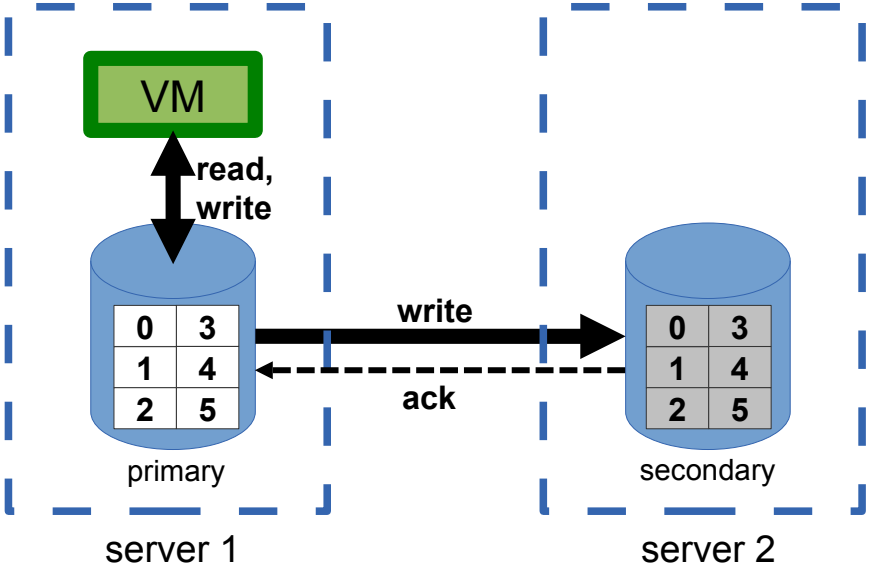
DRBD and Linstor

Recap: Traditional storage options

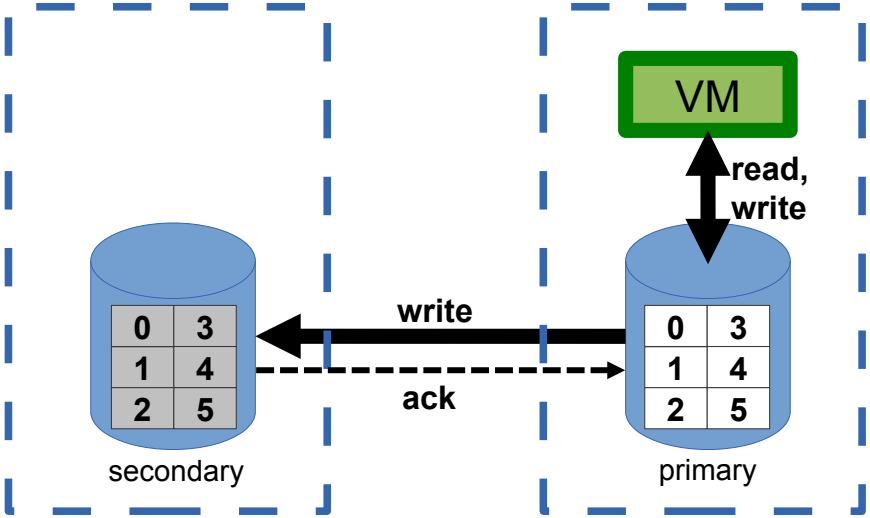
- Local storage (with RAID)
  - Migrations require copying the entire volume (slooooow)
  - Loss of node = loss of VMs on that node
- SAN or NAS
  - The storage server is a single point of failure. $$$
- Distributed software-defined storage e.g. Ceph
  - There are failure modes that can lock up the whole system
  - Performance issues are hard to diagnose
  - Your data is stored in opaque blobs

