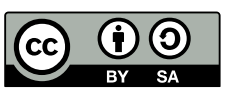
relax-and-recover.org

Relax and Recover (ReaR) Workshop



Agenda

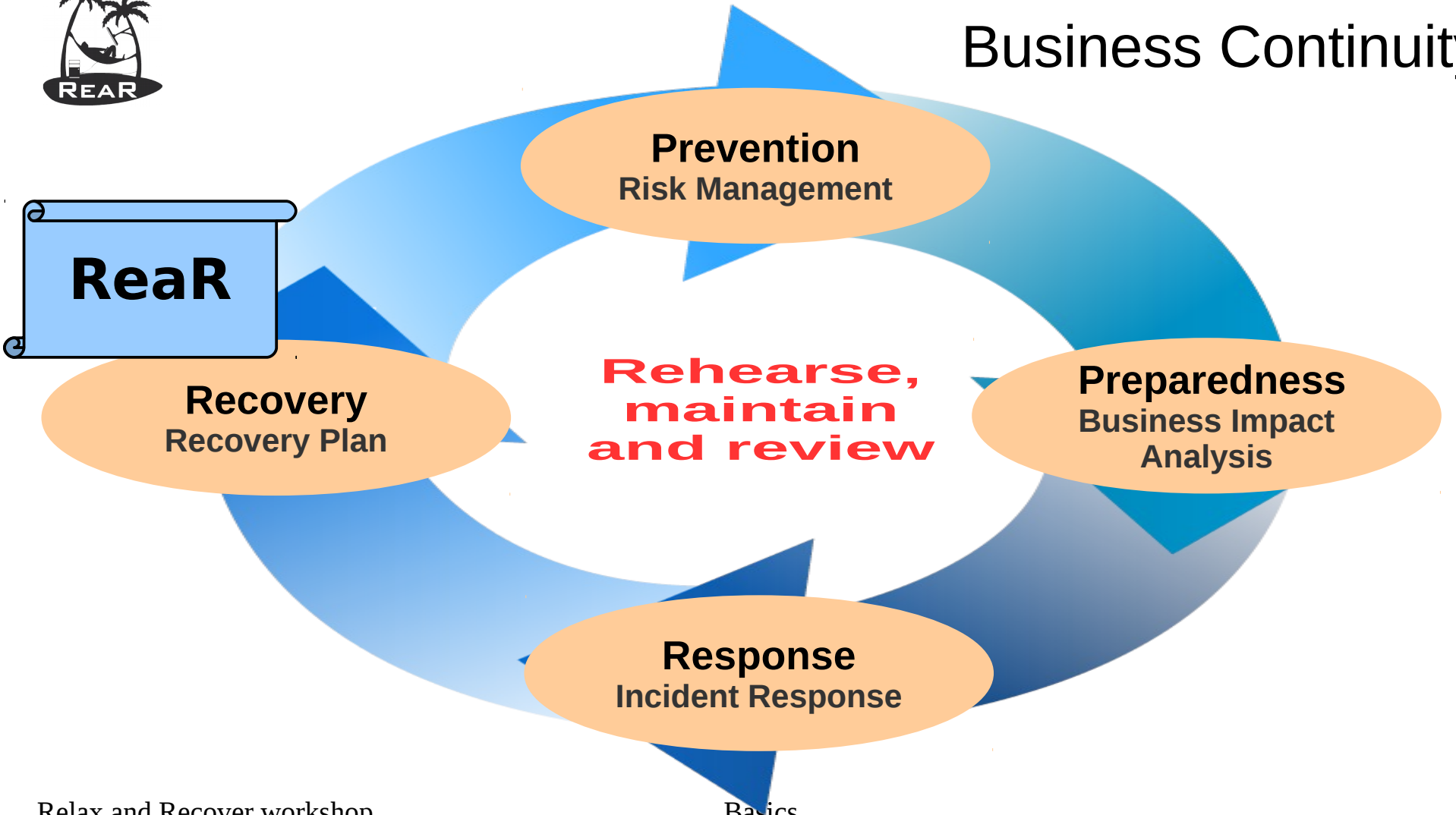
- Introduction and Basics
- Lab 1:
  - Basic Usage
  - Automated Disaster Recovery
- Secure Disaster Recovery & Advanced Usage Scenarios
- Architecture and Development
- Lab 2:
  - Debugging ReaR Issues
  - Extending ReaR with Custom Code
  - Contributing to ReaR
  - Advanced Usage Example

Introduction and Basics

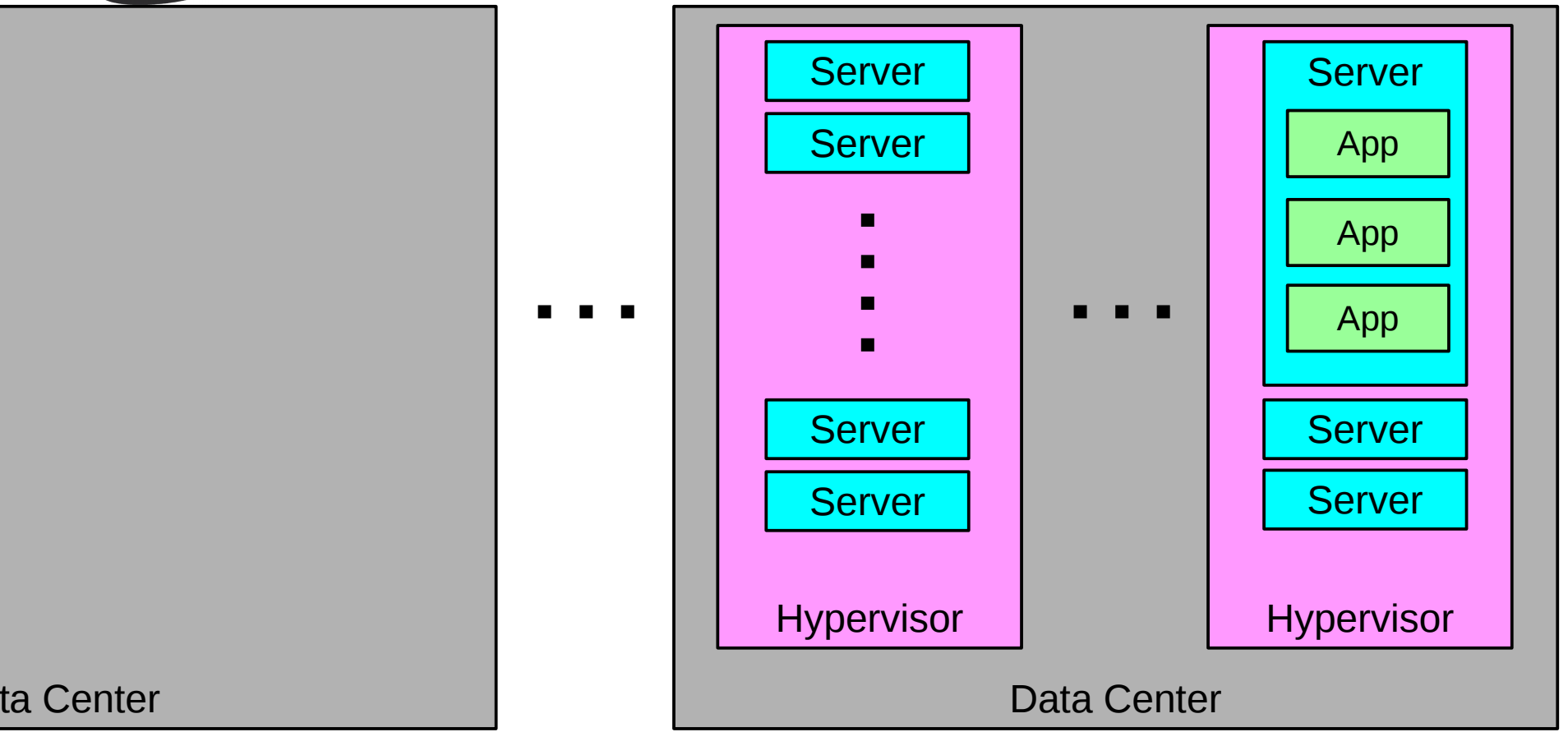
Ask Yourself: Mean Time to Restore Service

- After deploying a bad software update or configuration?
- After upgrading the Operating System to a faulty version?
  - On 50 servers? On 500 servers?
- After deleting the hard disk / SAN LUN of your main database?
  - After deleting 20 LUNs?
- After deleting the hard disk / SAN LUN of a Hypervisor?
  - All the LUNs of a virtualization cluster?
- After flooding the data center?

