App Workbench – Managing the Menagerie of Big Data Apps (https://www.bluedata.com/blog/2016/06/app-workbench-managing-big-data-apps/)

June 22, 2016
By Nanda Vijaydev (/blog/author/Nanda)

(https://twitter.com/intent/tweet?
text=App%20Workbench%20%E2%80%93%20Managing%20the%20Menagerie%20of%20Big%20Data%2(
workbench-managing-big-data-apps%2F&)

(http://www.linkedin.com/shareArticle?
mini=true&title=App%20Workbench%20%E2%80%93%20Managing%20the%20Menagerie%20of%20Big%
workbench-managing-big-data-apps/&summary=One+of+the+most+challenging+aspects+of+Big+Data+c

(mailto:?subject=Check out this blog post: https://www.bluedata.com/blog/2016/06/app-workbench-managing-big-data-apps/&body=I thought you would find this blog post interesting:
"App%20Workbench%20%E2%80%93%20Managing%20the%20Menagerie%20of%20Big%20Data%20App
Here is the link: https://www.bluedata.com/blog/2016/06/app-workbench-managing-big-data-apps/.)

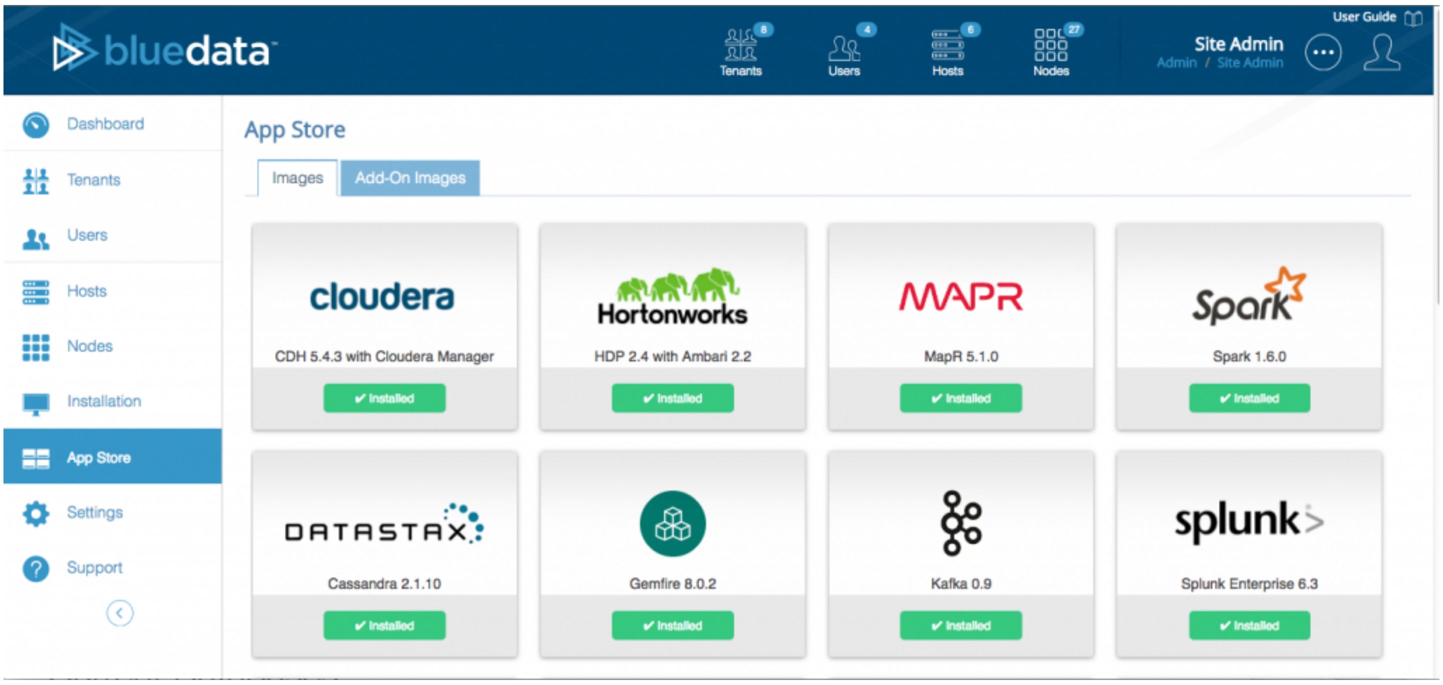
One of the most challenging aspects of Big Data deployments is keeping up with the dynamic nature of Big Data frameworks, distributions, applications, and their latest versions. The success or failure of a Big Data implementation may hinge on how well the organization handles support for the menagerie of applications and tools that data scientists, developers, analysts, and engineers want to use.

Finding the right set of Big Data expertise and implementing best practices for these applications is both difficult and time consuming. It's even more difficult due to the constant pace of Big Data innovation, as new versions and tools are released.

Speaking of new releases … we just announced the new summer release (https://www.bluedata.com/article/application-flexibility-and-choice-for-hadoop-and-spark) for the BlueData EPIC software platform today. And fortunately, our new software release introduces innovative new capabilities and functionality to help enterprises keep up with the pace of innovation in the Big Data ecosystem.

App Store and App Workbench

BlueData provides a Big-Data-as-a-Service software platform powered by Docker containers. The BlueData EPIC platform (https://www.bluedata.com/product) simplifies Big Data deployments and enables data scientists and analysts to spin up self-service Hadoop and Spark clusters, within minutes. Two key components of the solution are the "App Store (https://www.bluedata.com/product/appstore)" (which provides sample Docker images for common Big Data frameworks, applications, and tools) and the "App Workbench" (which allows administrators to manage, create, and update the preferred applications and versions for their end users).



With the new EPIC summer release (https://www.bluedata.com/article/application-flexibility-and-choice-for-hadoop-and-spark), we updated several distributions and applications that are provided as pre-configured Docker images in the App Store – including Cloudera, Hortonworks, MapR, Spark, Cassandra, Kafka, Splunk, and more – and these images can be installed via one-click deployment.

We also made significant enhancements to the App Workbench to allow administers to easily modify and update the base Docker images in their own App Store – or create new images for other applications and tools. In this blog post, I'll describe how to use the App Workbench and in doing so I'll highlight some of this new functionality.

The App Workbench focuses mainly on the following three use cases:

1. *Modify or upgrade an existing Hadoop or Spark distribution in the App Store. For example:*

- Modify Cloudera CDH version 5.4.3 and add a security patch to the base image
- Create a new CDH version 5.5.1 image starting from existing CDH 5.4.3 image

2. *Add a new application as an edge node for Hadoop or Spark with auto-provisioning. For example:*

- Splunk/Hunk could be one team's preferred tool for operational analytics; those end users may want "Splunk on Hadoop (Hunk)" as an edge node for new Hadoop deployments
- Users with a cluster dedicated for ETL may want Talend as an edge node, pre-wired for immediate use

3. *Create new images for Big Data applications and frameworks. For example:*

- Data science teams often need for tools beyond Hadoop and Spark – they'll want to add Kafka, Cassandra, and other applications like H2O for their testing, development, prototyping, and experimentation
- A user running Spark on YARN in a Hadoop cluster may be interested in trying Spark 1.6 or Spark 2.0 standalone – and they may want to add tools such as Zeppelin or Jupyter notebooks and Spark Job Server (https://www.bluedata.com/blog/2016/06/apache-spark-with-jupyter-spark-job-server/) to their Spark clusters

