From Docker to Big Data Clusters
A new era for SQL Server

Christophe Laporte

Consultant & formateur

Conseil IT

Christophe Laporte

Audit
Conseil
Formation
Remote DBA

/conseilit

@conseilit

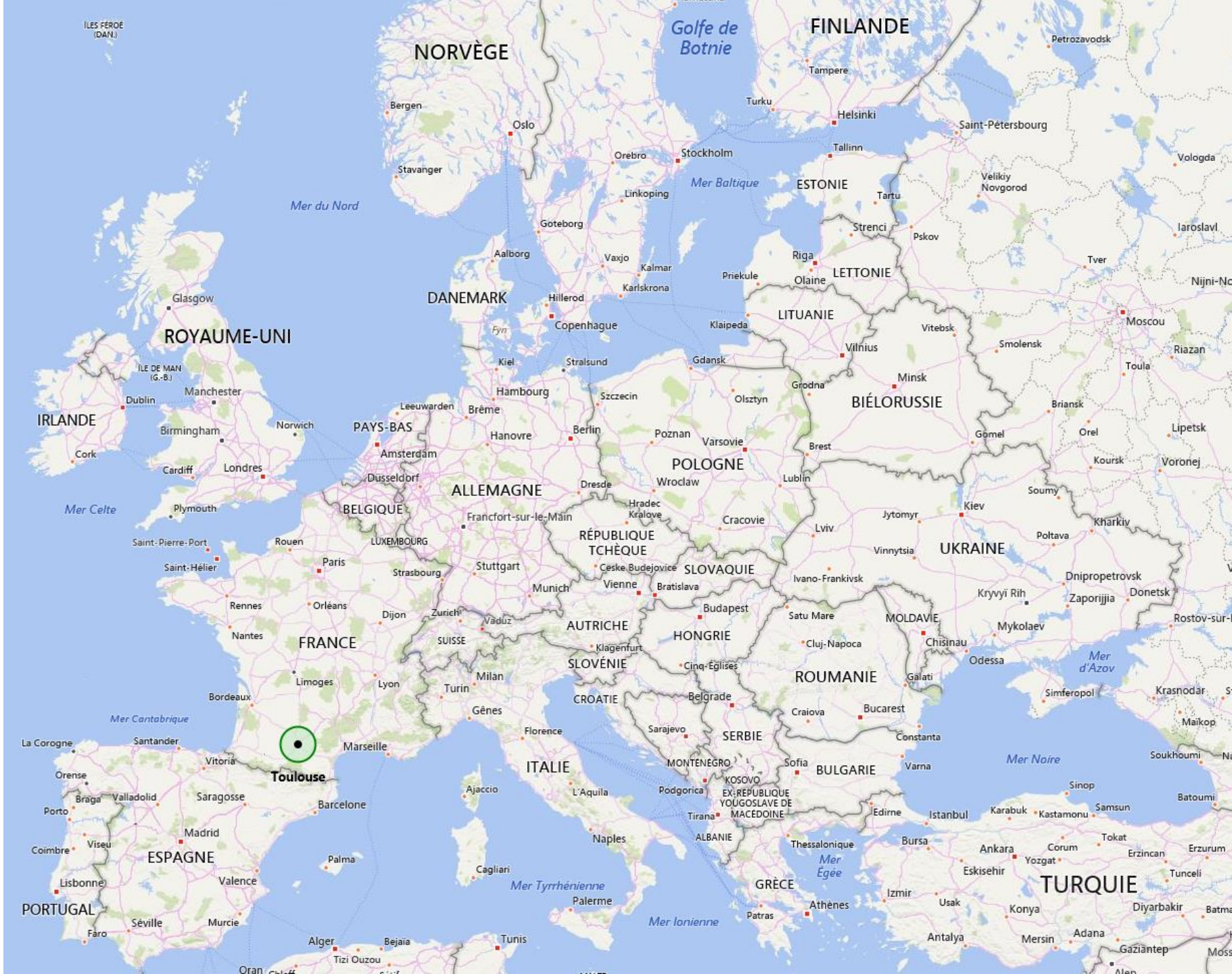
/christophelaporte

conseilit@outlook.com

Microsoft CERTIFIED Master

Microsoft CERTIFIED Trainer

~ depuis 1997 : SQL 6.5 / WinNT4



Montréal:UTC-5

Paris:UTC+1

Interventions heures non ouvrées : maintenance, migrations, dépannage …

VIRTUAL

PASS SQLSATURDAY

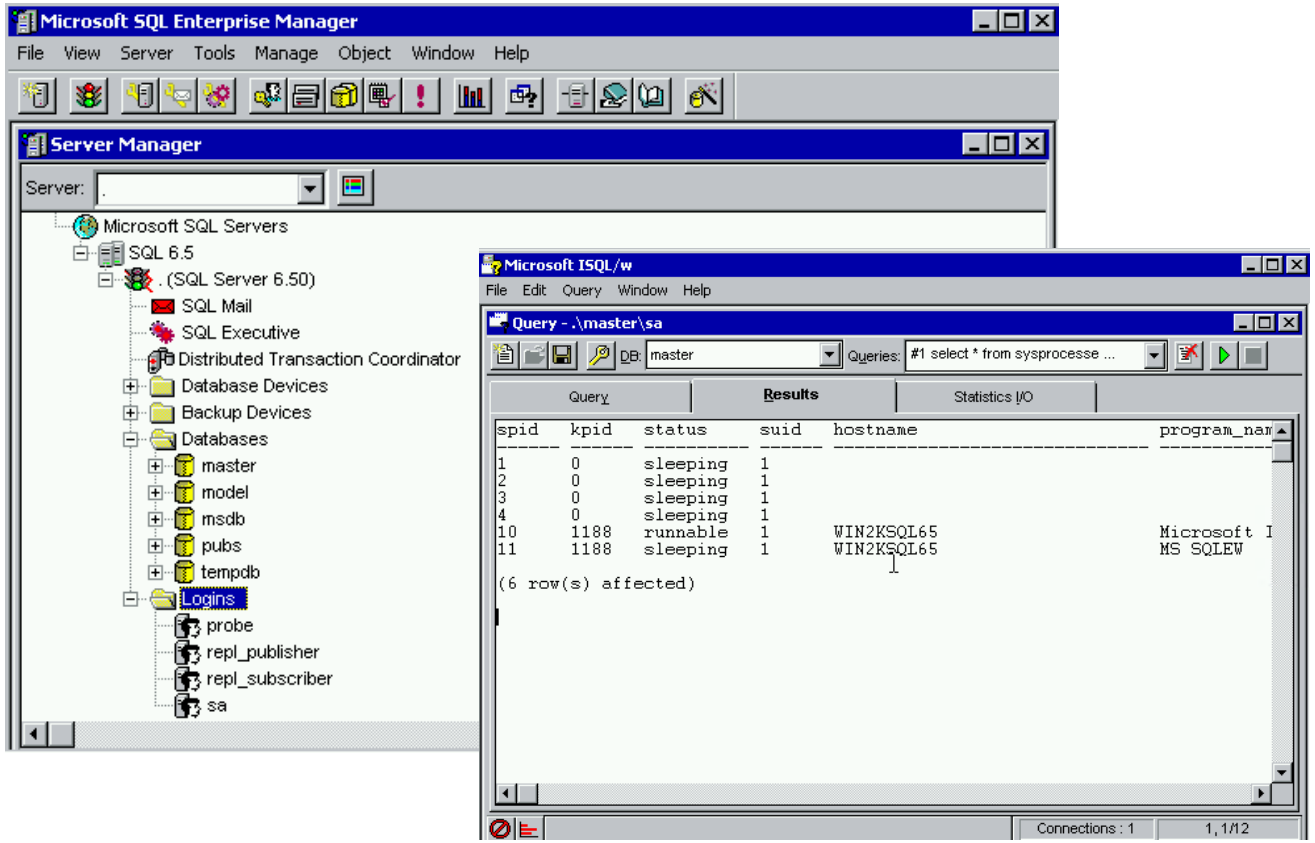
MONTREAL | NOV 21 2020

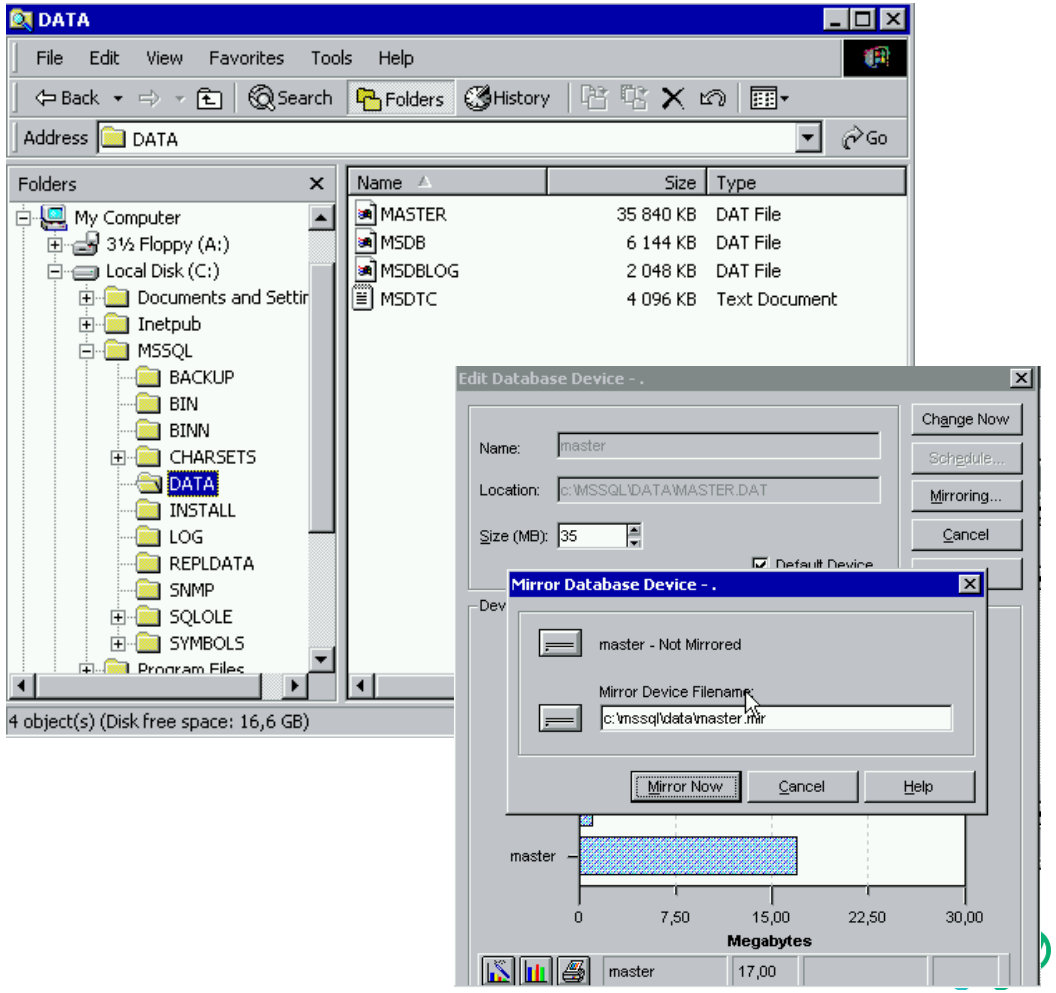
Thanks to our sponsors

Global		
Microsoft	RedHat	Google Cloud
Microsoft	Red Hat	Google Cloud

And PASS

1997 - SQL Server 6.5





SQL Server est en évolution constante

Nouvelles fonctionnalités

Temporal tables, Graph Databases

HA/DR capabilities

Amélioration de la sécurité

TDE, Always Encrypted

RLS, Dynamic Data Masking

Extensibilité

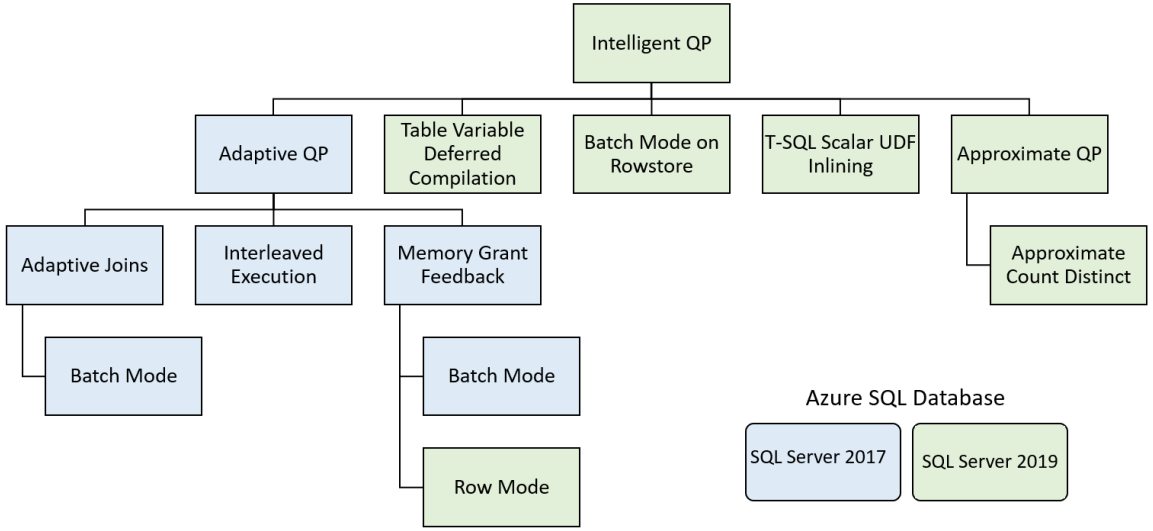
SQLCLR, Java, Python, R

Polybase

Amélioration de performance

InMemory OLTP, Columnstore index

Nouveau CE, TempDB, Intelligent QP



Un peu d'histoire …

Il y a 10 ans, se posaient les questions …

- Dois-je virtualiser SQL Server ?
- Est-ce que les performances seront bonnes ?
- Quel hyperviseur choisir ?

Aujourd'hui

- La quasi totalité des instances sont virtualisées
