Best Practices for Running Kafka on Docker Containers

Nanda Vijaydev, BlueData

Kafka Summit San Francisco

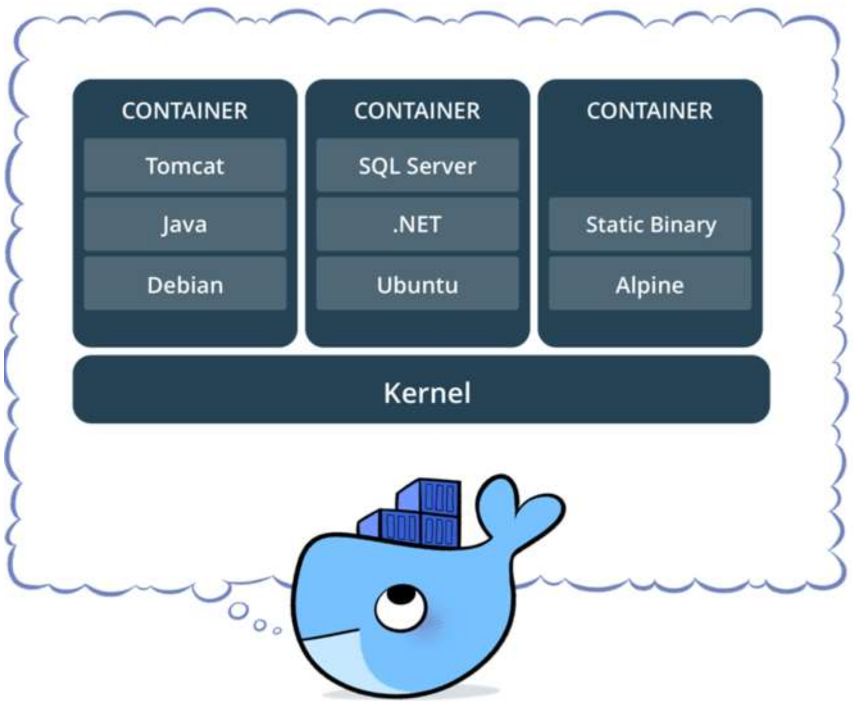
August 28, 2017

Agenda

- What is Docker?
- Deploying services on Docker
- Messaging systems (Kafka) on Docker: Challenges
- How We Did it: Lessons Learned
- Key Takeaways for Running Kafka on Docker
- Q & A

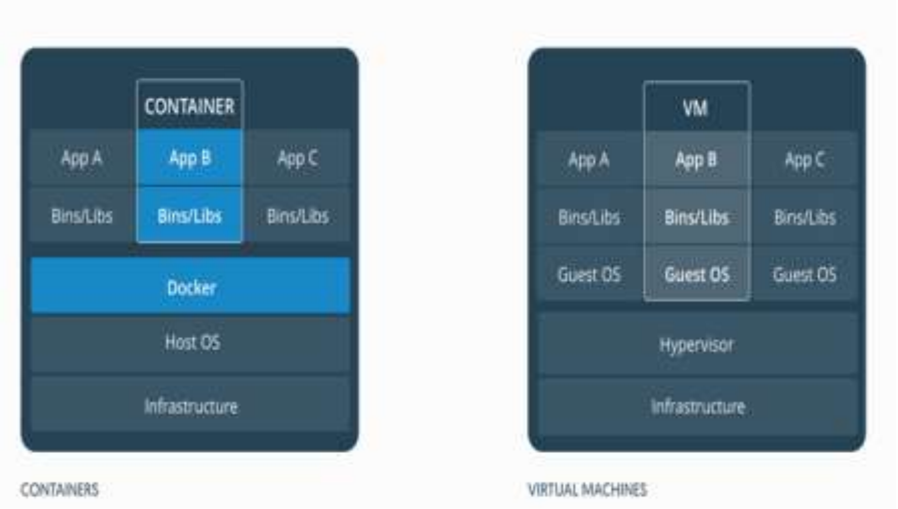
bluedata

What is a Docker Container?



- Lightweight, stand-alone, executable package of software to run specific services
- Runs on all major Linux distributions
- On any infrastructure including VMs, bare-metal, and in the cloud
- Package includes code, runtime, system libraries, configurations, etc.
- Run as an isolated process in user space

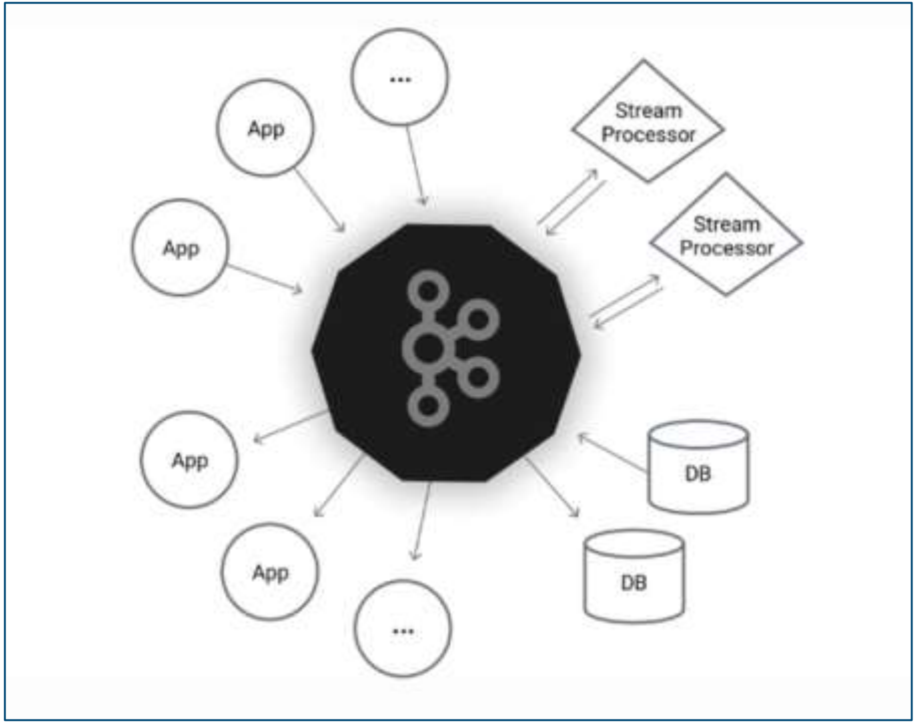
Docker Containers vs. Virtual Machines



- Unlike VMs, containers virtualize OS and not hardware
- More portable and efficient
- Abstraction at the app layer that packages app and dependencies
- Multiple containers share the base kernel
- Take up less space and start almost immediately

bluedata

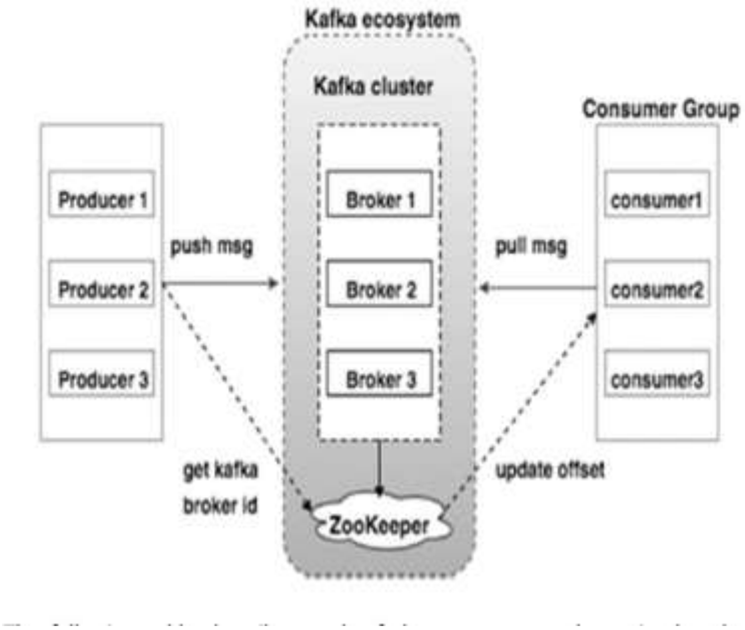
Kafka, Producers, and Consumers



- Independent services that send/receive messages over Kafka
- Can be written in many languages
- Purpose-built for specific actions
- Mostly operate on high frequency events and data
- Availability and scalability are important

bluedata

Considerations for Kafka Deployment



- Multiple services; each with its own requirements
- Single QOS for related containers and services (CPU & Memory)
- Storage – Local persistence & External Volumes
- Service monitoring and dependency management