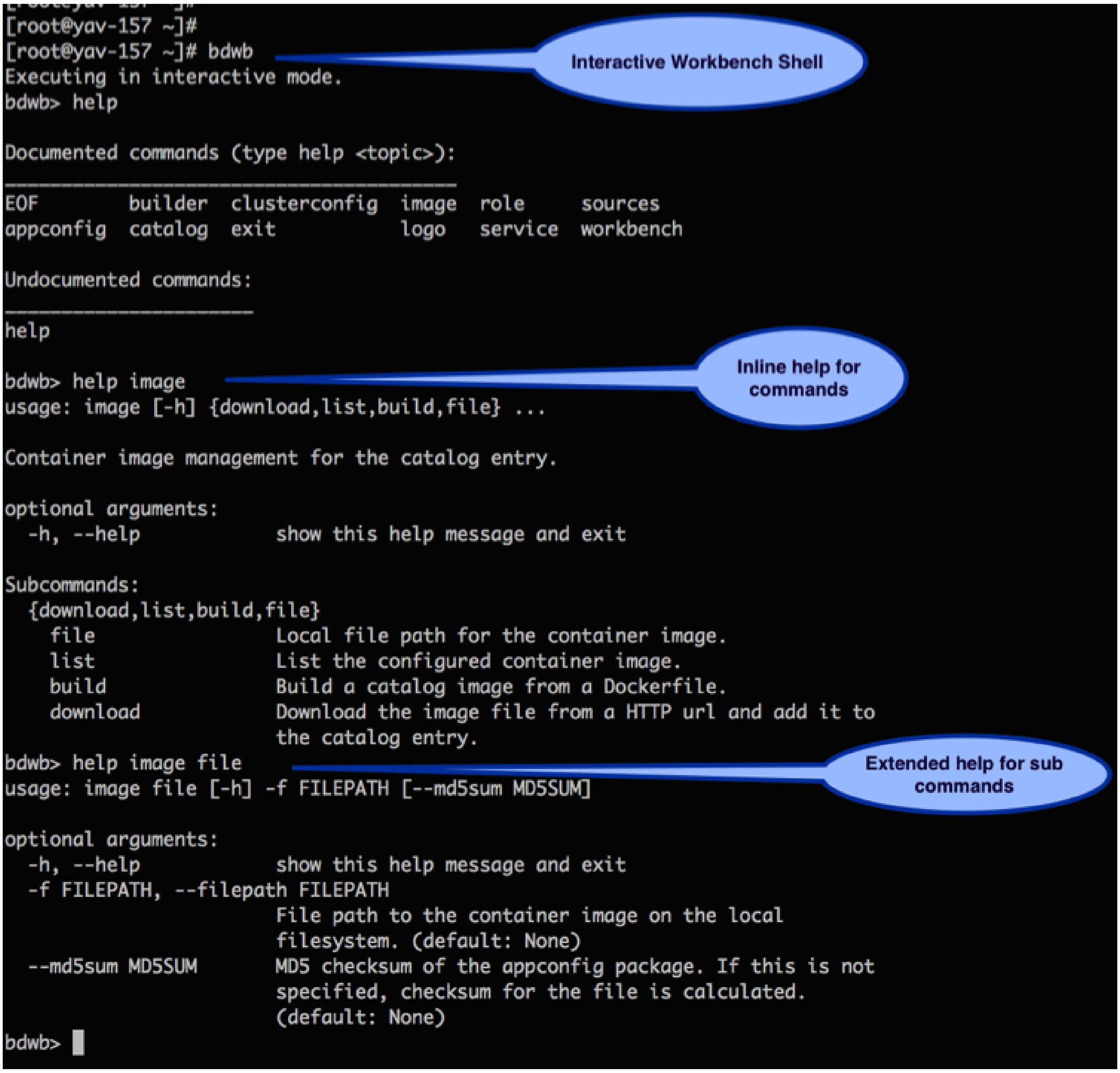
With the BlueData EPIC platform and our App Workbench, all of the above scenarios can be supported seamlessly. It's easy for an organization to maintain and run multiple applications and tools (and multiple versions) in parallel, to support a wide variety of Big Data use cases.

How the BlueData App Workbench Works

The BlueData App Workbench (aka **"bdwb"**) is a CLI framework, written in Python, that provides a rich set of APIs, macros, and a shell to:

- Create Docker images from Dockerfiles created using BlueData base images
- Orchestrate the run time environment for single and multi-node deployments
- Package and load images into the BlueData EPIC App Store (i.e. new catalog entries)

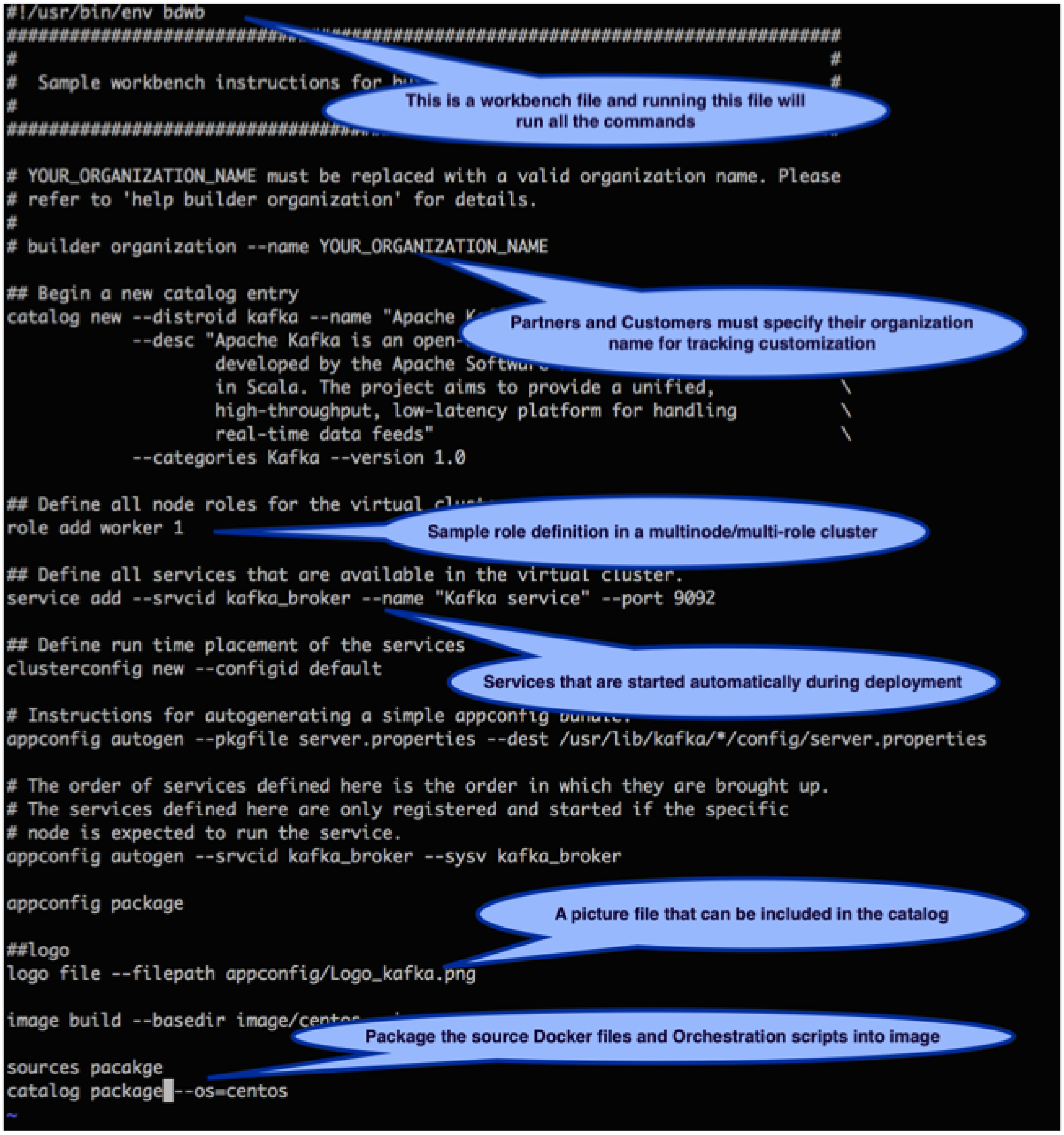
As shown in the command line screenshot below, "**bdwb**" supports an interactive shell; inline help for commands and subcommands; autocomplete and contextual help; and commands to build images, configure Docker instances, define new catalog entries and options, add a logo, and much more.



```
[root@yav-157 ~]#
[root@yav-157 ~]# bdwb
Executing in interactive mode.
bdwb> help

Documented commands (type help <topic>):
========================================
EOF        builder  clusterconfig  image  role     sources
appconfig  catalog  exit           logo   service  workbench

Undocumented commands:
======================
help

bdwb> help image
usage: image [-h] {download,list,build,file} ...

Container image management for the catalog entry.

optional arguments:
  -h, --help            show this help message and exit

Subcommands:
  {download,list,build,file}
    file                Local file path for the container image.
    list                List the configured container image.
    build               Build a catalog image from a Dockerfile.
    download            Download the image file from a HTTP url and add it to
                        the catalog entry.
bdwb> help image file
usage: image file [-h] -f FILEPATH [--md5sum MD5SUM]

optional arguments:
  -h, --help            show this help message and exit
  -f FILEPATH, --filepath FILEPATH
                        File path to the container image on the local
                        filesystem. (default: None)
  --md5sum MD5SUM       MD5 checksum of the appconfig package. If this is not
                        specified, checksum for the file is calculated.
                        (default: None)
bdwb> █
```

To add a new application or modify an existing one, users can run these commands interactively, or create a "**.wb**" file containing a series of commands. Much like a python script, running the "**.wb**" runs all the commands in the file.

