zfs

Configure des systèmes de fichiers ZFS

Description

La commande zfs configure des datasets ZFS dans des pools de stockage ZFS. Un dataset est identifié par un chemin unique dans l'espace de nom ZFS. par exemple **pool/{filesystem,volume,snapshot}** où la longueur maximum d'un nom de dataset est MAXNAMELEN (256octets). Un dataset peut être un des suivants :

filesystem Un dataset ZFS de type filesystem peut être monté dans l'espace de nom système standard et fonctionne comme tout autre système de fichier. Bien que les système de fichiers ZFS sont conçus pour être conforme POSIX, des problème connus existent qui empêche la conformité dans certains cas. Les applications qui dépendent des conformités standard peut échouer lors de la vérification de l'espace disque disponible.

volume Un volume logique exporté comme périphérique block ou brute. Ce type de dataset devrait seulement être utilisé sous des circonstances particulière. Les systèmes de fichier sont typiquement utilisé dans la plupart des environnements.

snapshot Une version lecture seul d'un filesystem ou volume à un point donné dans le temps. C'est spécifié en filesystem@name ou volume@name.

Hiérarchie des systèmes de fichier

Un pool de stockage ZFS est une collection logique de périphériques qui fournissent de l'espace pour les datasets. Un pool de stockage est également la racine de la hiérarchie de système de fichiers ZFS. La racine du pool peut être accédé comme système de fichier, tel que monter et démonter, prendre des snapshots, et définir des propriétés. Les caractéristiques de stockage physique, cependant, sont gérés par la commande zpool.

Snapshots

Un snapshot est une copie lecture seul d'un système de fichier ou un volume. Les snapshots peuvent être crées extrêment rapidement, et ne consomment initialement aucun espace dans le pool. Comme les données dans le dataset actif changent, le snapshot consomme plus de données et qui sont partagés avec le dataset actif. Les snapshots peuvent avoir des noms arbitraires. Les snapshots de volumes peut être clonés ou annulés, mais ne peuvent pas être accédés indépendamment.

Les snapshots de filesystem peuvent être accédés sous le répertoire .zfs/snapshot dans le système de fichier racine. Les snapshots sont automatiquement montés à la demande et peuvent être démontés à intervalles réguliers. La visibilité du répertoire .zfs peut être contrôlé par la propriété snapdir.

Clones

Un clone est un volume en écriture ou un système de fichier dont le contenu initial est le même qu'un autre dataset. Comme avec les snapshots, créer un clone est presque instantané, et ne consomme pas d'espace disque initialement.

Les clones peuvent seulement être créés depuis un snapshot. Quand un snapshot est cloné, il créé une dépendance implicit entre le parent et l'enfant. Même quand un clone est créé ailleurs dans la hiérarchie de dataset, le snapshot original ne peut pas être détruit tant qu'un clone existe. La propriété origin expose cette dépendance, et la commande destroy liste de telles dépendances, si elles existent.

La relation de dépendance des parent/enfant des clones peut être renversé en utilisant la sous-commande promote. Le système de fichier "origin" devient un clone du système de fichier spécifié, qui rend possible de détruire le système de fichier depuis lequel le clone avait été créé.

Points de montage

Créer un système de fichier ZFS est une opération simple, donc le nombre de systèmes de fichier par système est susceptible d'être nombreux. Pour faire face à cela, ZFS gère automatiquement le montage et le démontage des systèmes de fichiers sans avoir besoin d'éditer /etc/vfstab. Tous les systèmes de fichiers gérés automatiquement sont montés par ZFS au boot.

Par défaut, les systèmes de fichiers sont montés sous /path, où path est le nom du système de fichier dans l'espace de nom ZFS. Les répertoires sont créés et détruis si nécessaires.

Un système de fichier peut également avoir un point de montage dans la propriété **mountpoint**. Ce répertoire est créé si nécessaire, et ZFS monte automatiquement le système de fichier quand **zfs mount -a** est invoqué (sans éditer **/etc/vfstab**). **mountpoint** peut être hérité, donc si **pool/home** a un point de montage de **/export/stuff**, alors le **pool/home/user** hérite automatiquement d'un point de montage **/export/stuff/user**.

Une propriété **mountpoint** de système de fichier à **none** empêche le système de fichier d'être monté. Si nécessaire, les systèmes de fichiers ZFS peuvent également être gérés avec les outils traditionnels. Si un point de montage de système de fichier est mis à **legacy**, ZFS ne tente pas de gérer le système de fichier.

Zones

Un système de fichiers ZFS peut être ajouté à une zone non-globale en utilisant la sous-commande **zonecfg add fs**. Un système de fichier ZFS qui est ajouté à une zone non-globale doit avoir sa propriété **mountpoint** à **legacy**.

Les propriétés physiques d'un système de fichier ajouté sont contrôlé par l'administrateur global. Cependant, l'administrateur de zone peut créer, modifie, ou détruire des fichiers dans le système de fichier ajouté, en fonction de la manière dont le système de fichier est monté.

Un dataset peut également être délégué dans une zone non-globale en utilisant la sous-commande **zonecfg add dataset**. Vous ne pouvez pas déléguer un dataset à une zone et l'enfant du même dataset à une autre zone. L'administrateur de zone peut changer les propriétés du dataset ou un de ses enfant. Cependant, la propriété **quota** est contrôlée par l'administrateur global.

Un volume ZFS peut être ajouté comme périphérique à une zone non-globale en utilisant la sous-commande **zonecfg add device**. Cependant, ses propriétés physiques peuvent être modifiées seulement par l'administrateur global.

Après qu'un dataset est délégué à une zone non-globale, la propriété **zoned** est automatiquement définie. Un système de fichier zoned ne peut pas être monté dans la zone globale, vu que l'administrateur de zone peut avoir définis le point de montage à des valeurs inacceptables.

L'administrateur global peut effacer la propriété **zoned**, bien que cela devrait être fait avec une extrême attention. L'administrateur global devrait vérifier que tous les points de montage sont acceptable avant d'effacer cette propriété.

Déduplication

La déduplication est un processus pour supprimer les données redondantes au niveau block, réduisant la quantité globale de données stockée. Si un système de fichier a la propriété **dedup**, les blocks de données dupliqués sont supprimés de manière synchrone. Le résultat est que seul une donnée unique est stockée et les composants communs sont partagés entre les fichiers.

Propriétés natives

Les propriétés sont divisées en 2 types, les propriétés natives et les propriétés définis par l'utilisateur (user). Les propriétés natives exportent des statistiques internes ou contrôle le fonctionnement de ZFS. En plus, les propriétés natives sont soit éditables, soit lecture-seul. Les propriété user n'ont pas d'effet sur le fonctionnement de ZFS, mais peuvent être utilisés pour annoter les datasets d'une manière significative dans l'environnement.

