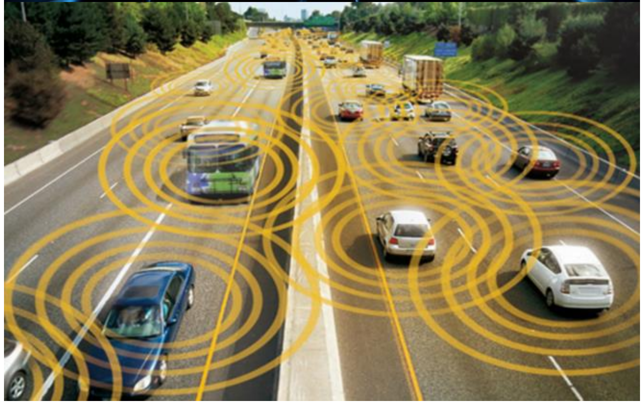
Bring Your Own Models (BYOM)- Machine Learning as a Service

Malini Bhandaru, Soila Kavulya & Luis Daniel Castellanos
Contributors: Konrad Kurdej, Weiting Chen
INTEL

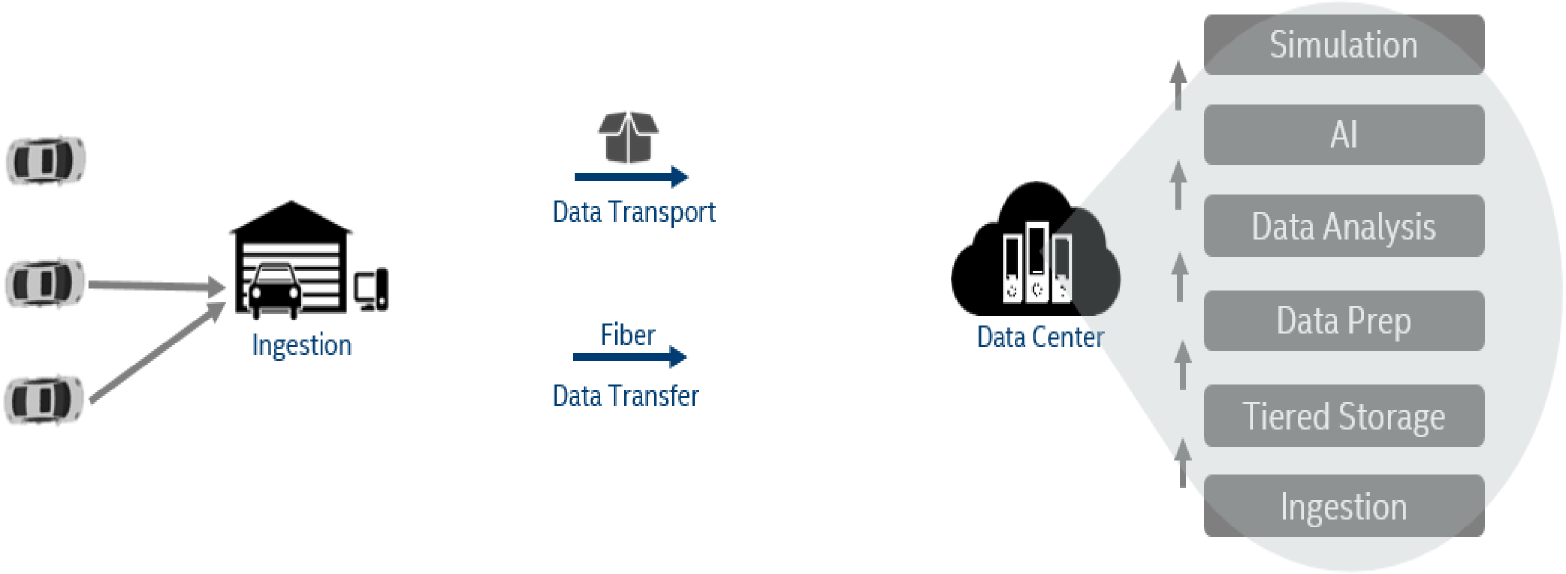
#ML9SAIS

Machine Learning Everywhere!

