High Availability Clustering with Red Hat® Enterprise Linux 8

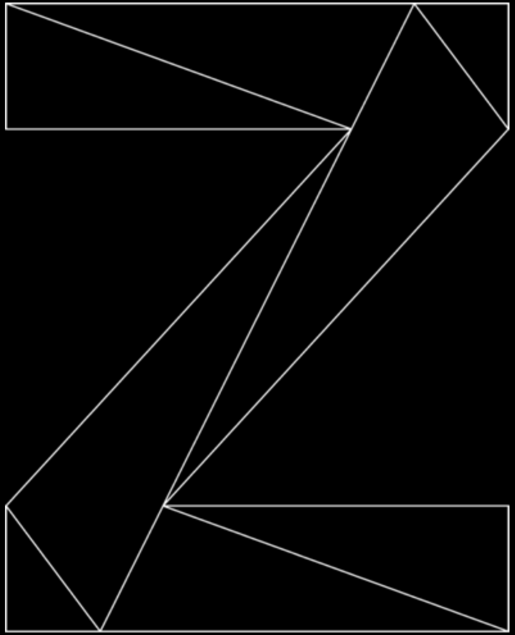
Solution Assurance

Bodo Brand
bodo.brand1@ibm.com

Daniel Kaiser
daniel.kaiser1@ibm.com

David Stark
david.stark@de.ibm.com

Document version: 2021-12



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.

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.

Agenda

- **Introduction**
  - **References**
  - **Cluster Types – Single site**
  - **Cluster Types – Multi site**
  - **IBM Z run levels**
  - **Components**
- **Concepts**
  - **Resources**
  - **Quorum**
  - **Planned/Unplanned Outage**
  - **Fencing/STONITH**
  - **Advanced Concepts**
- **Use Case:**
  - **LPAR HA Cluster with KVM as resource**
- **Appendix**

Introduction

Introduction - References

Documentation:

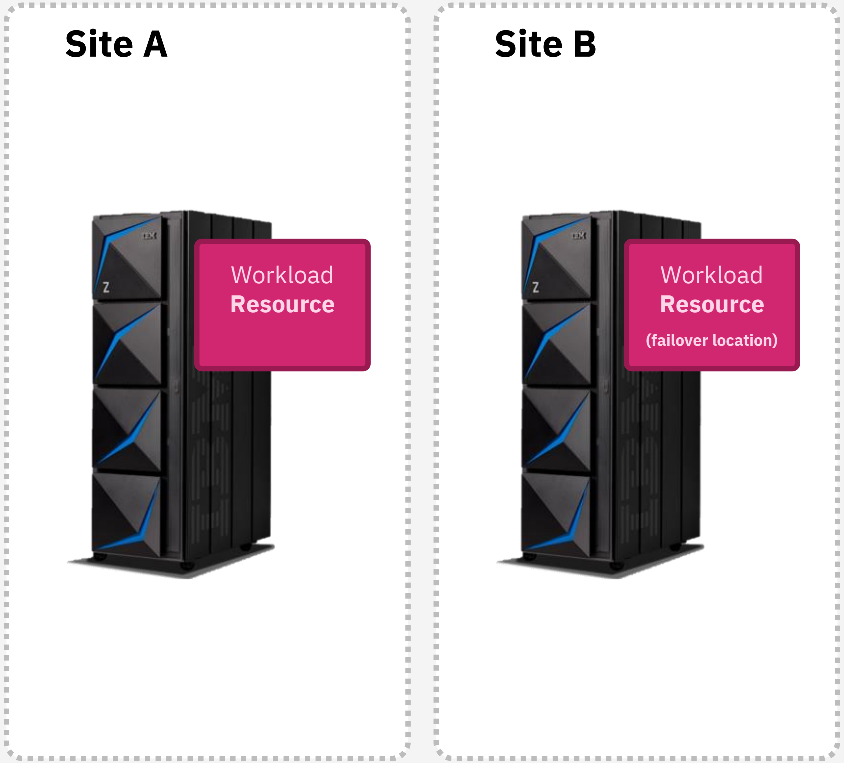
Official Red Hat® **documentation**:
- RHEL7: LINK
- RHEL8: LINK

Official Red Hat® **support statements**:
- z/VM specific
- further support statements

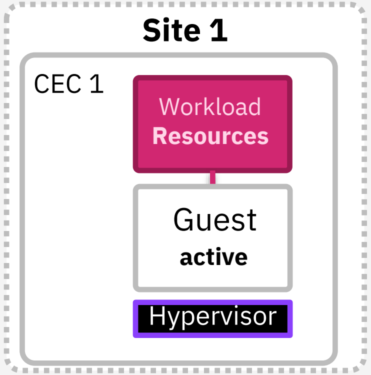
Official Red Hat® **version changes**:
- RHEL7: LINK
- RHEL8 Release Notes: LINK

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

Official Pacemaker documentation



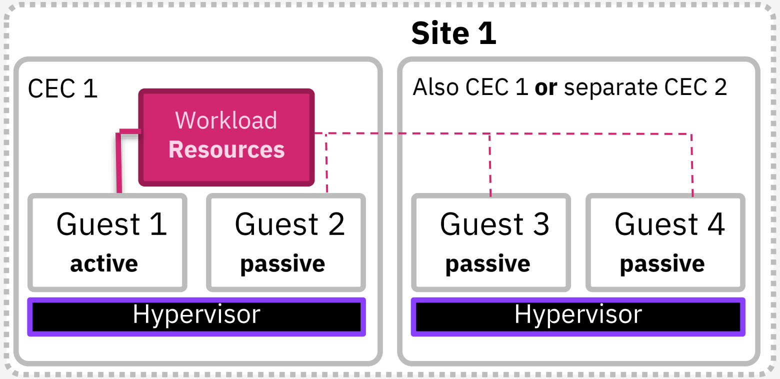
Introduction – Cluster Types – Single site



No Red Hat High Availability

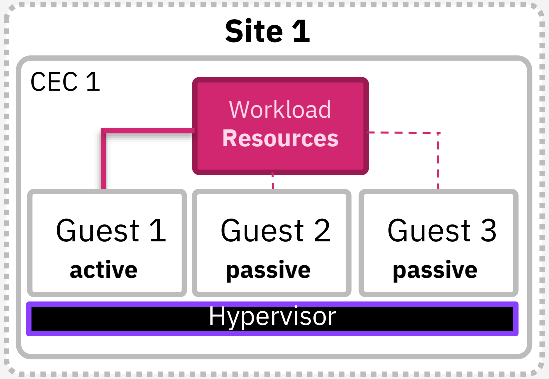
Hypervisor can be:
- LPAR
- z/VM
- KVM

Workload Resources can be anything e.g. an apache webserver



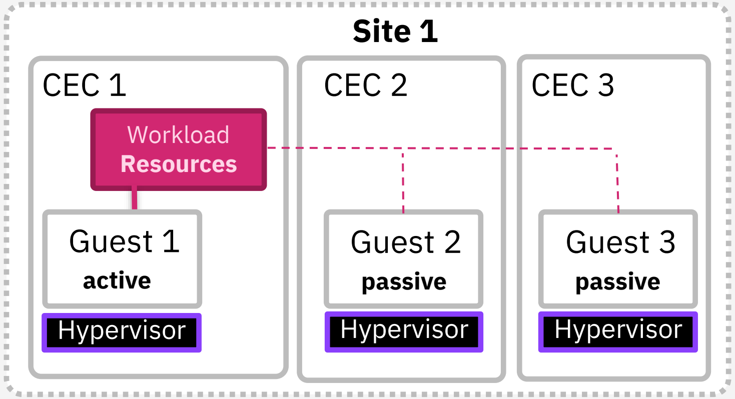
Cluster spanning two Hypervisors&CECs

Notes:
- A quorum server running separately might be needed



Active / Passive Failover

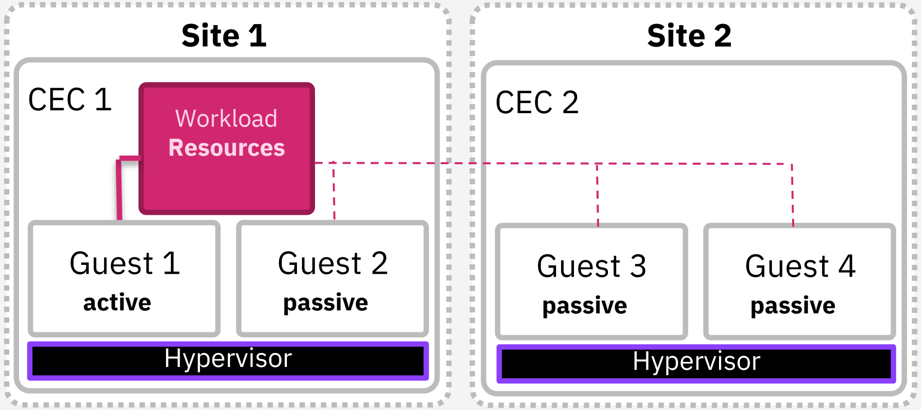
- Minimal setup to get started with HA.
- Does not sustain CEC or Hypervisor failures



Cluster spanning over more then two Hypervisors/CECs

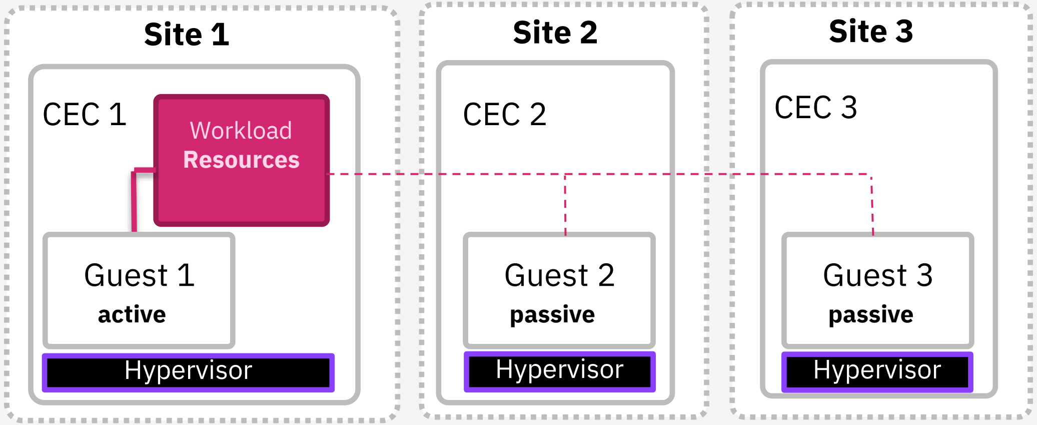
- Quorum server might not be needed!

Introduction – Cluster Types – Multi site



Multi Site Cluster with two sites

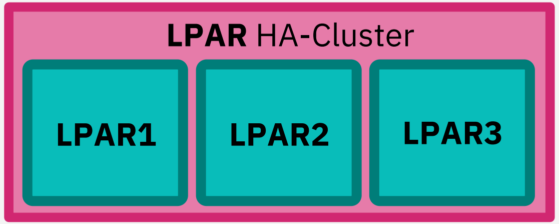
- You may have to use Replicated Storage instead of Shared Storage
- The HA settings have to keep the addtional latency in mind which might be introduced
- Reachability gets more complicated



Multi Site Cluster with more then two sites

- With that many CECs a separate quorum server might not be needed anymore
- With storage replication you might want to have synchron replication between two sites and asynchron replication with a third site

Introduction – IBM Z run levels



LPAR HA-Cluster

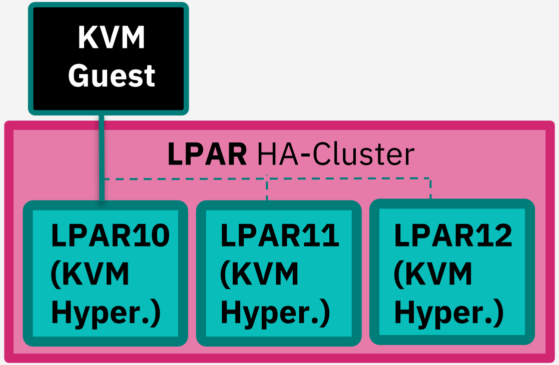
LPAR1

LPAR2

LPAR3

<u>Cluster spanning over three LPARs</u>

- PR/SM Hypervisor with optionally DPM



KVM Guest

LPAR HA-Cluster

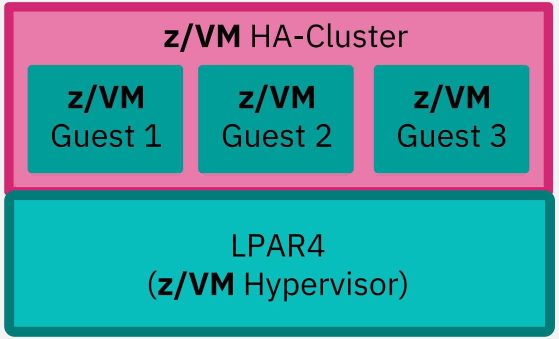
LPAR10 (KVM Hyper.)

LPAR11 (KVM Hyper.)

LPAR12 (KVM Hyper.)

