Configuration Manual

CONETTIX B6800 Central Station Software Receiver

Failover Configuration and Procedure

Contents

Introduction

This document titled B6800 Failover Configuration and Procedure has been written to provide how failover is to be handled based on our UL approval.

Document Scope

This document will discuss the overall benefits, strategies, and complexities of running the B6800 System in a failover configuration. Also provided is a set of scripts that allow you to set up a simple failover procedure using the standard B6800 installations across two servers, and instruction to help you through the installation, configuration and how to start a failover procedure up.

System Overview and Purpose

The purpose of this document is to outline the configuration and illustration for setting up the CONETTIX B6800 System in a simple failover configuration. This approach involves running two instances of the B6800: a primary (Active) and a secondary (Standby) system. The Active Sytems is responsible for capturing events from devices and forwarding them to automation, while the Standby System serves as a ready and waiting “hot-spare” backup in case the Active System fails.

In the event of a failure on the Active System, the Standby System is configured to automatically and seamlessly take over without impacting devices or automation. Real-time replication ensures that the active configuration is continuously synchronized with the standby system, eliminating the need for manual synchronization. There is no loss of events, data, and configuration during a failover event with this configuration as provided.

1.1 B6800 System Overview

The B6800 System is deployed and installed as a Virtual Machine, which runs on the Windows operating system using Hyper-V, the built-in virtualization or hypervisor solution on Windows. During installation, a dedicated Windows Installer is utilized to configure Hyper-V if it is not yet installed.

Within the Virtual Machine, the B6800 System is composed of multiple isolated containers, each serving a distinct function. These containers provide the necessary components and services for the B6800 platform. Additionally, the B6800 extensively utilizes PostgreSQL as the internal database, responsible for managing configuration and runtime persistence.

The virtual machine is running a Linux operating system. The distribution used is Alpine Linux, a small, fast, and stable base for system that are to be virtualized and require great stability.

The B6800 also utilized the stable PostgreSQL database as its underlying datastore. PostgreSQL will hold all the authentication, configuration, and run-time event data that the B6800 receives and sends.

Within the Linux operating system, the B6800 also utilizes a container framework called Podman for hosting and managing the system containers. Podman facilitates efficient operation and organization of the system components, enhancing scalability, portability, and management capabilities.

1.2 What are containers and why are they important to the B6800 System?

When running mission-critical applications where stability and uptime are crucial, a container-based architecture, using tools like Podman, offers significant benefits.

Containers provide isolated and portable environments for applications, running on a virtual machine (VM) to provide the necessary hardware and operating system resources.

Here are the advantages of a container-based architecture:

1) Isolation and Portability: Containers ensure strong isolation along with encapsulating applications and their dependencies within their own runtime environment. This isolation prevents conflicts between applications and allows for seamless portability across different infrastructure setups, such as local development environments, cloud platforms, or on-premises servers.

2) Efficiency and Resource Optimization: Containers are lightweight and share the host machine's operating system kernel. This approach requires fewer resources compared to running multiple virtual machines, resulting in improved efficiency and better resource utilization. It helps optimize hardware resources and reduces costs.

3) Rapid Deployment and Scalability: Containers enable fast application deployment and scaling. They can be quickly started or stopped, facilitating rapid updates and rollbacks. Additionally, containers can be replicated and scaled horizontally to meet changing demands, ensuring high availability and responsiveness even during peak usage periods.

4) Dependency Management and Consistency: Containers package applications with their dependencies, creating consistent and reproducible environments. This eliminates issues related to version conflicts or missing dependencies, improving stability, and reducing runtime errors.

5) Fault Isolation and Quick Recovery: In a container-based architecture, if a containerized application fails or experiences issues, it does not impact other containers or the underlying VM. Fault isolation enhances stability and uptime, as issues can be contained and resolved without affecting the overall system.

Tools like Podman provide mechanisms for automated recovery and quick container restarts in case of failures. By leveraging Podman in a container-based architecture running on a virtual machine, we gain stability, fault tolerance, and scalability.

The isolation provided by containers ensures that critical applications remain unaffected by issues in other parts of the system, reducing the risk of downtime. The ability to rapidly scale containers up or down allow for seamless handling of fluctuations in application load, ensuring consistent performance and availability.

Moreover, the portability and consistency of containers simplify maintenance and updates for mission-critical applications. Encapsulating dependencies and runtime environments ensure consistent deployment across different environments, reducing compatibility issues and minimizing downtime during updates or migrations.

1.3 Containers make failover better

A container-based approach significantly enhances the ability to implement a failover solution. By encapsulating applications and their dependencies within isolated containers, the failover process becomes more streamlined and efficient. In the event of a failure or issue with a containerized application, the failover mechanism can seamlessly switch to a standby container on another node within the cluster, ensuring minimal downtime and uninterrupted service availability.

1.4 PostgreSQL containers and failover

A container-based approach brings several benefits when replicating PostgreSQL databases between failover cluster nodes. Containers provide a lightweight and portable environment for running database instances, making it easier to replicate and synchronize databases across nodes. The encapsulation of the database and its dependencies within containers ensures consistency and reproducibility, simplifying the replication process. In the event of a node failure, the failover mechanism can quickly redirect client requests to a standby container running on another node, minimizing downtime, and ensuring continuous access to the replicated database.

1.5 UL Background

Release 1.1 of the B6800 System can be configured in a failover setup to satisfy the failover requirements from UL 864 (Fire) and UL 2610 (Intrusion).

This failover illustration is meant to show this capability exists to meet these key performance points from the UL standards:

From UL 864 10th edition section 40.2:

- (d) All supervising station signal processing equipment shall be completely duplicated with provision for automatic switchover to the backup system within 30 s without loss of any signals. The backup computer shall have equivalent or greater capabilities of the primary, such as memory, speed, data storage, and the like.
- (e) Failure of any part of the main system configuration, shall result in automatic switchover to the backup system without loss of any signals and shall be indicated by an audible or obvious visual indication at the constantly attended supervising station's operator interface, all within 30s.

