



\*PASS

SQLSATURDAY

EDINBURGH | 01 FEB 2020

# From Docker to Big Data Clusters

A new era for SQL Server ....

Christophe Laporte, Consultant, Conseil IT



# Christophe Laporte



/conseilit



@conseilit



/christophelaporte



conseilit@outlook.com

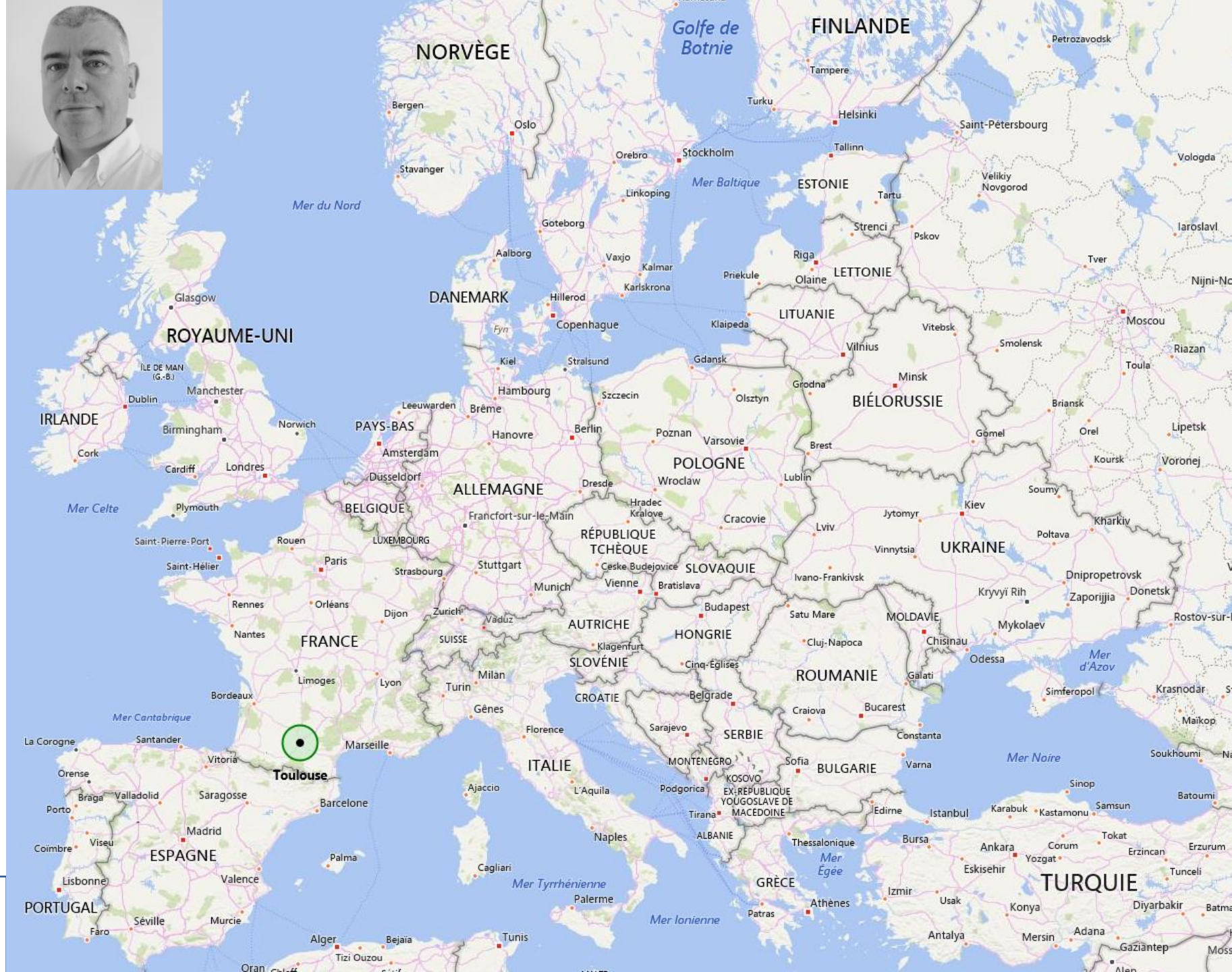


**Microsoft**  
CERTIFIED  
Master

**Microsoft**  
CERTIFIED  
Trainer



~ since 1997 : SQL 6.5 / WinNT4





# Toulouse – South West of France



# Thanks to our Sponsors



..and special THANK YOU to our **Volunteers** ([light blue](#) #SQLSat927 polo shirt)



Own your career with Interactive learning built by community and guided by data experts.

1

Attend an event

2

Join a  
Community

3

Explore More

Get  
involved.  
Get ahead.

*In-person*



\*PASS  
SUMMIT



\*PASS  
SQLSATURDAY



PASS  
LOCAL  
GROUPS



PASS CONNECTOR  
INSIGHTS



PASS.org

*Online*



24HOURS  
OF \*PASS



\*PASS  
MARATHON



PASS  
VIRTUAL  
GROUPS



PASS  
VOLUNTEERS



PASS SUMMIT 2020

# Everything is bigger in Texas.

Join us in Houston November 10 – 13, 2020  
For the largest gathering of Microsoft  
Data Professionals.

## Make Plans for PASS Summit 2020

Over 200 sessions by industry experts and 3 days of networking with people just like you. Don't miss out, future-proof your career at **PASSsummit.com**

# A bit of history

- 10 years ago, we were wondering ...
  - Should I virtualize SQL Server ?
  - What about performance issues ?
  - Which hypervisor ?
- Nowadays
  - Almost all SQL Server instances are virtualized OnPrem
  - And we have good performance
    - Even with Tier-1 workloads ...

# A bit of history

- 2016 : SQL Server 2017 on Linux announcement

## 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 on Linux

Remember, SQL Server  
was derived from  
Sybase ...



ubuntu

- A requirement from customers and ISVs
- New OS
  - An achievement ?
  - ... or a new the chapter for SQL Server ...



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

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

# Installing SQL Server on linux

- Quite straightforward
  - For basic installation

```
# ubuntu
wget -qO- https://packages.microsoft.com/keys/microsoft.asc | sudo apt-key add -
sudo add-apt-repository "$(wget -qO- https://packages.microsoft.com/config/ubuntu/16.04/mssql-server-2017.list)"
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 https://packages.microsoft.com/config/rhel/7/mssql-server-2017.repo
sudo yum install -y mssql-server
sudo /opt/mssql/bin/mssql-conf setup

# Suse
sudo zypper addrepo -fc https://packages.microsoft.com/config/sles/12/mssql-server-2017.repo
sudo zypper --gpg-auto-import-keys refresh
sudo zypper install -y mssql-server
sudo /opt/mssql/bin/mssql-conf setup
```

