Sneaking Data into Containers with the Whole Tale

Adam Brinckman, Mary Terese Campbell, Kyle Chard, Niall Gaffney,
Mihael Hategan, Matthew B. Jones, Kacper Kowalik, Sivakumar Kulasekaran,
Bertram Ludäscher, Bryce Mecum, Jaroslaw Nabrzyski, Damian Perez,
Victoria Stodden, Ian Taylor, Thomas Thelen, Matthew Turk, Kandace Turner,
Craig Willis and Sebastian Wyngaard

Speaker: Kacper Kowalik

SciPy'18

This material is based upon work supported by the **National Science Foundation under Grant No. OAC-1541450.**

Outline

- Introduction of Whole Tale
  - Motivation
  - Vision
- Existing solutions
- Overview of the architecture
- Getting data into remote environment via FUSE
- Outlook

Intro

Whole Tale is a *Data Infrastructure Building Block* project under **NSF**'s *Campus Cyberinfrastructure - Data, Networking, and Innovation Program*.

- Distributed team of people: http://wholetale.org/team.html
- Trying to bridge the distinction between **data, code** and **scientific paper.**
- Do all the above in a way that captures **provenance** and allows easily **reproducing** the results.
- Enabling **reuse of data** and **exploration** in unseen ways.

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

THE UNIVERSITY OF CHICAGO

UCSB
UNIVERSITY OF CALIFORNIA
SANTA BARBARA

TEXAS
The University of Texas at Austin

UNIVERSITY OF NOTRE DAME

Motivation: Data & Software importance

- Many scientific experiments, studies, and results are difficult, if not impossible, to replicate, verify, and/or reproduce.
- The scholarly publication has not kept pace with the changes in science.
- Data underpins most research -- whether acquired, derived, or obtained from a repository.
- Computation & software are integral and inseparable components, and are the means via which most research takes place.
- When software is a key part of the discovery process, it should be subject to the same philosophy of transparency as any method.
- **WholeTale aims to capture and preserve the journey towards discovery rather than just the endpoints.**

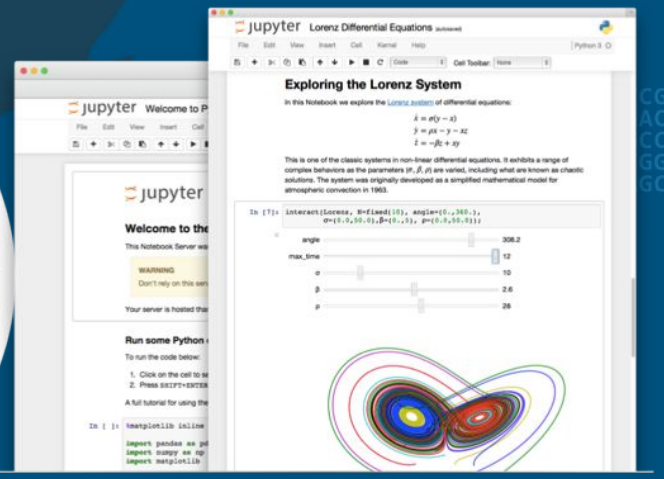
The Whole Tale Vision

- A living publication, preserving all digital scholarly objects, and can be shared and replayed
  - Input, intermediate, and derived data
  - Software and environment
  - Workflow process
  - Publication narrative
- Captures computational steps and provide compute environment
- Provides unique identifiers to objects (DOI)

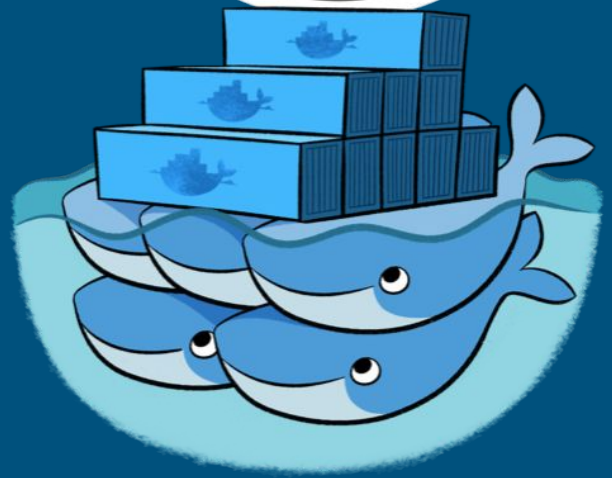
Whole Tale leverages & contributes to **existing Cyber Infrastructure and tools** to support the **whole science story.**

What is a “Tale”?

Data Drive



Front End



Container

Existing solutions

The primary goal of **BinderHub** is creating custom computing environments that can be used by many remote users. BinderHub enables an end user to easily specify a desired computing environment from a GitHub repo. BinderHub then serves the custom computing environment at a URL which users can access remotely.

https://mybinder.org/
https://binderhub.readthedocs.io/en/latest/

Existing solutions

https://swan.web.cern.ch/

SWAN (Service for Web based ANalysis) is a platform to perform interactive data analysis in the cloud.

