High Availability Clustering with RHEL 7/8 and z/VM®

Advanced Setup

—

Solution Assurance

Bodo Brand, David Stark

Document version: 03-2022

IBM Z

IBM

Notices and disclaimers

U.S. Government Users Restricted Rights – use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM.

Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. **This document is distributed "as is" without any warranty, either express or implied. In no event, shall IBM be liable for any damage arising from the use of this information, including but not limited to, loss of data, business interruption, loss of profit or loss of opportunity.** IBM products and services are warranted per the terms and conditions of the agreements under which they are provided.

IBM products are manufactured from new parts or new and used parts. In some cases, a product may not be new and may have been previously installed. Regardless, our warranty terms apply.

Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.

Performance data contained herein was generally obtained in a controlled, isolated environments. Customer examples are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.

References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.

Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.

It is the customer's responsibility to ensure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer follows any law.

Notices and disclaimers, *continued*

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products about this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. **IBM expressly disclaims all warranties, expressed or implied, including but not limited to, the implied warranties of merchantability and fitness for a purpose.**

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.

IBM, the IBM logo, ibm.com and IBM Z, IBM z15, and z/VM are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: www.ibm.com/legal/copytrade.shtml

RHEL HA - **Agenda**

- **Introduction**
- **Architecture and Requirements**
- **High Availability Concepts**
  - **Quorum**
  - **Planned/Unplanned Outage**
  - **Resources**

1. **First Steps – Cluster Setup**
2. **Live Guest Relocation (SSI)**
3. **Fencing/STONITH (SBD)**
4. **Cluster Timeouts**
5. **Resources**
   - **Shared Storage**
   - **Floating IP Address**
   - **Your Workload**
6. **Cluster Testing**

Guidance Notes

- Some of the operations must be run on all nodes and some only one one node.
- The "**Run on**" graphic on the right indicate on which of the nodes you must run the command.

- "**#**" at the beginning of the line indicate a privileged bash command.

- "**rhel7/8**" at the beginning of the line indicate that this command should run on RHEL 7.7+ and RHEL 8.3+.

```
rhel7/8# echo "Example command"
```

- The graphic on the right is used for illustration purposes.

Introduction

Introduction - References

Documentation:

Official Red Hat® **documentation**:
- RHEL7: LINK
- RHEL8: LINK

Official Red Hat® **support statements**:
- z/VM specific
- Components in General

Official Red Hat® **version changes**:
- RHEL7: LINK
- RHEL8 Release Notes: LINK

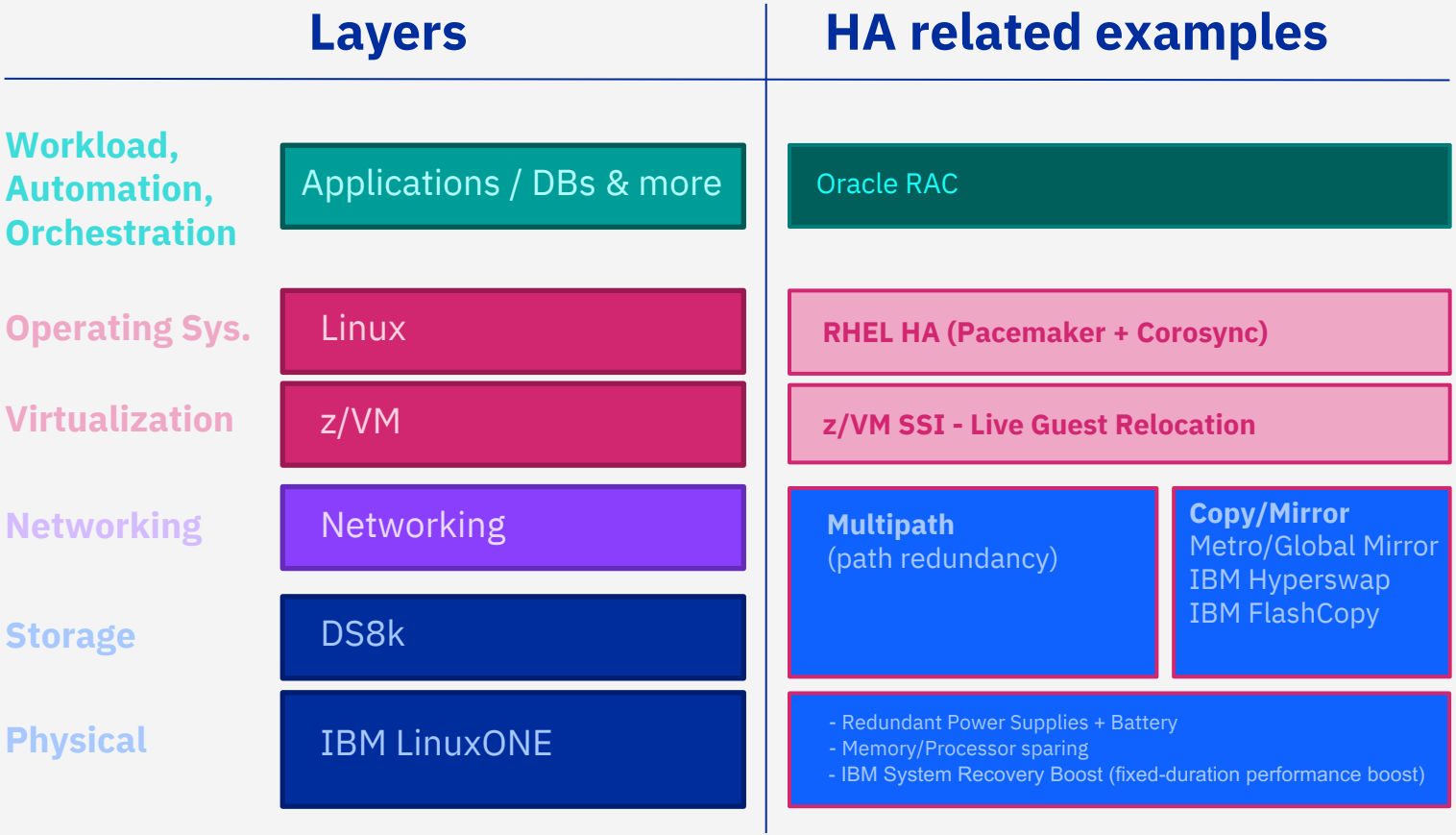
Redbooks® publication - HA on Linux
- HA services or applications **uptime** approaches 100%
- **HA withstands** failures that are caused by **planned or unplanned outages**

Pacemaker documentation

SIDE A

SIDE B

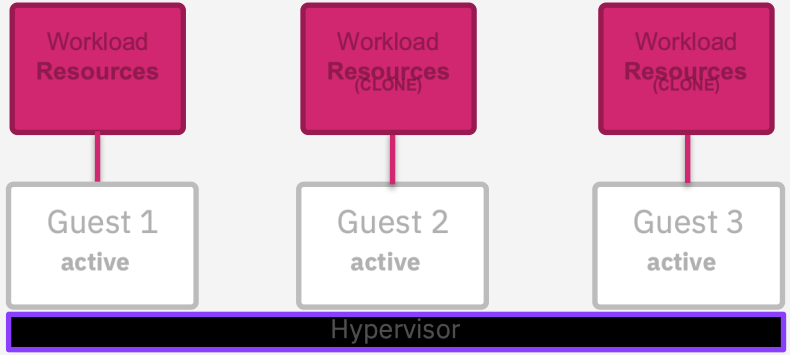
Introduction – High Availability Stack



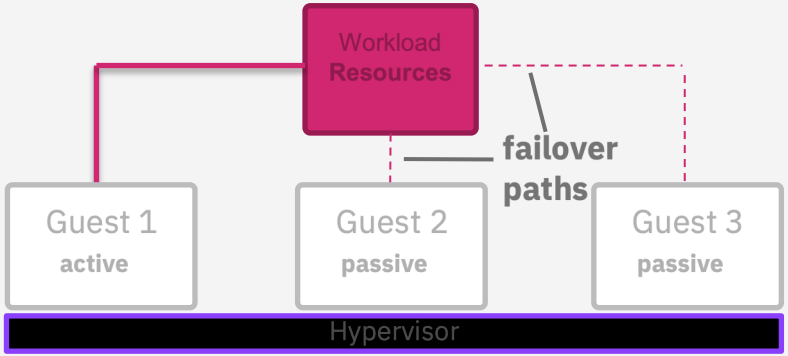
Introduction – HA Complexity

"No Red Hat High Availability"

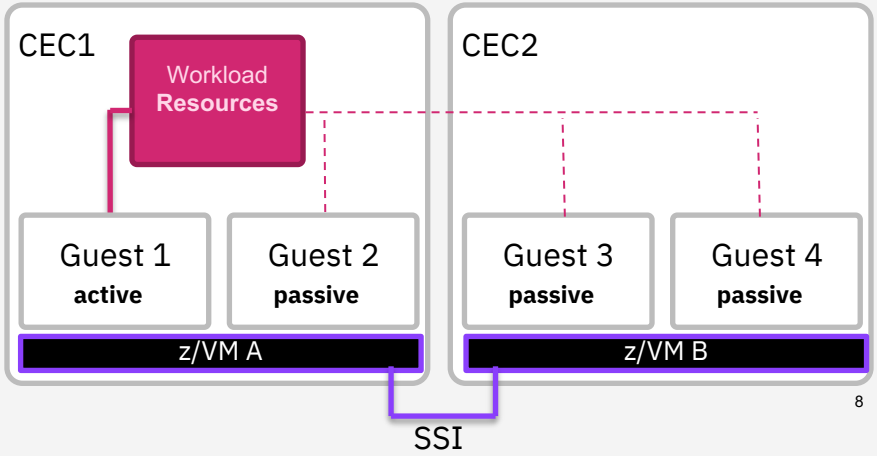
"Active/Active Cluster" often with load balancing



"Active / Passive Failover"



"Cluster spanning multiple sites with SSI"



Architecture and Requirements

Architecture and Details

Setup:

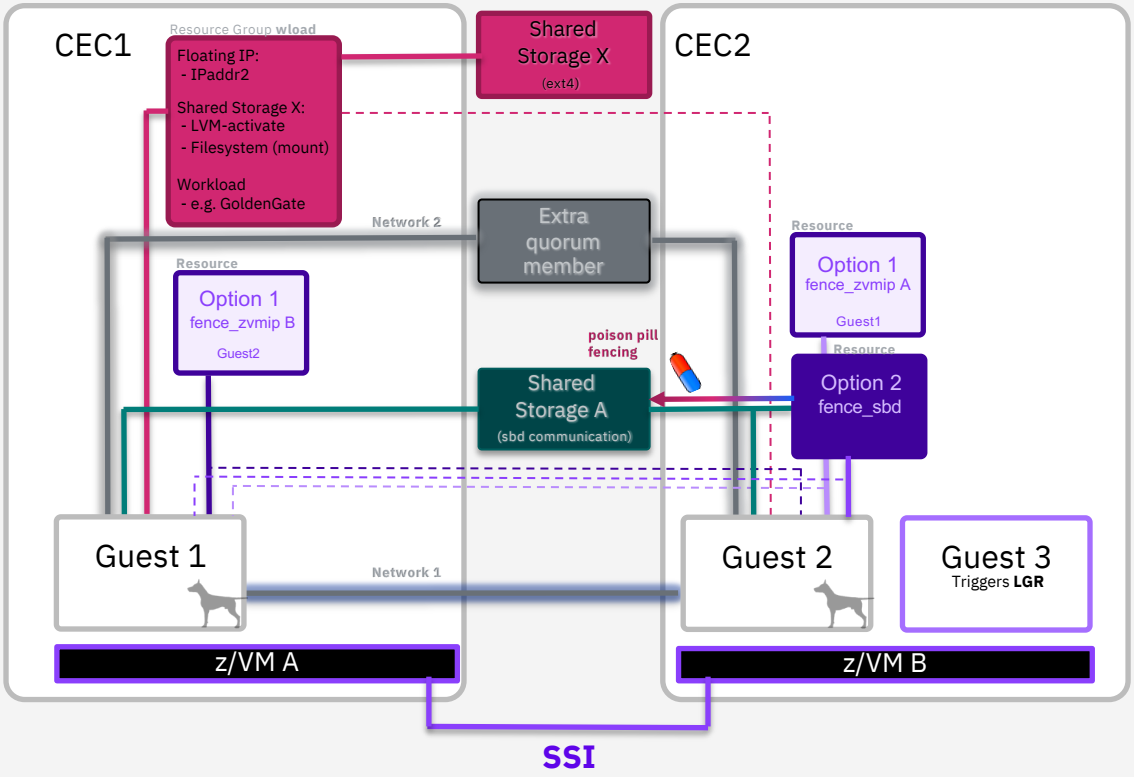
- z15™ and DS8000®
- 1 SSI Cluster (2 z/VMs and 2 guests)
- 1 guest on the third side
- 2+ ECKD DASD Fullpack Minidisk (shared storage)
- 3 ECKD DASD (z/VM guests OS)

z/VM guests:

