Sage

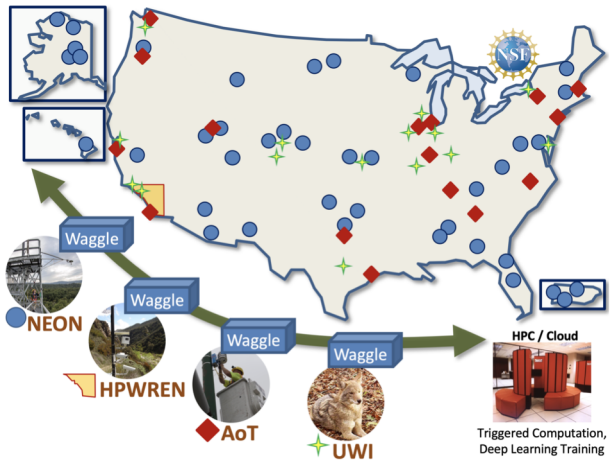
A Software-Defined Sensor Network

Cyberinfrastructure for Edge Computing

www.sagecontinuum.org

SAGE Edge Code Repository

Wolfgang Gerlach



Northwestern University
THE UNIVERSITY OF CHICAGO
Northern Illinois University
Argonne NATIONAL LABORATORY
Colorado State University
UC San Diego
GEORGE MASON UNIVERSITY
ARRAY of THINGS
ARM
neon National Ecological Observatory Network
HPWREN
SDSC SAN DIEGO SUPERCOMPUTER CENTER
LINCOLN PARK ZOO. FOR WILDLIFE. FOR ALL.

Overview

- Edge Code Repository
- Access to SAGE
- Example: access to SAGE object store

ECR: Edge Code Repository

“app store” for AI@Edge:

- Collection of edge code software that can be execute on SAGE nodes

- Process to test, profile and review edge code before it can be scheduled
  ⇨ metadata required by SAGE Edge Scheduler and node resource manager

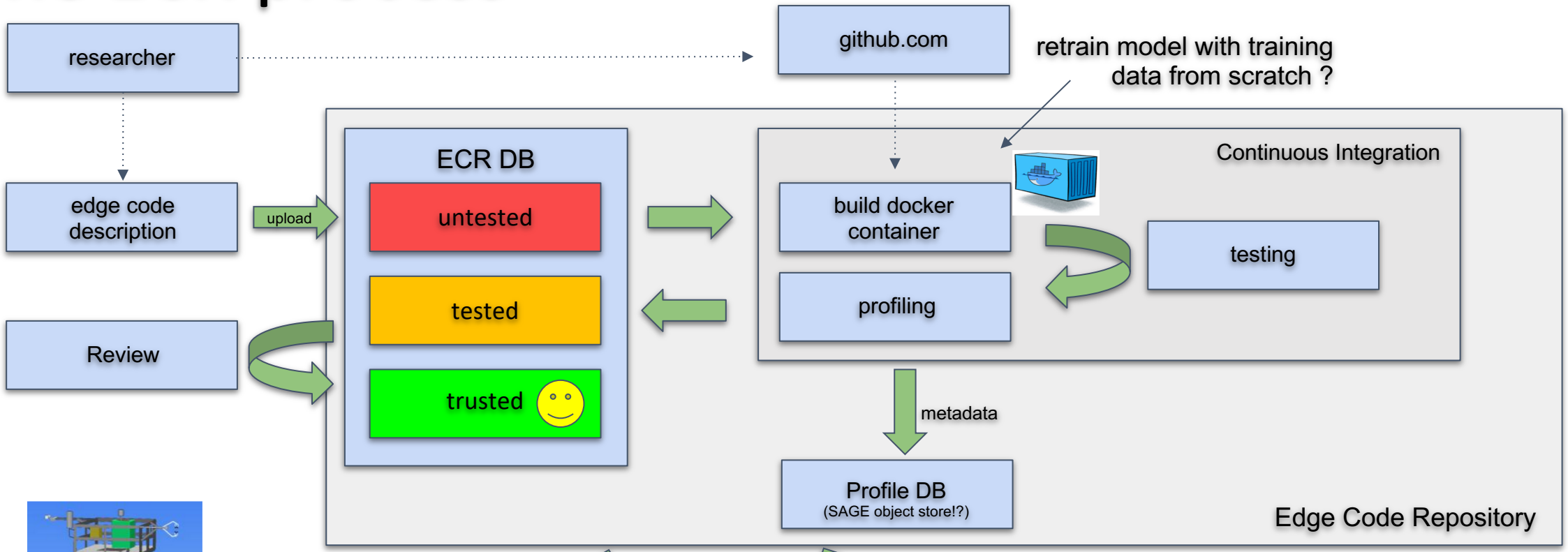
Sage

Edge Code Description (simplified)

