QCon 全球软件开发大会
INTERNATIONAL SOFTWARE DEVELOPMENT CONFERENCE

BEIJING 2018

京东如何基于容器打造高性能及效率的大数据平台

Zhen Fan fanzhen@jd.com

Weiting Chen weiting.chen@intel.com

主办方 Geekbang 极客邦科技 InfoQ

INTEL NOTICE & DISCLAIMER

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting www.intel.com/design/literature.htm.

Intel, the Intel logo, Intel® are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others

AGENDA

BACKGROUND

SPARK ON KUBERNETES
- Why use Spark-on-K8s
- How it works
- Current Status & Issues

JD.COM CASE STUDY
- JD.com’s MoonShot
- Experience Sharing

SUMMARY

BACKGROUND

ABOUT SPARK-ON-KUBERNETES

- https://github.com/apache-spark-on-k8s/spark
- Spark* on Kubernetes*(K8s) is a new project proposed by the companies including Bloomberg, Google, Intel, Palantir, Pepperdata, and Red Hat.
- The goal is to bring native support for Spark to use Kubernetes as a cluster manager like Spark Standalone, YARN*, or Mesos*.
- The feature is planning to be put into Spark 2.3.0 release(SPARK-18278).

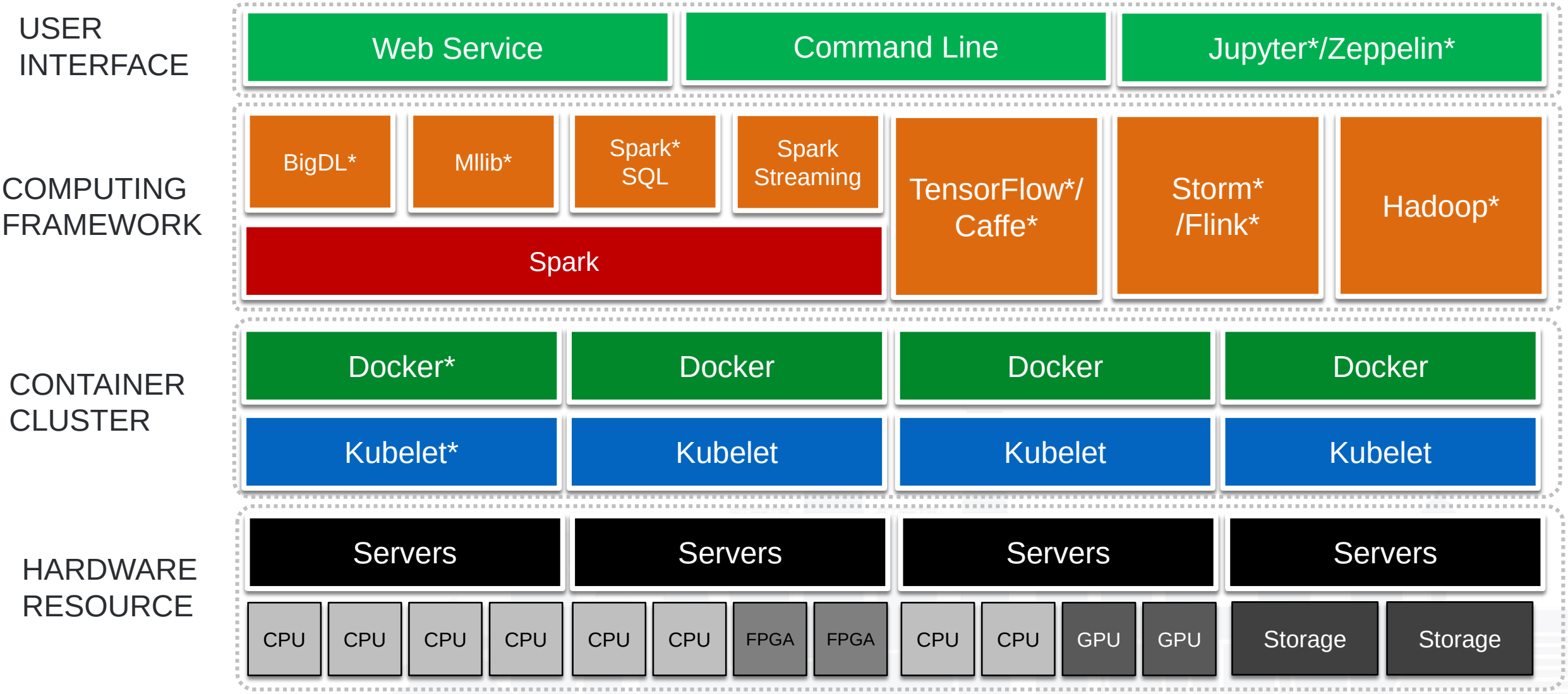
WHY JD.COM CHOOSE SPARK-ON-K8S

Heterogeneous Computing

CPU + GPU + FPGA

Customers are asking to use an unified cloud platform to manage their applications. Based on Kubernetes*, we can ease to set up a platform to support CPU, GPU, as well as FPGA resources for Big Data/AI workloads.