- Et la performance est au rendez-vous !
  - Y compris pour les charges de travail de tiers 1 …

Un peu d’histoire …

SQL Server 2017 sur Linux

Annoncé en mars 2016

Exécuter SQL Server sur un nouvel OS

Une demande de la part de clients / des ISVs

Announcing SQL Server on Linux

Mar 7, 2016 | Scott Guthrie - Executive Vice President, Cloud and Enterprise Group, Microsoft

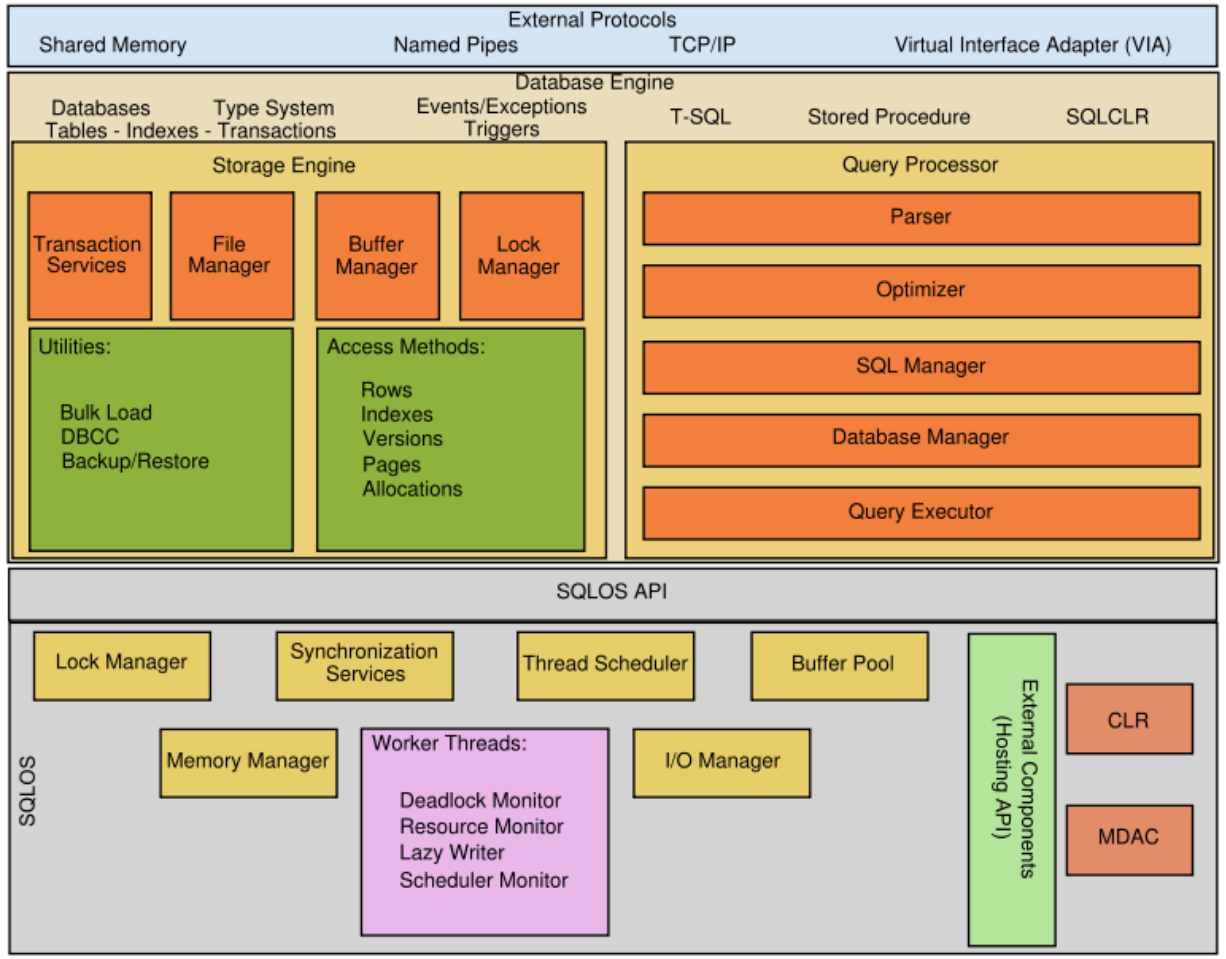
Extending SQL Server to Also Now Run on Linux

Today I’m excited to announce our plans to bring SQL Server to Linux as well. This will enable SQL Server to deliver a consistent data platform across Windows Server and Linux, as well as on-premises and cloud. We are bringing the core relational database capabilities to preview today, and are targeting availability in mid-2017.

SQL Server on Linux will provide customers with even more flexibility in their data solution. One with mission-critical performance, industry-leading TCO, best-in-class security, and hybrid cloud innovations – like Stretch Database which lets customers access their data on-premises and in the cloud whenever they want at low cost – all built in.

SQL Server sur Linux

Portage natif estimé à 5 ans



SQL Server sur Linux

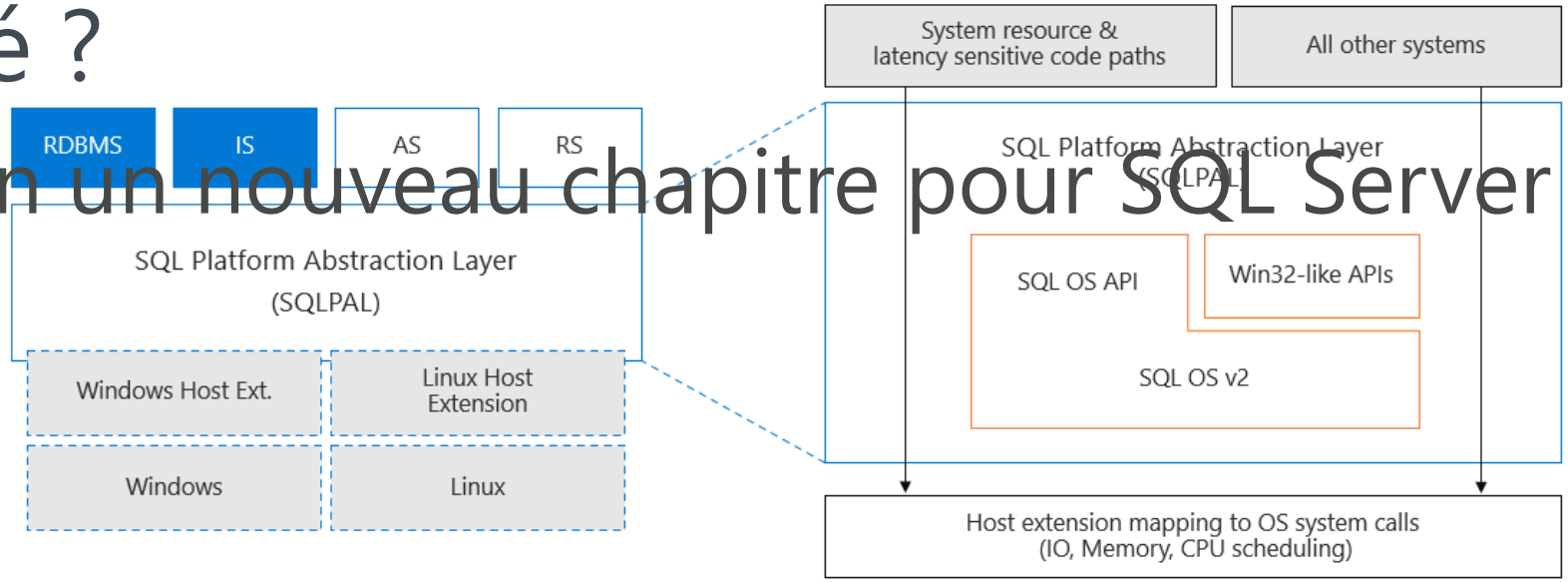
Portage natif estimé à 5 ans

3 semaines de prototypage du SQLPAL

Disponibilité publique en 24 mois

Une finalité ?

