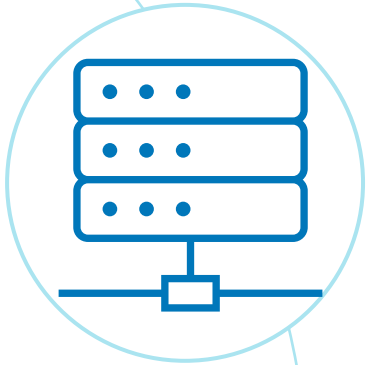


Réalisez un traitement dans un environnement Big Data sur le Cloud

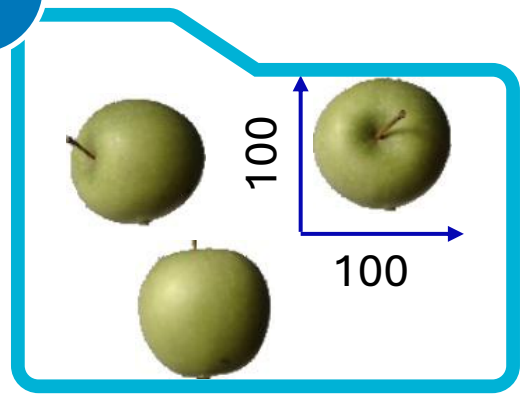
Projet 11

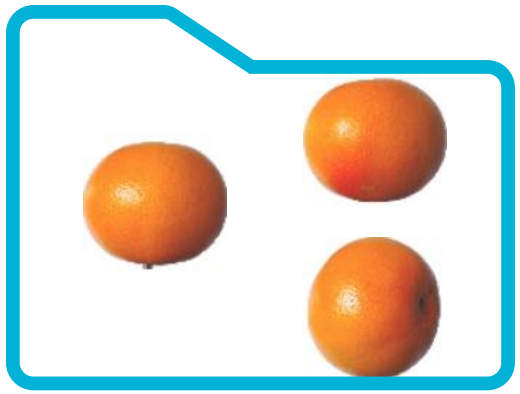
AgriTech souhaite contribuer à une agriculture plus raisonnée grâce à un usage **plus limité** des traitements phytosanitaires.
L'objectif est de proposer à terme un algorithme pour des **robots cueilleurs** qui identifiera les fruits afin d'appliquer le traitement adéquat.
Avant cela nous allons vous présenter la première phase du projet qu'est la conception du **moteur de classification.**
Sur la base du travail réalisé par l'alternant nous allons vous exposer la méthodologie de mise en production du modèle sur le **cloud.**

- Présentation des données
- Architecture
- Configuration
- Préparation des données
- Traitements
- Stockage résultats
- RGPD

Présentation des données

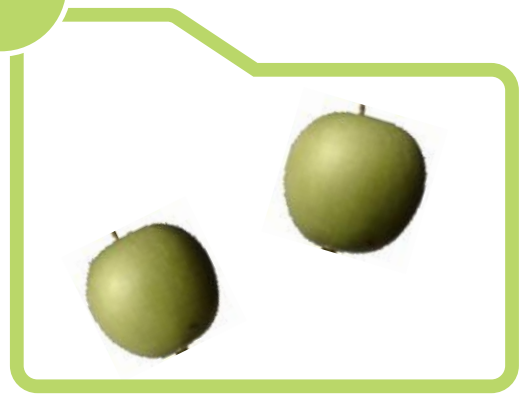
Train

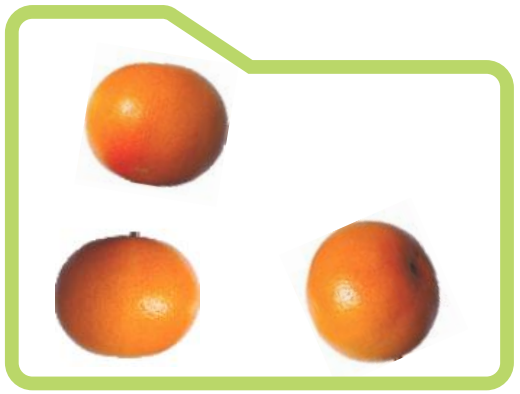




67.692
images

Test





22.688
images

```
result.show()
```

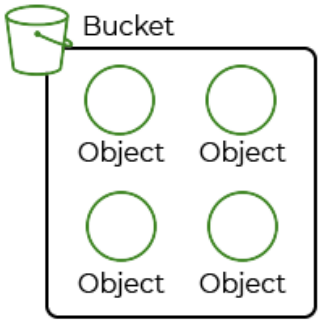
```
+--------------------+---------------+--------------------+
|                path|          label|        pca_features|
+--------------------+---------------+--------------------+
|s3://p11-ia/Test/...|Pineapple Mini 1|[-4.8636752277107...|
|s3://p11-ia/Test/...|    Watermelon 1|[-2.7510919463027...|
|s3://p11-ia/Test/...|    Watermelon 1|[-1.8006401827213...|
|s3://p11-ia/Test/...|Pineapple Mini 1|[-4.1113366599291...|
|s3://p11-ia/Test/...|   Cauliflower 1|[-5.0680672005228...|
|s3://p11-ia/Test/...|     Raspberry 1|[0.50051455921299...|
|s3://p11-ia/Test/...|   Cauliflower 1|[-6.0323608291431...|
|s3://p11-ia/Test/...|   Cauliflower 1|[-4.4727961268549...|
|s3://p11-ia/Test/...|   Cauliflower 1|[-4.5577351868231...|
|s3://p11-ia/Test/...|   Cauliflower 1|[-4.8705659298443...|
|s3://p11-ia/Test/...|     Pineapple 1|[-6.0206880865931...|
|s3://p11-ia/Test/...|   Cauliflower 1|[-5.5567624402274...|
|s3://p11-ia/Test/...|   Cauliflower 1|[-5.1424696226717...|
|s3://p11-ia/Test/...|   Cauliflower 1|[-6.0582350846046...|
|s3://p11-ia/Test/...|   Cauliflower 1|[-4.6452776976900...|
|s3://p11-ia/Test/...| Cucumber Ripe 1|[-0.2399424367527...|
|s3://p11-ia/Test/...|  Apple Golden 1|[-2.1089238473189...|
|s3://p11-ia/Test/...|Pineapple Mini 1|[-5.9859947365781...|
|s3://p11-ia/Test/...|  Pear Forelle 1|[1.88205655032966...|
|s3://p11-ia/Test/...|      Rambutan 1|[-1.7766259372150...|
+--------------------+---------------+--------------------+
only showing top 20 rows
```

