Syndicate: Software-defined Wide-area Storage

Jude Nelson
Princeton University

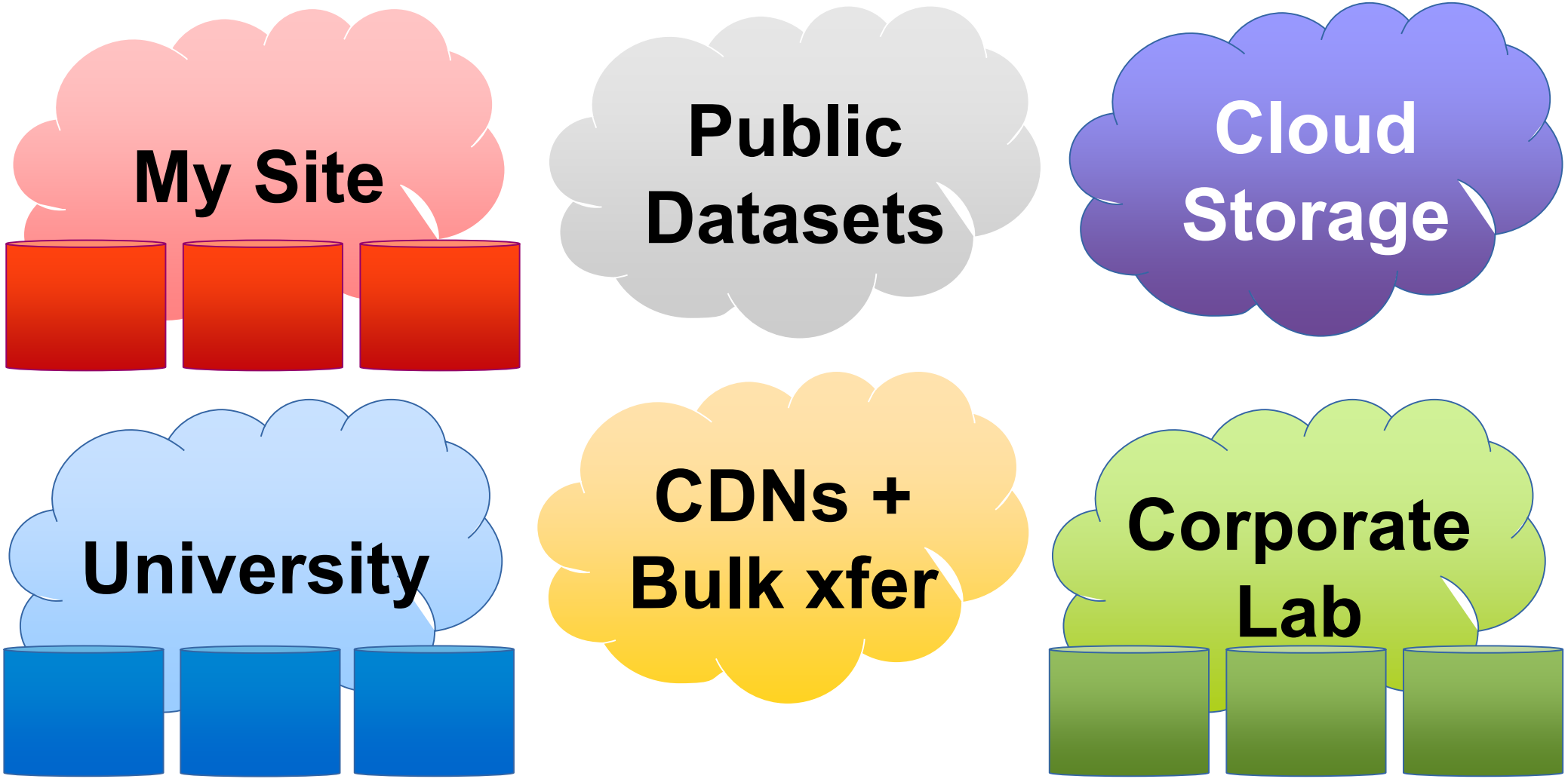
Background

- CCI*DIBBS NSF #1541318
- Princeton University + University of Arizona
  - OpenCloud + CyVerse (iPlant)
- Next-generation storage system
  - Coming online this year
  - Seeking community input and advise