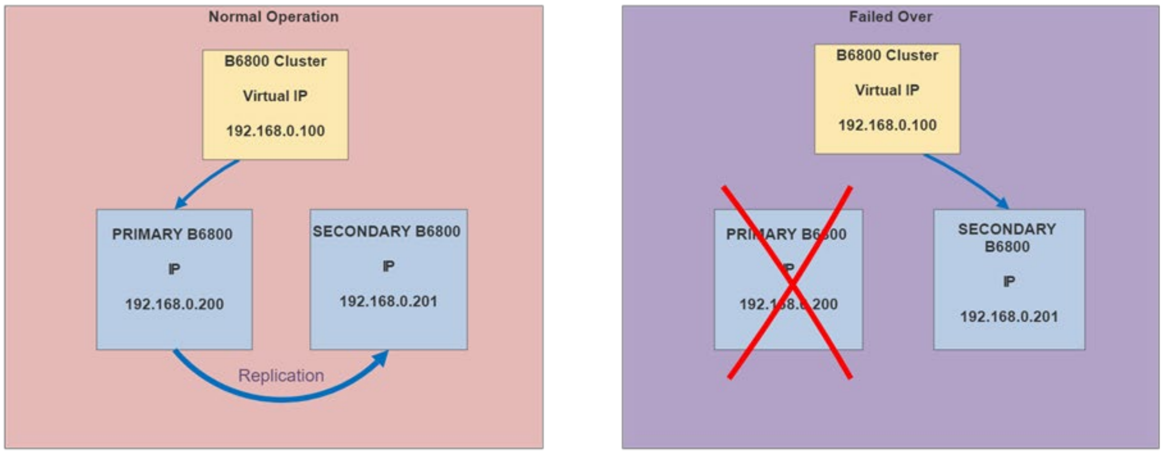
From UL 2610 section 37.4

- (h) All receiving equipment shall be completely duplicated with provision for switchover to the backup system within 30 seconds. The backup system shall be fully operational within 6 minutes of the loss of the primary system. This allows 30 seconds for the backup system to be fully energized and connected to necessary communication lines and other devices, followed by 5-1/2 minutes for the system to boot up, conduct memory tests, file system check, security verifications and prepare for full system operation. The backup computer shall have equivalent or greater capabilities of the primary, such as memory, speed and the like.

- (i)Failure of the main computer system, hard disk, and alarm monitor shall result in switchover to the backup system and shall be indicated by an audible or obvious visual indication.

Failover Configuration makes use of a Virtual IP

B6800 Failover Cluster Illustration



The Bosch B6800 failover configuration utilizes two identically configured B6800 systems for redundancy. These systems communicate and monitor each other through a network-based heartbeat. The active node handles incoming traffic, and a Virtual IP (VIP) directs the traffic to this node. A Linux-based service, called Keepalived, is employed in conjunction with a PostgreSQL cluster featuring synchronous replication for both connectivity and data redundancy.

1.6 Keepalived

A Virtual IP (VIP) is a unique IP address assigned to a group of servers, serving as a centralized access point for clients. Panels, Automation, and users connect to the VIP instead of individual B6800 IP addresses, abstracting away the underlying server infrastructure. Keepalived continuously monitors the server health and dynamically assigns the VIP to the active or available system. In the event of the active system failure, Keepalived swiftly transfers the VIP to the standby system, ensuring nearly uninterrupted service availability. This VIP-based approach enables fault tolerance and high availability by transparently directing client traffic to the appropriate server in the cluster.

1.7 PostgreSQL Replication

To ensure data consistency and synchronization, database replication is implemented between the active and standby nodes in the B6800 Failover configuration. This replication mechanism minimizes data discrepancies by keeping both databases closely aligned.

For Event or Automation traffic for the B6800 cluster, transactional persistence is enforced on both nodes. This guarantees that data is redundantly written to disk on both nodes before acknowledging to the caller, ensuring data integrity and eliminating the risk of data loss.

1.8 How Keepalived and PostgreSQL replication work together

Clients connect to the PostgreSQL cluster using the VIP provided by Keepalived, enabling connectivity to the active node regardless of its physical location.

In the event of a active node failure, the VIP is quickly transferred to the standby node, which seamlessly assumes the functionality and responsibilities of the primary node, minimizing downtime.

The PostgreSQL cluster consists of a active node (Node A) and a standby node (Node B), with synchronous replication and the *synchronous_commit* setting enabled to ensure data modifications are written to both nodes before confirming a successful transaction.

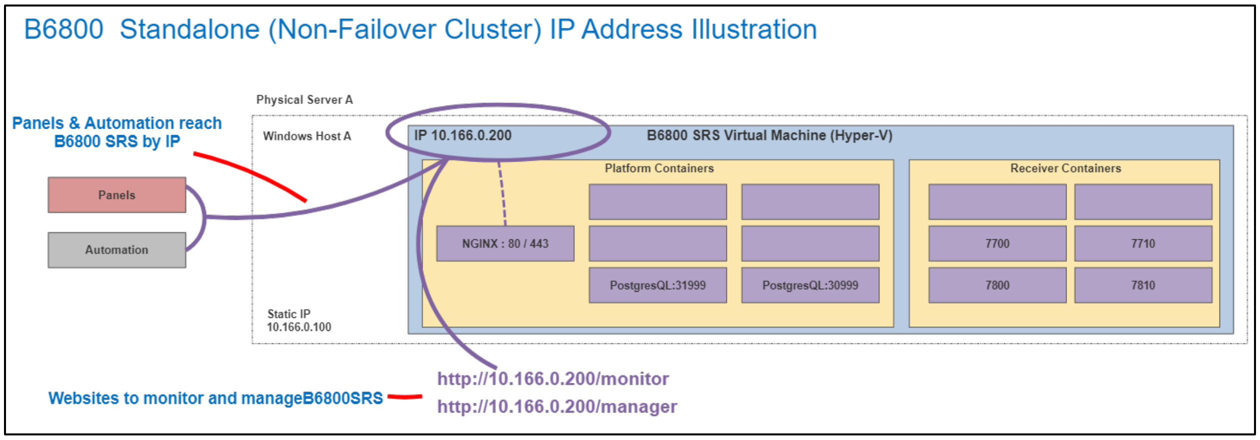
Synchronization between Keepalived and the PostgreSQL cluster during failover is achieved using the notify.sh script. This script updates the PostgreSQL cluster configuration, designating the new active (the backup) node as the surviving PostgreSQL server and the previous active node is disabled. It then restarts the PostgreSQL service on the surviving server.

Overall, this implemented solution combines Keepalived for server redundancy and synchronous replication in the PostgreSQL cluster to provide both server connectivity and data redundancy.

