NAME

zfs-allow - delegate ZFS administration permissions to unprivileged users

SYNOPSIS

zfs allow [**-dglu**] *user*|*group*[,*user*|*group*]<?> *perm*|*@setname*[,*perm*|*@setname*]<?> *filesystem*|*volume*
zfs allow [**-dl**] **-e**|**everyone** *perm*|*@setname*[,*perm*|*@setname*]<?> *filesystem*|*volume*
zfs allow -c *perm*|*@setname*[,*perm*|*@setname*]<?> *filesystem*|*volume*
zfs allow -s *@setname perm*|*@setname*[,*perm*|*@setname*]<?> *filesystem*|*volume*
zfs unallow [**-dglru**] *user*|*group*[,*user*|*group*]<?> [*perm*|*@setname*[,*perm*|*@setname*]<?>] *filesystem*|*volume*
zfs unallow [**-dlr**] **-e**|**everyone** [*perm*|*@setname*[,*perm*|*@setname*]<?>] *filesystem*|*volume*
zfs unallow [**-r**] **-c** [*perm*|*@setname*[,*perm*|*@setname*]<?>] *filesystem*|*volume*
zfs unallow [**-r**] **-s** *@setname* [*perm*|*@setname*[,*perm*|*@setname*]<?>] *filesystem*|*volume*

DESCRIPTION

zfs allow *filesystem*|*volume*

Displays permissions that have been delegated on the specified filesystem or volume. See the other forms of **zfs allow** for more information.

Delegations are supported under Linux with the exception of **mount**, **unmount**, **mountpoint**, **canmount**, **rename**, and **share**. These permissions cannot be delegated because the Linux mount(8) command restricts modifications of the global namespace to the root user.

zfs allow [**-dglu**] *user*|*group*[,*user*|*group*]<?> *perm*|*@setname*[,*perm*|*@setname*]<?> *filesystem*|*volume*

zfs allow [**-dl**] **-e**|**everyone** *perm*|*@setname*[,*perm*|*@setname*]<?> *filesystem*|*volume*

Delegates ZFS administration permission for the file systems to non-privileged users.

-d Allow only for the descendent file systems.

-e|**everyone**

Specifies that the permissions be delegated to everyone.

-g *group*[,*group*]<?>

Explicitly specify that permissions are delegated to the group.

-l Allow "locally" only for the specified file system.

-u *user*[,*user*]<?>

Explicitly specify that permissions are delegated to the user.

user|*group*[,*user*|*group*]<?>

Specifies to whom the permissions are delegated. Multiple entities can be specified as a comma-separated list. If neither of the **-gu** options are specified, then the argument is interpreted preferentially as the keyword **everyone**, then as a user name, and lastly as a group name. To specify a user or group named "everyone", use the **-g** or **-u** options. To specify a group with the same name as a user, use the **-g** options.

perm|*@setname*[,*perm*|*@setname*]<?>

The permissions to delegate. Multiple permissions may be specified as a comma-separated list. Permission names are the same as ZFS subcommand and property names. See the property list below. Property set names, which begin with **@**, may be specified. See the **-s** form below for details.

If neither of the **-dl** options are specified, or both are, then the permissions are allowed for the file system or volume, and all of its descendents.

Permissions are generally the ability to use a ZFS subcommand or change a ZFS property. The following permissions are available:

NAME	TYPE	NOTES
allow	subcommand	Must also have the permission that is being allowed
bookmark	subcommand	
clone	subcommand	Must also have the **create** ability and **mount** ability in the origin file system
create	subcommand	Must also have the **mount** ability. Must also have the **refreservation** ability to creat volume.
destroy	subcommand	Must also have the **mount** ability
diff	subcommand	Allows lookup of paths within a dataset given an object number, and the ability to cr
hold	subcommand	Allows adding a user hold to a snapshot
load-key	subcommand	Allows loading and unloading of encryption key (see **zfs load-key** and **zfs unload-k**
change-key	subcommand	Allows changing an encryption key via **zfs change-key**.
mount	subcommand	Allows mounting/umounting ZFS datasets
promote	subcommand	Must also have the **mount** and **promote** ability in the origin file system
receive	subcommand	Must also have the **mount** and **create**

		ability
release	subcommand	Allows releasing a user hold which might destroy the snapshot
rename	subcommand	Must also have the **mount** and **create** ability in the new parent
rollback	subcommand	Must also have the **mount** ability
send	subcommand	
share	subcommand	Allows sharing file systems over NFS or SMB protocols
snapshot	subcommand	Must also have the **mount** ability
groupquota	other	Allows accessing any **groupquota@**<*?*> property
groupobjquota	other	Allows accessing any **groupobjquota@**<*?*> property
groupused	other	Allows reading any **groupused@**<*?*> property
groupobjused	other	Allows reading any **groupobjused@**<*?*> property
userprop	other	Allows changing any user property
userquota	other	Allows accessing any **userquota@**<*?*> property
userobjquota	other	Allows accessing any **userobjquota@**<*?*> property
userused	other	Allows reading any **userused@**<*?*> property
userobjused	other	Allows reading any **userobjused@**<*?*> property
projectobjquota	other	Allows accessing any **projectobjquota@**<*?*> property
projectquota	other	Allows accessing any **projectquota@**<*?*> property
projectobjused	other	Allows reading any **projectobjused@**<*?*> property
projectused	other	Allows reading any **projectused@**<*?*> property