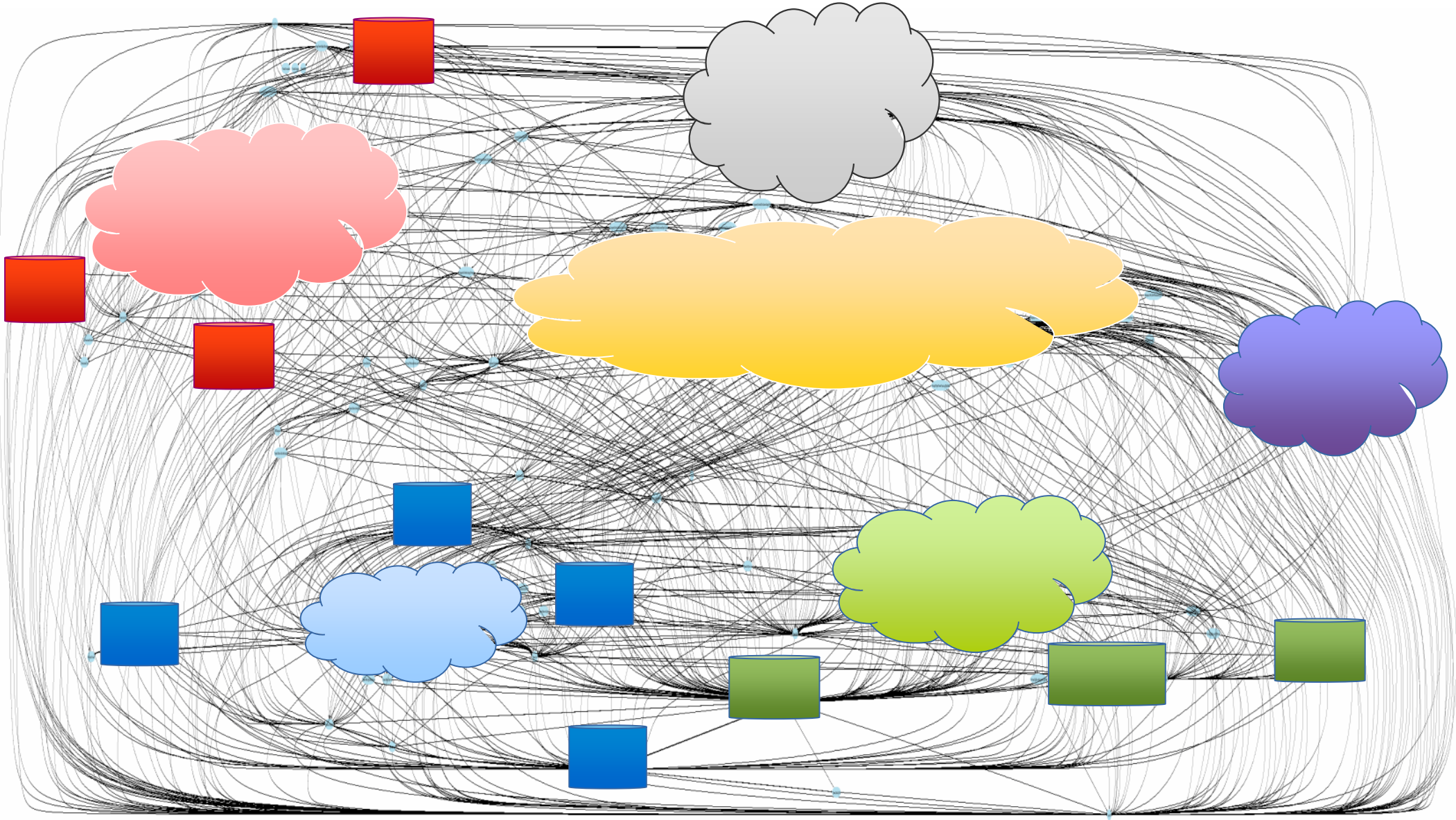
Outline

- Problem Formulation
- What is Syndicate?
- Sample Applications
- UI/UX
- Status

The Good: Lots of Data Sources!