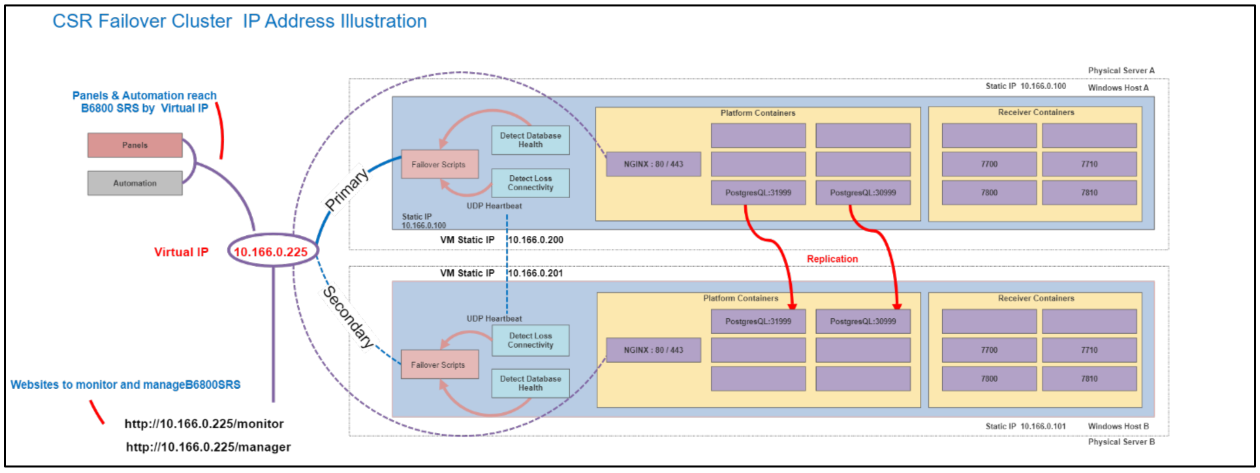
The failover configuration, including redundant B6800 systems, a network-based heartbeat, VIP usage, and database replication, ensures continuous availability, data integrity, and rapid failover capabilities in the face of active server/node failure.

1.9 Implementation

In a stand-alone B6800 System, the panels, automation and user connect to the server by utilizing the IP address of the Virtual Machine where the system is installed.


B6800 Standalone (Non-Failover Cluster) IP Address Illustration
Physical Server A
Panels & Automation reach B6800 SRS by IP
Windows Host A
IP 10.166.0.200
B6800 SRS Virtual Machine (Hyper-V)
Platform Containers
Receiver Containers
Panels
Automation
NGINX : 80 / 443
7700
7710
PostgresQL:31999
PostgresQL:30999
7800
7810
Static IP 10.166.0.100
Websites to monitor and manageB6800SRS
http://10.166.0.200/monitor
http://10.166.0.200/manager

In a failover configuration, a VIP (Virtual IP) is a virtual network address that serves as the entry point for accessing the cluster's services. Configuring the VIP in your applications allows them to interact with the platform cluster seamlessly.



Here's how the VIP works in relation to the other applications:

VIP as the Single-Entry Point: The VIP is the address that you configure in all your other applications to access the platform cluster. Instead of specifying the individual IP addresses of the cluster nodes, applications communicate with the cluster through the VIP. This simplifies the configuration and management of application connections.

Failover: In case of a node failure, the VIP moves and automatically redirects the traffic to a healthy node, ensuring high availability and seamless failover.

Transparent Service Access: When your applications use the VIP, they are shielded from the underlying complexity of the cluster infrastructure. They interact with the platform cluster as if it were a single, cohesive entity. Applications can send requests to the VIP without needing to be aware of the specific nodes handling those requests. This abstraction simplifies application development and maintenance.

Centralized Configuration: By using the VIP, you centralize the configuration of your applications to point to a single address. This simplifies application deployment and reduces the need for manual reconfiguration or redeployment when making changes to the cluster infrastructure.

Overall, configuring the VIP in your applications ensures a seamless and consistent connection to the platform cluster. It provides failover capabilities, transparent access to cluster services, scalability, and centralized configuration management. By using the VIP, your applications can effectively leverage the benefits and capabilities of the platform cluster without needing to manage the intricacies of individual node addresses.

B6800 Failover Sample Installation

For release 1.1, the failover features are included as scripts that can be run to make two standalone B6800 system installations into a two-node failover cluster.

To install the Bosch provided failover configuration, scripts will be run on each of two distinctly installed B6800 Systems. The scripts will configure each system to be either the "Primary" (Active) node or "Secondary" (Standby) node of the cluster. Once each server has been configured as one of the failover cluster nodes, the cluster can be started.

1.10 Prerequisites

Two distinct hardware platforms (hosts) with network adapters.

Two valid licenses of the B6800 System

The B6800 Installer application

Basic knowledge of using and configuration of a Linux OS. The installer is expected to be an IT professional with some previous experience establishing cluster configurations and familiarity with the terminology used.

1.11 Installation Overview

These are the high-level summary of what steps will be needed to configure the failover cluster. You don't need to do these steps just yet, that will come in the next "Detailed Installation Instructions". This is just to illustrate what the entire process will entail.

1. First, you will run the B6800 installer on both hosts.
    a. The failover is configured "on top" of two standalone receivers
2. After installation, apply the B6800 license to each system.
3. Figure out what your failover configuration will be.
    a. Determine which machine is intended to be the Active system which is to be the Standby system.
    b. Determine your network setup.
        i. What will be the IP address of each system?
        ii. What will be your Vrtual IP (VIP) Address?

4. Configure the first system as Active node on the cluster (see detailed instruction below).

5. Configure the second system as the Standby node on the cluster (see detailed instruction below).

6. Once the cluster is established and up and running, you can configure the cluster to work with your devices and your automation.

1.12 Beginning the Failover Installation- Detailed Steps and Instructions

We will be first installing the B6800 on two separate hosts. Once those installations are completed, use the already included scripts to modify the standard installations to become the cluster's "Primary" (Active) node/server, while the other will be the "Secondary" (Standby) node/server. Note that there are different scripts for the primary and secondary systems, but the installation process on each server is similar.

1.12.1 Cluster Network Decisions

To build a B6800 failover cluster using the provided scripts, you will need to get organized and dedicate some IP addresses from the local network.

Each node will need an IP address. This IP address is assigned during the installation of the B6800 system. It is recommended to assign either static IPs or DHCP reservations to the installed B6800 virtual machines. This ensures that the "real" IP of each failover cluster node remains unchanged.

The failover cluster will utilize a Virtual IP (also known as a floating IP). During startup, the cluster will create and assign this Virtual IP to the local network. It is crucial to ensure that no other "real" device is using this Virtual IP and that it is available and dedicated for the cluster use only.

For this illustration, make sure the virtual IP is on the same subnet as the B6800 Systems “real” IP addresses.

Select a Virtual IP that is not currently in use by any machine or server on the local network. Ensure that the chosen IP falls outside any DHCP range of IP addresses that might be dynamically assigned to other devices.

When it comes time establish each failover cluster node, you will need to provide specific information as command line parameters when running the scripts. This includes the desired "Virtual IP" for the cluster to run from and the static IP of each server/node (one for the primary and one for the secondary).

Needed before continuing:

Primary Server “Real” IP Address (the initial Primary, it would change after failover)

Secondary Server “Real” IP Address (the initial Secondary, it would change after failover)

Reserved Virtual IP address for the cluster IP

Once you have gathered the necessary information and made the required preparations, you are ready to proceed with the installation process. Let's begin!

