



**XKIT**

XROOTD · KUBERNETES INTEGRATION TESTING

**XKIT:**

**XRootD  
Kubernetes  
Integration  
Testing**

*Rob Currie, Wenlong Yuan*

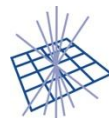
**CHEP 2024**

**21st October 2024**

# Motivations for XRootD Integration Testing

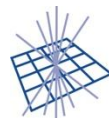
Rob's opinion: similar situation to when he worked on another grid project Ganga.

- *Unit-testing != Code Analysis != Integration Testing*
- 1. Large tool with large codebase & many uses.
- 2. Many communities using it to solve their problems.
- 3. Works extremely well.
- 4. Highly configurable with many plugins.
- 5. Not every community is running bleeding edge clients/versions.  
(some communities are better than others)
- Testing is difficult because the phase-space is so large.
  - > *3 large dimensions; client version, server version & network topology*
  - > *many compact dimensions, plugins options, server options, expected pass/fail*



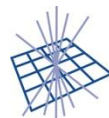
# Setting the Scene

- Larger UK grid sites use XRootD in different ways
- 5 large UK Tier 2 site configs, all similar, but none the same.
- Supporting different users, different release versions, different plugins combinations, ...
  - e.g: v4 client <-> v5 server using vector reads
  - vs: 3<sup>rd</sup> party copy v5 <-> dCache ...
- Question that has come up in testing:  
*“What was the ‘golden’ release/plugin version which worked for user X?”*