… ou bien un nouveau chapitre pour SQL Server …



SQLOS et SQLPAL

Inside SQLOS 2012

Bob Ward

Level: 500, 1/2 Day Session (3 hours)

Audience: DBA

Language: English

PASS SUMMIT 2012

Inside SQL Server 2017 on Linux

Bob Ward

Level: 500, MS Tiger Half-Day Session (2.5 hours)

Audiences: IT Pro IT Manager/Director/Executive

Language: English

PASS SUMMIT 2017

SQL Server sur Linux

Plusieurs distributions supportées

Fonctionnalités

- (StretchDB, Filetables, Filestream)

Performance

TPC-H data warehousing top results by TPC-H configuration (size)

Company	System	Performance Price/QphH	Database Operating System
Hewlett Packard Enterprise	HPE Proliant DL380 Gen10	1,244,450 QphH@3000GB 0.38 USD	Microsoft SQL Server 2017 Enterprise Edition SUSE Linux Enterprise Server 15
Hewlett Packard Enterprise	HPE Proliant DL380 Gen9	717,101 QphH@1000GB 0.61 USD	Microsoft SQL Server 2017 Enterprise Edition Red Hat Enterprise Linux Server 7.3
CISCO	Cisco UCS C460 M4 Server	1,115,298 QphH@10000GB 0.87 USD	Microsoft SQL Server 2016 Enterprise Edition Microsoft Windows Server 2016 Standard Edition

ubuntu

redhat

SUSE

SQL Server sur Linux

Installation simple

Configuration du serv

```
# ubuntu
wget -qO- https://packages.microsoft.com/keys/mi
sudo add-apt-repository "$(wget -qO- https://pac
sudo apt-get update
sudo apt-get install -y mssql-server
sudo /opt/mssql/bin/mssql-conf setup

# RedHat
sudo curl -o /etc/yum.repos.d/mssql-server.repo
sudo yum install -y mssql-server
sudo /opt/mssql/bin/mssql-conf setup


# Suse
sudo zypper addrepo -fc https://packages.microso
sudo zypper --gpg-auto-import-keys refresh
sudo zypper install -y mssql-server
sudo /opt/mssql/bin/mssql-conf setup
```

```
Preparing to unpack .../7-libc6-dbg_2.27-3ubuntu1_amd64.deb ...
Unpacking libc6-dbg:amd64 (2.27-3ubuntu1) ...
Selecting previously
Preparing to unpack
Unpacking libsss-nss
Selecting previously
Preparing to unpack
Unpacking mssql-serv
Setting up libc++abi
Setting up libcc1-0:
Setting up libc6-dbg
Setting up libsss-ns
Setting up gdbserver
Setting up libsasl2-
Setting up libbabelt
Setting up libc++1:a
Setting up gdb (8.1-
Setting up mssql-ser

+------------------
Please run 'sudo /op
to complete the setu
+------------------

Processing triggers
Processing triggers
Christophe@lxSQL-vm:
```

```
Christophe@lxSQL-vm:~$ sudo /opt/mssql/bin/mssql-conf setup
usermod: no changes
Choose an edition of SQL Server:
  1) Evaluation (free, no production use rights, 180-day limit)
  2) Developer (free, no production use rights)
  3) Express (free)
  4) Web (PAID)
  5) Standard (PAID)
  6) Enterprise (PAID) - CPU Core utilization restricted to 20 physical/40 hyperthreaded
  7) Enterprise Core (PAID) - CPU Core utilization up to Operating System Maximum
  8) I bought a license through a retail sales channel and have a product key to enter.

Details about editions can be found at
https://go.microsoft.com/fwlink/?LinkId=2109348&clcid=0x409

Use of PAID editions of this software requires separate licensing through a
Microsoft Volume Licensing program.
By choosing a PAID edition, you are verifying that you have the appropriate
number of licenses in place to install and run this software.

Enter your edition(1-8): 2
The license terms for this product can be found in
/usr/share/doc/mssql-server or downloaded from:
https://go.microsoft.com/fwlink/?LinkId=2104294&clcid=0x409

The privacy statement can be viewed at:
https://go.microsoft.com/fwlink/?LinkId=853010&clcid=0x409

Do you accept the license terms? [Yes/No]:Yes

Enter the SQL Server system administrator password:
Confirm the SQL Server system administrator password:
Configuring SQL Server...

ForceFlush is enabled for this instance.
ForceFlush feature is enabled for log durability.
Created symlink /etc/systemd/system/multi-user.target.wants/mssql-server.service → /lib/systemd/system/mssql-server.service.
Setup has completed successfully. SQL Server is now starting.
Christophe@lxSQL-vm:~$
```

Modèle de conception de micro-services

Hier : applications monolithiques

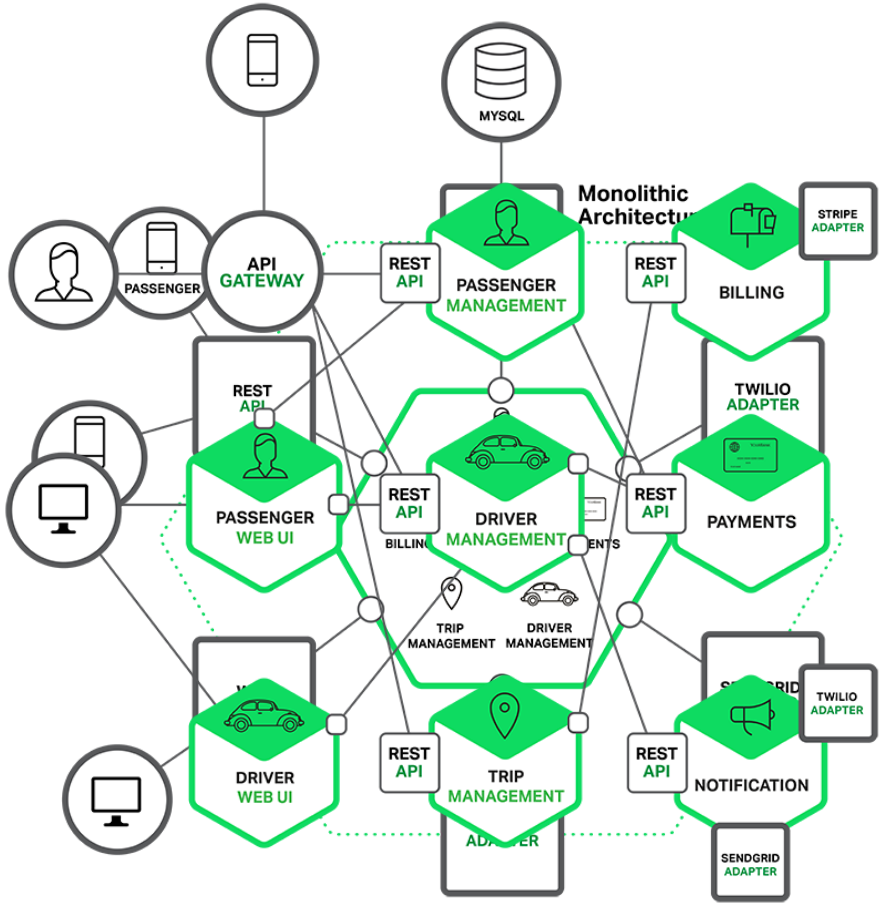
- Difficile à entretenir / évoluer

Aujourd'hui, place aux micro-services

- Nouvelle façon de développer des applications
- Eléments applicatifs pouvant évoluer indépendamment
- 1, 10 ou des centaines de conteneurs <-> application

Cela semble devenir une norme

- D'un point de vue infrastructure
- Totalement dans la « philosophie » DevOps



Introduction aux conteneurs

Virtualisation 1.0

Virtualisation matérielle (Hyper-V, VMware, …)

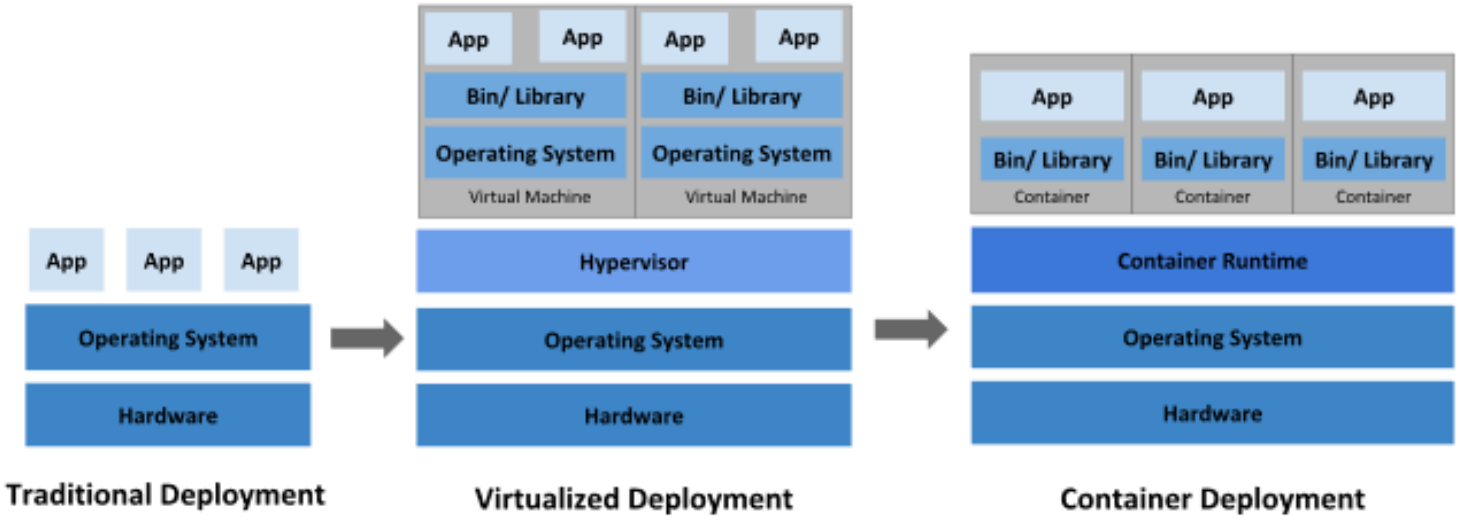
Virtualisation 2.0

Virtualisation d'OS, appelée containerisation

Terminologie

Image

Conteneur



Virtualisation 2.0 : containerisation

Empreinte système réduite

- léger -> meilleure efficacité des serveurs hôtes

Une seule image

- Déploiements multiples (dev / test / prod)
- Eviter : “Cela fonctionne sur ma machine” !

Isolation

- Le conteneur n’est pas visible depuis le réseau
- Le conteneur a un fonctionnement sans état