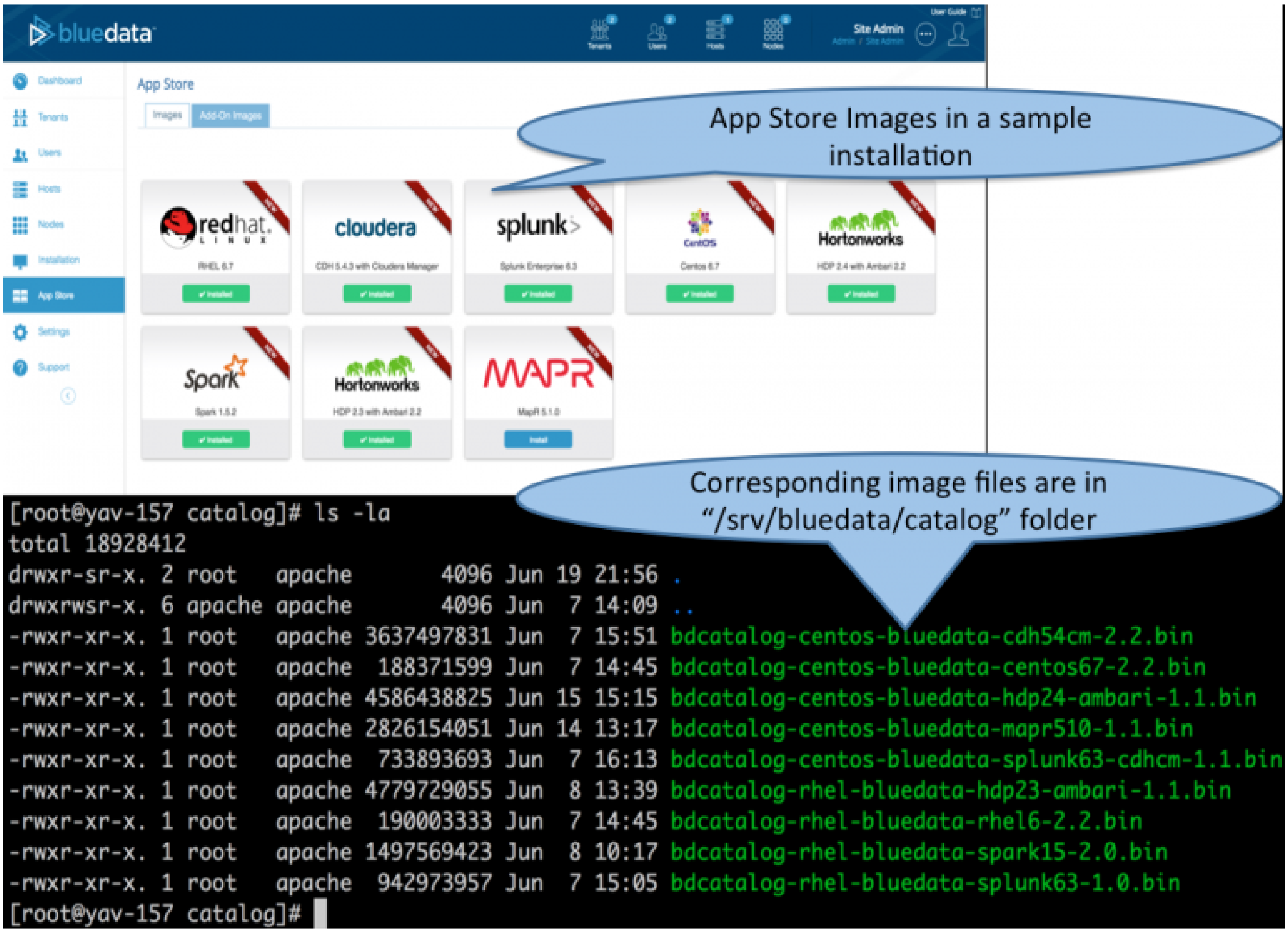
Below is a sample file. Note the "**#!/usr/bin/env bdwb**" in the first line:



```
#!/usr/bin/env bdwb
##################################################################################
#                                                                                #
#   Sample workbench instructions for bu                                         #
#
##########################################

# YOUR_ORGANIZATION_NAME must be replaced with a valid organization name. Please
# refer to 'help builder organization' for details.
#
# builder organization --name YOUR_ORGANIZATION_NAME

## Begin a new catalog entry
catalog new --distroid kafka --name "Apache K
            --desc "Apache Kafka is an open-
                    developed by the Apache Softwa
                    in Scala. The project aims to provide a unified,          \
                    high-throughput, low-latency platform for handling        \
                    real-time data feeds"                                     \
            --categories Kafka --version 1.0

## Define all node roles for the virtual clus
role add worker 1

## Define all services that are available in the virtual cluster.
service add --srvcid kafka_broker --name "Kafka service" --port 9092

## Define run time placement of the services
clusterconfig new --configid default

# Instructions for autogenerating a simple appconfig bundle.
appconfig autogen --pkgfile server.properties --dest /usr/lib/kafka/*/config/server.properties

# The order of services defined here is the order in which they are brought up.
# The services defined here are only registered and started if the specific
# node is expected to run the service.
appconfig autogen --srvcid kafka_broker --sysv kafka_broker

appconfig package

##logo
logo file --filepath appconfig/Logo_kafka.png

image build --basedir image/cento

sources pacakge
catalog package --os=centos
~
```

This is a workbench file and running this file will run all the commands
Partners and Customers must specify their organization name for tracking customization
Sample role definition in a multinode/multi-role cluster
Services that are started automatically during deployment
A picture file that can be included in the catalog
Package the source Docker files and Orchestration scripts into image

The following screenshot shows the App Store in the web-based BlueData EPIC interface, with corresponding binaries from a specific BlueData installation:



```
[root@yav-157 catalog]# ls -la
total 18928412
drwxr-sr-x. 2 root   apache       4096 Jun 19 21:56 .
drwxrwsr-x. 6 apache apache       4096 Jun  7 14:09 ..
-rwxr-xr-x. 1 root   apache 3637497831 Jun  7 15:51 bdcatalog-centos-bluedata-cdh54cm-2.2.bin
-rwxr-xr-x. 1 root   apache  188371599 Jun  7 14:45 bdcatalog-centos-bluedata-centos67-2.2.bin
-rwxr-xr-x. 1 root   apache 4586438825 Jun 15 15:15 bdcatalog-centos-bluedata-hdp24-ambari-1.1.bin
-rwxr-xr-x. 1 root   apache 2826154051 Jun 14 13:17 bdcatalog-centos-bluedata-mapr510-1.1.bin
-rwxr-xr-x. 1 root   apache  733893693 Jun  7 16:13 bdcatalog-centos-bluedata-splunk63-cdhcm-1.1.bin
-rwxr-xr-x. 1 root   apache 4779729055 Jun  8 13:39 bdcatalog-rhel-bluedata-hdp23-ambari-1.1.bin
-rwxr-xr-x. 1 root   apache  190003333 Jun  7 14:45 bdcatalog-rhel-bluedata-rhel6-2.2.bin
-rwxr-xr-x. 1 root   apache 1497569423 Jun  8 10:17 bdcatalog-rhel-bluedata-spark15-2.0.bin
-rwxr-xr-x. 1 root   apache  942973957 Jun  7 15:05 bdcatalog-rhel-bluedata-splunk63-1.0.bin
[root@yav-157 catalog]#
```

Now I'll review some of the new App Workbench features using two concrete use case examples:

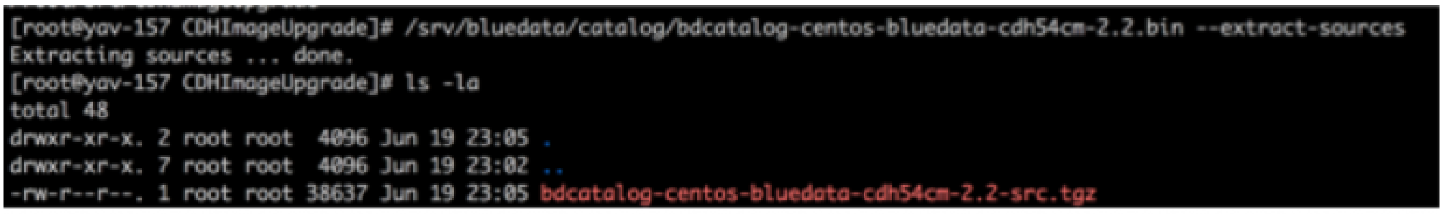
- Create a new Cloudera CDH 5.5.1 catalog entry in the App Store starting from an existing CDH 5.4.3 catalog entry
- Create a new Kafka image using a set of App Workbench commands

Create a new CDH 5.5.1 catalog entry from existing CDH 5.4.3 entry

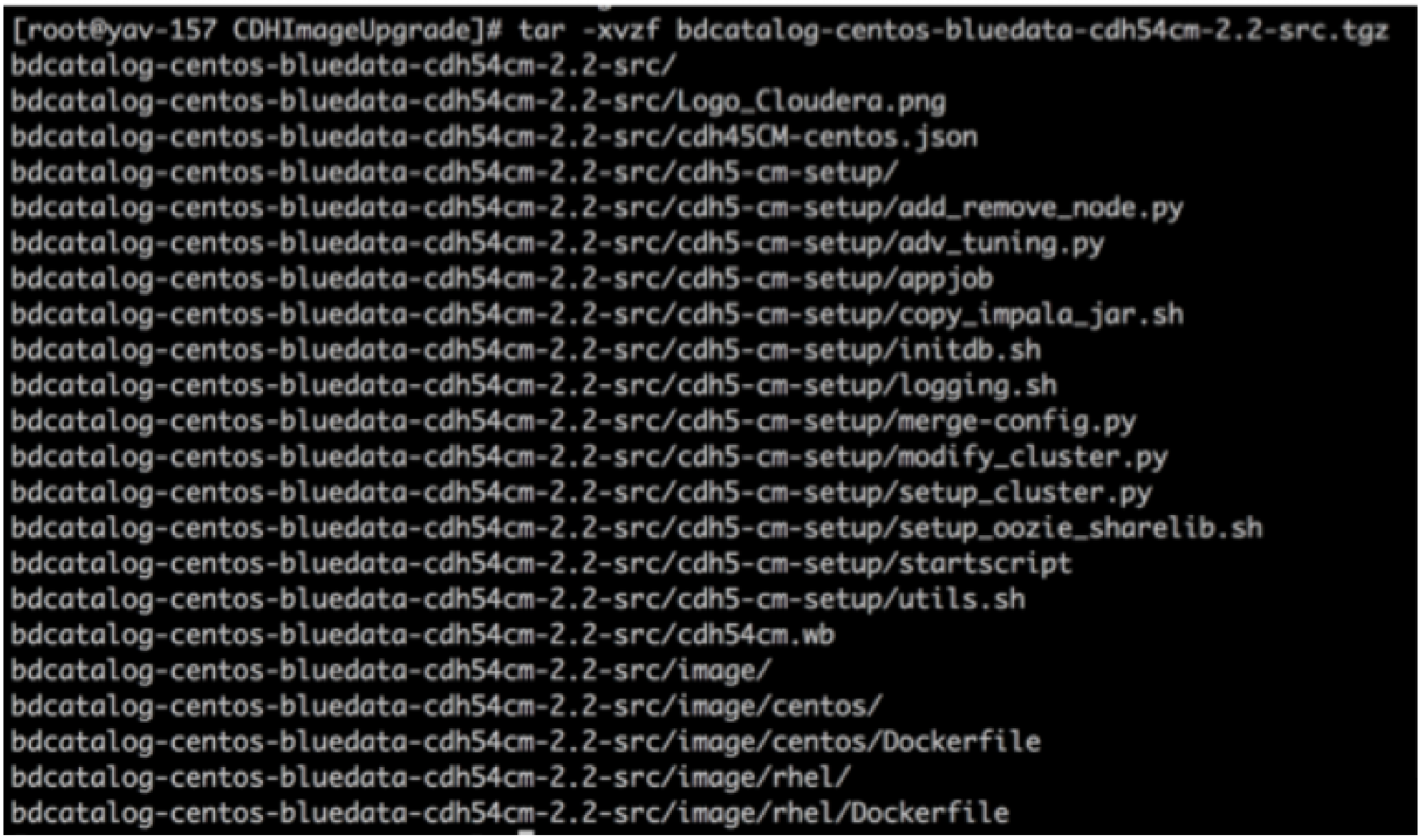
For each customer installation of BlueData EPIC, the list of distributions and applications available to end users in their App Store is controlled by the site administrator. Data scientists and developers can create new images in a development environment and then transfer the final bin file to the site administrator for installation.

The steps to create a new image of CDH 5.5.1 from an existing template CDH 5.4.3 are outlined below:

1. Create a source directory for the new image. In this example, we'll use the directory /root/src/CDHImageUpgrade
2. Extract the source files from the existing CDH 5.4.3 image into this directory. With BlueData EPIC, Dockerfiles and orchestration scripts are bundled into each image bin file as shown below.

