Liferay DXP Clustering

Liferay DXP can serve everything from the smallest to the largest web sites. Out of the box, it's configured optimally for a single server environment. If one server isn't sufficient to serve the high traffic needs of your site, Liferay DXP scales to the size you need.

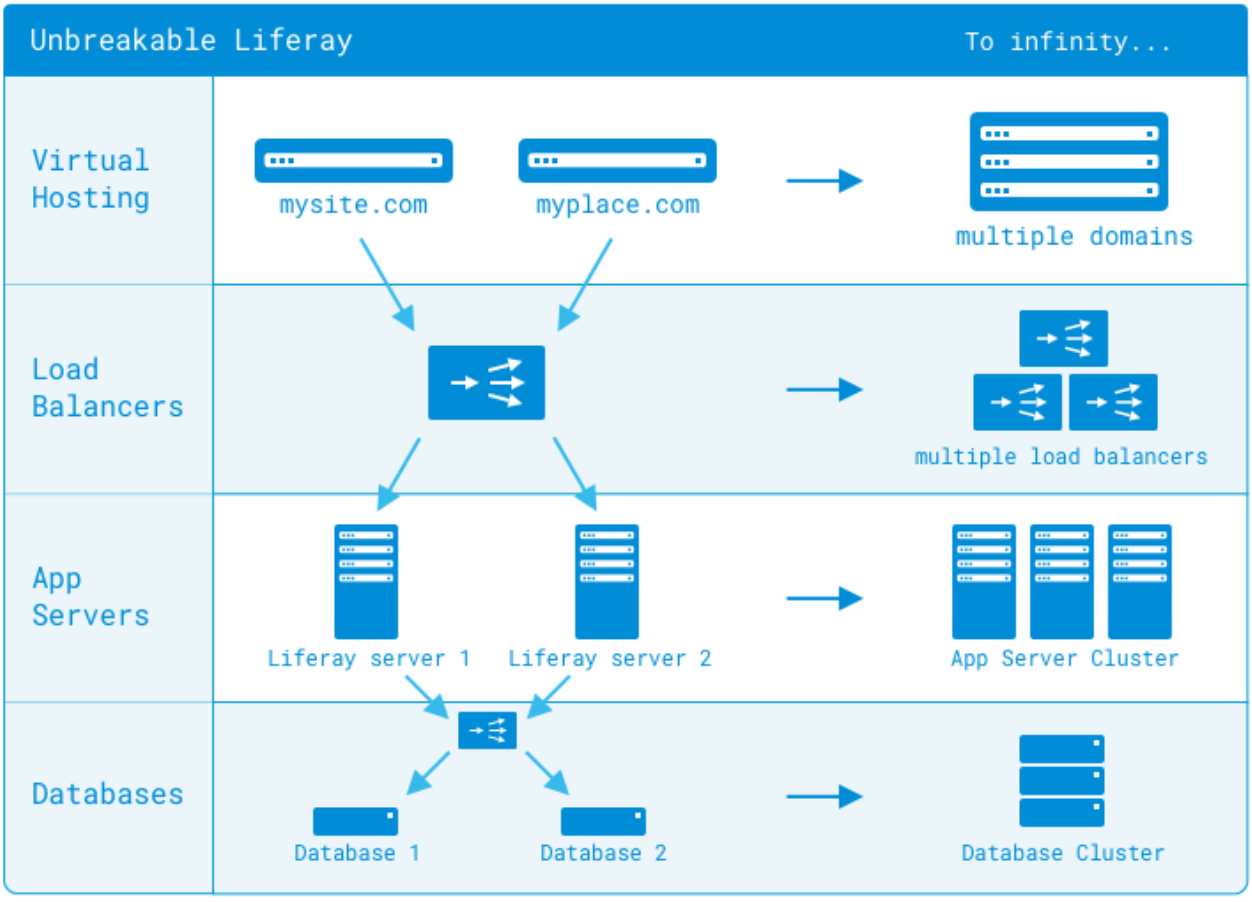


Figure 1: Liferay DXP is designed to scale to as large an installation as you need.

one application server node, there are several optimizations that need to be made. At a minimum, Liferay DXP should be configured in the following way for a clustered environment:

1. All nodes should be pointing to the same Liferay DXP database or database cluster.
2. Documents and Media repositories must have the same configuration and be accessible to all nodes of the cluster.
3. Search should be on a separate search server that is optionally clustered.
4. Cluster Link must be enabled so the cache replicates across all nodes of the cluster.
5. Hot deploy applications to each node individually.

