Deploying a Fedora Cluster

- Configuration
- Step-By-Step guides for deploying Fedora 4 clusters
  - Deploy cluster using the UDP Multicast protocol for node discovery and the TCP protocol for replication
    - JGroups configuration
    - Infinispan configuration
    - Modeshape configuration
    - Using Tomcat7
  - Deploy cluster using the UDP Multicast protocol for node discovery and replication
    - Using Tomcat7
  - Deploy cluster using the UDP Multicast protocol for node discovery and replication on a single machine
    - Using Tomcat7
- Using TCP for Discovery and Sync
  - TCPPING Element
  - TCP Element
- Deploying in AWS
- Load Balancing Fedora 4 using Apache and mod_jk
- Firewall Notes
- Simple Shell script to coordinate a cluster

To support horizontal scalability use cases as well as geographic distribution, Fedora 4 can be configured as a cluster of application servers.

Configuration

Fedora 4 is built in top of the JCR implementation Modeshape. Modeshape uses Infinispan as a distributed datastore, which in turn uses the Messaging Toolkit JGroups to transfer state between nodes.

Therefore the following resources and documents contain a lot of important information about configuring Fedora 4's underlying projects.

- Fedora configuration inventory
- Modeshape documentation
- Modeshape configuration section
- Infinispan documentation
- Infinispan configuration section
- JGroups documentation
- JGroups configuration section

Step-By-Step guides for deploying Fedora 4 clusters

Deploy cluster using the UDP Multicast protocol for node discovery and the TCP protocol for replication

A few configuration options have to be set in order to have Fedora 4 work as a cluster on a local machine:

- **-Xmx1024m** Set the Java heap to 1GB
- **-XX:MaxPermSize=256m** Set the Java PermGen size to 256MB
- **-Dfcrepo.modeshape.configuration=file:///path/to/repository.json** The Modeshape configuration used for clustering
- **-Djava.net.preferIPv4Stack=true** Tell Java to use IPv4 rather than IPv6
- **fcrepo.ispn.jgroups.configuration=/path/to/jgroups-fcrepo-tcp.xml** Set the JGroups configuration file holding the TCP Transport defintions
- **jgroups.udp.mcast_addr=239.42.42.42** Set the UDP multicast address for the JGroups cluster
- **fcrepo.infinispan.cache_configuration=/path/to/infinispan.xml** Set the Infinispan configuration file holding the Infinispan cluster configuration

JGroups configuration

In order to use the UDP Multicasting for node discovery and TCP for replication the following JGroups example configuration can be used:

jgroups-fcrepo-tcp.xml

```
<config xmlns="urn:org:jgroups" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="urn:
org:jgroups http://www.jgroups.org/schema/JGroups-3.0.xsd">
        <TCP bind_port="7800"
                loopback="false"
                recv_buf_size="${tcp.recv_buf_size:5M}"
                send_buf_size="${tcp.send_buf_size:640K}"
                max_bundle_size="64K"
                max_bundle_timeout="30"
                use_send_queues="true"
                sock_conn_timeout="300"
                timer_type="new3"
                timer.min_threads="4"
                timer.max_threads="10"
                timer.keep_alive_time="3000"
                timer.queue_max_size="500"
                thread_pool.enabled="true"
                thread_pool.min_threads="1"
                thread_pool.max_threads="10"
                thread_pool.keep_alive_time="5000"
                thread_pool.queue_enabled="true"
                thread_pool.queue_max_size="10000"
                thread_pool.rejection_policy="discard"
                oob_thread_pool.enabled="true"
                oob_thread_pool.min_threads="1"
                oob_thread_pool.max_threads="8"
                oob_thread_pool.keep_alive_time="5000"
                oob_thread_pool.queue_enabled="false"
                oob_thread_pool.queue_max_size="100"
                oob_thread_pool.rejection_policy="discard"/>
        <MPING timeout="1000"
                num_initial_members="1"/>
        <MERGE2 max_interval="30000"
                min_interval="10000"/>
        <FD_ALL timeout="150000"/>
        <VERIFY_SUSPECT timeout="150000" />
        <BARRIER />
        <pbcast.NAKACK2 use_mcast_xmit="false"
                discard_delivered_msgs="true"/>
        <UNICAST timeout="600,900,2500"/>
        <pbcast.STABLE stability_delay="2000" desired_avg_gossip="50000"
                max_bytes="4M"/>
        <pbcast.GMS print_local_addr="true" join_timeout="6000"
                view_bundling="true"/>
        <MFC max_credits="2M"
                min_threshold="0.4"/>
        <FRAG2 frag_size="60K" />
        <pbcast.STATE_TRANSFER />
</config>
```

