```
$ source ./compute/bin/activate
$ globus-compute-endpoint configure PEARC25-Tutorial-Endpoint
Created profile for endpoint named < PEARC25-Tutorial-Endpoint >
```

Configuration file: /home/vas/.globus\_compute/PEARC25-Tutorial-Endpoint /config.yaml

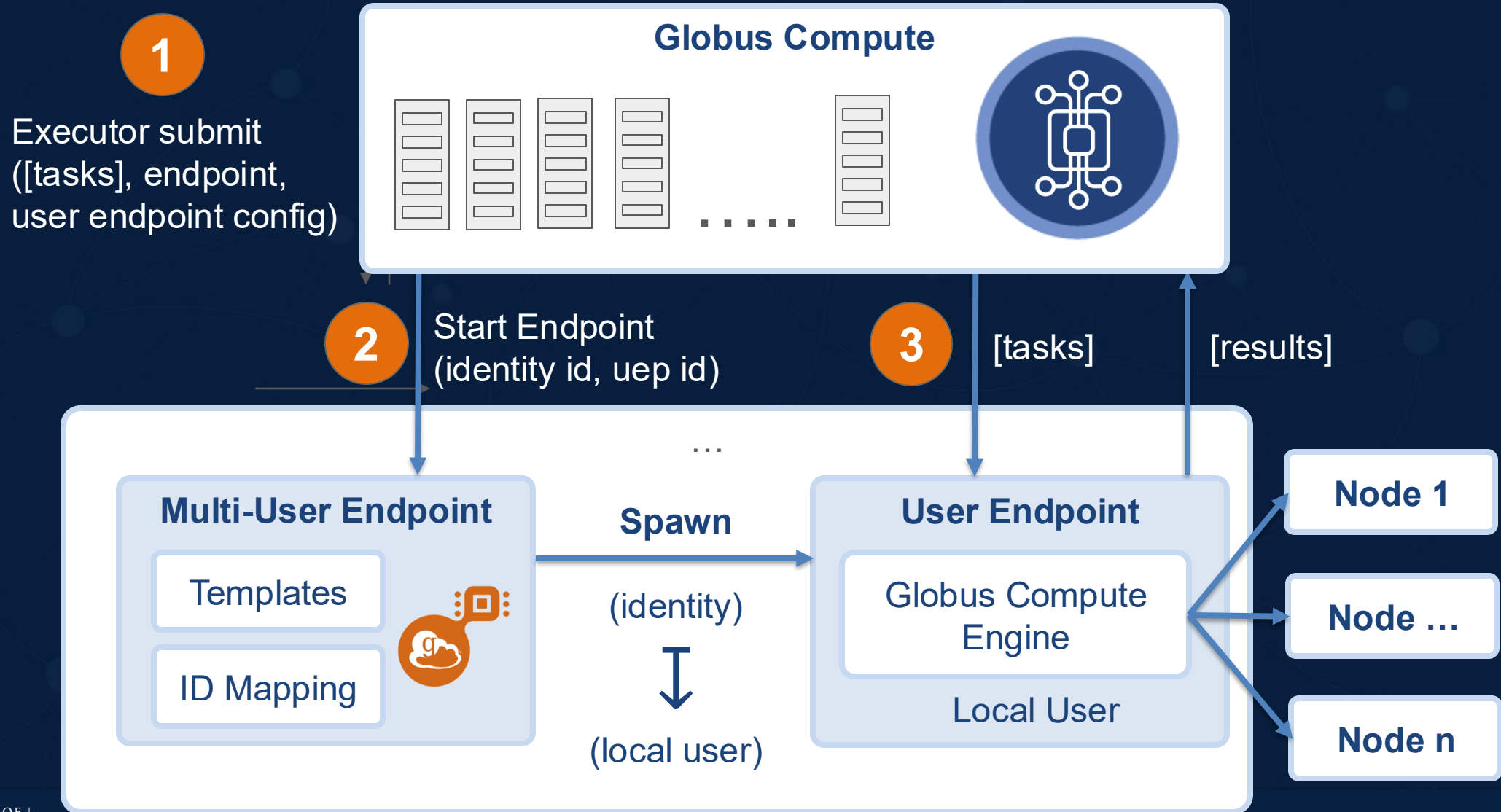
Use the `start` subcommand to run it:

```
$ globus-compute-endpoint start PEARC25-Tutorial-Endpoint
```

```
$ globus-compute-endpoint start PEARC25-Tutorial-Endpoint
Starting endpoint; registered ID: 54460200-b652-4f43-a918-02882fa6114a
```



# Multi-user Endpoints: Architecture





# Multi-user Compute Endpoints

## Benefits for Researcher

- No need to maintain multiple endpoints for different configurations
- Specify configuration at task submission
- No need to log in to the terminal

## Benefits for Administrator

- Templatable endpoint configurations, e.g., pre-select executor, enforce limits
- No orphaned user endpoints
- Standard Globus identity mapping
- Lower user support overhead



# 1. Configure compute endpoint

- **Globus compute agent must be installed where each postprocessing step will run**
  - Facility/instrument-adjacent cluster → Single-user endpoint
  - RCC cluster, cloud/other shared resource → Multi-user endpoint
- **Configure executor, e.g., Slurm job parameters**
- **Considerations: local account that compute tasks run as**
  - Must have access to compute endpoint and scratch space
  - Must have necessary allocations (# nodes, capability, time, ...)?