If you haven't configured your application server to use farms for deployment, the hot deploy folder should be a separate folder for all the nodes, and plugins will have to be deployed to all of the nodes individually. This can be done via a script. If you do have farms configured, you can deploy normally to any node's deploy folder, and your farm configuration should take care of syncing the deployment to all nodes.

Many of these configuration changes can be made by adding or modifying properties in your `portal-ext.properties` file. Remember that this file overrides the defaults in the `portal.properties` file. You can also browse its contents here. It's a best practice to copy the relevant section you want to modify from `portal.properties` into your `portal-ext.properties` file, and then modify the values there.

Note: This article documents a Liferay DXP-specific cluster configuration without getting into specific implementations of third party software, such as Java EE application servers, HTTP servers, and load balancers. Please consult your documentation for those components of your cluster to configure those components. Before creating a Liferay DXP cluster, make sure your OS is not defining the hostname of your box to the local network at 127.0.0.1.

Each step defined above is covered below to give you a step by step process for creating your cluster.

1. All Nodes Should Point to the Same Liferay DXP Database

Each node should have a data source that points to one Liferay DXP database (or a database cluster) that all the nodes will share. This means, of course, Liferay DXP cannot (and should not) use the embedded HSQL database that is shipped with the bundles (but you already knew that, right?). And, of course, the database server should be on a separate system from the server which is running Liferay DXP.

READ-WRITER DATABASE CONFIGURATION

Liferay DXP allows you to use two different data sources for reading and writing. This enables you to split your database infrastructure into two sets: one optimized for reading and one optimized for writing. Since all Liferay DXP's supported databases support replication, you can use your database vendor's replication mechanism to keep the database nodes in sync.

Enabling a read-writer database is simple. In your `portal-ext.properties` file:

1. Set the default database connection pool provider to `dbcp`, `tomcat`, or `c3po`. Note, provider HikariCP does not support read/write splitting. Here's an example setting:

   ```
   jdbc.default.liferay.pool.provider=dbcp
   ```

   All the portal JDBC configuration properties are documented here.

2. Configure two different data sources for Liferay DXP to use, one for reading, and one for writing:

   ```
   jdbc.read.driverClassName=com.mysql.jdbc.Driver
   jdbc.read.url=jdbc:mysql://dbread.com/lportal?useUnicode=true&characterEncoding=U
   jdbc.read.username=**your user name**
   jdbc.read.password=**your password**

   jdbc.write.driverClassName=com.mysql.jdbc.Driver
   jdbc.write.url=jdbc:mysql://dbreadwrite.com/lportal?useUnicode=true&characterEnco
   jdbc.write.username=**your user name**
   jdbc.write.password=**your password**
   ```

   To use the JNDI instead of the JDBC data sources, set the `*.username` and `*.password` properties above to your JNDI user name and password and set these additional properties:

   ```
   jdbc.read.jndi.name=**your read JNDI name**
   jdbc.write.jndi.name=**your read-write JNDI name**
   ```

3. Avoid using the `default` data source, by setting this:

And if you're using a `dbcp` or `tomcat` database connection pool provider, set these:

```
jdbc.default.validationQuery=
jdbc.read.validationQuery=SELECT releaseId FROM Release_
jdbc.write.validationQuery=SELECT releaseId FROM Release_
```

These settings are related to issue LPS-64624.

4. Enable the read-writer database configuration by uncommenting the following Spring configuration files from the `spring.configs` and `spring.infrastructure.configs` properties:

```
spring.configs=\
    [..]
    META-INF/dynamic-data-source-spring.xml,\
    [..]

spring.infrastructure.configs=\
    [..]
    META-INF/dynamic-data-source-infrastructure-spring.xml,\
    [..]
```

The Spring configuration portal properties are documented here.

