11falsenonelisttrue

Interfaces to Storage

When thinking about managing access, consider the available “interfaces” to the storage service.

The SMB Protocol (Samba)

SMB stands for “Server Message Block”. SMB is a network protocol used by Windows-based computers that allows systems within the same network to share files. In the Linux world, the protocol is employed by the Samba package. In the RIS Storage Service, the IBM Spectrum Scale software product uses the Ganesha Samba implementation.

Access to storage volumes is mediated by Wash U Active Directory (AD) groups. If you are a storage customer, you are a member of a group whose name begins with storage-. Usually it is based on your Lab Name or your Principle Investigator’s WashU Key ID, so storage-joe.user. As a member of this group you should be able to “mount” a SMB volume.

See Getting Connected.

Globus

Globus is an application that serves to move data into the storage service.

See
https://washu.atlassian.net/wiki/spaces/RUD/pages/1795588152/Moving+Data+With+Globus?atlOrigin=eyJpIjoiZTMwMDllMzhkZmY3NGJiZmFmZWFiNzE0NzI3ODYwMG
. In summary, a user must be a member of a Wash U AD group like storage-* to see data via Globus. First a user should confirm access via SMB. If that works, the same storage volume should be visible in the Globus application.

See
https://washu.atlassian.net/wiki/spaces/RUD/pages/1796145237/Moving+Data+With+Globus+CLI?atlOrigin=eyJpIjoiNjlkZTM4NzIxMGEwNDljY2I4ZWMxMTAyMDFjNTkzO
for using the Globus CLI in order to transfer data.

Via the Compute Service

If your research group also uses the integrated RIS Compute Service, you can access your Storage volumes from your Docker containers by specifying the path via an environment variable.

See
https://washu.atlassian.net/wiki/spaces/RUD/pages/1794965573/Access+Storage+Volumes?atlOrigin=eyJpIjoiZDM4ZjNiMmZiZWE3NDlhMWEzNTU1OTE1ZDIzN2RjNzQi
.

Wash U Active Directory Groups

At this point you are aware of the fact that access to RIS Services are mediated by membership in AD groups. The group names have a prefix for the service and are typically tied to the WashU Key ID of the Lab’s principal investigator (PI).

Wash U PI “Joe User” has a WashU Key ID joe.user. As a Storage and Compute consumer user Prof. User has membership in AD groups storage-joe.user and compute-joe.user.

Prof. User has a storage volume named after his ID joe.user and so there exists a storage Allocation of that name in the storage cluster, made visible via SMB at `smb://storage1.ris.wustl.edu/joe.user` and via filesystems in the Compute service at `/storage1/fs1/joe.user`.

For simple cases, this is perhaps “good enough” for most labs. PI Joe User asks that lab members are added to his `storage-joe.user` group and that’s that. But for cases where the storage volume contains lots of different kinds of data with different access control requirements we provide the use of project sub directories.

If Prof. User chooses, he can request **optional Project Sub-directories** (see Designing a Storage Layout). These project directories come along with **project AD groups**. Consider Prof. User requests a project named “First Project”. The following would exist:

> Allocation Name: joe.user
>
> Allocation Group: storage-joe.user
>
> Project Name: FirstProject
>
> Project Groups: storage-joe.user-firstproject-rw, storage-joe.user-firstproject-ro

The “ro” and “rw” refer to “read-only” and “read-write”.

Prof. User can now request that user Alice be given “read-write” permission while user Bob should be given “read-only” permission.

AD Groups and GPFS/NFSv4 Access Control Lists

The **primary** means by which RIS deploys ACLs with AD groups is by the use of GPFS/NFSv4 Access Control Lists. RIS Administrative Users have a view of the filesystem ACLs that Compute Users cannot see from inside Docker containers:

