Using new Open OnDemand 4.1 capabilities to enhance connection security for interactive applications on HPC systems

Open OnDemand Tips and Tricks

Stefano Ansaloni

University of Manitoba

April 2, 2026

University of Manitoba

Digital Research Alliance of Canada

Stefano Ansaloni

Cloud Computing Specialist at University of Manitoba
(part of the Advanced Research Computing team)

Software Developer and DevOps Specialist since 2017

Linux User/Admin since 2005

University of Manitoba

UManitoba HPC cluster: Grex

Hardware

Grex is a HPC system at University of Manitoba:

- First put in production in 2011
- Initial capacity:
  - 312 SGI Altix compute nodes
    (12 cores and 48GB memory per node – 3.6K cores and 14TB memory total)
  - QDR 40Gb/s InfiniBand network
  - 418TB Lustre filesystem (added in 2017)
- Renovated in 2024 (old nodes decommissioned)
- Current capacity:
  - 110 compute nodes (mixed layout – 10K cores, 50 GPUs and 55TB memory total)
  - EDR 100Gb/s and HDR 200Gb/s InfiniBand network
  - 2PB Lustre filesystem

Grex Wiki [Grex]

University of Manitoba

UManitoba HPC cluster: Grex

Software

Operating system:	AlmaLinux 8.10
Scheduler:	Slurm 25.05.5
Module system:	Lmod 8.7.32
Storage:	Lustre 2.14.x (appliance by DDN)

University of Manitoba

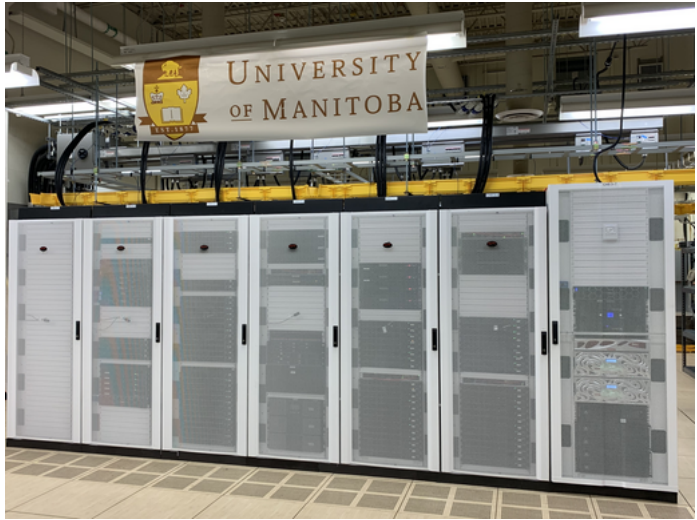
UManitoba HPC cluster: Grex

Configuration

- Slurm scheduling is “per-core”: different jobs from different users can run on the same node at the same time
- Network is “almost” private: only login nodes have a public IP and an external DNS entry (compute nodes name cannot be resolved from outside the cluster)