Tout dataset a un jeu de propriété qui exporte des statistiques sur le dataset aussi bien que contrôles divers modes. Les propriétés sont hérités du parent sauf s'ils sont spécifiés dans l'enfant. Certaines propriétés s'appliquent uniquement à certains type de datasets.

Les valeurs des propriétés numériques peuvent être spécifié en utilisant des suffixes human-readable (ex : k, KB, M, Gb, etc.). Les valeurs de propriétés non-numériques sont sensibles à la casse et doivent être en minuscule, sauf pour **mountpont**, **sharenfs**, et **sharesmb**.

Les propriétés natives suivantes consistent de statistiques lecture-seules sur le dataset. Ces propriétés ne peuvent être ni des jeux, ni hérités. Les propriétés natives s'appliquent à tous les types de dataset sauf mention contraire.

available La quantité d'espace disponible dans le dataset et tous ses enfants, assumant qu'il n'y a pas d'autre activité dans le pool. Parce que l'espace est partagé dans le pool, la disponibilité peut être limité par plusieurs facteurs, incluant la taille physique du pool, quotas, reservations, ou d'autres datasets dans le pool.

compressratio Le ratio de compression pour ce dataset, exprimé en multiplicateur. La compression peut être activée avec zfs set compression=on. Défaut : off.

creation La date de création du dataset

defer_destroy Cette propriété est on si le snapshot a été marqué pour une destruction déférée en utilisant la commande **zfs destroy -d**. Sinon, la propriété est off.

origin Pour les systèmes de fichier ou les volumes clonés, le snapshot depuis lequel le clone a été créé. L'origine ne peut pas être supprimé tant qu'un clone existe.

referenced La quantité de donnée qui est accessible par ce dataset, qui peut être partagé ou non avec d'autre datasets dans le pool. Quan un snapshot ou un clone est créé, il référence initialement la même quantité d'espace que le système de fichier ou le snapshot depuis lequel il a été créé, vu que son contenu est identique.

type le type de dataset : **filesystem**, **volume**, ou **snapshot**

used La quantité d'espace disque consommé par ce dataset et tous ses descendants. C'est la valeur qui est comparée au quota et à la réservation du dataset. L'espace utilisé n'inclus pas cette réservation du dataset, mais prend en compte les réservations de tous les dataset descendants. La quantité d'espace qu'un dataset consomme depuis son parent, aussi bien que la quantité d'espace qui sont libérés si ce dataset est détruit récursivement, est supérieur à son espace utilisé et sa réservation.

Quand des snapshots sont créés, leur espace est initialement partagé entre le snapshot et le système de fichier, et possiblement avec les snapshots précédents. Vu que le système de fichier change, l'espace qui a été précédemment partagé devient unique au snapshot, et compté dans l'espace utilisé du snapshot. Additionnellement, supprimer les snapshots peut augmenter la quantité d'espace unique à (et utilisé par) d'autre shapshots.

La quantité d'espace utilisé, disponible, ou référencé ne prend pas en compte les changements en attente, Ces changements en attente sont généralement pris en comptes dans les secondes qui suivent. Envoyer un changement au disque avec fsync ou O_SYNC ne garantit pas nécessairement que les informations d'utilisation de l'espace est mis à jours immédiatement.

usedby* Les propriétés **usedby*** décomposent les propriétés **used** dans les diverses raisons pour lequel l'espace est uilisé. Spécifiquement, **used = usedbychildren + usedbydataset + usedbyrefreservation +, usedbysnapshots**. Ces propriétés sont seulement disponible pour les datasets créés dans des pools version 13.

usedbychildren La quantité d'espace utilisé par l'enfant de ce dataset, qui serait libéré si tous le dataset de l'enfant était détruit.

usedbydataset La quantité de l'espace utilisé par ce dataset lui-même, qui serait libéré si le dataset était détruit.

usedbyrefreservation La quantité d'espace utilisé par un **refreservation** définis dans ce dataset, qui serait libéré si **refreservation** était supprimé.

usedbysnapshots La quantité d'espace consommé par les snapshots de ce dataset. En particulier, c'est la quantité d'espace qui aurait été libéré si tous les snapshots de ce dataset étaient supprimés. Noter que ce n'est pas simplement la somme des propriétés **used** des snaphots parce que l'espace peut être partagé par plusieurs snapshots.

userused@user La quantité d'espace consommé par l'utilisateur spécifié dans ce dataset. L'espace est calculé pour le propriétaire de chaque fichier, tel qu'affiché par **ls -l**. La quantité d'espace est affiché par **du** et **ls -s**.

Les utilisateurs non-privilégiés peuvent accéder seulement à leur propre espace. L'utilisateur root, ou un utilisateur qui a obtenu le privilège **userused** avec **zfs allow**, peut accéder à l'utilisation de tout le monde.

userused@... n'est pas affiché par **zfs get all**. Le nom de l'utilisateur doit être ajouté après les @, en utilisant un des formes suivantes :

- Nom POSIX (ex : joe)
- ID numérique POSIX (ex : 789)
- Nom SID (ex : joe.smith@mydomain)
- ID numérique SID (ex : S-1-123-456-789)

userrefs Cette propriété est définie au nombre d'utilisateurs dans ce snapshot, définis par la commande **zfs hold**.

groupused@group La quantité d'espace consommée par le groupe spécifié dans ce dataset. L'espace est calculé au groupe de chaque fichier, comme affiché par la commande **ls -l**. Les utilisateurs non-privilégiés peuvent seulement accéder à l'espace de leur propre groupe. L'utilisateur root, ou un utilisateur qui a obtenu le privilège **groupused** peuvent accéder à l'utilisation de tous les groupes.

volblocksize=blocksize Pour les volumes, spécifie la taille de block du volume. **blocksize** en peut pas être changé une fois le volume écrit, donc il doit être définis à la création. Défaut : 8Koctets. Tout puissance de 2 de 512o à 128Ko sont valides.

Les propriété natives suivantes peuvent être utilisée pour changer le comportement d'un dataset :

aclinherit=discard | noallow | restricted | passthrough | passthrough-x Contrôle comment les entrées d'ACL sont hérités quand des fichiers et des répertoires sont créés. Un système de fichier avec une propriété **aclinherit** à **discard** n'hérite d'aucune entrée ACL. Un système de fichier avec une propriété **aclinherit** à **noallow** hérite uniquement des ACL héritables qui spécifient des permissions "deny". La valeur "restricted" (le défaut) supprime les permissions **write_acl** et **write_owner** quand l'entrée d'ACL est héritée. Un système de fichier avec une propriété **aclinherit** à **passthrough** hérite de toutes les entrées d'ACL héritables sans modification. **passthrough-x** à la même signification, excepté que les ACE **owner@**, **group@** et **everyone@** héritent des permissions d'exécution uniquement si le mode de création de fichier demande également le bit d'exécution.