HETEROGENEOUS CLOUD SOLUTION



*Other names and brands may be claimed as the property of others.

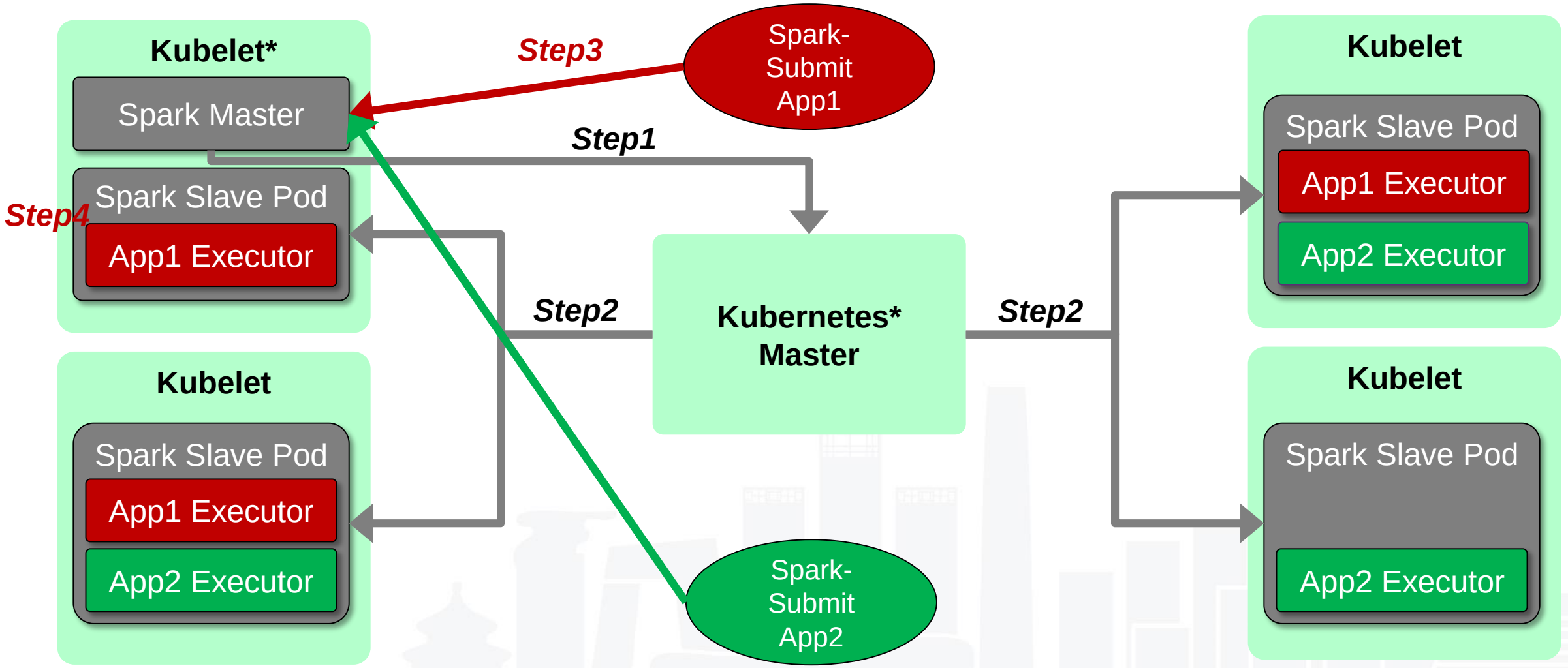
SPARK ON KUBERNETES

SPARK ON DOCKER SOLUTIONS

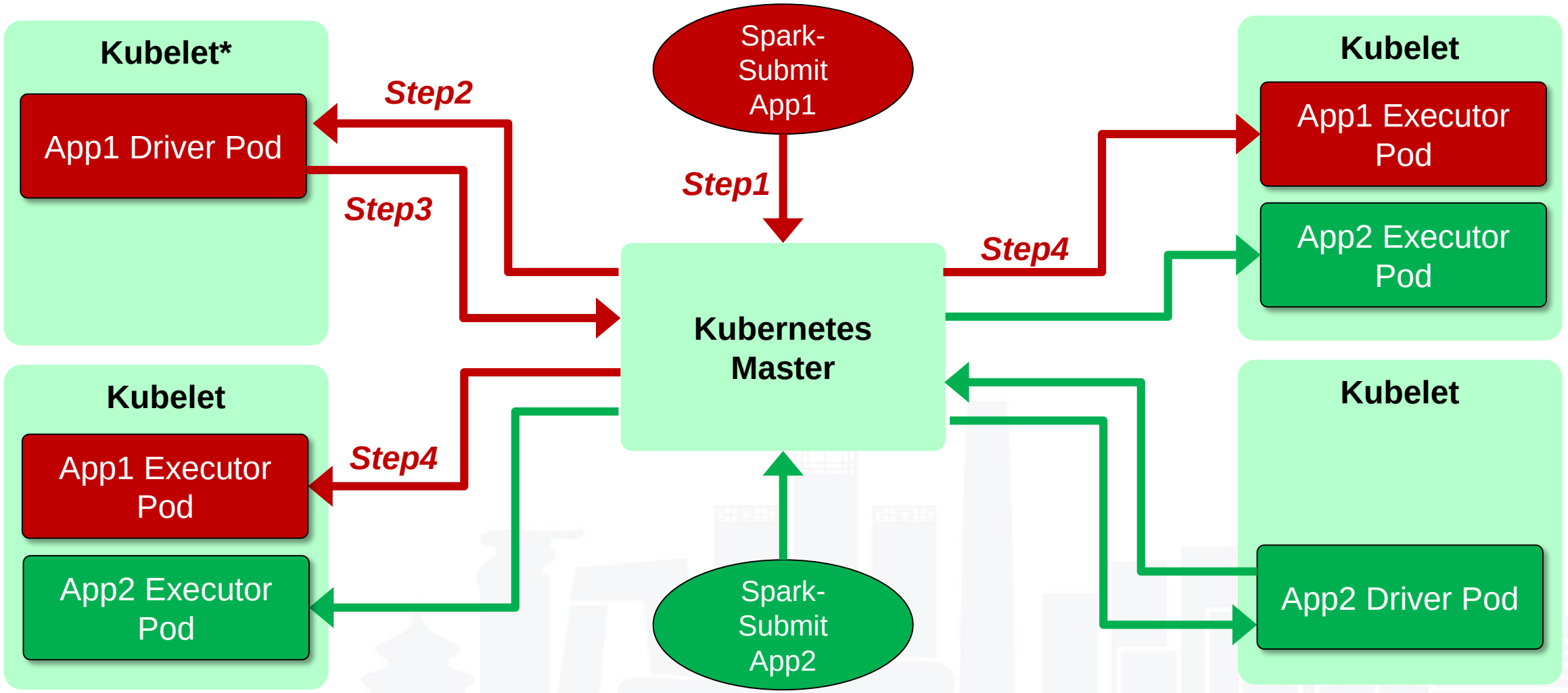
- **Solution1 - Spark* Standalone on Docker***
  - Run Spark standalone cluster in Docker.
  - Two-tiers resource allocation(K8s->Spark Cluster->Spark Applications).
  - Less efforts to migrate existing architecture into container environment.

- **Solution2 - Spark on Kubernetes***
  - Use native way to run Spark on Kubernetes like Spark Standalone, YARN, or Mesos.
  - Single tier resource allocation(K8s->Spark Applications) for higher utilization.
  - Must re-write the entire logical program for resource allocation via K8s.

SOLUTION1 - SPARK STANDALONE ON DOCKER



SOLUTION2 - SPARK ON KUBERNETES



HOW TO USE SPARK ON K8S

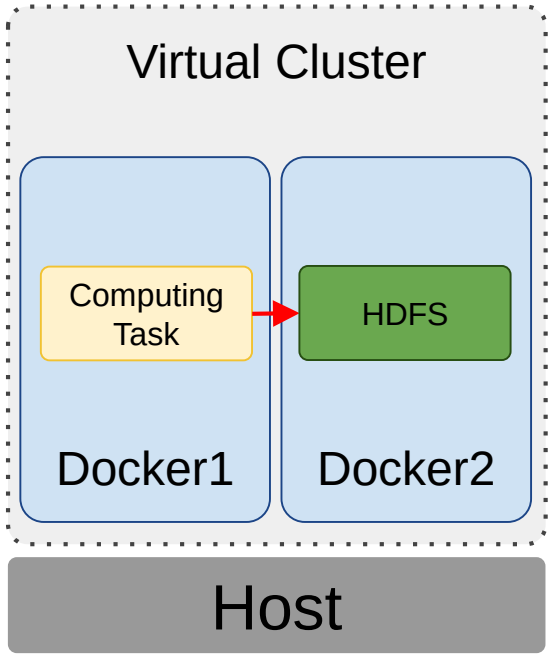
```
# bin/spark-submit \
  --deploy-mode cluster \
  --class org.apache.spark.examples.SparkPi \
  --master k8s://http://127.0.0.1:8080 \
  --kubernetes-namespace default \
  --conf spark.executor.instances=5 \
  --conf spark.executor.cores=4 \
  --conf spark.executor.memory=4g \
  --conf spark.app.name=spark-pi \
  --conf spark.kubernetes.driver.docker.image=localhost:5000/spark-driver \
  --conf spark.kubernetes.executor.docker.image=localhost:5000/spark-executor \
  --conf spark.kubernetes.initcontainer.docker.image=localhost:5000/spark-init \
  --conf spark.kubernetes.resourceStagingServer.uri=http://$ip:31000 \
  hdfs://examples/jars/spark-examples_2.11-2.1.0-k8s-0.1.0-SNAPSHOT.jar
```

KEY FEATURES

- Support Cluster Mode
- Client Mode Support is under reviewing.
- Support File Staging in local, HDFS, or running a File Stage Server container.
- Support Scala, Java, and PySpark.
- Support Static and Dynamic Allocation for Executors.
- Support running HDFS inside K8s or externally.
- Support for Kubernetes 1.6 - 1.7
- Pre-built docker images

