NAME

zfs-receive - create snapshot from backup stream

SYNOPSIS

zfs receive [**-FhMnsuv**] [**-o origin**=*snapshot*] [**-o** *property=value*] [**-x** *property*] *filesystem*|*volume*|*snapshot*
zfs receive [**-FhMnsuv**] [**-d**|**-e**] [**-o origin**=*snapshot*] [**-o** *property=value*] [**-x** *property*] *filesystem*
zfs receive -A *filesystem*|*volume*
zfs receive -c [**-vn**] *filesystem*|*snapshot*

DESCRIPTION

zfs receive [**-FhMnsuv**] [**-o origin**=*snapshot*] [**-o** *property=value*] [**-x** *property*] *filesystem*|*volume*|*snapshot*

zfs receive [**-FhMnsuv**] [**-d**|**-e**] [**-o origin**=*snapshot*] [**-o** *property=value*] [**-x** *property*] *filesystem*

Creates a snapshot whose contents are as specified in the stream provided on standard input. If a full stream is received, then a new file system is created as well. Streams are created using the **zfs send** subcommand, which by default creates a full stream. **zfs recv** can be used as an alias for **zfs receive**.

If an incremental stream is received, then the destination file system must already exist, and its most recent snapshot must match the incremental stream's source. For **zvols**, the destination device link is destroyed and recreated, which means the **zvol** cannot be accessed during the **receive** operation.

When a snapshot replication package stream that is generated by using the **zfs send -R** command is received, any snapshots that do not exist on the sending location are destroyed by using the **zfs destroy -d** command.

The ability to send and receive deduplicated send streams has been removed. However, a deduplicated send stream created with older software can be converted to a regular (non-deduplicated) stream by using the **zstream redup** command.

If **-o** *property=value* or **-x** *property* is specified, it applies to the effective value of the property throughout the entire subtree of replicated datasets. Effective property values will be set (**-o**) or inherited (**-x**) on the topmost in the replicated subtree. In descendant datasets, if the property is set by the send stream, it will be overridden by forcing the property to be inherited from the top-most file system. Received properties are retained in spite of being overridden and may be restored with **zfs inherit -S**. Specifying **-o origin**=*snapshot* is a special case because, even if **origin** is a read-only property and cannot be set, it's allowed to receive the send stream as a clone of the given snapshot.

Raw encrypted send streams (created with **zfs send -w**) may only be received as is, and cannot be re-

encrypted, decrypted, or recompressed by the receive process. Unencrypted streams can be received as encrypted datasets, either through inheritance or by specifying encryption parameters with the **-o** options. Note that the **keylocation** property cannot be overridden to **prompt** during a receive. This is because the receive process itself is already using the standard input for the send stream. Instead, the property can be overridden after the receive completes.

The added security provided by raw sends adds some restrictions to the send and receive process. ZFS will not allow a mix of raw receives and non-raw receives. Specifically, any raw incremental receives that are attempted after a non-raw receive will fail. Non-raw receives do not have this restriction and, therefore, are always possible. Because of this, it is best practice to always use either raw sends for their security benefits or non-raw sends for their flexibility when working with encrypted datasets, but not a combination.

The reason for this restriction stems from the inherent restrictions of the AEAD ciphers that ZFS uses to encrypt data. When using ZFS native encryption, each block of data is encrypted against a randomly generated number known as the "initialization vector" (IV), which is stored in the filesystem metadata. This number is required by the encryption algorithms whenever the data is to be decrypted. Together, all of the IVs provided for all of the blocks in a given snapshot are collectively called an "IV set". When ZFS performs a raw send, the IV set is transferred from the source to the destination in the send stream. When ZFS performs a non-raw send, the data is decrypted by the source system and re-encrypted by the destination system, creating a snapshot with effectively the same data, but a different IV set. In order for decryption to work after a raw send, ZFS must ensure that the IV set used on both the source and destination side match. When an incremental raw receive is performed on top of an existing snapshot, ZFS will check to confirm that the "from" snapshot on both the source and destination were using the same IV set, ensuring the new IV set is consistent.

The name of the snapshot (and file system, if a full stream is received) that this subcommand creates depends on the argument type and the use of the **-d** or **-e** options.

If the argument is a snapshot name, the specified *snapshot* is created. If the argument is a file system or volume name, a snapshot with the same name as the sent snapshot is created within the specified *filesystem* or *volume*. If neither of the **-d** or **-e** options are specified, the provided target snapshot name is used exactly as provided.

The **-d** and **-e** options cause the file system name of the target snapshot to be determined by appending a portion of the sent snapshot's name to the specified target *filesystem*. If the **-d** option is specified, all but the first element of the sent snapshot's file system path (usually the pool name) is used and any required intermediate file systems within the specified one are created. If the **-e** option is specified, then only the last element of the sent snapshot's file system name (i.e. the name of the source file system itself) is used as the target file system name.

-F Force a rollback of the file system to the most recent snapshot before performing the receive operation. If receiving an incremental replication stream (for example, one generated by **zfs send -R** [**-i**|**-I**]), destroy snapshots and file systems that do not exist on the sending side.

-d Discard the first element of the sent snapshot's file system name, using the remaining elements to determine the name of the target file system for the new snapshot as described in the paragraph above.

