Spark dans le cloud
Kubernetes + object storage = 

https://github.com/scauglog/prez

Introduction


YOU GET A K8S CLUSTER
YOU GET A K8S CLUSTER
EVERYBODY GETS A K8S CLUSTER

Les Données

- Github Archives?
- TPC-DS
  - 1TB - **10 TB** - 100 TB
  - Generator - **AWS**
  - CSV |

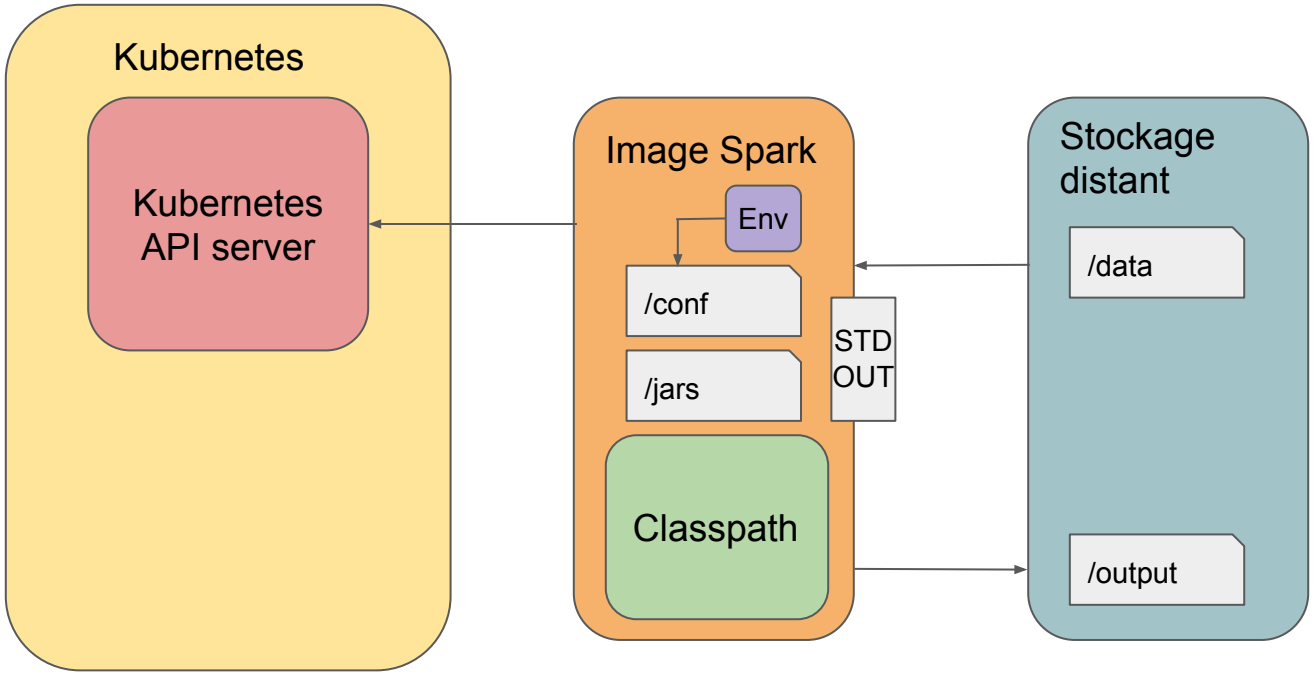
Les Queries

- CSV To Parquet
  - écriture intensif
- TPC-DS
  - SQL (Databricks benchmark)
  - LIMIT 100 (peu d’écriture)
  - Certaines sont complexe

Plateforme

- Scaleway
  - Pas chère
  - Français
  - k8s managé
  - Object Storage S3
  - GP1-XL
    - 48 CPU
    - 256Go RAM
    - 600 Go SSD
    - 10 Gbs Network

Architecture



Quelle image de conteneur pour Spark (3.2.1) ?

- Image Spark officiel
  - pas d’image officiel
  - raison juridique flou lié à la licence apache

Quelle image de conteneur pour Spark (3.2.1) ?

- Image Spark officiel
  - pas d'image officiel
  - raison juridique flou lié à la licence apache
- $SPARK_HOME/kubernetes/dockerfiles/spark/
  - nécessite d'avoir spark en local pour construire l'image

Quelle image de conteneur pour Spark (3.2.1) ?

- Image Spark officiel
  - pas d'image officiel
  - raison juridique flou lié à la licence apache
- $SPARK_HOME/kubernetes/dockerfiles/spark/
  - nécessite d'avoir spark en local pour construire l'image
- Image Ocean Spark (Datamechanics)
  - datamechanics/spark
  - 3.2.1
  - connecteur object storage embarqué (s3, gcs, wasbs, abfss, …)
  - image volumineuse

Quelle image de conteneur pour Spark (3.2.1) ?

- Image Spark officiel
  - pas d'image officiel
  - raison juridique flou lié à la licence apache
- $SPARK_HOME/kubernetes/dockerfiles/spark/
  - nécessite d'avoir spark en local pour construire l'image
- Image Ocean Spark (Datamechanics)
  - datamechanics/spark
  - 3.2.1
  - connecteur object storage embarqué (s3, gcs, wasbs, abfss, …)
  - image volumineuse
- Image Custom

Image Spark Custom

- Stage 1 : downloader
  - image curl
  - Téléchargement du tgz spark depuis le site spark et décompression
  - Téléchargement d'utilitaire autre (envsubst)

```
FROM  curlimages/curl  as downloader

RUN curl https://downloads.apache.org/spark/spark-  ${SPARK_VERSION} /spark-${SPARK_VERSION} -bin-hadoop3.2.tgz |
tar xvz -C /tmp

RUN mkdir -p /tmp/bin
RUN curl  -o /tmp/bin/envsubst -L https://github.com/a8m/envsubst/releases/download/v1.2.0/envsubst-`uname
-s`-`uname -m` && chmod +x /tmp/bin/envsubst
```

Image Spark Custom

- Stage 2 : builder
  - image maven
  - juste un pom.xml
  - création d'un fat jar pour télécharger les dépendances additionnels
  - plus simple que de récupérer les jar et leur dépendance une par une depuis maven central

```
FROM maven:3.8.4-jdk-11 as dep-builder

WORKDIR /app

COPY pom.xml ./
RUN mvn package
```

Image Spark Custom

- Stage 3
  - image java
  - création d'un user spark

```
FROM openjdk:${java_image_tag}

RUN addgroup --quiet --gid "${spark_uid}" "spark" && \
   adduser --disabled-password \
       --gecos \
       --quiet "spark" \
       --uid "${spark_uid}" \
       --gid "${spark_uid}" \
       --home "/opt/spark"
```

Image Spark Custom

- Stage 3
  - image java
  - création d'un user spark
  - installation de lib nécessaire pour spark (repris du dockerfile officiel)

```
RUN apt-get update && \
    ln -s /lib /lib64 && \
    apt install -y bash tini libc6 libpam-modules krb5-user libnss3 gettext-base && \
    rm /bin/sh && \
    ln -sv /bin/bash /bin/sh && \
    echo "auth required pam_wheel.so use_uid" >> /etc/pam.d/su && \
    chgrp root /etc/passwd && chmod ug+rw /etc/passwd && \
    rm -rf /var/cache/apt/*
```

Image Spark Custom

- Stage 3
  - image java
  - création d'un user spark
  - installation de lib nécessaire pour spark (repris du dockerfile officiel)
  - copie de spark depuis le stage downloader

```
COPY --from=downloader /tmp/bin/envsubst /usr/local/bin/
# Specify the User that the actual main process will run as
USER ${spark_uid}
COPY --chown=$spark_uid:$spark_uid --from=downloader ${spark_folder}/jars /opt/spark/jars
COPY --chown=$spark_uid:$spark_uid --from=downloader ${spark_folder}/bin /opt/spark/bin
COPY --chown=$spark_uid:$spark_uid --from=downloader ${spark_folder}/sbin /opt/spark/sbin
COPY --chown=$spark_uid:$spark_uid --from=downloader ${spark_folder}/examples /opt/spark/examples
COPY --chown=$spark_uid:$spark_uid --from=downloader ${spark_folder}/kubernetes/tests /opt/spark/tests
COPY --chown=$spark_uid:$spark_uid --from=downloader ${spark_folder}/kubernetes/dockerfiles/spark/entrypoint.sh
/opt/spark/
COPY --chown=$spark_uid:$spark_uid --from=downloader ${spark_folder}/data /opt/spark/data
```

Image Spark Custom

- Stage 3
  - image java
  - création d'un user spark
  - installation de lib nécessaire pour spark (repris du dockerfile officiel)
  - copie de spark depuis le stage downloader
  - copie des lib java depuis le stage builder
  - finalisation de l'image (workdir, entrypoint, SPARK_HOME)

```
COPY --chown=$spark_uid:$spark_uid --from=dep-builder /app/target/spark-dep-3.2.1-shaded.jar
/opt/spark/jars/spark-dep-3.2.1-shaded.jar
ENV SPARK_HOME /opt/spark
WORKDIR /opt/spark/work-dir
ENTRYPOINT [ "/opt/spark/entrypoint.sh" ]
```

Spark Image Custom

- multi stage build
- réduit la taille de l’image spark
  - embarque seulement le lib qui vous sont nécessaire
- réduit la taille du jar applicatif
  - spark+S3 connecteur sont “provided” par l’image de base
  - rebuild d’image applicative rapide
- possibilité de préconfigurer
  - spark-default.conf
  - lib additionnel

Créer son livrable d’application Spark

Créer son livrable d'application Spark

- stage de build maven
  - copie du pom
  - dependency:go-offline permet de ne pas re télécharger les dépendances à chaque build
  - copie des sources
  - build

```
FROM maven:3.8.4-jdk-11 as builder
WORKDIR /app
COPY pom.xml ./
RUN mvn dependency:go-offline
COPY src src
RUN mvn package
```

Créer son livrable d'application Spark

- stage de build maven
  - copie du pom
  - dependency:go-offline permet de ne pas re télécharger les dépendances à chaque build
  - copie des source
  - build
- stage final
  - réutilisation de l'image de base
  - copy du jar applicatif dans l'image

```
FROM rg.fr-par.scw.cloud/benchmark-spark/spark:3.2.1
COPY --chown=185:185 --from=builder /app/target/benchmark-1.0-SNAPSHOT-shaded.jar
/opt/spark/examples/jars/spark-benchmark.jar
```

spark-submit avec Kubernetes

Comment ça marche ?

La commande

```
spark-submit --master k8s://kubernetes-api-endpoint \
--deploy-mode cluster \
--name data \
--class data.Main \
--conf spark.executor.instances=3 \
--conf spark.kubernetes.container.image=franckcussac/data:1 \
--conf spark.kubernetes.authenticate.driver.serviceAccountName=spark-kube \
--conf spark.kubernetes.namespace=spark \
--conf spark.hadoop.fs.s3a.access.key=xxx \
--conf spark.hadoop.fs.s3a.secret.key=xxx \
local://opt/spark/examples/jar/spark-benchmark.jar \
s3a://dev-kube-datalake/data/input \
s3a://dev-kube-datalake/data/feedback \
s3a://dev-kube-datalake/data/output
```

La commande - job Spark

```
spark-submit --master k8s://kubernetes-api-endpoint \
--deploy-mode cluster \
--name data \
--class data.Main \
--conf spark.executor.instances=3 \
--conf spark.kubernetes.container.image=franckcussac/data:1 \
--conf spark.kubernetes.authenticate.driver.serviceAccountName=spark-kube \
--conf spark.kubernetes.namespace=spark \
--conf spark.hadoop.fs.s3a.access.key=xxx \
--conf spark.hadoop.fs.s3a.secret.key=xxx \
local://opt/spark/examples/jar/spark-benchmark.jar \
s3a://dev-kube-datalake/data/input \
s3a://dev-kube-datalake/data/feedback \
s3a://dev-kube-datalake/data/output
```

La commande - configuration Kubernetes

```
spark-submit --master k8s://kubernetes-api-endpoint \
--deploy-mode cluster \
--name data \
--class data.Main \
--conf spark.executor.instances=3 \
--conf spark.kubernetes.container.image=franckcussac/data:1 \
--conf spark.kubernetes.authenticate.driver.serviceAccountName=spark-kube \
--conf spark.kubernetes.namespace=spark \
--conf spark.hadoop.fs.s3a.access.key=xxx \
--conf spark.hadoop.fs.s3a.secret.key=xxx \
local://opt/spark/examples/jar/spark-benchmark.jar \
s3a://dev-kube-datalake/data/input \
s3a://dev-kube-datalake/data/feedback \
s3a://dev-kube-datalake/data/output
```

