Arvados - Story #15372

Revise group permissions to separate them from permissions on managed objects

06/17/2019 02:32 PM - Ward Vandewege

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assigned To:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:	Arvados Future Sprints		

Description

From a customer:

> When I share a project with a user and give her manage permissions she can see the sharing page in wb2 and open the sharing dialog in wb2 ... when i change this to write she cannot see and change the sharing anymore but still read and write within the project. I think this is the behavior which was expected in our current release.
>
> Now I remove her as user, add a group where she is member of, and give it write permissions. She cannot see and change the sharing but can read and write within the project ... expected behavior ... when changing the permissions to manage, however nothing changes .... she still cannot see and change the sharing but read and write within the project. My expectation was that giving manage permission to a user directly or to a group with this user as member does not make any difference.

This behavior is apparently working as designed, but is inconsistent with user expectations. Part of the existing behavior may be due to implementation details where projects are a form of group, but that isn't something that users are (or should be) aware of.

Here's a sketch of how I think the model should work:

- Groups have the following types of relationships:
    - Viewer - users who have permission to see that the group exists and know its name, but not its membership. It's arguably desirable that they be able to see the owner or some other point of contact, but I think a blinded "contact the owner" or "request membership" feature might be a better way to connect people.
    - Viewer+ - users who can see the members (and, separately, administrators?) of a group. We may be able to bundle this with Viewer to simplify the model.
    - Administrator - users who can edit add/remove group members, change the group name or description. Administrators aren't granted access to the objects shared with the group unless they are also a member (and they can, of course, trivially add themselves, but that creates an audit trail)
    - Member - users who belong to the group and are granted the same permissions that the group has. Perhaps in the future this could be extended to groups as well so that you could have nested groups. Members can't necessarily see the membership list, but they can see the name of the group. Group membership does **not** convey any privileges on the group itself other than being able to see its name.
    - Shared objects - objects which are shared with the group members on either a read-only, read/write, or manage basis. All group members have the same access to the object and it's children ie access privileges are **not** tailored on a member by member basis.

- Special cases:
    - "all users" group - users will be able to see the name of the group by virtue of their membership in it, but won't be able to see the full list unless they also have Viewer+ privileges
    - groups which are secret from their members (ie the admin's "Losers" group) - this model doesn't currently support this, but could be modified by removing member's default access to the names of projects that they are members of and granting this separately (one mechanism would be to share the group with itself, granting that access to its members)

Implementation (taken from #note-19, and updated)

Mnemonic	tail_uuid type	head_uuid type	detail	current	future
object viewer	user or role	not a group/role/project	target is readable	can_read	can_read
object collaborator	user or role	not a group/role/project	target is writable	can_write	can_write
object manager	user or role	not a	can add/remove	can_manage	can_manage

		group/role/project	sharing links with same head_uuid		
(Project) viewer	user or role	project	project+children+grandchildren are readable	can_read	can_read
(Project) collaborator	user or role	project	project+children+grandchildren are writable	can_write	can_write
(Project) manager	user or role	project	can add/remove sharing links with project/children/grandchildren as head_uuid	can_manage	can_manage
(User) viewer	user or role	user	user is readable	can_read	can_read
(User) collaborator	user or role	user	user is writable	can_write	can_write
(User) manager	user or role	user	can add/remove sharing links with same head_uuid	can_manage	can_manage
Viewer	user or role	role	role is readable	(n/a)	can_read
Viewer+	user or role	role	role membership list is readable (implies viewer)	(n/a)	can_list_members
Writer	user or role	role	role is writable (e.g., user can rename the target role)	(n/a)	can_write
Member	user or role	role	permissions on other objects via role (implies viewer, but not viewer+)	(n/a)	can_use_permissions
Member+	user or role	role	permissions on other objects via role (implies viewer+)	(n/a)	can_use_permissions + can_list_members
Administrator	user or role	role	can add/remove members and viewers (implies member+)	can_manage→, ←can_read	can_use_permissions + can_manage
Administrator-	user or role	role	can add/remove members and viewers (without implying member)	(n/a)	can_manage

