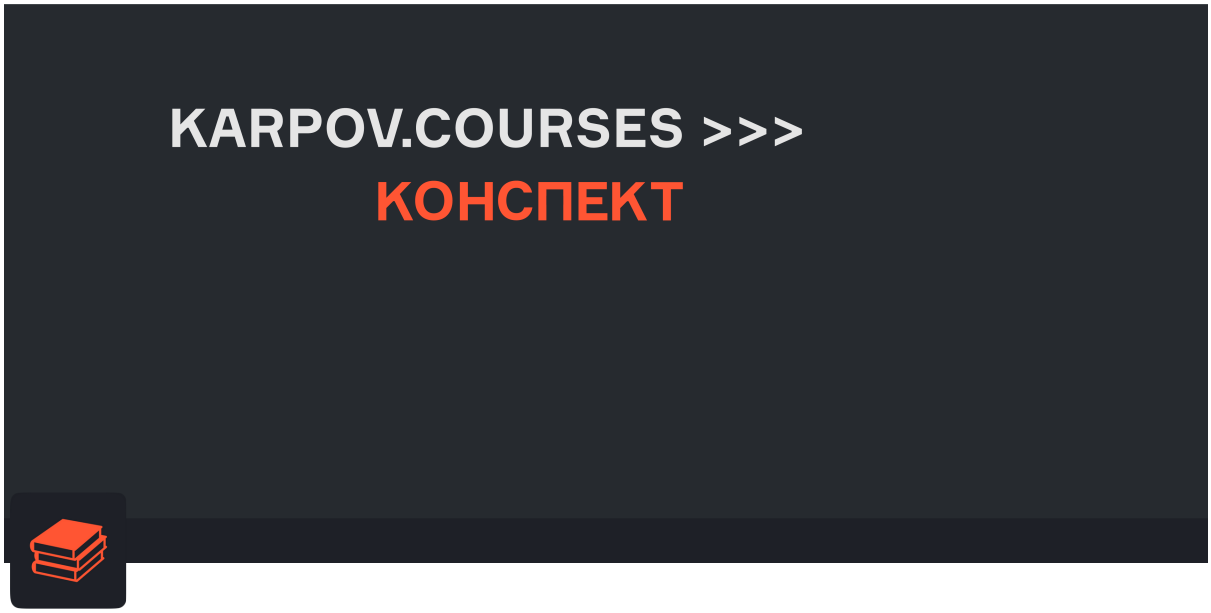


> Конспект > 7 урок > Практика: Разворачиваем Spark Kubernetes

> Оглавление

> Оглавление
> Почему стоит запускать Spark в Kubernetes
> Spark и Kubernetes
    > spark-submit
    > Spark Operator
    > Сравнение быстродействия
> Особенности Spark в Kubernetes

> Почему стоит запускать Spark в Kubernetes

- Изоляция сред (контейнеризация и dependency management). Обновив версию Spark, столкнемся с проблемой, что старые job's написанные на предыдущий версии перестали работать или работая, выдают другой результат. Spark в Kubernetes поможет избежать этого, так как каждый наш процессор упакован в отдельный контейнер и запуская новую версию,

запуск будет происходить в отдельном контейнере. Следовательно в одном кластере, может работать несколько версий Spark, так же все зависимости упаковываются в docker-контейнеры).

- Управление ресурсами. В классическом hadoop и кластере идет борьба за ресурсы, ресурсов не хватает на всех дата-инженеров, и они не могут все одновременно запустить свою job. Работая с Kubernetes, который поддерживает быстрое масштабирование, можно добавить десятки или сотни Node и соотвественно получить новые ресурсы для запуска приложений. Так же, если не запускается Spark приложение, он может масштабироваться в меньшую сторону и это позволяет эффективно использовать ресурсы.
- Гибкое масштабирование.
- Разделение `storage` и `compute` слоев. В Spark в Kubernetes наши данные лежат в s3 или в какой нибудь внешней системе (может быть hadoop, в котором, например, не хватило ядер). Запустив Spark в Kubernetes, получим нужное количество ядер и памяти, он возьмет данные из сторонней системе, обработает и по завершению работы данные запишутся в постоянных слоях.

> Spark и Kubernetes

Спарк можно заупускать в кубере с версии 2.3 (2018г), до версии 3.1 это фича считалась экперементальной.

Есть два способа запуска:

- spark-submit (традиционный способ запуска, похоже, на то как запускали бы свою job).
- Kubernetes Operator for Spark (использование специального оператора для Spark).

> spark-submit

Например, у нас есть некий клиент. Создается классическая строка spark-submit в которой указывается, что кластером будет выступать Kuber. Выполняя команду, отправляем ее в scheduler Kuber, внутри которого запустится Spark driver, который будет отвечать за запуск отдельных Executor'a.