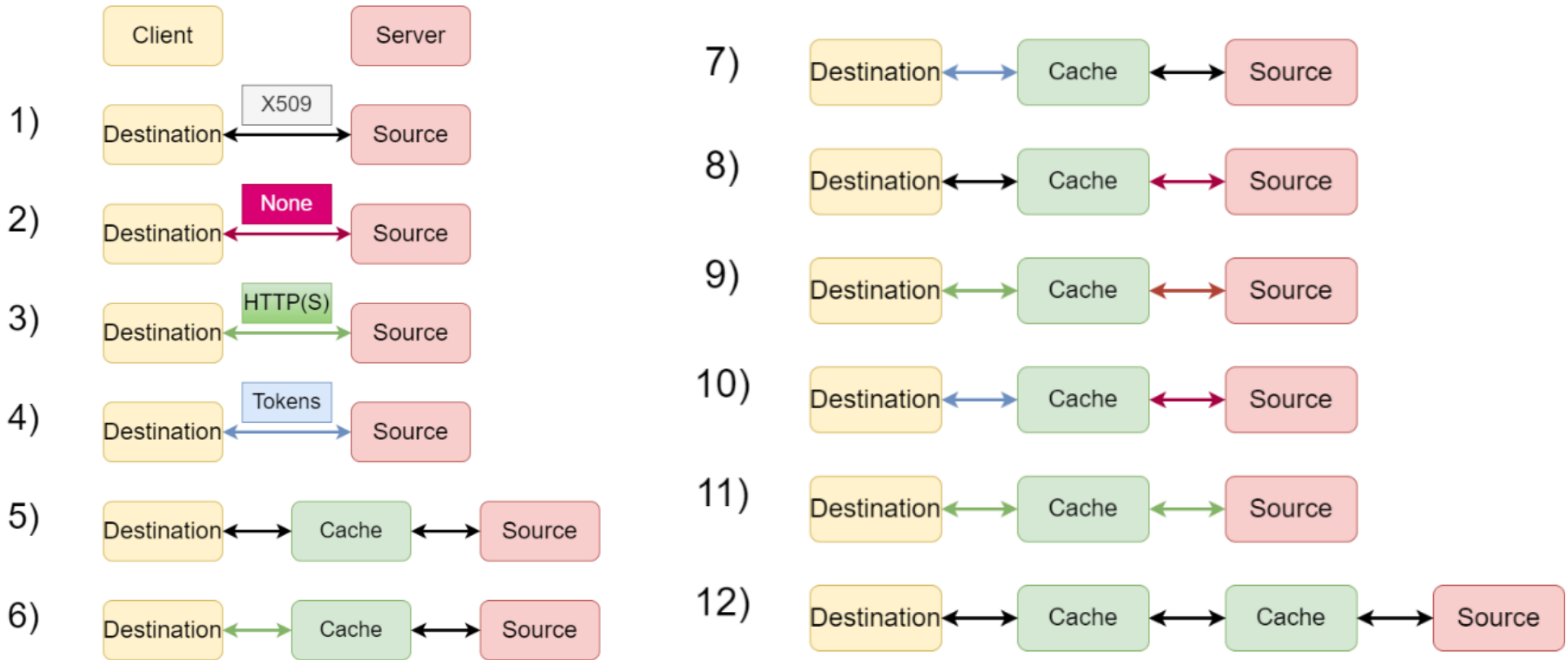


# UK Grid Software Deployments

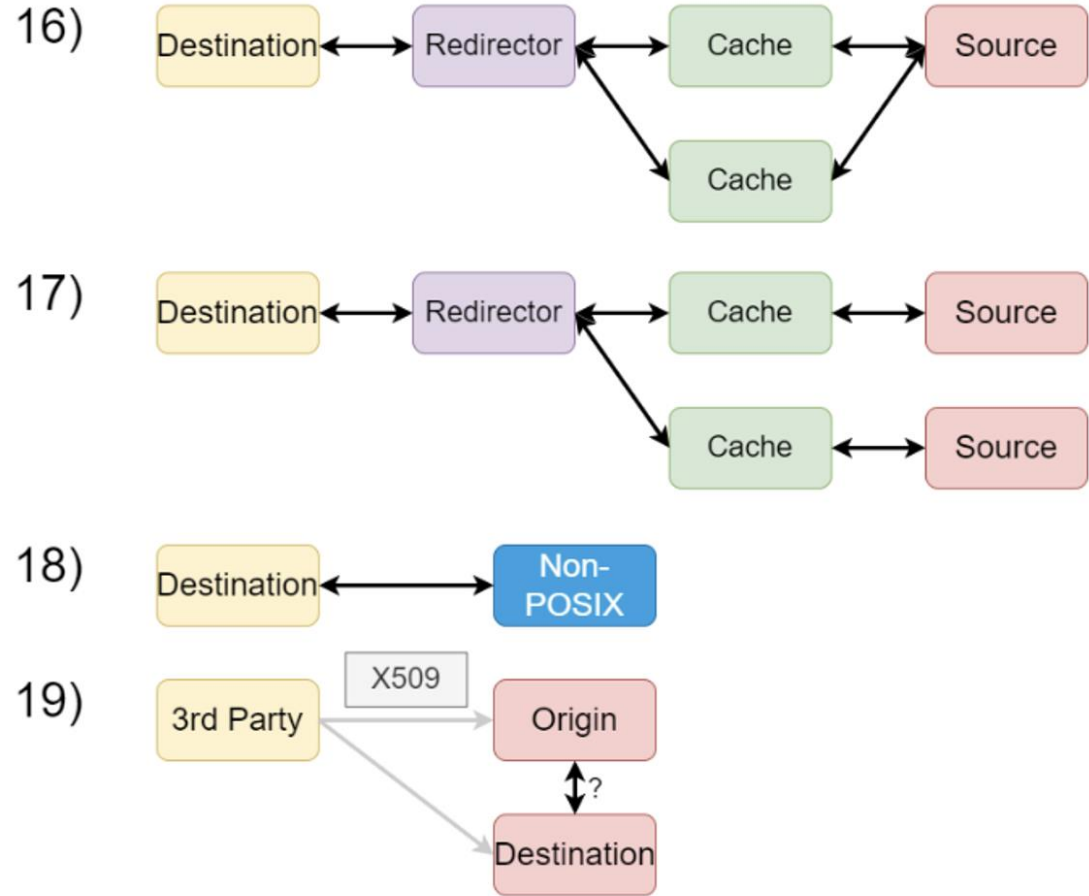
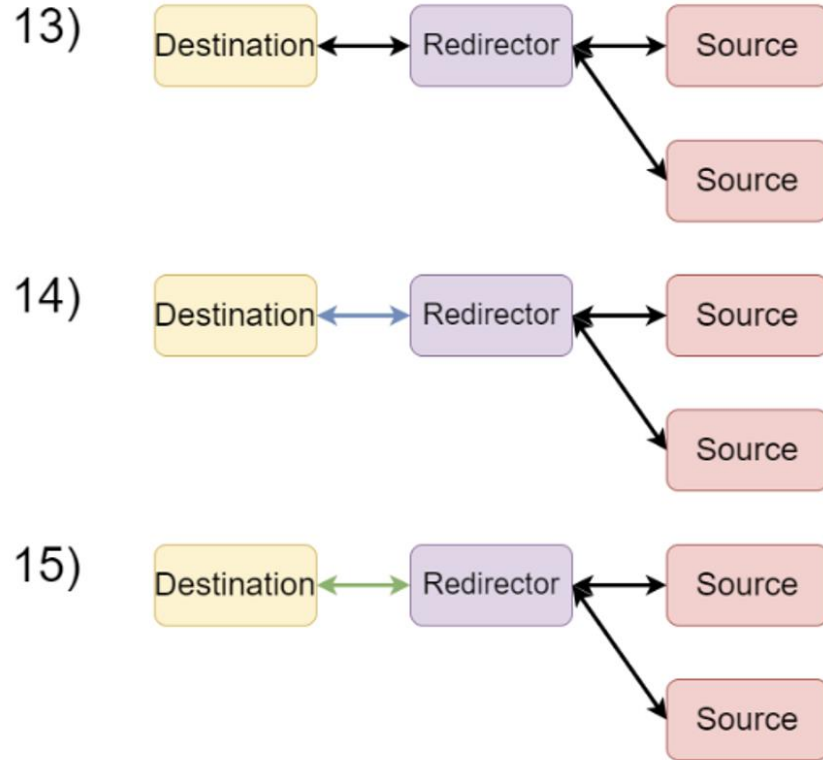
1. Grid site performs an install, does simple tests, possibly with a small test queue.
  2. Local & Remote VO experts check that everything is working as expected.
  3. Ideally, small problems then involve 3-4 people who may not be low-level experts.
  4. Issues impacting users tend to involve more people, take more effort ...
- Want to reduce person-power/effort needed to verify new packages for production configurations.
  - Virtual site deployments which 'look like' real-world sites reduces effort needed for 2.