La commande - configuration de l'application

```
spark-submit --master k8s://kubernetes-api-endpoint \
--deploy-mode cluster \
--name data \
--class data.Main \
--conf spark.executor.instances=3 \
--conf spark.kubernetes.container.image=franckcussac/data:1 \
--conf spark.kubernetes.authenticate.driver.serviceAccountName=spark-kube \
--conf spark.kubernetes.namespace=spark \
--conf spark.hadoop.fs.s3a.access.key=xxx \
--conf spark.hadoop.fs.s3a.secret.key=xxx \
local://opt/spark/examples/jar/spark-benchmark.jar \
s3a://dev-kube-datalake/data/input \
s3a://dev-kube-datalake/data/feedback \
s3a://dev-kube-datalake/data/output
```

La commande

```
spark-submit --master k8s://kubernetes-api-endpoint \
--deploy-mode cluster \
--name data-ked \
--class data.Main \
--conf spark.executor.instances=3 \
--conf spark.kubernetes.container.image=franckcussac/data:1 \
--conf spark.kubernetes.authenticate.driver.serviceAccountName=spark-kube \
--conf spark.kubernetes.namespace=spark \
--conf spark.hadoop.fs.s3a.access.key=xxx \
--conf spark.hadoop.fs.s3a.secret.key=xxx \
local://opt/spark/examples/jar/spark-benchmark.jar \
s3a://dev-kube-datalake/data/input \
s3a://dev-kube-datalake/data/feedback \
s3a://dev-kube-datalake/data/output
```

spark-on-k8s-operator

```yaml
apiVersion: sparkoperator.k8s.io/v1beta2
kind: SparkApplication
metadata:
 name: data
 namespace: spark
spec:
 mode: cluster
 mainClass: data.Main
 mainApplicationFile: local://opt/spark/examples/jar/spark-benchmark.jar
 image: franckcussac/data:1
 driver:
   serviceAccount: spark-kube
   env:
     - name: INPUT_FILE
       value: "s3a://dev-kube-datalake/data/input"
   envSecretKeyRefs:
     AWS_ACCESS_KEY_ID: {...}
     AWS_SECRET_ACCESS_KEY: {...}
 executor:
   instances: 3
```

spark-on-k8s-operator - job Spark

```yaml
apiVersion: sparkoperator.k8s.io/v1beta2
kind: SparkApplication
metadata:
 name: data
 namespace: spark
spec:
 mode: cluster
 mainClass: data.Main
 mainApplicationFile: local://opt/spark/examples/jar/spark-benchmark.jar
 image: franckcussac/data:1
 driver:
   serviceAccount: spark-kube
   env:
     - name: INPUT_FILE
       value: "s3a://dev-kube-datalake/data/input"
   envSecretKeyRefs:
     AWS_ACCESS_KEY_ID: {...}
     AWS_SECRET_ACCESS_KEY: {...}
 executor:
   instances: 3
```

spark-on-k8s-operator - configuration Kubernetes

```yaml
apiVersion: sparkoperator.k8s.io/v1beta2
kind: SparkApplication
metadata:
 name: data
 namespace: spark
spec:
 mode: cluster
 mainClass: data.Main
 mainApplicationFile: local://opt/spark/examples/jar/spark-benchmark.jar
 image: franckcussac/data:1
 driver:
   serviceAccount: spark-kube
   env:
     - name: INPUT_FILE
       value: "s3a://dev-kube-datalake/data/input"
   envSecretKeyRefs:
     AWS_ACCESS_KEY_ID: {...}
     AWS_SECRET_ACCESS_KEY: {...}
 executor:
   instances: 3
```

spark-on-k8s-operator - configuration de l'application

```yaml
apiVersion: sparkoperator.k8s.io/v1beta2
kind: SparkApplication
metadata:
 name: data
 namespace: spark
spec:
 mode: cluster
 mainClass: data.Main
 mainApplicationFile: local://opt/spark/examples/jar/spark-benchmark.jar
 image: franckcussac/data:1
 driver:
   serviceAccount: spark-kube
   env:
     - name: INPUT_FILE
       value: "s3a://dev-kube-datalake/data/input"
   envSecretKeyRefs:
     AWS_ACCESS_KEY_ID: {...}
     AWS_SECRET_ACCESS_KEY: {...}
 executor:
   instances: 3
```

spark-on-k8s-operator

```yaml
apiVersion: sparkoperator.k8s.io/v1beta2
kind: SparkApplication
metadata:
 name: data
 namespace: spark
spec:
 mode: cluster
 mainClass: data.Main
 mainApplicationFile: local://opt/spark/examples/jar/spark-benchmark.jar
 image: franckcussac/data:1
 driver:
   serviceAccount: spark-kube
   env:
     - name: INPUT_FILE
       value: "s3a://dev-kube-datalake/data/input"
   envSecretKeyRefs:
     AWS_ACCESS_KEY_ID: {...}
     AWS_SECRET_ACCESS_KEY: {...}
 executor:
   instances: 3
```

Les premiers résultats

Allons voir l’history server

Spark History Server sur Kubernetes

- disponible dans le tgz spark
- lis les logs depuis l'object storage
- utiliser le spark-default.conf pour configurer l'accès à l'object storage
  - configmap spark-default.conf avec variable d'env pour les secret
    - $SCW_SECRET_KEY
    - $SCW_ACCESS_KEY

```
kind: ConfigMap
apiVersion: v1
metadata:
 name: spark-hs-config
data:
 spark-defaults.conf : |-
   spark.hadoop.fs.s3a.access.key=$SCW _ACCESS_KEY
   spark.hadoop.fs.s3a.secret.key=$SCW_SECRET_KEY
   spark.hadoop.fs.s3a.endpoint=https://s3.fr-par.scw.cloud
   spark.history.fs.logDirectory={{ .Values.sparkEventLogStorage.logDirectory }}
```

Spark History Server sur Kubernetes

- disponible dans le tgz spark
- lis les logs depuis l'object storage
- utiliser le spark-default.conf pour configurer l'accès à l'object storage
  - …
  - initcontainer avec envsusbt
    - envfrom Secret
    - volumeMount configmap
    - volumeMount emptydir
    - envsusbt configMap -> emptyDir

```
initContainers:
- name: init-config
 image: "rg.fr-par.scw.cloud/benchmark-spark/spark:3.2.1"
 envFrom:
   - secretRef: SCW_ACCESS_KEY
     name: "scw-secrets"
 volumeMounts:
   - name: config-volume
     mountPath: /opt/spark/conf/
   - name: config-init-volume
     mountPath:
/opt/spark/prepared_conf/spark-defaults.conf
     subPath: spark-defaults.conf
 command:
   - '/bin/sh'
   - '-c'
   - >
     envsubst <
/opt/spark/prepared_conf/spark-defaults.conf >
/opt/spark/conf/spark-defaults.conf;
volumes:
 - name: config-volume
   emptyDir: {}
 - name: config-init-volume
   configMap:
     name: spark-hs-config
```

Spark History Server sur Kubernetes

- disponible dans le tgz spark
- lis les logs depuis l'object storage
- utiliser le spark-default.conf pour configurer l'accès à l'object storage
  - configmap spark-default.conf avec variable d'env pour les secret
    - $SCW_SECRET_KEY
    - $SCW_ACCESS_KEY
  - secret pour les crédentials
    - SCW_SECRET_KEY
    - SCW_ACCESS_KEY
  - initcontainer avec envsusbt
    - volumeMount configmap
    - envfrom Secret
    - volumeMount emptydir
    - envsusbt configMap -> emptyDir
- Sur les jobs spark configurer `spark.eventLog.dir` vers un stockage partagé (s3, gcs, wasb, …)

D'où vient le problème

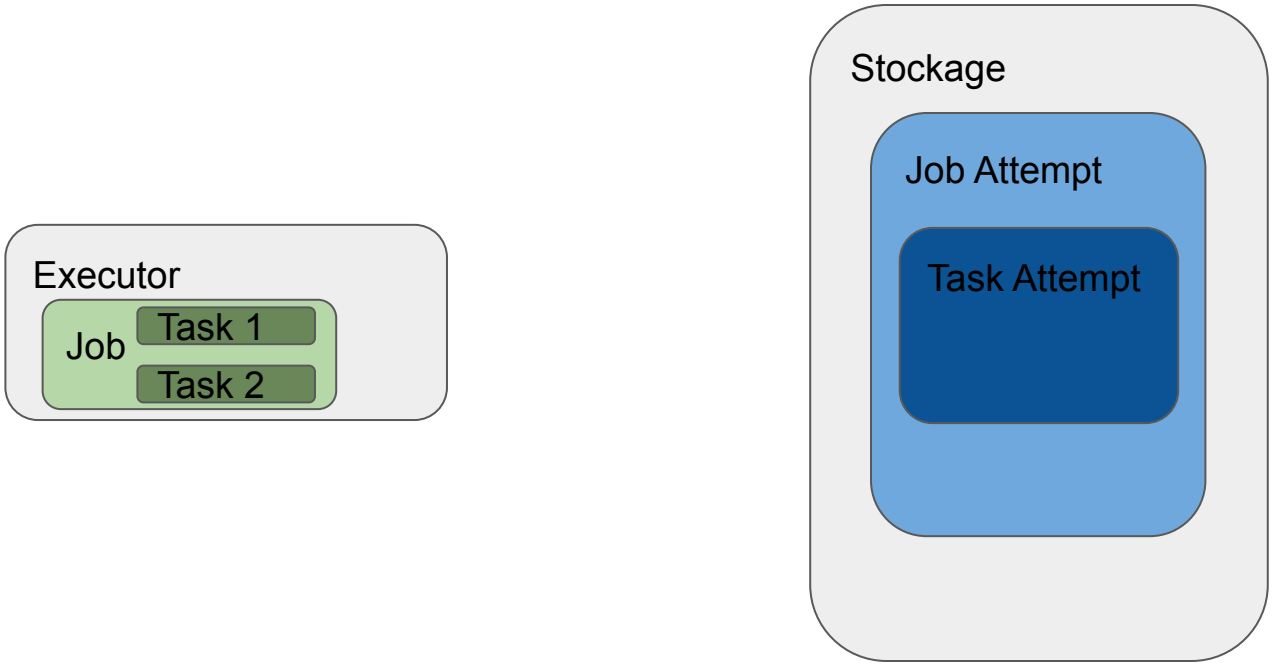
- Arrêt sur le stage d’écriture
- CPU bas
- Mémoire qui explose
- Écriture sur Object Storage long

Comment est implémenté l’écriture de fichier

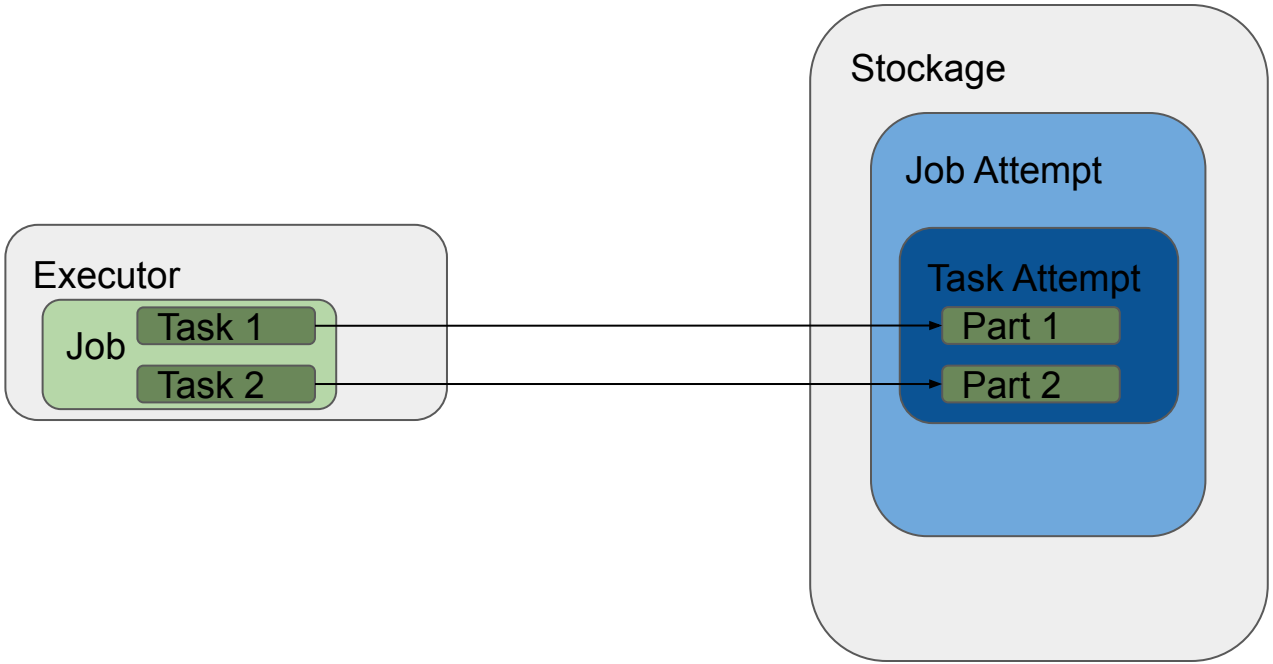
Écriture des données : Comportement par défaut