Notes

- A "role" group's UUID cannot be used as an owner_uuid (IOW a non-project group can't have children)
- A "project" group's UUID cannot be used as the tail_uuid of a permission link
- A user UUID can be a target of a can_use_permission link -- this allows access to all of the user's permissions and home project tree
- A group cannot have a group_class other than "role" or "project" (and group_class cannot be null)
- A group can have a can_use_permissions link to another group; this graph is (still) traversed, so groups can be composed
- A can_list_members link permits its subject (tail_uuid) to read all can_use_permissions links from other users/groups to the same head_uuid, as well as the member users/groups themselves
- can_manage implies can_list_members

Migration/implementation

- Obviously, update the materialized view of permissions and the "readable_by" query
- Clients that (might) need to be updated: Workbench1, Workbench2, group-sync, login-sync, arv-mount (probably drop Workbench1's group admin feature entirely rather than update it)
- Existing "Administrator" permissions convert to "Member+"

- Existing group→user links (e.g., "all users" links) convert to "Member"
- Existing project-sharing links (including read-only sharing, and sharing with role groups) can be left alone
- Warn/error if a non-project group is used as any object's owner_uuid, or a project group is used as a permission link's tail_uuid
- Update use of "all users" links in user activation code

Subtasks:

Task # 15392: Review	**New**
Task # 15731: [Workbench] Remove legacy group mechanism	**New**

Related issues:

Related to Arvados - Story #15730: [Workbench 2] Implement UI for new group p...	**New**
Related to Arvados - Story #15732: Update group sync tool for new group permi...	**New**
Related to Arvados - Feature #16571: Permission system supports seeing & shar...	**New**
Related to Arvados - Bug #16538: workbench 2 should not allow sharing "write"...	**Rejected**

History

#1 - 06/17/2019 02:32 PM - Ward Vandewege

- Description updated

#2 - 06/17/2019 02:32 PM - Ward Vandewege

- Description updated

#3 - 06/17/2019 02:38 PM - Tom Morris

- Project changed from Arvados Private to Arvados

- Description updated

#4 - 06/17/2019 02:39 PM - Tom Morris

- Subject changed from manage permissions bug? to manage permission doesn't work for group members?

#5 - 06/19/2019 03:27 PM - Tom Morris

- Target version changed from To Be Groomed to 2019-07-03 Sprint

#6 - 06/19/2019 03:33 PM - Tom Clegg

- Assigned To set to Tom Clegg

#7 - 06/19/2019 03:36 PM - Peter Amstutz

- Assigned To changed from Tom Clegg to Peter Amstutz

#8 - 06/21/2019 08:37 PM - Peter Amstutz

I suspect this is working as designed.

It depends on the access that the user has to the group.

If the user has "write" access to the group, that means the user only gets "write" access to things that the group can "write" or "manage".

If the user has "manage" access to the group, then the user also gets "manage" access to things the group can manage.

If the user has "manage" access to the group, and the groups gets "write" access, then the user only gets "write" access.

As a feature, we may be missing a permission level where a user is able to manage things the group can manage, but not manage the group itself.

#9 - 06/24/2019 07:13 PM - Tom Clegg

I think the expected behavior depends on what you mean by a user being a member of a group, which isn't quite clear in the description.

In this case the user **should** see sharing controls for the project. This is what Workbench1 does when you use the "add user to group" admin feature.

```
       can_manage            can_manage
user --------------> group --------------> project
```

In this case the user **should not** see sharing controls for the project. This might happen if something other than the Workbench1 admin page is used to add the user→group permission link.

```
       can_write             can_manage
```

```
user --------------> group --------------> project
```

Permissions are narrowed to the least powerful permission on the path.

#10 - 06/26/2019 01:16 PM - Peter Amstutz

- Status changed from New to In Progress

#11 - 07/03/2019 03:26 PM - Tom Morris

- Target version changed from 2019-07-03 Sprint to 2019-07-17 Sprint

To be investigated - does the group sync tool set things up the same way that workbench does?