- Analyse data without the need to install any software
- Jupyter notebook interface as well as shell access from the browser
- Use CERNBox as your home directory and synchronise your local user storage with the cloud
- Access experiments' and user data in the CERN cloud
- Share your work with your colleagues thanks to CERNBox
- Document and preserve science - create catalogues of analyses: encourage reproducible studies and learning by example

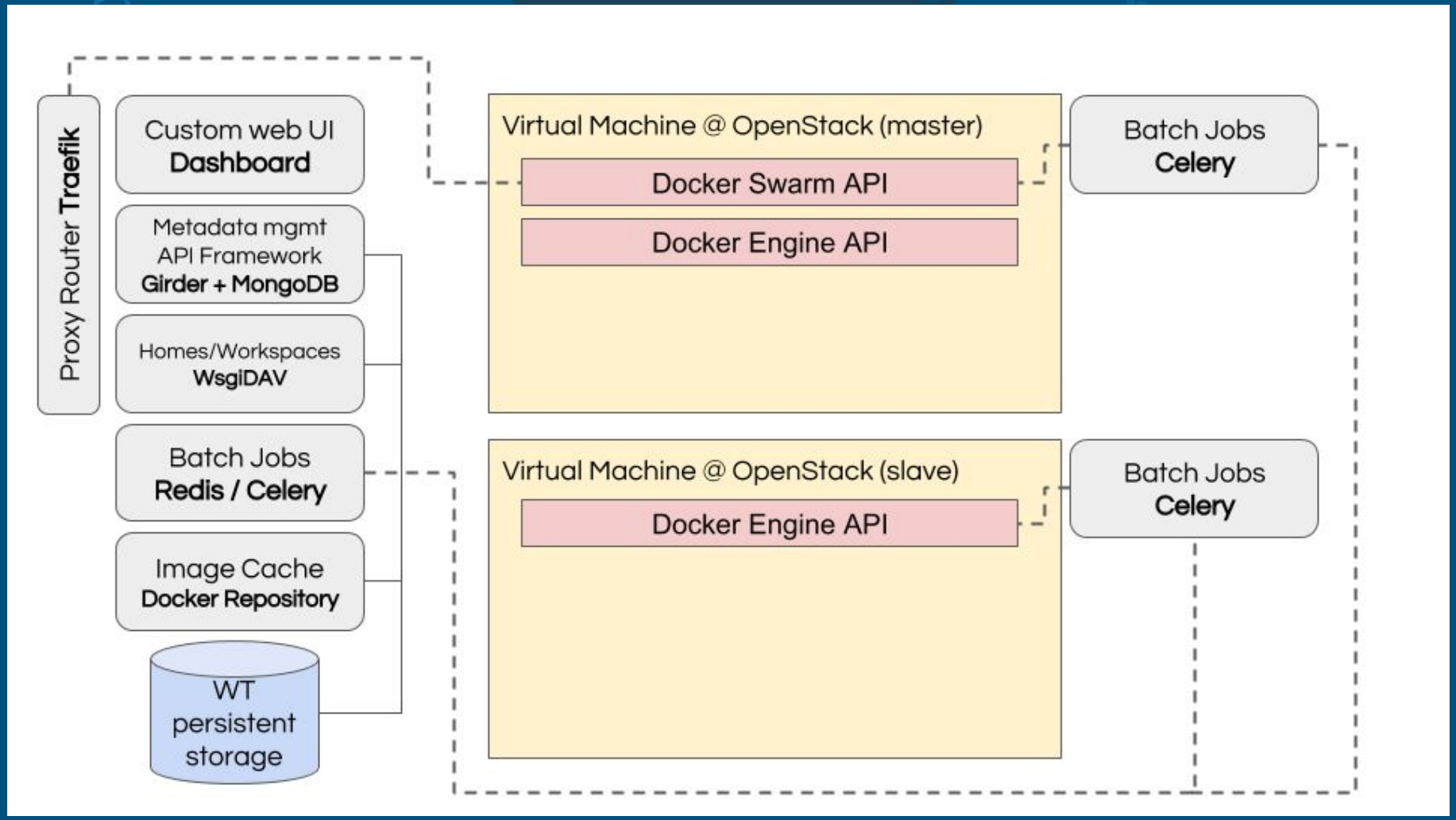
Existing CyberInfrastructure and Tools

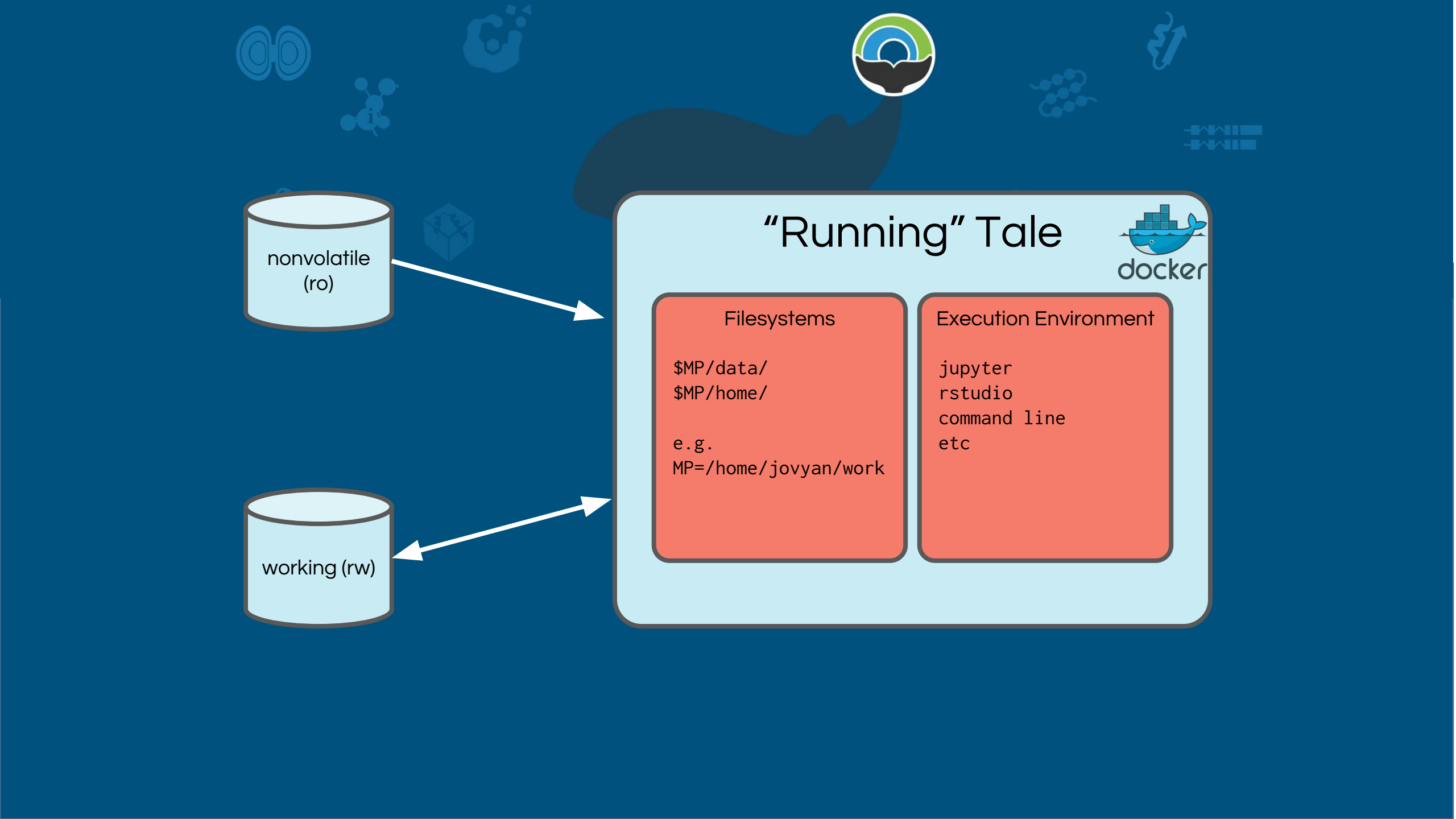
girder (gh:girder/girder)
girder-worker (gh:girder/girder_worker)
Celery (gh:celery/celery)
WsgiDAV (gh:mar10/wsgidav)
PyMongo (gh:mongodb/mongo-python-driver)
fusepy (gh:fusepy/fusepy)