- On z/VM 7.2
- Distro: RHEL 8.3+ (or 7.7+)

Legend:

- **Watchdog**
- **Poison Pill**
- **Failover Paths**



A more detailed graphic can be found in Appendix B

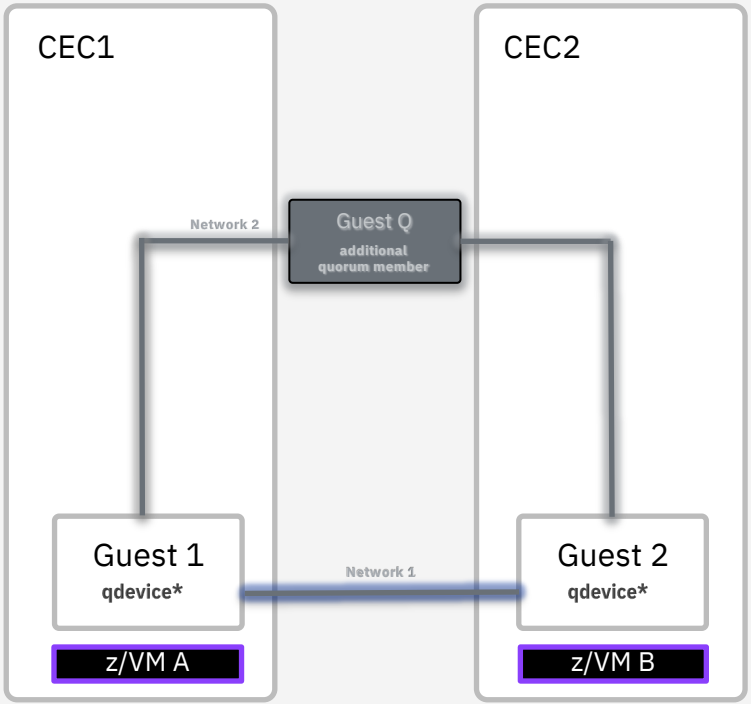
Quorum Concept

Quorum - Concepts

Corosync Votequorum

- Quantity of **votes are assigned to the systems**
- Only when a **majority of votes are present** the cluster **operations are allowed** to proceed.
- With 1 CEC an uneven amount of nodes ensures quorum when 1 node fails.
- With 2 CECs you either need:
  - A third cluster member on a neutral/third side
  - Or a quorum server on a neutral/third side which only votes but does not participate otherwise
- (See next slides for reasons on why a third side is needed)

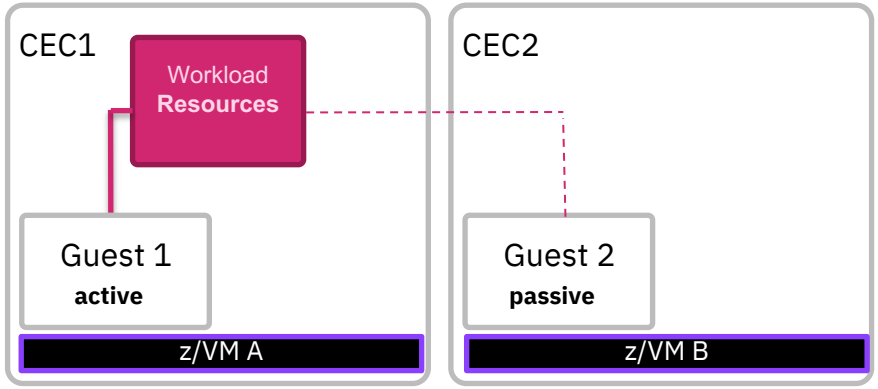
Example 2 CEC quorum setup:



Quorum – 2/3-Nodes Challenges

2-node cluster

split brain challenge

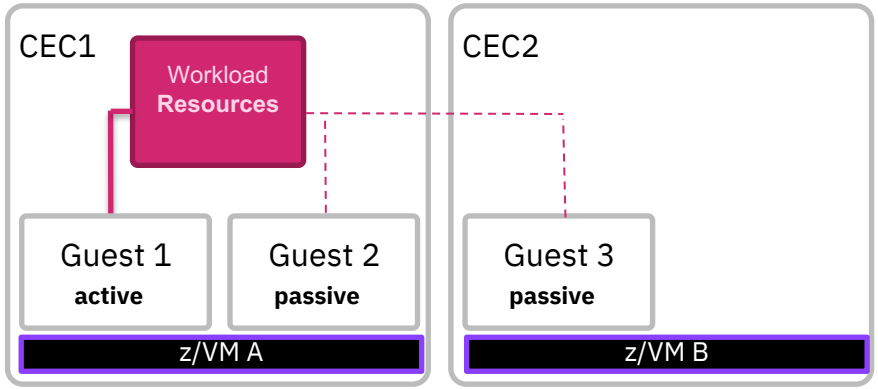


Does not work because in case of a network failure both CECs would be quorate or inquorate which results in running the workload twice or not at all. (**split brain**)

When you use a Tie-Breaker it **does not work** when the wrong CEC dies.

3-node cluster

quorum challenge

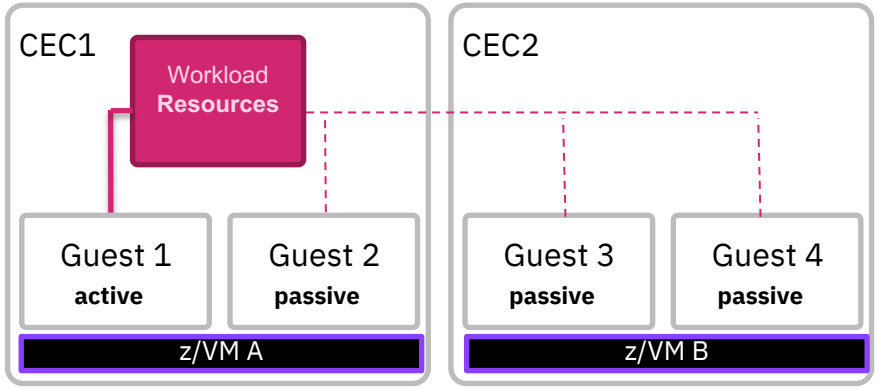


Does not work because in case of CEC1 failure CEC2 alone is not quorate!

Quorum – 4 Nodes Challenges and Solution

4-nodes cluster

split brain + quorum challenge

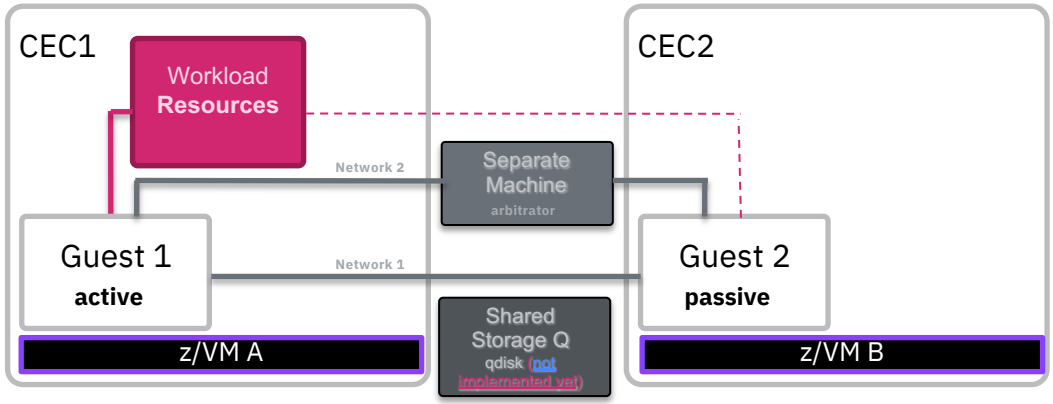


Does not work because in case of a network failure in between the CECs both CECs are not quorate.
A tiebreaker would only help one side.

Solution:

2 / 4 nodes + additional quorum member

solves challenges



A **separate machine** which participates as quorum member **prevents split brain** and other quorum challenges.

Planned / Unplanned Outage Concept

Planned / Unplanned Outage - Concepts

Planned Outage

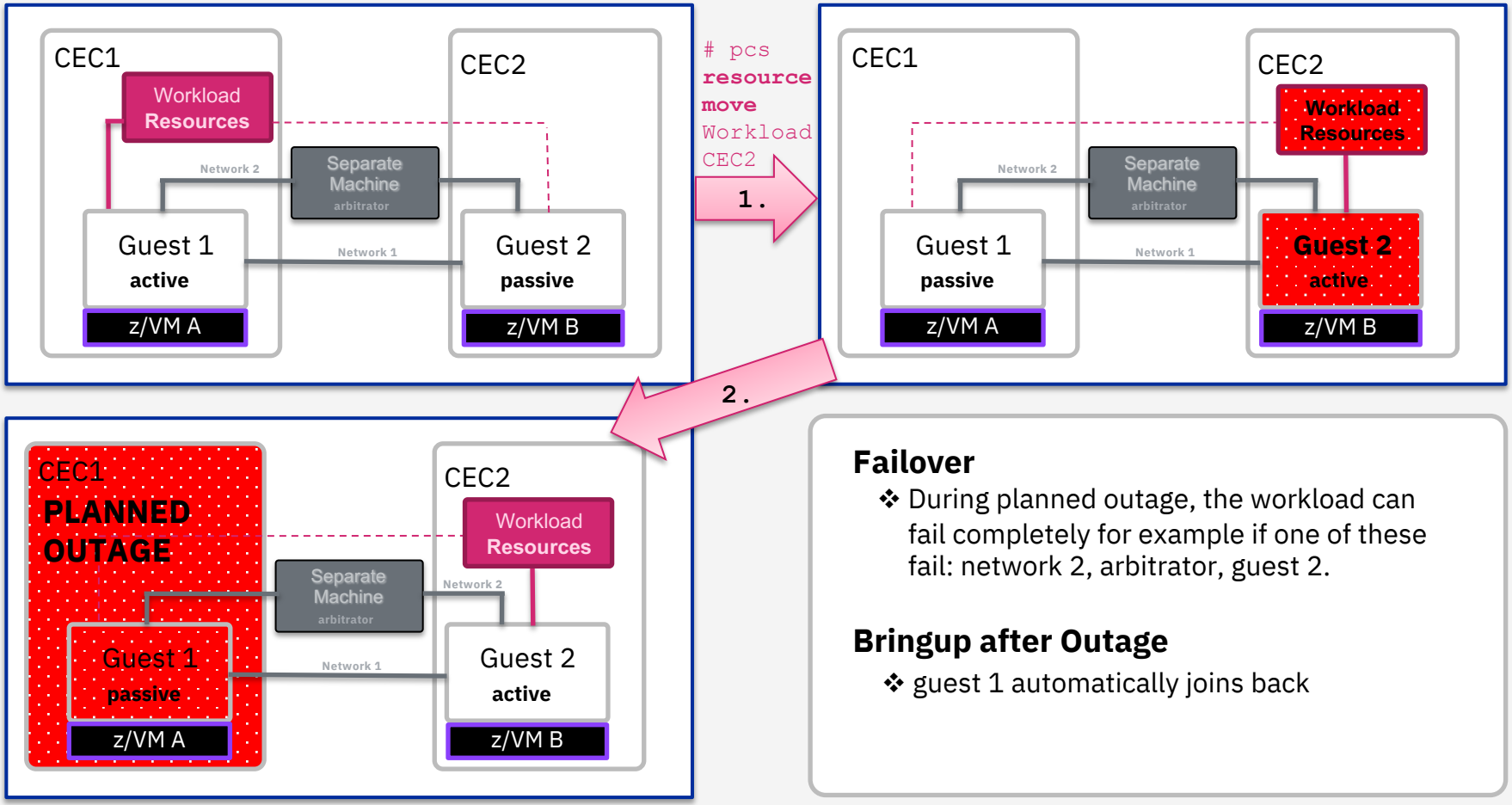
- Red Hat HA allows you to manually trigger the movement of workload to the other side
- When a SSI cluster is in use you can move the whole guest to the other side with Live Guest Relocation (LGR).
  - Note that only the LGR of passive guests might be supported (LINK) and therefore a movement of the workload in the cluster might be needed before moving.