Quand la valeur de propriété est définie à **passthrough** les fichiers sont créés avec un mode déterminé par les ACE héritables. Si aucune ACE héritable n'existe qui affecte ce mode, alors le mode est définis en accord avec le mode demandé depuis l'application.

aclmode=discard | groupmask | passthrough Contrôle comment une ACL est modifiée durant un chmod. Un système de fichier avec une propriété **aclmode** à **discard** supprime toutes les entrées ACL qui ne représentent pas le mode du fichier. Une propriété **aclmode** a **groupmask** (le défaut) réduit les permissions utilisateur ou groupe. Les permission sont réduit, tel qu'ils ne sont pas supérieurs que les bits de permissions du groupe, sauf si c'est une entrée utilisateur qui a le même UID que le propriétaire du fichier ou répertoire. Dans ce cas, les permissions d'ACL sont réduites pour qu'elle ne soient pas supérieurs aux bits de permission du propriétaire. Un système de fichier avec une propriété **aclmode** à **passthrough** indique qu'aucun changement n'est fait aux ACL autre que générer les entrées ACL nécessaires pour représenter le nouveau mode du fichier ou répertoire.

atime=on | off Contrôle si la date d'accès pour les fichiers est mis à jours quand ils sont lus. Désactiver cette propriété évite de produire du trafic d'écriture en lisant des fichiers et peut augmenter les performances d'écritures. Défaut : on.

canmount=on | off | noauto À off, le système de fichier ne peut pas être monté, et est ignoré par **zfs mount -a**. C'est similaire à définir **mountpoint** à **none**, excepté que le dataset aura une propriété mountpoint définie, qui peut être hérité. Définir cette propriété à **off** permet aux datasets d'être utilisé seulement comme mécanisme d'héritage de propriété. Un exemple est d'avoir 2 datasets avec le même nountpoint, donc l'enfant des 2 datasets apparaît dans le même répertoire, mais peut avoir des caractéristiques hérités différents.

Quand l'option **noauto** est définis, un dataset peut seulement être monté de démonté explicitement. Le dataset n'est pas monté automatiquement quand le dataset est créé ou importé, ni n'est monté par **zfs mount -a** ou démonté par **zfs umount -a**. Cette propriété n'est pas héritée.

checksum=on | off | fletcher2,| fletcher4 | sha256 Contrôle le checksum utilisé pour vérifier l'intégrité des données. La valeur par défaut est **on**, qui sélectionne automatiquement un algorithme approprié (actuellement, **fletcher4**). Changer cette propriété n'affecte que les données nouvellement écrites.

compression=on | off | lzjb | gzip | gzip-N | zle Contrôle l'algorithme de compression utilisé pour ce dataset. L'algorithme **lzjb** est optimisé pour les performances tout en proposant un taux de compression correct. **on** équivaut à **lzjb**. Changer cette propriété n'affecte que les données nouvellement écrites.

copies=1 | 2 | 3 Contrôle le nombre de copies de données stockées pour ce dataset. Ces copies sont en plus de toute redondances fournies par le pool. Les copies sont stockée sur différents disques, si possible. Changer cette propriété n'affecte que les données nouvellement écrites. Cependant, définir cette propriété à la création du système de fichier en utilisant l'option **-o copies=N**

dedup=on | off | verify | sha256 [,verify] Contrôle si la déduplication est en effet pour un dataset. Défaut : off. Le checksum utilisé pour la déduplication est sha256. Quand activé, l'algorithme de checksum écrase la propriété **checksum**. Définir la valeur à **verify** est équivalent à spécifier **sha256,verify**. À verify, quand 2 blocks ont la même signature, ZFS fait une comparaison supplémentaire bit à bit.

devices=on | off Contrôle si les périphérique peuvent être ouvert dans ce système de fichier. Défaut : on.

exec=on | off Contrôle si les processus peuvent être exécutés depuis ce système de fichier. Défaut : on.

mlslabel=label | none Label de sensibilité qui détermine si un dataset peut être monté dans une zone dans le système avec Trusted Extensions activé. Si le dataset labelisé correspond à la zone labelisée, la dataset peut être monté et accédé depuis la zone labélisée.

Quand cette propriété n'est pas définie, la valeur par défaut est none. Définir à none est équivalent à supprimer cette propriété.

Cette propriété peut être modifiée seulement quand Trusted Extensions est activé et seulement avec les privilèges appropriés. Les droits de le modifier ne peut pas être délégué. En changeant un label à un label supérieur ou en définissant le label initial, le privilège **{PRIV_FILE_UPGRADE_SL}** est requis. En changeant un label à un label inférieur ou à none, le privilège **{PRIV_FILE_DOWNGRADE_SL}** est requis. Changer le dataset à des labels autre que none peut être fait seulement quand le dataset n'est pas monté. Quand un dataset avec le label none est monté dans une zone labélisée, l'opération de montage définis automatiquement le mlslabel au label de cette zone.

mountpoint=path | none | legacy Contrôle le point de montage utilisé pour ce système de fichier. quand mountpoint est changé pour un système de fichier, le système de fichier et ses enfants qui héritent du point de montage sont démontés. Si la nouvelle valeur est **legacy**, alors ils restent démontés. Sinon, ils sont automatiquement remontés dans le nouvel emplacement si la propriété était précédemment **legacy** ou **none**, ou s'ils étaient montés avant que la propriété a été changée. En plus, tous système de fichiers partagé sont départagés et partagés dans le nouvel emplacement.

nbmand=on | off Contrôle si le système de fichier devrait être monté avec **nbmant** (Non Blocking mandatory locks). C'est utilisé pour les clients CIFS. Changer cette propriété prend effet seulement quand le système de fichier est démonté et remonté.

primarycache=all | none | metadata Contrôle ce qui est caché dans le cache primaire (ARC). Si cette propriété est à **all**, les données utilisateur et les métadonnées sont cachées. Si cette propriété est à **none**, rien n'est caché. Si cette propriété est définie à **metadate**, seul les metadate sont cachés. Défaut : all.

quota=size | none Limite la quantité d'espace qu'un dataset et ses descendants peuvent consommer. Cette propriété force une limite hard dans la quantité d'espace utilisée. Cela inclus tout l'espace consommé par ses descendants, incluant les systèmes de fichiers et les snapshots. Définir un quota dans un descendant d'un dataset qui a déjà un quota n'écrase par le quota de l'ancêtre, mais impose une limite additionnelle. Les quota ne peuvent pas être définis dans les volumes, vu que **volsize** agit comme quota implicite.

userquota@user=size | none Limite la quantité d'espace consommée par l'utilisateur spécifié. Similairement à la propriété **refquota**, le calcul de l'espace **userquota** n'inclus pas l'espace qui est utilisé par les datasets descendant, tel les snapshots et les clones. La consommation d'espace de l'utilisateur est identifié par la propriété **userspace@suser**.