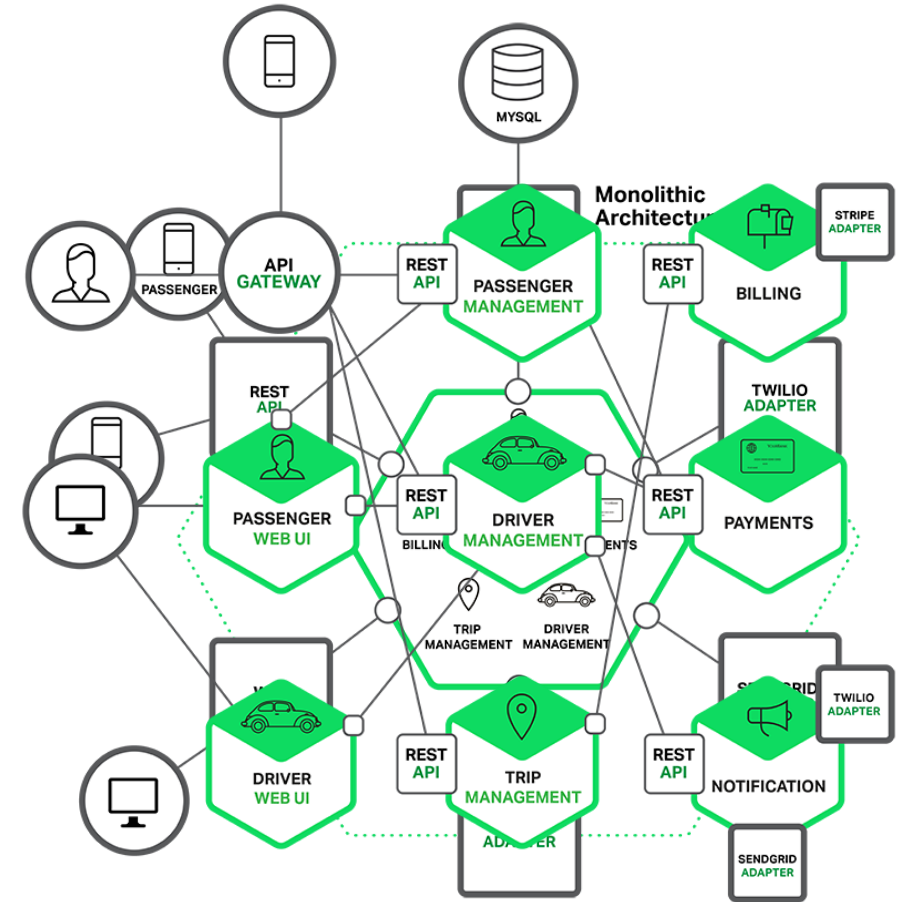
ubuntu®





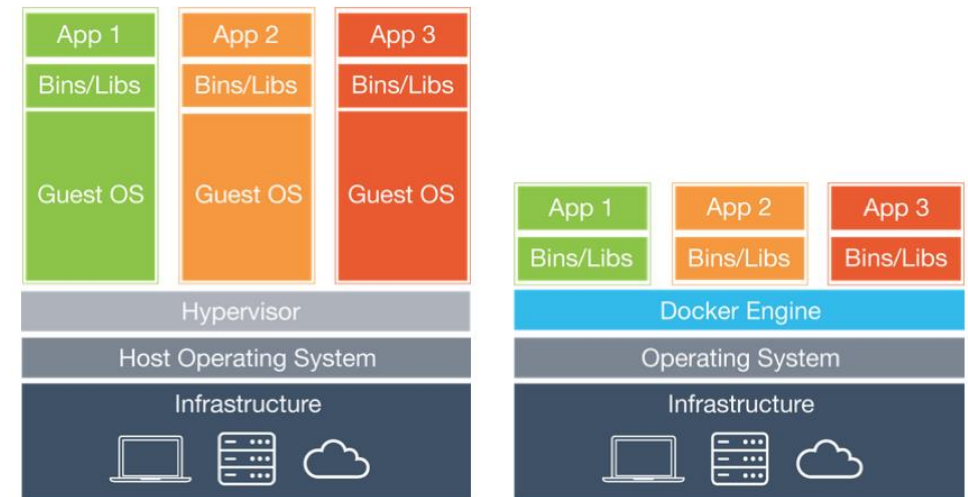
# Micro services design pattern

- Yesterday : monolithic applications
  - Hard to maintain / evolve
- Today : micro services
  - New way to develop applications
  - Lightweight pieces of SW evolving independently
  - 1, 10s or 100s of containers composed as a single application
- It seems to become a standard
  - From an infrastructure prospective
  - From a DevOps “philosophy”



# Containerization : virtualization v2.0

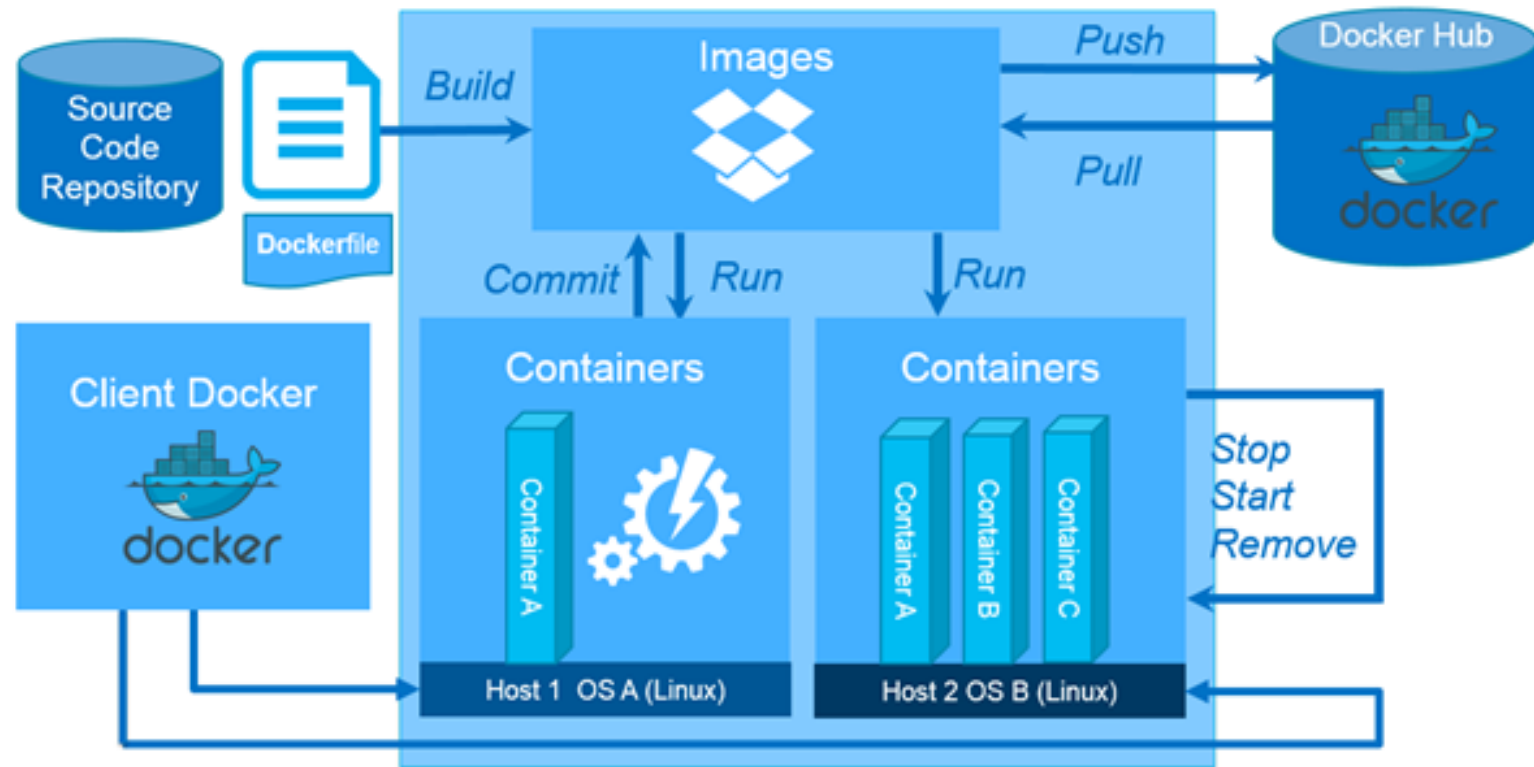
- Small system footprint
  - Lightweight -> better efficiency on host servers
- Single image
  - Multiple deployments (dev / test / prod)
  - Avoid : “it works on my computer” !
- Will always run the same
  - Regardless of where it is deployed