- Algorithme commit output file v1
- Écriture dans des fichiers temporaires
  - Safe sur un échec de Task
  - Safe sur un échec de Job

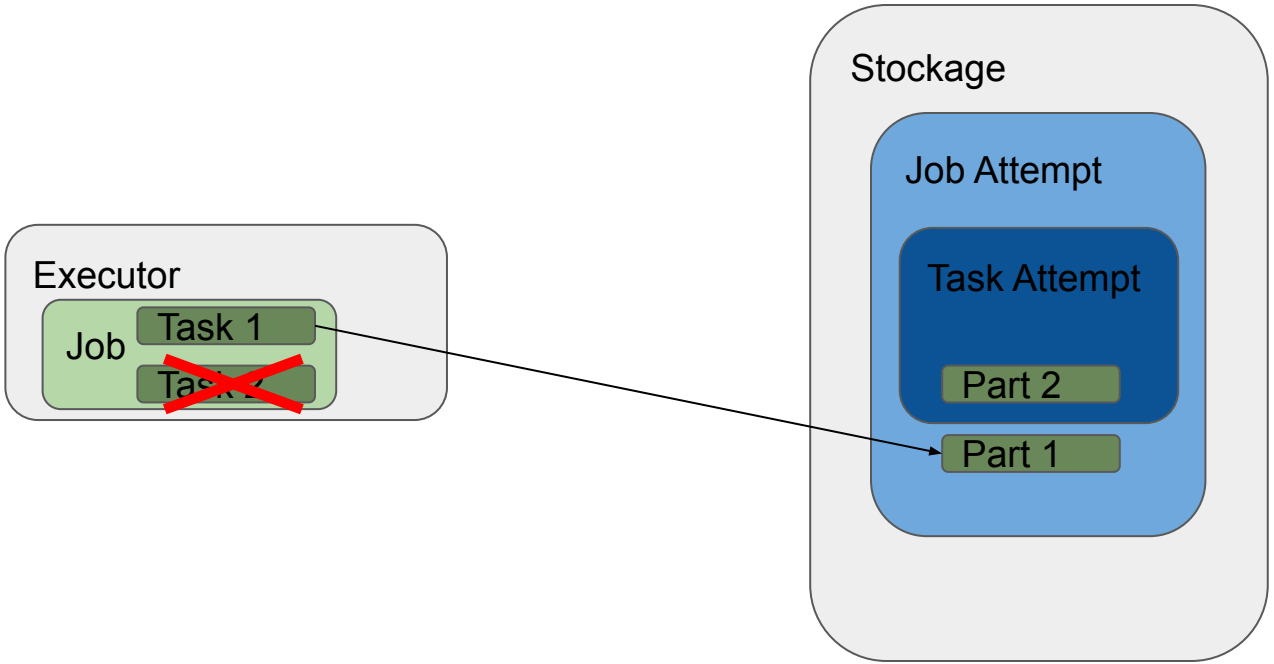
Task Commit


Executor
Job
Task 1
Task 2
Stockage
Job Attempt
Task Attempt

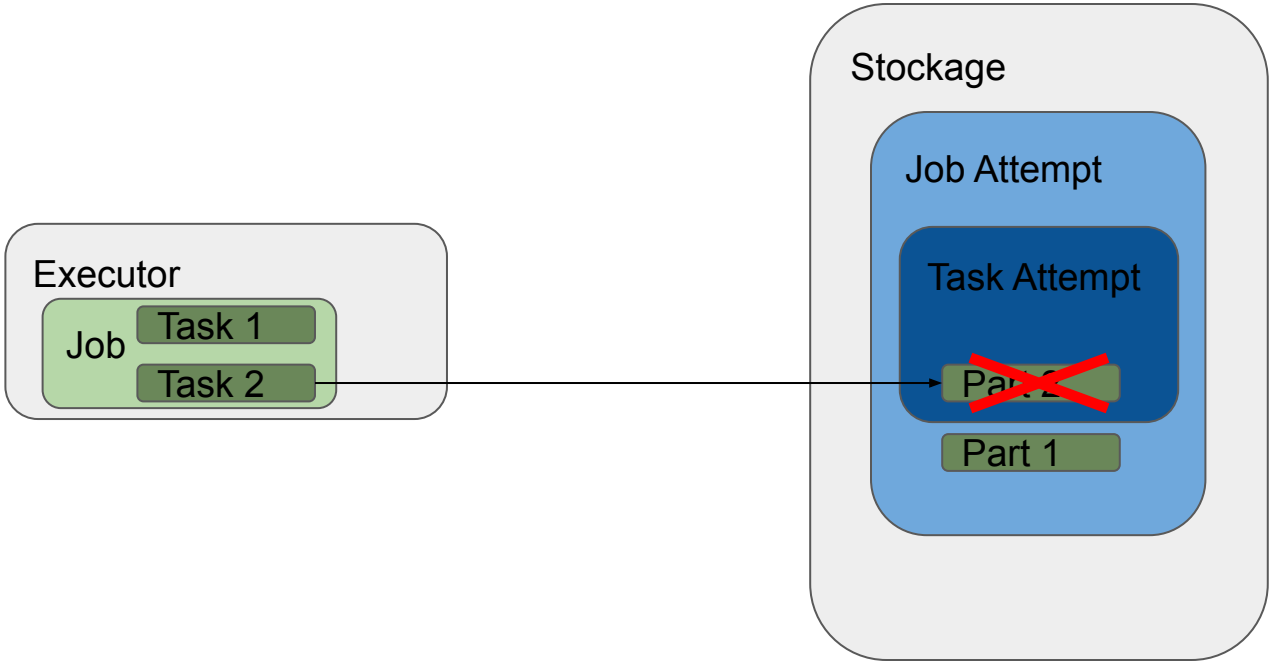
Task Commit


Stockage
Job Attempt
Task Attempt
Part 1
Part 2
Executor
Job
Task 1
Task 2

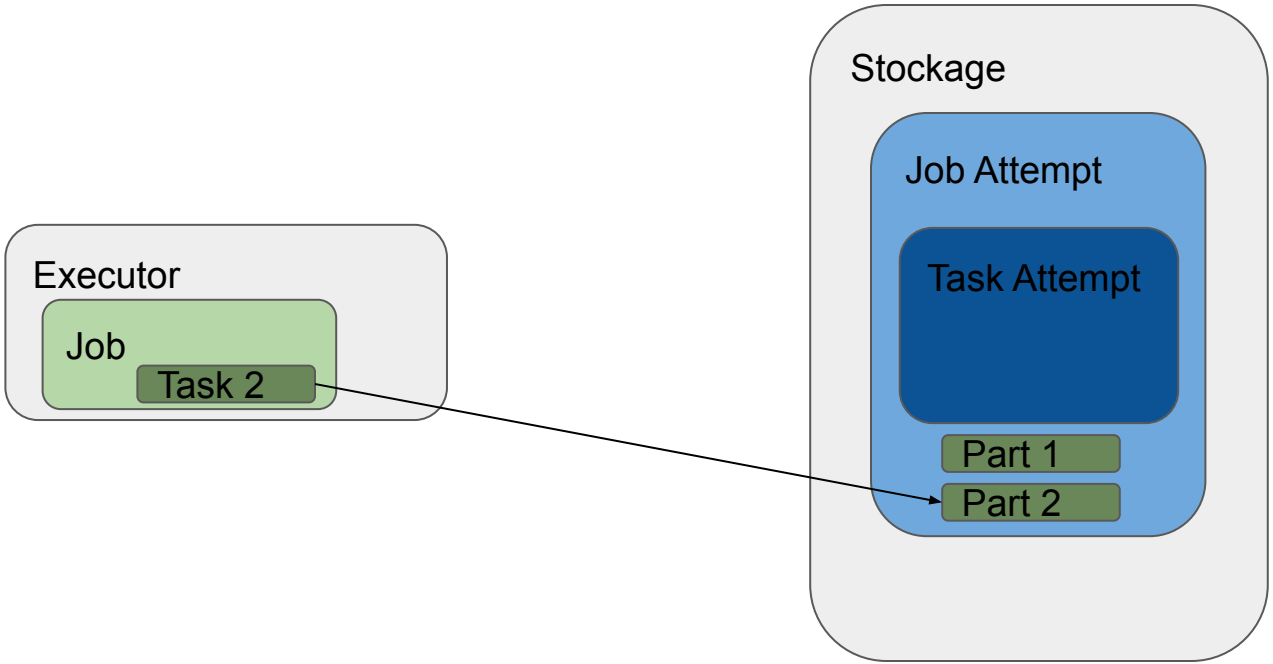
Task Commit


Stockage
Job Attempt
Task Attempt
Part 2
Part 1
Executor
Job
Task 1

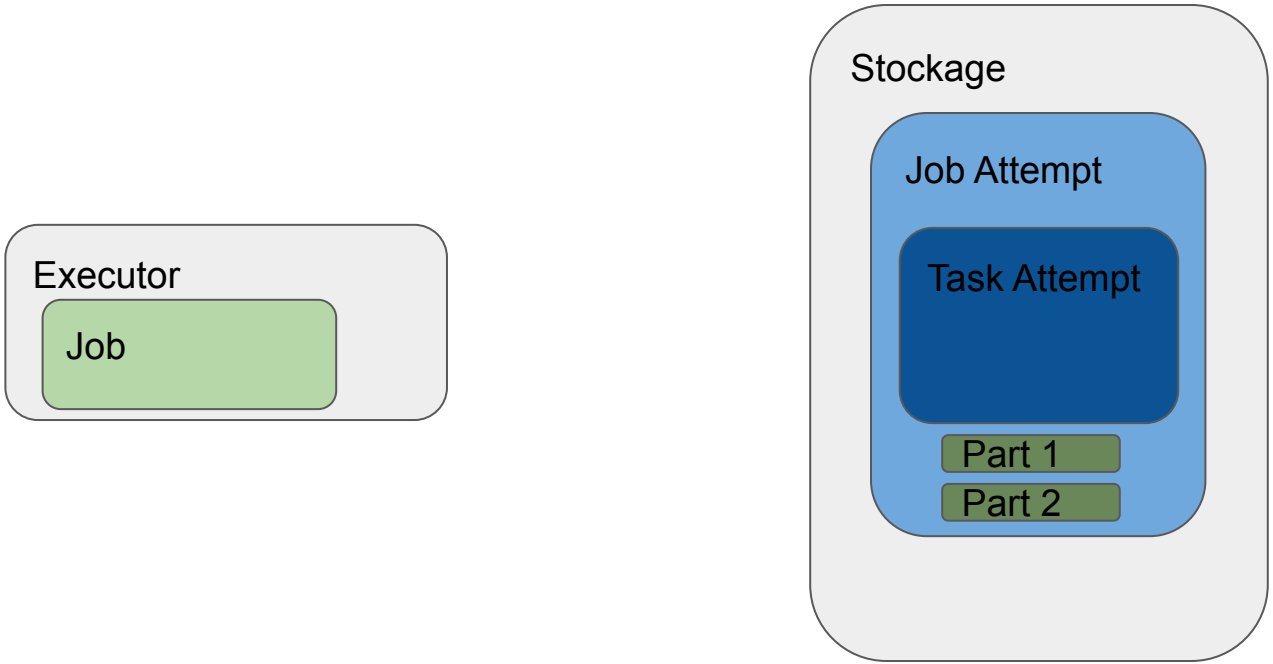
Task Commit


Stockage
Job Attempt
Task Attempt
Part 1
Executor
Job
Task 1
Task 2

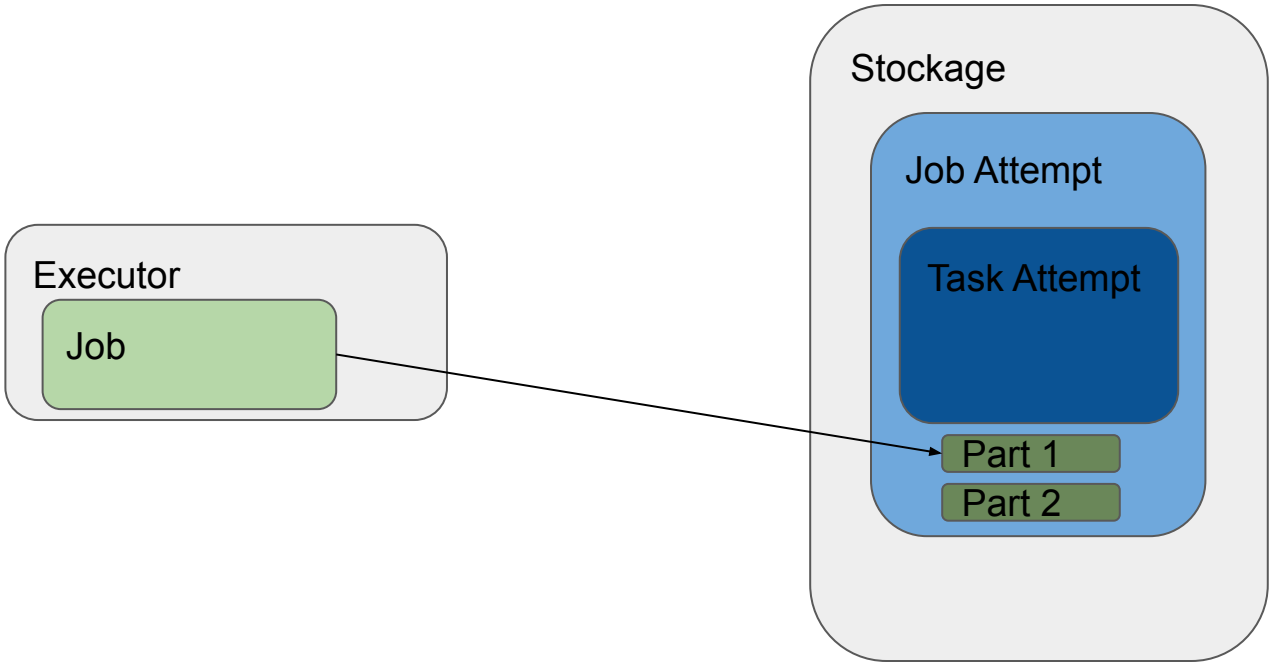
Task Commit


Stockage
Job Attempt
Task Attempt
Part 1
Part 2
Executor
Job
Task 2

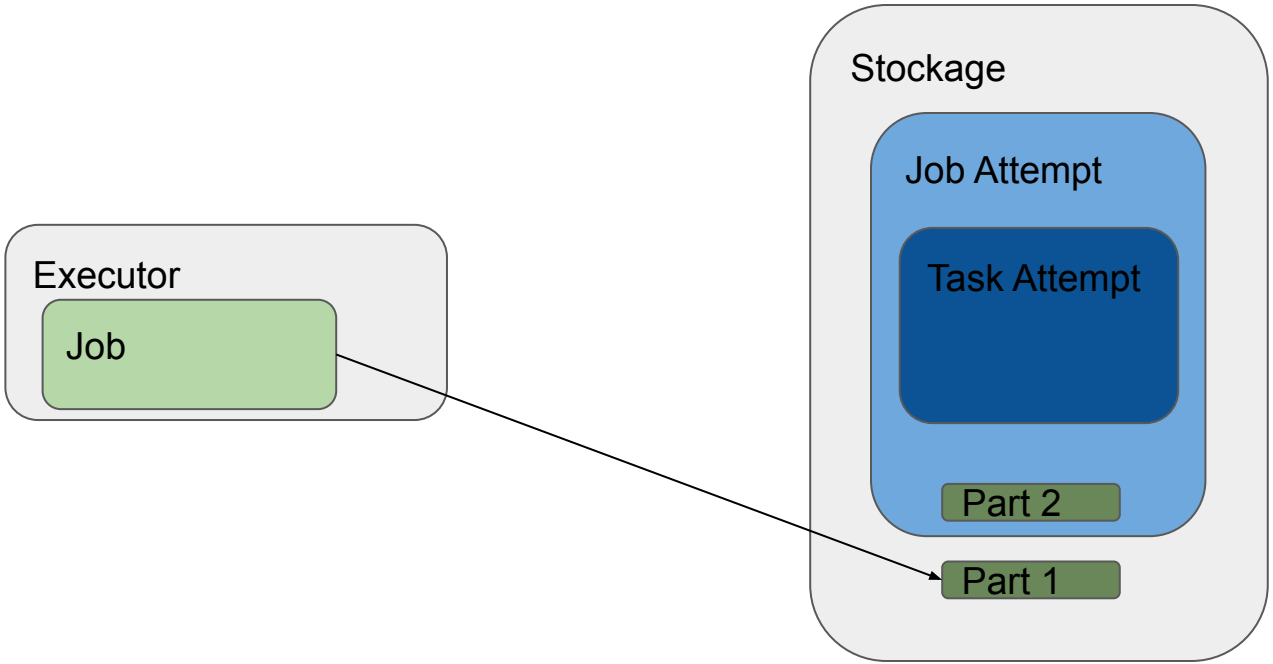
Job Commit


Executor
Job
Stockage
Job Attempt
Task Attempt
Part 1
Part 2

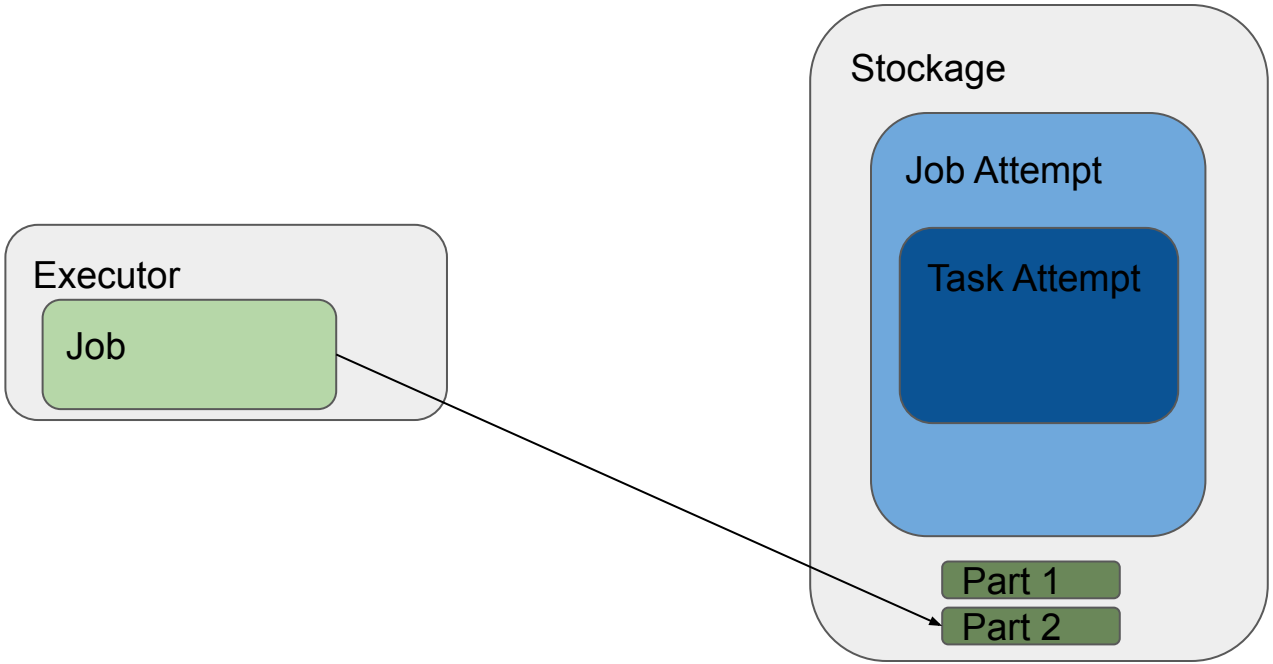
Job Commit


Stockage
Job Attempt
Task Attempt
Part 1
Part 2
Executor
Job

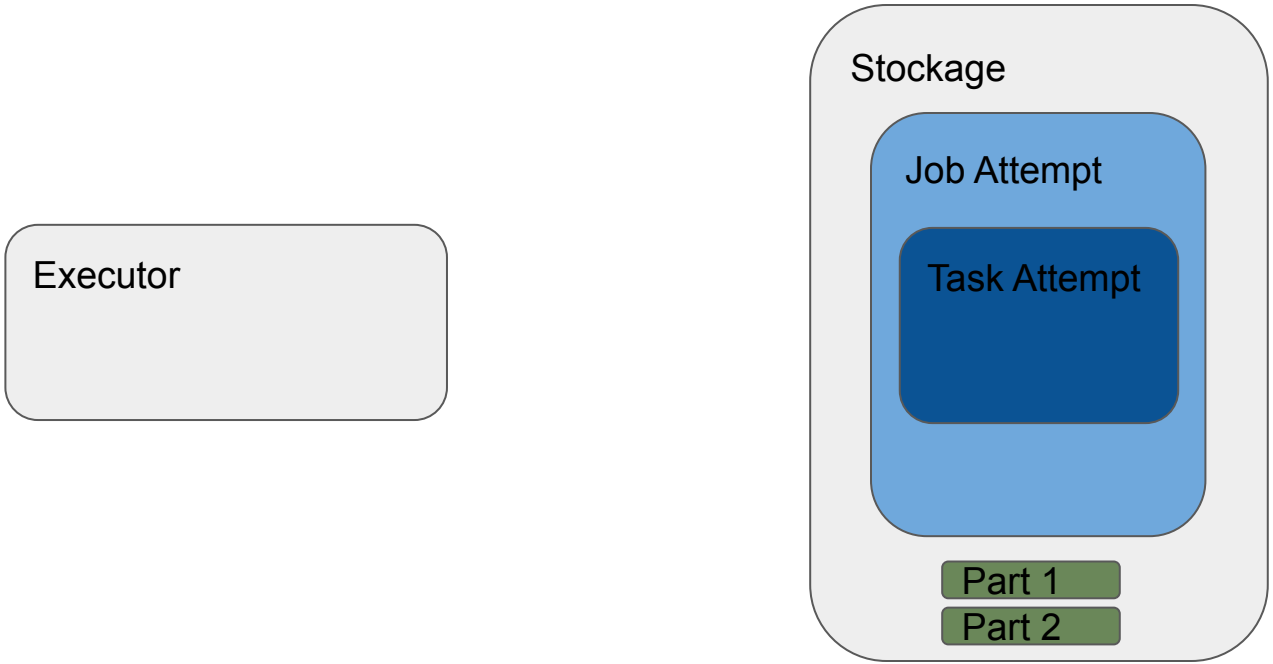
Job Commit


Stockage
Job Attempt
Task Attempt
Part 2
Part 1
Executor
Job

Job Commit


Stockage
Job Attempt
Task Attempt
Executor
Job
Part 1
Part 2

Job Commit



Algo v1 Commit task

- Supprime le dossier taskCommitted si il a précédemment été créé
- Renomme le dossier taskAttempt en taskCommitted (suppression du _temporary)

```
taskAttemptPath = '$dest/_temporary/$appAttemptId/_temporary/$taskAttemptID'