University of Manitoba

UManitoba HPC cluster: Grex



University of Manitoba

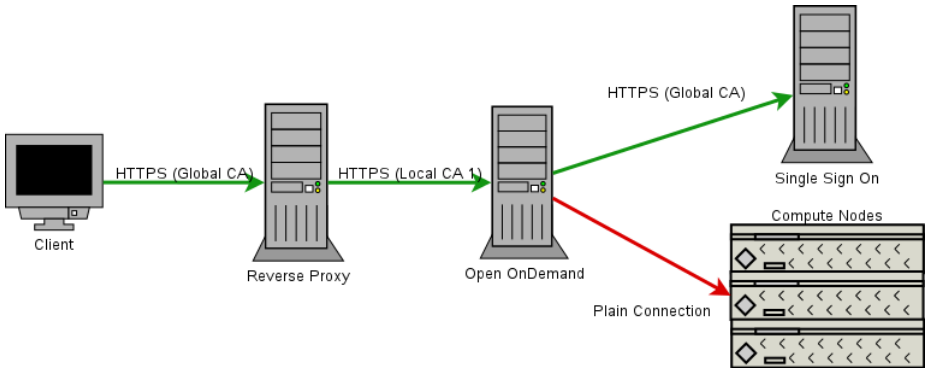
Open OnDemand on Grex

History

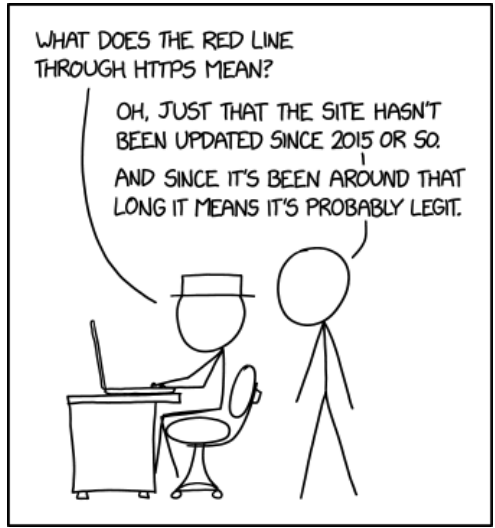
- Installed OOD version 1.8 around 2020
  (probably the first installation on a canadian academic system)
- Upgraded to version 2.x in 2021
- Upgraded to version 3.x in 2023
  (configured SSO with MFA – probably the first on a canadian academic system)
- Upgraded to version 4.0 in 2025
- Upgraded to version 4.1 in 2026
  (enabled TLS proxying – probably the first on a canadian academic system)

Open OnDemand on Grex

Previous configuration (before 4.1)



University of Manitoba



XKCD [2634]

University of Manitoba

Enabling Open OnDemand TLS proxying

Motivations

Obvious reasons:

- Increased security for users (especially on shared systems)
- Better system security posture
- "Encryption in flight" is a requirement when working with restricted/protected data (we currently do not allow such data)

Non-obvious reasons:

- N/A

University of Manitoba

Enabling Open OnDemand TLS proxying

Steps overview

1. Install/Update OOD version 4.1 or later

2. Set "`secure_node_uri`" and "`secure_rnode_uri`" in `ood_portal.yml`

3. Update the desired server applications:
   - 3.1 The application must be started with TLS
   - 3.2 Use either "`secure_node_uri`" or "`secure_rnode_uri`" in `view.html.erb` (depending on the application)

University of Manitoba

Enabling Open OnDemand TLS proxying

Step 1 – Install OOD 4.1

Installing or upgrading Open OnDemand to version 4.1 is straightforward in most cases thanks to the wonderful, well-maintained official documentation.

https://osc.github.io/ood-documentation/latest

Enabling Open OnDemand TLS proxying

Step 2 – Configure required parameters

The options that must be set in `ood_portal.yml` are:

- `secure_node_uri` (for server applications that know the *full* URI path)
- `secure_rnode_uri` (for server applications that know the *relative* URI path)

The suggestion is to set them to something meaningful, for example "`/secure-node`" and "`/secure-rnode`".

Do not forget to regenerate the portal configuration through the "`update_ood_portal`" utility.

Enabling Open OnDemand TLS proxying

Step 3 – Update server applications

Configured server applications must be started with TLS enabled (each software is unique, please refer to the specific application documentation).

Server applications also need to use the new "`secure_(r)node_uri`" paths inside their `view.html.erb`.

Problems

Some problems start to arise:

1. Not all server applications support TLS (e.g. CryoSPARC)
2. Some server applications support TLS only in their commercial version (e.g. RStudio Server)
3. When started with TLS, some server applications confuse Open OnDemand readiness check (e.g. Code Server)

Issue 2. can be "solved" by purchasing the commercial version of the application.

Issue 3. needs a change in the Open OnDemand source code.

Bigger problems

Something *important* is missing: the TLS certificate used by the server applications.

Possible solutions:

- obtain a certificate from a global CA (e.g. LetsEncrypt)
- generate a certificate from a local CA

Those solutions do not provide "confidentiality": having a single TLS certificate means sharing the private key with all the users – NOT ACCEPTABLE.

TLS certificates

Desired properties

The TLS certificate needs the following properties:

- unique per user
- specific to the nodes that are running the application
- *not* valid for the node default FQDN
- *not* valid for the node IP
- short-lived

Using a global CA could present limitations (e.g. rate limit, invalid FQDN).

A local CA is probably better (at least to have things started).

TLS certificates

Automation

Manually generating each TLS certificate is (almost) impossible.

Some kind of automation will be required to generate the TLS certificates on-the-fly with the desired properties every time a user launch a server application.

Luckily, Slurm provides that needed automation through its `prolog` and `epilog` features.