How We Did It: Design Decisions I

- Run Kafka (e.g. Confluent distribution) and related services and tools / applications unmodified
  - Deploy all services that run on a single bare-metal host in a single container
- Multi-tenancy support is key
  - Network and storage security
- Clusters of containers span physical hosts

How We Did It: Sample Dockerfile

```
# Confluent Kafka 3.2.1 docker image
FROM bluedata/centos7:latest
#Install java 1.8
RUN yum -y install java-1.8.0-openjdk-devel
#Download and extract Kafka installation tar file
RUN mkdir /usr/lib/kafka;curl -s http://packages.confluent.io/archive/3.2/confluent-3.2.1-2.11.tar.gz | tar xz -C /usr/lib/kafka/
##Create necessary directories for Kafka and Zookeeper to run
RUN mkdir /var/lib/zookeeper
…….
```

How We Did It: Design Decisions II

- Images built to “auto-configure” themselves at time of instantiation
  - Not all instances of a single image run the same set of services when instantiated
    - Zookeeper vs. Broker cluster nodes
  - Ability to scale on demand

bluedata

How We Did It: Deployment Configuration

```
#!/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
builder organization --name BlueData
## Begin a new catalog entry
catalog new --distroid confluent-kafka --name "Confluent Kafka 3.2.1"                  \
        --desc "The free, open-source streaming platform (Enterprise edition) based on Apache       \
                Kafka. Confluent Platform is the best way to get started        \
                with real-time data streams."                                   \
        --categories Kafka --version 4.0


## Define all node roles for the virtual cluster.
role add broker 1+
role add zookeeper 1+
role add schemareg 1+
role add gateway 0+
```

```
## Define all services that are available in the virtual cluster.
service add --srvcid kafka-broker --name "Kafka Broker service" --port 9092
service add --srvcid zookeeper --name "Zookeeper service" --port 2181
service add --srvcid schema-registry --name "Schema-registry service" --port 8081
service add --srvcid control-center --name "Control center service" --port 9021

## Dev Configuration. Multiple services are placed on same container
clusterconfig new --configid default
clusterconfig assign --configid default –role gateway –srvcids gateway   control-center
clusterconfig assign --configid default --role broker –srvcids kafka-broker  schema-registry
clusterconfig assign --configid default --role zookeeper --srvcids kafka-broker zookeeper

## Prod Configuration. Services run on dedicated nodes with special attributes
clusterconfig new --configid production
clusterconfig assign --configid production --role broker --srvcids kafka-broker
clusterconfig assign --configid production --role zookeeper --srvcids zookeeper
clusterconfig assign --configid production --role schemareg --srvcids schemareg
clusterconfig assign --configid production --role gateway --srvcids control-center
```

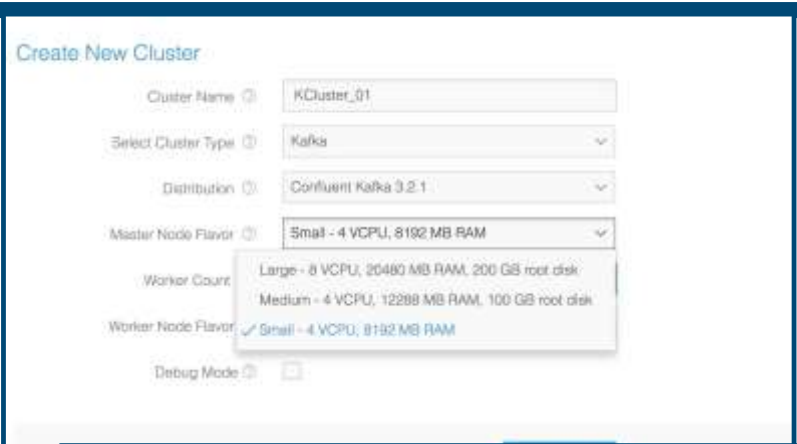
How We Did It: Deployment Configuration

```
#Configure your docker nodes with appropriate run time values

appconfig autogen --replace /tmp/zookeeper/myid –pattern @@ID@@ --macro UNIQUE_SELF_NODE_INT
appconfig autogen --replace /usr/lib/kafka/etc/kafka/server.properties  –pattern @@HOST@@ --macro GET_NODE_FQDN
appconfig autogen --replace /usr/lib/kafka/etc/kafka/server.properties  –pattern @@zookeeper.connet@@ --macro ZOOKEEPER_SERVICE_STRING
```

```
#Start services in the order specified
REGISTER_START_SERVICE_SYSV zookeeper
REGISTER_START_SERVICE_SYSV kafka-broker  –wait zookeeper
REGISTER_START_SERVICE_SYSV schema-registry  –wait zookeeper
```

How We Did It: Resource Allocation



Create New Cluster

Cluster Name: KCluster_01

Select Cluster Type: Kafka

Distribution: Confluent Kafka 3.2.1

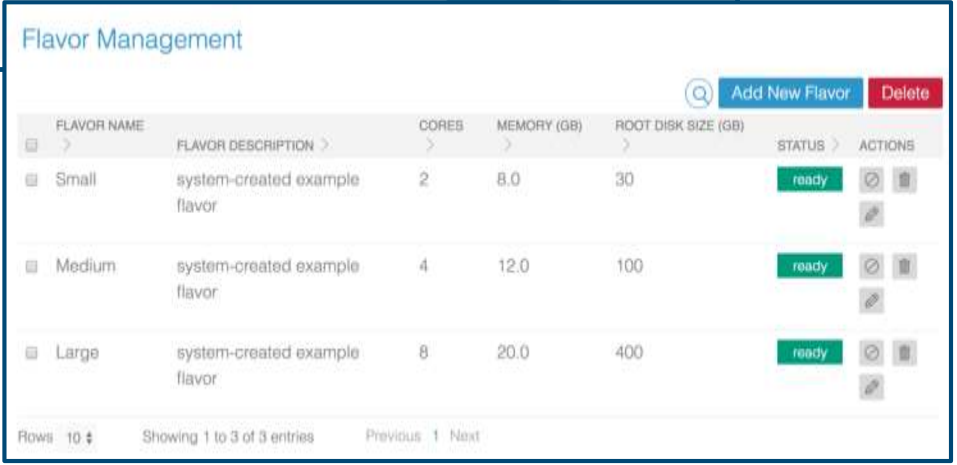
Master Node Flavor: Small - 4 VCPU, 8192 MB RAM

- Large - 8 VCPU, 20480 MB RAM, 200 GB root disk
- Medium - 4 VCPU, 12288 MB RAM, 100 GB root disk
- ✓ Small - 4 VCPU, 8192 MB RAM

Worker Count

Worker Node Flavor

Debug Mode □



Flavor Management

□	FLAVOR NAME	FLAVOR DESCRIPTION	CORES	MEMORY (GB)	ROOT DISK SIZE (GB)	STATUS	ACTIONS
□	Small	system-created example flavor	2	8.0	30	ready	
□	Medium	system-created example flavor	4	12.0	100	ready	
□	Large	system-created example flavor	8	20.0	400	ready	

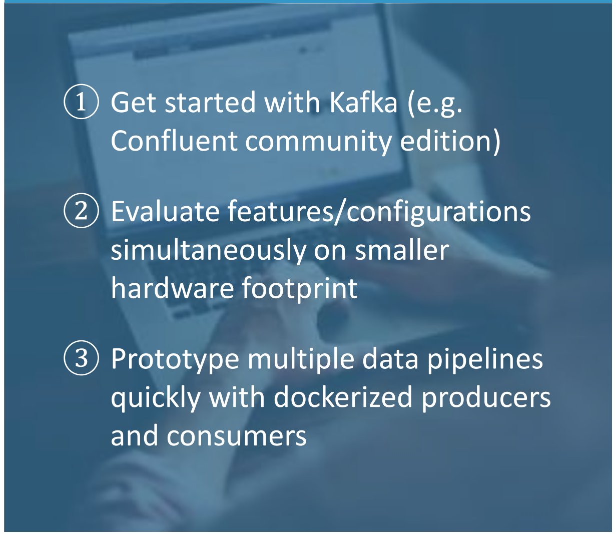
Rows 10 Showing 1 to 3 of 3 entries Previous 1 Next

