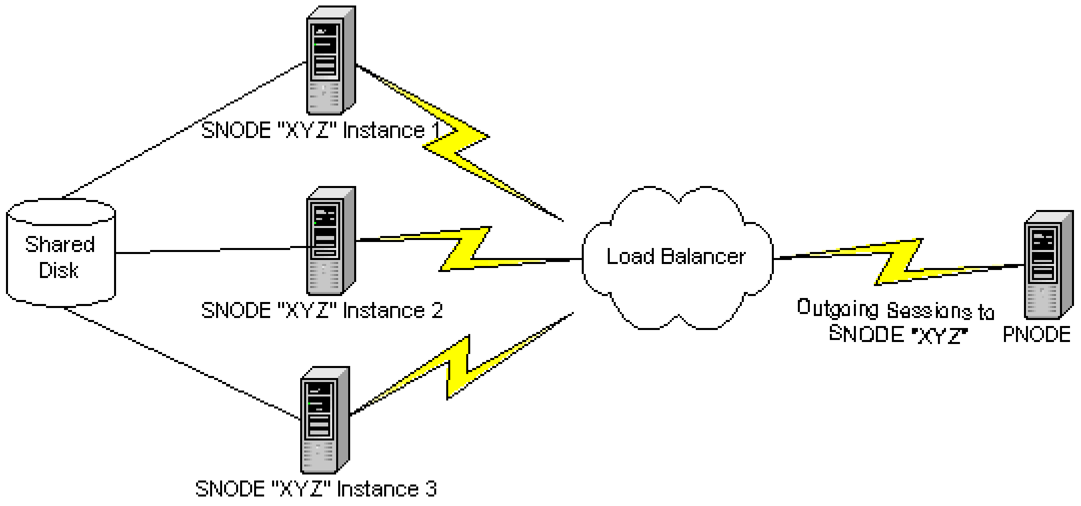
Considerations for Running Connect:Direct for UNIX Behind a Load Balancer - Updated

Date: November 17, 2020

Table of Contents

1. Introduction

Customers may wish to use a connection load balancer to distribute incoming Snode sessions across multiple instances of Connect:Direct for UNIX. In such an arrangement the Connect:Direct for UNIX instances behind the load balancer appear as a single Snode to the outside world. A given incoming session is distributed to one of the instances based on criteria defined in the load balancer. If an incoming session fails and is restarted by the remote PNODE, then the restarted session may be assigned to any of the instances behind the load balancer and will not necessarily be established with the original SNODE instance. Generally, from the point of view of the nodes behind the load balancer only incoming or "SNODE" sessions are affected by the load balancer; PNODE, or outgoing sessions operate the same way they normally do. Connect:Direct for UNIX includes an enhancement that allows COPY checkpoint/restart and RUN TASK resynchronization to work when Connect:Direct is set up in this way.



This document points out some extra considerations that come into play only when instances of Connect:Direct for UNIX are configured to operate behind a connection load balancer. These considerations can be categorized as:

- SNODE server characteristics
- SNODE Connect:Direct node setup
- Shared SNODE work area setup
- "Stranded" SNODE TCQ entries
- RUN TASK RESTART parameter
- Process Statistics
- Remote Snode alternate.comminfo parameter

2. SNODE Server Considerations

- The servers used for the Connect:Direct for UNIX instances behind the load balancer must all have access to a common shared cluster disk storage since any COPY statement source and destination files for SNODE processes must reside in directories accessible to all of the servers. All nodes must have access to a common SNODE work area. If you want to use NFS for the shared drive, the NFS version must be **v4** or greater.

- The system clocks on all the servers must be synchronized for COPY checkpoint/restart and RUN TASK synchronization to work.

- The administrator user ID used to install Connect:Direct for UNIX must be defined the same on each server and must be the same user and group number on each server.

- The servers should all be of the same hardware platform type and running the same Operating System.

3. SNODE Connect:Direct Node setup

- One Connect:Direct for UNIX node should be installed on each server behind the load balancer.
- Each node must be the same Connect:Direct for UNIX version and maintenance level.
- Each node must be installed by the same user ID.
- Each node must have the same C:D node name.
- Each node must have the same node-to-node connection listening port.
- Each node must specify the same path for the snode.work.path attribute of the ndm.path initialization parameter in the initialization parameter file.
- Each node must specify its local host name in the tcp.api listening address specification of the netmap local node entry.