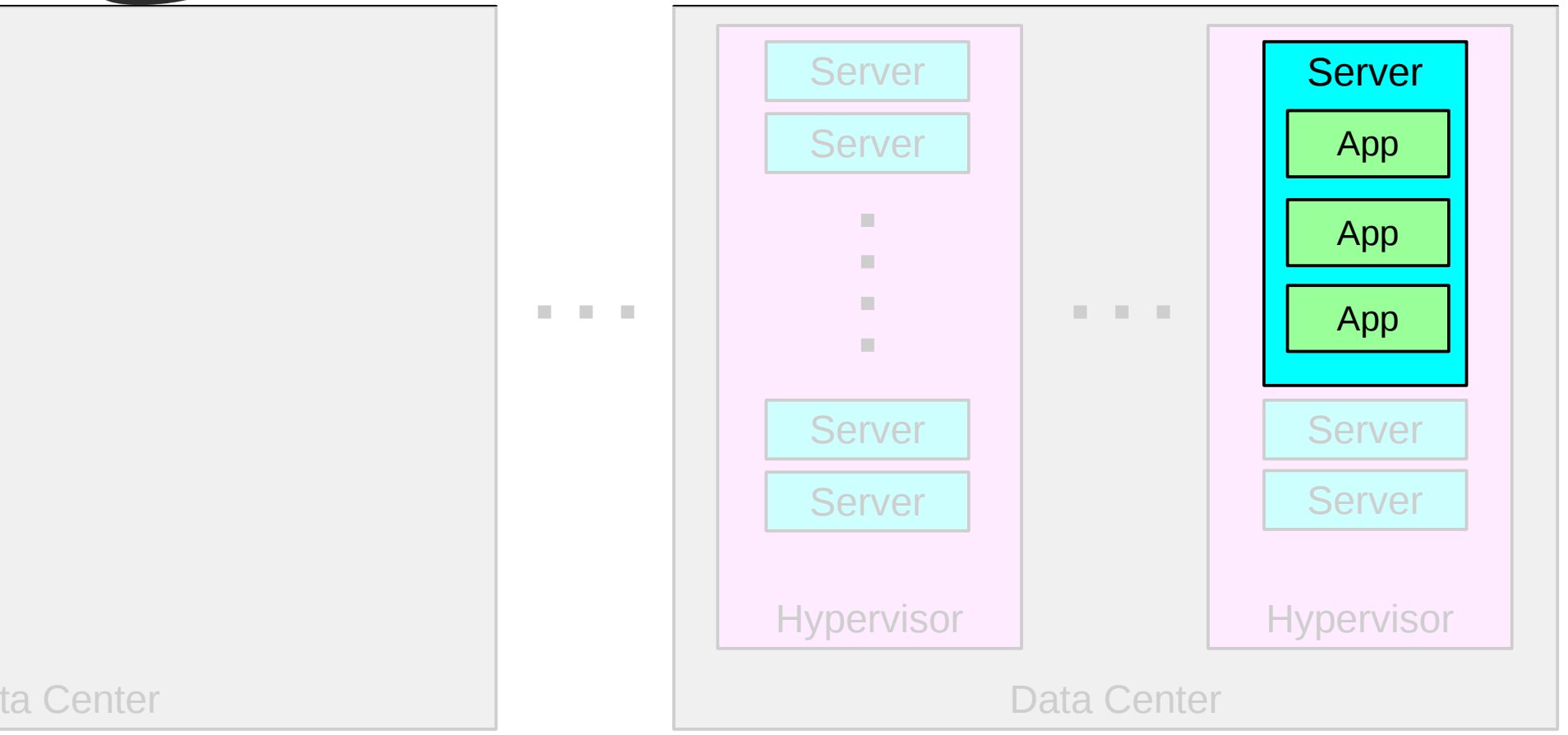
Business Continuity



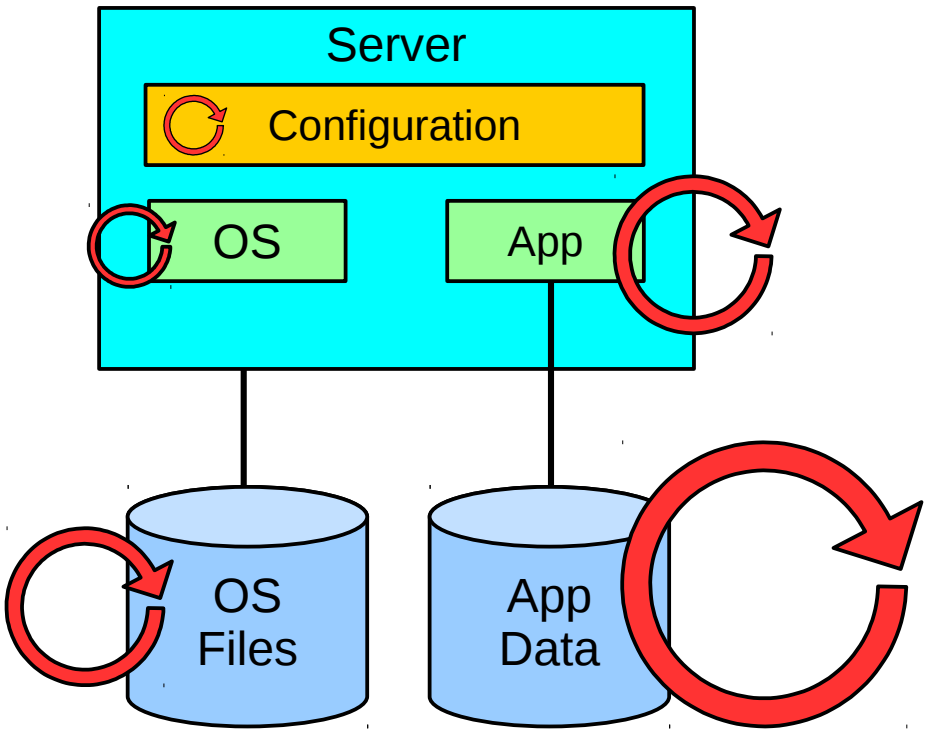
Scope

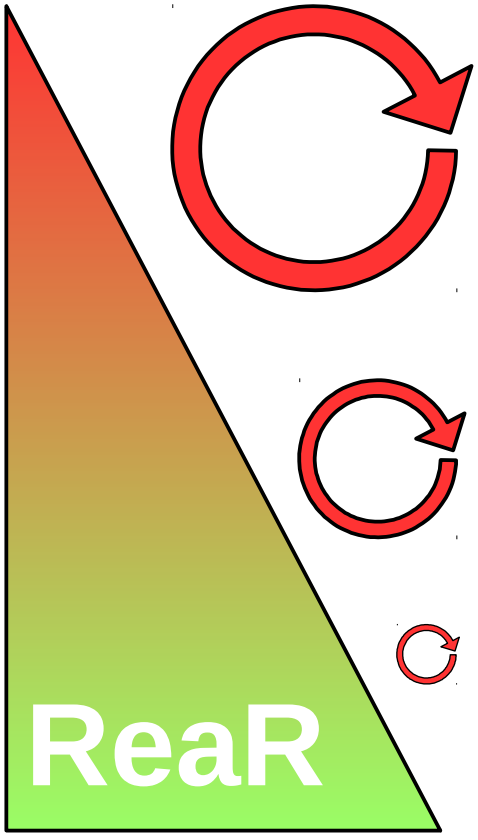

ta Center
Server
Server
Server
Server
Hypervisor
Server
App
App
App
Server
Server
Hypervisor
Data Center