- Autonomous Vehicles
- Genomics
- Finance
- Supply Chain



Autonomous Vehicles R&D Data Center



Big Data

1 – 20 TB/car/hour
- Weather Conditions
- Time of Day
- Road Conditions
- Location
- Edge Cases

Object Detection Models
Environment Models
Driver Models
Privacy Preservation Models

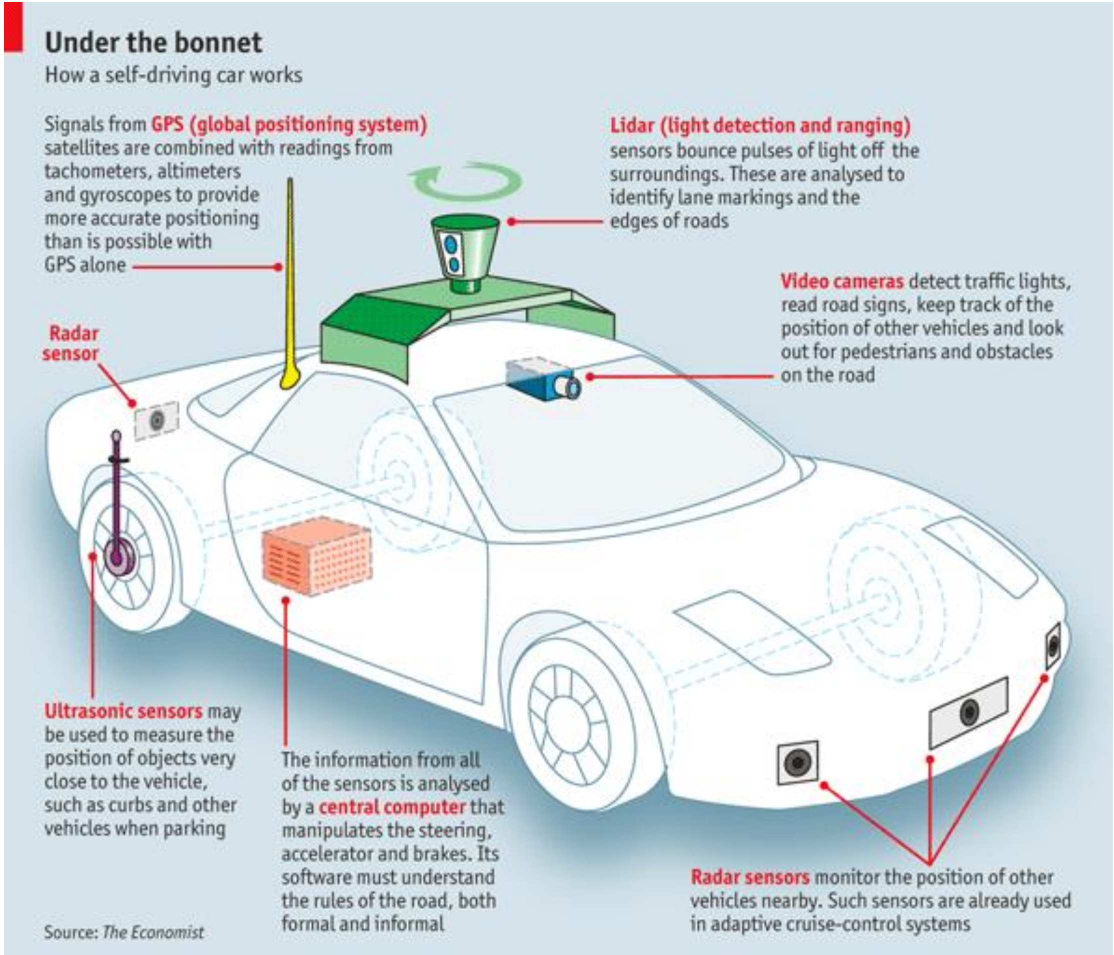


Image credit: https://www.wowwoodys.com/our-future-autonomous-cars/

Data Center Platform

- Fungible, Dynamic, Fast,
- Resilient
- Easy to Use

Storage
HDFS, Ceph, MySQL, S3

Orchestrator Stacks
Spark,
Hadoop,
YARN
Kubernetes

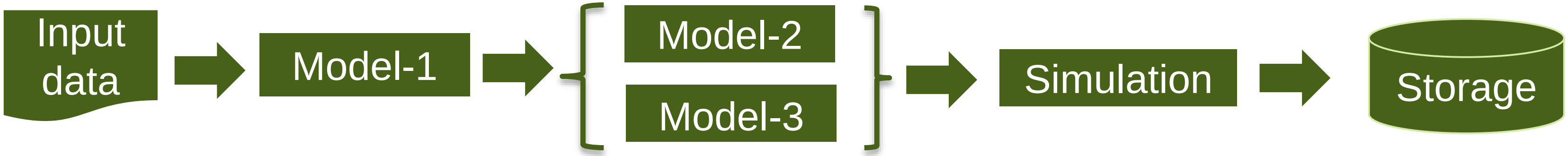
Hetero Hardware
CPUs
GPUS
FPGAs
Programmable
Inference Chips

AI Frameworks
TensorFlow
Caffe2, ..