The Bad: Lots of Data Flows



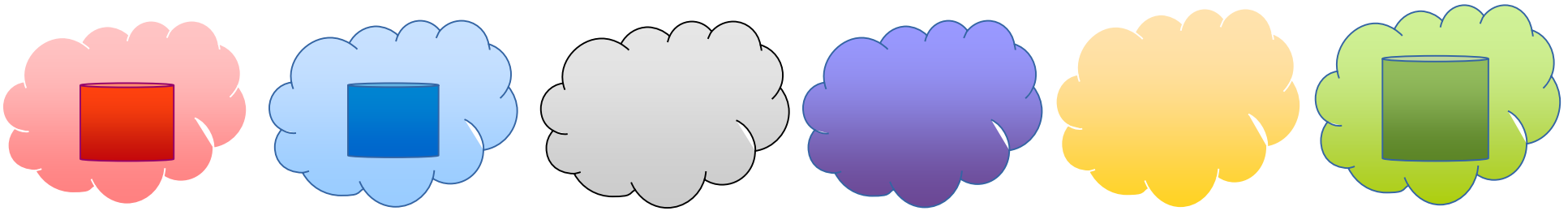
The Ugly: Storage Reintegration

Workflow logic

Drivers are only the beginning...

- Consistency
- Confidentiality
- Formatting
- Fault tolerance
- Access control
- Retention
- Authentication
- ...etc...