- Users to choose “flavors” while launching containers
- Storage heavy containers can have more disk space
- vCPUs * n = cpu-shares
- No over-provisioning of memory

bluedata

Kafka On Docker Use Cases

Prototyping



① Get started with Kafka (e.g. Confluent community edition)

② Evaluate features/configurations simultaneously on smaller hardware footprint

③ Prototype multiple data pipelines quickly with dockerized producers and consumers

Exploring the Value of Kafka

Departmental

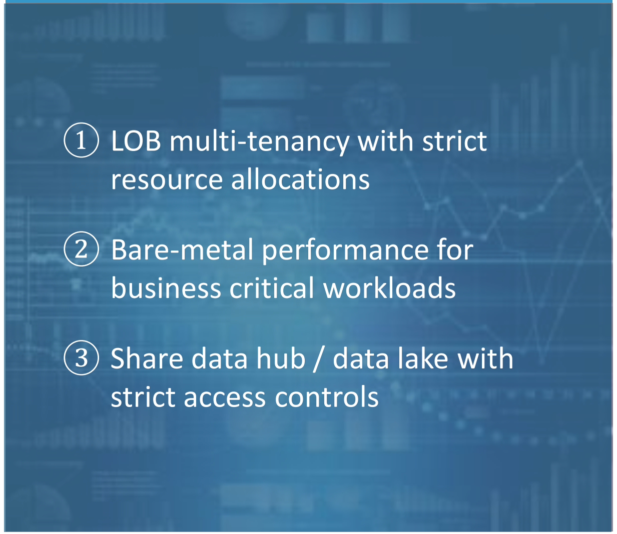
① Spin up dev/test clusters with replica image of production

② QA/UAT using production configuration without re-inventing the wheel

③ Offload specific users and workloads from production

Initial Departmental Deployments

Enterprise



① LOB multi-tenancy with strict resource allocations

② Bare-metal performance for business critical workloads

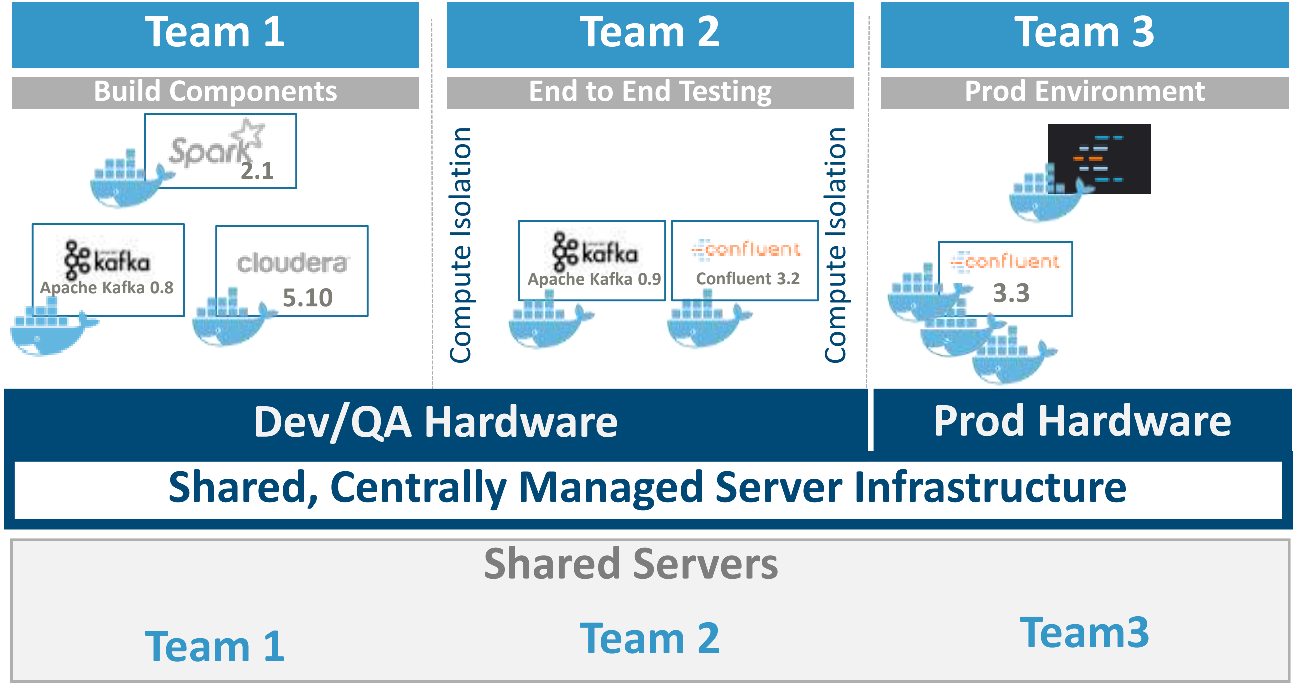
③ Share data hub / data lake with strict access controls

Enterprise-Wide, Mission-Critical Deployments

bluedata

Multi-Tenant Deployment

Multiple distributions, services, tools on shared, cost-effective infrastructure



Team 1

Build Components

Team 2

End to End Testing

Team 3

Prod Environment

Dev/QA Hardware

Prod Hardware

Shared, Centrally Managed Server Infrastructure

Shared Servers

Team 1

Team 2

Team3

Multiple teams or business groups

Evaluate different Kafka use cases (e.g. producers, consumers, pipelines)

Use different services & tools (e.g. Broker, Zookeeper, Schema Registry, API Gateway)

Use different distributions of standalone Kafka and/or Hadoop

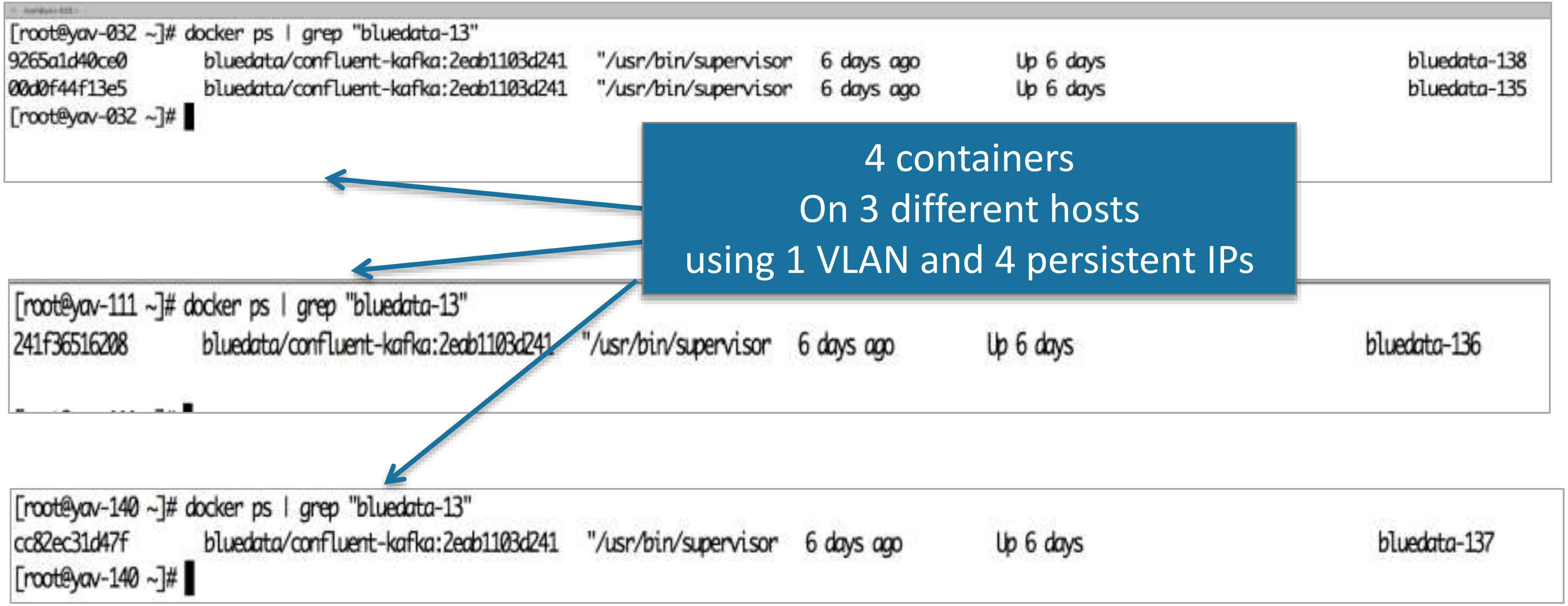
BlueData EPIC software platform

Shared server infrastructure with node labels

Shared data sets for HDFS access

bluedata

Multi-Host Kafka Deployment



```
[root@yav-032 ~]# docker ps | grep "bluedata-13"
9265a1d40ce0        bluedata/confluent-kafka:2eab1103d241   "/usr/bin/supervisor   6 days ago          Up 6 days                               bluedata-138
00d0f44f13e5        bluedata/confluent-kafka:2eab1103d241   "/usr/bin/supervisor   6 days ago          Up 6 days                               bluedata-135
[root@yav-032 ~]#
```