Infinispan configuration

The following example configuration has it's replication timeout set to 10 minutes in order to mitigate the problem of SyncTimeouts when spanning one transaction over a lot of operations

infinispan.xml

```
<infinispan xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="urn:infinispan:config:5.2 http://www.infinispan.org/schemas/infinispan-config-
5.2.xsd"
            xmlns="urn:infinispan:config:5.2">
  <global>
    <globalJmxStatistics enabled="true" allowDuplicateDomains="true"/>
    <transport clusterName="modeshape-cluster">
```

```
      <properties>
        <property name="configurationFile" value="${fcrepo.ispn.jgroups.configuration:config/jgroups-fcrepo-tcp.
xml}"/>
      </properties>
    </transport>
  </global>
  <default>
    <clustering mode="distribution">
      <sync replTimeout="600000"/>
      <l1 enabled="false" lifespan="0" onRehash="false"/>
      <hash numOwners="${fcrepo.ispn.numOwners:2}"/>
      <stateTransfer chunkSize="100" fetchInMemoryState="true"/>
    </clustering>
  </default>
  <namedCache name="FedoraRepository">
    <clustering mode="replication">
      <sync replTimeout="6000000"/>
      <l1 enabled="false" lifespan="0" onRehash="false"/>
      <stateTransfer chunkSize="100" fetchInMemoryState="true" timeout="120000"/>
    </clustering>
    <locking isolationLevel="READ_COMMITTED" writeSkewCheck="false" lockAcquisitionTimeout="150000"
useLockStriping="true" />
    <transaction transactionMode="TRANSACTIONAL" lockingMode="PESSIMISTIC"/>
    <loaders passivation="false" shared="false" preload="false">
      <loader class="org.infinispan.loaders.file.FileCacheStore" fetchPersistentState="true"
              purgeOnStartup="false">
        <properties>
          <property name="location" value="${fcrepo.ispn.repo.CacheDirPath:target/FedoraRepository/storage}"/>
          <property name="fsyncMode" value="perWrite"/>
        </properties>
      </loader>
    </loaders>
  </namedCache>
  <namedCache name="FedoraRepositoryMetaData">
    <clustering mode="distribution">
      <sync replTimeout="600000"/>
      <l1 enabled="false" lifespan="0" onRehash="false"/>
      <hash numOwners="${fcrepo.ispn.numOwners:2}"/>
      <stateTransfer chunkSize="100" fetchInMemoryState="true"/>
    </clustering>
    <locking concurrencyLevel="1000" lockAcquisitionTimeout="150000" useLockStriping="false" />
    <deadlockDetection enabled="true" spinDuration="1000"/>
    <eviction maxEntries="500" strategy="LIRS" threadPolicy="DEFAULT"/>
    <transaction
            transactionManagerLookupClass="org.infinispan.transaction.lookup.GenericTransactionManagerLookup"
            transactionMode="TRANSACTIONAL" lockingMode="PESSIMISTIC"/>
    <loaders passivation="false" shared="false" preload="false">
      <loader class="org.infinispan.loaders.file.FileCacheStore" fetchPersistentState="true"
              purgeOnStartup="false">
        <properties>
          <property name="location" value="${fcrepo.ispn.CacheDirPath:target/FedoraRepositoryMetaData/storage}"
/>
          <property name="fsyncMode" value="perWrite"/>
        </properties>
      </loader>
    </loaders>
  </namedCache>
  <namedCache name="FedoraRepositoryBinaryData">
    <clustering mode="distribution">
      <sync replTimeout="600000"/>
      <l1 enabled="false" lifespan="0" onRehash="false"/>
      <hash numOwners="${fcrepo.ispn.numOwners:2}"/>
      <stateTransfer chunkSize="100" fetchInMemoryState="true"/>
    </clustering>
    <locking concurrencyLevel="1000" lockAcquisitionTimeout="150000" useLockStriping="false" />
    <deadlockDetection enabled="true" spinDuration="1000"/>
    <eviction maxEntries="100" strategy="LIRS" threadPolicy="DEFAULT"/>
    <transaction
            transactionManagerLookupClass="org.infinispan.transaction.lookup.GenericTransactionManagerLookup"
            transactionMode="TRANSACTIONAL" lockingMode="PESSIMISTIC"/>
    <loaders passivation="false" shared="false" preload="false">
```

```
      <loader class="org.infinispan.loaders.file.FileCacheStore" fetchPersistentState="true"
              purgeOnStartup="false">
        <properties>
          <property name="location" value="${fcrepo.ispn.binary.CacheDirPath:target/FedoraRepositoryBinaryData
/storage}"/>
          <property name="fsyncMode" value="perWrite"/>
        </properties>
      </loader>
    </loaders>
  </namedCache>
</infinispan>
```