taskCommittedPath = '$dest/_temporary/$appAttemptId/$taskAttemptID'
jobAttemptPath = '$dest/_temporary/$appAttemptId/'
```

```
def commitTask(fs, jobAttemptPath, taskAttemptPath, dest):
  if fs.exists(taskAttemptPath) :
    fs.delete(taskCommittedPath, recursive=True)
  fs.rename(taskAttemptPath, taskCommittedPath)
```

Algo v1 Commit job

- déplace toutes les commitedTask du dossier jobAttempDir dans le dossier final
- crée le fichier _SUCCESS

```
def commitJob(fs, jobAttemptDir, dest):
  for committedTask in fs.listFiles(jobAttemptDir):
    mergePathsV1(fs, committedTask, dest)
  fs.touch("$dest/_SUCCESS")

def mergePathsV1(fs, src, dest) :
  if fs.exists(dest) :
    toStat = fs.getFileStatus(dest)
  else:
    toStat = None

  if src.isFile :
    if not toStat is None :
      fs.delete(dest, recursive = True)
    fs.rename(src.getPath, dest)
  else :
    # src is directory, choose action on dest type
    if not toStat is None :
      if not toStat.isDirectory :
        # Destination exists and is not a directory
        fs.delete(dest)
        fs.rename(src.getPath(), dest)
      else :
        # Destination exists and is a directory
        # merge all children under destination directory
        for child in fs.listStatus(src.getPath) :
          mergePathsV1(fs, child, dest + child.getName)
    else :
      # destination does not exist
         fs.rename(src.getPath(), dest)
```