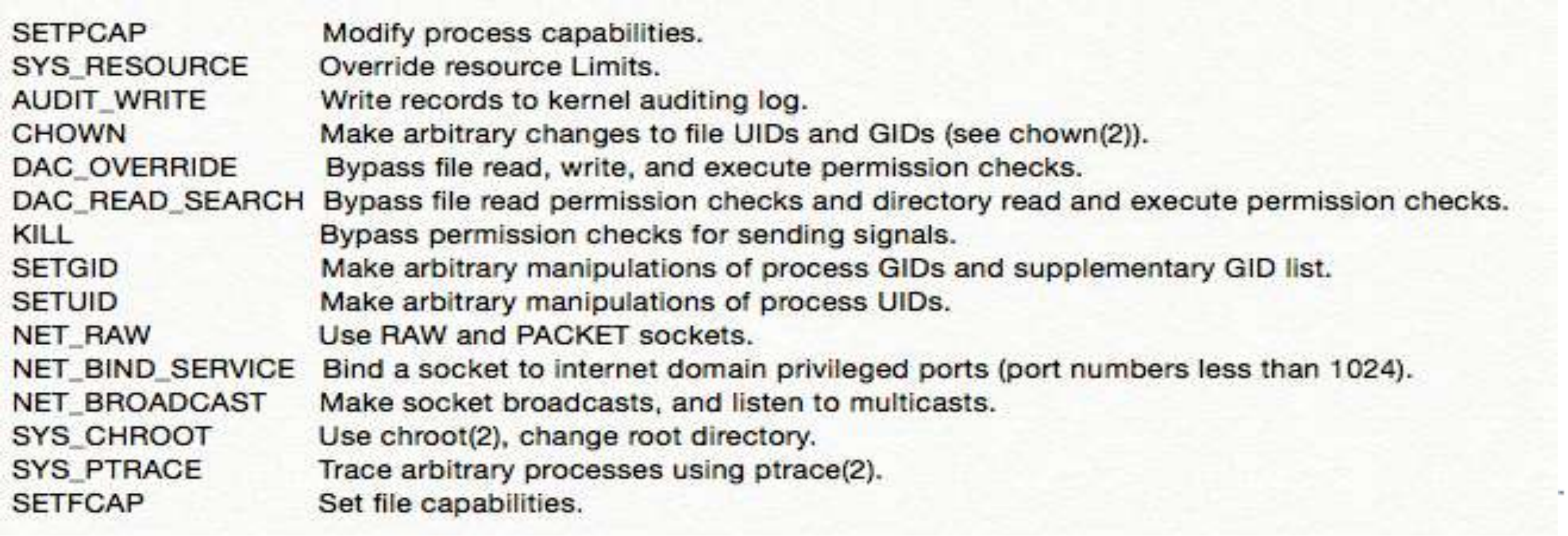
4 containers
On 3 different hosts
using 1 VLAN and 4 persistent IPs

```
[root@yav-111 ~]# docker ps | grep "bluedata-13"
241f36516208        bluedata/confluent-kafka:2eab1103d241   "/usr/bin/supervisor   6 days ago          Up 6 days                               bluedata-136
```

```
[root@yav-140 ~]# docker ps | grep "bluedata-13"
cc82ec31d47f        bluedata/confluent-kafka:2eab1103d241   "/usr/bin/supervisor   6 days ago          Up 6 days                               bluedata-137
[root@yav-140 ~]#
```

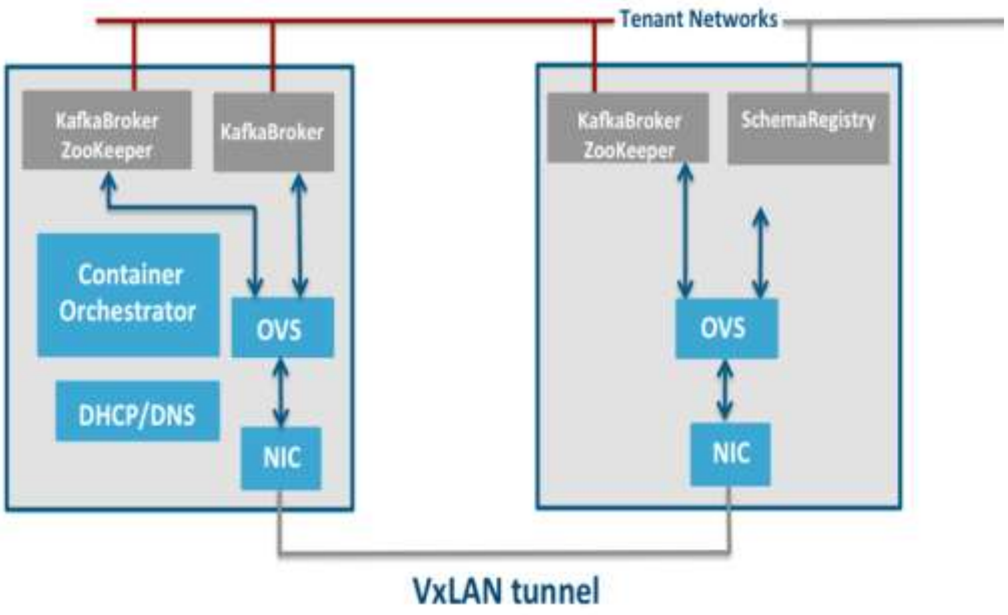
How We Did It: Security Considerations

- Security is essential since containers and host share one kernel
  - Non-privileged containers
- Achieved through layered set of capabilities
- Different capabilities provide different levels of isolation and protection
- Add “capabilities” to a container based on what operations are permitted



SETPCAP	Modify process capabilities.
SYS_RESOURCE	Override resource Limits.
AUDIT_WRITE	Write records to kernel auditing log.
CHOWN	Make arbitrary changes to file UIDs and GIDs (see chown(2)).
DAC_OVERRIDE	Bypass file read, write, and execute permission checks.
DAC_READ_SEARCH	Bypass file read permission checks and directory read and execute permission checks.
KILL	Bypass permission checks for sending signals.
SETGID	Make arbitrary manipulations of process GIDs and supplementary GID list.
SETUID	Make arbitrary manipulations of process UIDs.
NET_RAW	Use RAW and PACKET sockets.
NET_BIND_SERVICE	Bind a socket to internet domain privileged ports (port numbers less than 1024).
NET_BROADCAST	Make socket broadcasts, and listen to multicasts.
SYS_CHROOT	Use chroot(2), change root directory.
SYS_PTRACE	Trace arbitrary processes using ptrace(2).
SETFCAP	Set file capabilities.