- Multi OS
  - Windows, Linux, Mac
- Docker engine
  - Containers execution
- Docker client
  - Command line utility
- Terminology
  - Image
  - Container
  - Repository



# Docker – The command line utility

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)



# Running my first container



# 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/>

# But ... Wait ....

- We can run SQL Server on Linux
- So, why not running it in a container ?



```
# Run (Pull+Create+Start) the container in detach mode
```

```
docker run --detach \  
  --name sqldocker \  
  --hostname sqldocker \  
  --env 'MSSQL_PID=developer' \  
  --env 'SA_PASSWORD=Password1!' \  
  --env 'ACCEPT_EULA=Y' \  
  --volume /mssql:/var/opt/mssql/data \  
  --publish 1433:1433 \  
  mcr.microsoft.com/mssql/server:2019-latest
```

```
# Run (Pull+Create+Start) the container in detach mode
```

```
# Container name
```

```
# OS name
```

```
# Edition : developer is the default value
```

```
# Password for SA account
```

```
# You still need to acknowledge licence terms
```

```
# Redirect storage to persist data
```

```
# TCP endpoint to connect the container
```

```
# Image used to build and start the container
```

# Now ... What's next ?

- We need some orchestration for containers
  - Ensure container is healthy --> restart container
  - Ensure host is healthy --> Restart on a different host
  - Provide network access to the containers
  - Provide persistent storage across multiples nodes
  - Manage container resources (CPU, RAM ...)
- And we wish a similar deployment experience
  - OnPrem
  - Public Cloud
- And some scaling functions

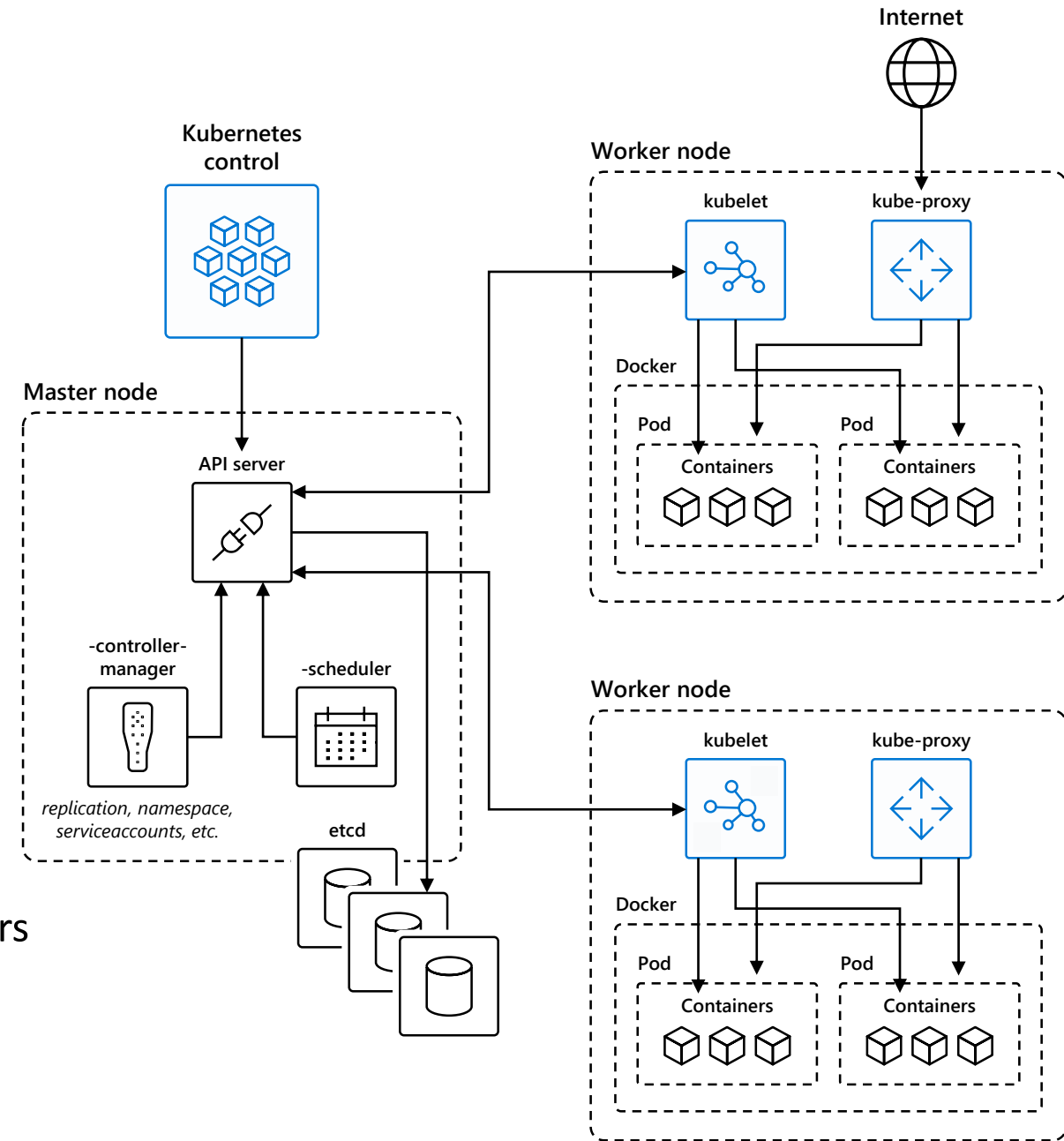
Sounds familiar ...  
Hey, that's a cluster !