The next time you start Liferay DXP, it uses the two data sources you have defined. Be sure you have correctly set up your two databases for replication before starting Liferay DXP.

2. Documents and Media Library Clustering

Liferay DXP's Documents and Media Library can mount several repositories at a time while presenting a unified interface to the user. By default, users can use the Liferay DXP repository, which is already mounted. This repository is built into Liferay DXP and can use one of several different store implementations as its back-end. In addition to this, users can mount many different kinds of third party repositories. In a cluster, Documents and Media must have the exact same configuration on all nodes. If you have a separate repository you've mounted, all nodes of the cluster must point to this repository. Your avenue for improving performance at this point is to cluster your third party repository, using the documentation for the repository you have chosen. If you don't have a third party repository, you can configure the Liferay DXP repository to perform well in a clustered configuration.

Note that the file systems used by the `File System` or `Advanced File System` stores must support concurrent requests and file locking.

Checkpoint: To test if the sharing works well, execute the following steps:

1. On Node 1 upload a document to the Documents and Media.
2. On Node 2 download the document. The download should be successful.
3. Repeat the test with reversed roles.

3. Clustering Search

Search should always run on a separate environment from your Liferay DXP server. Liferay DXP supports Elasticsearch or Solr, and either of those environments can also be clustered.

For more information on how to cluster Elasticsearch, see Elasticsearch's distributed cluster setup.

Once Liferay DXP servers have been properly configured as a cluster and the same for Elasticsearch, change Liferay DXP from *embedded* mode to *remote* mode. On the first connection, the two sets of clustered servers communicate with each other the list of all IP addresses; in case of a node going down, the proper failover protocols will enable. Queries and indices can continue to be sent for all nodes.

For more information on how to cluster Solr, see Apache Solr Cloud documentation.

Once Liferay DXP servers have been properly configured as a cluster, deploy the Liferay Solr 5 Adapter on all nodes. (This app is available for download from Liferay Marketplace here.) Create a Solr Cloud (cluster) managed by *Apache Solr Zookeeper*. Connect the Liferay DXP cluster to Zookeeper and finish the final configurations to connect the two clusters.

4. Enable Cluster Link

Enabling Cluster Link automatically activates distributed caching. Distributed caching enables some RMI (Remote Method Invocation) cache listeners that are designed to replicate the cache across a cluster. Cluster Link uses Ehcache, which has robust distributed caching support. The cache is distributed across multiple Liferay DXP nodes running concurrently. The Ehcache global settings are in the `portal.properties` file.

By default Liferay does not copy cached entities between nodes. If an entity is deleted or changed, for example, Cluster Link sends an *remove* message to the other nodes to invalidate this entity in their local cache. Requesting that entity on another node results in a cache *miss*; the entity is then retrieved from the database

miss. The miss triggers the node to retrieve the entity from the database and store it in its local cache.