4. Shared SNODE work area setup

A directory should be established for the shared SNODE work area used by the Connect:Direct for UNIX nodes behind the load balancer. The path to this directory must be specified in the snode.work.path attribute of the initialization parameter file for each instance. Following is an example for this parameter.

```
# Miscellaneous Parameters
ndm.path:path=/<local_install_path>/cdunix:\
        :snode.work.path=/<shared_disk_mount_point>/cdunix/shared:
```

SNODE return code files (steprc files) and COPY checkpoint information are created and stored in this area when the snode.work.path attribute is specified in the initialization parameters.

This directory should be owned by the Connect:Direct administrator ID and must be accessible to all of the servers behind the load balancer. It must be on a cluster file system. If you want to use NFS for the shared drive, you must use NFS v4 or greater.
The initial size of the shared work area is dependent upon the number of Connect:Direct for UNIX servers behind the load balancer and the total workload on the servers. You can start with 50Mb, or more. You will need to monitor the usage of this disk space from time to time and be prepared to increase the size if necessary.

5. “Stranded” SNODE TCQ entries

As mentioned above, when Connect:Direct instances are setup behind a load balancer and an incoming session fails and is restarted by the remote PNODE, then the restarted session may be assigned to any of the SNODE instances behind the load balancer and will not necessarily be established with the original SNODE instance. In this scenario, each SNODE instance that receives a session for a given process creates a TCQ entry for the process. (Note that each SNODE instance has its own TCQ file; these are not shared among SNODE instances. Only the work files created in the shared work area are shared among instances.)

Consider this scenario based on the drawing depicted in the introduction. If a process P is submitted on the remote PNODE, a session may be established with SNODE instance 1, which creates an SNODE TCQ entry in its TCQ file. If process execution is interrupted and the process is requeued and restarted, the restart session may be established with SNODE instance 2, which also creates an SNODE TCQ entry in its TCQ file. If the process runs to completion on the restart session with SNODE instance 2, then that instance deletes its TCQ entry for process P and also deletes any SNODE work files for process P from the shared SNODE work area. However, SNODE instance 1 still retains its TCQ entry for process P. This TCQ entry is “stranded” since the process has completed and the work files have been deleted.

Stranded SNODE TCQ entries are checked automatically when Connect:Direct for UNIX is initialized and also when the TCQ is scanned periodically during product execution. The stranded SNODE TCQ entries will be deleted according to the parameter value set for ‘ckpt.max.age’ in the “TCQ Information” section of the ‘initparm.cfg’ file. The default value is 8 days before automatic deletion occurs. No administrator action is required.

6. Run Task Restart parameter

The Run Task Restart parameter works slightly differently when Connect:Direct for UNIX is configured with shared SNODE work areas.

Ordinarily when process execution is interrupted during a Run Task step and subsequently restarted, then the setting of the Run Task Restart parameter determines whether or not the task is restarted only if it is determined that the task is no longer active. Connect:Direct for UNIX determines whether or not the task is still active by checking if the task system process is still active on the server.

When shared SNODE work areas are configured and the Run Task is on the SNODE, then at restart time it is generally not possible for Connect:Direct for UNIX to determine whether the original task is still active or not since the restart session may be with a different SNODE instance on a different server machine.

Therefore, in this scenario, the task is either restarted or not restarted based solely on the setting of the Run Task Restart parameter. Because of this, caution should be used when specifying Run Task Restart = Yes. It is possible that a task could be restarted even though it may be active on another server machine.

7. SNODE Process Statistics

Because of the fact that statistics files are not shared among the SNODE instances behind the load balancer, when a process is interrupted and restarted to a different SNODE instance, the statistics records for that process will be distributed between the two SNODE instances involved. Currently there is no provision for selecting all the statistics records for a given process in a single operation when the records are distributed across multiple SNODE instances.

8. PNODE to SNODE Considerations.

When the Connect:Direct for UNIX servers behind the load balancer are acting as Pnodes sending to an Snode, the remote Snode needs to have the Netmap record for the Pnodes configured with the “alternate.comminfo” parameter. This will allow Netmap checking to work successfully on the remote Snode.