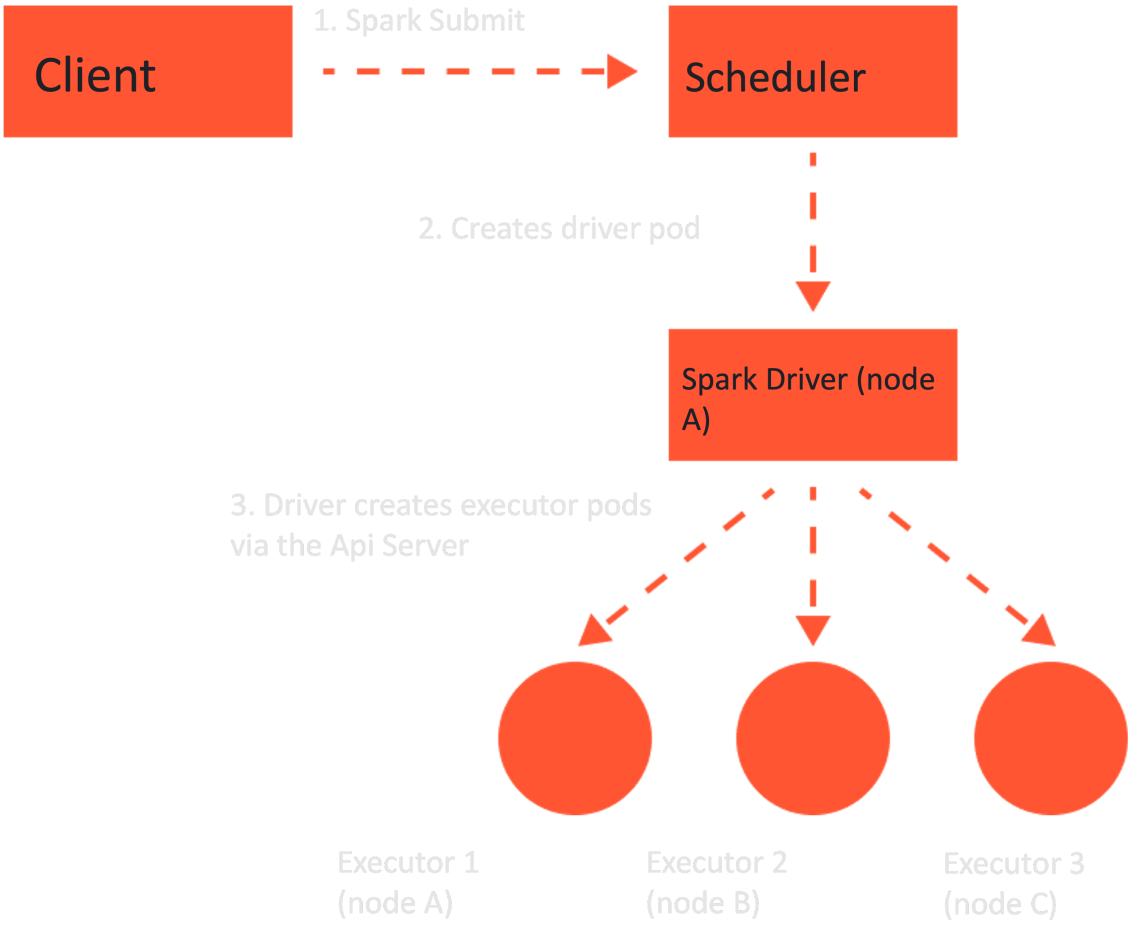


Схема работы spark-submit

1. Клиент создает Spark driver внутри k8s Pod.
2. Driver создает executors внутри k8s Pod, подключается к ним и использует код приложения.
3. Когда исполнение кода завершено, executor pods убиваются, driver же остается в статусе `completed`.

> Spark Operator

Использует кастомные ресурсы внутри Kuber, которые отвечают за работу со Spark. Используя Spark Operator, Kuber понимает, что работает Spark внутри него, появляется возможность действовать со Spark-job, как со специфическим Spark приложением.