Scope of Relax and Recover



Understanding Change Frequencies



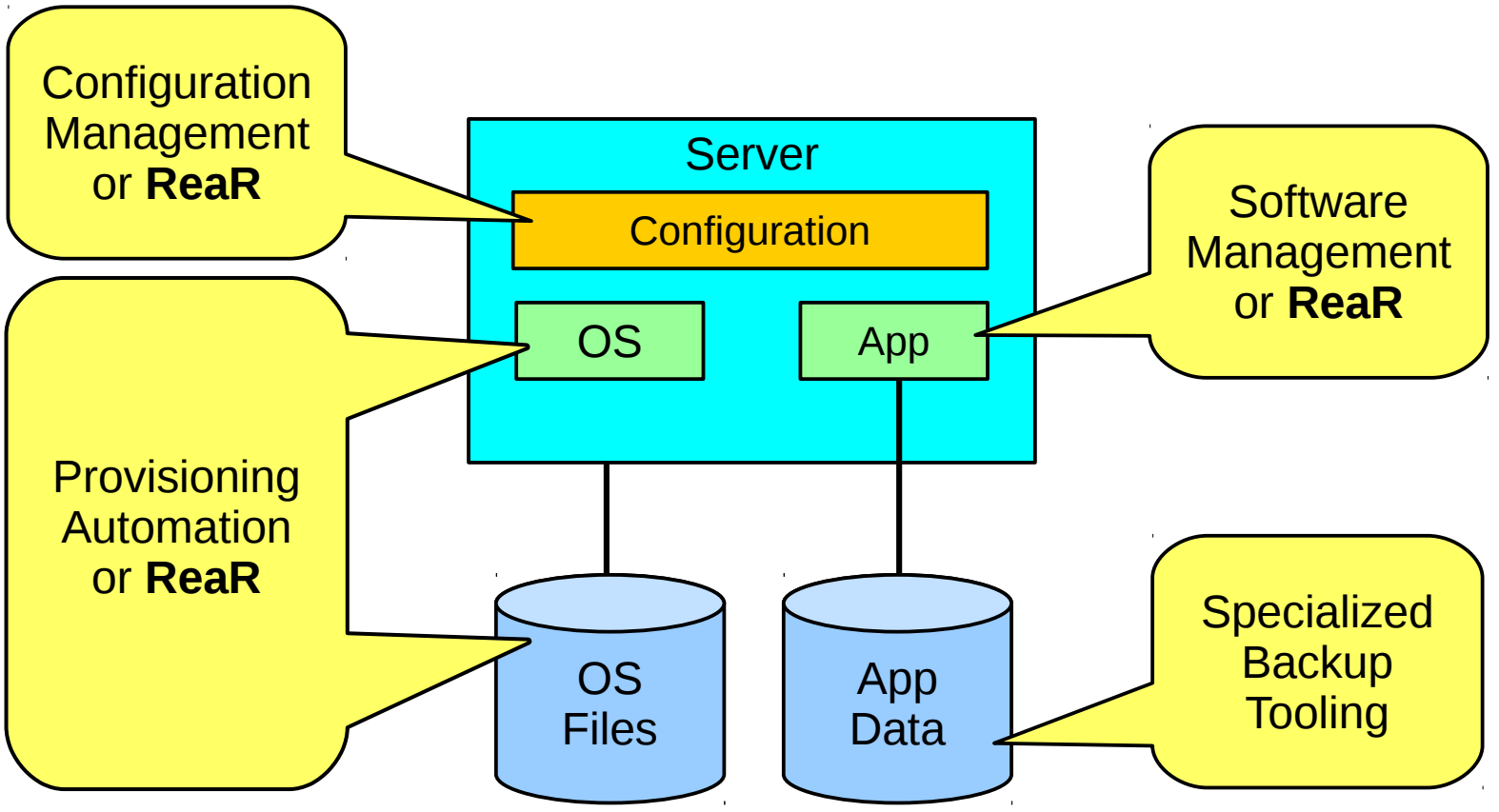


Constantly changing
Very high cost per time lost

Regularly changing

Seldomly changing

Use the Best Tool for the Restore Job



Manual → Automated

Backup → Restore → Recovery

Relax and Recover

Why are backups not enough?

- Backups of data are necessary!
- Are not enough in case of losing the complete Operating System (OS)!
- Reinstalling the OS from scratch takes hours
- Restoring the backups a few more hours
- Fine-tuning of configurations takes days
- Even months later issues pop up!
- It is absolute necessary to foresee an inventory of hard- and software

Disaster Recovery Plan (DRP)

- DRP addresses need to recover from an emergency with minimum impact to the enterprise
- Protects enterprise from major services failure
- Minimizes risk to enterprise from delays in providing services
- Guarantees reliability of standby systems by testing and simulation
- Minimizes personnel decision-making required during disaster recovery

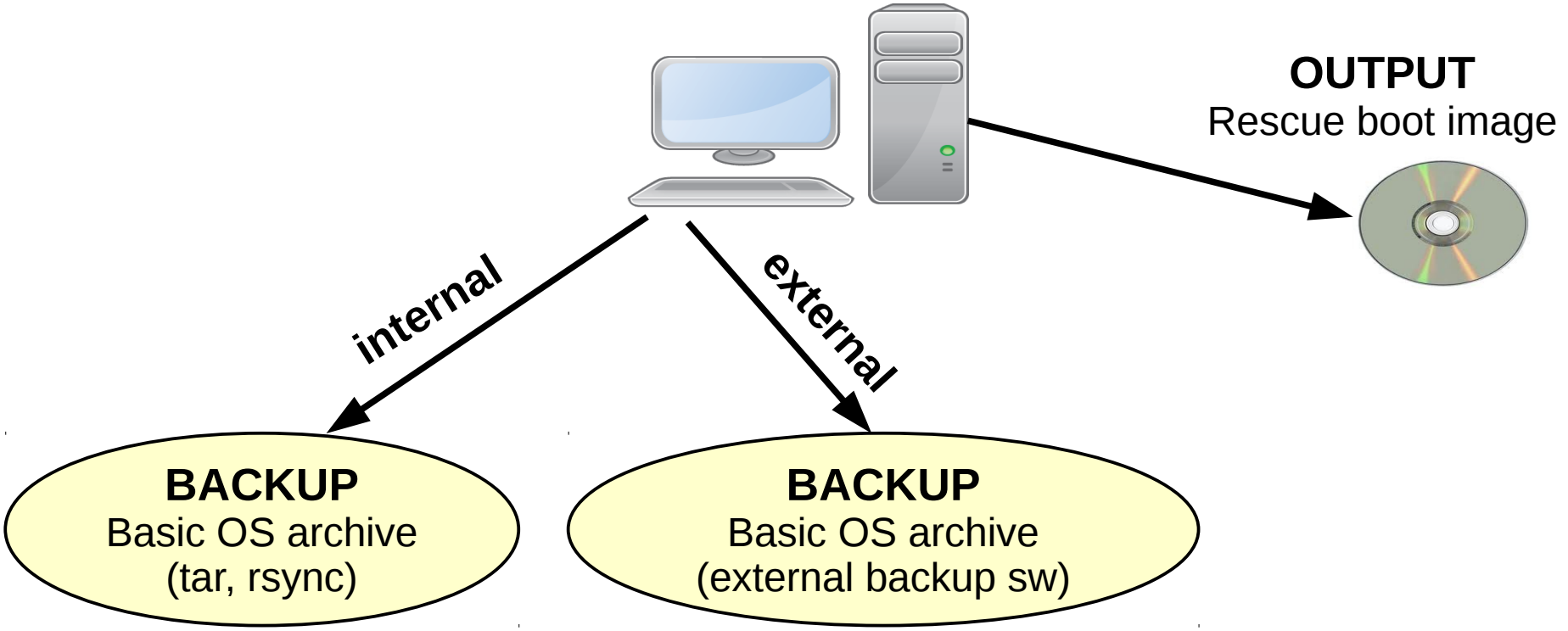
Relax and Recover as DR solution

- Rear is a tool that implements a DR work-flow for Linux
- Basically meaning:
  - Modular framework written in Bash
  - Easy to extend to own needs
  - Easy to deploy (set up and forget)
  - Integration for various Linux technologies
  - Integration with various back-up solutions
  - Attempts to make system recovery *as easy as possible*
- Rear runs *on-line* (no downtime to create a DR image)

Introduction to Relax and Recover (rear)

- Proven solution at large enterprise customers
- Rear established as standard solution for Linux disaster recovery in data centers
- Shipping with Fedora, openSUSE and RHEL 7.2 (and >)
- Integrates with many “commercial” backup software solutions, e.g. TSM, DP, NBU, NSR, …
- Integrates with OS backup software solutions as well, e.g. GNU tar, rsync, bacula, bareos, ...
- Scales well with large amounts of servers

DR Flow – BACKUP and OUTPUT



Decide on DR strategy

- Which backup mechanism to use?
  - Internal backup: GNU tar, rsync
  - External backup: bacula, bareos, commercial backup solution
- Where will the backups reside?
  - NFS share, CIFS share, external USB disk, tape, local spare disk, cloud storage, DVD
  - Remote network and/or storage location
- How shall we boot the rescue image?
  - Via DVD (ISO image), tape (OBDR), network (PXE), USB disk

Disaster Recovery - Media

- **Most important: External storage!**
- Bootable media: CD/DVD, USB key, LAN, tape ...
- Media usually combination boot and backup media:
  - Bootable CD/DVD, USB key with backup data on it
  - LAN boot (PXE) with backup data via CIFS, NFS ...
  - Bootable tapes - HP OBDR (CD emulation)
- Separation between boot media and backup data
  - Boot the system from a (small) USB key, CD/DVD or LAN
  - Recover the system with backup software, tar, rsync ...