Forcer les quotas utilisateur peut être retardé de quelques secondes. Ce délai signifie qu'un utilisateur peut excéder son quota avec que le système notifie qu'il est hors quota. Le système va commencer à refuser les écritures additionnelles avec le message d'erreur **EDQUOT**.

Les utilisateurs non-privilégiés peut seulement accéder à l'utilisation de l'espace de leur propre groupe. L'utilisateur root, ou un utilisateur qui a obtenu le privilège **userquota** avec **zfs allow** peut obtenir et définir le quota pour tout le monde.

Cette propriété n'est pas disponible pour les volumes, et dans les systèmes de fichier avant v4 et les pools avant v15. Les propriétés userquota ne sont pas affichés par **zfs get all**. Le nom de l'utilisateur doit être ajouté après les @, en utilisant un des formes suivantes :

groupquota@group=size|none Limite la quantité d'espace consommé par le groupe spécifié.

readonly=on|off Contrôle si le dataset peut être modifié. Défaut : off.

recordsize=size Spécifie une taille de block suggérée pour les fichiers dans le système de fichier. Cette propriété est conçue pour être utilisé avec des bases de données qui accèdent à des enregistrement de taille fixe.

refquota=size|none Limite la quantité d'espace qu'un dataset peut consommer. Cette propriété force une limite hard sur la quantité d'espace utilisé. Cette limite n'inclue pas l'espace utilisé par les descendants, incluant les systèmes de fichier et les snapshot.

refreservation=size|none La quantité minimum d'espace garantit pour un dataset, n'incluant pas ses descendants. Quand la quantité d'espace utilisé est inférieur à cette valeur, le dataset est traité comme s'il prenait cette espace. Si **refreservation** est définis, un snapshot est seulement permis s'il y a suffisamment d'espace disponible dans le pool en dehors de cette réservation.

reservation=size|none Quantité minimum garantie pour un dataset et ses descendants. Quand la quantité d'espace utilise est inférieur à cette valeur, le dataset est traité comme s'il prenait cet espace.

secondarycache=all|none|metadata Contrôle ce qui est mis en cache secondaire (L2ARC). Si cette propriété est définie à **all**, les données utilisateurs et les métadonnées sont cachés.

setuid=on|off Contrôle si le bit set-uid est respecté pour le système de fichier. Défaut : on

shareiscsi=on|off Comme **sharenfs**, indique si un volume ZFS est exporté comme target iSCSI. Les valeurs peuvent être on, off, et **type=disk**. Défaut : off.

sharesmb=on|off|opts Contrôle si le système de fichier est partagé en utilisant le service Solaris CIFS, et quelles options sont utilisée.

sharenfs=on|off|opts Contrôle si le système de fichier est partagé via nfs, et quelles options sont utilisée. Un système de fichier avec la propriété **sharenfs** à **off** est géré via les outils traditionnels tel share, unshare, et dfstab. Sinon, le système de fichier est automatiquement partagé via **zfs share** et **zfs unshare**. Quand **sharenfs** est changé pour un dataset, le dataset et ses enfants héritant de la propriété sont partagés avec de nouvelles options, seulement si la propriété était précédemment **off**, ou s'il étaient partagés avant que la propriété soit changé.

logbias = latency | throughput Fournis une astuce à ZFS sur la manipulation des requêtes dans ce dataset. Si **logbias** est à **latency** (par défaut), ZFS utilise les périphériques de log du pool (si configuré) pour manipuler les requête à faible latence. À **throughput**, ZFS n'utilise pas les périphérique de log du pool. À la place, ZFS optimise les opérations synchrone pour le pool et l'utilisation efficace des ressources.

snapdir=hidden | visible Contrôle si le répertoire **.zfs** est caché ou visible dans le root du système de fichier. Défaut : hidden.

version=1 | 2 | current La version de ce système de fichier, qui est indépendant de la version du pool. Cette propriété peut seulement être définie pour les dernières versions supportées.

volsize=size Pour les volumes, spécifie la taille logique du volume. Par défaut, créer un volume établis une réservation de taille égale. Pour les pools de stockage avec un numéro de version 9 ou plus, un **refreservation** est définis à la place.

vscan=on | off Contrôle si les fichiers régulier devraient être scannés pour les virus quand un fichier est ouvert et fermé. En plus d'activer ce service, le service de scan de virus doit également être activé. Défaut : off.

xattr=on | off Contrôle si les attributs étendus sont activés pour ce système de fichier. Défaut : on

zoned=on | off Contrôle si le dataset est géré depuis une zone non-globale. Défaut : off.

Les 3 propriétés suivantes ne peuvent pas être changée après la création du système de fichier. Si les propriétés ne sont pas définies avec les commandes **zfs create** et **zpool create**, ces propriété sont héritées du dataset parent.

casesensitivity=sensitive | insensitive | mixed Indique si l'algorithme de correspondance de nom de fichier utilisé par le système de fichier devrait être sensible à la casse.**mixed** indique que le système de fichier supporte les requêtes pour les 2 cas. Traditionnellement, les systèmes de fichiers UNIX et POSIX ont des noms de fichier sensible à la casse.

normalization = none | formC | formD | formKC | formKD Indique si le système de fichier devrait effectuer une normalisation unicode des noms de fichier quand 2 noms de fichier sont comparés. Si cette propriété est définie à une valeur autre que **none** et que la propriété **utf8only** n'est pas spécifiée, cette dernière est automatiquement mis à **on**. Défaut : none.

utf8only=on | off Indique si le système de fichier devrait rejeter les noms de fichier qui incluent des caractères qui ne sont pas présents dans le jeu UTF8.

Propriétés de point de montage temporaire

Quand un système de fichier est monté, soit via **mount** ou via **zfs mount**, ses options de montage sont définis en accord avec ses propriétés. La corrélation entre les propriété et les options de montage sont comme suit :

```
PROPERTY__________MOUNT OPTION
```

```
devices_____________devices/nodevices
exec________________exec/noexec
readonly____________ro/rw
setuid______________setuid/nosetuid
xattr_______________xattr/noxattr
```

En plus, ces options peuvent être définis sur une base par montage en utilisant l'option **-o**, sans affecter la propriété qui est stockée sur disque. Les valeurs spécifiées sur la ligne de commande écrasent celle stockées dans le dataset. L'option **-nosuid** est un alias pour **nodevices,nosetuid**.

