Lessons Learned From Dockerizing Spark Workloads

Thomas Phelan
Chief Architect, BlueData
@tapbluedata

Nanda Vijaydev
Data Scientist, BlueData
@nandavijaydev

February 8, 2017

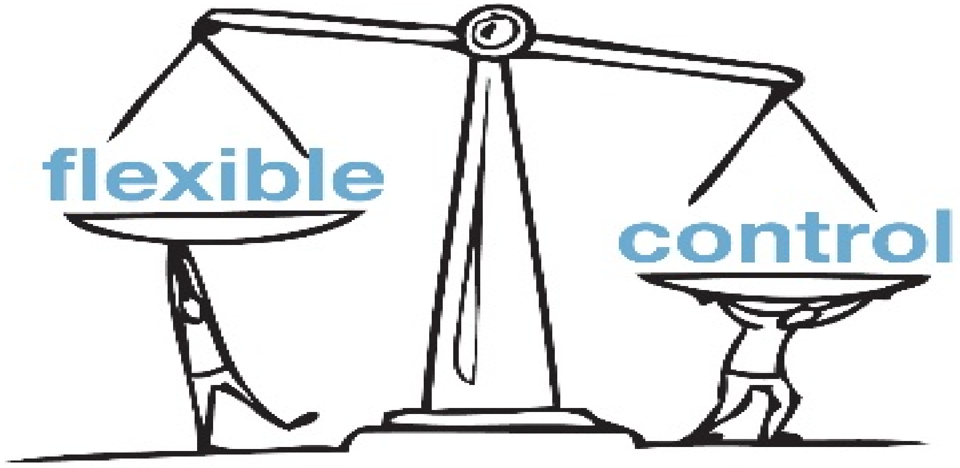
SPARK SUMMIT EAST 2017

Outline

- Docker Containers and Big Data
- Spark on Docker: Challenges
- How We Did It: Lessons Learned
- Key Takeaways
- Q & A

Distributed Spark Environments

- Data scientists want flexibility:
  - New tools, latest versions of Spark, Kafka, H2O, et.al.
  - Multiple options – e.g. Zeppelin, RStudio, JupyterHub
  - Fast, iterative prototyping
- IT wants control:
  - Multi-tenancy
  - Data security
  - Network isolation



Why “Dockerize”?

docker

Infrastructure

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

Applications

- Fool-proof packaging *(configs, libraries, driver versions, etc.)*
- Repeatable builds and orchestration
- Faster app dev cycles
- Lightweight *(virtually no performance or startup penalty)*

The Journey to Spark on Docker

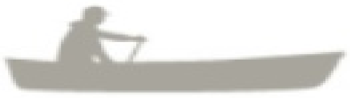
So you want to run Spark on Docker in a multi-tenant enterprise deployment?

Start with a clear goal in sight

Begin with your Docker toolbox of a single container and basic networking and storage

Warning: there are some pitfalls & challenges

Spark on Docker: Pitfalls



Navigate the river of container managers

- *Swarm ?*
- *Kubernetes ?*
- *AWS ECS ?*
- *Mesos ?*

Cross the desert of storage configurations

- *Overlay files ?*
- *Flocker ?*
- *Convoy ?*

Traverse the tightrope of network configurations

- *Docker Networking ?*
  Calico
- *Kubernetes Networking ?*
  Flannel, Weave Net

Spark on Docker: Challenges

Tame the lion of performance

Pass thru the jungle of software configurations

- *R ?*
- *Python ?*
- *HDFS ?*
- *NoSQL ?*

Trip down the staircase of deployment mistakes

- *On-premises ?*
- *Public cloud ?*

Finally you get to the top!

Spark on Docker: Next Steps?

But for an enterprise-ready deployment, you are still not even close to being done …

You still have to climb past: high availability, backup/recovery, security, multi-host, multi-container, upgrades and patches …

Spark on Docker: Quagmire

You realize it’s time
to get some help!

How We Did It: Spark on Docker

- Docker containers provide a powerful option for greater agility and flexibility – whether on-premises or in the public cloud
- Running a complex, multi-service platform such as Spark in containers in a distributed enterprise-grade environment can be daunting
- Here is how we did it for the BlueData platform ... while maintaining performance comparable to bare-metal

How We Did It: Design Decisions

- Run Spark (any version) with related models, tools / applications, and notebooks unmodified
- Deploy all relevant services that typically run on a single Spark host in a single Docker container
- Multi-tenancy support is key
  - Network and storage security
  - User management and access control

SPARK SUMMIT EAST 2017

How We Did It: Design Decisions

- Docker 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
    - Master vs. worker cluster nodes

