XKIT:

XRootD Kubernetes Integration Testing

Rob Currie, Wenlong Yuan

University of Edinburgh

THE UNIVERSITY of EDINBURGH

Intro

- Slightly packed set of slides.

- Unfortunately, must leave early today.

  Very happy to discuss any of this further by email:
  rob.currie<at>ed.ac.uk

- We're looking for some input/wisdom from the experts to improve this ☺

  Please tell me if/when I'm wrong.

Site History – Edinburgh Tier2 Version

THE UNIVERSITY
of EDINBURGH

- ??? - 2018: Ran DPM at our site which eventually migrated to dCache.
- 2021 – 2022: Gained experience(s) testing XRootD-PFC internally for (ATLAS) grid workflows.
- 2022 – 2023: DDOS'ed our own storage several times due to mis-configuration.
- 2022 – 2023: Had a student work to optimize our cache performance.
- 2023: Turned off our dCache ahead of CO7 EOL.
- 2024+ : Contributing to wider UK storage efforts which mostly use XRootD.

Setting the Scene

- Larger UK grid sites use XRootD in different ways. (See Alastair's Talk)
- 5 large Tier2 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: 3rd 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.**

Motivations for XRootD Integration Testing

<u>My opinion:</u> similar situation to when I worked on another grid project Ganga.

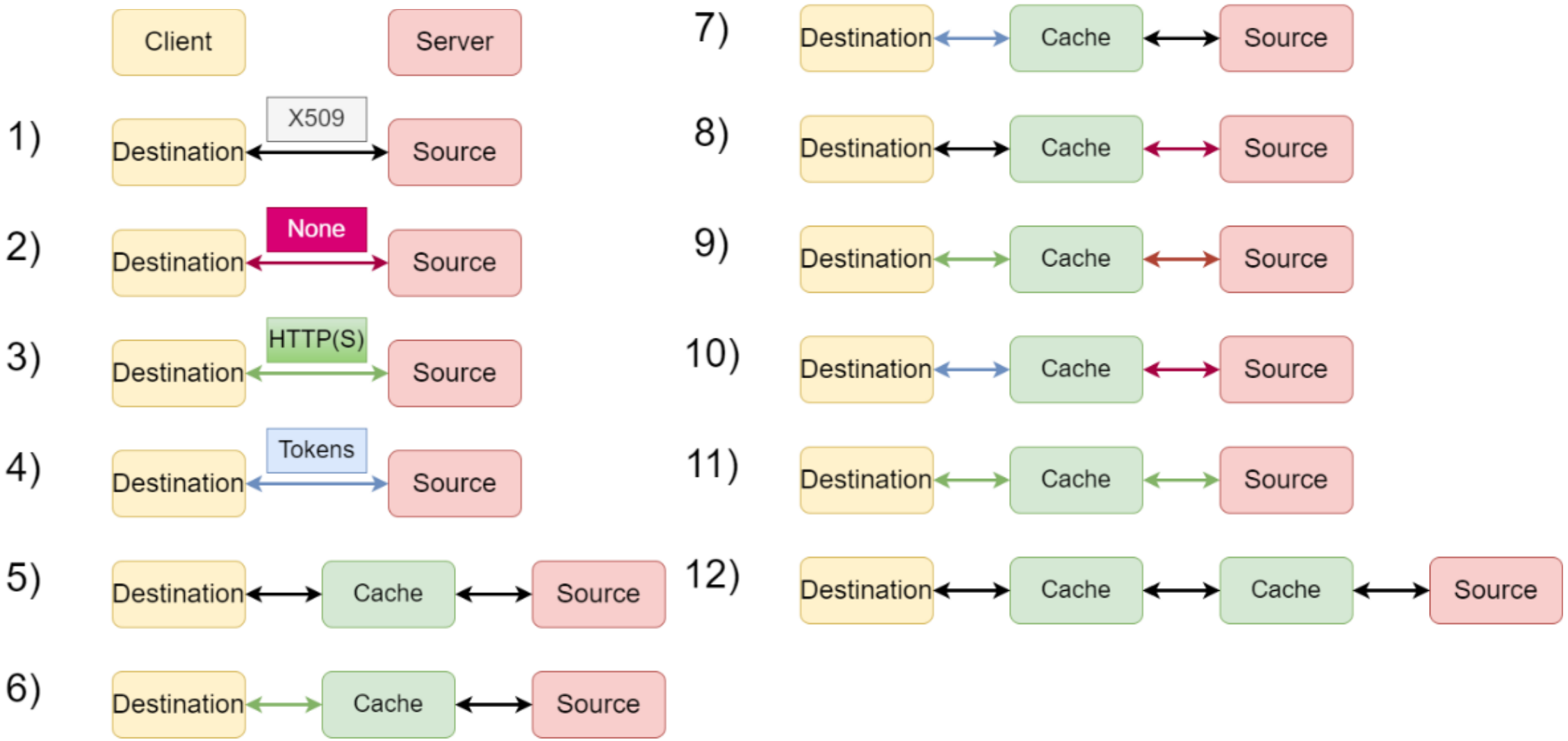
- **Unit-testing != Code Analysis != Integration Testing**