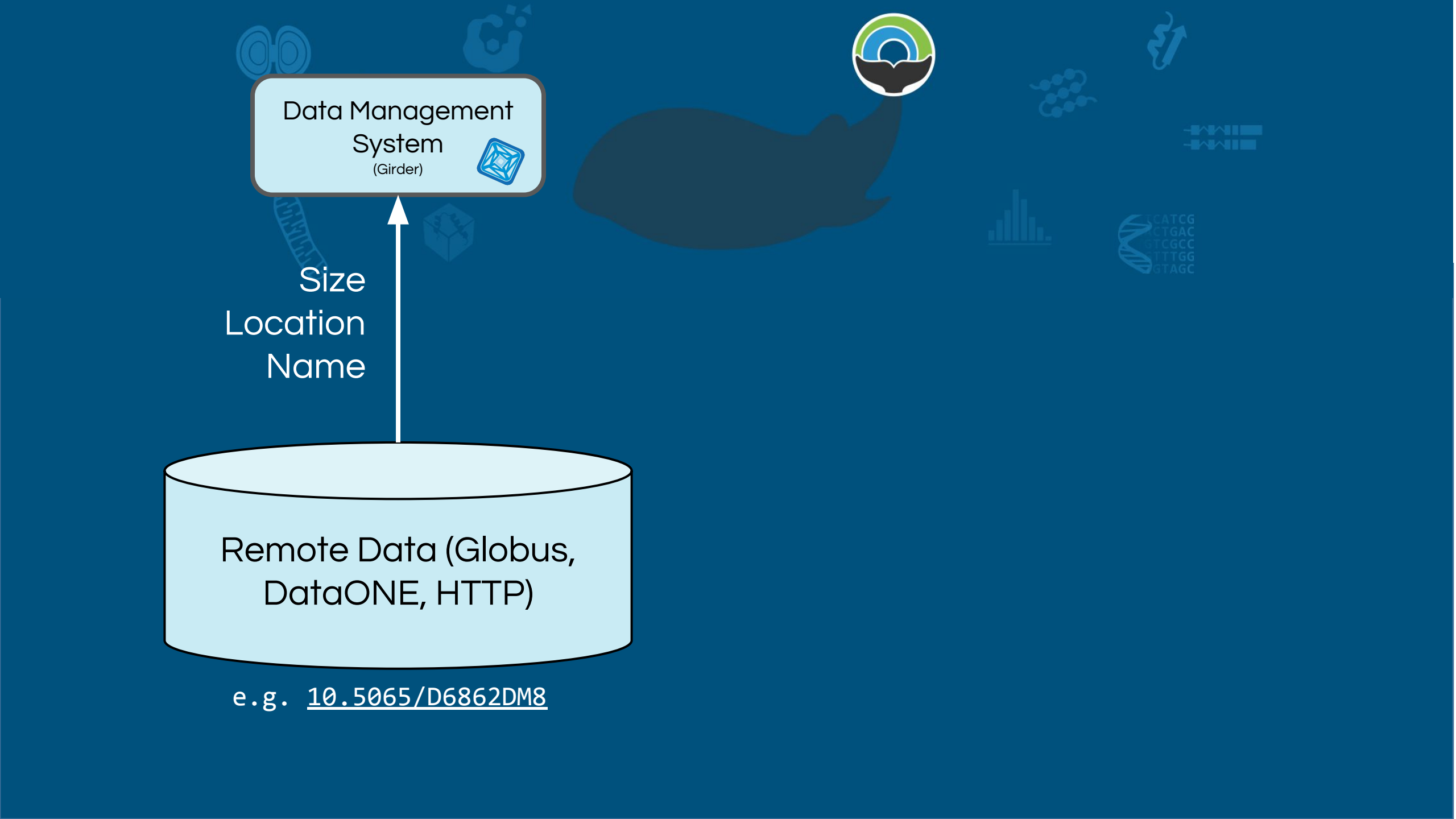
Træfik (gh:containous/traefik)
Redis (https://redis.io)
Docker (https://www.docker.com/)
Terraform (https://www.terraform.io/)
Ember.js (https://www.emberjs.com)

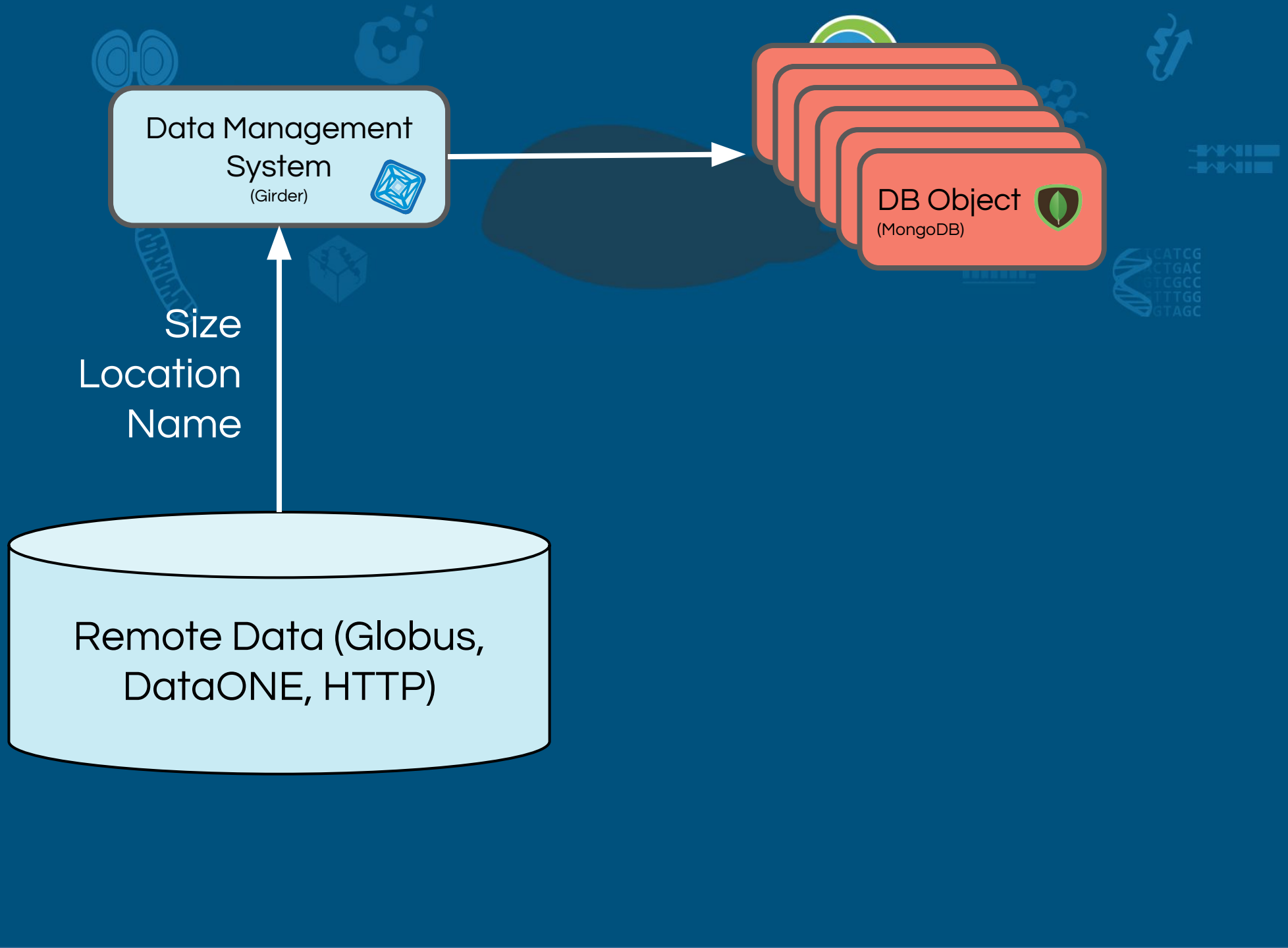
Jetstream: A National Science and Engineering Cloud (https://jetstream-cloud.org/)