Briques d'architecture choisies

Simple Storage Services

Service de stockage de fichiers (objets) avec différentes options notamment:

- La configuration de droits d'accès pour chaque fichier.
- Chiffrage de toute ou partie du contenu
- Versioning
- Configuration d’une date d'expiration des fichiers
- Réplication de fichiers sur plusieurs datacenters AWS

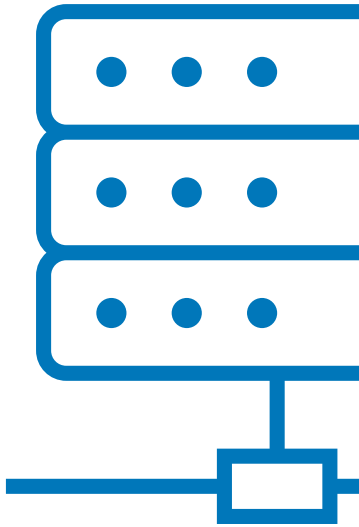


A noter que ce service n’est pas obligatoire car les serveurs EC2 peuvent stocker les données. Cependant lorsque les instances EC2 sont **résiliées, leurs données sont supprimées**, ce qui n’est pas le cas de S3

Briques d'architecture choisies

Elastic Compute Cloud

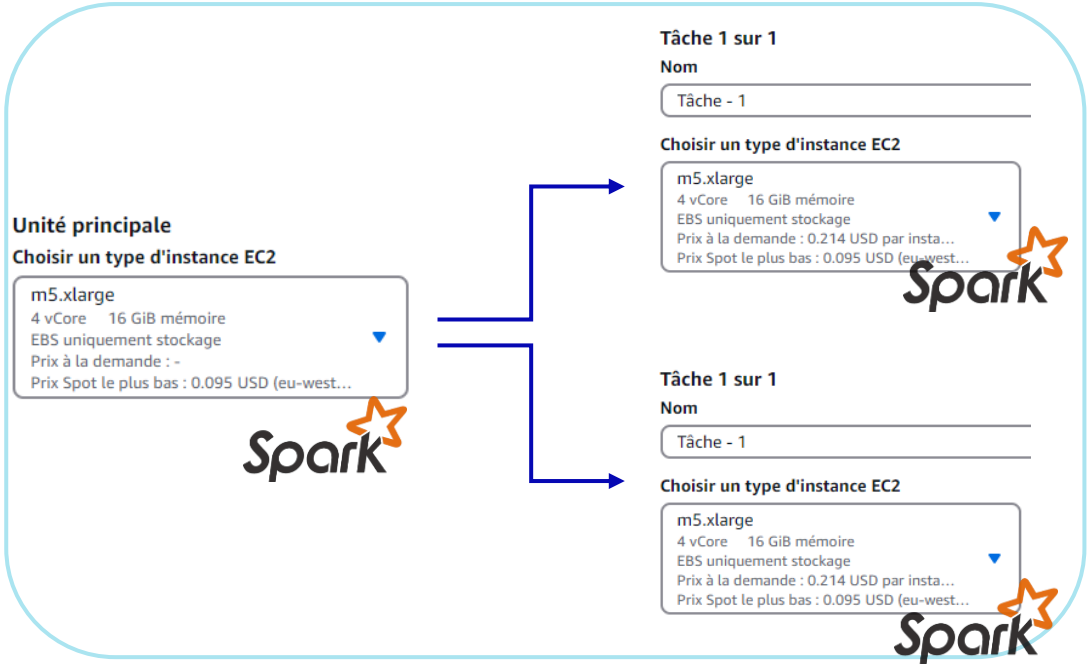
Service de location de serveur. Le ou les serveurs peuvent être utilisés vierges de toute installation ou préconfigurés.
Dans notre projet, nous passerons par des instances EMR donc préconfigurées avec les outils nécessaires (Spark, Hadoop… pour la construction de nos clusters.