## 2. Prepare compute code(s)

- **Package and register functions with Globus Compute**
  - Data processing
  - Metadata extraction
- **Ensure local user can load required environment(s)**
- **Ensure imported packages are in place**
- **Ensure Python versions match (serialization, ugh!)**



# Preparing for data sharing and/or publication

# Globus Search provides FAIRness foundation

- **Metadata store with fine-grained visibility controls**
- **Schema agnostic dynamic schemas**
- **Simple search using URL query parameters**
- **Complex search using search request document**



[docs.globus.org/api/search](https://docs.globus.org/api/search)



# Capture metadata early and often

- **...about the instrument: equipment, calibration data, software version, ...**
- **...about the run: researcher/project/experiment IDs, sample metadata, time/duration, preparation and/or procedural data, ...**
- **Hardcode less variable attributes, or enter before run**
- **Register Compute function(s) to extract dynamic (sample/run-based) variables**



# Set up Search index

- **Create index and assign roles**
  - Service account must have writer access
- **Decide on schema and search facets**
  - No Globus-imposed constraints
  - Extensible as needs evolve
- **Select ID type/service (DataCite DOI, ORCID ID, ...)**
  - Configure minting service (get credentials, decide on test vs. production environment)



# Persistent identifiers are key to data FAIRness

- Add a PID minting step to your flow
- Include PID when ingesting metadata to Globus Search

[docs.globus.org/api/flows/hosted-action-providers/ap-datacite-mint](https://docs.globus.org/api/flows/hosted-action-providers/ap-datacite-mint)



**DataCite**

FIND, ACCESS, AND REUSE DATA





# Creating and deploying flows for automation



# Flow definition

```
"StartAt": "TransferFiles",
"States": {
  "TransferFiles": {
    "Comment": "Transfer to a guest collection",
    "Type": "Action",
    "ActionUrl": "https://actions.automate.globus.org/transfer/transfer",
    "Parameters": {
      "source_endpoint_id.$": "$.input.source.id",
      "destination_endpoint_id.$": "$.input.destination.id",
      "transfer_items": [
        {
          "source_path.$": "$.input.source.path",
          "destination_path.$": "$.input.destination.path",
          "recursive.$": "$.input.recursive_tx"
        }
      ]
    },
    "ResultPath": "$.TransferFiles",
    "WaitTime": 60,
    "Next": "SetPermission"
  },
  "SetPermission": {
    .....
    "End": True
  }
}
```

Action

Action Provider URL

Action inputs

Timeout (seconds)

Next state



# Define/deploy the Globus flow

- **Start with published flow definitions**
  - [github.com/globus/globus-flows-trigger-examples](https://github.com/globus/globus-flows-trigger-examples)
  - [docs.globus.org/api/flows/authoring-flows/examples/](https://docs.globus.org/api/flows/authoring-flows/examples/)
- **Manage flow definitions in a version control system**
- **Validation tools**
  - Flows IDE: <https://globus.github.io/flows-ide/>
  - Globus CLI: *globus flows validate*
- **Be agile :-) build/test each action incrementally; add error handling**



# Create the flow



```
$ globus login  
$ cd ~/pearc25-tutorial  
$ globus flows create FLOW_NAME \  
> definition.json --input-schema schema.json
```

- **Success returns the flow ID**
- **Inspect the flow using the web app**




# Decide who/what will run the flow

- **Instruments are closed environments**
  - Little/no visibility into run progress
  - Restricted access to capture dir (outside of device software)
- ➔ **Flow trigger is sometimes custom code**
- ➔ **More often a human may need to run the flow**
- **Decide how to pass inputs (depends on trigger )**



# Permissions to run the flow


 Assign New Role


Assign To

☒ User  
☐ Group  
☐ All Logged In Users  
☐ Public (anonymous users)


Username or Email


Role

 ☐ Administrator  
can start this flow, view this flow and associated activity, and modify this flow

 ☐ Starter  
can start this flow and view associated activity

☐ Viewer  
can view this flow and associated activity

 ☐ Run Manager  
can view and manage all runs of this flow

 ☐ Run Monitor  
can view all runs of this flow

Add Role

Cancel

**Set  
permission  
for the service  
account to  
run the flow**





# Key consideration: Be identity aware



- **What identity is the flow running as?**
- **Does identity have access to target resources?**
  - Collections (ideally, guest collections)
  - Compute endpoint
  - Compute function
- **Does identity have the required role?**
  - Access Manager, if granting/revoking permissions



# Triggering flows on instruments ...and other resources





# Instruments don't play nice!

- **Determining the trigger event is not an exact science**
- **Custom code almost always required**
- **Locked-down environments may necessitate a manual step/triggering of the flow**

# Creating and triggering runs of our flow



- 1. Edit the monitor (trigger.py) script**
- 2. Ensure GCP is running on the instrument**
- 3. Run the monitor script**
- 4. Trigger the flow**



# Ask for help! Really, please.

- **Guidance on best practices**
- **Sounding board for your design/implementation**
- **Assistance with developing flows, solutions**
- **All at no cost to you ...just reach out**