#12 - 07/03/2019 04:11 PM - Tom Morris

If I'm understanding it correctly, the current model seems wrong to me. I expect the rights to manage a group to be orthogonal to the rights that membership in a group grants me. The rights that I have to an object should be the union of my individual rights plus the rights of all groups of which I'm a member. Granting a group member manage access to the group itself, just so that they're able to exercise their right to manage objects shared with the group, allows them to change membership of the group, which seems highly undesirable.

Permissions should be based on group membership, not access to the group.

#13 - 07/17/2019 02:50 PM - Tom Morris

- Status changed from In Progress to New

- Assigned To deleted (Peter Amstutz)

- Target version changed from 2019-07-17 Sprint to To Be Groomed

#14 - 09/18/2019 04:37 PM - Tom Morris

- Description updated

#15 - 09/18/2019 04:39 PM - Tom Morris

- Tracker changed from Bug to Story

- Subject changed from manage permission doesn't work for group members? to Revise group permissions to separate them from permissions on managed objects

- Description updated

#16 - 09/18/2019 08:08 PM - Peter Amstutz

Proposal:

Permission links separately express "access to record immediate" and "transitive access through record"

Access levels

0 - None
1 - View ("metadata only", for collections this means manifest does not have permission signatures)
2 - Read
3 - Write
4 - Manage

Viewer: "View_None"

User can see the group record (name, description) but not gain any access through the group. "View" access does not grant access to the link records that point to the group, so the client cannot access the user list.

Viewer+: "Read_None"

"Read" access to a record also confers permission to query links that point **to** the record. This allows the client to get the list of permission links that grant access to this group. (This would allow the client to get a list of user uuids but not actually grant access to the user records, this might be fine if the user has access to the "All Users" group).

Administrator: "Manage_Manage"

"Manage" grants the ability to edit the record, and also create/update permission links that target the record. Grants at most "manage" permission to anything accessible through the group.

Member: "Read_Manage"

"Read" grants ability to read the record, and see permission links pointing to the record. Grants at most "manage" permission to anything accessible through the group.

Traversing the graph

Normal member

Membership is defined as having any permission link with tail_uuid of the user and head_uuid of the group.

start at user A

has "Read_Manage" for group B

group B has "Read_Read" for project C

project C owns collection D (this is Manage_Manage)

1. Traverse A->B, current[traversal] = Manage
   1. Compute min(current[traversal], A->B[record access]) = min(Manage, Read) = Read, access level for B is 'Read'
   2. Compute min(current[traversal], A->B[traversal]) = min(Manage, Manage) and update current[traversal] = Manage
2. Traverse B->C, current[traversal] = Manage
   1. Compute min(current[traversal], B->C[record access]) = min(Manage, Read) = Read, access level for C is 'Read'
   2. Compute min(current[traversal], B->C[traversal]) = min(Manage, Read) and update current[traversal] = Read
3. Traverse C->D, current[traversal] = Read
   1. Compute min(current[traversal], C->D[record access]) = min(Read, Manage) = Read, access level for D is 'Read'
   2. Compute min(current[traversal], B->C[traversal]) = min(Read, Manage) and update current[traversal] = Read

Group admin

has "Manage_Manage" for group B

1. Traverse A->B, current[traversal] = Manage
   1. Compute min(current[traversal], A->B[record access]) = min(Manage, Manage) = Read, access level for B is 'Manage'
   2. Compute min(current[traversal], A->B[traversal]) = min(Manage, Manage) and update current[traversal] = Manage

Migration

- can_manage turns into Manage_Manage
- can_write turns into Write_Write when head_uuid is a project and Read_Manage when head_uuid is a user group
- can_read turns into Read_Read unless head_uuid is a user, then it is "View_None"

#17 - 09/18/2019 08:40 PM - Peter Amstutz

Could add a special case "if group is readable, and permission link exists group->user, then 'user' is viewable"