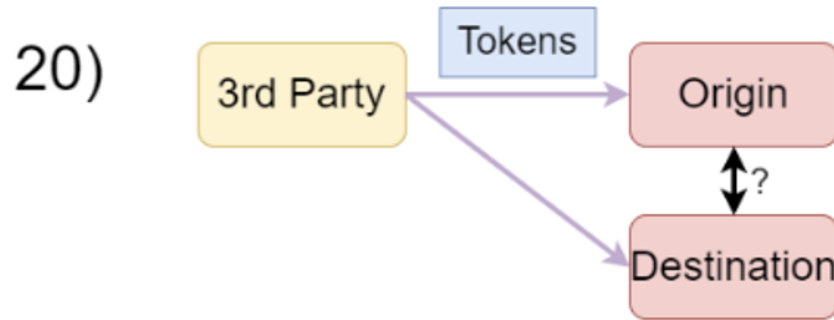
# How much do we want to test?



# How much do we want to test?

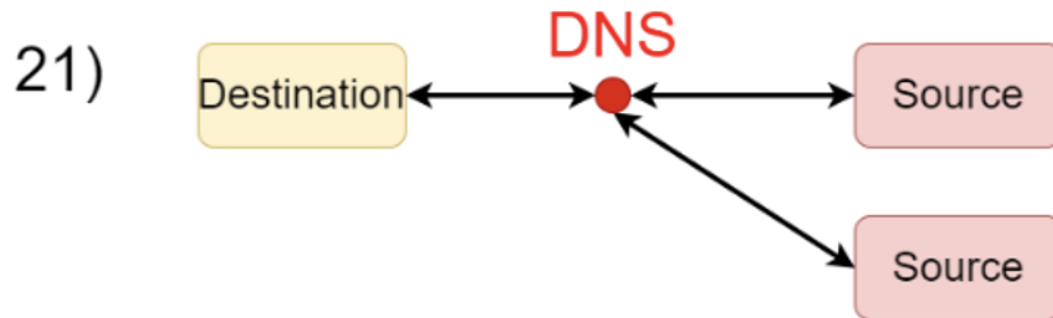


# How much do we want to test?



The topology of a “typical XRootD install” seems to vary even within UK.

Would be good to try and identify the key components of this.