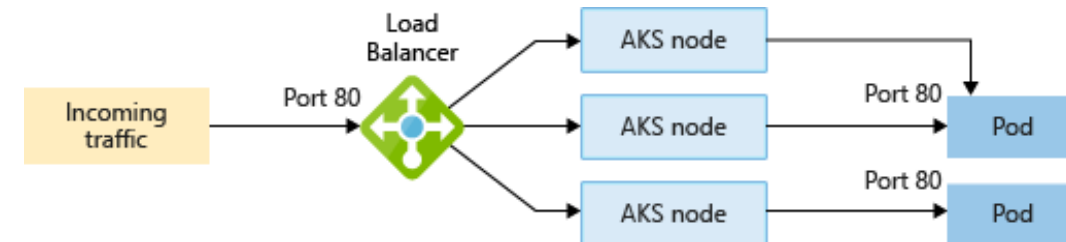
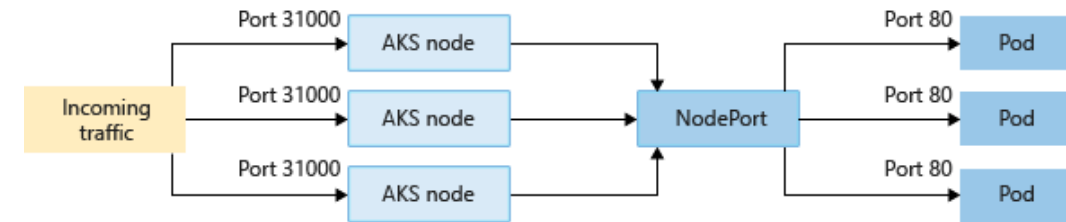
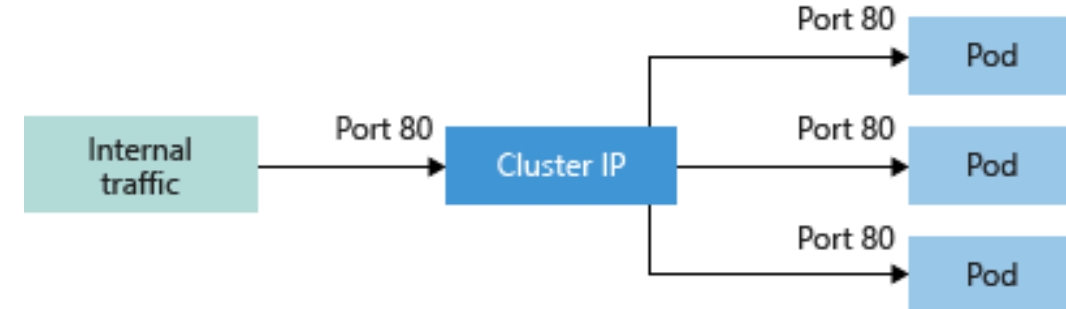
# Kubernetes for DBAs

- Also known as K8s
- Terminology
  - Container
  - Pod
    - Smallest management unit
    - 1..N containers
    - Unique @IP across the cluster
  - Master node
    - Responsible for pod scheduling
  - Worker node
    - Node of the K8s Cluster
    - Kubelet : responsible for running containers
    - Kube-proxy : manage network traffic
- Desired State Configuration



# Connecting to applications

- Connections goes through kube-proxy
  - Routing and NATing to the Pod
  - No matter the worker node
- Services
  - Exposes applications
  - Logical abstraction of one or more Pods
- Different types of service
  - ClusterIP
  - Node Port
  - Load Balancer



# Kubectl : the command line utility

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



# A bit of history (again)

- 10 years ago, we were wondering ...
  - Should I virtualize SQL Server ?
  - Almost all SQL Server instances are virtualized OnPrem
- 10 years ago, Microsoft was introducing 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 features, Windows Server capabilities, and marketplace offerings that will make it easier for developers to build profitable businesses from their Microsoft-based solutions.

# Containers services on Azure

- ACI : Azure Container Instance
  - [Easiest](#) way to spin up a container, without managing Kubernetes / Servers / ...
  - Charged for each GB and vCPU second your container group consumes

```
# 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;"
```

# Containers services on Azure

- AKS : Azure Kubernetes Service
  - Fully managed K8s cluster
  - Everything is configured for you
    - Storage
    - Network
    - K8s configuration (master, worker nodes)
- You have to create the cluster
- You have to maintain the cluster
- Charged for the virtual machines and the associated storage and networking resources consumed

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

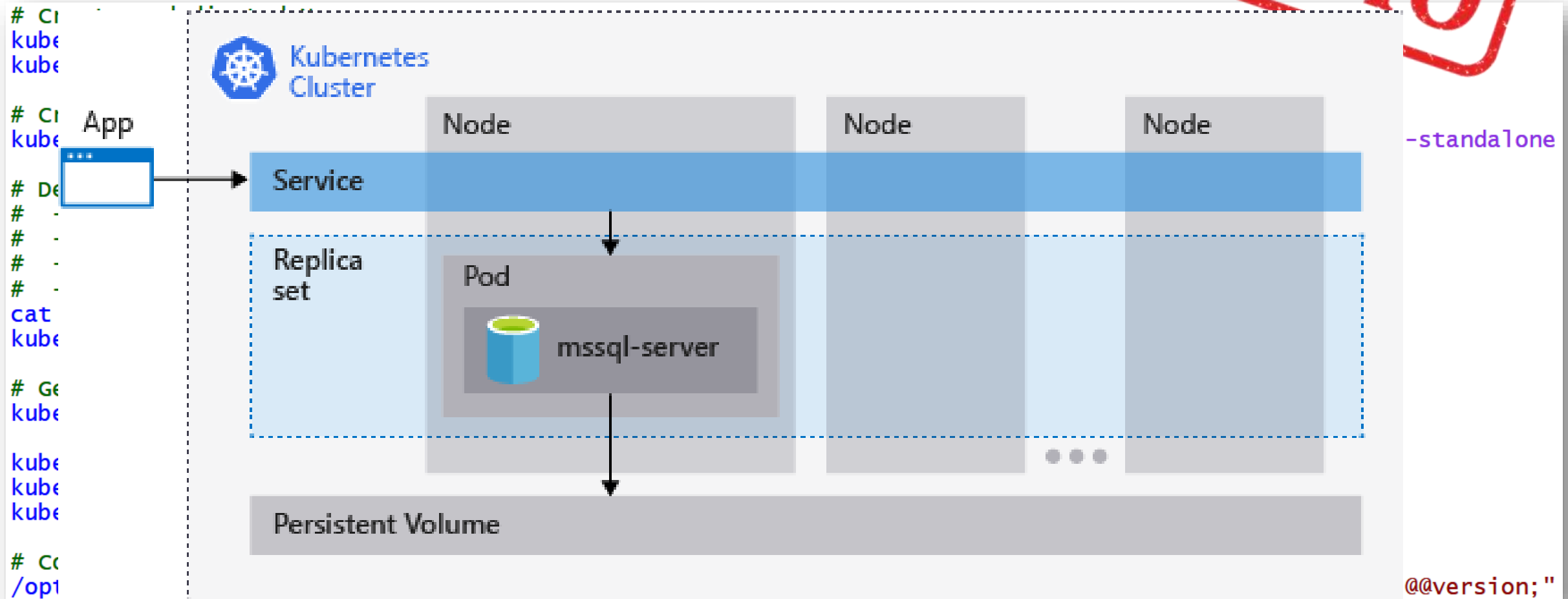
# List currently supported Kubernetes version
az aks get-versions --location francecentral --output table