CDU Example:

```
XYZ:\
   :comm.info=9.1.2.2;1364:\    <<This is the IP address of the load balancer>>
   :alternate.comminfo=9.1.2.3, 9.1.2.4, 9.1.2.5:\  <<These are the IP addresses
                                                      of the individual nodes
                                                      behind the load balancer>>
```

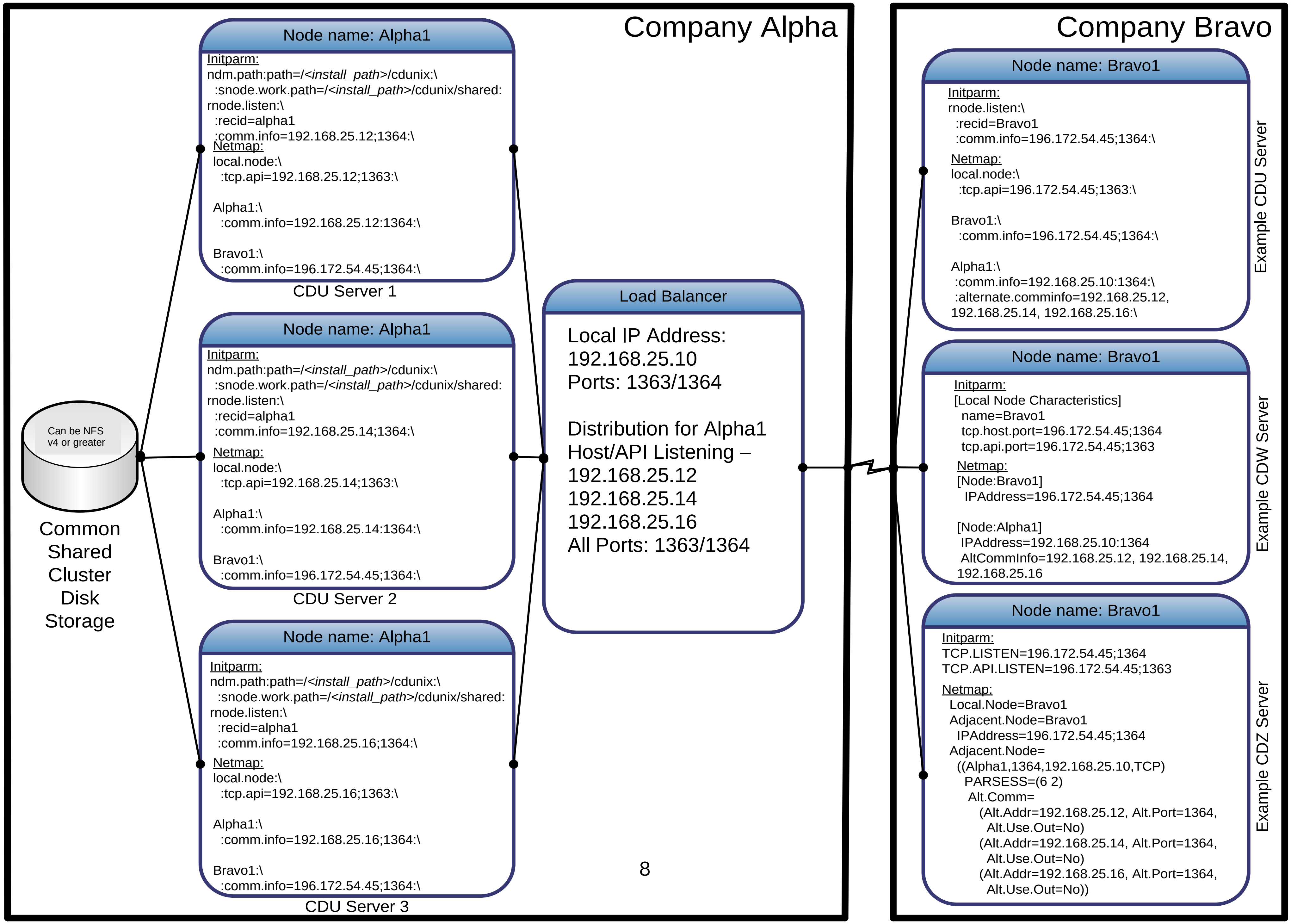
Support for distributing API connections through a load balancer is limited to being used for submit process commands without a maxdelay specification. When a process is submitted with maxdelay, the connection is idle until the process is completed. Load Balancers tend to kill idle connections. A “Best Practice” for submitting a long running process via a Load Balancer distributed API connection is to submit the process without maxdelay, keep the API connection open, and periodically poll the server for process results.
For example, a C:D File Agent operating outside of the CDU LB cluster could have its connections distributed via the load balancer. Query and configuration type commands should be submitted directly to each Connect:Direct for UNIX server via its own API Listening port.