- name
- description
- pointer to git repository
- Dockerfile
- hardware requirements (architecture, GPU...)
- provenance information:
  - how was model created/trained
  - link to training data
- specification of input parameters to control behavior
- …
- -> Unique identifier for use in Edge Code Scheduler (ECR)

Sage

The ECR process

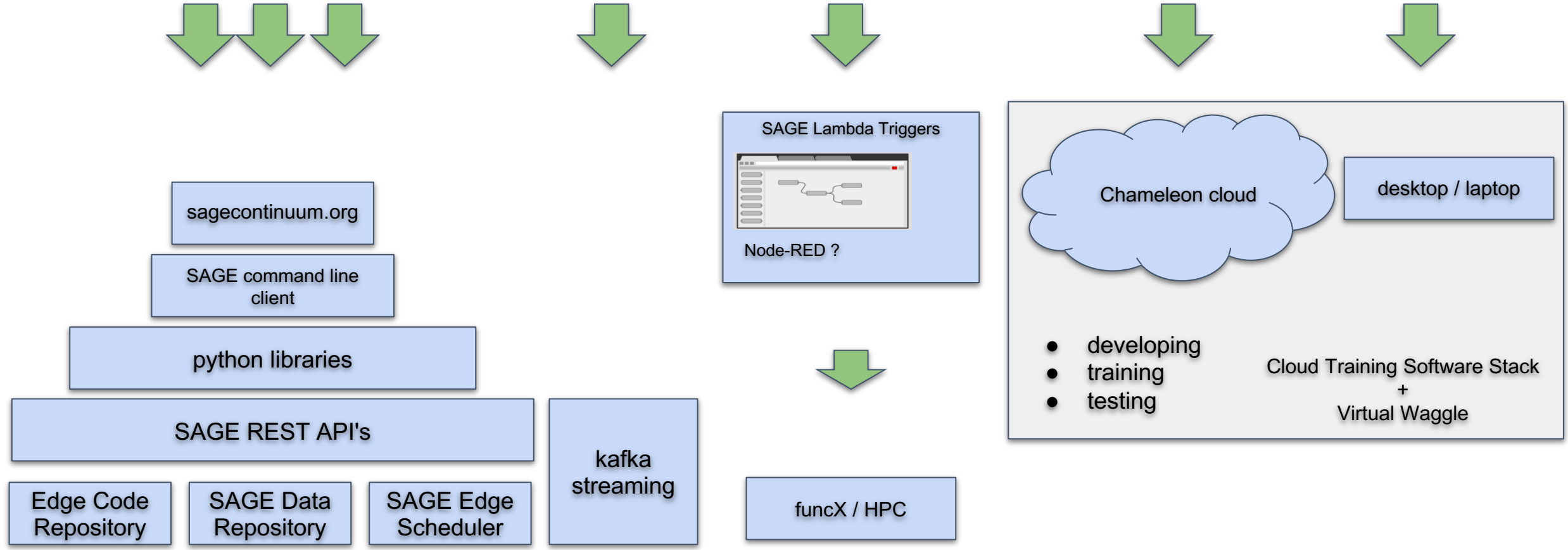


Sage

Testing and Profiling of Edge Code

- edge codes need to be reliable
  - developers need to provide realistic tests / input data for testing
- scheduler needs to know if edge code "fits"
  - RAM, CPU, GPU competition on the edge
  - collect metrics
  - -> Can edge code run concurrently ?
  - -> Can edge code be executed every N minutes ?
- RAM, CPU and GPU limits for edge code containers