# 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
```

# Deploying SQL Server on AKS

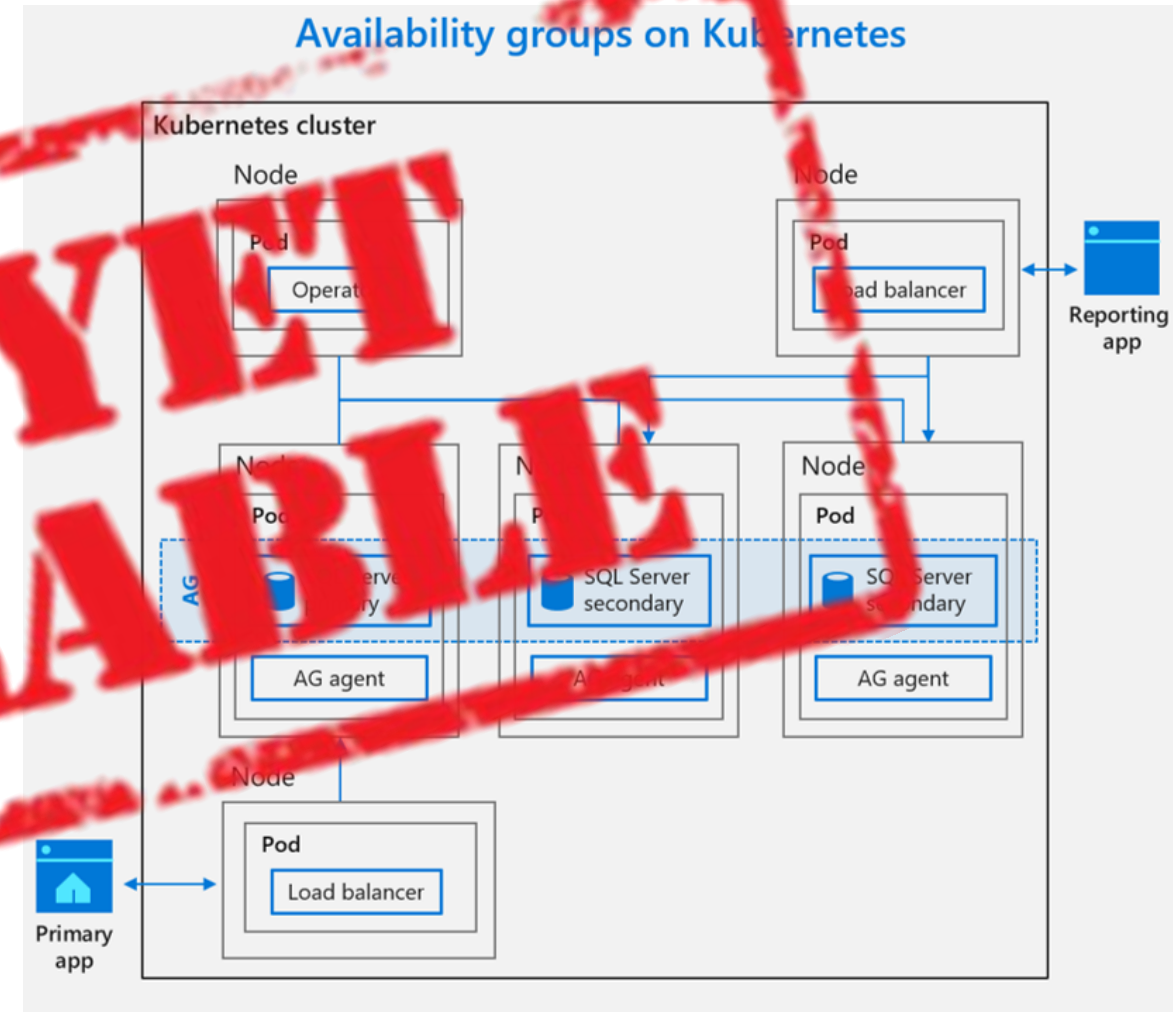
DEMO





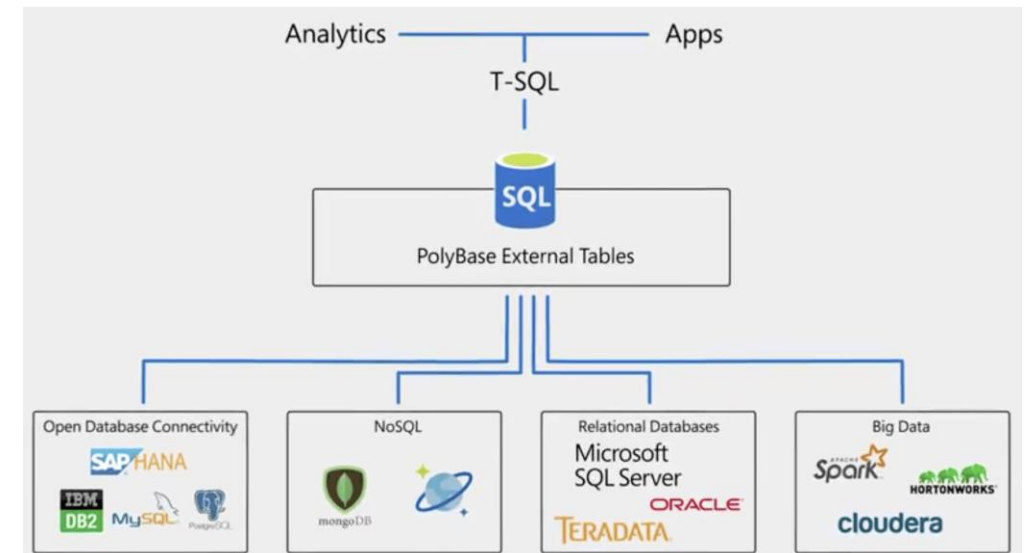
# SQL Server HA inside K8s

- *mssql-operator*
  - Implements the Kubernetes operator for SQL Server and Availability Groups.
- *mssql-server-k8s-health-agent*
  - Implements the logic to determine the health of a SQL Server Instance
- *mssql-ha-supervisor*
  - Implements the AG health detection and management logic, including the leader election logic to determine the Primary replica for the availability group. The leader election functionality is based on the functionality of the Kubernetes ConfigMap leader election.
