STRATEGIC PARTNER

Microsoft

GOLD SPONSOR

elitmind think bright

lingaro

predica.

SILVER SPONSOR

Future Processing

Data Community

Christophe Laporte

From Docker to BDC
A new era for SQL Server

SQLDay 2021

SQL Day

Christophe Laporte

Trainer & Consultant

/conseilit

@conseilit

/in/christophelaporte

conseilit@outlook.com

Microsoft CERTIFIED AZURE SOLUTIONS ARCHITECT EXPERT

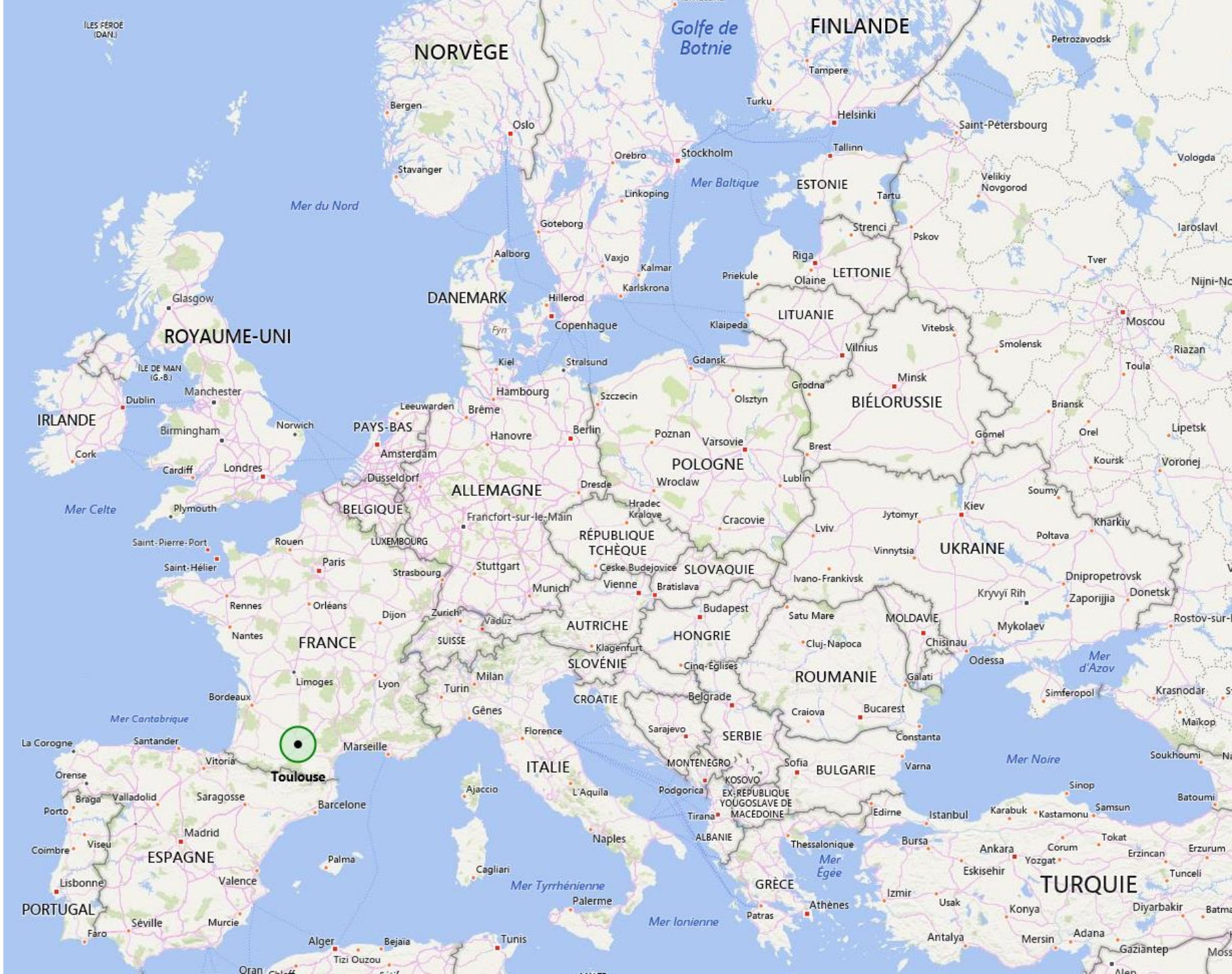
Microsoft CERTIFIED AZURE ADMINISTRATOR ASSOCIATE