Driver | Driver | Driver | Driver | Driver | Driver

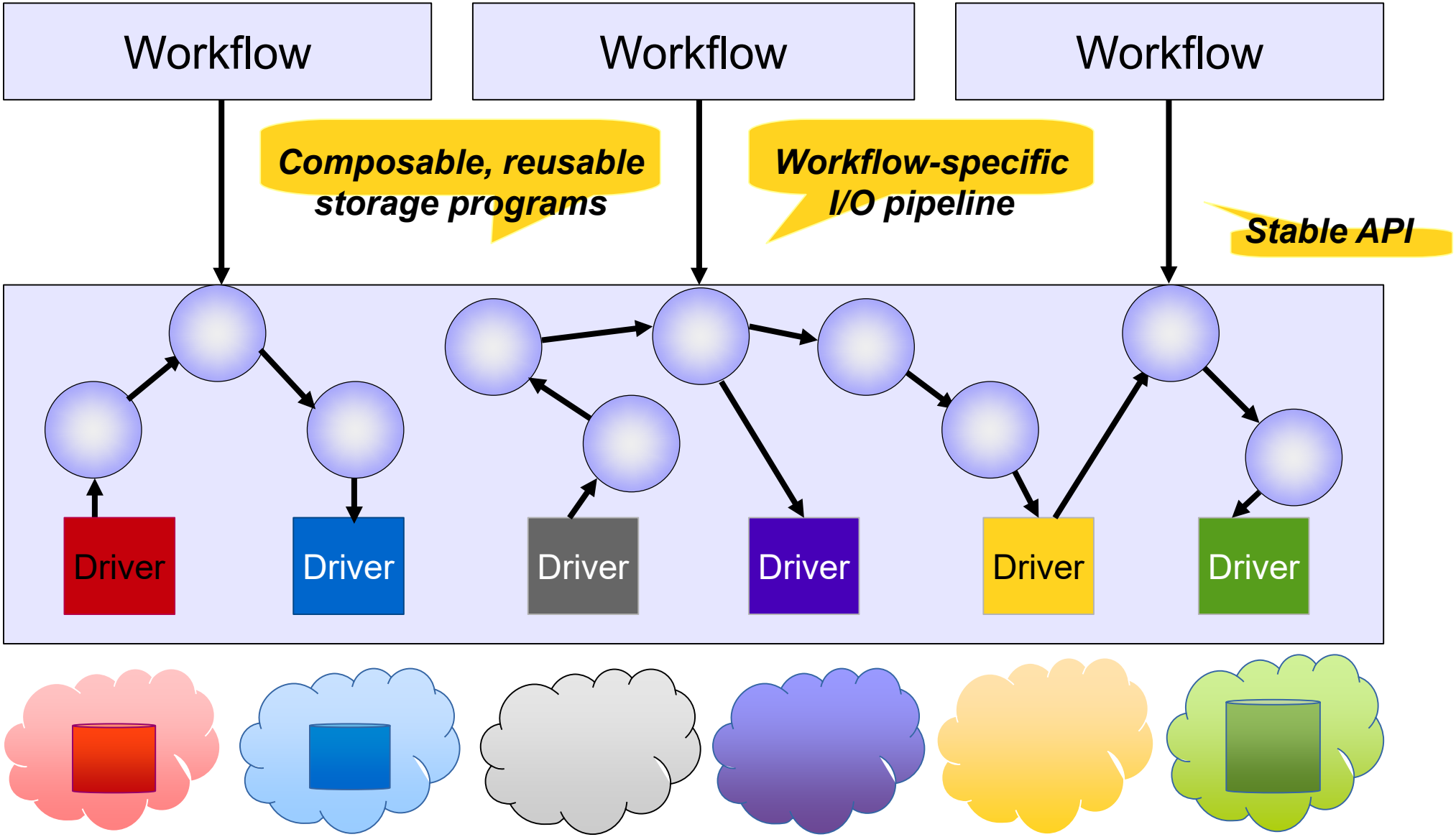


Each workflow implements a built-in bespoke storage system!

Prior Work

- iRODS
  - Intra-site programmable storage
- Parrot Virtual FS
  - Driver layer for legacy services
- CernVM FS
  - Wide-area
  - End-to-end guarantees
  - Read-only