Disaster Recovery – How It Works

- Store the disk layout
  - Partitioning, LVM and RAID configuration
  - File systems, file system labels ...
  - Boot loader (GRUB, GRUB2, LILO, UEFI)
- Store the files (tgz, rsync, through backup software ...)
- Create bootable rescue media with system configuration (and backup data)
- **Can be done online**
  - No business interruption
  - 100% compatible with original systems hard- and software
  - Works very reliably with Operating System files, not with databases …

Usage of rear

- Shell scripts are stored under `/usr/share/rear`
- Scripts are kept together according work-flows
  - `mkrescue` (only make rescue image)
  - `mkbackup` (including make rescue image)
  - `mkbackuponly` (excluding make rescue image)
  - `recover` (the actual recovery part)
- Easy to incorporate new scripts, e.g. for information gathering of Hard- and Software, or other goodies

Backup Types

- The major “backup types” available are
  - **NETFS**: NFS, CIFS, USB, TAPE, ISO, SSHFS
  - RSYNC: With rsync server
  - REQUESTRESTORE, EXTERNAL (noop)
  - BACULA, BAREOS, RBME, DUPLICITY (open source software)
  - DP, NBU, TSM, NSR, GALAXY[7], SESAM (commercial software)

BACKUP and OUTPUT methods

- BACKUP variable defines the “backup” method
  - NETFS, RSYNC, DUPLICITY, ….
- BACKUP_URL variable defines the location where to store the backup archive
- OUTPUT variable defines the “output” method
  - ISO, PXE, OBDR, USB
- OUTPUT_URL variable defines the location where to store the output image (ISO image, pxe configuration, extlinux configuration)

BACKUP type NETFS

isolinux
OUTPUT=ISO

pxelinux
network

OUTPUT=PXE
BACKUP=NETFS

(NFS|CIFS|local)
disks

OUTPUT=ISO
BACKUP=NETFS

Tape drive

OUTPUT=OBDR
BACKUP=NETFS

extlinux
External USB disks

OUTPUT=USB
BACKUP=NETFS

Location BACKUP_URL

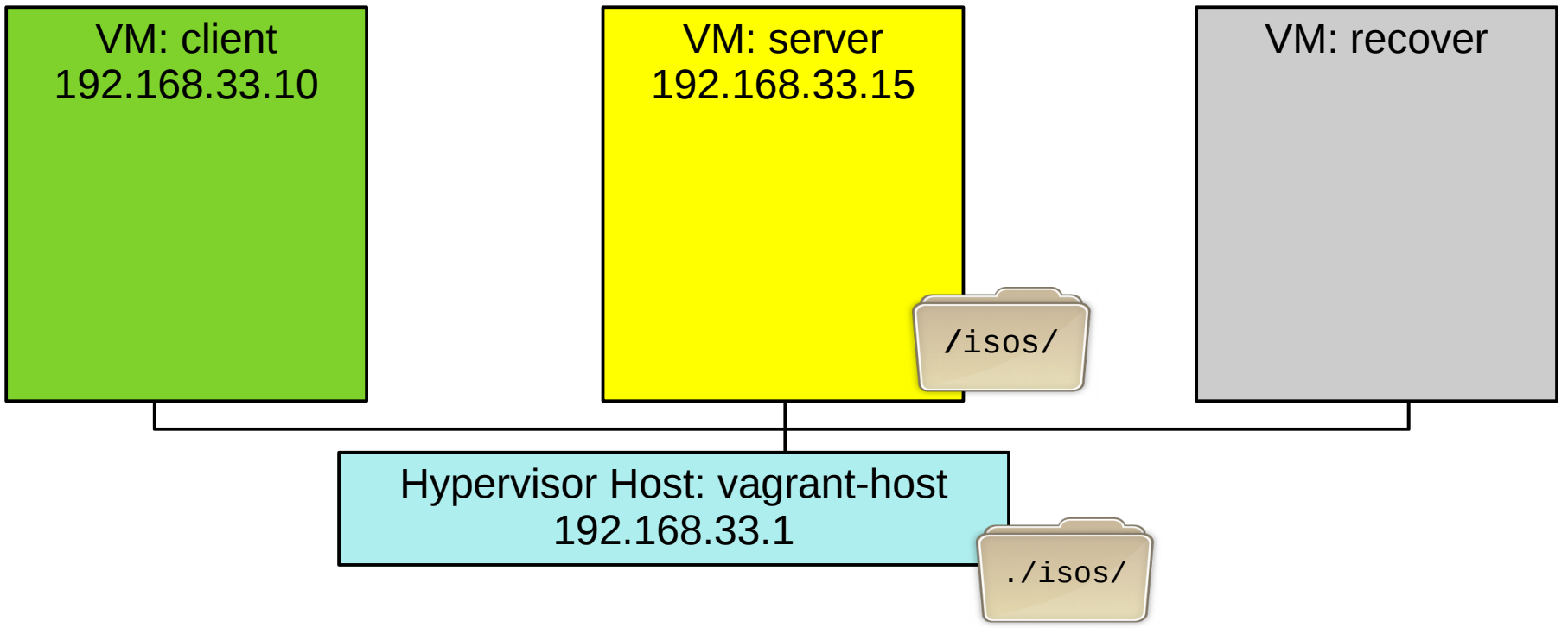
- **BACKUP=NETFS**
- **BACKUP_URL** can be
  - File type: `BACKUP_URL=file:///directory/`
  - NFS type: `BACKUP_URL=nfs://nfs-server/directory/`
  - CIFS type: `BACKUP_URL=cifs://samba/directory/`
  - USB type: `BACKUP_URL=usb:///dev/disk/by-label/REAR-000`
  - ISO type: `BACKUP_URL=iso://backup`
  - Tape type: `BACKUP_URL=tape:///dev/nst0`

Backup Program

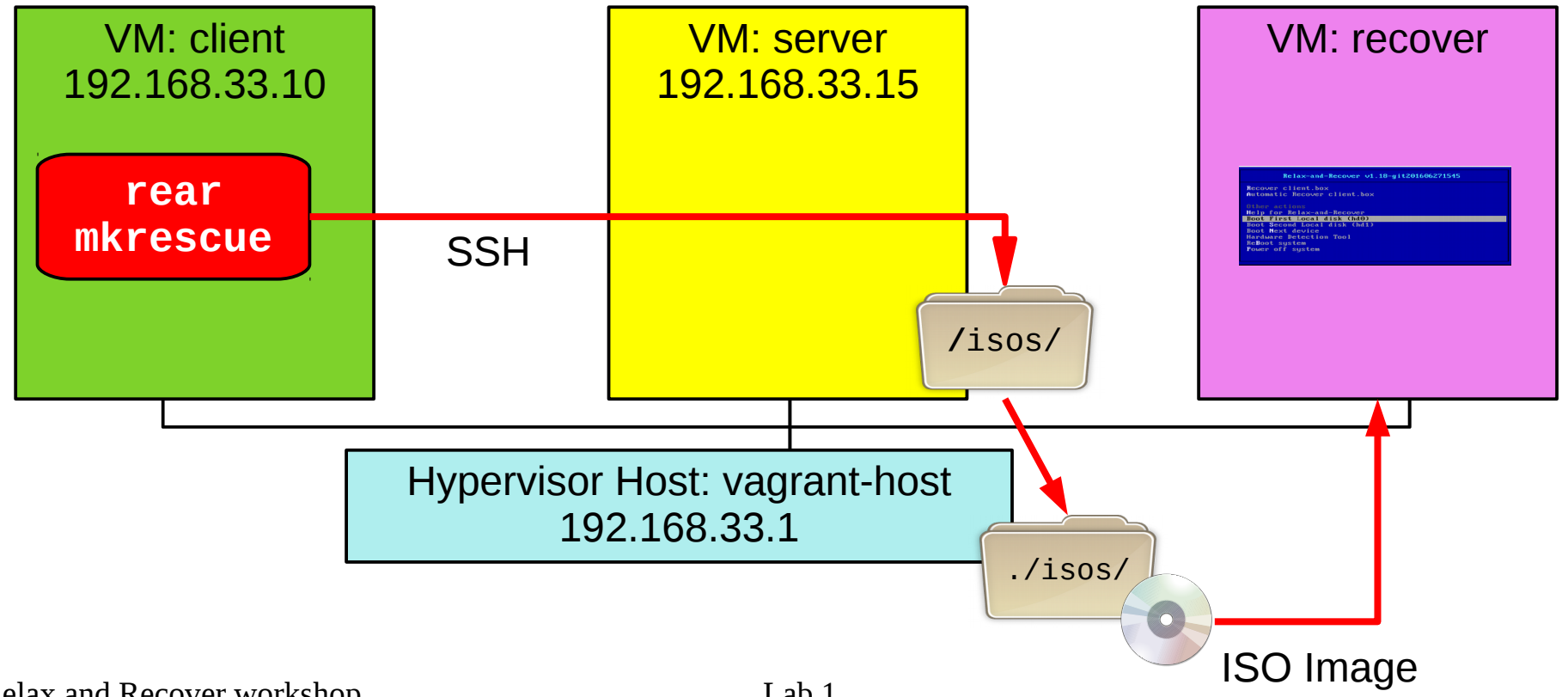
- BACKUP=NETFS
- /usr/share/rear/conf/default.conf contains:
  - Default: BACKUP_PROG=tar
  - However, BACKUP_PROG=rsync is possible for local storage
  - BACKUP_PROG_COMPRESS_OPTIONS="--gzip"
  - BACKUP_PROG_COMPRESS_SUFFIX=".gz"
  - BACKUP_PROG_EXCLUDE=('/tmp/*' '/dev/shm/*')