Unplanned Outage

- Red Hat HA automatically fails over in case of failure.
- Additionally, the node might be killed/fenced to make sure the resources are released before the Resource is brought up on the other node.

➢ See following slides for graphic illustrations.

Planned Outage - RHEL HA



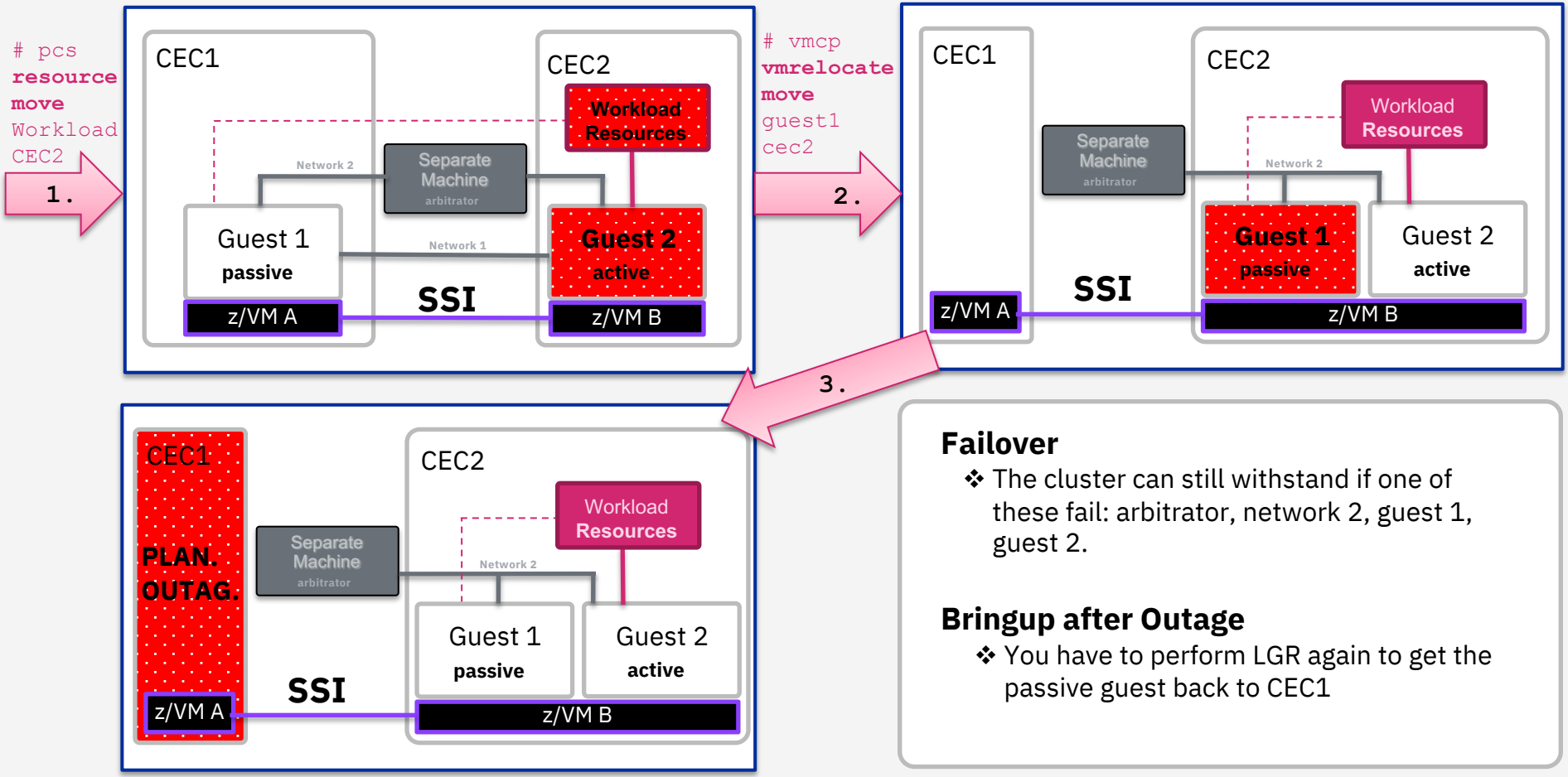
Failover

- During planned outage, the workload can fail completely for example if one of these fail: network 2, arbitrator, guest 2.

Bringup after Outage

- guest 1 automatically joins back

Planned Outage - RHEL HA + SSI



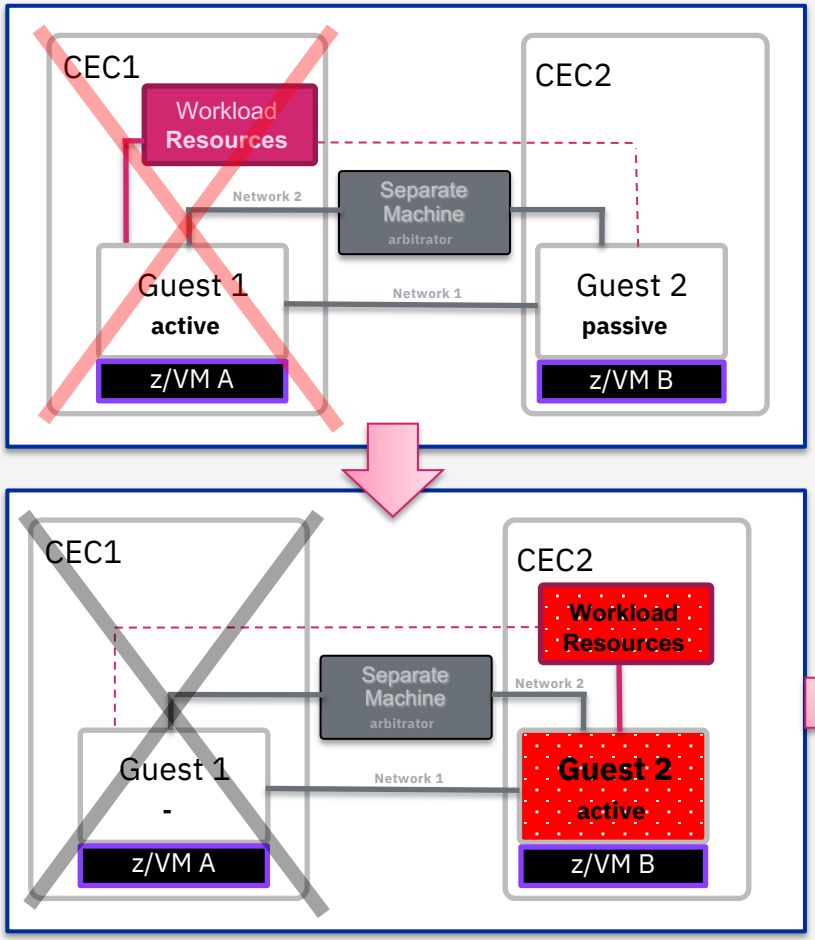
Failover

- The cluster can still withstand if one of these fail: arbitrator, network 2, guest 1, guest 2.

Bringup after Outage

- You have to perform LGR again to get the passive guest back to CEC1

UN-Planned Outage - RHEL HA



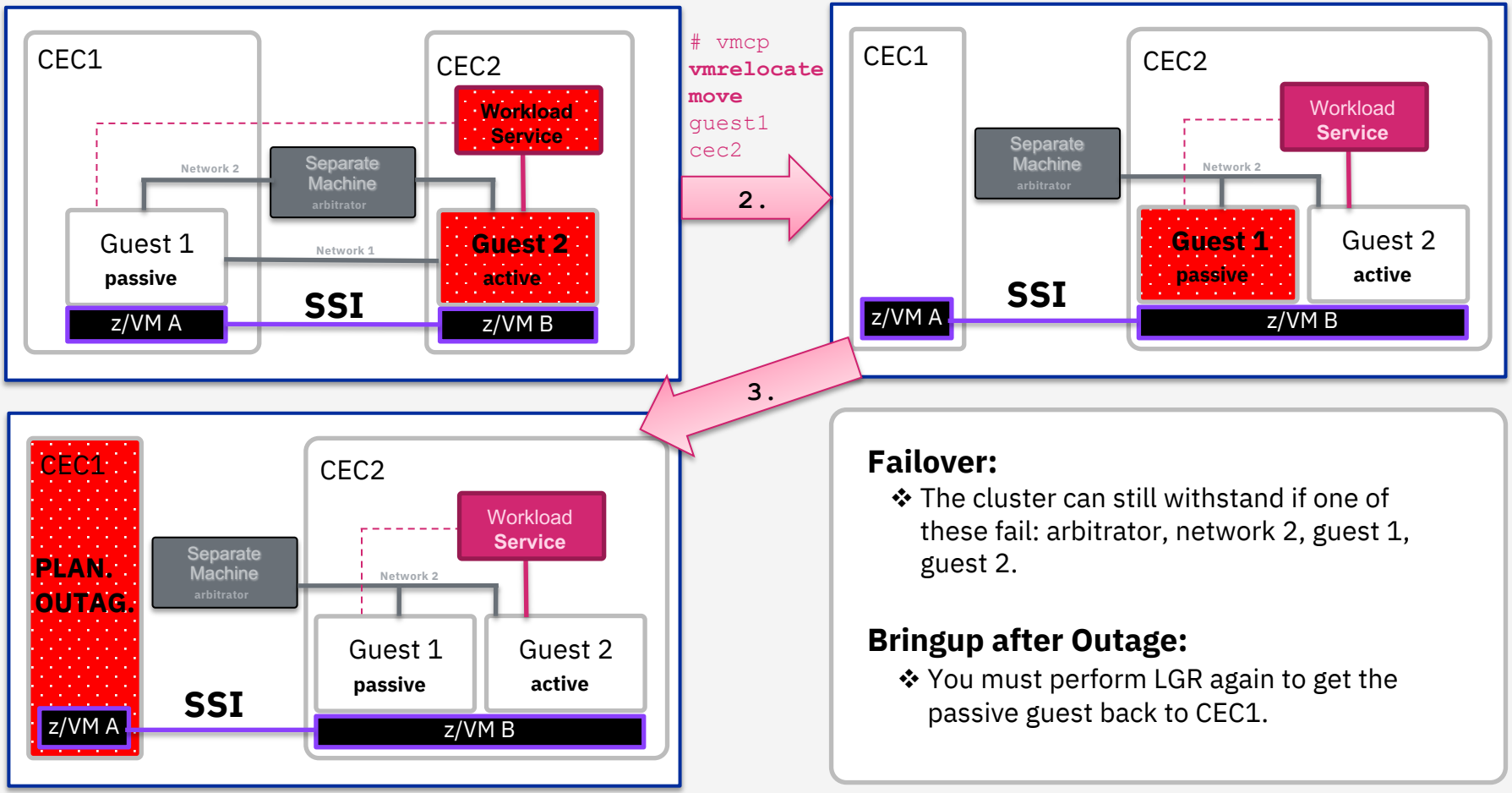
Failover

- Workload is automatically moved to guest 2 on CEC2
- Fencing of guest 1 fail
- **No big differences to non-SSI Cluster**

Bringup after Outage:

- Guest 1 automatically joins back

UN-Planned Outage - RHEL HA + SSI



Failover:

- The cluster can still withstand if one of these fail: arbitrator, network 2, guest 1, guest 2.

Bringup after Outage:

- You must perform LGR again to get the passive guest back to CEC1.