Briques d'architecture choisies

Elastic Map Reduce

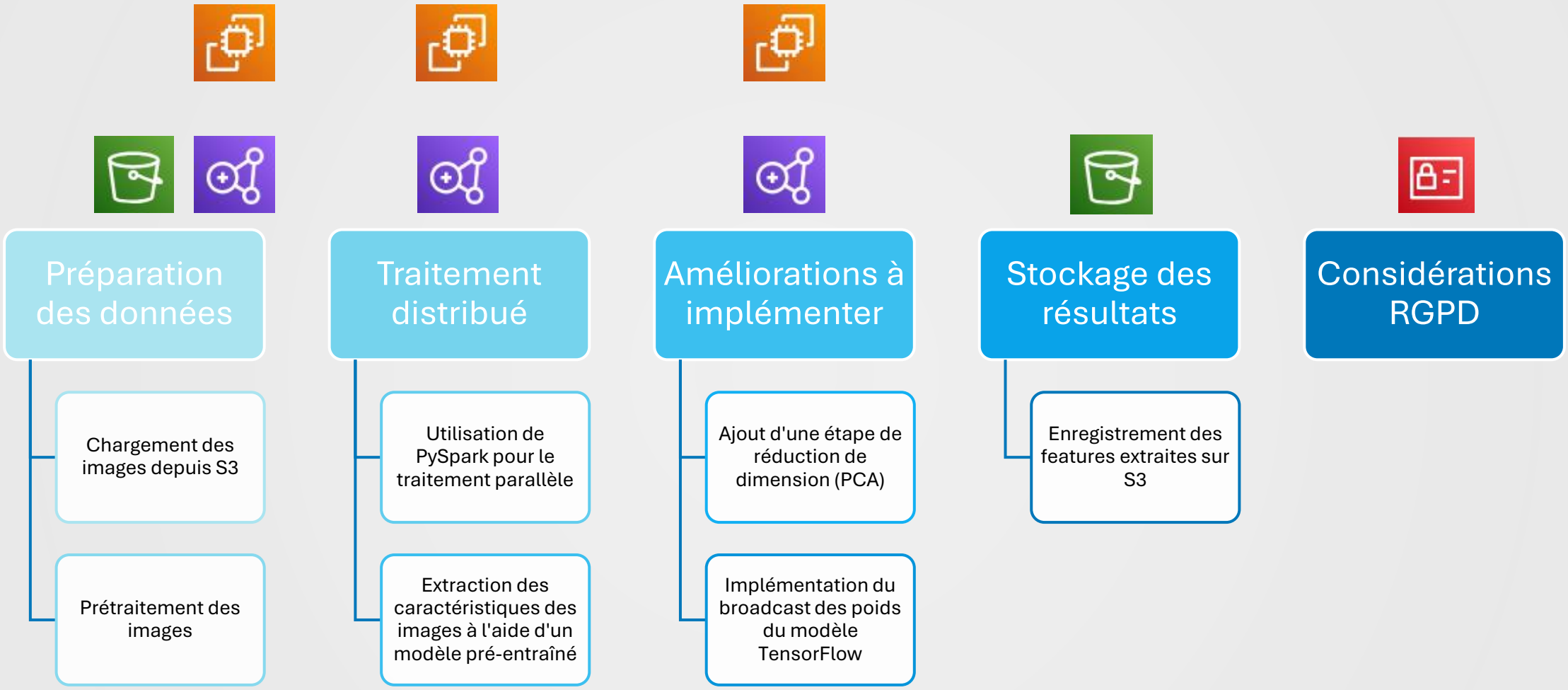
Service de calcul distribué de Amazon Web Services utilisant Spark comme moteur de traitement



Briques d'architecture choisies

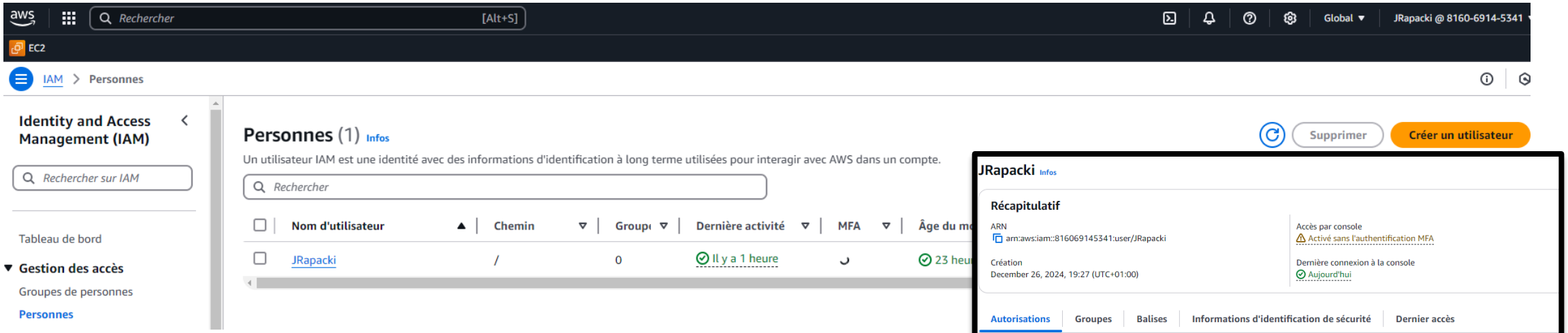
Idendity and Access Management

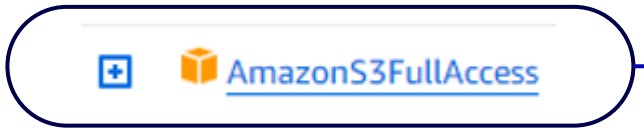
Service central d'AWS pour contrôler l'accès aux ressources AWS.
Il permet d'authentifier les utilisateurs et de gérer leurs autorisations


Préparation des données
Chargement des images depuis S3
Prétraitement des images
Traitement distribué
Utilisation de PySpark pour le traitement parallèle
Extraction des caractéristiques des images à l'aide d'un modèle pré-entraîné
Améliorations à implémenter
Ajout d'une étape de réduction de dimension (PCA)
Implémentation du broadcast des poids du modèle TensorFlow
Stockage des résultats
Enregistrement des features extraites sur S3
Considérations RGPD