Lab 1:
Basic Usage

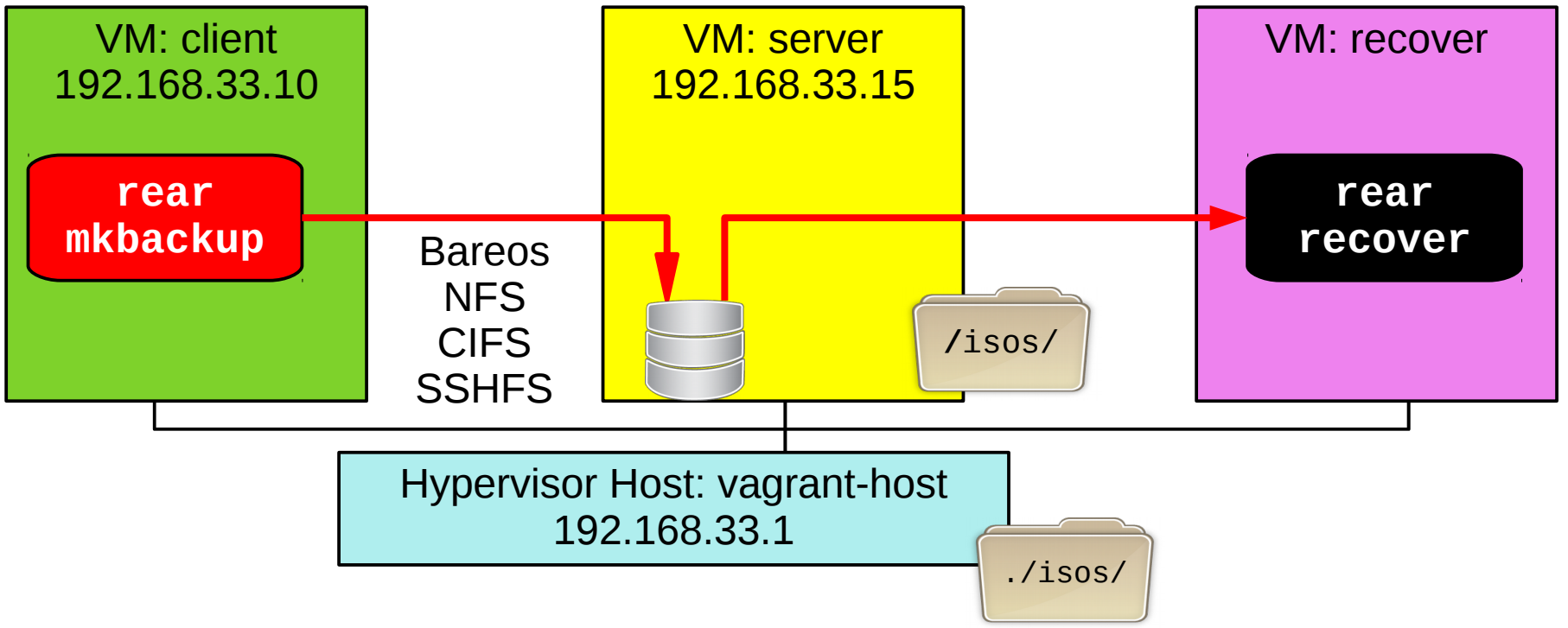
Lab Overview: Basic Setup



Lab Overview: Boot Media (CD)



Lab Overview: Backup & Restore



Set up vagrant environment

Set up vagrant environment

Install / Download:

- VirtualBox ≥ 5.0 https://www.virtualbox.org/wiki/Downloads
- vagrant ≥ 1.9.4 https://www.vagrantup.com/
  - For vagrant 1.9.4 (Fix https://github.com/mitchellh/vagrant/issues/8519):
    `vagrant plugin install vagrant-share --plugin-version 1.1.8`
- Download workshop files:
  https://github.com/rear/rear-workshop/archive/master.zip

(Should work on Linux, Mac, Windows, tested on Linux)

Set up vagrant environment (2)

- Download workshop files: https://github.com/rear/rear-workshop/archive/master.zip
- Unzip workshop: `unzip master.zip`
- Go into directory "`rear-workshop-master/centos7`"
- Type "`vagrant up`"
- Use "`vagrant ssh client`" to login on the client system
- Use "`vagrant ssh server`" to login on the server system

Set up vagrant environment (3)

- The last line of the vagrant up output should be:
  `==> client: Complete!`
- Account vagrant/vagrant (and root/vagrant)
- Another way to login is via ssh:
  - ssh root@192.168.33.10 (client root pw is vagrant)
  - ssh root@192.168.33.15 (server root pw is vagrant)

Install Relax-and-Recover

Check NFS & SSH access on server

- On client
  - This should show the exported NFS share:
    `showmount -e server`
  - Check that the following commands work without error:
    ```
    mount -v server:/export/nfs /mnt/
    touch /mnt/from-client
    umount -v /mnt
    ```
  - `ssh server touch /isos/from-client`
- Check that the file exists on host in centos7 directory:
  - `ls -l isos/`
- We confirmed write access from the client to the `isos` directory.

Install Relax-and-Recover

- On client:
  - `yum search rear`
  - `yum -y install rear`
  - `yum -y install rear-workshop`
    (for the configs)
  - Install some extra tools:
    - `yum -y install net-tools`
      (for netstat, route,...)
    - `yum -y install bind-utils`
      (for nslookup, dig,...)

Install Relax-and-Recover

- Explore ReaR on client:
  - `rear`
  - `rear dump`

Make a full backup with Bareos

Make a full backup with Bareos

- Login on client
- Start the “bconsole”
  * `run`
  Automatically selected Catalog: MyCatalog

  The defined Job resources are:
  1: client-backup
  2: client-restore
  3: client-backup-mysql
  4: client-restore-mysql
  Select Job resource (1-4): **1**
  Exit bconsole after the jobs starts
- Check progress while still on client:
  `ssh server tail -f /var/log/bareos/bareos.log`

Configure relax-and-recover

- Configure `/etc/rear/local.conf` as follow:

```
OUTPUT=ISO
OUTPUT_URL=sshfs://server/isos
ISO_DEFAULT=manual
BACKUP=BAREOS
USE_STATIC_NETWORKING=y
KERNEL_CMDLINE="$KERNEL_CMDLINE net.ifnames=0 vga=791"
```

- Or, copy the prepared configuration file:

```
cd /etc/rear
cp workshop/local-with-bareos.conf local.conf
```

- Run `rear -v dump`

Create rear rescue image for BACKUP=BAREOS topology

Create rear rescue image

- On client
- Run “rear -v mkrescue”
- Fix any error shown (BAREOS_RESTORE_JOB)
- Verify `/var/log/rear/rear-client.log`
- Check destination (OUTPUT_URL) on host for ISO

