Setting up a small hybris cluster

General Setup

VM initial setup

get CentOS 6.4 64 bit minimal ISO image from http://mirrors.cat.pdx.edu/centos/6.4/isos/x86_64/CentOS-6.4-x86_64-minimal.iso

Use your favorite VM software and create 3 VMs from this image:

- web: I allowed 1GB of RAM and one CPU core, network adapter bridged
- app1 and app2: I allowed 1 core and 2 GB of RAM for each and used the bridged adapters as well

Install the VMware tools for these new VM's (so you can get the time synchronization running and fix any possible problems with network adapter drivers):

- Select your VM, In Virtual Machine Menu, choose "Install VMware Tools"
- On the shell prompt in the VM we would need to execute the following commands:

```
yum install kernel-devel perl fuse-libs
mkdir /media/cdrom
mount /dev/cdrom /media/cdrom
cd /tmp
tar -xvf /media/cdrom/VM*
cd vmware-tools-distrib/
./vmware-install.pl
```

It mail fail to configure vmware tools giving an error bout thin-print, it happened to me in Centos 6.4. We do not need thin-print, you can fix this by editing /etc/init/vmware-tools-thinprint.conf . Comment out the following lines:

```
#pre-start exec /etc/vmware-tools/thinprint.sh
start
#post-stop exec /etc/vmware-tools/thinprint.sh
stop
```

About this Document

This is a detailed set of instructions on how to set up a small and simple hybris cluster consisting of one web server(reverse proxy) and 2 apps servers in the back. We will be starting with 3 Centos minimal images, we will name them web, app1 and app2 respectively. For the database we will use MySql but we won't create a different machine to host this, for simplicity it will be hosted on the web server.

What you'll need to go trough this exercise:

- PC or Mac with a dual or quad core CPU, 2 GHz speed and good amount of RAM (8 GB would be ideal)
- Your favorite VM software (VMware Fusion, VMware player, etc).
- ssh client if you run on windows (putty)

Save and now start the VMware tools with:

```
/etc/vmware-tools/services.sh start
```

- Now verify the current time is synced with the host
- Setup the host with a static ip. Configure /etc/sysconfig/network-scripts/ifcfg-eth0 to have a static ip address, make sure you add the correct gateway and dns servers, restart eth0 interface then test connection to internet (ping www.yahoo.com) .

Lets prepare the vms for our build. We will disable SELinux, add our application group and our application user and the application folder, appadm, appuser1and /app respectively: and make some additions to the profile.

```
setenforce 0
groupadd -g 8080 appadm
useradd  -u 8080 -g appadm appuser1
passwd appuser1
mkdir /apps
chown appuser1:appadm /apps
echo 'appuser1  ALL=(ALL:ALL) ALL' >> /etc/sudoers
echo 'export HISTTIMEFORMAT="%d/%m/%y %T "' >>
~/.bash_profile
echo 'export HISTTIMEFORMAT="%d/%m/%y %T "' >>
/home/appuser1/.bash_profile
#Optional : to be able to execute scripts in your
current directory do this for /root/.bash_profile and
/home/appuser1/.bash_profile
#modify the line PATH=$PATH:$HOME/bin  to
PATH=.:$PATH:$HOME/bin, note the  period right after the
equal sign. Save and execute the .bash_profile
source ~/.bash_profile
#Disable the firewall on all 3 servers. Once the
configuration is proven you could lock it up for only
required ports.
chkconfig iptables off
iptables --flush
```

Productivity tasks

Let's add the host entries for each of the servers in our hosts file on our client computer so we can reference them by name. On mac the hosts files is located in /etc/hosts, on windows c:\windows\system32\drivers\etc\hosts :

below is what I added on my hosts file. you could add these also on each of the 3 servers host files

```
10.5.0.71 web
10.5.0.74 app1
10.5.0.77 app2
```

On your terminal in the client computer: generate a key pair. The rest of the instructions are only suitable for Mac or a linux type desktop, but you could do similar things using putty and putty keygen in windows.