Resources Concept

(Managed) Resources

A cluster contains one or multiple resources. Each resource has following properties:

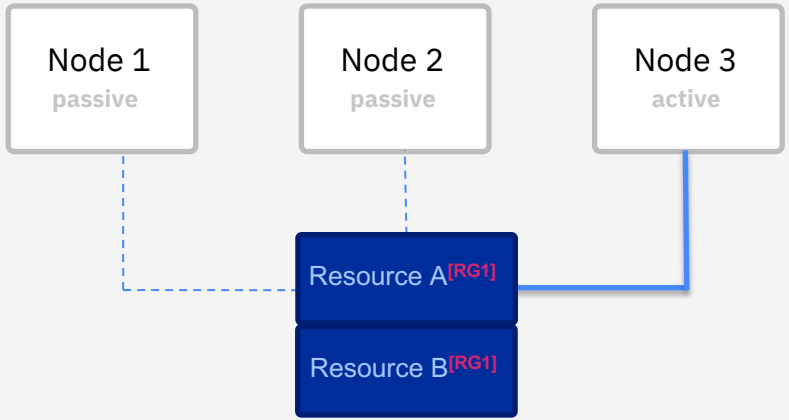
- type (e.g., apache) and resource identifier (e.g., Website)

Resource:

- Implemented as Resource Agent (RA)
- executable/service conforming to a standard (usually ocf **or systemd**)
- handles all **operations: (start, stop, monitor)**
- **attributes** for configuration (e.g., configfile=a.conf)
- **constraints** (**location, order, colocation**)

Resource Groups [RG1]:

- Resources in a resource group **start and stop in order**
- When one of the resources moves in the group, the other resources in that group move with it



Predefined Ressources:

- List all predefined Resources

```
# pcs resource list
```

- Look into the Resource docu

```
# pcs resource describe x
```

- Add Resources to cluster

```
# pcs resource create ...
```

Define your own Ressource

- Article from Red Hat ®: LINK
- OCF compliant RA: LINK
  - XML definition file
  - Operations implemented in any programming language
  - Exit codes standardized
- Add to pacemaker search location: `/usr/lib/ocf`

First Steps – Cluster Setup

Qnetd Server and Network Setup

Separate Machine Informations:

- Use a guest which is running on a **third side**
  - **Avoid a single point of failure** by having a **separate physical network** connecting both guests with the **third side** Guest Q

Separate Network Setup

- Example commands to create a second/new network interface (if-2)

Guest 1
Guest 2
qnetd

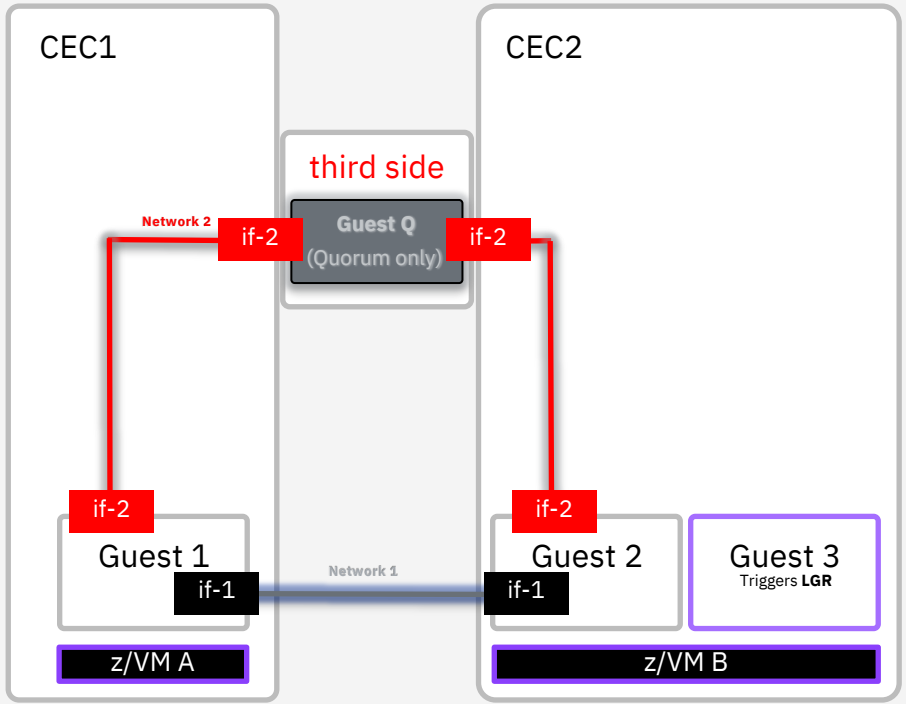
```
rhel7/8# cio_ignore -r 0.0.b100-0.0.b102
rhel7  # vim /etc/zipl.conf
rhel8  # vim /boot/loader/entries/*.conf
```

add to parmline:

```
rd.znet=qeth,0.0.b100,0.0.b101,0.0.b102
rhel7/8# zipl -V
rhel7/8# reboot
rhel7/8# cd /etc/sysconfig/network-scripts
rhel7/8# cp ifcfg-encbdf0 ifcfg-enccw0.0.b100
rhel7/8# vim ifcfg-enccw0.0.b100
```

do setup for second network interface

```
rhel7/8# reboot
```



RHEL HA Installation and Firewall Configuration

Step 1.0 – RHEL HA Installation

Run on: Guest 1, Guest 2, qnetd

```
rhel7/8# subscription-manager register --auto-attach

rhel8  # dnf config-manager --set-enabled \
           rhel-8-for-s390x-highavailability-rpms
rhel7  # yum install -y yum-utils
rhel7  # yum-config-manager --enable \
           rhel-ha-for-rhel-7-for-system-z-rpms

rhel7/8# yum update -y
rhel7/8# yum install -y pcs
```

Run on: Guest 1, Guest 2

```
rhel7/8# yum install -y pacemaker \
                        fence-agents-all \
                        corosync-qdevice
```

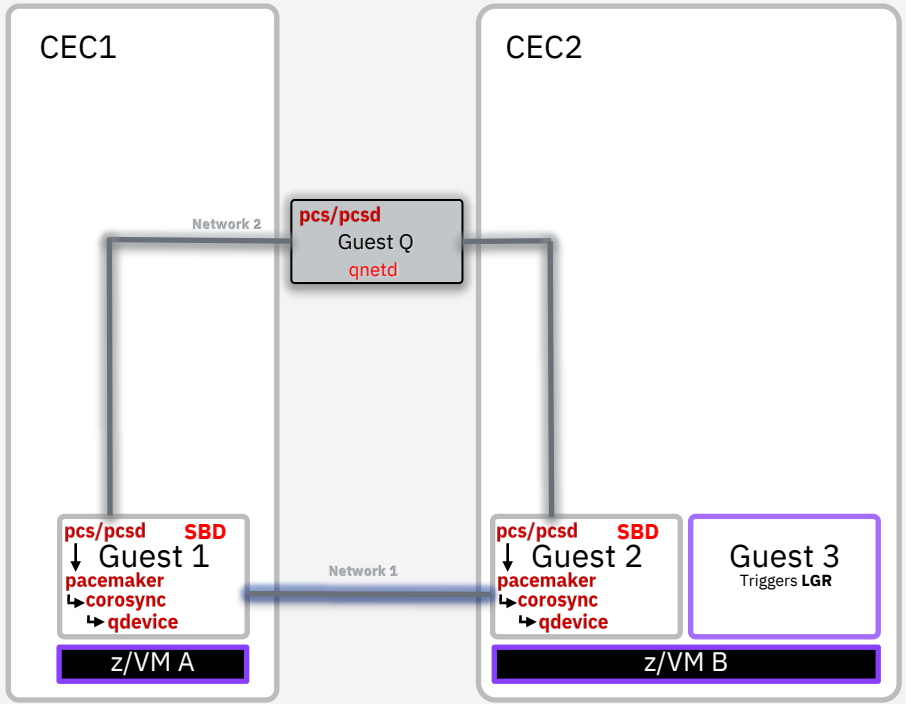
Run on: qnetd

```
rhel7/8# yum install -y corosync-qnetd
```

Step 1.1 – Firewall Configuration

Run on: Guest 1, Guest 2, qnetd

```
rhel7/8# firewall-cmd --permanent \
           --add-service=high-availability
rhel7/8# firewall-cmd --reload
```



RHEL HA Cluster Setup

Step 1.2 – Prepare Cluster

Run on: Guest 1, Guest 2, qnetd

❖ Create a linux® user used by the cluster

```
rhel7/8# passwd hacluster
```

Note:
- Same password for each node recommended.

❖ Enable the cluster controlling and configuration daemon

```
rhel7/8# systemctl enable pcsd.service --now
```

Step 1.3 – Auth Nodes

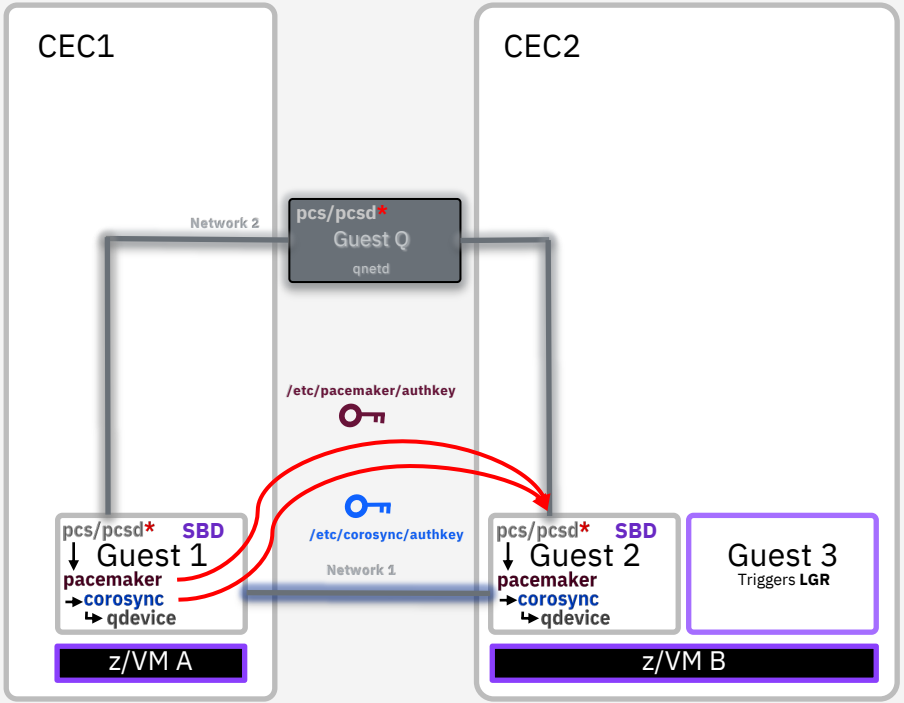
Run on: Guest 1

```
rhel7# pcs cluster auth guest1 guest2 qnetd
rhel8# pcs host auth guest1 guest2 qnetd
Username: hacluster
Password: …
```

Step 1.4 – Setup Cluster

Run on: Guest 1

```
rhel7# pcs cluster setup --name \
  my_cluster guest1 guest2
rhel8# pcs cluster setup \
  my_cluster guest1 guest2
```



Startup of the Cluster and Status

Step 1.5 – Start and Enable Services

Run on: Guest 1

```
rhel7/8# pcs cluster start --all
rhel7/8# pcs cluster enable --all
```

Step 1.6 – Status and CLI Tools

Run on: Guest 1

- The pcs CLI tool allows you to configure the cluster (pacemaker + corosync) and view the status
  - Full cluster status:

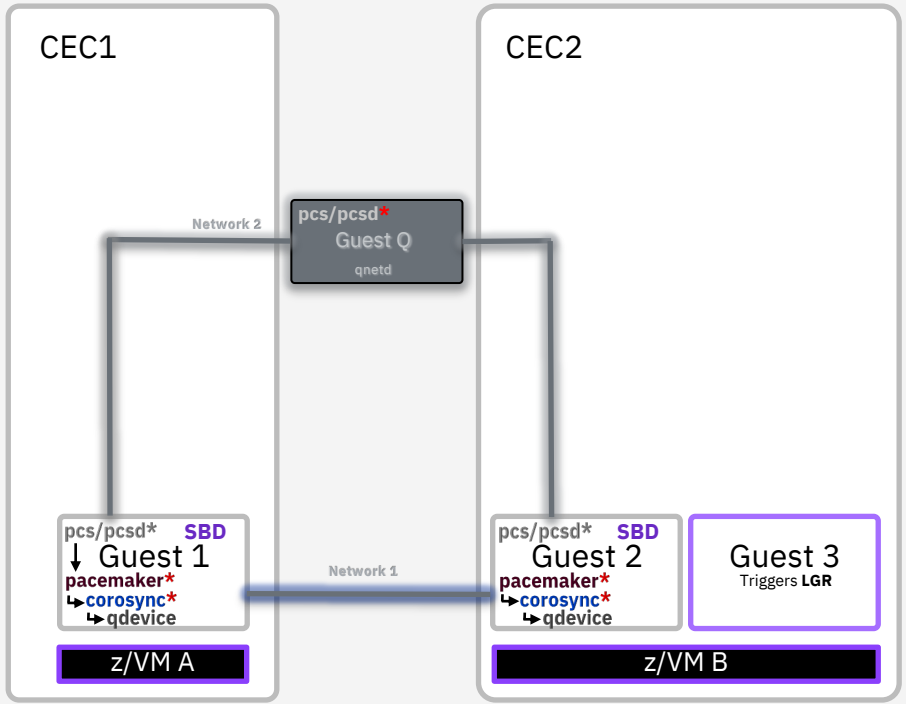
    ```
    rhel7/8# pcs status --full
    ```
  - Pacemaker configuration