Create rear rescue image

```
# rear -v mkrescue
Relax-and-Recover 2.00 / Git
Using log file: /var/log/rear/rear-client.log
Creating disk layout
Creating root filesystem layout
Copying logfile /var/log/rear/rear-client.log into initramfs as'/tmp/rear-client-partial-2017-05-08T15:43:32+0200.log'
Copying files and directories
Copying binaries and libraries
Copying kernel modules
Creating initramfs
Making ISO image
Wrote ISO image: /var/lib/rear/output/rear-client.iso (132M)
Copying resulting files to sshfs location
Saving /var/log/rear/rear-client.log as rear-client.log to sshfs location
```

Recover with Bareos

- Halt the client on vagant-host: `vagrant halt client`
- Verify on the host that the `isos/client` directory contains the ISO image
- Start the recovery: `vagrant up recover`
- Choose “Recover client” in boot menu
- Recover your 'client' system onto vm 'recover'
  - Check network with '`ip a s`'
  - Recover with '`rear -v recover`'

Recover with Bareos (2)

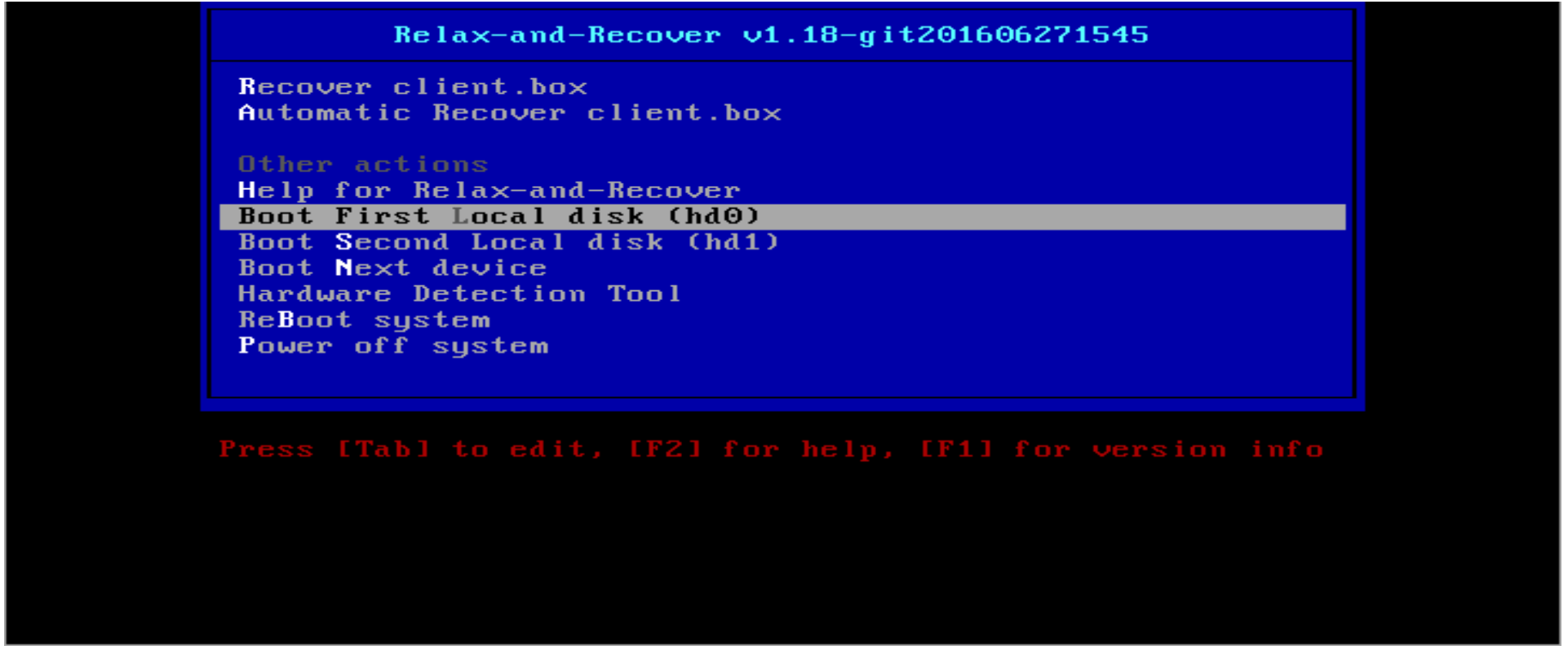
```
Original disk /dev/vda does not exist in the target system. Please choose an app
ropriate replacement.
1) /dev/vda
2) Do not map disk.
#? 1
2016-06-28 13:21:18 Disk /dev/vda chosen as replacement for /dev/vda.
Disk /dev/vda chosen as replacement for /dev/vda.
This is the disk mapping table:
    /dev/vda /dev/vda
Please confirm that '/var/lib/rear/layout/disklayout.conf' is as you expect.

1) View disk layout (disklayout.conf)  4) Go to Relax-and-Recover shell
2) Edit disk layout (disklayout.conf)  5) Continue recovery
3) View original disk space usage      6) Abort Relax-and-Recover
#? 5
Partition primary on /dev/vda: size reduced to fit on disk.
Please confirm that '/var/lib/rear/layout/diskrestore.sh' is as you expect.

1) View restore script (diskrestore.sh)
2) Edit restore script (diskrestore.sh)
3) View original disk space usage
4) Go to Relax-and-Recover shell
5) Continue recovery
6) Abort Relax-and-Recover
#?
```

Recover with Bareos (3)

```
Catalog:          MyCatalog
Priority:         10
Plugin Options:   *None*
OK to run? (yes/mod/no):
Job queued. JobId=2
You have messages.

waiting for job to start
JobId 2 Job client-restore.2016-09-05_15.17.55_12 is running.
waiting for job to finish
Restore job finished.

Please verify that the backup has been restored correctly to '/mnt/local'
in the provided shell. When finished, type exit in the shell to continue
recovery.

rear> df
Filesystem                     1K-blocks    Used Available Use% Mounted on
devtmpfs                          195988       0    195988   0% /dev
tmpfs                             250420       0    250420   0% /dev/shm
tmpfs                             250420    4248    246172   2% /run
tmpfs                             250420       0    250420   0% /sys/fs/cgroup
/dev/mapper/VolGroup00-LogVol00  38764728 1238724  35928892   4% /mnt/local
/dev/vda2                          487571  195247    262628  43% /mnt/local/boot
rear> _
```

Recover with Bareos (4)

Reboot the VM and select now (2 times):



Recover with Bareos (5)

- Check that the recovered VM is as the client was before, e.g. with:
  `yum install -y samba`
- Remove the recovery machine:
  `vagrant destroy recover -f`
- Start the client again:
  `vagrant up client`

Troubleshoot the configuration files

Troubleshoot the configuration files

- Go to directory `/etc/rear/workshop`
- Type:

```
for f in ls *.conf ; do
    bash -n $f
done
```

- And now type:

```
for f in ls *.conf ; do
    source $f
done
```

Using NFS as backup destination

Configure /etc/rear/local.conf

- Replace the content of `/etc/rear/local.conf` with:

```
OUTPUT=ISO
OUTPUT_URL=sshfs://server/isos
ISO_DEFAULT=manual
BACKUP=NETFS
BACKUP_URL=nfs://server/export/nfs
USE_STATIC_NETWORKING=y
KERNEL_CMDLINE="$KERNEL_CMDLINE net.ifnames=0 vga=791"
```

- Or, copy the prepared configuration file:

```
cd /etc/rear
cp workshop/local-with-nfs.conf local.conf
```

Using NFS as backup destination

- Run a **simulation** and understand the difference
  - `rear -s mkrescue`
  - `rear -s mkbackup`
- Run on client: `rear -v mkbackup`
- Halt the client vm: `vagrant halt client`
- Start the recover vm:
  - `vagrant up recover`
  - Select “Recover client” in boot menu
  - Run: `rear -v recover`

Using sshfs as backup destination

Configure /etc/rear/local.conf

- Install: `yum -y install sshfs`
- Replace the content of `/etc/rear/local.conf` with

```
OUTPUT=ISO
OUTPUT_URL=sshfs://server/isos
ISO_DEFAULT=manual
BACKUP=NETFS
BACKUP_URL=sshfs://root@server/export/archives
USE_STATIC_NETWORKING=y
KERNEL_CMDLINE="$KERNEL_CMDLINE net.ifnames=0 vga=791"
```

- Or, copy the prepared configuration file:

```
cd /etc/rear
cp workshop/local-with-sshfs.conf local.conf
```