Fonctionne de la même manière

- Quel que soit l’environnement cible

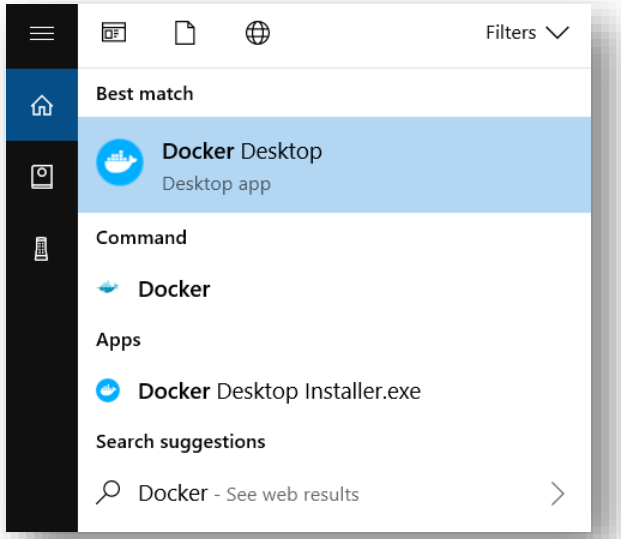
Déploiement rapide

Multi OS

- Windows, Linux, Mac
- Ligne de commande identique
- Comportement identique

Premiers pas …

- Docker Desktop for Windows 10
- Linux




```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
sudo apt-get update
apt-cache policy docker-ce
sudo apt-get install -y docker-ce
```

docker

Docker engine

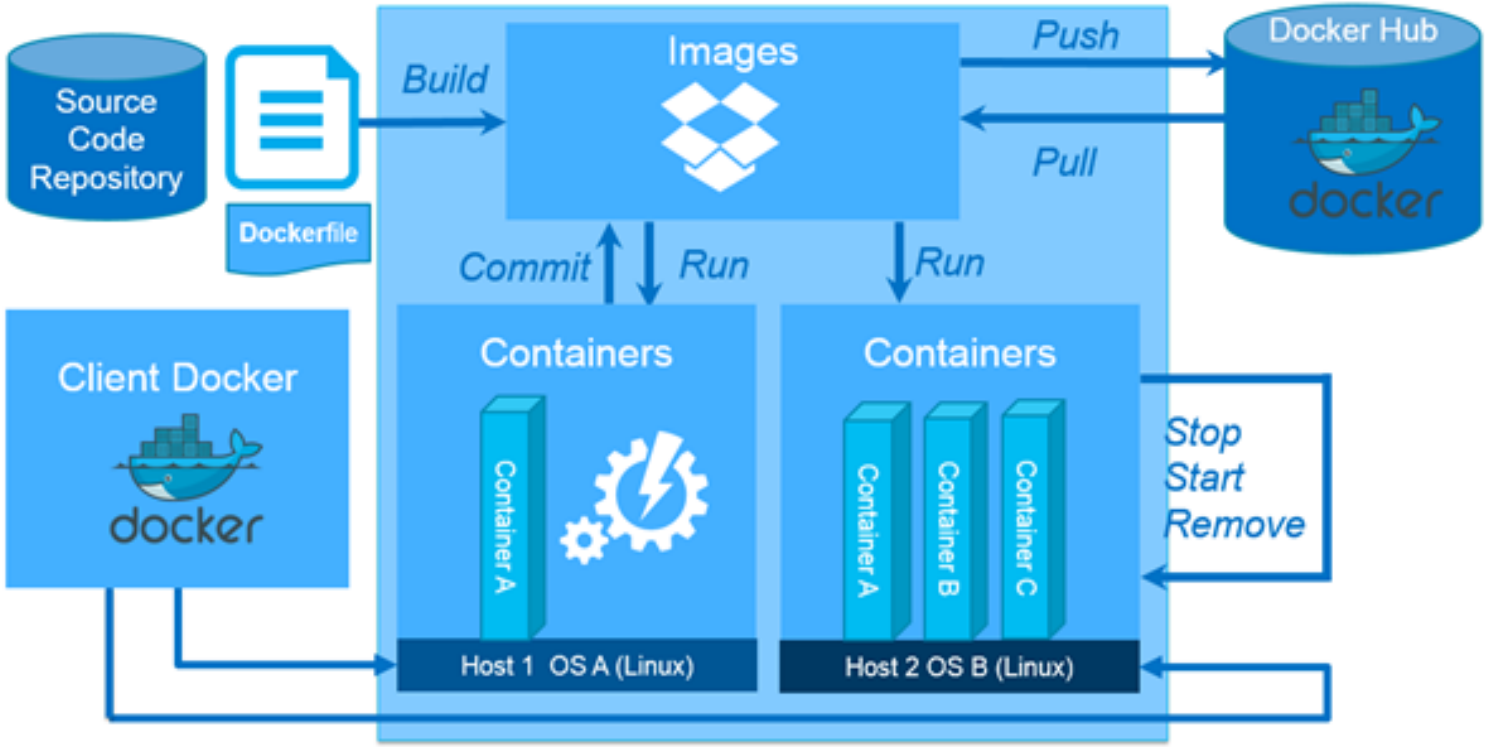
Exécution des containers

Docker client

Ligne de commande

Produits alternatifs

Podman
Mesos Containerizer
LXC
Rkt (prononcer « Rocket »)



Docker – utilitaire en ligne de commande

Command	Description
Docker search	Find an image on a repository
Docker pull	Download an image from the repository
Docker build	Create an image from a Dockerfile
Docker create	Create a container
Docker start	Start a container
Docker run	All-in-one command to pull, create and start a container
Docker stop	Stop a container
Docker rm	Remove the container – but not the image (Docker RMI)

Mon premier conteneur


```
# Survival kit : Docker commands
docker

## Display Docker version and info
docker version
docker info

## Docker images CLI commands
docker image --help
docker image ls  # <=>  docker images

## Docker container CLI commands
docker container --help
docker container ls       #  <=>  docker ps
docker container ls --all #  <=>  docker ps -a

# Running my first container
docker run hello-world
```

```
root@lxDocker:/home/chris# docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
1b930d010525: Pull complete
Digest: sha256:9572f7cdcee8591948c2963463447a53466950b3fc15a247fcad1917ca215a2f
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

Minute papillon….

Il est possible d'exécuter SQL Server sous Linux

- Un nouvel OS, mais pas vraiment un but ultime …
- Le début d'un nouveau chapitre

Pourquoi ne pas exécuter SQL Server dans un container ?

- Exécuter SQL Server sous linux était donc une étape indispensable

Quelques points d'attention cependant

- Redirection de ports TCP
- Redirection de stockage pour assurer la persistance des données

SQL Server inside a container


```
# Run (Pull+Create+Start) the container in detach mode
docker run  --detach \                                  # Run (Pull+Create+Start) the container in detach mode
            --name sqldocker \                          # Container name
            --hostname sqldocker \                      # OS name
            --env 'MSSQL_PID=developer' \               # Edition : developer is the default value
            --env 'SA_PASSWORD=Password1!' \            # Password for SA account
            --env 'ACCEPT_EULA=Y' \                     # You still need to acknowledge licence terms
            --volume /mssql:/var/opt/mssql/data \       # Redirect storage to persist data
            --publish 1433:1433 \                       # TCP endpoint to connect the container
            mcr.microsoft.com/mssql/server:2019-latest  # Image used to build and start the container
```

Et maintenant ?

Comme une impression de déjà vu … Hey, c'est un cluster !

Il faudrait ajouter un peu d'orchestration

- Vérifier la santé du conteneur --> restart container
- Vérifier la santé du host --> Restart sur host différent
- Fournir un accès au conteneur depuis le réseau
- Fournir un stockage persistant accessible par tous les nœuds
- Gérer les ressources des conteneurs (CPU, RAM …)

Voire même des fonctionnalités de scaling …

Et si possible fournir une expérience de déploiement similaire

- OnPrem
- Cloud Public

Kubernetes pour les DBAs : les bases

Aussi connu sous le nom K8s

Master Node

- Orchestration des conteneurs

Worker Node

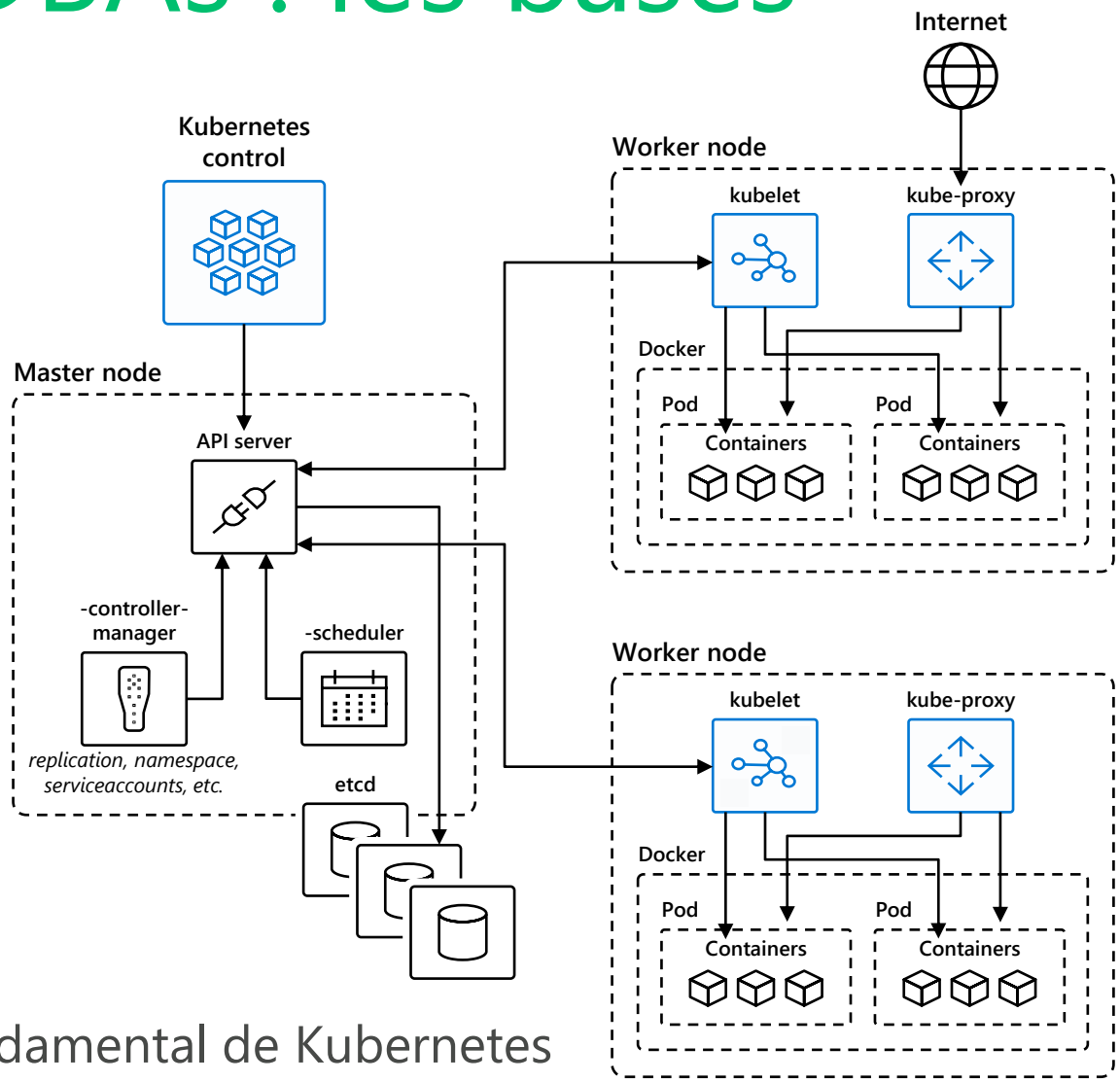
- Kubelet : responsable de l'exécution
- Kube-proxy : gestion du trafic réseau

Pod

- Unité élémentaire d'exécution
- Constitué de 1..N containers
- Dispose d'une @IP unique

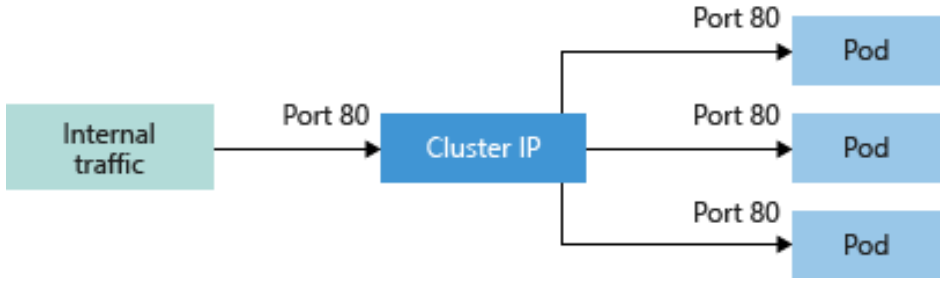
Haute disponibilité

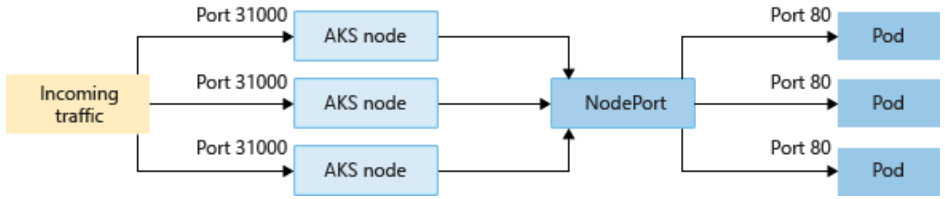
- « par défaut »
- Desired state Configuration : un concept fondamental de Kubernetes

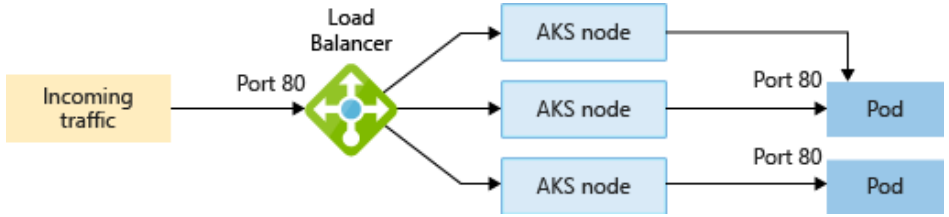


Kubernetes pour les DBAs : les bases

- Les connections passent par le kube-proxy
  - Routage et translation de port vers le Pod
  - Quel que soit de worker node
- Services
  - Exposent des applications
  - Abstraction logique d'un ou plusieurs Pods
- Diffèrent types de services
  - ClusterIP
  - Node Port
  - Load Balancer







Kubectl : Utilitaire en ligne de commande

Command	Description
kubectl create|apply -f ./somefile.yaml	Resource creation
kubectl delete -f ./somefile.yaml	Resource deletion
kubectl run nginx –image=nginx	Run a single instance from Nginx image
Kubectl get pods	List Pods
kubectl get service(s)	List Services
kubectl get deployment(s)	List Deployments
kubectl get node(s)	List Nodes of the cluster
kubectl logs <pod-name>	Display container / pod logs
kubectl exec -it <pod-name> — bash	Run a command inside a container

Un peu d’histoire (encore)

Il y a 10 se posait la question

- Dois-je virtualiser SQL Server ?
- Aujourd’hui, la quasi totalité des instances SQL Server sont virtualisées

Il y a 10 ans, Microsoft levait le voile sur … Azure

Microsoft Cloud Services Vision Becomes Reality With Launch of Windows Azure Platform

November 17, 2009 |

LOS ANGELES — Nov. 17, 2009 — Microsoft Corp. today announced the availability of the Windows Azure platform at the Microsoft Professional Developers Conference (PDC). In his opening keynote address, Ray Ozzie, chief software architect at Microsoft, described Windows Azure and SQL Azure as core elements of the company’s cloud services strategy. The company also announced a set of new Windows Azure

SQL Server sur Azure

Machines virtuelles

Recommandé pour des migrations rapides et pour les applications requérant un accès au niveau OS

Instances managées

Migration facilité vers une instance gérée

Azure SQL Databases

Recommandé pour les applications modernes possédant plusieurs niveaux de services

SQL Server containers on Azure

Azure Containers Instances

Azure Kubernetes Service

Services de conteneurs sur Azure : ACI

Azure Container Instance

Moyen simple d'exécuter un conteneur, pas de cluster Kubernetes à gérer

Facturation du stockage et de la consommation CPU à la seconde