Propriétés utilisateur

En plus des propriétés natives standard, ZFS supporte les propriétés utilisateurs arbitraires. Les propriétés utilisateur n'ont pas d'effet sur le fonctionnement de ZFS, mais les applications et les administrateurs peut les utiliser pour annoter les datasets.

Les noms de propriété utilisateur doivent contenir un caractère ' :' pour les distinguer des propriétés natives. Ils peuvent contenis les lettre minuscules, des nombres, et les caractères : ' :', '-', '.', '_'. La convention attendue est que le nom de propriété est divisée en 2 portions tel que **module :property**, mais cet espace de nom n'est pas forcé par ZFS. Les noms de propriété utilisateur ne peuvent pas avoir plus de 256 caractères, et ne peuvent pas commencer par un '-'.

En utilisant des propriétés utilisateur, il est fortement suggéré d'utiliser reverse DNS pour le composant module pour réduire les chances que 2 packages indépendants utilisent le même nom de propriété pour différents buts. Les noms de propriété commençant avec **com.sun.** sont réservés.

Les valeurs des propriétés utilisateur sont des chaînes arbitraires, sont toujours hérités, et ne sont jamais validées. Tous les commandes qui opèrent sur les propriétés peuvent être utilisé pour manipuler les propriétés utilisateurs également. Utiliser la commande **zfs inherit** pour supprimer des propriétés utilisateur. Si la propriété n'est pas définie dans un dataset parent, il est supprimé entièrement. Les valeur de propriétés sont limitées à 1024 caractères.

Volumes ZFS comme Swap ou périphériques de Dump

Durant une installation initiale ou une mise à jours depuis un système de fichier UFS, un périphérique de swap et un périphérique de dump sont créés dans le pool root des volume ZFS. Par défaut, la taille de la zone de swap est basée sur la moitié de la mémoire physique jusqu'à 2Go. La taille du périphérique de dump dépend des prérequis du kernel à l'installation. Des volumes ZFS séparés doivent être utilisés pour les périphérique swap et dump. Ne pas swapper dans un fichier dans un système de fichier ZFS. Une configuration de fichier swap ZFS n'est pas supportée. Si vous devez changer votre zone de swap ou dump un fois le système installé, utiliser les commandes **swap** et **dumpadm**.

Sous Commandes

Toutes les sous-commandes qui modifient l'état sont loggés de manière persistante dans le pool sous leur forme originelle.

zfs create [-p] [-o property=value] ... filesystem Crée un nouveau système de fichie zfs. Le système de fichier est automatiquement monté en accord avec les propriété **mountpoint** héritées du parent.

-p Crée tous les datasets parent non-existant. les Datasets créés de cette manière sont automatiquement montés en accord avec le mountpoint hérité de leur parent. Toute propriété spécifié sur la ligne de commande est ignorée. Si le système de fichier cible existe déjà, l'opération se termine avec succès.

-o property=value Définis la propriété spécifiée. Peut être spécifié plusieurs fois

zfs create [-ps] [-b blocksize] [-o property=value] ... -V size volume Crée un volume à la taille donnée. Le volume est exporté comme périphérique block dans **/dev/zvol/**{**dsk,rdsk**}**/path**, où path est le nom du volume dans l'espace de nom zfs. La taille représente la taille logique tel qu'exporté par le périphérique. Par défaut, une réservation de taille égal est créée.

-p Crée tous les datasets parent non-existant. les Datasets créés de cette manière sont automatiquement montés en accord avec le mountpoint hérité de leur parent. Toute propriété spécifié sur la ligne de commande est ignorée. Si le système de fichier cible existe déjà, l'opération se termine avec succès.

-s Créer un volume sparse sans réservation.

-o property=value Définis la propriété spécifiée. Peut être spécifié plusieurs fois

-b blocksize Équivaleunt à **-o volblocksize=blocksize**.

zfs destroy [-rRf] filesystem|volume Détruit le dataset donné. Par défaut, la commande départage tous systèmes de fichier partagé, démonte tous les systèmes de fichier actuellement monté, et refuse de détruire un dataset qui a une dépendance active (enfant ou clone).

-r Détruit récursivement tout enfant

-R Détruit récursivement toutes les dépendances, incluant les clones en dehors de la hiérarchie cible.

-f Force à démonter les systèmes de ficiher en utilisant **umount -f**. N'a d'effet que sur les systèmes de fichier, et montés.

zfs destroy [-rRd] snapshot Détruit le snapshot donné si et seulement si la commande **zfs destroy** l'aurait détruit sans l'option **-d**. Si le snapshot n'est pas qualifié pour une destruction immédiate, il est marqué pour une suppression déferrée. Dans cet état, il est utilisable et visible jusqu'à ce que les conditions pour sa suppression soient rencontrées.

-d Défère la suppression

-r Détruit ou marque pour suppression tous les snapshots avec ce nom dans le système de fichier descendant.

-R Détruit récursivement toutes les dépendances.

zfs snapshot [-r] [-o property=value] ... filesystem@snapname|volume@snapname Créer un snapshot avec le nom donné. Toutes les modifications précédente dans le système de fichier font partie du snaphot.

-r Créer des snapshot récursivement pour tous les datasets descendants. Les snapshots sont pris automatiquement, donc tous les snapshots pris correspondent au même moment dans le temps.

-o property=value Définis la propriété spécifiée.

zfs rollback [-rRf] snapshot Applique un précédent snapshot a un dataset. Par défaut, la commande refuse d'appliquer un snapshot autre que le plus récent. Les options **-rR** ne détruisent pas récursivement les snapshots enfant. Seul le snapshot récurisf top-level est détruit par une de ces options. pour appliquer un snapshot récursif complet, il faut l'appliquer individuellement aux snapshots enfant.

-r Détruit récursivement tous snapshots plus récent que celui spécifié

-R Détruit récursivement tous des snapshots plus récent, ainsi que les clones de ces snapshots.

-f Utilisé avec -R force à démonter tous systèmes de fichier clone qui doivent être détruits

zfs clone [-p] [-o property=value] ... snapshot filesystem|volume Crée un clone d'un snapshot donné.

-p créé tous les datasets parent non-existants. Les datasets crées de cette manière sont automatiquement montés en accord avec la propriété **mountpoint** héritée de leur parent.