Lab 8 with sshfs

- Make a backup and ISO image (on client):
  - `rear -v mkbackup`
  - NETFS mounted via sshfs fuse module
- Halt the client vm: `vagrant halt client`
- Start the recover vm:
  - `vagrant up recover`
  - Choose “Recover client”
  - Run: `rear -v recover`

Using CIFS as backup destination

Configure /etc/rear/local.conf

- Install: `yum -y install cifs-utils`
- Replace the content of /etc/rear/local.conf with

```
OUTPUT=ISO
OUTPUT_URL=sshfs://server/isos
ISO_DEFAULT=manual
BACKUP=NETFS
BACKUP_URL=cifs://server/homes
BACKUP_OPTIONS="cred=/etc/rear/.cifs"
USE_STATIC_NETWORKING=y
KERNEL_CMDLINE="$KERNEL_CMDLINE net.ifnames=0 vga=791"
```

- Or, copy the prepared configuration file:

```
cd /etc/rear
cp workshop/local-with-cifs.conf local.conf
```

Configure CIFS credentials

- Create or copy the credentials file:
  `cp /etc/rear/workshop/.cifs to /etc/rear/`
- Make a backup and ISO image (on client):
  - `rear -v mkbackup`
  - NETFS mounted via cifs
- Halt the client vm: `vagrant halt client`
- Start the recover vm:
  - `vagrant up recover`
  - Run: `rear -v recover`

Secure Disaster Recovery
&
Advanced Usage

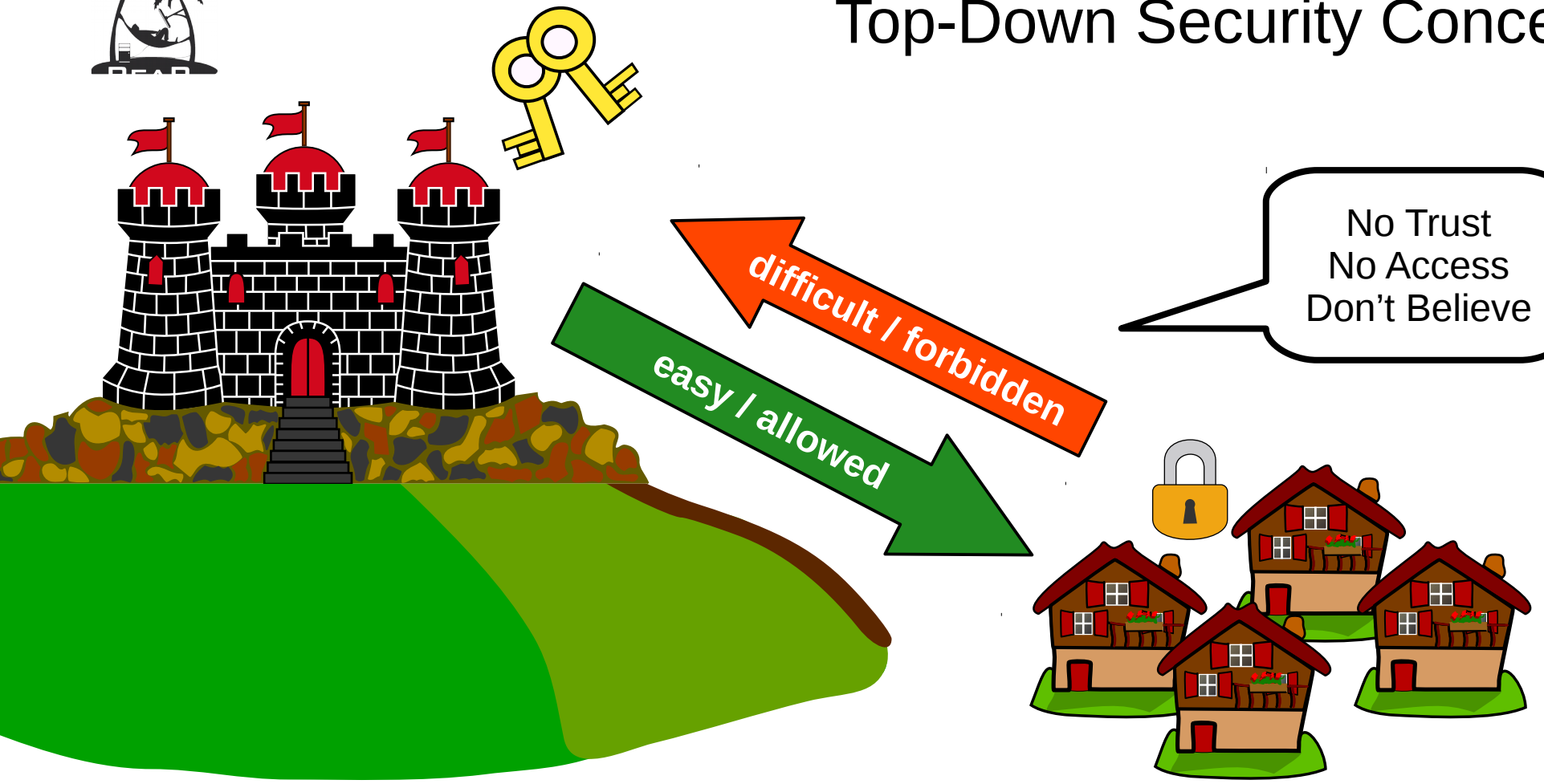
Why is everything we did in the lab really bad security practice?

How will an attacker exploit our setup?

Potential Attack Vectors in Lab 1

- Anybody on client can read your backup and secrets within
- Backups (with secrets) of other clients can be also read
- Use the secrets pulled from the backup archive
- Modify the backup of other clients
- SSH key of root: Run any program on “server”
- Trigger Bareos backup/restore on other clients, modify Bareos
- Attack plan:
  1) Modify backup of important server to include back door
  2) Make sysadmin recover the important server, e.g. via social engineering or DoS attack
  3) Use back door ☺

Top-Down Security Concept



ReaR to the Rescue

- Designed to be secure by default
- Rescue image does not contain exploitable secrets,
  as long as you don’t put them there.
- Most BACKUP and OUTPUT methods are not secure!
- Secure solutions are based on
  - Pull from outside instead of push
  - If push, then write-only access without overwriting existing data
  - Triggering actions like file restore from outside
  - Verifying that the data was not tampered with

ReaR is a tool box ⇒ build your own secure solution.

Security Best Practices

- Don’t use `BACKUP=NETFS`, backup should be pulled from secure system. Some commercial backup software solves this better.
  See also RSYNC BACKUP MADE EASY: https://github.com/schlomo/rbme
- Pull rescue image or use drop protocols to push image:
  FTP: `OUTPUT_URL=ftp://server/`
  HTTP: `OUTPUT_URL=http://server/`
  SMTP: `RESULT_MAILTO=rear@domain`
- Orchestrate recovery from secure system:
  - Provide access to rescue image only by need, not always
  - Watch for suspicious behavior, e.g. many rescue images in short time

DISASTER RECOVERY LINUX MANAGER

About:

DRLM is a Centralized Management Open Source solution for small-to-large Disaster Recovery implementations using ReaR.

Is an easy-to-use software to manage your growing ReaR infrastructure. Is written in the bash language (like ReaR) and offers all needed tools to efficiently manage your GNU/Linux disaster recovery backups, reducing Disaster Recovery management costs.

ReaR is great solution, but when we're dealing with hundreds of systems, could be complex to manage well all ReaR deployments.

With DRLM you can, easily and centrally, deploy and manage ReaR installations for all your GNU/Linux systems in your DataCenter(s).

DRLM is able to manage all required services (TFTP, DHCP-PXE, NFS, ...) with no need of manual services configuration. Only with few easy commands, the users will be able to create, modify and delete ReaR clients and networks, providing an easy way to boot and recover your GNU/Linux systems through network with ReaR.

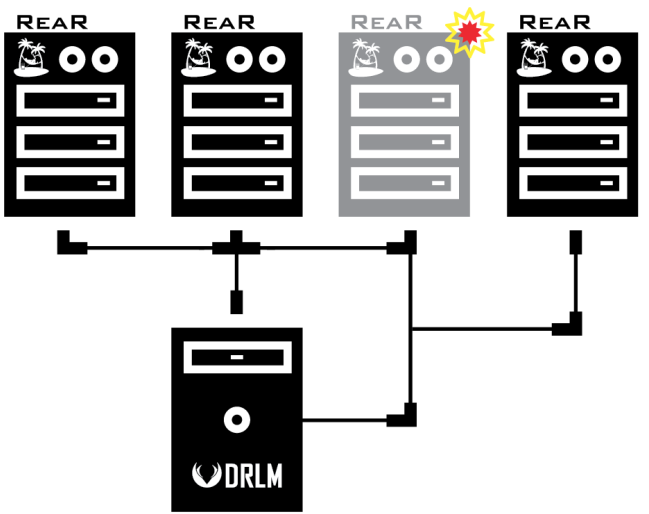
Furthermore DRLM acts as a central scheduling system for all ReaR installations. Is able to start rear backups remotely and store the rescue-boot/backup in DR images easily managed by DRLM.

