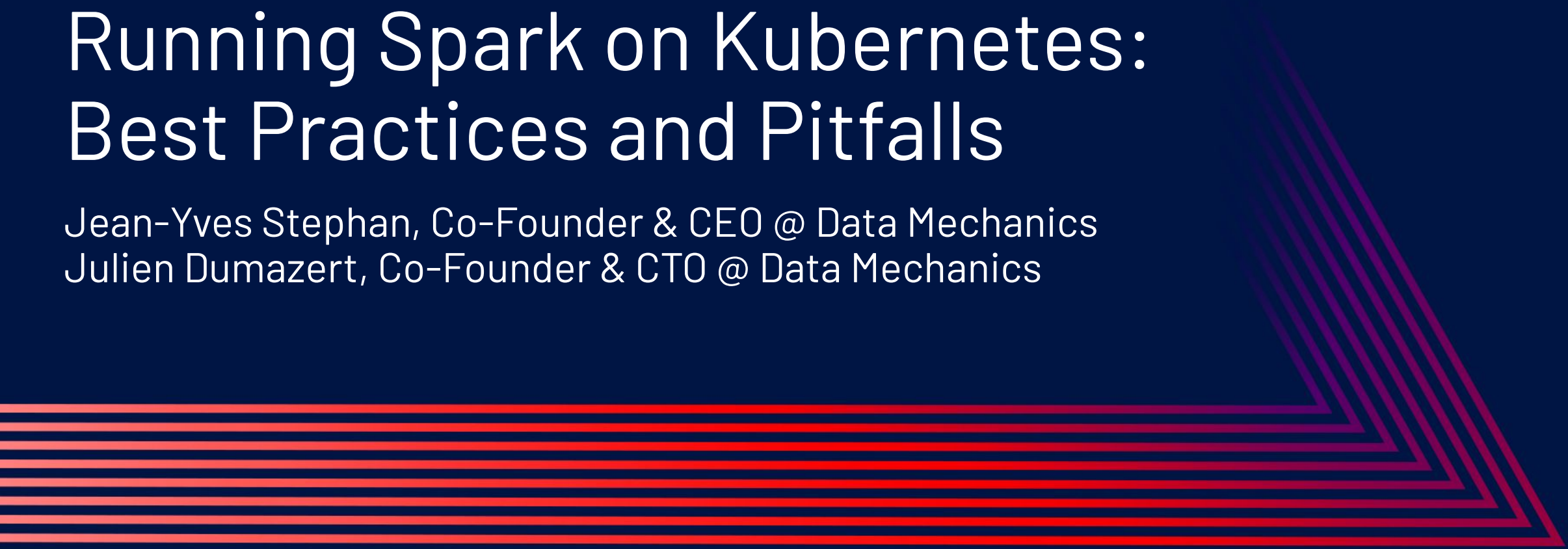
Running Spark on Kubernetes: Best Practices and Pitfalls

Jean-Yves Stephan, Co-Founder & CEO @ Data Mechanics
Julien Dumazert, Co-Founder & CTO @ Data Mechanics



SPARK+AI SUMMIT

Who We Are

Jean-Yves "JY" Stephan
Co-Founder & CEO @ Data Mechanics
jy@datamechanics.co

Previously:
Software Engineer and
Spark Infrastructure Lead @ Databricks

Julien Dumazert
Co-Founder & CTO @ Data Mechanics
julien@datamechanics.co

Previously:
Lead Data Scientist @ ContentSquare
Data Scientist @ BlaBlaCar

Who Are You?

Poll: What is your experience with running Spark on Kubernetes?

- **61%** - I've never used it, but I'm curious about it.
- **24%** - I've prototyped using it, but I'm not using it in production.
- **15%** - I'm using it in production.

This slide was edited after the conference to show the results for the poll.
You can see and take the poll at https://www.datamechanics.co/spark-summit-poll