1.12.2 Common Steps- Same for Primary and Secondary Servers/nodes

These first steps are common to each B6800 System. Please follow these instructions first on each system before creating the “Primary” and “Secondary” nodes using the provided scripts.

First, some common steps for each of the cluster nodes:

1) Login to the console of each Host VM of each installed B6800 from the Hyper-V Management Console.

#Login: root
#Password: (the root password that you entered during the installation)

2) After login, you should be in /root directory, but if not:

#cd /root
#ls *.tar

The failover scripts are in a TAR file stored in the **/root** directory. You should see a file called **failoversample.tar**/

We've included a script to help you untar this file right in the root directory. Just run:
#./untar.sh

(Yes, you will need the "period" and the "forward slash" before the name of the script... it's a Linux thing!)

The untar script will deliver all necessary failover scripts and files to the proper directories.

At this point, you should have all the scripts that you need delivered ready to run.

This is the end of the common steps for both primary and secondary servers. From here on, only follow the instruction for the specific node you are working on.

Note: Optional! Create a VM "export" here if you'd like on each server. Then you can always restart from here.

Now it's time to build out the specifics to make both a Primary and Secondary failover cluster node.

1.12.3 Primary Cluster Node Instructions

On the VM that is to become your primary (initial active), make sure you are in the /root directory. If not, change to that directory using the following command:
#cd /root

Follow these steps to run the following script to set up the primary node.

The script takes three parameters as command line arguments:

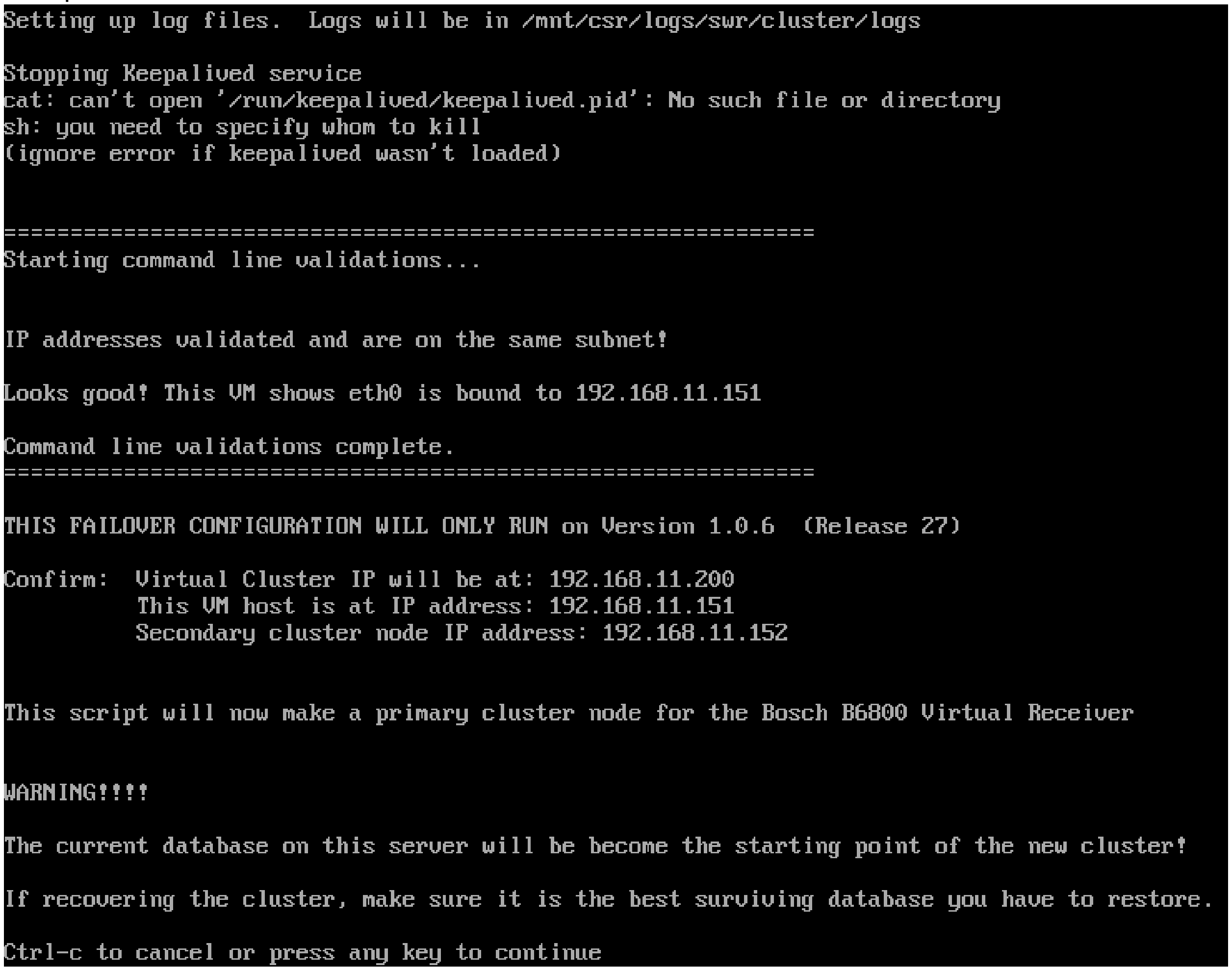
- The first parameter is the IP address of the local VM, which is the VM's "real" IP. You also specified this when you installed the B6800.
- The second parameter is the virtual IP address of the cluster, which is the IP address that the CSR will be available at for all UI, panels, and automation systems.
- The third parameter is the IP address of the secondary cluster node.

Script: **makeclusterprimary.sh**

Here's the syntax of the command:
#./makeclusterprimary.sh <ipaddressofthisvm> <vipaddressofcluster> <ipaddressofsecondaryserver>
(Spaces between the IP addresses)

Remember to include the period and slash before the script name. For example:
#./makeclusterprimary.sh 192.168.0.100 192.168.0.200 192.168.0.101

The script will set up the environment on this installation to allow it to become a cluster primary node. When the script nearly finishes, the primary node should be setup. You should see the next screen.



```
Setting up log files.  Logs will be in /mnt/csr/logs/swr/cluster/logs

Stopping Keepalived service
cat: can't open '/run/keepalived/keepalived.pid': No such file or directory
sh: you need to specify whom to kill
(ignore error if keepalived wasn't loaded)


=============================================================
Starting command line validations...


IP addresses validated and are on the same subnet!

Looks good! This VM shows eth0 is bound to 192.168.11.151

Command line validations complete.
=============================================================

THIS FAILOVER CONFIGURATION WILL ONLY RUN on Version 1.0.6  (Release 27)

Confirm:  Virtual Cluster IP will be at: 192.168.11.200
          This VM host is at IP address: 192.168.11.151
          Secondary cluster node IP address: 192.168.11.152


This script will now make a primary cluster node for the Bosch B6800 Virtual Receiver


WARNING!!!!

The current database on this server will be become the starting point of the new cluster!

If recovering the cluster, make sure it is the best surviving database you have to restore.

Ctrl-c to cancel or press any key to continue
```

DON'T press the final confirmation until the secondary node is ready. The last step the script will perform will be to start the cluster and broadcast the Virtual IP to the network.