9. FASP Considerations.

CDU uses both TCP and UDP for FASP connections. When the CDU environment is the Snode, destination address affinity persistence, also known as ‘sticky persistence’, needs to be enabled in the load balancer. DAAP supports both TCP and UDP protocols. This will direct session requests to the same server based solely on the destination IP address of a packet.
Consult the following document for more details on using FASP behind a load balancer.
Using a Load Balancer with FASP for Connect:Direct for UNIX

10. Example Illustrations.

The first is an example of the logical representation of the connectivity of the load balancer environment. This illustration contains examples of the configuration settings used to enable the connectivity between the two representative companies.
This is an example only – your environment may be different.
The second is an example of the physical connectivity that one might use to setup this environment.
This is an example only – your environment may be different.



CDU Server 1 (Node name: Alpha1)

```
Initparm:
ndm.path:path=/<install_path>/cdunix:\
  :snode.work.path=/<install_path>/cdunix/shared:
rnode.listen:\
  :recid=alpha1
  :comm.info=192.168.25.12;1364:\
Netmap:
local.node:\
  :tcp.api=192.168.25.12;1363:\

Alpha1:\
  :comm.info=192.168.25.12:1364:\

Bravo1:\
  :comm.info=196.172.54.45;1364:\
```

CDU Server 2 (Node name: Alpha1)

```
Initparm:
ndm.path:path=/<install_path>/cdunix:\
  :snode.work.path=/<install_path>/cdunix/shared:
rnode.listen:\
  :recid=alpha1
  :comm.info=192.168.25.14;1364:\
Netmap:
local.node:\
  :tcp.api=192.168.25.14;1363:\

Alpha1:\
  :comm.info=192.168.25.14:1364:\

Bravo1:\
  :comm.info=196.172.54.45;1364:\
```

CDU Server 3 (Node name: Alpha1)

```
Initparm:
ndm.path:path=/<install_path>/cdunix:\
  :snode.work.path=/<install_path>/cdunix/shared:
rnode.listen:\
  :recid=alpha1
  :comm.info=192.168.25.16;1364:\
Netmap:
local.node:\
  :tcp.api=192.168.25.16;1363:\

Alpha1:\
  :comm.info=192.168.25.16;1364:\

Bravo1:\
  :comm.info=196.172.54.45;1364:\
```

Load Balancer

Local IP Address:
192.168.25.10
Ports: 1363/1364

Distribution for Alpha1
Host/API Listening –
192.168.25.12
192.168.25.14
192.168.25.16
All Ports: 1363/1364

Example CDU Server (Node name: Bravo1)

```
Initparm:
rnode.listen:\
  :recid=Bravo1
  :comm.info=196.172.54.45;1364:\
Netmap:
local.node:\
  :tcp.api=196.172.54.45;1363:\

Bravo1:\
  :comm.info=196.172.54.45;1364:\

Alpha1:\
 :comm.info=192.168.25.10:1364:\
 :alternate.comminfo=192.168.25.12,
192.168.25.14, 192.168.25.16:\
```