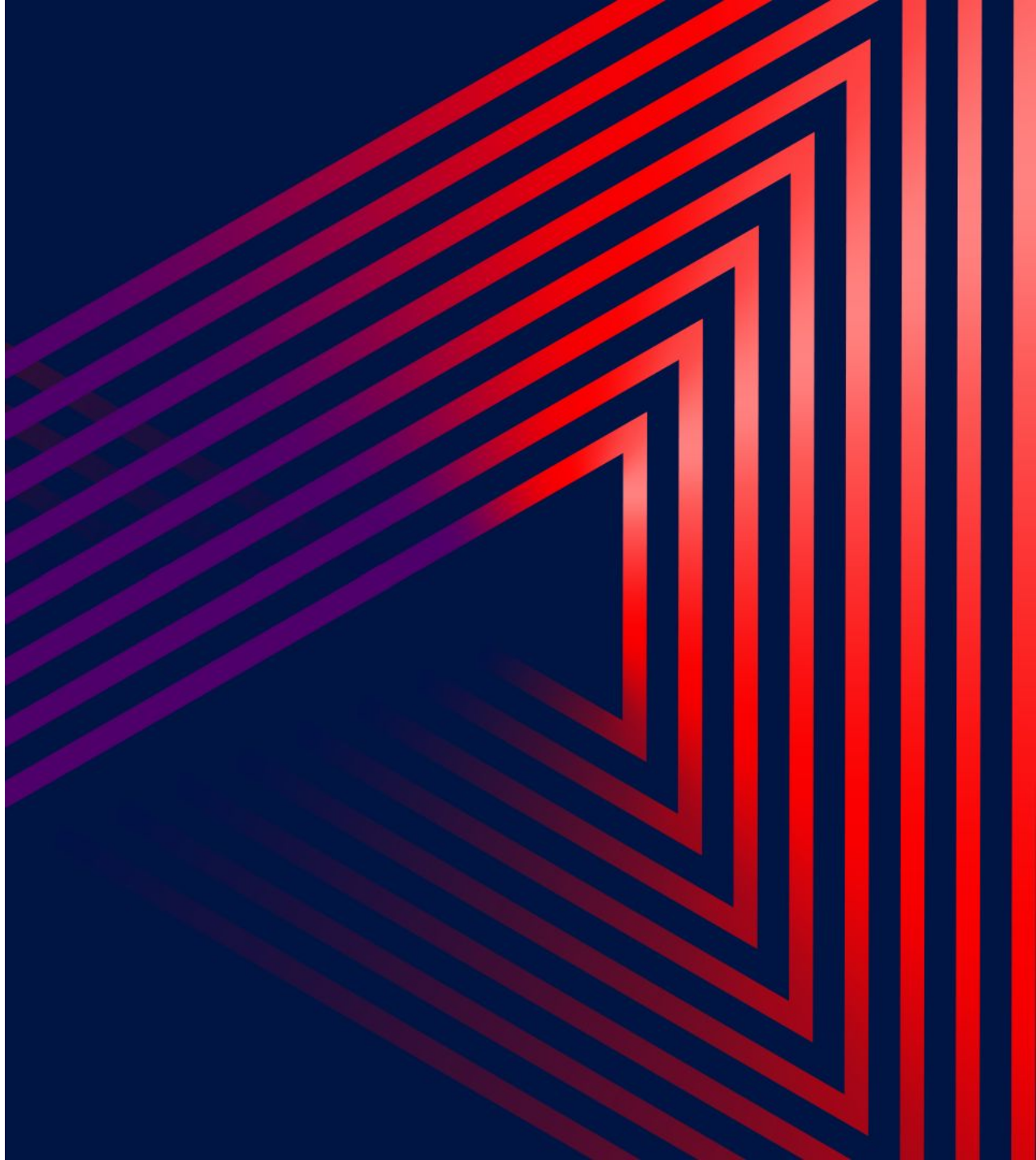
Agenda

A quick primer on Data Mechanics

Spark on Kubernetes

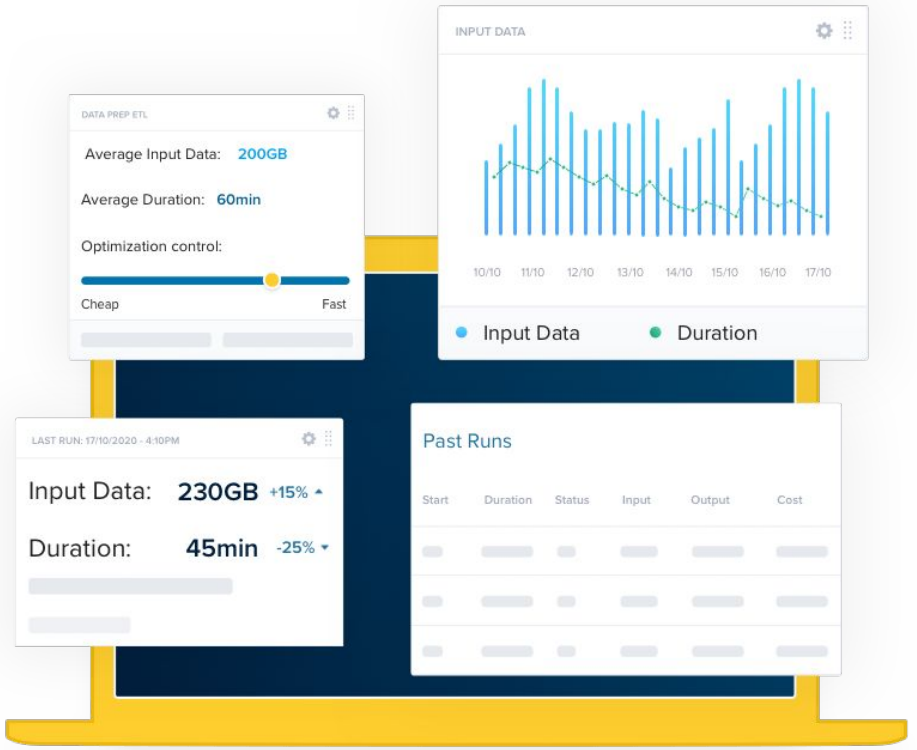
- Core Concepts & Setup
- Configuration & Performance Tips
- Monitoring & Security
- Future Works

Conclusion: Should you get started?



Data Mechanics - A *serverless* Spark platform

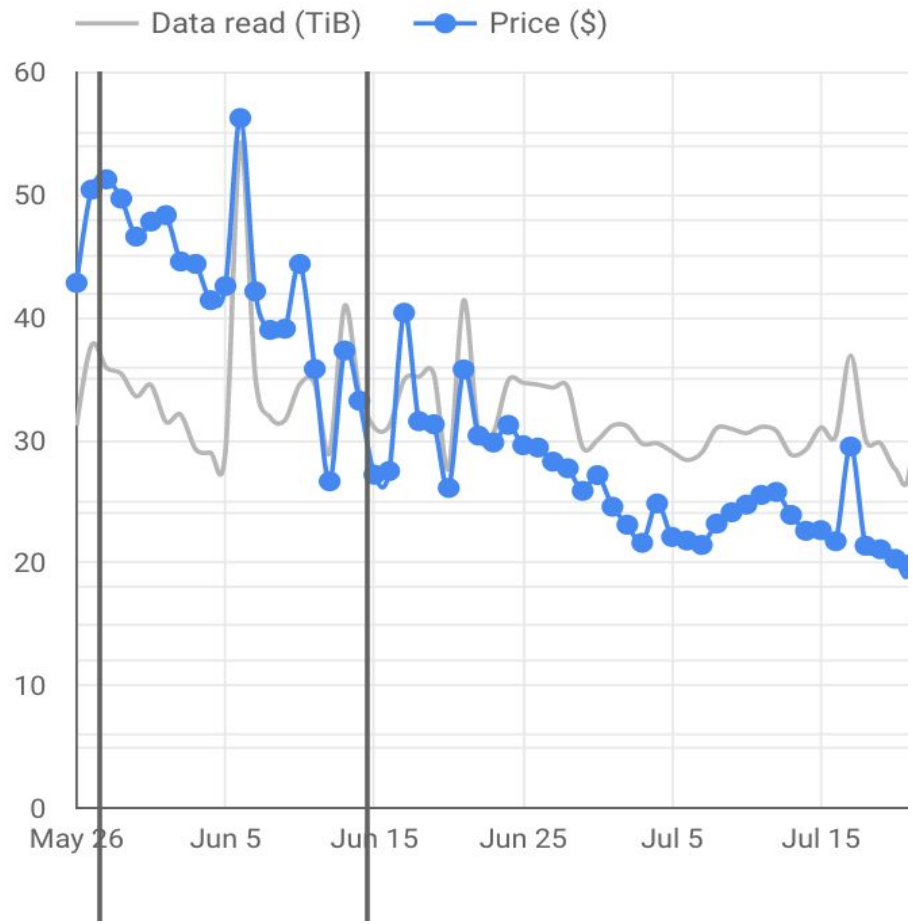
- Applications start and autoscale in seconds.
- Seamless transition from local development to running at scale.
- Tunes the infra parameters and Spark configurations automatically for each pipeline to make them fast and stable.