Before moving on to setting up the secondary node, don't forget to perform the common steps detailed above for both servers.

1.12.4 Secondary Cluster Node Instructions

Before proceeding with the instructions below, ensure that you have followed the common steps for both servers as detailed earlier in this document.

To set up the secondary cluster node:

On the VM that is to become your secondary (initial standby), make sure you are in the /root directory. If not, change to that directory using the following command:
#cd /root

Follow these steps to run the following script to set up the secondary node.

The script takes in three parameters:

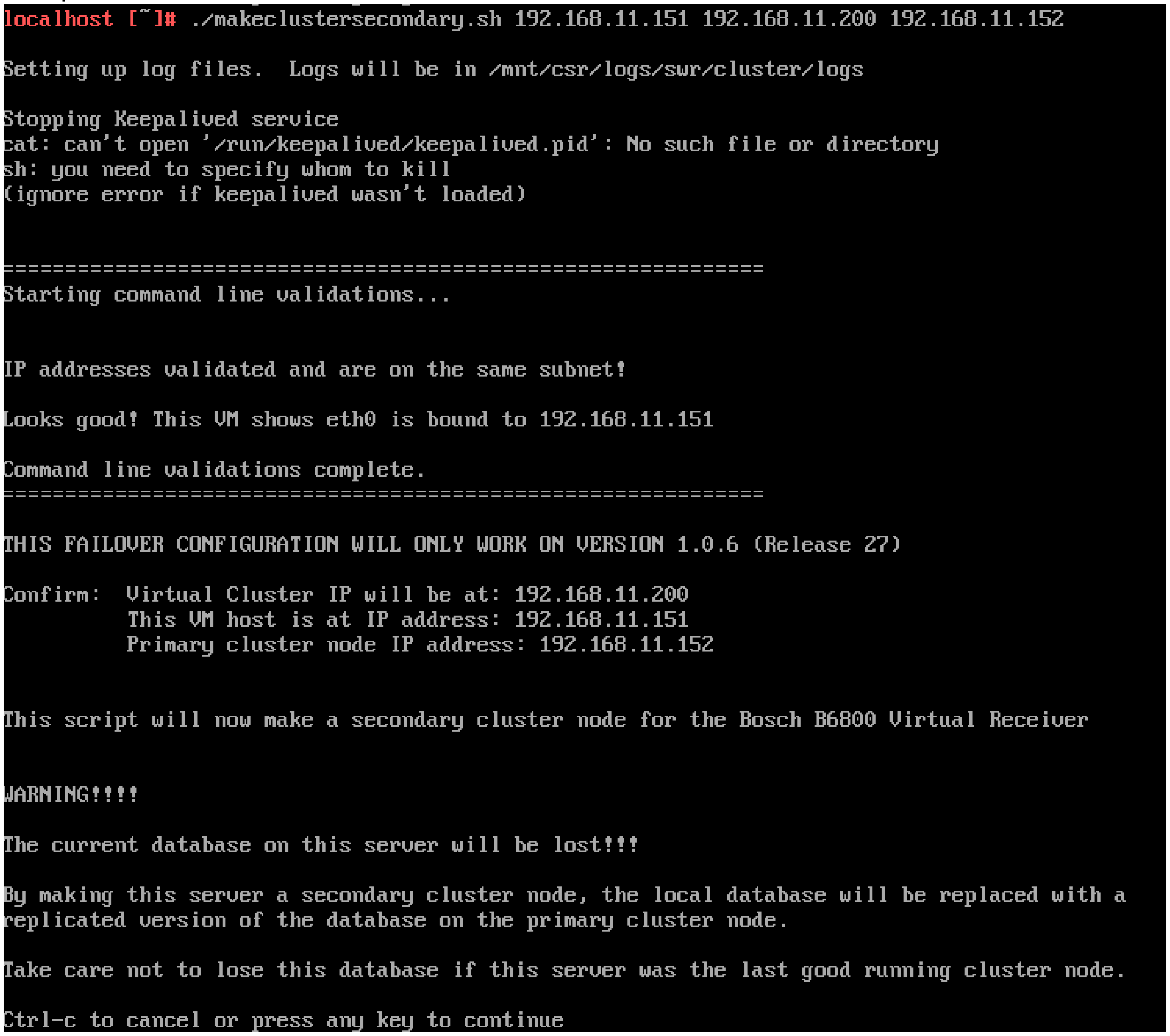
- The first parameter is the IP address of the local VM, which is the VM's “real” IP.
- The second parameter is the virtual IP address of the cluster, which is the IP address that the CSR will be available at for all UI, panels, and automation systems.
- The third parameter is the IP address of the primary cluster node.

Script: **makeclustersecondary.sh**

Here's the syntax of the command:
#./makeclustersecondary.sh <ip_address_of_this_vm> <vip_address_of_cluster> <ip_address_of_primary_vm>
(Spaces between the IP addresses)

