



Déployez un modèle dans
le cloud

PLAN

- 1/ Présentation de la problématique
- 2/ Présentation du Big Data
- 3/ Présentation de la base de données
- 4/ Présentation de l'architecture cloud
- 5/ Présentation du travail effectué
- 6/ Conclusion et suite du projet



Fruits!

Data scientist dans start-up de l'AgriTech, "Fruits!"

Mettre en oeuvre des solutions innovantes de récolte de fruits : développer des robots cueilleurs tout en respectant la biodiversité

Objectifs :

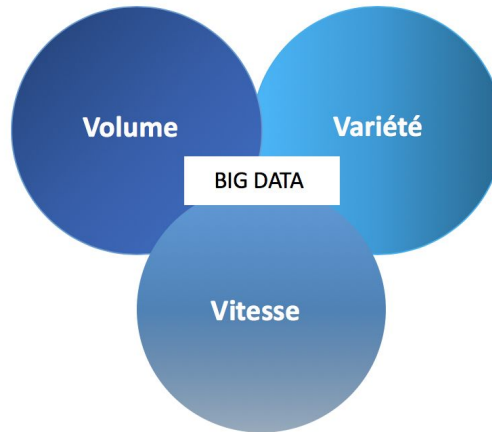
- Mettre à disposition une application publique
- Mettre en place une architecture Big Data

Mission :

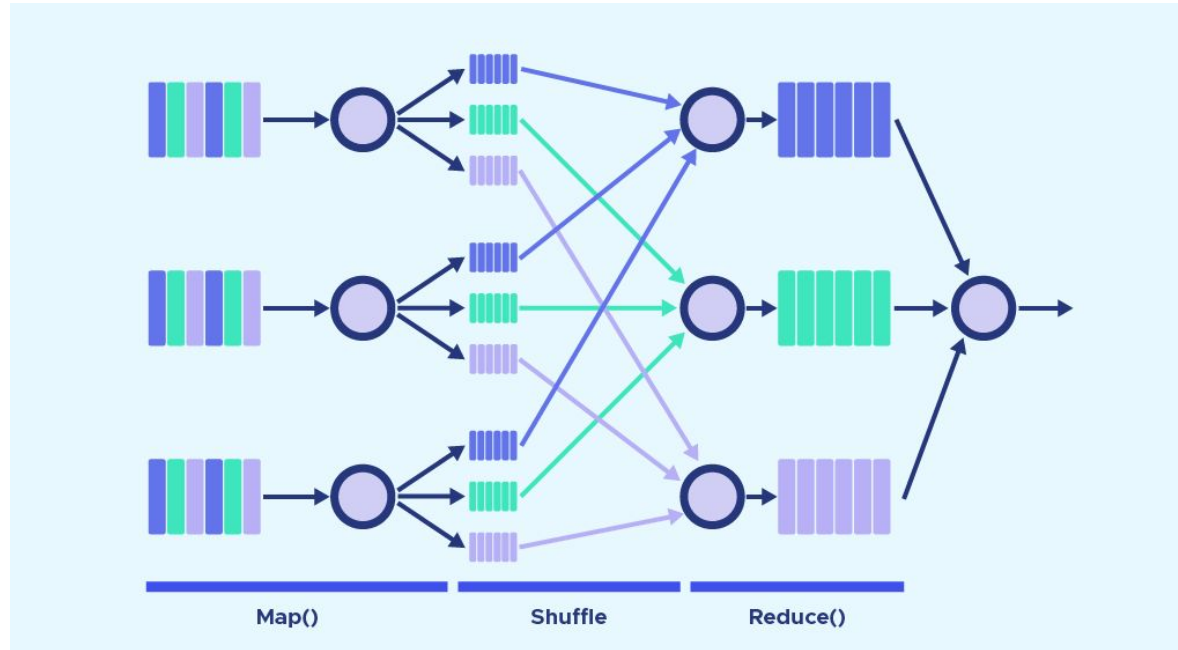
- Développer architecture Big Data
- Première chaîne de traitement (preprocessing et réduction de dimension)

- **Big Data ?**

- Quantité de données excède la faculté d'une machine à les stocker et les traiter dans un temps acceptable (référence capacité de la RAM)
- Quantité de ressources de calcul : paralléliser les calculs sur plusieurs machines



- Calculs distribués : MapReduce



- Dataset Kaggle : Fruits 360

- Photos de 131 espèces de fruits (photos sur 3 axes à 360°)
- Arrière plan reconstruit sur fond blanc
- Dimensions 100pxl x 100pxl en RGB \Rightarrow (100, 100, 3)
- Deux dossiers :
 - Train avec 67 692 images
 - Test avec 22 688 images