Glue Technologies
- Kafka
- Oozie
- Argo

Containers

Models Galore, Usages Rich



MODEL
Name, Description,
type (mega | lean)
Framework & Version
Input, Output
ImageID: Container Registry ID
Training_Sets: { S1, S2 , S3}
Training_Label_Freq {L1:f1, L2:f2 ..}
Validation_Sets: {V1, V2}
Accuracy, Recall, Precision,
Speed, Size
Infrastructure:CPU/GPU/FPGA ..

Resources & API

Model

- **CRUD, Validate dependencies,**
- name, description, framework, version, hardware preference
- Tags (sharable, input-sensor ..)

Data Transformer

- **CRUD**
- Image resizer, compression, crypto, ..

Pipeline

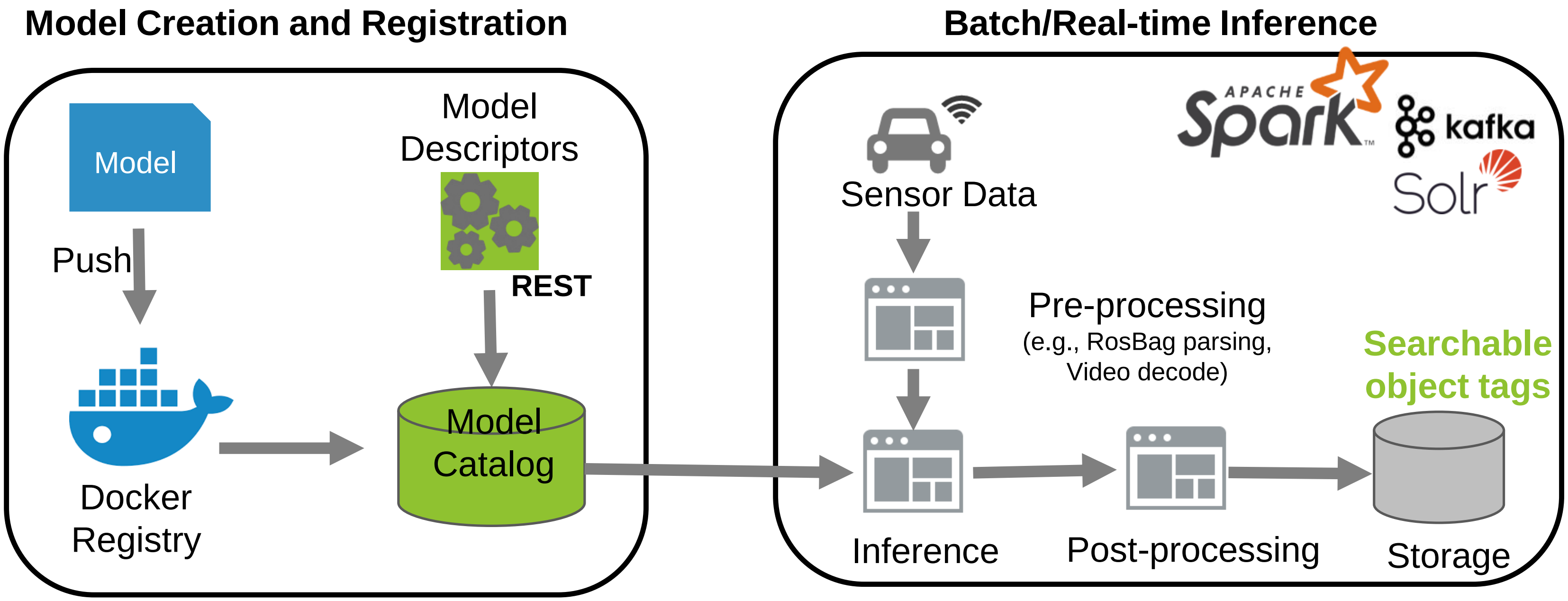
- **CRUD, Start/Stop/Pause**
- workflow specifications, language