Configuration

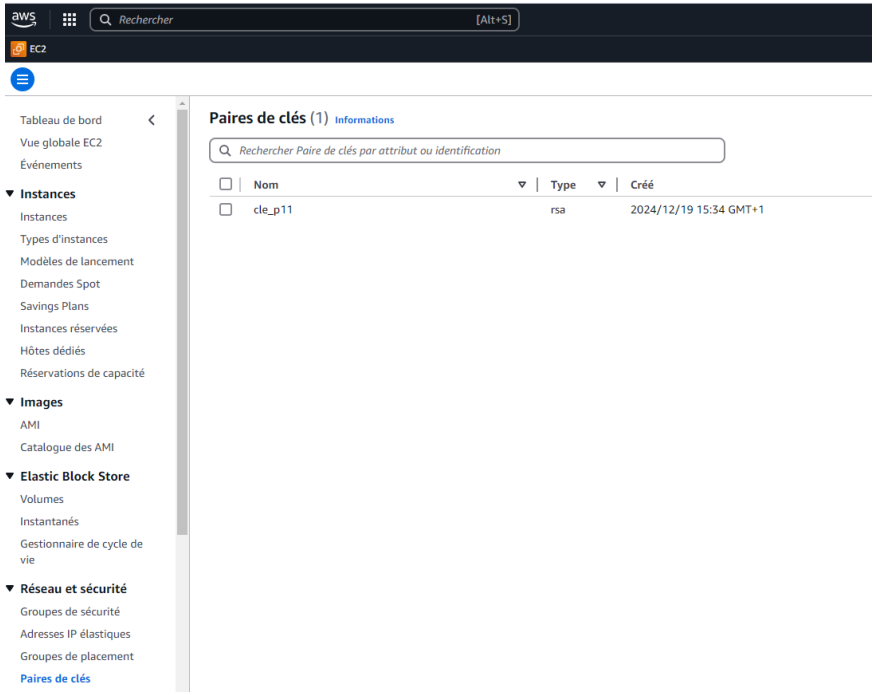
Création user | SSH | Création cluster | Chargement des données | Traitement des images | Stockage des résultats

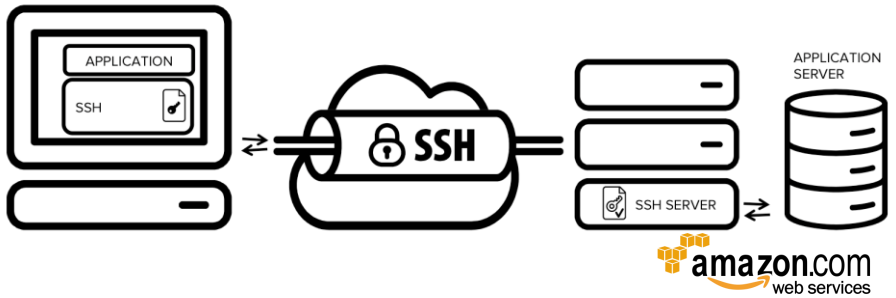




Configuration

Création user | SSH | Création cluster | Chargement des données | Traitement des images | Stockage des résultats

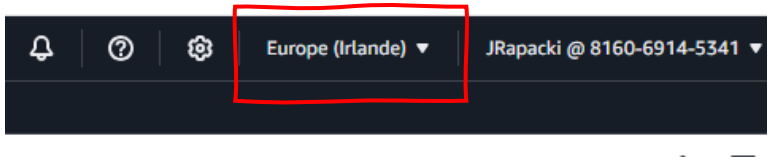


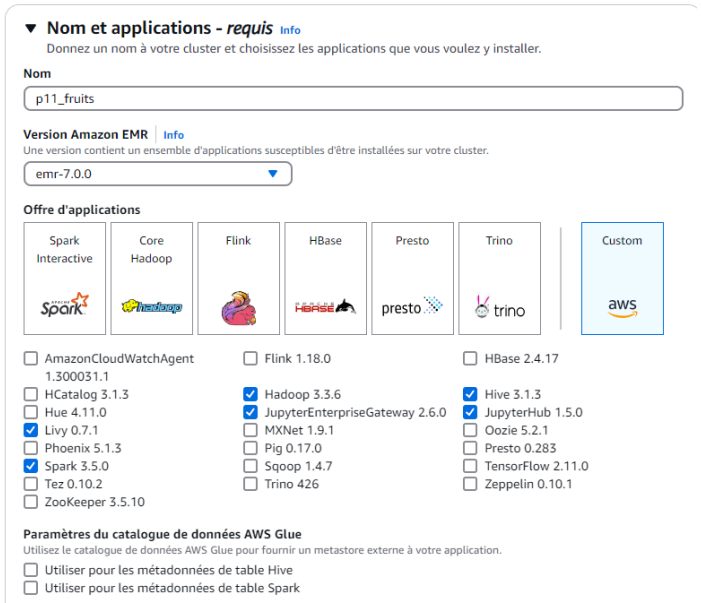


Le tunnel SSH va crypter nos échanges avec les serveurs d'AWS. Il permet aussi d'éviter la saisie de mot de passe à chaque connexion.

Configuration

Création user | SSH | Création cluster | Chargement des données | Traitement des images | Stockage des résultats