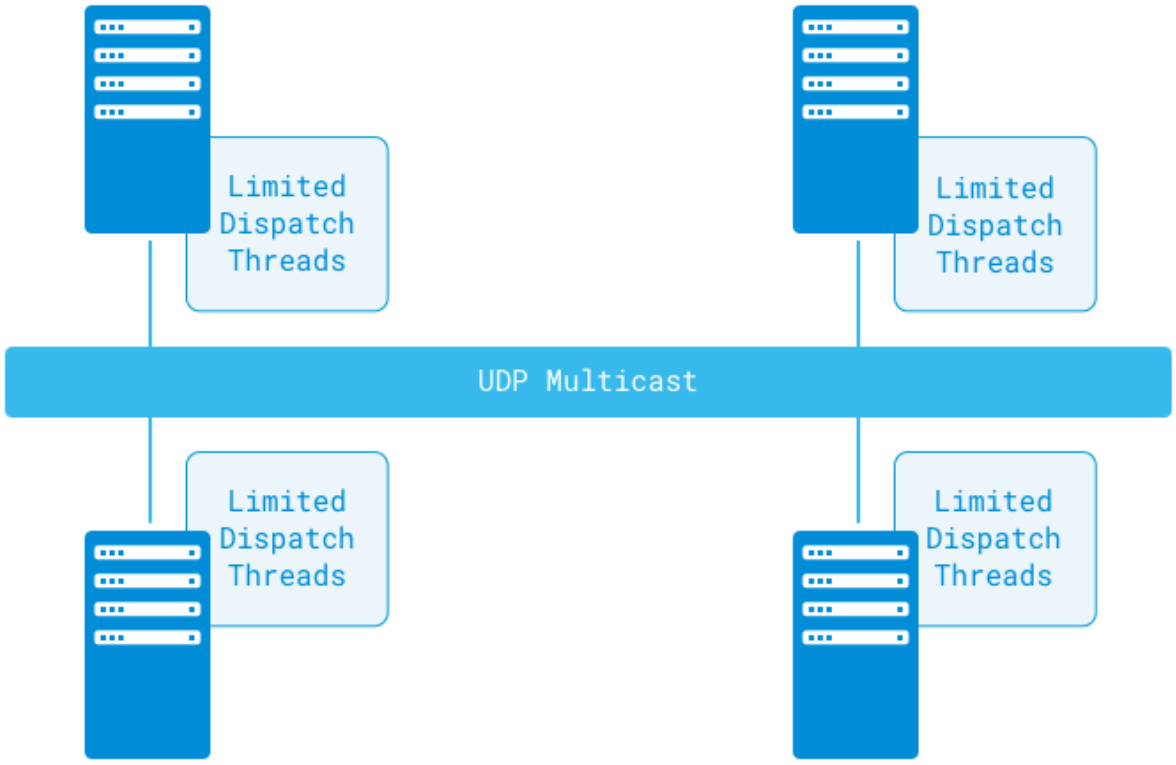


Figure 2: Liferay DXP's cache algorithm is extremely efficient.

To enable Cluster Link, add this property to `portal-ext.properties`:

```
cluster.link.enabled=true
```

Cluster Link depends on JGroups, and provides an API for nodes to communicate. It can

- Send messages to all nodes in a cluster
- Send messages to a specific node
- Invoke methods and retrieve values from all, some, or specific nodes
- Detect membership and notify when nodes join or leave

```
-------------------------------------------------------------------
GMS: address=oz-52865, cluster=liferay-channel-control, physical address=192.168.1.10:
-------------------------------------------------------------------
```

Cluster Link contains an enhanced algorithm that provides one-to-many type communication between the nodes. This is implemented by default with JGroups's UDP multicast, but unicast and TCP are also available.

MULTICAST OVER UDP

When you enable Cluster Link, Liferay DXP's default clustering configuration is enabled. This configuration defines IP multicast over UDP. Liferay DXP uses two groups of channels from JGroups to implement this: a control group and a transport group. If you want to customize the channel properties, you can do so in `portal-ext.properties` :

```
cluster.link.channel.name.control=[your control channel name]
cluster.link.channel.properties.control=[your control channel properties]
```

Please see JGroups's documentation for channel properties. The default configuration sets many properties whose settings are discussed there.

Multicast broadcasts to all devices on the network. Clustered environments on the same network communicate with each other by default. Messages and information (e.g., scheduled tasks) sent between them can lead to unintended consequences. Isolate such cluster environments by either separating them logically or physically on the network, or by configuring each cluster's `portal-ext.properties` to use different sets of multicast group address and port values.

JGroups sets a bind address automatically, using `localhost` by default. In some configurations, however, `localhost` is bound to the internal loopback network (`127.0.0.1` or `::1`), rather than the host's real address. As long as DXP's `cluster.link.autodetect.address` Portal Property points to a server that's contactable, DXP uses that server to automatically detect your host's real address. Here's the default setting:

```
cluster.link.autodetect.address=www.google.com:80
```

Contacting Google may not work if your server is behind a firewall.