save the public and private keys in your ~/.ssh/ on the client computer and set the appropriate permissions:

```
ssh-keygen -t rsa
chmod 700 ~/.ssh
chmod 600 ~/.ssh/*
```

Execute the following command for each of the 3 servers to add your public key to each of the servers in the correct place. The example below is for appuser1 user on app2 server. You could do a similar thing for root user on each server.

```
cat ~/.ssh/id_rsa.pub | ssh appuser1@app2 'cat >>
authorized_keys && mkdir -p .ssh && chmod 700 .ssh &&
 cat authorized_keys >> .ssh/authorized_keys && chmod
600 .ssh/authorized_keys && rm authorized_keys'
#now set your ~/.ssh/config file in your client computer
for defaults, here is how mine looks like:
cat ~/.ssh/config
Host web
HostName web
User appuser1
IdentityFile ~/.ssh/id_rsa
Host app1
HostName app1
User appuser1
IdentityFile ~/.ssh/sid_rsa
Host app2
Hostname app2
User appuser1
IdentityFile ~/.ssh/id_rsa
```

From the terminal you could just type ssh app1 and you'll be logged in to the app1 server.

Building and configuring the web server

We will need the following software packages installed. Some of them may be handy later for monitoring and troubleshooting.

On the server named web execute the following commands as root.

```
yum install make gcc man wget
open-ssl openssl-devel telnet pcre-devel lynx tree zip
unzip
wget
http://pkgs.repoforge.org/rpmforge-release/rpmforge-rele
ase-0.5.3-1.el6.rf.x86_64.rpm
rpm -Uhv rpmforge-release*.rf.x86_64.rpm
yum install htop
```

Building from source and installing apache on the web server

Log in as appuser1 on web .

Lets download,compile and and install the latest stable version of apache.

```
su - appuser1
mkdir -p /apps/apache/src
cd /apps/apache/src
wget
http://www.us.apache.org/dist/httpd/httpd-2.4.6.tar.gz
wget http://www.us.apache.org/dist/apr/apr-1.5.0.tar.gz
wget
http://www.us.apache.org/dist/apr/apr-util-1.5.3.tar.gz
tar -xzf httpd-2.4.6.tar.gz
tar -xzf apr-1.5.0.tar.gz
tar -xzf apr-util-1.5.3.tar.gz
mv apr-1.5.0 httpd-2.4.6/srclib/apr
mv apr-util-1.5.3 httpd-2.4.6/srclib/apr-util
cd httpd-2.4.6
#The command below is split in multiple lines for
legibility(note the \ character) . Copy and pass all the
way including --with-included-apr
./configure --prefix=/apps/apache/2.4.6   \
  --enable-so                              \
  --enable-ssl                             \
  --enable-rewrite                         \
  --enable-deflate                         \
  --enable-expires                         \
  --enable-headers                         \
  --enable-proxy                           \
  --enable-proxy-http                      \
  --enable-proxy-ajp                       \
  --enable-proxy-balancer                 \
  --with-included-apr
make
make install
ln -s /apps/apache/2.4.6 /apps/apache/current
```

Note that we have not followed the linux standard to download source in /usr/local/src and install it in /usr/local/ . We did that to simplify the work as we wont need to su moving forward if we do everything in our own /apps folder.

Edit .bash_profile and add the following lines

```
export APACHE_HOME=/apps/apache/current
export PATH=$APACHE_HOME/bin:$PATH
```

```
source ~/.bash_profile
which https
/apps/apache/current/bin/httpd
#allow unprivileged users to start apache
cd $APACHE_HOME/bin
sudo chown root httpd
sudo chmod u+s httpd
```

SSL, Reverse Proxy and balancer Settings for apache

Edit $APACHE_HOME/conf/httpd.conf

append the following lines to the end of the file. Change the ip addresses bellow with the ip addresses of your app1 and app2.