<u>Cluster spanning over three LPARs with KVM Guests</u>

- **KVM Guest** is seen as any other workload in the cluster



z/VM HA-Cluster

z/VM Guest 1

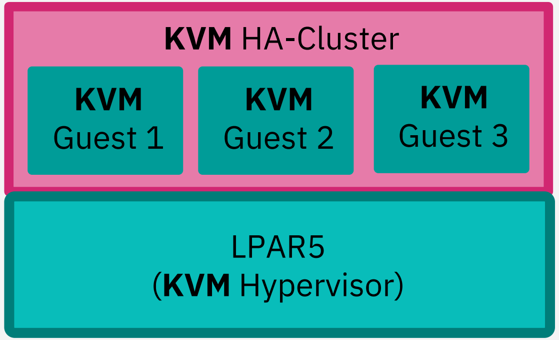
z/VM Guest 2

z/VM Guest 3

LPAR4
(**z/VM** Hypervisor)

<u>Cluster spanning over three z/VM Guests</u>

- **KVM Guest** is seen as any other workload in the cluster



KVM HA-Cluster

KVM Guest 1

KVM Guest 2

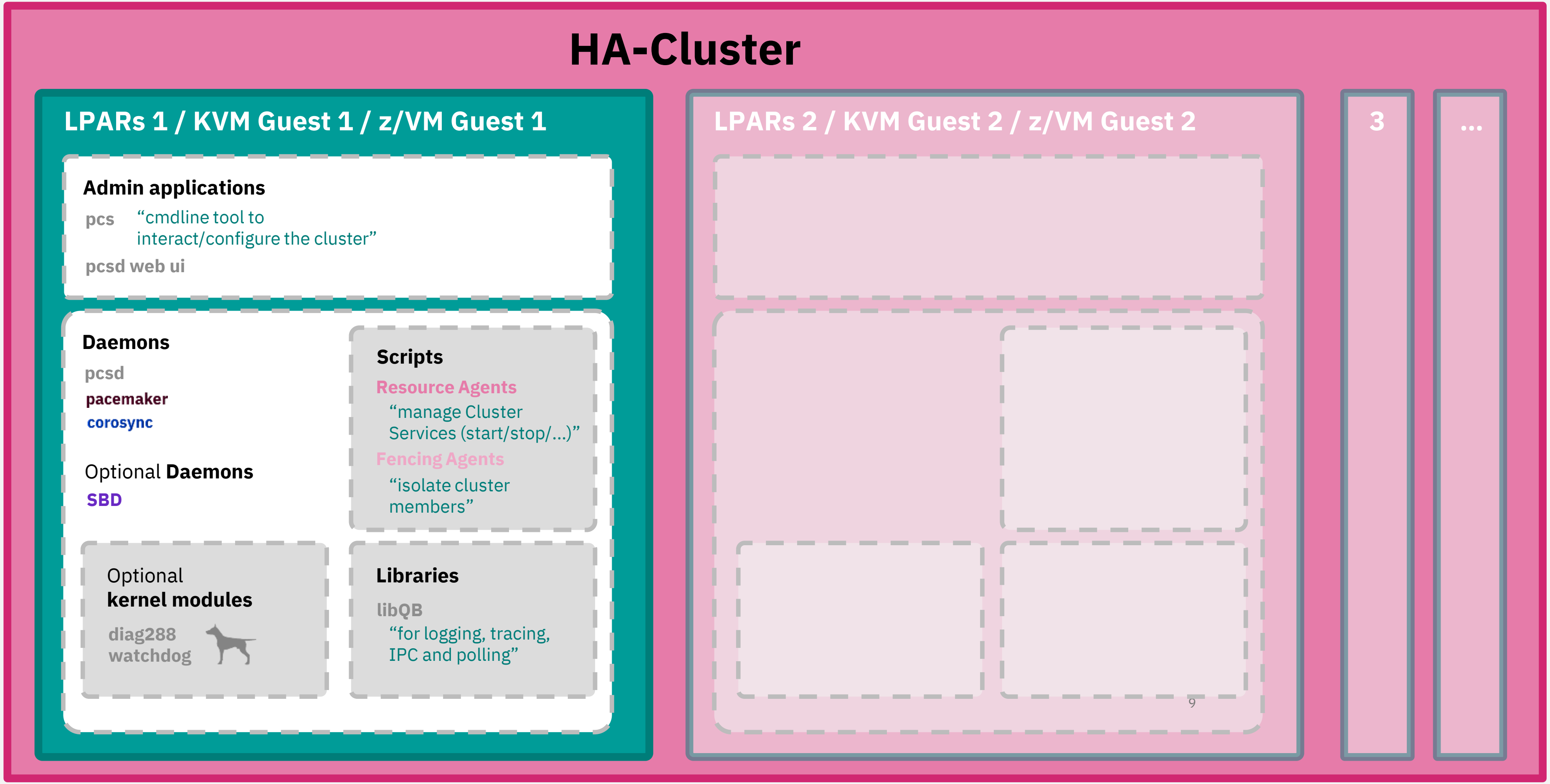
KVM Guest 3

LPAR5
(**KVM** Hypervisor)

<u>Cluster spanning over three KVM Guests</u>

- **KVM Guest** is seen as any other workload in the cluster

Introduction – Components (Minimal)



HA-Cluster

LPARs 1 / KVM Guest 1 / z/VM Guest 1

Admin applications

pcs “cmdline tool to interact/configure the cluster”

pcsd web ui

Daemons

pcsd

pacemaker

corosync

Optional **Daemons**

SBD

Scripts

Resource Agents
“manage Cluster Services (start/stop/…)”

Fencing Agents
“isolate cluster members”

Optional
kernel modules

diag288
watchdog

Libraries

libQB
“for logging, tracing, IPC and polling”

LPARs 2 / KVM Guest 2 / z/VM Guest 2

3

…

Concepts

(Managed) Resources

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

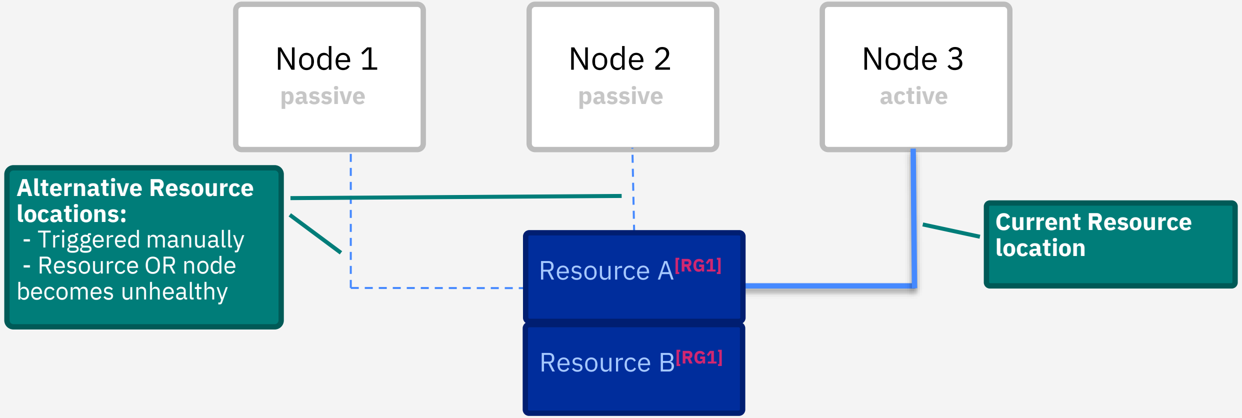
- type (e.g., apache) and resource identifier (e.g., Website)

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



Alternative Resource locations:
- Triggered manually
- Resource OR node becomes unhealthy

Current Resource location

Predefined Ressources:

- List all predefined Resources

```
# pcs resource list
```

- Look into the Resource docu

```
# pcs resource describe ...
```

- 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`

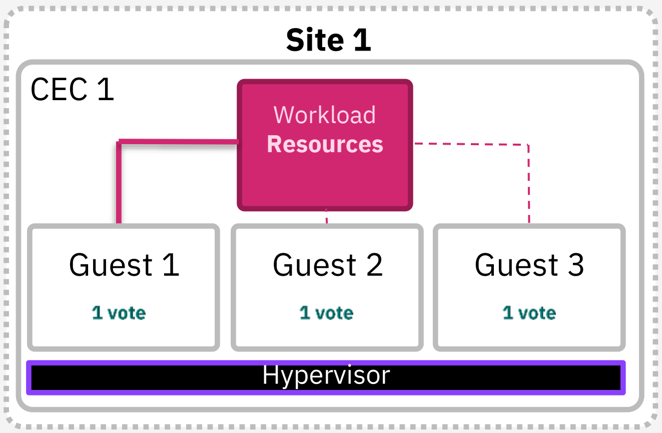
Quorum

Quorum

Corosync Votequorum

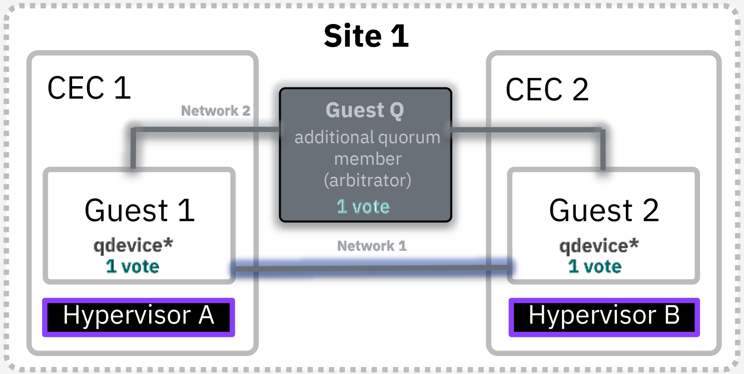
- ❖ Quorum decides how many guests can fail before the cluster becomes non-operational
- ❖ Quantity of **votes are assigned to the systems**
- ❖ Only when a **majority of votes are present** the cluster **operations are allowed** to proceed.
- ❖ With 1 CEC an uneven amount of nodes ensures quorum when 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)

Uneven number of cluster members



Notes:
- Usually 1 vote each
- cluster is quorate / functional with 2 votes.

Even number of cluster members



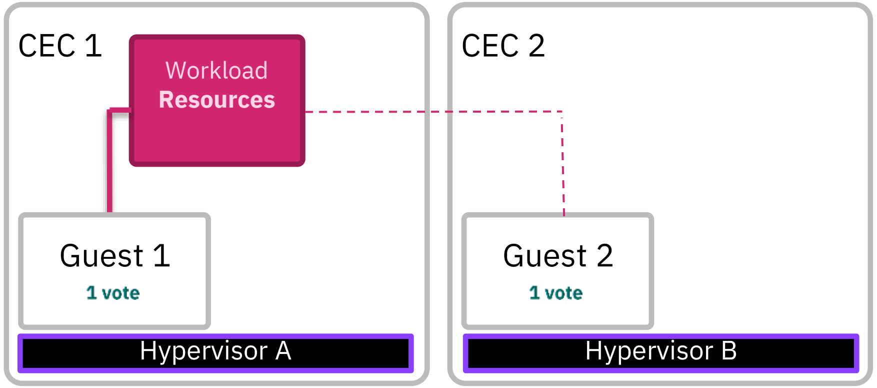
Notes:
- Guest Q should be reachable through different network connection
- can be used by multiple independet clusters

Quorum – 2/3-Nodes Challenges

2-node cluster

split brain challenge

Operative with 1 vote


CEC 1

Workload Resources

Guest 1

1 vote

Hypervisor A

CEC 2

Guest 2

1 vote

Hypervisor B

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**)

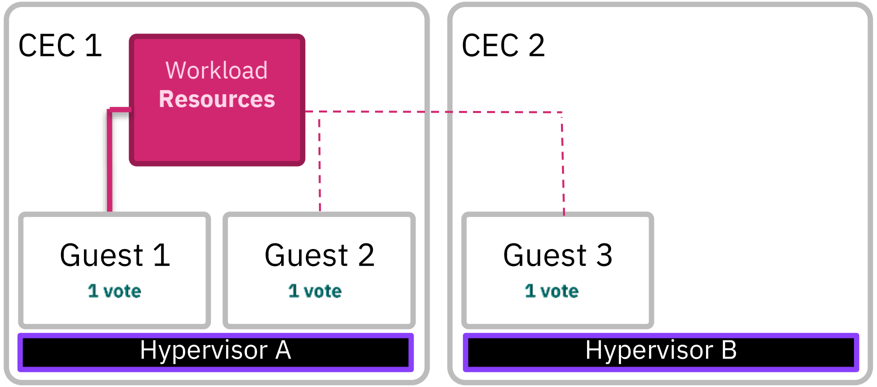
Note:

- When a CEC/Hypervisor dies, it affects the network and guest at the same time. A Tie-Breaker where the “lower node id” (e.g. guest 1) wins is not be able to handle this situation well.

3-node cluster

quorum challenge

Operative with 2 vote


CEC 1

Workload Resources

Guest 1

1 vote

Guest 2

1 vote

Hypervisor A

CEC 2

Guest 3

1 vote

Hypervisor B

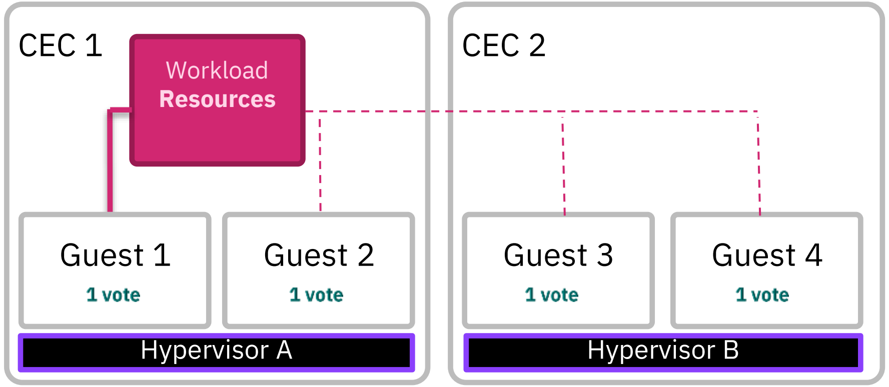
Does not work because in case of CEC 1 failure CEC 2 alone is not quorate!

Quorum – 4 Nodes Challenges and Solution

4-nodes cluster

split brain + quorum challenge

Operative with 3 vote

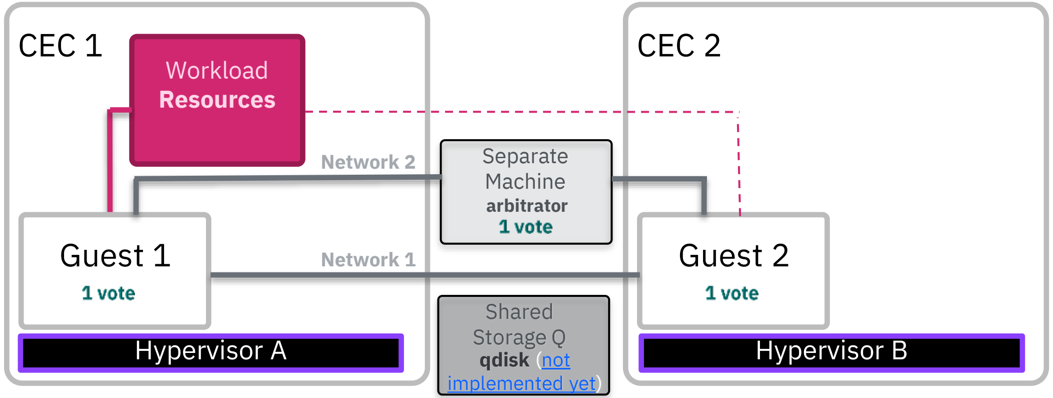


Does not work because in case of a network failure in between the CECs both CECs are not quorate.