1. Disable address auto-detection by setting the `cluster.link.autodetect.address` property to an empty value:

   ```
   cluster.link.autodetect.address=
   ```

2. Set the following properties to your host's IP address:

   ```
   cluster.link.bind.addr["cluster-link-control"]=[place your IP address or host nam
   cluster.link.bind.addr["cluster-link-udp"]=[place your IP address or host name he
   ```

Your network configuration may preclude the use of multicast over TCP, so below are some other ways you can get your cluster communicating. Note that these methods are all provided by JGroups.

Checkpoint:

1. If you are binding the IP address instead of using `localhost`, make sure the right IP addresses are declared using:

   ```
   cluster.link.bind.addr["cluster-link-control"]=localhost
   cluster.link.bind.addr["cluster-link-udp"]=localhost
   ```

2. Test your load and then optimize your settings if necessary.

UNICAST OVER TCP

If your network configuration or the sheer distance between nodes prevents you from using UDP Multicast clustering, you can configure Liferay DXP to use TCP Unicast. You'll definitely need this if you have a firewall separating any of your nodes or if your nodes are in different geographical locations.

1. Add a parameter to your app server's JVM:

   ```
   -Djgroups.bind_addr=[place your IP address or host name here]
   ```

   Use the node's IP address or host name.

```
- TCPPing
- JDBCPing
- S3_Ping
- Rackspace_Ping
```

If you aren't sure which one to choose, use TCPPing. This is used in the rest of these steps; the others are covered below.

3. Download the latest `com.liferay.portal.cluster.multiple-[version].jar` file from Liferay's Nexus repository. In this JAR's `lib` folder is a file called `jgroups-[version].Final.jar`. Open it and find `tcp.xml`. Extract this file to a location accessible to Liferay DXP. Use this file on all your nodes.

4. If you're vertically clustering (i.e., you have multiple Liferay DXP servers running on the same physical or virtual system), you must change the port on which discovery communicates for all nodes other than the first one, to avoid TCP port collision. To do this, modify the TCP tag's `bind_port` parameter:

```
<TCP bind_port="[some unused port]"
    ...
/>
```

Since the default port is `7800`, provide some other unused port.

5. Add to the same tag the parameter `singleton_name="liferay_cluster"`. This merges the transport and control channels to reduce the number of thread pools. See JGroups documentation for further information.

Usually, no further JGroups configuration is required. However, in a very specific case, if *(and only if)* cluster nodes are deployed across multiple networks, then the parameter `external_addr` must be set on each host to the external (public IP) address of the firewall. This kind of configuration is usually only necessary when nodes are geographically separated. By setting this, clustered nodes that are deployed to separate networks (e.g. separated by different firewalls) can communicate together. This configuration will likely be flagged in security audits of your system. See JGroups documentation for more information.

6. Save the file. Modify that node's `portal-ext.properties` file to point to it:

You're now set up for Unicast over TCP clustering! Repeat this process for each node you want to add to the cluster.

JDBC Ping

Rather than use TCP Ping to discover cluster members, you can use a central database accessible by all the nodes to help them find each other. Cluster members write their own and read the other members' addresses from this database. To enable this configuration, replace the `TCPPING` tag with the corresponding `JDBCPING` tag:

```
<JDBC_PING
    connection_url="jdbc:mysql://[DATABASE_IP]/[DATABASE_NAME]?useUnicode=true&amp;cha
    connection_username="[DATABASE_USER]"
    connection_password="[DATABASE_PASSWORD]"
    connection_driver="com.mysql.jdbc.Driver"/>
```

The above example uses MySQL as the database. For further information about JDBC Ping, please see the JGroups Documentation.

S3 Ping

Amazon S3 Ping can be used for servers running on Amazon's EC2 cloud service. Each node uploads a small file to an S3 bucket, and all the other nodes read the files from this bucket to discover the other nodes. When a node leaves, its file is deleted.

To configure S3 Ping, replace the `TCPPING` tag with the corresponding `S3_PING` tag:

```
<S3_PING
    secret_access_key="[SECRETKEY]"
    access_key="[ACCESSKEY]"
    location="ControlBucket"/>
```

Supply your Amazon keys as values for the parameters above. For further information about S3 Ping, please see the JGroups Documentation.