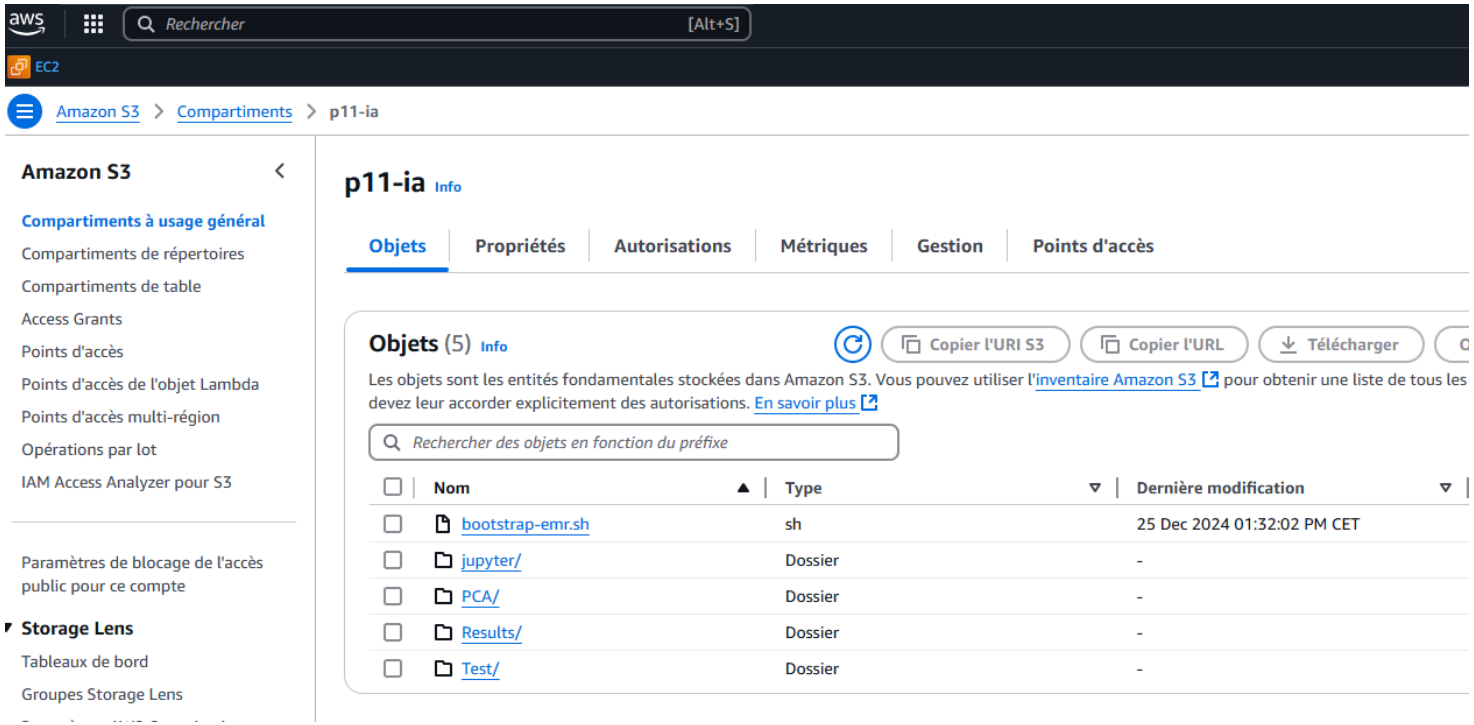


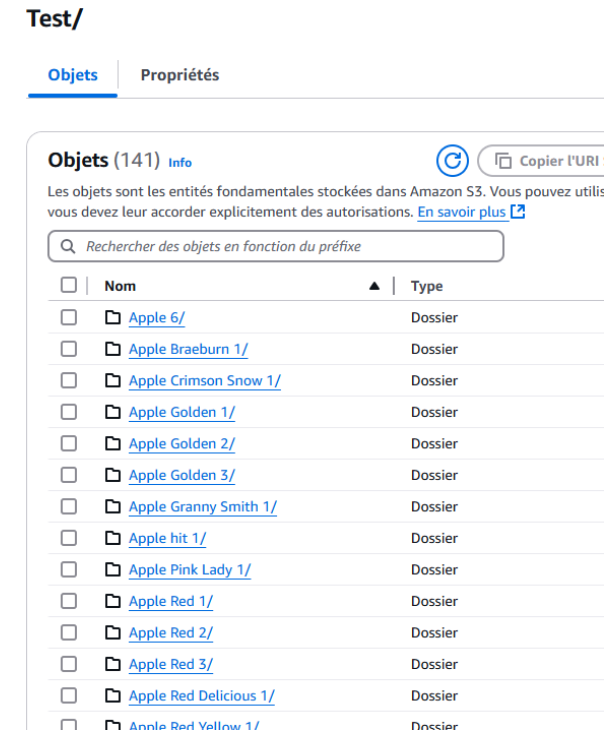


La version EMR est la **7.0.0** car la dernière version (7.6.0) provoquait des soucis de compatibilité avec la librairie Pandas notamment.
La dernière version demandait l'ajout d'autre librairies via le bootstrap (« rich » entre-autre). Malgré ces modification le problème avec Pandas était présent

Préparation des données

Création user | SSH | Création cluster | **Chargement des données** | Traitement des images | Stockage des résultats





Préparation des données

Création user | SSH | Création cluster | **Chargement des données** | Traitement des images | Stockage des résultats

La définition de la stratégie de sécurité du stockage S3 est très importante car elle doit permettre la lecture des données mais aussi l'écriture pour l'enregistrement de nos features

Éditeur de politique

```
19        "Resource": [
20            "arn:aws:s3:::aws-logs-816069145341-eu-west-1/elasticmapreduce",
21            "arn:aws:s3:::aws-logs-816069145341-eu-west-1/elasticmapreduce/*"
22        ]
23    },
24    {
25        "Effect": "Allow",
26        "Action": [
27            "s3:GetBucketVersioning",
28            "s3:GetObject",
29            "s3:GetObjectTagging",
30            "s3:GetObjectVersion",
31            "s3:ListBucket",
32            "s3:PutObject",
33            "s3:ListBucketMultipartUploads",
34            "s3:ListBucketVersions",
35            "s3:DeleteObject",
36            "s3:ListMultipartUploadParts"
37        ],
38        "Resource": [
39            "arn:aws:s3:::elasticmapreduce",
40            "arn:aws:s3:::aws-logs-816069145341-eu-west-1/elasticmapreduce",
41            "arn:aws:s3:::elasticmapreduce/*",
42            "arn:aws:s3:::aws-logs-816069145341-eu-west-1/elasticmapreduce/*",
43            "arn:aws:s3:::*.elasticmapreduce/*",
44            "arn:aws:s3:::p11-ia/*",
45            "arn:aws:s3:::p11-ia"
46        ]
```

```
"arn:aws:s3:::p11-ia/*"
"arn:aws:s3:::p11-ia"
```

Préparation des données

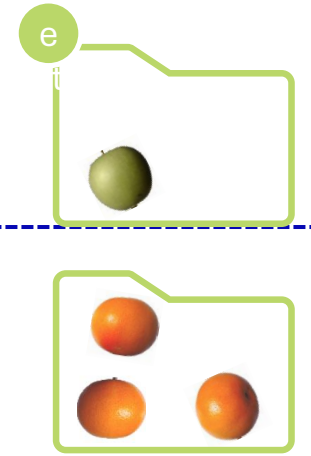
MobileNetV2 a été pré-entraîné sur la base d'image appelée **ImageNet**.