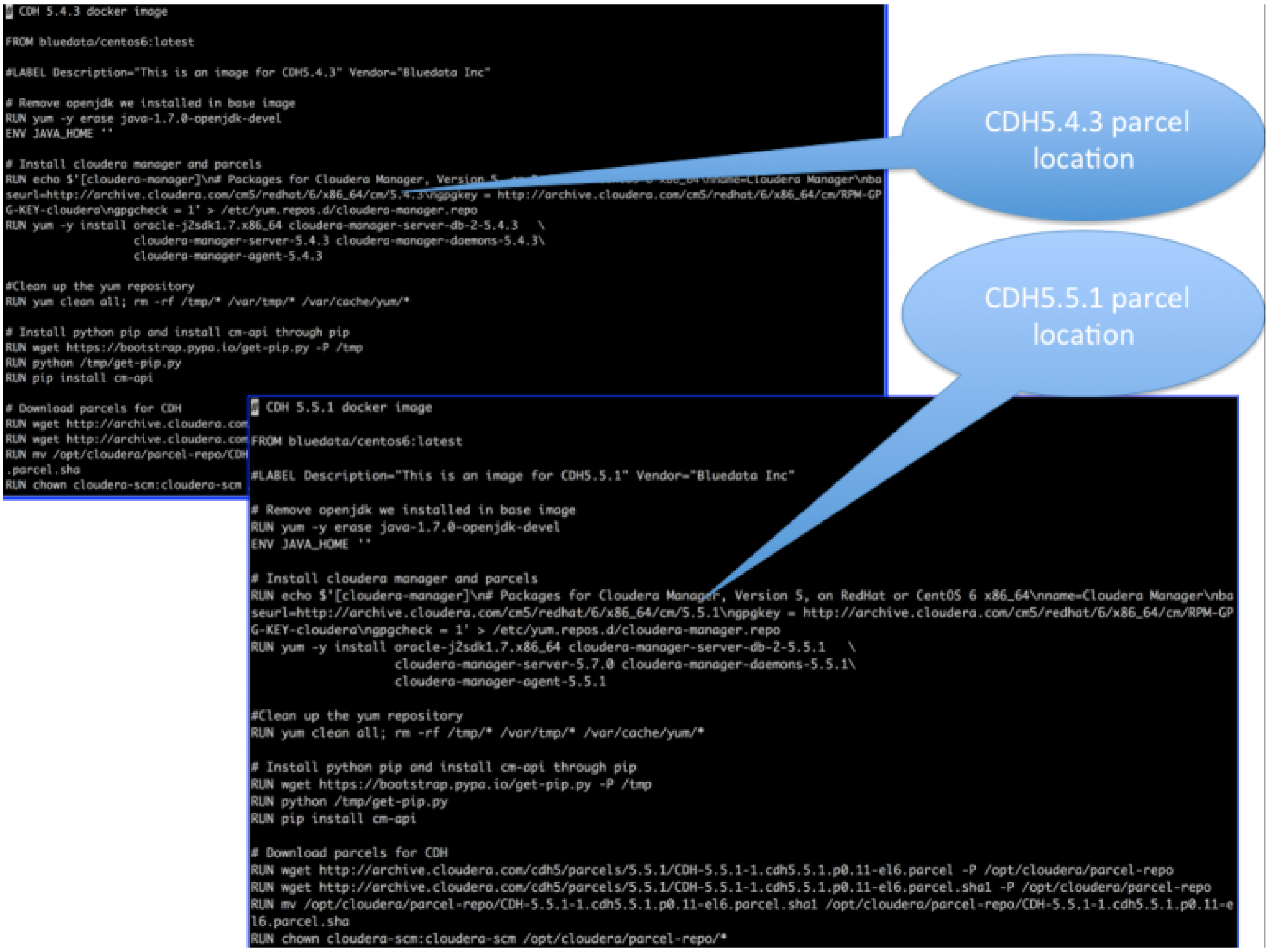


```
[root@yav-157 CDHImageUpgrade]# /srv/bluedata/catalog/bdcatalog-centos-bluedata-cdh54cm-2.2.bin --extract-sources
Extracting sources ... done.
[root@yav-157 CDHImageUpgrade]# ls -la
total 48
drwxr-xr-x. 2 root root  4096 Jun 19 23:05 .
drwxr-xr-x. 7 root root  4096 Jun 19 23:02 ..
-rw-r--r--. 1 root root 38637 Jun 19 23:05 bdcatalog-centos-bluedata-cdh54cm-2.2-src.tgz
```



```
[root@yav-157 CDHImageUpgrade]# tar -xvzf bdcatalog-centos-bluedata-cdh54cm-2.2-src.tgz
bdcatalog-centos-bluedata-cdh54cm-2.2-src/
bdcatalog-centos-bluedata-cdh54cm-2.2-src/Logo_Cloudera.png
bdcatalog-centos-bluedata-cdh54cm-2.2-src/cdh45CM-centos.json
bdcatalog-centos-bluedata-cdh54cm-2.2-src/cdh5-cm-setup/
bdcatalog-centos-bluedata-cdh54cm-2.2-src/cdh5-cm-setup/add_remove_node.py
bdcatalog-centos-bluedata-cdh54cm-2.2-src/cdh5-cm-setup/adv_tuning.py
bdcatalog-centos-bluedata-cdh54cm-2.2-src/cdh5-cm-setup/appjob
bdcatalog-centos-bluedata-cdh54cm-2.2-src/cdh5-cm-setup/copy_impala_jar.sh
bdcatalog-centos-bluedata-cdh54cm-2.2-src/cdh5-cm-setup/initdb.sh
bdcatalog-centos-bluedata-cdh54cm-2.2-src/cdh5-cm-setup/logging.sh
bdcatalog-centos-bluedata-cdh54cm-2.2-src/cdh5-cm-setup/merge-config.py
bdcatalog-centos-bluedata-cdh54cm-2.2-src/cdh5-cm-setup/modify_cluster.py
bdcatalog-centos-bluedata-cdh54cm-2.2-src/cdh5-cm-setup/setup_cluster.py
bdcatalog-centos-bluedata-cdh54cm-2.2-src/cdh5-cm-setup/setup_oozie_sharelib.sh
bdcatalog-centos-bluedata-cdh54cm-2.2-src/cdh5-cm-setup/startscript
bdcatalog-centos-bluedata-cdh54cm-2.2-src/cdh5-cm-setup/utils.sh
bdcatalog-centos-bluedata-cdh54cm-2.2-src/cdh54cm.wb
bdcatalog-centos-bluedata-cdh54cm-2.2-src/image/
bdcatalog-centos-bluedata-cdh54cm-2.2-src/image/centos/
bdcatalog-centos-bluedata-cdh54cm-2.2-src/image/centos/Dockerfile
bdcatalog-centos-bluedata-cdh54cm-2.2-src/image/rhel/
bdcatalog-centos-bluedata-cdh54cm-2.2-src/image/rhel/Dockerfile
```

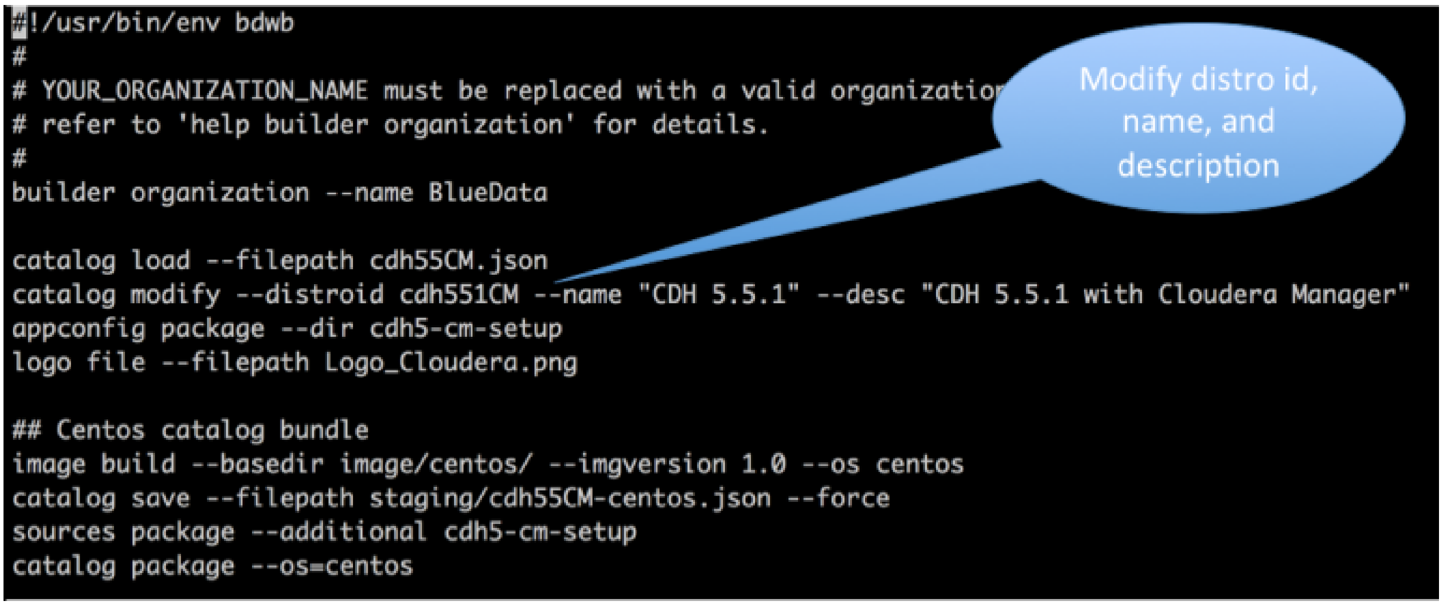
3. Modify the Dockerfile to replace the 5.4.3 parcel links with 5.5.1 parcel links.



```
# CDH 5.4.3 docker image

FROM bluedata/centos6:latest

#LABEL Description="This is an image for CDH5.4.3" Vendor="Bluedata Inc"

# Remove openjdk we installed in base image
RUN yum -y erase java-1.7.0-openjdk-devel
ENV JAVA_HOME ''

# Install cloudera manager and parcels
RUN echo $'[cloudera-manager]\n# Packages for Cloudera Manager, Version 5 [illegible] Cloudera Manager\nba
seurl=http://archive.cloudera.com/cm5/redhat/6/x86_64/cm/5.4.3\ngpgkey = http://archive.cloudera.com/cm5/redhat/6/x86_64/cm/RPM-GP
G-KEY-cloudera\ngpgcheck = 1' > /etc/yum.repos.d/cloudera-manager.repo
RUN yum -y install oracle-j2sdk1.7.x86_64 cloudera-manager-server-db-2-5.4.3   \
                   cloudera-manager-server-5.4.3 cloudera-manager-daemons-5.4.3\
                   cloudera-manager-agent-5.4.3

#Clean up the yum repository
RUN yum clean all; rm -rf /tmp/* /var/tmp/* /var/cache/yum/*

# Install python pip and install cm-api through pip
RUN wget https://bootstrap.pypa.io/get-pip.py -P /tmp
RUN python /tmp/get-pip.py
RUN pip install cm-api

# Download parcels for CDH
RUN wget http://archive.cloudera.com
RUN wget http://archive.cloudera.com
RUN mv /opt/cloudera/parcel-repo/CDH
.parcel.sha
RUN chown cloudera-scm:cloudera-scm
```