How We Did It: Network Architecture



- Connect containers across hosts
- Persistence of IP address across container restart
- DHCP/DNS service required for IP allocation and hostname resolution
- Deploy VLANs and VxLAN tunnels for tenant-level traffic isolation

Storage – Internal To Host File System

Data Volume

- A directory on host FS
- Data not deleted when container is deleted

Device Mapper Storage Driver

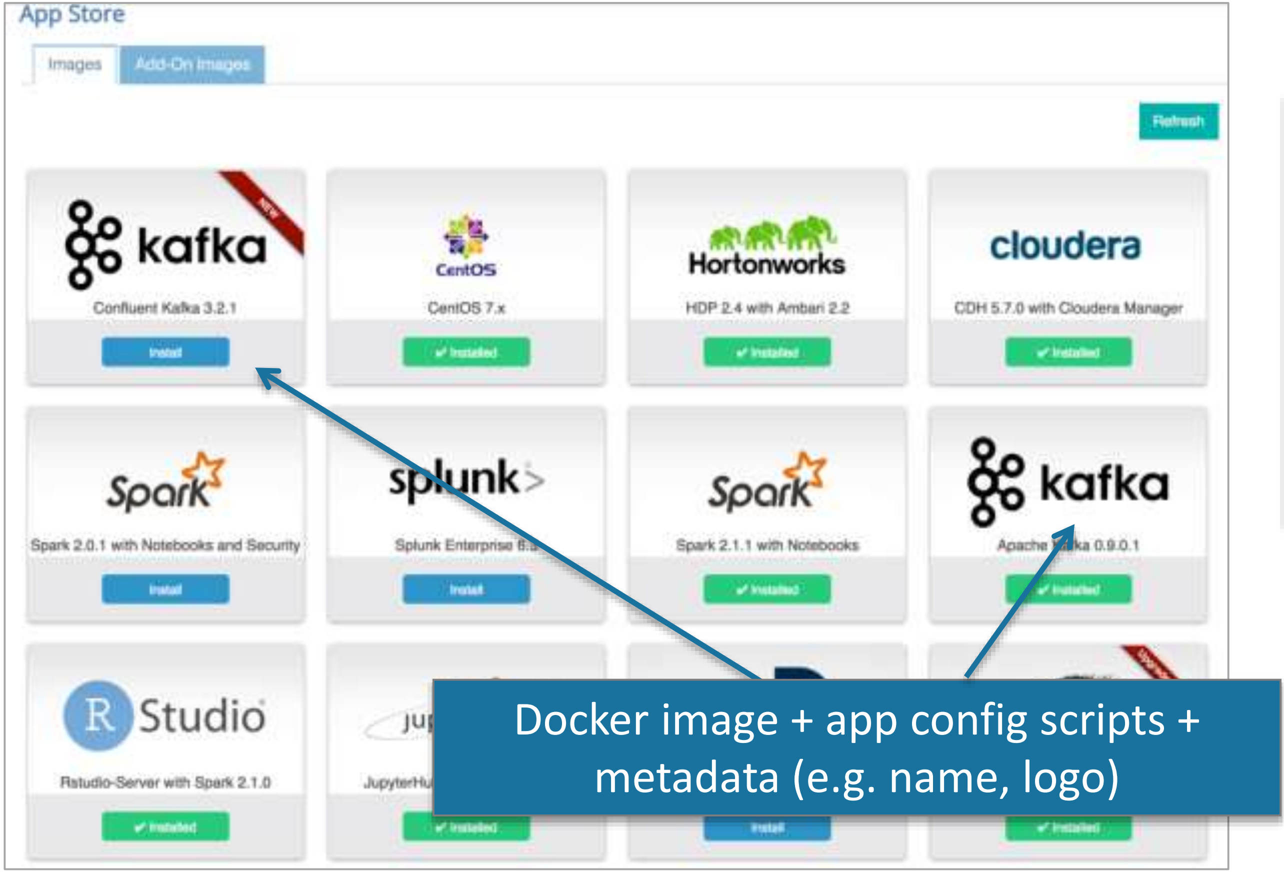
- Default – OverlayFS
- We use direct-lvm thinpool with devicemapper
- Data is deleted with container

bluedata

Storage - External Volumes

- Storage is external to host FS, accessed over the network
- Separates container from storage
- Cloud providers have storage services such as S3, EBS
- You can also connect to HDFS, NFS, Gluster
- Services such as REX-Ray provide external volume support

App Store for Kafka, Spark, & More

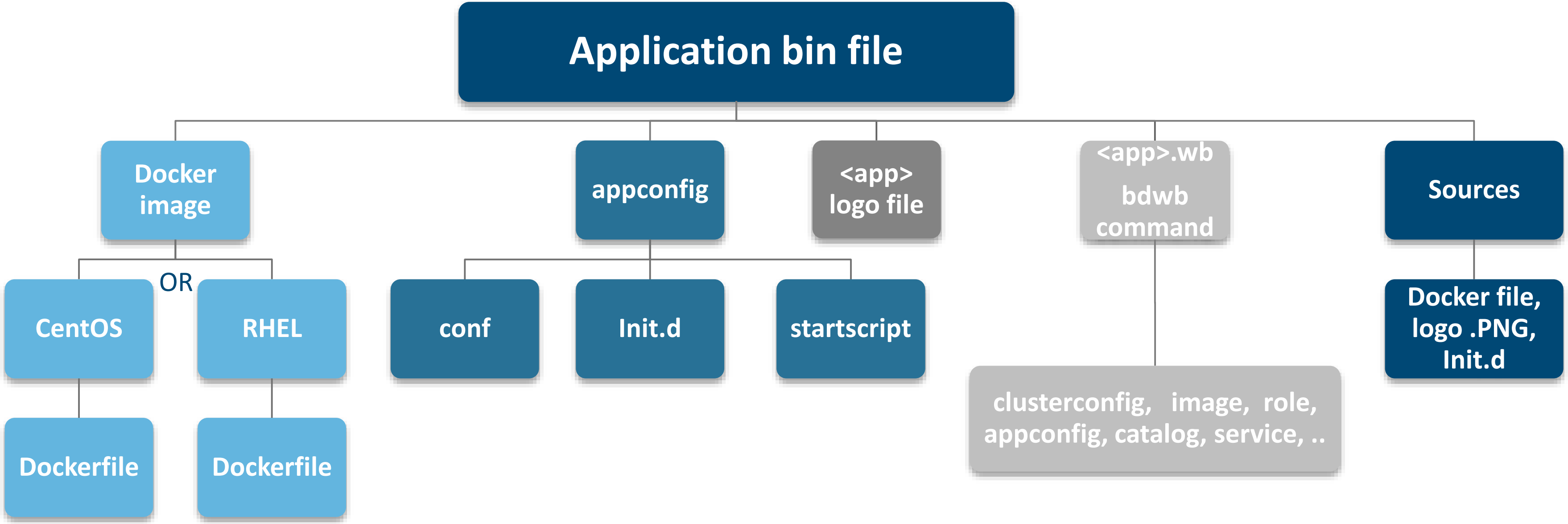


Docker image + app config scripts + metadata (e.g. name, logo)

Pre-built images, or author your own Docker app images with our App Workbench

bluedata

BlueData Application Image (.bin file)



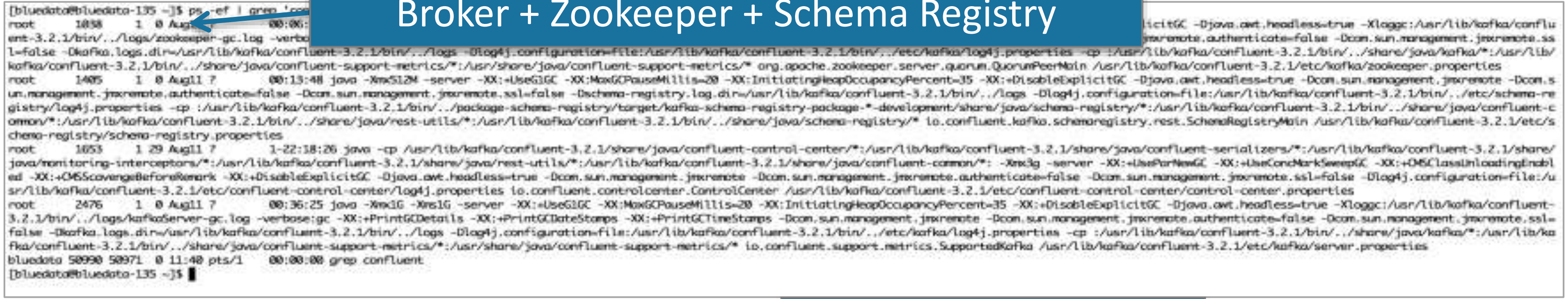
Software Bits

Runtime

Development
(e.g. extract .bin and modify to create new bin)