Modeshape configuration

The following configuration has indexing disabled completely in order to increase ingest performance

repository.json

```
{
    "name" : "repo",
    "jndiName" : "",
    "workspaces" : {
        "predefined" : ["default"],
        "default" : "default",
        "allowCreation" : true
    },
    "clustering" : {
        "clusterName" : "modeshape-cluster"
    },
    "query" : {
        "enabled" : "false",
    },
    "storage" : {
        "cacheName" : "FedoraRepository",
        "cacheConfiguration" : "${fcrepo.infinispan.cache_configuration:config/infinispan/clustered/infinispan.
xml}",
        "binaryStorage" : {
            "type" : "cache",
            "dataCacheName" : "FedoraRepositoryBinaryData",
            "metadataCacheName" : "FedoraRepositoryMetaData"
        }
    },
    "security" : {
        "anonymous" : {
            "roles" : ["readonly","readwrite","admin"],
            "useOnFailedLogin" : false
        },
        "providers" : [
            { "classname" : "org.fcrepo.http.commons.session.BypassSecurityServletAuthenticationProvider" }
        ]
    },
    "node-types" : ["fedora-node-types.cnd"]
}
```

Using Tomcat7

1. *Build the fcrepo war file or download the prebuilt fcrepo war file*
    a. Build the War file as described on this page
    b. Fetch the WAR file from the download page
2. *Get Tomcat*
    a. Download Tomcat 7 and unpack it (Tomcat 7.0.50 is used in this example)

```
#> wget http://mirror.synyx.de/apache/tomcat/tomcat-7/v7.0.50/bin/apache-tomcat-7.0.50.tar.gz
#> tar -zxvf apache-tomcat-7.0.50.tar.gz
#> mv apache-tomcat-7.0.50 tomcat7
```

3. *Put the WAR file into tomcat's webapp directory or create a symbolic link*
    a. Copy the fcrepo-webapp-VERSION.war file

```
#> cp fcrepo-webapp-VERSION.war tomcat7/webapps/fcrepo.war
```

4. *Set the send/recv buffer sizes if neccessary*
    a. Use the following commands to set the buffer size. To persist these settings between reboots, also put them in */etc/sysctl.conf*.

```
#> sysctl net.core.rmem_max=26214400
#> sysctl net.core.wmem_max=5242880
```

5. *Start instances*
    a. Using a custom configuration by pointing Fedora 4 to custom configuration files:

```
#> CATALINA_OPTS="-Xmx1024m -XX:MaxPermSize=256m -Djava.net.preferIPv4Stack=true -Djgroups.udp.
mcast_addr=239.42.42.42 -Dfcrepo.modeshape.configuration=file:///path/to/repository.json -Dfcrepo.
ispn.jgroups.configuration=/path/to/jgroups-fcrepo-tcp.xml  -Dfcrepo.infinispan.
cache_configuration=/path/to/infinispan.xml" bin/catalina.sh run
```

Deploy cluster using the UDP Multicast protocol for node discovery and replication

Issues with UDP Multicasting

Currently there are still issues using UDP Mulitcasting for replication, while using UDP for node discovery works as intended.