-e Discard all but the last element of the sent snapshot's file system name, using that element to determine the name of the target file system for the new snapshot as described in the paragraph above.

-h Skip the receive of holds. There is no effect if holds are not sent.

-M

Force an unmount of the file system while receiving a snapshot. This option is not supported on Linux.

-n Do not actually receive the stream. This can be useful in conjunction with the **-v** option to verify the name the receive operation would use.

-o origin=*snapshot*

Forces the stream to be received as a clone of the given snapshot. If the stream is a full send stream, this will create the filesystem described by the stream as a clone of the specified snapshot. Which snapshot was specified will not affect the success or failure of the receive, as long as the snapshot does exist. If the stream is an incremental send stream, all the normal verification will be performed.

-o *property=value*

Sets the specified property as if the command **zfs set** *property=value* was invoked immediately before the receive. When receiving a stream from **zfs send -R**, causes the property to be inherited by all descendant datasets, as through **zfs inherit** *property* was run on any descendant datasets that have this property set on the sending system.

If the send stream was sent with **-c** then overriding the **compression** property will have no effect on received data but the **compression** property will be set. To have the data recompressed on receive remove the **-c** flag from the send stream.

Any editable property can be set at receive time. Set-once properties bound to the received data, such as **normalization** and **casesensitivity**, cannot be set at receive time even when the datasets are

newly created by **zfs receive**. Additionally both settable properties **version** and **volsize** cannot be set at receive time.

The **-o** option may be specified multiple times, for different properties. An error results if the same property is specified in multiple **-o** or **-x** options.

The **-o** option may also be used to override encryption properties upon initial receive. This allows unencrypted streams to be received as encrypted datasets. To cause the received dataset (or root dataset of a recursive stream) to be received as an encryption root, specify encryption properties in the same manner as is required for **zfs create**. For instance:

zfs send *tank/test@snap1* | **zfs recv -o encryption=on -o keyformat=passphrase -o keylocation=***file:///path/to/keyfile*

Note that **-o keylocation=prompt** may not be specified here, since the standard input is already being utilized for the send stream. Once the receive has completed, you can use **zfs set** to change this setting after the fact. Similarly, you can receive a dataset as an encrypted child by specifying **-x encryption** to force the property to be inherited. Overriding encryption properties (except for **keylocation**) is not possible with raw send streams.

-s If the receive is interrupted, save the partially received state, rather than deleting it. Interruption may be due to premature termination of the stream (e.g. due to network failure or failure of the remote system if the stream is being read over a network connection), a checksum error in the stream, termination of the **zfs receive** process, or unclean shutdown of the system.

The receive can be resumed with a stream generated by **zfs send -t** *token*, where the *token* is the value of the **receive_resume_token** property of the filesystem or volume which is received into.

To use this flag, the storage pool must have the **extensible_dataset** feature enabled. See zpool-features(7) for details on ZFS feature flags.

-u File system that is associated with the received stream is not mounted.

-v Print verbose information about the stream and the time required to perform the receive operation.

-x *property*

Ensures that the effective value of the specified property after the receive is unaffected by the value of that property in the send stream (if any), as if the property had been excluded from the send stream.

If the specified property is not present in the send stream, this option does nothing.

If a received property needs to be overridden, the effective value will be set or inherited, depending on whether the property is inheritable or not.

In the case of an incremental update, **-x** leaves any existing local setting or explicit inheritance unchanged.

All **-o** restrictions (e.g. set-once) apply equally to **-x**.

zfs receive -A *filesystem|volume*
Abort an interrupted **zfs receive -s**, deleting its saved partially received state.

zfs receive -c [**-vn**] *filesystem|snapshot*
Attempt to repair data corruption in the specified dataset, by using the provided stream as the source of healthy data. This method of healing can only heal data blocks present in the stream. Metadata can not be healed by corrective receive. Running a scrub is recommended post-healing to ensure all data corruption was repaired.

It's important to consider why corruption has happened in the first place. If you have slowly failing hardware - periodically repairing the data is not going to save you from data loss later on when the hardware fails completely.

EXAMPLES

Example 1: Remotely Replicating ZFS Data
The following commands send a full stream and then an incremental stream to a remote machine, restoring them into *poolB/received/fs@a* and *poolB/received/fs@b*, respectively. *poolB* must contain the file system *poolB/received*, and must not initially contain *poolB/received/fs*.

```
# zfs send pool/fs@a |
    ssh host zfs receive poolB/received/fs@a
# zfs send -i a pool/fs@b |
    ssh host zfs receive poolB/received/fs
```

Example 2: Using the **zfs receive -d** Option
The following command sends a full stream of *poolA/fsA/fsB@snap* to a remote machine, receiving it into *poolB/received/fsA/fsB@snap*. The *fsA/fsB@snap* portion of the received snapshot's name is determined from the name of the sent snapshot. *poolB* must contain the file system *poolB/received*. If *poolB/received/fsA* does not exist, it is created as an empty file system.

```
# zfs send poolA/fsA/fsB@snap |
    ssh host zfs receive -d poolB/received
```

SEE ALSO

zfs-send(8), zstream(8)