```
# create a resource group
az group create --name sqlserver-aci --location westeurope

# and create a container inside ACI
az container create --resource-group sqlserver-aci  \
                    --name mssqlaci \
                    --image mcr.microsoft.com/mssql/server:2019-latest \
                    --ip-address public --ports 1433 \
                    --environment-variables ACCEPT_EULA=Y MSSQL_SA_PASSWORD=P@ssw0rd1! \
                    --dns-name-label conseilit-sqlserver-aci \
                    --cpu 4 --memory 16

# and finally connect to the SQL Server instance
/opt/mssql-tools/bin/sqlcmd -S conseilit-sqlserver-aci.westeurope.azurecontainer.io \
                            -U SA -P 'P@ssw0rd1!' -Q "SELECT name from sys.databases;"
```

Cas d'usage

Scénarii DevOps (exécution du conteneur, test et destruction)

Ne convient pas à des traitements de longue durée

Services de conteneurs sur Azure : AKS

Cluster Kubernetes intégralement géré

- Stockage
- Réseau
- Configuration K8s configuration (master, worker nodes)

Vous devez

- créer le cluster
- maintenir le cluster

Facturation pour la consommation

- des machines virtuelles
- du stockage
- des ressources réseau

```
# Create a resource group
az group create --name k8s-group --location francecentral

# List currently supproted Kubernetes version
az aks get-versions --location francecentral --output table

# Create the cluster
az aks create --name k8s-cluster \
--resource-group k8s-group \
--generate-ssh-keys \
--node-vm-size Standard_B8ms \
--node-count 3 \
--kubernetes-version 1.14.7

# Get Nodes and Pods
kubectl get nodes -o wide
kubectl get pods -o wide --all-namespaces
```

Services de conteneurs sur Azure : AKS

Parfaitement adapté aux environnements de production

Haute disponibilité … par défaut !

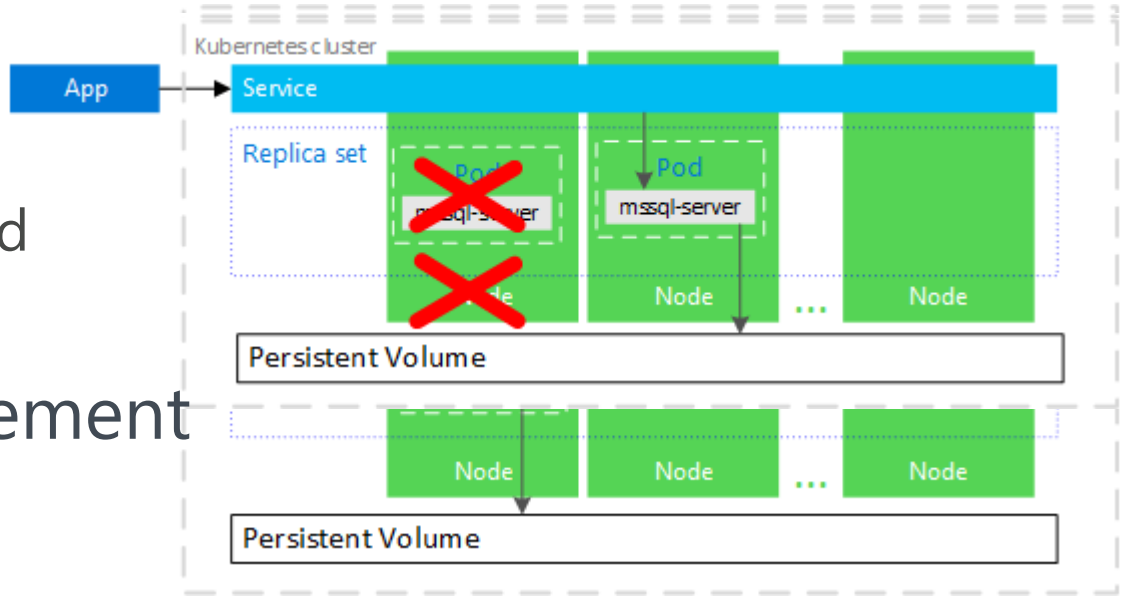
- Stockage persistant transversal aux nœuds du cluster
- K8s est responsable du maintient de la configuration désirée

Basculement automatique

- Quand une panne survient
- K8s essaie un redémarrage sur le même nœud
- Ou bien sur un nœud différent

Applications se reconnectent immédiatement

- Même Service => adresse IP / port identique
- K8s va rediriger le trafic vers le nouveau pod



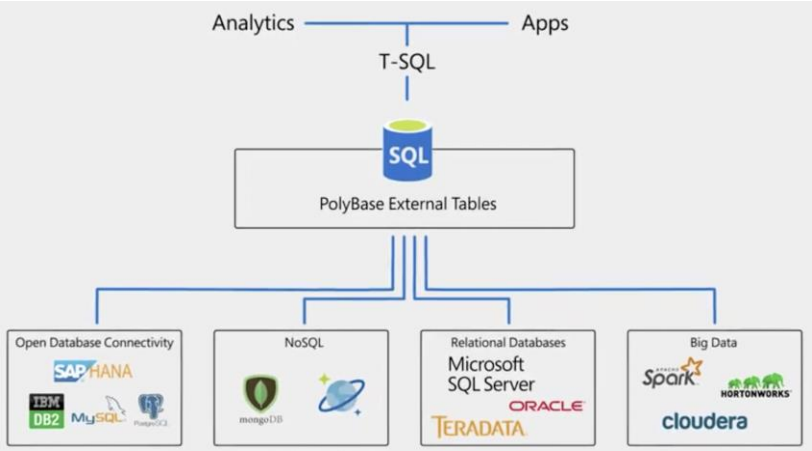
SQL Server sur AKS

```
# Create a dedicated Namespace
kubectl create namespace sqlsaturday
kubectl get namespaces

# refresh every 2 seconds the resources created
watch kubectl get all --namespace sqlsaturday

# Create a secret to be used by SQL Server deployment
kubectl create secret generic mssql \
            --from-literal=SA_PASSWORD="MyC0m9l&xP@ssw0rd"  \
            --namespace sqlsaturday

