



# Using btrfs Snapshots for Full System Rollback

**Matthias G. Eckermann**

Senior Product Manager

[mge@suse.com](mailto:mge@suse.com)

Enterprise End User Summit, New York, June 2014  
2014-06-20 15:44 UTC



Why this?

# Minimizing Downtime

**RAS**

**System  
Rollback**



**High Availability**

**Live Kernel  
Patching**

# Minimizing Downtime

RAS



High Availability

Live Kernel  
Patching



# System Rollback

## Reduce *Operational* Downtime

Goal: Go back to **well-known** system state

Peace of mind for:

- System administrative tasks
- Package and patch installation
- System upgrades

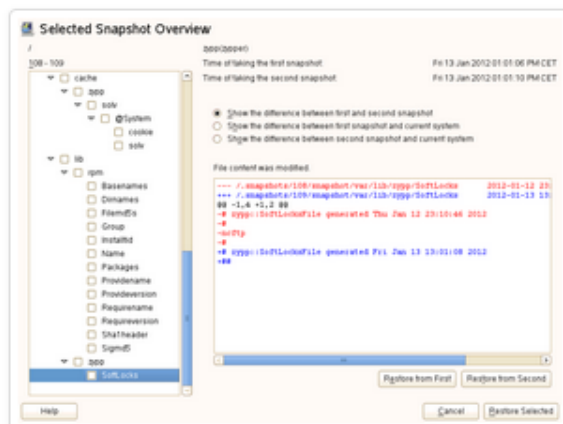
Introducing “snapper”

# Travel back in time and compare...

The ultimate snapshot tool for Linux

Download Binaries »

Download Source »



## Watch it in action

Greg Kroah-Hartman and Matthias Eckermann play sysadmins and ruin a web server configuration.



## Contribute

Snapper is open source. Port it to your distribution or integrate it with an application.

Fork us on GitHub »

<http://www.snapper.io/>



# Snapper – Btrfs integration

## Basic integration into

- Installer
  - Btrfs as root fs
  - Recommendation for subvolume layout
- Partitioner
  - Create Btrfs
  - Create subvolumes

## Tools

- Snapper
  - manage snapshots
  - Automatically create snapshots
  - Display differences between snapshots
  - Roll-back
  - User Interfaces
    - CLI
    - DBUS
    - Graphical (YaST)
  - In development (GSoC)
    - GUI integration



# Introducing “Snapper” – CLI example

## Snapper headers:

- **Type** : [ Pre | Post | Single ]
- **#** : Nr of snapshot
- **Pre #** : if type is “Post” the matching Pre nr.
- **Date** : timestamp
- **Cleanup** : cleanup algorithm for this snapshot
- **Description** : A fitting description of the snapshot (free text)
- **Userdata** : key=value pairs to record all sorts of useful information about the snapshot in an easily parsable format

# Special Use case: Snapper and ITIL

## # @Begin of implementation Change:

```
snapper create \  
  --type pre \  
  --description "ChgMgt Work order: Upgrade syslog  
configuration to forward log entries to central log  
server" \  
  --userdata \  
"WorkOrder=201201253030000012-1,  
State=InProgress, Agent=jdoe@example.com"
```

## # @End of implementation Change:

```
snapper create \  
  --type post --pre-number 240 \  
  --description "Done: ChgMgt Work order: Upgrade syslog  
configuration to forward log entries to central log  
server" \  
  --userdata "WorkOrder=201201253030000012-1,  
State=Closed, Agent=jdoe@example.com"
```



Rollback

# File based Rollback

## How it works

- The system uses the same instance of a subvolume: “working instance”
- single files are copied from the snapshot to the “working instance” – using CoW

## Benefits

- Subvolumes are treated as read-only
- Subvolumes can be used for Backup
- Supports Pick and Choose

## Disadvantages

- rollback not “atomic”

→ Implemented in Snapper as “undochange”



# Rollback per Subvolume

## How it works

- Instead of the original subvolume, the snapshot is mounted with the options “subvol=<name>”
  - Remember: snapshots are subvolumes
- “btrfs subvolume set-default ...” for permanent assignments

## Benefits

- “atomic” operation
- Very fast

## Disadvantages

- Additional complexity
  - May require explicit mounting of subvolumes
- No “rollback” per single file

→ “Full System Rollback”



# Snapshot/Rollback – Overview

## Past & Present

- “snapper undochange”
- Selective Rollback for
  - Package updates
  - Administrative changes
- No rollback of
  - Kernel / initrd
  - Bootloader
  - System data, e.g. /var/log