Solution:

2 / 4 nodes + additional quorum member

Operative with 2 vote



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

Planned / Unplanned Outage

Planned / Unplanned Outage

Planned Outage

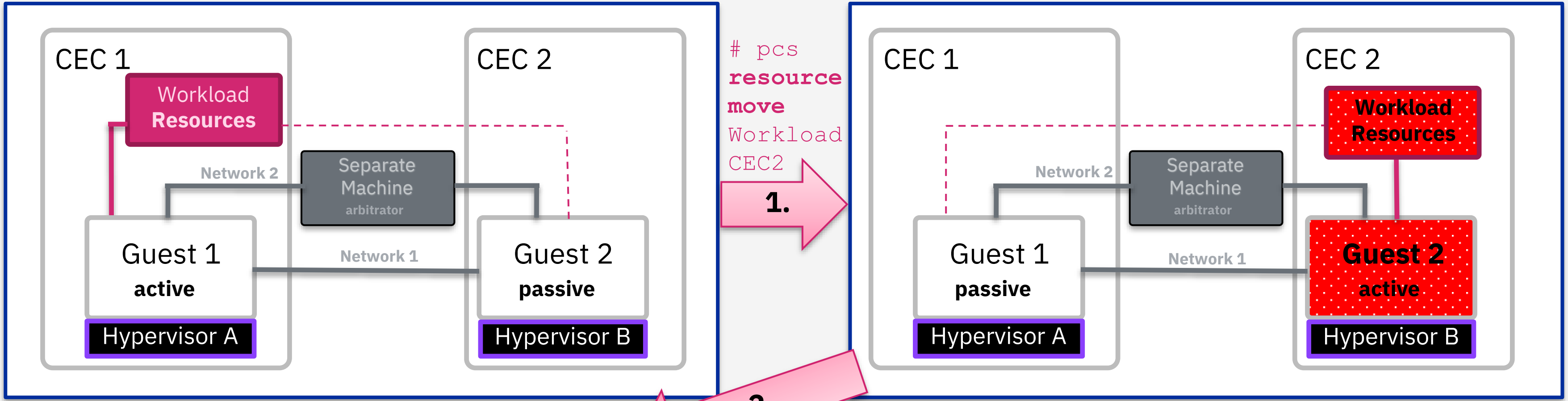
- Red Hat HA allows you to manually trigger the movement of workload to another cluster member
- Live Guest Relocation (LGR) (**z/VM**) / Live Migration (**KVM**)
  - **z/VM Guests Cluster**: For Live Guest Relocation to work you need a **SSI cluster**.
    - Only LGR of passive guests might be supported (LINK)
  - **KVM Guests Cluster**: For Live Migration to work you usually need Shared Storage (or Replicated Storage) which is mounted read and write between the Hypervisors.
  - **LPARs Cluster with KVM Guest Workload:** Live Migration of the Resource is automatically tried when all requirements are met.

Unplanned Outage

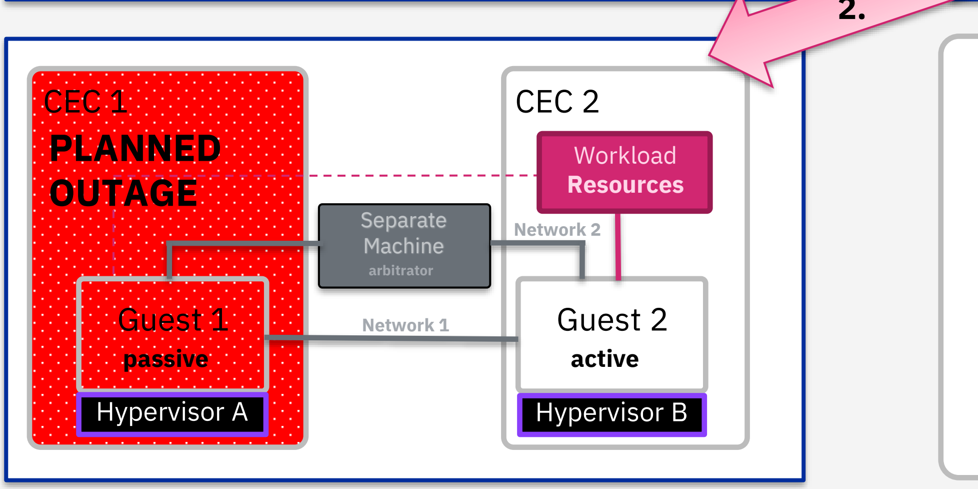
- Red Hat HA automatically fails over in case of failure as soon as the node released all Resources (see Fencing/STONITH concept).
- Live Guest Relocation (**z/VM**) / Live Migration (**KVM**)
  - Cannot be used as the Guest as you would move a corrupted/broken guest in this case.

➢ See following slides for graphic illustrations.

Planned Outage – 2 CECs Example



```
# pcs
resource
move
Workload
CEC2
```



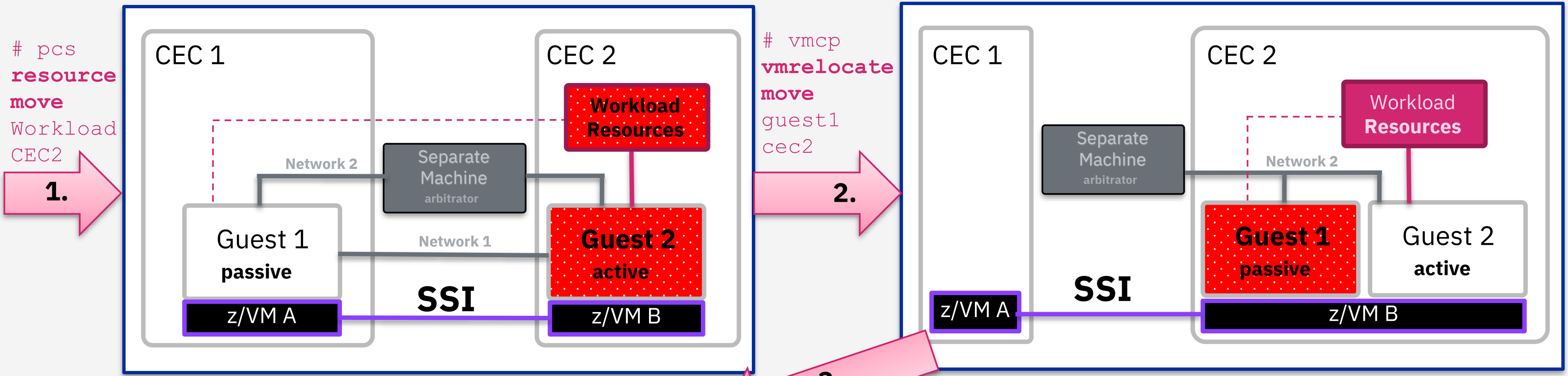
Failover

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

Bringup after Outage

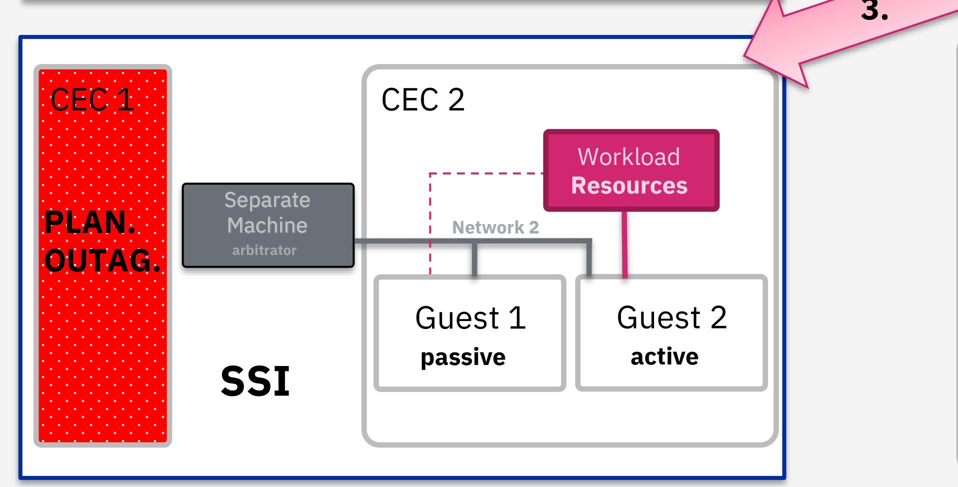
❖ guest 1 automatically joins back

Planned Outage – 2 CECs Example with SSI (z/VM)



```
# pcs resource move Workload CEC2
```

```
# vmcp vmrelocate move guest1 cec2
```



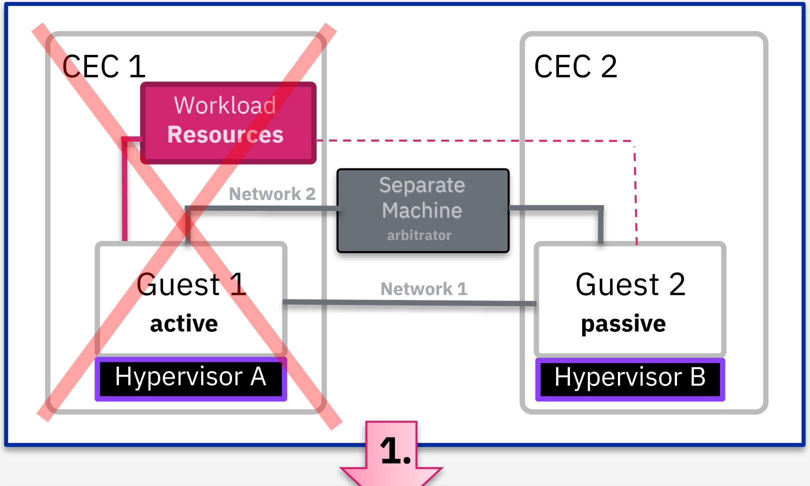
Failover

- ❖ 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 CEC 1

UN-Planned Outage – 2 CECs Example

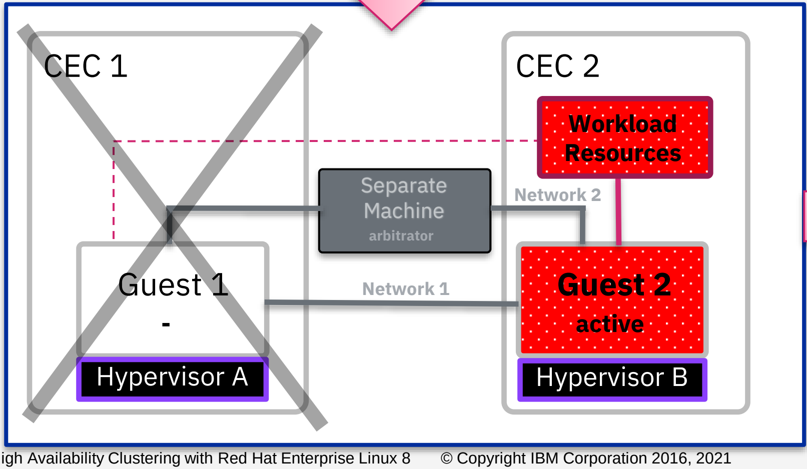


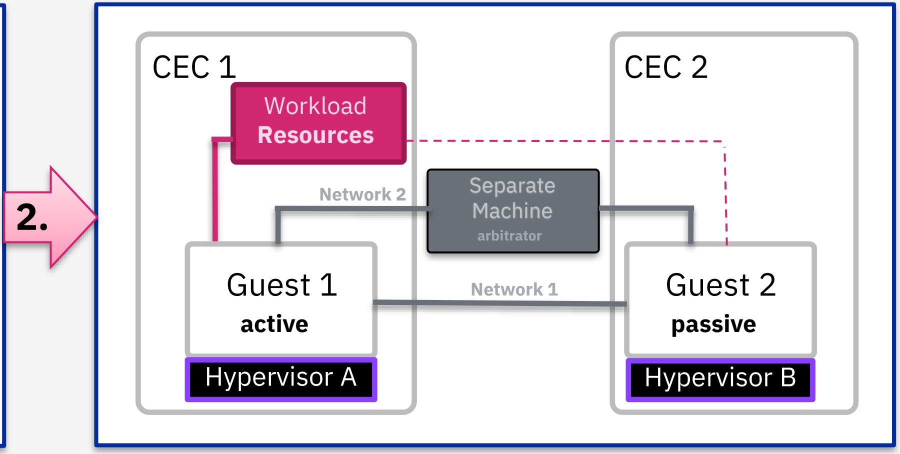
Failover

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

Bringup after Outage:

- ❖ Guest 1 automatically joins back
- ❖ Workload resources move back to Guest 1 by default (configurable)





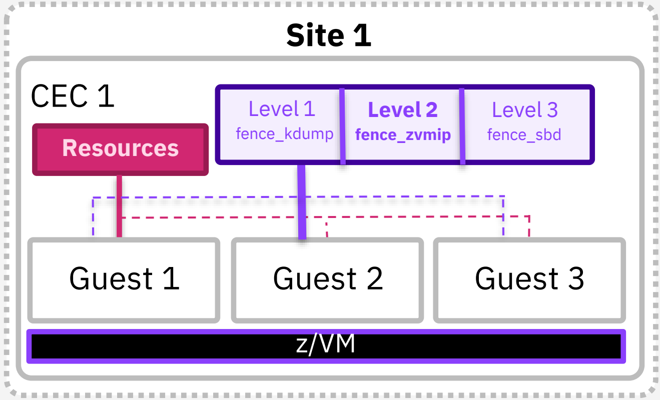
Fencing / STONITH

Fencing / STONITH

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
- ❖ Depending on your HA Setup you might want to use a combination of the following available fencing agents:
  - ❖ **fence_zvmip (via z/VM SMAPI)**: 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)
  - ❖ **fence_sbd (via SBD)**: 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.
  - ❖ **fence_ibmz (via HMC API):** Performs a deactivate, activate and load operation on a LPAR. Both Classic (PR/SM) and DPM is included. (available with RHEL8.6+ and RHEL9+)
  - ❖ **fence_kdump :** This fence agent just detects if the failing guest is currently taking a kdump. If yes fencing is considered complete.
  - ❖ **fence_virsh (via KVM - virsh):** Simply ssh's to the Hypervisor and fences the guest through virsh commands.