# Deploy a SQL Server Pod with a single YAML file containing
#  - Storage Class
#  - Persistent Volume Claim
#  - Deployment
#  - Service
cat AKS-SQLServer-AllinOne.yaml
kubectl apply -f AKS-SQLServer-AllinOne.yaml --namespace sqlsaturday
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mssql-deployment
  labels:
    app: mssql
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mssql
  template:
    metadata:
      labels:
        app: mssql
    spec:
      terminationGracePeriodSeconds: 10
      hostname: mssqlinst1
      securityContext:
        fsGroup: 1000
      containers:
      - name: mssql
        image: mcr.microsoft.com/mssql/server:2019-CU8-ubuntu-18.04
        ports:
        - containerPort: 1433
        env:
        - name: MSSQL_PID
          value: "Developer"
        - name: ACCEPT_EULA
          value: "Y"
        - name: MSSQL_AGENT_ENABLED
          value: "true"
        - name: MSSQL_SA_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mssql
              key: SA_PASSWORD
        volumeMounts:
        - name: mssqldb
          mountPath: /var/opt/mssql
      volumes:
      - name: mssqldb
        persistentVolumeClaim:
          claimName: mssql-data
```

Always On Availability Groups dans K8s

Mais …



K8s peut exécuter tout type d'application

K8s peut exécuter SQL Server

Aujourd'hui SQL Server est plus qu'un SGBD

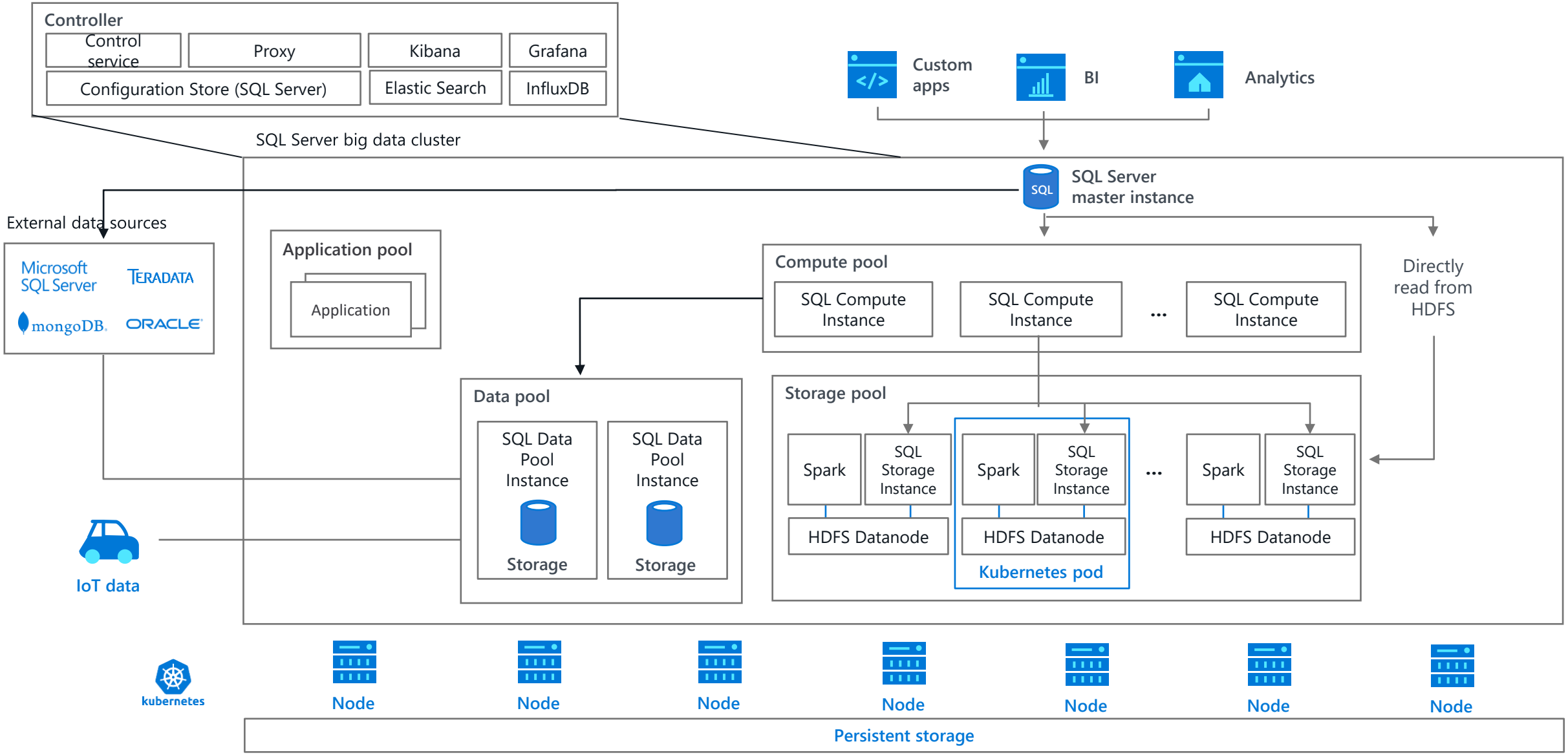
- SQL Server propose la virtualisation de données avec Polybase

Un Pod peut héberger plusieurs conteneurs

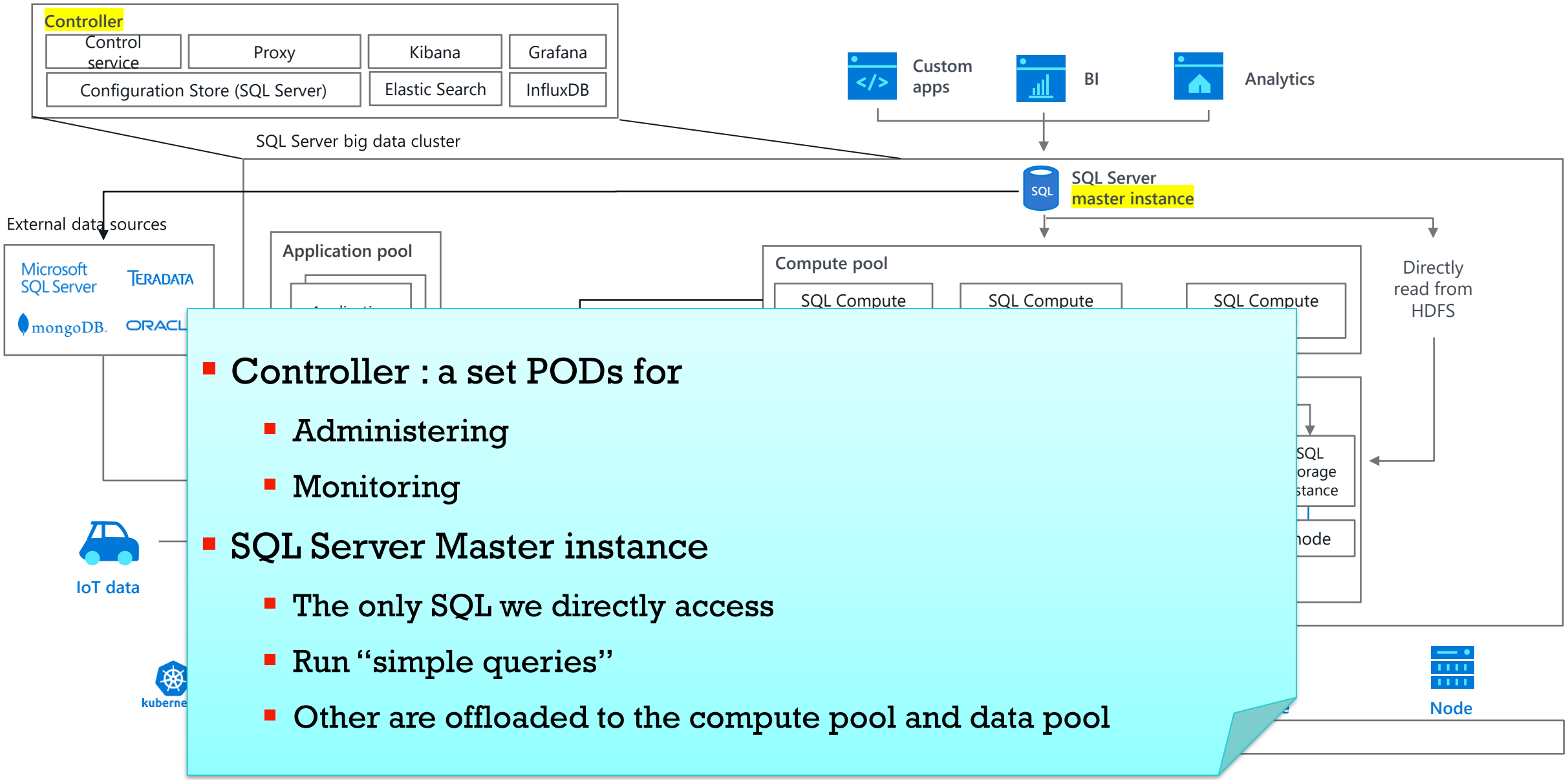
Ajoutons des conteneurs « Big Data»

- Avec un moteur Spark
- Et un stockage HDFS

SQL Server 2019 Big Data Cluster



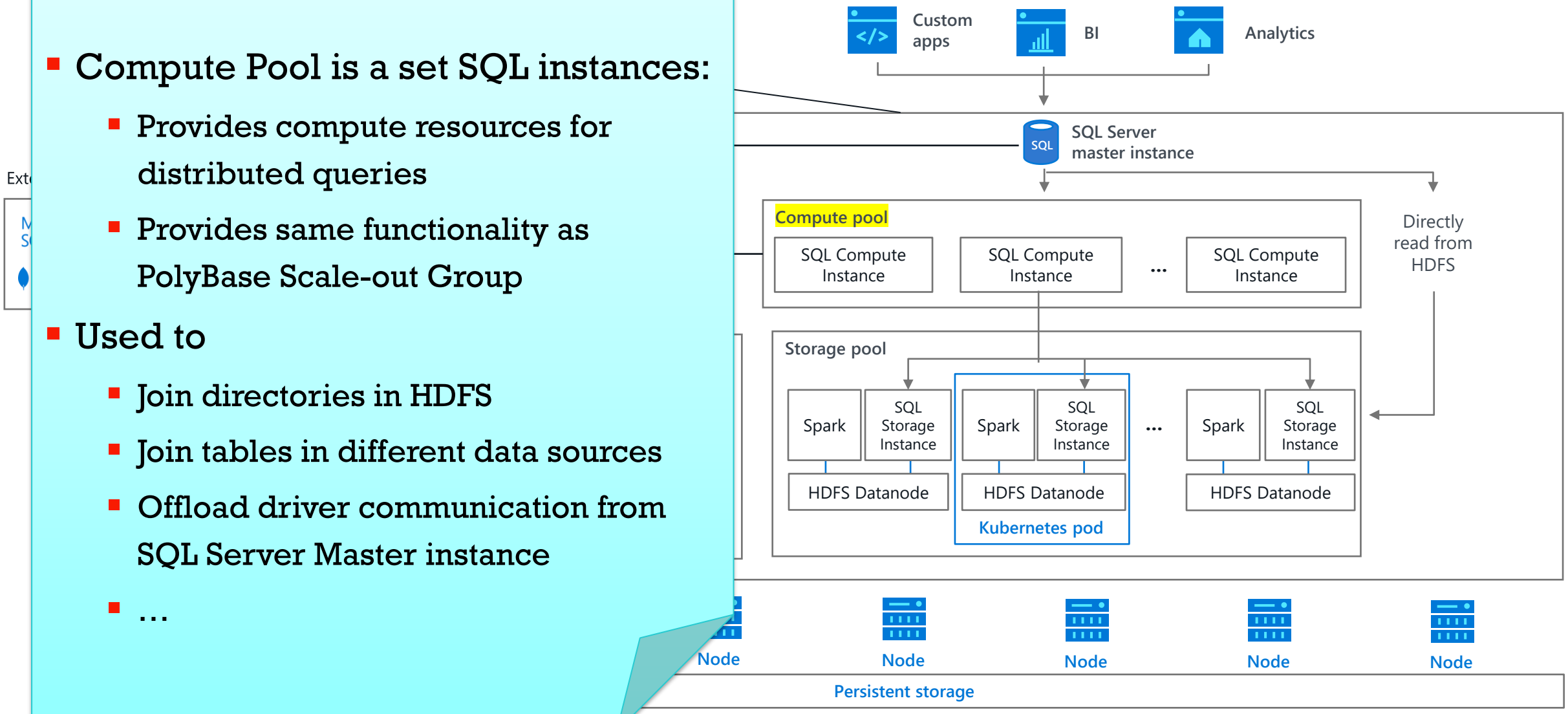
Control plane



- Controller : a set PODs for
  - Administering
  - Monitoring
- SQL Server Master instance
  - The only SQL we directly access
  - Run "simple queries"
  - Other are offloaded to the compute pool and data pool

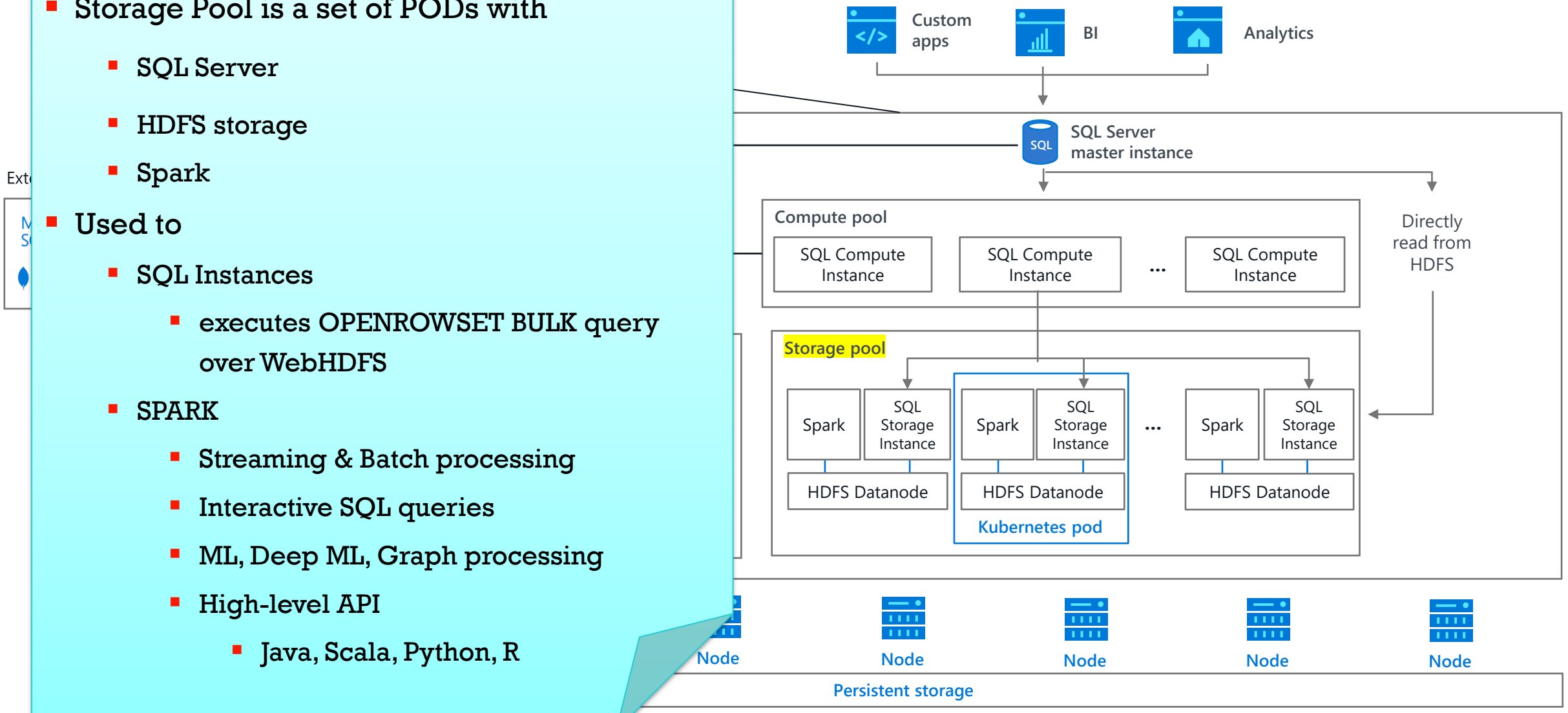
Compute plane

- Compute Pool is a set SQL instances:
  - Provides compute resources for distributed queries
  - Provides same functionality as PolyBase Scale-out Group
- Used to
  - Join directories in HDFS
  - Join tables in different data sources
  - Offload driver communication from SQL Server Master instance
  - …

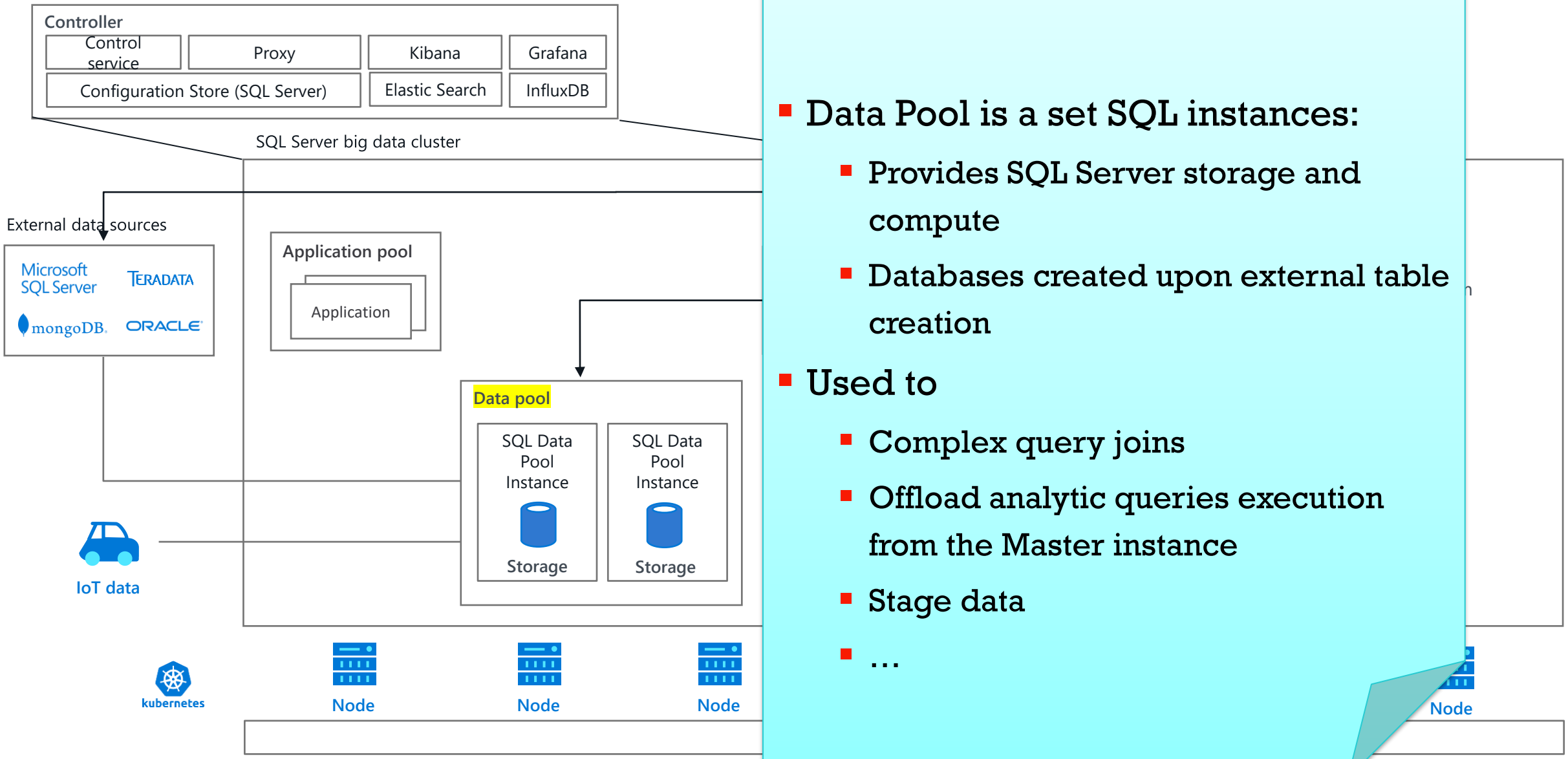


Data plane : Storage pool

- Storage Pool is a set of PODs with
  - SQL Server
  - HDFS storage
  - Spark
- Used to
  - SQL Instances
    - executes OPENROWSET BULK query over WebHDFS
  - SPARK
    - Streaming & Batch processing
    - Interactive SQL queries
    - ML, Deep ML, Graph processing
    - High-level API
      - Java, Scala, Python, R



Data plane : Data pool



- Data Pool is a set SQL instances:
  - Provides SQL Server storage and compute
  - Databases created upon external table creation
- Used to
  - Complex query joins
  - Offload analytic queries execution from the Master instance
  - Stage data
  - …

SQL Server 2019 BDC

[7]