- *mssql-server-k8s-init-sql*
  - Implements the logic for deployment and initialization of a desired state configuration to a SQL Server instance.

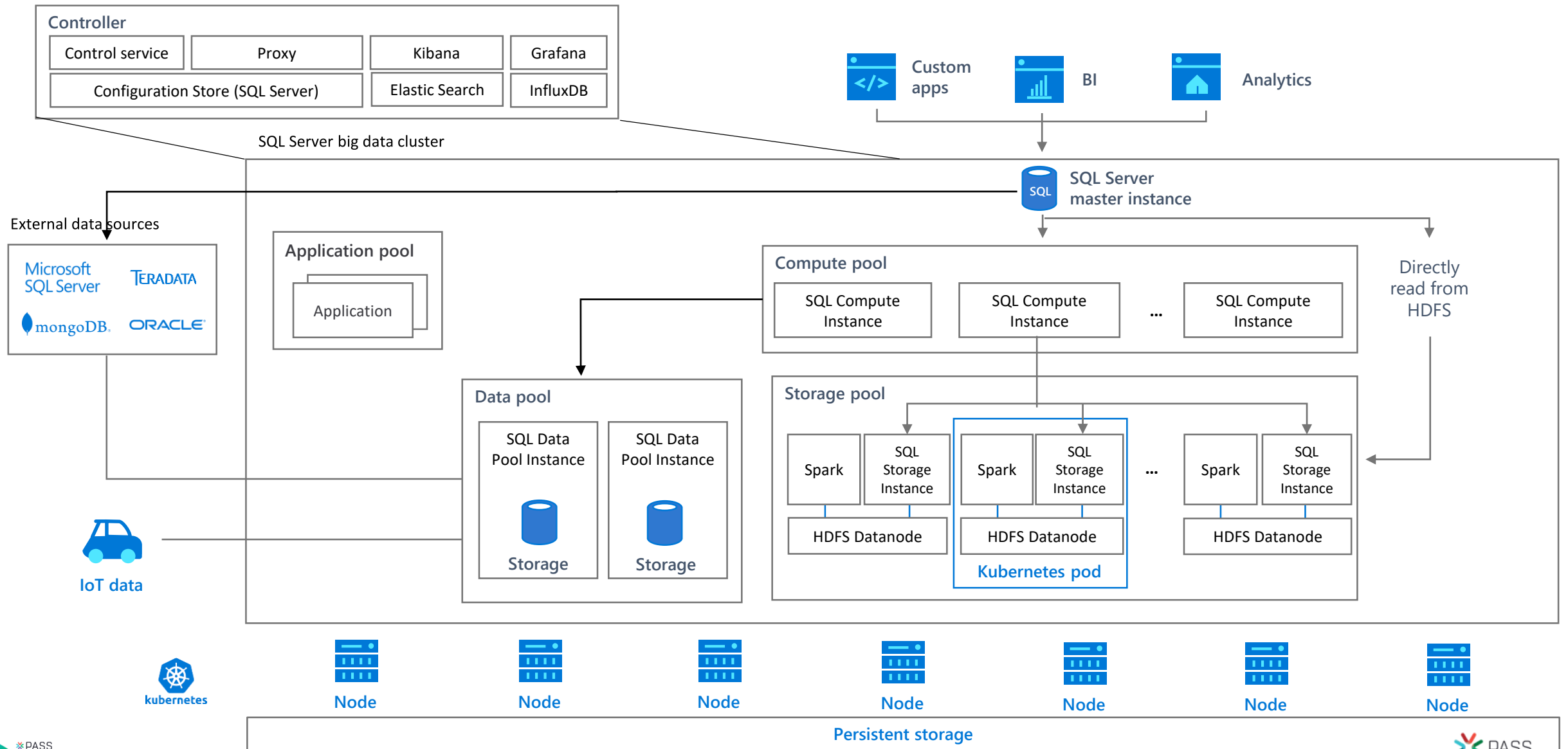


# But ... Wait ....

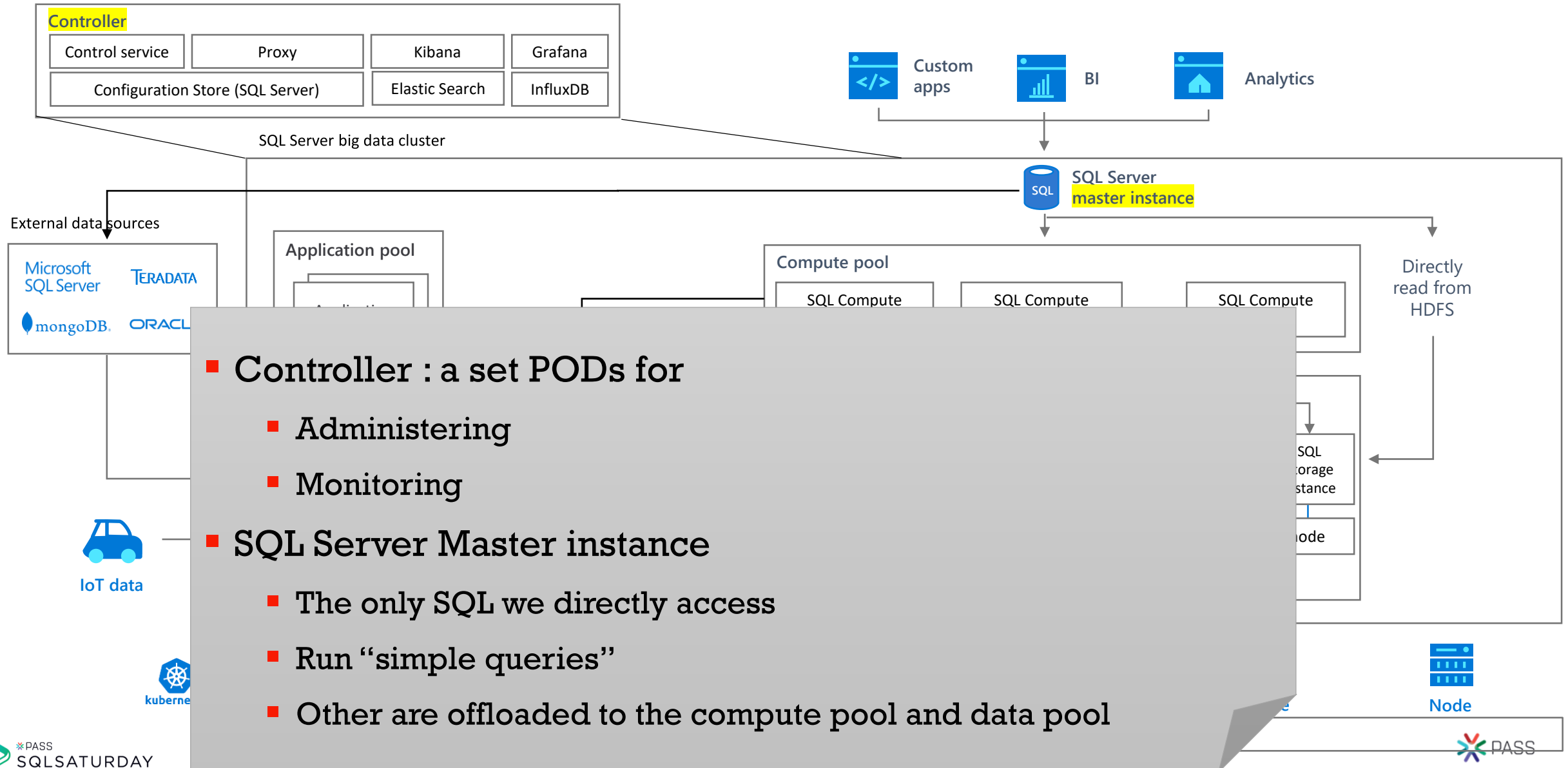
- Nowadays SQL Server is more than a SGBD
  - SQL Server offers Data Virtualization with Polybase
- K8s can run almost all kind of applications
- K8s can run SQL Server
  - ~~with AlwaysOn Availability Groups~~
- Pods can host multiple containers
- Hummmm ....
- Let's add some « Big Data » containers
  - With Spark engine
  - And some kind of HDFS storage