It is very important to define the proxy settings within the virtual host tag even for port 80 otherwise problems may occur when browsing sites in ssl.

```
 <VirtualHost *:80>
ProxyRequests Off
ProxyBadHeader Ignore
ProxyStatus On
# note, when adding additional balancermember's, you
need to make the same
# change in your server.xml / jvmroute configuration for
the route name
<Proxy balancer://hybriscluster>
BalancerMember ajp://app1:8009 route=hash1 keepalive=On
ping=1 ttl=60
BalancerMember ajp://app2:8009 route=hash2 keepalive=On
ping=1 ttl=60
ProxySet stickysession=JSESSIONID|jsessionid
lbmethod=byrequests timeout=2
</Proxy>
ProxyPass / balancer://hybriscluster/
ProxyPassReverse / balancer://hybriscluster/
</VirtualHost>
```

Note that we will use the route identifier to route requests back to the original app server. We will

eventually need to set app1 server with a jvmRoute of "hash1" and the app2 server with a jvmRoute of "hash2" in the local.properties file. The app server will then append the correct route when generating the cookie and with the above config file apache will know to route subsequent requests from the same sessions to the appropriate server.

Performance tuning tips:

Also in server.xml on the AJP connector we would need to define the keepAliveTimeout on the AJP connector to match the ttl value we defined for the workers, maybe give tomcat a little longer since it's better if the client(apache) closes the connection. Note that the keepAliveTimeout attribute in the AJP connector is in seconds, so the value would be 61000 since the tomcat attribute is in seconds. This would mean that apache will possibly close the socket before tomcat, but if apache has not closed the socket tomcat would make sure that inactive connection is closed and not using another thread. This may be beneficial with the BIO AJP where a connection = 1 thread in tomcat, so the number of tomcat threads are reduced if they are not used.

Note also that if defining a load factor for load balancer members, you should set that to 1, or a small number. Setting the load factor to 100 for all members seems to be causing the first node in the list to take on most of the traffic and is not recommended.

Create the ssl certs:

```
cd  $APACHE_HOME/conf/
openssl req -new -x509 -days 365 -sha1 -newkey rsa:1024
\
-nodes -keyout server.key -out server.crt \
-subj '/O=Company/OU=Department/CN=www.yoursite.com'
```

Enable ssl and proxy related modules; edit httpd.conf

Search for this line and uncomment it:

include conf/extra/httpd-ssl.conf

also uncomment LoadModule ssl_module modules/mod_ssl.so and LoadModule socache_shmcb_module modules/mod_socache_shmcb.s and LoadModule slotmem_shm_module modules/mod_slotmem_shm.so

Save the file

Configure the SSL conf file:

```
vi $APACHE_HOME/conf/extra/httpd-ssl.conf
```

Add the following lines NEAR the bottom of the file, just above the '</VirtualHost>' tag.
We also tack on the X-Forwarded-Proto header so we distinguish between http and ssl requests.

```
ProxyRequests Off
ProxyBadHeader Ignore
ProxyStatus On
RequestHeader set X-Forwarded-Proto "https"
<Proxy balancer://hybriscluster>
BalancerMember ajp://app1:8009 route=hash1 keepalive=On
ping=1
BalancerMember ajp://app2:8009 route=hash2 keepalive=On
ping=1
ProxySet stickysession=JSESSIONID|jsessionid
lbmethod=byrequests timeout=2
</Proxy>
ProxyPass / balancer://hybriscluster/
ProxyPassReverse / balancer://hybriscluster/
```

```
./bin/apachectl start
```

Start apache:

You may get some warnings about parameters, these should be fine.

OPTIONAL - if you want, enable server-status .

add the following to your $APACHE_HOME/conf/extra/httpd-info.conf

```
<Location /server-status>
SetHandler server-status
Order Deny,Allow
Deny from all
Allow from 127.0.0.1
</Location>
```

restart the service

```
apachectl restart
apachectl status
```

The status will tell you something like this: *"Service Unavailable, The server is temporarily unable to service your request due to maintenance downtime or capacity problems. Please try again later."*

Do not panic, this is ok, the server was configured to forward requests using ajp to the app servers and these have not been configured yet. We proved apache listens on 443 and 80 using telnet.