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.

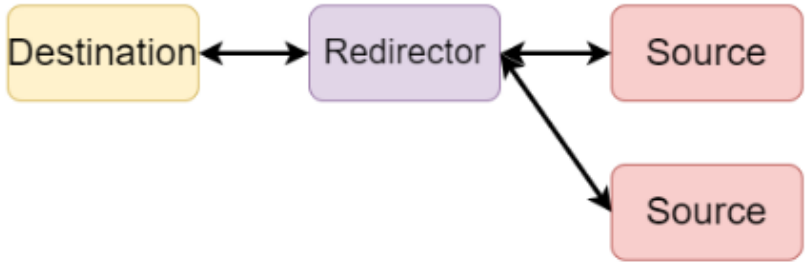
> 3 large dimensions; client version, server version & network topology
> many compact dimensions, plugins options, server options, expected pass/fail

How much do we want to test?

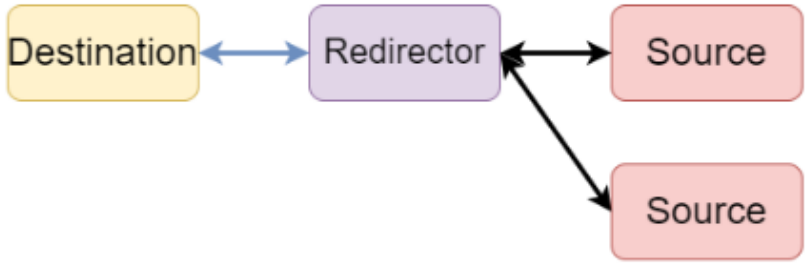


How much do we want to test?

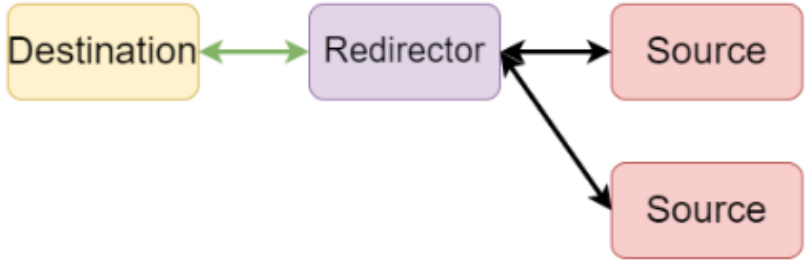
13)



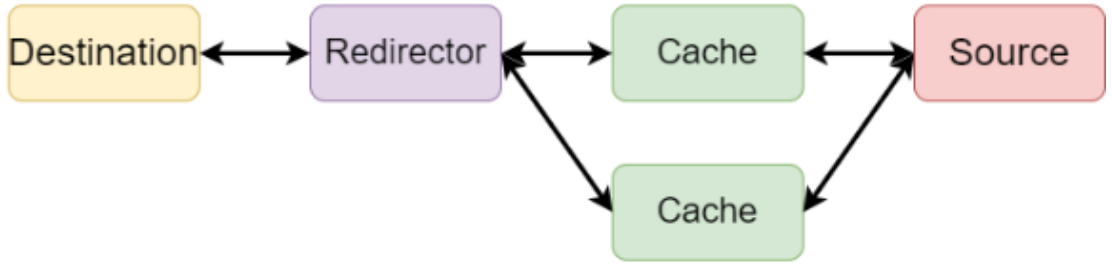
14)