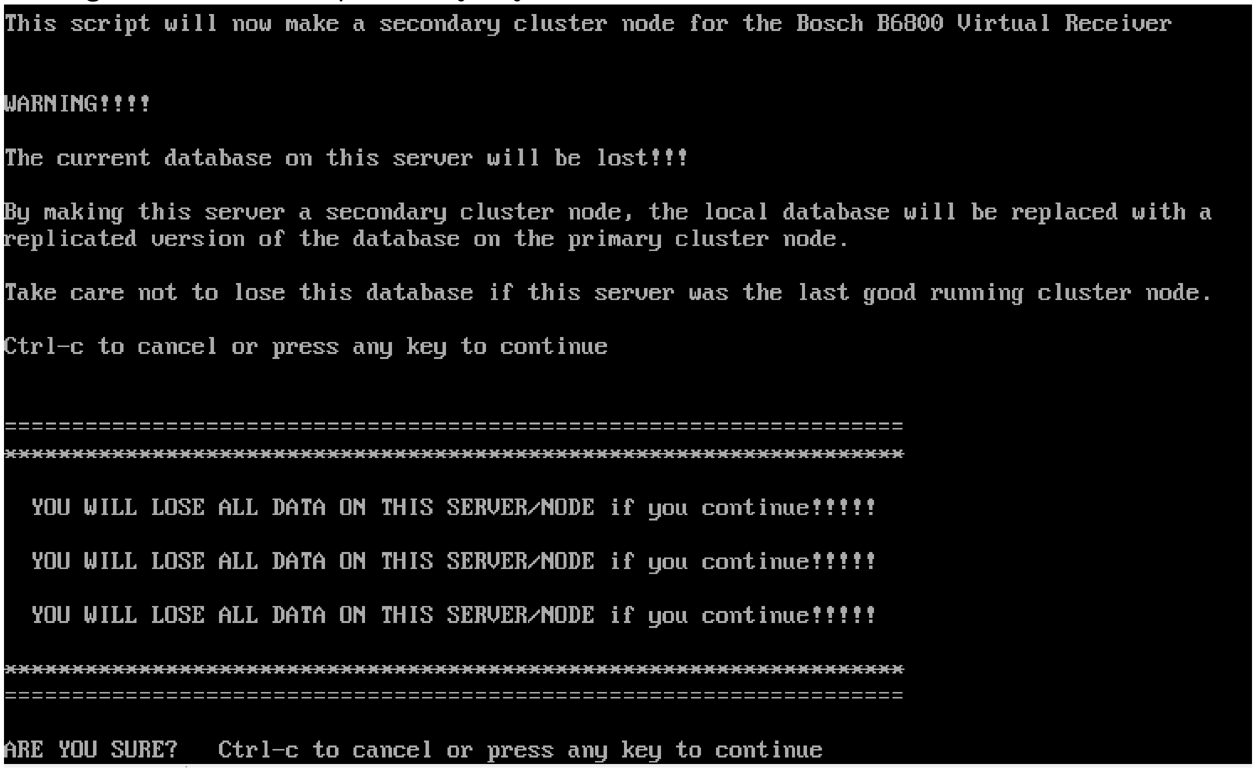
Remember to include the period and slash before the script name. For example:
#./makeclustersecondary.sh 192.168.0.101 192.168.0.200 192.168.0.100

The script will set up the environment on this installation to allow it to become a cluster secondary node. When the script nearly finishes, the secondary node should be setup. You should see the next screen.



```
localhost [~]# ./makeclustersecondary.sh 192.168.11.151 192.168.11.200 192.168.11.152

Setting up log files.  Logs will be in /mnt/csr/logs/swr/cluster/logs

Stopping Keepalived service
cat: can't open '/run/keepalived/keepalived.pid': No such file or directory
sh: you need to specify whom to kill
(ignore error if keepalived wasn't loaded)


============================================================
Starting command line validations...


IP addresses validated and are on the same subnet!

Looks good! This VM shows eth0 is bound to 192.168.11.151

Command line validations complete.
============================================================

THIS FAILOVER CONFIGURATION WILL ONLY WORK ON VERSION 1.0.6 (Release 27)

Confirm:  Virtual Cluster IP will be at: 192.168.11.200
          This VM host is at IP address: 192.168.11.151
          Primary cluster node IP address: 192.168.11.152


This script will now make a secondary cluster node for the Bosch B6800 Virtual Receiver


WARNING!!!!

The current database on this server will be lost!!!

By making this server a secondary cluster node, the local database will be replaced with a
replicated version of the database on the primary cluster node.

Take care not to lose this database if this server was the last good running cluster node.

Ctrl-c to cancel or press any key to continue
```

A confirmation screen should eventually appear to confirm all you configuration settings. If all correct, press any key to continue.



```
This script will now make a secondary cluster node for the Bosch B6800 Virtual Receiver


WARNING!!!!

The current database on this server will be lost!!!

By making this server a secondary cluster node, the local database will be replaced with a
replicated version of the database on the primary cluster node.

Take care not to lose this database if this server was the last good running cluster node.

Ctrl-c to cancel or press any key to continue


==================================================================
******************************************************************

  YOU WILL LOSE ALL DATA ON THIS SERVER/NODE if you continue!!!!!

  YOU WILL LOSE ALL DATA ON THIS SERVER/NODE if you continue!!!!!

  YOU WILL LOSE ALL DATA ON THIS SERVER/NODE if you continue!!!!!

******************************************************************
==================================================================

ARE YOU SURE?   Ctrl-c to cancel or press any key to continue
```

The script for making the secondary node of the cluster will warn you that the database on this server will be wiped.

WARNING: Make sure you don’t lose your only good set of data!

When you create a secondary node, the database will be wiped, and replicated from the primary node. Be careful not to lose or overwrite your best or only surviving copy of the database.

The script will set up the environment on this installation to allow it to become a cluster secondary node. When the script finishes, the primary node should be setup.

Finally, the script will display that the setup is complete. **DON’T** press the final confirmation until the primary node is ready.

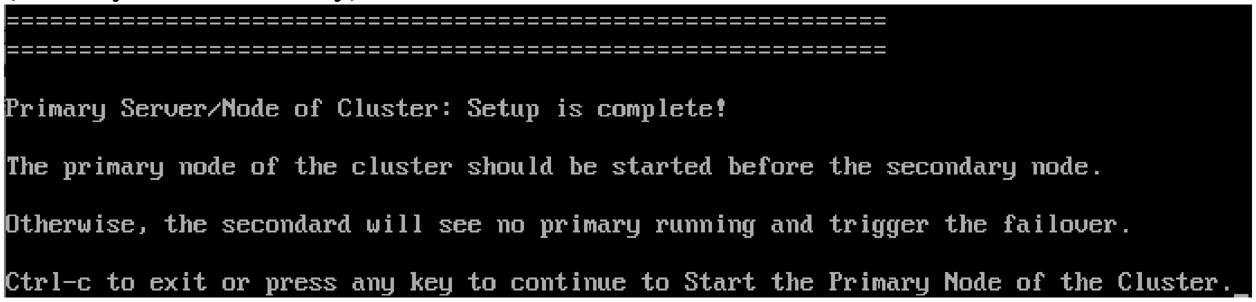
1.12.5 Starting the cluster

Both cluster nodes are now setup but aren’t yet working together as a failover cluster.

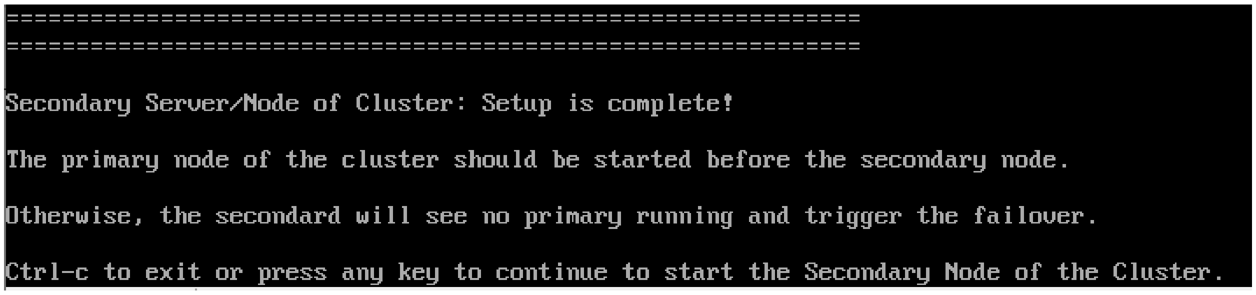
The order you start the cluster nodes is especially important!

