Comprehensive Guide to Clustering and High Availability in Liferay 7.4

Infrastructure Architecture Blueprint

November 6, 2026

Abstract

This document presents an architectural blueprint, infrastructure prerequisites, example configuration snippets, and validation procedures for deploying multiple Liferay 7.4 application nodes. High availability depends on removing single points of failure in the load balancer, database, document storage, search tier, and application services. Session failover and cache synchronization are distinct concerns and must be validated against the exact Liferay distribution and update level.

Contents

1 Architectural Topology **2**
- 1.1 Infrastructure Component Matrix . . . 2
- 1.2 Example Network Plan . . . 2

2 Global Properties Setup (`portal-ext.properties`) **2**
- 2.1 Decoupled Media and Document Repository . . . 2

3 Advanced Inter-Node Communication via JGroups **3**

4 Application Server Session Replication (Tomcat Stack) **3**

5 Deployment Protocol and Verification Steps **4**
- 5.1 Sequential Runtime Startup Protocol . . . 4
- 5.2 Network Cluster Verification . . . 4
- 5.3 Functional and Failover Tests . . . 5

6 Troubleshooting Checklist **5**

7 Production Readiness and Security **5**

8 Conclusion **5**

1 Architectural Topology

A typical deployment places a load balancer in front of two or more Liferay nodes, which use appropriately shared or highly available backend services. Application nodes should avoid node-local persistent business data, but they are not automatically stateless: HTTP sessions, caches, and background jobs may still require explicit design.

1.1 Infrastructure Component Matrix

Table 1: Decoupled Infrastructure Prerequisites

Component	Clustered deployment requirement
Load Balancer	Layer 7 proxy (e.g., HAProxy, NGINX, AWS ALB), health checks, TLS handling, and usually sticky sessions. Make the load-balancing tier redundant.
Application Nodes	Uniform, compatible Liferay 7.4 builds, modules, configurations, and JVM settings; avoid node-local document persistence.
Database Engine	A shared supported relational database, preferably with a tested HA/failover design and properly sized connection pools.
Document Store	A supported shared document library store, such as correctly configured shared filesystem storage or a supported object-storage integration.
Search Engine	An external search deployment compatible with the exact Liferay 7.4 release; verify the supported Elasticsearch/OpenSearch version and topology.

1.2 Example Network Plan

Role	Example address	Purpose
Load balancer	`10.0.1.5`	Public traffic entry
Liferay node 1	`10.0.1.10`	Application runtime
Liferay node 2	`10.0.1.11`	Application runtime
Database	`10.0.2.10`	Shared relational data
Search	`10.0.2.20`	External search service

2 Global Properties Setup (`portal-ext.properties`)

The following is an *illustrative* cluster-link configuration. Verify the actual property keys, channel naming, and supported transport stack for your precise Liferay CE or DXP build before deploying. Place supported properties in `${LIFERAY_HOME}/portal-ext.properties` or use the appropriate configuration mechanism.

```
# Cluster-link settings vary by distribution and update level.
cluster.link.enabled=true
cluster.link.bind.addr.autodetect=false

# Only use these channel properties if supported by your build.
cluster.link.channel.properties.control=tcp.xml
cluster.link.channel.properties.transport.0=tcp.xml
```

Listing 1: Illustrative cluster-link settings; verify against installed version

2.1 Decoupled Media and Document Repository

Each node must access the same logical document library. A local filesystem on each node is not an acceptable shared document repository. The store implementation and configuration keys

are version-specific; the example below is retained for reference, *not* as a guaranteed Liferay 7.4 drop-in configuration.

```
# Example only: confirm store implementation class and root property.
dl.store.impl=com.liferay.portal.store.file.system.AdvancedFileSystemStore
dl.store.file.system.root.dir=/mnt/shared/liferay/data/
```

Listing 2: Illustrative filesystem store settings; verify implementation and keys

Check permissions, mount consistency, atomic operations, latency, backup policy, and supported store-provider configuration. An object store such as S3 requires its own supported integration rather than simply a filesystem mount.

3 Advanced Inter-Node Communication via JGroups

Cluster-link configuration is not interchangeable with HTTP session replication. The following XML illustrates a TCP transport with static discovery. The correct JGroups protocol names, schema, and stack ordering depend on the JGroups version bundled with the installed Liferay release. Do not replace a vendor-provided stack without validating it.

```
<config xmlns="urn:org:jgroups"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <!-- Example only: verify protocol availability and stack order. -->
    <TCP bind_addr="10.0.1.10"
         bind_port="7800"
         recv_buf_size="${tcp.recv_buf_size:5M}"
         send_buf_size="${tcp.send_buf_size:5M}"
         max_bundle_size="64K"
         sock_conn_timeout="300"/>
    <TCPPING initial_hosts="10.0.1.10[7800],10.0.1.11[7800]"
             port_range="0"/>
    <MERGE3 min_interval="10000" max_interval="30000"/>
    <FD_SOCK/>
    <FD_ALL timeout="9000" interval="3000"/>
    <VERIFY_SUSPECT timeout="1500"/>
    <pbcast.NAKACK2 use_mcast_xmit="false"/>
    <UNICAST3/>
    <pbcast.STABLE/>
    <pbcast.GMS/>
    <MFC max_credits="2M" min_threshold="0.4"/>
    <FRAG2 frag_size="60K"/>
</config>
```

Listing 3: Illustrative JGroups TCP/TCPPING configuration

Note: Each node needs an appropriate bind address. Ensure firewall rules permit the configured cluster ports between nodes. Static discovery addresses must be kept current; in orchestrated environments use a supported discovery mechanism.