DATA PROCESSING MODEL

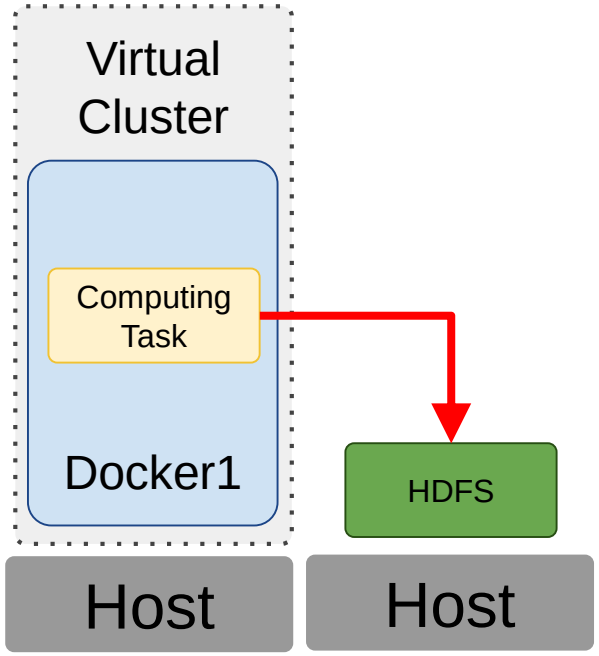
PATTERN 1: Internal HDFS*



Use HDFS as file sharing server. HDFS runs in the same host to give elasticity to add/reduce compute nodes by request.

Please refer to Spark & HDFS.

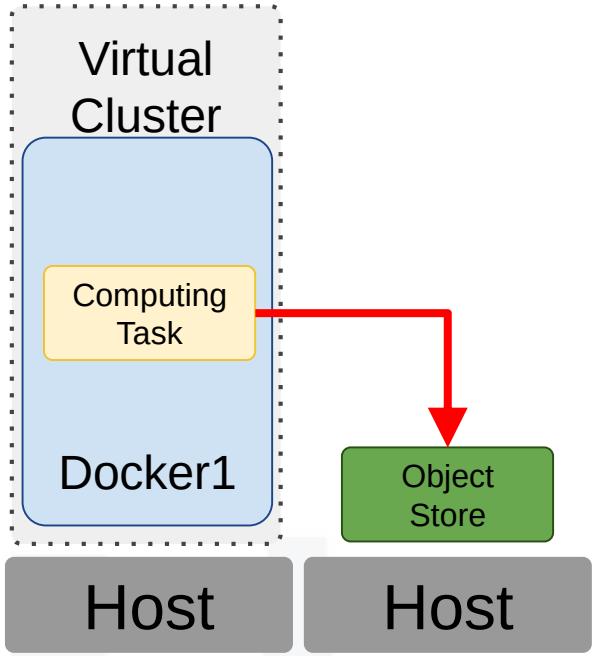
PATTERN 2: External HDFS



Use HDFS as file sharing server.
HDFS runs outside in a long-running cluster to make sure data is persisted.

Please refer to PR-350

PATTERN 3: Object Store



Launch a File Staging Server to share data between nodes.
Input and Output data can put in an object store
Streaming data directly via object level storage like **Amazon S3, Swift**.

Storage Plan for Spark* on K8s*

The design rule is based on "whether the data must be persisted".

spark.local.dir:
For Spark Data Shuffling.
Use Ephemeral Volume.
Now it uses docker-storage with diff. storage backend.
EmptyDir is WIP.

File Staging Server:
For sharing data such as Jar or dependence file between computing nodes.
Now it uses docker-storage.
Local Storage support in Persist Volume(PV) is WIP.