Want to test/check/know-how-to-use all features and best practice(s).

# Test Management

- XRootD Integration Testing requires 2 parts:

## Client:

- Test cmdline tools (xrscp, xrdfs, ...)
- Test Python3 client API(s)
- Double-check everything works as expected
- Might aim test the C++ API (something closer to user-code in HEP)

## Server:

- Want to verify server behaviour (logs/output)
- Want to test read/write transfers work as expected
- Check server-side features configs haven't changed

***Containers to the Rescue!***





# XRootD Package/Image Management

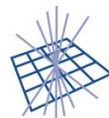
- XRootD is already used in Containers, but we want a minimal container for testing!

We are now 'rolling our own' container-images:

1. Using the **rpm** build recipe from the XRootD github repo (standing on the shoulders of giants!)
2. Built rpms from source on Alma9 base image(s)
3. Packages installed via dnf with all *normal* extensions for XRootD and dependencies
4. Image is tagged with release version
5. New images published to dockerhub
6. **No security/configuration/gremlins baked into images**

***IF someone else is doing a better job we can use their base images(!).***

*Deploying these containers means we have additional runtime control how we mount in CRL/config/data/cute-cuddly-kittens from our host into the container.*



# Service Management

- OK, now we have an image, so can launch containers/run-tests. 😊
- We started with **docker-compose** to manage multiple services.
- **This ended quickly.**
- Setting up a single transfer of:

**POSIX → PFC → Destination** DNS gets annoying 😞

- Docker/Podman(-compose) aren't friendly to mocking real world security setups.

***“Let's fix the problem of complex container management, with... more containers!”***



# Service Management (2)

- Each XRootD service needs the following:

- ✓ CRL/VOMS mounted/updated from host
- ✓ Server config mounted from host
- ✓ Test data mounted from host \*
- ✓ DNS entries pointing to instance
- ✓ Hostcert mounted from host (per-instance)
- ✓ External network connectivity



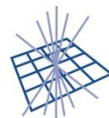
- After evaluating a few options, we decided to **go with Kubernetes**



# “There’s an API for that!”

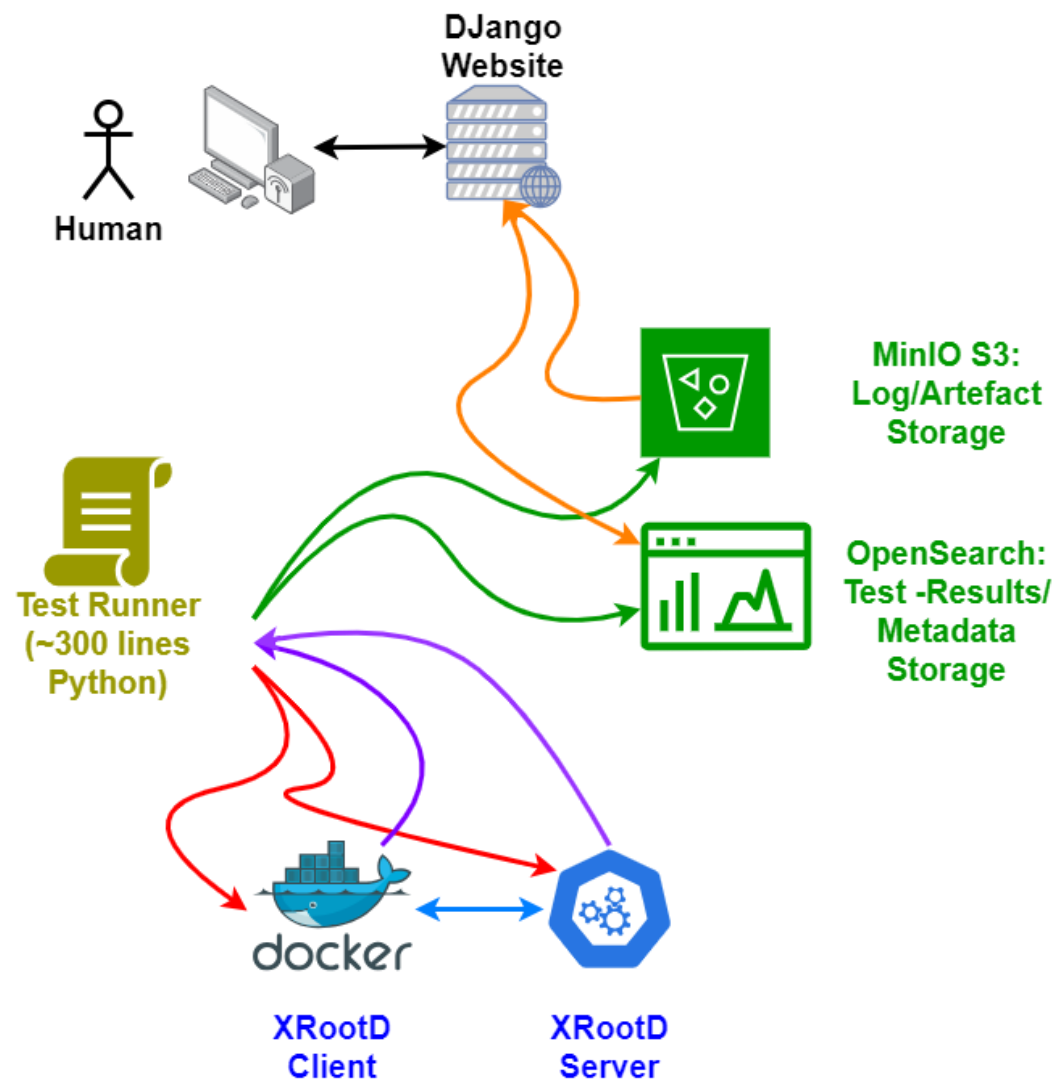
- Almost everything “speaks” Python3 these days.  
(The less we code, *the less we debug*, trying to keep things minimal)
- Kubernetes, Docker, S3, OpenSearch, Django, ...
- Most of the ‘*heavy lifting*’ for projects like this has been done for us.
- With that in mind, we decided to start working out what to do.
- Not *all work* is in Python3... but enough.