Another option: block-level replication

- "RAID1 over the network": access local disk, writes replicated

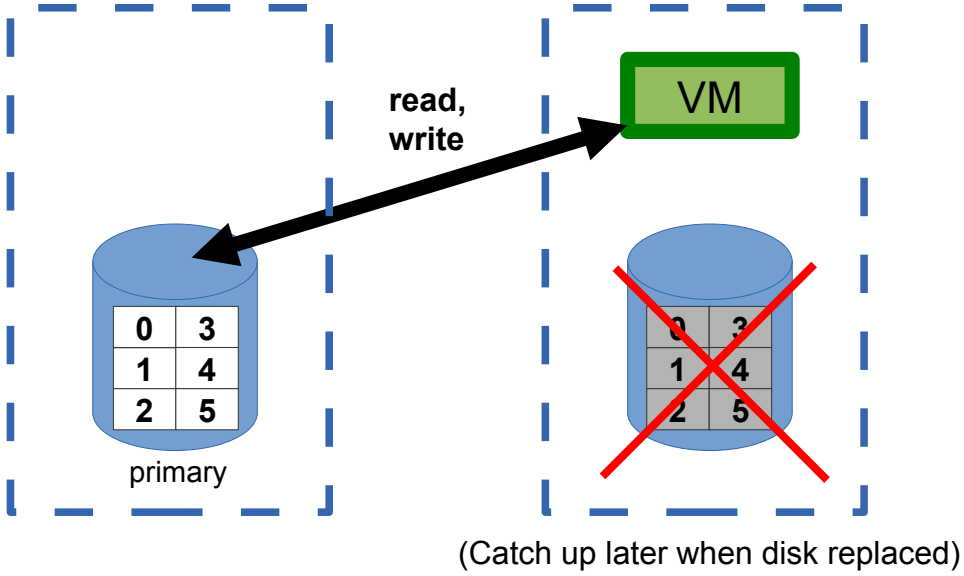


Reverse roles when VM migrates



Disk failure

- Transparently redirect all I/O over network

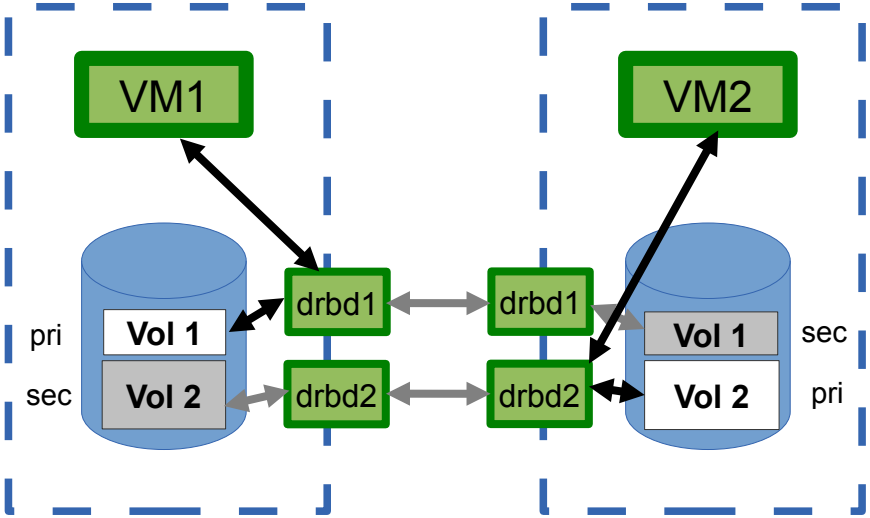


DRBD: Distributed Replicated Block Device

- Has been part of the Linux kernel for many years
  - Hugely robust and well tested. It's "only" mirroring…
- Your data is stored **as-is** on the underlying storage
  - Plus a little bit of metadata at the end to track replication state
- Out-of-box configuration is via **static** configuration files (e.g. mirror /dev/sda1 on one machine to /dev/sda1 on another)
  - You could run LVM on top of DRBD to break up the space
- This is not great if we want to migrate individual VMs
  - We want separate DRBD instances created dynamically per VM

Layered DRBD

- DRBD-over-LVM (or DRBD-over-ZVOL)



Solution: Linstor

- Written by Linbit, the originators of DRBD
- Management layer on top of DRBD 9
  - Allows up to 32 replicas of each volume
  - Allows diskless clients (i.e. VM can run even where there's no replica)
  - Most Linux distros have DRDB 8, so you need to install kernel module
- Free and open source, but commercial support available
- Integrations for Proxmox VE, Kubernetes, OpenStack etc
- Prometheus metrics and Grafana dashboard
- Why is this not more widely known??