https://www.datamechanics.co

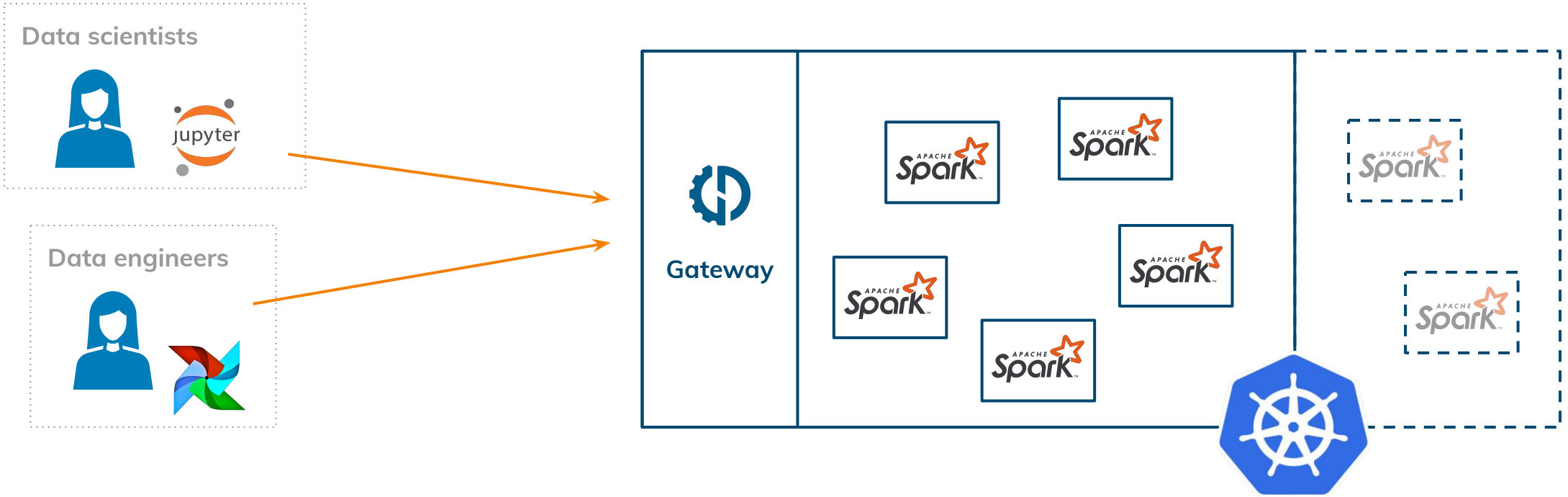
Customer story: Impact of automated tuning on Tradelab

- **Stability**: Automatic remediation of OutOfMemory errors and timeouts
- **2x** performance boost on average (speed and cost savings)



For details, watch our SSAI 2019 Europe talk
How to automate performance tuning for Apache Spark