```
USE DemoDB;
GO
SELECT TOP 10 * FROM [dbo].[WxLog]
```

Commands completed successfully.

(10 rows affected)

Total execution time: 00:00:13.527

	Date	Time	Baro	QNH	Gust Speed	Gust
1	11/05/2013	13:25	1024.00	1024.00	16.92	270
2	11/05/2013	13:26	1024.00	1024.00	16.92	248
3	11/05/2013	13:27	1024.00	1024.00	16.20	248
4	11/05/2013	13:28	1024.00	1024.00	16.92	293
5	11/05/2013	13:29	1024.00	1024.00	9.36	248
6	11/05/2013	13:30	1024.00	1024.00	14.04	293
7	11/05/2013	13:31	1024.00	1024.00	9.72	293
8	11/05/2013	13:32	1024.00	1024.00	12.60	293
9	11/05/2013	13:33	1024.00	1024.00	12.24	293
10	11/05/2013	13:34	1024.00	1024.00	12.60	270

[3]

```
# Read the CSV file(s) into a spark dataframe and print schema
results = spark.read \
    .option("inferSchema", "true") \
    .csv('/csvfiles/temperature-last-year_poolhouse.csv') \
    .toDF("DateTime","Humidity","Temperature","Temperature_range (low)","Temperature_range (high)")
results.printSchema()
```

Starting Spark application

ID	YARN Application ID	Kind	State	Spark UI	Driver log	Current session?
5	application_1573997616589_0001	pyspark	idle	Link	Link	✓

SparkSession available as 'spark'.

```
root
 |-- DateTime: string (nullable = true)
 |-- Humidity: string (nullable = true)
 |-- Temperature: string (nullable = true)
 |-- Temperature_range (low): string (nullable = true)
 |-- Temperature_range (high): string (nullable = true)
```

[11]