You will need to start the Primary Server/Node first, and then a few seconds later start the Secondary Server/Node.

Once both servers are configured and setup, you will see the following screen on each (Primary and Secondary).



```
============================================================
============================================================

Primary Server/Node of Cluster: Setup is complete!

The primary node of the cluster should be started before the secondary node.

Otherwise, the secondard will see no primary running and trigger the failover.

Ctrl-c to exit or press any key to continue to Start the Primary Node of the Cluster._
```



```
============================================================
============================================================

Secondary Server/Node of Cluster: Setup is complete!

The primary node of the cluster should be started before the secondary node.

Otherwise, the secondard will see no primary running and trigger the failover.

Ctrl-c to exit or press any key to continue to start the Secondary Node of the Cluster.
```

To Start the cluster:

1. Press any key on the Primary system
2. Count to 5
3. Press any key on the Secondary system.

Again, it is crucial to note that you must start the cluster primary server before the secondary server. If you do not follow this ordering, the secondary server will not detect a working primary server and will instead trigger the failover to immediately happen.

The Primary will start and wait for the Secondary to come online before the B6800 will actual start working and allow event and automation traffic.

While the primary cluster node is waiting for the secondary to come online to establish the cluster, all and any event traffic will not be received and/or acknowledged.

Only once the cluster is up and running with both active nodes will it be able to start working.

That's it! You should now have a fully functioning B6800 Failover Cluster.

1.13 Accessing the B6800 In a “Failover Cluster: Mode

Don’t forget! The windows shortcuts on each server when you installed the B6800 only “point” to that Host’s own instance.

You likely will want to make some new shortcuts that now point to the Virtual IP address of the cluster, and not each server’s instance. You can also make shortcuts to the “DOMAIN” name that was specified during the installation.

http://<virtualip>/monitor
http://<virtualip>/manager

All your panels and automation software should also be configured to point to this new “virtual Ip” address or Domain Name. This is critical, as it is the moving virtual Ip that will enable the communications to “move” over to the failover node of the cluster should something happen to the primary system.

1.14 Running on a Failed Cluster after Failover Event

After a failover event, the cluster will be in a degraded state, operating on a single server or node. This can either be the original primary or the original secondary server, depending on the specific type of failover scenario.

It is of utmost importance to exercise caution during this phase, as the surviving server may hold the only remaining copy of the latest database.

For example, if the primary server’s network connection is unplugged, the software is still running on that host, it’s just that nothing can reach it. The system might even display it’s Web UIs if browsed to on the host windows system, even though the secondary node has taken over, and the virtual IP has shifted over to the secondary node. So, while the failed primary node/server may still have a working B6800 and a functional database, it is crucial to note that it is rapidly becoming outdated compared to the still operating and connected server. Plugging this primary server back into the network will not re-establish a cluster automatically, the two B6800 systems will need to be put back into a working cluster configuration by manual intervention.

After any failover scenario, the surviving node continues to receive events and interact with automation systems.

To ensure data consistency and minimize discrepancies, it is essential to carefully manage the failed system. Since the failed node is going to be out of sync with the active database, the VM of the failed node should be Shutdown from the Hyper-V Manager or the VM console.

1.15 How do I rebuild a Cluster after failover has occurred?

1.15.1 Overview

Every failover scenario is unique, and the appropriate actions will vary depending on the circumstances. Here are some considerations:

- Network Glitch or Disconnected Server: If the failover is due to a temporary network issue or a disconnected server that remains a viable cluster node, you may not need to replace or reinstall the server. Simply reconnecting it to the network and then re-establishing a running cluster should suffice.

- Catastrophic Failure and Host Replacement: In the case of a host experiencing a catastrophic failure, such as hardware damage, it may be necessary to replace and reinstall the system. This involves setting up a new host with the B6800 installed and licenses reapplied.

- Both Systems Failing: In rare cases, both systems in the cluster may fail simultaneously or before you can recover from the first failure event. In such situations, you will need to assess the extent of the failure and determine the best course of action. This could involve replacing both hosts and rebuilding the cluster from scratch.

    - Recovering from this “disaster” scenario will rely much more on backup assets to recover any exist configurations or event data. Strategies can include utilizing host Windows images, VM backup images, VM Image Exports, VM replication, database backups as well as numerous other varying approaches.

It is important to have a plan in place for “Disaster Recovery” before the event happens to minimize any downtime and impact of the event.

When dealing with a failed or missing system, some manual intervention will be required. Here's an overview of the steps involved:

- Choose the New Primary Node: Determine which system will become the new primary node. Typically, this will be the surviving node that is still operational or the server with the best and latest database.

- Bring the Failed Host Back Online: Repair or reconnect the failed host as needed. This may involve rebooting the host or resolving network issues. Note that the failed system will not automatically reconnect to the cluster.

1.15.2 Scenario: The Primary System/Node of the Cluster has failed

What do I do if the primary system fails for good? Can the surviving secondary server be incorporated into a new cluster?

Yes! If the primary system has died, the secondary server still has all your configuration and run-time data preserved.

You will need to carefully get your cluster rebuilt to resume a redundant configuration. Steps:

1. Leave the surviving secondary node running while you build the replacement system.

2. Build a new system to replace the failed system.

3. Install B6800 software on the new system (must be same version of B6800!!!)

4. Apply the B6800 License to the new system. This is critical. You don't want your new cluster to failover some day and find you haven't applied a license to your replacement secondary system!

5. No need to configure the new secondary B6800 in any way. It will fetch all the configuration information when it joins the new cluster.

6. Next, we need to prepare the new system to become the new secondary/backup node in the newly repaired cluster. (The surviving "old" secondary system will become the new primary node of the new cluster). On the new replacement system, run the failover script *makeclustersecondary.sh*. Let the script do all its setup to completion, but don't pass the last step just yet!!! Before we start the cluster, we will first need to convert the old surviving and still running secondary node to the new primary node.

7. Once you have your new secondary built, and setup, and primed and ready to become the new secondary, we can convert the old secondary to the new primary. To accomplish this, simply run the makeclusterprimary.sh script on the running secondary node.