-o property=value Définis la propriété spécifiée.

zfs promote clone-filesystem Détache un clone de système de fichier de son snapshot original. Cela permet de détruire le système de fichier depuis lequel ce clone a été créé. La relation de dépendance de clone parent-enfant est réservée, donc le système de fichier d'origine devient un clone du système de fichier spécifié. Le snapshot qui a été cloné, et tous les snapshots précédents à ce snapshot, sont désormais possédés par le clone promus. L'espace qu'ils utilisent se déplace du système de fichier original avec le clone promus, il doit donc y avoir suffisamment d'espace pour ces snapshots. Aucun nouvel espace n'est consommé par cette opération, mais l'espace est ajusté. Le clone promu ne doit pas avoir de conflits de noms de snapshot.

zfs rename filesystem|volume|snapshot filesystem|volume|snapshot

zfs rename [-p] filesystem|volume filesystem|volume Renomme le dataset donné. La nouvelle cible peut être localisée n'importe où dans la hiérarchie ZFS, à l'exception des snapshots. Les snapshots peuvent seulement être renommés dans le système de fichier parent ou les volumes. En renommant un snapshot, le système de fichier parent du snapshot n'a pas besoin d'être spécifié comme partie du second argument. Les système de fichie renomés peuvent hériter de nouveaux points de montage, auquel cas ils sont démonté et remontés dans le nouveau point.

-p Crée tous les datasets parent non-existant. Les datasets ainsi créés de cette manière sont automatiquement montés en accord avec la propriété **mountpoint** hérité du parent.

zfs rename -r snapshot snapshot Renome récursivement les snapshots de tous les datasets descendants. Les snapshots sont les seuls datasets qui peuvent être renommés récursivement.

zfs list [-r|-d depth] [-H] [-o property[,...]] [-t type[,...]] [-s property] ... [-S property] ... [filesystem|volume|snapshot] ...L

-H Utilisé pour le scripting. N'affiche pas les en-têtes, et sépare les champs par une simple tabulation.

-r Affiche récursivement les enfants du dataset.

-d depth Affiche récursivement les enfants du dataset, en limitant la profondeur de la récursion.

-o property Liste de propriétés à afficher. Peut être une propriété native, utilisateur, **name** pour affichier le nom du dataset, et **space** pour afficher l'utilisation disque.

-s property Une propriété pour le trie dans l'ordre ascendant basée sur la valeur de la propriété..

-S property Idem mais dans l'ordre descendant.

-t type Liste de type à affichier (filesystem, snapshot, volume ou all).

zfs set property=value filesystem|volume|snapshot ... Définis la propriété à la valeur donnée pour chaque dataset.

zfs get [-r|-d depth] [-Hp] [-o all | field[,...] [-s source[,...]] all | property[,...] filesystem|volume|snapshot ... Affiche les propriété pour les datasets donné. si aucun dataset n'est donné, affiche les propriétés pour tous les datasets.

-r Affiche les propriétés des enfants du dataset.

-d depth Affiche récursivement les enfants du dataset, en limitant la profondeur de la récursion.

-H Sort dans un format plus facile à parser par les scripts

-o field Définis les champs à afficher, parmi : name,property,value,received,source,all

-s source Liste de sources à afficher. chaque source doit être un parmi : local,default,inherited,temporary,received,none

-p Affiche les nombres en valeurs parsable (exactes)

zfs inherit [-rS] property filesystem|volume|snapshot ... Efface les propriétés spécifiée, forcant l'héritage du parent.

-r Hérite récursivement de la propriété donnée pour tous les enfants

-S Revient à la valeur de propriété reçue si possible.

zfs upgrade [-v] affiche une liste de systèmes de fichiers qui ne sont pas à la dernière version

zfs upgrade [-r] [-V version] [-a | filesystem] Met à jours les systèmes de fichiers à une version plus récente. En générale, la version est dépendante de la version du pool.

-a Met à jours tous les systèmes de fichier dans tous les pools importés

filesystem Met à jours le système de fichier spécifié

-r Met à jours le système de fichier spécifié et tous ses descendants.

-V version Met à jours à la version spécifiée.

zfs userspace [-niHp] [-o field[,...]] [-sS field]... [-t type [,...]] filesystem | snapshot Affiche l'espace consommé, et les quotas, de chaque utilisateur dans le système de fichier spécifié ou le snapshot. Cela correspond aux propriétés userused@user et userquota@user

-n Affiche l'ID numérique au lieu du nom

-H Sort dans un format plus facile à parser par les scripts

-p Affiche les nombres en valeurs parsable (exactes)

-o field Définis les champs à afficher, parmi : type,name,used,quota. Défaut : affiche tous les champs

-s field Trie la sortie par ce champs. Peut être spécifié plusieurs fois.

-S field idem, mais trie dans l'ordre inverse

-t type[,...] Affiche seulement les types spécifiés du jeu suivant : all,posixuser,smbuser,posixgroup,smbgroup. Défaut : posixuser,smbuser

-i Traduit SID en POSIX ID.

zfs groupspace [-niHp] [-o field[,...]] [-sS field]... [-t type [,...]] filesystem | snapshot Affiche l'espace consommé, et les quotes, pour chaque groupe dans le système de fichier spécifié ou le snapshot. Cette sous commande est identique à **zfs userspace**, excepté que le type d'affichage pas défaut est **-t posixgroup,smbgroup**

zfs mount Affiche tous les systèmes de fichiers montés

zfs mount [-vO] [-o options] -a | filesystem Monte les systèmes de fichiers ZFS. Invoqué automatiquement durant le processus de démarrage

-o options liste d'options de montage à utiliser temporairement

-O montage overlay

-v Reporte la progression du montage

-a Monte tous les systèmes de fichiers zfs disponible.

filesystem Monte le système de fichier spécifié

zfs unmount [-f] -a | filesystem|mountpoint Démonte les systèmes de fichiers zfs. Invoqué automatiquement durant le processus d'arrêt

-f Force à démonter tous les systèmes

-a Démonte tous les systèmes de fichiers zfs

filesystem|mountpoint Démonte le système de fichier spécifié.

zfs share -a | filesystem Partage un système de fichier zfs

-a Partage tous les systèmes de fichier zfs. Invoqué automatiquement durant le processus de démarrage

filesystem Partage le systèmes de fichier en accord avec les propriétés sharenfs et sharesmb.

zfs unshare -a | filesystem|mountpoint Ne partage plus les systèmes de fichier zfs. Invoqué automatiquement durant le processus d'arrêt

-a dé-partage tous les systèmes de fichier zfs

filesystem|mountpoint dé-partage le système de fichier spécifié

zfs send [-DvRp] [-[iI] snapshot] snapshot Créé une représentation flux du second snapshot, qui est écris sur la sortie standard. La sortie peut être redirigée dans un fichier ou sur un système différent.