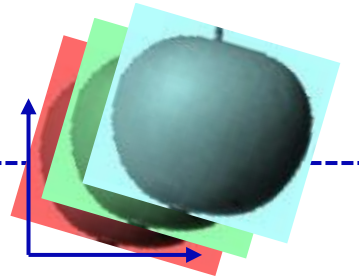
Le jeu de données ImageNet le plus utilisé, ILSVRC 2012-2017, est composé d'environ 1.5 million d'images, réparties en environ 90 % d'images d'entraînement, 3 % de validation et 7 % de test. 1000 classes d'objets y sont identifiées.

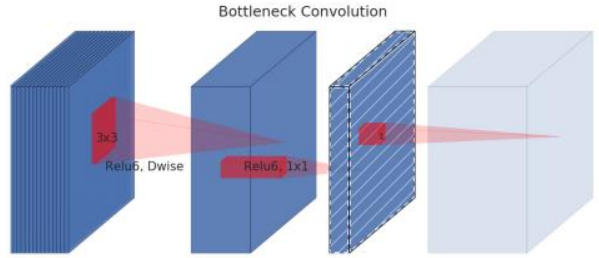
Le modèle est donc capable de classer 1000 catégories d'objet mais ce n'est pas l'objectif du projet. Le but est d'obtenir des caractéristiques permettant de classer l'objet.

Ces caractéristiques ont la forme d'un tenseur **1,1,1280.** C'est-à-dire que chaque image est caractérisée par **1280 variables**.

Traitement des données







[[v1,v2,v3.....v1280]]

```
[ ]  df.head()
```

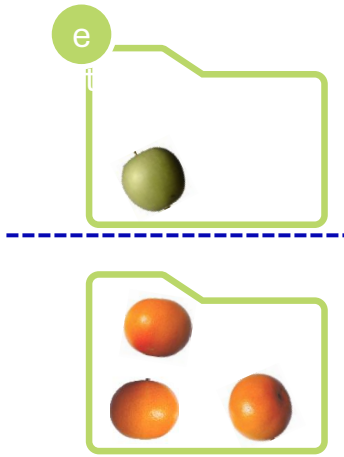
	path	label	features
0	file:/content/drive/MyDrive/Colab Notebooks/PR...	Lychee	[0.9803019, 2.1587136, 0.0, 0.0, 0.024135016, ...
1	file:/content/drive/MyDrive/Colab Notebooks/PR...	Melon Piel de Sapo	[1.3291984, 0.027475864, 0.06831567, 0.0, 0.0,...
2	file:/content/drive/MyDrive/Colab Notebooks/PR...	Apple Red Yellow 1	[0.45133322, 0.0, 0.0, 0.0, 0.0, 0.009732636, ...
3	file:/content/drive/MyDrive/Colab Notebooks/PR...	Tangelo	[0.38730422, 0.020577567, 0.0, 0.0, 0.0, 0.023...
4	file:/content/drive/MyDrive/Colab Notebooks/PR...	Pear Stone	[0.04176776, 0.0, 0.42859992, 0.0, 0.23425241,...

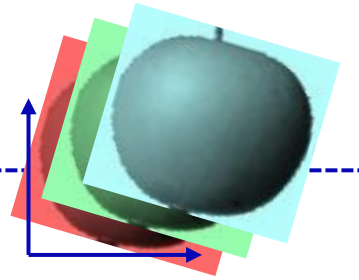
On valide que la dimension du vecteur de caractéristiques des images est bien de dimension 1280 :

```
df.loc[0,'features'].shape
```

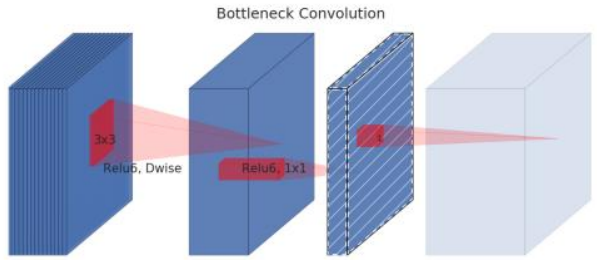
```
(1280,)
```

Traitement des données





224



[[v1,v2,v3.....v1280]]

```
[ ]  df.head()
```

	path	label	features
0	file:/content/drive/MyDrive/Colab Notebooks/PR...	Lychee	[0.9803019, 2.1587136, 0.0, 0.0, 0.024135016, ...
1	file:/content/drive/MyDrive/Colab Notebooks/PR...	Melon Piel de Sapo	[1.3291984, 0.027475864, 0.06831567, 0.0, 0.0,...
2	file:/content/drive/MyDrive/Colab Notebooks/PR...	Apple Red Yellow 1	[0.45133322, 0.0, 0.0, 0.0, 0.0, 0.009732636, ...
3	file:/content/drive/MyDrive/Colab Notebooks/PR...	Tangelo	[0.38730422, 0.020577567, 0.0, 0.0, 0.0, 0.023...
4	file:/content/drive/MyDrive/Colab Notebooks/PR...	Pear Stone	[0.04176776, 0.0, 0.42859992, 0.0, 0.23425241,...