    ```
    rhel7/8# pcs cluster cib
    ```
- Pacemaker and corosync have their own cli tools:
  - Pacemaker configuration

    ```
    rhel7/8# cibadmin -Q
    ```
  - Show corosync object database

    ```
    rhel7/8# corosync-cmapctl
    ```
  - Dump live corosync flight data

    ```
    rhel7/8# corosync-blackbox
    ```



Start Qnetd Service and Qdevice Daemon

Step 1.7 – Start Qnetd Service

Run on: qnetd

```
rhel7/8# pcs qdevice setup model net --enable –start
rhel7/8# systemctl status corosync-qnetd
```

Step 1.8 – Setup Qdevice Daemon

Run on: Guest 1

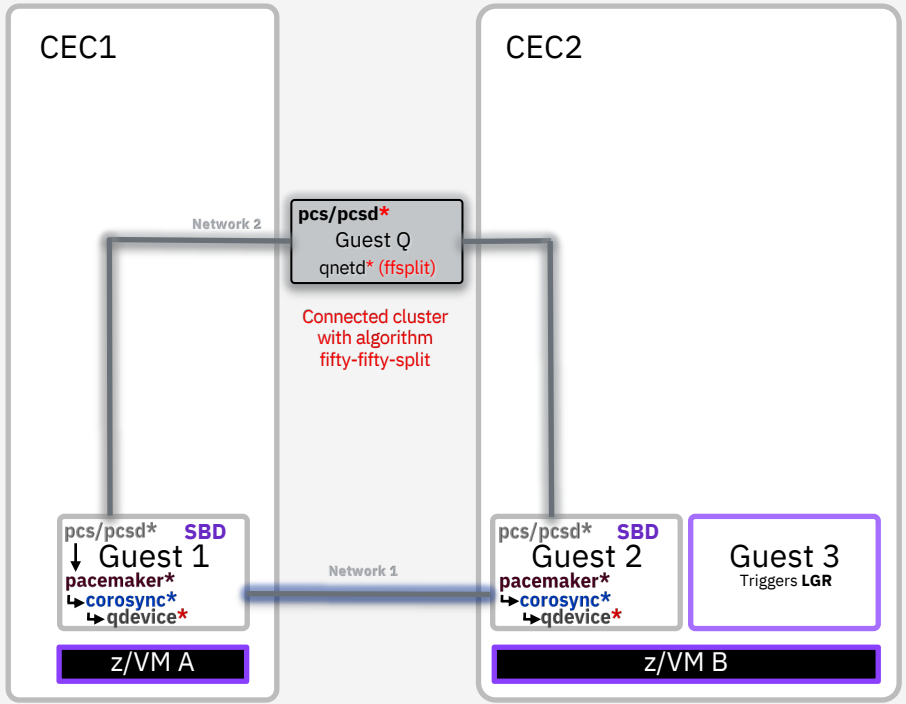
- Add qdevice daemon to cluster (connected to qnetd)

```
rhel7/8# pcs quorum device add model net \
          host=xxx.xxx.xxx.xxx algorithm=ffsplit
```

- Check status of qdevice daemon

```
rhel7/8# /usr/sbin/corosync-qdevice-tool -s -v
```

Note:
- You can connect multiple clusters to Guest Q.



Live Guest Relocation (SSI)

Live Guest Relocation

Step 2.0 – Live Guest Relocation

Run on: Guest 3

- Check on which side a guest is running

```
# vmcp q guestX extend at all | cut -d" " -f1
```

- Test Live Guest Relocation (**from same side**)

```
# vmcp vmrelocate test guest2 ZVM_A
```

- Test Live Guest Relocation (**from other side**)

```
# vmcp at ZVM_A cmd vmrelocate test guest1 ZVM_B
```

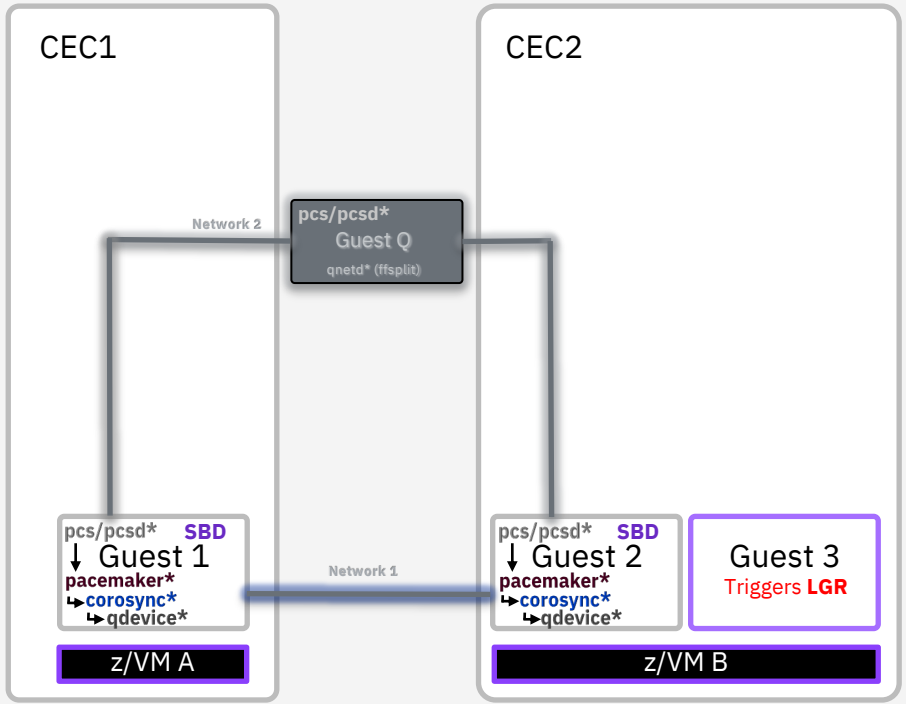
Note: You might have to detach certain z/VM disks for live guest relocation to work e.g.:

```
# vmcp 'DET 0190'
# vmcp 'DET 019D'
# vmcp 'DET 019E'
# vmcp 'DET 0592'
```

- Perform Live Guest Relocation (from same side)

```
# vmcp vmrelocate move guest2 ZVM_A
```

Note:
- Make sure the z/VM userid for guest3 has enough permissions to trigger LGR.



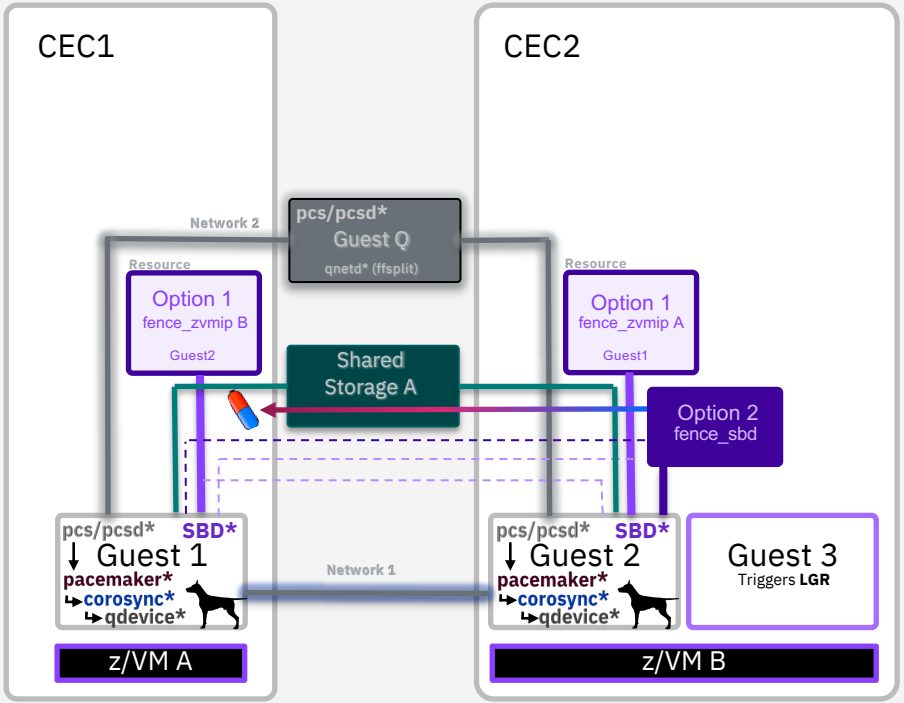
Fencing / STONITH (SBD)

Fencing / STONITH concept

Concept of Fencing/STONITH

- Ensures that it is not possible for a guest to run resources if the guest is not intended to do so
- Different fencing methods are available
  - **via z/VM SMAPI**: (option 1) In a 2 CEC setup you have to use 2 fence-agents specifying the other side. This fence agent is not SSI aware which means you would have to change both fence-agents every time you do LGR. (Instructions)
  - **via SBD**: (option 2) SBD watches the cluster health locally and triggers self fencing if needed. Additionally, SBD watches a shared disk where the fence-agent can write a poison pill to which also triggers fencing.

➢ SBD is set up in the following slides as this method works with 2 CECs and SSI



self-fencing / watchdog-timeout fencing

Watchdogs must be fed otherwise they kill the guest.

AND

poison pill fencing

Watchdogs fed with a poison pill kills the guest!

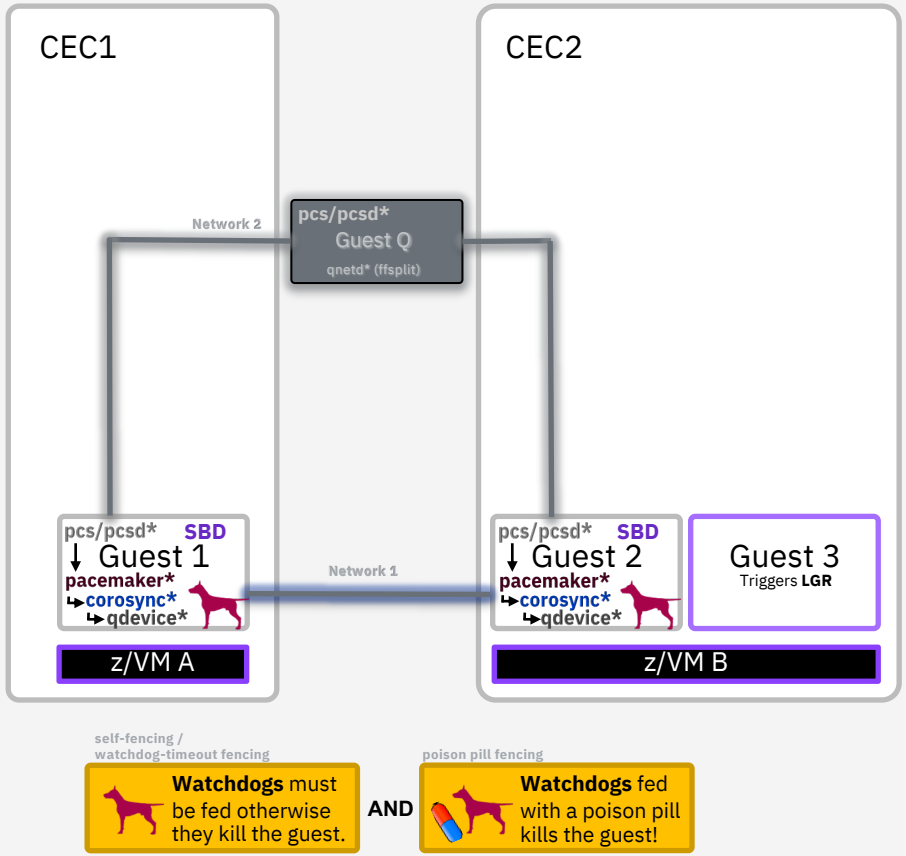
Setup SBD Fencing - Watchdog

Step 3.0 – Setup Watchdog

Run on:
- Guest 1
- Guest 2

- Enable watchdog kernel module
```
# modprobe diag288_wdt
```
- Show watchdog
```
# wdctl
```
- Test if watchdog works (should halt after 30s)
```
# timeout=30; echo; echo; echo "Opening watchdog device to start countdown from $timeout"; echo start > /dev/watchdog; echo "Counting down. The system should halt near 0 if the watchdog is functional."; while true; do if [ $timeout -le 0 ]; then echo "Timeout expired.  System should be halting."; else echo "$timeout..."; fi; timeout=$((timeout-1)); sleep 1; done;
```
- Check z/VM command executed on timeout
```
# cat /sys/module/diag288_wdt/parameters/cmd
```
- Make watchdog loading persistent
```
# echo "diag288_wdt" > /etc/modules-load.d/watchdog.conf
```