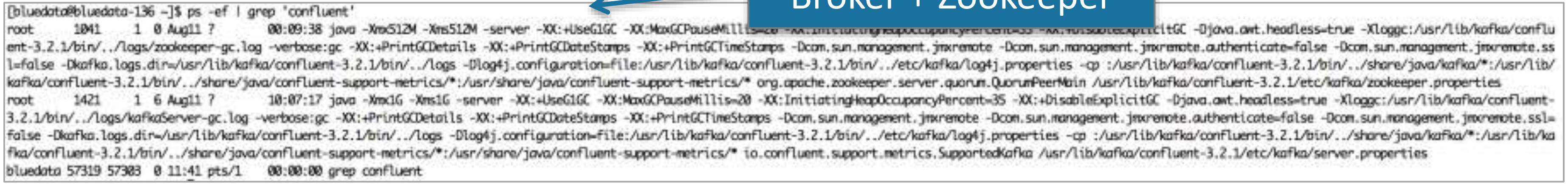
Different Services in Each Container

Broker + Zookeeper + Schema Registry



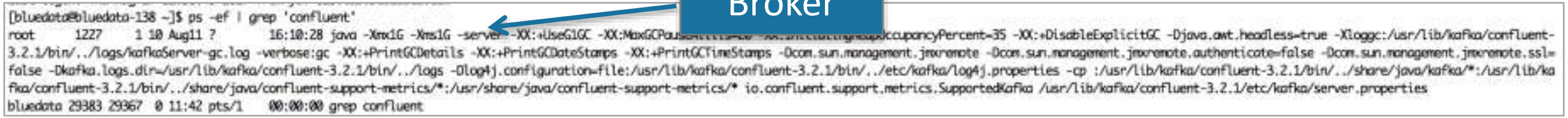
```
[bluedata@bluedata-135 ~]$ ps -ef | grep 'con
root      1038     1  0 Aug1                    licitGC -Djava.awt.headless=true -Xloggc:/usr/lib/kafka/conflu
ent-3.2.1/bin/../logs/zookeeper-gc.log -verbo                    jmxremote.authenticate=false -Dcom.sun.management.jmxremote.ss
l=false -Dkafka.logs.dir=/usr/lib/kafka/confluent-3.2.1/bin/../logs -Dlog4j.configuration=file:/usr/lib/kafka/confluent-3.2.1/bin/../etc/kafka/log4j.properties -cp :/usr/lib/kafka/confluent-3.2.1/bin/../share/java/kafka/*:/usr/lib/
kafka/confluent-3.2.1/bin/../share/java/confluent-support-metrics/*:/usr/share/java/confluent-support-metrics/* org.apache.zookeeper.server.quorum.QuorumPeerMain /usr/lib/kafka/confluent-3.2.1/etc/kafka/zookeeper.properties
root      1405     1  0 Aug11 ?        00:13:48 java -Xmx512M -server -XX:+UseG1GC -XX:MaxGCPauseMillis=20 -XX:InitiatingHeapOccupancyPercent=35 -XX:+DisableExplicitGC -Djava.awt.headless=true -Dcom.sun.management.jmxremote -Dcom.s
un.management.jmxremote.authenticate=false -Dcom.sun.management.jmxremote.ssl=false -Dschema-registry.log.dir=/usr/lib/kafka/confluent-3.2.1/bin/../logs -Dlog4j.configuration=file:/usr/lib/kafka/confluent-3.2.1/bin/../etc/schema-re
gistry/log4j.properties -cp :/usr/lib/kafka/confluent-3.2.1/bin/../package-schema-registry/target/kafka-schema-registry-package-*-development/share/java/schema-registry/*:/usr/lib/kafka/confluent-3.2.1/bin/../share/java/confluent-c
ommon/*:/usr/lib/kafka/confluent-3.2.1/bin/../share/java/rest-utils/*:/usr/lib/kafka/confluent-3.2.1/bin/../share/java/schema-registry/* io.confluent.kafka.schemaregistry.rest.SchemaRegistryMain /usr/lib/kafka/confluent-3.2.1/etc/s
chema-registry/schema-registry.properties
root      1653     1 29 Aug11 ?        1-22:18:26 java -cp /usr/lib/kafka/confluent-3.2.1/share/java/confluent-control-center/*:/usr/lib/kafka/confluent-3.2.1/share/java/confluent-serializers/*:/usr/lib/kafka/confluent-3.2.1/share/
java/monitoring-interceptors/*:/usr/lib/kafka/confluent-3.2.1/share/java/rest-utils/*:/usr/lib/kafka/confluent-3.2.1/share/java/confluent-common/*: -Xmx3g -server -XX:+UseParNewGC -XX:+UseConcMarkSweepGC -XX:+CMSClassUnloadingEnabl
ed -XX:+CMSScavengeBeforeRemark -XX:+DisableExplicitGC -Djava.awt.headless=true -Dcom.sun.management.jmxremote -Dcom.sun.management.jmxremote.authenticate=false -Dcom.sun.management.jmxremote.ssl=false -Dlog4j.configuration=file:/u
sr/lib/kafka/confluent-3.2.1/etc/confluent-control-center/log4j.properties io.confluent.controlcenter.ControlCenter /usr/lib/kafka/confluent-3.2.1/etc/confluent-control-center/control-center.properties
root      2476     1  0 Aug11 ?        00:36:25 java -Xmx1G -Xms1G -server -XX:+UseG1GC -XX:MaxGCPauseMillis=20 -XX:InitiatingHeapOccupancyPercent=35 -XX:+DisableExplicitGC -Djava.awt.headless=true -Xloggc:/usr/lib/kafka/confluent-
3.2.1/bin/../logs/kafkaServer-gc.log -verbose:gc -XX:+PrintGCDetails -XX:+PrintGCDateStamps -XX:+PrintGCTimeStamps -Dcom.sun.management.jmxremote -Dcom.sun.management.jmxremote.authenticate=false -Dcom.sun.management.jmxremote.ssl=
false -Dkafka.logs.dir=/usr/lib/kafka/confluent-3.2.1/bin/../logs -Dlog4j.configuration=file:/usr/lib/kafka/confluent-3.2.1/bin/../etc/kafka/log4j.properties -cp :/usr/lib/kafka/confluent-3.2.1/bin/../share/java/kafka/*:/usr/lib/ka
fka/confluent-3.2.1/bin/../share/java/confluent-support-metrics/*:/usr/share/java/confluent-support-metrics/* io.confluent.support.metrics.SupportedKafka /usr/lib/kafka/confluent-3.2.1/etc/kafka/server.properties
bluedata 50990 50971  0 11:40 pts/1    00:00:00 grep confluent
[bluedata@bluedata-135 ~]$ ▌
```

Broker + Zookeeper



```
[bluedata@bluedata-136 ~]$ ps -ef | grep 'confluent'
root      1041     1  0 Aug11 ?        00:09:38 java -Xmx512M -Xms512M -server -XX:+UseG1GC -XX:MaxGCPauseMill                    itGC -Djava.awt.headless=true -Xloggc:/usr/lib/kafka/conflu
ent-3.2.1/bin/../logs/zookeeper-gc.log -verbose:gc -XX:+PrintGCDetails -XX:+PrintGCDateStamps -XX:+PrintGCTimeStamps -Dcom.sun.management.jmxremote -Dcom.sun.management.jmxremote.authenticate=false -Dcom.sun.management.jmxremote.ss
l=false -Dkafka.logs.dir=/usr/lib/kafka/confluent-3.2.1/bin/../logs -Dlog4j.configuration=file:/usr/lib/kafka/confluent-3.2.1/bin/../etc/kafka/log4j.properties -cp :/usr/lib/kafka/confluent-3.2.1/bin/../share/java/kafka/*:/usr/lib/
kafka/confluent-3.2.1/bin/../share/java/confluent-support-metrics/*:/usr/share/java/confluent-support-metrics/* org.apache.zookeeper.server.quorum.QuorumPeerMain /usr/lib/kafka/confluent-3.2.1/etc/kafka/zookeeper.properties
root      1421     1  6 Aug11 ?        10:07:17 java -Xmx1G -Xms1G -server -XX:+UseG1GC -XX:MaxGCPauseMillis=20 -XX:InitiatingHeapOccupancyPercent=35 -XX:+DisableExplicitGC -Djava.awt.headless=true -Xloggc:/usr/lib/kafka/confluent-
3.2.1/bin/../logs/kafkaServer-gc.log -verbose:gc -XX:+PrintGCDetails -XX:+PrintGCDateStamps -XX:+PrintGCTimeStamps -Dcom.sun.management.jmxremote -Dcom.sun.management.jmxremote.authenticate=false -Dcom.sun.management.jmxremote.ssl=
false -Dkafka.logs.dir=/usr/lib/kafka/confluent-3.2.1/bin/../logs -Dlog4j.configuration=file:/usr/lib/kafka/confluent-3.2.1/bin/../etc/kafka/log4j.properties -cp :/usr/lib/kafka/confluent-3.2.1/bin/../share/java/kafka/*:/usr/lib/ka
fka/confluent-3.2.1/bin/../share/java/confluent-support-metrics/*:/usr/share/java/confluent-support-metrics/* io.confluent.support.metrics.SupportedKafka /usr/lib/kafka/confluent-3.2.1/etc/kafka/server.properties
bluedata 57319 57303  0 11:41 pts/1    00:00:00 grep confluent
```