Syndicate: Programmable Storage

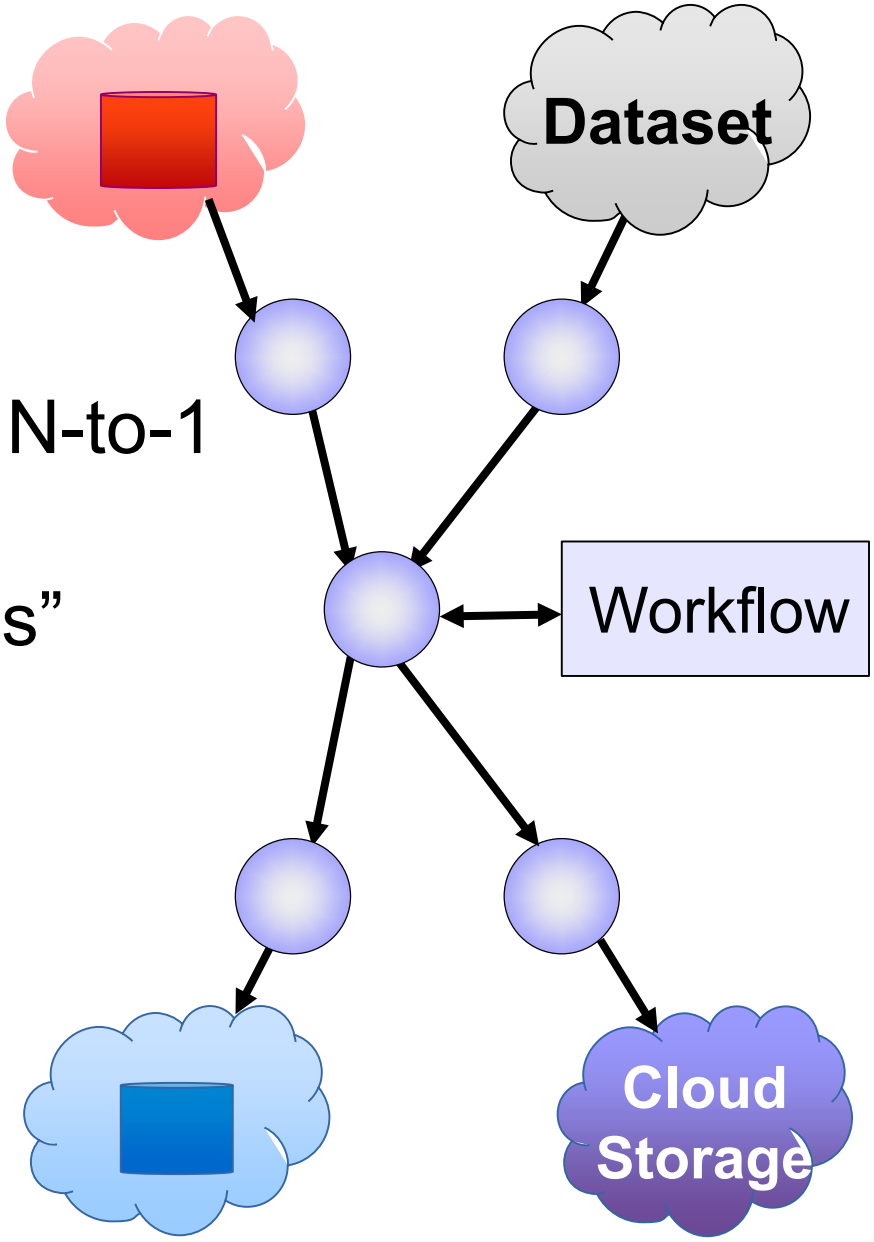


Why Syndicate?

- Spans multiple sites and services
  - End-to-end authenticity
  - End-to-end correctness
  - No central points of trust
- Minimizes operational costs
  - Isolates, composes reusable storage logic
  - Reprogrammable fabric → Immutable workflows
  - Self-managing (SDN-like)