## Present & Future

- “snapper rollback”
- Full Rollback for
  - Package updates
  - Administrative changes
  - **Kernel / initrd (initramfs)**
- No rollback of
  - Bootloader
  - Customer data: “/home”, if on own partition (default)
  - System data, e.g. /var/log

High Demand



# Full System Rollback



# Full System Rollback – Use Cases

- “I have changed something, the system is still working, but now some functionality is missing / performance is bad / ...”
  - reset the system, reboot, enjoy!
  - “Reboot *later* mode”
- “Something changed, and the system is not booting anymore” (worst case scenario)
  - need immediate reboot
  - “Reboot *now* mode”

# Reboot *Later* Mode

- Administrator is in a current read-write filesystem, but wants to rollback
- “snapper list” to view and select a snapshot
- Call

“snapper rollback <number>”

this will

- Create a new read-only snapshot of the currently running system
- Create a new read-write snapshot of the snapshot <number>, lineary after the just recently created read-only snapshot
- “setdefault” to the new read-write snapshot

and reboot

# Reboot Now Mode

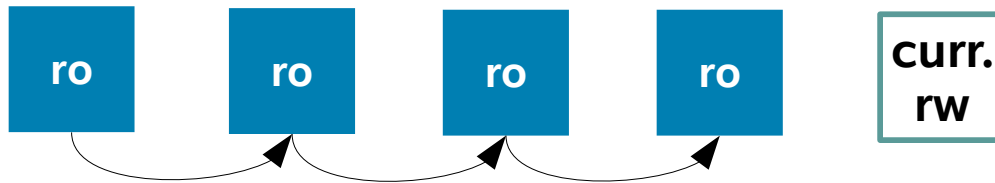
- Boot into an existing read-only snapshot
- The system should work, as all variable data are on writable subvolumes anyways.
- To continue to work in this snapshot, the admin should call
  - “snapper rollback”
  - this will
    - Create a new read-only snapshot of the old read-write one, linearly after the last existing one
    - Create a new read-write snapshot of the snapshot you are currently in in read-only mode, linearly after the just recently created read-only snapshot
    - “setdefault” to the new read-write snapshot