File Output Committer v1

- deux déplacement du fichier
  - lors du taskCommit
    - `'$dest/_temporary/$appAttemptId/_temporary/$taskAttemptID/000.parquet' -> '$dest/_temporary/$appAttemptId/$taskAttemptID/000.parquet'`
  - lors du jobCommit
    - `'$dest/_temporary/$appAttemptId/$taskAttemptID/000.parquet' -> '$dest/000.parquet'`

Caractéristique des systèmes de stockage

Store	**connector**	**Rename Performance**
Amazon S3	s3a	O(data) (COPY+DELETE)
Scaleway Object Storage	s3a	O(data) (COPY+DELETE)
Azure Storage	wasb	O(files in directory)
Azure Datalake Gen 2	abfs	O(1)
Google GCS	gs	O(1)
HDFS	hdfs	O(1)

Solution 1 : Changer l’algorithme de commit

- mapreduce.fileoutputcommitter.algorithm.version = 2
- Écriture dans des fichiers temporaires
  - Safe sur un échec de Task

Temps d'exécution : 4H
- 30 executors
  - 28Go
  - 5 CPU
- 28 Milliards lignes

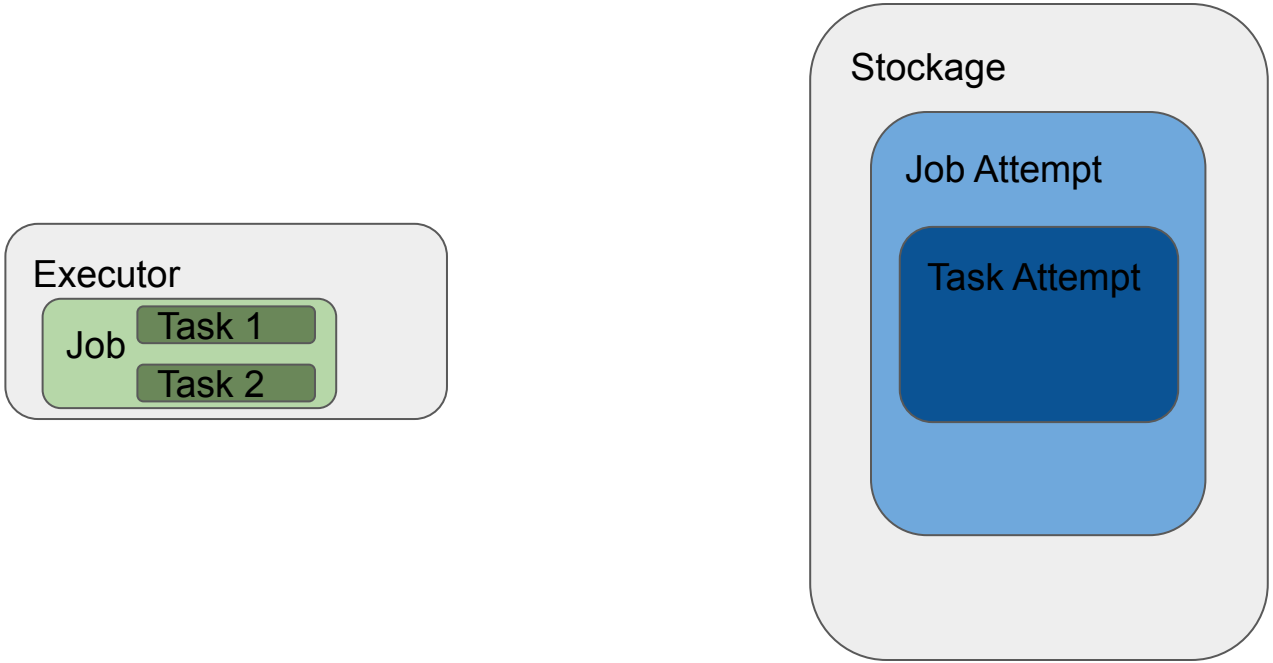
Test Parquet : algorithme v1 vs algorithme v2

- 64 partitions
- 3 executors
  - 3 coeurs
  - 16Go de RAM
- Algorithme v1 : 16 minutes
- Algorithme v2 : 8 minutes

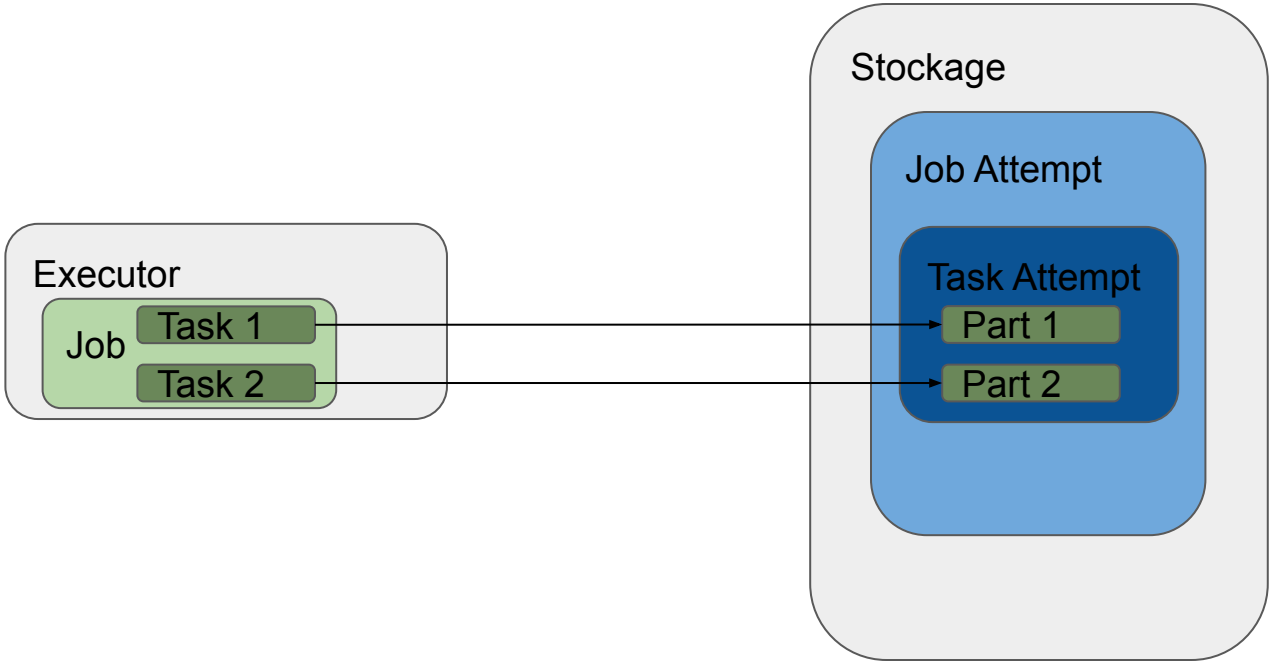
Test Parquet : algorithme v1 vs algorithme v2

- 64 partitions
- 2 executors
  - 2 coeurs
  - 8Go de RAM
- Algorithme v1 : 9 minutes 40 secondes
- Algorithme v2 : 6 minutes 40 secondes