# That's SQL Server 2019 Big Data Cluster



# Control plane





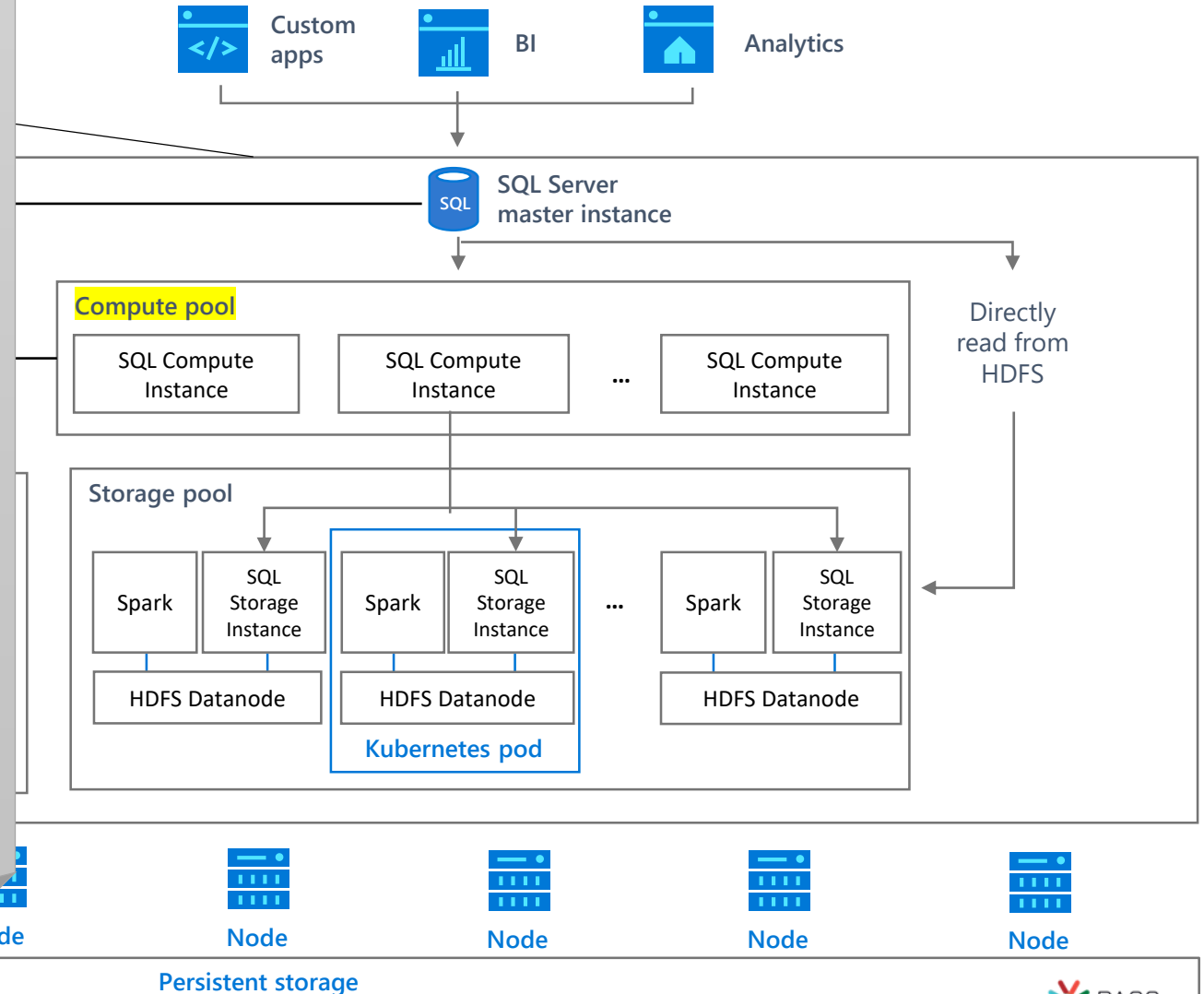
# Compute plane

- **Compute Pool is a set SQL instances:**

- Provides compute resources for distributed queries
- Provides same functionality as PolyBase Scale-out Group

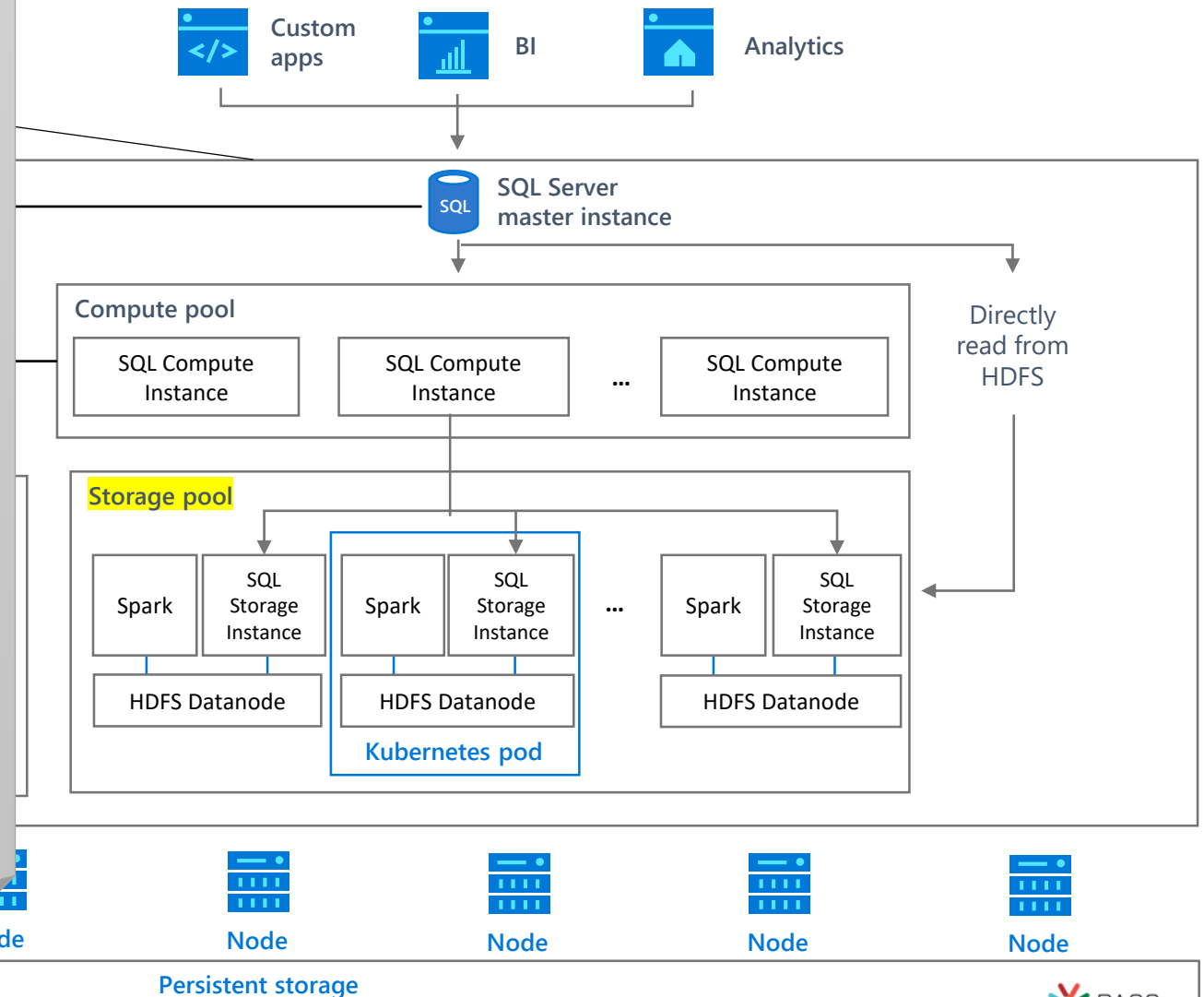
- **Used to**

- Join directories in HDFS
- Join tables in different data sources
- Offload driver communication from SQL Server Master instance
- ...

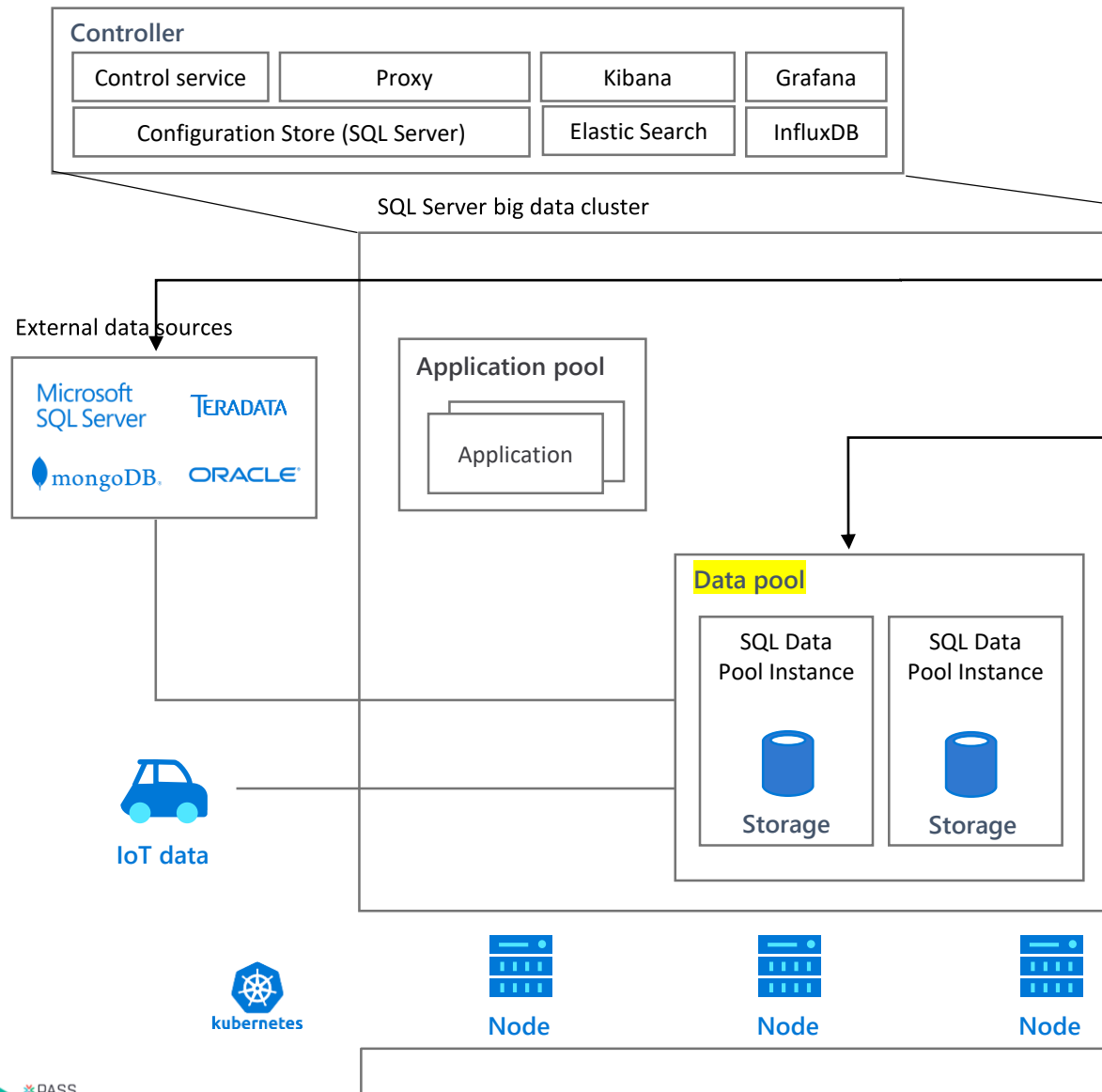


# Data plane : Storage pool

- 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
  - ...

# Let's play with SQL Server 2019 BDC



