The New Btrfs Filesystem for Linux: Features and Tools
Lecture
Legal Notices

Novell, Inc., makes no representations or warranties with respect to the contents or use of this documentation, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc., reserves the right to revise this publication and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes.

Further, Novell, Inc., makes no representations or warranties with respect to any software, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc., reserves the right to make changes to any and all parts of Novell software, at any time, without any obligation to notify any person or entity of such changes.

Any products or technical information provided under this Agreement may be subject to U.S. export controls and the trade laws of other countries. You agree to comply with all export control regulations and to obtain any required licenses or classification to export, re-export or import deliverables. You agree not to export or re-export to entities on the current U.S. export exclusion lists or to any embargoed or terrorist countries as specified in the U.S. export laws. You agree to not use deliverables for prohibited nuclear, missile, or chemical biological weaponry end uses. See the Novell International Trade Services Web page (http://www.novell.com/info/exports/) for more information on exporting Novell software. Novell assumes no responsibility for your failure to obtain any necessary export approvals.

Copyright © 2009 Novell, Inc. All rights reserved. No part of this publication may be reproduced, photocopied, stored on a retrieval system, or transmitted without the express written consent of the publisher.

Novell, Inc., has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more of the U.S. patents listed on the Novell Legal Patents Web page (http://www.novell.com/company/legal/patents/) and one or more additional patents or pending patent applications in the U.S. and in other countries.

Novell, Inc.
404 Wyman Street, Suite 500
Waltham, MA 02451
U.S.A.
www.novell.com

Online Documentation: To access the latest online documentation for this and other Novell products, see the Novell Documentation Web page (http://www.novell.com/documentation).

Novell Trademarks

For Novell trademarks, see the Novell Trademark and Service Mark list (http://www.novell.com/company/legal/trademarks/tmlist.html).

Third-Party Materials

All third-party trademarks are the property of their respective owners.
# Contents

**ATT Live SUS18**  The Btrfs Filesystem  4

SECTION 1:  Introduction to Btrfs  6

SECTION 2:  Btrfs Architecture  14

SECTION 3:  Btrfs Migration  20

SECTION 4:  Btrfs Copy on Write  24

SECTION 5:  Btrfs Subvolumes  28

SECTION 6:  Btrfs Snapshots and Rollback  39

SECTION 7:  Snapper  46
The Btrfs Filesystem

SUSE Linux Enterprise
9057 - Storage Technologies
Section 5

ATT Live
Session SUS18
Objectives

- Introduction to Btrfs
- Btrfs Architecture
- Btrfs Migration
- Btrfs Copy on Write
- Btrfs Subvolumes
- Btrfs Snapshots and Rollback
- Snapper
- Use Btrfs on Multiple Devices
Introduction to Btrfs
What is Btrfs?

- Rising star in the open source filesystem universe
- The Btrfs community is very active and hungry for additional support
- Moving forward with existing Ext data...
  
  ...Support for offline **in-place migration** from Ext2/3 -to-> Btrfs

Btrfs is the 'rising star' in the open source file system universe. It builds the foundation for the Ceph distributed filesystem and its RADOS object store layer for "cloud" technologies.

Btrfs encompasses ideas from the ext series of file systems (*ext2, ext3, ext4) as well as the Reiser file system, the XFS file system and the HP aufs file system.

Btrfs implements most of the same concepts as the Sun/Oracle ZFS but does so in a much more open ended manner that will allow much more headroom for development. Its architecture is much more scalable than ZFS. Btrfs was built with scalability and extensibility in mind.

Btrfs is very actively developed and greater stability, performance and new features are being added at a tremendous pace.

It is one of the key features enhancements of SLES 11 **SP2**.
History of the B-Tree File System (BtrFS)

- Linux correspondent for ZFS
- Based on the B-tree structure of XFS
- Developed by Chris Mason from Oracle since 2007
- Included in Linux kernel since version 2.6.29-rc1 (January 2009)
- Version 3.0 available since July 2011
- Technology preview in SLE 11 SP1, supported in SLE 11 SP2, default in SLE 12

Alternative acceptance of BtrFS: “Butter FS” or “Better FS”

ZFS was developed in 2006 by Sun but its license was incompatible to Linux.