A couple of configuration options have to be set in order to have Fedora 4 work as a cluster on a local machine:

- **Xmx1024m** Set the Java heap to 1GB
- **XX:MaxPermSize=256m** Set the Java PermGen size to 256MB
- **fcrepo.modeshape.configuration=file:///path/to/repository.json** The Modeshape configuration used for clustering
- **java.net.preferIPv4Stack=true** Tell Java to use IPv4 rather than IPv6
- **fcrepo.ispn.jgroups.configuration=/path/to/jgroups-fcrepo-udp.xml** Set the JGroups configuration file holding the UDP Transport defintions
- **jgroups.udp.mcast_addr=239.42.42.42** Set the UDP multicast address for the JGroups cluster
- **fcrepo.infinispan.cache_configuration=/path/to/infinispan.xml** Set the Infinispan configuration file holding the Infinispan cluster configuration

Using Tomcat7

1. *Build the fcrepo war file or download the prebuilt fcrepo war file*
    a. Build the War file as described on this page
    b. Fetch the WAR file from the download page
2. *Get Tomcat*
    a. Download Tomcat 7 and unpack it (Tomcat 7.0.50 is used in this example)

```
#> wget http://mirror.synyx.de/apache/tomcat/tomcat-7/v7.0.50/bin/apache-tomcat-7.0.50.tar.gz
#> tar -zxvf apache-tomcat-7.0.50.tar.gz
#> mv apache-tomcat-7.0.50 tomcat7
```

3. *Put the WAR file into tomcat's webapp directory or create a symbolic link*
    a. Copy the fcrepo-webapp-VERSION.war file

```
#> cp fcrepo-webapp-VERSION.war tomcat7/webapps/fcrepo.war
```

4. *Setup the cluster configuration (Optional)*
    a. This github project contains a sample configuration for a cluster in distributed mode.
    b. Change the configuration as required. Description is available at the JGroups, Infinispan and Modeshape documentations

c. Make sure to point Fedora 4 to the configuration files by updating the file `$TOMCAT_HOME/bin/setenv.sh` (create if necessary) using the properties `fcrepo.modeshape.configuration, fcrepo.ispn.jgroups.configuration` and `fcrepo.infinispan.cache_configuration`

5. *Set the send/recv buffer sizes if neccessary*
   a. Use the following commands to set the buffer size

```
#> sysctl net.core.rmem_max=5242880
#> sysctl net.core.wmem_max=5242880
```

6. *Start instance*
   a. Using the default clustered configuration (Replication mode):

```
#> CATALINA_OPTS="-Xmx1024m -XX:MaxPermSize=256m -Dfcrepo.modeshape.configuration=config/clustered
/repository.json -Djava.net.preferIPv4Stack=true -Djgroups.udp.mcast_addr=239.42.42.42" bin
/catalina.sh run
```

   b. Using a custom configuration by pointing Fedora 4 to custom configuration files:

```
#> CATALINA_OPTS="-Xmx1024m -XX:MaxPermSize=256m -Djava.net.preferIPv4Stack=true -Djgroups.udp.
mcast_addr=239.42.42.42 -Dfcrepo.modeshape.configuration=file:///path/to/repository.json -Dfcrepo.
ispn.jgroups.configuration=/path/to/jgroups-fedora-udp.xml  -Dfcrepo.infinispan.
cache_configuration=/path/to/infinispan.xml" bin/catalina.sh run
```

Deploy cluster using the UDP Multicast protocol for node discovery and replication on a single machine

Issues with UDP Multicasting

Currently there are still issues using UDP Mulitcasting for replication, while using UDP for node discovery works as intended.

A couple of configuration options have to be set in order to have Fedora 4 work as a cluster on a local machine:

- **Xmx1024m** Set the Java heap to 1GB
- **XX:MaxPermSize=256m** Set the Java PermGen size to 256MB
- **fcrepo.modeshape.configuration=file:///home/ruckus/dev/tomcat7-8081/repository.json** The Modeshape configuration used for clustering
- **java.net.preferIPv4Stack=true** Tell Java to use IPv4 rather than IPv6
- **fcrepo.ispn.jgroups.configuration=/home/ruckus/dev/tomcat7-8081/jgroups-fcrepo-udp.xml** Set the JGroups configuration file holding the UDP Transport defintions
- **jgroups.udp.mcast_addr=239.42.42.42** Set the UDP multicast address for the JGroups cluster
- **fcrepo.infinispan.cache_configuration=/home/ruckus/dev/tomcat7-8081/infinispan.xml** Set the Infinispan configuration file holding the Infinispan cluster configuration
- **jms.port=61617** The port used by ActiveMQ's JMS protocol. This needs to be distinct for every instance of fcrepo4
- **stomp.port=61614** The port used by ActiveMQ Stomp protocol. This needs to be distinct for every instance of fcrepo4