Linstor architecture

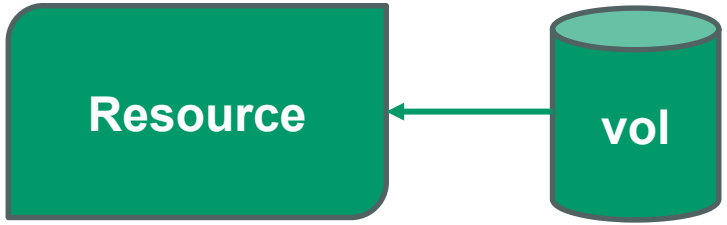
- Stateless "satellite" on each node to manipulate DRBD/LVM, plus a central "controller" with database
- All written in Java ☹
  - Expect ~1GiB RAM usage on each node
  - But it stays completely out of the data path
  - Even if a satellite or controller crashes or restarts, replication is completely unaffected
- Controller exposes an HTTP API (which Proxmox plugin uses)
- CLI talks to API; optional web UI also available

Linstor terminology

- A participating server is a **node**
  - its role is "satellite", "controller" or "combined"
  - you add them manually *("linstor node create …")*
- Satellite nodes have named **storage pools**
  - LVM volume group, LVM thin pool, or ZFS zpool
  - you add them manually to each node *("linstor storage-pool create …")*
  - one node can provide multiple storage pools
    - e.g. different volume groups for SSD and HDD
    - e.g. thin pools and regular volume groups
- On a node, **volumes** are space allocated from storage pools

Resources and Volumes

- Volume replication is managed by a "**resource**"
- Resources have names; volume is identified by resource name + volume number (starting from zero)
- One resource *could* have multiple volumes, but normally it's 1:1