and reboot

# User view on Snapshot History

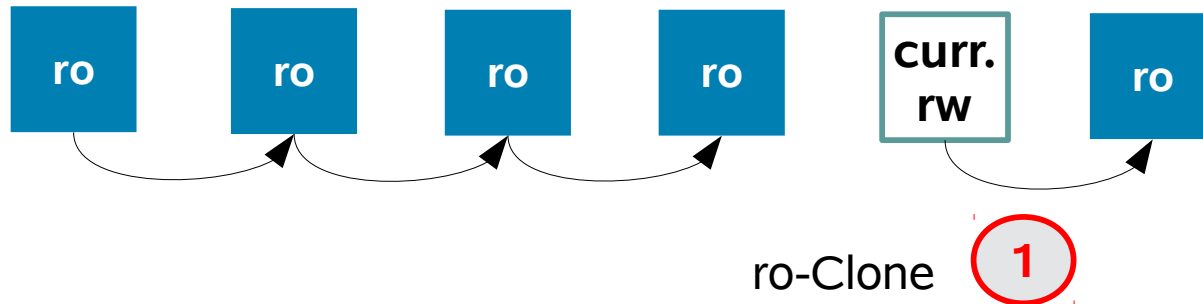
Snapshot / Rollback

# User view on Snapshot History (1)

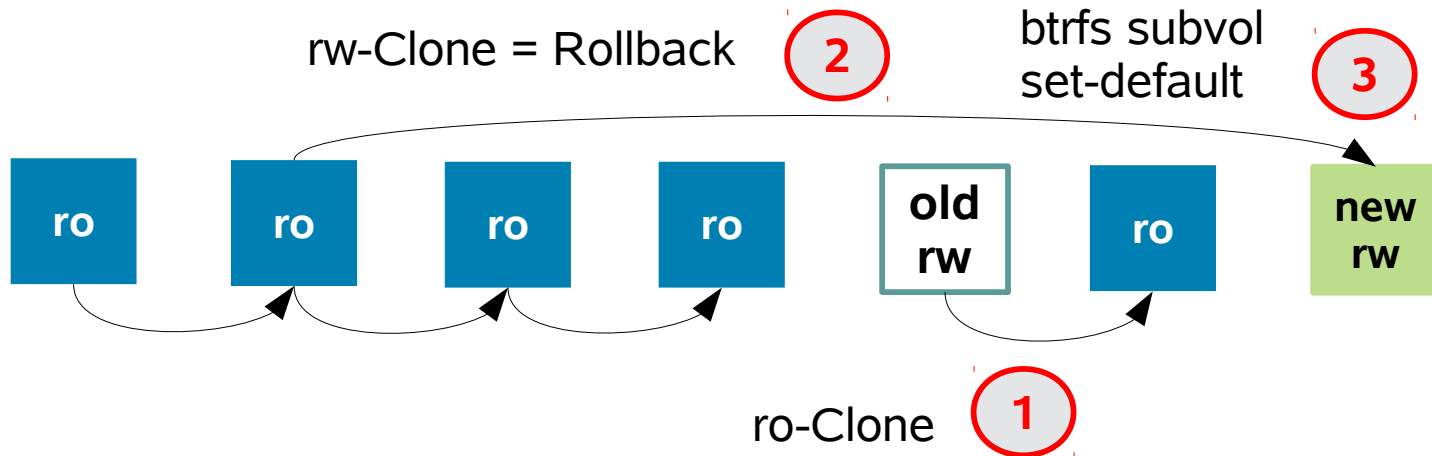


## Snapshot / Rollback

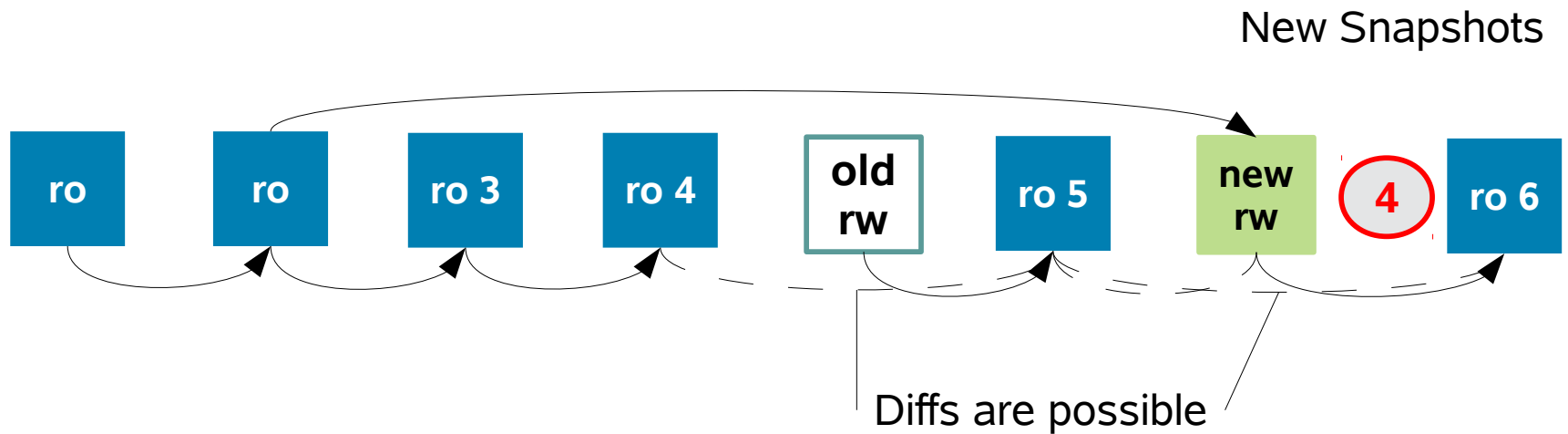
# User view on Snapshot History (2)



# User view on Snapshot History (3)



# User view on Snapshot History (4)



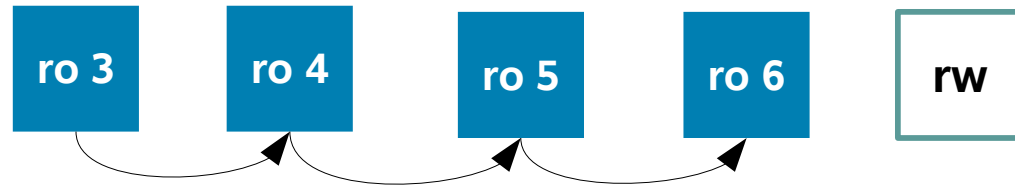


Snapshot / Rollback

# User view on Snapshot History (5)

Condensed view

What happens,  
if we rollback again?