SPARK SUMMIT EAST 2017

How We Did It: Design Decisions

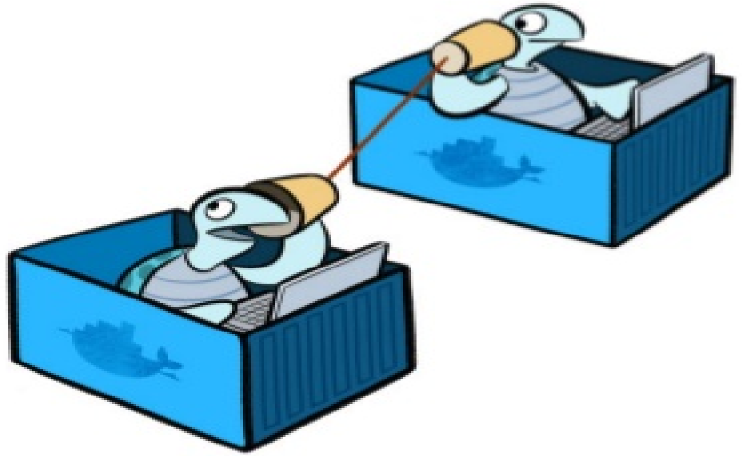
- Maintain the promise of Docker containers
  - Keep them as stateless as possible
  - Container storage is always ephemeral
  - Persistent storage is external to the container

STATELESS

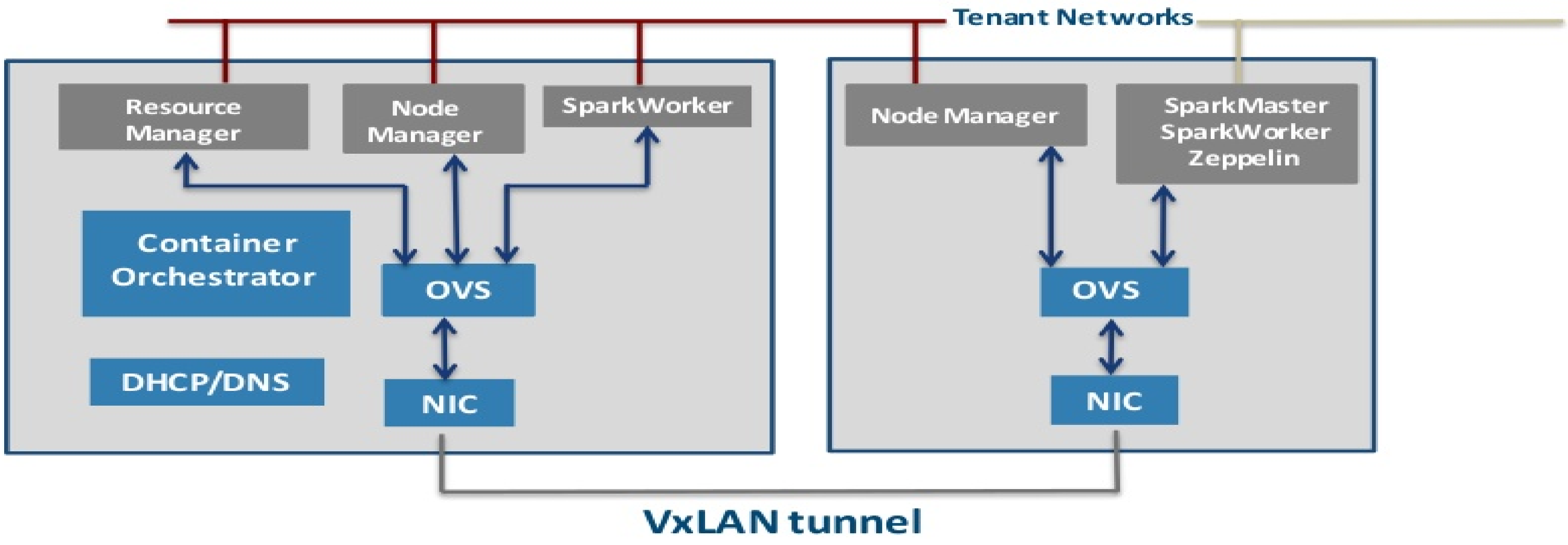
How We Did It: CPU & Network

- Resource Utilization
  - CPU cores vs. CPU shares
  - Over-provisioning of CPU recommended
  - No over-provisioning of memory

- Network
  - Connect containers across hosts
  - Persistence of IP address across container restart
  - Deploy VLANs and VxLAN tunnels for tenant-level traffic isolation



How We Did It: Network



How We Did It: Storage

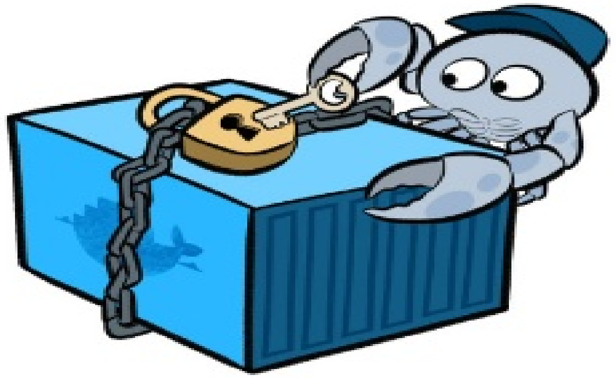
- Storage
  - Tweak the default size of a container's /root
    - Resizing of storage inside an existing container is tricky
  - Mount logical volume on /data
    - No use of overlay file systems
  - BlueData DataTap (version-independent, HDFS-compliant)
    - Connectivity to external storage
- Image Management
  - Utilize Docker's image repository

TIP: Mounting block devices into a container does not support symbolic links (IOW: /dev/sdb will not work, /dm/… PCI device can change across host reboot).