Btrfs implements the same concepts as the Sun/Oracle ZFS but does so in a much more open ended manner that will allow for much more headroom for development. Its architecture is much more scalable than ZFS. Btrfs was built with scalability and extensibility in mind

Reiser4 FS also based on B-Tree

Chris Mason was ReiserFS developer at SUSE before he went to Oracle

Btrfs summarizes ideas from the ext series of file systems (ext2, ext3, ext4) as well as the Reiser file system, the XFS file system and the HP aufs file system.
How is Btrfs a "Better" File System

Solves Storage Challenges

- Massive Scalability
- Data Integrity
- Dynamic Expand and Shrink
- Built-in Volume Management
- Ubiquitous use: Server, Cloud-thru-Desktop, Even Mobile
- Cloud ready
Btrfs: The "Better" FS

- Built-in volume management with subvolumes
- Built-in copy on write
- Powerful snapshot capabilities
- Massive Scalability
  - 18.4 Exabytes (16EB w/ effective shrink)
    * See Notes
- Other Capabilities:
  - Compression
  - Built-in data integrity (checksums)
  - Desktops: SSD optimization (eg: MeeGo has chosen btrfs as default)
- Technology preview in SLE 11 SP1, supported in SLE 11 SP2, default in SLE 12.

The max file size in btrfs filesystem is truly massive at 16 Exabytes.

Here's a diagram for perspective:

```
18 446 744 073 709 551 616
```

```
^   ^   ^   ^   ^   ^   ^   ^
One Byte <---------- 8 bits
-------------Thousand Kilobyte
---------------Million Megabyte
-----------------Billion Gigabyte
-------------------Trillion Terabyte
----------------------Quadrillion Petabyte
--------------------------Quintillion Exabyte ......(ie, 18.4 Exabytes)
<=----------------------------Sextillion Zettabyte (or 18.4 BILLION )
<=--------------------------------Septillion Yottabyte ( GIGABYTES! )
```

Max scalability of Btrfs, which includes effective features, is 16EB.

Perhaps one should think of online shrink as "effective shrink", the filesystem keeps explicit back-references of each block. This way Btrfs can quickly determine which blocks need to be relocated from the device, which is requested to be removed.

This depends only on the number of allocated blocks on the device. There is no need to scan the whole device to find all the blocks.

Other solutions, such as Device Mapper, may have to scan the entire device to perform the same shrink action.

The Btrfs architecture makes online shrink considerably more effective.
There is a YaST Module for Btrfs

YaST partitioner should be used to create and manage btrfs filesystems to be in a supported state on SUSE Linux Enterprise systems.
Btrfs support

Btrfs is supported
- As “/” filesystem
- For migration from ext2/3/4 (except “/”)
- Everything you can do with the YaST partitioner is supported

Exceptions to official support then
- “/boot” as Btrfs
- RAID
- Integrated Volume Management
- Compression and Encryption

Recommendation for data volumes: xfs
- Performance and scalability are proven for 8+ years
- No extra costs for xfs in SUSE Linux Enterprise

Full text in the release notes:
http://www.suse.com/releasenotes/x86_64/SUSE-SLES/11-SP2/#fate-306585

FAQ
(from Q&A w/ Matthias Eckerman, SUSE PM)

1. Is there any limitation for copy on write in terms of transactions

THE DISK SPACE IS THE LIMITATION. As every change may need additional blocks, two sets of parameters must be chosen carefully according to the use case:
- Disk space available to the whole volume
- Automated cleanup rules

btrfs / snapshots in SUSE® Linux Enterprise 11 SP2
Question and Answer (2)

2. Is there a recommendation of how many % of free space should be kept?

In SUSE Linux Enterprise 11 SP2, the default for a root filesystem on btrfs is 20 GiB, while the system will need not more than 8 GiB usually. In other words, when starting there should be more than 70% free. For safe operation we suggest that a filesystem is not filled up to more than 90%, i.e. 10% free. Future development for snapper might introduce automated cleanup rules also based on free space or fill-level.
(FAQ cont.)

3. If the subvolumes you want to link to a particular “/” are identified with the “@” character (defining a namespace), how do we switch between (toggling the @)?