MVP Microsoft® Most Valuable Professional

Microsoft CERTIFIED Master

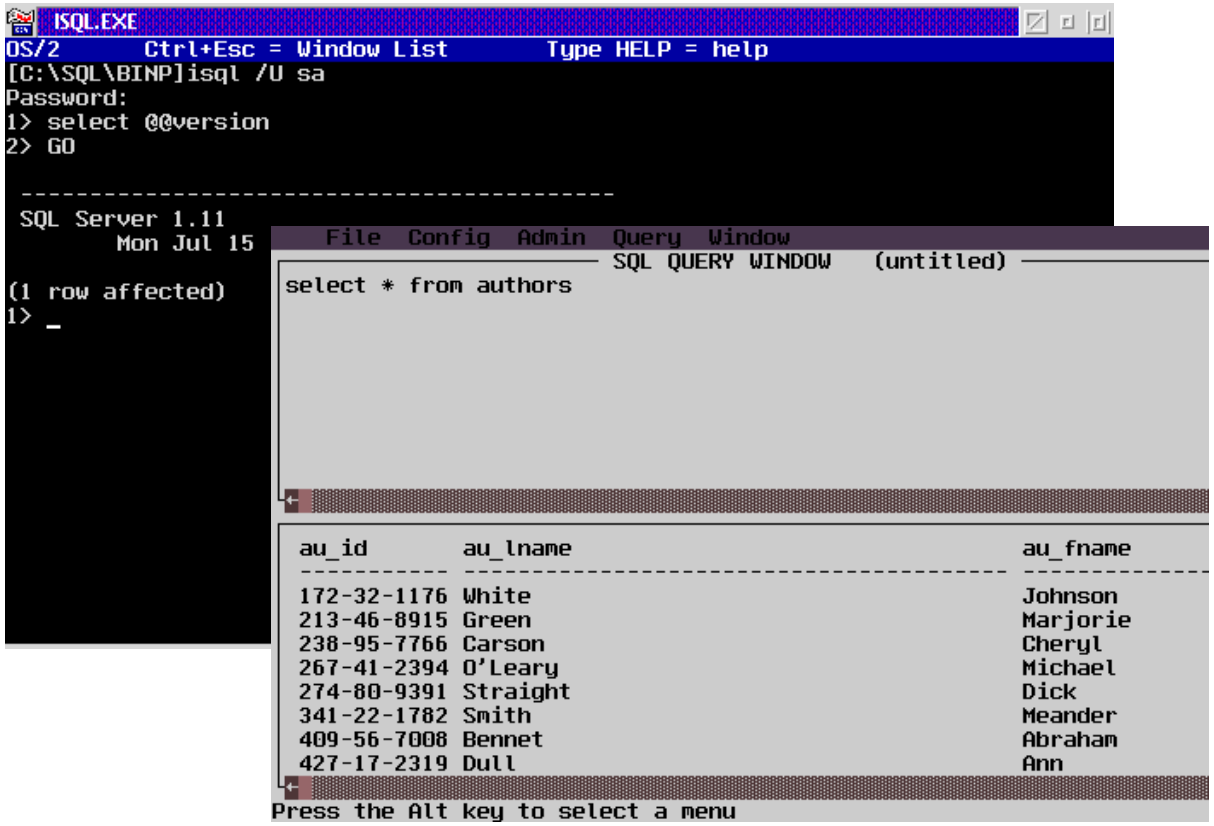
Microsoft CERTIFIED Trainer

~ since 1997 : SQL 6.5 / WinNT4



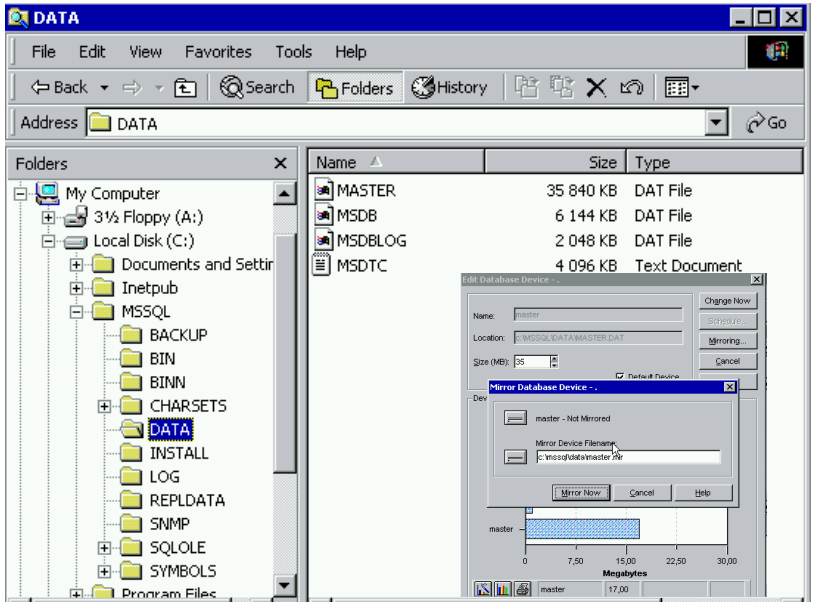
Once upon a time

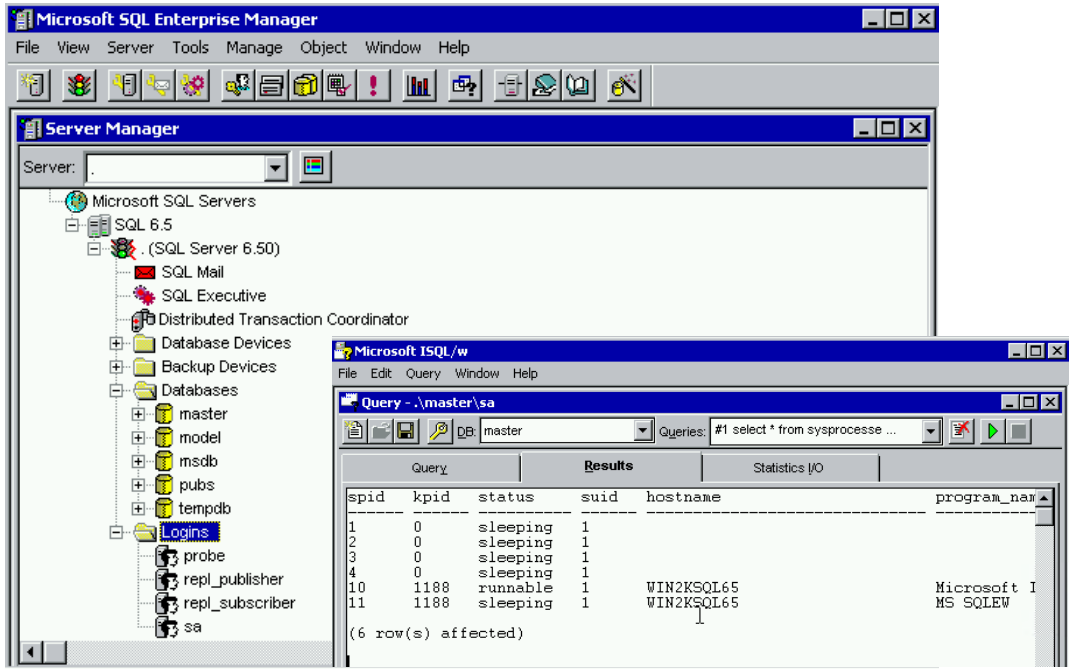
- 1989 - SQL Server was born
  - Project to port Sybase onto OS/2
- 1993 – SQL Server 4.21a
  - First landing on Windows NT
- 1995 – SQL Server 6.0
  - End of collaboration with Sybase



When my journey started

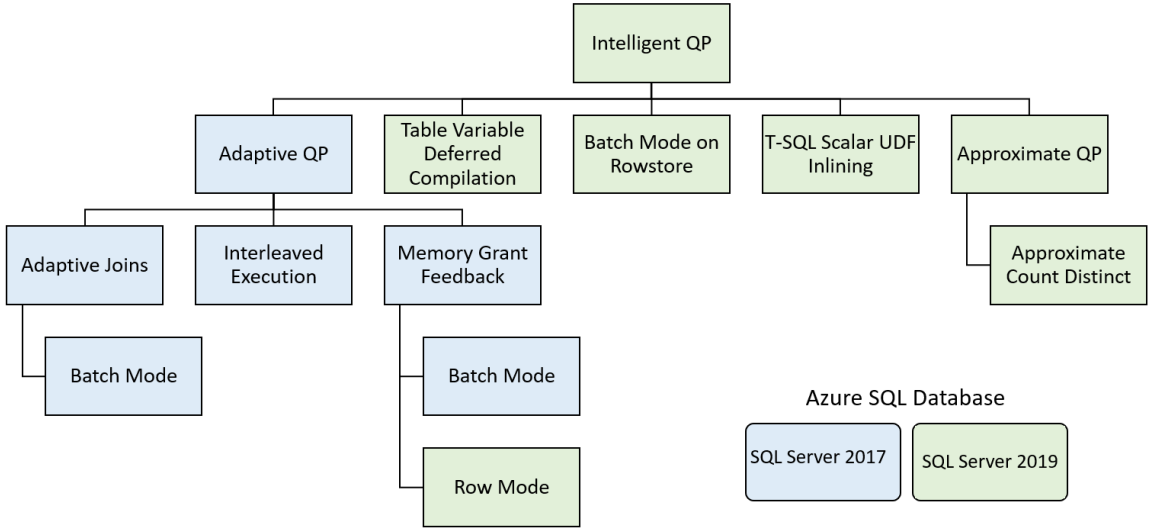
- 1998 – SQL Server 6.5
  - SQL Trace, Clustering, ANSI syntax
- 1998 – SQL Server 7.0
  - C source code converted to C++





SQL Server is evolving

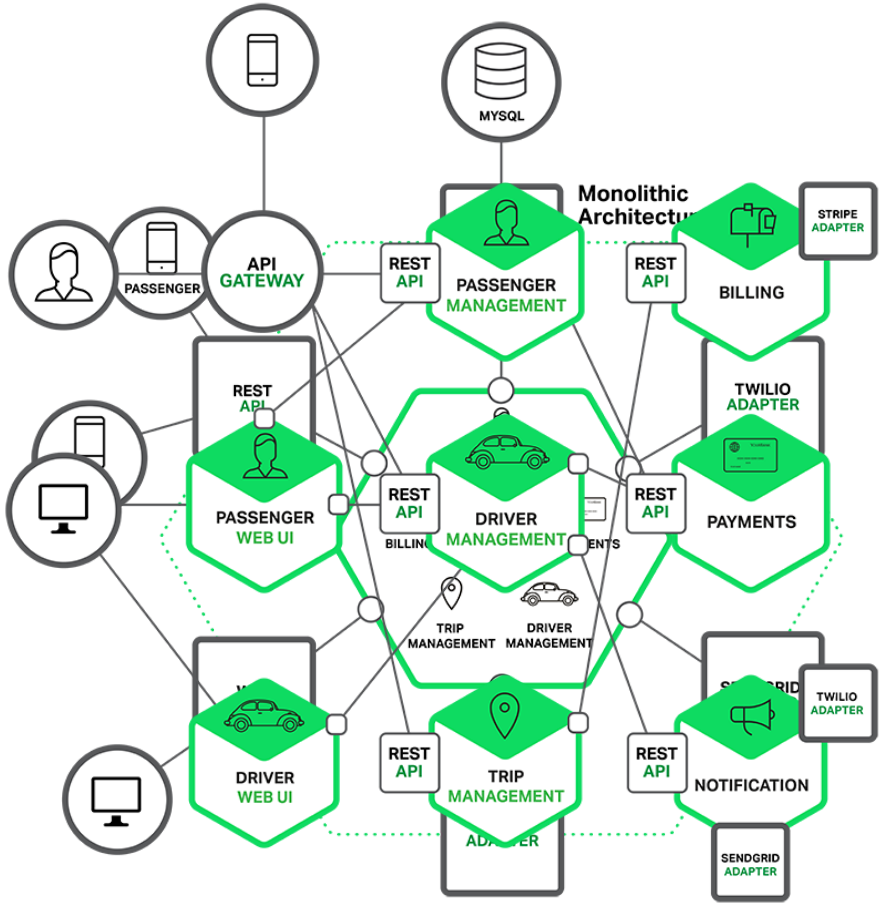
- New functionalities
  - Temporal tables, Graph Databases
  - HA/DR capabilities
- Security improvements
  - TDE, Always Encrypted
  - RLS, Dynamic Data Masking
- Extensibility
  - SQLCLR, Java, Python, R
  - Polybase
- Performance enhancements
  - InMemory OLTP, Columnstore index
  - New CE, TempDB scalability, Intelligent QP



Applications are evolving

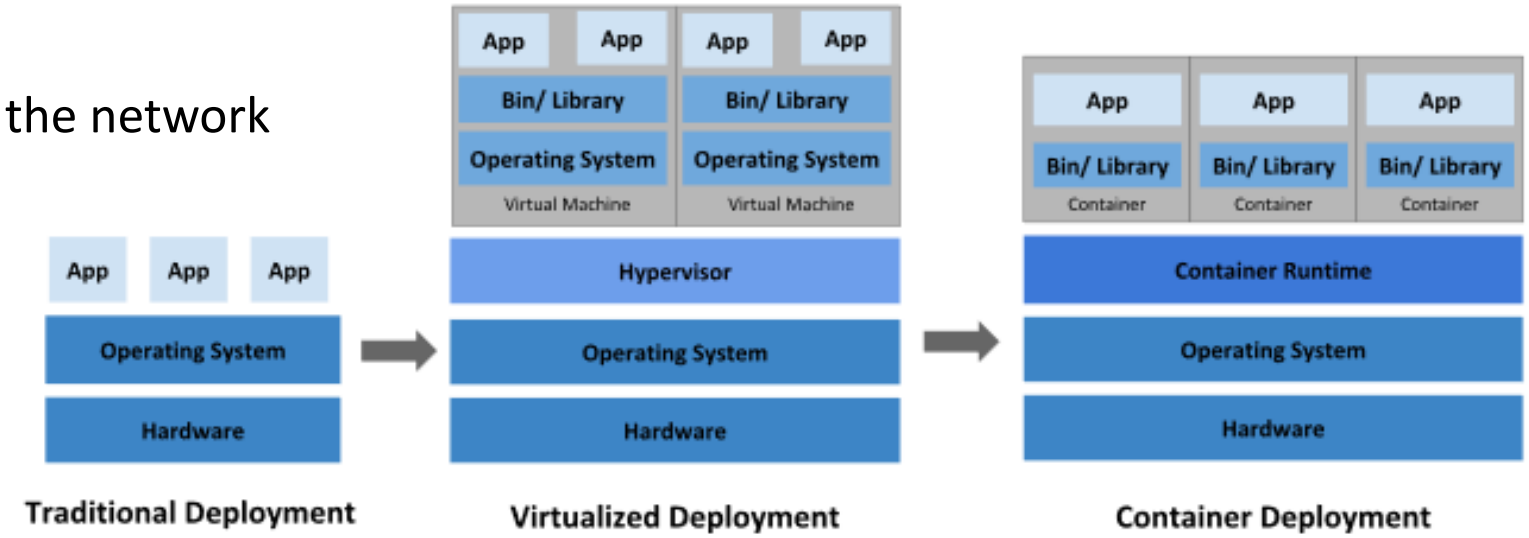
Micro services design pattern

- 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”