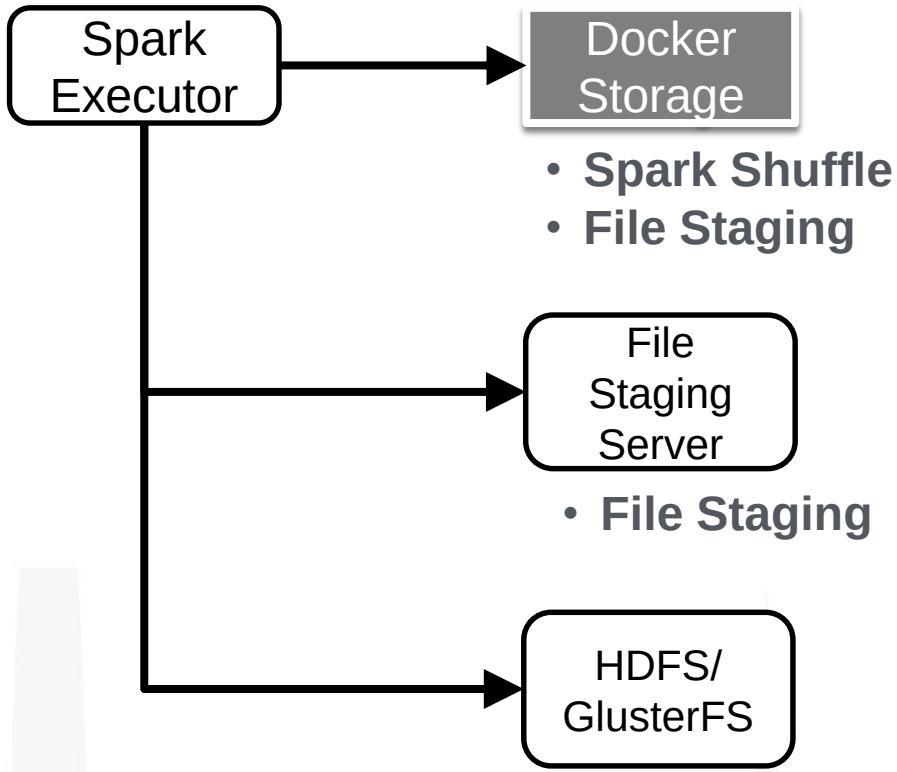
STORAGE SUPPORT

- Spark* Shuffle: Uses Ephemeral Volumes
- Docker* Storage: Use devicemapper
- Shared Volumes:

1. #439 Use EmptyDir for File Staging to share jar file.
2. Local in Spark Executors(Docker Storage)
3. Remote HDFS*
4. Create a Staging Server Container

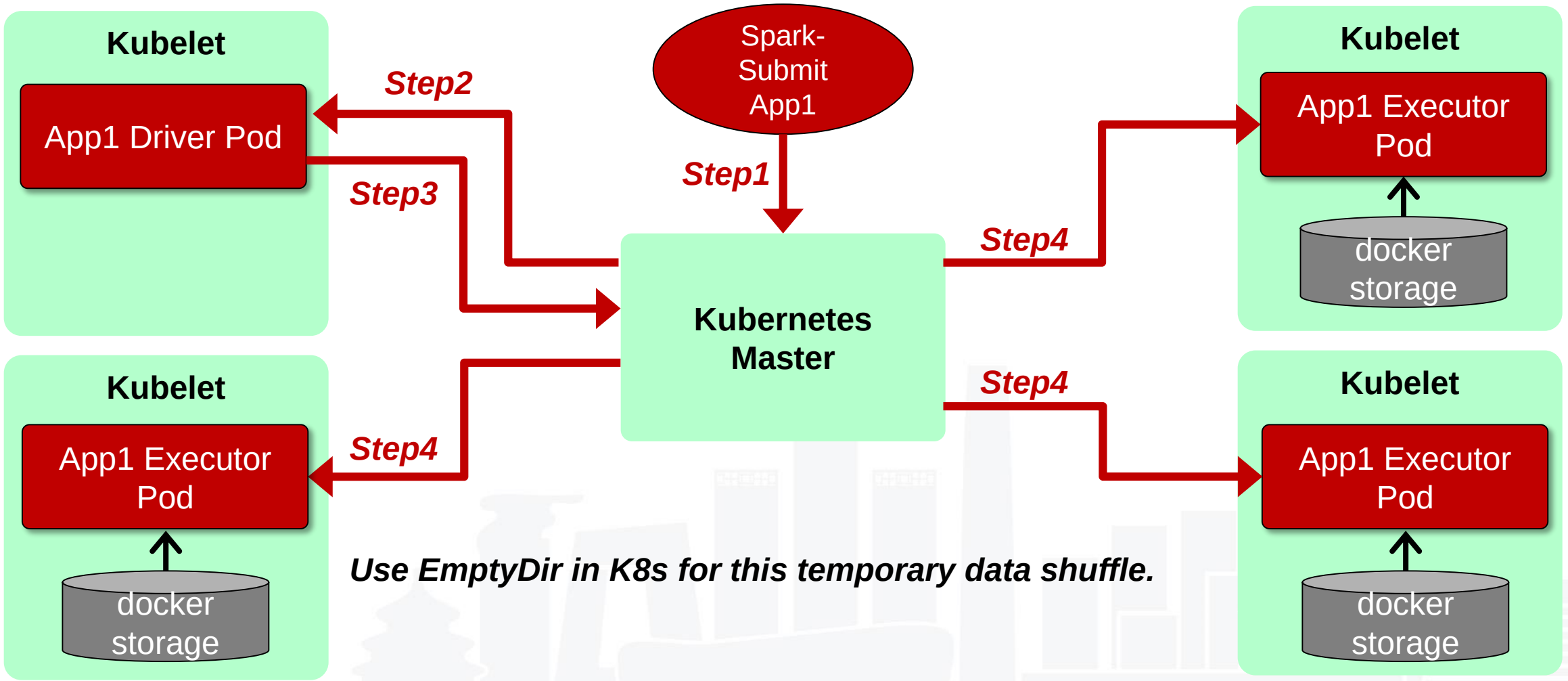
- Persistent Volumes(Ongoing):

1. #306 Use PV
2. Input/Output Data
3. Remote HDFS
4. Remote Object Storage such as GlusterFS*