Syndicate Programming Model

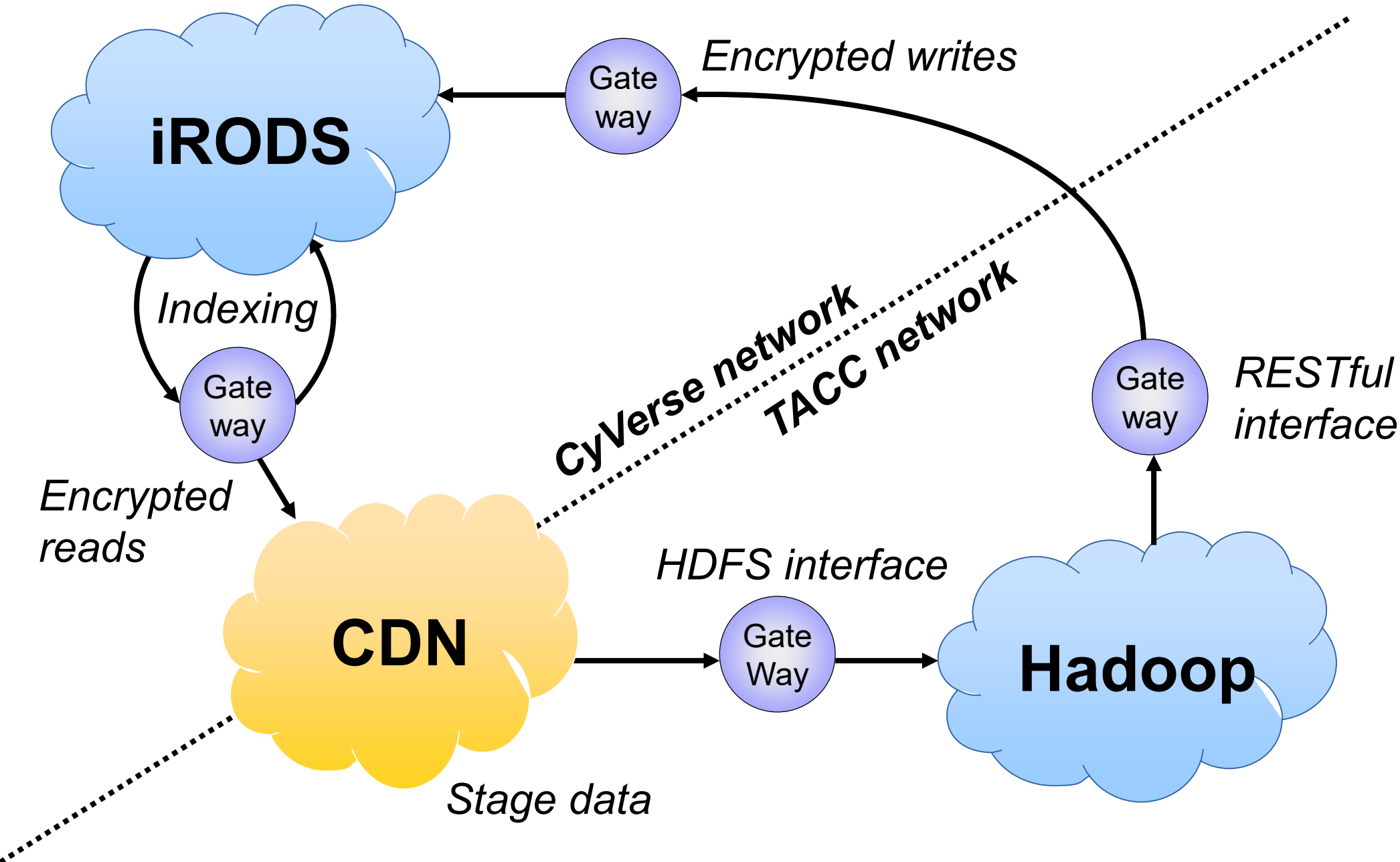
- Storage Programs
  - UNIX-y data plane
  - I/O flow: typed byte stream
  - Composition: 1-to-1, 1-to-N, N-to-1
- Gateways
  - A storage program's "process"
  - Stable workflow interface
- Syndicate
  - The "shell" for gateways
  -