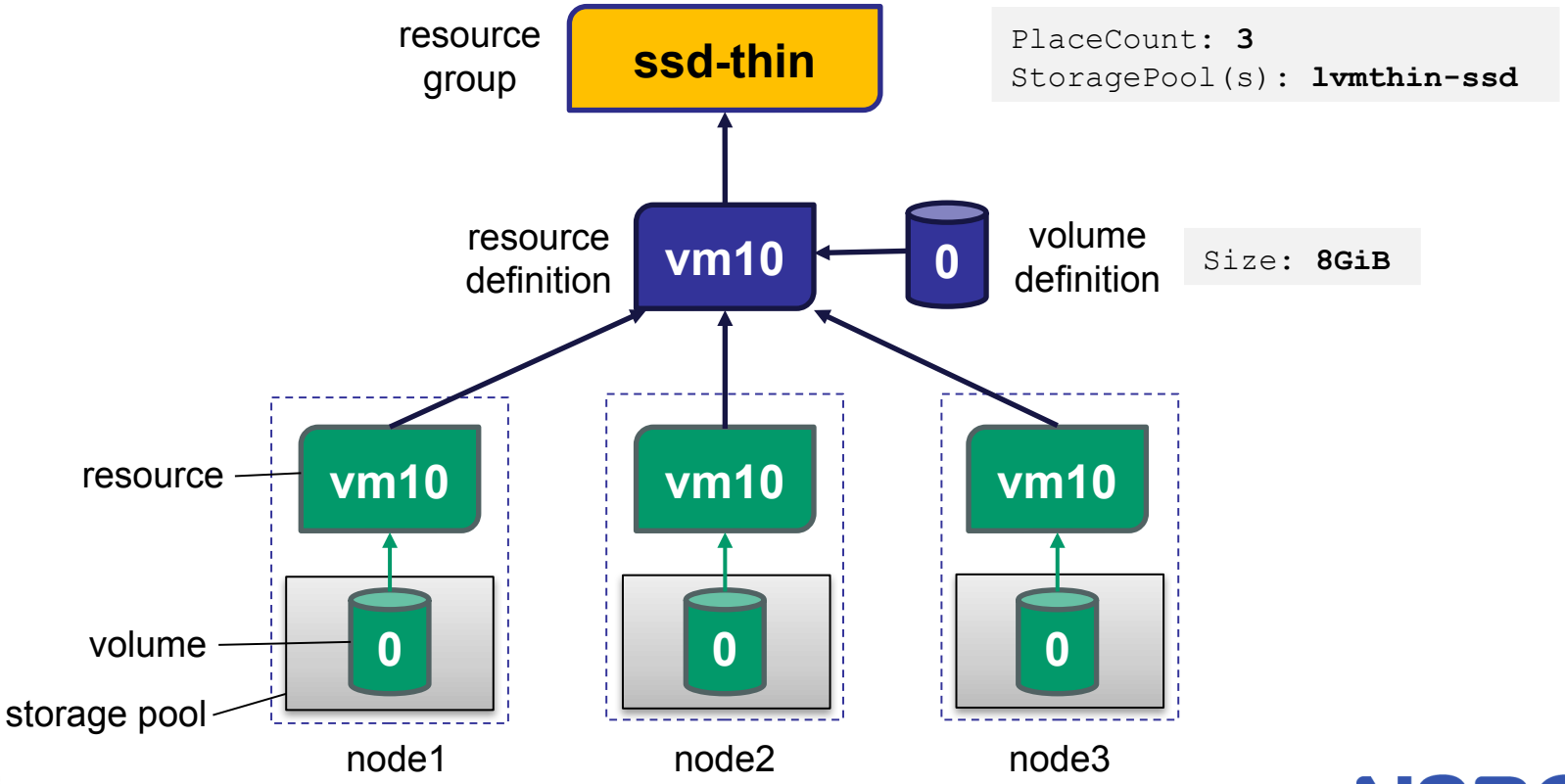


accessed as
`/dev/drbdXXXX`

Resource definitions

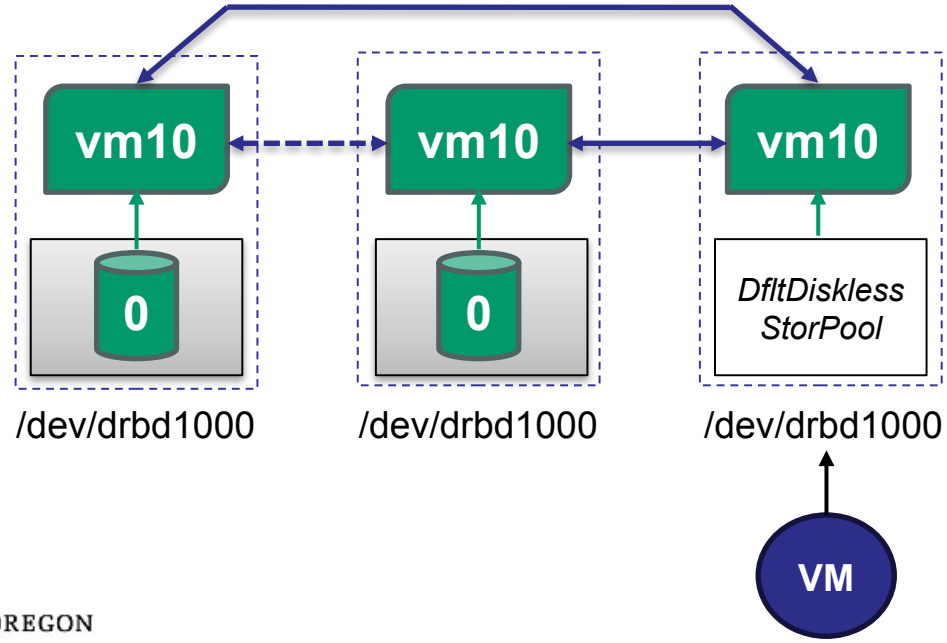
- A **resource** (and volume) is the manifestation of a single replica
- A **resource definition** (and volume definition) describes the overall collection of replicas – e.g. the volume size you want
  - What you think of as a "virtual disk" for your VM
  - Has the same name as its associated resources
- A **resource group** is a set of common parameters inherited by a resource definition
  - e.g. how many replicas you want to maintain; which storage pool(s) to allocate space from
  - In effect, it's a "class of service" for your resource definitions