Note:
- Watchdog timeout cannot be lower than 15s! (link)



Setup SBD Fencing - Shared Storage

Step 3.1 – Setup SBD Shared Storage

- Shared Storage is already assumed to be setup. **(see appendix A for a cheat sheet)**

Run on: Guest 1

- Create SBD header on disk partition

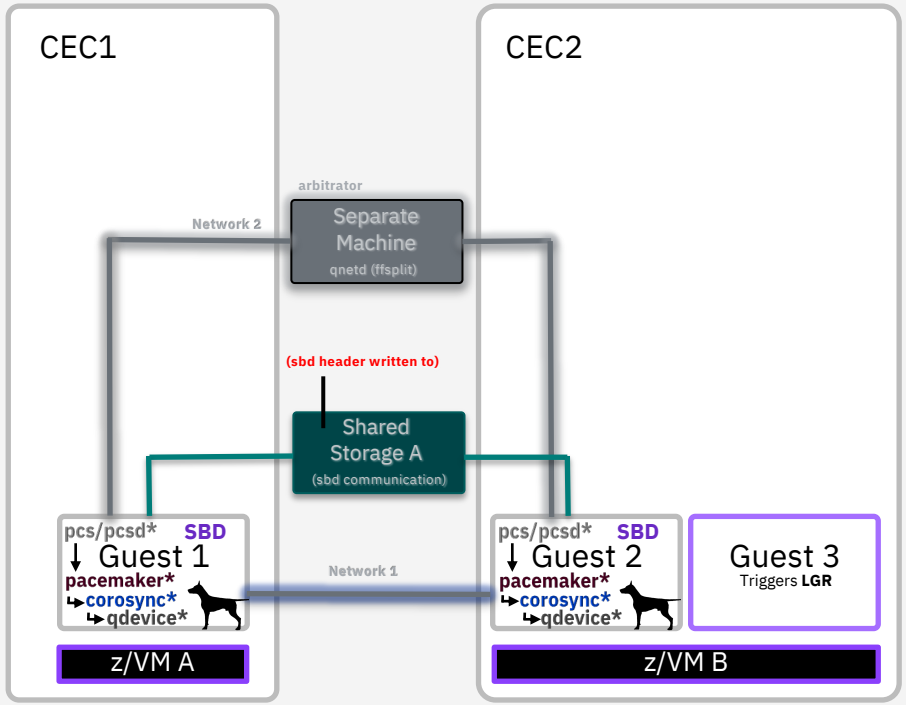
```
rhel7/8# pcs stonith sbd device setup \
    --device=/dev/disk/by-path/ccw-0.0.0200-part1 \
    watchdog-timeout=15 \
    msgwait-timeout=30
```

- Alternative command to create SBD header

```
# sbd -v \
      -d /dev/disk/by-path/ccw-0.0.0200-part1 \
      create
```

- Show SBD header

```
rhel7/8# pcs stonith sbd status --full
```



self-fencing / watchdog-timeout fencing

Watchdogs must be fed otherwise they kill the guest.

AND

poison pill fencing

Watchdogs fed with a poison pill kills the guest!

Setup SBD Fencing - Daemon

Step 3.2 – Setup SBD

Run on: Guest 1

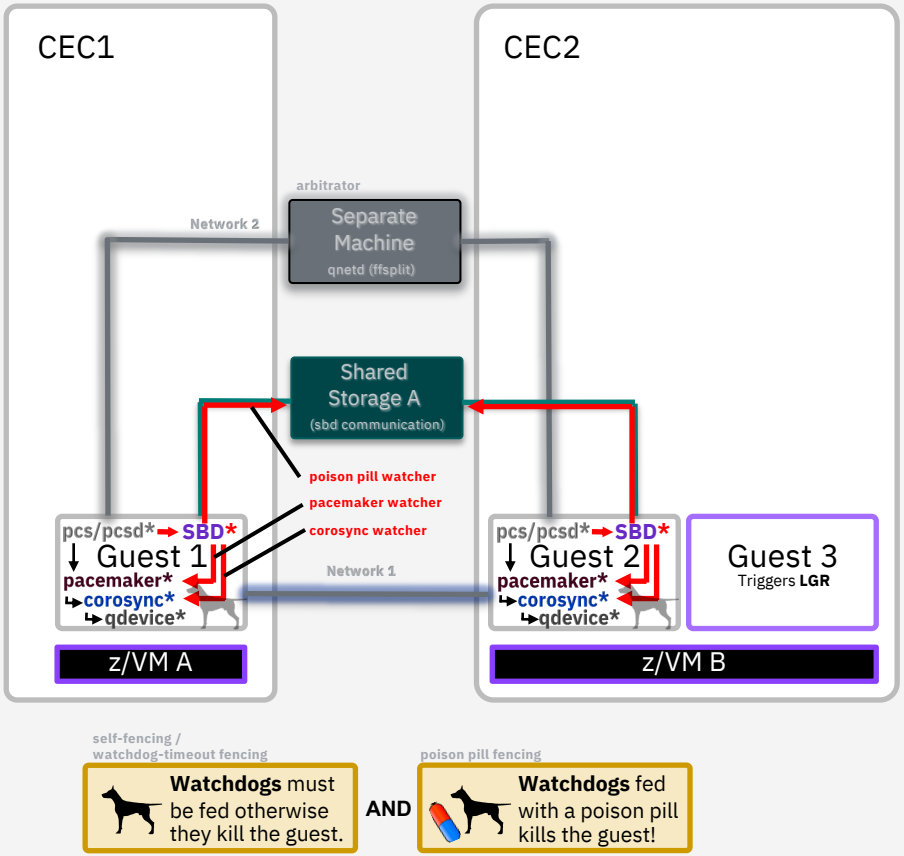
❖ Enable SBD systemd daemon in cluster

```
rhel7# pcs stonith sbd enable \
       --watchdog=/dev/watchdog \
       --device=/dev/disk/by-path/ccw-0.0.0200-part1 \
       SBD_DELAY_START=yes SBD_WATCHDOG_TIMEOUT=15
rhel8# pcs stonith sbd enable \
       watchdog=/dev/watchdog \
       device=/dev/disk/by-path/ccw-0.0.0200-part1 \
       SBD_DELAY_START=60 SBD_WATCHDOG_TIMEOUT=15
```

Note:
- SBD_* are environment variables for the SBD systemd service.
- SBD_WATCHDOG_TIMEOUT **only applies** when SBD runs in diskless mode.
-> when disks are defined the watchdog timer written to the disk header is used.
- **The diag288 watchdog minimum timeout is 15 seconds. (LINK)**
- SBD_DELAY_START postpones the start of the pacemaker systemd daemon
- SBD_DELAY_START should be longer then: corosync token timeout (5) + consensus timeout (6) + pcmk_delay_max (0) + msgwait (30) = 41 seconds. Otherwise, you might run into the issue that pacemaker starts with exit code 100.

❖ Restart cluster

```
rhel7/8# pcs cluster stop --all
rhel7/8# pcs cluster start --all
```



Setup SBD Fencing - Fence Agent

Step 3.4 – Setup SBD Fence Agent

Run on: Guest 1

❖ Show default power_timeout which indicates how long the fencing process waits before pacemaker must be up again after fencing

```
rhel7/8# fence_sbd -o metadata|grep -A 2 power_timeout
```

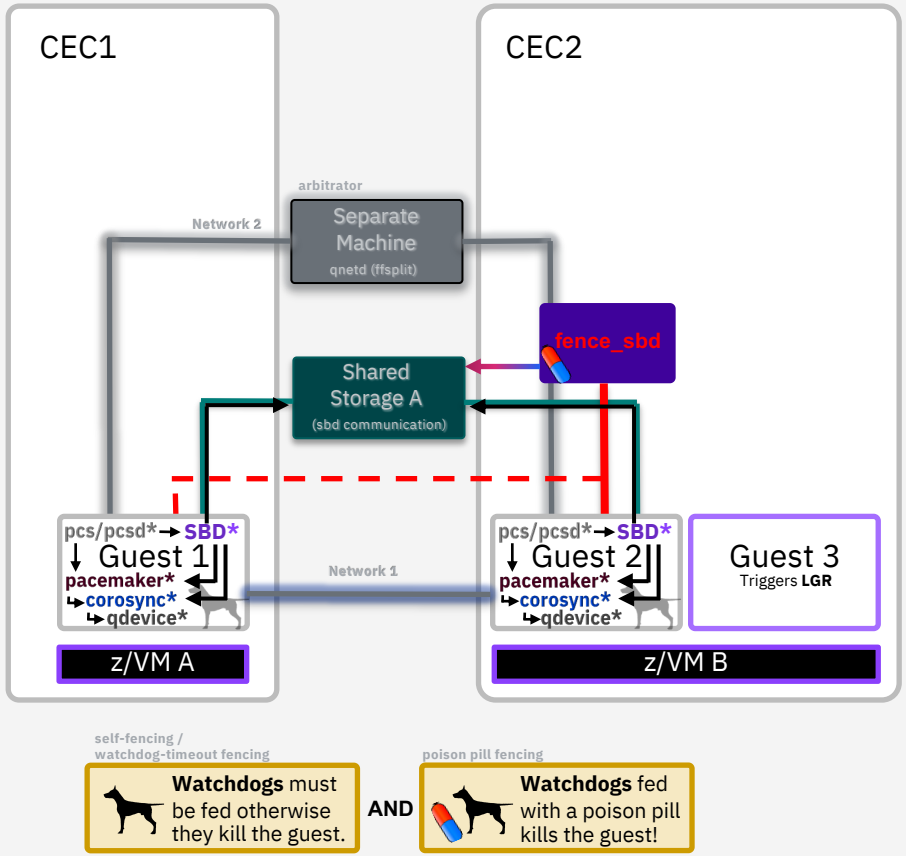
❖ Create SBD fence agent and set power_timeout

```
rhel7/8# pcs stonith create sbd fence_sbd \
     devices="/dev/disk/by-path/ccw-0.0.0200-part1" \
     power_timeout=45
```

Note:
- power_timeout should be bigger then msgwait timeout

❖ Show settings of SBD fence agent

```
rhel7  # pcs stonith show sbd
rhel8  # pcs stonith config
```



Setup SBD Fencing - Testing

Step 3.5 – Test SBD fencing

Run on: Guest 2

❖ Test sending of messages through A

```
rhel7/8# sbd -d /dev/disk/by-path/ccw-0.0.0200-part1 \
      message guest1 test
```

Run on: Guest 1

❖ Look at the Log of the SBD systemd service

```
rhel7/8# journalctl -u sbd -f
```