(pour faciliter les calculs de l'exercice \Rightarrow échantillon de 4 photos pour 6 espèces)

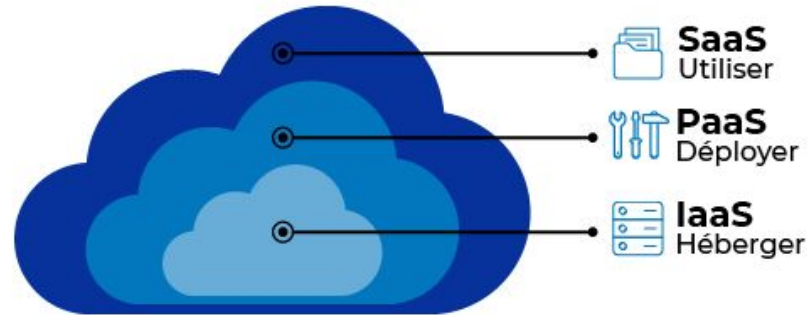
- Big Data ⇒ Cloud
Plusieurs solutions :



Location de serveurs selon la puissance de calcul nécessaire
Accès à des serveurs dans zones géographiques selon utilisation

⇒ Service de mise à disposition de machines (configurées et remplacées/entretenuées)

- Plusieurs type de cloud



PaaS (Platform as a Service): fournit accès à l'infrastructure et fonctionnalités, le nombre de machines nécessaires pour distribuer le travail est géré automatiquement

- Amazon Web Service (AWS)

- Services Amazon EC2 : "Elastic Compute Cloud"

Service pour lancer les serveurs

Configuration système d'exploitation, mémoire et stockage

"Élastique" : possibilité d'ajouter ou enlever des serveurs selon les besoins

⇒ Lancer notre instance



Type d'instance

t2.medium

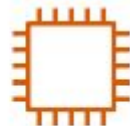
Famille: t2 2 vCPU 4 GiB Mémoire

À la demande Linux tarification: 0.0528 USD par heure

À la demande Windows tarification: 0.0708 USD par heure



Elastic IP address



Instance EC2

- Amazon Web Service (AWS)

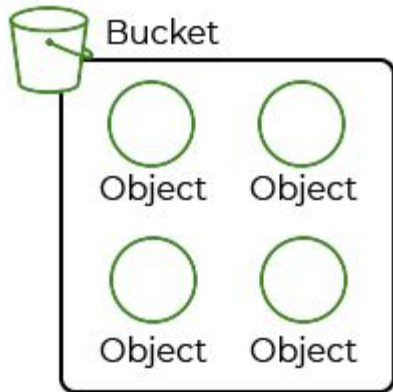
- Services Amazon S3 : "Simple Storage Service"

Service de stockage des données sur internet

⇒ Créer nos buckets S3 pour stocker les images et les résultats
Redéfinir les droits d'accès aux buckets (service IAM)



Amazon S3



	Nom	
<input type="radio"/>	sparkyfruitp8	
<input type="radio"/>	sparkyfruitp8-results	



- Configuration instance EC2

- Installer Jupyter Notebook
- Installer les librairies nécessaires (tensorflow, boto3 etc.)
- Installer PySpark