aclinherit	property
aclmode	property
acltype	property
atime	property
canmount	property
casesensitivity	property
checksum	property
compression	property
context	property
copies	property
dedup	property
defcontext	property
devices	property
dnodesize	property
encryption	property
exec	property
filesystem_limit	property
fscontext	property
keyformat	property
keylocation	property
logbias	property
mlslabel	property
mountpoint	property
nbmand	property
normalization	property
overlay	property
pbkdf2iters	property
primarycache	property
quota	property
readonly	property
recordsize	property
redundant_metadata	property
refquota	property
refreservation	property
relatime	property
reservation	property
rootcontext	property
secondarycache	property
setuid	property
sharenfs	property

sharesmb	property
snapdev	property
snapdir	property
snapshot_limit	property
special_small_blocks	property
sync	property
utf8only	property
version	property
volblocksize	property
volmode	property
volsize	property
vscan	property
xattr	property
zoned	property

zfs allow -c *perm*|*@setname*[,*perm*|*@setname*]<?> *filesystem*|*volume*

Sets "create time" permissions. These permissions are granted (locally) to the creator of any newly-created descendent file system.

zfs allow -s *@setname perm*|*@setname*[,*perm*|*@setname*]<?> *filesystem*|*volume*

Defines or adds permissions to a permission set. The set can be used by other **zfs allow** commands for the specified file system and its descendents. Sets are evaluated dynamically, so changes to a set are immediately reflected. Permission sets follow the same naming restrictions as ZFS file systems, but the name must begin with **@**, and can be no more than 64 characters long.

zfs unallow [**-dglru**] *user*|*group*[,*user*|*group*]<?> [*perm*|*@setname*[,*perm*|*@setname*]<?>] *filesystem*|*volume*

zfs unallow [**-dlr**] **-e**|**everyone** [*perm*|*@setname*[,*perm*|*@setname*]<?>] *filesystem*|*volume*

zfs unallow [**-r**] **-c** [*perm*|*@setname*[,*perm*|*@setname*]<?>] *filesystem*|*volume*

Removes permissions that were granted with the **zfs allow** command. No permissions are explicitly denied, so other permissions granted are still in effect. For example, if the permission is granted by an ancestor. If no permissions are specified, then all permissions for the specified *user*, *group*, or **everyone** are removed. Specifying **everyone** (or using the **-e** option) only removes the permissions that were granted to everyone, not all permissions for every user and group. See the **zfs allow** command for a description of the **-ldugec** options.

-r Recursively remove the permissions from this file system and all descendents.

zfs unallow [**-r**] **-s** *@setname* [*perm*|*@setname*[,*perm*|*@setname*]<?>] *filesystem*|*volume*

Removes permissions from a permission set. If no permissions are specified, then all permissions are removed, thus removing the set entirely.

EXAMPLES

Example 1: Delegating ZFS Administration Permissions on a ZFS Dataset

The following example shows how to set permissions so that user *cindys* can create, destroy, mount, and take snapshots on *tank/cindys*. The permissions on *tank/cindys* are also displayed.

```
# zfs allow cindys create,destroy,mount,snapshot tank/cindys
# zfs allow tank/cindys
---- Permissions on tank/cindys --------------------------------------
Local+Descendent permissions:
        user cindys create,destroy,mount,snapshot
```

Because the *tank/cindys* mount point permission is set to 755 by default, user *cindys* will be unable to mount file systems under *tank/cindys*. Add an ACE similar to the following syntax to provide mount point access:

```
# chmod A+user:cindys:add_subdirectory:allow /tank/cindys
```

Example 2: Delegating Create Time Permissions on a ZFS Dataset

The following example shows how to grant anyone in the group *staff* to create file systems in *tank/users*. This syntax also allows staff members to destroy their own file systems, but not destroy anyone else’s file system. The permissions on *tank/users* are also displayed.

```
# zfs allow staff create,mount tank/users
# zfs allow -c destroy tank/users
# zfs allow tank/users
---- Permissions on tank/users ---------------------------------------
Permission sets:
        destroy
Local+Descendent permissions:
        group staff create,mount
```

Example 3: Defining and Granting a Permission Set on a ZFS Dataset

The following example shows how to define and grant a permission set on the *tank/users* file system. The permissions on *tank/users* are also displayed.

```
# zfs allow -s @pset create,destroy,snapshot,mount tank/users
# zfs allow staff @pset tank/users
# zfs allow tank/users
---- Permissions on tank/users ---------------------------------------
Permission sets:
    @pset create,destroy,mount,snapshot
Local+Descendent permissions:
    group staff @pset
```

Example 4: Delegating Property Permissions on a ZFS Dataset

The following example shows to grant the ability to set quotas and reservations on the *users/home* file system. The permissions on *users/home* are also displayed.

```
# zfs allow cindys quota,reservation users/home
# zfs allow users/home
---- Permissions on users/home ---------------------------------------
Local+Descendent permissions:
    user cindys quota,reservation
cindys% zfs set quota=10G users/home/marks
cindys% zfs get quota users/home/marks
NAME              PROPERTY  VALUE  SOURCE
users/home/marks  quota     10G    local
```

Example 5: Removing ZFS Delegated Permissions on a ZFS Dataset

The following example shows how to remove the snapshot permission from the *staff* group on the **tank/users** file system. The permissions on **tank/users** are also displayed.

```
# zfs unallow staff snapshot tank/users
# zfs allow tank/users
---- Permissions on tank/users ---------------------------------------
Permission sets:
    @pset create,destroy,mount,snapshot
Local+Descendent permissions:
    group staff @pset
```