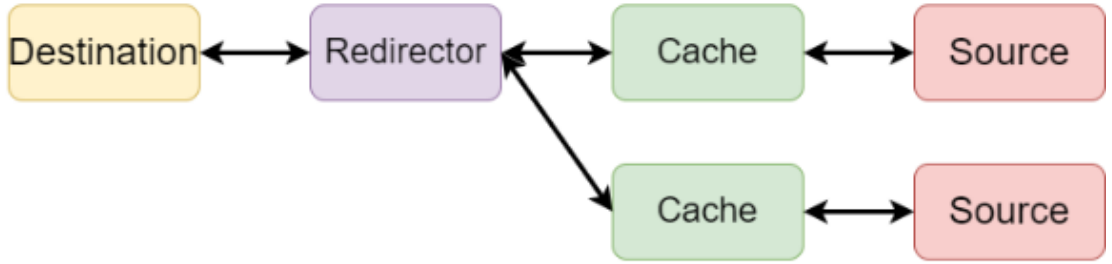
15)



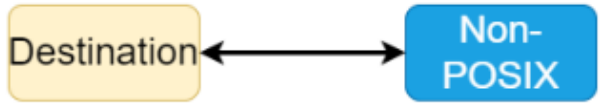
16)



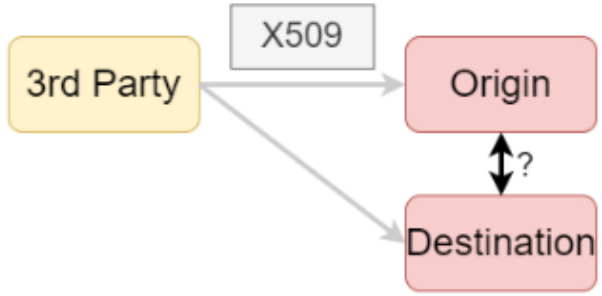
17)



18)

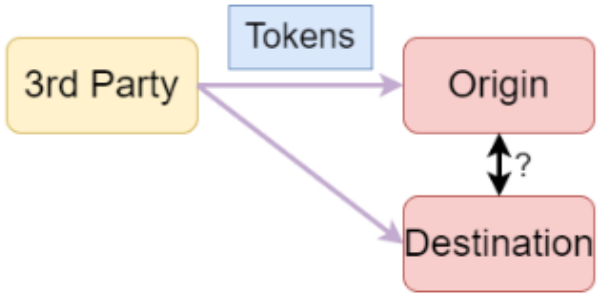


19)

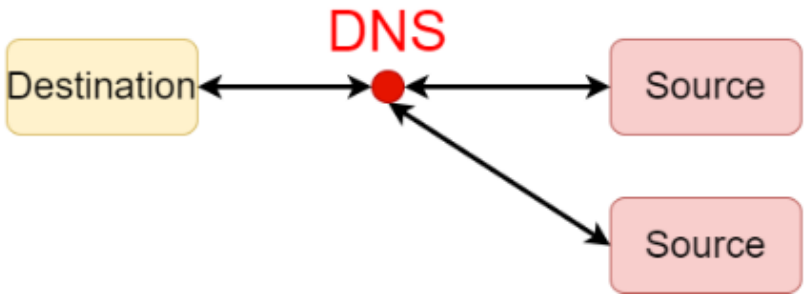


How much do we want to test?

20)



21)



The topology of a “typical XRootD install” seems to vary 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 (xrdcp, 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 is already used in Containers

- **But we want a minimal container for testing!**
- Container design often ends up optimizing for 1 of 2 goals:

- **Deployability**: *

Container design used by **perfSonar**, **Gitlab…**
Deploying several services within a single container.
Not-so-great for seeing what's going on, debugging, or fixing/testing…

- **Reproducibility**:

This is what you see in more commercially supported containers.
Closer to the UNIX philosophy of **<u>"do one thing and do it well"</u>**
Minimal, **great for testing**.

* Yes, I just made up a word…

XRootD Package/Image Management

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

- Why is this important? Containers are backed by images;

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**

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

~~Container Orchestration~~

- OK, now we have an image, so can launch containers/run-tests.