TIP: Docker images can get large. Use "docker squash" to save on size.

How We Did It: Security

- 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



How We Did It: Security

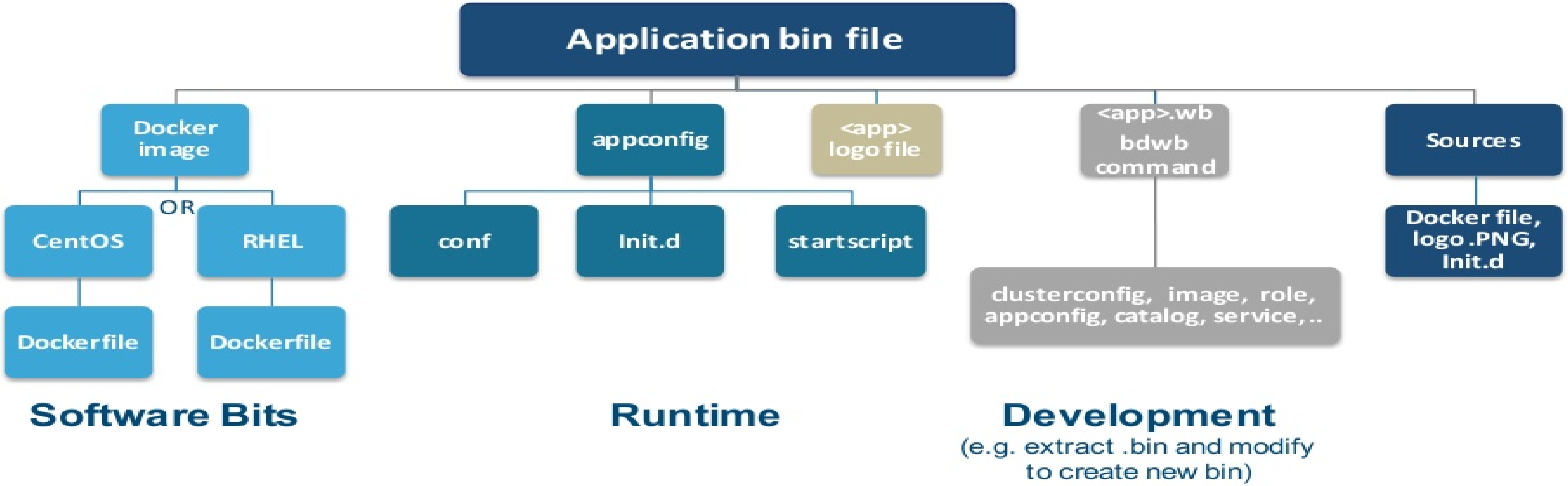
- Add "capabilities" to a container based on what operations are permitted

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: Sample Dockerfile

```
# Spark-1. docker image for RHEL/CentOS 6.x
FROM centos:centos6

# Download and extract spark
RUN mkdir /usr/lib/spark; curl -s http://archive.apache.org/dist/spark/spark-2.0.1/spark-2.0.1-
bin-hadoop2.6.tgz | tar xz -C /usr/lib/spark/

## Install zeppelin
RUN mkdir /usr/lib/zeppelin; curl -s https://s3.amazonaws.com/bluedata-
catalog/thirdparty/zeppelin/zeppelin-0.7.0-SNAPSHOT.tar.gz | tar xz -C /usr/lib/zeppelin

ADD configure_spark_services.sh /root/configure_spark_services.sh
RUN chmod -x /root/configure_spark_services.sh && /root/configure_spark_services.sh
```

BlueData App Image (.bin file)



Multi-Host

root@yav-028:~

```
[root@yav-028 ~]# docker ps
CONTAINER ID        IMAGE                           COMMAND                  CREATED             STATUS              PORTS               NAMES
df43180810b6        bluedata/spark15:88eb70e22e9e   "/usr/bin/supervisor     4 minutes ago       Up 4 minutes                            bluedata-8
7ff8d4d3c515        bluedata/spark15:88eb70e22e9e   "/usr/bin/supervisor     4 minutes ago       Up 4 minutes                            bluedata-6
[root@yav-028 ~]#
```