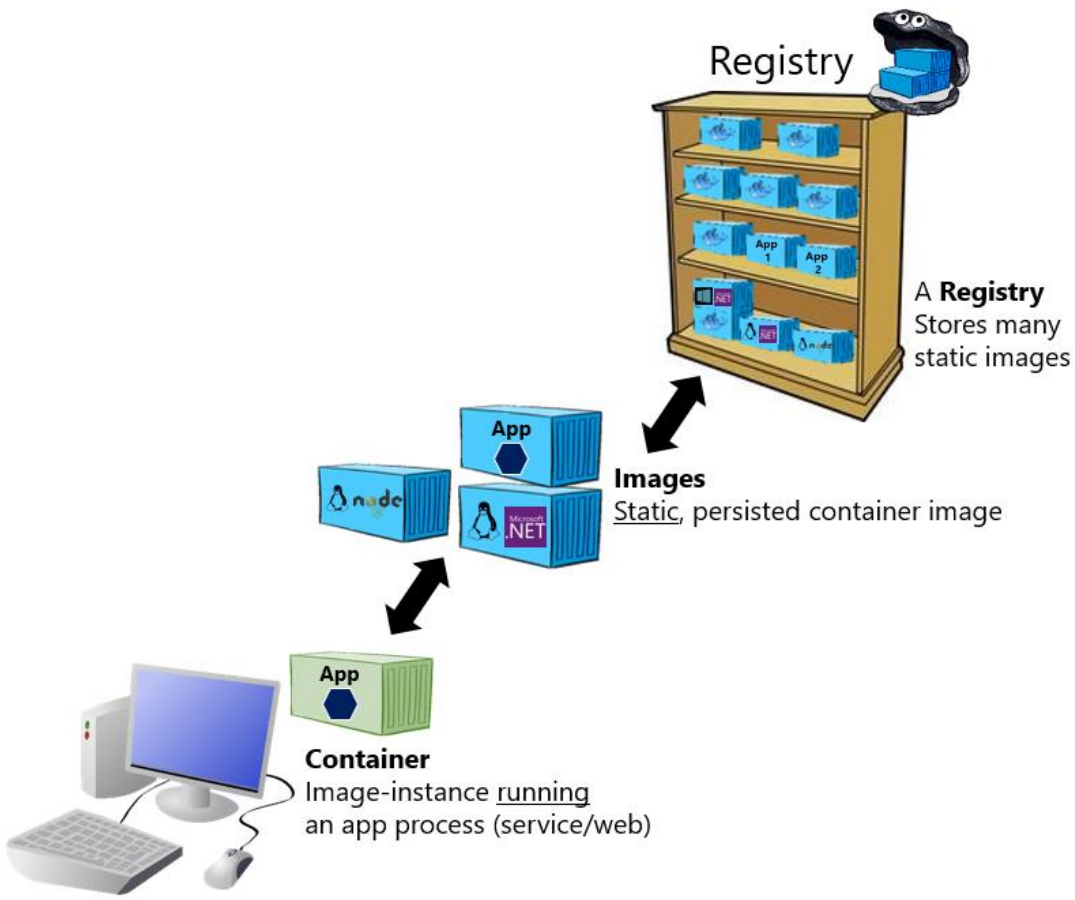
Containers for DBAs

- Virtualization 1.0
  - Hardware virtualization (Hyper-V, VMware, …)
- Virtualization 2.0
  - OS virtualization known as containerization
- Isolation
  - Container not visible from the network
  - Stateless



Containers for DBAs

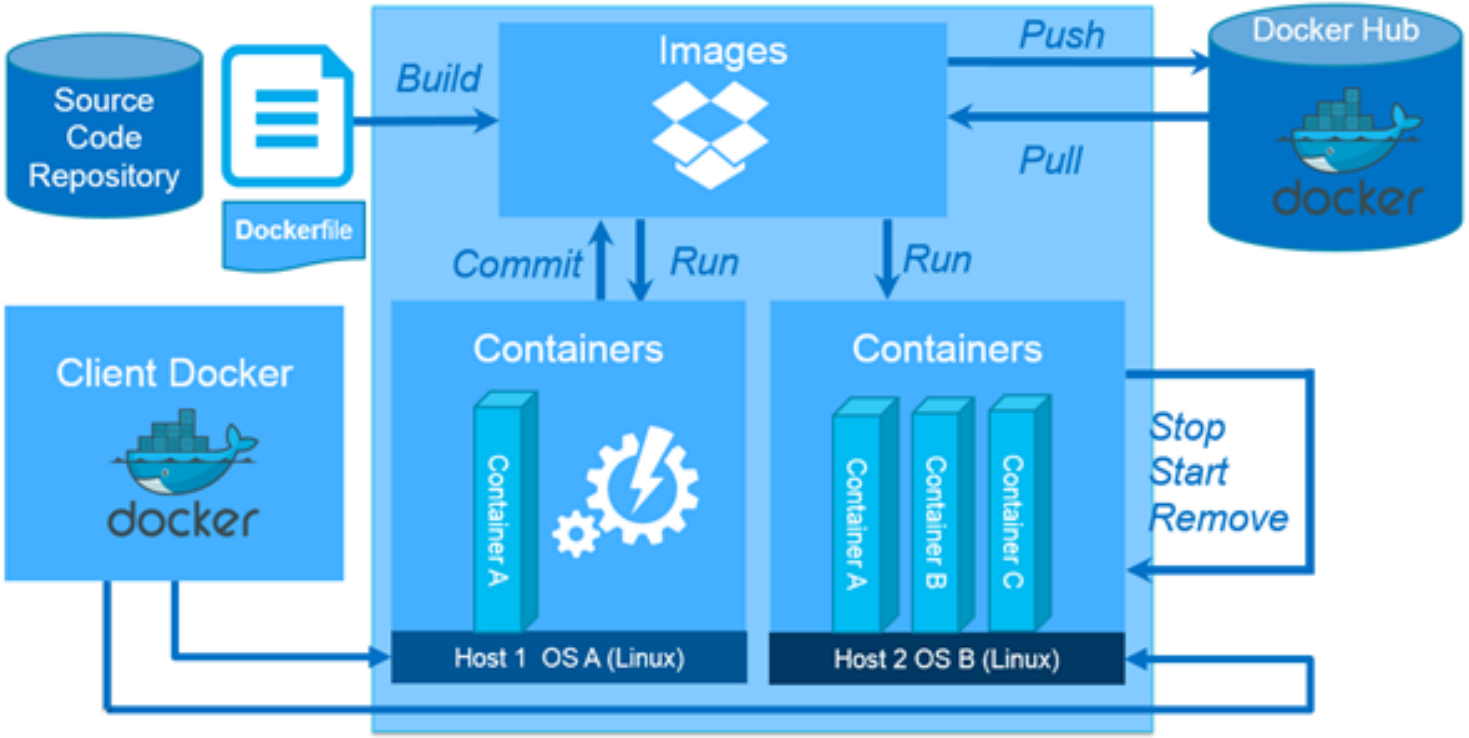
- Small system footprint
  - Lightweight -> better efficiency on host servers
- Single image
  - Multiple identical deployments (dev / test / prod)
  - Avoid : “it works on my computer” !
- Will always run the same
  - Regardless of where it is deployed
- Fast deployment



Docker containers, images, and registries | Microsoft Docs

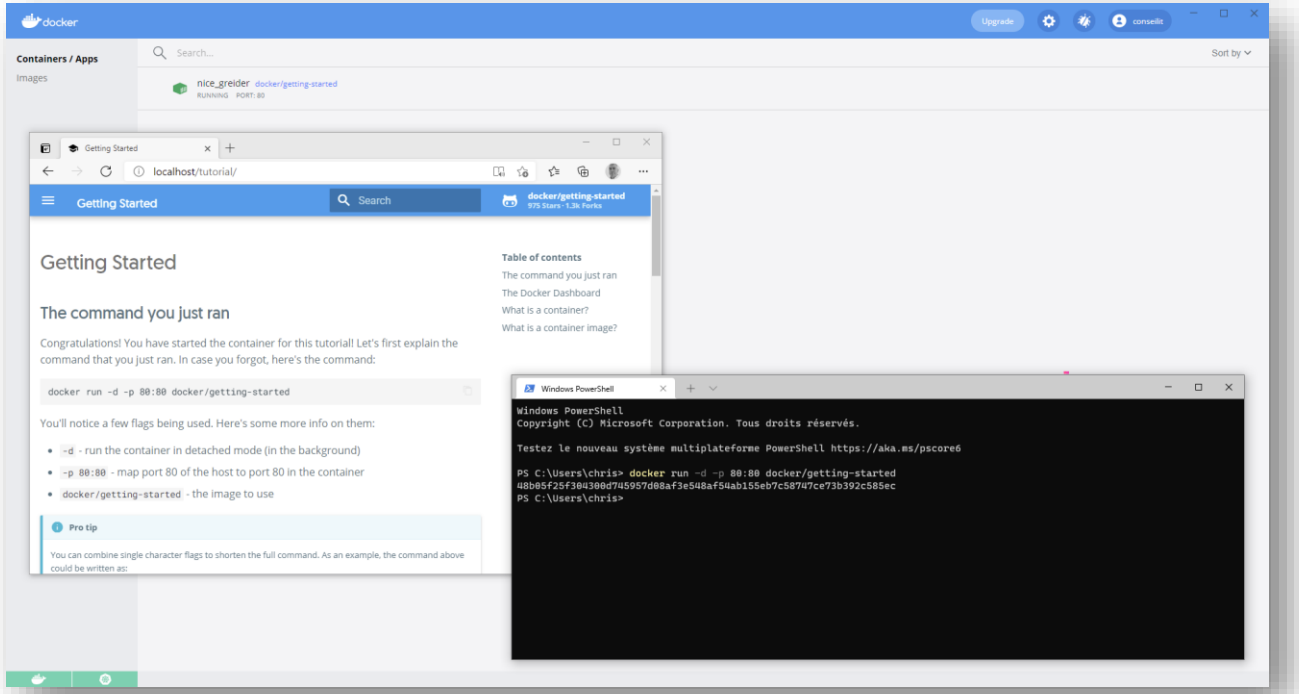
docker

- Docker engine
  - Containers execution
- Docker client
  - Command line utility
- Docker alternatives
  - Podman
  - Mesos
  - LXC
  - Rkt



docker

- Multi OS
  - Windows, Linux, Mac
  - Same command line
  - Same behavior
- Easiest way to use it ?
  - 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 command line