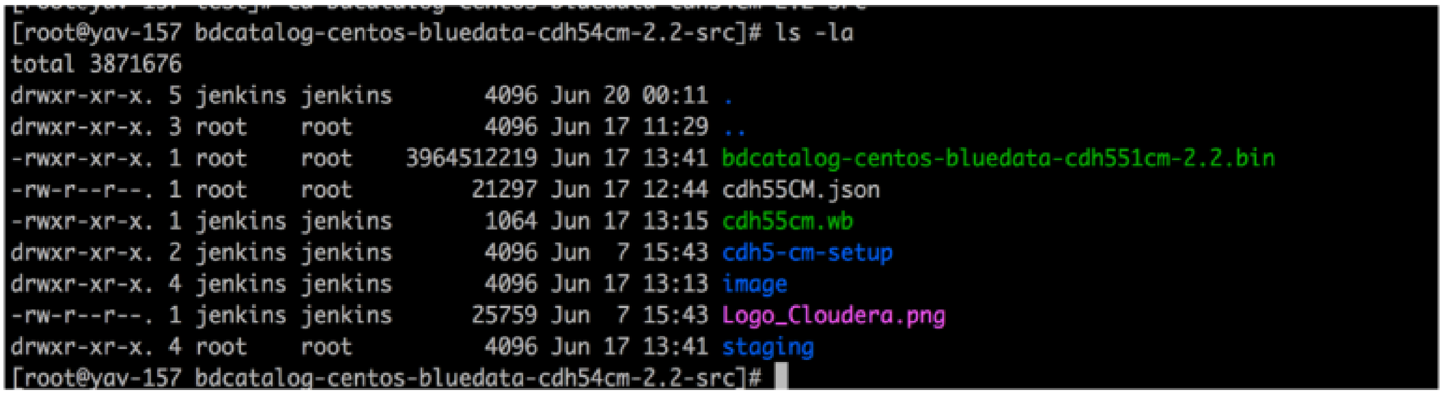
```
# CDH 5.5.1 docker image

FROM bluedata/centos6:latest

#LABEL Description="This is an image for CDH5.5.1" Vendor="Bluedata Inc"

# Remove openjdk we installed in base image
RUN yum -y erase java-1.7.0-openjdk-devel
ENV JAVA_HOME ''

# Install cloudera manager and parcels
RUN echo $'[cloudera-manager]\n# Packages for Cloudera Manager, Version 5, on RedHat or CentOS 6 x86_64\nname=Cloudera Manager\nba
seurl=http://archive.cloudera.com/cm5/redhat/6/x86_64/cm/5.5.1\ngpgkey = http://archive.cloudera.com/cm5/redhat/6/x86_64/cm/RPM-GP
G-KEY-cloudera\ngpgcheck = 1' > /etc/yum.repos.d/cloudera-manager.repo
RUN yum -y install oracle-j2sdk1.7.x86_64 cloudera-manager-server-db-2-5.5.1   \
                   cloudera-manager-server-5.7.0 cloudera-manager-daemons-5.5.1\
                   cloudera-manager-agent-5.5.1

#Clean up the yum repository
RUN yum clean all; rm -rf /tmp/* /var/tmp/* /var/cache/yum/*

# Install python pip and install cm-api through pip
RUN wget https://bootstrap.pypa.io/get-pip.py -P /tmp
RUN python /tmp/get-pip.py
RUN pip install cm-api

# Download parcels for CDH
RUN wget http://archive.cloudera.com/cdh5/parcels/5.5.1/CDH-5.5.1-1.cdh5.5.1.p0.11-el6.parcel -P /opt/cloudera/parcel-repo
RUN wget http://archive.cloudera.com/cdh5/parcels/5.5.1/CDH-5.5.1-1.cdh5.5.1.p0.11-el6.parcel.sha1 -P /opt/cloudera/parcel-repo
RUN mv /opt/cloudera/parcel-repo/CDH-5.5.1-1.cdh5.5.1.p0.11-el6.parcel.sha1 /opt/cloudera/parcel-repo/CDH-5.5.1-1.cdh5.5.1.p0.11-e
l6.parcel.sha
RUN chown cloudera-scm:cloudera-scm /opt/cloudera/parcel-repo/*
```

4. Modify the distro id, name, and description to match the CDH 5.5.1 details.



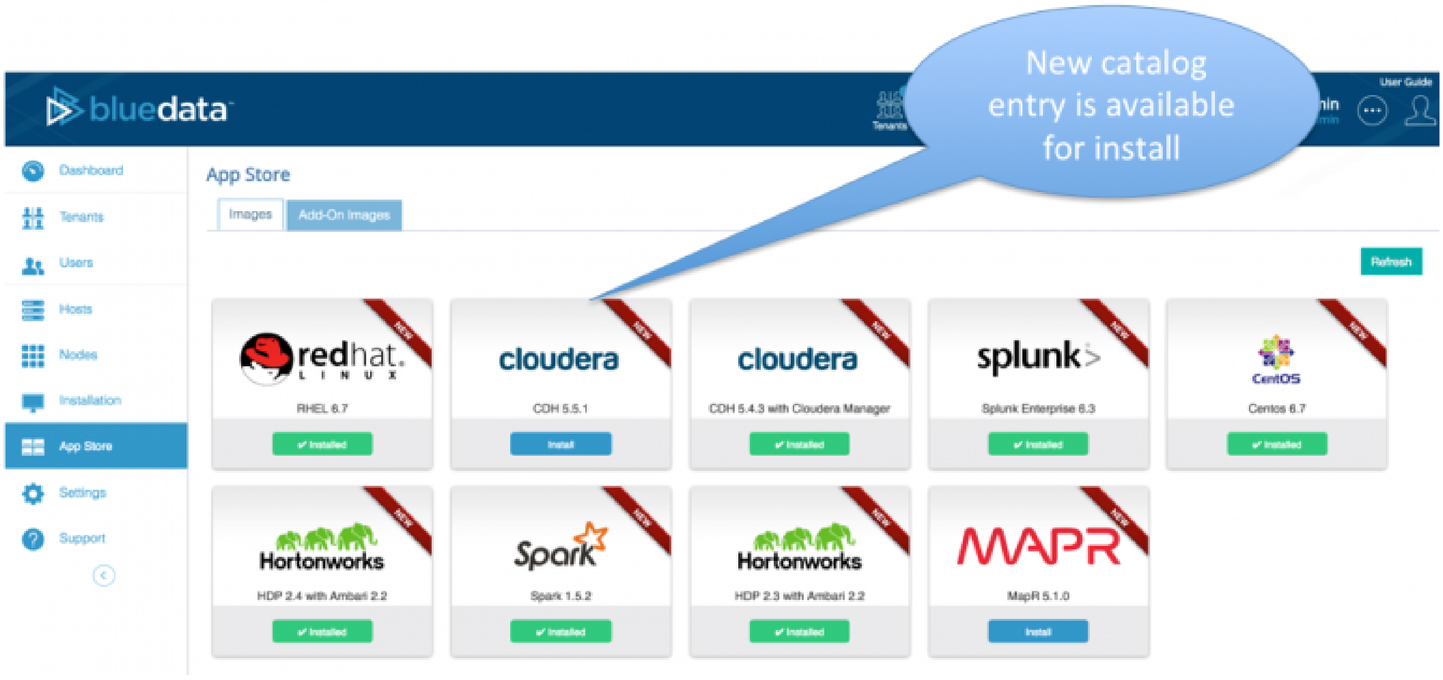
```
#!/usr/bin/env bdwb
#
# YOUR_ORGANIZATION_NAME must be replaced with a valid organizatio
# refer to 'help builder organization' for details.
#
builder organization --name BlueData

catalog load --filepath cdh55CM.json
catalog modify --distroid cdh551CM --name "CDH 5.5.1" --desc "CDH 5.5.1 with Cloudera Manager"
appconfig package --dir cdh5-cm-setup
logo file --filepath Logo_Cloudera.png

## Centos catalog bundle
image build --basedir image/centos/ --imgversion 1.0 --os centos
catalog save --filepath staging/cdh55CM-centos.json --force
sources package --additional cdh5-cm-setup
catalog package --os=centos
```

5. Create a new image file by simply running the ".wb" file (Example : > ./cdh54cm.wb). This will create the Docker image file and create the "**.bin**" file for the distribution as shown below.