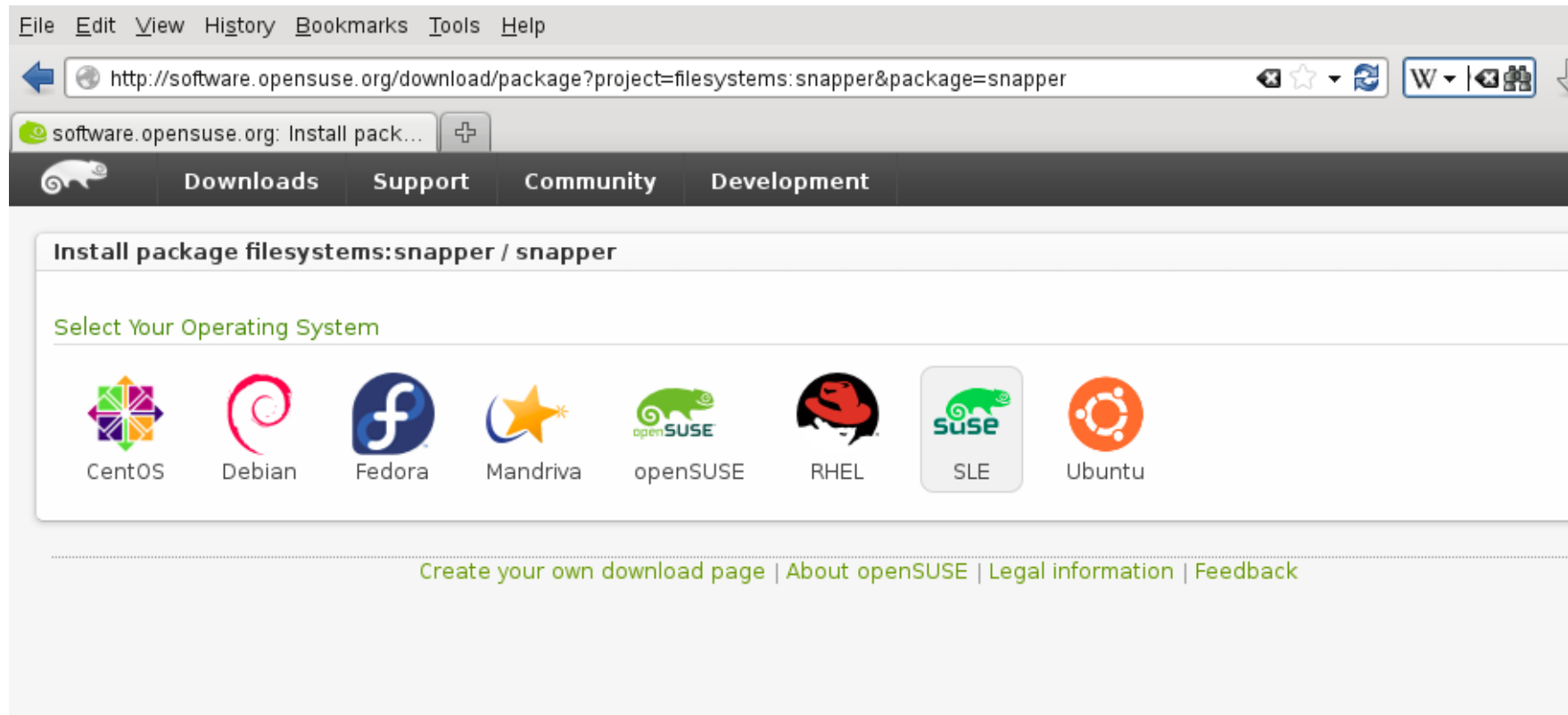


Caveat: this does not reflect 1:1 what happens technically.



Summary

# “Snapper” availability ...



The screenshot shows a web browser window with the URL `http://software.opensuse.org/download/package?project=filesystems:snapper&package=snapper`. The page title is "Install package filesystems:snapper / snapper". Below the title, there is a section titled "Select Your Operating System" with a grid of icons for different operating systems: CentOS, Debian, Fedora, Mandriva, openSUSE, RHEL, SLE (highlighted), and Ubuntu. At the bottom of the page, there are links for "Create your own download page", "About openSUSE", "Legal information", and "Feedback".

<http://software.opensuse.org/download/package?project=filesystems:snapper&package=snapper>



# System Rollback

## Reduce *Operational* Downtime

Goal: Go back to **well-known** system state

Peace of mind for:

- System administrative tasks
- Package and patch installation
- System upgrades

Go ahead, try btrfs and snapper today!

Your questions!?

Thank you.