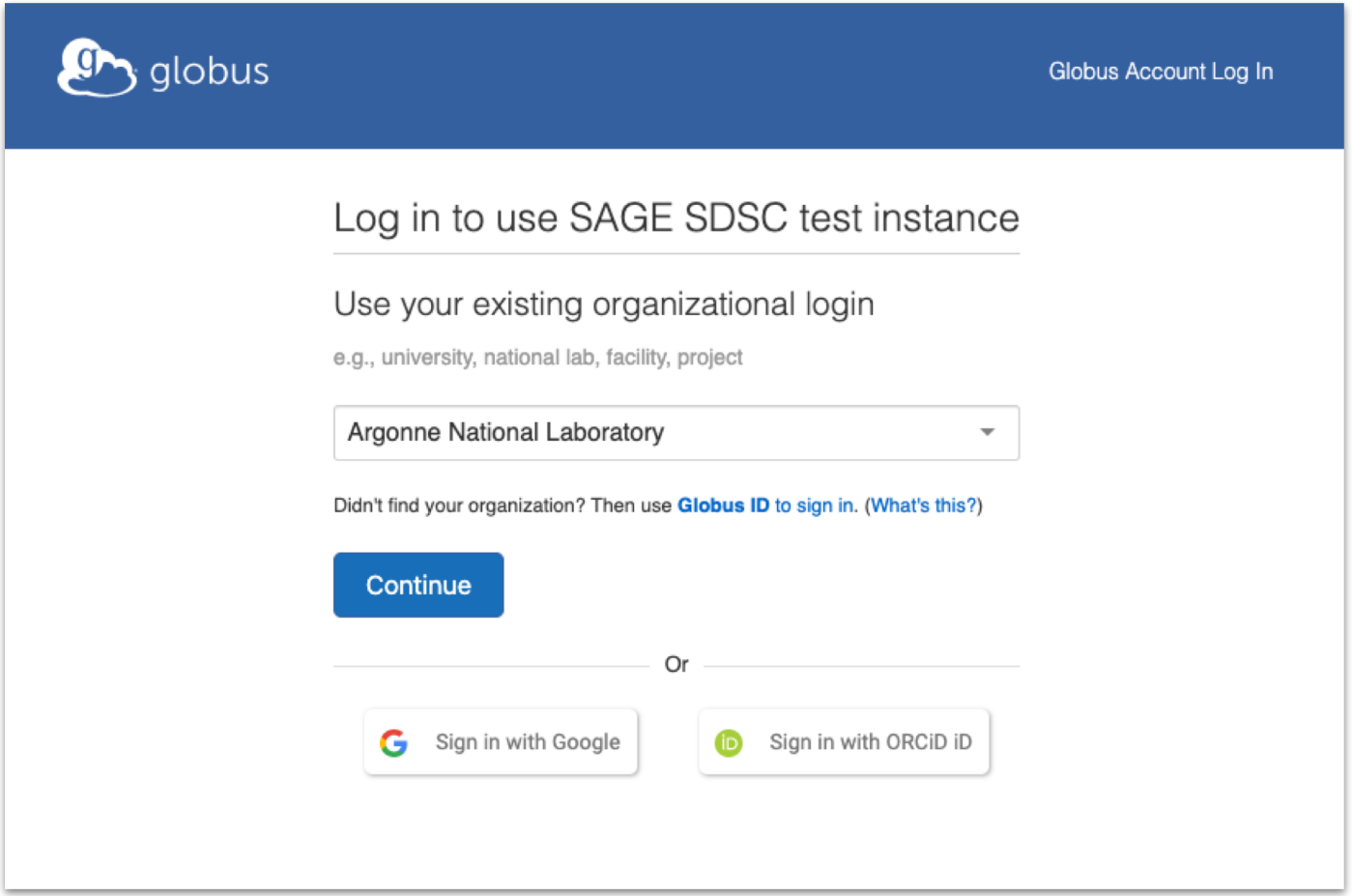
Access to SAGE resources



Sage

SAGE Authentication

- login via institutional login
- OAuth2 tokens for data access and other resources



Sage

SAGE access, example: SAGE object store

- Training data (pictures, video) (from edge and/or users)
- ML models
- Similar to S3, but each bucket contains only one data set
  - (implementation & documentation: https://github.com/sagecontinuum/sage-storage-api)
- Access:
  - SAGE website (not yet)
  - JSON REST API
  - python client library
  - command line client
- Authentication via SAGE user tokens
- Access management: `FULL_CONTROL`, `WRITE`, `READ`, `WRITE_ACP`, `READ_ACP`

Sage

SAGE access, example: CLI

Create bucket

```
> export SAGE_USER_TOKEN=user:testuser
> export SAGE_HOST=http://localhost:8080
> ./sage-cli.py storage bucket create --datatype training-data --name camera7
{
  "id": "797b9cf4-4506-486a-86ac-b2299ab83243",
  "name": "camera7",
  "owner": "testuser",
  "type": "training-data",
  "time_created": "2020-05-06T18:41:19Z",
  "time_last_updated": "2020-05-06T18:41:19Z"
}
```

Sage

https://github.com/sagecontinuum/sage-cli

SAGE access, example: API

Upload files into bucket

```
> curl -X PUT \
    "localhost:8080/api/v1/objects/797b9cf4-4506-486a-86ac-b2299ab83243/" \
    -H "Authorization: sage <secret_token>" \
    -F 'file=@camera_picture1.jpg'
{
  "bucket-id": "797b9cf4-4506-486a-86ac-b2299ab83243",
  "key": "/camera_picture1.jpg"
}
```

Sage

https://github.com/sagecontinuum/sage-storage-api

Summary/Roadmap

- 2021-Q3: Cyber Infrastructure completed (excl. web portal and edge scheduler)
  - User management & Authentication [90%]
  - SAGE data repository
    - SAGE object store [80%]
    - Near real-time data stream [discussion]
    - SAGE sensor data store [discussion]
  - SAGE Edge Code Repository (api, db, CI for testing/profiling) [design]
  - CTSS [software 50%, working on smooth integration with Chameleon]
  - VW [working early prototype]
- 2021-Q4: SAGE Scheduler completed [design]
- 2022-Q1: SAGE Web Portal fully deployed [framework]

Sage