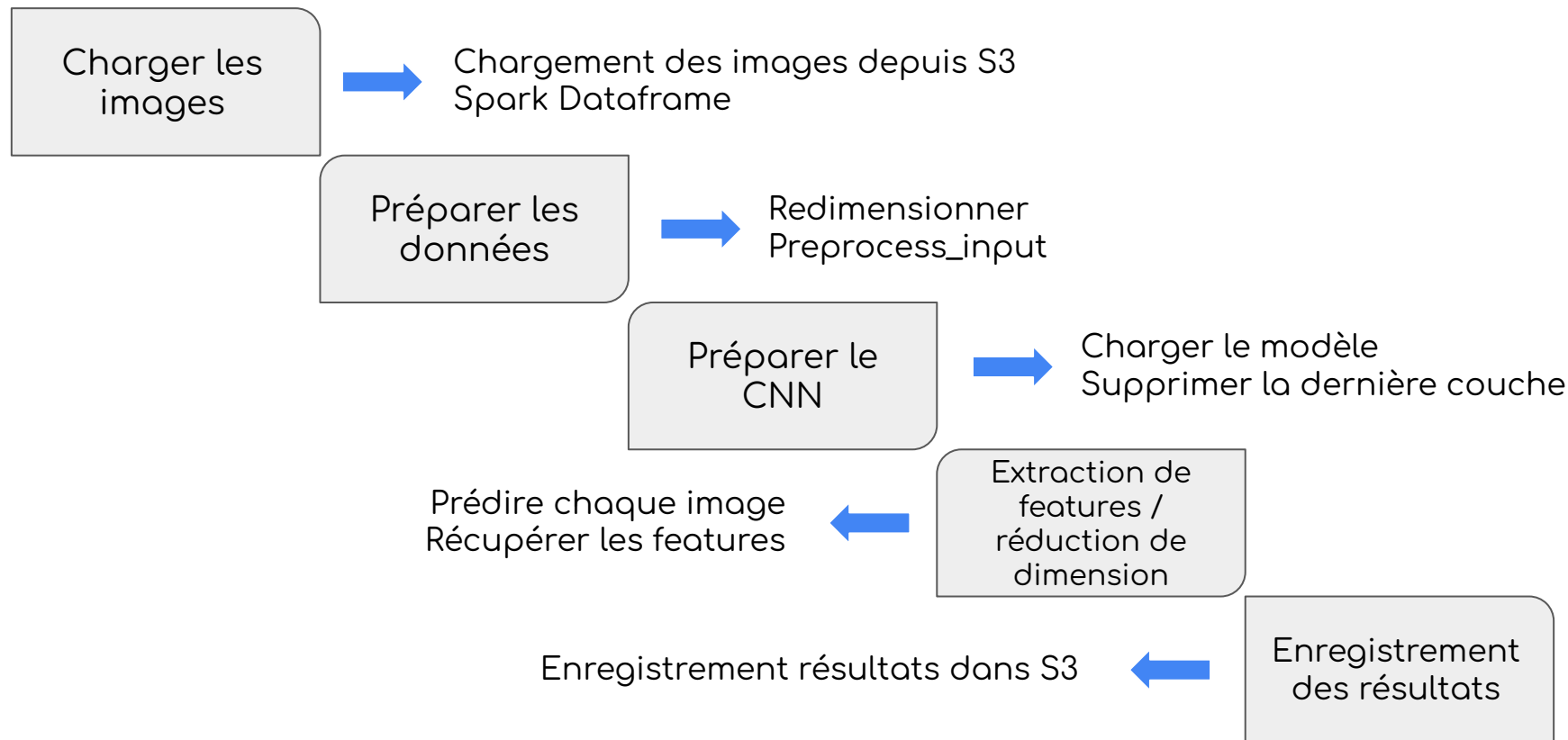
- PySpark

Librairie permettant d'utiliser le langage Apache Spark avec Python

Langage permettant d'effectuer des analyses et calculs sur de gros volumes de données, de manière distribuée

⇒ Langage de programmation pour le Big Data





Charger les
images

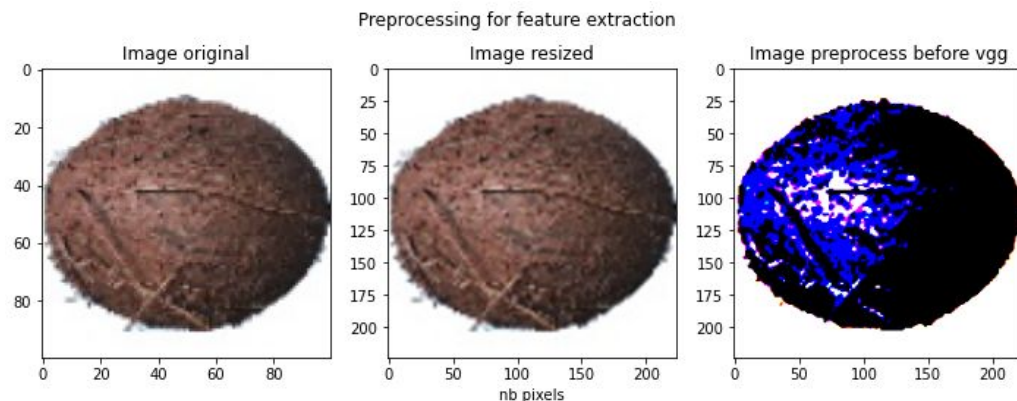
```
<class 'pyspark.sql.dataframe.DataFrame'>
root
|-- path: string (nullable = true)
|-- modificationTime: timestamp (nullable = true)
|-- length: long (nullable = true)
|-- content: binary (nullable = true)
```

path	modificationTime	length	content
s3a://sparkyfruit...	2022-11-16 10:36:06	5969	[FF D8 FF E0 00 1...
s3a://sparkyfruit...	2022-11-16 10:36:05	5768	[FF D8 FF E0 00 1...
s3a://sparkyfruit...	2022-11-16 10:36:06	5654	[FF D8 FF E0 00 1...
s3a://sparkyfruit...	2022-11-16 10:36:06	5449	[FF D8 FF E0 00 1...
s3a://sparkyfruit...	2022-11-16 10:36:04	4984	[FF D8 FF E0 00 1...