Set up the Mysql Database Server and initial hybris schema

We will use web to host the mysql database in this example. Follow the instructions below.

```
mkdir -p /apps/mysql/download
cd /apps/mysql/download
wget
http://dev.mysql.com/get/Downloads/MySQL-5.6/mysql-5.6.1
4-linux-glibc2.5-x86_64.tar.gz
wget
http://dev.mysql.com/get/Downloads/Connector-J/mysql-con
nector-java-5.1.27.tar.gz
cd ..
tar -xvf
download/mysql-5.6.14-linux-glibc2.5-x86_64.tar.gz
mv mysql-5.6.14-linux-glibc2.5-x86_64 5.6.14
ln -s /apps/mysql/5.6.14 /apps/mysql/current
vi ~/.bash_profile
export MYSQL_HOME=/apps/mysql/current
export PATH=$MYSQL_HOME/bin:$PATH
source .bash_profile
which mysql
/apps/mysql/current/bin/mysql
cd $MYSQL_HOME
mkdir logs run
```

Configure my.cnf, see a working configuration in this attachment: my.cnf

```
./scripts/mysql_install_db --basedir=/apps/mysql/5.6.14
--ldata=/apps/mysql/5.6.14/data --user=appuser1
echo "$MYSQL_HOME/bin/mysqld_safe --basedir=$MYSQL_HOME
--datadir=$MYSQL_HOME/data
--log-error=$MYSQL_HOME/logs/error.log --skip-syslog
--socket=/tmp/mysql.sock
—pid-file=$MYSQL_HOME/run/mysql.pid & " >> start.sh
chmod 755 start.sh
start.sh
```

We need to create the schema and the schema owner for the hybris application. We also need to make sure the dbuser1 schema owner can log in from the app servers into the database:

```
mysql -u root
```

```
mysql> create database b2c504;
Query OK, 1 row affected (0.00 sec)
mysql> create user 'dbuser1'@'localhost' identified by
'dbuser1';
Query OK, 0 rows affected (0.00 sec)
mysql> grant all privileges on b2c504.* to
'dbuser1'@'localhost';
mysql> create user 'dbuser1'@'app1' identified by
'dbuser1';
Query OK, 0 rows affected (0.00 sec)
mysql> grant all privileges on b2c504.* to
'dbuser1'@'app1';
Query OK, 0 rows affected (0.00 sec)
mysql> create user 'dbuser1'@'app2' identified by
'dbuser1';
Query OK, 0 rows affected (0.00 sec)
mysql> grant all privileges on b2c504.* to
'dbuser1'@'app2';
Query OK, 0 rows affected (0.00 sec)
mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)
```

Building the app Servers

We will need the following software packages installed. Some of them may be handy later for monitoring and troubleshooting

```
Install the following as root:
yum install man wget unzip zip telnet
wget
http://pkgs.repoforge.org/rpmforge-release/rpmforge-rele
ase-0.5.3-1.el6.rf.x86_64.rpm
rpm -Uhv rpmforge-release*.rf.x86_64.rpm
yum install htop
su - appuser1
cd /apps
mkdir -p java/download hybris/download
```

Install Java

```
cd /apps/java/download
wget --no-cookies --no-check-certificate --header
"Cookie: gpw_e24=http%3A%2F%2Fwww.oracle.com"
"http://download.oracle.com/otn-pub/java/jdk/7u45-b18/jd
k-7u45-linux-x64.tar.gz"
cd ..
tar -xzf
download/jdk-7u45-linux-x64.tar.gz\?AuthParam\=138490090
2_44a4e09a764831dab92db73351b851f6
ln -s /apps/java/jdk1.7.0_45/ /apps/java/current
```

Add the following in the .bash_profile:

```
export JAVA_HOME=/apps/java/current
export PATH=$JAVA_HOME/bin:$PATH
```

Save and execute the .bash_profile"

```
source ~/.bash_profile
which java
/apps/java/current/bin/java
```