Using Tomcat7

1. *Build the fcrepo war file or download the prebuilt fcrepo war file*
   a. Build the War file as described on this page
   b. Fetch the WAR file from the download page
2. *Get Tomcat*
   a. Download Tomcat 7 and unpack it (Tomcat 7.0.50 is used in this example)

```
#> wget http://mirror.synyx.de/apache/tomcat/tomcat-7/v7.0.50/bin/apache-tomcat-7.0.50.tar.gz
#> tar -zxvf apache-tomcat-7.0.50.tar.gz
#> mv apache-tomcat-7.0.50 tomcat7-8080
```

3. *Put the WAR file into tomcat's webapp directory or create a symbolic link*
   a. Copy the fcrepo-webapp-VERSION.war file

```
#> cp fcrepo-webapp-VERSION.war tomcat7-8080/webapps/fcrepo.war
```

4. *Get the Infinispan XML configuration file*
    a. Download from github and put it into tomcat7-8080

```
#> wget -O infinispan.xml https://gist.github.com/fasseg/8646707/raw
#> mv infinispan.xml tomcat7-8080/
```

5. *Get the Modeshape JSON configuration file*
    a. Download from Github and put into tomcat7-8080

```
#> wget -O repository.json https://gist.github.com/fasseg/8646727/raw
#> mv repository.json tomcat7-8080/
```

6. *Get the JGroups UDP Transport configuration file*
    a. Download from Github and put into tomcat7-8080

```
#> wget -O jgroups-fedora-udp.xml https://gist.github.com/fasseg/8646743/raw
#> mv jgroups-fedora-udp.xml tomcat7-8080/
```

7. *Set the send/recv buffer sizes if neccessary*
    a. Use the following commands to set the buffer size

```
#> sysctl net.core.rmem_max=5242880
#> sysctl net.core.wmem_max=5242880
```

8. *Copy the whole tomcat folder in order to create a second instance*
    a. `#> cp -R tomcat7-8080/ tomcat7-8081`

9. *Change the connector ports of the second Tomcat instance to 8081, 8006 and 8010*
    a.
```
#> sed -i 's/8080/8081/g' tomcat7-8081/conf/server.xml
#> sed -i 's/8005/8006/g' tomcat7-8081/conf/server.xml
#> sed -i 's/8009/8010/g' tomcat7-8081/conf/server.xml
```

10. *Start the first instance*
    a.
```
#> CATALINA_OPTS="-Xmx1024m -XX:MaxPermSize=256m -Dfcrepo.modeshape.configuration=file:///path/to
/repository.json -Djava.net.preferIPv4Stack=true -Dfcrepo.ispn.jgroups.configuration=/path/to
/jgroups-fedora-udp.xml -Djgroups.udp.mcast_addr=239.42.42.42 -Dfcrepo.infinispan.
cache_configuration=/path/to/infinispan.xml" tomcat7-8080/bin/catalina.sh run
```

11. *Start the second instance*
    a.
```
#> CATALINA_OPTS="-Xmx1024m -XX:MaxPermSize=256m -Dfcrepo.modeshape.configuration=file:///path/to
/repository.json -Djava.net.preferIPv4Stack=true -Dfcrepo.ispn.jgroups.configuration=/path/to
/jgroups-fedora-udp.xml -Djgroups.udp.mcast_addr=239.42.42.42 -Dfcrepo.infinispan.
cache_configuration=/path/to/infinispan.xml -Djms.port=61617 -Dstomp.port=61614" tomcat7-8081/bin
/catalina.sh run
```

12. *Check that the instances are reachable and that the clustersize is '2':*
    a.
```
#> wget http://localhost:8080/fcrepo/rest
#> wget http://localhost:8081/fcrepo/rest
```

    b. Navigate to http://localhost:8080/fcrepo/rest

Using TCP for Discovery and Sync

If you cannot use UDP, which is recommended for larger clusters, then TCP can be used. There are some things to watch out for when configuring TCP.

TCPPING Element

Each host has a different TCPPING configuration. This can be tricky to configure:

- The initial_hosts attribute should only include remote hosts for the local node, not itself.
- The num_initial_members should match the count of remote hosts, i.e. it does not include the local node either.
- If you are running just one port on each host, then port_range will be 0.
- IP resolution is important if you use DNS names. The locally resolved IP of each remote host must match the TCP@bind_addr in the remote host config.

TCP Element

If you use a hostname for bind_addr, make sure that it resolved to the IP you want, probably an external one and not the loopback IP. This is easy to miss.