4 Application Server Session Replication (Tomcat Stack)

Sticky sessions reduce cross-node session movement, but do not themselves provide session failover. Tomcat replication is an optional, separately validated design decision and may not be supported or recommended for every Liferay 7.4 topology. The sample below uses multicast membership and therefore will not work on networks that block multicast. It is *not* a production-ready recommendation.

```
<Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster"
         channelSendOptions="8">
```

```
    <Manager className="org.apache.catalina.ha.session.DeltaManager"
             expireSessionsOnShutdown="false"
             notifyListenersOnReplication="true"/>
    <Channel className="org.apache.catalina.tribes.group.GroupChannel">
        <Membership className="org.apache.catalina.tribes.membership.McastService"
                    address="228.0.0.4" port="45564"
                    frequency="500" dropTime="3000"/>
        <Receiver className="org.apache.catalina.tribes.transport.nio.NioReceiver"
                  address="10.0.1.10" port="4000"
                  autoBind="100" selectorTimeout="5000" maxThreads="6"/>
        <Sender className="org.apache.catalina.tribes.transport.ReplicationTransmitter">
            <Transport className="org.apache.catalina.tribes.transport.nio.PooledParallelSender
    "/>
        </Sender>
        <Interceptor className="org.apache.catalina.tribes.group.interceptors.
    TcpFailureDetector"/>
        <Interceptor className="org.apache.catalina.tribes.group.interceptors.
    MessageDispatchInterceptor"/>
    </Channel>
    <Valve className="org.apache.catalina.ha.tcp.ReplicationValve"/>
    <Valve className="org.apache.catalina.ha.session.JvmRouteBinderValve"/>
    <ClusterListener className="org.apache.catalina.ha.session.ClusterSessionListener"/>
</Cluster>
```

Listing 4: Illustrative Tomcat DeltaManager configuration

If the chosen Tomcat version and Liferay distribution support this approach, consult their documentation for the correct `server.xml` placement, distributable web application requirements, session serialization, and security. The XML belongs within the appropriate Tomcat `<Host>` configuration only when that integration is supported.

5 Deployment Protocol and Verification Steps

5.1 Sequential Runtime Startup Protocol

1. **Prepare shared services:** Confirm database readiness, search compatibility, document store access, backups, network connectivity, and matching application versions.

2. **Node 1 baseline:** Start one node and verify database access, deployed modules, search connectivity, and document operations. Run schema upgrades only through the supported upgrade process.

3. **Node 2 expansion:** Start the second node with the same release and compatible configuration. Inspect logs for startup, cluster membership, and cache errors.

4. **Load balancer:** Enable both nodes behind readiness-aware health checks and validate TLS/forwarded headers and session affinity.

5.2 Network Cluster Verification

Depending on the exact release, clustering configuration and status may be exposed in the Liferay control panel or require logs/management interfaces. A possible navigation path is:

Control Panel → System Settings → Infrastructure → Cluster Executor

Do not assume this menu exists in every Community Edition or DXP build. Verify cluster membership through the supported management interface and node logs.

5.3 Functional and Failover Tests

1. Confirm requests reach both application nodes under normal load.
2. Upload a document on node 1 and retrieve it through node 2.
3. Modify representative portal content and confirm visibility across nodes.
4. Verify search indexing and query behavior after content changes.
5. Test node shutdown and unexpected termination separately; document whether logged-in users must reauthenticate.
6. Validate database and search service failover independently; application clustering alone does not provide backend HA.
7. Observe job scheduling, background processing, message queues, and logs for duplicate or missed work.

6 Troubleshooting Checklist

Symptom	Investigate
Nodes cannot discover peers	Bind addresses, TCP ports, firewalls, DNS/static discovery list, and JGroups stack compatibility.
Intermittent logout	Sticky-session configuration, cookie domain, session serialization, and actual failover design.
Missing documents	Store implementation, common root, permissions, mount consistency, and object-store configuration.
Stale content or search	Cluster events, cache invalidation, indexer status, and search connectivity.
Repeated startup failures	Release/module mismatches, database schema state, connection limits, and upgrade logs.

7 Production Readiness and Security

Use private networks for database, search, and inter-node traffic; enforce least privilege, encrypted connections where supported, secret management, centralized logs, metrics, backups, and tested restore procedures. Design and test redundant load balancers, database failover, document-store durability, and search recovery. Review supported clustering features for the exact CE or DXP version before enabling example properties.

8 Conclusion

A resilient Liferay 7.4 architecture combines multiple application nodes with highly available shared services and tested operational procedures. Sticky sessions, inter-node messaging, shared document storage, and session failover solve different problems and must be validated separately. Treat all sample configuration snippets in this guide as architecture illustrations until matched against the installed release's documentation and tested in staging.