4 containers
on 2 different hosts
using 1 VLAN and 4
persistent IPs

root@yav-210:~

```
[root@yav-210 ~]# docker ps
CONTAINER ID        IMAGE                           COMMAND                  CREATED             STATUS              PORTS               NAMES
72eb670b2dd2        bluedata/spark15:88eb70e22e9e   "/usr/bin/supervisor     4 minutes ago       Up 4 minutes                            bluedata-9
efeb3165b3b5        bluedata/spark15:88eb70e22e9e   "/usr/bin/supervisor     4 minutes ago       Up 4 minutes                            bluedata-7
[root@yav-210 ~]#
```

SPARK SUMMIT EAST 2017

Services per Container

Master Services

bluedata@bluedata-6:~

```
[bluedata@bluedata-6 ~]$ ps -ef | grep master | grep -v grep
root       898     1  3 07:24 ?        00:00:17 /usr/lib/jvm/java-openjdk/bin/java -cp /usr/lib/spark/spark-1.5.2-bin-hadoop2.4/sbin/../conf/:/usr/lib/spark/spark
-1.5.2-bin-hadoop2.4/lib/spark-assembly-1.5.2-hadoop2.4.0.jar:/usr/lib/spark/spark-1.5.2-bin-hadoop2.4/lib/datanucleus-core-3.2.10.jar:/usr/lib/spark/spark-1.5.2-
bin-hadoop2.4/lib/datanucleus-rdbms-3.2.9.jar:/usr/lib/spark/spark-1.5.2-bin-hadoop2.4/lib/datanucleus-api-jdo-3.2.6.jar:/etc/hadoop/conf/ -Xms1g -Xmx1g -XX:MaxPe
rmSize=256m org.apache.spark.deploy.master.Master --ip bluedata-6.bdlocal --port 7077 --webui-port 8080
[bluedata@bluedata-6 ~]$ ps -ef | grep worker | grep -v grep
root      2048     1  2 07:24 ?        00:00:12 /usr/lib/jvm/java-openjdk/bin/java -cp /usr/lib/spark/spark-1.5.2-bin-hadoop2.4/sbin/../conf/:/usr/lib/spark/spark
-1.5.2-bin-hadoop2.4/lib/spark-assembly-1.5.2-hadoop2.4.0.jar:/usr/lib/spark/spark-1.5.2-bin-hadoop2.4/lib/datanucleus-core-3.2.10.jar:/usr/lib/spark/spark-1.5.2-
bin-hadoop2.4/lib/datanucleus-rdbms-3.2.9.jar:/usr/lib/spark/spark-1.5.2-bin-hadoop2.4/lib/datanucleus-api-jdo-3.2.6.jar:/etc/hadoop/conf/ -Xms1g -Xmx1g -XX:MaxPe
rmSize=256m org.apache.spark.deploy.worker.Worker --webui-port 8081 spark://bluedata-6.bdlocal:7077
[bluedata@bluedata-6 ~]$
```

Worker Services

bluedata@bluedata-7:~

```
[bluedata@bluedata-7 ~]$
[bluedata@bluedata-7 ~]$
[bluedata@bluedata-7 ~]$
[bluedata@bluedata-7 ~]$
[bluedata@bluedata-7 ~]$ ps -ef |grep master | grep -v grep
[bluedata@bluedata-7 ~]$ ps -ef |grep worker | grep -v grep
root       926     1  1 07:24 ?        00:00:08 /usr/lib/jvm/java-openjdk/bin/java -cp /usr/lib/spark/spark-1.5.2-bin-hadoop2.4/sbin/../conf/:/usr/lib/spark/spark
1.5.2-bin-hadoop2.4/lib/spark-assembly-1.5.2-hadoop2.4.0.jar:/usr/lib/spark/spark-1.5.2-bin-hadoop2.4/lib/datanucleus-api-jdo-3.2.6.jar:/usr/lib/spark/spark-1.5.2
bin-hadoop2.4/lib/datanucleus-rdbms-3.2.9.jar:/usr/lib/spark/spark-1.5.2-bin-hadoop2.4/lib/datanucleus-core-3.2.10.jar:/etc/hadoop/conf/ -Xms1g -Xmx1g -XX:MaxPerm
ize=256m org.apache.spark.deploy.worker.Worker --webui-port 8081 spark://bluedata-6.bdlocal:7077
[bluedata@bluedata-7 ~]$
```

SPARK SUMMIT EAST 2017

Container Storage from Host

Container Storage

Host Storage

```
[bluedata@bluedata-6 ~]$ df / /data
Filesystem           1K-blocks     Used Available Use% Mounted on
/dev/mapper/docker-253:1-36831402-7ff8d4d3c51599f302a2a38cd15efcd81e4e38e7ee305fd3ebbdfb801904e5a7
                      20511356 2389232  17073548  13% /
/dev/mapper/volBDSCStore-bluedata--6
                      10190136   23040   9642808   1% /data
[bluedata@bluedata-6 ~]$
```