Putting it all together



Diskless Resources

- Allow access to storage without creating a local copy
- All I/O is forwarded over the network



Tie Breaker Resources

- When there exactly **two** resources, a third Tie Breaker resource is automatically created
- Just like a normal diskless resource, except automatically removed when no longer required
- Having three resources protects against some split brain scenarios when one node suddenly vanishes

Moving resources

- This is really easy!
- "linstor resource create …" to say where you want a new replica
  - it will immediately mirror-sync (unless you asked for a diskless resource)
- When done, "linstor resource delete …" the unwanted replica
- When you're an expert, there are shortcuts to do this in one step

Linstor command line

- There is tab-completion to help
- Shortened versions are easier to type

```
linstor resource-group list
linstor resource-definition list
linstor resource-definition list-properties <rsrc>
linstor volume-definition list
linstor resource list
linstor volume list
linstor resource create <nodes..> <rsrc>
linstor resource delete <nodes..> <rsrc>
…etc
```

```
linstor rg l
linstor rd l
linstor rd lp <rsrc>
linstor vd l
linstor r l
linstor v l
linstor r c <n..> <rsrc>
linstor r d <n..> <rsrc>
…etc
```

What the linstor-proxmox plugin does

- Lets you create "storage" types for Linstor resource groups
- Lets you create VM disks
  - created as "resource definitions" in Linstor
  - random name; an auxiliary property stores the VM ID
    - because Proxmox lets you detach a disk from one VM and attach it to another
- Lets you copy VM disks of other types to/from Linstor
- Ensures a Linstor resource exists on the target node when you migrate a VM (but maybe only a diskless one)
- Manages snapshots
  - LVM_THIN or ZFS / ZFS_THIN storage pools only

But it DOESN'T:

- Let you see which nodes the resources (replicas) are on
- Add or remove replicas
- Therefore, after migration your VM may end up running "disklessly" without you realising
- You need the Linstor CLI or GUI to monitor and fix this
  - Or configure DrbdOptions/auto-diskful, where Linstor will automatically move disk storage after <N> minutes of running disklessly

Note about volume sizes (Linstor on LVM)

- Ask Linstor to create a volume, e.g. 1GiB
  - That's 1024MiB, which in LVM is 256 extents @ 4MiB each
- However, DRBD needs some space for metadata
  - e.g. tracking which parts of the mirror are in sync or outdated
- Therefore Linstor requests a larger volume
  - Next size up is 1028MiB (257 extents)
  - But DRBD takes less than 4MiB for metadata
  - So the remaining space you get is *slightly more than* 1GiB
- This isn't usually a problem – unless you try to squeeze the same VM into exactly 1GiB later!

Getting Linstor and DRBD 9

- Free repository for Debian, including Proxmox plugin
  http://packages.linbit.com/public/ proxmox-<vers> drbd-9
- Free repository for Ubuntu
  https://launchpad.net/~linbit/+archive/ubuntu/linbit-drbd9-stack
- Production-grade repositories for various distros, including RHEL-based ones, available by subscription
- Or you can build from source

Summary

- Linstor is a great middle-ground between local storage and fully distributed storage
- Easy to understand, manage and debug
- Use it for free, or buy binary packages and/or support if you want

Linstor Lab