On valide que la dimension du vecteur de caractéristiques des images est bien de dimension 1280 :

```
df.loc[0,'features'].shape
```

```
(1280,)
```

Traitement des données ACP

Conversion des features array -> vecteur pour l'ACP

Entrée [17]:
```
from pyspark.ml.linalg import Vectors,VectorUDT
from pyspark.sql.functions import udf
from pyspark.sql.types import DoubleType
from pyspark.ml.functions import vector_to_array
```
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…

Entrée [18]:
```
def to_vector(array):
    return Vectors.dense(array)

vector_udf = udf(to_vector, VectorUDT())
```
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…

Entrée [19]:
```
#convertion des features array en vecteurs
features_to_vec_df =  features_df.withColumn("features", vector_udf(features_df["features"]))
```
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…

Traitement des données

ACP

Détermination du k optimal pour la variance expliquée

Entrée [20]:
```
from pyspark.ml.feature import PCA
from pyspark.ml.feature import VectorAssembler
from pyspark.sql.functions import col


```
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…

Entrée [21]:
```

pca = PCA(k=1000, inputCol="features", outputCol="pcaFeatures")
model = pca.fit(features_to_vec_df)
result = model.transform(features_to_vec_df)
```
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…

Entrée [23]:
```
explained_variance = model.explainedVariance.toArray()
cumulative_variance = explained_variance.cumsum()
```
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…

Entrée [24]:
```
k_optimal = (cumulative_variance >= 0.90).argmax() + 1
```
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…

Entrée [25]:
```
k_optimal
```
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…

```
np.int64(191)
```

Traitement des données

ACP

Finalisation, vérification

Entrée [33]:
```
df_PCA = pd.read_parquet(PATH_PCA, engine='pyarrow')
```

Entrée [34]:
```
df_PCA.head()
```

	path	...	array_pcafeatures
0	s3://p11-ia/Test/Pineapple Mini 1/140_100.jpg	...	[-6.428251636300699, 4.352973200107818, -0.224...
1	s3://p11-ia/Test/Watermelon 1/284_100.jpg	...	[-2.751091946278985, 2.1131146099898794, -8.28...
2	s3://p11-ia/Test/Watermelon 1/r_53_100.jpg	...	[-1.8006401821512406, 4.482510896774938, -8.08...
3	s3://p11-ia/Test/Watermelon 1/135_100.jpg	...	[-2.213427618870578, 2.526063557442736, -8.576...
4	s3://p11-ia/Test/Cauliflower 1/r_186_100.jpg	...	[-5.068067199610342, 2.5937451196174237, -0.49...

[5 rows x 3 columns]

Entrée [35]:
```
df_PCA.shape
```

```
(23619, 3)
```

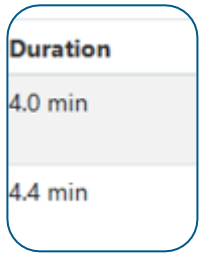
Entrée [37]:
```
df_PCA.loc[0,'array_pcafeatures'].shape
```

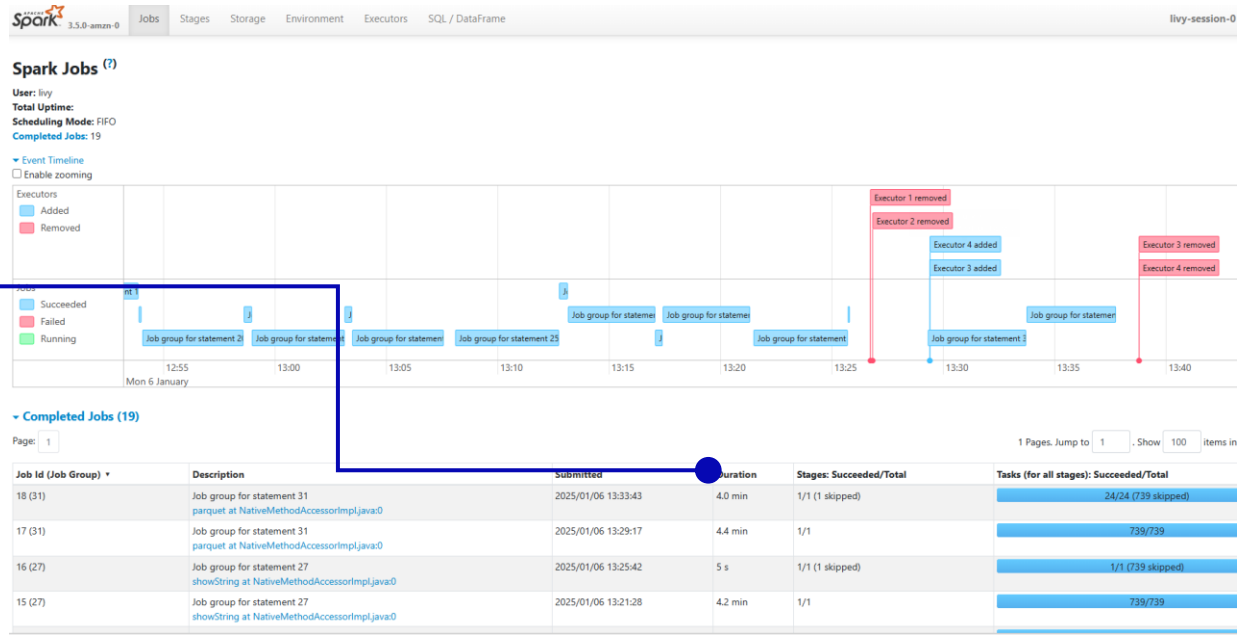
```
(191,)
```

Traitement des données

Création user | SSH | Création cluster | Chargement des données | **Traitement des images** | Stockage des résultats