Demo “scripts”

# Demonstration #1

## Using Snapshots with YaST2 snapper

#	Description	Tools
1	Track and visualize YaST activities with snapper	Start YaST2 Create a new user (yast users) Start YaST snapper View the changes
2	Undo some of the changes	Start YaST snapper Select the snapshot pair created by the “yast users” module
3	Install a new package by either using YaST2 and visualize the result using YaST2 snapper	YaST2

### Functions

- Automatic snapshots
- Integration with YaST and Zypp
- Rollback
- Hook scripts for integration
- YaST GUI and NCurses



# Demonstration#2

## btrfs subvols and snapshots

#	Description	Tools
1	List the currently available subvolumes	“btrfs subvol ...”
2	List the currently available subvolumes – display details about the “parents” of the subvolumes	“btrfs subvol ... -p ...”
3	Create your own subvolume “/mydata”	“btrfs subvol ...”



# Demonstration#3

## Snapper Command Line

#	Description	Tools
1	List the currently available snapshots	“snapper list ...”
2	Show, which files differ in the snapshot pair created by “yast users”	“snapper status ...”
3	Show the difference <i>only</i> in /etc/shadow in the snapshot pair created by “yast users”	“snapper diff ...”
4	Undo a change	remove unnecessary files in the user's home directory, e.g. “.xim.template” “snapper undochange ...”
5	Create your own snapshot and modify its description	“snapper create ... -d <description>” “snapper modify ... <num>”
6	Add a Key-Value pair to an existing snapshot	“snapper modify ... --userdata ... <num>”
7	Create your own snapshot pair	“snapper create --type pre ...” “snapper list” “snapper create --type post --pre-number <n1>...”
8	Change description and user data of some more of your snapshots	“snapper modify ... <num>”
9	Rollback the full system to a former state	“snapper rollback ... <num>   current”

# Demonstration#4

## Full System Rollback

#	Description	Tools
1	List the currently available snapshots	“snapper list”
2	Install a new kernel	“zypper in ...”
3	List the currently available snapshots	“snapper list”
4	Show the current kernel version and reboot	“uname -a ; reboot”
5	List the currently available snapshots	“snapper list”
6	Rollback to the former version and reboot	“snapper rollback <num> ; reboot”
7	List the currently available snapshots	“snapper list”
8	Show the current kernel version	“uname -a”

# Demonstration#5

## Migrating from ext3 to btrfs

#	Description	Tools
1	Create a logical volume or partition or loop device. Name it “toconvert” or the like.	
2	Create an ext3 filesystem on this “toconvert” and mount it to “/toconvert”	“mkfs.ext3 -b 4096 ...” “mkdir”, “mount”
3	Create some directories and files on “/toconvert”. Optional: create md5 checksums	“mkdir”, “vi” Optional: “md5sum”
4	Perform the conversion and mount the filesystem again	“umount /toconvert” “btrfs-convert ...” “mount ...”
5	Control, if your data are all there and check the optional md5sums	“find”, “md5sum”, ...
6	Understand, how btrfs saves the old ext3 filesystem metadata. “/toconvert/ext2_saved/image”	“mount -o loop ...”

# Demonstration#6

## Using snapper for /home/\$USER

### Requirements

- /home/\$USER is a btrfs subvolume
- “snapper” with DBUS interface (SLES 11 SP3, openSUSE 12.3, ...)
- Create a snapper configuration for the user allowing him/her to do snapshots

### Additional Option

- Automated snapshotting on login/logout
- Requires small changes to the configuration of Linux PAM (Pluggable Configuration Modules)



**Corporate Headquarters**  
Maxfeldstrasse 5  
90409 Nuremberg  
Germany

+49 911 740 53 0 (Worldwide)  
[www.suse.com](http://www.suse.com)

Join us on:  
[www.opensuse.org](http://www.opensuse.org)

## **Unpublished Work of SUSE. All Rights Reserved.**

This work is an unpublished work and contains confidential, proprietary and trade secret information of SUSE. Access to this work is restricted to SUSE employees who have a need to know to perform tasks within the scope of their assignments. No part of this work may be practiced, performed, copied, distributed, revised, modified, translated, abridged, condensed, expanded, collected, or adapted without the prior written consent of SUSE. Any use or exploitation of this work without authorization could subject the perpetrator to criminal and civil liability.

## **General Disclaimer**

This document is not to be construed as a promise by any participating company to develop, deliver, or market a product. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. SUSE makes no representations or warranties with respect to the contents of this document, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. The development, release, and timing of features or functionality described for SUSE products remains at the sole discretion of SUSE. Further, SUSE reserves the right to revise this document and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes. All SUSE marks referenced in this presentation are trademarks or registered trademarks of Novell, Inc. in the United States and other countries. All third-party trademarks are the property of their respective owners.