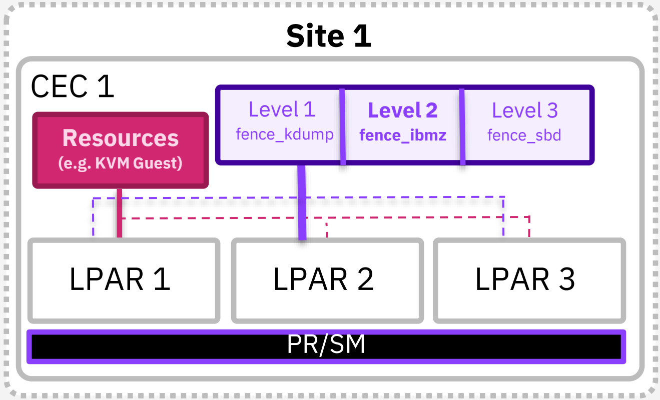
z/VM Guest Cluster – fencing example



Specifics:

- ❖ Level 3 (fence_sbd): You can specify the z/VM command executed on SBD fencing

LPARs Cluster – fencing example



Specifics:

- ❖ Level 2 (fence_ibmz): Check support for Redhat version. Should be able to handle both:
  - ❖ classic (PR/SM)
  - ❖ DPM

Advanced Concepts

Advanced concepts for reference

Remark

- ❖ Not covered technically in this presentation

Cluster notifications & Error conditions

- ❖ When errors happen in the cluster the cluster might not proceed without manual intervention
- ❖ Getting notified of cluster problems in time can be crucial for High Availability

Cluster User Permissions

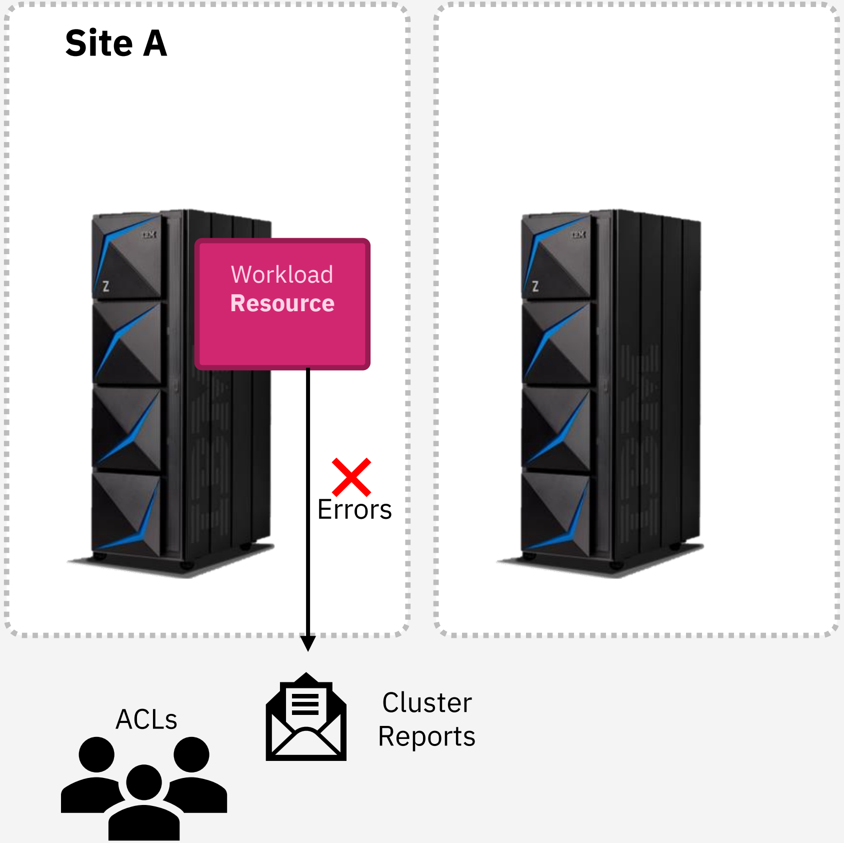
- ❖ To make sure a specific role (e.g. a cluster operator) can only perform actions specific to his job role you can configure ACLs (Access Control Lists)

Deal with Multi Site Clusters

- ❖ To prevent "split-brain" in multi-site clusters the booth ticket manager spans an overlay cluster over existing clusters on different sites

Disaster Recovery

- ❖ A secondary cluster can be specified as recovery site



LPAR HA Cluster with KVM as resource

Agenda

- ❖ **Architecture and Requirements**

- ❖ **Steps for setup creation**
  1. **First Steps – Cluster Setup**
  2. **Quorum**
  3. **Fencing/STONITH**
     4. fence_ibmz
     5. fence_sbd
     6. fence_kdump
     7. Fencing levels
  8. **GFS2 (Shared Storage)**
  9. **VirtualDomain (KVM Guest)**
  10. **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.

Run on:

LPAR 1

LPAR 2

...

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

Run on:

LPAR 1

LPAR 2

...

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

Architecture and Requirements

Architecture and Requirements

Our Test Setup:

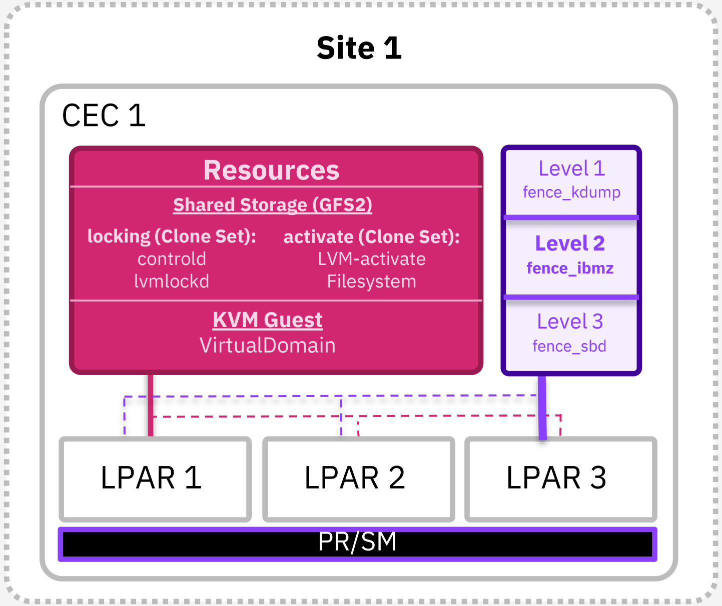
- ❖ z15™ and DS8000®
- ❖ 3 LPARs (on one CEC)
  - ❖ 3 ECKD DASD (LPAR guest OS)
- ❖ Shared Storage (for SBD and KVM)
  - ❖ 2+ ECKD DASD

LPARs:

- ❖ PR/SM mode **or** DPM mode
- ❖ Distro: RHEL 8.5

Legend:

¦¦¦ **Failover Paths**



❖ **Clone Set**:
Is a Resource Group which is cloned to all other cluster members.

First Steps – Cluster Setup

Installation and Firewall

Step 1.0 – Installation

Run on:

LPAR 1
LPAR 2
LPAR 3

```
rhel8# subscription-manager register --auto-attach

rhel8# dnf config-manager --set-enabled \
           rhel-8-for-s390x-highavailability-rpms

rhel8# yum update -y
rhel8# yum install -y pcs


rhel8# yum install -y pacemaker \
                      fence-agents-all
```

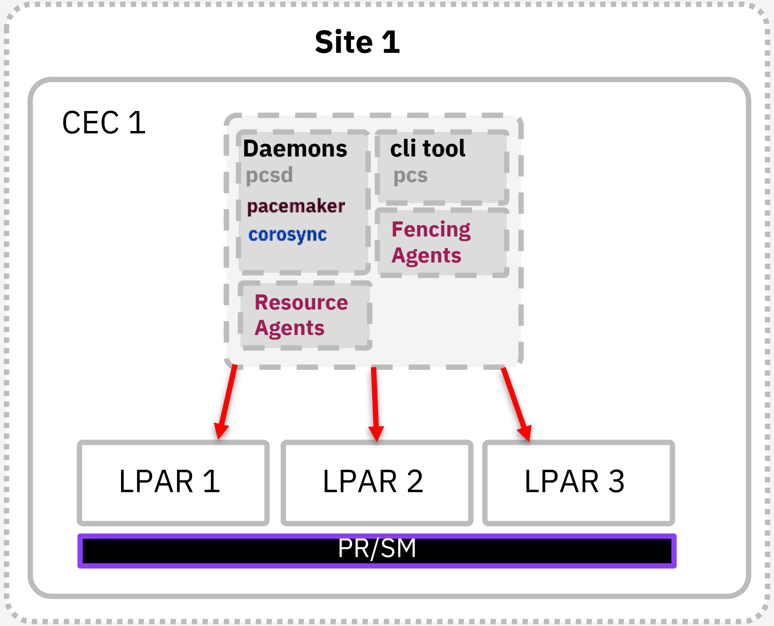
Note:
- fence_ibmz will only be installed with later Red Hat Enterprise Linux versions. Manual installation from upstream will be described later in this Use Case.

Step 1.1 – Firewall Configuration

Run on:

LPAR 1
LPAR 2
LPAR 3

```
rhel8# firewall-cmd --permanent \
           --add-service=high-availability
rhel8# firewall-cmd --reload
```



Cluster Setup

Step 1.2 – Prepare Cluster

Run on:

LPAR 1
LPAR 2
LPAR 3

❖ Create a linux® user used by the cluster

```
rhel8# passwd hacluster
```

Note:
- Same password for each node is recommended.

❖ Enable the cluster controlling and configuration daemon

```
rhel8# systemctl enable pcsd.service --now
```

Step 1.3 – Auth Nodes

Run on:

LPAR 1

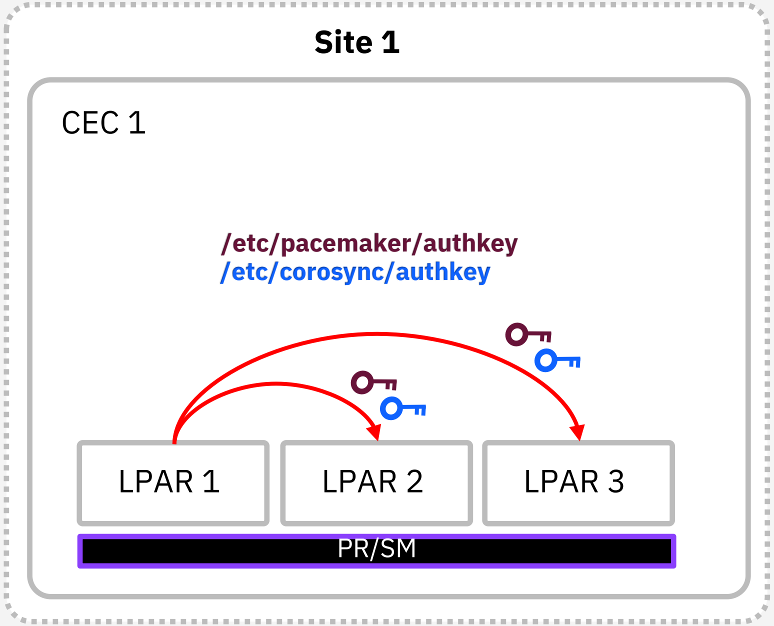
```
rhel8# pcs host auth LPAR1 LPAR2 LPAR3
Username: hacluster
Password: …
```

Step 1.4 – Setup Cluster

Run on:

LPAR 1

```
rhel8# pcs cluster setup \
          my_cluster LPAR1 LPAR2 LPAR3
```



Startup of the Cluster and Status

Step 1.5 – Start and Enable Services

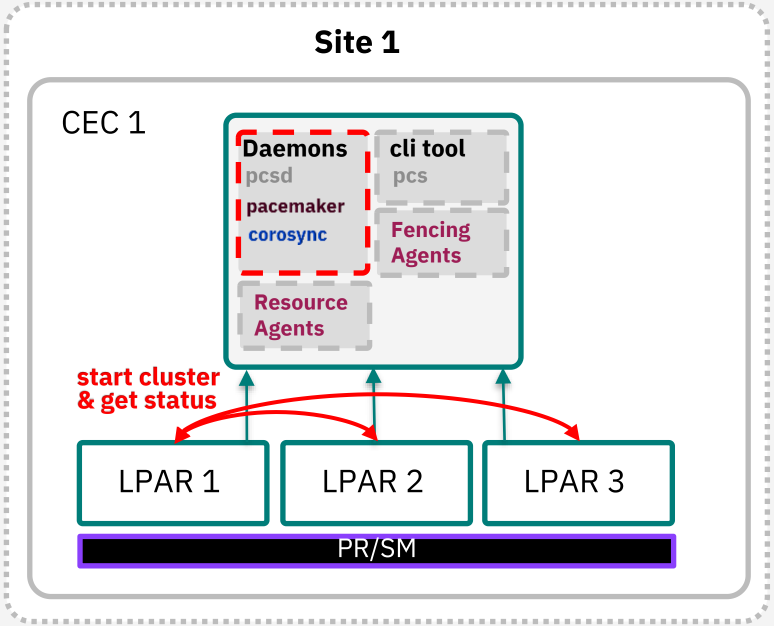
Run on: LPAR 1

```
rhel8# pcs cluster start --all
rhel8# pcs cluster enable --all
```

Step 1.6 – Status and CLI Tools

Run on: LPAR 1

- ❖ The pcs CLI tool allows you to configure the cluster (pacemaker + corosync) and view the status
  - ❖ Full cluster status:
    ```
    rhel8# pcs status --full
    ```
  - ❖ Pacemaker configuration
    ```
    rhel8# pcs cluster cib
    ```
- ❖ Pacemaker and corosync have their own cli tools:
  - ❖ Pacemaker configuration
    ```
    rhel8# cibadmin -Q
    ```
  - ❖ Show corosync object database
    ```
    rhel8# corosync-cmapctl
    ```
  - ❖ Dump live corosync flight data
    ```
    rhel8# corosync-blackbox
    ```



Quorum

Quorum

Step 2.0 – Considerations

Run on:

LPAR 1

❖ With wait_for_all enabled the whole cluster only becomes quorate/functional for the first time when all cluster members are available

```
rhel8# pcs quorum update wait_for_all=1
```

Note:
- For example, when starting three LPARs consecutively. Without enabling wait_for_all the last LPAR might be fenced from the two already available LPARs.