This is possible using the not too well documented mount option “subvolrootid”. One could use:

```
mount -o subvol=SINGLELEVELDIR,subvolrootid=NUMBER
```

Certainly, enabling this functionality in snapper in the future should put the burden into software, not onto the administrator. (There is a FATE request in for this feature)

4. Can the Copy on Write feature be used for replication between SANs over WAN links?

Please use infrastructures such as DRBD for that purpose.

5. Is btrfs optimized for SSD out of the box and how?

Please use the mount option “ssd”, to enable btrfs's ssd allocation scheme.

There is an alternative allocation scheme called “ssd_spread”.

The btrfs Wiki writes: “Mount -o ssd_spread is more strict about finding a large unused region of the disk for new allocations, which tends to fragment the free space more over time. Mount -o ssd_spread is often faster on the less expensive SSD devices. The default for autodetected SSD devices is mount -o ssd.”

https://btrfs.wiki.kernel.org/index.php/FAQ#What_is_the_difference_between_mount_-_o_ssd_and_mount_-_o_ssd_spread.3F

6. Can btrfs partitions be encrypted?

Yes, this is possible.

However, we have not implemented support for this into our YaST2 partitioner. Thus we recommend that you are using one of the other methods of encrypting storage, such as device mapper based encryption. – As a sidenote, this is a good argument, why combining LVM2 with btrfs can be useful, ...

7. Does btrfs support RAID?

Yes, the btrfs filesystem supports this.

HOWEVER, WE HAVE NOT IMPLEMENTED FULL SUPPORT FOR THIS INTO OUR YAST2 PARTITIONER.

Thus we recommend that you are using one of the other methods to create a software RAID with Linux; both are supported by the YaST2 partitioner directly:

MD stack

DM (device mapper) stack with LVM2. – This is another argument, why combining LVM2 with btrfs can be useful, ...

Novell, Inc. Copyright 2012-ATT LIVE-1-HARDCOPY PERMITTED. NO OTHER PRINTING, COPYING, OR DISTRIBUTION ALLOWED.
LAB 5-1: Install and Configure Btrfs

**Summary:** In this exercise, you install btrfsprogs on your machine and format an empty virtual disk with Btrfs using YaST.

**Duration:** 10 min.
Btrfs Architecture
Btrfs Terminology (1/3)

**inode**
- A data structure in a traditional UNIX file system that stores all information about a regular file/directory/filesystem object except its data and file name (metadata)

**file**
- An “leaf node” entry in a file system comprised of a hard link (file name) and an inode
- Small file (less than one leaf block) are stored in the btree entry and larger files are stored in an extent

**directory**
- A special type of file that contains hard links rather than data
  - a list of association structures (name,inode #)
- Directories are indexed in two different ways:
  - optimized for filename lookup
  - optimized for reading of data

Notes:
Btrfs Terminology (2/3)

**subvolume**

- A named btree that holds files and directories
- can be mounted directly or access via the top-level subvolume
  - represented as a directory in the top-level subvolume
- A btrfs filestsem has at least one subvolume referred to as the default subvolume
- Snapshots are performed at the subvolume level
- Similar in to some extent to LVM logical volumes but are contained within a single btrfs filesystem

Notes:
### Btrfs Terminology (3/3)

**extent**  
- Smallest allocatable portion of disk space in an extent based file system  
  - can be comprised of 1 or more contiguous blocks  
  - allows for shorter list of allocatable units associated with a file  
  - Contiguous area of storage reserved for a file  
  - an entire extent is allocated when a new file is created

**extent block groups**  
- Allow allocation optimization by breaking the disk up into chunks of 256MB or more  
- Record info about the number of blocks available  
  - file and directories have a preferred block group where they will try first for allocations

**extent tree**  
- Divide up storage into a number of flexible allocation policies assignable to a collection of tree roots

---

**Notes:**
Btrfs Commands

btrfs
- main btrfs administration command
- comprised of several “sub” commands

btrfs filesystem
- performs filesystem tasks

btrfs subvolume
- performs subvolume tasks

btrfs device
- performs device tasks

btrfs scrub
- performs scrub tasks

mkfs.btrfs
- creates a btrfs file system

btrfs-convert
- converts an ext[2,3,4] file system to btrfs

Notes:
Btrfs Migration
Migration From Ext2/3/4

- Convert from Ext2, Ext3, Ext4 to Btrfs
- Requirements:
  - File System is not mounted
  - Minimum: 15% free space on the device
- Roll back is complete, not partial!