Deploying in AWS

Java options for Tomcat or Jetty

```
JAVA_OPTS="$JAVA_OPTS -Xmx1024m -XX:MaxPermSize=256m -Dfcrepo.modeshape.configuration=file:///config/clustered
/repository.json"
JAVA_OPTS="$JAVA_OPTS -Dfcrepo.infinispan.cache_configuration=config/infinispan/clustered/infinispan.xml"

JAVA_OPTS="${JAVA_OPTS} -Dfcrepo.home=/tmp/wherever"

JAVA_OPTS="${JAVA_OPTS} -Djgroups.tcp.address=<private-ip-address-of-ec2-instance>"

JAVA_OPTS="${JAVA_OPTS} -Dfcrepo.ispn.numOwners=2 -Djava.net.PreferIPv4Stack=true"

# The jgroups-ec2.xml file is included in ispn's jars
JAVA_OPTS="${JAVA_OPTS} -Dfcrepo.ispn.jgroups.configuration=jgroups-ec2.xml"

# This property overwrites the S3 bucketname variable in jgroups-ec2.xml
JAVA_OPTS="${JAVA_OPTS} -Djgroups.s3.bucket=<some-bucket-that-you-have-already-created>"
```

See this example on Fedora Cluster Installation in AWS

Load Balancing Fedora 4 using Apache and mod_jk

Load balancing can be achieved by using an Apache server with mod_jk in front of the Fedora 4 cluster. Using mod_jk one has to create as many workers in the workers.properties configuration file as there are Fedora 4 nodes.

See this example on the RedHat pages

Firewall Notes

If your cluster has a firewall between nodes, you'll need to open the following ports:

- UDP multicast source address and port for JGroups messaging

  Example: if your nodes are on the 192.168.47.x network, then this would be your iptables rule:

UDP multicast firewall rule
`-A INPUT -m pkttype --pkt-type multicast -s 192.168.47.0/24 -j ACCEPT`

- TCP replication: TCP bind address and port (source)
  Example: if your nodes are on the 192.168.47.x network, and the TCP bind_port is 7800, then this would be your iptables rule:

TCP replication firewall rule

```
-A INPUT -m state --state NEW -m tcp -p tcp -s 192.168.47.0/24 --dport 7800 -j ACCEPT
```

- UDP replication: UDP bind address and port (source)
  Example: if your nodes are on the 192.168.47.x network, and the UDP bind_port is 47788, then this would be your iptables rule:

UDP replication firewall rule

```
-A INPUT -m state --state NEW -m udp -p udp -s 192.168.47.0/24 --dport 47788 -j ACCEPT
```

Simple Shell script to coordinate a cluster

For pushing configurations and wars/jars, start, stop, restart and purge the Ubuntu 12.04 LTS cluster this small script gets used on the FIZ cluster.

- In order to make this work without having to input passwords all the time for the sudo and ssh calls on the cluster nodes, I distributed a public ssh key on the cluster nodes for ssh auth and allowed the fcrepo user to execute `sudo` calls to `rm, cp, service` calls without a password.
- The configuration of the FIZ cluster can be accessed here: https://github.com/fasseg/fiz-fcrepo-cluster-config. There is also a setenv.sh file in there which we symlinked to $TOMCAT_HOME/bin/setenv.sh, that sets the environment variables for the repository, jgroups and infinispan configuration.
- So on each node the layout on the file system looks like this:
  - `/data/fcrepo` (the exploded war file, owned by fcrepo)
  - `/home/fcrepo/fiz-cluster-config` (the configuration and setenv.sh file, owned by fcrepo)
  - `/var/lib/tomcat7/webapps/fedora` (owned by root) symlinks to /data/fcrepo
- Using this setup jar updates can be pushed by the shell script to /data/fcrepo/WEB-INF/lib directly.
- Pushing a new WAR file to the nodes requires unpacking the WAR to /data/fcrepo therefore access to /tmp is required.
- The script is setup for six nodes with know IPs. So the `node[]` array will have to change for different configurations, as should the range defined in the for statements in `start_cluster() purge_cluster()` and `stop_cluster()`.