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)

Hello world from Docker


```
## 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/
```

Containers for DBAs

- Containers can run all kind of applications
  - So, why not running SQL Server in a container ?
- Need some extra stuff like
  - TCP port redirection
  - Persistent storage
- Official images from Microsoft
  - SQL Server **2016** (windows containers)

Docker & SQL Server

Docker Pull

- Images officielles Microsoft
  - SQL Server 2016 SP1
  - SQL Server v.Next

```
# Pull SQL Server Express 2016 SP1 from Docker Hub
docker pull microsoft/mssql-server-windows-express

# Pull SQL Server Developer 2016 SP1 from Docker Hub
docker pull microsoft/mssql-server-windows-developer

# Pull SQL Server developper v.Next from Docker Hub
docker pull microsoft/mssql-server-windows
```

Fichier Dockerfile

```
FROM microsoft/dotnet35

MAINTAINER Christophe Laporte

ENV sqlinstance SQL
ENV sqlsapassword Password1
ENV sql c:\\sql
ENV sqldata c:\\sql\\data
ENV sqlbackup c:\\sql\\backup

COPY . /install

WORKDIR /install

RUN  /install/sqlexpr_x64_enu.exe /q /x:/install/setup \
  && /install/setup/setup.exe /q /ACTION=Install /INSTANCENAME=%sqlinstance%
/FEATURES=SQLEngine /UPDATEENABLED=0 \
      /SECURITYMODE=SQL /SAPWD=%sqlsapassword% /SQLSVCACCOUNT="NT
AUTHORITY\System"  /SQLSYSADMINACCOUNTS="BUILTIN\ADMINISTRATORS" \
      /INSTALLSQLDATADIR=%sqldata% /SQLUSERDBLOGDIR=%sqldata%
/SQLBACKUPDIR=%sqlbackup% \
      /TCPENABLED=1 /NPENABLED=0 /IACCEPTSQLSERVERLICENSETERMS \
  && powershell ./Set-SqlExpressStaticTcpPort %sqlinstance% \
  && powershell ./Move-dirs-and-stop-service %sqlinstance% %sql% %sqldata% %sqlbackup% \
  && del sqlexpr_x64_enu.exe \
  && rmdir .\setup /s /q

CMD powershell ./start detached %sqlinstance% %sqldata% %sqlbackup%
```

SQLSaturday Montreal 2017

SQL Server 2017 on Linux

https://blogs.microsoft.com/blog/2016/03/07/announcing-sql-server-on-linux/

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.

Installing SQL Server on linux

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
```

Installing SQL Server on linux

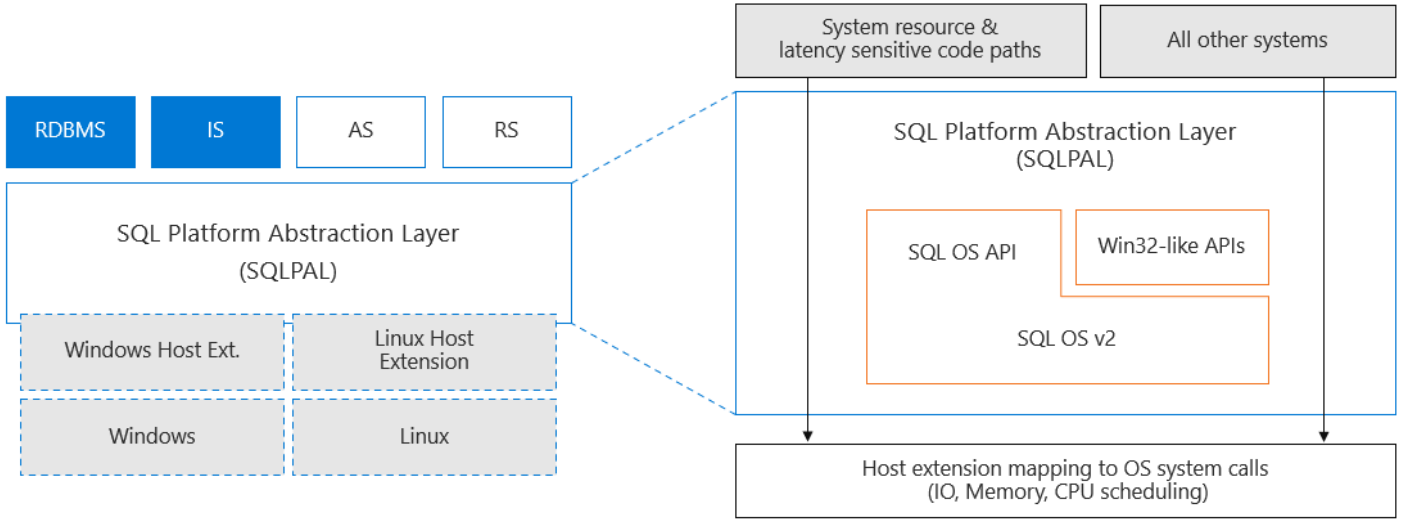
Straightforward for basic installation

```
Preparing to unpack .../7-libc6-dbg_2.27-3ubuntu1_amd64.deb ...
Unpacking libc6-dbg:amd64 (2.27-3ubuntu1) ...
Selecting previously unselected package libsss-nss-idmap0.
Preparing to unpack .../8-libsss-nss-idmap0_1.16.1-1ubuntu1.5_amd64.deb ...
Unpacking libsss-nss-idmap0 (1.16.1-1ubuntu1.5) ...
Selecting previously unselected package mssql-server.
Preparing to unpack .../9-mssql-server_15.0.4033.1-2_amd64.deb ...
Unpacking mssql-server (15.0.4033.1-2) ...
Setting up libc++abi1:amd64 (6.0-2) ...
Setting up libcc1-0:amd64 (8.4.0-1ubuntu1~18.04) ...
Setting up libc6-dbg:amd64 (2.27-3ubuntu1) ...
Setting up libsss-nss-idmap0 (1.16.1-1ubuntu1.5) ...
Setting up gdbserver (8.1-0ubuntu3.2) ...
Setting up libsasl2-modules-gssapi-mit:amd64 (2.1.27~101-g0780600+dfsg-3ubuntu2.1) ...
Setting up libbabeltrace1:amd64 (1.5.5-1) ...
Setting up libc++1:amd64 (6.0-2) ...
Setting up gdb (8.1-0ubuntu3.2) ...
Setting up mssql-server (15.0.4033.1-2) ...

+------------------------------------------------------------+
Please run 'sudo /opt/mssql/bin/mssql-conf setup'
to complete the setup of Microsoft SQL Server
+------------------------------------------------------------+

Processing triggers for libc-bin (2.27-3ubuntu1) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Christophe@lxSQL-vm:~$
```

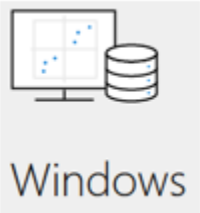
```
Christophe@lxSQL-vm:~$ sudo /opt/mssql/bin/mssql-conf setup
usermod: no changes
Choose an edition of SQL Server:
  1) Evaluation (free, no production use rights, 180-day limit)
  2) Developer (free, no production use rights)
  3) Express (free)
  4) Web (PAID)
  5) Standard (PAID)
  6) Enterprise (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:~$
```

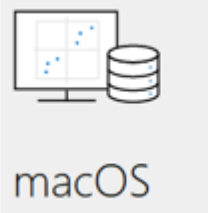
SQL Server on Linux

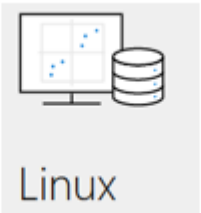
- Some figures
  - 5 years estimated to port SQL Server natively on Linux
  - 3 weeks to prototype using SQLPAL
  - 24 months to release
- Why ?
  - ISVs expectation
  - Customers wishes
  - Win market shares



SQL Server everywhere







- SQL Server **2017**
  - Official support inside containers
  - Official support on Linux