- We **started with docker-compose** to manage multiple services.
- **This ended quickly**.

- When setting up a single transfer of:

  POSIX → PFC → Destination DNS gets annoying

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

This is a shame; we're making use of docker-compose a LOT at our site.

Service Management

Let’s fix the problem of complex container management, with… more containers!

Container Management

- Each XRootD instance needs:

  → CRL/VOMS mounted/updated from host
  → Config mounted from host
  → Test data mounted from host *
  → DNS entries pointing to instance
  → Hostcert mounted from host (per-instance)
  → External network connectivity

- Just like a thousand people before us; **go with Kubernetes**

*So far, biggest use-case is POSIX, but plan to test CEPH-FS

“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.

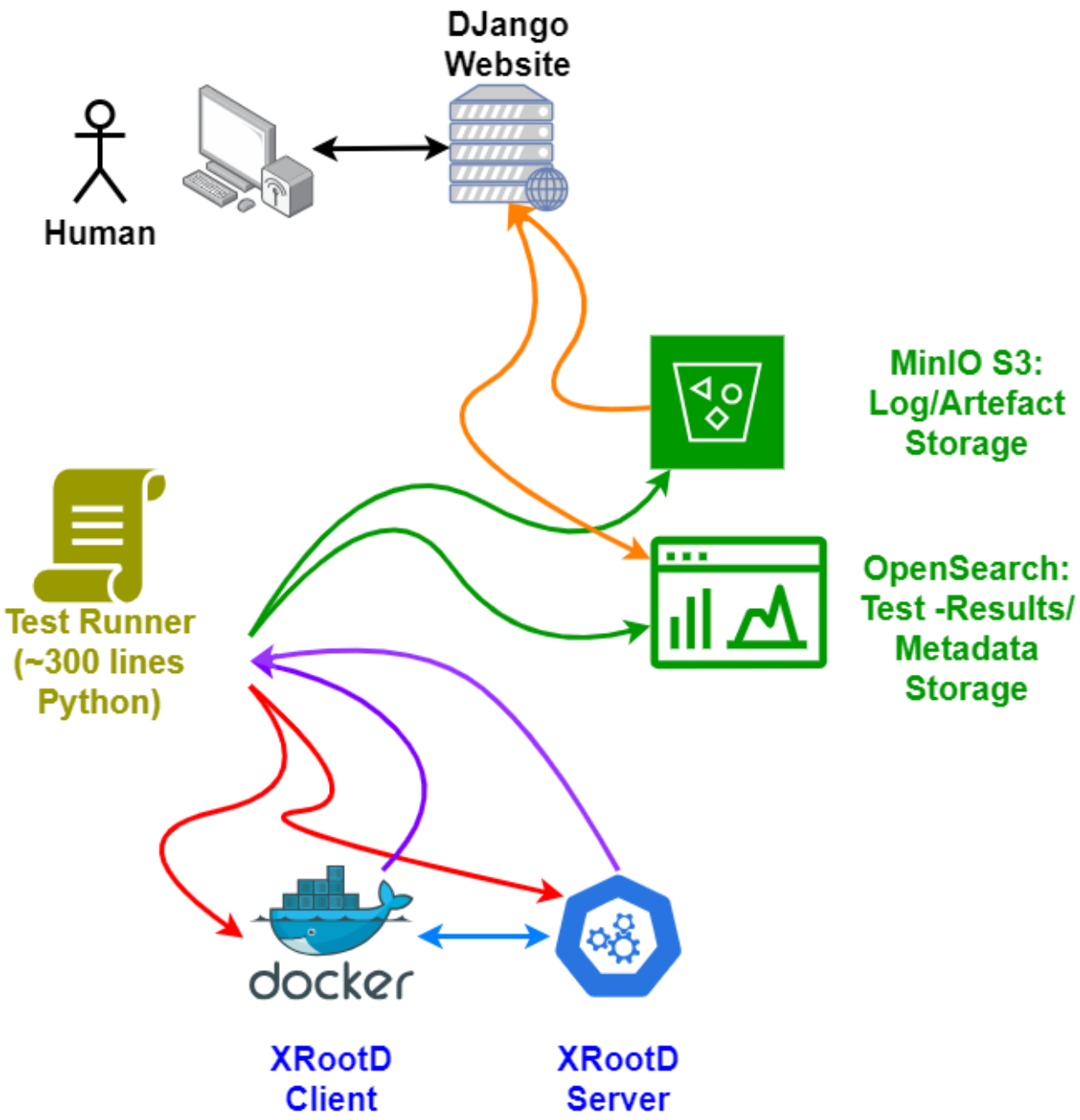
- With that in mind, we decided to start working out what to do.