STATIC RESOURCE ALLOCATION

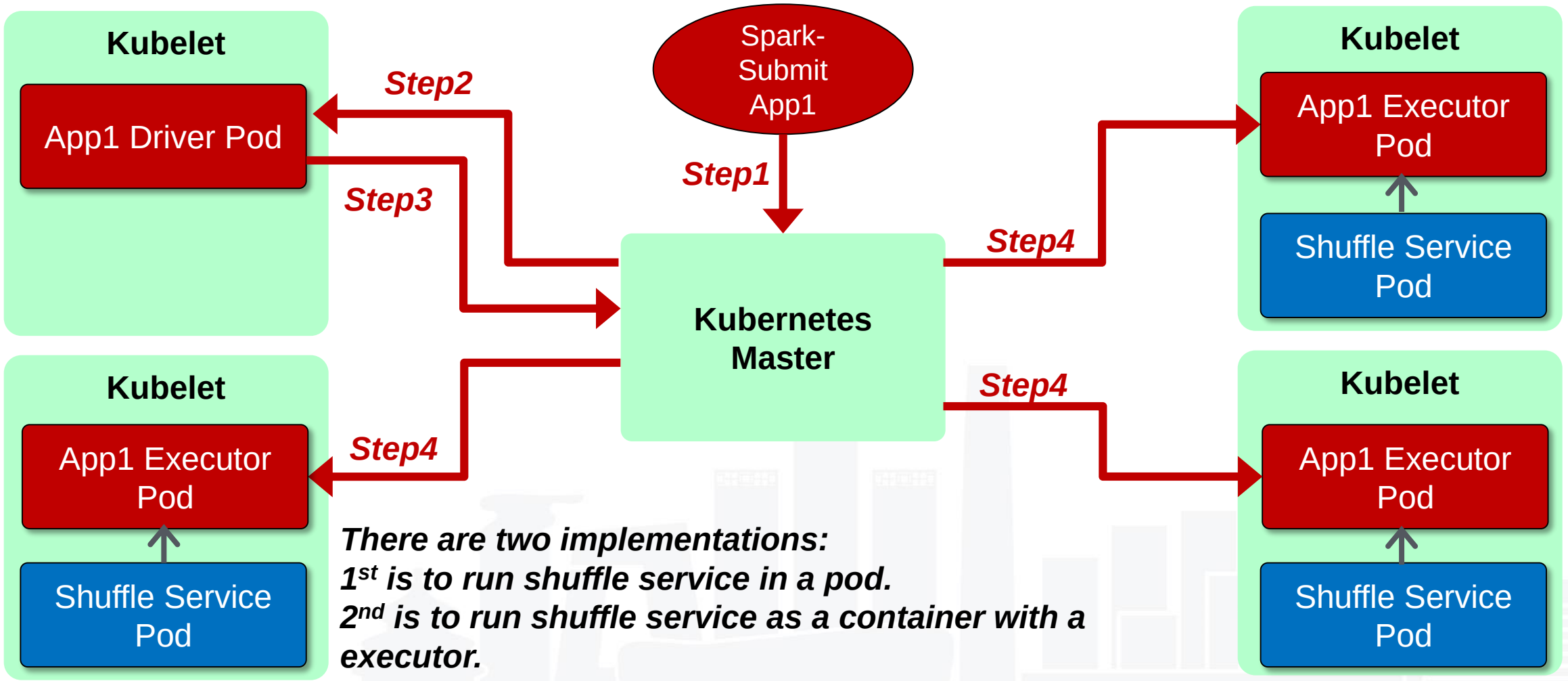
The resources are allocated in the beginning and cannot change during the executors are running.
Static resource allocation uses local storage(docker-storage) for data shuffle.



Use EmptyDir in K8s for this temporary data shuffle.

DYNAMIC RESOURCE ALLOCATION

The resources are allocated in the beginning, but applications can change the resource in run time.
Dynamic resource allocation uses shuffle service container for data shuffle.



There are two implementations:
1st is to run shuffle service in a pod.
2nd is to run shuffle service as a container with a executor.