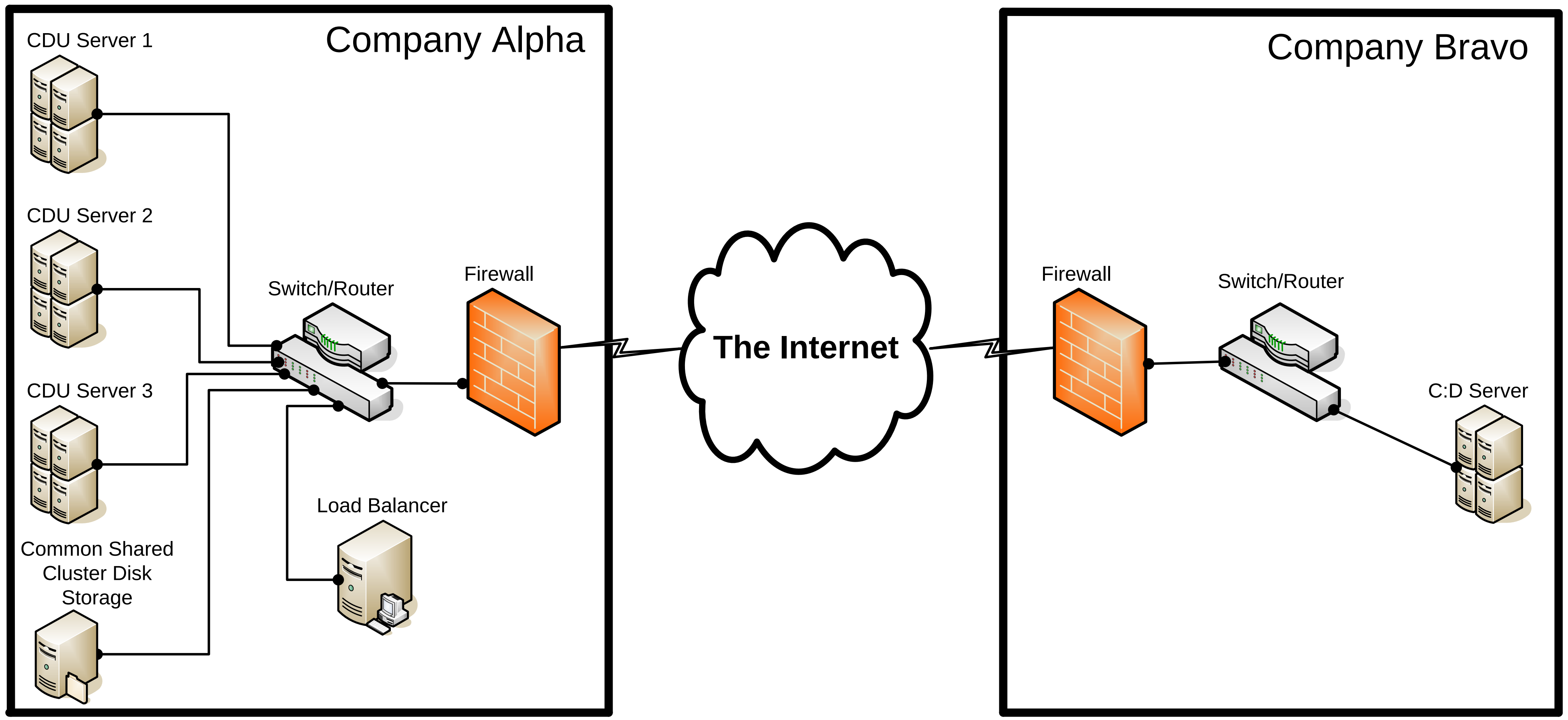
Example CDW Server (Node name: Bravo1)

```
Initparm:
[Local Node Characteristics]
  name=Bravo1
  tcp.host.port=196.172.54.45;1364
  tcp.api.port=196.172.54.45;1363
Netmap:
[Node:Bravo1]
  IPAddress=196.172.54.45;1364

[Node:Alpha1]
 IPAddress=192.168.25.10:1364
 AltCommInfo=192.168.25.12, 192.168.25.14,
192.168.25.16
```

Example CDZ Server (Node name: Bravo1)

```
Initparm:
TCP.LISTEN=196.172.54.45;1364
TCP.API.LISTEN=196.172.54.45;1363
Netmap:
 Local.Node=Bravo1
 Adjacent.Node=Bravo1
   IPAddress=196.172.54.45;1364
 Adjacent.Node=
   ((Alpha1,1364,192.168.25.10,TCP)
     PARSESS=(6 2)
      Alt.Comm=
         (Alt.Addr=192.168.25.12, Alt.Port=1364,
           Alt.Use.Out=No)
         (Alt.Addr=192.168.25.14, Alt.Port=1364,
           Alt.Use.Out=No)
         (Alt.Addr=192.168.25.16, Alt.Port=1364,
           Alt.Use.Out=No))
```


Company Alpha
CDU Server 1
CDU Server 2
CDU Server 3
Common Shared Cluster Disk Storage
Switch/Router
Firewall
Load Balancer
The Internet
Company Bravo
Firewall
Switch/Router
C:D Server