- Not *all work* is in Python3... but most of it.

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?

XRootD CI test results

localhost:8000/display_table/

XRootD Test Results

client_image	server_image	client_output	server_output	testTime	testName	testStatus	@timestamp
gridppedi/xrdtesting:xrd-v5.7.0	gridppedi/xrdtesting:xrd-v5.7.0	pythonTestOutputs/read.py_C_xrd-v5.7.0_S_xrd-5.6.2_clientOutput.log	pythonTestOutputs/read.py_C_xrd-v5.7.0_S_xrd-5.6.2_serverOutput.log	2024-07-22T15:40:36.595529	read.py	GOOD	2024-07-22T15:40:36.595820
gridppedi/xrdtesting:xrd-v5.7.0	gridppedi/xrdtesting:xrd-v5.7.0	pythonTestOutputs/read.py_C_xrd-v5.7.0_S_xrd-5.6.2_clientOutput.log	pythonTestOutputs/read.py_C_xrd-v5.7.0_S_xrd-5.6.2_serverOutput.log	2024-07-22T15:31:27.728273	read.py	GOOD	2024-07-22T15:31:27.728584
gridppedi/xrdtesting:xrd-v5.7.0	gridppedi/xrdtesting:xrd-v5.7.0	pythonTestOutputs/read.py_C_xrd-v5.7.0_S_xrd-5.6.2_clientOutput.log	pythonTestOutputs/read.py_C_xrd-v5.7.0_S_xrd-5.6.2_serverOutput.log	2024-07-22T15:32:48.912504	read.py	GOOD	2024-07-22T15:32:48.912898
gridppedi/xrdtesting:xrd-v5.7.0	gridppedi/xrdtesting:xrd-v5.7.0	pythonTestOutputs/read.py_C_xrd-v5.7.0_S_xrd-5.6.2_clientOutput.log	pythonTestOutputs/read.py_C_xrd-v5.7.0_S_xrd-5.6.2_serverOutput.log	2024-07-22T15:35:25.426305	read.py	GOOD	2024-07-22T15:35:25.426606
gridppedi/xrdtesting:xrd-v5.7.0	gridppedi/xrdtesting:xrd-v5.7.0	pythonTestOutputs/read.py_C_xrd-v5.7.0_S_xrd-5.6.2_clientOutput.log	pythonTestOutputs/read.py_C_xrd-v5.7.0_S_xrd-5.6.2_serverOutput.log	2024-07-22T15:40:20.524026	read.py	GOOD	2024-07-22T15:40:20.524350
gridppedi/xrdtesting:xrd-v5.7.0	gridppedi/xrdtesting:xrd-v5.7.0	pythonTestOutputs/read.py_C_xrd-v5.7.0_S_xrd-5.6.2_clientOutput.log	pythonTestOutputs/read.py_C_xrd-v5.7.0_S_xrd-5.6.2_serverOutput.log	2024-07-22T15:46:49.528588	read.py	GOOD	2024-07-22T15:46:49.528985
gridppedi/xrdtesting:xrd-v5.7.0	gridppedi/xrdtesting:xrd-v5.7.0	pythonTestOutputs/read.py_C_xrd-v5.7.0_S_xrd-5.6.2_clientOutput.log	pythonTestOutputs/read.py_C_xrd-v5.7.0_S_xrd-5.6.2_serverOutput.log	2024-07-22T15:51:34.707189	read.py	BAD	2024-07-22T15:51:34.707649