-D Effectue un traitement **dedup** sur le flux.

-i snapshot génère un flux incrémentale depuis le premier snapshot dans le second. La source incrémentale (le premier snapshot) peut être spécifié comme dernier composant du nom du snapshot (par exemple, la partie après le @), et il est assumé être du même système de fichier que le second snapshot. Si la destination est un clone, la source peut être le snapshot d'origine, qui doit être spécifié en entier (ex : pool/fs@origin).

-I snapshot Génère un package flux qui envoie tous les snapshots intermédiaire depuis le premier snapshot jusqu'au second.

-R Génère un package flux de réplication, qui va répliquer le système de fichier spécifié, et tous les systèmes de fichier descendants, jusqu'au snapshot nommé. Toutes les propriétés, snapshots, systèmes de fichiers descendants et clones sont préservés. Si -i et -I sont utilisé avec -R, un flux de réplication incrémentale est généré.

-p Envoie les propriétés

-v Affiche des informations sur le flux généré.

zfs receive [-vnFu] filesystem|volume|snapshot

zfs receive [-vnFu] [-d | -e] filesystem Crée un snapshot dont le contenu est spécifié dans le flux fournis sur l'entrée standard. Si un flux complet est reçu, le nouveau système de fichier est créé également.

Si un flux incrémentale est reçu, le système de destination doit déjà exister, et son snapshot le plus récent doit matcher la source du flux incrémentale. Pour les zvols, le lien du périphérique de destination est détruit et recréé, ce qui signifie que le zvol ne peut être accédé durant l'opération.

Quand un flux de réplication de snapshot généré par **zfs send -R** est reçu, tout snapshot qui n'existent pas dans l'emplacement d'origine est détruit via **zfs destroy -d**.

Le nom du snapshot (et du système de fichier, si un flux complet est reçu) que cette sous-commande créé dépends du type d'argument et des options -d ou -e.

Si l'argument est un nom de snapshot, il est créé. Si l'argument est un système de fichier ou un nom de volume, un snapshot avec le même nom que le snapshot envoyé est créé dans le système de fichier ou le volume spécifié. Si l'option -e ou -d est spécifiée, le nom du snapshot est déterminé en ajoutant le nom du snapshot au système de fichier spécifié. Si -d est spécifié, tous sauf le nom du pool du snapshot envoyé est ajouté. (ex : b/c@1 ajouté depuis a/b/c@1), et si -e est spécifié, seul la fin du chemin est ajouté (ex : c@1).

-d Utilise tout sauf le premier élément du chemin du snapshot envoyé pour déterminer le nom du nouveau snapshot.

Utilise le dernier élément du chemin du snapshot envoyé pour déterminer le nom du nouveau snapshot.

-u Le système de fichier qui est associé avec le flux reçu n'est pas monté

-v Affiche des informations sur le flux et le temps requis pour effectuer l'opération.

-n Ne reçoit pas le flux. Utile avec -v pour vérifier le nom.

-F Force le rollback du système de fichier au snapshot le plus récent avant d'effectuer l'opération.

zfs allow filesystem | volume Affiche les permissions qui ont été délégués dans le système de fichier spécifié ou le volume.

zfs allow [-ldug] "everyone"|user|group [,...] perm|@setname [,...] filesystem| volume

zfs allow [-ld] -e perm|@setname [,...] filesystem | volume Délègue les permissions d'administration ZFS pour les systèmes de fichier à des utilisateurs non-privilégiés.

[-ug] "everyone"|user|group [,...] Spécifie à que des autorisation sont déléguées. Plusieurs entités peuvent être spécifiées. Si -u et -g n'est pas spécifié, l'argument est interprété préférentiellement pour tout le monde, puis comme nom d'utilisateur, et enfun comme nom de groupe. Pour spécifier un utilisateur ou un groupe nommé "everyone", utiliser -u ou -g. Pour spécifier un groupe avec le même nom que l'utilisateur, utiliser l'option -g.

[-e] perm|@setname [,...] Spécifie les permissions à déléguer à "everyone". Plusieurs permissions peuvent être spécifiée. Les noms de permission sont les même que la même sous-commande ZFS ou nom de propriété. Voir la liste plus bas. Les noms de jeu de propriété, qui commencent par @, peuvent être spécifiés.

[-ld] filesystem|volume Spécifie où les permissions sont déployées. Si -l et -d n'est pas pécifié, ou les 2 le sont, les permissions sont permises pour le système de fichier ou le volume, et tous ses descendants. Il seul -l est utilisé, c'est permis localement seulement pour le système de fichier spécifié. Si seul -d est utilisé, c'est permis seulement pour les systèmes de fichiers descendants.

Les permissions sont généralement la capacité d'utiliser une sous-commande ou changer une propriété zfs. Les permissions suivantes sont permises :

```
NAME_____________TYPE__________ NOTES
allow____________subcommand____Doit également avoir la permission qui est permise
clone____________subcommand____Doit également avec la capacité 'create' et 'mount' dans le système de
fichier d'origine
create___________subcommand____Doit également avoir la capacité 'mount'
destroy__________subcommand____Doit également avoir la capacité 'mount'
hold_____________subcommand____Permet d'ajouter un utilisateur maintenu dans un snapshot
mount____________subcommand____Permet de monter/démonter des datasets
promote__________subcommand____Doit également avoir les capacités 'mount' et 'promote' dans le système de
fichie d'origine
receive__________subcommand____Doit également avoir les capacités 'mount' et 'create'
release__________subcommand____Permet d'enlever un user qui peut détuire le snapshot
rename___________subcommand____Doit également avec la capacité 'create' et 'mount' dans le nouveau parent
rollback_________subcommand
send_____________subcommand
share____________subcommand____Permet de partager des systèmes de fichier sur NFS ou SMB
snapshot_________subcommand
groupquota_______other_________Permet d'accéder à toutes les propriétés groupquota@...
groupused________other_________Permet de lire toutes les propriété groupused@...
userprop_________other_________Permet de changer toute propriété utilisateur
userquota________other_________Permet d'accéder à toutes les propriétés les userquota@...
userused_________other_________Permet de lire toutes les propriétés userused@...
aclinherit_______property
aclmode__________property
atime____________property
canmount_________property
casesensitivity__property
checksum_________property
compression______property
copies___________property
dedup____________property
devices__________property
exec_____________property
logbias__________property
mlslabel_________property
mountpoint_______property
```

```
nbmand__________property
normalization____property
primarycache_____property
quota____________property
readonly_________property
recordsize_______property
refquota_________property
refreservation___property
reservation______property
secondarycache___property
setuid___________property
shareiscsi_______property
sharenfs_________property
sharesmb_________property
snapdir__________property
utf8only_________property
version__________property
volblocksize_____property
volsize__________property
vscan____________property
xattr____________property
zoned____________property
```

zfs allow -c perm|@setname[,...] filesystem|volume Définis les permission "create time". Ces permission sont données localement au créateur d'un nouveau système de fichier descendant créé.