Dataset

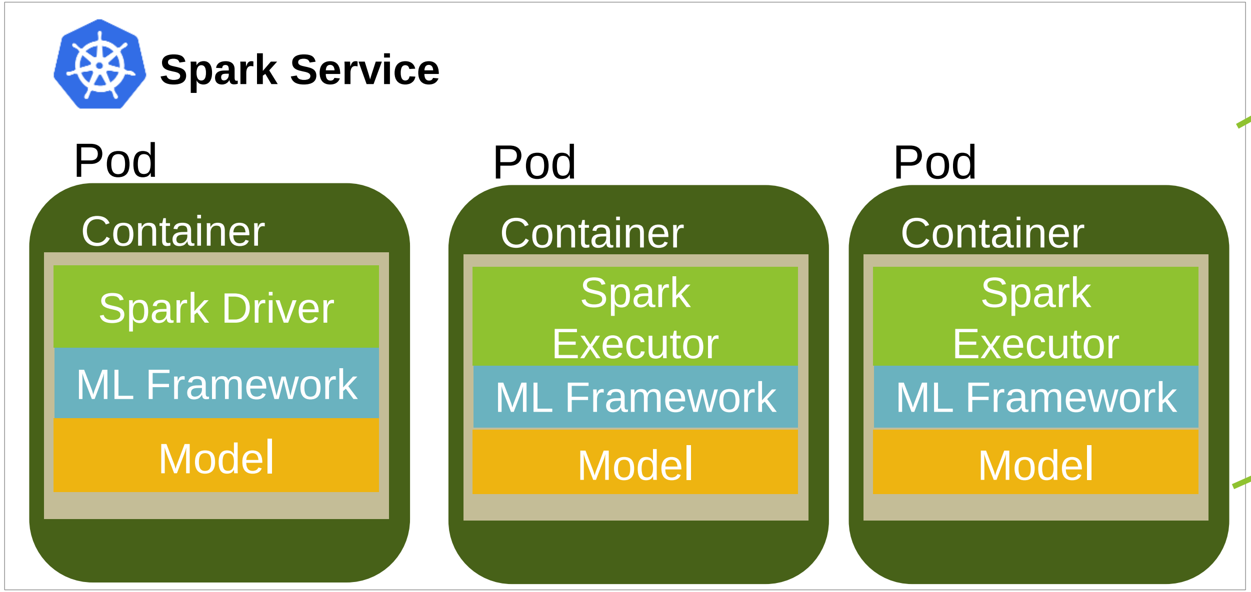
- **CRUD**
- Name, description, data location
- (s3,hdfs, local file system)