```
results.show(5)
```

```
+-------------------+--------+-----------+----------------------+-----------------------+
|           DateTime|Humidity|Temperature|Temperature_range (low)|Temperature_range (high)|
+-------------------+--------+-----------+----------------------+-----------------------+
|           DateTime|Humidity|Temperature|   Temperature_range...|    Temperature_range...|
|2018-05-14 00:00:00|      80|      10.06|                   8.8|                   11.2|
|2018-05-15 00:00:00|      88|      11.83|                  10.5|                   13.6|
|2018-05-16 00:00:00|      83|      13.47|                  11.7|                   16.6|
|2018-05-17 00:00:00|      84|      14.69|                  12.9|                   18.1|
+-------------------+--------+-----------+----------------------+-----------------------+
only showing top 5 rows
```

[12]
```
results.filter("Humidity > 70").show()
```

DateTime	Humidity	Temperature	Temperature_range (low)	Temperature_range (high)
2018-05-14 00:00:00	80	10.06		
2018-05-15 00:00:00	88	11.83		
2018-05-16 00:00:00	83	13.47		
2018-05-17 00:00:00	84	14.69		
2018-05-18 00:00:00	82	15.91		
2018-05-19 00:00:00	76	17.69		
2018-05-28 00:00:00	82	18.27		
2018-05-29 00:00:00	82	19		
2018-05-30 00:00:00	84	18.23		
2018-05-31 00:00:00	79	18.97		
2018-06-03 00:00:00	73	20.36		
2018-06-04 00:00:00	73	20.77		
2018-06-05 00:00:00	79	19.86		
2018-06-06 00:00:00	80	18.21		
2018-06-07 00:00:00	73	19.86		
2018-06-10 00:00:00	76	21.46		
2018-06-11 00:00:00	79	19.29		
2018-06-12 00:00:00	76	18.69		
2018-06-15 00:00:00	71	19.54		
2018-06-18 00:00:00	72	19.03		

only showing top 20 rows

[13]
```
results.filter("Humidity > 70").filter("Temperature > 15").show()
```

DateTime	Humidity	Temperature	Temperature_range (low)	Temperature_range (high)
2018-05-19 00:00:00	76	17.69	13.9	21.6
2018-05-28 00:00:00	82	18.27	16.4	19.8
2018-05-29 00:00:00	82	19	16.4	22.2
2018-05-30 00:00:00	84	18.23	16.2	20.6
2018-05-31 00:00:00	79	18.97	15.7	23.4
2018-06-03 00:00:00	73	20.36	15.9	25.2
2018-06-04 00:00:00	73	20.77	15.5	26
2018-06-05 00:00:00	79	19.86	16.8	25.7
2018-06-06 00:00:00	80	18.21	15.5	22.3
2018-06-07 00:00:00	73	19.86	16	24.7
2018-06-10 00:00:00	76	21.46	18.7	25
2018-06-11 00:00:00	79	19.29	16.8	23.6
2018-06-12 00:00:00	76	18.69	14.1	32.9
2018-06-15 00:00:00	71	19.54	16.3	23.8
2018-06-18 00:00:00	72	19.03	14.7	23.5
2018-09-06 00:00:00	73	20.93	19.4	23.8
2018-10-09 00:00:00	82	16.55	13.7	20.7
2018-10-10 00:00:00	77	18.77	16.4	22.6
2018-10-11 00:00:00	74	19	14.2	23.5
2018-10-12 00:00:00	74	20.63	16.5	25.2

only showing top 20 rows

We can switch a TSQL like syntax to query the dataframe

```
[18]  1 results.select("temperature","Humidity").show(10)
```

```
+-----------+--------+
|temperature|Humidity|
+-----------+--------+
|Temperature|Humidity|
|      10.06|      80|
|      11.83|      88|
|      13.47|      83|
|      14.69|      84|
|      15.91|      82|
|      17.69|      76|
|      19.07|      67|
|      19.26|      65|
|      19.31|      69|
+-----------+--------+
only showing top 10 rows
```

We can also use some real TSQL statements.

Let's creae a kind of view ands make some queries

```
[20]  1 results.createOrReplaceTempView("meteo")
```

```
[21]  1 spark.sql("SELECT * from meteo").show(10)
```

```
+-------------------+--------+-----------+----------------------+-----------------------+
|           DateTime|Humidity|Temperature|Temperature_range (low)|Temperature_range (high)|
+-------------------+--------+-----------+----------------------+-----------------------+
|           DateTime|Humidity|Temperature|   Temperature_range...|    Temperature_range...|
|2018-05-14 00:00:00|      80|      10.06|                   8.8|                   11.2|
|2018-05-15 00:00:00|      88|      11.83|                  10.5|                   13.6|
|2018-05-16 00:00:00|      83|      13.47|                  11.7|                   16.6|
|2018-05-17 00:00:00|      84|      14.69|                  12.9|                   18.1|
|2018-05-18 00:00:00|      82|      15.91|                  11.1|                   20.8|
|2018-05-19 00:00:00|      76|      17.69|                  13.9|                   21.6|
```

[23]
```
spark.sql("SELECT MIN(Temperature),MAX(Temperature),AVG(Temperature) from meteo").show()
```

```
+----------------+----------------+--------------------------------+
|min(Temperature)|max(Temperature)|avg(CAST(Temperature AS DOUBLE))|
+----------------+----------------+--------------------------------+
|           -0.09|     Temperature|              15.283779680952737|
+----------------+----------------+--------------------------------+
```

[24]
```
spark.sql("SELECT DateTime,Temperature,LEAD(Temperature) OVER (order by DateTime) as NextValue,avg(Temperat
```

```
+-------------------+-----------+---------+------------------+
|           DateTime|Temperature|NextValue|           avgTemp|
+-------------------+-----------+---------+------------------+
|2018-05-14 00:00:00|      10.06|    11.83|15.283779680952737|
|2018-05-15 00:00:00|      11.83|    13.47|15.283779680952737|
|2018-05-16 00:00:00|      13.47|    14.69|15.283779680952737|
|2018-05-17 00:00:00|      14.69|    15.91|15.283779680952737|
|2018-05-18 00:00:00|      15.91|    17.69|15.283779680952737|
|2018-05-19 00:00:00|      17.69|    19.07|15.283779680952737|
|2018-05-20 00:00:00|      19.07|    19.26|15.283779680952737|
|2018-05-21 00:00:00|      19.26|    19.31|15.283779680952737|
|2018-05-22 00:00:00|      19.31|    20.69|15.283779680952737|
|2018-05-23 00:00:00|      20.69|    21.14|15.283779680952737|
|2018-05-24 00:00:00|      21.14|    20.15|15.283779680952737|
|2018-05-25 00:00:00|      20.15|    21.54|15.283779680952737|
|2018-05-26 00:00:00|      21.54|    21.87|15.283779680952737|
|2018-05-27 00:00:00|      21.87|    18.27|15.283779680952737|
|2018-05-28 00:00:00|      18.27|       19|15.283779680952737|
|2018-05-29 00:00:00|         19|    18.23|15.283779680952737|
|2018-05-30 00:00:00|      18.23|    18.97|15.283779680952737|
|2018-05-31 00:00:00|      18.97|    22.18|15.283779680952737|
|2018-06-01 00:00:00|      22.18|    21.65|15.283779680952737|
|2018-06-02 00:00:00|      21.65|    20.36|15.283779680952737|
+-------------------+-----------+---------+------------------+
only showing top 20 rows
```

We can also work on multiple files in the same folder

```
allfiles = spark.read \
    .option("inferSchema", "true") \
    .csv('/csvfiles/*.csv') \
    .toDF("DateTime","Humidity","Temperature","Temperature_range (low)","Temperature_range (high)")
allfiles.count()
```

```
2844
```

[26]

```
allfiles.select("temperature","Humidity").summary().show()
```

```
+-------+--------------------+------------------+
|summary|         temperature|          Humidity|
+-------+--------------------+------------------+
|  count|                2844|              2844|
|   mean|   21.64290264575081| 43.94069418930773|
| stddev|   8.124551245802312|18.989245280131374|
|    min|               -0.09|             10.01|
|    25%|                17.8|             26.58|
|    50%|               22.04|              46.0|
|    75%|               24.14|              57.0|
|    max|Temperature_range...|       Temperature|
+-------+--------------------+------------------+
```

It is also possible to use the JOIN operator between dataframes

[27]

```
salledebain = spark.read \
    .option("inferSchema", "true") \
    .csv('/csvfiles/temperature-last-year_salledebain.csv') \
    .toDF("DateTime","Humidity","Temperature","Temperature_range (low)","Temperature_range (high)")

salon = spark.read \
    .option("inferSchema", "true") \
    .csv('/csvfiles/temperature-last-year_salon.csv') \
    .toDF("DateTime","Humidity","Temperature","Temperature_range (low)","Temperature_range (high)")

salledebain.select("DateTime","temperature","Humidity").join(salon.select("DateTime","temperature","Humidity"),"DateTime").show(10)

```

```
+-------------------+-----------+--------+-----------+--------+
|           DateTime|temperature|Humidity|temperature|Humidity|
+-------------------+-----------+--------+-----------+--------+
|           DateTime|Temperature|Humidity|Temperature|Humidity|
|2018-05-14 00:00:00|      21.32|      57|      19.75|      52|
|2018-05-15 00:00:00|      21.27|      58|      19.67|      55|
|2018-05-16 00:00:00|      21.15|      61|      20.42|      55|
|2018-05-17 00:00:00|      21.14|      64|      21.16|      58|
|2018-05-18 00:00:00|      21.63|      67|      21.81|      58|
|2018-05-19 00:00:00|      21.83|      68|      22.17|      59|
|2018-05-20 00:00:00|      21.78|      64|       22.8|      57|
|2018-05-21 00:00:00|       22.1|      63|      23.09|      55|
|2018-05-22 00:00:00|      22.55|      68|      23.22|      57|
+-------------------+-----------+--------+-----------+--------+
only showing top 10 rows
```

SQL Server est entré dans une nouvelle ère

SQL Server est en constante évolution

Nouvelles fonctionnalités

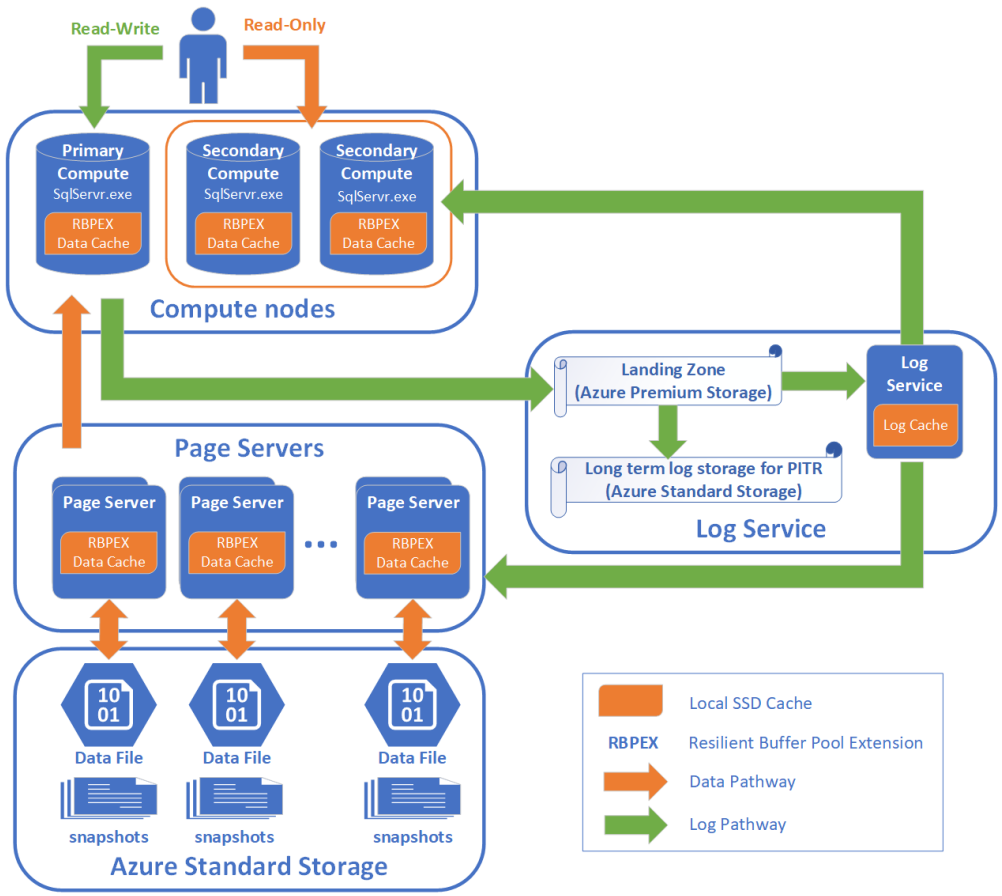
Multi plateforme

Windows, Linux, Docker, K8s

Et même une nouvelle architecture

ScaleOut niveau stockage et calcul

Casse le modèle monolithique de SQL Server



Christophe Laporte

/conseilit

@conseilit

/christophelaporte

conseilit@outlook.com

Q&A

Merci pour votre attention