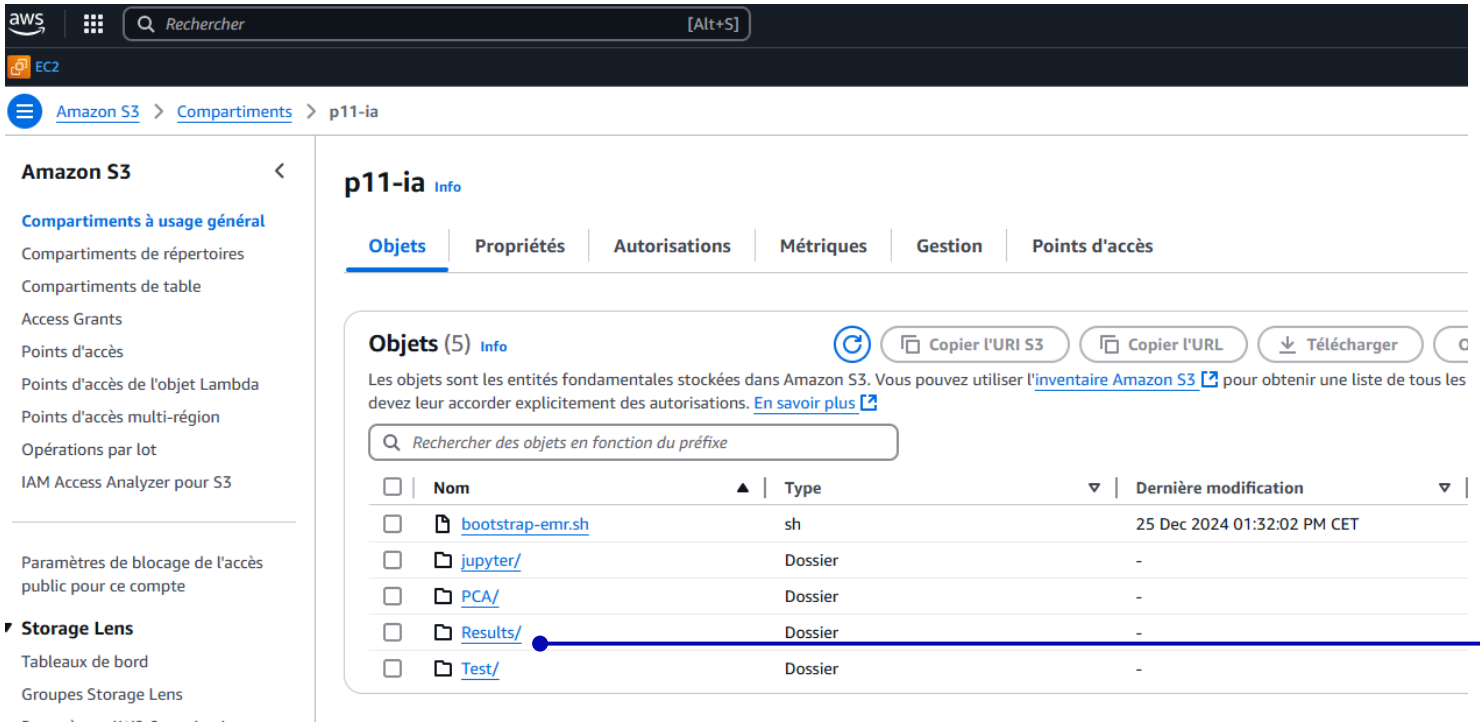
Les parties les plus exigeantes en termes de ressources de calcul sont sans surprise la génération des factures (~22000 matrices de 1x1280) ainsi que la réduction de dimension (ACP)

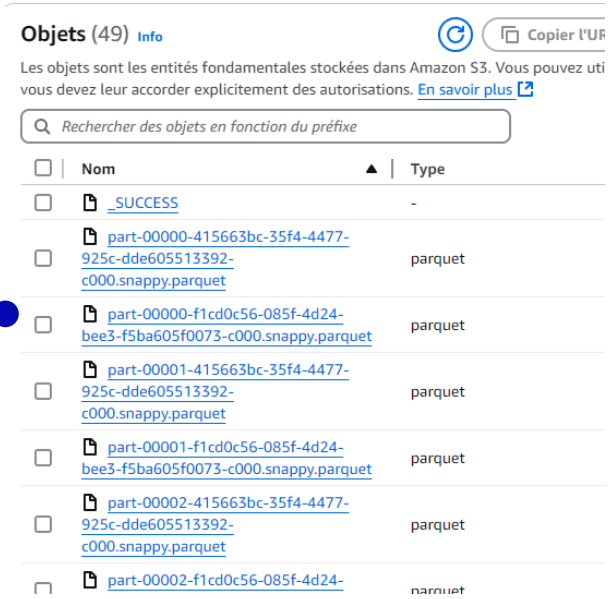




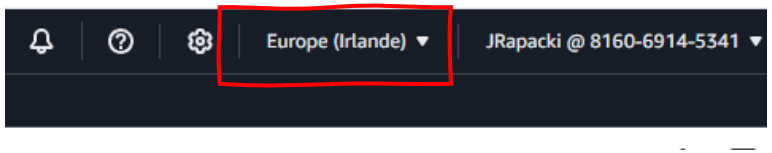
Stockage

Création user | SSH | Création cluster | Chargement des données | Traitement des images | Stockage des résultats





RGPD



Données anonymes

5 PRINCIPE

Licéité, loyauté et transparence : Les données personnelles doivent être traitées de manière licite, équitable et transparente.

Limitation des finalités : Les données doivent être collectées pour des finalités spécifiques, explicites et légitimes.

Minimisation des données : Les données collectées doivent être adéquates, pertinentes et limitées à ce qui est nécessaire.

Exactitude : Les données personnelles doivent être exactes et tenues à jour si nécessaire

Intégrité et confidentialité : Les données doivent être traitées de façon à garantir leur sécurité et leur protection contre le traitement non autorisé ou illicite, la perte, la destruction ou les dégâts d'origine accidentelle

Tunnel SSH + droits S3

Conclusion

Nous avons pu nous appuyer sur le notebook de l'alternant pour mettre en œuvre un calcul distribué sur AWS.

Au-delà du respect nécessaire de la RGPD, c'est surtout la sécurité qui nous semble être une priorité dès lors que l'on a recours au cloud.

Un accent doit être mis sur la gestion des rôles, utilisateurs et stratégies de compartiments S3.

La rapidité de calcul pourra être facilement améliorée par une simple augmentation du nombre de serveurs et de leur puissance.