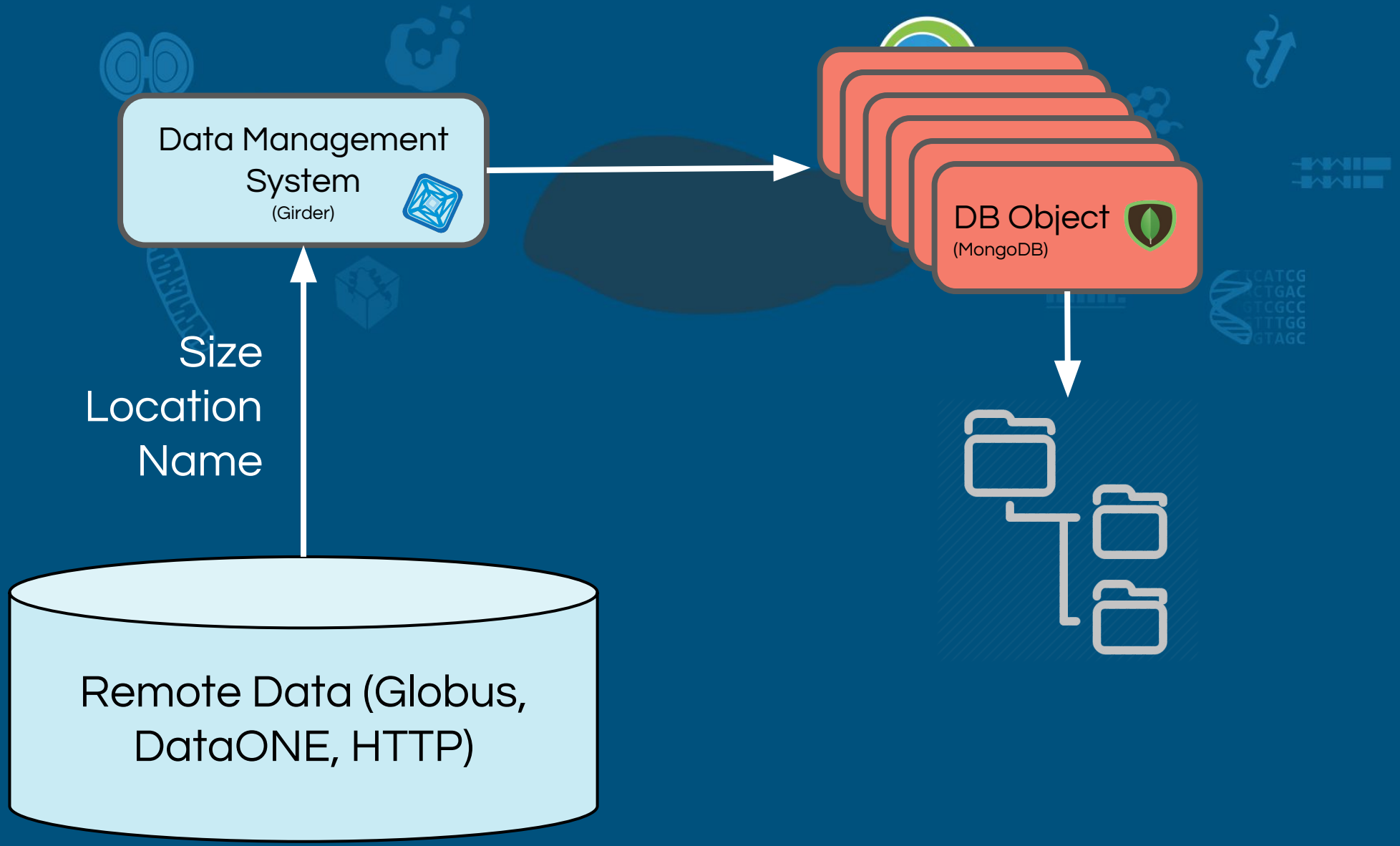
Architecture

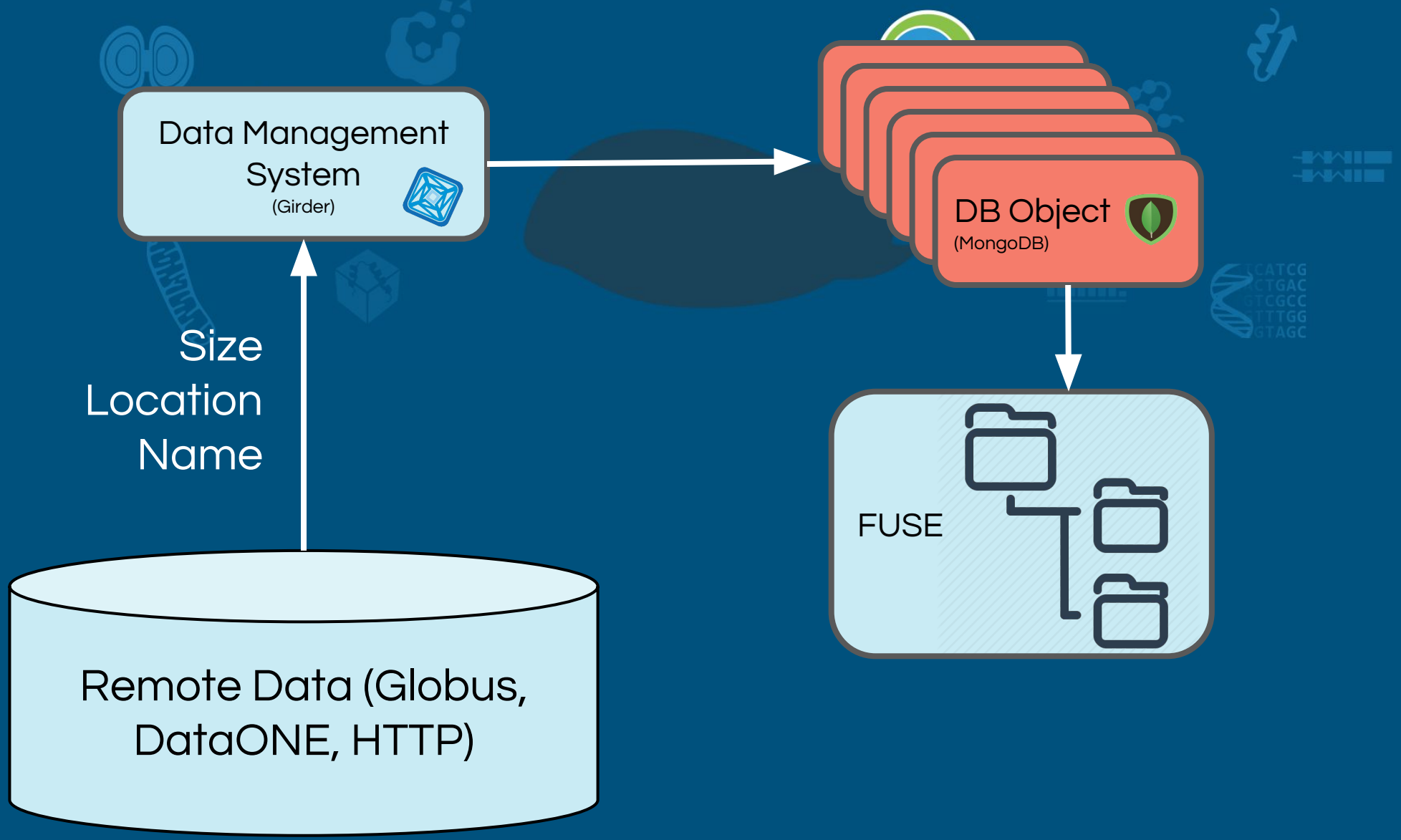


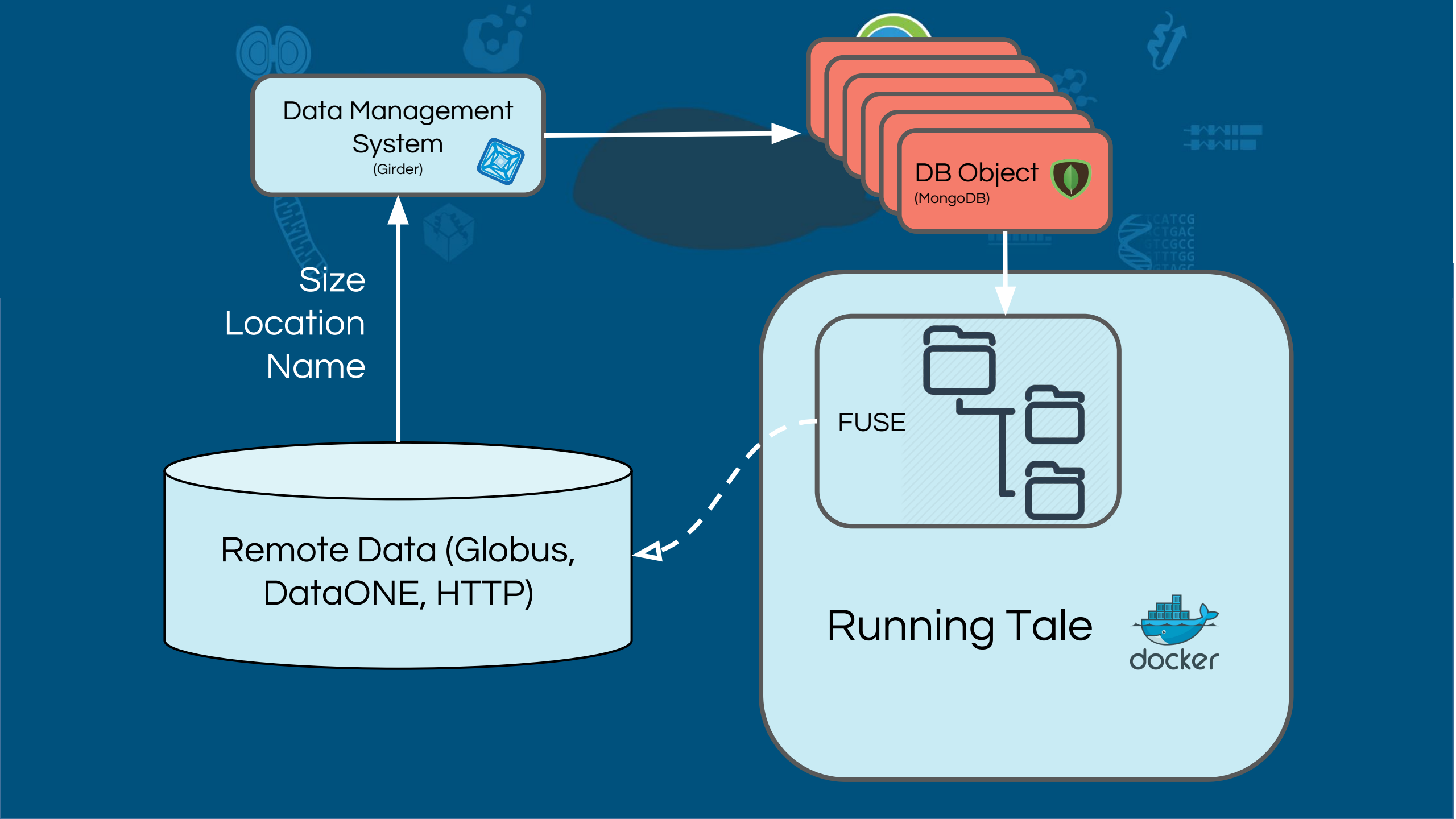

nonvolatile
(ro)
working (rw)
"Running" Tale
docker
Filesystems
$MP/data/
$MP/home/
e.g.
MP=/home/jovyan/work
Execution Environment
jupyter
rstudio
command line
etc