Run on: Guest 2

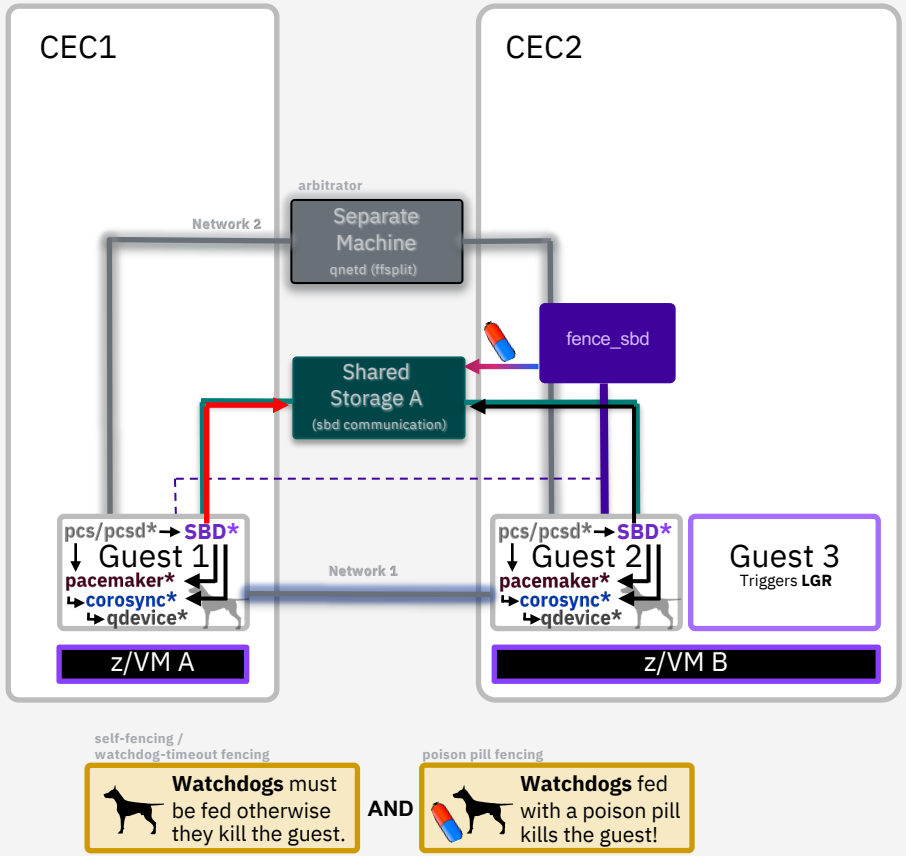
❖ Send poison pill from guest2 to guest1

```
rhel7/8# time pcs stonith fence guest1
```

Note:
- other system should reboot and the pacemaker systemd service should be delayed by the SBD systemd service by msgwait-timeout seconds.

❖ Helpful debugging options
- ❖ Increase SBD verbosity: add "-v" to SBD_OPTS in /etc/sysconfig/sbd
- ❖ Look at systemd startup

```
rhel7/8# systemd-analyze critical-chain
```



Cluster Timeouts

Setup Timeouts - Totem

Step 4.0 – Setup Totem Timeout

Run on: Guest 1

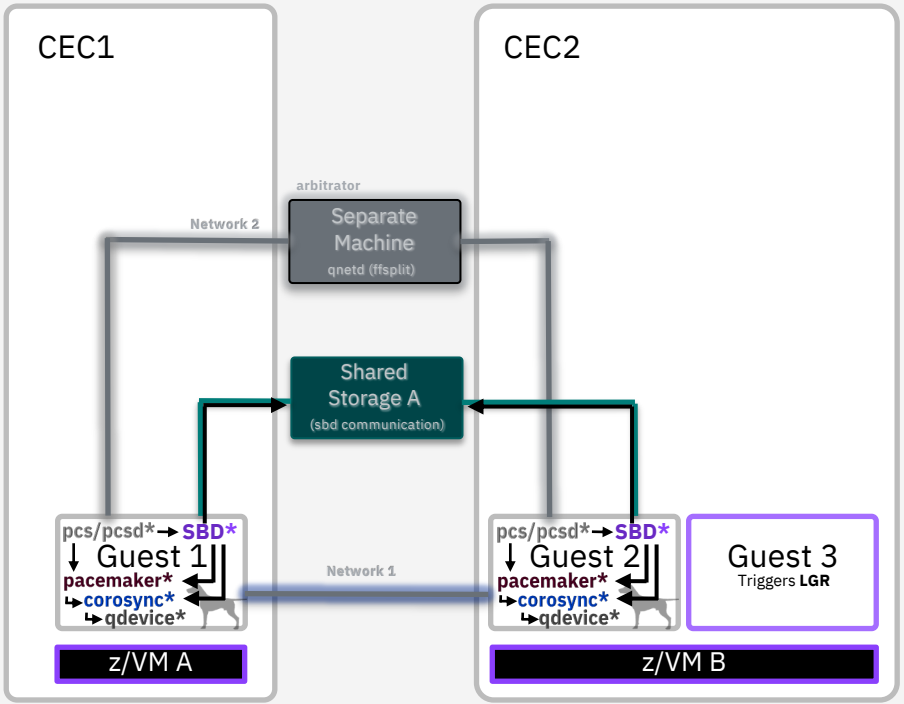
- "totem token timeout" specifies in milliseconds until a token loss is declared

```
rhel7# vim /etc/corosync/corosync.conf
-> add `token: 5000` in totem section
rhel7# pcs cluster sync
rhel7# pcs cluster reload corosync
rhel7/8# pcs cluster config update \
            totem token=5000
```

Note:
- For totem token limits check out the corosync support policies

- Check totem token timeout

```
rhel7/8# corosync-cmapctl | \
         grep "runtime.*totem.token "
```



Shared Storage

Shared Storage – Create Resources

Step 5.1 – Add LVM Resource to Cluster

Run on: Guest 1

- Shared Storage is already assumed to be setup **(see appendix A for a cheat sheet)**

- Change LVM option in /etc/lvm/lvm.conf to

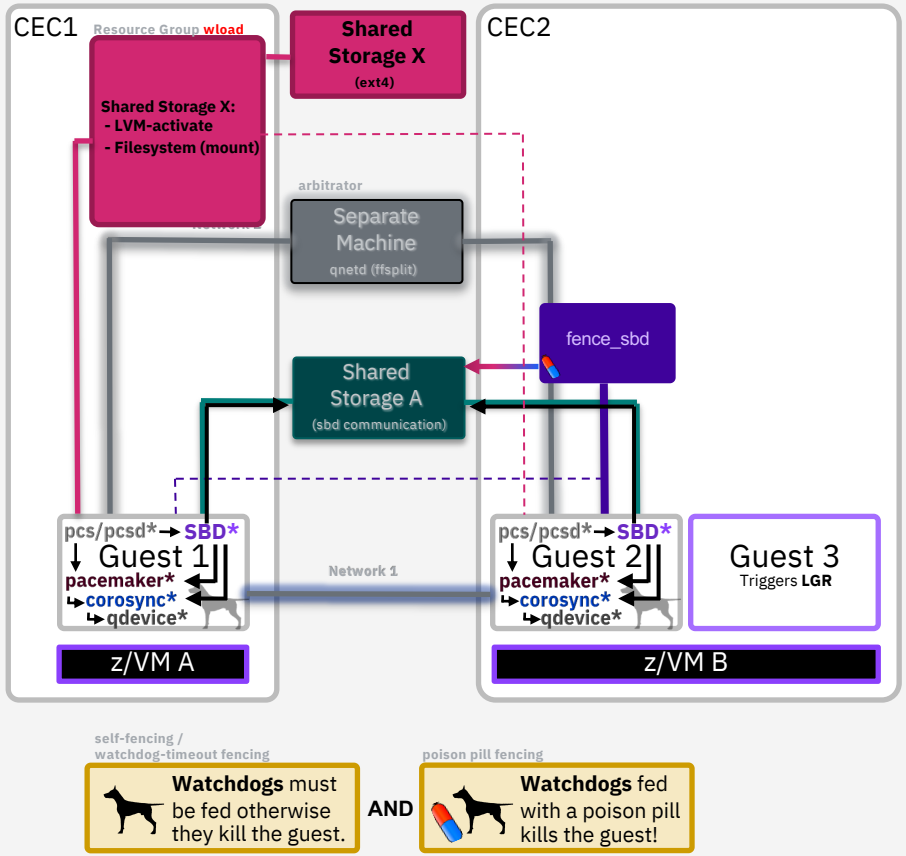
```
system_id_source = "uname"
```

- Create logical volume

```
rhel7/8# pvcreate \
           /dev/disk/by-path/ccw-0.0.0200-part1
rhel7/8# vgcreate my_vg \
           /dev/disk/by-path/ccw-0.0.0200-part1
rhel7/8# lvcreate -n my_lv -l 100%FREE my_vg
rhel7/8# mkfs.ext4 /dev/my_vg/my_lv
```

- Add required Resources to cluster

```
rhel7/8# pcs resource create my_lvm LVM-activate \
           vgname=my_vg \
           vg_access_mode=system_id \
           --group wload
rhel7/8# pcs resource create my_fs Filesystem \
           device="/dev/my_vg/my_lv" \
           directory="/var/www" fstype="ext4" \
           --group wload --after my_lvm
```



Floating IP Address

Floating IP Address – Create Resource

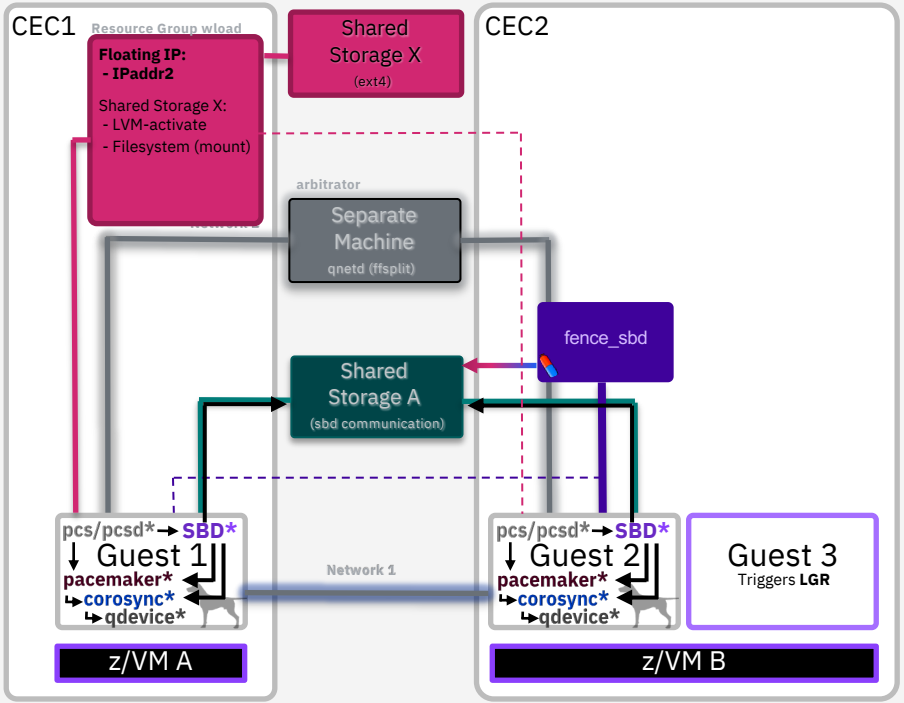
Step 5.2 – Setup Floating IP Address

Run on: Guest 1

❖ Add floating IP resource to cluster

```
rhel7/8# pcs resource create ClusterIP \
         ocf:heartbeat:IPaddr2 \
         ip="${FLOATING_IP_ADDRESS}" \
         --group wload \
         --before my_lvm
```

Note:
- Floating IP addresses is like a static IP address in the network.
- `${FLOATING_IP_ADDRESS}` is a placeholder.



self-fencing / watchdog-timeout fencing

Watchdogs must be fed otherwise they kill the guest.

AND

poison pill fencing

Watchdogs fed with a poison pill kills the guest!