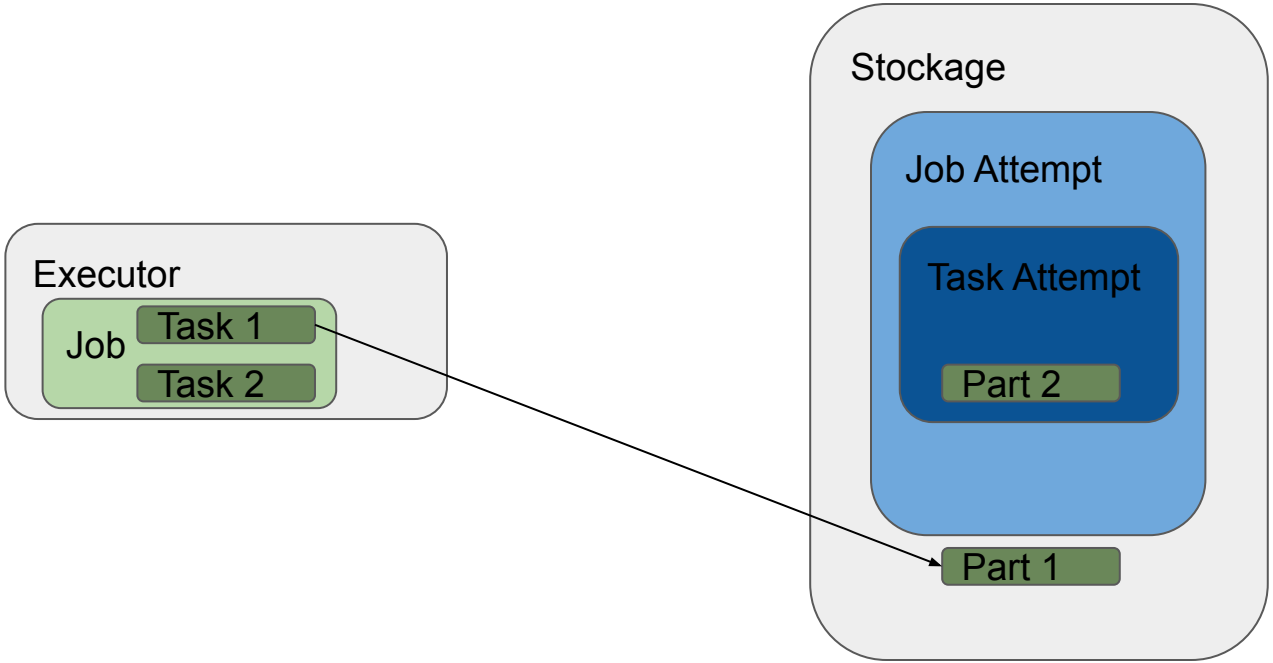
Task Commit


Executor
Job
Task 1
Task 2
Stockage
Job Attempt
Task Attempt

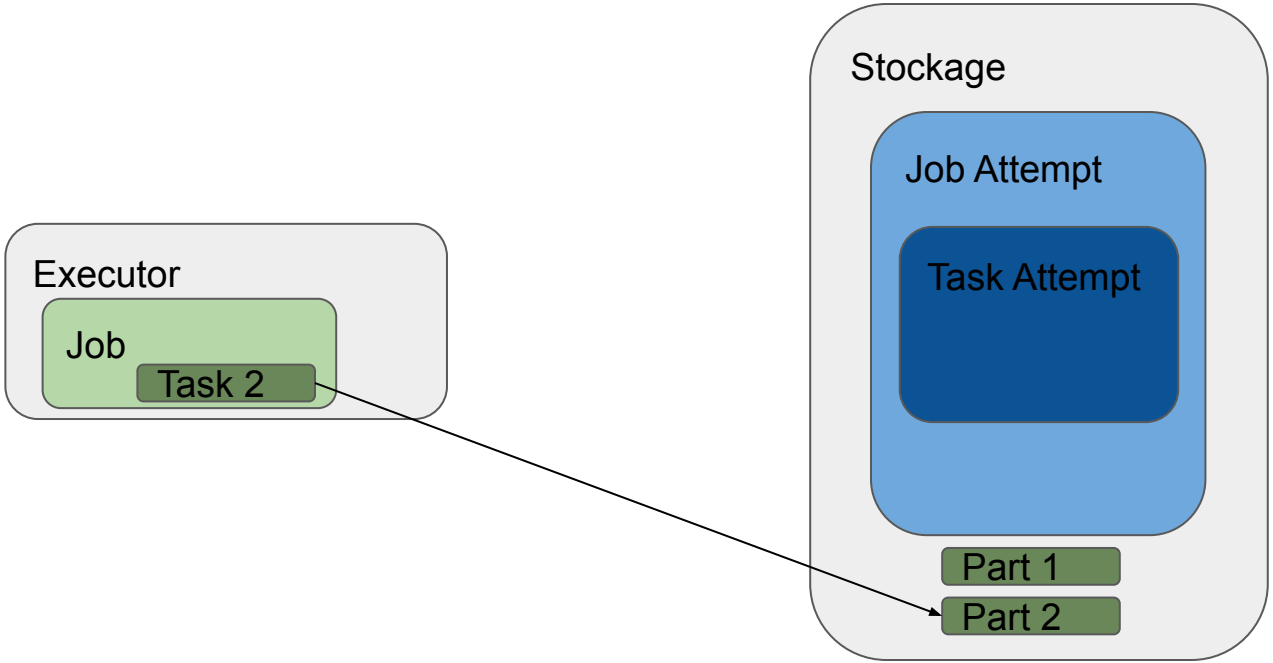
Task Commit


Stockage
Job Attempt
Task Attempt
Part 1
Part 2
Executor
Job
Task 1
Task 2

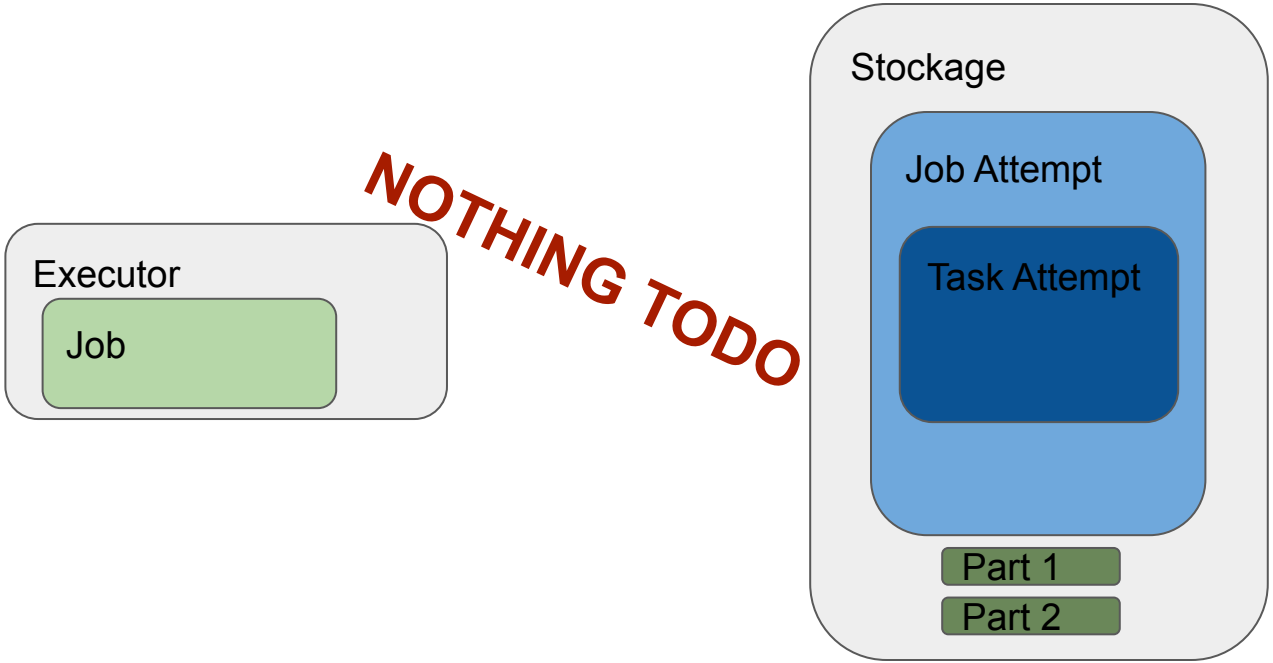
Task Commit


Stockage
Job Attempt
Task Attempt
Part 2
Part 1
Executor
Job
Task 1
Task 2

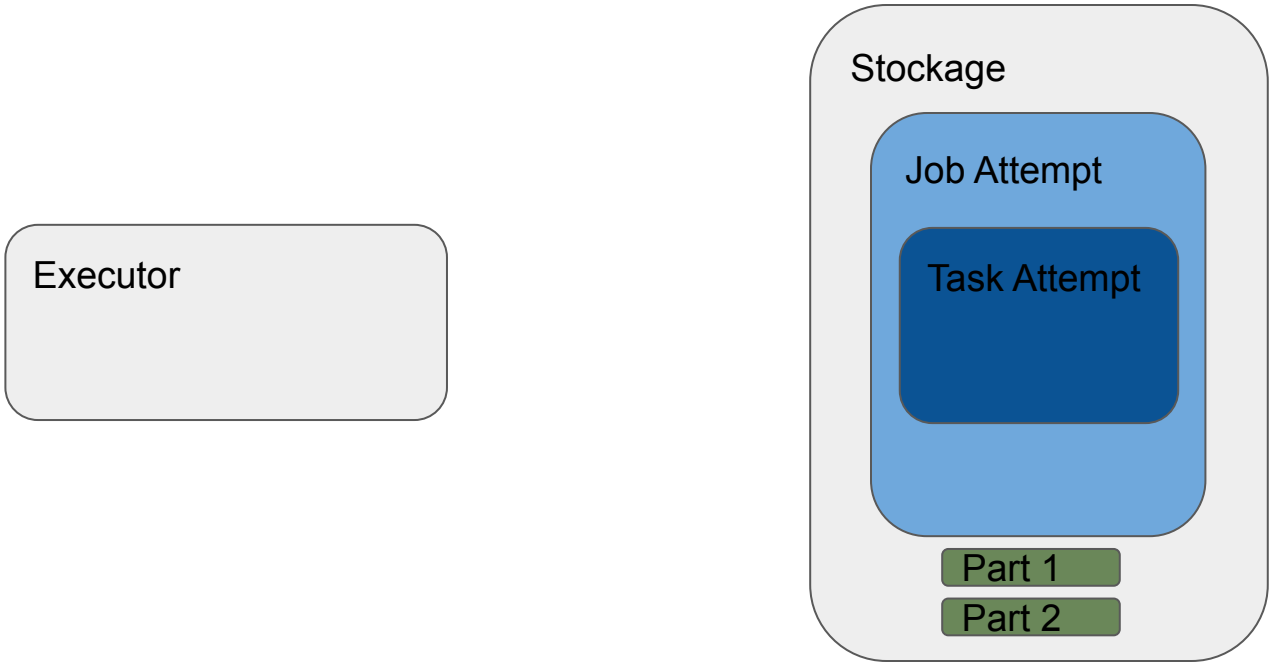
Task Commit


Stockage
Job Attempt
Task Attempt
Executor
Job
Task 2
Part 1
Part 2

Job Commit


Executor
Job
NOTHING TODO
Stockage
Job Attempt
Task Attempt
Part 1
Part 2

Job Commit



Algo v2 commit task

- Si le fichier taskAttempt existe dans la destination on le supprime
- Déplacement du fichier taskAttempt dans le dossier final

```
def commitTask(fs, jobAttemptPath, taskAttemptPath, dest):
  if fs.exists(taskAttemptPath) :
       mergePathsV2(fs, taskAttemptPath, dest)

def mergePathsV2(fs, src, dest) :
  if fs.exists(dest) :
    toStat = fs.getFileStatus(dest)
  else:
    toStat = None

  if src.isFile :
    if not toStat is None :
      fs.delete(dest, recursive = True)
    fs.rename(src.getPath, dest)
  else :
    # destination is directory, choose action on source type
    if src.isDirectory :
      if not toStat is None :
        if not toStat.isDirectory :
          # Destination exists and is not a directory
          fs.delete(dest)
          fs.mkdirs(dest)                                    #
          for child in fs.listStatus(src.getPath) :          # HERE
            mergePathsV2(fs, child, dest + child.getName)    #

        else :
          # Destination exists and is a directory
          # merge all children under destination directory
          for child in fs.listStatus(src.getPath) :
            mergePathsV2(fs, child, dest + child.getName)
      else :
        # destination does not exist
        fs.mkdirs(dest)                                      #
        for child in fs.listStatus(src.getPath) :            # HERE
             mergePathsV2(fs, child, dest + child.getName)   #
```

Algo v2 commitJob

- écrit un fichier vide _SUCCESS

```
def commitJob(fs, jobAttemptDir, dest):
    fs.touch("$dest/_SUCCESS")
```

File Output Committer v2

- Un seul déplacement de fichier
  - lors du taskCommit
    - `'$dest/_temporary/$appAttemptId/_temporary/$taskAttemptID/000.parquet' -> '$dest/000.parquet'`
  - Si le job échoue il faut supprimer tout les fichiers et recommencer

Solution 2: changer de committer

S3A Committer :

- StagingCommitter
  - Directory
  - Partitioned
- MagicCommitter

Comment choisir son committer ?

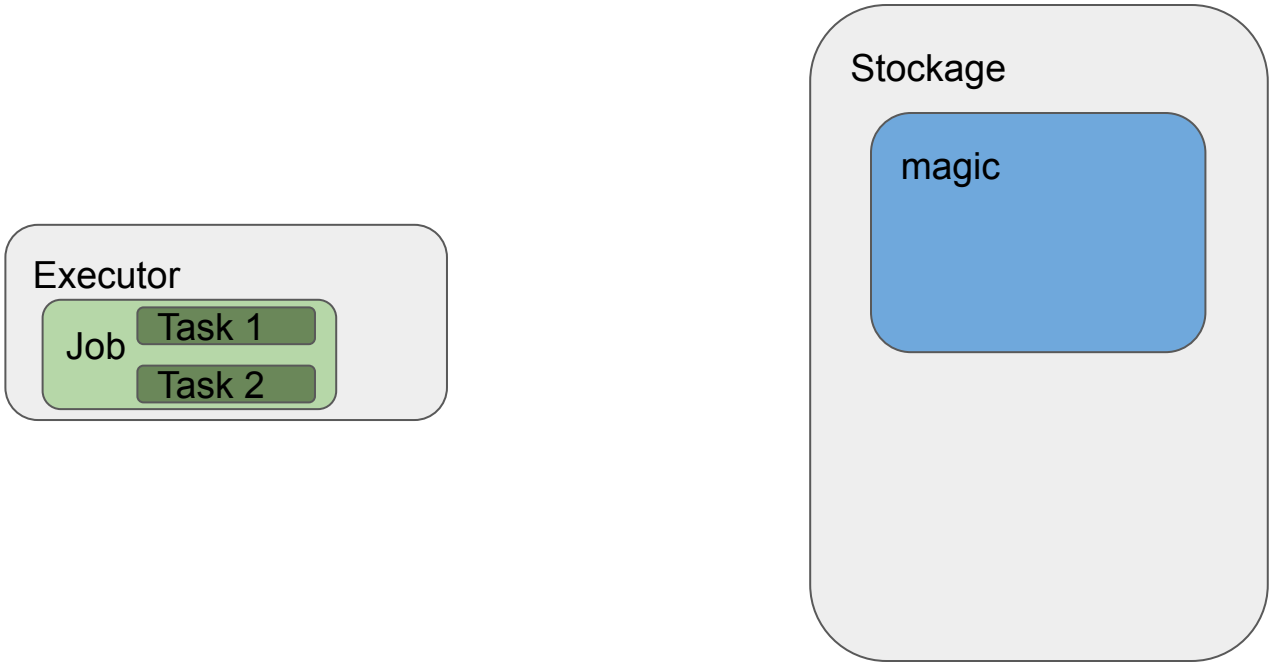
1. Écrire dans un fichier existant et partitionné : Partitioned Committer
2. Écrire un gros fichier : Magic Committer
3. Sinon : Directory Committer si le disque dur des executors le permet