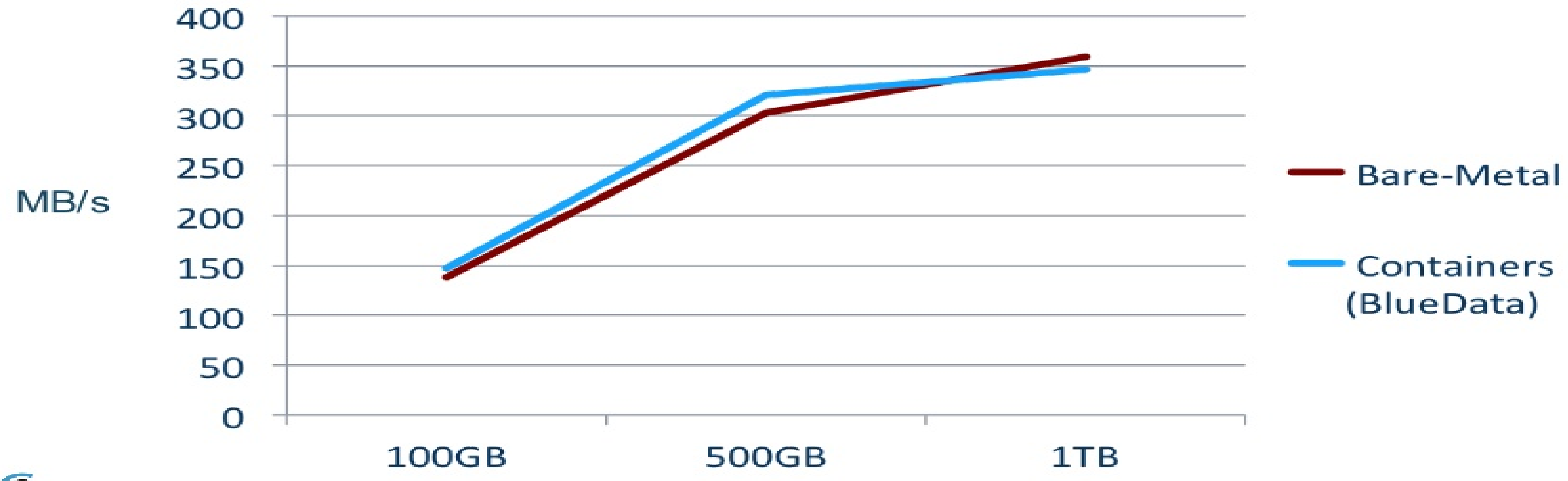
```
[bluedata@bluedata-7 ~]$ df / /data
Filesystem           1K-blocks     Used Available Use% Mounted on
/dev/mapper/docker-253:1-65799449-efeb3165b3b5ca8c4b67e5becd2fa5b5cd639bed6cb4c75452576a143accf17b
                      20511356 2372952  17089828  13% /
/dev/mapper/volBDSCStore-bluedata--7
                      10190136   23032   9642816   1% /data
[bluedata@bluedata-7 ~]$
```

```
[root@yav-028 ~]# lvdisplay
  --- Logical volume ---
  LV Path                /dev/volBDSCStore/bluedata-6
  LV Name                bluedata-6
  VG Name                volBDSCStore
  LV UUID                z3Hw0q-45Kz-4Hyw-kgsk-4v1w-Rs1l-mq5Do1
  LV Write Access        read/write
  LV Creation host, time yav-028.lab.bluedata.com, 2016-08-16 07:23:53 -0700
  LV Status              available
  # open                 1
  LV Size                10.00 GiB
  Current LE             10
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           253:4

  --- Logical volume ---
  LV Path                /dev/volBDSCStore/bluedata-8
  LV Name                bluedata-8
  VG Name                volBDSCStore
  LV UUID                cJUDzc-yElI-M8E0-z7uE-VMYf-VZMN-rRjKAj
  LV Write Access        read/write
  LV Creation host, time yav-028.lab.bluedata.com, 2016-08-16 07:24:00 -0700
  LV Status              available
  # open                 1
  LV Size                10.00 GiB
  Current LE             10
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           253:6

  --- Logical volume ---
  LV Path                /dev/volGroup/lv_swap
```