We're deployed on k8s in our customers cloud account



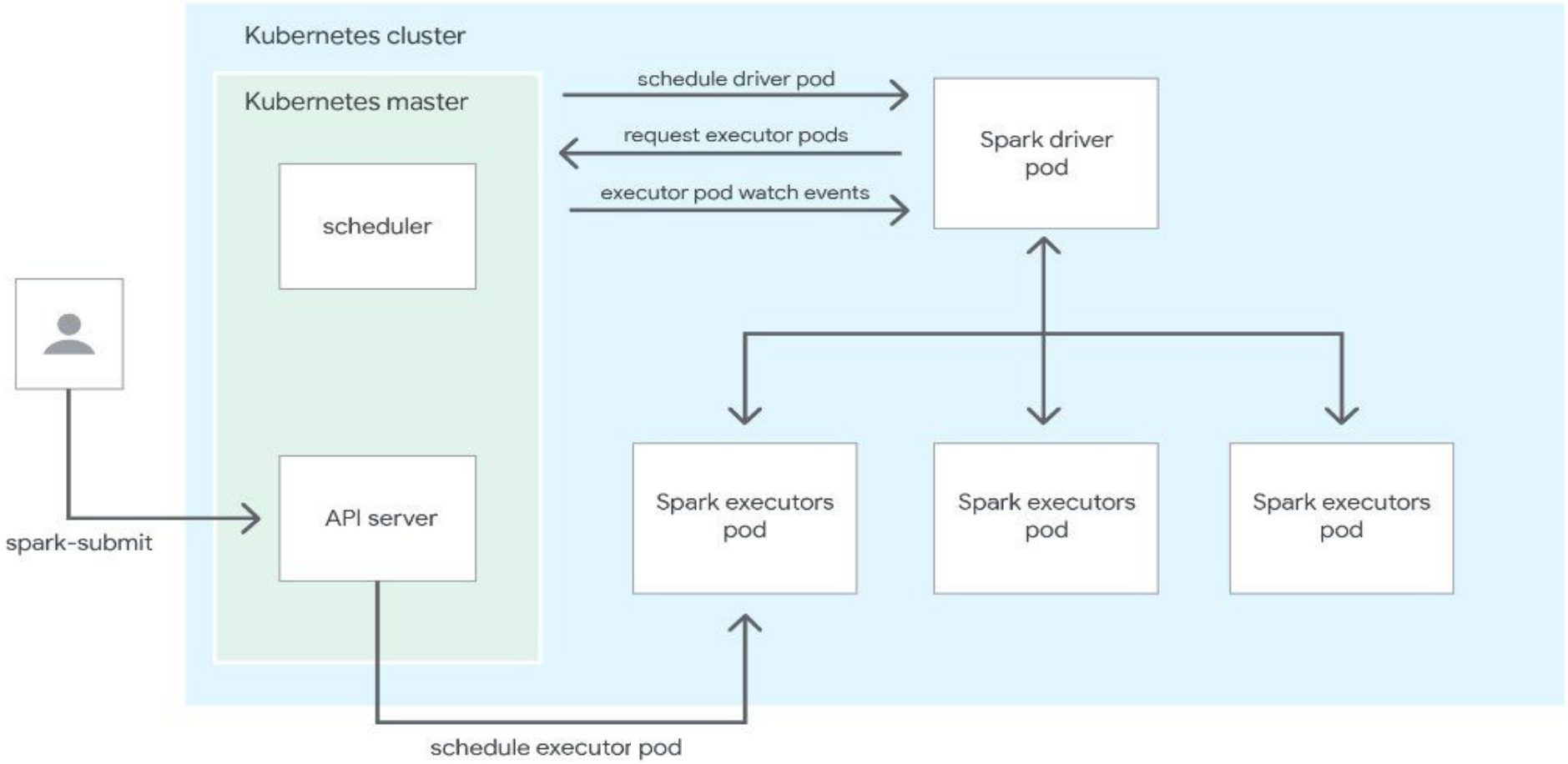
Spark on Kubernetes: Core Concepts & Setup

Where does Kubernetes fit within Spark?

Kubernetes is a new cluster-manager/scheduler for Spark.

- Standalone
- Apache Mesos
- Yarn
- Kubernetes (since version 2.3)

Spark on Kubernetes - Architecture



Source

Two ways to submit Spark applications on k8s

Spark-submit

- "Vanilla" way from Spark main open source repo
- Configs spread between Spark config (mostly) and k8s manifests
- Little pod customization support before Spark 3.0
- App management is more manual

spark-on-k8s operator

- Open-sourced by Google (but works on any platform)
- Configs in k8s-style YAML with sugar on top (configmaps, volumes, affinities)
- Tooling to read logs, kill, restart, schedule apps
- Requires a long-running system pod

App management in practice

Spark-submit

```
# Run an app
$ spark-submit --master k8s://https://<api-server> …

# List apps
k get pods -label "spark-role=driver"
NAME            READY   STATUS      RESTARTS   AGE
my-app-driver   0/1     Completed   0          25h

# Read logs
k logs my-app-driver

# Describe app
# No way to actually describe an app and its parameters…
```

spark-on-k8s operator

```
# Run an app
$ kubectl apply -f <app-manifest>.yaml

# List apps
$ k get sparkapplications
NAME     AGE
my-app   2d22h

# Read logs
sparkctl log my-app

# Describe app
$ k get sparkapplications my-app -o yaml
apiVersion: sparkoperator.k8s.io/v1beta2
kind: SparkApplication
  arguments:
  - gs://path/to/data.parquet
  mainApplicationFile: local:///opt/my-app/main.jar
  ...
status:
  applicationState:
    state: COMPLETED
  ...
```

Dependency Management Comparison

YARN

- Lack of isolation
  - Global Spark version
  - Global Python version
  - Global dependencies
- Lack of reproducibility
  - Flaky Init scripts
  - Subtle differences in AMIs or system

Kubernetes

- Full isolation
  - Each Spark app runs in its own docker container
- Control your environment
  - Package each app in a docker image
  - Or build a small set of docker images for major changes and specify your app code using URIs

Spark on Kubernetes: Configuration & Performance Tips

A surprise when sizing executors on k8s

Assume you have a k8s cluster with 16GB-RAM 4-core instances.

Do one of these and you'll never get an executor!

- Set spark.executor.cores=4
- Set spark.executor.memory=11g

k8s-aware executor sizing

What happened?

→ Only a fraction of capacity is available to Spark pods, and spark.executor.cores=4 requests 4 cores!