Fast conversion because only the Btrfs meta data are created. The data blocks are not touched.

btrfs uses a file system snapshot labeled "extX_saved" to archive the Ext file system.

Snapshot can be deleted if you will not roll-back to Ext.

Caveat: when rolling back all data which has been added after the conversion into btrfs will be lost.
Convert and Roll-Back

- `dal:~ # btrfs-convert /dev/sdc1`
  creating btrfs metadata.
  creating ext2fs image file.
  cleaning up system chunk.
  conversion complete.
  `dal:~ #`

- `dal:~ # btrfs-convert -r /dev/sdc1`
  rollback complete.
  `dal:~ #`
LAB 5-4: Perform an In-place Upgrade of an ext3 File System to Btrfs

Summary: In this exercise, you convert an existing ext3 file system to btrfs.

Special Instructions
Use the following values in the exercise:

(none)

Duration: ? min.

Lab Notes:
Btrfs Copy on Write
Does "Butter" FS Come from (a) C.o.W?

According to Chris Mason it does.
He's the main author of btrfs,
and says it should be called “butter”fs,
as it comes from the “CoW”.

26
Copy on Write (1)

“Normal” Write
- Existing blocks of a file are overwritten, when the content changes

Copy on Write
- If content of a block changes, the changed block is put beside the non-changed block
- Instead the metadata (block-list) changes

Benefit
- Implementation of “transactions” in the filesystem is easy, as old content is still available
Copy on Write (2)

“Normal” Write

Copy on Write

Sun

is

shining

FREE

Sun

was

shining

FREE

Sun

is

shining

FREE

Sun

is

was

FREE
Btrfs Subvolumes
Subvolumes (1)

- A complete filesystem tree
- Usually appears as a sub-directory in the “parent” fs
- Can be mounted separately not “just a subdirectory”

Similar to
- two “foreign” filesystems, which are
- using the same pool of data blocks (and other infrastructure)

Benefits
- different parts (subvolumes) of a filesystem can have different attributes, such as quotas or snapshotting rules
- Copy on Write is possible across volumes

Basic commandline management
- “btrfs subvolume …”
- “mount -o subvol=…”

The `df` command does not show the correct amount of free disk space on Btrfs file systems.

The appropriate way is built into the `btrfs` command:

**Syntax:**  `btrfs filesystem df Path`

**Example:**

dal:~ # btrfs filesystem df /data

    System: total=4.00MB, used=4.00KB
    Data+Metadata, RAID0: total=818.00MB, used=92.00KB

dal:~ #
Subvolumes (2)

Normal Filesystem

With Subvolumes
Snapshots

- Copy on Write on a
  - full subvolume tree
  - instead of a single file only

- Every snapshot is again a subvolume of its own
- Snapshots (as subvolumes) can be mounted and accessed as every other subvolume
- Snapshots can be created read-only
- Basic commandline management
  - “btrfs subvolume snapshot ...”
Manage Subvolumes Using YaST (1/2)
Manage Subvolumes Using YaST (2/2)
Managing Subvolumes on the Command Line

- `btrfs subvolume create [Path/]Name`
  
  Example: `btrfs subvolume create /data/SV4`

- `btrfs subvolume set-default ID Path`
  
  Example: `btrfs subvolume set-default 264 /data`

- `btrfs subvolume delete [Path/]Name`
  
  Example: `btrfs subvolume delete /data/SV4`

- `btrfs subvolume list [-p] Path`
  
  Example: `btrfs subvolume list -p /data`

- `btrfs subvolume find-new [Path/]Name GenPointer`
  
  Example: `btrfs subvolume find-new /data/SV3 35`

---

`Btrfs subvolume list:` Option `-p` also displays the parent ID.

```bash
da1:~ # btrfs subvolume find-new /data/SV3 35
inode 257 file offset 0 len 99 disk start 0 offset 0 gen 35 flags INLINE test
transid marker was 35
```

“35” = Value of the generation pointer
transid marker = Current value of the generation pointer

It may take a few seconds before find-new displays the correct information.
Mount Subvolumes

- Mount a Subvolume:
  - mount -o subvol=Name /dev/sdXX /Mountpoint
  - mount -o subvolid=ID /dev/sdXX /Mountpoint

- btrfs subvolume Command Parameter

"Normal" file system: One partition (or logical volume) = One file system

Btrfs: Several virtual file systems can be created inside the Btrfs filesystem and can be mounted on their own.