Slurm [Prolog and Epilog Guide]

University of Manitoba

TLS certificates

Proposed solution

Proposed solution for automatic TLS certificates creation:

- configure wildcard DNS CNAMEs for compute nodes in your DNS server
- generate a local CA
- use Slurm prolog to
  - get the username running the job
  - get the list of compute nodes running the job
  - generate a TLS certificate valid for "`username.node_fqdn`"
  - save the TLS certificate in a known, private location
- use Slurm epilog to delete the user TLS certificate

University of Manitoba

TLS certificates

Open OnDemand configuration

The file "`ood_portal.yml`" needs to be updated accordingly:

- `host_regex` must match "`username.node_fqdn`"
- `ssl_proxy` must allow TLS certificates signed by the internal CA

As always, do not forget to regenerate the portal configuration through the "`update_ood_portal`" utility.

OOD Documentation [Configuration Reference]

University of Manitoba

Something is missing

Desktop applications

We enabled TLS proxying only for server application (which is a great thing).

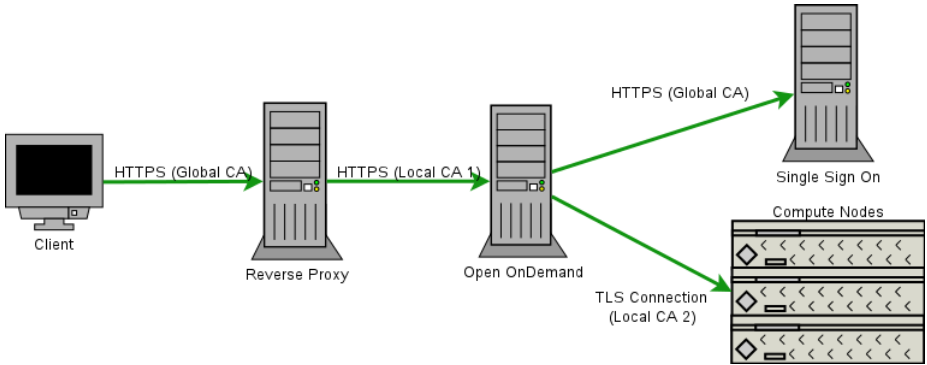
What about desktops/desktop applications?

Open OnDemand does not support it at this moment, but we tried to enable it anyway:

- modified Open OnDemand Dashboard app source code
- configured websockify with TLS

Open OnDemand on Grex

Current configuration (since 4.1)



University of Manitoba

DEMO

Final thoughts

We are very happy that OOD devs implemented this great feature, letting administrators increase the security posture of their systems.

Some caveats apply:

- wildcard DNS CNAMEs could be replaced with something else (dynamic DNS entries creation?)
- TLS could give users a false sense of security (it does enhance system posture, but users should always be vigilant)
- desktops/desktop applications are not officially supported

University of Manitoba

Final thoughts

Extra: websockify

Websockify needs to be available on all compute nodes in order for them to be able to run desktops/desktop applications.

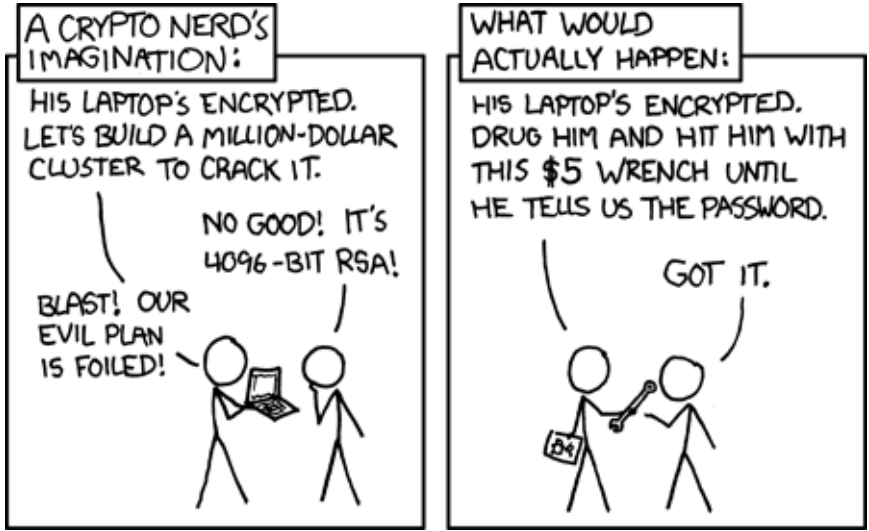
Standard (python) websockify drags in a lot of dependencies (most notably, `numpy`).

In 2024 we created our own custom Golang implementation of websockify.

Is websockify still needed?
Websocket (with TLS) functionality integrated in TurboVNC 3.0 (May 3 2022).

TurboVNC [Features]
TurboVNC [Issue 127]
Medium [Websocket Performance Comparison]

University of Manitoba



XKCD [538]

University of Manitoba

Questions?

Thank you