Apache Beam, Spark, Argo

Model Deployment Pipeline



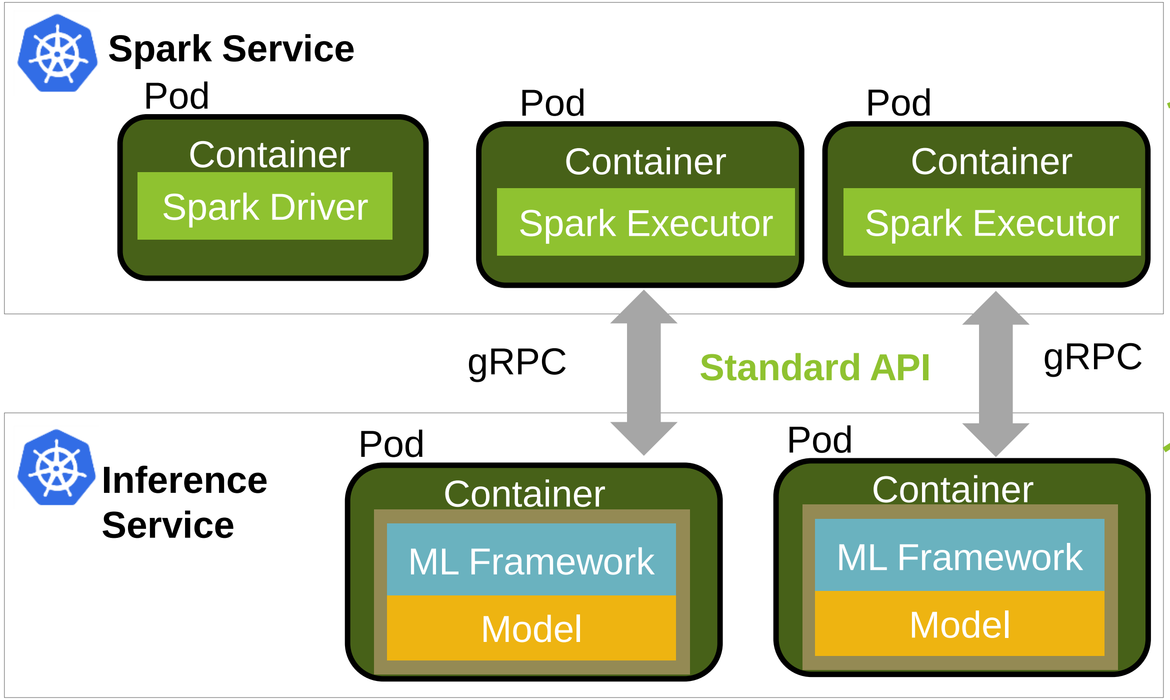
BYOM Options: Monolithic



Pros
- Simple deployment
- Container life-cycle in full sync with workload
- No version tracking or mismatch concerns
- Data locality

Cons
- Larger container footprint
- Tight coupling between model and Spark engine