Each subvolume is a subtree of the root tree.

First Btrfs on a volume is a subtree called "default"

Mount option "-o subvol" can be used for subvolumes that are located in the root directory of the Btrfs.

ID of the Subvolume can be get from the "list" command. For example:

btrfs subvolume list /data
LAB 5-2: Create a New Btrfs Subvolume and a Snapshot Subvolume

Summary: In this exercise, you create a new btrfs subvolume name /data and create a snapshot of it.

Special Instructions
Use the following values in the exercise:

*(none)*

Duration: ? min.

Lab Notes:
LAB 5-3: Convert an Existing Directory into a Subvolume

**Summary:** In this exercise, you make /home a btrfs subvolume while maintaining all of the file that are in the /home directory.

**Special Instructions**
Use the following values in the exercise:

*(none)*

**Duration:** ? min.

Lab Notes:
Btrfs Snapshots and Rollback
Btrfs Snapshot Basics

- Snapshot and the original point to the same blocks
- Snapshots reside on the same subvolume that has been snapshotted
- Are CoW writable
- Can be snapshotted again
  - ie, Snapshot of a snapshot
- Are performed at the subvolume level
  - ie, Subvolumes make snapshotting of Btrfs Possible

Initially a snapshot does not occupy additional disk space.

Partitions containing snapshots need to be larger than “normal” partitions.
SUSE Btrfs Snapshot Features

- Single file rollback (“undochange”)
- User interfaces: CLI and YaST2 integration
- Snapshots for YaST2 and zypper activities
- Automated snapshots (time based)
- Automated snapshot cleanup (time/number based)
- Allow to work with several kernels
  - implemented independently of btrfs (in ZYpp)
Rollback Type 1 – per Subvolume

• How it works
  • Instead of the original subvolume, the snapshot is mounted with the options “subvol=<name>”
    – Remember: snapshots are subvolumes
  • Talking about the “/” filesystem, the “subvol” can also be hard coded using “btrfs subvolume set-default ...”

• Benefits
  • “Atomic” operation
  • Very fast

• Disadvantages
  • Additional complexity
    – May require explicit mounting of subvolumes
  • No “rollback” per single file
Rollback Type 2 – File based “undochange”

• 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”
    - may need some more computing time
Snapshotting “/” – Challenges

Multiple Kernels
- Separate /boot
- Zypper integration

System integrity and Compliance
- Don't allow to roll back certain log-files etc.
- Solution: subvolumes instead of directories for
  /tmp
  /opt
  /srv
  /var/spool
  /var/log
  /var/run
  /var/tmp
Snapshotted “/” – Traditional Method

“/” filesystem is mounted as any other Linux Filesystem: root volume of the btrfs filesystem = “/” in a user's view

Benefits
- No difference for people coming from ext3, xfs, …
- no “set-default” necessary

Disadvantages
- subvolume based rollback introduces additional complexity
- less flexibility for future improvements,
  > e.g. no “parallel root filesystems”
Introduction to the Snapper Tool

- SUSE Toolchain co-developed for openSUSE and SUSE Linux Enterprise
  - “snapper” command line tool
  - YaST2 integration for snapshot rollback
  - Unique functionality: selective rollback

http://lizards.opensuse.org/2011/04/01/introducing-snapper/
The **snapper** Command

- CLI Utility for file system snapshot management
- snapper can:
  - Create snapshots
  - delete snapshots
  - compare changes between snapshots
  - undo changes between snapshots
- For each subvolume being snapshotted, a snapper configuration file must exist for that subvolume
  - created by the `snapper create-config` command
- **snapper** distinguishes 3 types of snapshots:
  - pre / post
  - single

**Snapper Snapshot Types:**

**pre / post**
Pre / post snapshots always come in a pair. A **pre** snapshot is performed before a change to the system takes place and then a **post** snapshot is performed immediately after the change takes place. These two snapshots can then be compared to sow exactly what changed in the file system.