Depuis que S3 est consistent par défaut, Magic Committer est recommandé

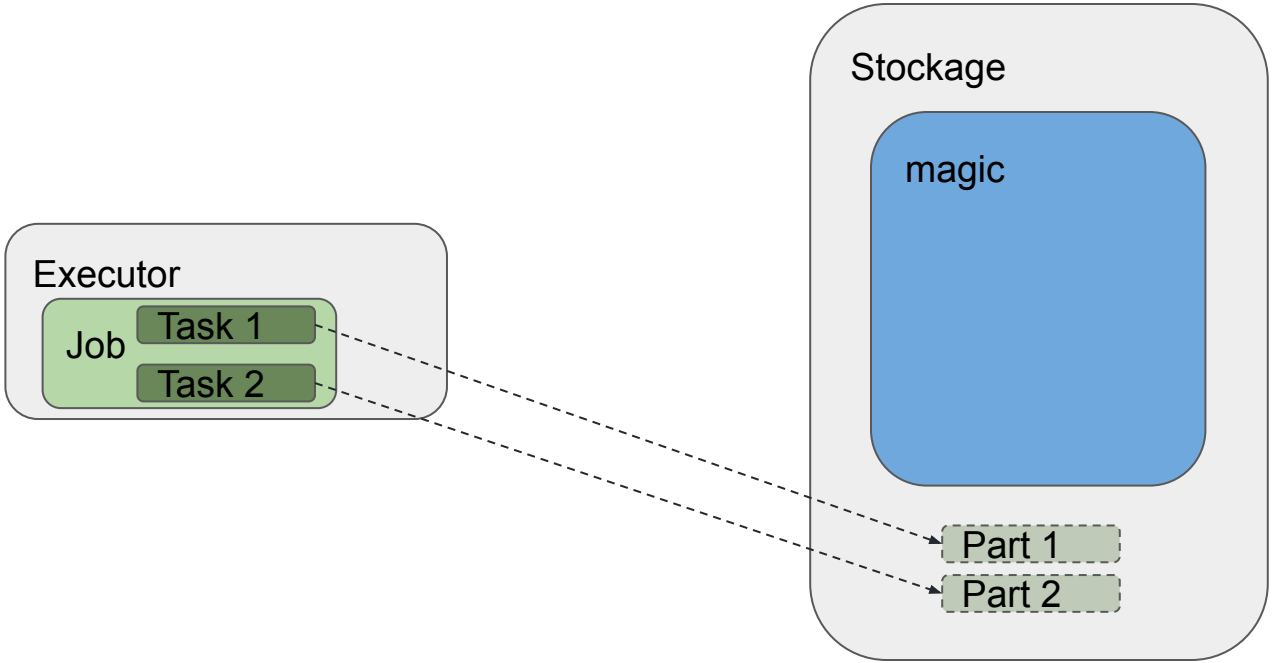
Magic Committer

- Utilise S3 comme FileSystem
- Écrit les fichiers au fil de l’eau dans S3
- Commit la liste des fichiers écrit dans S3
- Commit dans S3 à la fin du job tous les fichiers écrits

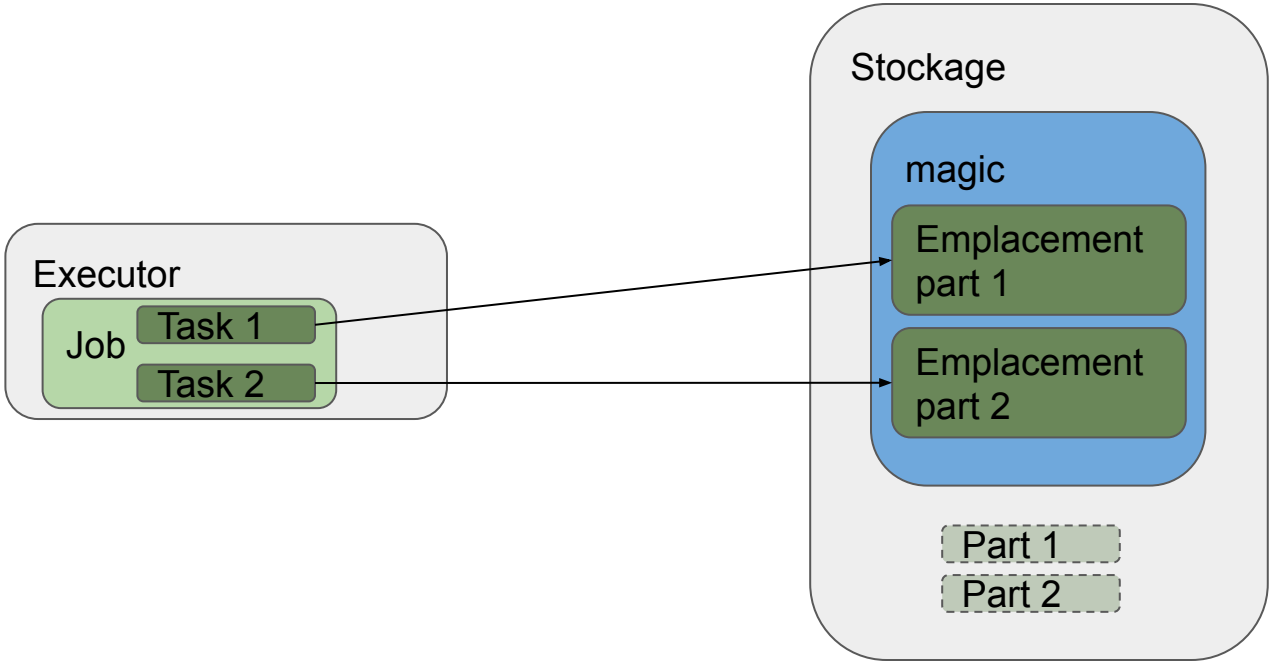
Task Commit



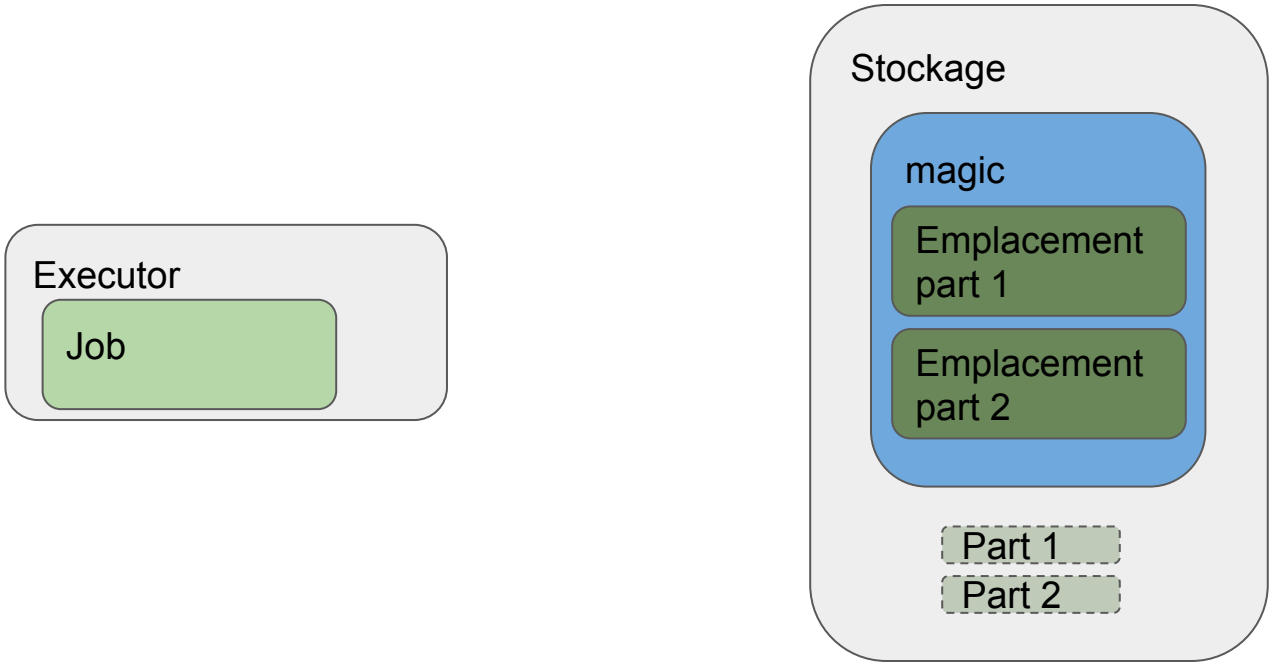
Task Commit



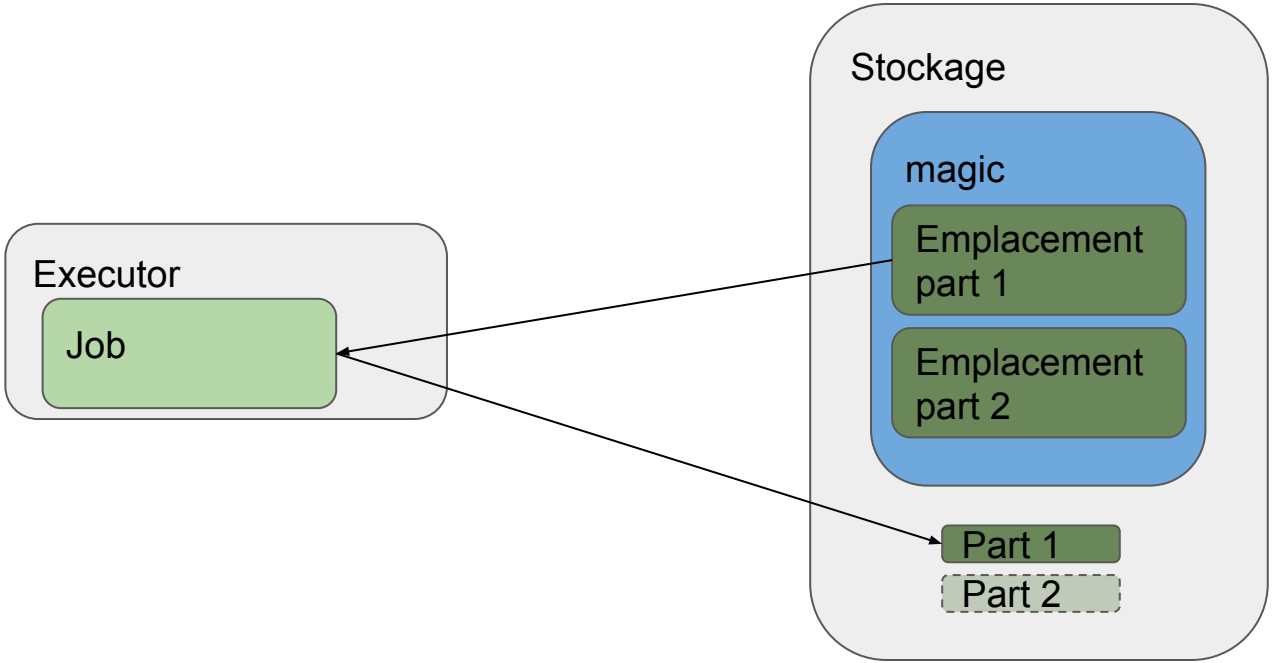
Task Commit


Stockage
magic
Emplacement part 1
Emplacement part 2
Part 1
Part 2
Executor
Job
Task 1
Task 2