Data Management System
(Girder)
Size
Location
Name
Remote Data (Globus, DataONE, HTTP)
e.g. 10.5065/D6862DM8

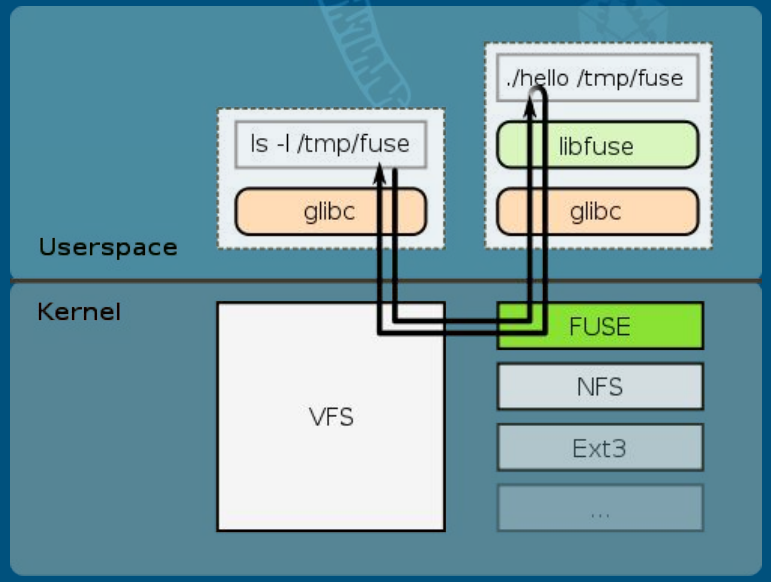

Data Management System
(Girder)
DB Object
(MongoDB)
Size
Location
Name
Remote Data (Globus, DataONE, HTTP)


Data Management System
(Girder)
DB Object
(MongoDB)
Size
Location
Name
Remote Data (Globus, DataONE, HTTP)


Data Management System
(Girder)
DB Object
(MongoDB)
Size
Location
Name
FUSE
Remote Data (Globus, DataONE, HTTP)


Data Management System
(Girder)
DB Object
(MongoDB)
Size
Location
Name
FUSE
Remote Data (Globus, DataONE, HTTP)
Running Tale
docker

What's FUSE?