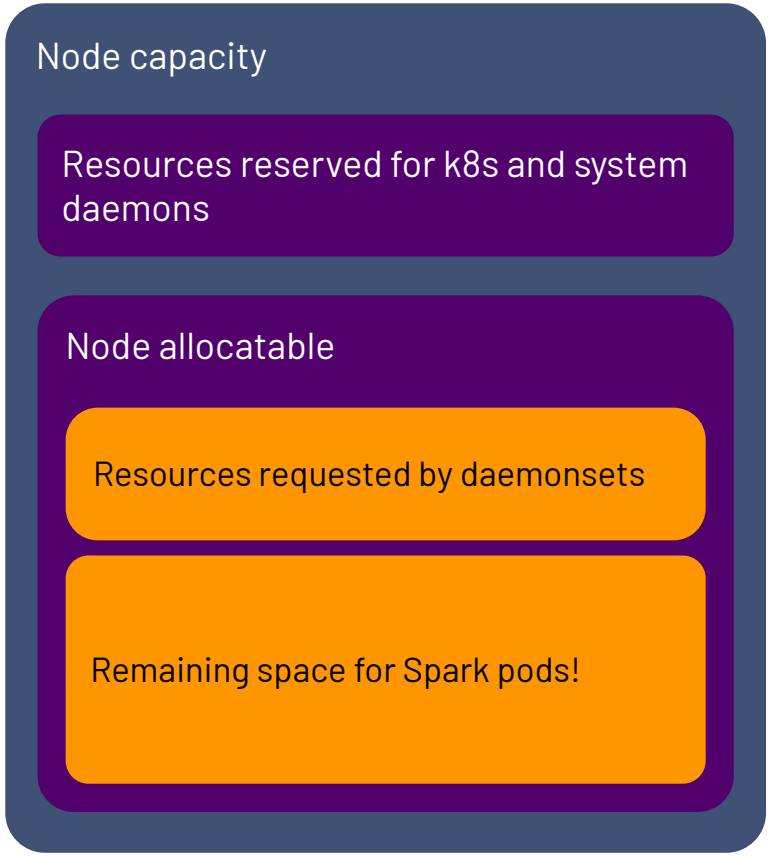
Compute available resources

- Estimate node allocatable: usually 95%
- Measure what's taken by your daemonsets (say 10%)

→ 85% of cores are available

Configure Spark

spark.executor.cores=4
spark.kubernetes.executor.request.cores=3400m



More configuration tips here

Dynamic allocation on Kubernetes

- Full dynamic allocation is not available. When killing an exec pod, you may lose shuffle files that are expensive to recompute.
  There is ongoing work to enable it (JIRA: SPARK-24432).

- In the meantime, a soft dynamic allocation is available from Spark 3.0
  Only executors which do not hold active shuffle files can be scaled down.

```
spark.dynamicAllocation.enabled=true
spark.dynamicAllocation.shuffleTracking.enabled=true
```

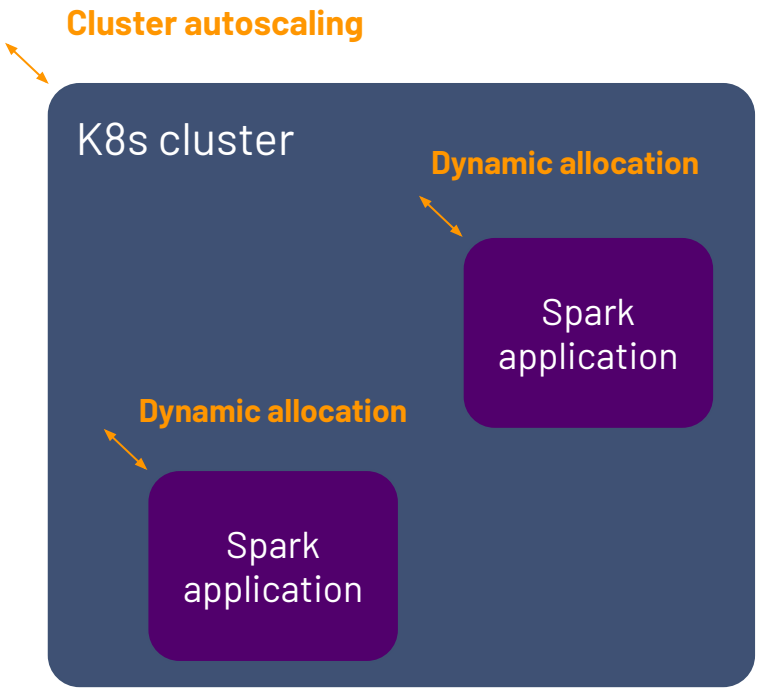
Cluster autoscaling & dynamic allocation

k8s can be configured to autoscale if pending pods cannot be allocated.

Autoscaling plays well with dynamic allocation:

- <10s to get a new exec if there is room in the cluster
- 1-2 min if the cluster needs to autoscale

Requires to install the cluster autoscaler on AKS (Azure) and EKS (AWS). It is natively installed on GKE (GCP).

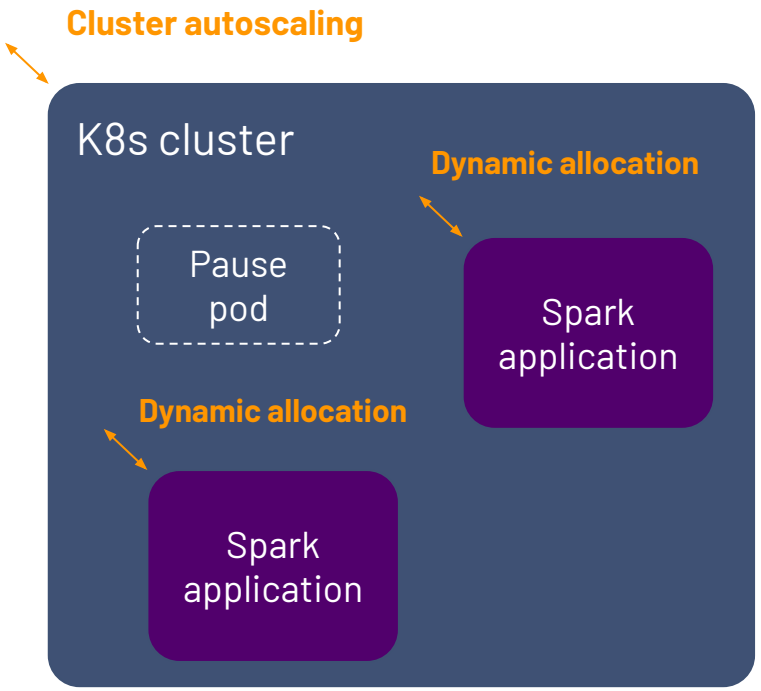


Overprovisioning to speed up dynamic allocation

To further improve the speed of dynamic allocation, overprovision the cluster with low-prio pause pods:

- The pause pods force k8s to scale up
- Spark pods preempt pause pods' resources when needed

Cluster autoscaler doc about overprovisioning.