**single**
Single snapshots are standalone and have no relationship to other snapshots
Snapper Configurations

- Stored in `/etc/snapper/configs/`
- `snapper create-config Path`
- `snapper list-configs`

Example:
```
da1:~ # snapper list-configs
Config | Subvolume
-------+----------
root   | /data
```

copies allow you to specify when to make snapshots, how many of each type should be kept etc.

Config templates are in `/etc/snapper/config-templates`.

One configuration for each partition or subvolume

Default configuration `/ ("root")`
Snapper Command Parameters

- `snapper [-c ConfigName] create -d "Description"
- `snapper list`
- `snapper diff SnapshotNo1 SnapshotNo2`
- `snapper diff --file File SnapshotNo1 SnapshotNo2`
- `snapper rollback SnapshotNo1 SnapshotNo2`
- `snapper undochange`
- `snapper [-c ConfigName] delete SnapshotNo1`
Snapper: Important Files and Directories

/etc/snapper/ - Main configuration directory
/etc/snapper/configs/ - Directory containing configuration files for managed subvolumes
/etc/snapper/filters/ - Directory containing files that list the files and directories that are to be filtered (ignored)
/etc/snapper/config-templates - Directory containing managed subvolume configuration file templates

Notes:
Snapper: Automatic Snapshot Creation

- The snapper utility also automatically creates snapshots of defined subvolumes on a time based interval with cron
- Utilities such as zypper and YaST automatically create snapshots when they run
- Snapper can also automatically clean up (delete) old automatically created snapshots

Notes:
Snapper: Subvolume Config File

```plaintext
# subvolume to snapshot
SUBVOLUME="/"

# filesystem type
FSTYPE="btrfs"

# run daily number cleanup
NUMBER_CLEANUP="yes"

# limit for number cleanup
NUMBER_MIN_AGE="1800"
NUMBER_LIMIT="100"

# create hourly snapshots
TIMELINE_CREATE="yes"

# cleanup hourly snapshots after some time
TIMELINE_CLEANUP="yes"

# limits for timeline cleanup
TIMELINE_MIN_AGE="1800"
TIMELINE_LIMIT_HOURLY="10"
TIMELINE_LIMIT_DAILY="10"
TIMELINE_LIMIT_MONTHLY="10"
TIMELINE_LIMIT_YEARLY="10"

# cleanup empty pre-post-pairs
EMPTY_PRE_POST_CLEANUP="yes"

# limits for empty pre-post-pair cleanup
EMPTY_PRE_POST_MIN_AGE="1800"
```

In case your root filesystem is btrfs, Snapper will have created a config that makes a “pre” and “post” snapshot pair whenever you use YaST or zypper in addition to the “timeline” snapshots.

By default, the last 100 YaST and zypper snapshots are being kept. If this number is exceeded, the oldest snapshot(s) will be deleted.
Snapper: Snapshot Directory Layout

/.snapshots/

| 1/ | -snapshot/ |
|    |   -(files and directories) |
|    |   -info.xml |

Single snapshot

| 2/ | -snapshot/ |
|    |   -(files and directories) |
|    |   -info.xml |

Pre snapshot

| 3/ | -snapshot/ |
|    |   -(files and directories) |
|    |   filelist-2.txt |
|    |   -info.xml |

Post snapshot

List of changes

Notes:
Snapper: YaST Module

List of Snapshots

Displays Differences

Notes:
LAB 5-5: Configure Snapper to Manage a New Subvolume

Summary: In this exercise, you configure Snapper to manage the /home subvolume. You then create and list snapshots of the /home subvolume.

Special Instructions
Use the following values in the exercise:

*(none)*

Duration: ? min.

Lab Notes:
LAB 5-6: Use the Snapper YaST Module to Restore Files from a Snapshot

Summary: In this exercise, you use the Snapper YaST module to role back changes to a file from a snapshot.

Special Instructions
Use the following values in the exercise:

*(none)*

Duration: ? min.

Lab Notes:
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.