```
# Creates a container with SQL Server 2017 Windows
docker run --detach `
           --name sqldocker `
           --hostname sqldocker `
           --publish 1433:1433 `
           --volume c:\mssql\sqldocker:c:\mssql `
           --env sa_password=Password1! `
           --env ACCEPT_EULA=Y `
           microsoft/mssql-server-windows-express
```

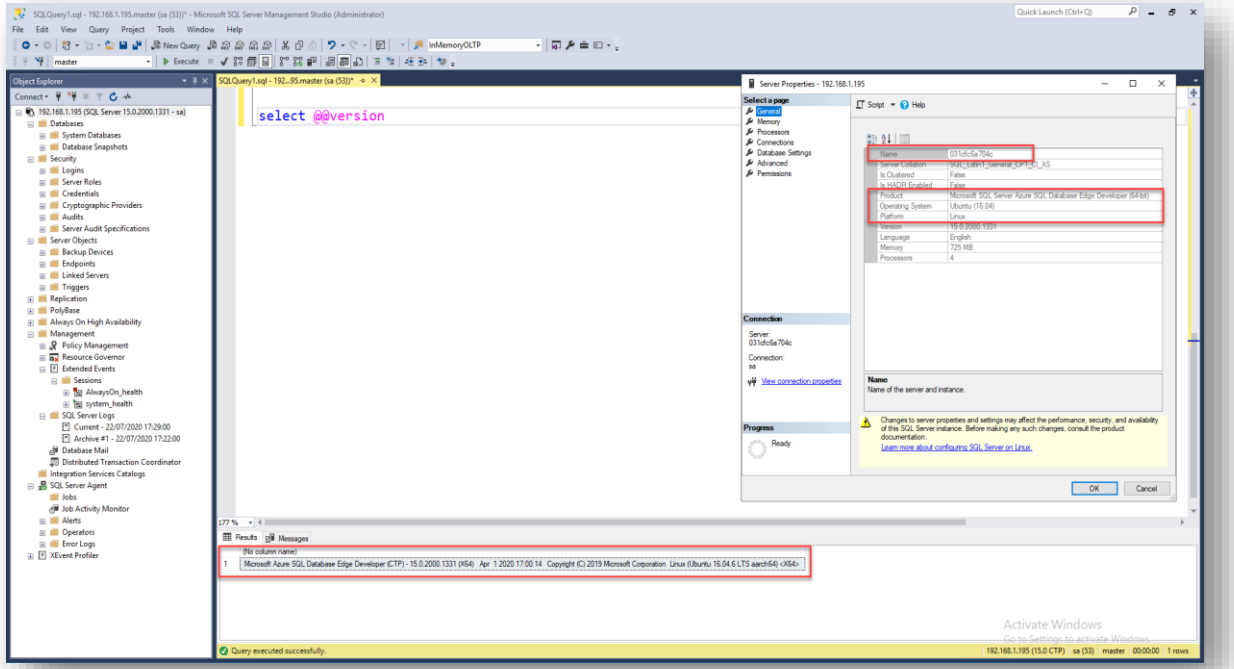

```
# Creates a container with SQL Server 2017 Linux
docker run  --detach \
            --name sqldocker \
            --hostname sqldocker \
            --env 'MSSQL_PID=developer' \
            --env 'SA_PASSWORD=Password1!' \
            --env 'ACCEPT_EULA=Y' \
            --publish 1433:1433 \
            microsoft/mssql-server-linux
```

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
```

SQL Server inside a container

- Bonus : Azure SQL Edge
  - SQL Server inside a container
  - Running on Raspberry Pi (ARM64)
  - Data streaming, time series
  - In-database ML and graph features




```
ubuntu@UbuntuRPi3:~$ sudo docker images
REPOSITORY                                                          TAG                 IMAGE ID            CREATED             SIZE
mcr.microsoft.com/azureiotedge-hub                                  1.0                 5848317c21b9        4 days ago          237MB
mcr.microsoft.com/azureiotedge-agent                                1.0                 beff1f7b2802        4 days ago          219MB
mcr.microsoft.com/azureiotedge-diagnostics                          1.0.9.3             094e654f1ecb        4 weeks ago         8.03MB
marketplace.azurecr.io/microsoftsqledge-preview/azure-sql-database-edge   latest              3108f728f69d        3 months ago        1.79GB
ubuntu@UbuntuRPi3:~$ sudo docker ps
CONTAINER ID        IMAGE                                                                   COMMAND                  CREATED             STATUS              PORTS
                                                    NAMES
031cfc6a704c        marketplace.azurecr.io/microsoftsqledge-preview/azure-sql-database-edge:latest   "/bin/sh -c 'dotnet …"   5 minutes ago       Up 5 minutes        1401/tcp, 0.0.0.0:1433->1433/tcp
                                                    AzureSQLDatabaseEdge
0dba03be2715        mcr.microsoft.com/azureiotedge-hub:1.0                                  "/bin/sh -c 'echo \"$…"  16 minutes ago      Up 16 minutes       0.0.0.0:443->443/tcp, 0.0.0.0:5671->5671
/tcp, 0.0.0.0:8883->8883/tcp   edgeHub
c32e4f71250d        mcr.microsoft.com/azureiotedge-agent:1.0                                "/bin/sh -c 'echo \"$…"  42 minutes ago      Up 39 minutes
                                                    edgeAgent
ubuntu@UbuntuRPi3:~$
```

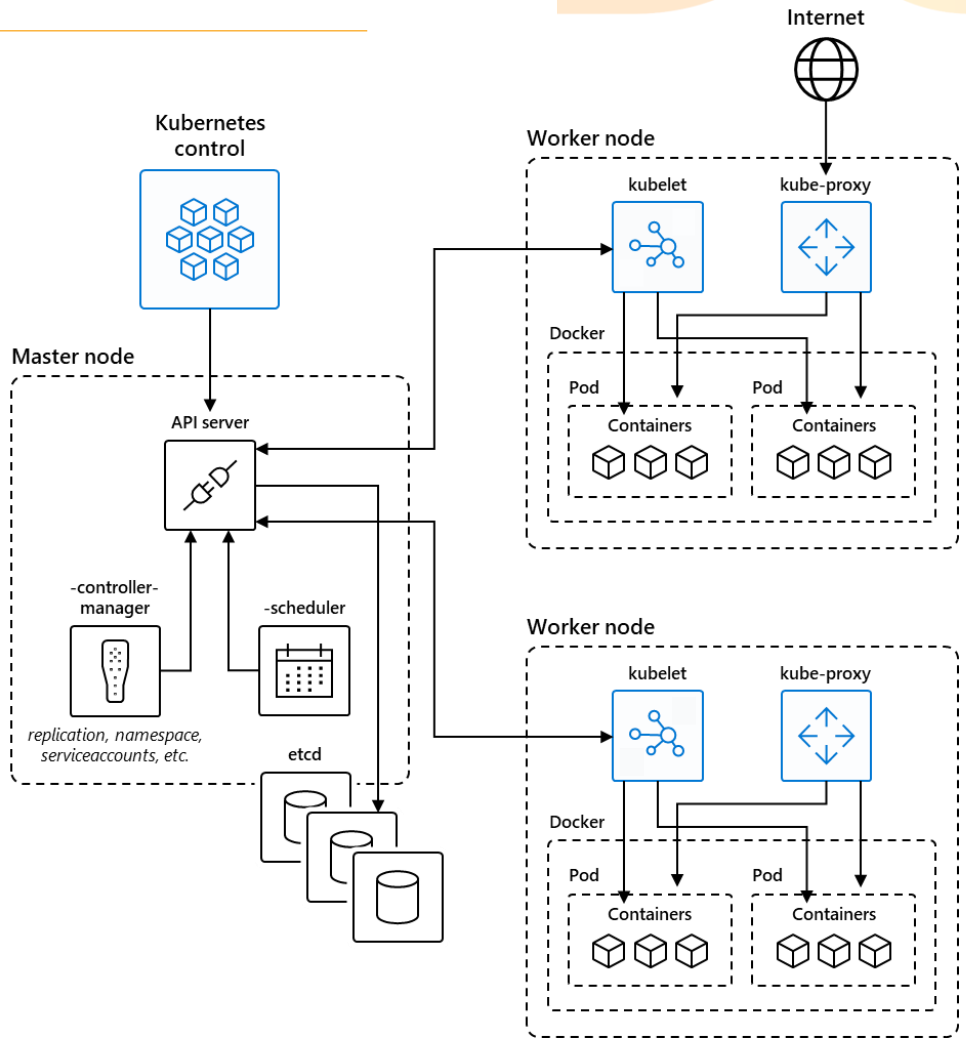
Now … What's next ?

- 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
- And some scaling functions
  - Manage container resources (CPU, RAM …)
  - Allow scaling out by adding more containers
- And we wish a similar deployment experience
  - OnPrem
  - Public Cloud

Sounds familiar … Looks like a cluster !

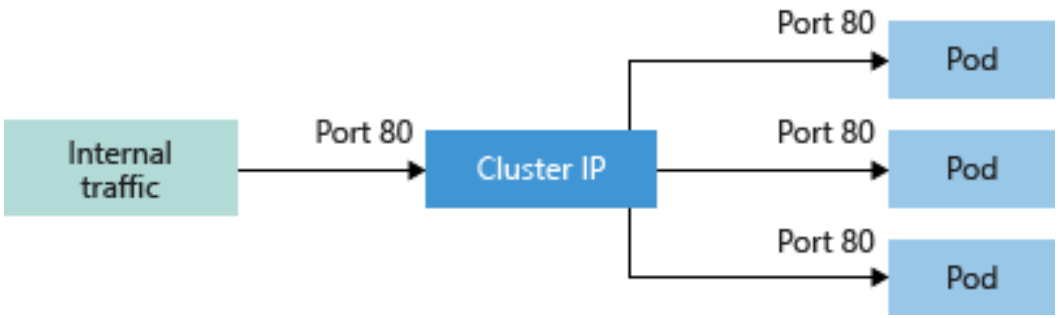
Kubernetes for DBAs

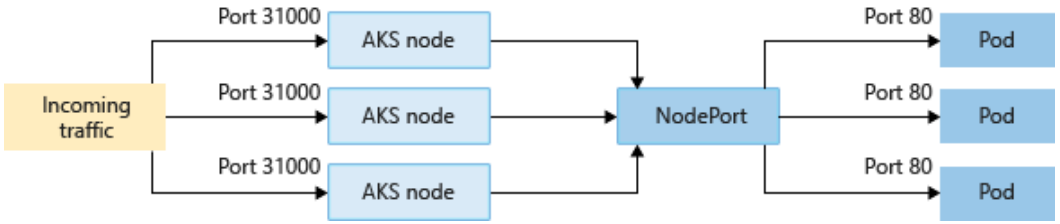
- Also known as K8s
  - Pod
    - Smallest management unit
    - Made of 1..N containers
    - Unique @IP across the cluster
  - Master node
    - Responsible for pod scheduling
  - Worker node
    - Node of the K8s Cluster
    - Kubelet : responsible for running containers
    - Kube-proxy : manage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