Install Hybris

Get the latest accelerator and build it.

```
cd /apps/hybris/src
wget --user cristian.popa@hybris.com --ask-password
https://download.hybris.com/resources/releases/5.0.4/hyb
ris-commerce-accelerator-for-b2c-5.0.4.0.zip
cd ..
unzip
src/hybris-commerce-accelerator-for-b2c-5.0.4.0.zip
mv hybris b2c-504
ln -s /apps/hybris/b2c-504 /apps/hybris/currentb2c
```

Add the platform home environment var, it may come handy. I prefer cd $PLATFORM_HOME rather than typing the full path:

```
vi ~/.bash_profile
export
PLATFORM_HOME=/apps/hybris/currentb2c/bin/platform
$source ~/.bash_profile

cd $PLATFORM_HOME
. ./setantenv.sh
ant clean all
```

At the prompts enter "production" and "3G"

Configure database connections

Edit config/local.properties file and add the following. This was copied from project.properties and adapted for my configuration, make sure you touch up on the ip address of the mysql server:

```
## Additional run-time parameters for the MySQL database
mysql.optional.tabledefs=CHARSET=utf8 COLLATE=utf8_bin
## A type of MySQL database tables, for example InnoDB
for transactional or MyISAM for non-transactional tables
mysql.tabletype=InnoDB
 db.url=jdbc:mysql://web/b2c504?useConfigs=maxPerformanc
e&characterEncoding=utf8
db.driver=com.mysql.jdbc.Driver
db.username=dbuser1
db.password=dbuser1
```

We also need to get the mysql adapter so hybris can connect to mysql database:

```
mkdir -p /apps/mysql/downloads
cd /apps/mysql/downloads
wget http://dev.mysql.com/get/Downloads/Connector-J/mysq
l-connector-java-5.1.27.tar.gz
tar -xvf mysql-connector-java-5.1.27.tar.gz
cp mysql-connector-java-5.1.27/mysql-connector-java-5.1.
27-bin.jar $PLATFORM_HOME/lib/dbdriver
```

Configure proxy and cluster settings

We would need to configure the proxy settings in local.properties.

```
proxy.ssl.port=443
tomcat.ajp.useipv=true
#proxy.http.port  and tomcat.ajp.secureport are only
needed when using 2 separate connectors for ajp (one for
originally secure requests, and one for the plain
requests, this is not a typical setup)
#proxy.http.port=80
#tomcat.ajp.secureport=8010
```

Now let's configure some cluster specific settings. In project.properties you will find documentation for cluster options below, and much more info.

> **UDP vs Jgroups**
> Please note that at the time this was written, the udp home grown protocol was best practice to set up clustering. Starting version 5.1 JGroups protocol is used to communicate cache invalidation messages between nodes. JGroups can be configured to use either TCP or UDP, the recommended method is to use JGroups over UDP. See Advanced clustering settings with JGroups for setting up a JGroups clustering over UDP.

I used multicast setup since multicast was enabled in my network. Check your adapter settings to see if multicast is on:

```
ifconfig eth0 | grep MULTICAST
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
```

For app1 server we'll have:

```
clustermode=true
cluster.id=1
cluster.maxid=2
cluster.broadcast.methods=udp
tomcat.jvm.route=hash${cluster.id}
```

For app2 server we'll have:

```
clustermode=true
cluster.id=2
cluster.maxid=2
cluster.broadcast.methods=udp
tomcat.jvm.route=hash${cluster.id}
```

We will keep the default multicast ip and UDP port from the project.properties so we won't touch those: 230.0.0.1 and 9997 have been defined in the project.properties file.

Remember that the cluster* properties have to do with cache invalidation mechanism and the session persistence settings are controlled by the tomcat.jvm.route property.