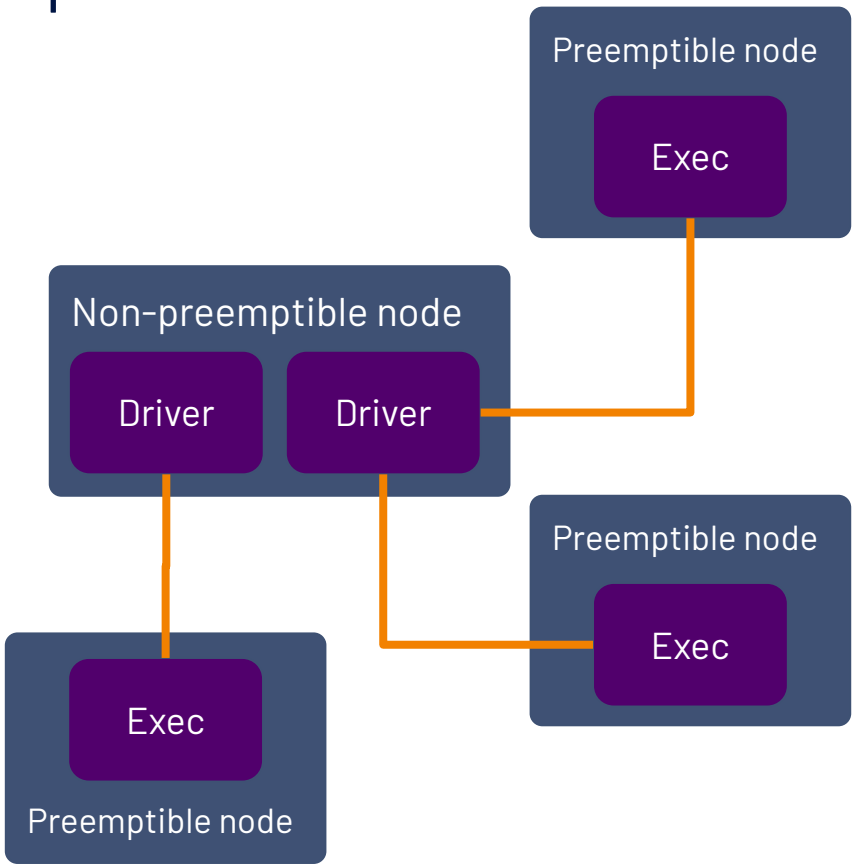


Further cost reduction with spot instances

Spot (or preemptible) instances can reduce costs up to 75%.

- If an executor is killed, Spark can recover
- If the driver is killed, game over!

Node selectors and affinities can be used to constrain drivers on non-preemptible nodes.



I/O with an object storage

Usually in Spark on Kubernetes, data is read and written to an object storage.

Cloud providers write optimized committers for their object storages, like the S3A Committers.

If it's not the case, use the version 2 of the Hadoop committer bundled with Spark:

```
spark.hadoop.mapreduce.fileoutputcommitter.algorithm.version=2
```

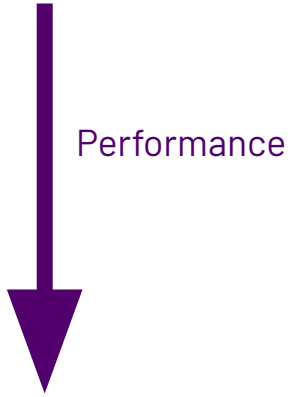
The performance boost may be up to 2x! (if you write many files)

Improve shuffle performance with volumes

I/O speed is critical in shuffle-bound workloads, because Spark uses local files as scratch space.

Docker filesystem is slow → Use volumes to improve performance!

- emptyDir: use a temporary directory on the host (by default in Spark 3.0)
- hostPath: Leverage a fast disk mounted in the host (NVMe-based SSD)
- tmpfs: Use your RAM as local storage (⚠️ dangerous)



We ran performance benchmarks to compare Kubernetes and YARN.
Results will be published on our blog early July 2020.
(Sneak peek: There is no performance penalty for running on k8s if you follow our recommendations)

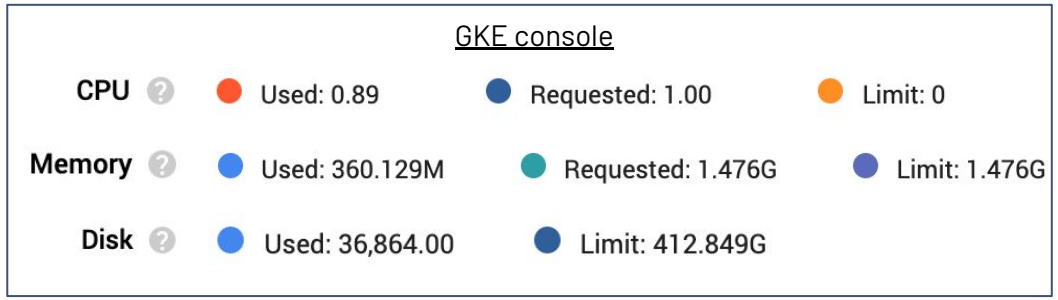
Spark on Kubernetes: Monitoring & Security

Monitor pod resource usage with k8s tools

Workload-agnostic tools to monitor pod usages:

- Kubernetes dashboard (installation on EKS)
- The GKE console

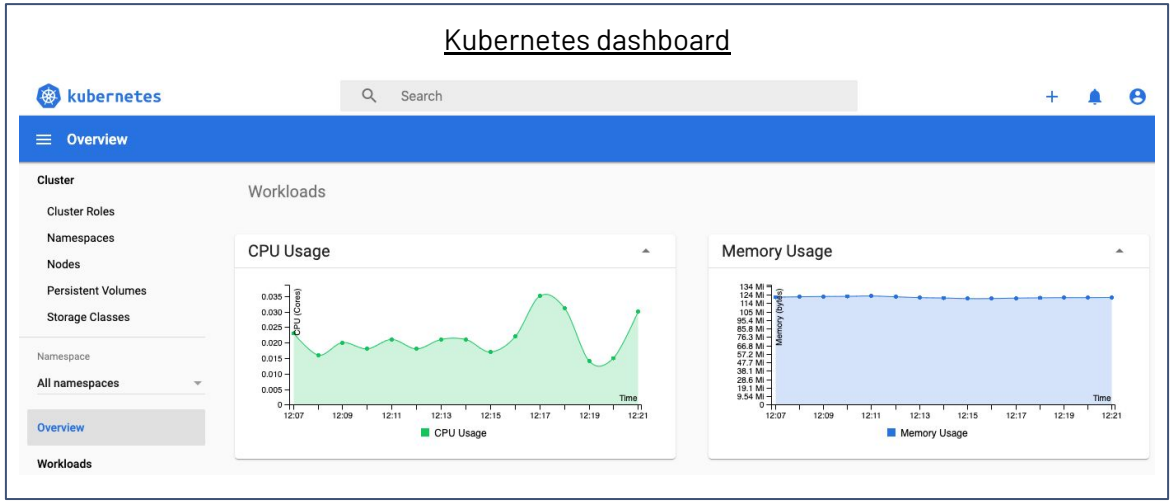
GKE console



Issues:

- Hard to reconcile with Spark jobs/stages/tasks
- Executors metadata are lost when the Spark app is completed

Kubernetes dashboard



Spark history server

Setting up a History server is relatively easy:

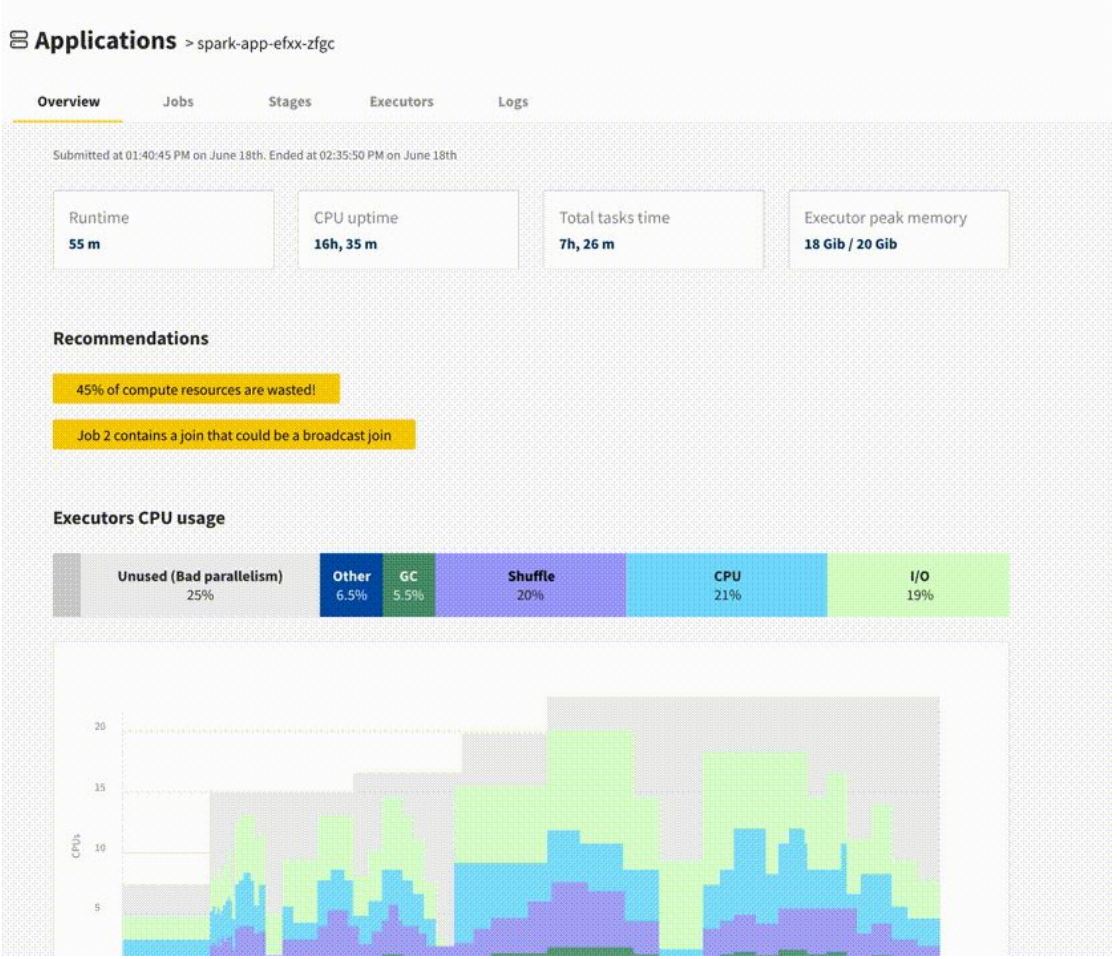
- Direct your Spark event log file to S3/GCS/Azure Storage Account with the spark.eventLog.dir config
- Install the Spark history server Helm chart on your cluster

What's missing: resource usage metrics!

“Spark Delight” - A Spark UI replacement

- We’re building a better Spark UI
  - better UX
  - new system metrics
  - automated performance recommendations
  - free of charge
  - cross-platform

- Not released yet, but we’re working on it! Learn more and leave us feedback.



Note: This slide was added after the conference.
Sorry for the self-promotion. We look forward to the feedback from the community!