Your Workload

Add your Cluster Workload

Step 5.3 – Add Workload to Cluster

Run on: Guest 1

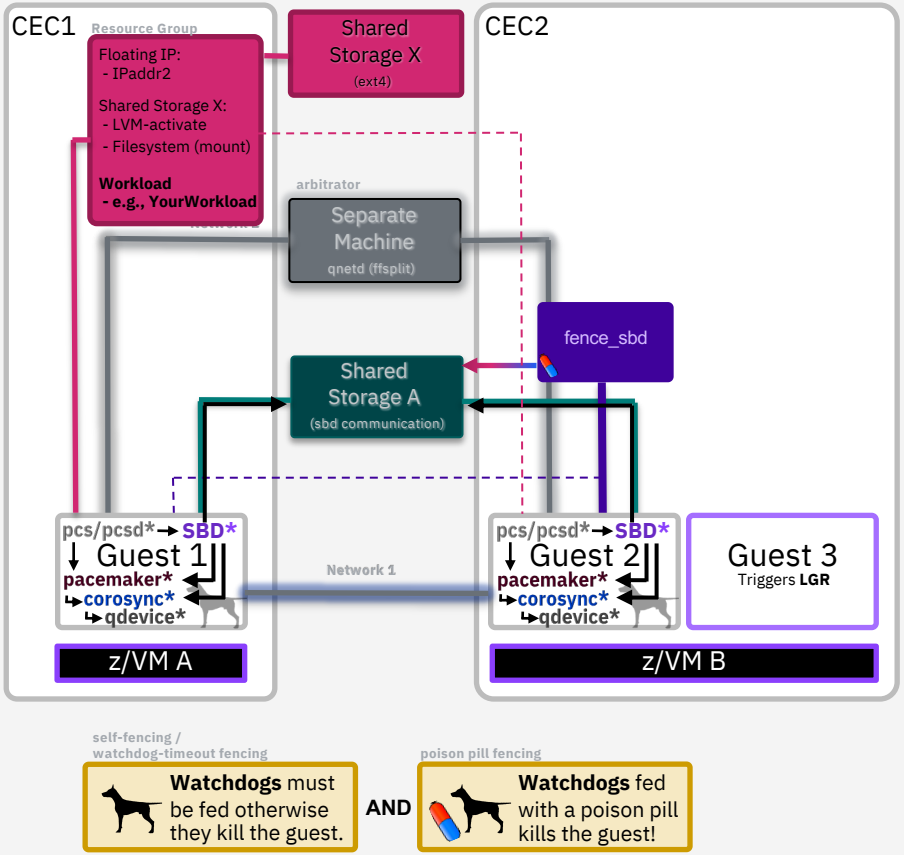
- Check if a Resource Agent (RA) exist for your workload. E.g. For the apache webserver a RA exists

```
rhel7/8# pcs resource list
```

Note:
- If no RA exists for your workload you might have to create a RA yourself. Checkout the previous slide about resources and Resource Agents.

- Add workload resource after filesystem mount

```
rhel7/8# pcs resource create YourWorkload \
            your-workload-ra \
            --group wload \
            --after my_fs
```



Cluster Testing

Cluster Testing

Step 6.0 – Test Workload Failover

❖ Watch the cluster status (live)

```
# watch -n1 pcs cluster status
```

Run on: Guest 2

❖ Trigger a kernel panic

```
# echo c > /proc/sysrq-trigger
```

Run on: Guest 1

❖ Or trigger a logoff via the z/VM hypervisor

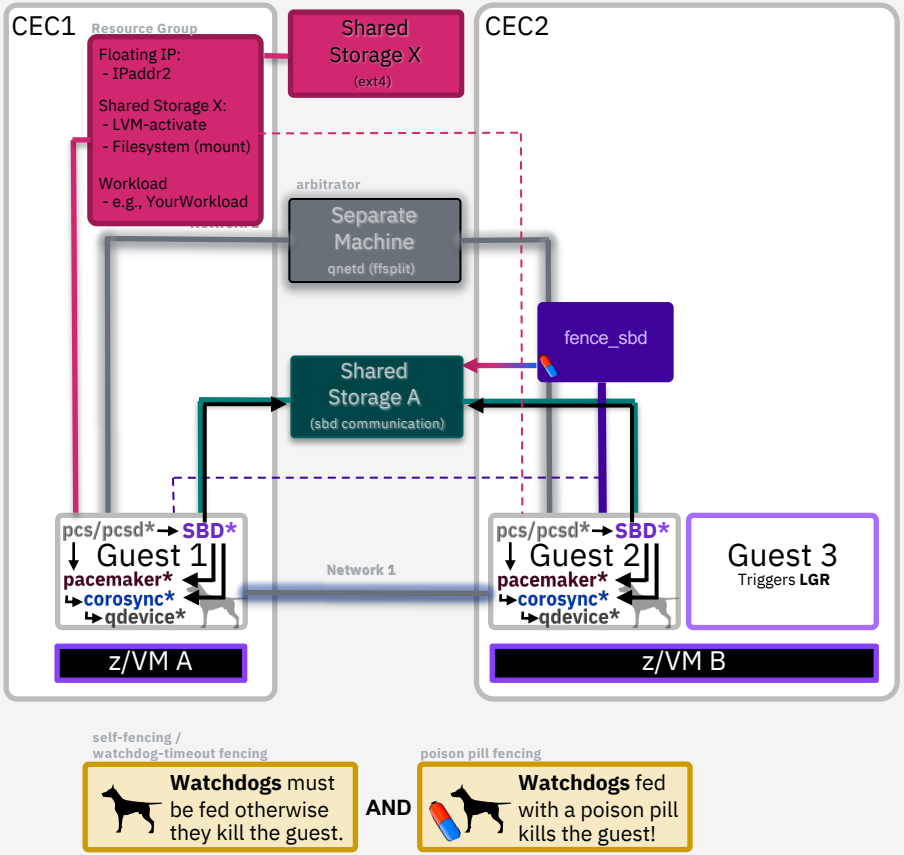
- ❖ Log into the z/VM guest user id and type

```
# cp logoff
```

Run on: z/VM A

❖ Check systemd logs (errors, warnings...)

```
# journalctl -u corosync
# journalctl -u pacemaker
# journalctl -u sbd
```



Appendix

Appendix A - Shared Storage – z/VM Shared Storage

Create Fullpack Minidisk in z/VM

This might be required to be executed multiple times when setting up shared storage in z/VM.

Run on: z/VM

❖ Create new user LINSHARE

```
USER LINSHARE NOLOG
MDISK 0200 3390 DEVNO 1111 MWV
```

❖ Link Minidisks

```
# vmcp 'link * 0200 0200 rw'
```

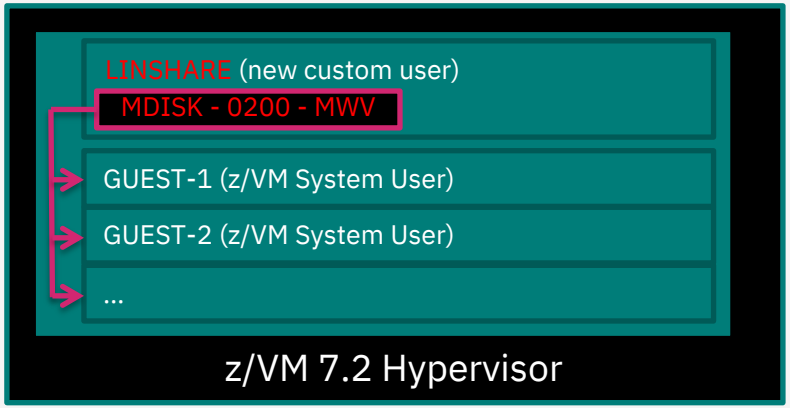
Run on: every guest in cluster

❖ De-ignore, enable and make DASD persistent:

```
# chzdev -e dasd 0.0.0200
# echo 0.0.0200 >> /etc/dasd.conf
```

❖ Format dasd and create partition over whole dasd

```
# dasdfmt -b 4096 -d cdl \
          -p /dev/disk/by-path/ccw-0.0.0200
# fdasd -a /dev/disk/by-path/ccw-0.0.0200
```



Appendix B – Detailed Architecture

Cluster Resources

(can be grouped, ordered, ..)

Cluster operations:
start, stop, monitor

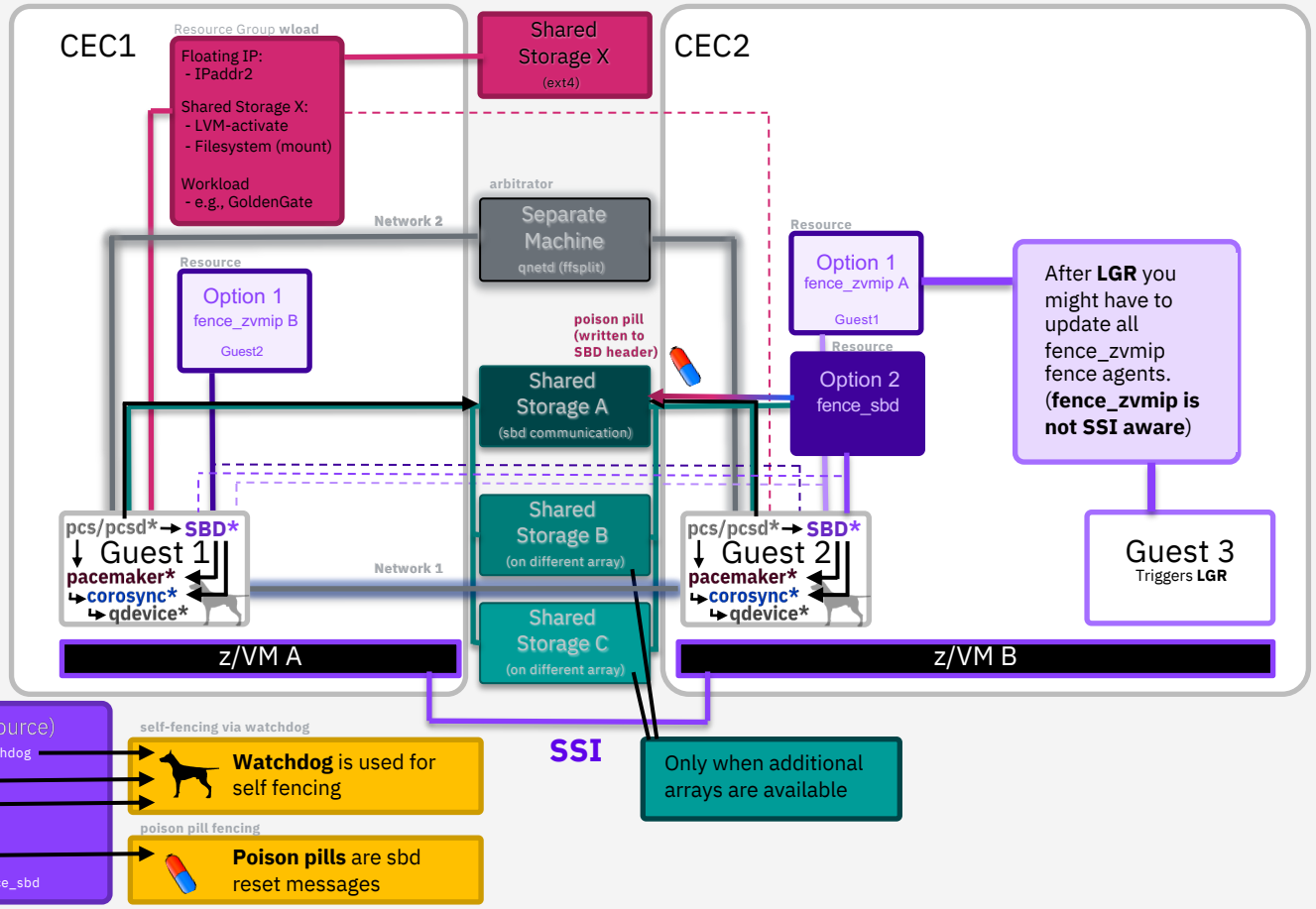
Cluster Quorum

Cluster operations:
start, stop, monitor
↳ needs quorum

avoids split-brain

Daemon

abbreviated with * in this graphic



sustain more node failures then standard quorum rules allow

qdevice* = systemd:qdevice

model: net — arbitrator only option. qdisk in future?

algorithm: fifty-fifty-split — ? decides One vote. Score decides.

Heuristics (optional) scripts (pings etc.) which all must pass; result send to qnetd (before Tiebreaker)

Tiebreaker in case all values are the same

sbd daemon (watch mode)

SBD* = systemd:sbd (Resource)
- inquisitor — inits /dev/watchdog
- corosync watcher
- pacemaker watcher
- disk watcher

Creates or uses sbd slot on /dev/by-path/ccw-0.0.0200-part1 and waits for instructions from fence_sbd

self-fencing via watchdog

Watchdog is used for self fencing

poison pill fencing

Poison pills are sbd reset messages