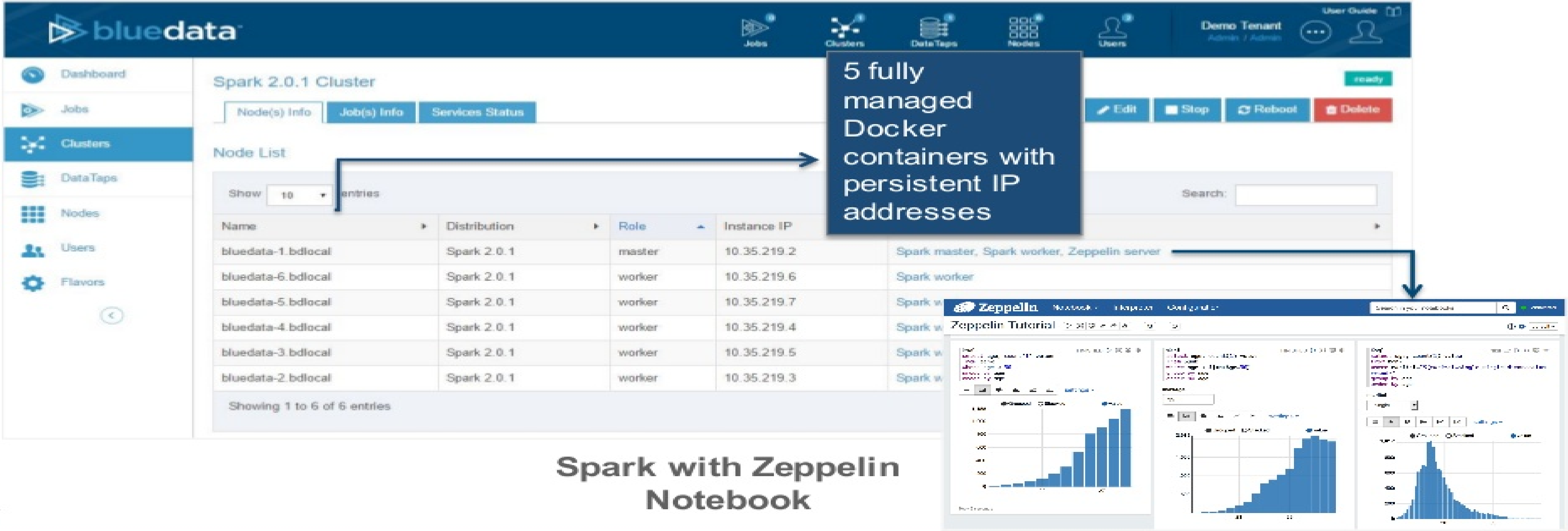
Performance Testing: Spark

- Spark 1.x on YARN
- HiBench - Terasort
- Data sizes: 100Gb, 500GB, 1TB
- 10 node physical/virtual cluster
- 36 cores and112GB memory per node
- 2TB HDFS storage per node (SSDs)
- 800GB ephemeral storage