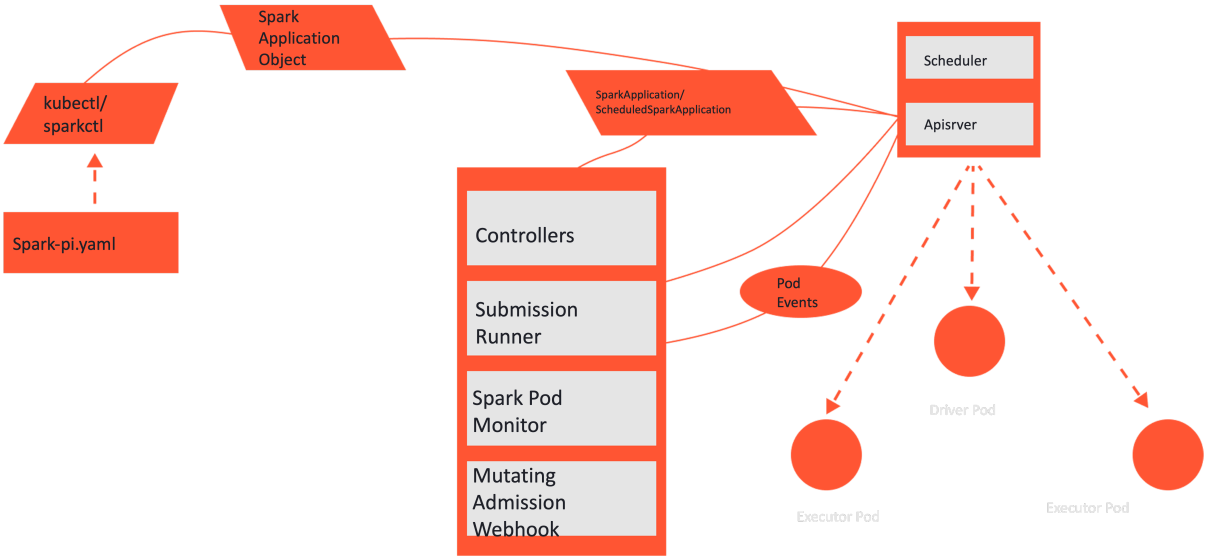
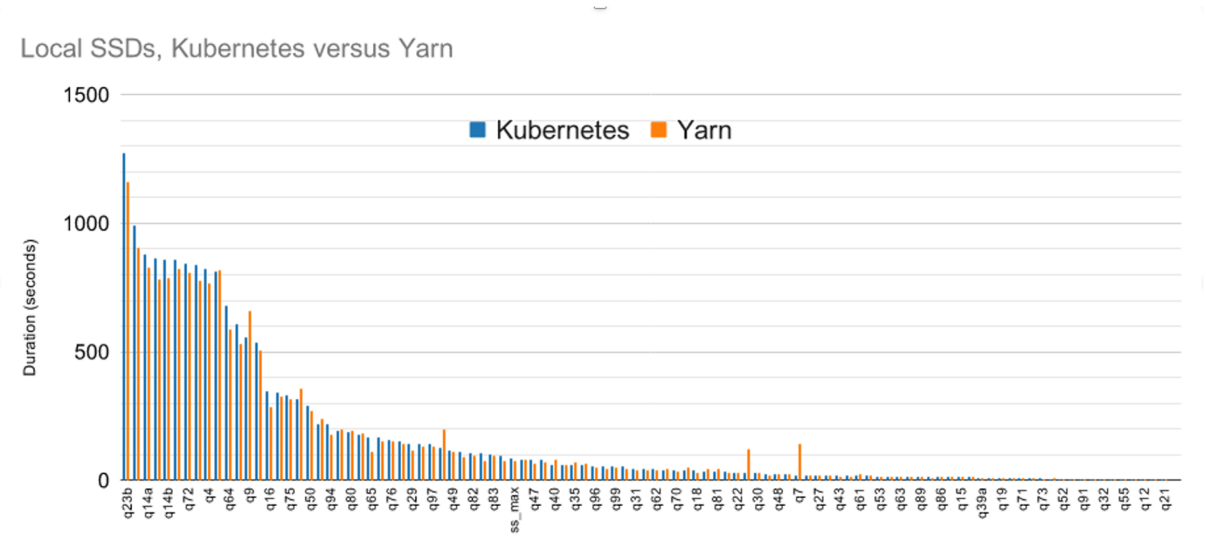


Схема работы Spark Operator

- Spark Application Controller
- Submission runner
- Spark Pod Monitor
- Mutating Admission Webhook

> Сравнение быстродействия

Spark в классическом варианте быстрее, но не намного, всего лишь 4-5%. Ссылка на статью.



Сравнение быстродействия

> Особенности Spark в Kubernetes

- **Быстродействие зависит от типа используемых дисков для временных файлов**. Например, в облаках быстродействие будет меньше, чем на диске который внутри машины. Так же влияет тип диска SSD и HDD. Используя разные типы дисков, всегда будут разные результаты производительности.
- **При сайзинге экзекьюторов необходимо учитывать потребление ресурсов на Node самим Kubernetes, сторонними сервисами.** Самому Kubernetes на каждой Node необходимы ресурс под агенты самого же Kuber'a, они сами потребляют оперативку и ресурсы процессора, поэтому нужно оставлять запас.
- **Dynamic Allocation работает в Kuber'e по-другому для Spark**. Работает для тех экзьютеров, на которые нету persistent данных в хранилище.