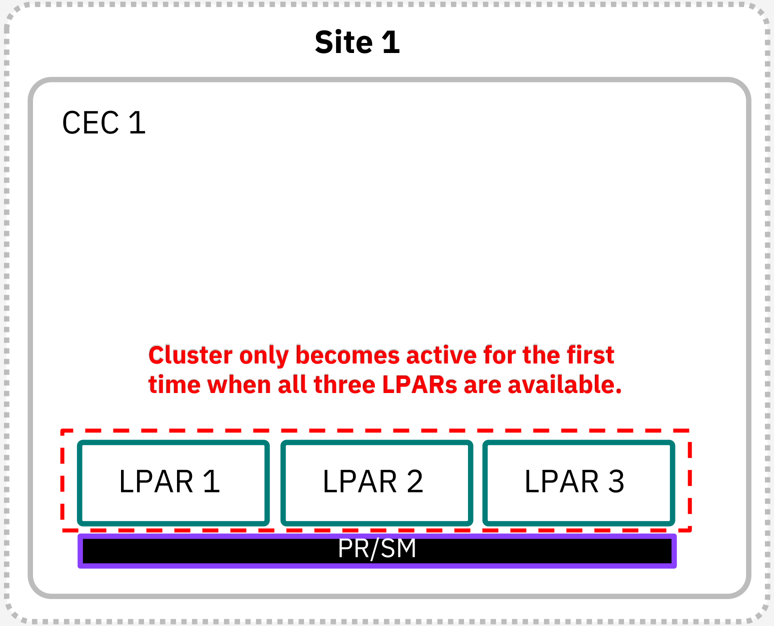
❖ "totem token timeout" specifies in milliseconds until a token loss is declared

```
rhel8# pcs cluster config update \
                 totem token=5000
```

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

❖ Check totem token timeout

```
rhel8# corosync-cmapctl | \
       grep "runtime.*totem.token "
```



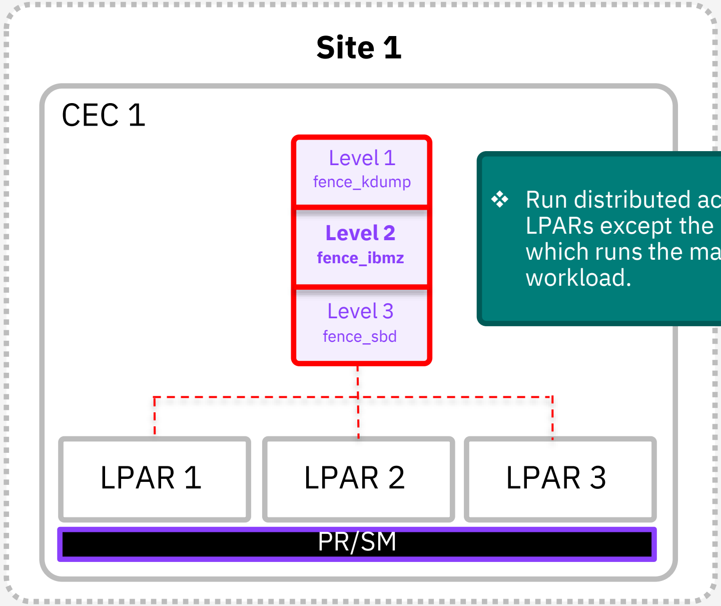
Cluster only becomes active for the first time when all three LPARs are available.

Fencing / STONITH

Fencing / STONITH

Step 3.0 – Considerations

- The main fencing method will be power fencing over the HMC:
  - **fence_ibmz (Level 2):** Provides solid fencing because it is a power fencing method which triggers fencing externally via the HMC API.
- When the HMC is not available, SBD is used as backup fence agent:
  - **fence_sbd (Level 3)**: As last resort, self fencing is a reliable backup option which might take a bit longer but should take effect in the worst cases. The poison pill is used to speed up this fence method in some failure cases.
- For debugging purposes, we also include:
  - **fence_kdump (Level 1):** When you take a kdump (either automatically or manually) you want to prevent other fencing methods to trigger. This way the fencing is considered successful when a LPAR kdumps.



- Run distributed across all LPARs except the LPAR which runs the main workload.

Fencing / STONITH fence_ibmz (Level 2)

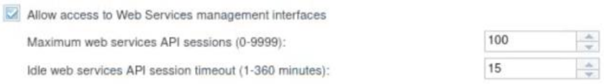
fence_ibmz – Prerequisites

Step 4.0 – Create a HMC user

Run on:

HMC - Classic mode

- **Create minimal viable role (Set Scope and Permissions of HMC user)**
  - Minimum tasks required:
    Deactivate, Activate, Load, View Activation Profiles
  - Scope to cluster members only.
  - Allow access to Web Services management interfaces

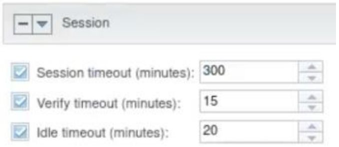


☑ Allow access to Web Services management interfaces

Maximum web services API sessions (0-9999): 100

Idle web services API session timeout (1-360 minutes): 15

  - Keep timeouts configured here in mind. The default values are usually very high which should not affect fencing actions.

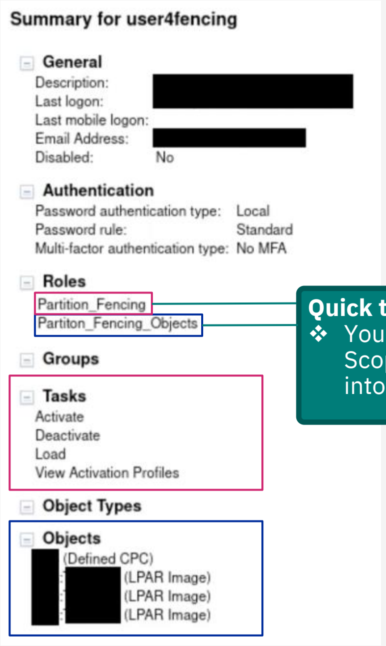


Session

☑ Session timeout (minutes): 300

☑ Verify timeout (minutes): 15

☑ Idle timeout (minutes): 20



Summary for user4fencing

General

Description: [redacted]

Last logon:

Last mobile logon:

Email Address: [redacted]

Disabled: No

Authentication

Password authentication type: Local

Password rule: Standard

Multi-factor authentication type: No MFA

Roles

Partition_Fencing

Partiton_Fencing_Objects

Groups

Tasks

Activate

Deactivate

Load

View Activation Profiles

Object Types

Objects

[redacted] (Defined CPC)

[redacted] (LPAR Image)

(LPAR Image)

(LPAR Image)

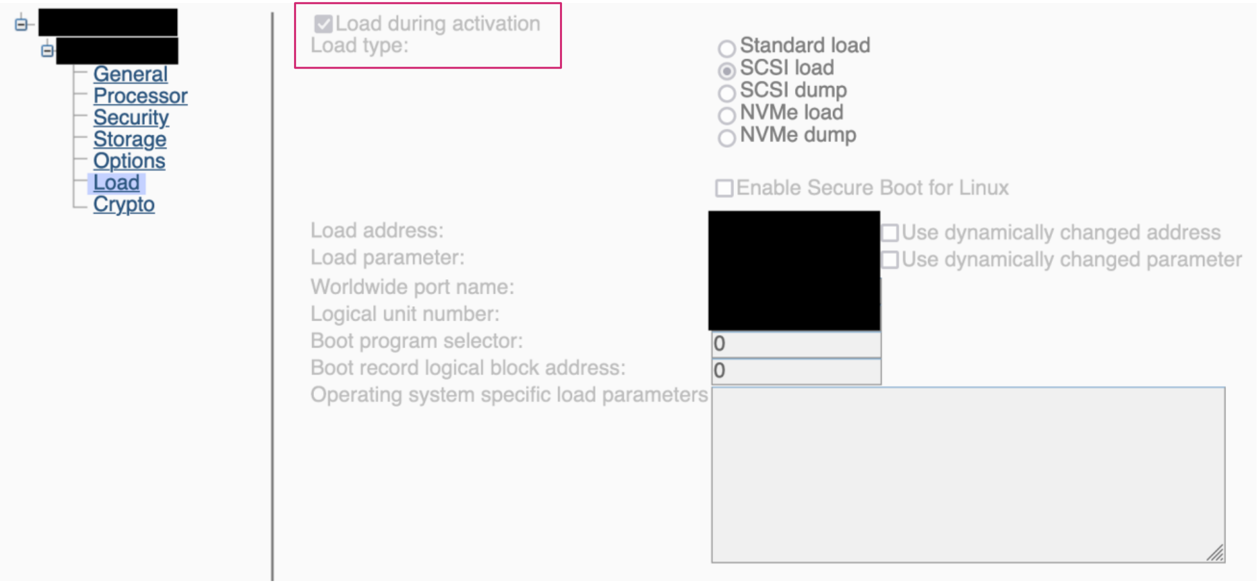
Quick tip:

- You can separate the Object Scope and Task permissions into two roles.

fence_ibmz – optional HMC SCSI configuration

Step 4.1 – Use HMC activate-on-load

- Set **load during activation** allows you to:
  - skip the additional load task (saves time)
  - can be found in the activation profile of each LPAR
  - works with any supported storage type
  - currently this option is **required** for SCSI usage



☑ Load during activation

Load type:

○ Standard load
● SCSI load
○ SCSI dump
○ NVMe load
○ NVMe dump

□ Enable Secure Boot for Linux

Load address: [redacted] □ Use dynamically changed address

Load parameter: □ Use dynamically changed parameter

Worldwide port name:

Logical unit number:

Boot program selector: 0

Boot record logical block address: 0

Operating system specific load parameters

fence_ibmz – TLS CA Certificates

Step 4.2 – Install and Trust HMC TLS Certificate

- Get Root Certificate and all Intermediate CA Certificates used in the chain of the server certificate (in PEM format) and then trust them by executing:

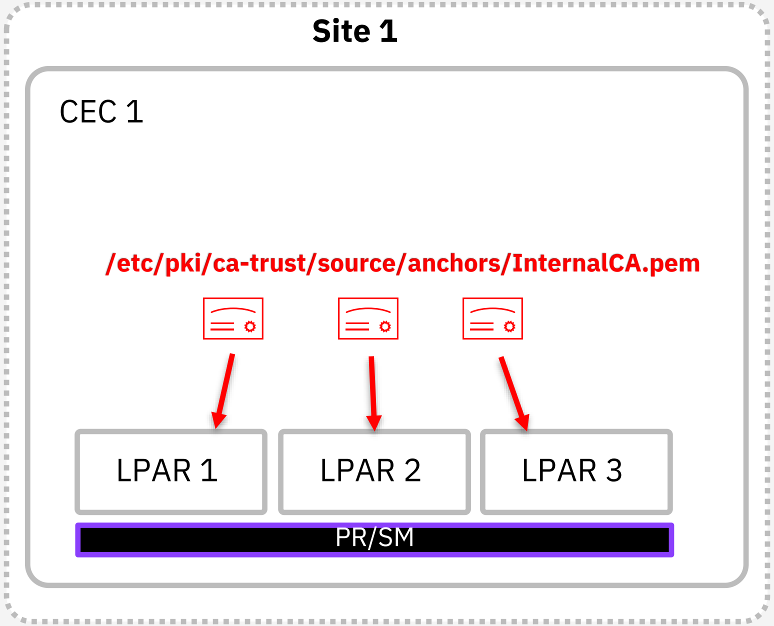
```
# cp CA_CERT.pem /etc/pki/ca-trust/source/anchors/
# update-ca-trust
```

- Verify that the Certificates are in the trust store:

```
# trust list | less
```

- Verify that the whole certificate chain to the HMC is trusted:

```
# openssl s_client -showcerts -connect ${HMC_URL}:443 \
                   -verify_return_error < /dev/null
```



fence_ibmz – Installation

Remark

- ❖ **In newer RHEL** versions fence_ibmz might be already installed with the fence-agents-all package

Step 4.3 – Setup fence_ibmz

Run on:

LPAR 1

LPAR 2

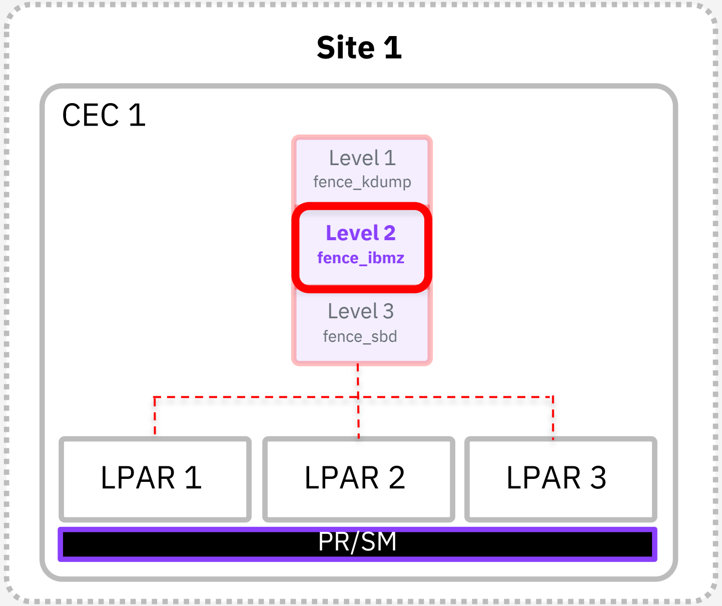
LPAR 3

- ❖ Installation via Upstream

```
# dnf install -y wget
# wget https://raw.githubusercontent.com/ClusterLabs/ \
fence-agents/master/agents/ibmz/fence_ibmz.py
# sed -i 's+@PYTHON@+/usr/libexec/platform-python+' \
fence_ibmz.py
# sed -i 's+@FENCEAGENTSLIBDIR@+/usr/share/fence+' \
fence_ibmz.py
# cp fence_ibmz.py /usr/sbin/fence_ibmz
# chmod +x /usr/sbin/fence_ibmz
```

- ❖ Verify Installation

```
# pcs stonith list|grep fence_ibmz
```



fence_ibmz – Add to Cluster

Step 4.4 – Add fence_ibmz to cluster

Run on:

LPAR 1

❖ Add fence agent to cluster

```
# pcs stonith create fence_ibmz fence_ibmz \
            ip=${HMC_URL} \
            username="${HMC_USER}" \
            password="${HMC_USER_PASSWORD}" \
            ssl_secure=true \
            pcmk_host_map="lpar1:CEC1/LPAR1;lpar2:CEC1/LPAR2;lpar3:CEC1/LPAR3"
```

Note:
- pcmk_host_map: the second value is case sensitive!
- It is possible to hide the password by providing a password_script: LINK.

❖ Add debug log output for verification

```
# pcs stonith update fence_ibmz verbose=1 debug_file=/tmp/fence_ibmz.log
```

❖ Trigger fencing manually to verify the fence agent

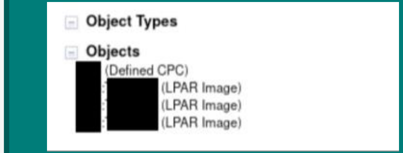
```
# pcs stonith fence lpar2
# cat /tmp/fence_ibmz.log | less
```

Note:
- stonith-timeout and stonith-action options might be ignored when triggering manual fencing: LINK.