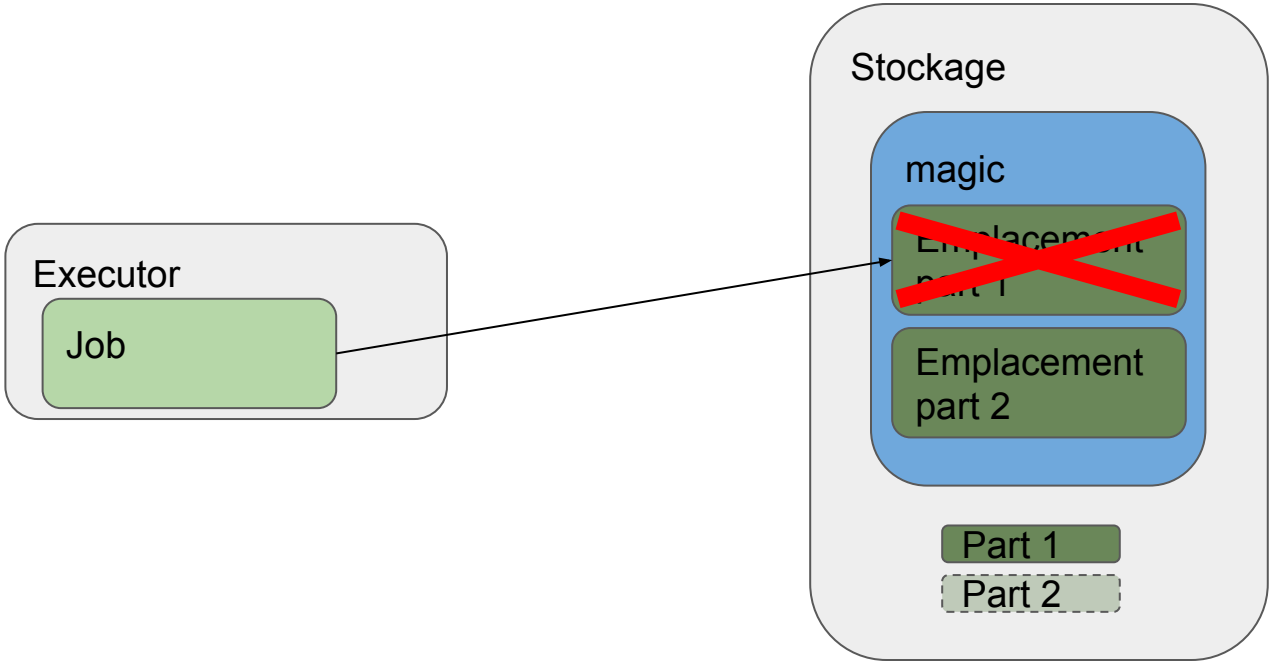
Task Commit


Executor
Job
Stockage
magic
Emplacement part 1
Emplacement part 2
Part 1
Part 2

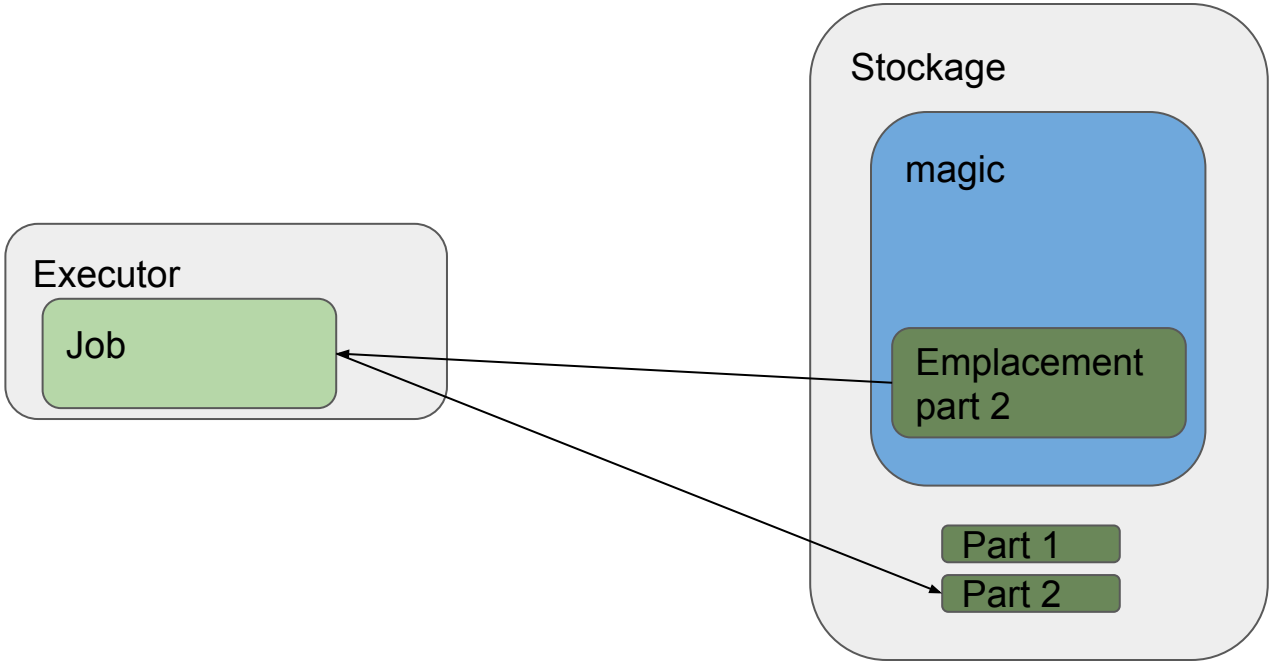
Job Commit


Stockage
magic
Emplacement part 1
Emplacement part 2
Executor
Job
Part 1
Part 2

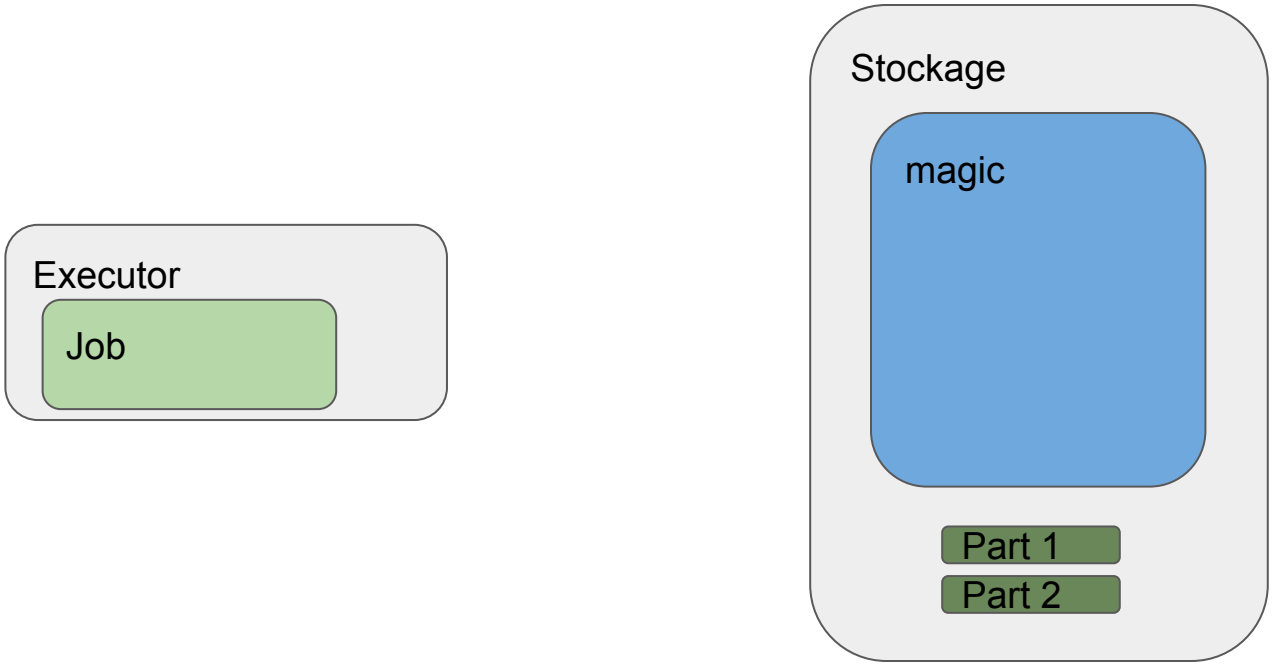
Job Commit


Executor
Job
Stockage
magic
Emplacement
part 2
Part 1
Part 2

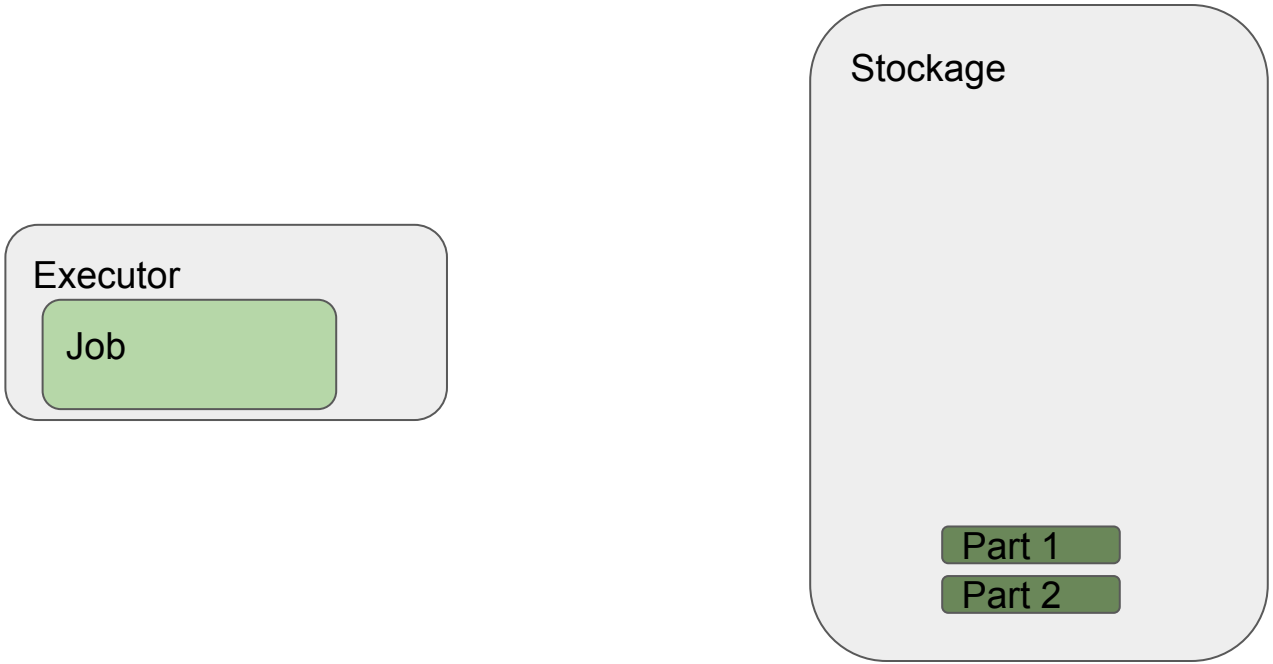
Job Commit


Stockage
magic
Executor
Job
Emplacement
part 2
Part 1
Part 2

Job Commit



Job Commit



Magic Committer

- Aucun déplacement de fichier
- 2 fois plus de fichiers à écrire
- Beaucoup de suppression de petits fichiers

Magic Committer : Hadoop conf

```
"fs.s3a.committer.magic.enabled" : "true"
"fs.s3a.committer.name" : "magic"
```

ATTENTION

Parquet implémente son propre Committer

Magic Committer avec Parquet : Hadoop conf

```
"fs.s3a.committer.magic.enabled": "true"
"fs.s3a.committer.name": "magic"
"mapreduce.outputcommitter.factory.scheme.s3a": "org.apache.hadoop.fs.s3a.commit.S3ACommitterFactory"
"spark.sql.sources.commitProtocolClass": "org.apache.spark.internal.io.cloud.PathOutputCommitProtocol"
"spark.sql.parquet.output.committer.class": "org.apache.spark.internal.io.cloud.BindingParquetOutputCommitter"
"mapreduce.fileoutputcommitter.algorithm.version": "2"
```

Test Parquet : Magic Committer vs algorithme v2

- 720 020 485 lignes (~500 partitions)
- 5 executors
  - 3 coeurs
  - 16Go de RAM
- Magic Committer : 16 minutes
- Algorithme v2 : 15 minutes

Test Parquet : Magic Committer vs algorithme v2

- 720 020 485 lignes (~500 partitions)
- 5 executors
  - 2 coeurs
  - 8Go de RAM
- Magic Committer : 16 minutes 30
- Algorithme v2 : 18 minutes

Conclusion

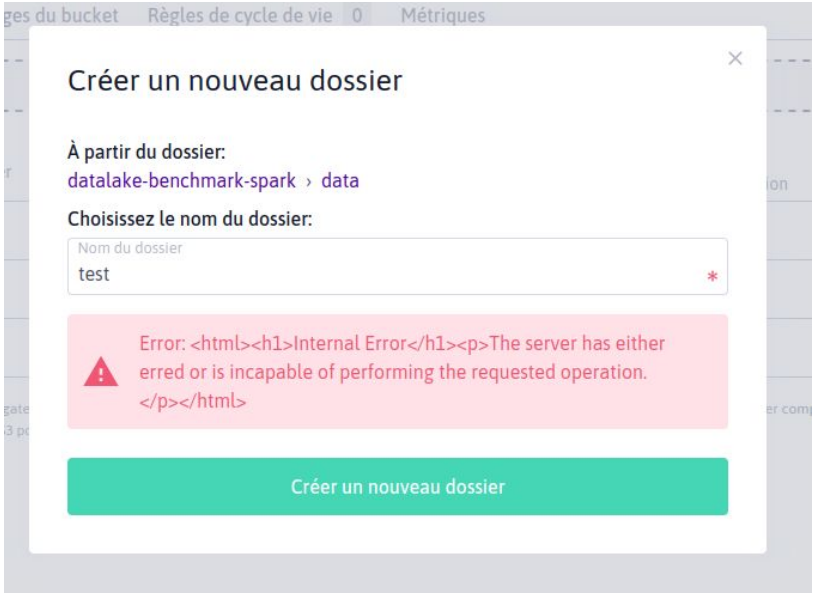
- Packager son jar dans une image préconfigurée
- Passer du *spark-submit* à un manifest yaml
- Lancer un spark history server
- Faire attention au committer si on est sur S3
  - Algo V2
  - Magic Committer

Et Object Storage qui crash ?

Les erreurs Object Storage

Algorithme v2 :

- > 70 executors
  - 3 coeurs
  - 16Go de RAM
- Écriture bloquée



Les erreurs Object Storage

Magic Committer :

- > 40 executors
  - 3 coeurs
  - 16Go de RAM
- 400 Bad Request
  - AmazonS3Exception
- 500 Internal Error
  - Max redirections to a leader have been reached

link

- https://hadoop.apache.org/docs/r3.2.0/hadoop-aws/tools/hadoop-aws/committer_architecture.html

Merci pour votre attention