Spark on Docker: Performance



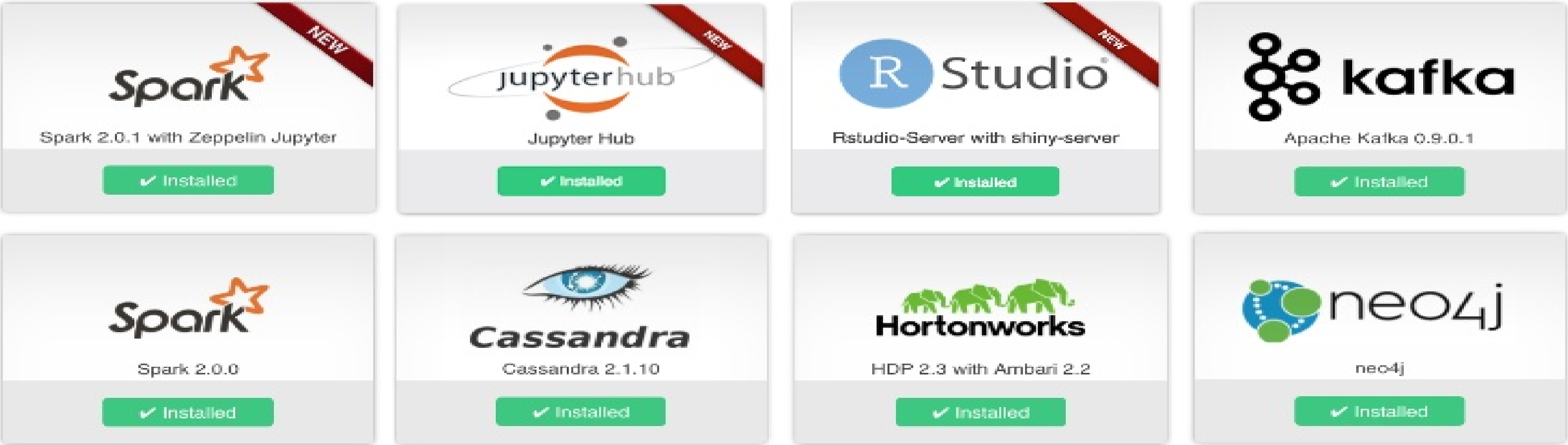
“Dockerized” Spark on BlueData



Spark with Zeppelin Notebook

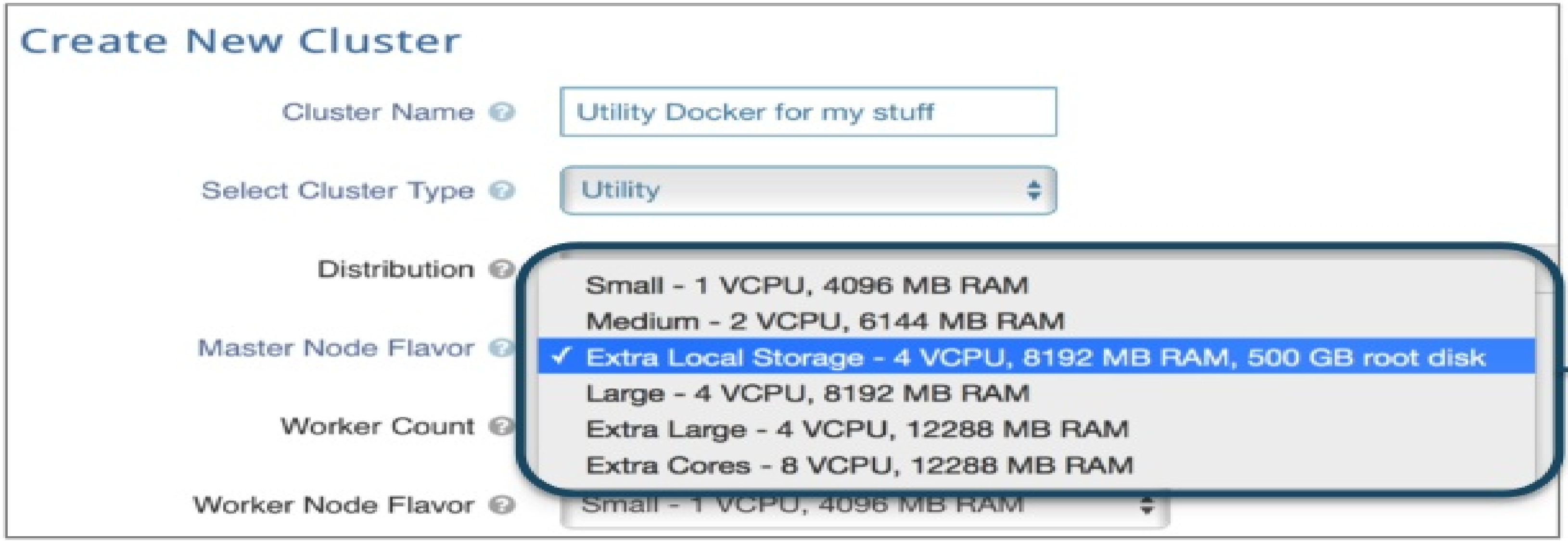
SPARK SUMMIT EAST 2017

Pre-Configured Docker Images



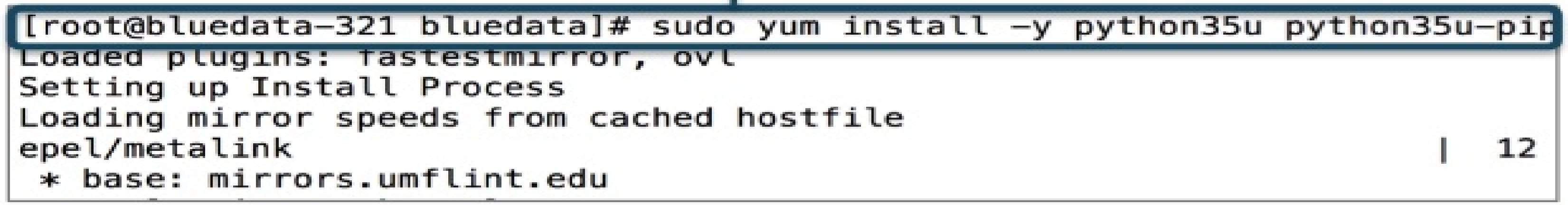
Choice of data science tools and notebooks (e.g. JupyterHub, RStudio, Zeppelin) and ability to “bring-your-own” tools and versions

On-Demand Elastic Infrastructure



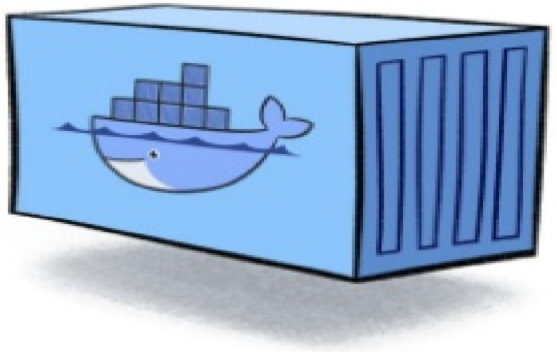
Create "pristine" Docker containers with appropriate resources

Log in to container and install your tools & libraries



Spark on Docker: Key Takeaways

- All apps can be “Dockerized”, including Spark
  - Docker containers enable a more flexible and agile deployment model
  - Faster app dev cycles for Spark app developers, data scientists, & engineers
  - Enables DevOps for data science teams