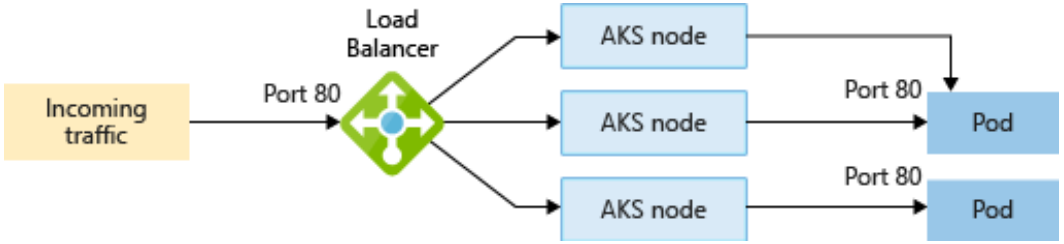


Connecting to applications

- 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





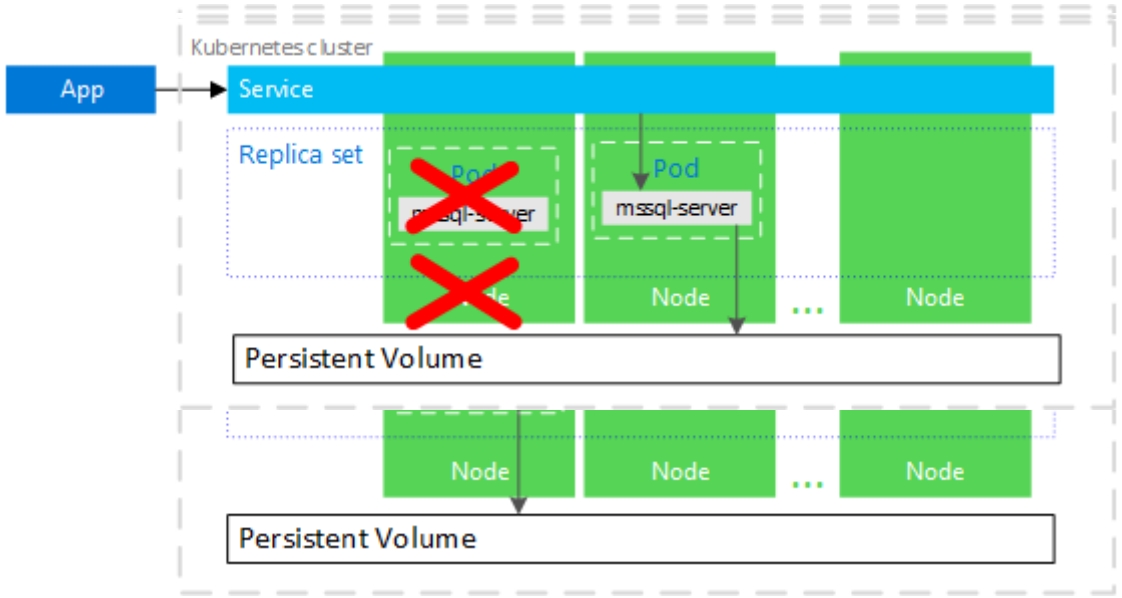


Kubectl command line

Command	Description
kubectl 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

SQL Server inside K8s

- Suitable for production
- HA enabled by default for the instance !
  - Persistent Volume available across the cluster nodes
  - K8s is responsible to maintain Desired State Configuration
- Failover
  - When a failure occurs
  - K8s tries to spin up a pod on the same node
  - Or on another worker node
- Applications will reconnect
  - Same service means same IP address / port
  - K8s will redirect network traffic to the new pod



A bit of history

- Year 2010
  - we were wondering … if virtualizing SQL server was the right choice
  - And Microsoft was releasing some cloud services …
  - A managed SQL Server was one of them !

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.

A bit of history

- Nowadays : IaaS & PaaS

SQL virtual machines

Best for migrations and applications requiring OS-level access

Managed instances

Best for most lift-and-shift migrations to the cloud

Azure SQL Databases

Best for modern cloud applications. Hyperscale and serverless options are available

A bit of history

- Nowadays : CaaS

Azure Container Instances

Azure Kubernetes Service

SQL Server inside ACI

```
# 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;"
```

SQL Server inside AKS


```
# Create a dedicated Namespace
kubectl create namespace ns-sqlday2021
kubectl get namespaces

# refresh every 2 seconds the resources created
watch kubectl get all --namespace ns-sqlday2021

# Create a secret to be used by SQL Server deployment
kubectl create secret generic mssql \
            --from-literal=SA_PASSWORD="MyC0m9l&xP@ssw0rd"  \
            --namespace ns-sqlday2021

# 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 ns-sqlday2021
```