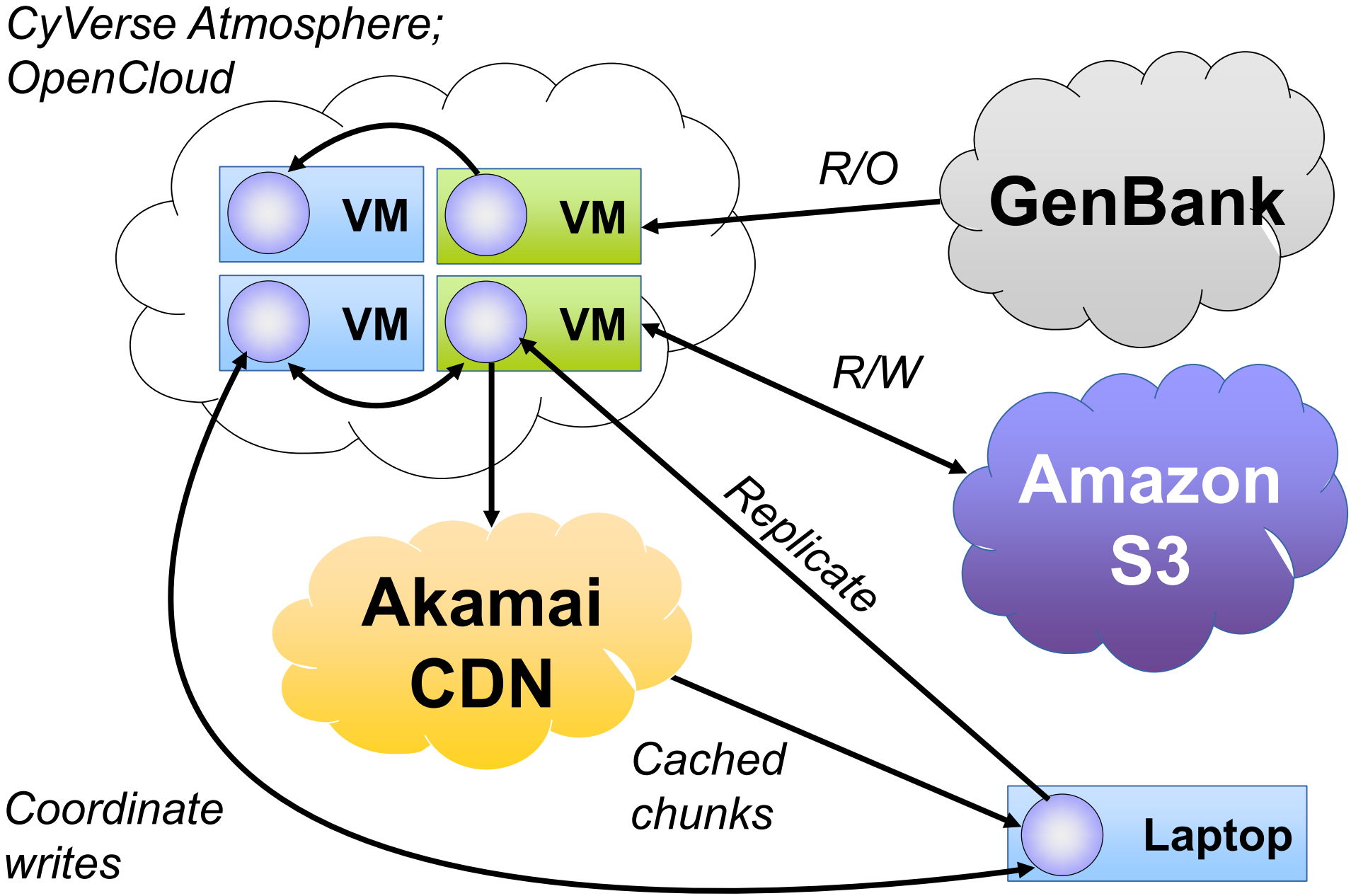
Syndicate Usage

- Volume
  - Tagged filesystem abstraction
  - Set of cooperating gateways
  - Workflow-specific data-plane behavior
- Users
  - Own, control, and run gateways
  - Volume owner: controls admission
  -

Real-world Volume (1)



Real-world Volume (2)



Spanning Multiple Networks

- Global control plane
  - Membership; configuration; I/O pipeline construction
  - Metadata Service (MS) in Google AppEngine
- Blockstack (USENIX ATC 2016)
  - Public LDAP-like DB
  - Control plane trust anchor
  - *All nodes independently construct the same DB*
    - DB journal embedded in a PoW blockchain
    - **No central points of trust!**

User Experience

1) PI makes user accounts
2) Users make volumes
3) Volume owners make and assign gateways
4) Users point client at volume owners

- Client looks up volume owners in Blockstack
- Client discovers accessible volumes
- Client configures and runs gateways

Operator Experience

1) Bake Syndicate into VM images
2) Run site-local Blockstack server
3) Run Syndicate MS in Google AppEngine
4)
5)
6)
7) (optional) Run gateways on users' behalf

- MS is *untrusted*
- Helps gateway discovery
- Authentication through Blockstack

System Status

- Driver support
  - Amazon S3, Google Drive, Box.net, Dropbox, …
  - GenBank, M-Lab, iRODS, local disk, …
  - FUSE, Node.js, HDFS, shell programs, …
- Blockstack in production since 2015
  - https://github.com/blockstack
- Syndicate is alpha
  - Usable, with quirks
  - https://github.com/syndicate-storage

Thank you!

Questions?