mmgetacl /storage1/fs1/joe.user/Active #NFSv4 ACL #owner:root #group:root special:owner@:rwxc:allow:DirInherit (X)READ/LIST (X)WRITE/CREATE (X)APPEND/MKDIR (X)SYNCHRONIZE (X)READ_ACL (X)READ_ATTR (X)READ_NAMED (-)DELETE (X)DELETE_CHILD (X)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL (X)WRITE_ATTR (X)WRITE_NAMED special:owner@:rw-c:allow:FileInherit (X)READ/LIST (X)WRITE/CREATE (X)APPEND/MKDIR (X)SYNCHRONIZE (X)READ_ACL (X)READ_ATTR (X)READ_NAMED (-)DELETE (-)DELETE_CHILD (X)CHOWN (-)EXEC/SEARCH (X)WRITE_ACL (X)WRITE_ATTR (X)WRITE_NAMED group:ris-it-admin:rwx-:allow:DirInherit (X)READ/LIST (X)WRITE/CREATE (X)APPEND/MKDIR (X)SYNCHRONIZE (X)READ_ACL (X)READ_ATTR (X)READ_NAMED (-)DELETE (X)DELETE_CHILD (-)CHOWN (X)EXEC/SEARCH (-)WRITE_ACL (X)WRITE_ATTR (X)WRITE_NAMED group:ris-it-admin:rw--:allow:FileInherit (X)READ/LIST (X)WRITE/CREATE (X)APPEND/MKDIR (X)SYNCHRONIZE (X)READ_ACL (X)READ_ATTR (X)READ_NAMED (-)DELETE (X)DELETE_CHILD (-)CHOWN (-)EXEC/SEARCH (-)WRITE_ACL (X)WRITE_ATTR (X)WRITE_NAMED group:storage-joe.user:rwx-:allow:DirInherit (X)READ/LIST (X)WRITE/CREATE (X)APPEND/MKDIR (X)SYNCHRONIZE (X)READ_ACL (X)READ_ATTR (X)READ_NAMED (-)DELETE (X)DELETE_CHILD (-)CHOWN (X)EXEC/SEARCH (-)WRITE_ACL (X)WRITE_ATTR (X)WRITE_NAMED group:storage-joe.user:rw--:allow:FileInherit (X)READ/LIST (X)WRITE/CREATE (X)APPEND/MKDIR (X)SYNCHRONIZE (X)READ_ACL (X)READ_ATTR (X)READ_NAMED (-)DELETE (X)DELETE_CHILD (-)CHOWN (-)EXEC/SEARCH (-)WRITE_ACL (X)WRITE_ATTR (X)WRITE_NAMED > ls -ld /storage1/fs1/joe.user/Active drwx------. 13 root root 8192 Jan 9 17:00 /storage1/fs1/joe.user/Active]]>

Here we see that members of the `storage-joe.user` group have `read-write` permission on the storage volume, even though the allocation directory is initially owned by root. These permissions are also inherited by any files created inside the allocation.

Now that Prof. Joe User has a First Project subdirectory, with more specific AD groups, we can observe more specific access controls:

mmgetacl /storage1/fs1/joe.user/Active/First_Project/ #NFSv4 ACL #owner:root #group:root special:owner@:rwxc:allow:DirInherit (X)READ/LIST (X)WRITE/CREATE (X)APPEND/MKDIR (X)SYNCHRONIZE (X)READ_ACL (X)READ_ATTR (X)READ_NAMED (-)DELETE (X)DELETE_CHILD (X)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL (X)WRITE_ATTR (X)WRITE_NAMED special:owner@:rw-c:allow:FileInherit (X)READ/LIST (X)WRITE/CREATE (X)APPEND/MKDIR (X)SYNCHRONIZE (X)READ_ACL (X)READ_ATTR (X)READ_NAMED (-)DELETE (-)DELETE_CHILD (X)CHOWN (-)EXEC/SEARCH (X)WRITE_ACL (X)WRITE_ATTR (X)WRITE_NAMED group:ris-it-admin:rwx-:allow:DirInherit (X)READ/LIST (X)WRITE/CREATE (X)APPEND/MKDIR (X)SYNCHRONIZE (X)READ_ACL (X)READ_ATTR (X)READ_NAMED (-)DELETE (X)DELETE_CHILD (-)CHOWN (X)EXEC/SEARCH (-)WRITE_ACL (X)WRITE_ATTR (X)WRITE_NAMED group:ris-it-admin:rw--:allow:FileInherit (X)READ/LIST (X)WRITE/CREATE (X)APPEND/MKDIR (X)SYNCHRONIZE (X)READ_ACL (X)READ_ATTR (X)READ_NAMED (-)DELETE (X)DELETE_CHILD (-)CHOWN (-)EXEC/SEARCH (-)WRITE_ACL (X)WRITE_ATTR (X)WRITE_NAMED group:storage-joe.user:rwx-:allow:DirInherit (X)READ/LIST (X)WRITE/CREATE (X)APPEND/MKDIR (X)SYNCHRONIZE (X)READ_ACL (X)READ_ATTR (X)READ_NAMED (-)DELETE (X)DELETE_CHILD (-)CHOWN (X)EXEC/SEARCH (-)WRITE_ACL (X)WRITE_ATTR (X)WRITE_NAMED group:storage-joe.user:rw--:allow:FileInherit (X)READ/LIST (X)WRITE/CREATE (X)APPEND/MKDIR (X)SYNCHRONIZE (X)READ_ACL (X)READ_ATTR (X)READ_NAMED (-)DELETE (X)DELETE_CHILD (-)CHOWN (-)EXEC/SEARCH (-)WRITE_ACL (X)WRITE_ATTR (X)WRITE_NAMED group:storage-joe.user-first_project-rw:rwx-:allow:DirInherit (X)READ/LIST (X)WRITE/CREATE (X)APPEND/MKDIR (X)SYNCHRONIZE (X)READ_ACL (X)READ_ATTR (X)READ_NAMED (-)DELETE (X)DELETE_CHILD (-)CHOWN (X)EXEC/SEARCH (-)WRITE_ACL (X)WRITE_ATTR (X)WRITE_NAMED group:storage-joe.user-first_project-rw:rw--:allow:FileInherit (X)READ/LIST (X)WRITE/CREATE (X)APPEND/MKDIR (X)SYNCHRONIZE (X)READ_ACL (X)READ_ATTR (X)READ_NAMED (-)DELETE (X)DELETE_CHILD (-)CHOWN (-)EXEC/SEARCH (-)WRITE_ACL (X)WRITE_ATTR (X)WRITE_NAMED group:storage-joe.user-first_project-ro:r-x-:allow:DirInherit (X)READ/LIST (-)WRITE/CREATE (-)APPEND/MKDIR (X)SYNCHRONIZE (X)READ_ACL (X)READ_ATTR (X)READ_NAMED (-)DELETE (-)DELETE_CHILD (-)CHOWN (X)EXEC/SEARCH (-)WRITE_ACL (-)WRITE_ATTR (-)WRITE_NAMED group:storage-joe.user-first_project-ro:r---:allow:FileInherit (X)READ/LIST (-)WRITE/CREATE (-)APPEND/MKDIR (X)SYNCHRONIZE (X)READ_ACL (X)READ_ATTR (X)READ_NAMED (-)DELETE (-)DELETE_CHILD (-)CHOWN (-)EXEC/SEARCH (-)WRITE_ACL (-)WRITE_ATTR (-)WRITE_NAMED > ls -ld /storage1/fs1/joe.user/Active/First_Project drwx------. 4 root root 8192 Jan 2 16:01 /storage1/fs1/joe.user/Active/First_Project]]>