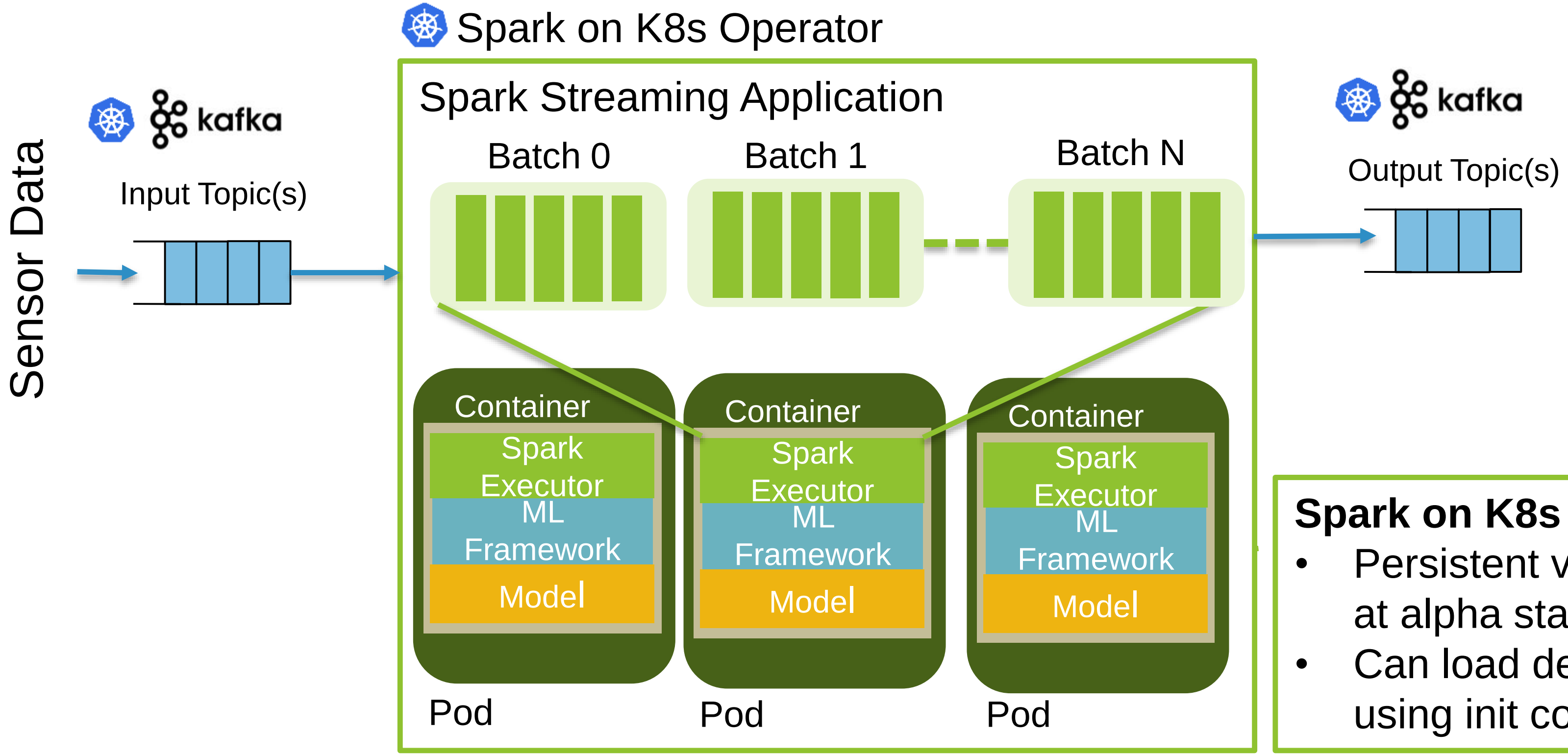
BYOM Options: Just-In-Time Compose



Pros
- Small container footprint
- Multi-framework friendly
- Auto scales
- Standard API

Cons
- More complex orchestration workflow
- Additional mechanisms needed for data locality, e.g., pod affinity

Deployment on K8s



Spark on K8s operator

- Persistent volumes are at alpha stage
- Can load dependencies using init container

Demo

https://videoportal.intel.com/media/0_70vbt74e

Future

- Support for Just-in-time-Composition
  - Tackling dependencies
- Resource scheduling, HW accelerator aware
- Hardware specific models (CPU/GPU/FPGA ..)
- Pipeline options with speed, accuracy, and resource availability projections

Conclusion

- Across domains Bring-your-own-model genuine need for both R&D and Production Systems
- System and Orchestration developers typically not Machine Learning specialists –
  - reduce the barrier to adoption

SPARK+AI SUMMIT 2018

Thank You!

Kavulya, Soila P soila.p.kavulya@intel.com

Luis Daniel Castellanos (luis.daniel.castellanos@intel.com)

Kurdej, Konrad konrad.kurdej@intel.com

Bhandaru, Malini K malini.k.bhandaru@intel.com

Chen, Weiting <weiting.chen@intel.com>

Please join us in the BYOM effort!