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 CI test results

localhost:8000/display_table/

XRootD Test Results

client_image	server_image	client_output	server_output	testTime	testName	testStatus	@timestamp
gridppedi/xrdtesting:xrd-v5.7.0	gridppedi/xrdtesting:xrd-v5.7.0	pythonTestOutputs/read.py_C_xrd-v5.7.0_S_xrd-5.6.2_clientOutput.log	pythonTestOutputs/read.py_C_xrd-v5.7.0_S_xrd-5.6.2_serverOutput.log	2024-07-22T15:40:36.595529	read.py	GOOD	2024-07-22T15:40:36.595820
gridppedi/xrdtesting:xrd-v5.7.0	gridppedi/xrdtesting:xrd-v5.7.0	pythonTestOutputs/read.py_C_xrd-v5.7.0_S_xrd-5.6.2_clientOutput.log	pythonTestOutputs/read.py_C_xrd-v5.7.0_S_xrd-5.6.2_serverOutput.log	2024-07-22T15:31:27.728273	read.py	GOOD	2024-07-22T15:31:27.728584
gridppedi/xrdtesting:xrd-v5.7.0	gridppedi/xrdtesting:xrd-v5.7.0	pythonTestOutputs/read.py_C_xrd-v5.7.0_S_xrd-5.6.2_clientOutput.log	pythonTestOutputs/read.py_C_xrd-v5.7.0_S_xrd-5.6.2_serverOutput.log	2024-07-22T15:32:48.912504	read.py	GOOD	2024-07-22T15:32:48.912898
gridppedi/xrdtesting:xrd-v5.7.0	gridppedi/xrdtesting:xrd-v5.7.0	pythonTestOutputs/read.py_C_xrd-v5.7.0_S_xrd-5.6.2_clientOutput.log	pythonTestOutputs/read.py_C_xrd-v5.7.0_S_xrd-5.6.2_serverOutput.log	2024-07-22T15:35:25.426305	read.py	GOOD	2024-07-22T15:35:25.426606
gridppedi/xrdtesting:xrd-v5.7.0	gridppedi/xrdtesting:xrd-v5.7.0	pythonTestOutputs/read.py_C_xrd-v5.7.0_S_xrd-5.6.2_clientOutput.log	pythonTestOutputs/read.py_C_xrd-v5.7.0_S_xrd-5.6.2_serverOutput.log	2024-07-22T15:40:20.524026	read.py	GOOD	2024-07-22T15:40:20.524350
gridppedi/xrdtesting:xrd-v5.7.0	gridppedi/xrdtesting:xrd-v5.7.0	pythonTestOutputs/read.py_C_xrd-v5.7.0_S_xrd-5.6.2_clientOutput.log	pythonTestOutputs/read.py_C_xrd-v5.7.0_S_xrd-5.6.2_serverOutput.log	2024-07-22T15:46:49.528588	read.py	GOOD	2024-07-22T15:46:49.528985
gridppedi/xrdtesting:xrd-v5.7.0	gridppedi/xrdtesting:xrd-v5.7.0	pythonTestOutputs/read.py_C_xrd-v5.7.0_S_xrd-5.6.2_clientOutput.log	pythonTestOutputs/read.py_C_xrd-v5.7.0_S_xrd-5.6.2_serverOutput.log	2024-07-22T15:51:34.707189	read.py	BAD	2024-07-22T15:51:34.70764

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

(Still empty as of ~~August~~ September 2024 ☹ aiming for uploading tests before CHEP)

Just starting to populate this repo for testing ☺

Tier2 Site Perspective

From the Site's Perspective

- This has ***lots*** of moving parts (*and only 2 fractional admins*):

  DNS, VOMS, Kubernetes, multiple new systems to update/maintain, S3, OpenSearch/ElasticSearch, message queues, credentials...

- Slight concern:

  Did we just replace effort required for one piece of work with more effort required for somewhere else?...

- All these services are being re-used by some other project.

  Not just throwing up lots of services for a single goal.

From the Site’s Perspective

- Work on this allows us to:

1. Support the in-development protoDUNE DAQ offline monitoring
2. Support DUNE-DM monitoring
3. Support GridPP-FTS monitoring
4. Support UoE PPE-Labs clean-room certification
5. Gain valuable experience with Kubernetes
6. Supporting GridPP UK Storage efforts

Conclusions

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 3rd-party copy between endpoints, internal and external.
- Plan to integrate with higher-level testing system for tracking different client/server tests and outputs.


THANK YOU FOR
LISTENING