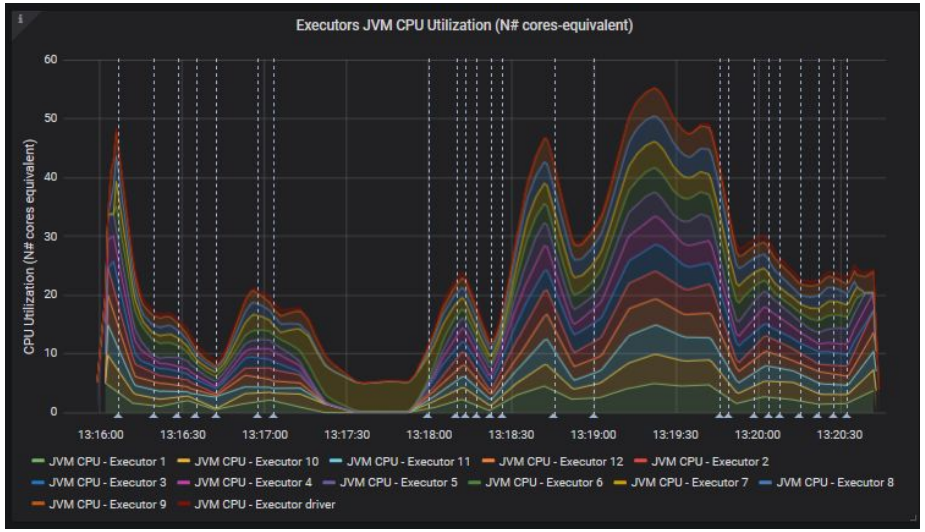
Export Spark metrics to a time-series database

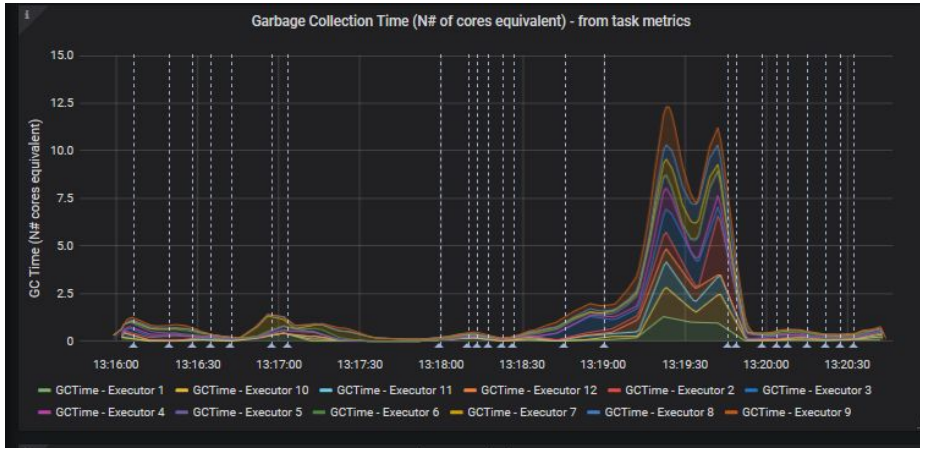
Spark leverages the DropWizard library to produce detailed metrics.

The metrics can be exported to a time-series database:

- InfluxDB (see spark-dashboard by Luca Canali)
- Prometheus
  - Spark has a built-in Prometheus servlet since version 3.0
  - The spark-operator proposes a Docker image with a Prometheus java agent for older versions

Use sparkmeasure to pipe task metrics and stage boundaries to the database





Luca Canali, spark-dashboard

Security

Kubernetes security best practices apply to Spark on Kubernetes for free!

Access control

Strong built-in RBAC system in Kubernetes
Spark apps and pods benefit from it as native k8s resources

Secrets management

Kubernetes secrets as a first step
Integrations with solutions like HashiCorp Vault

Networking

Mutual TLS, Network policies (since v1.18)
Service mesh like Istio

Spark on Kubernetes: Future Works

Features being worked on

- Shuffle improvements: Disaggregating storage and compute
  - Use remote storage for persisting shuffle data: SPARK-25299
  - Goal: Enable full dynamic allocation, and make Spark resilient to node loss (e.g. spot/pvm)

- Better Handling for node shutdown
  - Copy shuffle and cache data during graceful decomissioning of a node: SPARK-20624

- Support local python dependency upload (SPARK-27936)

- Job Queues and Resource Management

Spark on Kubernetes: Should You Get Started?

We chose Kubernetes for our platform - should you?

Pros

- Native Containerization
- A single cloud-agnostic infrastructure for your entire tech stack with a rich ecosystem
- Efficient resource sharing guaranteeing both resource isolation and cost efficiency

Cons

- Learning curve if you're new to Kubernetes
- A lot to setup yourself since most managed platforms do not support Kubernetes
- Marked as experimental (until 2.4) with missing features like the External Shuffle service.

For more details, read our blog post
The Pros and Cons of Running Apache Spark on Kubernetes

Checklist to get started with Spark-on-Kubernetes

- Setup the infrastructure
  - Create the Kubernetes cluster
  - Optional: Setup the spark operator
  - Create a Docker Registry
  - Host the Spark History Server
  - Setup monitoring for Spark application logs and metrics
- Configure your apps for success
  - Configure node pools and your pod sizes for optimal binpacking
  - Optimize I/O with proper libraries and volume mounts
  - Optional: Enable k8s autoscaling and Spark app dynamic allocation
  - Optional: Use spot/preemptible VMs for cost reduction
- Enjoy the Ride !

Our platform helps with this, and we're happy to help too!

Data
Mechanics

The Simplest Way To Run Spark
https://www.datamechanics.co

Thank you!

Appendix

Cost reduction with cluster autoscaling

Configure two node pools for your k8s cluster

- Node pool of small instances for system pods (e.g. ingress controller, autoscaler, spark-operator)
- Node pool of larger instances for Spark applications

Since node pools can scale down to zero on **all** cloud providers,

- you have large instances at your disposal for Spark apps
- you only pay for a small instance when the cluster is idle!