Other Pings

MODIFYING THE CACHE CONFIGURATION WITH A MODULE

It's recommended to test your system under a load that best simulates the kind of traffic your system needs to handle. If you'll be serving up a lot of message board messages, your script should reflect that. If web content is the core of your site, your script should reflect that too.

As a result of a load test, you may find that the default distributed cache settings aren't optimized for your site. In this case, you should tweak the settings yourself. You can modify the Liferay DXP installation directly or you can use a module to do it. Either way, the settings you change are the same. A benefit of working with modules is that you can install a module on each node and change the settings without taking down the cluster. Modifying the Ehcache settings with a module is recommended over modifying the Ehcache settings directly.

We've made this as easy as possible by creating the project for you. Download the project and unzip it into a Liferay Workspace, in the workspace's `modules` folder. To override your cache settings, you only have to modify one Ehcache configuration file, which you'll find in this folder structure:

- src
  - main
    - java
      - resources
        - ehcache
          - override-liferay-multi-vm-clustered.xml

In the sample project, this file contains a configuration for Liferay DXP's `GroupImpl` object which handles sites. You may wish to add other objects to the cache; in fact, the default file caches many other objects. For example, if you have a vibrant community, a large portion of your traffic may be directed at the message boards portlet, as in the example above. To cache the threads on the message boards, configure a block with the `MBMessageImpl` class:

```
<cache
        eternal="false"
        maxElementsInMemory="10000"
        name="com.liferay.portlet.messageboards.model.impl.MBMessageImpl"
        overflowToDisk="false"
        timeToIdleSeconds="600"
>
</cache>
```

the Liferay Foundation application suite's `com.liferay.portal.ehcache-[version].jar` file. You can get this JAR from the `Liferay Foundation.lpkg` file in the `osgi/marketplace` folder. The file you want is `liferay-multi-vm-clustered.xml`, in the `/ehcache` folder inside the `com.liferay.portal.ehcache-[version].jar` file. Once you have the file, replace the contents of the `override-liferay-multi-vm-clustered.xml` file above with the contents of this file. Now you'll be using the default configuration as a starting point.

Once you've made your changes to the cache, save the file, build, and deploy the module, and your settings override the default settings. In this way, you can tweak your cache settings so that your cache performs optimally for the type of traffic generated by your site. You don't have restart your server to change the cache settings. This is a great benefit, but beware: since Ehcache doesn't allow for changes to cache settings while the cache is alive, reconfiguring a cache while the server is running flushes the cache.

5. Hot Deploy to All Nodes

If you want to deploy any module or WAR file onto the cluster, it must be deployed to all nodes of the cluster. Because Liferay DXP now installs applications as OSGi bundles, this means you cannot rely on your application server's means of installing WAR files (even if you only intend to install WAR files) to deploy an application to the entire cluster. Instead, the application must be placed in Liferay DXP's `deploy` folder on each node.

This, as you might imagine, can be done with a script. Write a shell script that uploads applications to each node using sftp or some other service. This way, when you deploy an application, it is uploaded to each node's `deploy` folder and installed by each running Liferay DXP installation.

Summary

Setting up Liferay DXP on a cluster takes five steps:

1. Point all nodes at the same database or database cluster.
2. Make sure the Documents and Media repository is accessible to all nodes.
3. Install Elasticsearch or Solr on a separate system or cluster.
4. Enable Cluster Link for cache replication.
5. Hot deploy applications to each node individually.

« Document Repository Configuration Updating a Cluster »

¿Fue útil este artículo?

Sí No

Usuarios a los que les pareció útil: 2 de 4

¿Necesitas más ayuda?

Abre un ticket

Contacta con nosotros

Pregunta en la Comunidad

Liferay University
Forma a tu equipo en los puntos esenciales del uso de Liferay con videos detallados y ejercicios prácticos diseñados para simular una formación presencial

Liferay Global Services
Aumenta tu conocimiento sobre el producto y el desarrollo gracias a expertos que te ayudarán a maximizar tu ROI

activación

Soporte de la cuenta

Preguntas frecuentes a soporte

© 2023 LIFERAY INC. ALL RIGHTS RESERVED