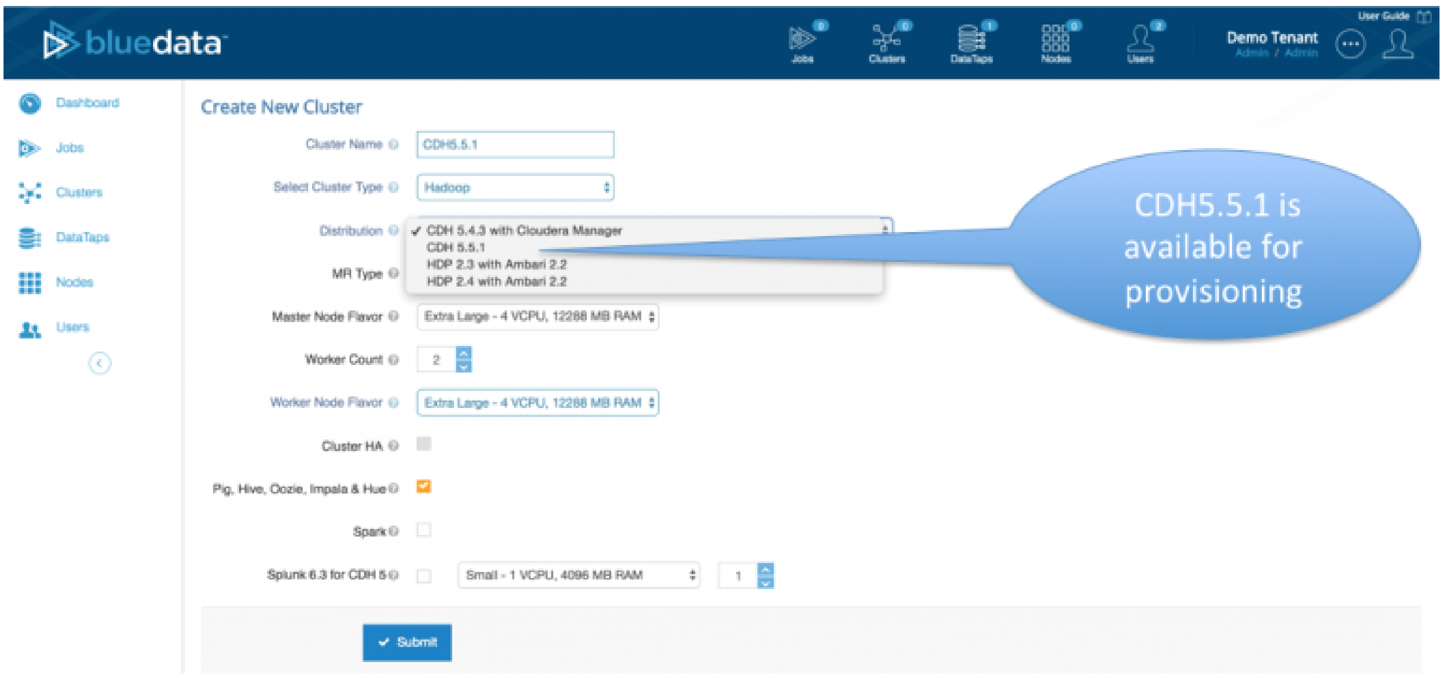


```
[root@yav-157 bdcatalog-centos-bluedata-cdh54cm-2.2-src]# ls -la
total 3871676
drwxr-xr-x. 5 jenkins jenkins       4096 Jun 20 00:11 .
drwxr-xr-x. 3 root    root          4096 Jun 17 11:29 ..
-rwxr-xr-x. 1 root    root    3964512219 Jun 17 13:41 bdcatalog-centos-bluedata-cdh551cm-2.2.bin
-rw-r--r--. 1 root    root         21297 Jun 17 12:44 cdh55CM.json
-rwxr-xr-x. 1 jenkins jenkins       1064 Jun 17 13:15 cdh55cm.wb
drwxr-xr-x. 2 jenkins jenkins       4096 Jun  7 15:43 cdh5-cm-setup
drwxr-xr-x. 4 jenkins jenkins       4096 Jun 17 13:13 image
-rw-r--r--. 1 jenkins jenkins      25759 Jun  7 15:43 Logo_Cloudera.png
drwxr-xr-x. 4 root    root          4096 Jun 17 13:41 staging
[root@yav-157 bdcatalog-centos-bluedata-cdh54cm-2.2-src]#
```

6. Copy the new bin file into the "**srv/bluedata/catalog**" directory on the BlueData EPIC controller host.



7. From the BlueData EPIC web-based user interface, click on the new image's "install" button in the App Store. Within minutes, the image will be ready for use. The create cluster screen will now display the new CDH 5.5.1 distribution as shown below:



Create New Cluster

Cluster Name	CDH5.5.1
Select Cluster Type	Hadoop
Distribution	✓ CDH 5.4.3 with Cloudera Manager CDH 5.5.1 HDP 2.3 with Ambari 2.2 HDP 2.4 with Ambari 2.2
MR Type	
Master Node Flavor	Extra Large - 4 VCPU, 12288 MB RAM
Worker Count	2
Worker Node Flavor	Extra Large - 4 VCPU, 12288 MB RAM
Cluster HA	☐
Pig, Hive, Oozie, Impala & Hue	☑
Spark	☐
Splunk 6.3 for CDH 5	☐ Small - 1 VCPU, 4096 MB RAM 1

Create a new Kafka image using a set of workbench commands

Some use cases require our customers to create new application images from scratch and add them to the App Store. To do this, the user needs to understand how the application would be manually deployed and how it typically integrates with the rest of their Big Data ecosystem.