cluster

```
#!/bin/bash
nodes[0]=192.168.42.101
nodes[1]=192.168.42.102
nodes[2]=192.168.42.103
nodes[3]=192.168.42.104
nodes[4]=192.168.42.105
nodes[5]=192.168.42.106
start_node() {
        if [[ -z "$1" ]]; then
                echo "No node argument suplied [1-6]"
                exit 1
        fi
        nodeip=${nodes[(($1 - 1))]}
        echo -n "Starting node $nodeip..."
        ssh -qt fcrepo@$nodeip "sudo service tomcat7 start > /dev/null"
        curl -sf http://${nodeip}:8080/fcrepo/rest > /dev/null
        until [ $? -eq 0 ]
        do
                curl -sf http://${nodeip}:8080/fcrepo/rest > /dev/null
        done
        echo "done."
}
stop_node() {
        if [[ -z "$1" ]]; then
                echo "No node argument suplied [1-6]"
                exit 1
```

```
        fi
        nodeip=${nodes[(($1 - 1))]}
        echo -n "Stopping node $nodeip..."
        ssh -qt fcrepo@$nodeip "sudo service tomcat7 stop > /dev/null"
        curl -sf http://${nodeip}:8080/fcrepo/rest > /dev/null
        until [ $? -gt 0 ]
        do
                curl -sf http://${nodeip}:8080/fcrepo/rest > /dev/null
        done
        echo "done."
}
restart_node() {
        stop_node $1
        start_node $1
}
start_cluster() {
        echo "Starting cluster"
        for node in 1 2 3 4 5 6
        do
                start_node $node
        done
}
stop_cluster() {
        echo "Stopping cluster"
        for node in 1 2 3 4 5 6
        do
                stop_node $node
        done
}
restart_cluster() {
        stop_cluster
        start_cluster
}
status() {
        echo "Status of FIZ Fedora 4 cluster"
        for nodeip in "${nodes[@]}"
        do
                curl -sf http://${nodeip}:8080/fcrepo/rest > /dev/null
                if [ $? -gt 0 ];then
                        echo "$nodeip is OFFLINE"
                else
                        echo "$nodeip is online"
                fi
        done
}
purge() {
        echo "purging cluster"
        for nodeip in "${nodes[@]}"
        do
                echo -n "purging ${nodeip}..."
                ssh -qt fcrepo@$nodeip "sudo rm -Rf /var/lib/tomcat7/fcrepo4-data/* /var/lib/tomcat7/work
/Catalina/localhost/*"
                echo "done"
        done
}
push_config() {
        for nodeip in "${nodes[@]}"
        do
                echo "pushing config file $2 to $nodeip"
                scp $1 fcrepo@${nodeip}:fcrepo-config/
        done
}
restart_purge() {
        stop_cluster
        purge
        start_cluster
}
push_war() {
        stop_cluster
        purge
        rm -Rf /tmp/fcrepo
```

```
        mkdir /tmp/fcrepo
        unzip -qq $1 -d /tmp/fcrepo
        for nodeip in "${nodes[@]}"
        do
                echo -n "pushing WAR file to ${nodeip}..."
                ssh -qt fcrepo@${nodeip} "sudo rm -Rf /tmp/fcrepo"
                scp -qr /tmp/fcrepo fcrepo@${nodeip}:/tmp
                ssh -qt fcrepo@${nodeip} "sudo rm -Rf /opt/fcrepo/*"
                ssh -qt fcrepo@${nodeip} "sudo mv /tmp/fcrepo /opt/"
                echo "done.";
        done
}
push_jar() {
        stop_cluster
        purge
        for nodeip in "${nodes[@]}"
        do
                echo -n "pushing JAR to ${nodeip}..."
                scp -q $1 fcrepo@${nodeip}:/opt/fcrepo/WEB-INF/lib
                echo "done."
        done
}

case "$1" in
        start)
        start_cluster
        ;;
        stop)
        stop_cluster
        ;;
        restart)
        restart_cluster
        ;;
        stop-node)
        stop_node $2
        ;;
        start-node)
        start_node $2
        ;;
        restart-node)
        restart_node $2
        ;;
        status)
        status
        ;;
        purge)
        purge
        ;;
        push-config)
        push_config $2
        ;;
        restart-purge)
        restart_purge
        ;;
        push-war)
        push_war $2
        ;;
        push-jar)
        push_jar $2
        ;;
        *)
        echo $"Usage: $0 {start|stop|restart|restart-purge|start-node|stop-node|restart-node|status|purge|push-
config|push-war|push-jar}"
        exit 1
esac
```