You can easily enable or disable the last or any previous backups to restore any client with a single command line.

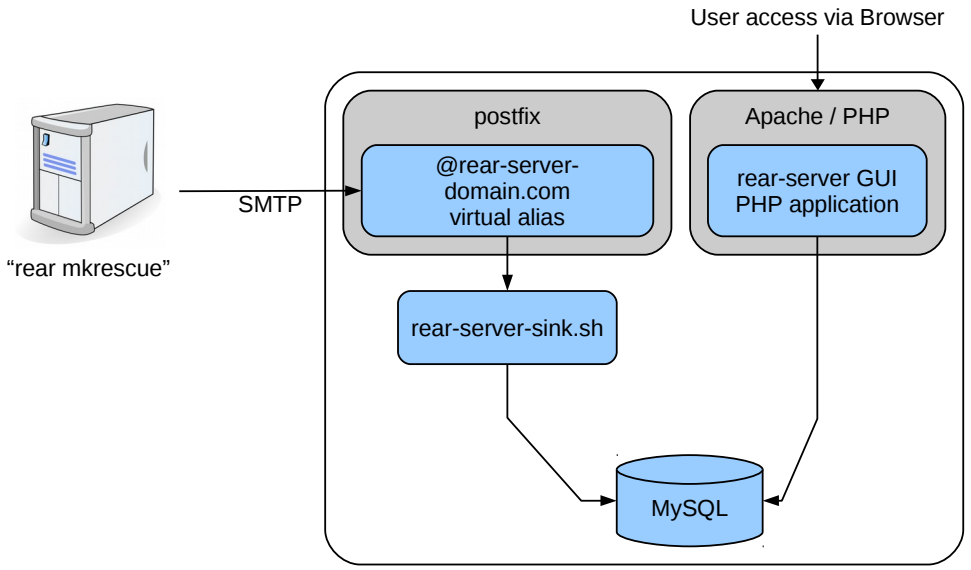
Currently DRLM supports PXE and NETFS(nfs) OUTPUT/BACKUP methods of ReaR, but the Development of DRLM non stops here, we are working on new 2.0 version with new features to improve performance, usability and more ReaR methods, in order to become, together with ReaR, the reference when talking about Disaster Recovery of GNU/Linux systems.

DRLM

http://drlm.org/



Idea for rear-server



- Use SMTP or HTTP as secure, easily available and cross-domain routable transport protocol
- Version 1: Central dashboard
- Version 2: Trigger DR via UI
- Version 3: Integrate with system management solutions

See https://goo.gl/6K61Kb for more infos, looking for sponsor

Advanced Features 1

- Use `PROGS` and `COPY_AS_IS` to add your own content
- Add your own commands or scripts via `POST_RECOVERY_SCRIPT, PRE_RECOVERY_SCRIPT, POST_BACKUP_SCRIPT, PRE_BACKUP_SCRIPT`
- Create bootable recovery media on USB storage:
  - `BACKUP=NETFS`
    `BACKUP_URL=usb:///dev/disk/by-label/REAR-000`
    `OUTPUT=USB`
    `UDEV_WORKFLOW=mkbackup`
  - `rear format /dev/sdb`
  - Automatically trigger backup via udev rule:
    `ACTION=="add", SUBSYSTEM=="block", ENV{ID_FS_LABEL}=="REAR-000", RUN+="/usr/sbin/rear udev"`
- Use `rear -C CONFIG` to configure different profiles
  https://github.com/rear/rear/blob/master/doc/user-guide/11-multiple-backups.adoc

Advanced Features 2

- Automate entire process via network boot `OUTPUT=PXE`
- Set correct time on recovery TIMESYNC, TIMESYNC_SOURCE
- Automate P2V, P2P, V2V migration by answering questions in mappings files:
  - `/etc/rear/mappings/disk_devices:`
    `/dev/cciss/c0d0 /dev/sda`
  - `/etc/rear/mappings/ip_addresses:`
    `eth0 213.203.238.113/25`
    `eth1 dhcp`
  - `/etc/rear/mappings/mac:`
    `00:11:85:c2:b8:d5 00:50:56:b3:75:ad eth0`
    `00:11:85:c2:b8:d7 00:50:56:b3:08:8c eth1`
  - `/etc/rear/mappings/routes:`
    `default 213.203.238.1 eth0`
    `192.168.33.0/24 213.203.238.45 eth0`
  - Provide `/etc/rear/disklayout.conf` if you know better
  - Provide `/etc/rear/lun_wwid_mapping.conf` for SAN migrations

Architecture & Development

Internals

- **Bash** framework
- Main program: `/usr/sbin/rear`
- Configuration: `/usr/share/rear/conf` and `/etc/rear`
- Functions: `/usr/share/rear/lib/*-functions.sh`
- Workflows: `/usr/share/rear/lib/*-workflow.sh`
- Stages: `/usr/share/rear/*`
- Multi-dimensional script merging by `ARCH`, `OS`, `OS_VERSION`, `BACKUP`, `OUTPUT`, `BACKUP/OUTPUT` and more. Use rear -s to see how it works.

Coding

- https://github.com/rear/rear/wiki/Coding-Style
- Use Bash arrays and other advanced Bash features
  Learn from the Advanced Bash-Scripting Guide http://tldp.org/LDP/abs/html/
  and frequently consult `man bash`
- Document all global variables in `default.conf`
- Test on as many Linux distributions as you can

Lab 2
Development
Debugging
Advanced Usage

Development – Preparation

Please use Linux for development, we all do.

- Use “client” VM with NFS configuration from Lab 1
- Configure plain SSH access from host to client:
  ```
  vagrant ssh-config client >>~/.ssh/config
  ```
- Git clone ReaR sources:
  ```
  git clone git@github.com:rear/rear.git
  ```
- Install build dependencies:
  ```
  ssh client sudo yum install -y rpm-build git
  ```
- Create backup to restore:
  ```
  ssh client sudo rear -v mkbackuponly
  ```

Development – Change Code and Test

Use the editor of your choice to work on ReaR.

- Push ReaR sources to client
  ```
  rsync -av --del rear/ client:rear/
  ```
- Build & install RPM
  ```
  ssh client "cd rear ; make rpm; sudo rpm -Uhv
  --force *.rpm ; sudo rear -v mkrescue"
  ```
- `vagrant up recover`
- `vagrant ssh recover -- -t rear -v recover`

Debugging

- Read the logs – read the logs – read the logs!
- See which scripts are called: `rear -s mkrescue`
- Step-by-step, watch the data files: `rear -S mkrescue`
- Enable debug mode `-d` or debugscript mode `-D`, read the log and inspect the work directory in `/tmp/rear.*`
- Set a break point:
  BREAKPOINT=layout/save/default/400_check_backup_special_files.sh rear -v checklayout
- Add debug output to scripts or add a script that dumps stuff
- Use rear shell to try out internals:

```
rear shell
# SourceStage layout
```

Contributing to ReaR

Follow the instructions in our development guide at http://relax-and-recover.org/development/ to contribute something to ReaR, e.g. fix a spelling mistake.

Read also the existing documentation, guides and links to blog articles in http://relax-and-recover.org/documentation/ for ideas.

Help (or sponsoring) wanted especially with

- Testing and test automation, see also https://github.com/rear/rear-integration-tests
- Documentation
- Cleanup of legacy code paths

Try some advanced features

- Create a disk mapping file to skip the question if `/dev/sda` should be recovered onto `/dev/sda` ☺
- Configure vsftpd as drop-only FTP server, find out how to allow next upload without harming previous one?
- Configure VirtualBox PXE booting and exercise PXE-based recovery.
  See https://github.com/defunctzombie/virtualbox-pxe-boot
- Create a backup of the Lab laptop onto USB thumb drive, erase it and recover it from the USB media

Feedback

Need Assistance?

Relax-and-Recover support options:
http://relax-and-recover.org/support/

Professional Support:

Gratien D'haese
IT3 Consultants

Heinlein Support GmbH

Schlomo Schapiro Open Source Consulting