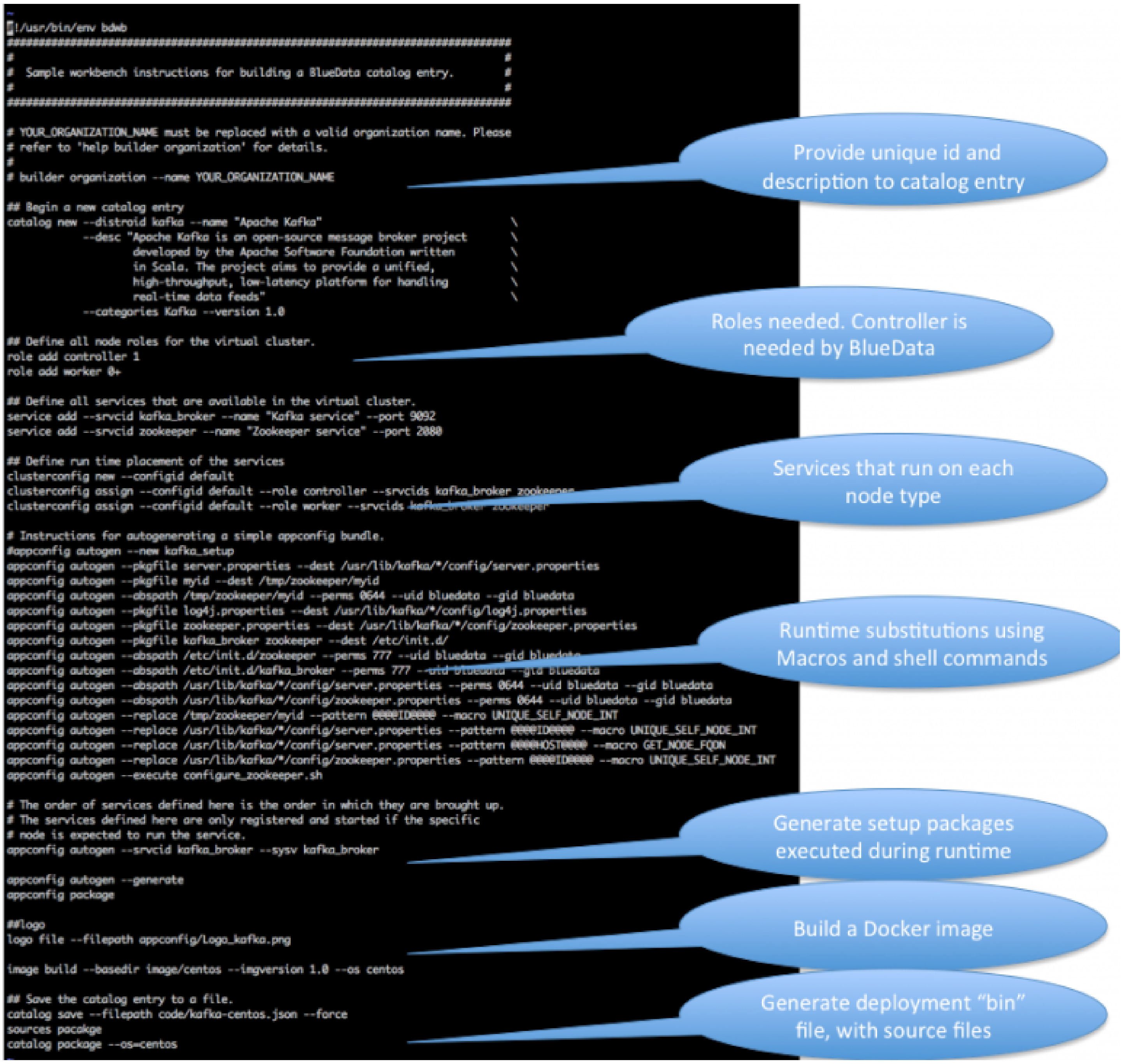
They could then use this knowledge of the application to create a Docker image and automate deployment with BlueData EPIC. In this example, we'll show how that would work for creating a new Kafka application image.

In general, developers need to follow the steps below:

1. Create utility containers in BlueData EPIC, the install and validate the application
2. Create a Dockerfile using manual instructions and generate init.d scripts for the services
3. List the commands to be executed in the "wb" file, shown below for Kafka
4. Run the wb file

The steps above generate a Docker image, setup scripts, and a bin file for application deployment. If an application requires advanced features in the setup scripts, users can always edit scripts manually and repackage them using the interactive shell or wb files.

The screenshot below demonstrates the creation of a Kafka bin file from scratch. Running this script file generates the "bin" for a Kafka 0.8.1.1 image.



```
#!/usr/bin/env bdwb
##############################################################################
#                                                                            #
#  Sample workbench instructions for building a BlueData catalog entry.      #
#                                                                            #
##############################################################################

# YOUR_ORGANIZATION_NAME must be replaced with a valid organization name. Please
# refer to 'help builder organization' for details.
#
# builder organization --name YOUR_ORGANIZATION_NAME

## Begin a new catalog entry
catalog new --distroid kafka --name "Apache Kafka"                        \
            --desc "Apache Kafka is an open-source message broker project  \
                    developed by the Apache Software Foundation written    \
                    in Scala. The project aims to provide a unified,       \
                    high-throughput, low-latency platform for handling     \
                    real-time data feeds"                                  \
            --categories Kafka --version 1.0

## Define all node roles for the virtual cluster.
role add controller 1
role add worker 0+

## Define all services that are available in the virtual cluster.
service add --srvcid kafka_broker --name "Kafka service" --port 9092
service add --srvcid zookeeper --name "Zookeeper service" --port 2080

## Define run time placement of the services
clusterconfig new --configid default
clusterconfig assign --configid default --role controller --srvcids kafka_broker zookeeper
clusterconfig assign --configid default --role worker --srvcids kafka_broker zookeeper

# Instructions for autogenerating a simple appconfig bundle.
#appconfig autogen --new kafka_setup
appconfig autogen --pkgfile server.properties --dest /usr/lib/kafka/*/config/server.properties
appconfig autogen --pkgfile myid --dest /tmp/zookeeper/myid
appconfig autogen --abspath /tmp/zookeeper/myid --perms 0644 --uid bluedata --gid bluedata
appconfig autogen --pkgfile log4j.properties --dest /usr/lib/kafka/*/config/log4j.properties
appconfig autogen --pkgfile zookeeper.properties --dest /usr/lib/kafka/*/config/zookeeper.properties
appconfig autogen --pkgfile kafka_broker zookeeper --dest /etc/init.d/
appconfig autogen --abspath /etc/init.d/zookeeper --perms 777 --uid bluedata --gid bluedata
appconfig autogen --abspath /etc/init.d/kafka_broker --perms 777 --uid bluedata --gid bluedata
appconfig autogen --abspath /usr/lib/kafka/*/config/server.properties --perms 0644 --uid bluedata --gid bluedata
appconfig autogen --abspath /usr/lib/kafka/*/config/zookeeper.properties --perms 0644 --uid bluedata --gid bluedata
appconfig autogen --replace /tmp/zookeeper/myid --pattern @@@@ID@@@@ --macro UNIQUE_SELF_NODE_INT
appconfig autogen --replace /usr/lib/kafka/*/config/server.properties --pattern @@@@ID@@@@ --macro UNIQUE_SELF_NODE_INT
appconfig autogen --replace /usr/lib/kafka/*/config/server.properties --pattern @@@@HOST@@@@ --macro GET_NODE_FQDN
appconfig autogen --replace /usr/lib/kafka/*/config/zookeeper.properties --pattern @@@@ID@@@@ --macro UNIQUE_SELF_NODE_INT
appconfig autogen --execute configure_zookeeper.sh

# The order of services defined here is the order in which they are brought up.
# The services defined here are only registered and started if the specific
# node is expected to run the service.
appconfig autogen --srvcid kafka_broker --sysv kafka_broker

appconfig autogen --generate
appconfig package

##logo
logo file --filepath appconfig/Logo_kafka.png

image build --basedir image/centos --imgversion 1.0 --os centos

## Save the catalog entry to a file.
catalog save --filepath code/kafka-centos.json --force
sources pacakge
catalog package --os=centos
```

Provide unique id and description to catalog entry
Roles needed. Controller is needed by BlueData
Services that run on each node type
Runtime substitutions using Macros and shell commands
Generate setup packages executed during runtime
Build a Docker image
Generate deployment "bin" file, with source files

That's it. It wasn't so hard, was it? By following these simple steps, our customers can have their their own App Store populated with Docker images for the latest versions of the Big Data applications and tools that their users want – providing the ultimate in flexibility, choice, and configurability.

And once these images have been added to the App Store, their data scientists and developers can instantly spin up ready-to-run clusters for these tools (with pre-built security, SSH, networking, and remote data access) using BlueData EPIC's self-service interface.

The App Store and App Workbench (https://www.bluedata.com/product/appstore) are integral components of the BlueData EPIC software platform. We use the App Workbench internally to add new applications to the App Store, and we're working with many of our ecosystem partners (https://www.bluedata.com/partners) and customers to ensure that this process is simple and extensible to support a wide variety of use cases.

Big Data is a dynamic and rapidly evolving space, with new applications and new versions emerging constantly. Here at BlueData, we're committed to helping our customers keep up with these innovations – to arm their data analysts, developers, and data science teams with all the tools they need for discovering new business insights, uncovering new opportunities, and delivering competitive advantage with Big Data analytics.

Posted in Big Data Analytics (https://www.bluedata.com/blog/category/big-data-analytics/) Data Science (https://www.bluedata.com/blog/category/data-science/) Tagged Big Data Applications (https://www.bluedata.com/blog/tag/big-data-applications/) Data Science (https://www.bluedata.com/blog/tag/data-science/)

Apache Spark Integrated with Jupyter and Spark Job Server (https://www.bluedata.com/blog/2016/06/apache-spark-with-jupyter-spark-job-server/)
Big-Data-as-a-Service. On-Prem or in the Cloud. It’s BDaaS (https://www.bluedata.com/blog/2016/06/big-data-as-a-service-on-prem-or-cloud-bdaas/)