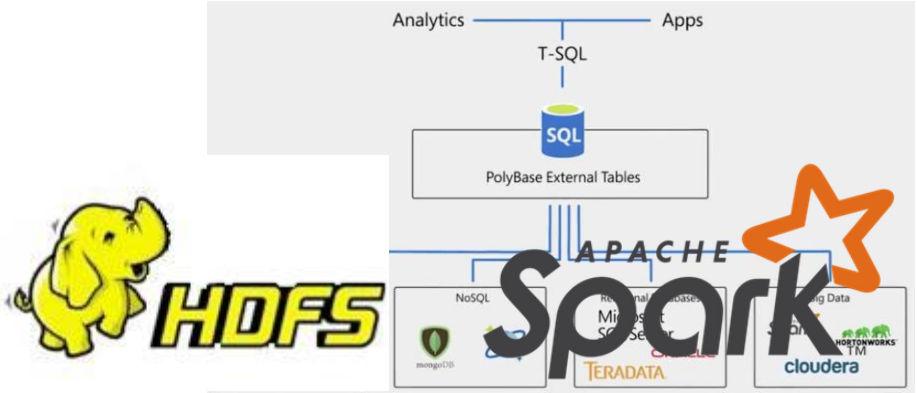
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mssql-deployment
  labels:
    app: mssql
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mssql
  template:
    metadata:
      labels:
        app: mssql
    spec:
      terminationGracePeriodSeconds: 10
      hostname: mssqlinst1
      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
```

Azure Kubernetes Services

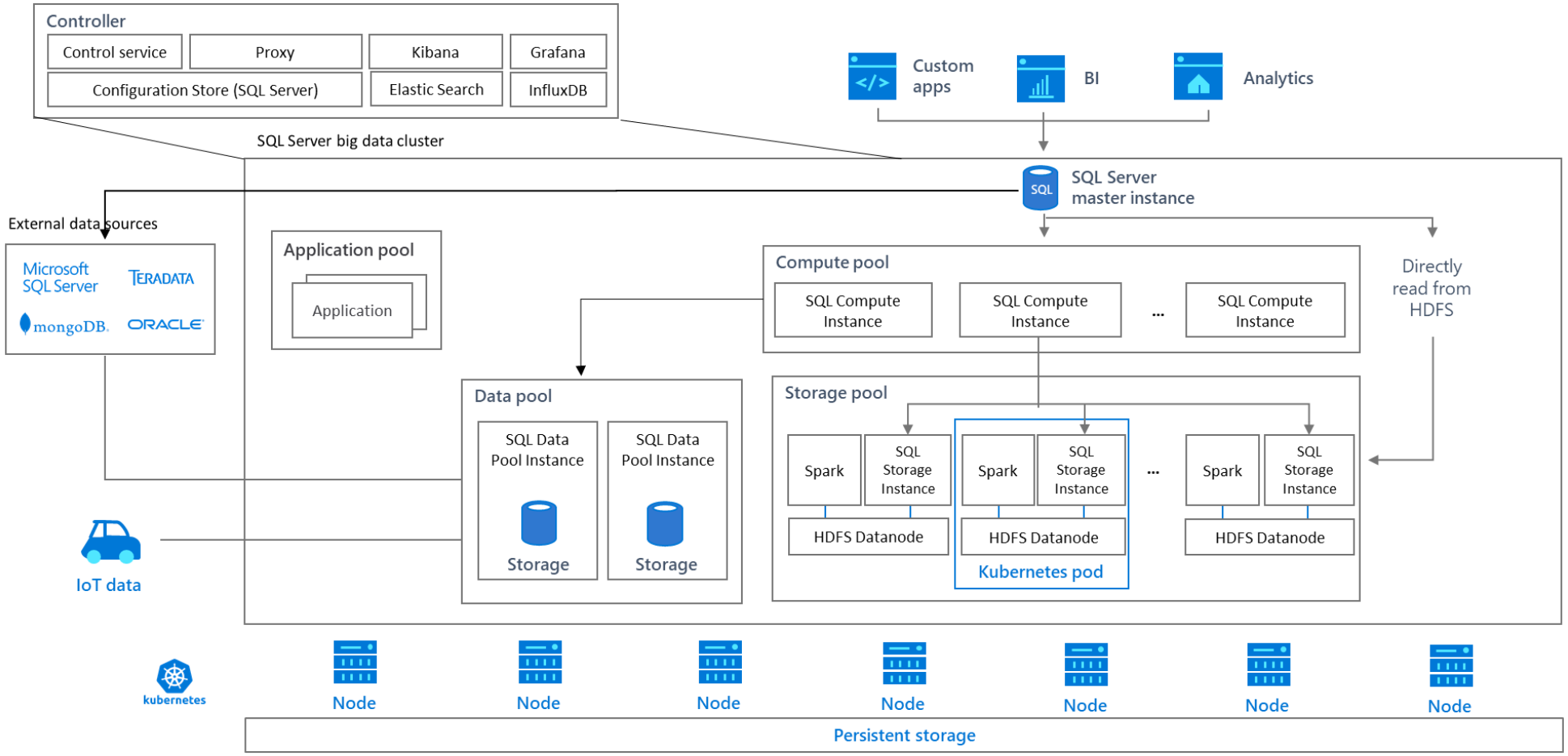
- Always On Availability Groups

But … Wait ….

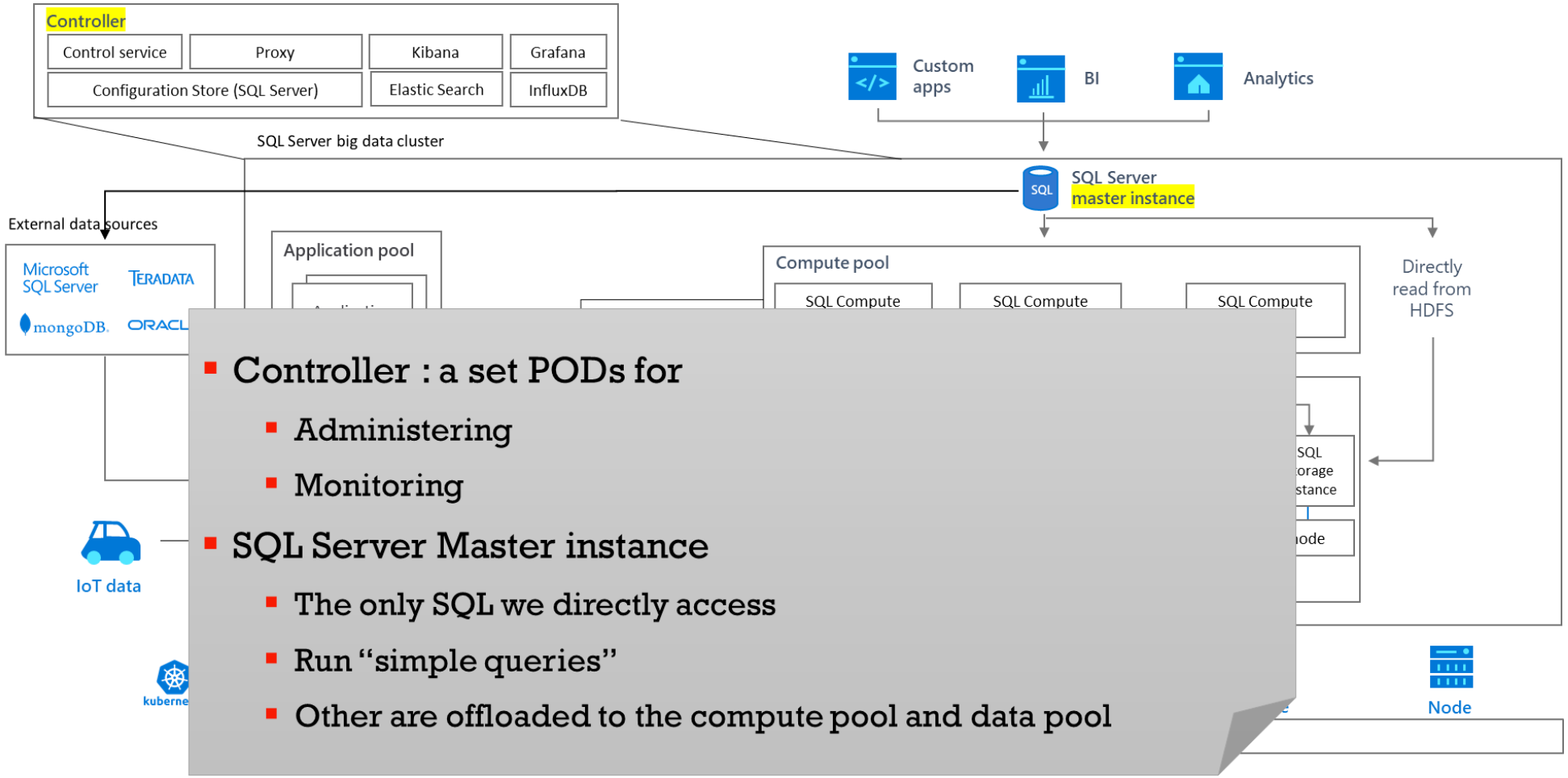
- K8s can run SQL Server
- K8s and containers can run almost all kind of applications
- Nowadays SQL Server is more than a SGBD
  - SQL Server offers Data Virtualization with Polybase
- Pods can host multiple containers
- Let’s add some « Big Data» containers
  - With Spark engine
  - And some kind of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



Big Data Cluster



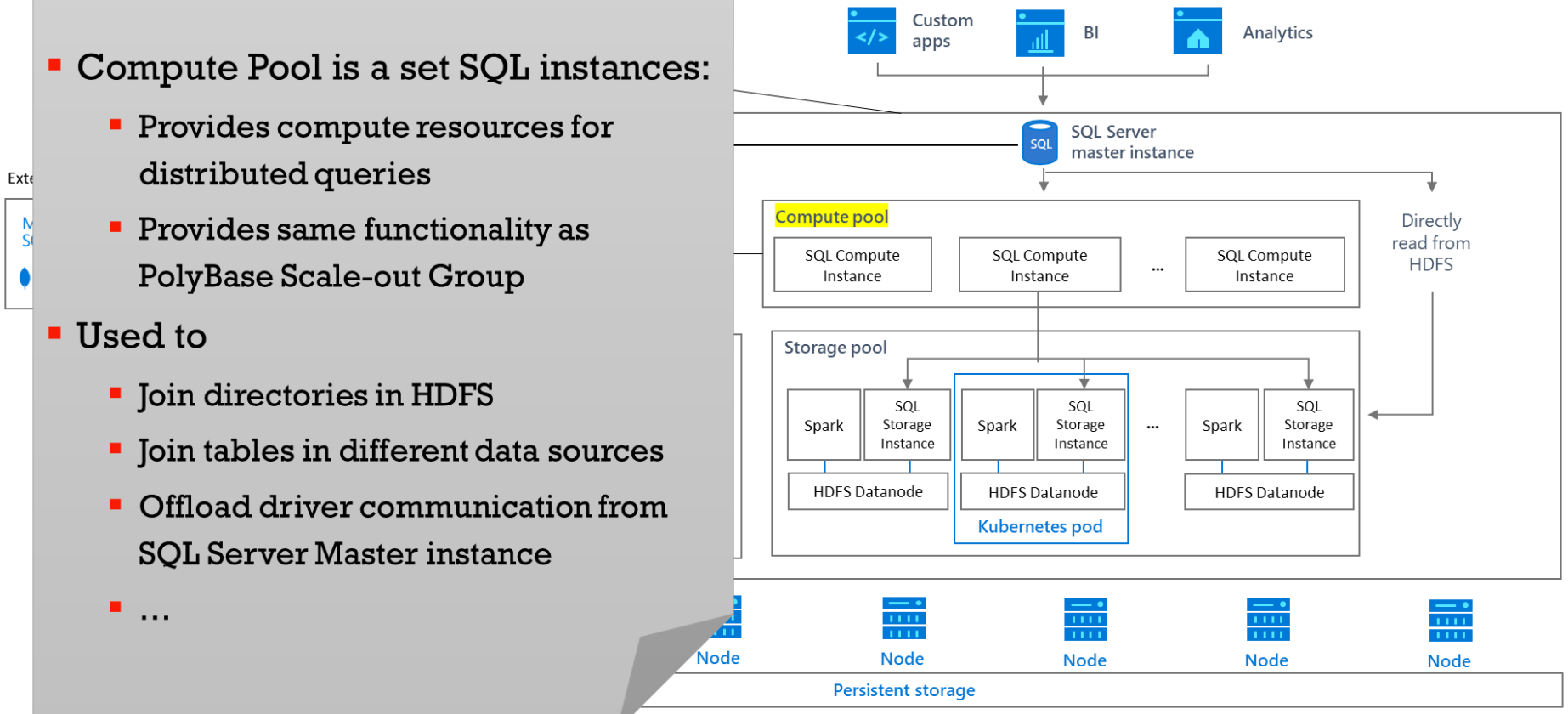
Control plane



- 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

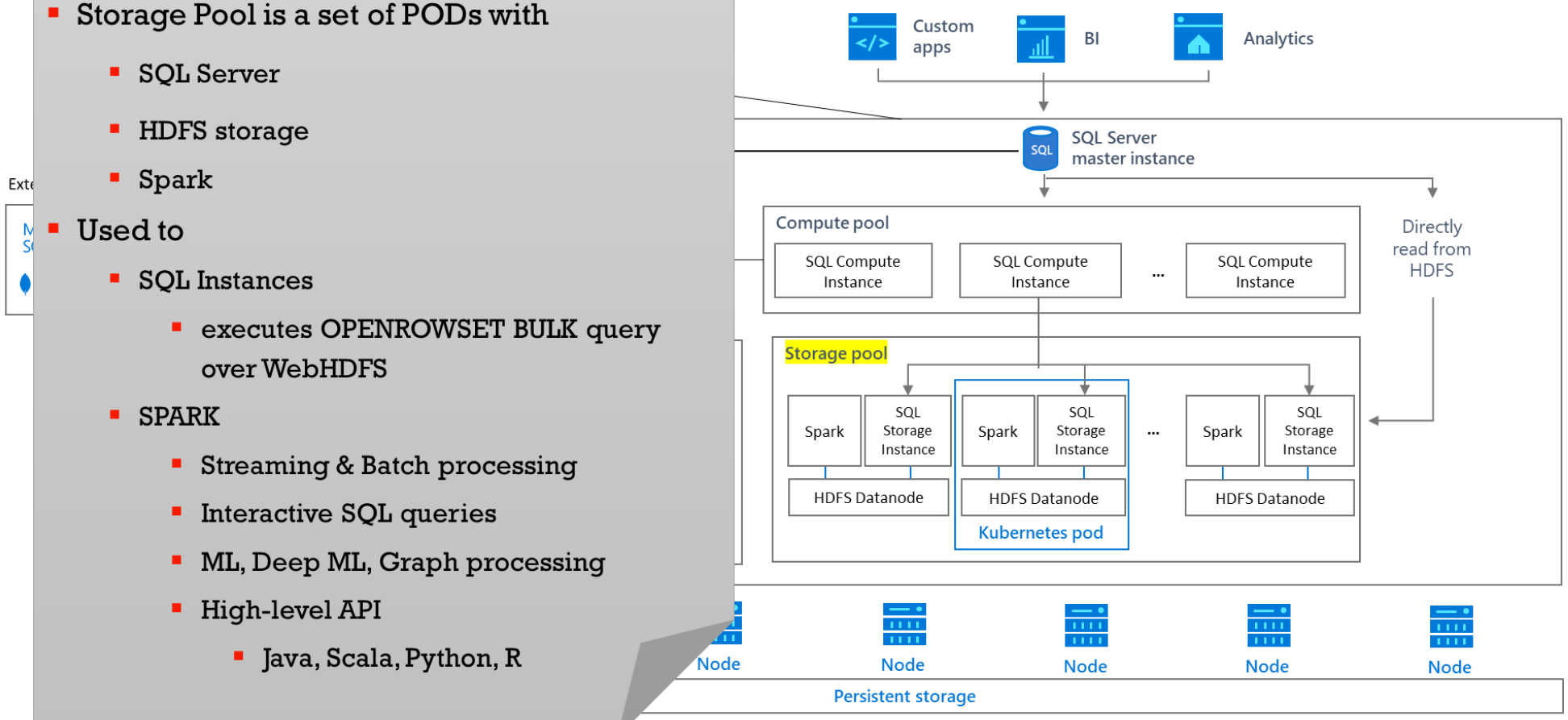
Compute plane

- 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
  - …

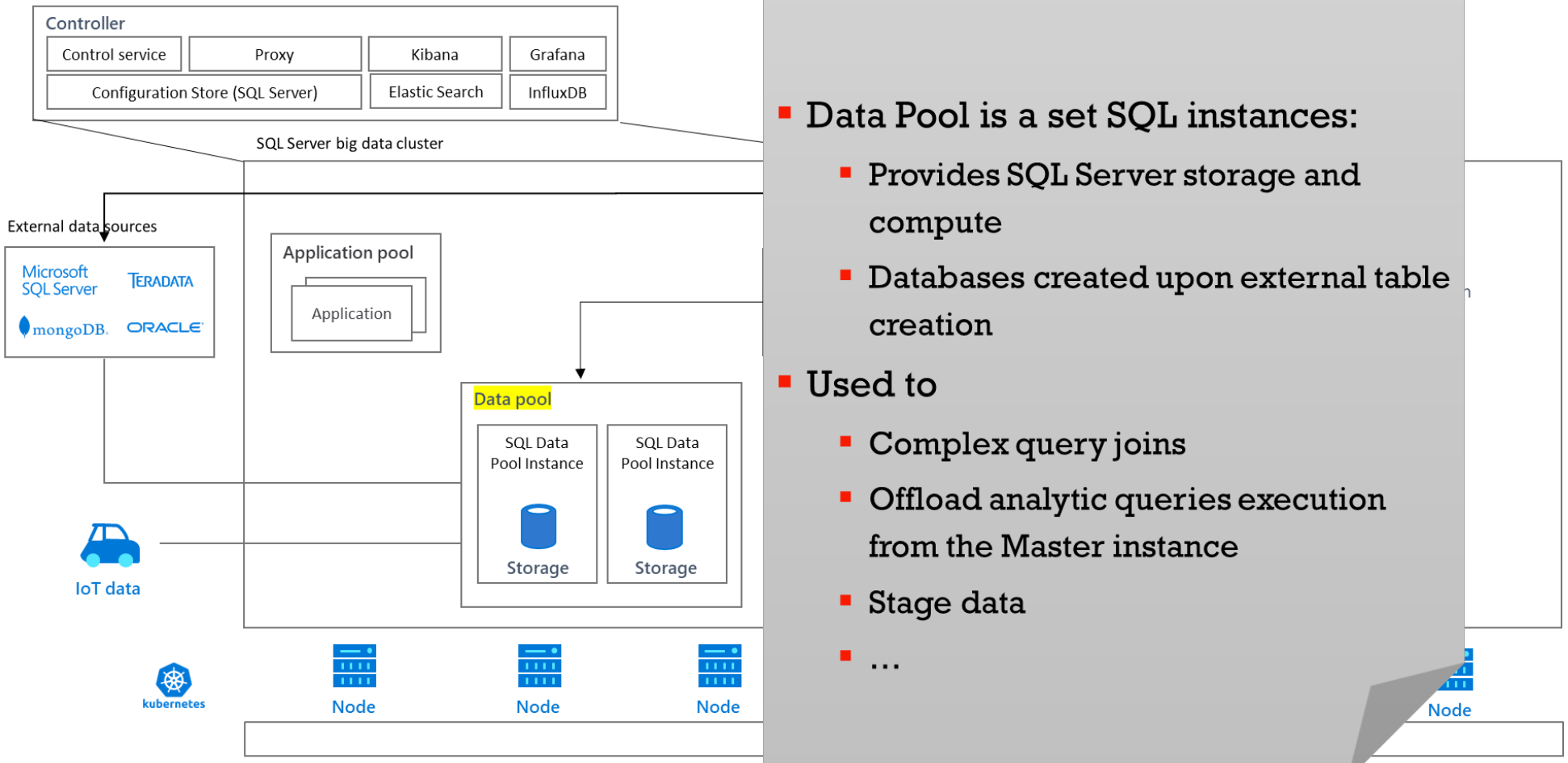


Data plane : Storage pool

- 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	Dri
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)
```