path	content	label
s3a://sparkyfruit...	[FF D8 FF E0 00 1...	Cocos
s3a://sparkyfruit...	[FF D8 FF E0 00 1...	Cocos
s3a://sparkyfruit...	[FF D8 FF E0 00 1...	Cocos

Préparer les données

```
def preprocess(content):  
    """  
    Preprocesses raw image bytes for prediction.  
    """  
    img = Image.open(io.BytesIO(content)).resize([224, 224])  
    arr = img_to_array(img)  
    return preprocess_input(arr)
```



Préparer le CNN

```
def model_fn():  
    """  
    Returns a VGG16 model with top layer removed  
    and broadcasted pretrained weights.  
    """  
    model_vgg = VGG16(weights=None, include_top=False, pooling='max')  
    model_vgg.set_weights(bc_model_weights.value)  
    return model_vgg
```

```
def featurize_series(model, content_series):  
    """  
    Featurize a pd.Series of raw images using the input model.  
    :return: a pd.Series of image features  
    """  
    input_ = np.stack(content_series.map(preprocess))  
    preds = model.predict(input_)  
    # For some layers, output features will be multi-dimensional  
    # tensors.  
    # We flatten the feature tensors to vectors for easier storage  
    # in Spark DataFrames.  
    output = [p.flatten() for p in preds]  
    return pd.Series(output)
```

Extraction de
features /
réduction de
dimension

```
@pandas_udf('array<float>', PandasUDFType.SCALAR_ITER)
def featurize_udf(content_series_iter):
    ...

    This method is a Scalar Iterator pandas UDF wrapping
    our featurization function.
    The decorator specifies that this returns a
    Spark DataFrame column of type ArrayType(FloatType).

    :param content_series_iter: This argument is an
    iterator over batches of data, where each batch
    is a pandas Series of image data.
    ...

    # With Scalar Iterator pandas UDFs, we can load
    # the model once and then re-use it
    # for multiple data batches. This amortizes the
    # overhead of loading big models.
    model = model_fn()
    for content_series in content_series_iter:
        yield featurize_series(model, content_series)
```

- User-Defined Fonction (UDF)

Permet de créer et appliquer des fonctions non préexistantes dans Spark



⇒ Pandas UDF

Prends la/les colonne(s) en entier au format pandas.Series et retourne le résultat au format pandas.Series (vs. UDF qui applique la fonction ligne par ligne)

Enregistrement
des résultats

<input type="checkbox"/>	Nom ▲	Type ▼
<input type="checkbox"/>	 results_udf.csv	csv
<input type="checkbox"/>	 results_udf/	Dossier

	A	B	C	D	E	F	G	H	I
1	,path,label,features								
2	0,s3a://sparkyfruitp8/Cocos/r_0_100.jpg,Cocos,"[6.50632019e+01	0.00000000e+00	3.40743065e+01	1.69850445e+01					
3	5.69933701e+00	3.60631065e+01	8.24240398e+00	5.85966873e+00					
4	1.22910440e+00	1.88920059e+01	7.66068935e+00	0.00000000e+00					
5	0.00000000e+00	0.00000000e+00	0.00000000e+00	3.51385808e+00					

<input type="checkbox"/>	Nom ▲	Type ▼
<input type="checkbox"/>	 _SUCCESS	-
<input type="checkbox"/>	 part-00000-1706bc51-4732-40b2-9fb6-7b89776dca91-c000.snappy.parquet	parquet

- Conclusion

- Découverte architecture Big Data et Cloud
- Utilisation des services Amazon Web Services (AWS) :
 - Création d'une instance EC2
 - Création d'un espace de stockage S3
- Réalisation des calculs distribués avec PysPark
- Réalisation de la première chaîne de traitement
 - Passage d'une image de dimension 100x100x3 (30 000 features) à 512 features (CNN)

- Suite, perspectives

- Utilisation d'un autre modèle CNN
- Ajouter seconde étape de réduction de dimension (ACP)
- Etudier les services d'AWS (AWS EMR) pour développement et expansion de l'application :
 - Instances EC2 avec plus de capacités (plus chères)
 - Stockage S3 Intelligent-Tiering