First in our config/tomcat/conf/server.xml we need to search for "Engine tag" and add the jvmroute attributeas shown below:

> **jvmRoute could also be passed as an environment variable as a tomcat option**
> There is a way to set the jvmRoute without editing the sever.xml. Note that this would not override the jvmRoute property if that is set in server.xml, but it would act as a default.
>
> Assuming in your local.properties you have defined your tomcat.jvm.route=hash${cluster.id}, then you could just add the following to your general options:
>
> tomcat.generaloptions= .....-DjvmRoute=${tomcat.jvm.route}

```
 <Engine name="Catalina" defaultHost="localhost"
jvmRoute="${tomcat.jvm.route}">
```

In the Host tag also add the following valve. This will help the app server to distinguish between http and https requests. Note that our internal proxy is the ip address of the web server.

```
<Valve

className="org.apache.catalina.valves.RemoteIpValve"

protocolHeader="x-forwarded-proto"
                />
```

```
cd $PLATFORM_HOME
. ./setantenv.sh
ant clean all
```

Verify multicast using UDP sniff. MAke sure all of the above have been configured on each app server. Open a terminal session for each app server:

```
cd resources/ant/
ant udpsniff
.........
[java] INFO [main] [SolrCoreRegistry] Initializing Solr
core components
[java] INFO [main] [DefaultTaskService] Task engine
starting up (min:10 max:10 idle:20s interval:10s)
[java] INFO [Task-master-poll] [DefaultTaskService]
Polling thread started.
[java] INFO [main] [UDPSniffer] UDP Multicast Receiver
(udpsniff) configured with the following parameter:
[java] INFO [main] [UDPSniffer] Interface:
0.0.0.0/0.0.0.0, NetworkInterface: null
[java] INFO [main] [UDPSniffer] Receiving packets from
group /230.0.0.1
[java] INFO [main] [UDPSniffer]
/10.5.0.74:[?://?|ver:4010100|15481864823210240-18340207
230448-7|99|1of1 (content: 6 bytes)]
[java] INFO [main] [UDPSniffer]
/10.5.0.77:[?://?|ver:4010100|15481864823210240-35932395
636736-2|99|1of1 (content: 29 bytes)]
[java] INFO [main] [UDPSniffer]
/10.5.0.74:[?://?|ver:4010100|15481864823210240-18340207
230448-8|99|1of1 (content: 29 bytes)]
[java] INFO [main] [UDPSniffer]
/10.5.0.77:[?://?|ver:4010100|15481864823210240-35932395
636736-3|99|1of1 (content: 6 bytes)]
[java] INFO [main] [UDPSniffer]
/10.5.0.77:[?://?|ver:4010100|15481864823210240-35932395
636736-4|99|1of1 (content: 29 bytes)]
[java] INFO [main] [UDPSniffer]
/10.5.0.74:[?://?|ver:4010100|15481864823210240-18340207
230448-9|99|1of1 (content: 29 bytes)]
```

Looks like the cluster is set up ok. So lets start the server

Start the hybris server:

```
cd $PLATFORM_HOME
hybrisserver.sh start
tail –f
$PLATFORM_HOME/../../log/tomcat/console_mmddyyyy.log
```

to see the startup log in progress

Now navigate to https://web then try http://web

You should get a logon prompt for HAC on both and you should be able to log in.

At this point let's initialize the system from hac.

Once initialization is complete, let's try the accelerators.

http://web/yacceleratorstorefront?clear=true&site=apparel-uk

The above should work.

Now lets try

https://web/yacceleratorstorefront?clear=true&site=apparel-uk

This will redirect you to the HAC logon prompt. This is because by default in Hybris 5.0.4 the spring security for the accelerator requires http channel for anonymous requests that are not data sensitive. You could adjust the security to allow any request for anonymous Pages. You could edit the ext-template/yacceleratorstorefront/web/webroot/WEB-INF/config/spring-security-config.xml and change the following section. The only changed I made was to set requires-channel="any" for the /** pattern.

```
<!-- OPEN / ANONYMOUS pages Run all other (public) pages
openly. Note that while credentials are secure, the
session id can be sniffed.
If this is a security concern, then this line should be
re-considered -->
<security:intercept-url pattern="/**"
requires-channel="any" method="POST" /> <!-- Allow posts
on either secure or insecure -->
<security:intercept-url pattern="/**"
requires-channel="any" /> <!-- Everything else should be
insecure -->
```

Now the accelerator front page should load when requested using https.