zfs allow -s @setname perm|@setname[,...] filesystem|volume Définis ou ajoute des permissions à un jeu de permission. Le jeu peut être utilisé par d'autres commande **zfs allow** pour le système de fichier spécifié et ses descendants. Les jeux sont évalués dynamiquement, donc changer un set est reflété dynamiquement. Les jeux de permission suivent les même restrictions de nommage que les systèmes de fichiers ZFS, mais le nom doit commencer par '@' et ne peut pas faire plus de 64 caractères.

zfs unallow [-rldug] "everyone"|user|group[,...] [perm|@setname[, ...]] filesystem|volume

zfs unallow [-rld] -e [perm|@setname [,...]] filesystem|volume

zfs unallow [-r] -c [perm|@setname[,...]] filesystem|volume Supprime les permissions qui ont été données avec la commande **zfs allow**. Aucune permissions n'est explicitement refusée, donc les permissions données restent effectives. Si aucune permission n'est spécifiée, toutes les permissions pour l'utilisateur, groupe, ou everyone sont supprimés. Spécifier everyone ou l'option -e supprime seulement les permissions qui ont été donnée à everyone.

-r Supprime récursivement les permissions dans le système de fichier et ses descendants

zfs unallow [-r] -s @setname [perm|@setname[,...]] filesystem|volume Supprime les permissions d'un jeu de permissions. Si aucune permission n'est spécifiée, supprime toutes les permissions et supprime le jeu.

zfs hold [-r] tag snapshot... Ajoute une référence simple, nommée avec l'argument tag, au snapshot ou aux snapshots spécifiés Chaque snapshot a son propre espace de nom de tag, et les tags doivent être unique dans cet espace. Si un hold existe dans un snapshot, tenter de détruire ce snapshot en utilisant **zfs destroy** retourne EBUSY.

-r Spécifie qu'un hold avec le tag donné est appliqué récursivement aux snapshots de tous les systèmes de fichiers descendants.

zfs holds [-r] snapshot... Liste toutes les références utilisateurs existant pour un snapshot donné

-r Liste les holds qui sont définis dans les snapshots descendants nommés, en plus de lister les holds dans le snapshot nommé.

zfs release [-r] tag snapshot... Supprime une simple référence, nommée avec l'argument tax, du ou des snapshots spécifiés. Le tag doit déjà exister.

-r Supprime récursivement un hold dans les snapshots et tous les système de fichier descendants.

Exemples

Créer une hiérarchie de système de fichier zfs :

zfs create pool/home
zfs set mountpoint=/export/home pool/home
zfs create pool/home/bob
Créer un snapshot ZFS. Ce snapshot est monté à la demande dans le .zfs/snapshot dans le système de fichier pool/home/bob :
zfs snapshot pool/home/bob@yesterday
Créer et détruire plusieurs snapshots :
zfs snapshot -r pool/home@yesterday
zfs destroy -r pool/home@yesterday
Désactive et active la compression de système de fichier
zfs set compression=off pool/home
zfs set compression=on pool/home/anne
Lister les datasets
zfs list
Définir un quota dans un système de fichier
zfs set quota=50G pool/home/bob
Lister les propriétés ZFS :
zfs get all pool/home/bob
Lister une simple propriété
zfs get -H -o value compression pool/home/bob
Liste toutes les propriété avec les paramètres locaux :
zfs get -r -s local -o name,property,value all pool/home/bob
Rollback un système de fichier
zfs rollback -r pool/home/anne@yesterday
Créer un clone :
zfs clone pool/home/bob@yesterday pool/clone
Détacher un clone :
zfs create pool/project/production
zfs snapshot pool/project/production@today
zfs clone pool/project/production@today pool/project/beta
zfs promote pool/project/beta
zfs rename pool/project/production pool/project/legacy
zfs rename pool/project/beta pool/project/production
zfs destroy pool/project/legacy
Hériter des propriétés ZFS :
zfs inherit checksum pool/home/bob pool/home/anne
Répliquer les données ZFS à distance :
zfs send pool/fs@a | ssh host zfs receive poolB/received/fs@a
poolB doit contenir le système de fichier poolB/received et ne doit pas contenir initialement pooB/received/fs
zfs send -i a pool/fs@b | ssh host zfs receive poolB/received/fs
Utiliser zfs receive -d
zfs send poolA/fsA/fsB@snap | ssh host zfs receive -d poolB/received
Définir les propriétés utilisateurs :
zfs set com.example :department=12345 tank/accounting
Créer un volume ZFS comme target iSCSI :
zfs create -V 2g pool/volumes/vol1
zfs set shareiscsi=on pool/volumes/vol1
iscsitadm list target
Maintenir un historique de snapshots dans un schéma de nommage consistant :
zfs destroy -r pool/users@7daysago
zfs rename -r pool/users@6daysago @7daysago
zfs rename -r pool/users@5daysago @6daysago
zfs rename -r pool/users@yesterday @5daysago
zfs rename -r pool/users@yesterday @4daysago
zfs rename -r pool/users@yesterday @3daysago
zfs rename -r pool/users@yesterday @2daysago
zfs rename -r pool/users@today @yesterday
zfs snapshot -r pool/users@today
Définis la propriété sharenfs dans un système de fichier zfs :
zfs set sharenfs='rw=@123.123.0.0/16,root=neo' tank/home

Déléguer les permissions d'administration dans un dataset :
zfs allow cindys create,destroy,mount,snapshot tank/cindys
zfs allow tank/cindys
Parce que les permissions du point de montage tank/cindys est à 755 par défaut, cindys ne sera pas capable de monter les systèmes de fichier sous tank/cindys. Définir une ACL similaire à la syntaxe suivante :
chmod A+user :cindys :add_subdirectory :allow /tank/cindys
Déléguer la permission create time dans un dataset :
zfs allow staff create,mount tank/users
zfs allow -c destroy tank/users
zfs allow tank/users
Définir et donner un jeu de permission dans un dataset :
zfs allow -s @pset create,destroy,snapshot,mount tank/users
zfs allow staff @pset tank/users
zfs allow tank/users
Déléguer les permissions de propriété dans un dataset :
zfs allow cindys quota,reservation users/home
zfs allow users/home
Supprimer les permissions déléguer dans un dataset :
zfs unallow staff snapshot tank/users
zfs allow tank/users