#18 - 10/01/2019 02:29 PM - Peter Amstutz

Might be worth considering how our permission system might interact with the GA4GH Data Use Ontology

https://www.ga4gh.org/news/data-use-ontology-approved-as-a-ga4gh-technical-standard/

#19 - 10/15/2019 10:09 PM - Tom Clegg

Group permission type	detail	current	future
(non-group target)	target is visible	can_read	can_read
(Project) viewer	project+children+grandchildren are visible	can_read	can_read
(Project) collaborator	project+children+grandchildren are writable	can_write	can_write
Viewer	group is visible	(n/a)	can_read
Viewer+	group membership list is visible (implies viewer)	(n/a)	can_list_members
Member	permissions on other objects via group (implies viewer, but not viewer+)	(n/a)	can_use_permissions
Member+	permissions on other objects via group (implies viewer+)	(n/a)	can_use_permissions + can_list_members
Administrator	can add/remove members and viewers (implies member and	can_manage→, ←can_read	can_use_permissions + can_manage

	viewer+)		
Administrator-	can add/remove members and viewers (without implying member)	(n/a)	can_manage

Notes

- A "role" group's UUID cannot be used as an owner_uuid (IOW a non-project group can't have children)
- A "project" group's UUID cannot be used as the tail_uuid of a permission link
- A user UUID can be a target of a can_use_permission link -- this allows access to all of the user's permissions and home project tree
- A group cannot have a group_class other than "role" or "project" (and group_class cannot be null)
- A group can have a can_use_permissions link to another group; this graph is (still) traversed, so groups can be composed
- A can_list_members link permits its subject (tail_uuid) to read all can_use_permissions links from other users/groups to the same head_uuid, as well as the member users/groups themselves
- can_manage implies can_list_members

Migration/implementation

- Obviously, update the materialized view of permissions and the "readable_by" query
- Clients that (might) need to be updated: Workbench1, Workbench2, group-sync, login-sync, arv-mount (probably drop Workbench1's group admin feature entirely rather than update it)
- Existing "Administrator" permissions convert to "Member+"
- Existing group→user links (e.g., "all users" links) convert to "Member"
- Existing project-sharing links (including read-only sharing, and sharing with role groups) can be left alone
- Warn/error if a non-project group is used as any object's owner_uuid, or a project group is used as a permission link's tail_uuid
- Update use of "all users" links in user activation code

#20 - 10/16/2019 05:45 PM - Tom Morris

- Related to Story #15730: [Workbench 2] Implement UI for new group permission scheme added

#21 - 10/16/2019 05:46 PM - Tom Morris

- Target version changed from To Be Groomed to Arvados Future Sprints

- Story points set to 5.0

#22 - 10/16/2019 05:48 PM - Tom Morris

- Related to Story #15732: Update group sync tool for new group permission scheme added

#23 - 11/13/2019 05:20 PM - Lucas Di Pentima

- Description updated

#24 - 06/29/2020 03:52 PM - Peter Amstutz

- Related to Feature #16571: Permission system supports seeing & sharing with a group without having access to group contents. added

#25 - 06/30/2020 03:25 PM - Tom Clegg

- Related to Bug #16538: workbench 2 should not allow sharing "write" or "manage" to all users group added

#26 - 07/22/2020 04:06 PM - Tom Clegg

- Description updated

#27 - 07/22/2020 04:09 PM - Tom Clegg

- Description updated

#28 - 07/22/2020 04:11 PM - Tom Clegg

- Description updated

#29 - 07/22/2020 04:18 PM - Tom Clegg

- Description updated

#30 - 07/22/2020 04:35 PM - Tom Clegg

- Description updated

#31 - 07/22/2020 04:44 PM - Tom Clegg

- Description updated

#32 - 07/22/2020 05:04 PM - Tom Clegg

- Description updated

#33 - 07/22/2020 05:07 PM - Tom Clegg

- Description updated