    - Note: This will take the B6800 offline for a few minutes as the primary cluster node is rebuilt and reconfigured.

8. Once the primary node is setup and ready to start…. You simply press "any key" to start the primary node up, and then quickly after (as was done originally), press any key to start the new secondary node and have it joining the primary and establish the new cluster.

9. Very quickly (in a few seconds in most cases) the data and configuration will replicate to the new secondary node, and you will again have a redundant two node cluster.

1.15.3 Scenario: The Secondary Server/Node of the Cluster has failed

What if the secondary system is the one that fails? How do I rebuild my cluster in this case?

When the failover occurred, the primary cluster node will have recognized that the secondary node has failed or gone offline and degrade the cluster. It will begin functioning as a standalone B6800 after about 10 seconds after it can't see the secondary node.

To rebuild a working cluster, it isn't as simple as making a new secondary server and reintroducing it to the surviving primary node. In this simple sample configuration, the surviving primary server will not return to a primary cluster node on its own. The cluster must be re-established completely by rerunning the make cluster scripts.

Steps:

1. Leave the surviving primary node running while you build the replacement server. It is up and running and should be working in standalone mode.

2. Build a new system to replace the failed secondary server.

3. Install B6800 software on the system (must be same version of B6800!!!)

4. Apply the B6800 License to the new system. This is critical. You don't want your new cluster to failover some day and find you haven't applied a license to your secondary system!

5. No need to configure the new secondary B6800 in any way. It will fetch all the configuration information when it joins the new cluster.

6. Next, we need to prepare the new system to become the new secondary/backup node in the newly repaired cluster. The surviving “old” primary system, will again become the new primary node of the new cluster. On the new replacement server, run the failover script makeclustersecondary.sh. Let the script do all its setup to completion, but don’t pass the last step just yet!!! Before we start the cluster, we will first need to re-establish the cluster by converting the surviving system from a standalone to running as the primary node in the new cluster.

7. To accomplish this, simply run the makeclusterprimary.sh script on the still running (old primary) system. Note: This will take the B6800 offline for a few minutes as the primary cluster node is rebuilt and reconfigured.

8. Once the primary node is setup and ready to start.... You simply press “any key” to start the primary node up, and then quickly after, press any key to start the new secondary node and have it joining the primary and establish the new cluster.

9. Very quickly (in a few seconds in most cases) the data and configuration will replicate to the new secondary node, and you will again have a redundant two node cluster.

The "*makeclusterprimary.sh*" and "*makeclustersecondary.sh*" scripts can be run at any time to re-create a cluster.... You can run them interchangeably on each system to switch which system in your clustered pair of servers will run as the primary and the secondary.

The scripts are written in a way that to be able to modify the configuration of the VM from primary to secondary and then back again. This allows the scripts to be run to handle most any scenario of cluster rebuild after most scenarios of failure.

The scripts can even be run against a working cluster to swap the server’s role within the existing cluster. This is useful when you need to swap out a working but older piece of hardware with a newer version.

It's important to highlight the following:

- The "*makeclusterprimary.sh*" script will always preserve the underlying database, ensuring its continuity during failover.

- Running the "*makeclusterprimary.sh*" script will always take down the cluster for the duration of time it takes to bring the two cluster nodes back on together.

Re-establishing the cluster will likely require a short maintenance window between 3-10 minutes.

- Exercise caution with the "*makeclustersecondary.sh*" script, as it will always **clear or erase the database** on the secondary system. The secondary system becomes the database replication target and receives the same database as the new primary.

Disaster Recovery and Planning

Recovering from a failover cluster failover event involves more than just the reestablishment of the cluster itself. It requires a comprehensive strategy that includes various disaster recovery assets to ensure the recovery of the failed cluster and its data. Some key components of this strategy include database backups and server VM exports.

First and foremost, having regular database backups is crucial. Database backups capture the state of the data at a specific point in time, providing a reliable restore point in case of cluster failure. These backups can be used to restore the databases on the primary and secondary servers, ensuring that critical data is preserved and can be recovered efficiently.

In addition to database backups, server images and VM exports can play a vital role in the recovery process. Server VM exports capture the entire state of the virtual machines, including the operating system, applications, and configurations. By exporting the server VMs, you create a snapshot that can be used to restore the entire cluster environment, including all the necessary software components and settings. This approach simplifies the recovery process by providing a consistent and comprehensive recovery point for the entire cluster.

By leveraging a combination of database backups, server VM exports, and other disaster recovery assets, organizations can ensure a comprehensive recovery strategy for failover cluster events. This approach minimizes data loss, reduces downtime, and helps restore the cluster to its previous state efficiently and effectively. It is important to regularly test and update the disaster recovery plan to account for any changes in the cluster configuration or infrastructure to maintain a robust and reliable recovery process.

Summary

The failover scripts included with the B6800 serve as a UL validated failover configuration. As the B6800 expands in future releases, this process may change to meet additional needs or improvements.

FAQs

Can I turn a standalone B6800 into a Failover Cluster Configuration?

- Yes. A standalone B6800 can become the Primary Server in a B6800 Failover Cluster. All existing data and configuration will be preserved for as then Primary Node/Server of a cluster.

While I am creating the Failover Cluster, is the B6800 offline?

- Yes, the act of creating the cluster and replicated databases will take a running B6800 offline. The cluster creation usually only takes a few minutes using the scripts provided, but nevertheless, there will be a short maintenance window where the B6800 will be offline.

Can I go back to standalone B6800 after establishing a failover cluster?

- Not currently. The provided scripts do not support this pathway. Make a backup image of your host server or export the VM image with a known working configuration to have a “cold-standby” B6800 on the ready.

How long does it take to failover?

- If the network is disconnected, or the primary node fails, the failover occurs in 15-20 seconds on the minimum recommended hardware. The UI may take a bit longer to update and show activity, but the receiver is up and able to accept traffic in under 30 seconds.

- Is there ever a loss of data or events during failover?
  - No, there should be no loss of events. Before failover, all event data is replicated to the backup node immediately before responding to panels or automation. This “commit twice” design guarantees that the data is saved on both cluster nodes. Then during failover, the surviving server simple starts adding new event data to the surviving database. There should be no loss of data, and only a short interruption of communication during the failover event itself.

Can the two servers’ “boot” into the working cluster?

- No. A cluster must be established in precise order. If the secondary server comes online before the primary server, it likely would interpret that as the “primary is down” and assume control of the cluster. The primary node must be started first, and the secondary node soon after to establish the working cluster. Always just rerun the *makeclusterprimary.sh* and *makeclustersecondary.sh* scripts to restart a cluster.

Bosch Security Systems B.V. Torenallee 49

5617 BA Eindhoven

Netherlands

www.boschsecurity.com

© Bosch Security Systems B.V., 2024