```
[7] 1 USE DemoDB;
     2 GO
     3 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] 1 # Read the CSV file(s) into a spark dataframe and print schema
     2 results = spark.read \
     3     .option("inferSchema", "true") \
     4     .csv('/csvfiles/temperature-last-year_poolhouse.csv') \
     5     .toDF("DateTime", "Humidity", "Temperature", "Temperature_range (low)", "Temperature_range (high)")
     6 results.printSchema()
```

Starting Spark application

ID	YARN Application ID	Kind	State	Spark UI	Driver log	Current session?
5	application_1573997616589_0001	pyspark	idle	<a href="#">Link</a>	<a href="#">Link</a>	✓

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] 1 results.show(5)
```

```
+-----+-----+-----+-----+-----+
|      DateTime|Humidity|Temperature|Temperature_range (low)|Temperature_range (high)|
+-----+-----+-----+-----+-----+
|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] 1 results.filter("Humidity > 70").show()
```

DateTime	Humidity	Temperature	Temperature_range (low)	Temperature_range (high)
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		
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] 1 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 create a kind of view and 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|
|2018-05-20 00:00:00|      67|      19.07|          16.4|          24.5|
```

[23]

```
1 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]

```
1 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

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

2844

```
[26] 1 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] 1 salledebain = spark.read \  
2     .option("inferSchema", "true") \  
3     .csv('/csvfiles/temperature-last-year_salledebain.csv') \  
4     .toDF("DateTime", "Humidity", "Temperature", "Temperature_range (low)", "Temperature_range (high)")  
5  
6 salon = spark.read \  
7     .option("inferSchema", "true") \  
8     .csv('/csvfiles/temperature-last-year_salon.csv') \  
9     .toDF("DateTime", "Humidity", "Temperature", "Temperature_range (low)", "Temperature_range (high)")  
10  
11 salledebain.select("DateTime", "temperature", "Humidity").join(salon.select("DateTime", "temperature", "Humidity"), "DateTime").show(10)  
12
```

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



# Conclusion

- SQL Server is constantly evolving
  - New features / capabilities
  - Even new architecture ...
- DBAs must constantly acquire new skills
  - Containers and orchestration
  - Cloud (Azure, AWS, GCP)
- SQL Server is entering a new era
  - Multi platform
  - Containerization is the next step of virtualization
  - Working with “Big Data” and Relational Databases is easiest than ever



Windows

Run SQL Server 2019 database engine on Windows.



Linux

Run SQL Server 2019 database engine on Linux.



Docker

Run SQL Server 2019 database engine container image with Docker.



Big data analytics

Run SQL Server 2019 big data analytics container images with Kubernetes.



Thank you