Here we see that members of `storage-joe.user-first_project-rw` are given `read-write` permission while the `storage-joe.user-first_project-ro` group have `read-only` permission. These permissions are also inherited by files underneath the same project directory. Note that the creation of a project also necessitates ACL entries on the parent directories above it so that project group members may traverse the filesystem to get to the project directories.

It is important to manage membership in groups, as this is the way you control access to your data.

The primary `storage-key` group should be **the most trusted** (smallest) group

Project specific groups `storage-key-project-(ro|rw)` group should **specific to that data set**.

POSIX Permissions

From the point of view of a Compute Service execution node, the POSIX permissions on storage volumes may appear as follows:

In this case where there are no POSIX permissions for ordinary users, access is completely governed by the NFSv4 ACLs. When a user creates a new directory from the shell, it looks like this:

is submitted to queue . <> <> Using default tag: latest latest: Pulling from library/alpine Digest: sha256:c19173c5ada610a5989151111163d28a67368362762534d8a8121ce95cf2bd5a Status: Image is up to date for alpine:latest docker.io/library/alpine:latest /home/joe.user $ /home/joe.user $ cd /data /data $ mkdir newdir /data $ touch newdir/newfile]]>

That example shows the filesystem view from inside the container. The filesystem the permissions look like this:

ls -ld /data/newdir drwx------. 2 joe.user domain users 8192 Nov 8 11:35 /data/newdir]]>

From outside the container one could examine the ACL as well:

mmgetacl /storage1/fs1/joe.user/Active/newdir #NFSv4 ACL #owner:joe.user #group:domain users special:owner@:rwxc:allow:DirInherit:Inherited (X)READ/LIST (X)WRITE/CREATE (X)APPEND/MKDIR (X)SYNCHRONIZE (X)READ_ACL (X)READ_ATTR (X)READ_NAMED (-)DELETE (X)DELETE_CHILD (X)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL (X)WRITE_ATTR (X)WRITE_NAMED special:owner@:rw-c:allow:FileInherit:InheritOnly:Inherited (X)READ/LIST (X)WRITE/CREATE (X)APPEND/MKDIR (X)SYNCHRONIZE (X)READ_ACL (X)READ_ATTR (X)READ_NAMED (-)DELETE (-)DELETE_CHILD (X)CHOWN (-)EXEC/SEARCH (X)WRITE_ACL (X)WRITE_ATTR (X)WRITE_NAMED group:ris-it-admin:rwx-:allow:DirInherit:Inherited (X)READ/LIST (X)WRITE/CREATE (X)APPEND/MKDIR (X)SYNCHRONIZE (X)READ_ACL (X)READ_ATTR (X)READ_NAMED (-)DELETE (X)DELETE_CHILD (-)CHOWN (X)EXEC/SEARCH (-)WRITE_ACL (X)WRITE_ATTR (X)WRITE_NAMED group:ris-it-admin:rw--:allow:FileInherit:InheritOnly:Inherited (X)READ/LIST (X)WRITE/CREATE (X)APPEND/MKDIR (X)SYNCHRONIZE (X)READ_ACL (X)READ_ATTR (X)READ_NAMED (-)DELETE (X)DELETE_CHILD (-)CHOWN (-)EXEC/SEARCH (-)WRITE_ACL (X)WRITE_ATTR (X)WRITE_NAMED group:storage-joe.user:rwx-:allow:DirInherit:Inherited (X)READ/LIST (X)WRITE/CREATE (X)APPEND/MKDIR (X)SYNCHRONIZE (X)READ_ACL (X)READ_ATTR (X)READ_NAMED (-)DELETE (X)DELETE_CHILD (-)CHOWN (X)EXEC/SEARCH (-)WRITE_ACL (X)WRITE_ATTR (X)WRITE_NAMED group:storage-joe.user:rw--:allow:FileInherit:InheritOnly:Inherited (X)READ/LIST (X)WRITE/CREATE (X)APPEND/MKDIR (X)SYNCHRONIZE (X)READ_ACL (X)READ_ATTR (X)READ_NAMED (-)DELETE (X)DELETE_CHILD (-)CHOWN (-)EXEC/SEARCH (-)WRITE_ACL (X)WRITE_ATTR (X)WRITE_NAMED]]>

The user can use `chmod` as normal:

Example

Wash U PI “Joe User” has a WashU Key ID `joe.user`. As a Storage and Compute user he has membership in AD groups `storage-joe.user` and `compute-joe.user`. Prof. User employs a trusted postdoc named Terry, and lab members Alice and Bob, who are team leads for First Project and Second Project, respectively.

With SMB, using WashU Key credentials, the team has access to `smb://storage1.ris.wustl.edu/joe.user`

From Compute they can supply `LSF_DOCKER_VOLUMES=/storage1/fs1/joe.user:/data` to Docker jobs and then see his storage volume:

ls -l /data total 46054464 d--------- 2 root root 65536 Jan 25 2019 firstproject d--------- 2 root root 65536 Jan 25 2019 secondproject]]>

In this case, the following AD groups exist:

`storage-joe.user`	Contains Prof. Joe User himself, and his most trusted lab member, Terry.
`storage-joe.user-firstproject-ro`	Contains anyone who needs read-only access to First Project.
`storage-joe.user-firstproject-rw`	Contains Alice.
`storage-joe.user-secondproject-ro`	Contains anyone who needs read-only access to Second Project.
`storage-joe.user-secondproject-rw`	Contains Bob

Project Quotas: Moving Data

It is highly recommended to avoid use of `rsync -a` or `mv` when moving data between directories with set project quotas. Use of these commands do not allow group ownership to update resulting in a discrepancy in expected project quota usage.

Examples

- Allocation: smith
  - Project: firstproject (Project Quota: 5GB)
    - File: john-file1.txt (File Size: 5GB)
  - Project: secondproject (Project Quota: 5GB)
- Allocation: jones
  - Project: analysis1 (Project Quota: 10GB)

Scenario 1

Copy file to new fileset using `rsync -a` to preserve attributes.:

rsync -av smith/firstproject/john-file1.txt jones/analysis1/mary-file1.txt]]>

Result:

Allocation: smith (5GB total usage)

Project: firstproject (Project Quota usage: 5/5GB)

File: john-file1.txt

Project: secondproject (Quota usage: 0/5GB)

Allocation jones (5GB total usage)

Project: analysis1 (Project Quota usage: 0/10GB)

File: mary-file1.txt (File Size: 5GB)

Scenario 2

Copy file to new project in the same fileset using `rsync -a` to preserve attributes.:

rsync -av smith/firstproject/john-file1.txt smith/secondproject/output-file1.txt]]>

Result:

Allocation: smith (10GB total usage)

Project: firstproject (Project Quota usage: 10/5GB)

File: john-file1.txt

Project: secondproject (Quota usage: 0/5GB)

File: output-file1.txt

Allocation jones (0GB total usage)

Project: analysis1 (Project Quota usage: 0/10GB)