This image was made by User:Sven „FUSE structure", CC-BY-SA
https://commons.wikimedia.org/wiki/File:FUSE_structure.svg

Filesystem in Userspace (gh:libfuse/libfuse)

Define typical i/o requests (read/open/link), mount, profit!

Packages worth mentioning:

fusepy - (gh:fusepy/fusepy)

PyFilesystem2 - (gh:PyFilesystem/pyfilesystem2)

```python
import requests, tempfile
from fuse import Operations, LoggingMixIn

class RESTGirderFS(LoggingMixIn, Operations):

    def read(self, path, size, offset, fh):
        obj = get object from metadata_server(path)   # translate path to object
        # Request data via HTTP
        req = requests.get(
            '%sitem/%s/download' % (self.girder cli.urlBase, obj[" id"]),
            headers={'Girder-Token': self.girder_cli.token}, stream=True)
        # Download data and store it locally
        with tempfile.NamedTemporaryFile(prefix='wtdm', delete=False) as tmp:
            for chunk in req.iter_content(chunk_size=65536):
                tmp.write(chunk)

        with open(tmp.name, 'rb') as fp:
            fp.seek(offset)
            return fp.read(size)  # open & read from local file
```

```python
import requests, tempfile
from fuse import Operations, LoggingMixIn

class RESTGirderFS(LoggingMixIn, Operations):

    def read(self, path, size, offset, fh):
        obj = get object from metadata_server(path)   # translate path to object
        # Request data via HTTP
        req = requests.get(
            '%sitem/%s/download' % (self.girder cli.urlBase, obj[" id"]),
            headers={'Girder-Token': self.girder_cli.token}, stream=True)
        # Download data and store it locally
        with tempfile.NamedTemporaryFile(prefix='wtdm', delete=False) as tmp:
            for chunk in req.iter_content(chunk_size=65536):
                tmp.write(chunk)

        with open(tmp.name, 'rb') as fp:
            fp.seek(offset)
            return fp.read(size)  # open & read from local file
```

AuthN

```python
import requests, tempfile
from fuse import Operations, LoggingMixIn

class RESTGirderFS(LoggingMixIn, Operations):

    def read(self, path, size, offset, fh):
        obj = get object from metadata server(path)   # translate path to object
        # Request data via HTTP and collect statistics
        req = requests.get(
            '%sitem/%s/download' % (self.girder cli.urlBase, obj[" id"]),
            headers={'Girder-Token': self.girder_cli.token}, stream=True)
        # Download data and store it locally
        with tempfile.NamedTemporaryFile(prefix='wtdm', delete=False) as tmp:
            for chunk in req.iter_content(chunk_size=65536):
                tmp.write(chunk)

        with open(tmp.name, 'rb') as fp:
            fp.seek(offset)
            return fp.read(size)  # open & read from local file
```

AuthN

```python
import requests, tempfile
from fuse import Operations, LoggingMixIn

class RESTGirderFS(LoggingMixIn, Operations):

    def read(self, path, size, offset, fh):
        obj = get object from metadata server(path)   # translate path to object
        # Request data via HTTP and collect statistics
        req = requests.get(
            '%sitem/%s/download' % (self.girder cli.urlBase, obj[" id"]),
            headers={'Girder-Token': self.girder_cli.token}, stream=True)
        # Download data and store it locally
        with tempfile.NamedTemporaryFile(prefix='wtdm', delete=False) as tmp:
            for chunk in req.iter_content(chunk_size=65536):
                tmp.write(chunk)
        # Track provenance
        with open(tmp.name, 'rb') as fp:
            fp.seek(offset)
            return fp.read(size)  # open & read from local file
```

AuthN

Outlook

Project page - http://wholetale.org/

Roadmap - https://wholetale.readthedocs.io/release/milestones.html

Prototype UI - https://dashboard.wholetale.org/

Beta version (coming this summer) - https://dashboard.stage.wholetale.org/

Code and development - https://github.com/whole-tale/

Thank you!

Why ‘reinvent’ the wheel^H^H Home?

- “In-house” solution based on WsgiDAV https://github.com/whole-tale/wt_home_dirs
- Integrated with Girder and supporting OAuth
- Reliability and performance: https://wholetale.readthedocs.io/development/design_notes/webdav.html

What does it give us?

- Unified workspace shared across our **web UI**, **running containers** and **your laptop.**