There are a few more things to consider. We initialized the system, but that happened on only one node of the cluster(meaning the media and the solr indexes exist only on one node only, because by default solr and media storage are embedded on the app server) . If you browse trough the store fronts from each of the servers individually, you'll see that on one of the app servers, the pictures would not load and clicking on various menu items would not bring any results(solr index does not exist on the server).

Media Sharing

Let's start by configuring the media to be shared between the 2 app servers. In our case I will create an nfs server in one of the nodes and I will share that location with the other node. Then we would tell hybris to use the share for media storage.

App2 will be our nfs server and app2 will be our client.

On app2:

```
su - root
yum install nfs-utils nfs-utils-lib
chkconfig nfs on
service rpcbind start
service nfs start
mkdir /hybrismedia
chown hybris /hybrismedia
vi /etc/exports
#add the following 2 lines so app1 and web are allowed
to browse the nfs share.
/hybrismedia
app1(rw,sync,no_root_squash,no_subtree_check)
/hybrismedia
web(rw,sync,no_root_squash,no_subtree_check)
#once you saved the /etc/exports
exportfs -a
```

On app1:

```
su - root
yum install nfs-utils nfs-utils-lib
mkdir /hybrismedia
chown hybris /hybrismedia
mount app2:/hybrismedia /hybrismedia
#check to make sure its mounted
df -h
#check to see if we can write to it
touch  /hybrismedia/test
#check on the other server and see if it exists in
/hybrismedia folder
#persistent mount
vi /etc/fstab
#add the following line at the end of /etc/fstab to make
sure this will be mounted automatically at the next
reboot
app2:/hybrismedia  /hybrismedia   nfs
auto,noatime,nolock,bg,nfsvers=3,intr,tcp,actimeo=1800 0
0
```

Now we would check which server(app1 or app2) has data in $PLATFORM_HOME/../../data/media/sys_master and copy all of it into the shared nfs:

cp -r /apps/hybris/currentb2c/data/media /hybrismedia

The only thing left is to tell hybris where the media is located:

In local.properties on each node:

```
media.replication.dirs=/hybrismedia
media.read.dir=/hybrismedia
```

Recompile the application on each server . Go into the storefronts on each server, media should display now. The only thing that is not working is the search.

We will move solr onto its own instance next to resolve the search issue.

Moving solr onto its own instance

We will be installing solr on tomcat.

Download and install tomcat. I installed in /apps/tomcat.

In /apps/tomcat/conf/server.xml modify the http connector to listen on the port used by solr. 8983.

```
 <Connector port="8983" protocol="HTTP/1.1"
              connectionTimeout="20000"
              redirectPort="8443" />
```

Copy solr.war file from $HYBRIS_BIN/ext-commerce/solrfacetsearch/resources/solr/server/webapps/solr.war to tomcat webapps directory.

Create a directory where your solr home will reside and change to that directory.

```
mkdir /apps/solr
cd /apps/solr
#Copy the solr files from hybris  into your SOLR_HOME
directory:
cp
$HYBRIS_BIN/ext-commerce/solrfacetsearch/resources/solr/
server/* /apps/solr
```

We would need to make a few solr libraries availableand the log4j properties config file for solr.

```
cp /apps/solr/lib/ext/* /tomcat/lib/
cp /apps/solr/resorces/log4j.properties /tomcat/lib
```

We would also need to customize our startup so it knows wher solr home is located. In /tomcat/bin create an executable file called setenv.sh with these contents.

```
#!/bin/sh
export JAVA_OPTS="-Dsolr.solr.home=/apps/solr/"
```

Now start tomcat and verify you can open solr:

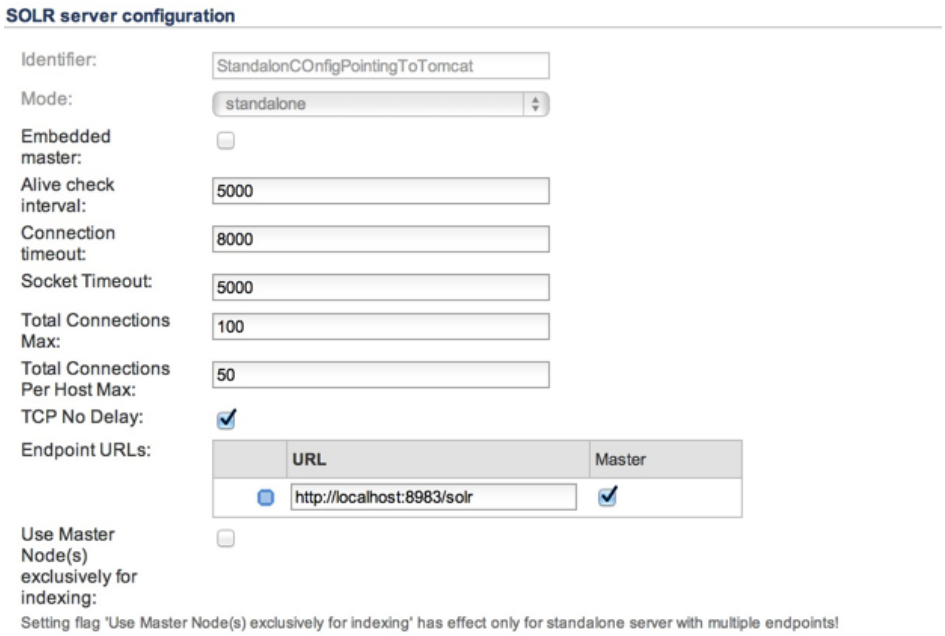
http://solrserverip:8983/solr

Now that solr runs on tomcat, we would need to configure Hybris to work with the stand alone solr instance.

First let's configure this solr instance to have a master and a slave core. Edit the replication Handler in /apps/solr/conf to look like this.

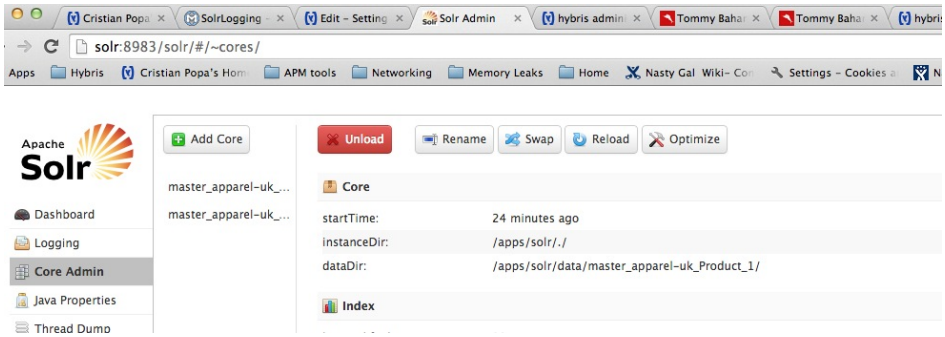
```
  <requestHandler
name="/replication" class="solr.ReplicationHandler" >
    <!--
       To enable simple master/slave replication,
uncomment one of the
       sections below, depending on whether this solr
instance should be
       the "master" or a "slave".  If this instance is a
"slave" you will
       also need to fill in the masterUrl to point to a
real machine.
    -->
       <lst name="master">
         <str name="replicateAfter">commit</str>
         <str name="replicateAfter">startup</str>
         <str
name="confFiles">schema.xml,stopwords.txt</str>
       </lst>
       <lst name="slave">
         <str
name="masterUrl">http://localhost:8983/solr</str>
         <str name="pollInterval">00:00:60</str>
       </lst>
    -->
  </requestHandler>
```

Now Go to hybris in the hmc and set your solrconfig to point to this solr instance. System -> Facet Search Config . Choose your index, in this case I selected apparel uk index. On the solr configuration tab, wight click on the existing entry in "Solr Server Configuration". Edit in new window with these settings:



Save your configuration, now go to your indexer operation wizard and perform a full index.

Go to your solr standalone url, and make sure you have 2 cores created, one for the master and one for the slave.



Now search should work in the apparel-uk store front.