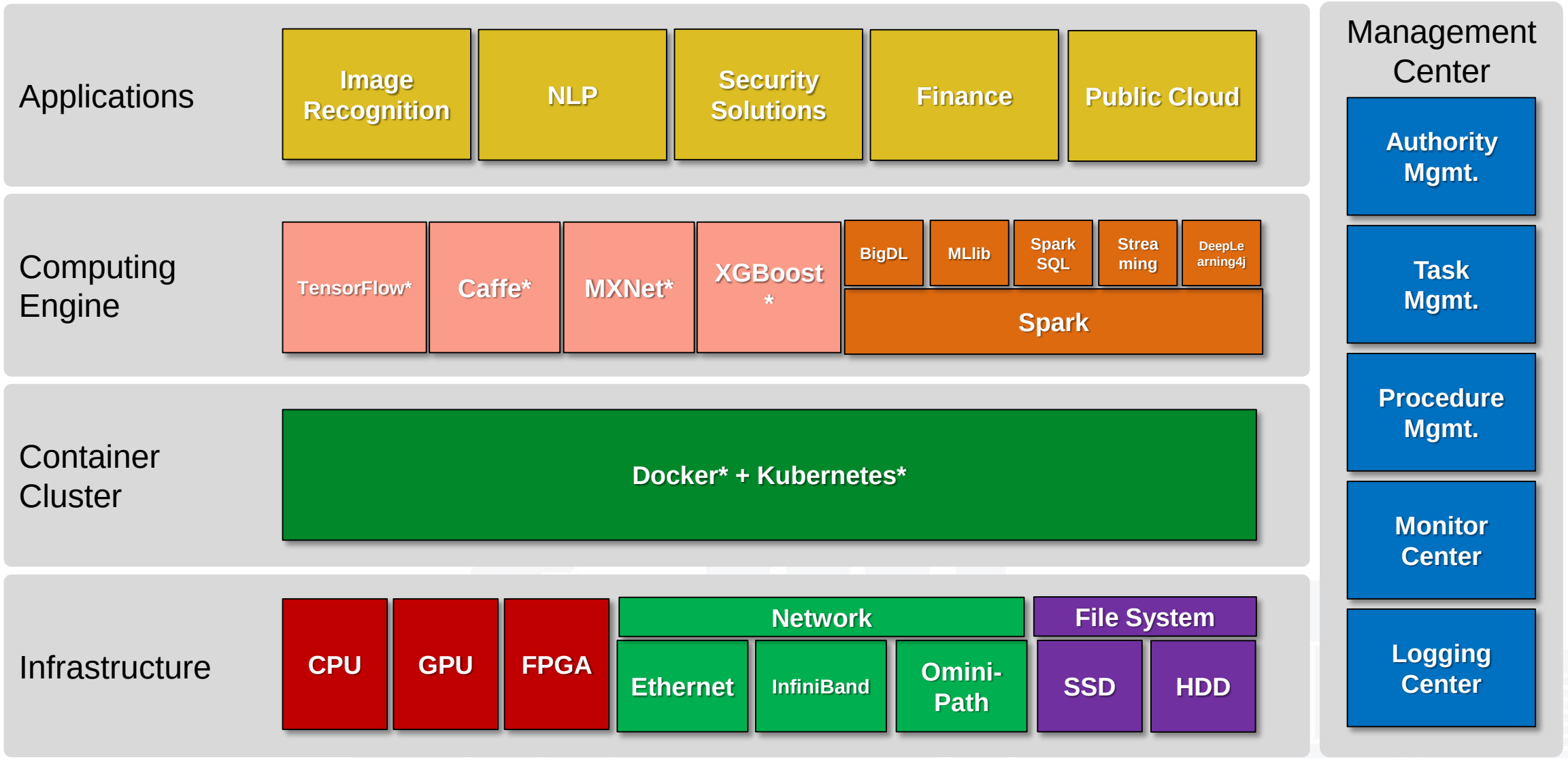
JD.COM CASE STUDY

JD.com MOONSHOT

- JD has used K8s* as cloud infrastructure management for several years.
- JD would like to use K8s to manage all the computing resources including CPU, GPU, FPGA, …etc.
- Target for all AI workloads; Using the same cluster for training/inference.
- Across multiple Machine Learning framework including Caffe, TensorFlow, XGBoost, MXNet, BigDL …etc.
- To optimize workloads for different resource allocation.
- Multi-tenancy support by different user accounts and resource pool.

Reference: https://mp.weixin.qq.com/s?__biz=MzA5Nzc2NDAxMg%3D%3D&mid=2649864623&idx=1&sn=f476db89b3d0ec580e8a63ff78144a37

MOONSHOT ARCHITECTURE



JD.com CONSIDERATION

- Separate Compute and Storage cluster
  - Use Kubernetes as resource manager for compute resource
  - Use Stand-alone HDFS Cluster for data persistent
- Network Choices
  - Host Network for batch job
  - Calico Network for inference job
- Data locality depends on the workload types
- Feature Enabling

NETWORK SOLUTIONS

- Flannel*

  A simple and easy to configure layer 3 network fabric designed for K8s. It runs flanneld on each host to allocate subnet and uses etcd to store network configuration. Flannel supports several backends including VXLAN, host-gw, UDP, …etc.

- Calico*

  An approach to virtual networking and network security for containers, VMs, and bare metal services, which provides a rich set of security enforcement capabilities running on top of a highly scalable and efficient virtual network fabric. Calico uses BGP to set up the network and it also supports IPIP methods to build up a tunnel network.

- Host Network
- OpenVSwitch*
- Others

Why CALICO?

- No overlay required
  Little overhead comparing to bare metal. Sometimes, overlay network(encapsulating packets inside an extra IP header) is an option, not MUST.
- Simple & Scalable
  The architecture is simple, the deployment is simple as well. We can easily deploy thousands of nodes in k8s by using yaml file.
- Policy-driven network security
  In many scenarios of JD.com, for example, multi-tenancy is necessary to make network isolation. Calico enables developers and operators to easily define network policy with fine granularity such as allowed or blocked connections.
- Widely deployed, and proven at scale
  We leverage the experience from other big companies who share their issues in the community. These experience are very valuable for us at the very beginning of moonshot. Fortunately, Calico has passed the verified in our production environment.