***It’s time for the running tests!***



# The Plan...

Testing Strategy
1) Deploy Configuration and Launch Containers
2) Wait for tests to run
3) Collect container artefacts
4) Store Test Results/Logs
4) Display Results to User



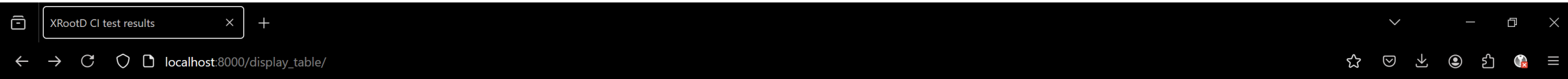
# What do we have so far?

(Not bad for <100 lines of Python!)

Containers on  
DockerHub

Test Client & Server  
logfiles on (*private!*) S3

Test Metadata,  
success/fail,  
timestamps, ...



## XRootD Test Results

client_image	server_image	client_output	server_output	testTime	testName	testStatus	@timestamp
<a href="#">gridppedi/xrdtesting:xrd-v5.7.0</a>	<a href="#">gridppedi/xrdtesting:xrd-v5.7.0</a>	<a href="#">pythonTestOutputs/read.py_C_xrd-v5.7.0_S_xrd-5.6.2_clientOutput.log</a>	<a href="#">pythonTestOutputs/read.py_C_xrd-v5.7.0_S_xrd-5.6.2_serverOutput.log</a>	2024-07-22T15:40:36.595529	read.py	GOOD	2024-07-22T15:40:36.595820
<a href="#">gridppedi/xrdtesting:xrd-v5.7.0</a>	<a href="#">gridppedi/xrdtesting:xrd-v5.7.0</a>	<a href="#">pythonTestOutputs/read.py_C_xrd-v5.7.0_S_xrd-5.6.2_clientOutput.log</a>	<a href="#">pythonTestOutputs/read.py_C_xrd-v5.7.0_S_xrd-5.6.2_serverOutput.log</a>	2024-07-22T15:31:27.728273	read.py	GOOD	2024-07-22T15:31:27.728584
<a href="#">gridppedi/xrdtesting:xrd-v5.7.0</a>	<a href="#">gridppedi/xrdtesting:xrd-v5.7.0</a>	<a href="#">pythonTestOutputs/read.py_C_xrd-v5.7.0_S_xrd-5.6.2_clientOutput.log</a>	<a href="#">pythonTestOutputs/read.py_C_xrd-v5.7.0_S_xrd-5.6.2_serverOutput.log</a>	2024-07-22T15:32:48.912504	read.py	GOOD	2024-07-22T15:32:48.912898
<a href="#">gridppedi/xrdtesting:xrd-v5.7.0</a>	<a href="#">gridppedi/xrdtesting:xrd-v5.7.0</a>	<a href="#">pythonTestOutputs/read.py_C_xrd-v5.7.0_S_xrd-5.6.2_clientOutput.log</a>	<a href="#">pythonTestOutputs/read.py_C_xrd-v5.7.0_S_xrd-5.6.2_serverOutput.log</a>	2024-07-22T15:35:25.426305	read.py	GOOD	2024-07-22T15:35:25.426606
<a href="#">gridppedi/xrdtesting:xrd-v5.7.0</a>	<a href="#">gridppedi/xrdtesting:xrd-v5.7.0</a>	<a href="#">pythonTestOutputs/read.py_C_xrd-v5.7.0_S_xrd-5.6.2_clientOutput.log</a>	<a href="#">pythonTestOutputs/read.py_C_xrd-v5.7.0_S_xrd-5.6.2_serverOutput.log</a>	2024-07-22T15:40:20.524026	read.py	GOOD	2024-07-22T15:40:20.524350
<a href="#">gridppedi/xrdtesting:xrd-v5.7.0</a>	<a href="#">gridppedi/xrdtesting:xrd-v5.7.0</a>	<a href="#">pythonTestOutputs/read.py_C_xrd-v5.7.0_S_xrd-5.6.2_clientOutput.log</a>	<a href="#">pythonTestOutputs/read.py_C_xrd-v5.7.0_S_xrd-5.6.2_serverOutput.log</a>	2024-07-22T15:46:49.528588	read.py	GOOD	2024-07-22T15:46:49.528985
<a href="#">gridppedi/xrdtesting:xrd-v5.7.0</a>	<a href="#">gridppedi/xrdtesting:xrd-v5.7.0</a>	<a href="#">pythonTestOutputs/read.py_C_xrd-v5.7.0_S_xrd-5.6.2_clientOutput.log</a>	<a href="#">pythonTestOutputs/read.py_C_xrd-v5.7.0_S_xrd-5.6.2_serverOutput.log</a>	2024-07-22T15:51:34.707189	read.py	BAD	2024-07-22T15:51:34.707640

# What do we have so far?

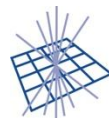
- Simple, *entirely dynamically generated* web-UI.  
Not *yet* public, plan to '*hide*' host behind an OAuth login.
- Using a github organization for managing the various pieces of this:  
<https://github.com/gridpp-Edi>
- **Tests repo:**  
<https://github.com/gridpp-Edi/xrootd-ci-tests>
- **Server configs repo:**  
<https://github.com/gridpp-Edi/xrootd-helm-charts>

*We aim to publish and share all ASAP.*



# From the Site's Perspective

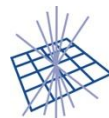
- On the face of it, this has *lots* of moving parts:
  - DNS, VOMS, Kubernetes, multiple new systems to update/maintain, s3, OpenSearch/ElasticSearch, message queues, credentials...
- However, these services are being re-used by some other project.
- Work on this allows us to:
  - Support the in-development protoDUNE DAQ offline monitoring
  - Support DUNE-DM monitoring
  - Support GridPP-FTS monitoring
  - Support UoE PPE-Labs clean-room certification
  - Gain valuable experience with Kubernetes
  - Support GridPP storage efforts





# Conclusions

- Successfully run initial tests against XRootD using our ‘pipeline’.
  - Data transfers in/out of ‘Virtual site’ using containers.
- Have worked out most of the annoying bits in setting this up.
- Have a minimal web-UI which we aim to share ASAP.



# Conclusions – Next Steps

- Need to expand our testing topology (helm charts).
  - So far have server-side configs for simple XRD-POSIX and XRD-PFC.
  - Only testing X509 auth but want to do more.
- Need to flesh out some additional tests.
  - Successfully written/read data from POSIX via different API.
  - Want to automatically test 3<sup>rd</sup>-party copy between endpoints, internal and external.
- Plan to integrate with higher-level testing system for tracking different client/server tests and outputs.