Quick tip:

❖ Each pair consists of: HOSTNAME:CECNAME/LPARNAME

❖ See HMC Objects:



⊟ Object Types
⊟ Objects
[redacted] (Defined CPC)
[redacted] (LPAR Image)
(LPAR Image)
(LPAR Image)

fence_ibmz – Considerations

Step 4.5 – Further Considerations

Run on:

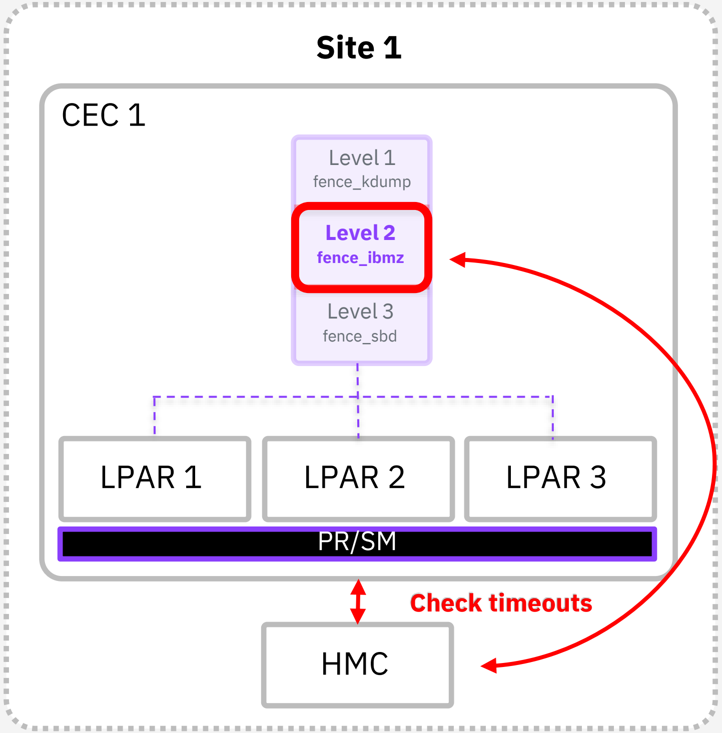
LPAR 1

- ❖ The stonith-timeout defines how long to wait for STONITH action (e.g. on, off) to complete. (default 60s)
  - ❖ Can be overwritten by pcmk_xxx_timeout on a fence agent basis

- ❖ Considering the deactivate operation on the HMC can take up to 900 seconds (by default), you can overwrite the STONITH action timeout for the fence agent:

```
# pcs stonith update fence_ibmz \
                     pcmk_reboot_timeout=1810 \
                     pcmk_off_timeout=905 \
                     pcmk_on_timeout=905
```

Note:
- pcmk_reboot_timeout should not be relevant here as the fence operation maps the reboot action to off and on internally
- When the system is at it limits the HMC task can take significantly longer



Fencing / STONITH fence_sbd (Level 3)

SBD Fencing – Watchdog

Step 5.0 – Setup Watchdog

Run on:

LPAR 1
LPAR 2
LPAR 3

- ❖ Enable watchdog kernel module

```
# modprobe diag288_wdt
```

- ❖ Show watchdog

```
# wdctl
```

- ❖ Make watchdog loading persistent

```
# echo "diag288_wdt" > /etc/modules-load.d/watchdog.conf
```

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

Step 5.1 – Setup SBD Shared Storage

Run on:

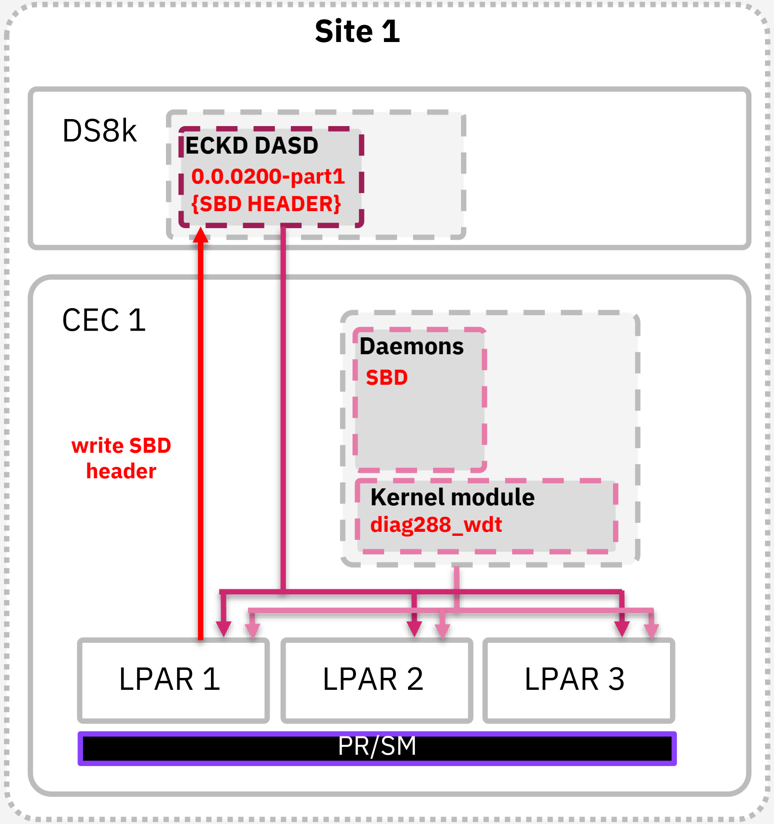
LPAR 1

- ❖ Shared Storage is already assumed to be setup.
- ❖ Create SBD header on disk partition

```
rhel8# pcs stonith sbd device setup \
          --device=/dev/disk/by-path/ccw-0.0.0200-part1 \
          watchdog-timeout=15 \
          msgwait-timeout=30
```

- ❖ Show SBD header

```
rhel8# pcs stonith sbd status --full
```



SBD Fencing – Daemon

Step 5.2 – Setup SBD

Run on:

LPAR 1

❖ Enable SBD systemd daemon in cluster

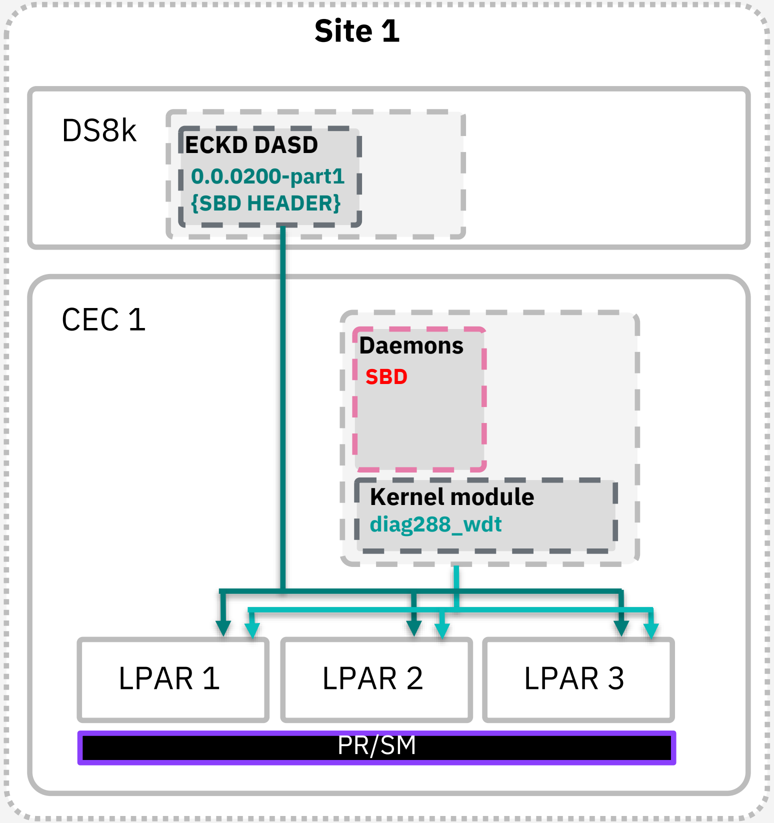
```
rhel8# pcs stonith sbd enable \
       watchdog=/dev/watchdog \
       device=/dev/disk/by-path/ccw-0.0.0200-part1 \
       SBD_DELAY_START=60 SBD_WATCHDOG_TIMEOUT=15
```

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

```
rhel8# pcs cluster stop --all
rhel8# pcs cluster start --all
```



SBD Fencing – Fence Agent

Step 5.3 – Setup SBD Fence Agent

Run on:

LPAR 1

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

```
rhel8# fence_sbd -o metadata|grep -A 2 power_timeout
```

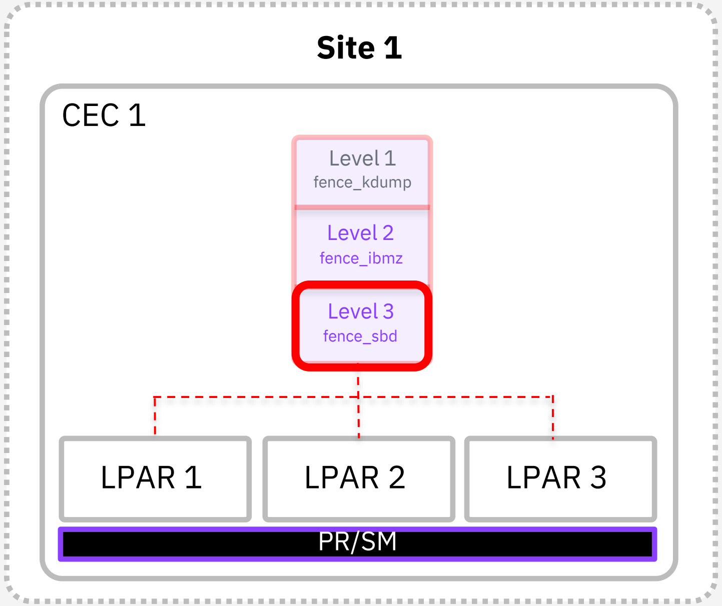
- Create SBD fence agent and set power_timeout

```
rhel8# pcs stonith create fence_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

```
rhel8# pcs stonith config
```



SBD Fencing – Testing

Step 5.4 – Test SBD fencing

Run on: LPAR 2

- ❖ Test sending of messages through A

```
rhel8# sbd -d /dev/disk/by-path/ccw-0.0.0200-part1 \
       message lpar1 test
```

Run on: LPAR 1

- ❖ Look at the Log of the SBD systemd service

```
rhel8# journalctl –u sbd -f
```

Run on: LPAR 2

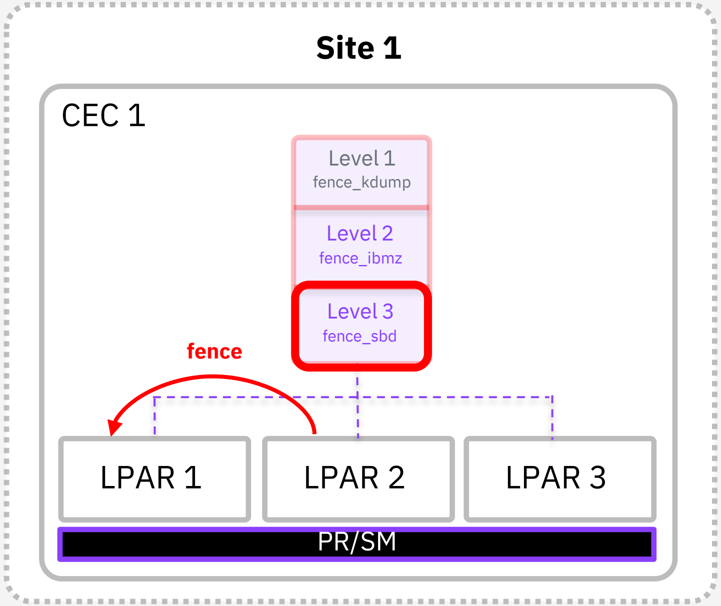
- ❖ Send poison pill from lpar2 to lpar1

```
rhel8# pcs stonith disable fence_ibmz
rhel8# time pcs stonith fence lpar1
rhel8# pcs stonith enable fence_ibmz
```

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

```
rhel8# systemd-analyze critical-chain
```



Fencing / STONITH fence_kdump (Level 1)

Fence_kdump

Step 6.0 – Configure kdump

Run on: LPAR 1

- ❖ Setup kdump

```
rhel8# systemctl enable kdump --now
```

Run on: LPAR 1 LPAR 2 LPAR 3

- ❖ Firewall rules

```
rhel8# firewall-cmd --add-port=7410/udp --permanent
rhel8# systemctl reload firewalld
rhel8# systemctl restart firewalld
```

- ❖ Edit kdump configuration

```
rhel8# vim /etc/kdump.conf
```

- Set Port to send the kdump message (see `man fence_kdump_send`)
  Set interval to every 10 seconds (forever):

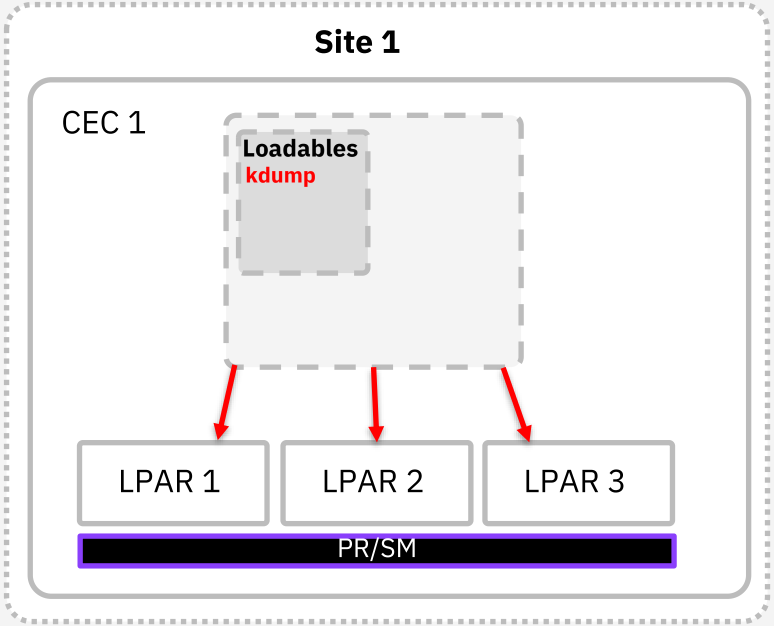
```
fence_kdump_args -p 7410 -f auto -c 0 -i 10
```

- all hostnames (cluster members) to send the kdump message to:

```
fence_kdump_nodes lpar1 lpar2 lpar2
```

- ❖ Restart kdump

```
rhel8# systemctl restart kdump
```



Additional documentation

- ❖ Red Hat article - fence_kdump: LINK
- ❖ Solution Assurance guide - kdump: LINK

Fence_kdump – fence agent

Step 6.1 – Add kdump fence agent

Run on:

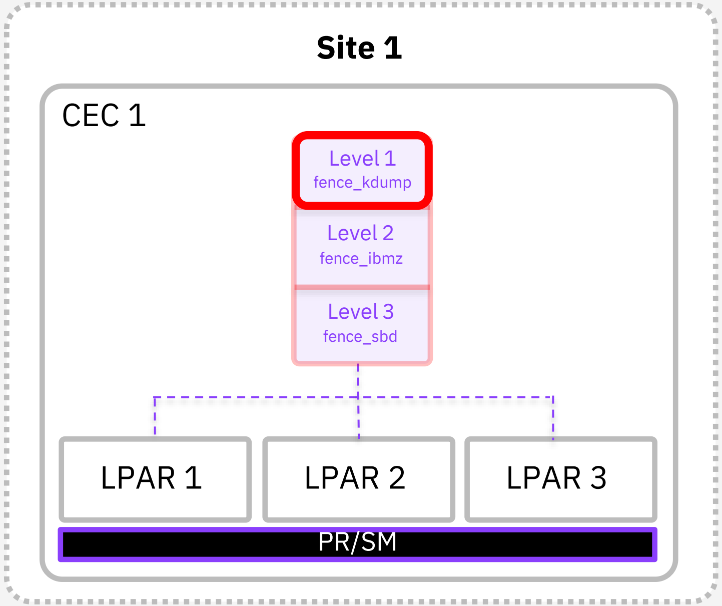
LPAR 1

- Create kdump fence agent

```
rhel8# pcs stonith create kdump fence_kdump \
           pcmk_reboot_action="off" \
           pcmk_host_list="lpar1 lpar2 lpar3" \
           verbose=1
```

- Will be tested in the cluster testing chapter

- See following chapter to add fence levels to your cluster.



Fencing / STONITH
Add fencing levels

Fencing levels

Step 7.0 – fence-levels:

Run on:

LPAR 1

❖ To order the execution of fence agents, fence levels are introduced. They are executed from low to high.

```
rhel8# pcs stonith level add 1 regexp%lpar[0-9] fence_kdump
rhel8# pcs stonith level add 2 regexp%lpar[0-9] fence_ibmz
rhel8# pcs stonith level add 3 regexp%lpar[0-9] fence_sbd
```

❖ Show and verify fencing levels:

```
rhel8# pcs stonith level
rhel8# pcs stonith level verify
```

❖ When you want to remove levels:

```
rhel8# pcs stonith level remove 1
```

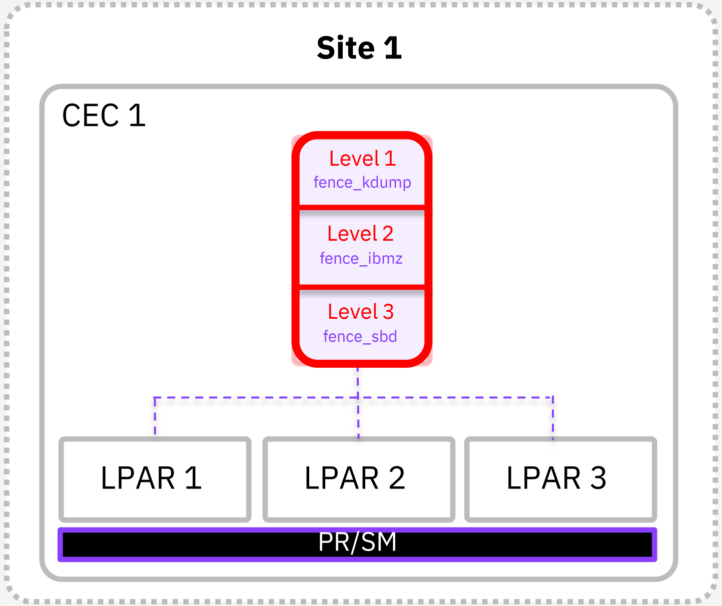
❖ Additional considerations:

- When your fencing is misconfigured, or the node has still a healthy cluster communication (e.g. when using fabric fencing) the node to be fenced is notified of its own fencing. In this case the fence-reaction property decides what happens. Panic is the safest choice and reboots the node.

```
rhel8# pcs property set fence-reaction=panic
```

- To prevent multiple fencing operations in parallel you can disable concurrent-fencing. In a 3-node cluster that can only withstand the failure of one node we might not need concurrent-fencing:

```
rhel8# pcs property set concurrent-fencing=false
```



GFS2
(Shared Storage)

GFS2 – Installation and Locking

Step 8.0 – GFS2 Packages

Run on:

LPAR 1
LPAR 2
LPAR 3

```
rhel8# subscription-manager repos \
       --enable=rhel-8-for-s390x-resilientstorage-rpms
rhel8# yum update -y
rhel8# yum install -y lvm2-lockd gfs2-utils dlm
```

Step 8.1 – GFS2 properties and locking

Run on:

LPAR 1

By default, the cluster stops all resources when the quorum is lost.
The GFS2 resource cannot be stopped because it relies on the quorum.
For this reason, the behavior must be changed to freeze all resources instead:

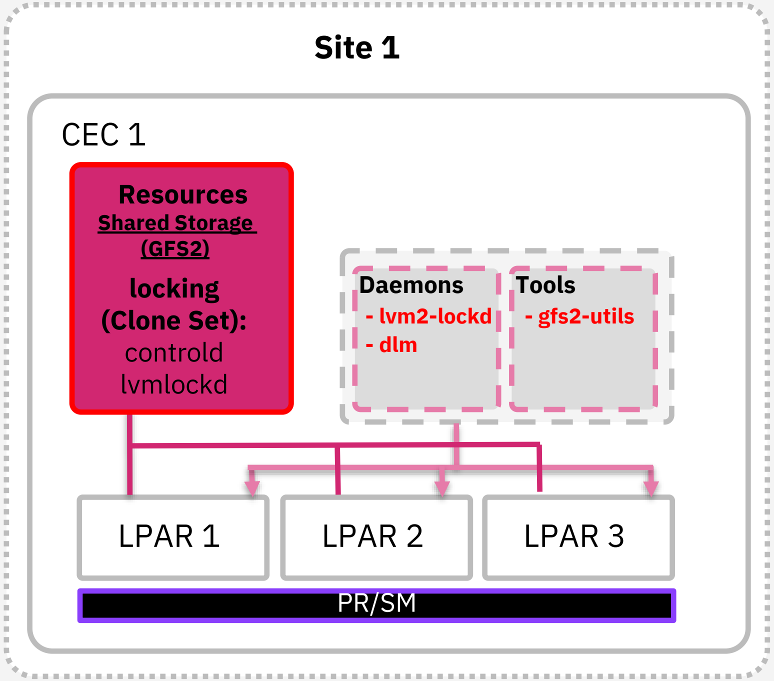
```
rhel8# pcs property set no-quorum-policy=freeze
```

DLM is used by lvmlockd for basic locking (read/write):

```
rhel8# pcs resource create dlm --group locking \
         ocf:pacemaker:controld op monitor interval=30s \
         on-fail=fence
rhel8# pcs resource clone locking interleave=true
```

Lvmlockd locks lvm metadata, validates caching of lvm metadata
and prevents activation conflicts :

```
rhel8# pcs resource create lvmlockd --group locking \
         ocf:heartbeat:lvmlockd op monitor interval=30s \
         on-fail=fence
```



GFS2 – Filesystem

Step 8.2 – Create GFS2 Filesystem

Run on: LPAR 1

- ❖ Shared Storage is already assumed to be setup (e.g. with chzdev `-e dasd 0.0.0201`)

- ❖ Create VG, LV and make GFS2 filesystem

Run on: LPAR 1

```
rhel8# vgcreate --shared shared_vg1 \
          /dev/disk/by-path/ccw-0.0.0201-part1
```

Note:
- Now the VG should be already visible on all LPARs

- ❖ Start the lock space on the other LPARs

Run on: LPAR 2-3

```
rhel8# vgchange --lock-start shared_vg1
```

- ❖ Create shared Logical Volume

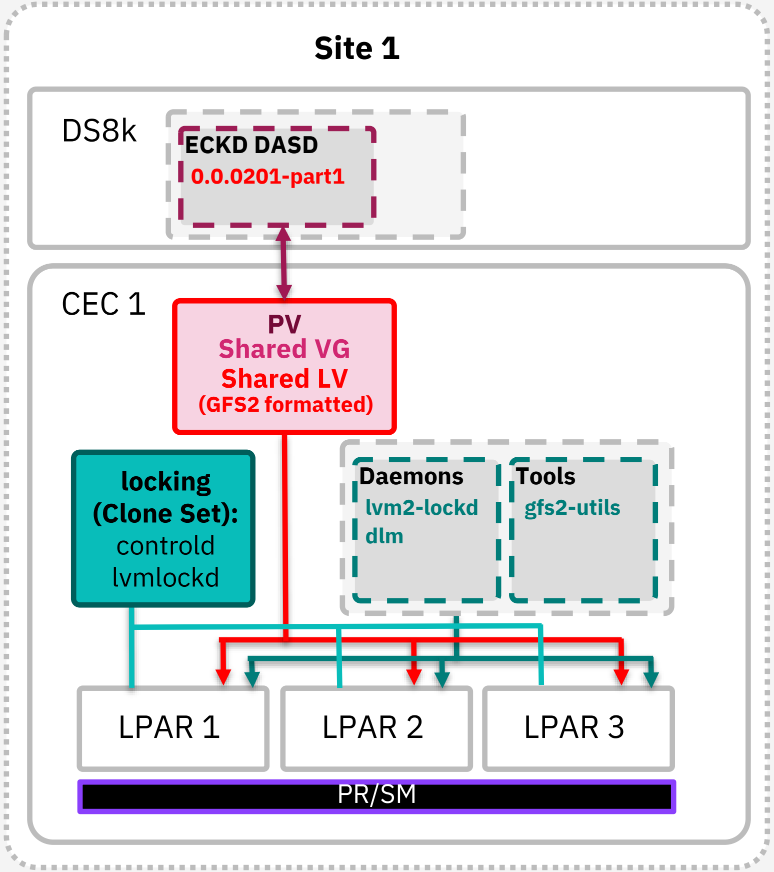
Run on: LPAR 1

```
rhel8# lvcreate --activate sy -l 100%FREE \
                -n shared_lv1 shared_vg1
```

- ❖ Create GFS2 filesystem

```
rhel8# mkfs.gfs2 -j3 -p lock_dlm \
                 -t my_cluster:gfs2-demo1 \
                 /dev/shared_vg1/shared_lv1
```

Note:
- Make sure to create 3 journals (1 journal for each cluster member)



GFS2 – Create Resources

Step 8.3 – Add LVM Resource to Cluster

Run on:

LPAR 1

❖ Create LVM Resource and required constraints

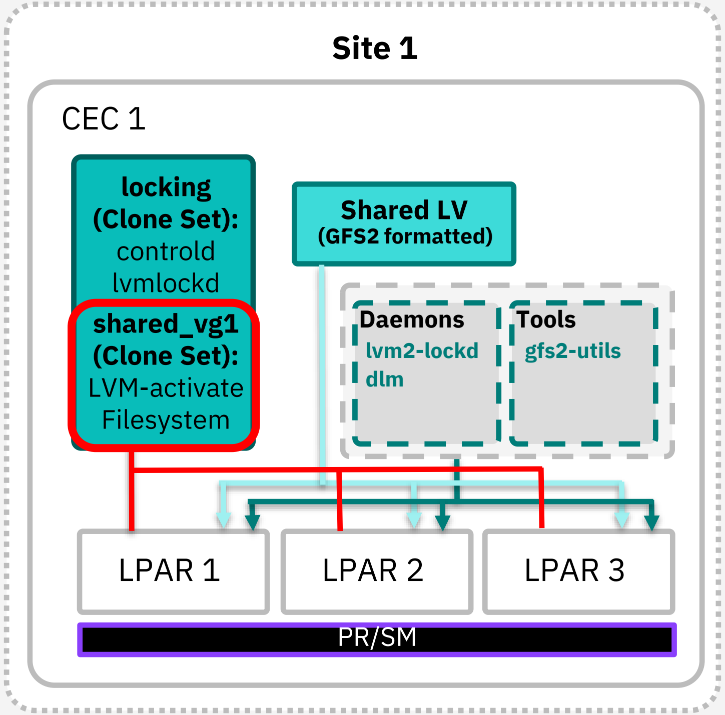
```
rhel8# pcs resource create sharedlv1 --group shared_vg1 \
          ocf:heartbeat:LVM-activate lvname=shared_lv1 \
          vgname=shared_vg1 activation_mode=shared \
          vg_access_mode=lvmlockd

rhel8# pcs resource clone shared_vg1 interleave=true

rhel8# pcs constraint order start locking-clone then shared_vg1-clone
rhel8# pcs constraint colocation add shared_vg1-clone with locking-clone
```

❖ Create Filesystem Resource

```
rhel8# pcs resource create sharedfs1 --group shared_vg1 \
          ocf:heartbeat:Filesystem \
          device="/dev/shared_vg1/shared_lv1" \
          directory="/shared-fs-1" fstype="gfs2" \
          options=noatime,nodiratime,context=system_u:object_r:svirt_image_t:s0 \
          op monitor interval=10s on-fail=fence
```



Site 1
CEC 1
locking (Clone Set): controld lvmlockd
shared_vg1 (Clone Set): LVM-activate Filesystem
Shared LV (GFS2 formatted)
Daemons
lvm2-lockd
dlm
Tools
gfs2-utils
LPAR 1
LPAR 2
LPAR 3
PR/SM

VirtualDomain

VirtualDomain – Creation

Step 9.0 – Create KVM Guest

Run on:

LPAR 1

- Check the Redhat Virtualization documentation: LINK. When following commands passes you are ready to create KVM guests:

```
rhel8# virt-host-validate
```

- Create a minimal kvm guest. E.g.:

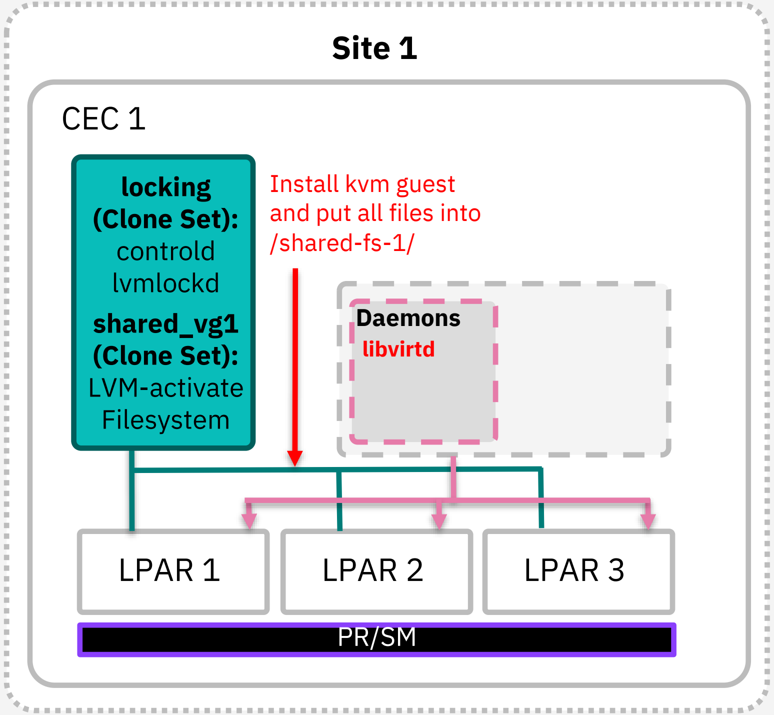
```
rhel8# virt-install \
         --name rhel8-guest1 \
         --memory 3000 \
         --vcpus 1 \
         --disk "size=4" \
         --location ${INSTALL_SERVER_URL}/s390x/RHEL8.5/DVD/ \
         --os-variant "rhel8.5" \
         --network "network=default" \
         --initrd-inject "/root/${YOUR_KICKSTART_FILE}.ks" \
         --extra-args "ks=file:///${YOUR_KICKSTART_FILE}.ks" \
         --noautoconsole
```

- Dump the guest configuration to the GFS2 shared directory

```
rhel8# virsh dumpxml rhel8-guest1 > /shared-fs-1/rhel8-guest1.xml
```

- Move the qcow image to the shared directory.

```
rhel8# mkdir /shared-fs-1/images
rhel8# virsh shutdown rhel8-guest1
rhel8# mv /var/lib/libvirt/images/* /shared-fs-1/images/
rhel8# restorecon -vr '/shared-fs-1'
```



Site 1

CEC 1

locking (Clone Set): controld lvmlockd

shared_vg1 (Clone Set): LVM-activate Filesystem

Install kvm guest and put all files into /shared-fs-1/

Daemons

libvirtd

LPAR 1

LPAR 2

LPAR 3

PR/SM

VirtualDomain – Add KVM Guest

Step 9.1 – Add VirtualDomain Resource

Run on:

LPAR 1

❖ Make sure the kvm hypervisor is reachable via ssh from all cluster members (without password prompting).

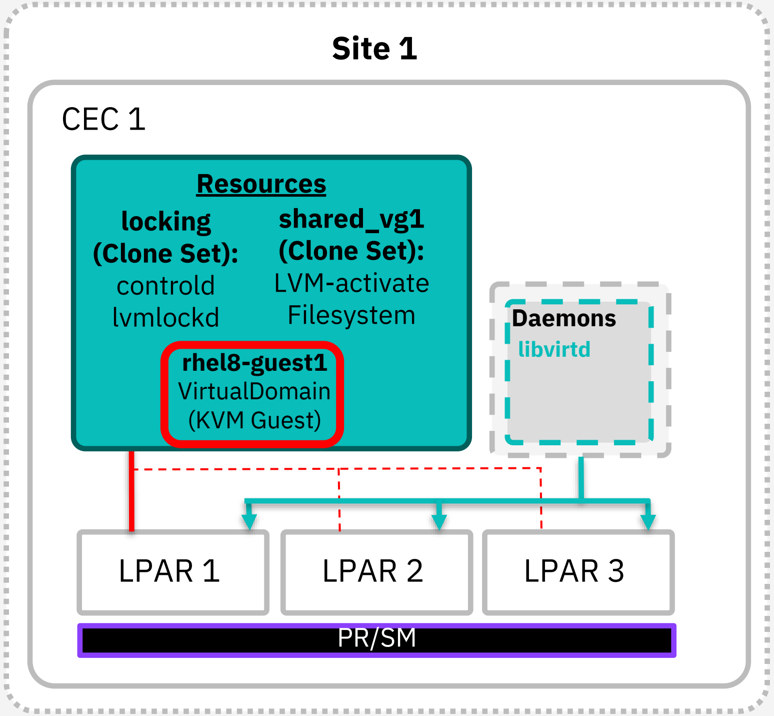
```
rhel8# virsh -c qemu+ssh://lpar2/system
```

❖ Add VirtualDomain Resource and required constraint:

```
rhel8# pcs resource create rhel8-guest1 \
          ocf:heartbeat:VirtualDomain \
          config="/shared-fs-1/rhel8-guest1.xml" \
          hypervisor="qemu:///system" \
          migration_transport="ssh" \
          meta allow-migrate=true

rhel8# pcs constraint order start shared_vg1-clone then rhel8-guest1
```

Note:
- The allow-migrate option allows live migration of the KVM guest when you move it manually.



Cluster Testing

Cluster Testing

Step 10.0 – Test Workload Failover

Run on: LPAR 2

❖ Watch the cluster status (live)

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

Run on: LPAR 1

❖ Trigger a kernel panic

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

Note:
- you should see a kdump written to the /var/crash directory

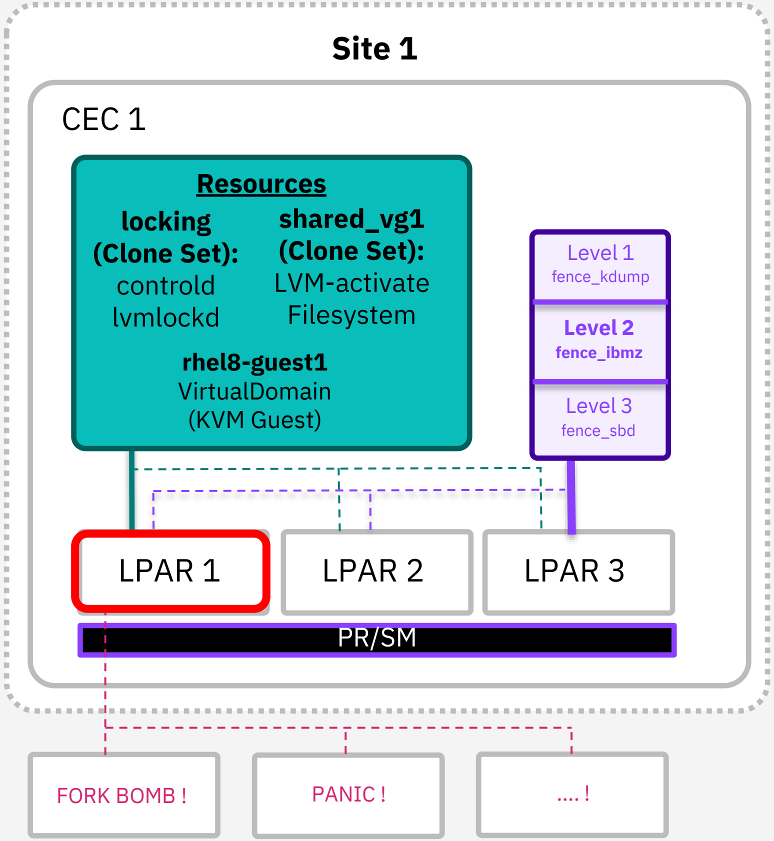
❖ Check systemd logs (errors, warnings…)

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

❖ Fork Bomb:

```
# :() { :|:& }; :
```

Note:
- It might take some time for the cluster to hang
- The deactivate step in the fence_ibmz fence agent might take longer


Site 1
CEC 1
Resources
locking (Clone Set): controld lvmlockd
shared_vg1 (Clone Set): LVM-activate Filesystem
rhel8-guest1 VirtualDomain (KVM Guest)
Level 1 fence_kdump
Level 2 fence_ibmz
Level 3 fence_sbd
LPAR 1
LPAR 2
LPAR 3
PR/SM
FORK BOMB !
PANIC !
.... !

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:

- ❖ Create new user LINSHARE

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

z/VM

- ❖ Link Minidisks

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

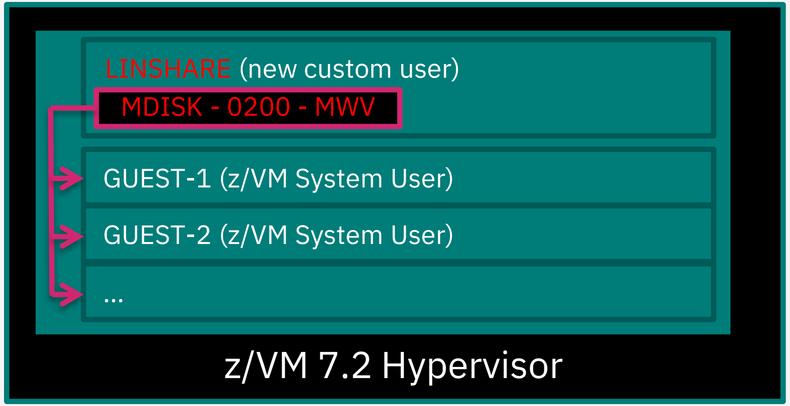
- ❖ De-ignore, enable and make DASD persistent:

```
# chzdev -e dasd 0.0.0200
```

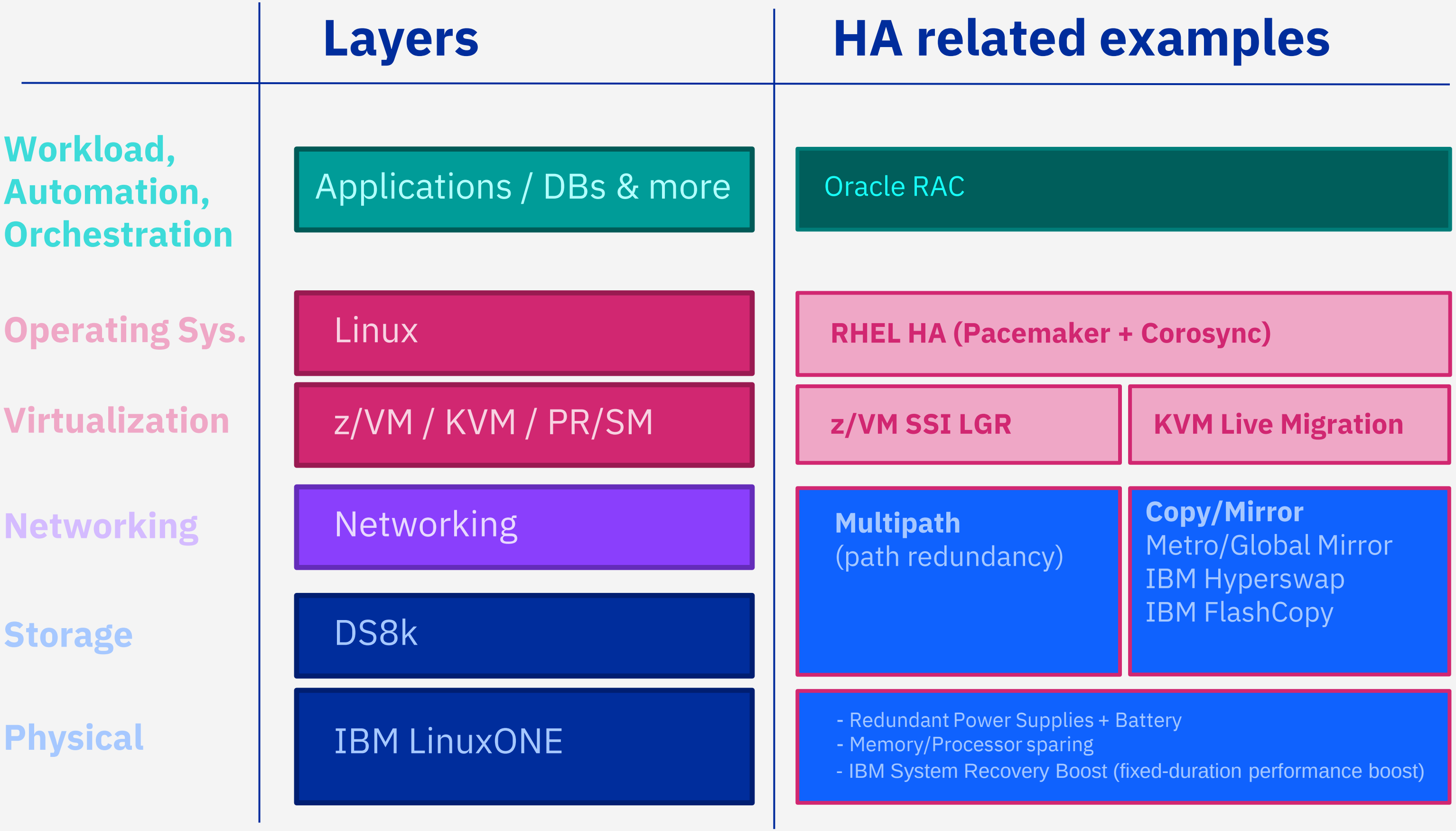
every guest in cluster

- ❖ 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
```


LINSHARE (new custom user)
MDISK - 0200 - MWV
GUEST-1 (z/VM System User)
GUEST-2 (z/VM System User)
...
z/VM 7.2 Hypervisor

Appendix B – Introduction – High Availability Stack



	Layers	HA related examples	
Workload, Automation, Orchestration	Applications / DBs & more	Oracle RAC	
Operating Sys.	Linux	RHEL HA (Pacemaker + Corosync)	
Virtualization	z/VM / KVM / PR/SM	z/VM SSI LGR	KVM Live Migration
Networking	Networking	**Multipath** (path redundancy)	**Copy/Mirror** Metro/Global Mirror IBM Hyperswap IBM FlashCopy
Storage	DS8k		
Physical	IBM LinuxONE	- Redundant Power Supplies + Battery - Memory/Processor sparing - IBM System Recovery Boost (fixed-duration performance boost)	