NETWORK PERFORMANCE RESULT

All scenarios use ab command to connect to nginx* server with different IP address.
“ab -n 1000000 -c 100 -H"Host: nginx.jd.local" 172.20.141.72:80/index.html “

No.	Scenario	Concurrency #	Total Time(s)	Request per Second	Waiting Time(ms)
1	Client -> Nginx	50	50.044	19982	0.05
2	Weave: Client -> iptables -> Weave -> Pod	50	132.839	7527	0.133
3	Calico with IPIP: Client -> iptables -> Calico -> Pod	50	111.136	8998	0.111
4	Calico with BGP: Client -> iptables -> Calico -> Pod	50	59.218	16886	0.059

JD.com decides to pick up Calico since Calico provides better performance than Weave and Calico can still provide tunnel method(via IPIP) to set up network.

DATA LOCALITY ISSUE

- In cloud environment, compute and storage resource are separated. This could highlight data locality issue with performance drop.
- Some possible solutions can help to resolve data locality issues
- Choose right workloads, most workloads only need to read data and write data at beginning and end phase.
- HDFS* on Kubernetes
- Alluxio*

DATA LOCALITY IMPACT

Workloads	Types	Locality	Datasize	Cluster Size	Network	Execution Time	Notes
Terasort	IO	Local	320GB	5	1Gb	2119.926sec	1x
Terasort	IO	Remote	320GB	5 Spark + 3 Hadoop	1Gb	4212.029sec	1.98x
Terasort	IO	Local	320GB	5	10Gb	500.198sec	1x
Terasort	IO	Remote	320GB	5 Spark + 3 Hadoop	10Gb	548.549sec	1.10x
Kmeans	CPU	Local	240GB	5	10Gb	1156.235sec	1x
Kmeans	CPU	Remote	240GB	5 Spark + 3 Hadoop	10Gb	1219.138sec	1.05x

Note1: This testing is using 5-nodes bare metal cluster.
Note2: 4 SATA SSD per Spark and Hadoop node
Note3: Performance may impact in different configuration including the number of disk, network bandwidth, as well as different platform.

SPARK PERFORMANCE COMPARISON

Task#	Remote in K8s (sec)	Remote in Yarn (sec)	Local in Yarn (sec)	1 - k8s/yarn local (X)	1 - k8s/yarn remote (X)
Task 01	424.33	446.63	398.33	-6.53%	4.99%
Task 02	996.00	1008.00	1155.67	13.82%	1.19%
Task 03	420.67	431.33	391.33	-7.50%	2.47%
Task 04	399.67	423.39	413.33	3.31%	5.60%
Task 05	525.33	552.17	489.33	-7.36%	4.86%
Task 06	659.67	701.31	647.67	-1.85%	5.96%
Task 08	2224.33	2259.00	1717.67	-29.5%	1.53%
Task 10	1013.33	1219.00	958.00	-5.78%	16.87%
Task 11	1539.33	1740.33	1351.33	-13.91%	11.55%

Run 3 times per task and list result in average.
Separate compute and storage bring some performance loss compared to yarn local.
Spark on K8s(remote) provide better performance comparing to yarn remote.

New Feature Enabling in JD.com

Client Mode Support

- Spark SQL must use Client Mode via Spark Shell
- Zeppelin/Jupyter Support

Logging and Monitoring System Support

- Filebeat integration

Multiple Disks for Shuffle Service

- Single Disk(EmpDir) by default

Distributed Image Repository

- Need to support Huge Traffic

SUMMARY

CURRENT ISSUES

- Spark* Shell for Client Mode hasn't verified yet.
- PySpark Support
- Dynamic Resource Allocation Support
- Storage Backend Support
- Staging Server in Spark and Hadoop
- Container Launch Time may take too long(DNS Resolved Issue)
- Reliability needs to be verified
  - Run more workloads in long time

FUTURE WORKS

- Spark on K8s Feature Support
- Needs more Customer Case Study
- Intel Feature Enabling
  - Intel RDT integration
    https://www.intel.com/content/www/us/en/architecture-and-technology/resource-director-technology.html
  - Intel DPDK for Software Define Network(SDN)
  - Intel HW Features Enabling

SUMMARY

- Spark* on K8s* provides a cloud native way to run Spark on Cloud which not only can get better resource utilization but also integrate with more big data services.
- JD.com’s Moon Shot uses K8s to create a heterogeneous cloud infrastructure, it can support both CPU and GPU for their AI workloads.
- Spark on K8s is still under developing and there are many issues/features are waiting to be fixed/implemented.

关注QCon微信公众号，
获得更多干货！

Thanks!

INTERNATIONAL SOFTWARE DEVELOPMENT CONFERENCE

主办方 Geekbang. 极客邦科技 InfoQ

IN-CLUSTER & OUT-CLUSTER CLIENT MODE