Spark on Docker: Key Takeaways

- Deployment requirements:
  - Docker base images include all needed Spark libraries and jar files
  - Container orchestration, including networking and storage
  - Resource-aware runtime environment, including CPU and RAM

Spark on Docker: Key Takeaways

- Data scientist considerations:
  - Access to data with full fidelity
  - Access to data processing and modeling tools
  - Ability to run, rerun, and scale analysis
  - Ability to compare and contrast various techniques
  - Ability to deploy & integrate enterprise-ready solution

Spark on Docker: Key Takeaways

- Enterprise deployment challenges:
  - Access to container secured with ssh keypair or PAM module (LDAP/AD)
  - Fast access to external storage
  - Management agents in Docker images
  - Runtime injection of resource and configuration information

Spark on Docker: Key Takeaways

- "Do It Yourself" will be costly & time-consuming
  - Be prepared to tackle the infrastructure challenges and pitfalls to Dockerize Spark
  - Your business value will come from data science and applications – not the plumbing
- There are other options:
  - BlueData = a turnkey solution, for on-premises or on AWS

	Turnkey Big Data platform (BlueData)	Do-It-Yourself
1	Base Docker images include Big Data development libraries	
2	Orchestration software, including networking infrastructure	
3	Resource and infrastructure aware runtime environment	
4	Always runs Docker in non-privileged mode for security	
5	Secured with LDAP/AD or ssh keypair	
6	Pre-built access (via DataTap) to HDFS in every container	
7	Tools to view, monitor and manage individual containers or clusters	
8	Includes agents to restart services in etc/init.d	
9	Improve image flexibility by injecting runtime cluster configurations	
10	Container storage from local logical volume	

Thank You

Contact Info:

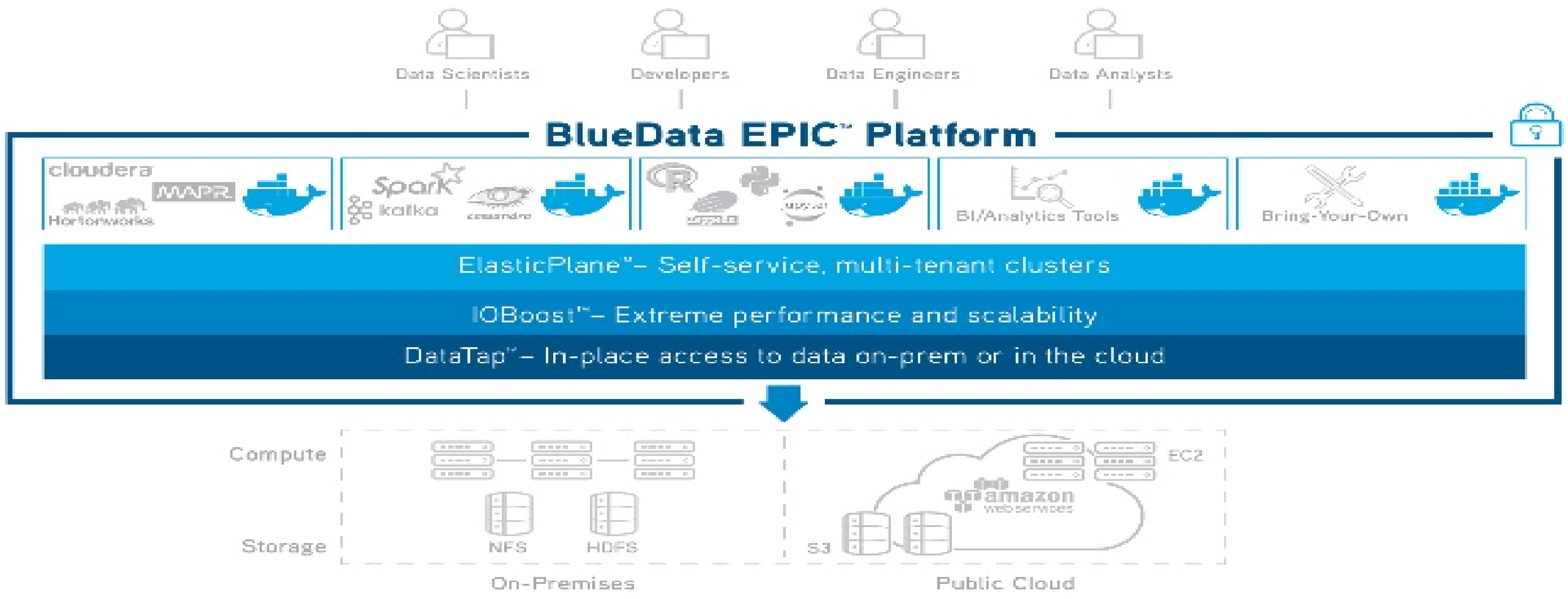
@tapbluedata

@nandavijaydev

tap@bluedata.com

nanda@bluedata.com

www.bluedata.com



Visit BlueData's booth in the expo hall