Broker

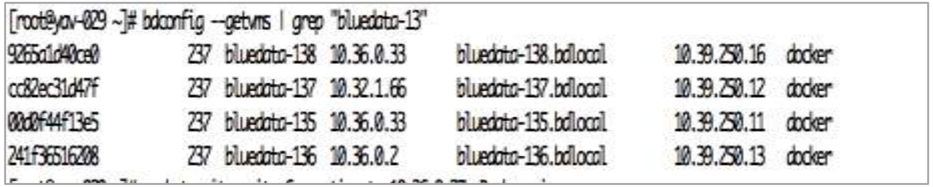


```
[bluedata@bluedata-138 ~]$ ps -ef | grep 'confluent'
root      1227     1 10 Aug11 ?        16:10:28 java -Xmx1G -Xms1G -server -XX:+UseG1GC -XX:MaxGCPau                    ccupancyPercent=35 -XX:+DisableExplicitGC -Djava.awt.headless=true -Xloggc:/usr/lib/kafka/confluent-
3.2.1/bin/../logs/kafkaServer-gc.log -verbose:gc -XX:+PrintGCDetails -XX:+PrintGCDateStamps -XX:+PrintGCTimeStamps -Dcom.sun.management.jmxremote -Dcom.sun.management.jmxremote.authenticate=false -Dcom.sun.management.jmxremote.ssl=
false -Dkafka.logs.dir=/usr/lib/kafka/confluent-3.2.1/bin/../logs -Dlog4j.configuration=file:/usr/lib/kafka/confluent-3.2.1/bin/../etc/kafka/log4j.properties -cp :/usr/lib/kafka/confluent-3.2.1/bin/../share/java/kafka/*:/usr/lib/ka
fka/confluent-3.2.1/bin/../share/java/confluent-support-metrics/*:/usr/share/java/confluent-support-metrics/* io.confluent.support.metrics.SupportedKafka /usr/lib/kafka/confluent-3.2.1/etc/kafka/server.properties
bluedata 29383 29367  0 11:42 pts/1    00:00:00 grep confluent
```

bluedata

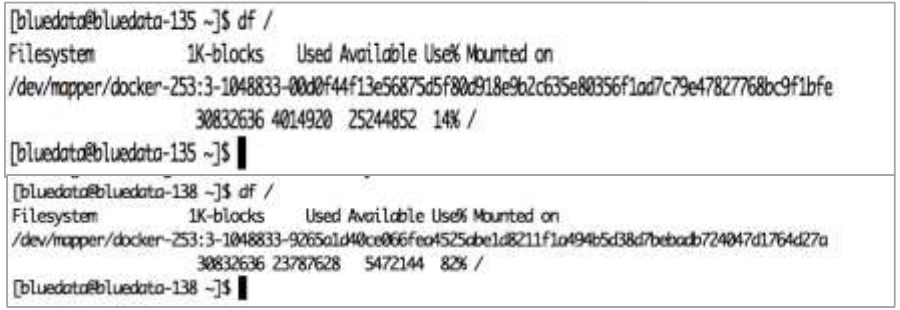
Container Storage On Host

Kafka Cluster Containers



```
[root@yav-029 ~]# bdconfig --getvms | grep "bluedata-13"
9265a1d40ce0          237  bluedata-138  10.36.0.33    bluedata-138.bdlocal     10.39.250.16  docker
cc82ec31d47f          237  bluedata-137  10.32.1.66    bluedata-137.bdlocal     10.39.250.12  docker
00d0f44f13e5          237  bluedata-135  10.36.0.33    bluedata-135.bdlocal     10.39.250.11  docker
241f36516208          237  bluedata-136  10.36.0.2     bluedata-136.bdlocal     10.39.250.13  docker
```

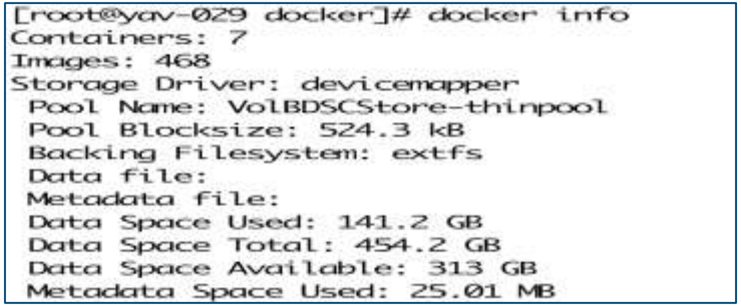
Container Storage



```
[bluedata@bluedata-135 ~]$ df /
Filesystem           1K-blocks    Used Available Use% Mounted on
/dev/mapper/docker-253:3-1048833-00d0f44f13e56875d5f80d918e9b2c635e80356f1ad7c79e47827768bc9f1bfe
                      30832636 4014920  25244852  14% /
[bluedata@bluedata-135 ~]$
```

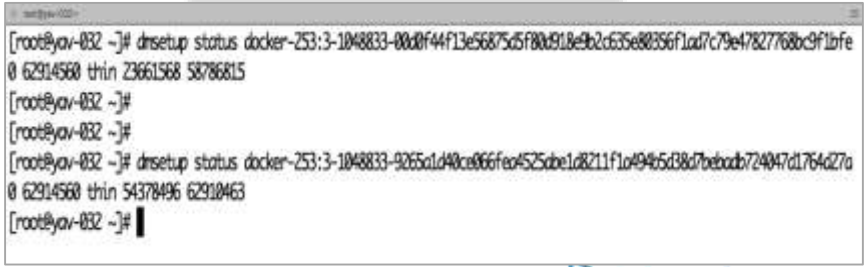
```
[bluedata@bluedata-138 ~]$ df /
Filesystem           1K-blocks     Used Available Use% Mounted on
/dev/mapper/docker-253:3-1048833-9265a1d40ce066fea4525abe1d8211f1a494b5d38d7bebadb724047d1764d27a
                      30832636 23787628   5472144  82% /
[bluedata@bluedata-138 ~]$
```

Container Hosts



```
[root@yav-029 docker]# docker info
Containers: 7
Images: 468
Storage Driver: devicemapper
 Pool Name: VolBDSCStore-thinpool
 Pool Blocksize: 524.3 kB
 Backing Filesystem: extfs
 Data file:
 Metadata file:
 Data Space Used: 141.2 GB
 Data Space Total: 454.2 GB
 Data Space Available: 313 GB
 Metadata Space Used: 25.01 MB
```