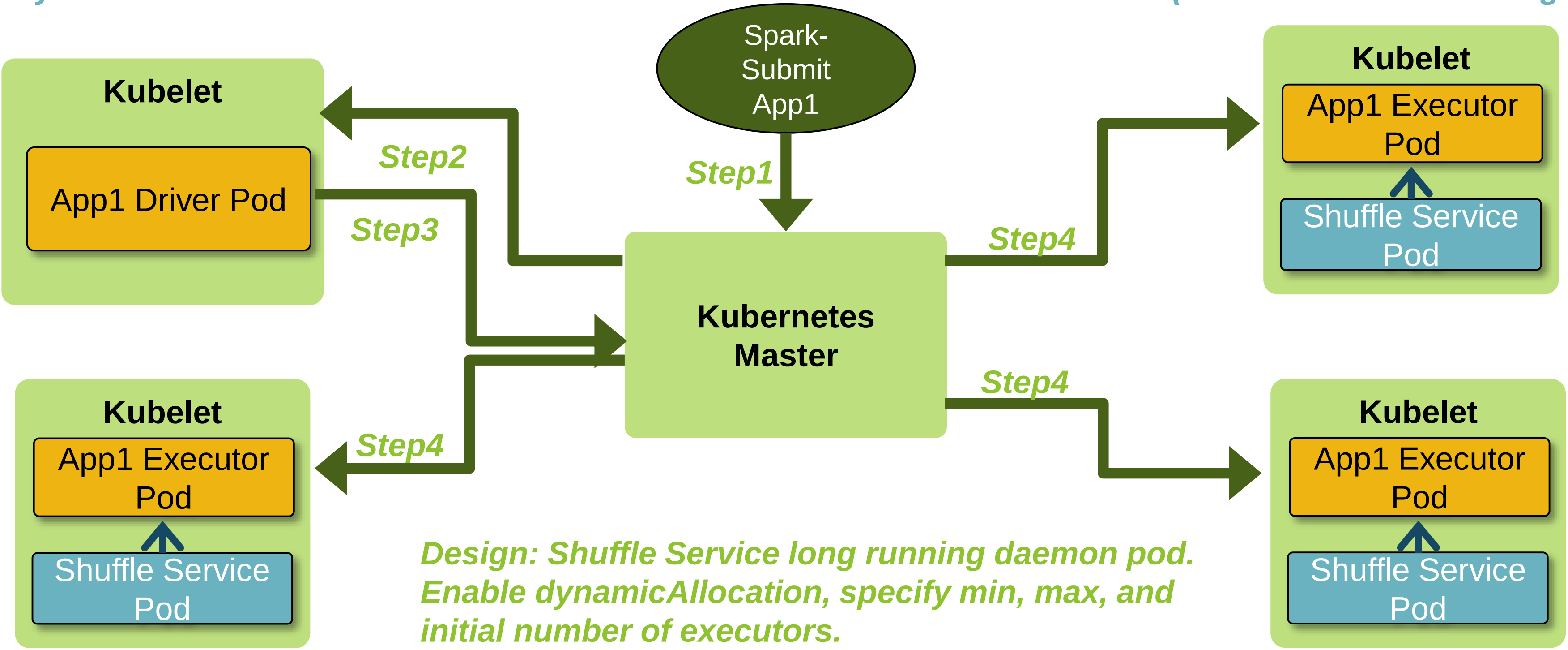
References

- Autonomous Driving : https://medium.com/@andrewng/self-driving-cars-are-here-aea1752b1ad0, https://www.wired.com/story/embark-self-driving-truck-deliveries
- Docker Registry: https://docs.docker.com/registry
- Spark on Kubernetes: https://github.com/apache-spark-on-k8s/spark
- HDFS on K8: https://spark-summit.org/2017/events/hdfs-on-kubernetes-lessons-learned/
- RDD: https://www.slideshare.net/datamantra/anatomy-of-rdd
- TensorFlow Serving: https://www.tensorflow.org/serving/
- https://jaceklaskowski.gitbooks.io/mastering-apache-spark/content/spark-dynamic-allocation.html
- Kubeflow: ML toolkit for Kubernetes: https://github.com/kubeflow/kubeflow, https://cloud.google.com/blog/big-data/2018/03/simplifying-machine-learning-on-open-hybrid-clouds-with-kubeflow
- https://beam.apache.org/documentation/pipelines/design-your-pipeline/

Upstreamed: Dynamic Resource Allocation

Weiting Chen

Resources are allocated at start but applications can request change at runtime.
Dynamic resource allocation uses shuffle service container for data shuffle (instead of Docker storage)



Design: Shuffle Service long running daemon pod.
Enable dynamicAllocation, specify min, max, and initial number of executors.