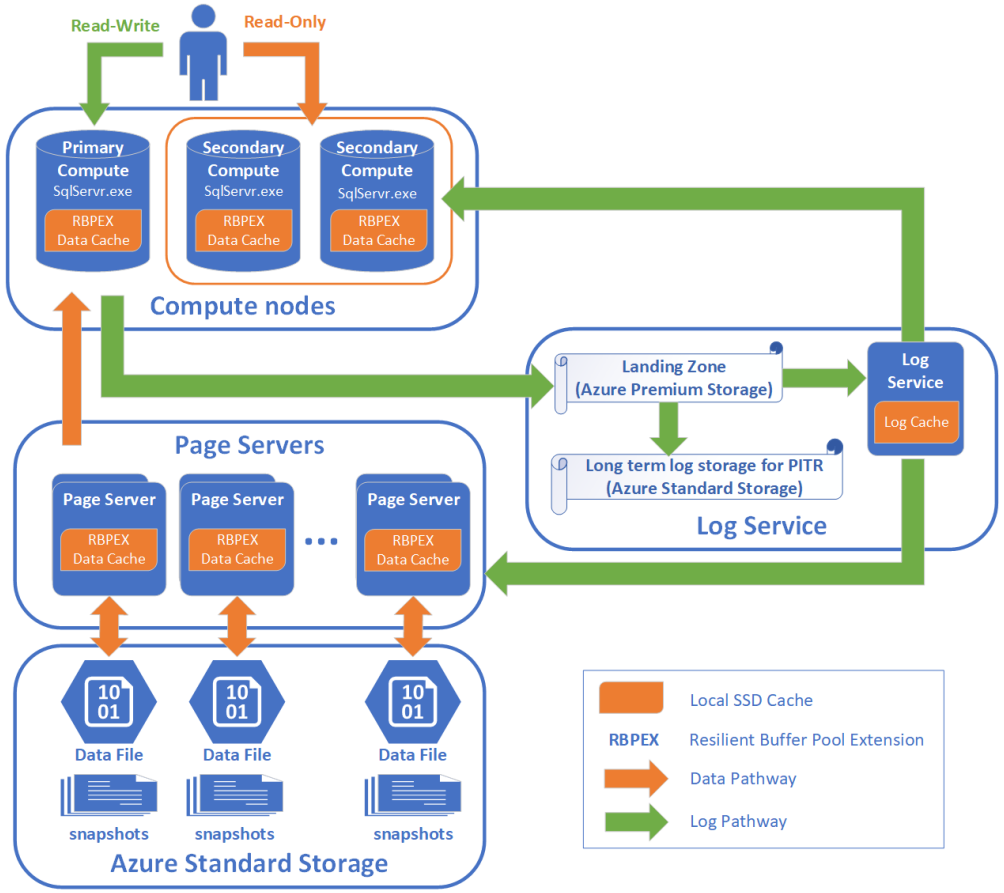
[11]

```
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
```

Entering a new era

- SQL Server is constantly evolving
  - New features / capabilities
- SQL Server on Linux was a milestone
- SQL Server is everywhere
  - Windows, Linux, Docker, K8s
  - OnPrem, Cloud, on the Edge
- Even a new architecture …
  - Compute and storage scale out
  - Breaks the monolithic SQL engine



Thanks for attending

STRATEGIC PARTNER

Microsoft

GOLD SPONSOR

elitmind think bright

lingaro

predica.

SILVER SPONSOR

Future Processing

/conseilit

@conseilit

/christophelaporte

conseilit@outlook.com

STRATEGIC PARTNER

Microsoft

GOLD SPONSOR

elitmind think bright

lingaro

predica.

SILVER SPONSOR

Future Processing