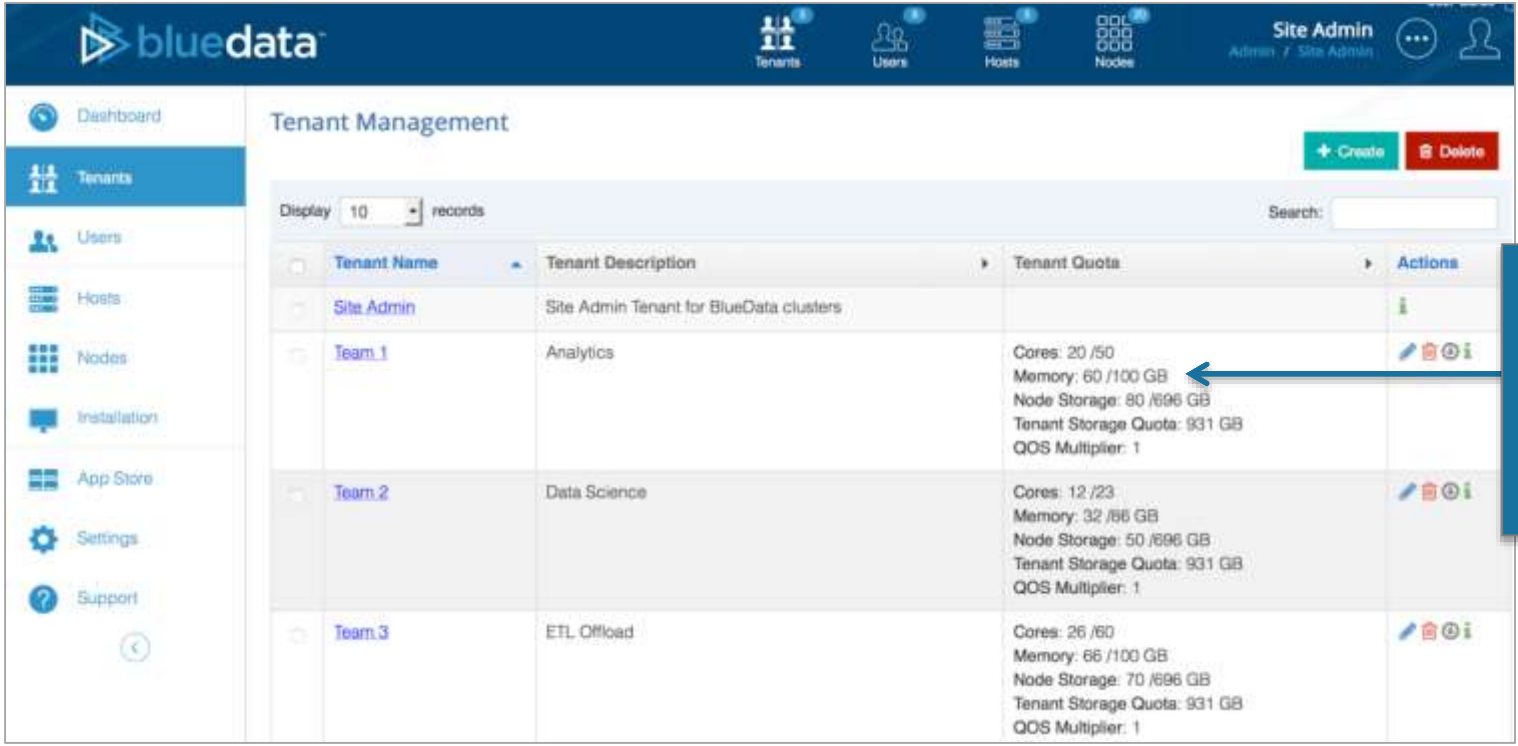
Host Storage



```
[root@yav-032 ~]# dmsetup status docker-253:3-1048833-00d0f44f13e56875d5f80d918e9b2c635e80356f1ad7c79e47827768bc9f1bfe
0 62914560 thin 23661568 58786815
[root@yav-032 ~]#
[root@yav-032 ~]#
[root@yav-032 ~]# dmsetup status docker-253:3-1048833-9265a1d40ce066fea4525abe1d8211f1a494b5d38d7bebadb724047d1764d27a
0 62914560 thin 54378496 62910463
[root@yav-032 ~]#
```

Multi-Tenant Resource Quotas

Aggregate compute, memory, & storage quotas for Docker containers



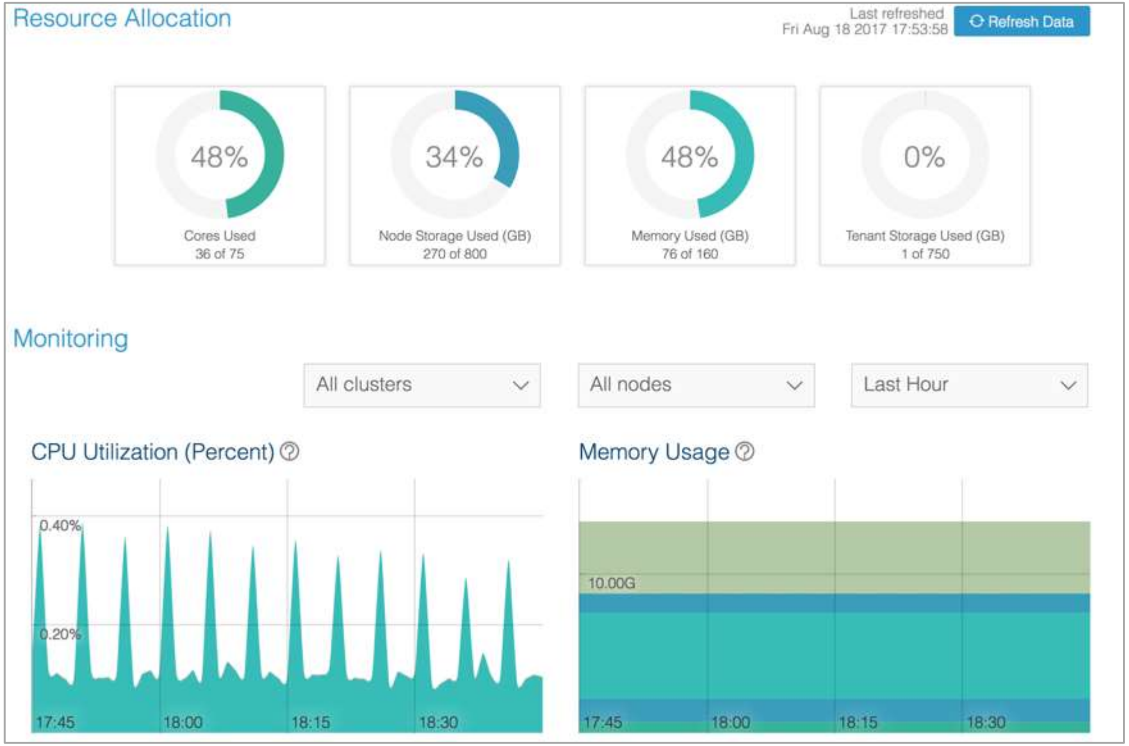
Tenant Management

	Tenant Name	Tenant Description	Tenant Quota	Actions
□	Site Admin	Site Admin Tenant for BlueData clusters		
□	Team 1	Analytics	Cores: 20 /50 Memory: 60 /100 GB Node Storage: 80 /696 GB Tenant Storage Quota: 931 GB QOS Multiplier: 1	
□	Team 2	Data Science	Cores: 12 /23 Memory: 32 /86 GB Node Storage: 50 /696 GB Tenant Storage Quota: 931 GB QOS Multiplier: 1	
□	Team 3	ETL Offload	Cores: 26 /60 Memory: 66 /100 GB Node Storage: 70 /696 GB Tenant Storage Quota: 931 GB QOS Multiplier: 1	

Aggregate Docker container storage, memory and cores (CPU shares) for all containers in tenant "Team 1"

Monitoring Containers

Resource monitoring



Resource Allocation

Last refreshed
Fri Aug 18 2017 17:53:58

Cores Used 48%
36 of 75

Node Storage Used (GB) 34%
270 of 800

Memory Used (GB) 48%
76 of 160

Tenant Storage Used (GB) 0%
1 of 750

Monitoring

All clusters | All nodes | Last Hour

CPU Utilization (Percent)

Memory Usage

Several open source and commercial monitoring options available

We use Elasticsearch with Metricbeat plugin

Containers = the Future of Apps

Infrastructure

- Agility and elasticity
- Standardized environments *(dev, test, prod)*
- Portability *(on-premises and cloud)*
- Higher resource utilization

Applications

- Fool-proof packaging *(configs, libraries, driver versions, etc.)*
- Repeatable builds and orchestration
- Faster app dev cycles

Kafka on Docker: Key Takeaways

- Enterprise deployment requirements:
  - Docker base image includes all needed services (Kafka, Zookeeper, Schema registry, etc.), libraries, jar files
  - Container orchestration, including networking and storage, depends on standards enforced by enterprises
  - Resource-aware runtime configuration, including CPU and RAM
  - Sequence-aware app deployment needs more thought

Kafka on Docker: Key Takeaways

- Enterprise deployment challenges:
  - Access to container secured with ssh keypair or PAM module (LDAP/AD)
  - Access to Kafka from Data Science applications
  - Management agents in Docker images
  - Runtime injection of resource and configuration information
- Consider a turnkey software solution (e.g. BlueData) to accelerate time to value and avoid DIY pitfalls

bluedata

Nanda Vijaydev

@NandaVijaydev

www.bluedata.com